ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

FOG COMPUTING ARCHITECTURE FOR E-TEXTILE APPLICATIONS

Ph.D. THESIS

Kadir ÖZLEM

Department of Computer Engineering

Computer Engineering Programme

DECEMBER 2024

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

FOG COMPUTING ARCHITECTURE FOR E-TEXTILE APPLICATIONS

Ph.D. THESIS

Kadir ÖZLEM
(504182521)

Department of Computer Engineering

Computer Engineering Programme

Thesis Advisor: Assoc. Prof. Dr. Gökhan İNCE
Co-Advisor: Assoc. Prof. Dr. Özgür ATALAY

DECEMBER 2024

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

E-TEKSTİL UYGULAMALARI İÇİN SİS BİLİŞİM MİMARİSİ

DOKTORA TEZİ

Kadir ÖZLEM
(504182521)

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Tez Danışmanı: Doç. Dr. Gökhan İNCE
Eş Danışman: Doç. Dr. Özgür ATALAY

ARALIK 2024

Kadir ÖZLEM, a Ph.D. student of ITU Graduate School student ID 504182521 successfully defended the thesis entitled "FOG COMPUTING ARCHITECTURE FOR E-TEXTILE APPLICATIONS", which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Assoc. Prof. Dr. Gökhan İNCE** Istanbul Technical University	....
Co-advisor :	**Assoc. Prof. Dr. Özgür ATALAY** Istanbul Technical University	....
Jury Members :	**Assoc. Prof. Dr. İlkay ÖKSÜZ** Istanbul Technical University	....
	Prof. Dr. Emine Dilara KOÇAK Marmara University	....
	Prof. Dr. Hürriyet YILMAZ Formed Healthcare	....
	Asst. Prof. Dr. Ayşe YILMAZER METİN Istanbul Technical University	....
	Prof. Dr. Ali Gökhan YAVUZ Turkish-German University	....

Date of Submission : 27 November 2024
Date of Defense : 17 December 2024

To my spouse,

FOREWORD

Firstly, I would like to express my gratitude to my thesis advisor, Assoc. Prof. Dr. Gökhan İNCE. Throughout my master's and doctoral journey, he has consistently stood by my side and provided invaluable supervision. Additionally, collaborating with him on faculty duties has been an honor for me. Secondly, I am indebted to my co-advisor, Assoc. Prof. Dr. Özgür ATALAY, for his support and the opportunities he has provided me. Through his guidance, I have had the chance to participate in numerous academic studies. Additionally, I would like to extend my gratitude to Assist. Prof. Dr. Aslı ATALAY. Engaging in brainstorming sessions with her to solve issues that emerged in our projects has been a source of great pleasure for me. Being a part of the Soft Sensors Laboratory team, formed alongside all three of my mentors, is an honor for me.

I would like to express my gratitude to Prof. Dr. Emine Dilara KOÇAK, Prof. Dr. Hürriyet YILMAZ, and Assoc. Prof. Dr. İlkay ÖKSÜZ, my thesis progress committee, for their valuable contributions. The constructive feedback provided by them throughout the course of my thesis has significantly enhanced the quality of my work. The guidance they provided and their impact on my life is immensely valuable to me. Additionally, from the academic and educational activities I engaged in with them, I have learned a great deal. Therefore, I am especially grateful to them.

I am indebted to the Soft Sensors Laboratory team, of which I have been a part since its inception. Contributing to each project within the team is a privilege I deeply value. I would like to extend my gratitude, especially to Abdulkadir PAZAR, Fidan KHALILBAYLI, Ayşe Feyza YILMAZ, and Çağatay GÜMÜŞ for their contributions to the research work in this thesis. Furthermore, I would like to express my gratitude to Selim Enes KILIÇASLAN, who provided assistance in addressing the crucial aspect of connectivity issues between sensors and mobile phones, which is one of the pivotal components of my thesis. In addition, I would like to extend my thanks to Uğur AYVAZ, Hasbi SEVİNÇ, Cemal Fatih KUYUCU, Meral KORKMAZ KUYUCU, Hend ELMOUGHNI, Ezgi PAKET, Ömür Fatmanur ERZURUMLUOĞLU, İlknur ÇELİK, Nada AL-AZZAWI, İrem YÜNCÜLER, Ibrahim AHMED, Mehmet Fatih ÇELEBİ, Münire Sibel ÇETİN, and Bahman TAHERKHANI, with whom I collaborated and co-authored publications during my doctoral studies. I would like to express my gratitude to all other undergraduate, and graduate students as well as interns whose names I may have omitted, for their contributions and efforts in our collaborative work.

I extend my thanks to the former Faculty Dean, Prof. Dr. Sema Fatma OKTUĞ, for the support and trust extended to me since becoming a member of the faculty. Particularly, I wish to express separate and special thanks to our former department chairs, Prof. Dr. Mustafa Ersel KAMAŞAK and Prof. Dr. Sema Fatma OKTUĞ, for providing our team with laboratory space and enabling the continuation of our work. Additionally, I express my gratitude to Assoc. Prof. Dr. Yusuf YASLAN, Assoc. Prof. Dr. Şerif BAHTİYAR, and all other esteemed academic staff on the faculty who have guided me. Furthermore, I would like to thank Lecturer Tacettin AYAR, who has been leading the way for me since my first day at the faculty and has generously offered support in every aspect. I would like to extend special thanks to Asst. Prof. Dr. Yusuf Hüseyin ŞAHİN, who took on the administrative responsibilities of my courses, enabling me to complete my thesis studies.

This thesis has been supported by a grant from the Scientific Research Project Unit of Istanbul Technical University, Grant No: MDK-2021-43137. Moreover, this work has received support from the European Research Council (ERC) under the European Union's Horizon Research and Innovation Programme Project "Textile-Based Wearable Soft Robotics with Integrated Sensing, Actuating, and Self-Powering Properties - TEXWEAROTS" (Grant No: 101042402), the EU Marie Skłodowska-Curie IF Project "Textile-based Soft Sensing Actuators for Soft Robotic Applications—TexRobots" (Grant No: 842786), the Scientific and Technological Research Council of Türkiye (TUBITAK), project "Development of Textile-Based Robotics for Drop Foot Syndrome" (Grant No: 120C118), TUBITAK project "Fabric-Based Soft Actuators for Wearable Applications" (Grant No: 118M668), and TUBITAK-IRASME project "'STEWART-Sports acTivitiEs With weARable Technology' (Grant No: 221N310). Additionally, I would like to extend our gratitude to GitHub and JetBrains for providing their products free of charge during the developmental phase of this study. Particularly, I wish to thank Digital Ocean for generously providing us with free credits to conduct cloud testing.

Finally, and most importantly, I would like to express my gratitude to my mother, father, and sister for their unwavering support throughout my thesis and my entire educational journey, providing me with every opportunity. Especially, I would like to express my gratitude to my partner, my wife, and my beloved Semanur for her support and patience throughout the thesis process. Her support has always provided me with motivation and the opportunity to complete my thesis to the best of my ability. Despite occasionally disrupting my work by climbing onto me or biting my monitor, I would like to thank to our cat, Mıncır, for providing entertainment and boosting my energy during moments of exhaustion. I would also like to extend my special thanks to all my relatives who supported me throughout the thesis period. Thanks for all.

DECEMBER 2024 Kadir ÖZLEM
(Research and Teaching Assistant)

TABLE OF CONTENTS

Page

FOREWORD .... **x**
TABLE OF CONTENTS .... **xiii**
ABBREVIATIONS .... **xv**
SYMBOLS .... **xix**
LIST OF TABLES .... **xxi**
LIST OF FIGURES .... **xxv**
SUMMARY .... **xxvii**
ÖZET .... **xxxi**
1. INTRODUCTION **1**
1.1 Purpose of the Thesis 3
1.2 Contribution of the Thesis 4
1.3 Organization of the Thesis 6
2. LITERATURE REVIEW **7**
2.1 Electronic Textiles 7
2.1.1 Sensors 7
2.1.1.1 Textile based resistive sensors 8
2.1.1.2 Textile based capacitive sensors 9
2.1.1.3 Textile based inductive sensors 11
2.1.1.4 Textile electrodes 11
2.1.2 Actuators 12
2.1.2.1 Cable driven textile actuators 13
2.1.2.2 Fluidic textile actuators 13
2.1.2.3 Shape-changing actuators 14
2.2 Applications 15
2.2.1 Gait phase recognition system 15
2.2.2 Assistive soft robotic control 16
2.3 Computing Systems 18
2.3.1 Cloud computing 18
2.3.2 Edge computing 21
2.3.3 Fog computing 23
2.3.3.1 Fog computing in smart cities 23
2.3.3.2 Fog computing in connected vehicles 24
2.3.3.3 Fog computing in smart grid 25
2.3.3.4 Fog computing in smart homes 26
2.3.3.5 Fog computing in healthcare 26
2.3.3.6 Fog computing in non-textile wearables 27
2.3.3.7 Fog computing in e-textiles 30
2.4 Positioning of the Thesis 30
3. FogETex FRAMEWORK **33**
3.1 Hardware Components 33
3.1.1 Textile-based IoT devices 33
3.1.2 Gateway devices 35
3.1.3 Broker nodes 36
3.1.4 Worker nodes 36
3.1.5 Cloud 37
3.2 Software Components 38
3.2.1 Resource manager 39
3.2.2 Computing module 43
3.2.2.1 Data preprocessing module 43
3.2.2.2 Deep learning module 44
3.2.3 Proxy module 44
3.2.4 User interface 45

3.3 Network Structure 47
3.3.1 RESTful API communication 48
3.3.2 Socket communication 50
3.3.3 Scalability handling 52
3.4 Concurrency Control Techniques 52
3.4.1 Single-threaded data processing via WebSocket IPC 53
3.4.2 Multi-threaded data processing via WebSocket IPC 54
3.4.3 Multi-process data processing via WebSocket IPC 55
3.4.4 Multi-process data processing via RESTful API 55
3.4.5 Multi-process data processing via FIFO IPC 56
4. GAIT PHASE RECOGNITION SYSTEM USING FogETex 59
4.1 T-IoT Device Design 59
4.1.1 Measurement based on textile-based capacitive strain sensor 60
4.1.2 Measurement based on IMU sensor 61
4.2 Data Labeling 62
4.3 Dataset 63
4.4 Data Preprocessing 63
4.5 Deep Learning Model 64
4.6 Experiments 64
4.6.1 Experimental setup 65
4.6.1.1 Wi-Fi testbed 65
4.6.1.2 Cellular testbed 67
4.6.2 Experimental scenarios 67
4.6.3 Evaluation criteria 69
4.6.3.1 Prediction accuracies 69
4.6.3.2 Time characteristics 70
4.6.3.3 Device resource usage 70
4.6.3.4 Network bandwidth usage 71
4.6.3.5 Stress test 71
4.7 Results 71
4.7.1 Results on prediction accuracies 71
4.7.2 Results on time performance 72
4.7.3 Results on resource usage 78
4.7.4 Results on network bandwidth usage 79
4.7.5 Stress test 79
4.7.6 Performance benchmarking 81
4.8 Discussion 82
4.9 Limitations 83
4.10 Conclusion 84
5. ASSISTIVE SOFT ROBOTIC GLOVE CONTROL USING FogETex 87
5.1 System Architecture and Material Designs 88
5.1.1 Design of the sensing T-IoT glove 88
5.1.2 Design of the actuating T-IoT glove 89
5.2 Data Processing 90
5.2.1 Data processing in the sensing T-IoT glove 91
5.2.2 Data processing in the cloud architecture 92
5.2.3 Control of the actuating T-IoT glove 94
5.3 Experiments 95
5.3.1 Characterization of T-IoT gloves 95
5.3.2 Dataset and labelling 96
5.3.3 Experimental setup 100
5.3.4 Evaluation criteria 102
5.3.4.1 Accuracy criteria 103
5.4 Results 103
5.4.1 Characterization of sensing T-IoT gloves 104
5.4.2 Characterization of actuating T-IoT gloves 105
5.4.3 Results on accuracy 105
5.4.4 Results on time performance 109
5.4.5 Results on resource usage 113
5.4.6 Results on network bandwidth usage 114
5.4.7 Results on concurrency control techniques 114
5.4.8 Discussion 120

5.5 Conclusion 121
6. CONCLUSION **123**
REFERENCES **127**
APPENDICES **153**
APPENDIX A : Assistive Soft Robotic Glove Control Supporting Information 155
CURRICULUM VITAE **164**

ABBREVIATIONS

3D	: Three-dimensional
4G	: the fourth-generation technology standard for cellular networks
5G	: The fifth-generation technology standard for cellular networks
ADSL	: Asymmetric Digital Subscriber Line
AI	: Artificial Intelligence
AP	: Access Point
API	: Application Programming Interface
ARM	: Advanced RISC Machine / Acorn RISC Machine
AWS	: Amazon Web Services
BLE	: Bluetooth Low Energy
CN	: Candidate Node
CNT	: Carbon NanoTube
COVID-19	: COronaVIrus Disease 2019
CPU	: Central Processing Unit
CTMC	: Continuous-Time Markov Chain
CV	: Connected Vehicles
DDR	: Double Data Rate
DP	: Distal Phalanx
DT	: Decision Tree
E-commerce	: Electronic Commerce
E-textile	: Electronic Textile
ECG	: ElectroCardioGram
EEG	: ElectroEncephaloGram
EKG	: ElektroKardiyoGram (in Turkish)
EMG	: ElectroMyoGram
FIFO	: First In First Out
FogETex	: Fog Computing Framework for Electronic Textile Applications
FOV	: Field Of View
GIL	: Global Interpreter Lock
GUI	: Graphical User Interface
Gym	: GYMnasium
HF	: HealthFog System
HTTP	: Hyper-Text Transfer Protocol
I2C	: Inter-Integrated Circuit
IaaS	: Infrastructure as a Service
IEEE	: Institute of Electrical and Electronics Engineers
IMU	: Inertial Measurement Unit
iOS	: iPhone Operating System
IoT	: Internet of Things
IP	: Internet Protocol
ISP	: Internet Service Provider

IT	: Information Technology
KNN	: K-Nearest Neighbors
KVM	: Kernel-based Virtual Machine
LAN	: Local Area Network
LiPo	: Lithium Polymer
LoRaWAN	: Long Range Wide Area Network
LPDDR	: Low-Power Double Data Rate
LR	: Logistic Regression
LSTM	: Long Short-Term Memory
LTE	: Long-Term Evolution
MATLAB	: MATrix LABoratory
MCP	: MetaCarpoPhalangeal joint
MEMS	: MicroElectroMechanical Systems
ML	: Machine Learning
MLP	: Multilayer Perceptron
PaaS	: Platform as a Service
PAM	: McKibben/Pneumatic Artificial Muscle
PAN	: Personal Area Network
PCT	: Patent Cooperation Treaty
PDA	: Personal Data Assistant
PIP	: Proximal InterPhalangeal joint
PPG	: PhotoPlethysmoGraphy
QoS	: Quality of Service
RAM	: Random-Access Memory
REST	: REpresentational State Transfer
RFID	: Radio-Frequency IDentification
SaaS	: Software as a Service
SD	: Standard Deviation
SDHC	: Secure Digital High Capacity
SDN	: Software-Defined Network
SDRAM	: Synchronous Dynamic Random-Access Memory
SMA	: Shape Memory Alloys
SMP	: Shape Memory Polymer
SOAP	: Symbolic Optimal Assembly Program
SoC	: System-on-Chip
SQC	: Signal Quality and Comfort
SSD	: Solid-State Drive
T-IoT	: Textile-based Internet of Things
TCP	: Transmission Control Protocol
TLS	: Traffic Light System
TPU	: Thermoplastic PolyUrethane
USA	: United State of America
USB	: Universal Serial Bus
VDSL	: Very High-speed Digital Subscriber Line
VFC	: Vehicular Fog Computing
VPS	: Virtual Private Server
WAN	: Wide Area Network

Web	**:**	World Wide Web
Wi-Fi	**:**	Wireless Fidelity
XGB	**:**	eXtreme Gradient Boosting
XGBoost	**:**	eXtreme Gradient Boosting
YOLO	**:**	You Only Look Once

SYMBOLS

α	: Angle formed by the DP and MCP points at the PIP point
β	: Angle of the arc formed by the bending of the finger
$\Delta\lambda$	: Difference between the longitudes of the user and CN
$\Delta\phi$	: Disparity between the latitudes of the user and CN
θ	: Angular distance between two points
μ	: Mean of feature
σ	: Standard deviation of feature
ϕ_1	: Latitude of the user
ϕ_2	: Latitude of the Candidate Node
a	: Square of half the cord length between two points
c	: Class number
C	: Capacitance
C_{min}	: Minimum capacitance value
C_{ref}	: Stray capacitance
C_x	: Capacitance value of the sensor
d	: Distance between the user and CN
$F1_c$	: F1 score for class c
FN_c	: False negative prediction of class c
FN	: False negative
FP_c	: False positive prediction of class c
FP	: False positive
I	: Current
M	: Number of classes
n	: Input size of the function
N	: Sample numbers
O	: Computational Complexity
P_c	: Precision score for class c
Q	: Stored charge
R_c	: Recall score for class c
R	: Mean radius of the Earth
$Score_c$	: Score of the c-th class
T	: Charging time
TN	: True negative
TP_c	: True positive prediction of class c
TP	: True positive
V_{dd}	: Supply voltage
V_{in}	: Input voltage
V_{out}	: Supply voltage
V	: Voltage
x	: Feature value

y_c	: True probability of the class
y_i	: Actual label of the i-th sample
$\hat{y}_c$	: Predicted probability of the class
$\hat{y}_i$	: Predicted label of the i-th sample
Z	: Normalized feature value

LIST OF TABLES

Page

Table 4.1 : Demographic information of test subjects. **63**
Table 4.2 : Comparison performance of FogETex with other works. **81**
Table 5.1 : Test Subject Demographic Information **97**
Table 5.2 : Model Parameters of The Classifiers. **102**
Table 5.3 : F1 Scores of Individual Finger Models **107**
Table 5.4 : Average Performance of All Models **107**

LIST OF FIGURES

Page

Figure 1.1 : Focus area of the thesis. **4**
Figure 3.1 : General Overview of the Proposed System Architecture. **34**
Figure 3.2 : FogETex Hardware and Software Components. **37**
Figure 3.3 : Worker User Interface. **45**
Figure 3.4 : Broker User Interface. **46**
Figure 3.5 : Cloud User Interface. **46**
Figure 3.6 : Gateway Device Graphical User Interface. **48**
Figure 3.7 : Connection Diagram of Users with A Fog Node in their LAN. **49**
Figure 3.8 : Connection Diagram of Users Without a Fog Node in their LAN.... **50**
Figure 3.9 : Single-threaded Data Processing via WebSocket IPC. **53**
Figure 3.10 : Multi-threaded Data Processing via WebSocket IPC **54**
Figure 3.11 : Multi-process Data Processing via WebSocket IPC. **55**
Figure 3.12 : Multi-process Data Processing via RESTful API. **56**
Figure 3.13 : Multi-process Data Processing via FIFO IPC. **56**
Figure 4.1 : T-IoT Device for Gait Phase Recognition System: a) Actual Photo and b) Schematic Illustration. **60**
Figure 4.2 : T-IoT Device Placement on the Knee and IMU Axes (X and Z). **61**
Figure 4.3 : Gyro-z data. **62**
Figure 4.4 : Experimental Setup Device Connection Diagram. **66**
Figure 4.5 : Model train loss. **72**
Figure 4.6 : Macro F1 scores of trained model. **73**
Figure 4.7 : Confusion matrix of trained model. **73**
Figure 4.8 : Arbitration time in different scenarios. **74**
Figure 4.9 : Latency in different devices and testbeds. **75**
Figure 4.10 : Queuing delay in different devices and testbeds. **76**
Figure 4.11 : Execution time in different devices and testbeds. **77**
Figure 4.12 : Total response time in different devices and testbeds. **77**
Figure 4.13 : Jitter in different devices and testbeds. **78**
Figure 4.14 : CPU and memory usage in different devices and testbeds. **79**
Figure 4.15 : Network bandwidth usage in different devices and testbeds. **80**
Figure 4.16 : Mean response time in different devices with varying numbers of users. **80**
Figure 5.1 : Telerehabilitation over the cloud with the medical staff wearing a sensing T-IoT glove and the human patient wearing an actuating T-IoT glove for telerehabilitation: a) Sensing T-IoT glove, b) Actuating T-IoT glove. **88**
Figure 5.2 : System architecture of sensor and control signals transmission. **91**

Figure 5.3 : Label Configuration for Bending Angle Calculation of Finger Movement: (a) Label Placement on the Index Finger, (b) Bending Arc Representation of Label Positions................................ **96**
Figure 5.4 : a) Sensing T-IoT glove and its fingers. b) Sensing T-IoT glove calibration and fist movement: open hand and close hand. c) Single finger closing and opening: I. Thumb, II. Index, III. Middle, IV. Ring, V. Pinkie. d) Motions with a combination of fingers closing and opening: I. Open thumb and other fingers close, II. Cylindrical pinch with thumb and index, III. Cylindrical pinch with thumb and middle, IV. Cylindrical pinch with thumb and ring, V. Cylindrical pinch with thumb and pinkie... **98**
Figure 5.5 : a) Sensing T-IoT glove calibration and fist movement: open hand and close hand. b) Actuating T-IoT glove calibration and fist movement: open hand and close hand. c) Sensing T-IoT Glove single finger closing and opening: I. Thumb, II. Index, III. Middle, IV. Ring, V. Pinkie. d) Actuating T-IoT Glove single finger closing and opening: I. Thumb, II. Index, III. Middle, IV. Ring, V. Pinkie... **99**
Figure 5.6 : Experimenter Graphical User Interface of the Sensing T-IoT Glove Data Acquisition and Labelling System. **99**
Figure 5.7 : T-IoT Glove Finger Sensor Virtualization Interface.................. **101**
Figure 5.8 : Capacitance Change of Textile-based Sensor During Index Finger Movement.. **104**
Figure 5.9 : Change of Index Finger Bending Angle During Flexion and Extension Movement of Textile-based Actuator. **106**
Figure 5.10 : Confusion Matrices of Random Forest Classifier for Different Fingers... **108**
Figure 5.11 : Arbitration time for different classifiers. **109**
Figure 5.12 : Latency for different classifiers. **110**
Figure 5.13 : Queuing delay for different classifiers............................... **111**
Figure 5.14 : Execution time for different classifiers. **112**
Figure 5.15 : Total response time for different classifiers.......................... **112**
Figure 5.16 : Jitter for different classifiers.. **113**
Figure 5.17 : CPU and memory usage for different classifiers. **114**
Figure 5.18 : Network bandwidth for different classifiers. **115**
Figure 5.19 : Stress test results of multi-threaded data processing via WebSocket IPC... **116**
Figure 5.20 : Stress test results of multi-process data processing via WebSocket IPC... **117**
Figure 5.21 : Stress test results of multi-process data processing via RESTful IPC. **118**
Figure 5.22 : Stress test results of multi-process data processing via FIFO IPC. .. **118**
Figure 5.23 : Performance comparison of various concurrency control techniques. **119**
Figure 5.24 : Performance comparison of multi-worker and cloud. **120**
Figure A.1 : Confusion Matrices of Logistic Regression Classifier for Different Fingers... **155**
Figure A.2 : Confusion Matrices of Decision Tree Classifier for Different Fingers. **156**
Figure A.3 : Confusion Matrices of K-Nearest Neighbors Classifier for Different Fingers... **157**

Figure A.4 : Confusion Matrices of Multi-layer Perceptron Classifier for Different Fingers. **158**
Figure A.5 : Confusion Matrices of XGBoost Classifier for Different Fingers. ... **159**

**FOG COMPUTING ARCHITECTURE
FOR E-TEXTILE APPLICATIONS**

SUMMARY

Textile products are present in almost every aspect of human life. With the introduction of electronic textiles, textile products have become capable of converting various physiological and environmental stimuli into electrical signals, many of which are of vital importance to humans. Therefore, these products require real-time (low-latency) and robust computing systems. However, due to comfort considerations, they cannot accommodate powerful computing resources.

In this thesis study, a novel Fog computing-based framework for Electronic Textiles (FogETex) is proposed to meet the needs of e-textile applications. FogETex is a Platform-as-a-Service (PaaS) model that is cross-platform supported, scalable, and operates in real-time. This framework encompasses end-to-end integration of the system including Textile-based Internet of Things (T-IoT) devices, fog devices, and the cloud.

The FogETex framework consists of a three-layer architecture: the edge layer, the fog layer, and the cloud layer. The edge layer includes T-IoT devices that collect data from e-textile sensors and transmit it to the gateway device. Gateways are typically mobile phones that users carry in their daily lives. These devices are responsible for forwarding the collected data to the fog layer and visualizing the processed data. If the T-IoT device is equipped with its own Wi-Fi or LTE module, it can directly transmit data to the fog layer without requiring a gateway device.

The second layer includes broker and worker devices. The worker device is responsible for handling incoming computational requests, while the broker device manages the fog node. The broker monitors resource utilization data sent in real time by the worker devices at regular intervals to determine if any devices are overloaded. Based on resource usage, the broker assigns the most suitable worker device when a new user connects to the fog node. For security reasons, only the broker device within the fog node has a connection to devices on the Wide Area Network (WAN). As a result, in outdoor applications, data is transferred to the worker devices via the broker. In this setup, the broker acts as a proxy between the worker devices and the users.

The third and top layer is the cloud. The cloud device assigns users to an appropriate fog node based on availability information provided by the broker. While the cloud determines the suitable fog node, it does not interfere with the worker assignments within the fog node itself. This structure ensures decentralized management. Even if one node fails, the others can continue performing their tasks independently. Additionally, the cloud and broker devices, besides managing their primary responsibilities, are also capable of providing computational services.

Therefore, during system overloads, these devices can step in to serve users, ensuring continued functionality.

Since e-textile sensors generate time-series data and many sensors collect tens of data points per second, communication between the gateway device and the worker device is established using a WebSocket structure. This approach eliminates the need to repeatedly establish connections for every data transmission, enabling asynchronous and bidirectional data flow. On the other hand, operations such as device allocation requests made by the user to the cloud or broker are one-time processes and are managed via a RESTful API developed specifically for this purpose. Additionally, each device is equipped with a user interface that allows system administrators to monitor the status of the devices. This data can be utilized to make decisions about provisioning additional devices for overloaded fog nodes, ensuring optimal system performance.

To bring this thesis to fruition and understand the nature of the e-textile applications, a variety of applications were developed using electronic textiles in areas such as gait phase detection and hand motion recognition. On the other hand, to ensure that the developed framework functions as a comprehensive end-to-end system rather than a data processing platform, research was also conducted in textile-based soft robotics, another domain of smart textiles. These efforts include exoskeleton gloves for individuals with muscle weakness. Selected case scenarios from these applications were used to test the FogETex system.

For the first application of the proposed framework, a deep learning-based gait phase analysis application using textile-based capacitive sensors is employed. In this case study, knee movements were captured using a textile-based capacitive sensor placed on the test subject's knee. The sensor data was converted into gait phases using a deep learning-based machine learning method. In the next stage, these gait phase data are intended to be used as control signals for the artificial muscle actuator developed for foot drop treatment.

FogETex was evaluated in terms of time characteristics, resource usage, and network bandwidth usage using a mock client to determine the ideal system performance and an actual client to conduct real-world tests. All these tests were repeated on worker, broker, and cloud devices to validate indoor applications. Additionally, for outdoor applications, tests were conducted by connecting worker and cloud devices through WAN. The broker device acted as a proxy between the worker device and the user in this test. The fog devices outperformed the cloud system in these metrics.

In this case scenario, the performance of the FogETex framework was analyzed across different devices in applications with a single sensor. Additionally, a stress test was conducted to evaluate the framework's capability to handle multiple users. It was found that worker devices could serve up to 6 users, broker devices up to 18, and the cloud up to 14 users. The system demonstrated superior performance when three or more worker devices were employed compared to other configurations. Considering rental and device costs, the worker devices were deemed more cost-effective in terms of performance. Lastly, the FogETex framework was compared with other systems in the literature that could serve as competitors and are widely used in various studies. The comparison revealed that FogETex outperformed its counterparts in metrics such as latency, execution time, response time, and operational frequency.

To demonstrate the versatile applicability of the proposed FogETex framework further, a cloud-based remote manipulation system was developed, integrating e-textiles and textile-based soft robotic systems. The objective of this research is to combine a textile-based sensorized glove with an air-driven soft robotic glove, operated wirelessly using the developed control system architecture. The sensing glove equipped with capacitive sensors on each finger captures the movements of the medical staff's hand. Meanwhile, the pneumatic rehabilitation glove designed to aid patients affected by impaired hand function due to stroke, brain injury, or spinal cord injury replicates the movements of the medical personnel. The proposed artificial intelligence-based system detects finger gestures and actuates the pneumatic system, responding within an average response time of 48.4 ms. The evaluation of the system further in terms of accuracy and transmission quality metrics verifies the feasibility of the proposed system integrating textile gloves into IoT infrastructure, enabling remote motion sensing and actuation. In addition, the system was tested using various concurrency and inter-process communication methods. The system was also tested with multiple worker devices. It was observed that the system could serve up to 10 devices with 1 worker, up to 22 devices with 2 workers, up to 26 devices with 3 workers, and up to 23 devices with the cloud system.

On the other hand, this research also tested the FogETex system in multi-sensor e-textile applications. Models developed using various machine learning methods were introduced to the system as different applications, demonstrating that the framework can run multiple applications simultaneously. Although the framework was designed as a fog computing architecture, it can also operate exclusively as a cloud or edge computing system. In this study, it was confirmed that the framework can function effectively even without fog devices. Furthermore, the developed system successfully integrated e-textiles and soft robotics, proving its capability to operate as a complete end-to-end solution.

The results from both applications demonstrated that the FogETex framework operates in real-time and with robust performance. While the primary goal of the FogETex system is to be utilized in e-textile applications, it can also process signals generated by e-textiles to control textile-based soft robotic structures. Thus, it serves as a framework that encompasses both e-textiles and soft robotics domains. Besides being developed primarily for electronic textile applications, FogETex framework can accommodate other IoT devices as well.

E-TEKSTİL UYGULAMALARI İÇİN SİS BİLİŞİM MİMARİSİ

ÖZET

Tekstil ürünleri, insan yaşamının hemen her alanında yer almaktadır. Elektronik tekstillerin ortaya çıkışıyla birlikte, tekstil ürünleri çeşitli fizyolojik ve çevresel uyarıları elektrik sinyallerine dönüştürme yeteneğine kavuşmuştur ve bunların birçoğu insanlar için hayati öneme sahiptir. Bu nedenle, bu ürünlerin gerçek zamanlı (düşük gecikmeli) ve sağlam bilgi işlem sistemlerine ihtiyaçları vardır. Ancak, konfor gereklilikleri nedeniyle güçlü bilgi işlem kaynaklarına yer verilememektedir.

Bu tez çalışmasında, e-tekstil uygulamalarının ihtiyaçlarını karşılamak için yeni bir sis bilişim tabanlı çerçeve (FogETex) önerilmiştir. FogETex, platformlar arası destek sunan, ölçeklenebilir ve gerçek zamanlı çalışan bir Hizmet Olarak Platform (PaaS) modelidir. Bu çerçeve, Tekstil tabanlı Nesnelerin İnterneti (T-IoT) cihazları, sis cihazları ve bulut dahil olmak üzere sistemin uçtan uca entegrasyonunu kapsamaktadır.

FogETex çerçevesi, uç katman, sis katmanı ve bulut katmanı olmak üzere üç katmanlı bir mimariden oluşmaktadır. Uç katman, e-tekstil sensörlerinden veri toplayan ve bu verileri ağ geçidi cihazına ileten T-IoT cihazlarını içerir. Ağ geçitleri genellikle kullanıcıların günlük yaşamlarında taşıdığı mobil telefonlardan oluşmaktadır. Bu cihazlar, toplanan verileri sis katmanına iletmek ve işlenen verileri görselleştirmekle sorumludur. Eğer T-IoT cihazı kendi Wi-Fi veya LTE modülüne sahipse, bir ağ geçidi cihazına ihtiyaç duymadan verileri doğrudan sis katmanına iletebilir.

İkinci katman, aracı (broker) ve işçi (worker) cihazlarını içerir. İşçi cihazı, gelen işlem taleplerini yönetmekten sorumluyken, aracı cihaz sis düğümünü yönetir. Aracı, işçi cihazları tarafından düzenli aralıklarla gerçek zamanlı olarak gönderilen kaynak kullanımı verilerini izler ve herhangi bir cihazın aşırı yüklenip yüklenmediğini belirler. Kaynak kullanımına bağlı olarak, yeni bir kullanıcı sis düğümüne bağlandığında, aracı en uygun işçi cihazını atar. Güvenlik nedenleriyle, sis düğümü içinde yalnızca aracı cihazın Geniş Alan Ağı'ndaki (WAN) cihazlarla bağlantısı bulunur. Bu nedenle, dış mekan uygulamalarında veriler aracı cihaz aracılığıyla işçi cihazlarına aktarılır. Bu yapıda, aracı, işçi cihazları ile kullanıcılar arasında bir vekil (proxy) görevi görür.

Üçüncü ve en üst katman, buluttur. Bulut cihazı, aracının sağladığı kullanılabilirlik bilgilerine dayalı olarak kullanıcıları uygun bir sis düğümüne atar. Bulut, uygun sis düğümünü belirlerken, sis düğümündeki işçi atamalarına müdahale etmez. Bu yapı, merkeziyetsiz bir yönetim sağlar. Bir düğüm arızalansa bile, diğer düğümler bağımsız olarak görevlerini yerine getirmeye devam edebilir. Ayrıca, bulut ve aracı cihazları, birincil sorumlulukları olan yönetimin yanı sıra, işlem hizmetleri sağlama kapasitesine de sahiptir. Bu nedenle, sistem aşırı yük altında olduğunda, bu cihazlar kullanıcıları hizmet vermek için devreye girebilir ve işlevselliğin devamını sağlar.

E-tekstil sensörleri zaman serisi verisi ürettiği ve birçok sensör saniyede onlarca veri noktası topladığı için, ağ geçidi cihazı ile işçi cihazı arasındaki iletişim WebSocket yapısı kullanılarak kurulmuştur. Bu yaklaşım, her veri iletimi için bağlantıların tekrar tekrar kurulmasına gerek kalmadan, asenkron ve iki yönlü veri akışını mümkün kılar. Diğer taraftan, kullanıcının buluta veya aracıya yaptığı cihaz tahsis talepleri gibi işlemler bir kerelik işlemler olup, bu amaçla özel olarak geliştirilen bir RESTful API üzerinden yönetilmektedir. Ayrıca, her cihaz, sistem yöneticilerinin cihazların durumunu izlemelerini sağlayan bir kullanıcı arayüzü ile donatılmıştır. Bu veriler, aşırı yüklenmiş sis düğümleri için ek cihazların sağlanmasına yönelik kararlar alınmasında kullanılabilmektedir ve sistem performansının optimal seviyede tutulmasını sağlamaktadır.

Bu tezin hayata geçirilmesi ve e-tekstillerin doğasının anlaşılabilmesi için, adım fazı tespiti ve el haraketi yakalama gibi alanlarda elektronik tekstiller kullanılarak çeşitli uygulamalar geliştirilmiştir. Öte yandan, geliştirilen çerçevenin yalnızca bir veri işleme platformu değil, kapsamlı bir uçtan uca sistem olarak işlev görmesini sağlamak amacıyla, akıllı tekstillerin bir diğer alanı olan tekstil tabanlı yumuşak robotlar üzerine de araştırmalar yapılmıştır. Bu çalışmalar, kas zayıflığı olan bireyler için dış iskelet eldivenleri sistemlerini içermektedir. Bu uygulamalardan seçilen vaka senaryoları, FogETex sistemini test etmek için kullanılmıştır.

Önerilen çerçevenin ilk uygulaması olarak, tekstil tabanlı kapasitif sensörler kullanarak derin öğrenme tabanlı bir yürüme evresi analiz uygulaması kullanılmıştır. Bu vaka çalışmasında, diz hareketleri, test konuğunun dizine yerleştirilen tekstil tabanlı kapasitif bir sensör ile yakalanmıştır. Sensör verileri, derin öğrenme tabanlı bir makine öğrenmesi yöntemi kullanılarak yürüme evrelerine dönüştürülmüştür. Bir sonraki aşamada, bu yürüme evresi verilerinin, ayak düşüklüğü tedavisi için geliştirilen yapay kas aktüatörü için kontrol sinyalleri olarak kullanılması planlanmaktadır.

FogETex, ideal sistem performansını belirlemek amacıyla bir sahte istemci kullanılarak ve gerçek dünyadaki testleri yapmak için gerçek bir istemci kullanılarak, zaman özellikleri, kaynak kullanımı ve ağ bant genişliği kullanımı açısından değerlendirildi. Tüm bu testler, iç mekan uygulamaları için doğrulama yapmak amacıyla işçi, aracı ve bulut cihazları üzerinde tekrarlanmıştır. Ayrıca, dış mekan uygulamaları için işçi ve bulut cihazları WAN üzerinden bağlanarak testler gerçekleştirilmiştir. Aracı cihaz, bu testte işçi cihazı ile kullanıcı arasında bir vekil olarak görev yapmaktadır. Sis cihazları, bu metriklerde bulut sistemini geride bırakmıştır.

Bu vaka senaryosunda, FogETex çerçevesinin performansı, tek bir sensörle yapılan uygulamalarda farklı cihazlar arasında analiz edilmiştir. Ayrıca, çerçevenin birden fazla kullanıcıyı yönetme kapasitesini değerlendirmek için bir stres testi yapılmıştır. Yapılan testlerde, işçi cihazlarının 6 kullanıcıya kadar hizmet verebildiği, aracı cihazlarının 18, bulut cihazlarının ise 14 kullanıcıya kadar hizmet verebildiği bulunmuştur. Sistem, üç veya daha fazla işçi cihazı kullanıldığında, diğer yapılandırmalara kıyasla üstün performans sergilemiştir. Kiralama ve cihaz maliyetleri göz önünde bulundurulduğunda, işçi cihazlarının performans açısından daha maliyet etkin olduğu değerlendirilmiştir. Son olarak, FogETex çerçevesi, literatürdeki diğer sistemlerle karşılaştırılmış ve bu sistemlerin, çeşitli çalışmalarda yaygın olarak kullanılan rakipler olarak değerlendirildiği görülmüştür. Karşılaştırma, FogETex'in gecikme, yürütme süresi, yanıt süresi ve işlem frekansı gibi metriklerde rakiplerini geride bıraktığını ortaya koymuştur.

Önerilen FogETex çerçevesinin çok yönlü uygulanabilirliğini daha da göstermek için e-tekstil ve tekstil tabanlı yumuşak robotik sistemlerin entegrasyonunu içeren bulut tabanlı bir uzaktan manipülasyon sistemi geliştirilmiştir. Bu araştırmanın amacı, geliştirilen kontrol sistemi mimarisi kullanılarak kablosuz olarak çalıştırılan, tekstil tabanlı sensörlü bir eldiven ile hava tahrikli bir yumuşak robotik eldiveni birleştirmektir. Her bir parmağa kapasitif sensörler yerleştirilmiş sensörlü eldiven, tıbbi personelin el hareketlerini yakalar. Bu sırada, felç, beyin yaralanması veya omurilik yaralanması nedeniyle el fonksiyonu bozulmuş hastalara yardımcı olmak amacıyla tasarlanan pnömatik rehabilitasyon eldiveni, tıbbi personelin hareketlerini taklit etmektedir. Önerilen yapay zeka tabanlı sistem, parmak jestlerini algılar ve pnömatik sistemi harekete geçirir, ortalama yanıt süresi 48.4 ms içinde yanıt vermektedir. Sistemin doğruluk ve iletim kalitesi metrikleri açısından yapılan değerlendirme, tekstil eldivenlerinin IoT altyapısına entegrasyonunu ve uzaktan hareket algılama ve aktüatörlük sağlama işlevselliğini doğrulamaktadır.

Öte yandan, bu araştırma, FogETex sistemini çoklu sensörlü e-tekstil uygulamalarında da test edilmesini sağlamıştır. Farklı makine öğrenmesi yöntemleri kullanılarak geliştirilen modeller, sistemi farklı uygulamalar olarak tanıtarak, çerçevenin birden fazla uygulamayı aynı anda çalıştırabileceğini göstermiştir. Çerçeve, bir sis bilişim mimarisi olarak tasarlanmış olsa da, yalnızca bir bulut veya uç bilişim sistemi olarak da çalışabilmektedir. Bu çalışmada, çerçevenin sis cihazları olmadan da etkili bir şekilde çalışabileceği doğrulanmıştır. Ayrıca, geliştirilen sistem, e-tekstil ve yumuşak robotikleri başarıyla entegre ederek, tamamlayıcı bir uçtan uca çözüm olarak çalışma kapasitesini kanıtlamıştır. Bunlara ek olarak sistem farklı eşzamanlılık ve prosesler arası haberleşme methodları ile test edilmiştir. Sistem ek olarak, çoklu işçi cihazı ile de test edilmiştir. Sistemin, 1 işçi cihazı ile 10, 2 işçi cihazı ile 22, 3 işçi cihazı ile 26 ve bulut sisteminde de 23 cihaza kadar hizmet verebildiği gözlemlenmiştir.

Her iki uygulamadan elde edilen sonuçlar, FogETex çerçevesinin gerçek zamanlı çalıştığını ve sağlam bir performans sergilediğini göstermiştir. FogETex sisteminin birincil amacı, e-tekstil uygulamalarında kullanılması olmakla birlikte, aynı zamanda e-tekstiller tarafından üretilen sinyalleri işleyerek tekstil tabanlı yumuşak robotik yapıları kontrol edebilmektedir. Böylece, hem e-tekstiller hem de yumuşak robotik alanlarını kapsayan bir çerçeve olarak hizmet vermektedir. Başlangıçta elektronik tekstil uygulamaları için geliştirilmiş olmasına rağmen, FogETex çerçevesi diğer IoT cihazlarını da barındırabilmektedir.

1. INTRODUCTION

Textile products hold a prominent role in the routine lives of individuals. From the onset of the day, people engage with various textile materials, and this interaction persists until they retire to bed at night. Notably, even during their sleep, individuals remain in contact with textiles. The advent of electronic textiles (e-textiles) has made it possible to transform these interactions into digital data, facilitating human motion, gaming, pressure mapping, rehabilitation, healthcare, smart wearables, and smart garments [1]. On the other hand, textile products can be operated as actuators by integrating controlled deformation features into the fabric. In contrast to conventional rigid sensors and actuators, e-textile products offer the advantages of being lightweight, soft, breathable, and comfortable [2]. Furthermore, thanks to the significant advances in mass production through centuries of textile development, the use of textile products is expected to rise progressively. This advancement not only increases the production capacity of electronic textile products but also amplifies their overall positive attributes, thereby improving their potential.

The collaboration of individuals from different engineering and science fields within electronic textiles technology has led to the emergence of various technologies and products in sectors such as healthcare [3], aerospace [4], entertainment [5], agriculture [6], and education among many others. The development of health-focused products within the field of electronic textiles enables the collection of vital health data, including ElectroCardioGram (ECG), ElectroMyoGram (EMG), and respiratory information. Furthermore, the integration of pneumatic artificial muscle [7] and exoskeleton actuators [8] into electronic textile technology aims to enhance the quality of life for patients. Sensors within the garments worn by aerospace passengers allow for sensing capabilities, additionally providing haptic feedback to the individual wearing the suit [9]. In the realm of entertainment, there are diverse application examples utilizing electronic textiles technology, including game controllers [10] and interactive education game mats [11]. In addition, the utilization of textile displays as interactive interfaces improve activities such as sports, making them more enjoyable and engaging [12]. In the field of agriculture, electronic textiles technology is used in various applications, including monitoring chloride levels in the soil [13] and enabling the gentle harvesting of sensitive fruits through the use of sensors [6]. In the domain of education, there is a focus on enhancing learning experiences for preschool children through the implementation of interactive educational methods [14].

The rapid expansion of electronic textile applications causes an increase in data and processing load [15]. To deal with this processing load, it is essential to establish appropriate architectural solutions. Electronic textile applications are expected to operate in real-time, and the textile products in which sensors and actuators are embedded often have low battery capacity due to comfort considerations. The energy harvesting studies [16]–[18] within the domain of electronic textiles are potential candidates for solving the battery problem in sensing electronic textile applications. However, the amount of energy they can generate with current technology is not high enough to power circuits, so they cannot yet offer a solution to this problem [19]. Efforts are made to prolong battery life by incorporating low-energy microcontrollers in the electronic circuitry of textile products. However, tasks that demand high processing power and energy, such as data processing and machine learning, should be offloaded to dedicated computing architectures. Although traditional sensor-to-cloud architectures are candidates for meeting needs due to their characteristics such as scalability and high performance, they will not be suitable for electronic textile applications due to factors such as intermittent delay, high bandwidth requirements, and security concerns. Network issues or errors in the data center can lead to inaccurate results, posing a risk to human safety, particularly in healthcare applications. Such inaccuracies may potentially result in unintended injuries or adverse effects.

Fog computing systems are well-suited for electronic textile applications due to their advantages. These systems offer real-time computation capabilities, enhanced security measures, flexible placement options for computing units, reduced bandwidth requirements, improved energy efficiency, support for the user or patient mobility, and seamless integration with existing infrastructure [20]. As a result, fog computing systems provide an optimal choice for addressing the unique requirements of electronic textile applications.

1.1 Purpose of the Thesis

In the literature on resource-constrained tiny IoT devices, rigid sensor structures are often encountered. In contrast, electronic textile devices are flexible, with a primary focus on comfort constraints. The flexibility and comfort provided by textile structures also introduce challenges, including sensor-circuit connection issues, low resolution, stability issues, cycle errors, and operational range limitations. Due to comfort considerations, the data acquisition and transmission circuits need to contain as few components as possible, to make them small and lightweight. Due to the vastness of these challenges, many studies focus solely on individual components, such as sensors, and remain at the proof-of-concept stage in laboratory settings without transitioning to practical applications [1]. Furthermore, systems developed for e-textile applications must inherently operate effectively in both indoor and outdoor environments, possess low latency, facilitate real-time data processing, and maintain session information along with time-series data processing. To our knowledge, there is currently no framework in the literature that utilizes a fog computing architecture specifically tailored for electronic textiles.

The primary aim of this thesis is not only to develop a fog computing framework specialized for e-textile uses but also to create applications to test this framework. It is essential to understand their characteristics to design a system tailored to the nature of e-textile applications. For this purpose, various systems such as gait phase recognition system and hand motion recognition system have been developed using electronic textiles, making this one of the most significant objectives of the thesis. Consequently, the framework has not been developed in a theoretical vacuum. While the various applications reinforce the framework's foundation, it aims to unify all these efforts under a single umbrella.

While many studies focus on a specific problem area and provide solutions, they often fail to translate into real-world applications. In this thesis, the goal is not merely to process sensor data; rather, it is to utilize processed data for purposes that can simplify people's lives. Textile structures are uniquely suited to soft robotics, as they can directly interact with people and enable beneficial applications. Therefore, establishing systems that process signals from e-textile sensors to control textile-based soft robotic structures is also an objective of this thesis. Through this work, the goal is to realize end-to-end systems where sensing, computing, and actuating function in unison. In this way, as shown in Figure 1.1, focus area of the thesis is to develop human-centered systems by integrating textile, health, electronics, and computer sciences.

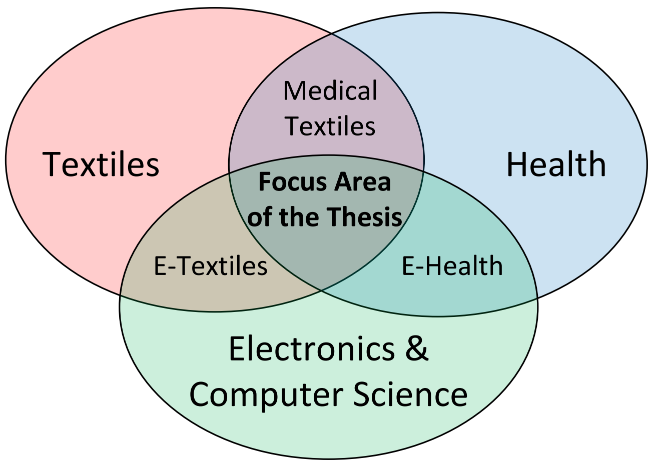


Figure 1.1 : Focus area of the thesis.

1.2 Contribution of the Thesis

In this thesis, a fog computing framework is developed specifically tailored for electronic textile applications. This framework consists of edge, fog, and cloud layers. The edge layer is where data is generated from e-textile products. In the fog layer, the worker device is responsible for processing the data produced in the edge layer, while the broker manages the devices in the fog node. The cloud layer is responsible for overseeing the entire architecture. The framework designed can be used for both indoor and outdoor applications. Deep learning-based applications involve a high amount of multiply and accumulate operations and memory access [21]. Therefore, a deep learning-based gait phase recognition application using Textile-based Internet of Things (T-IoT) device with capacitive strain sensor data was utilized to test the performance of the developed framework under high computational load. The framework was tested using various experimental scenarios in terms of its time performance, resource usage, and network bandwidth usage. In addition to these, a system stress test and a performance comparison with a similar study were also conducted.

The main contribution of this thesis to the literature is the development of a framework specifically tailored for electronic textiles using a fog computing architecture. This framework serves both indoor and outdoor clients in real-time with low response times to meet the needs of e-textile applications. Additionally, the system retains session information for processing time-series data, and data flows continuously. This contribution is particularly relevant as it addresses the growing demand for efficient, low-latency solutions in wearable technologies, an area where existing fog computing frameworks often fall short in real-world, distributed environments. In contrast to numerous fog computing frameworks, the team has developed all software components involved in the framework, ranging from sensor acquisition to cloud infrastructure, as well as the hardware of the T-IoT device. A typical use case, which utilizes a deep learning-based model to process time series data generated from textile-based sensors, has been applied within the framework to operate in real-time. One of the most significant contributions of this thesis is the introduction of the T-IoT device concept to the literature for the first time.

To develop a framework suitable for e-textiles, it is essential to thoroughly analyze their characteristics. For this purpose, gait phase recognition and hand motion recognition applications were developed using textile-based capacitive sensors. In the gait phase recognition system, deep learning was utilized to track step phases from single-sensor data. On the other hand, the hand motion recognition system implemented a multi-sensor electronic textile application. Additionally, to gain a broader perspective on e-textiles, a review paper focusing on security and privacy has been prepared, contributing to the literature.

Strengthening the foundation of this thesis, a deep learning-based gait phase recognition system [22], which involves high computational load, was chosen to test the developed fog computing system. This application provided a basis for conducting the necessary tests for e-textile applications. The test results confirmed that the developed framework can serve with low latency in both indoor and outdoor applications. Through this thesis, a fog computing system tailored to the needs of e-textile applications has been developed. Although the system is specifically focused on e-textiles and fog computing, it is also adaptable to other sensor and computing architectures.

Within the scope of this thesis, a telerehabilitation application has been developed using a sensing T-IoT glove and an actuating T-IoT glove to save time for patients and medical staff. This application enabled the holistic testing of the developed framework, encompassing both sensing and actuating functionalities. Implemented on the cloud to overcome distance limitations, the application demonstrates that the framework is not restricted to fog systems but can operate on various platforms independently, proving its adaptability across different systems. Additionally, with this application, the system was also tested in the fog environment using various concurrency and inter-process communication techniques. Furthermore, the system was tested as a whole with a multi-worker setup.

1.3 Organization of the Thesis

The thesis consists of a total of six chapters and is organized as follows: Chapter 2 provides a literature review on electronic textiles, textile-based actuators, and cloud, edge, and fog computing. Chapter 3 presents a detailed explanation of the FogETex framework developed for e-textile applications. In Chapter 4, the gait phase recognition system is described. In this chapter, the gait phase recognition system is integrated into the FogETex framework, and its suitability for indoor and outdoor applications is tested. Chapter 5 focuses on the assistive soft robotic glove, where a telerehabilitation application is developed using sensing and actuating T-IoT gloves. The developed framework is employed within this application under cloud computing. Finally, Chapter 6 concludes the thesis and presents directions for future work.

2. LITERATURE REVIEW

In this chapter, the literature has been reviewed under two main headings: electronic textiles and computing systems. In Section 2.1, the focus is on the specific area of this thesis, examining electronic textile structures, including sensors and actuators. In Section 2.1.1, cloud computing, edge computing, and fog computing systems are investigated.

2.1 Electronic Textiles

This thesis focuses on the sensor and actuator structures of electronic textiles. First, attention is given to e-textile sensors, which serve as the source of data. Information is provided on textile-based resistive, capacitive, and inductive sensors, as well as textile electrodes. Second, the study focuses on textile-based actuators designed to manipulate the environment or target objects to provide practical benefits. This section examines cable-driven, fluidic, and shape-changing actuators.

2.1.1 Sensors

The evolution of conductive yarn and fabric technologies has led to the integration of electrical current within textile products. In its initial stages, conductive fabrics were primarily conceived to leverage textile materials as heating elements. Hence, the primary objective was to address the issue of copper cable breakage arising from the repeated bending, folding, and unfolding associated with electrical blanket products available in the market [23]–[25]. Subsequently, with the advent of conductive yarn technology, it became feasible to create small conductive pathways within textile materials [26,27].

Over time, the evolution of conductive yarn and fabric technologies facilitated the emergence of electronic textiles, enabling the incorporation of sensory capabilities into textile products [28]. These advancements allowed for the detection of various stimuli—such as pressure, force, stretching, optical changes, chemical attributes, as well as monitoring temperature and humidity in the external environment—within textile products. Electronic textiles serve as an efficacious means to imbue fabrics with the capability to perceive diverse physical stimuli and responses [29]. Certainly, instances of electronic textile applications manifest across various domains, including commercial spheres [30], medical applications [31,32], military contexts [33], and aerospace industries [34]. The scholarly literature continually expands with the addition of novel electronic textile applications on a daily basis. Electronic textiles are categorized into resistive sensors, capacitive sensors, inductive sensors, and textile electrodes based on their electrical reactions to various physical stimuli. This section will analyze the various types of e-textile sensors and their respective applications across different usage domains.

2.1.1.1 Textile based resistive sensors

Textile-based resistive sensors are characterized by alterations in their resistance values in response to environmental stimuli, encompassing factors like force, temperature variations, magnetic fields, chemical influences, and optical changes. The alteration in resistance can be quantified through suitable electronic circuits and subsequently utilized as data in diverse applications.

Resistive sensors are categorized into various types based on their physical responses, including piezoresistive sensors, which react to pressure; thermoresistive sensors, sensitive to temperature changes; magnetoresistive sensors, affected by magnetic fields; chemiresistive sensors, responsive to chemical stimuli; and photoresistive sensors, influenced by optical variations [35]. This part concentrates on piezoresistive and thermoresistive sensors, commonly applied in human activity recognition methodologies.

Piezoresistive sensors: They exhibit alterations in resistance due to pressure or mechanical stretching exerted upon them. Pressure sensors find application in various fields such as touch and grip detection [36,37], posture detection [38], and analysis of plantar pressure distribution [39]. These sensors identify and measure applied pressure within these contexts. Motion capture systems utilize textile-based sensors to determine joint angles, such as those of fingers [40], knees, elbows [41], and shoulders [42], by measuring the stretching or elongation of the sensors. Indeed, through sensor stretching, it is feasible to acquire vital signals such as respiration rate. An example study demonstrates a textile-based resistive strain sensor used for assessing respiratory rate. This sensor facilitates the monitoring of diaphragm expansion, enabling the determination of the respiration rate. The tension applied to the stitches formed with conductive yarn in this sensor causes a reduction in the contact points of these stitches, consequently leading to an increase in the resistance value of the sensor [43]. Simple electrical measurement methods suffice for the evaluation of piezoresistive textile sensors. Notwithstanding the straightforwardness of the measurement technique, these sensors exhibit characteristics such as high response times, low linearity, and considerable hysteresis [44].

Thermoresistive sensors: They are commonly applied for the measurement of human body temperature [45]. In the medical domain, body temperature stands as one of the essential vital signs among the four primary indicators. Heart rate, respiration rate, and blood pressure are the other vital signals. Changes in temperature directly induce variations in the resistance values of these sensors. Thus, by incorporating a measurement circuit into the textile sensor, real-time monitoring of the body temperature of the individual can be achieved, facilitating prompt notifications to individuals or healthcare institutions during critical scenarios [46].

2.1.1.2 Textile based capacitive sensors

Capacitive sensors are formed through the insertion of dielectric materials, such as silicone [47], foam [48], and thermoplastic polyurethane [49], between two conductive fabrics. This configuration operates as a parallel plate electrode. The capacitance of the sensor diminishes as a consequence of either the elongation of the sensor length or the reduction in the distance separating the conductive fabrics. Textile-based capacitive sensors demonstrate distinct characteristics, including low linearity, reduced response time, and heightened hysteresis, distinguishing them from resistive sensors. In contrast to resistive sensors, textile-based capacitive sensors exhibit characteristics marked by low linearity, decreased response time, and reduced hysteresis [50].

Another capacitive sensor production technique involves employing the interdigital sensor technique. In this method, the capacitive sensor electrodes are designed not to overlap but rather in a comb-like structure, allowing them to be designed to interlock with each other. Hence, the change in capacitance is observed as the distance between the “teeth” of the comb structure decreases or increases based on the stretching direction of the sensor. Through this manufacturing technique, the sensor can be produced as a single-layer and thinner structure. Actually, this sensor technique is commonly encountered in products utilizing MicroElectroMechanical Systems (MEMS) technology [51]. Atalay [52] utilized conductive fabric from textiles to form the electrodes and placed silicone material between these electrodes to produce a capacitive interdigital sensor. Martinez-Estrada et al. [53] produced an interdigital sensor using weaving techniques with conductive and cotton threads. In another study, Yilmaz et al. [54] produced an interdigital capacitive sensor solely using knitting techniques, incorporating stretchable conductive yarn and regular yarn. As a result, they were able to create a more flexible sensor in a single piece compared to others, enabling the production of a single-piece, more flexible sensor unit.

As an alternative technique, capacitive sensors can be exclusively fabricated through yarn technology. The inner composition of the yarn comprises conductive silver fibers, while the outer layer consists of a dielectric material like cotton, enabling the creation of capacitance between two or more yarns. As this sensor extends, the centers of the yarns draw nearer, subsequently resulting in an increase in the capacitance value of the sensor [55].

Textile-based capacitive strain sensors find application in gloves designed for finger movement tracking [47], human body motion tracking systems [44,56], as well as in monitoring respiration rates. By employing machine learning techniques to process the data from these sensors, it becomes feasible to develop activity recognition systems [57]. An example study on step length estimation for indoor navigation purposes develops machine learning models to estimate step-by-step movements using textile-based capacitive strain sensors combined with Inertial Measurement Unit (IMU) sensors [58].

Textile-based capacitive pressure sensors enable the detection of touch and force properties, thereby catering to applications within the realm of soft robotics [2]. Sensor arrays can be established by arranging multiple capacitive pressure sensors in a contiguous fashion, facilitating a collective sensing capability. Consequently, such sensor arrays find utility in the advancement of applications like foot pressure mapping [59], position detection [60], gesture recognition [61], fall detection [62], and education game [11].

2.1.1.3 Textile based inductive sensors

Compared to other textile sensor technologies, textile-based inductive sensors represent a more recent technological development. Inductance is established by configuring nested loops in various patterns, including round [63], rectangular [64], and T-shaped [65], utilizing conductive yarn. The inductance produced by the sensor rises correspondingly with an increase in the number of loops incorporated within its design. Instances of applications exist in the literature, including heart rate monitoring [63] and motion tracking [64,65] utilizing textile-based inductive sensors.

An example study demonstrates a textile-based inductive sensor, where loops are configured in a rectangular shape to generate inductance. The sensors positioned around the hip joint area capture the multi-axial angle changes. The process of determining angle values involved employing the random forest regression algorithm subsequent to various data preprocessing stages, including sliding window techniques and feature generation [64].

2.1.1.4 Textile electrodes

Unlike other e-textile sensors that generate electrical responses like resistance, capacitance, or inductance, textile electrodes establish a strong electrical contact point between the human body and measurement modules. Hence, wearing a t-shirt or a textile band embedded with textile electrodes facilitates the reception of diverse body signals—such as ECG [66]–[68], EMG [69,70], and EEG [71,72]—eliminating the need for uncomfortable medical electrodes. Due to the very low voltages and high noise inherent in these signals, the employed conductive fabrics and conductive yarns are anticipated to possess high electrical conductivity. By integrating wireless capabilities into these products, it becomes feasible to develop applications capable of uninterrupted signal data collection from the heart and muscles of patients, transmitting this information directly to emergency health services [73].

An example study presents an ECG monitoring system employing textile electrodes. This system facilitates the instantaneous transmission of the ECG signal to a mobile device through Bluetooth technology. Employing the beat detection algorithm enables the instantaneous extraction of an individual's heart rate information. This data can subsequently be integrated into motion capture systems to create applications like anomaly detection and calorie calculation [73].

2.1.2 Actuators

Actuators are devices that convert the energy provided to them into mechanical energy, such as displacement, rotation, force, or motion. Linear electric motors and hydraulic or pneumatic pistons are examples of actuators commonly found in the industry. However, these products are large, rigid, heavy, noisy, and inflexible structures. While they offer solutions to many problems in the machine industry, they are particularly unsuitable for fragile and delicate applications. Especially in applications involving interaction with humans, there is a need for soft, compliant, lightweight, and silent actuators. Textile-based soft actuators meet this demand. When these actuators come together to form systems, they are referred to as soft robotics [74].

Textile-based soft robotic structures have a wide range of applications. These include locomotion assistance, thermoregulation, grasping and reaching assistance, shape-changing for dressing, haptic [75] and communication, as well as therapeutic compression. Due to the comfort, lightweight, and adaptability features provided by soft robotic structures in these applications, it is an open domain for new applications and use cases. Textile-based actuator structures can be categorized based on their working mechanisms into cable-driven actuators, fluidic textile actuators, and shape-changing actuators [76].

2.1.2.1 Cable driven textile actuators

Cable-driven actuator systems apply force to a textile product by pulling a flexible cable through an external mechanical actuator system. During the development of these systems, cables are fastened to garments using an anchor point [77]. An engine placed on the other end pulls this cable, applying force to the designated point. In these types of actuators, cables made of steel [78], aramid-containing polymers [79], polyethylene [80], and woven belts [81] can be utilized.

Integrated systems within clothing assist individuals in performing challenging muscle movements. Hence, individuals are enabled to perform various joint movements such as walking [82], reaching [83], and grasping [84]. Developed robotic systems can be used to apply rehabilitation therapy to patients. Moreover, haptic feedback systems are developed for virtual world applications to make simulations more realistic. An example study involves a virtual reality application developed for drone simulation, providing haptic feedback to the user [85].

2.1.2.2 Fluidic textile actuators

Fluidic textile actuators differ from cable-driven actuators as they undergo shape changes by expanding with gas, air, or liquid supplied into them, depending on the application. This expansion behavior generates pushing force in the desired area. For these types of actuator systems to function, external systems such as tanks, compressors, and pumps capable of supplying air, gas, or liquid, as well as pipelines for conveying these fluids, are required. Apart from that, an additional layer is integrated into textile products to prevent liquid or gas leakage, forming an artificial muscle. These structures are preferred due to their characteristics such as simplicity, attaining high power, good energy efficiency, and functionality [86].

Fluidic textile actuators are primarily represented by the McKibben/Pneumatic Artificial Muscle (PAM) within the linear actuator group. PAM actuators are designed to assist human movement. In these types of actuators, expansion occurs sideways with air pressure, causing contraction in length. Thus, it generates tensile force [86].

PAM actuators can be used in various applications such as locomotion assistance [87] and upper body joint motion [88].

Another group of fluidic textile actuators is Programming Fluidic Textile Actuators. In these types of actuators, fabric structures are designed to rotate in a specific direction. Thus, complex movements such as bending [89], rotational [90], and lifting [91] can be achieved. Flexible and non-flexible fabric structures are combined in these actuators to create anisotropy. Anisotropy can be achieved through methods such as combining woven and knitted fabrics [92,93], using knitted fabrics with varying flexibilities [94], adding seams to a uniform knitted fabric [95], employing pleated fabric structures‘ [96], and utilizing 3D knitting techniques‘ [97]. Additionally, creating pockets in an accordion shape can also generate pushing force when assembled together [98]. Actuators of this kind can be used in various applications such as grasping [99], skin locomotion [100], locomotion assistance [98], and joint movement [101]. An example study features an exoskeleton glove developed using bending actuators created with the 3D knitting technique. Air bladders are placed between the layers of three-layered knitted actuators, which are composed of a stretchable upper layer and non-stretchable middle and bottom layers. When the air bladder between the upper and middle layers is inflated, the resulting anisotropy causes the actuator to bend. Similarly, when the air bladder between the bottom and middle layers is inflated, the actuator performs an extension movement due to the non-stretchable structure of these layers. By placing five actuators onto a glove, an exoskeleton glove was constructed [8].

2.1.2.3 Shape-changing actuators

Shape-changing actuators incorporate active yarn technology, enabling them to be designed into various shapes by utilizing changes in the length of the yarn with external heat or electrical energy. Within these types of actuators, there are various technologies present; Shape Memory Alloys (SMAs) [102] and Shape Memory Polymers (SMPs) [103] undergo changes in length due to their distinct crystal structures at different temperatures. Carbon NanoTube (CNT) [104]–[106] and Dielectric Elastomer Actuators (DEA) [107] operate by receiving electrical energy to perform the desired movements. Apart from those, some actuators operate by mass transfer due to other environmental factors such as moisture-driven [108] and hygroscopic [109] actuators. In the literature, there are several applications of shape-changing textile actuators such as thermoregulation [110], haptic feedback [111], bending [112], locomotion [113], and lifting [114]. An example study demonstrates the thermoregulation application of shape-changing actuators, where ventilation flaps open to decrease the body temperature in response to perspiration [115].

2.2 Applications

In this section, a literature review has been conducted on the gait phase recognition and assistive soft robotic glove control studies, which were developed to test the FogETex framework.

2.2.1 Gait phase recognition system

Different aspects of the gait cycle can be determined by measuring the motion or tracking the location of the foot by utilizing various sensors including cameras, non-wearable devices such as sensor floors [116], wearable devices and their combinations [117]. In the camera-based approach, the joint angles or limb positions are extracted from each frame of the captured video, or features of the frame are learned by utilizing various machine learning methods [118,119]. Even though vision-based and non-wearable devices provide accurate results in gait phase detection, most of them are only available in specialized laboratory setups [120].

Wearable devices can be divided into two different groups which are force measurement sensors or angular velocity and accelerometer measurement sensors. Force measurement sensors [121] are generally in the shape of an insole, and require the shoe to be worn constantly which is not always applicable to individuals with walking abnormalities [120]. Another commonly used sensor in this field is the [122,123], which performs poorly when the individual paces at a lower speed. Thus, the IMUs are usually combined with other sensors to increase the reliability of the system [124]. Flexible strain sensors are also utilized in gait and posture classification [120], and the main idea is the change in the electrical resistance or capacitance value in accordance with the elongation of the sensor [125].

For gait segmentation, numerous traditional machine learning and deep learning approaches have been investigated in order to assist the decision-making in clinical studies and develop a control system [126]–[128]. Some examples of the widely utilized models in the gait segmentation can be given as Random Forest [129], Support Vector Machine [130], k-Nearest Neighbour [129], Neural Networks [131]. Overcoming the issue of the high dimensionality and variability nature of the data and increasing the reliability of signal segmentation can be defined as advantages of DL over traditional ML techniques [128].

2.2.2 Assistive soft robotic control

The human hand plays a crucial role in our everyday activities, serving as a sophisticated and intricate tool that allows us to accomplish tasks accurately and effectively [132]. Physical and neurological conditions such as stroke, burns, fractures, and ligament injuries can diminish finger range of motion and grip strength, while also increasing joint stiffness, significantly impacting individuals' physical, psychological, and economic welfare, and substantially impeding their capacity to carry out activities of daily living [133,134]. Hand dysfunctions necessitate exercises assisted by medical staff to restore functionality, and success in musculoskeletal rehabilitation relies on numerous factors, including the timing, intensity, and frequency of the exercises [135]–[137].

Many patients face obstacles that prevent them from accessing rehabilitation programs, such as financial constraints or difficulties in reaching therapy centers due to distance. Additionally, a lack of medical staffs restricts individuals from receiving the required rehabilitation for their physical enhancement [138,139]. Particularly during pandemics, medical staff-dependent rehabilitation methods become especially vital [140]. To overcome such hindrances, remote rehabilitation approaches have been developed within diverse IoT scenarios, involving the real-time collection and processing of data through sensors [141,142]. This makes remotely guided training more accessible, affordable, and results-driven [143] thereby influencing the motivation of patients in therapy sessions [144].

Hand exoskeleton robots, which mimic human hand movements and aid in rehabilitation by performing tasks such as grasp-release exercises, are increasingly utilized in medical environments. Robots often controlled remotely through assistive robotic gloves, are also commonly employed in virtual reality applications [145,146]. Clinical research has demonstrated that stroke patients who participate in intensive repetitive movements through robotic hand therapy experience notable enhancements in hand motor functions. Furthermore, their central nervous system adeptly incorporates feedback from multiple senses to aid in motor learning when confronted with hand movements induced from various sources [147,148].

Studies concerning hand functionality can be categorized into two main areas: gesture recognition and motion control [149]. Signals stemming from various sources have been investigated for the control of robotic hands, including activating buttons [150], physiological signals like EEG [151] and EMG [152,153], voice commands [154], eye tracking [155], and even movements of the feet [156]. For instance, vision-based hand tracking utilizes cameras to monitor hand movements by employing machine learning techniques trained on extensive image datasets [157,158]. Another approach involves wearable hand tracking based on Inertial Measurement Units (IMUs) and compasses. This method typically involves attaching accelerometers, gyroscopes, and magnetometers to the hand to measure its orientation, and then reconstructing the hand configuration by collecting angle data from each finger [159].

The earlier developed sensor designs feature components that are difficult to utilize, intricate to manufacture, and susceptible to fragility [160]. In vision-based systems, a fundamental challenge arises when objects fall outside the camera's Field Of View (FOV), which remains unaddressed even with the application of machine-learning techniques [161]. For IMU and compass-based systems, their vulnerability to magnetic field interference makes them impractical to use in close proximity to ferromagnetic objects [159,162]. Given these considerations, flexible, lightweight, and stretchable sensing technologies present a compelling solution as they provide excellent adaptability and can precisely capture the signals produced by fingers, facilitating safe interactions [93,163,164]. Unlike rigid glove systems that may limit finger movement, soft gloves offer flexibility, allowing users with different finger sizes to maintain natural motion without any constraints [165].

Assistive robotic glove systems have the capability to discern intricate hand features, gather low-dimensional data, and achieve faster hand gesture recognition speeds [166,167]. Currently, rigid and soft robotic gloves have been designed to address the need for increased repetitions, utilizing various actuating systems such as pneumatic, hydraulic, electric, and tendon-driven systems to provide mechanical power [168]. Advanced robotic systems feature complex mechanisms that can produce the required forces and movements with precision for hand therapy. Yet, their intricate mechanical design, heaviness, and bulkiness present notable obstacles when developing hand exoskeletons [169]. Soft robots' physical flexibility presents an encouraging answer to the limitations faced by traditional robots in terms of safety and adaptability [170]–[172]. Assistive devices, utilizing soft materials such as textiles [8,93] or elastomers [173], are inherently designed to be safe, lightweight, compliant, and non-restrictive, facilitating prolonged wear [154,174,175].

2.3 Computing Systems

Sensors based on textiles can be manufactured for various applications. These sensors require a framework for collecting, processing, and transmitting their data to end users. However, as user and sensor count in the system grows, it leads to increased data flow and computational load. In this section, various computing systems aimed at addressing these issues are explored.

2.3.1 Cloud computing

Cloud computing emerged due to the widespread accessibility of the Internet, advancements in virtualization technologies, continuous information flow, and the resulting accumulation and processing demands of big data. The first examples of cloud computing date back to the 1960s. Through the concept of "time-sharing," terminal computers are connected to mainframes, enabling multiple users to access computing services from a centralized system [176]. In the 1990s, with the public availability of the Internet and the rapid expansion of the World Wide

Web, significant advancements in data storage and application accessibility occurred, laying the foundation for cloud computing infrastructure [177]. The development of virtualization technologies, such as VMWare, Linux Kernel-based Virtual Machine (KVM), and Xen, in the early 2000s, further optimized physical resources for user utilization. The ability to customize and share a single resource among multiple users allowed cloud computing technology to expand [178].

The establishment of Amazon Web Services (AWS) in 2006 marked a milestone for cloud computing. With AWS, users gained increased access to scalable infrastructure tailored to their needs [179]. Today, cloud computing serves as a critical infrastructure tool in artificial intelligence and big data applications. Its high data processing capacity offers innovative solutions across industries and supports a wide range of applications [180] Key application areas for cloud computing include Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS), data backup and storage, big data analytics, machine learning and artificial intelligence, IoT (Internet of Things), telerehabilitation and healthcare services, media and entertainment, education, as well as finance and banking.

Today, cloud computing enables individuals and organizations to use their technological infrastructure more flexibly and efficiently. Cloud computing consists of three main service models: SaaS, PaaS, and IaaS. SaaS provides users with remote access to software applications. Typically subscription-based, this system allows users to access software over the Internet. Examples of such software include email, calendars, and office tools [177]. PaaS simplifies the development, testing, and deployment processes of applications for developers, without the need for publishing them first [181]. IaaS, on the other hand, is primarily used by system engineers or network architects and includes virtual machine storage solutions. With IaaS, companies can reduce hardware and setup costs [178,180]. These service models offer scalability, flexibility, and cost-efficiency while addressing diverse user needs [182,183].

Cloud computing provides users with reliable, scalable, and cost-effective solutions for data storage and backup. Compared to traditional methods, cloud-based systems offer geographically independent access to data from any location while enhancing security to minimize data loss [184]. In cloud-based storage systems, data is replicated across multiple servers, ensuring protection against physical hardware failures, software malfunctions, or natural disasters [185]. Service providers such as Amazon S3, Google Cloud Storage, Dropbox, OneDrive, and Yandex Disk offer flexible solutions that enable users to store large amounts of data. Furthermore, security measures like data encryption and access management safeguard data privacy [186]. These features are especially beneficial in sectors like finance, healthcare, and education, where data privacy is a high priority [187]. With the automatic backup feature provided by cloud computing, companies benefit from regular data backups, replacing the manual operations traditionally performed by Information Technology (IT) teams. This reduces operational burdens and secures company data more effectively [188]. Cloud storage enables quick and easy data access while maintaining data integrity and minimizing data loss through reliable backup procedures [189]. Consequently, cloud computing technology offers both individual users and organizations secure and economically efficient solutions for data management [190].

In modern technology, artificial intelligence, machine learning, and big data analytics serve as foundational components and are increasingly integrated into every technological domain. Alongside this integration, the demand for computational power continues to grow. Big data analytics enables valuable insights by analyzing large and diverse datasets. Through machine learning, these insights can be transformed into meaningful predictions [191]. Machine learning systems possess the ability to learn from data, allowing for the development of predictive models. Cloud computing systems are responsible for training these models and adapting them as new data becomes available [192]. Artificial Intelligence (AI), in contrast, encompasses a broader concept that includes not only machine learning but also algorithms and systems designed to perform tasks in a manner similar to human capabilities [193]. Together, these three technologies are driving transformative changes across industries, including healthcare [194], finance [195], automotive [196], and e-commerce [197]. For instance, big data analytics and machine learning enable early diagnosis from patient data [198,199], while AI systems facilitate the development of autonomous vehicles [200]. Additionally, cloud computing enables large language models to be deployed and serve millions of users simultaneously [201].

Beyond these applications, cloud computing supports a wide range of other fields. In the realm of IoT, cloud computing plays a crucial role in data processing, data flow, and storage [202]. In telerehabilitation and healthcare, it facilitates interactions between patients and medical staff, enabling applications in patient monitoring, treatment, and diagnostics [203,204]. Another significant area for cloud computing is media [205] and social media [206]. While cloud computing helps disseminate information to millions through media channels, social media platforms transform individuals from mere recipients to active sources of content. Social media enables real-time interactions among billions, with cloud computing systems serving as the backbone of this massive connectivity [207]. In the entertainment industry, cloud systems provide the infrastructure for various areas, including gaming [208], movies [209], and music [210]. In education, cloud computing supports the storage and sharing of educational materials and the operation of online testing systems [211]. Following the COVID-19 pandemic, existing educational systems have leveraged cloud solutions to enable students to continue their lessons online [212]. In finance and banking, users can perform banking services anytime from anywhere without visiting branches. Cloud systems not only provide the infrastructure for these transactions but also implement solutions to ensure their security [213].

Cloud computing technology presents several advantages, including cost-effectiveness, reduced energy consumption, enhanced resource management efficiency, and industry-specific solutions. However, challenges such as security and privacy concerns, service migration, and service continuity also persist within cloud computing [214]. Notably, due to the centralized nature of servers and the distances between users and these servers, latency issues arise, creating the need for additional computational systems to support real-time applications.

2.3.2 Edge computing

Edge computing differs from traditional cloud computing by advocating for computations to occur on edge devices rather than in a centralized manner. While cloud computing involves transmitting raw data to servers for processing, edge computing processes raw data on local edge devices, transmitting only the processed data to the server. This approach reduces the computational load on the cloud, leading to smaller cloud systems and decreased costs [215]. With processing taking place closer to the data source, response times decrease significantly. Furthermore, by prioritizing calculations on nearby edge devices rather than power-intensive devices, energy consumption is minimized within latency constraints [216].

With the proliferation of IoT devices, edge computing has gained traction as a significant topic. Security cameras are omnipresent in streets, squares, homes, schools, and various other locations. The instantaneous transfer of a single camera's video stream to a cloud system poses challenges, especially when considering the simultaneous transfer and processing of video streams from thousands of devices. Edge computing has emerged as a solution, facilitating real-time video analysis for applications like traffic monitoring [217,218], autonomous drones [219], and public safety measures [220].

Edge computing technology enables the control of factory production lines using sensor data, including parameters like temperature, pressure, sound, and Radio-Frequency IDentification (RFID), among others. Based on this sensor data, actuators within the production line are regulated, enabling the detection and removal of defective products from the production cycle. Additionally, through smart meter systems, real-time measurements of energy consumption, production count, and other pertinent information within the production line can be instantly tracked [221,222].

Besides, edge computing finds extensive applications in the healthcare sector [223]–[225]. Utilizing body sensor networks, data regarding body temperature, blood glucose levels, blood oxygen saturation, blood pressure, and post-operative monitoring can be gathered and processed on edge servers, transmitting this vital information to hospitals. This allows for remote patient monitoring, enabling healthcare professionals to conduct assessments even while patients are at home [226].

In addition to these application areas, beyond those application areas, edge computing plays a pivotal role in shaping smart cities through various applications such as short-term energy consumption estimation [227], vehicular networks [228], and traffic optimization [228]. Additionally, the literature highlights its significance in domains like smart homes [215,229], as well as collaborative edge computing [230,231].

2.3.3 Fog computing

Fog computing is an architecture that suggests processing data generated by IoT devices in a decentralized manner on local devices, as opposed to processing it in centralized cloud systems. It is a paradigm characterized by low latency, location awareness, mobility, a large number of nodes, heterogeneity, real-time response, and geographic distribution [232]. With these distinctive features, it distinguishes itself from other computing methods and has been the subject of extensive research. Some of the prominent areas of research include the domains of smart cities, connected vehicles, smart grids, smart homes, healthcare management, wearables, and e-textiles.

2.3.3.1 Fog computing in smart cities

In the context of smart cities, Minh et al. [233] have developed the FogFly prototype, based on fog computing, for adaptive traffic signal control systems, which are a method for addressing traffic jams. The FogFly application utilizes the network topology of iFogSim. It has been observed that the developed prototype outperforms traditional cloud-based methods in terms of latency, energy consumption, and operational costs. In another study, Tang et al. [234] proposed the idea of processing data at the location of its generation, as opposed to centralized computing systems that introduce latency. They addressed the real-time phase timing of single interaction problems using a genetic optimization algorithm. Additionally, they employed a cloud-based system for regional optimization.

Regarding resource management in the Traffic Light System (TLS), Jang et al. [235] proposed a Software-Defined Network (SDN) structure for dynamic resource allocation. This approach was motivated by the increasing vehicle traffic and the corresponding surge in traffic report messages. The proposed SDN system allows for the dynamic allocation of resources, especially during rush hours when resource demands increase and available bandwidth decreases. To reduce both network bandwidth usage and response times, they employed a fog-based architecture, while computation-intensive processes were offloaded to the cloud.

In a different study, Serdaroglu et al. [236] have proposed a location-aware air quality monitoring system for smart cities. Their system can serve up to 960 clients using 120 air quality stations. In addition, Aliyu et al. [237] have developed a fog computing-based shopping recommendation system to enhance customer shopping experiences. Customers connected to the Wi-Fi network in a shopping mall receive personalized recommendations based on their preferences. Price, congestion, and shopping rankings are optimized accordingly. Finally, Talaat et al. [238] propose a fog computing-based fire detection system using the YOLO-v8 algorithm. With this method, it is expected to increase fire detection accuracy, reduce false alarms, and lower costs

2.3.3.2 Fog computing in connected vehicles

Connected Vehicles (CV) aim to reduce traffic congestion and maximize driver safety. Sodhro et al. [239] have developed an AI-based secure and interference-free mobility management algorithm for driver safety and traffic monitoring. They also provided a security and delay-tolerant wireless channel model that enhances the quality of service for passengers. Additionally, they proposed an architecture for reliable and efficient multi-layer fog-based vehicle-to-vehicle communication. Finally, they optimize Quality of Service (QoS) for concepts like mobility, reliability, and packet loss. Zhang et al. [240] have introduced a vehicular network architecture based on software-defined and fog computing to meet the requirements of low latency and high reliability. They have developed solutions that address resource allocation and handover management problems, leading to improvements in these areas.

IoT technology is emerging as the future technology for various application domains, with smart cities being one of its prominent applications. Ning et al. [241] proposes a three-tier vehicular fog computing architecture to minimize latency in collecting event information in the city and transmitting it to vehicles. They have developed a Vehicular Fog Computing enabled (VFC-enabled) offloading scheme for real-time traffic management. In this scheme, both moving and parked vehicles act as fog nodes.

2.3.3.3 Fog computing in smart grid

Due to the limited energy resources on Earth, it is crucial to use energy efficiently. To enhance energy efficiency and effectively track and improve the entire journey of energy from production to transmission and consumption, smart grid systems are being developed. Wang et al. [242] have proposed a fog computing-based dynamic billing system for smart grids to enable real-time pricing of electricity bills. Their focus was on the security and privacy aspects of fog-based systems, and they have introduced a privacy-preserving data aggregation model.

In smart grids, storing excess electrical energy is a challenging task, and surplus energy often goes unused or is lost. Jaiswal et al. [243] have proposed a fog-based system for energy usage prediction in smart grids. In smart grids, the data from smart meters can create bottlenecks in cloud devices due to an increase in the number of sensors, leading to high latency issues. Their suggested system aims to address both latency and the balance between electricity production and consumption, providing a solution to these challenges.

In addition, Forcan and Maksimović [244] proposed a hybrid communication architecture based on cloud-fog computing for voltage profile monitoring and power loss estimation. They prepared and tested their proposed system through simulations using MATLAB Simulink, and it was found that the fog computing-based system significantly reduced the total simulation time.

Finally, Li et al. [245] proposed a dynamic game model for resource allocation in ubiquitous smart grids with finite electrical fog computing devices. This approach aimed to improve resource utilization, optimize resource allocation, and reduce system overheads in order to achieve efficient resource management for fog computing devices in smart grids.

2.3.3.4 Fog computing in smart homes

Due to advancements in the Internet of Things, sensors, and communication technologies, our homes have become smarter. Silva et al. [246] proposed a solution for addressing availability issues caused by the susceptibility to errors in gas leakage and temperature sensors used in smart homes. They introduced a Continuous-Time Markov Chain (CTMC) model in the fog layer to mitigate this problem. Additionally, they aimed to increase availability by combining the CTMC model with sensitivity analysis. Through their modeling efforts, they achieved a high availability rate.

In a different study, Bhatia [247] developed a fog-based framework for monitoring the health of pets in smart homes. The system monitors animals in real-time through sensors such as EMG, ECG, and PhotoPlethysmoGram (PPG), and it can send alerts to caregivers or veterinarians in case of emergencies. They use a deep learning-based temporal-Artificial Neural Network (t-ANN) method to determine the condition and emergencies of animals. Additionally, they have developed a visualization system for pet owners to access their pets' health information on demand.

Furthermore, Gill et al. [248] proposed a resource management method based on particle swarm optimization to improve resource management in fog-enabled cloud systems in smart homes. With their developed resource allocation method, they achieved improvements ranging from 10% to 14% in parameters such as network bandwidth, response time, latency, and energy consumption.

2.3.3.5 Fog computing in healthcare

In the field of healthcare management, Hassen et al. [249] have developed an e-health system based on the IoT and fog computing to address the increasing health issues associated with the growing elderly population worldwide. The developed system utilizes fog computing to collect physiological and general health parameters from elderly individuals at specified intervals. Through the developed Android application, both elderly individuals and their families can monitor the health status of the relevant individuals and communicate with healthcare providers such as system administrators and doctors. The system is also capable of sending recommendations, notifications, and alerts when necessary.

Furthermore, Kamruzzaman et al. [250] have proposed a fuzzy logic-supported machine learning framework based on fog computing for healthcare systems. In this system, machine learning methods are used to analyze ECG data of patients in real-time and predict whether they exhibit symptoms of illness. The fuzzy logic system is employed to determine the capacity of the fog computing system and ensure the efficient utilization of resources. This results in reduced costs, energy consumption, and latency, while improving maintenance efficiency.

Besides, Arunkumar et al. [251] have proposed a fog computing framework that incorporates optimization and ensemble learning for the detection of heart disorders. In this system, patient data is preprocessed, and features are extracted. Using Galactic Swarm optimization, the features are optimized to reduce errors and increase accuracy. Finally, various machine learning algorithms such as bagging, boosting, XGBoost, Multi-Layer Perceptron, and Partitions are used for classification. The majority voting classifier method is employed to combine the results of multiple machine learning techniques, ensuring the best possible predictions. Healthcare systems inherently require a more secure and privacy-preserving environment.

Almas et al. [252] propose a context-based adaptive trust solution for time-critical healthcare systems that incorporate fog computing, utilizing Bayesian and similarity measures. This proposed solution has been simulated using various applications. Due to its linear complexity ($O(n)$), the recommended solution operates more efficiently compared to other solutions.

2.3.3.6 Fog computing in non-textile wearables

Medical wearables collect vital signs from individuals. In this context, Beri et al. [253] have developed a device containing temperature, blood pressure, ECG, and pulse oximeter sensors to monitor the health status of pregnant women. They have proposed a fog computing-based e-healthcare system to process the data and detect emergency health conditions in pregnant women. Real-time tests were conducted on 80 pregnant women using the developed device, and it was determined that it achieved a 98.75% success rate in predictions. In another study, Klonoff [254] recommended the use of fog computing-based systems for diabetes management due to the need for fast responses to sensor data and the inherent intolerance for delays in such systems. They discussed the use of wireless devices such as blood glucose monitors, insulin pens, insulin pumps, and closed-loop control systems in these fog computing-based systems to address the delays caused by cloud computing.

Besides, Monteiro et al. [255] developed a tele-treatment system for monitoring the exercises of Parkinson's patients. In this system, sounds produced by patients during exercise are collected via a smartwatch, and clinical features such as loudness, short-time energy, zero crossing rate, and spectral centroid are extracted on a fog computing device. These features are then transmitted to the clinician via the cloud. To reduce cloud data traffic, high-dimensional audio files recorded for analysis are processed on the fog interface. This system enables the remote tracking of patients' exercises, independent of location. Similarly, Paul et al. [256] proposed a fog-based health monitoring system that utilizes wearable technology products like smart glasses, smartwatches, and fitness bands. Data collected from these wearables are processed at the fog layer, while cloud devices store the data. When a situation requiring action is detected based on predictions, the decision is made and managed on the cloud. The proposed system was simulated using the iFogSim simulator.

In addition to other works, Neel Mani et al. [257] suggested a system for processing and visualizing data on sleep patterns, muscle effort, heart rate, respiration rate, fitness activity, and time tracking from smartwatches through fog computing devices. The system enables users to monitor their activities in real-time and review historical data to stay fit. Additionally, they proposed a five-layer architecture comprising data center/cloud, network, edge domain, smart sensors, and smart monitoring. In another work, Moghadas et al. [258] introduced a fog-based patient monitoring system for arrhythmia detection by processing data from medical electrodes and ECG sensor modules connected to patients via Raspberry Pi. In this system, the ECG sensor module is wired to the electrodes, and the data is collected by Arduino UNO, and transmitted via Bluetooth to the fog computing device. This device detects whether the patient is having a heart attack, and in case of emergency, notifies the patient or physician.

Moreover, Kharel et al. [259] developed a fog-based smart health monitoring system using a pulse oximeter. The system continuously transmits pulse data to a fog server using LoRaWAN communication technology. The fog server stores the incoming data, processes it, and displays the results. Real-time data processing is done on the cloud as the fog server transmits the data instantaneously. Besides, Ijaz et al. [260] proposed a smart healthcare system utilizing wearable biosensors, comprising wearable, intelligent fog, and cloud layers. In the wearable layer, data collected from biosensors is transferred to a Personal Data Assistant (PDA), which detects and eliminates faulty data. In the intelligent fog layer, hidden Markov models are used to assess patients' health, and an alert is sent to family members or medical units when necessary. The cloud layer is used for storing processed data. As a case study, they monitored quarantined patients remotely during the COVID-19 pandemic.

Furthermore, Tuli et al. [261] proposed a blockchain-based framework that integrates IoT-Fog-Cloud from end to end. This framework addresses limitations seen in other Internet of Things frameworks, such as platform independence, security, resource management, and multi-application support. It employs blockchain for authentication and encryption of sensitive data. The system was tested using a pulse oximeter as the IoT device, a smartphone as the gateway, and Raspberry Pi devices as workers. As a case study, they conducted a sleep apnea analysis. Building upon their previous research, Tuli et al. [262] proposed an IoT and Fog computing environment based on ensemble deep learning for the automated diagnosis of heart diseases. They developed a lightweight Fog architecture capable of processing sensor data such as body oxygen, heart rate, ECG, and glucose levels from individuals. In the implementation of the system, they used Raspberry Pi devices as workers and conducted tests using their developed Android-based software.

While many studies on wearable devices focus on healthcare applications, Medina et al. [263] developed a fog computing system for tracking the daily activities of residents in their homes. The system uses accelerometers for movement detection, Bluetooth low energy for location tracking, binary pressure sensors for detecting whether someone is seated or lying down, and cookie sensors to identify activities like brushing teeth or drinking. All these sensors detect residents' activities using a linguistic approach.

2.3.3.7 Fog computing in e-textiles

Beyond all the research areas mentioned so far, the literature on electronic textile studies utilizing fog computing systems is relatively limited. A detailed investigation reveals only two studies in this domain. Constant et al. [264] proposed an architecture for wearable IoT devices that provides fog-based data transmission and filtering services. They deployed their proposed system on Raspberry Pi and Intel Edison development boards and compared the performance of these devices for fog computing. Additionally, they developed a glove for physiotherapists in which they used commercially available film-based flex sensors [265].

In another study, Wu et al. [266] developed textile-based ECG electrodes for medical applications based on fog computing. They optimized the signal quality and comfort parameters of these electrodes. The electrodes are designed to receive signals and transmit data via Bluetooth in an electronic circuit. In tests conducted on 20 different test subjects with no cardiovascular history, the design using a combination of cotton (30%) and nylon fiber-coated silver (70%) demonstrated the best performance in terms of Signal Quality and Comfort (SQC), based on signal-to-noise ratio and comfort surveys. However, the integration of these electrodes into a fog-based system is suggested as future work.

2.4 Positioning of the Thesis

Considering the current state of the art, the development of a fog-based framework for e-textile products emerges as an open research area. The FogETex framework has been developed for electronic textiles, filling the entire gap from developing e-textile-based IoT devices to integrating fog-cloud computing, providing an end-to-end solution. This framework ensures that e-textile applications operate in real-time in both indoor and outdoor environments. Additionally, the system processes time-series data in real-time while maintaining session information for users, with data flowing continuously. Unlike many other fog computing systems, in this thesis, all software and hardware components, from sensor acquisition to the cloud infrastructure, have been implemented. In the first application, a gait phase recognition system is implemented in real-time using a capacitive sensor placed on the ankle, along with a deep learning-based model. In the second application, a system where soft robotics and e-textile sensors are integrated with a computing system has been implemented to work end-to-end.

3. FogETex FRAMEWORK

The FogETex framework provides a platform-independent fog computing and network architecture specialized for electronic textile applications. Figure 3.1 illustrates the general overview of the system architecture proposed in this study. The edge layer, positioned at the bottom of the system hierarchy, encompasses T-IoT devices, actuators, and gateway devices. The middle layer, known as the fog layer, contains both worker and broker devices. At the top layer, there is a cloud device to which all other devices are connected. The fog computing system caters to Local Area Network (LAN) devices in settings such as homes, hospitals, and office spaces. Additionally, it has the capability to establish connections with the fog infrastructure over a Wide Area Network (WAN) using various network technologies including 4G, LTE, 5G, ADSL, VDSL, and Fiber Internet. As a result, the system continues to operate seamlessly in diverse environments such as central parks, forests, and beaches. The hardware components, software elements, and network architecture constituting the system are elaborated upon.

3.1 Hardware Components

This section provides an explanation of the physical devices mostly comprising edge and fog layers of the FogETex framework, as well as their interrelationships.

3.1.1 Textile-based IoT devices

Textile-based IoT (T-IoT) devices in the edge layer are responsible for generating data within the system. These devices provide physical or electrical signals via textile-based sensors or textile electrodes. Sensors can be capacitive, resistive, or inductive depending on their application areas. In addition, conductive fabrics can be utilized as electrodes to capture signals from the human body, such as those originating from the heart, muscles, or brain. This allows for the establishment of an electrical

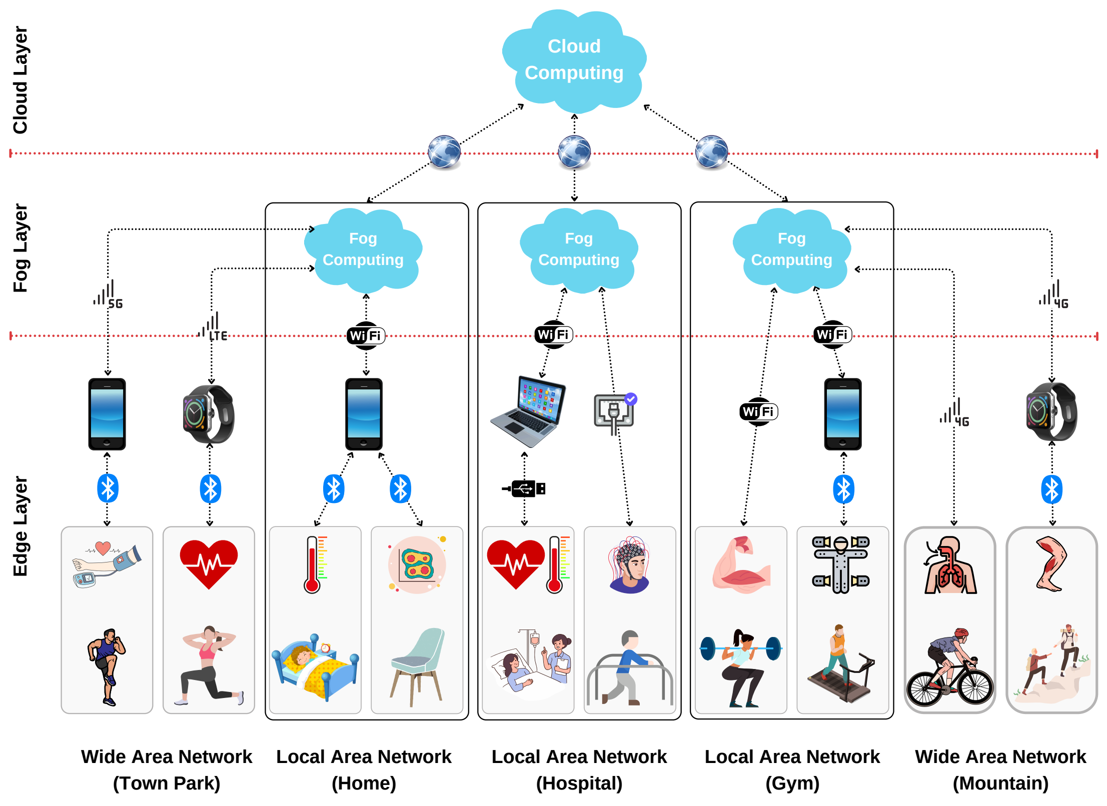


Figure 3.1 : General Overview of the Proposed System Architecture.

connection between the body and the electronic circuit, enabling the acquisition of these signals.

The signals from these sensors or electrodes can be collected directly by appropriate electronic circuits connected to a microcontroller or a microprocessor. The gathered textile sensor data can be transmitted to the gateway device via Serial Port, Bluetooth, Wi-Fi, or directly to the broker/worker nodes within the fog layer. The microcontrollers in this layer operate in close integration with the sensors. Comfort is one of the most crucial parameters in textile-based products; therefore, in the design of electronic circuits for e-textile applications, it is essential to use lightweight and flexible materials as much as possible. Hence, heavy batteries cannot be employed. To ensure extended battery life, energy-efficient microcontrollers are employed. Indeed, raw data is typically transferred directly to the gateway device. In communication with the gateway device, which is one layer above, low-energy communication protocols such as Bluetooth Low Energy (BLE) are preferred to extend battery life. The T-IoT device and the gateway device form a Personal Area Network (PAN) using BLE. If the fog node and the gateway device are on the same network, they connect via LAN; if not, they connect through the WAN. The gateway uses Wi-Fi for indoor activities and LTE for outdoor activities.

3.1.2 Gateway devices

Gateway devices are responsible for connecting T-IoT devices to the Internet. They serve as a bridge between the fog layer and T-IoT devices. They transmit the data generated by textile sensors to the fog devices, receive and process the data in the fog layer, and are also responsible for storing and visualizing the processed data. Devices in this layer include those with wired or wireless Internet connectivity such as smartphones, computers, and smartwatches. If the T-IoT device is equipped with a module that enables direct Internet connectivity, such as Wi-Fi, Ethernet, or LTE module, it can also act as a gateway device. In this case, it can directly connect to the fog layer and manage data communication on its own. However, since e-textile products are typically mobile and have low energy capacity, communication protocols such as BLE are preferred over high-energy-consuming modules like Wi-Fi and LTE. Due to the mobile nature of e-textile products and the limited range of BLE, Wi-Fi or LTE is used for communication between the fog and edge layers. For all these reasons, a device equipped with BLE, Wi-Fi, and LTE communication technologies is preferred. The smartphones are the most suitable candidates for this role. Therefore, a mobile application developed on the smartphone is responsible for receiving data from T-IoT devices, transmitting it to the fog layer, and visualizing the incoming data.

In the communication scenario, if there is a fog node established within the LAN to which the gateway device is connected, it primarily receives services from there. In the event that there is no fog node in the LAN or if the devices in the fog node are overloaded, a connection is established with the nearest available fog node outside the LAN. For instance, in Figure 3.1, individuals exercising in the town park are connected to a fog node located at home which is the closest available node, enabling them to access services. On the other hand, individuals in the mountains are connected to a fog node in the gym assigned by the cloud node.

3.1.3 Broker nodes

Each fog node is composed of worker nodes and only one broker node. The broker node serves as the manager for its respective fog node as shown in Figure 3.2. The initiation of a broker device in the network leads to the creation of a fog node. Worker devices can connect to the fog node by establishing a connection with the broker device in the respective network. The broker node continually receives work requests from the gateway device and assigns the workload to the available worker nodes with the least load. To perform this task, it periodically collects CPU and memory usage information from the worker devices through configurable intervals. The broker node also promptly conveys the availability information of the fog node to the cloud node. Consequently, when a new user arrives, the cloud node can assign them to the nearest and least congested fog node.

Additionally, data from devices connected via the WAN is routed to the worker nodes through the broker node. As a result, even if the broker device were to experience a failure due to an external attack, the other worker devices could continue to provide services to the devices within the LAN.

The broker device in the fog node can have low performance, similar to that of the worker devices, depending on the number of devices receiving services from the WAN. The greater the load on the fog node and the amount of data passing through the broker, the more powerful a broker will be needed.

3.1.4 Worker nodes

The worker node provides computation services to users within the system. Worker nodes are assigned to users by the broker node. Subsequently, users establish a socket connection with the worker node. This enables bidirectional communication between the user and the worker node. The worker node performs the computation tasks received from the user and sends real-time computation results back to the same user. While performing data computations, a session record is maintained for each user. This allows each user to access their own data preprocessing buffer and machine learning model. The worker node continually transmits resource usage information to

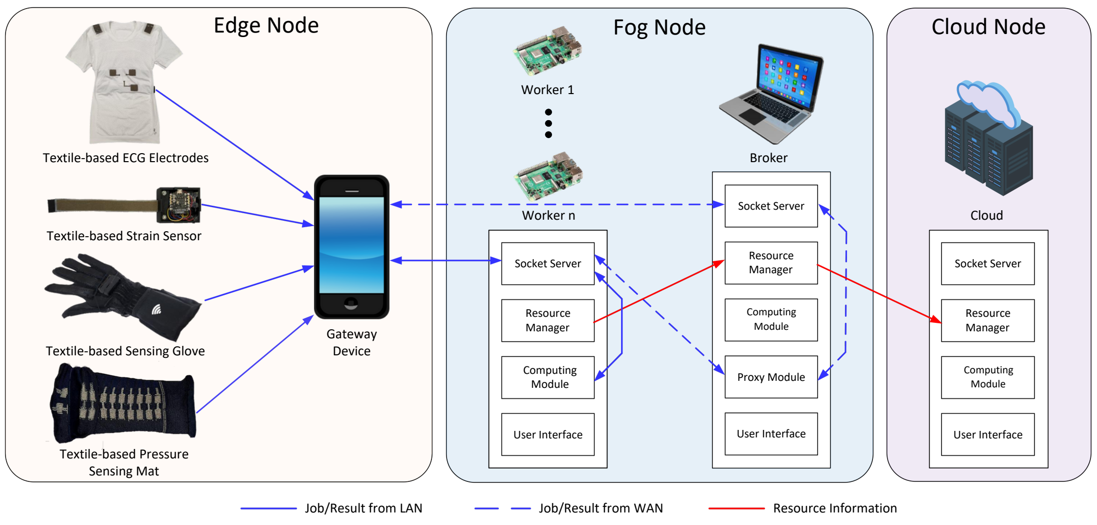


Figure 3.2 : FogETex Hardware and Software Components.

the broker node in real time. This prevents the assignment of new users to nodes that have exhausted their computation resources, thereby mitigating overload issues.

3.1.5 Cloud

The cloud node serves as the central hub for the entire architecture. While individual modules within the system can perform their tasks independently without the cloud node, the presence of the cloud node is essential for newly added worker devices and users to integrate into the system. In order to establish a flexible architecture, new users and devices initially communicate with the cloud node. As a result of a broker node connecting to the cloud node, a new fog node is formed. Worker devices, when initially connecting to the cloud, first learn the IP address of the broker node to which they belong. Then, they establish a connection with the broker node to transmit resource usage information to the broker node. The broker node, in turn, transmits the resource usage information of the worker nodes that connect to and disconnect from the fog node to the cloud layer.

Consequently, when a new user wishes to receive services from the system, they first connect to the cloud, and the cloud assigns them the most suitable fog node. Then, the user requests the assignment of a worker node from the broker node. The user receives computing services from the assigned worker node. In this context, the cloud node acts as the component that regulates and manages the system. Furthermore, system administrators can monitor all broker and worker devices connected to the system and their activities through the user interface provided by the cloud node.

3.2 Software Components

Devices within the FogETex framework require software components, including the resource manager, the computing module, and the proxy module. The computing module, which encompasses the data preprocessing module and deep learning module, provides computational services for T-IoT devices. Additionally, within the framework infrastructure, there is the user interface software component, allowing system administrators to control connected devices and view their resources. Hence, the need for additional worker devices to support the fog node can be easily identified, and overload issues can be detected early on. Figure 3.2 illustrates the included software components within the FogETex framework.

The cloud node, broker node, and worker node execute common software. Device types can be configured for each device during the framework installation using a snippet script with the parameter. This program also stores the device type in a cache file, so even in the event of a configuration change due to an update, the device type can be retrieved from the cache file, ensuring the preservation of device type configuration information. Updates to the software components can be distributed to the nodes via a GitHub repository. System administrators can send update requests to the devices via HTTP RESTful API at specified intervals. The update request triggers a background update script. After the updates are completed, this script retrieves the device type from the cache file and allows the device to resume its operation in the defined manner.

The presence of the same source code on all devices does not necessarily mean that all service devices operate in the same way. The customization of framework software components is based on the device type configuration. In this section, the common and distinct parts of the software components in the service devices are explained.

3.2.1 Resource manager

Resource management is crucial for ensuring system availability and preventing overload issues. The resource manager module plays a role in the proper management of resources and the assignment of the most suitable worker device to users. The primary goal of the FogETex framework is to leverage the computational power of worker devices to provide computational services to users. Additionally, user models, data buffers, and user information are stored in RAM to facilitate fast processing. Intermediate data is also temporarily held in memory during data processing. After the computation process is complete, these data are cleaned from memory by the garbage collector.

In the system, all devices providing computational services continuously monitor the RAM and CPU usage of the device every second. When one of these two values exceeds the predefined threshold, the device reports itself as “busy” to the layer above. Additionally, network bandwidth usage, system uptime, framework operation time, as well as incoming and outgoing request counts, are continuously monitored in real-time within the system. This information is collected for reporting purposes and analysis by system administrators. Resource monitoring and notification intervals can be modified from the configuration file, allowing for more precise or longer-term data monitoring settings. This flexibility enables system administrators to determine the most suitable values of different applications.

Worker devices transmit resource usage data and availability statuses to broker devices, which then utilize this information to assign users to the most suitable worker device. Broker devices also transmit their own resource usage data and the resource usage data of all devices connected to the fog node to the cloud. Furthermore, if all devices in the fog node are busy, the fog node reports to the cloud layer that it is busy. If at least one node is not busy, it reports the node as available. This information is used when assigning users to fog nodes to determine if a node is available or not.

Devices transmit resource and availability information from bottom to top over sockets, eliminating the need to reestablish a connection for each data transmission.

Additionally, when a socket connection is lost, the higher-level device is notified and will refrain from assigning tasks to the disconnected node during allocation processes. When the socket connection is initially established, the respective node shares information such as its local IP, public IP, device coordinates, CPU properties, RAM amount, disk information, and similar details with the higher layer. This information can be utilized for allocation and analysis purposes when needed.

In the system, the allocation of suitable devices is facilitated by the information provided by the resource manager. When a device that was previously assigned to a fog node wishes to reconnect to the system, it can request a new worker device from the broker using its old IP information. However, when a device connects to the system for the first time and needs to find a suitable fog node, it must initially connect to the cloud node and request the assignment of the most appropriate node. At this point, the gateway, worker, and broker devices within the system must have the domain/static IP information associated with the cloud node. Therefore, when setting up a new system with the framework, this information should be entered into the configuration settings of the devices.

Algorithm 1 illustrates the method by which the cloud node assigns a fog computing node to a user. If a fog node exists in the LAN and at least one worker device on that node is not busy, the user is directly assigned to this node, and the local IP address of the broker device on the node is sent to the user. If there are no devices within the LAN, the user is assigned to the geographically closest and available fog node, and the public IP address of the respective network is sent to the user.

To calculate the distance between the user and fog nodes the Haversine formula [267] is used. In this formula, the earth is accepted as a perfect spherical shape. In fact, the earth is oblate spherical and has hills, valleys, and canyons. In the calculation made using this formula, the error is 0.3%, and the maximum error is approximately 22 km [267]. Data on the Internet is transmitted through fiber optic cables in the backbone, and the delay in fiber optics is approximately 5 µs/km [268]. The maximum delay difference that could occur due to the calculation error would be 0.1 ms. Of course, this error is relevant for very long distances; at the distances for which fog

Algorithm 1: Fog Node Assignment

```
Parameter: $IP$, $Latitude$, $Longitude$
Input     : $BrokerDevices$
Output    : $BrokerIP$, $DeviceType$
for $broker$ in $BrokerDevices$ do
    if $broker.PublicIP = IP$ and $!broker.Busy$ then
        $BrokerIP \leftarrow broker.LocalIP$;
        $DeviceType \leftarrow$ “$LAN''$”;
        $return$;
    end
end
$CN \leftarrow null$;
$CNdistance \leftarrow Inf$;
for $broker$ in $BrokerDevices$ do
    if $broker.Busy$ then
        continue;
    end
    $distance \leftarrow Distance(broker, Latitude, Longitude)$;
    if $CN = null$ or $CNdistance > distance$ then
        $CN \leftarrow broker$;
        $CNdistance \leftarrow distance$;
    end
end
if $CN \neq null$ then
    $BrokerIP \leftarrow CN.PublicIP$;
    $DeviceType \leftarrow$ “$WAN''$”;
end
$return$;
```

computing is designed to operate, the latency difference caused by the error is around a few µs. Therefore, since small latency differences cannot affect system performance considerably, the approximate distance is enough for the proposed application. Firstly, the square of half the cord length between two points is specified as a and calculated as follows:

$$a = sin^2(\frac{\Delta\phi}{2}) + cos(\phi_1) \cdot cos(\phi_2) \cdot sin^2(\frac{\Delta\lambda}{2}), \tag{3.1}$$

where ϕ_1 is the latitude of the user, ϕ_2 is the latitude of the Candidate Node (CN), $\Delta\phi$ is the difference of the latitudes of the user and CN, and $\Delta\lambda$ is the difference of the longitudes of the user and CN. Then, the angular distance, θ, between two points is calculated as follows:

$$\theta = 2 \cdot atan2(\sqrt{a}, \sqrt{1-a}). \tag{3.2}$$

Using angular distance and the radius of the Earth, the distance between two points is calculated as follows:

$$d = R \cdot \theta, \tag{3.3}$$

where R is the mean radius of the Earth.

Using the provided broker IP address, the user requests the assignment of a suitable worker node from the broker device. For devices to connect to the broker from outside using the public IP, the port number of the service program must be forwarded to the public IP using the port forwarding method. The method for selecting the most suitable worker node by the broker device is provided in Algorithm 2.

Algorithm 2: Worker Node Assignment

```
Input   : $WorkerDevices$
Output  : $WorkerIP$
$WorkerIP \leftarrow null$;
$CNload \leftarrow Inf$;
for $worker$ in $WorkerDevices$ do
    if $worker.Busy$ then
        continue;
    end
    if $CNload > worker.CPUload$ then
        $WorkerIP \leftarrow worker.LocalIP$;
        $CNload \leftarrow worker.CPUload$;
    end
end
$return$;
```

The broker device assigns the user to the worker node with the lowest CPU load and is not busy among the worker devices connected to the fog node, returning the local IP address of the device. If the fog node assigned to the user is within the LAN, the user directly connects to the worker device. However, if the fog node is on the WAN, the user connects to the worker device via the public IP through the broker.

3.2.2 Computing module

The computing module in the FogETex framework is responsible for executing the tasks requested by users. Computing modules exist in the cloud node, broker, and worker devices within the system. When necessary, computation tasks can be executed not only on worker devices but also on cloud and broker devices. In cases where there are no fog nodes in the system, users can obtain this service from the cloud. Additionally, when the worker devices assigned to their fog node are not available, users can also receive this service from the broker device. Therefore, we can refer to broker, worker, and cloud nodes as computing service devices. In the ideal scenario, it is not desirable for broker and cloud devices to perform computing services, as they are better suited for their primary tasks. Therefore, the first attempt is to assign a worker node to users. Otherwise, broker and cloud nodes can provide computing services.

The computing module consists of two sub-modules: the data preprocessing module and the deep learning module. Users send their data to the service-providing device, which first preprocesses the data, and then the deep learning module performs predictive tasks based on the model in the system for the specific application. These two modules are designed using the Adapter design pattern [269], allowing new applications to be easily added to the system without the need for changes to the core infrastructure of the system.

3.2.2.1 Data preprocessing module

T-IoT devices need to be as lightweight and compact as possible since they are used on textile products. Therefore, the use of powerful batteries in these devices is not feasible due to weight constraints. Similarly, powerful microcontrollers cannot be used in these devices to extend battery life. For these reasons, T-IoT devices focus on collecting raw sensor data rather than performing extensive computational tasks. All computation tasks, including data preprocessing, are carried out on worker devices. This module is responsible for tasks such as filtering the raw sensor data, extracting nominal, ordinal, quantitative, and aggregated features using windowing techniques.

Electronic textile sensors typically produce time-series data. Directly applying time-series data to machine learning models is not always appropriate due to the presence of mechanical and electrical noise in the sensor data. Therefore, sensor data is initially filtered and subjected to normalization to mitigate noise. Subsequently, the time-series data is stored in a buffer, and time-series features are extracted using the windowing method. The deep learning module utilizes these features to make application-specific predictions.

3.2.2.2 Deep learning module

The deep learning module is responsible for making sense of the extracted features from the data preprocessing module and providing users with application-specific outputs regarding detection, recognition, classification, regression, clustering, and prediction tasks. The deep learning module operates as a replica of the brain, consisting of many neurons and their interconnections. Each neuron's value involves mathematical computations that increase in complexity depending on the number of neurons in the previous layer. Calculating the weights of the neural network and output values of all neurons requires significant computational power. The system's highest computational power is utilized by this module. Depending on the available hardware resources, these calculations can also be performed by graphics processing units or tensor processing units, enabling faster response times.

3.2.3 Proxy module

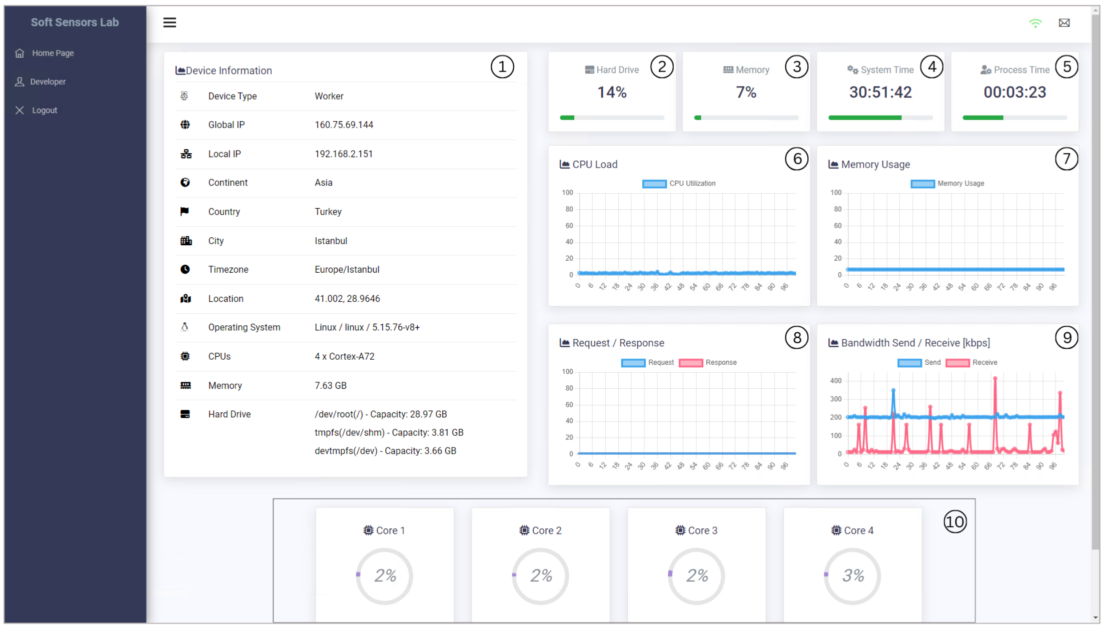
The proxy module serves as a bridge between gateway devices and worker devices. Allowing all devices to be accessible from the Internet makes them vulnerable to external attacks. As a result, there is a risk that an attack on one fog node's devices could put the entire node at risk. If all fog nodes become unavailable, it can impact many users. Therefore, in the FogETex framework, direct access to worker devices via the WAN is restricted. Only LAN devices can request services from them. This restriction helps enhance security by reducing the attack surface exposed to the Internet.

When a device from the WAN wants to request services from a worker device, it will send its requests to the broker device, as if it were requesting them directly from the worker device. The broker device then forwards these requests to the worker device and relays the responses back to the gateway device. In this way, the broker device acts as a man-in-the-middle between the worker node and the gateway device.

In the event of a potential attack, the broker device may sustain damage, but the other worker devices will continue to provide services to LAN devices. This is expected to reduce the availability problems in worker devices, enhancing the resilience of the overall system to attacks and ensuring continued service availability to LAN devices.

3.2.4 User interface

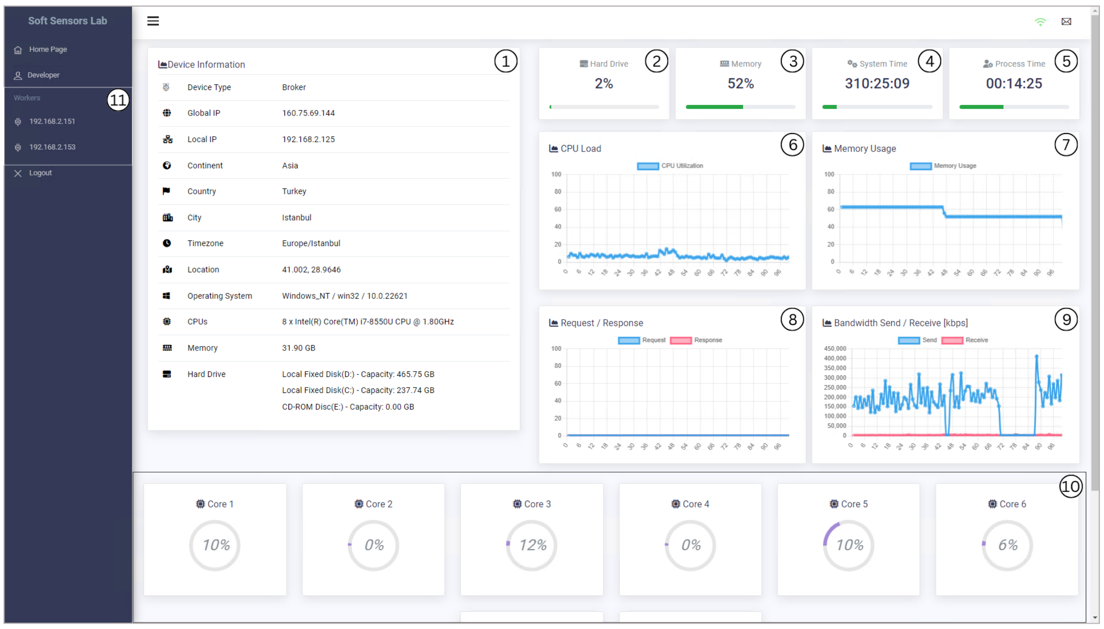
The user interface module contains system-level information such as real-time resource consumption, hardware details, and device-specific information for worker, broker, and cloud devices as shown in Figure 3.3-3.5. This information is primarily used by system administrators for device monitoring, network structure monitoring, and troubleshooting issues like overloading.



Device Information

Device Type	Worker
Global IP	160.75.69.144
Local IP	192.168.2.151
Continent	Asia
Country	Turkey
City	Istanbul
Timezone	Europe/Istanbul
Location	41.002, 28.9646
Operating System	Linux / linux / 5.15.76-v8+
CPUs	4 x Cortex-A72
Memory	7.63 GB
Hard Drive	/dev/root(/) - Capacity: 28.97 GB tmpfs(/dev/shm) - Capacity: 3.81 GB devtmpfs(/dev) - Capacity: 3.66 GB

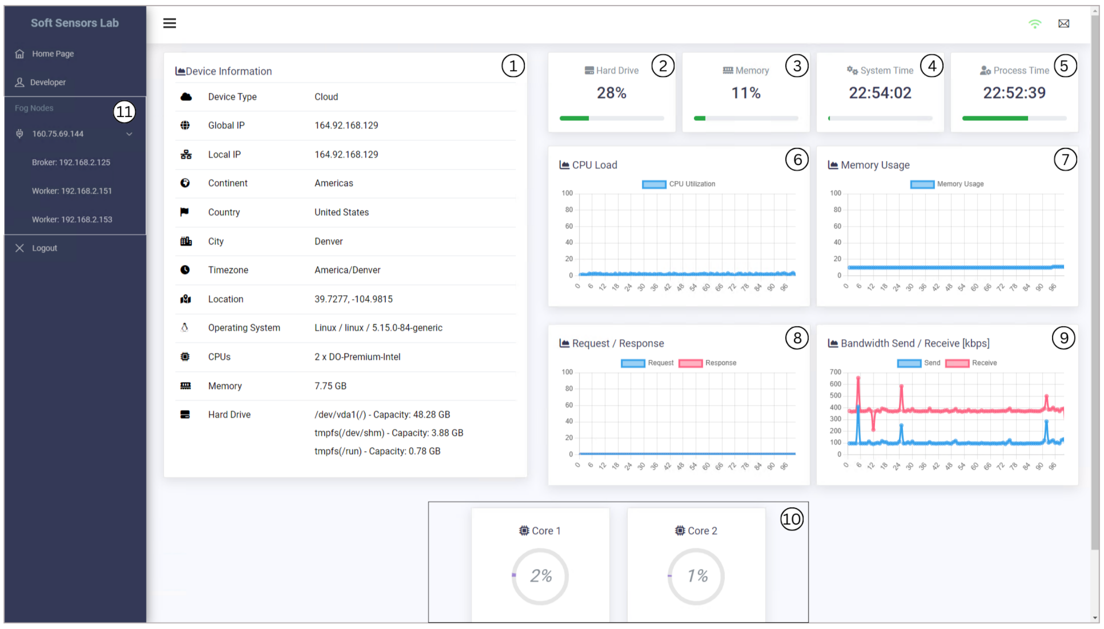
Figure 3.3 : Worker User Interface.



Device Information

Device Type	Broker
Global IP	160.75.69.144
Local IP	192.168.2.125
Continent	Asia
Country	Turkey
City	Istanbul
Timezone	Europe/Istanbul
Location	41.002, 28.9646
Operating System	Windows_NT / win32 / 10.0.22621
CPUs	8 x Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz
Memory	31.90 GB
Hard Drive	Local Fixed Disk(D:) - Capacity: 465.75 GB Local Fixed Disk(C:) - Capacity: 237.74 GB CD-ROM Disc(E:) - Capacity: 0.00 GB

Figure 3.4 : Broker User Interface.



Device Information

Device Type	Cloud
Global IP	164.92.168.129
Local IP	164.92.168.129
Continent	Americas
Country	United States
City	Denver
Timezone	America/Denver
Location	39.7277, -104.9815
Operating System	Linux / linux / 5.15.0-84-generic
CPUs	2 x DO-Premium-Intel
Memory	7.75 GB
Hard Drive	/dev/vda1(/) - Capacity: 48.28 GB tmpfs(/dev/shm) - Capacity: 3.88 GB tmpfs(/run) - Capacity: 0.78 GB

Figure 3.5 : Cloud User Interface.

With the user interface, system administrators can view ① static information provided by the resource manager, such as device type, local and global IP addresses, device location data, CPU cores and specifications, RAM information, and hard disk details. This interface provides a comprehensive view of the system's status and components, allowing administrators to manage and optimize the system effectively. In addition, it is possible to monitor ② disk usage, ③ memory usage, ④ device operating times, ⑤ the runtime of the FogETex system, ⑥ CPU load over time, ⑦ memory usage over time, ⑧ the number of requests and responses sent to the device over time, ⑨ network bandwidth usage over time, and ⑩ CPU core loads.

Furthermore, the user interface can also display ⑪ connected devices for the broker device and the cloud device. In the user interface of the broker device, connected worker devices can be viewed, and in the user interface of the cloud device, connected broker devices and their associated worker devices can be displayed. This allows for a comprehensive understanding of the entire system topology through the user interface of the cloud device.

Figure 3.6 illustrates the user interface of the gateway device. The user interface on the left allows the selection of Bluetooth devices and establishes a connection with the T-IoT device. Once the connection is established via Bluetooth, the user interface on the right opens. From this screen, a connection with the fog computing system can be established, and data transfer begins automatically. A predefined amount of data received via Bluetooth is sent to the worker device. Through the interface, the frequency of incoming data via Bluetooth, the real-time response time of the worker device, and the count of transmitted data can be monitored.

3.3 Network Structure

The FogETex architecture has a hybrid network structure. It uses the Hyper-Text Transfer Protocol (HTTP) request protocol for one-time requests and the Web socket protocol for sequential requests. Attempting to use the same communication protocol for different needs in the system results in unnecessary delays and the excessive use of

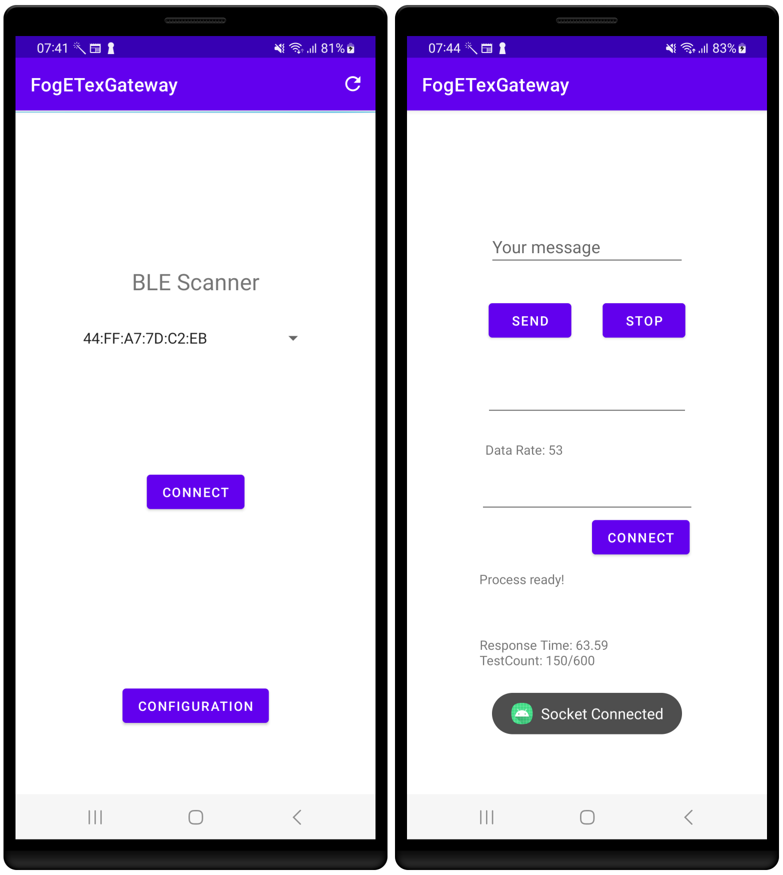


Figure 3.6 : Gateway Device Graphical User Interface.

memory, CPU, and network resources. Therefore, in this framework, the most suitable protocol is used for each task to save time and resources.

3.3.1 RESTful API communication

The HTTP protocol is used for fog node and worker node assignment queries and Web page requests of the user interface. During these queries, the user sends a single request to the device, and then the communication is terminated. RESTful APIs are used for node assignments, and a Web application is utilized within the user interface, all of which operate as services on the same HTTP server.

Figure 3.7 illustrates the connection diagram for users with fog nodes within the LAN. In this scenario, it is assumed that the user joins the network for the first time. Since the user is not aware of any devices in the system, it starts by sending an HTTP request query to the RESTful API of the cloud device, requesting the assignment of a suitable fog node. The cloud device also uses Algorithm 1 to return the local IP address of the broker device within the LAN. This time, the user requests the assignment of a suitable worker device from the broker's RESTful API. The broker uses Algorithm 2 to assign the worker device with the least load to the user. When determining the worker with the lowest load in the fog node, the first condition is that the worker device must not be busy. The busy condition is defined as having memory or CPU usage above pre-determined threshold values. Among the non-busy workers, the device with the least CPU usage is assigned to the user. The reason for selecting CPU as a parameter is that the incoming job requests demand intensive CPU resources.

If the user has previously connected to a fog node within their network, they will request the worker assignment directly from the broker instead of going to the cloud.

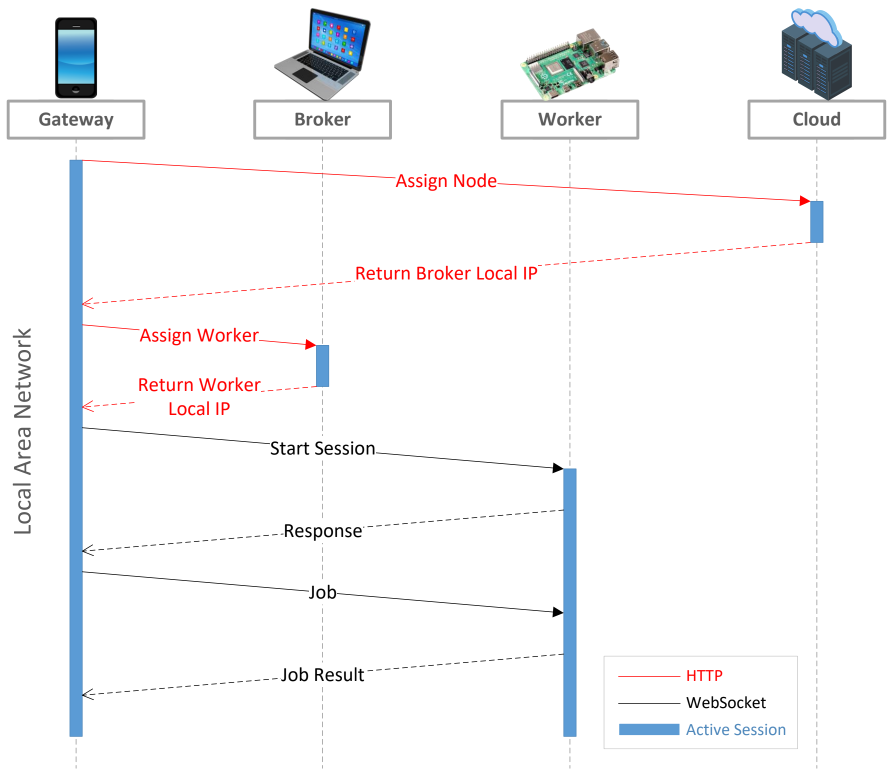


Figure 3.7 : Connection Diagram of Users with A Fog Node in their LAN.

If there is no fog node in the LAN, the appropriate worker device assignment is made according to Figure 3.8, and the user performs calculation requests using the network protocols in that network. Similarly, a query is made to the cloud, but since there is no suitable fog node in the LAN, Algorithm 1 returns the global IP address of the nearest and available fog node to the user based on location. Subsequently, the user accesses the broker device using this global IP and once again requests the assignment of a worker device through the RESTful API. If a user has received services from a fog node over the WAN, and wishes to establish a connection again, they must revert to the cloud to initiate the inquiry process from the beginning. This ensures that they do not miss any newly opened fog nodes that might be closer or within their LAN.

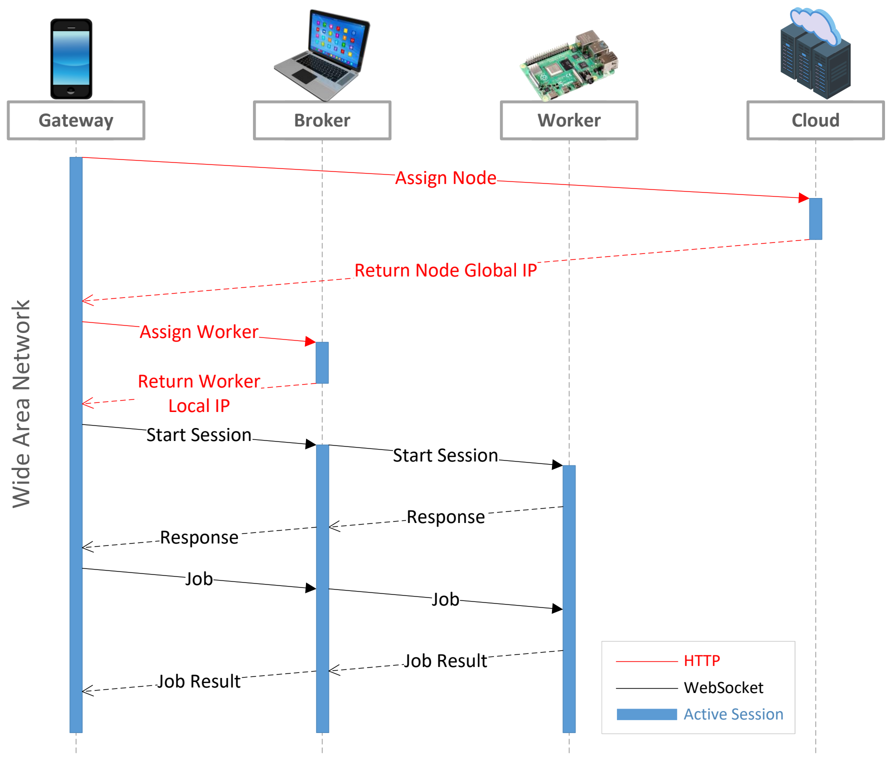


Figure 3.8 : Connection Diagram of Users Without a Fog Node in their LAN.

3.3.2 Socket communication

After the assignment of a worker device, users transmit sensor data sequentially and in raw format to the worker device, expecting data in the same format in return. If this process were attempted using RESTful API, the user would constantly send data to the other end and establish a connection for each job request. If delays occurred in job requests, there would be a high number of active connections at the same time, causing increased resource utilization at the service point.

In the Web socket architecture, asynchronous bidirectional data transfer can occur between the user and the service-providing device over a single connection. Incoming job requests accumulate in a queue along with the assigned user ID, and when the computation process for the respective user is completed, the job request is sent back to the user via socket.

In Figure 3.7, a user who wants to establish a connection over the LAN can directly communicate with the worker node using the local IP address of the worker node. After the socket connection is established, the user sends a "start session" message to indicate the desire to receive services. In the background, the worker device creates an object for itself, including specialized data buffers, machine learning models, calibration parameters, and other variables. Once all these processes are completed, the user is informed that the application is ready, and it can now receive services. Afterwards, the gateway device transmits the sensor data it receives via Bluetooth to the worker node and uses the incoming responses for various tasks such as control of the system, visualization, alert systems, and analysis within the application. The session on the worker device remains open as long as the user is actively connected. When the connection is lost, the session is closed, and all data specifically held for the respective user is cleared.

If there is no suitable fog node in the user's LAN, the user is assigned the nearest and most suitable fog node via the WAN (Figure 3.8). Then, the broker in the fog node shares the local IP address of a suitable worker device with the user. Since the user is on the WAN, a direct connection cannot be established with the worker via the local IP. First, the user informs the socket server of the broker device using the local IP address and establishes a connection. The socket server passes this information to the proxy module in the background, creating a virtual user. This virtual user then connects to the worker using its own socket client. Job requests are first sent to the broker, and the broker forwards them to the worker. The worker then communicates its responses to the virtual user, which is connected to it. Afterward, the proxy module relays these messages to the user through its socket server. This process ensures a more secure way for devices on the WAN to receive services. The sessions active in the proxy module on the broker device and the computing module on the worker device remain open as long as the user is actively receiving services. When the connection between the user and the proxy module is lost, the proxy module severs the connection to the worker, subsequently closing sessions on both the broker and worker devices.

3.3.3 Scalability handling

FogETex architecture follows a hierarchical structure where fog nodes establish a top-down hierarchy for devices to find suitable fog nodes, although they have their own administrative autonomy. However, the availability information of devices is still generated by worker and broker devices. The cloud device does not assign tasks or devices to nodes that do not indicate their availability. To set up a new fog node in the system, the broker node needs to connect to the cloud. After the broker node is connected, worker devices on the same LAN can join the system by learning the IP address of the broker device via the RESTful API of the cloud device. Worker devices are connected to the broker, and the broker, in turn, is connected to the cloud, sharing resource information from bottom to top in this hierarchy.

Adding a new broker or worker node to the system is straightforward. The devices only require the static IP or domain information of the cloud node and the device type to be entered. All other organizational aspects are managed by the FogETex framework. There is no limitation on the number of fog nodes and workers in the system. However, if multiple broker devices connect from the same network, the first one to connect will be assigned as the primary broker for the workers, while the others will remain passive. Therefore, additional worker devices can be easily added to the system as needed and made available to users.

3.4 Concurrency Control Techniques

Due to the wide range of machine learning libraries and the relatively simpler development process, the computing module of the FogETex framework has been developed using Python to enable other developers to easily write applications for the framework. The biggest obstacle to effectively utilizing processor cores is the Global Interpreter Lock (GIL) in the Python interpreter. The GIL mechanism only allows one thread to execute Python code at a time, thus preventing concurrency issues that may arise in memory and data management. However, it also introduces performance issues in parallel operations and CPU-intensive tasks [270]. Therefore, performance improvements using different concurrency techniques are required in this context.

The FogETex framework is designed to serve multiple users efficiently. Effective utilization of processor cores is crucial for accommodating a larger number of users. The FogETex framework incorporates single-threaded, multi-threaded, and multi-process concurrency methods. All three methods utilize the WebSocket Inter-Process Communication (IPC) mechanism.

Since the socket server and the computing module operate in separate processes on computing service devices, inter-process communication is critical for system performance. In addition to the WebSocket IPC method, the FogETex framework supports First In First Out IPC (FIFO IPC) and RESTful API IPC methods through the multi-process concurrency approach. In total, five different concurrency and IPC methods can be configured within the FogETex system.

3.4.1 Single-threaded data processing via WebSocket IPC

Single-threaded data processing via the WebSocket IPC method is the initially proposed approach. Figure 3.9 illustrates the data communication flow of this method. It has a simpler structure compared to other methods. In all methods, users send computation requests to the computing service devices through a WebSocket connection.

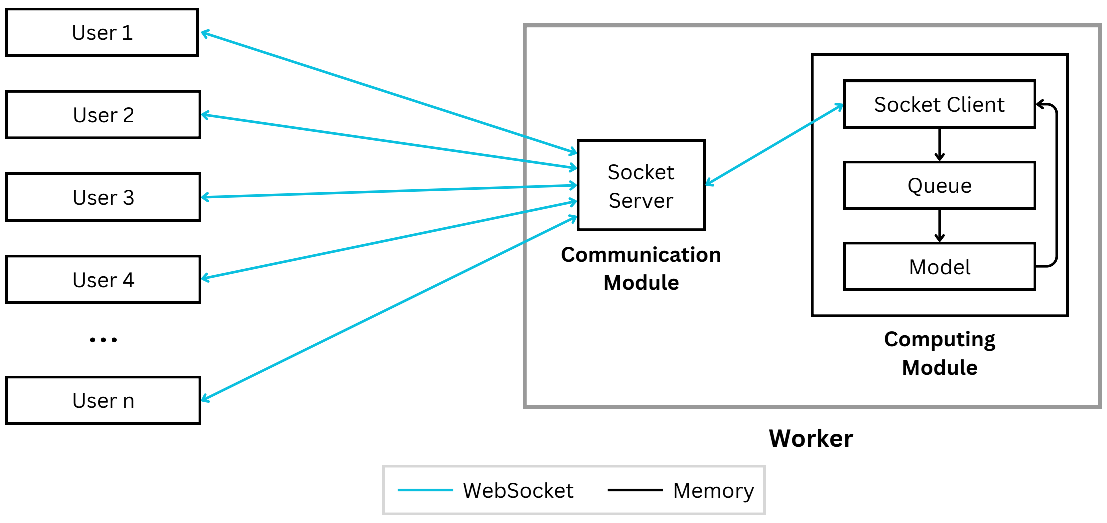


Figure 3.9 : Single-threaded Data Processing via WebSocket IPC.

Data received by the Socket Server is forwarded to the computing module via WebSocket. The socket client module within the computing module adds the data to a queue. Once the computation requests that arrived earlier are completed, the data is processed using the developed model and returned to the user through the socket client and socket server. Since all operations within the computing module occur in the same process, data is transferred directly through memory.

3.4.2 Multi-threaded data processing via WebSocket IPC

The second method is multi-threaded data processing via the WebSocket IPC method (Figure 3.10). In the previous method, each incoming computation request is added to the queue, and the new request must wait for the completion of the earlier ones. In the multi-threaded method, computation requests in separate threads do not have to wait for each other. This aims to reduce queue times.

In this method, each thread has its own dedicated socket client, queue, and model. This allows computation requests in different threads to be executed in parallel. Particularly in busy-wait operations, when the processor is idle, the computing service device can serve other users through another thread.

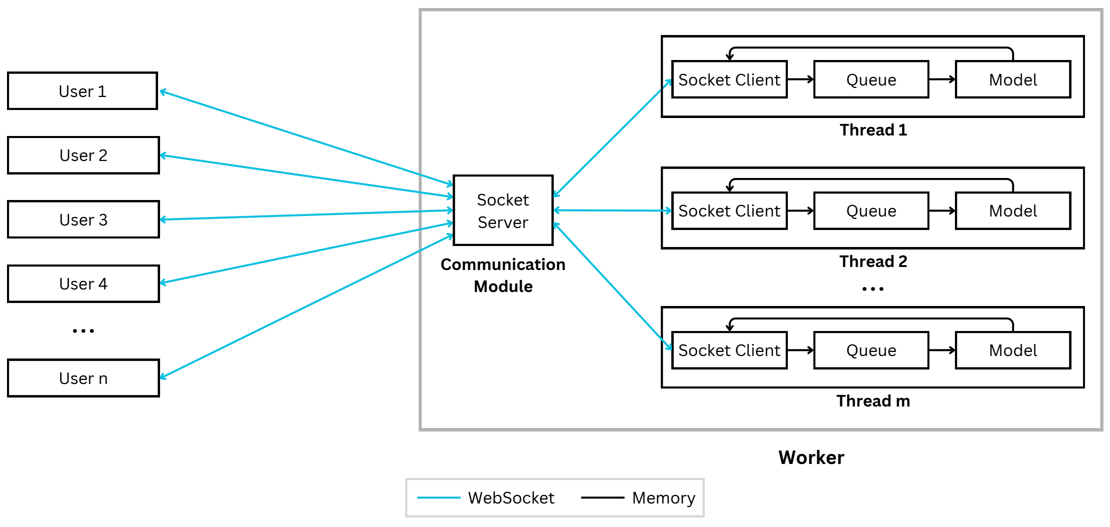


Figure 3.10 : Multi-threaded Data Processing via WebSocket IPC

3.4.3 Multi-process data processing via WebSocket IPC

The third method is multi-threaded data processing via WebSocket IPC (Figure 3.11). Due to the Global Interpreter Lock (GIL) issue in Python, multi-threaded structures can behave like single-threaded ones. Therefore, this method was developed to achieve higher performance. Similar to the multi-threaded method, data comes from the socket server via WebSocket. The key difference is that in this method, threads are replaced by processes. This approach aims to take better advantage of the processor's computational capacity by allowing processes to run on different processor cores.

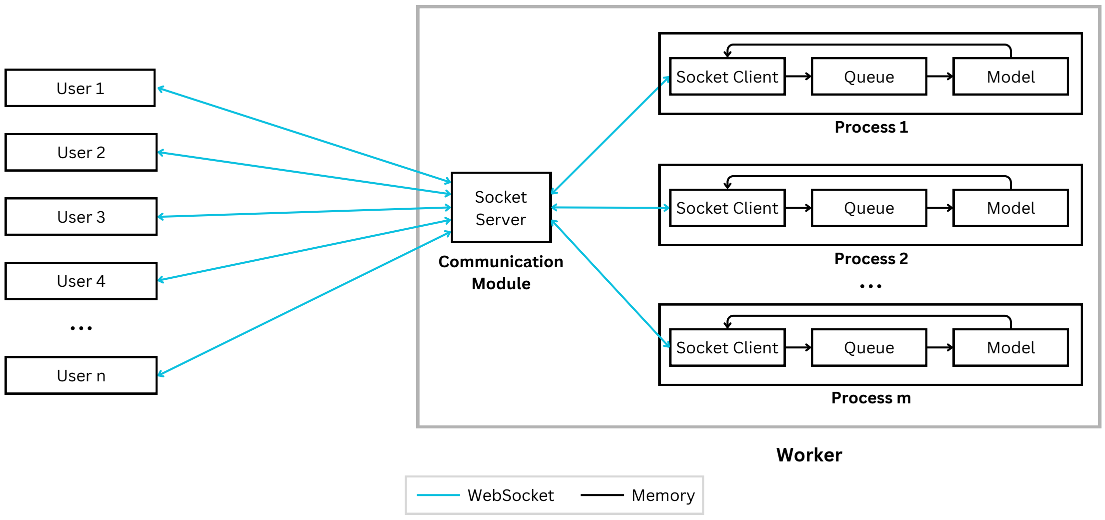


Figure 3.11 : Multi-process Data Processing via WebSocket IPC.

3.4.4 Multi-process data processing via RESTful API

The fourth method is multi-process data processing via RESTful API (Figure 3.12). This method, like the third method, uses a multi-process concurrency approach but employs a RESTful API for IPC. Each process contains its own RESTful API server. When a computation request arrives, the socket server sends the data to the RESTful API using the HTTP method. Each incoming data request immediately begins the execution process. The HTTP connection remains open throughout the computation, and the response is awaited by the socket server. With this method, queue delays have been reduced to zero.

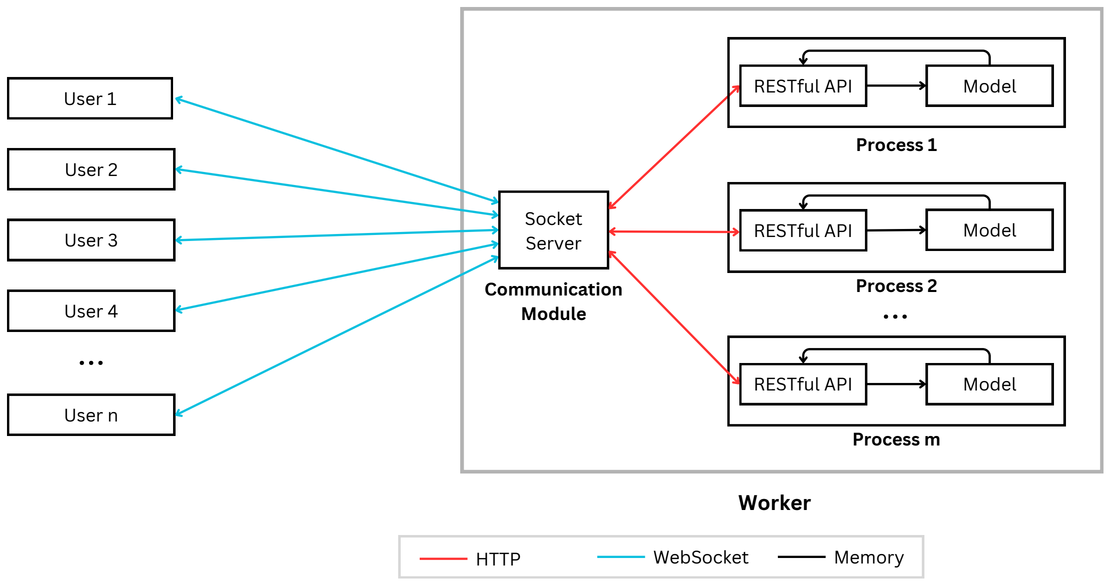


Figure 3.12 : Multi-process Data Processing via RESTful API.

3.4.5 Multi-process data processing via FIFO IPC

The fifth method is multi-process data processing via FIFO IPC (Figure 3.13). In this method, data between the socket server and the computing module is transferred using a special IPC method called FIFO. FIFO IPC, also known as the pipe method, involves two processes, one acting as the sender and the other as the receiver. Therefore, two FIFOs are required for full-duplex communication between the processes. The input

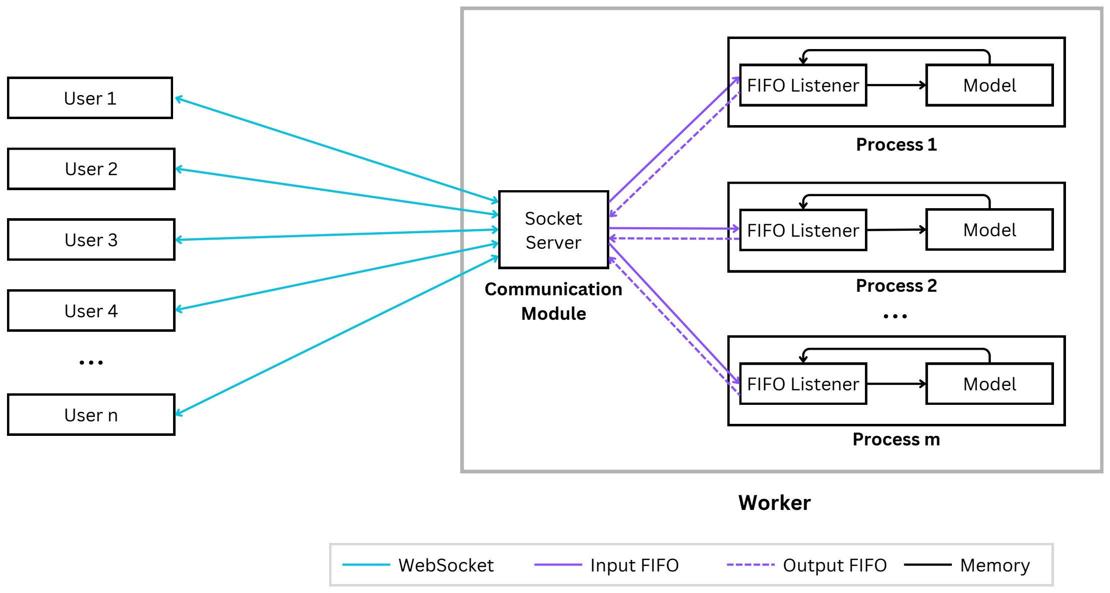


Figure 3.13 : Multi-process Data Processing via FIFO IPC.

FIFO facilitates communication from the socket server to the FIFO listener, while the output FIFO handles communication in the reverse direction.

Each time new data arrives, the FIFO listener sends the computation request to the model and does not read new data until the computation is complete. In this way, the FIFO structure also serves as a hidden queue.

Data is transmitted to processes through memory, making this method expected to be more efficient compared to other approaches. Additionally, running operations in different processes will allow for better utilization of multi-core processors.

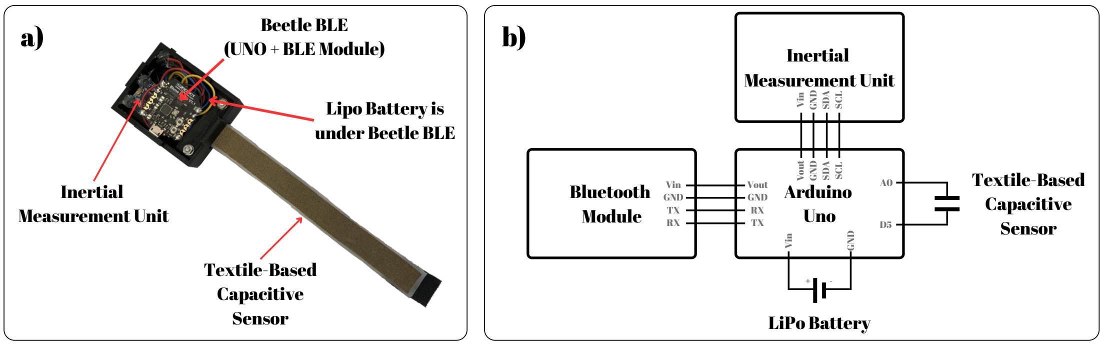
4. GAIT PHASE RECOGNITION SYSTEM USING FogETex

Gait analysis is utilized in various applications such as rehabilitation, robotics, gait biometrics, biomechanics, sports analysis, and disease monitoring [271]. In our previous study, we developed a deep learning-based gait phase recognition system using textile-based capacitive strain sensor [22]. It is important to emphasize that the aim of our current study is not to design and implement a gait recognition system from scratch and evaluate its performance using the given T-IoT system, which was indeed already investigated in [22]. Instead, we will utilize the data created by the participants during the interaction with this system in that study for the sake of servicing T-IoT data to be fed into our proposed FogETex framework. Therefore, the next subsections elaborate on the workload created by a T-IoT application with respect to the data processing and deep learning methods.

4.1 T-IoT Device Design

The T-IoT device includes a textile-based capacitive sensor [44], a DFRobot Beetle BLE development board (Transmitter), an MPU6050 6-axis Inertial Measurement Unit (IMU), and a Lithium Polymer (LiPo) battery as shown in Figure 4.1(a). The T-IoT device is integrated into a knee brace with its capacitive sensor placed [47] on the knee joint. The textile-based capacitive sensor is formed by adding a silicon layer, which serves as the dielectric, between two conductive fabrics acting as electrodes, and it functions as a parallel plate capacitor. The stretching caused by joint movement leads to an increase in capacitance. Similarly, a decrease in the stretch results in a reduction in capacitance.

The electronic circuit design of the T-IoT device is shown in Figure 4.1(b). The Beetle BLE (Transmitter) consists of an Arduino Uno and a Bluetooth module inside. The Arduino Uno features an Atmel ATmega 328 microcontroller. Embedded software, written in C++, is used to collect data at a sampling frequency of 50 Hz. The microcontroller measures the capacitance of the textile-based strain sensor using a voltage divider method and simultaneously receives real-time data from the IMU sensor via I2C protocol, sending this information to the gateway device via Bluetooth module (Texas Instrument, CC2540). The T-IoT device is powered by a single LiPo battery connected to the Beetle BLE.



a)
Beetle BLE (UNO + BLE Module)
Lipo Battery is under Beetle BLE
Inertial Measurement Unit
Textile-Based Capacitive Sensor
b)
Inertial Measurement Unit
Vin GND SDA SCL
Bluetooth Module
Vin GND TX RX
Vout GND RX TX
Arduino Uno
A0
D5
Vin
GND
Textile-Based Capacitive Sensor
LiPo Battery

Figure 4.1 : T-IoT Device for Gait Phase Recognition System: a) Actual Photo and b) Schematic Illustration.

4.1.1 Measurement based on textile-based capacitive strain sensor

The capacitive sensor placed on top of the knee brace changes its length depending on the bending of the knee. The capacitance of the sensor increases proportionally with the length. This allows for the detection of knee movement based on the sensor's capacitance value.

Textile-based capacitors cannot typically utilize capacitive measurement methods such as charge/discharge due to their capacitance values falling between 30 and 100 pF. Therefore, an alternative measurement method known as voltage divider is employed. In this method, one end of the capacitive sensor is connected to a digital pin of the microcontroller, while the other end is connected to an analog pin. Charging occurs via the digital pin. The analog pin generates a noise capacitance within the microcontroller known as stray capacitance. The charged capacitance forms a voltage divider with the stray capacitance. Voltage measurement is carried out via the analog pin, and the capacitance value of the sensor (C_x) is calculated using the following equation.

$$C_x = \frac{V_{in} \cdot C_{ref}}{V_{out} - V_{in}} \tag{4.1}$$

where V_{in} represents the input voltage measured by the Analog-to-Digital Converter (ADC), C_{ref} denotes the stray capacitance (33.1 pF), and V_{out} is the supply voltage that charges the capacitance. Given the known values of V_{out} and C_{ref}, the capacitance value of the sensor can be easily determined.

4.1.2 Measurement based on IMU sensor

The gyroscope integrated inside the IMU sensor is responsible for measuring angular velocity in three different directions. These directions are referred to as the x (pitch), y (roll), and z (yaw) axes. The Inertial Measurement Unit (IMU) was positioned on the knee in such a way that the gyroscope's x-axis (gyro-x) was aligned with the direction of gravity, the y-axis (gyro-y) was perfectly parallel to the ground, and the z-axis (gyro-z) was perpendicular to the surface of the leg as shown in Figure 4.2. This was done in order to get gait phase information that was representative of the individual's gait. Because of this, gyro-z measurements have a negative slope when they are being performed in flexion motion, but they have a positive slope when they are being performed in extension motion.

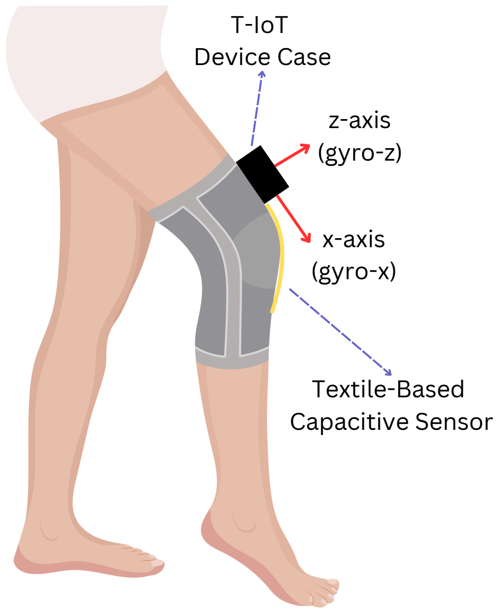


Figure 4.2 : T-IoT Device Placement on the Knee and IMU Axes (x and z).

4.2 Data Labeling

Using a Butterworth filter of the third order, the z component of the gyroscope data that was obtained is filtered in order to remove noise that is caused by gravity, vibration in the leg region, and other external influences at the same time. Phase detection utilizing gyro-z data has been accomplished through the utilization of the local extrema detection approach, which refers to a location in an open interval at which the greatest or minimum value of the function is attained [272]–[275]. Calculations were made to determine the occurrences of local minima and maxima in order to identify phase-shifting locations. One example is shown in Figure 4.3, which is a plot of the gyro-z data for two consecutive steps. The local minima points correspond to

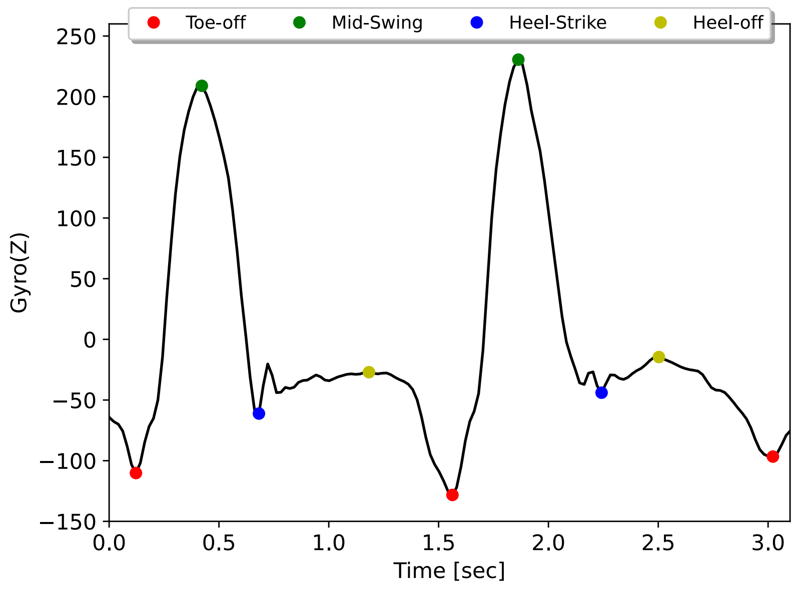


Figure 4.3 : Gyro-z data.

the toe-off and heel-strike phases. During these phases, the leg engages in flexion movement, which results in a decrease in angular velocity in the regions along the z-axis. During the mid-swing and heel-off phases, the leg swings forward and provides extension action, which ultimately results in a positive angular velocity along the z-axis. The information that is acquired from these measurements is then utilized as ground truth values for the purpose of training the model on capacitive sensor data.

4.3 Dataset

Data was collected with a sampling frequency of 50 Hz from a total of 5 subjects aged between 21 and 35 on tracks prepared for step sizes of 20, 30, 40, 50, and 60 cm (Table 4.1). Participants were instructed to walk on the tracks by stepping on markers placed on the ground. Each participant was asked to perform 5 consecutive steps on each track, and each test was repeated 10 times. Thus, there are a total of 50 test data for each test subject. The data for each step size is divided into 80% for training and 20% for testing.

Table 4.1 : Demographic information of test subjects.

Gender	Age	Height	Weight
Male	21	170 cm	65 kg
Female	35	160 cm	62 kg
Male	32	165 cm	72 kg
Female	22	172 cm	63 kg
Male	21	183 cm	75 kg

4.4 Data Preprocessing

In the system, incoming capacitance data is processed through a real-time Butterworth filter to reduce noise. The filtered data is maintained in a 10-sample window. During the training phase, with a slide count of 5, the first, second, and third derivatives of the capacitance values are computed to extract velocity, acceleration, and jerk features. These features, along with the filtered capacitance value, undergo standard normalization using the following formula for each feature individually:

$$Z = \frac{x - \mu}{\sigma} \tag{4.2}$$

where Z is the normalized feature value, x is the feature value, μ is the mean of the feature, σ is the standard deviation of the feature. To perform real-time normalization, the values of μ (mean) and σ (standard deviation) for each feature must be known. In real-time systems, these values are typically obtained through a calibration process to enable the normalization process to be conducted in real-time.

During both the training and testing phases, the first test file from the trials with different step lengths for each test subject is utilized as calibration data to calculate the normalization parameters. The features of the test subject data from the same course are then normalized using these parameters. Subsequently, the machine learning model is trained, and after training, the model is tested.

In the proposed real-time system, the initial 6 seconds of data are used for calibration. During this period, the model does not make predictions and only stores the extracted features in a list. At the end of this interval, μ and σ values are computed for each feature. Subsequently, when new data arrives, the normalized values of the features are sent to the model, and the model predicts the gait phase in real time.

4.5 Deep Learning Model

A machine learning model based on Long Short-Term Memory (LSTM) was employed to predict the gait phase from capacitive sensor data of the T-IoT device. LSTM-based models offer the advantage of capturing temporal dependencies in time series data [276], which is particularly relevant as capacitive sensors, like other e-textile sensors, generate time series data. The model consists of two LSTM modules, each with four hidden layers, and a fully connected linear neural network module with 64 neurons. The entire model was trained using the Adam optimizer with a learning rate of 0.001. The performance of the model was evaluated based on the multi-class cross-entropy loss function, which was defined as follows:

$$Loss = \sum_{c=1}^{M} y_c \cdot log(\hat{y}_c), \tag{4.3}$$

where c is the class number, M is number of classes, y_c is true probability of the class, and $\hat{y}_c$ is the predicted probability of the class.

4.6 Experiments

The effectiveness of the FogETex framework was demonstrated by deploying it to devices that constitute the fog architecture. The gate phase recognition system was used in performance tests as a dedicated application. Tests were conducted using a mock client to allow for analyzing the system in ideal conditions by eliminating operating system delays and Bluetooth jitter caused by the communication with the T-IoT device. Additionally, the same tests were repeated on a physical mobile device as the actual client to represent real-world conditions which were disregarded in the mock client.

4.6.1 Experimental setup

The devices and their respective configurations used in the tests of the FogETex framework are provided below:

① **Broker node:** Dell Inspiron 15 5000 (Intel Core i7-8550U CPU @1.80GHz, 32GB DDR4 RAM, 240GB SSD, and Windows 11 Enterprise 64-bit). Software: Node.js v18.15.0 and Python 3.9.13.
② **Worker node:** Raspberry Pi 4 Model B (Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC CPU @1.8GHz, 8GB LPDDR4-3200 SDRAM, 32GB Micro SDHC Class 10).
③ **Network switch:** ZYXEL GS-1100-16 GIGABIT 16 Port Switch.
④ **Wi-Fi Access Point (AP):** Huwai HG8245X6 Fiber Optic Modem (ISP: Turkcell Superonline, Download speed: 500Mbps, Upload speed: 20Mbps).
⑤ **Cloud device:** Digital Ocean VPS (2 x Dedicated Premium Intel CPU, 8GB RAM, 50GB SSD, Cloud Location: Frankfurt/Germany).
⑥ **Mock client:** Lenovo ThinkPad X1Carbon (Intel Core i7-5600U CPU @2.60GHz, 8GB DDR3 RAM, 240GB SSD, and Windows 10 Pro 64-bit). Software: Node.js v18.15.0 and Python 3.7.15.
⑦ **Gateway device (Actual client):** Samsung A14 mobile phone (4GB RAM 128GB Memory with Android 13).
⑧ **T-IoT device:** DFRobot Beetle BLE v1.1 with Bluetooth 4.0 and textile-based capacitive sensor.
⑨ **LTE modem:** iPhone 8 Plus (3GB RAM, 64GB Memory with iOS 16).

Figure 4.4 depicts the connection diagram of the devices used for the experimental setup to represent a typical usage scenario utilizing FogETex framework. The experimental setup consists of two test beds: a Wi-Fi (IEEE 802.11ac) testbed and a cellular (LTE) testbed.

4.6.1.1 Wi-Fi testbed

In the Wi-Fi (IEEE 802.11ac) testbed, clients and the fog node are in the same LAN. Broker node ① and worker node ② are connected via a network switch ③ linked

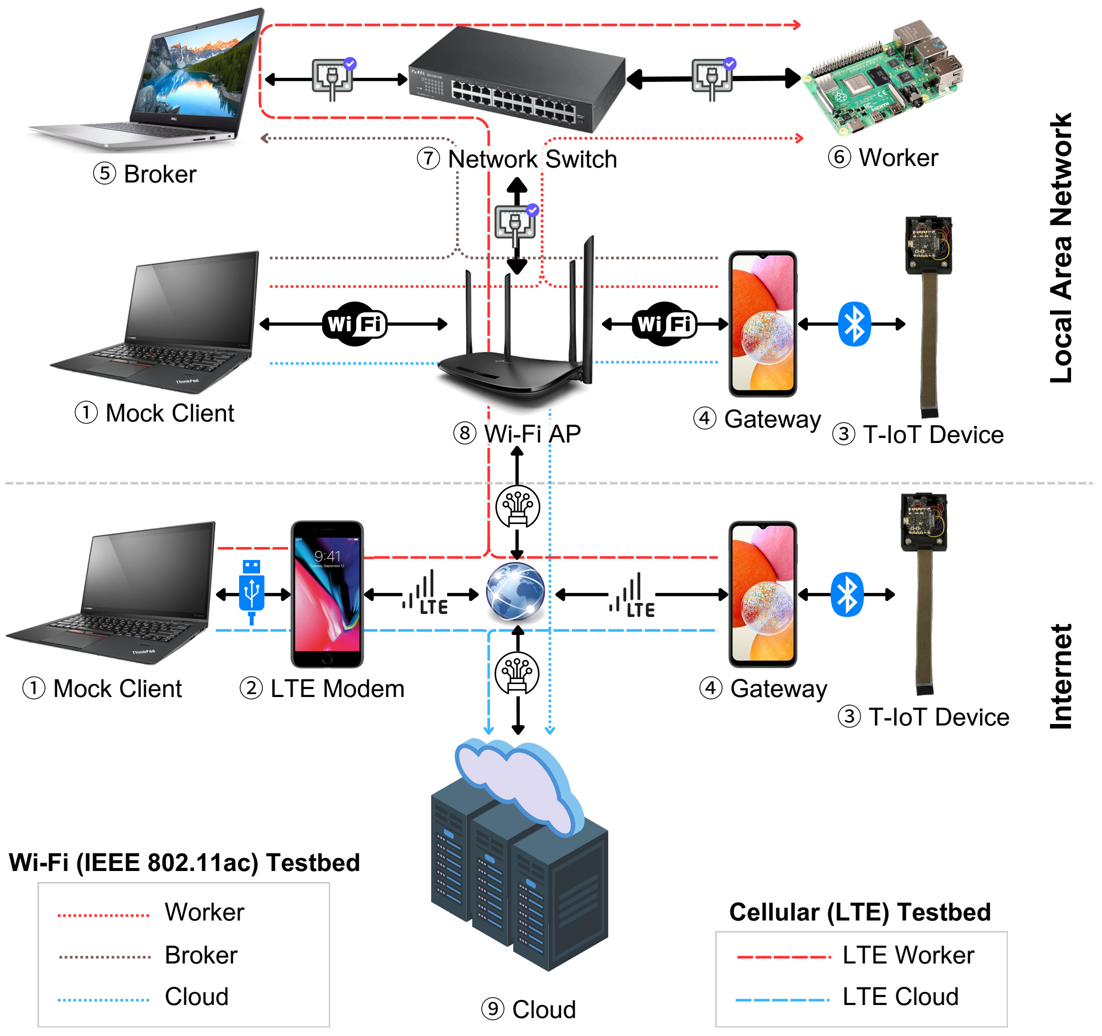


Figure 4.4 : Experimental Setup Device Connection Diagram.

to the Wi-Fi AP ④. The connections between the worker, broker, network switch, and Wi-Fi AP are established via Ethernet connections. The fog node is linked to the Internet through fiber Internet via the Wi-Fi AP. The devices in the LAN access the cloud node ⑤ via the Fiber Internet.

The worker node is responsible for computational serving to clients, but the broker node and cloud node can also give this service. Therefore, the worker node, the broker node, and the cloud node are defined as computing service devices. Since there is one cloud node and one broker node in the system, performance comparisons were conducted using a single worker device to ensure testing in the same environment. This article primarily focuses on the development of a system catering to the needs of electronic textiles, rather than dynamic resource allocation and computing offloading in multi-worker systems.

The sole responsibility of the mock client ⑥ is to send previously recorded T-IoT data to the computing service devices every 20 ms so that we can measure the performance of these devices. The mock client connects to the Wi-Fi AP using the wireless networking standard IEEE 802.11ac in this testbed.

The gateway device ⑦ conducts real-time device testing. The T-IoT device ⑧ measures the capacitance of the strain sensor and transmits this data to the gateway device via Bluetooth. The gateway device transfers the data received from the T-IoT device to the respective computing service device via Wi-Fi in the Wi-Fi testbed. As a result, the performance of the framework is monitored under the control of a real-time device.

4.6.1.2 Cellular testbed

In the cellular (LTE) testbed, clients transfer the data to the worker node ② and the cloud node ⑤ through LTE. The LTE testbed represents receiving services from a fog node over the WAN. The broker node ① in the fog node acts as an intermediary for assigning workers within the fog and serves as a proxy for cellular tests. Hence, the broker node is not utilized as a computing service device in the LTE testbed.

The mock client ⑥ sends its data to the worker node and the cloud node through an LTE modem ⑨ connected via USB. The gateway device ⑦ is connected to the fog node and cloud node via LTE. The attributes such as the source, quantity, and quality of the sensor data are the same as in the Wi-Fi testbed. Differently, the mock client and gateway device send their data to computing service devices over the Internet.

4.6.2 Experimental scenarios

The performance of the FogETex framework was evaluated through system tests using a mock client and an actual client. The tests were conducted using recorded data from the mock client. Test data is initially stored in memory and then sent to computing service devices at 20 ms intervals based on the test configuration. Both outgoing and incoming data are tracked with timestamp information. On the actual client side, real-time data is received from the T-IoT device with a sampling frequency of 50 Hz, and this data is instantly forwarded to the computing service devices.

The resource consumption data of the computing service devices (CPU load, memory usage, and bandwidth) are sent in real-time to the mock client program through a Web socket connection using a NodeJS-based program. The code for measuring system resources in the program is the same as the resource manager mentioned in Sec 3.2.1. The threshold values for CPU and memory resources of computing service devices have been set to 70% heuristically. Devices exceeding this threshold report themselves as "busy" due to resource usage beyond this limit.

Seven identical data sets were used in the experiments, each set belonging to data acquisition session and being 5 minutes long. For the arbitration tests, we established and closed connections 25 times consequently. Unless otherwise specified in characteristic graphs, bar graphs represent the mean and standard deviation values of the results obtained in the experiments.

The Wi-Fi testbed has three main experimental scenarios "Connection to worker, broker, and cloud". In these scenarios, both the mock client and the actual client connect separately to those respective devices to receive service. In all these scenarios, the IP information of the respective devices is known by the clients, allowing them to connect directly. The arbitration process includes an additional "Discovery and connection to worker" experimental scenario for client devices connecting to the system for the first time and seeking service from the worker. In this specific scenario as outlined in Figure 3.7, the appropriate fog node and worker node are assigned, followed by the connection to the worker device.

In the LTE testbed, there exist only "Connection to worker and cloud" scenarios. The clients establish connections to both devices via LTE and receive computational services. In this test bed, the broker device acts as a proxy and is responsible for communication between workers and clients. In the arbitration process, the clients in the WAN, despite having received services from worker devices before, request a new fog node assignment with each connection to ensure connectivity to the nearest node. In the “Discovery and connection to worker” scenario, the procedure outlined in Figure 3.8 is followed. In the “Connection to cloud’F scenario, the IP address of the cloud device is known, allowing direct connection to the device.

4.6.3 Evaluation criteria

The metrics used in the evaluation of the FogETex framework are defined as prediction accuracies, time characteristics, device resource usage, and network bandwidth usage, consequently.

4.6.3.1 Prediction accuracies

The accuracy of the gait analysis deep learning model was examined in test scenarios using Cross Entropy Loss function results, F1 Score, and a confusion matrix. In total, the model was trained for 100 epochs. Cross-entropy results were calculated based on Equation 4.3. F1 score was used to prevent misleading results depending on the dataset’s erroneous outcomes, and it was calculated as follows, using precision and recall:

$$Precision = \frac{TP}{TP+FP}, \tag{4.4}$$

$$Recall = \frac{TP}{TP+FN}, \tag{4.5}$$

$$F1\,Score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}. \tag{4.6}$$

The gait analysis process involves a multiclass classification problem, and when calculating, the macro F1 score metric was preferred and calculated using the following equation:

$$Macro\,F1\,Score = \frac{\sum_{c=1}^{M} F1\,Score_m}{M}, \tag{4.7}$$

where M is the number of the classes.

To examine the misclassifications of the gait analysis model on the test data in more detail, the confusion matrix at the end of 100 epochs is presented.

4.6.3.2 Time characteristics

The FogETex framework provides real-time computing services, making time characteristics of utmost importance. Therefore, six time characteristics that are crucial for clients to connect to the system and receive services have been defined as follows:

Arbitration time: It refers to the duration it takes for a client to find a suitable worker node and for the worker to prepare the necessary setup for the client. In these setup processes, the model for the application is loaded, and the necessary buffers and variables are prepared. After this period, the client can send data to the system. It encompasses the preparation time for the computing service device, excluding "Discovery and connection to worker" scenarios.

Latency: It is the time it takes for a client to send data and for that data to enter the computation queue.

Queuing delay: It is the time that a computation request spends waiting in the queue.

Execution time: It is the total duration during which the incoming computation request goes through preprocessing and the deep learning modules in sequence.

Total response time: It is the duration from sending the computation request to its return to the client.

Jitter: It represents the variation in total response time and is calculated by taking its standard deviation.

4.6.3.3 Device resource usage

This metric is used to monitor resource usage on computing service devices. It provides CPU and memory usage data for the respective device at one-second intervals. High resource usage can lead to increased system latency and reduced service capacity for other users.

4.6.3.4 Network bandwidth usage

One of the parameters of fog computing architecture is having low network bandwidth. Therefore, the total network bandwidth usage data for different device configurations is provided as an evaluation criterium. Network bandwidth data represents the total of incoming and outgoing data.

4.6.3.5 Stress test

The system is expected to handle peak loads and dynamic resource demands under different workloads. In this way, the system's ability to serve many users can be analyzed. Each test performed with the mock client was repeated for different numbers of users. Users were created within the mock client using multithreading.

4.7 Results

In this section, the experimental results of the gait phase recognition system and the FogETex framework are examined. First, the prediction accuracies of the gait phase recognition system are reviewed. Since the main focus of this thesis is the fog computing framework, time performance and resource utilization are more critical than accuracy results. In next sections, time performance, resource utilization, network bandwidth usage, stress test, and performance benchmarking results are analyzed. These results are used to perform a performance analysis of the framework using the gait phase recognition system.

4.7.1 Results on prediction accuracies

Figure 4.5 shows the decrease in cross-entropy loss as the number of epochs increases. It was observed that the loss did not change significantly after 80 epochs, and therefore, the total number of epochs was set to 100. After 100 epochs, the loss value of the model was calculated to be 0.6.

Figure 4.6 depicts the increase in F1Score as the number of epochs progresses. In the first 5 epochs, the F1Score exceeds 0.75 and shows a gradually increasing trend.

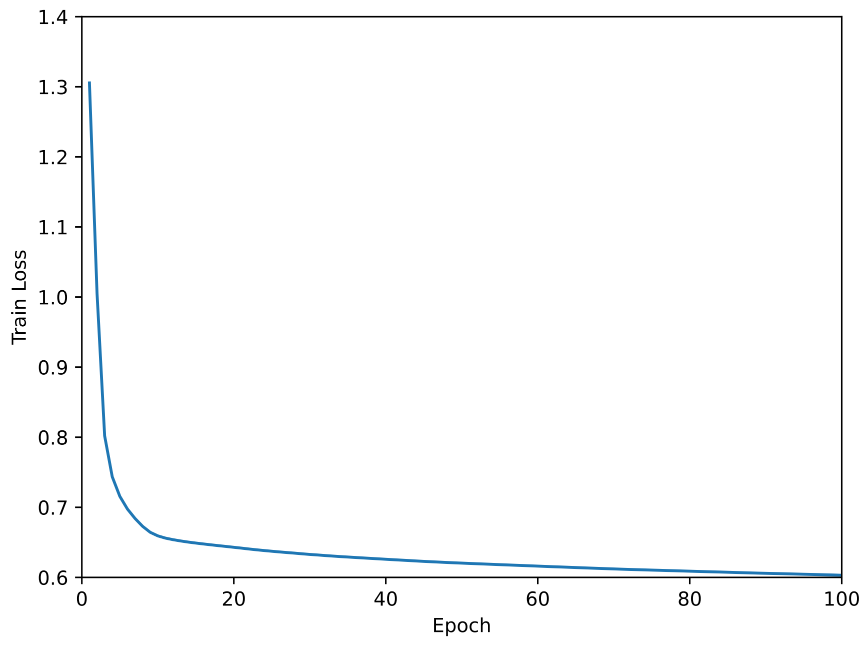


Figure 4.5 : Model train loss.

After 100 epochs, an F1Score of 0.79 was achieved. Considering that the measurement frequency of the sensor is 50 Hz, approximately 39 correct predictions per second can be assumed. Depending on the application of gait analysis, these 39 values can be combined in various ways to further enhance application performance.

Figure 4.7 illustrates the distribution of predictions made based on real classes. The toe-off, mid-swing, and heel-off classes achieved an accuracy score of more than 0.84. However, heel-strike achieved a performance of around 0.64, and it was observed that approximately 24% of the samples labeled as heel-strike were actually in the heel-off class, which is the next class label.

4.7.2 Results on time performance

Figure 4.8 provides the mean arbitration times for different scenarios. In Wi-Fi testbed and using mock client, the allocation time in the "Discovery and connection to worker" scenario is longer compared to "Connection to worker and broker" scenarios. This is because of the additional fog node discovery process. The broker node, being more powerful, allocated the resources in less time than the worker node. In the "Connection to cloud" scenario, the mock client accesses the cloud node via the Fiber Internet network, thus this scenario requires a longer allocation time compared to "Connection

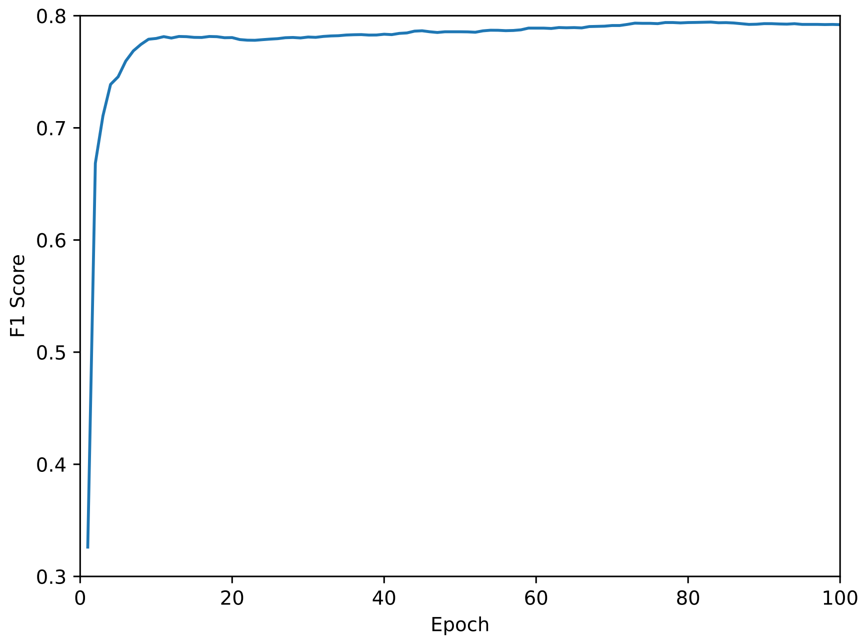


Figure 4.6 : Macro F1 scores of trained model.

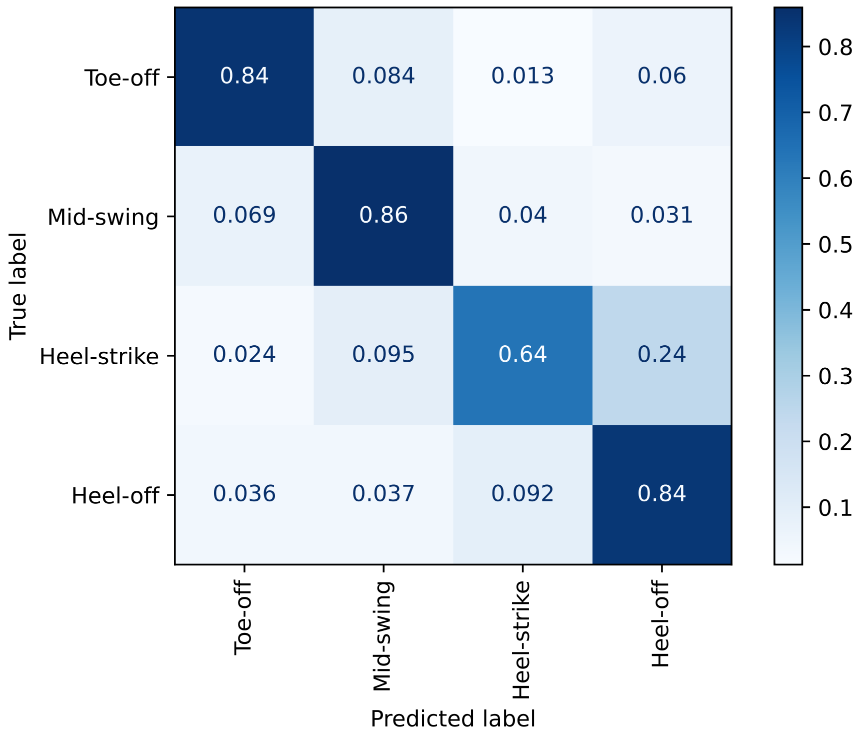


Figure 4.7 : Confusion matrix of trained model.

to worker and broker" scenarios but a shorter allocation time than the scenarios in the LTE testbed. In the LTE testbed, "Discovery and connection to worker" scenario follows a process similar to "Discovery and connection to worker" scenario in the Wi-Fi testbed, first discovering devices. Differently, the connection is established via the proxy module. Therefore, it has a longer allocation time than "Connection to cloud" scenario. In the "Connection to cloud" scenario, the mock client connects to the Internet via a cellular network, thus it has a longer allocation time than the same scenario in the Wi-Fi testbed.

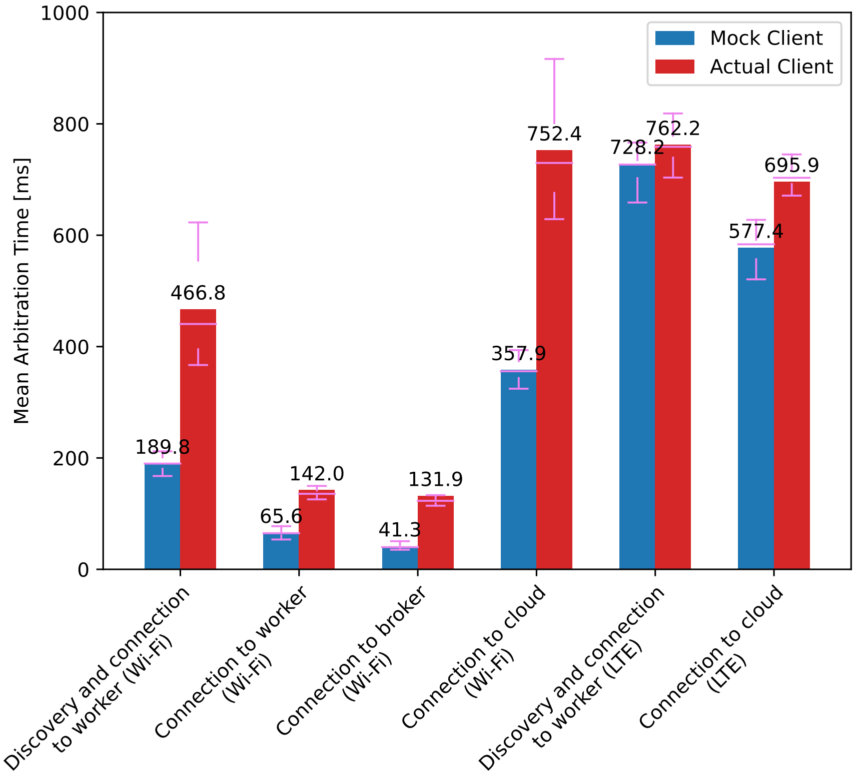


Figure 4.8 : Arbitration time in different scenarios.

When comparing the mean arbitration times of the actual client and the mock client, there is a slight difference in device performance with the actual client showing slightly lower performance. However, in all test scenarios, this trend seems to be consistent with the arbitration time being shorter than 1 second.

Figure 4.9 gives the mean latency in different devices and testbeds. The latency in worker and broker devices (Wi-Fi) is similar and lower than the rest of the units. Cloud has higher latency due to its multi-hop connection. However, worker (LTE)

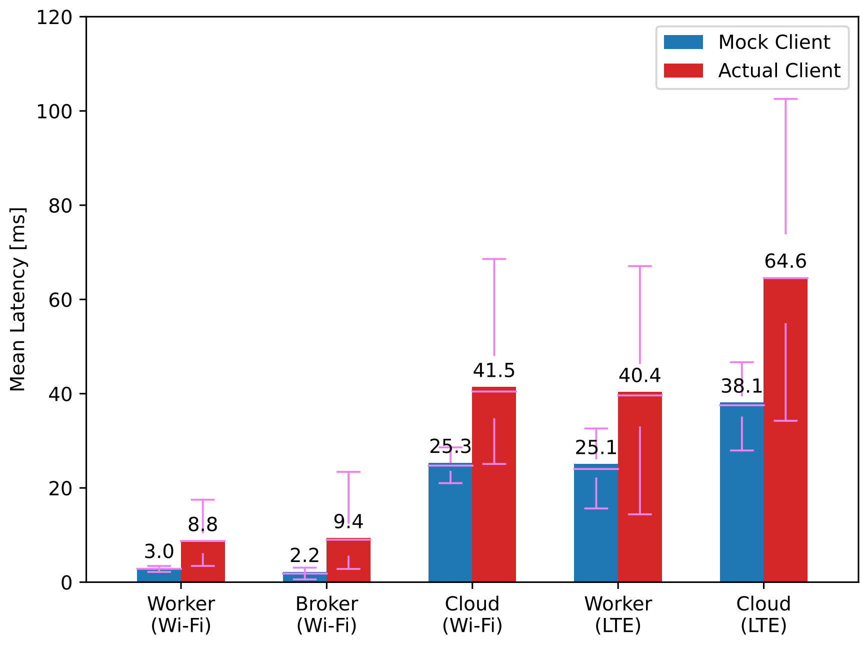


Figure 4.9 : Latency in different devices and testbeds.

and cloud (Wi-Fi) have similar latency. This is because worker (LTE) involves longer data transmission times through the air and the use of additional devices like the proxy module. Due to the impact of the cellular network on data, cloud (LTE) has higher latency compared to all other cases. When comparing the mock client and the actual client, the mock client has lower latency, and the latency variation is lower compared to the actual client.

Figure 4.10 provides the mean queuing delay in different devices and testbeds. The broker device has the highest queuing delay because it is involved in various tasks such as proxy and node management. Other devices have substantially lower queuing delays since they have fewer tasks. In devices connecting via cellular networks, queuing times are slightly increased due to variations in data arrival times. For instance, cloud (Wi-Fi) has 1.4 ms mean queuing delay for the actual client, whereas cloud (LTE) has 1.8 ms for the same client. Similarly, the queuing delays for the mock client are lower than the queuing delays for actual clients as the mock client sends data more stably.

Figure 4.11 demonstrates the mean execution time in different devices and testbeds. As expected, the worker device has the highest computation time. The broker device has a lower execution time than the other devices because it has a more powerful processor.

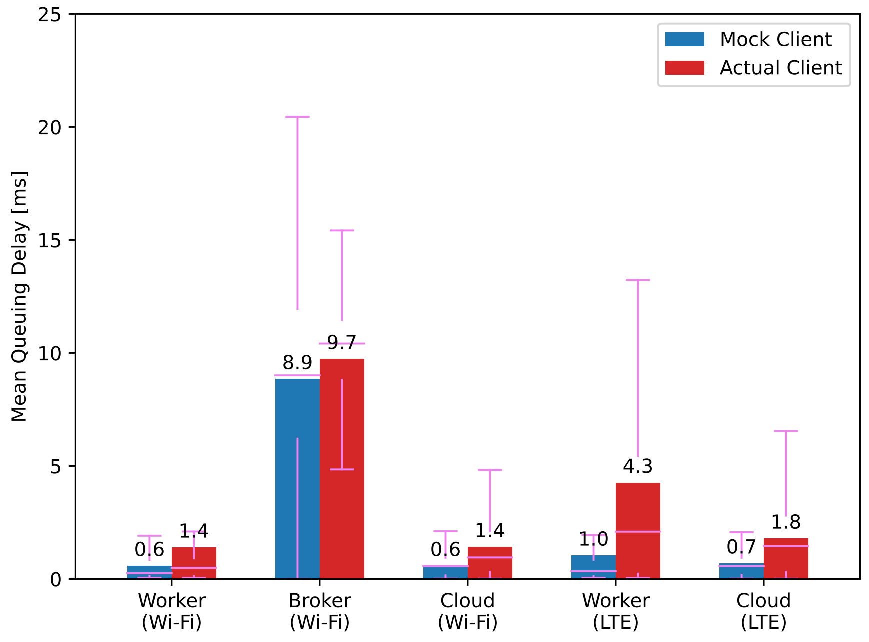


Figure 4.10 : Queuing delay in different devices and testbeds.

The connection types do not have any impact on the execution time. Similarly, the queuing delays for the mock client and the actual client have approximately the same execution times.

Figure 4.12 provides the total response time in different devices and testbeds. Worker devices cause the lowest response time among their respective connection types, indicating that workers perform their duty both adequately and effectively in the fog architecture. The broker device also has a shorter response time compared to the cloud device due to lower network delay. The response time for the actual client is higher than the response time for the mock client. On average, the lowest total response time is 10.5 ms for the mock client test and 22.3 ms for the actual client device. The highest response time belongs to cloud (LTE), which is 131 ms for the actual client.

Figure 4.13 illustrates the variation in jitter, showing that the worker (Wi-Fi) and the broker (Wi-Fi) have approximately 10 ms of jitter. However, this duration increases in the LTE testbed. Both worker (LTE) and cloud (LTE) have the same level of jitter. Considering that high jitter is not a desirable feature, worker devices outperform other devices as expected. Jitter for the actual client and the mock client have similar values in different scenarios.

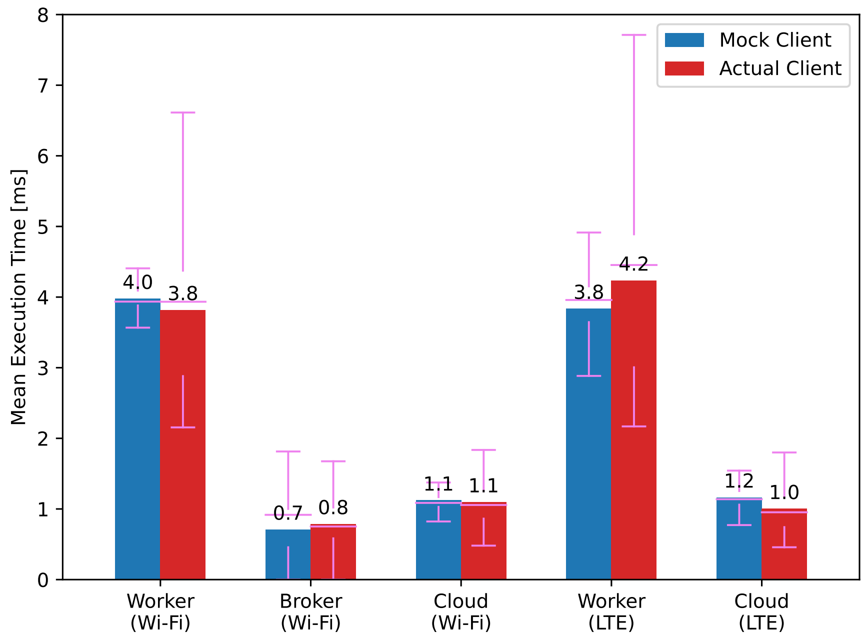


Figure 4.11 : Execution time in different devices and testbeds.

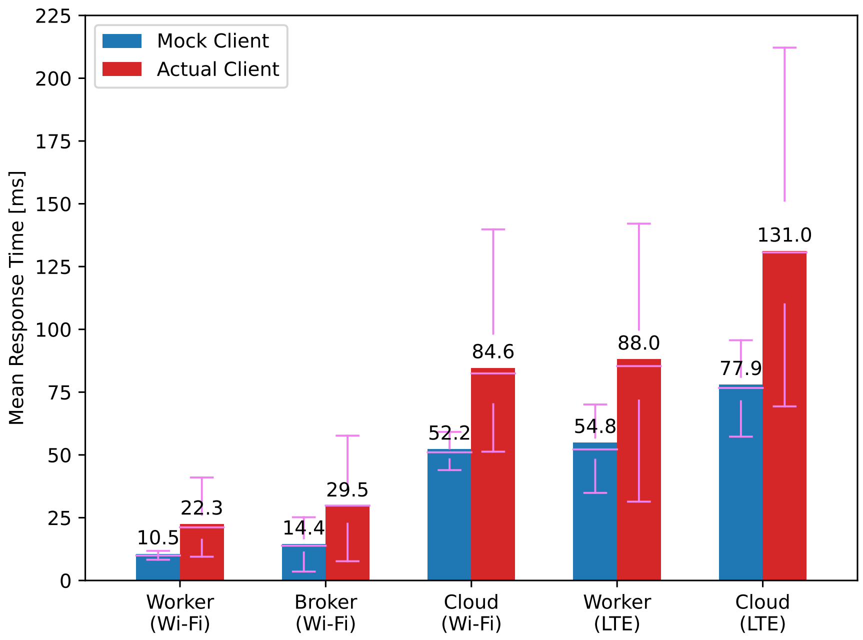


Figure 4.12 : Total response time in different devices and testbeds.

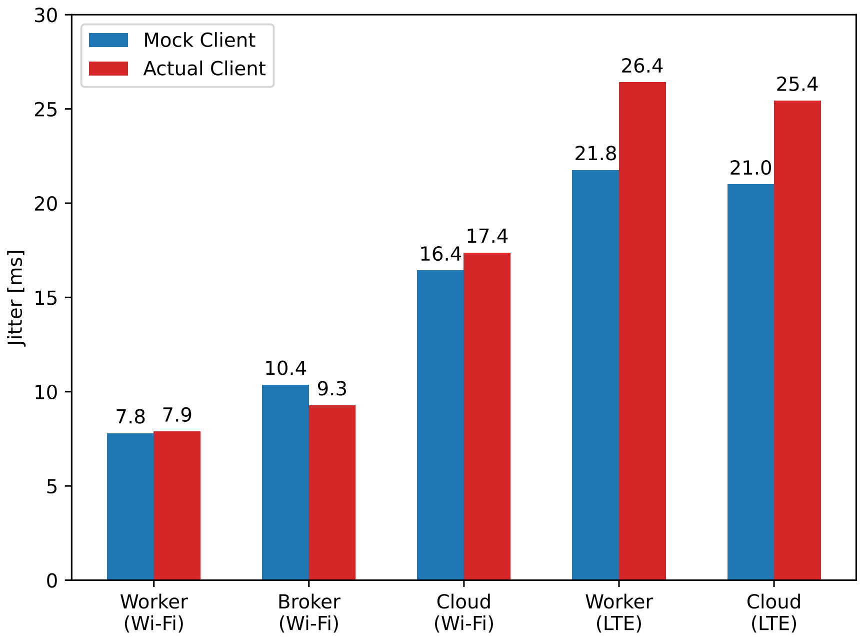


Figure 4.13 : Jitter in different devices and testbeds.

4.7.3 Results on resource usage

Figure 4.14 shows the CPU and memory usage results of computing service devices in different testbeds. The broker device has the lowest CPU usage rate because it has a powerful system. The proxy module, acting as a data transmitter without performing calculations, has used less CPU. The cloud device is relatively more powerful than worker devices and is optimized for computation, so it has lower CPU usage. Despite being the device with the highest CPU usage, the worker device has a very low value, around 10%.

The worker device has the lowest memory usage in all testbeds. The cloud device has slightly higher memory usage because it stores resource data for broker and worker devices. The broker device shows higher memory usage than expected. This is because the broker device has the only user interface in the operating system, which increases memory usage. Overall, memory usage is well below the threshold values set in our experiments with the FogETex framework.

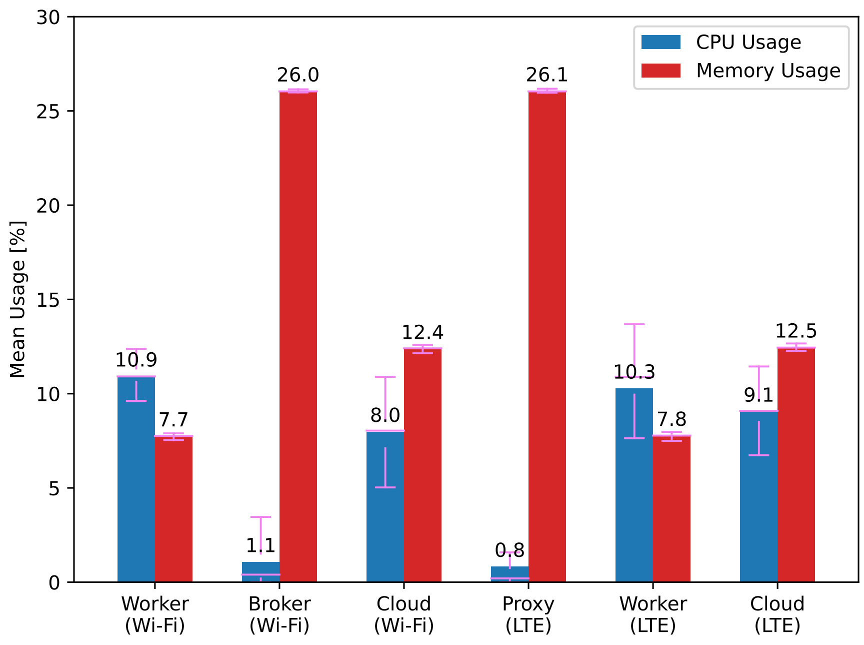


Figure 4.14 : CPU and memory usage in different devices and testbeds.

4.7.4 Results on network bandwidth usage

Figure 4.15 shows network bandwidth usage in different devices and testbeds. The lowest bandwidth usage is observed in the worker (Wi-Fi). The worker and cloud devices in the Wi-Fi testbed and LTE testbed show similar bandwidth usage. However, the broker device has higher bandwidth usage due to its various tasks, such as transferring resource data from worker devices, fog node assignment, and proxy. The proxy operation shows even higher bandwidth usage because it performs two-way message transmission during its task in the LTE testbed.

4.7.5 Stress test

Figure 4.16 shows the average response time of different devices based on the varying number of users. The error bar in the line graph represents the standard deviation, indicating jitter. For this application, worker (Wi-Fi) has the lowest average response time up to 6 users. Similarly, worker (LTE) performs better than cloud (LTE) up to 6 users. The broker shows worse results than worker (Wi-Fi) up to 6 users but can serve up to 18 users and consistently outperforms the cloud at all user counts. In the

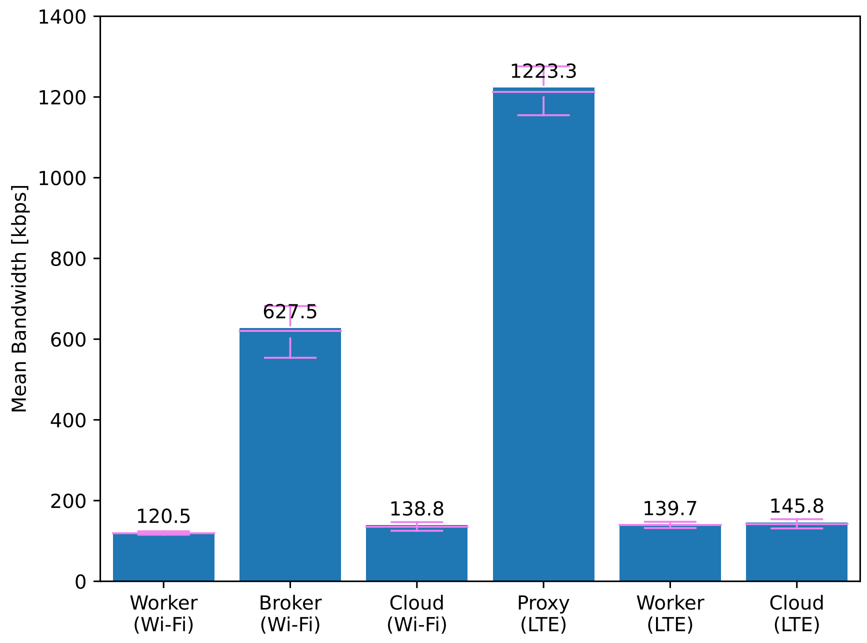


Figure 4.15 : Network bandwidth usage in different devices and testbeds.

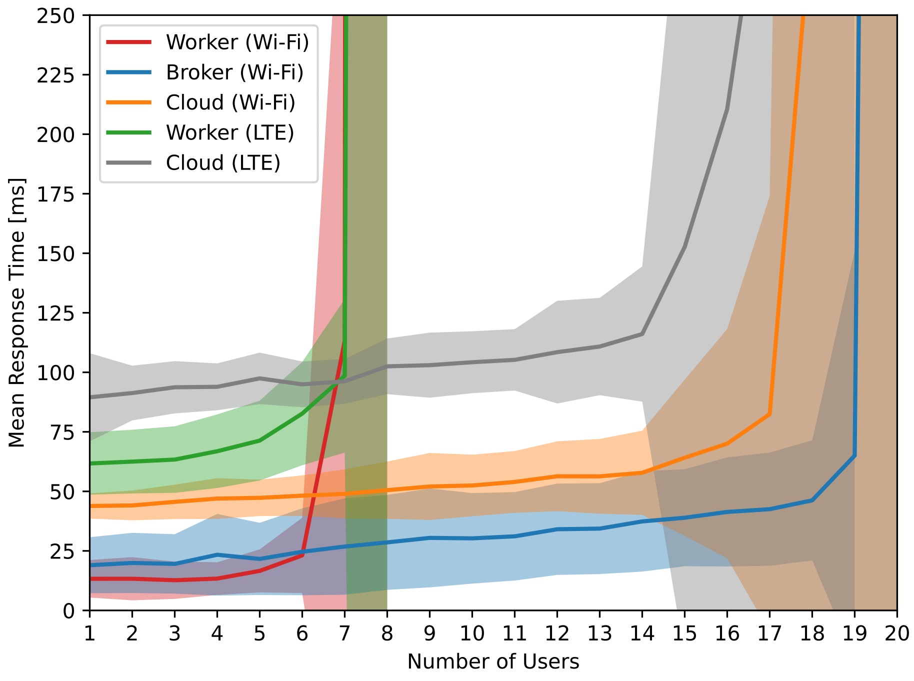


Figure 4.16 : Mean response time in different devices with varying numbers of users.

cloud, the system continues to operate successfully up to 14 users. After this point, jitter increases rapidly, and the system becomes unable to serve more than 16 users using LTE and 17 users using Wi-Fi.

Although the number of users a single worker can serve is lower compared to other devices, it is expected that a fog node will contain dozens of workers, depending on the need. Considering the cost of the devices, the worker is significantly cheaper than both the broker and the cloud rental costs, making it more advantageous in terms of price performance.

4.7.6 Performance benchmarking

In this section, we aimed to compare the system's performance with other studies. However, due to implementation constraints, many studies have been conducted through simulations, and many do not adhere to a common standard. As a result, performance comparisons could only be made with the study HealthFog [262], which has a similar structure to our work. In fact, several studies, such as Smart VetCare [247] and FETCH [277], also utilize the FogBus [261] architecture, just like HealthFog. HealthFog was chosen for comparison because it includes a deep learning module, like our system, and it is developed in Python, making it the most similar study to ours. Our gait phase recognition system has been integrated into this study for comparison.

Table 4.2 shows the performance comparison between HealthFog and our study. Our system outperforms in many metrics, with a significant difference observed particularly in response time and latency. In our system, the response time for worker, broker, and

Table 4.2 : Comparison performance of FogETex with other works.

	Worker		**Broker**		**Cloud**	
Metric	HF	FogETex	HF	FogETex	HF	FogETex
Arbitration Time [ms]	82.4	**65.6**	90.4	**41.3**	**209.9**	357.9
Latency [ms]	89.2	**3.0**	71.7	**2.2**	93.3	**25.3**
Queuing Delay [ms]	**0.5**	0.6	13.8	**8.9**	**0.4**	0.6
Execution Time [ms]	4.6	**4.0**	2.4	**0.7**	1.3	**1.1**
Response Time [ms]	101.3	**10.5**	107.3	**14.4**	174.2	**52.2**
Jitter [ms]	8.2	**7.8**	**8.1**	10.4	33.0	**16.4**
Frequency [Hz]	9.9	**50.0**	9.3	**50.0**	5.7	**50.0**

cloud is 10.5 ms, 14.4 ms, and 52.2 ms, respectively, whereas in the HealthFog (HF) study, it is 101.3 ms, 107.3 ms, and 174.2 ms. Due to the structure of the HealthFog system and its high response time, the maximum operational frequency is 9.9 Hz, while our system can consistently handle data at a fixed frequency of 50 Hz and respond to these requests.

4.8 Discussion

The worker device has shown successful performance when looking at characteristics that affect the user, such as latency, total response time, queuing delay, and jitter. This demonstrates that fog computing possesses the characteristics it should have inherently. In the LTE testbed, it has also outperformed the cloud. While it shows slightly worse results in terms of execution time compared to broker and cloud devices, with more than one worker in the system, these additional devices will decrease the average execution time as the number of users increases.

Worker devices have higher CPU usage as they are relatively less powerful devices. The main goal here is to establish a system with high computational power using low-cost devices. Similar to execution time, when multiple worker devices are connected, the total processing power is expected to be higher than that of the broker and cloud. In terms of memory usage, however, worker devices have shown better results.

Morover, the worker device has demonstrated the most successful performance in terms of network bandwidth. Thus, it has shown that fog computing systems possess many essential features such as low response time, low latency, real-time response, and low network bandwidth. This confirms that FogETex is an appropriate platform for e-textile applications relying on machine learning-based computational requirements.

On the other hand, when comparing the mock client and the actual client, the mock client performed better. In the mock client, data is read from memory and transmitted to computing devices every 20 ms. The mock client provides an ideal testing environment for computing devices. However, the actual client showed lower performance due to several factors, such as being a less powerful device, issues caused by traffic in the data transmitted from the T-IoT device, background applications, and the load created by the graphical user interface of the mobile application.

4.9 Limitations

The proposed FogETex framework offers low latency, response time, queuing time, jitter, and resource usage, but it also presents several challenges and limitations such as distributed system management, security and privacy issues, data consistency, latency and bandwidth limitations, heterogeneity, energy efficiency, and scalability. While the distributed architecture is beneficial in terms of latency, it also increases the complexity of the system, making resource management more difficult and raising maintenance costs. Since electronic textile products typically handle human-related data, additional administrative efforts will be required to ensure data security and user privacy within the distributed fog architecture.

Another challenge is data synchronization and management. Data comes from various devices, and it is critical that it is transmitted without corruption, as any discrepancies could lead to incorrect predictions by the system. Although fog computing reduces latency and network bandwidth usage, Internet infrastructure may not be sufficiently robust in all areas, necessitating additional enhancements in those regions. The use of different types of edge devices could lead to software and hardware issues over time. While the proposed framework is designed to work across various device types and operating systems, hardware and software changes may require additional effort.

One of the most significant limitations of this study is that the tests are conducted with a single worker. Although the developed framework supports an unlimited number of workers, an analysis of different numbers of worker devices in indoor and outdoor operations and a review of the results are necessary. Based on these results, system administrators can better determine the required number of devices and their configurations according to the number of users.

Another limitation is the decrease in battery life and potential issues with data transfer in T-IoT devices due to daily usage. At this point, the system may experience data loss. To address this, intermittent computing techniques can be used, allowing the

T-IoT device to operate in low-power mode. During this mode, data transfer over Bluetooth can be interrupted, and the device can switch to data collection mode only. Once the battery level returns to a non-critical state, the collected data can be sent to the fog system.

When displaying resource usage in the system, it is more appropriate to normalize it based on device performance. However, since the system includes devices with different architectures, normalization using metrics like processor clock count may not be accurate. Therefore, resource usage had to be presented with their original values rather than being normalized.

In addition, since the system was tested on real devices, it is more susceptible to external factors, leading to limitations in performance comparisons. An emulator system that includes T-IoT devices and gateways could be developed to address this issue. This way, external factors in the system could be controlled, allowing for more accurate performance comparisons.

In addition to these, while fog nodes are independent within the system, there is a limitation of a centralized approach in resource monitoring and fog node assignments. Fully distributed or federated system architectures could make it easier to manage the system internally. Although the complexity level would increase, the problem of node issues affecting each other would decrease.

Using a large number of devices and continuously operating them will increase energy consumption. Depending on changing needs, there may be limitations in terms of energy efficiency. Finally, scalability is a key limitation for ensuring the continuity of services within the system. Although the framework offers dynamic flexibility in adding devices, physically installing these devices and identifying regional needs for devices will still be necessary.

4.10 Conclusion

In this thesis, we proposed a novel fog computing-based framework for electronic textile applications. The FogETex framework challenges the computational load of low-power microcontrollers commonly used in electronic textile products, thus enabling designers to address concerns about comfort. Since the framework is developed as the platform as a service model, it can be used not only for e-textile products but also for other applications that require fog computing architecture. The framework has been developed using a cross-platform software approach, allowing it to run on different processors and operating systems. In the infrastructure, computation services can be sent in sequence through bidirectional peer-to-peer connections using Web socket connections. For single requests, queries are made through HTTP RESTful API connections. The framework has been developed using NodeJS for communication processes and Python for deep learning-based computation services. A user interface is developed for resource monitoring of devices and tracking the fog structure.

The system is tested with a real-time and real-world problem using deep learning-based gait phase analysis, in which textile-based capacitive sensors are used for phase prediction. The entire system, from the sensor to the fog node and cloud, is implemented and tested from a holistic perspective. In the framework tests, system performance in an ideal environment using mock client tests as well as real-world results using an actual client test with a mobile application were examined. The time characteristics, resource usage, and network bandwidth usage of the FogETex architecture were studied in different devices and testbeds. The worker devices in the FogETex infrastructure have shown low latency, low response time, low queuing time, low jitter, and low bandwidth usage in different experimental scenarios, demonstrating that they meet the requirements of fog computing. This system supports a large number of users and has a lower latency compared to its competitors in the literature.

5. ASSISTIVE SOFT ROBOTIC GLOVE CONTROL USING FogETex

In this chapter, we present a novel IoT system for remote rehabilitation using FogETex framework. This system utilizes a sensing Textile-based IoT glove (T-IoT [278] glove) as the master and a pneumatic actuating T-IoT glove as the slave within a remote rehabilitation framework. In this study, we also validate the performance of the closed-loop control system through the cloud computing system. Equipped with capacitive textile sensors on each finger, the T-IoT glove is paired with a wireless transmitter incorporating a machine learning-based recognition of the finger movements of the medical staff. The actuating T-IoT glove mimics the finger motions captured by the sensors to assist individuals remotely with the required motions.

The sensing T-IoT glove system integrates a Beetle Bluetooth Low Energy (BLE) for the acquisition of sensor signals and transmitting them to the cloud via the medical staff's client program. The cloud system employed signal processing and machine learning analysis techniques to facilitate comprehensive telerehabilitation, requiring preprocessing and pattern identification to efficiently control the actuating T-IoT glove. Cloud services are utilized to boost response time with minimal latency and fast transmission, as well as to reduce computational load [142,144]. This entails transferring the healthcare system's service module to the cloud, assigning computing resources according to users' health conditions, and facilitating prompt interaction through Transmission Control Protocol (TCP) and Internet Protocol (IP). The computation unit in the cloud performs gesture recognition with a 93.95% accuracy using Machine Learning (ML) techniques applied to the sensor data from the sensing T-IoT glove. It generates real-time control commands for the actuating T-IoT glove. The actuating T-IoT glove replicates the recognized actions, while computational tasks are efficiently handled by the cloud system, leaving only the communication tasks to run on edge devices with low computational resources.

To our knowledge, no existing system offers hand gesture recognition through a sensing T-IoT glove and consequent remote control of the actuating T-IoT glove utilizing machine learning over a cloud computing system all performed in real-time. We anticipate that combining cloud computing with machine learning-driven signal analysis will open up fresh opportunities, facilitating distinctive automatic detection, recognition, and prediction abilities in areas such as healthcare assessments and home-based rehabilitation.

5.1 System Architecture and Material Designs

The structure of the proposed telerehabilitation system based on cloud computing is illustrated in Figure 5.1. It comprises three main components: a sensing T-IoT glove for gesture recognition, an actuating T-IoT glove for rehabilitation, and an intermediate cloud computing architecture connecting the two gloves.

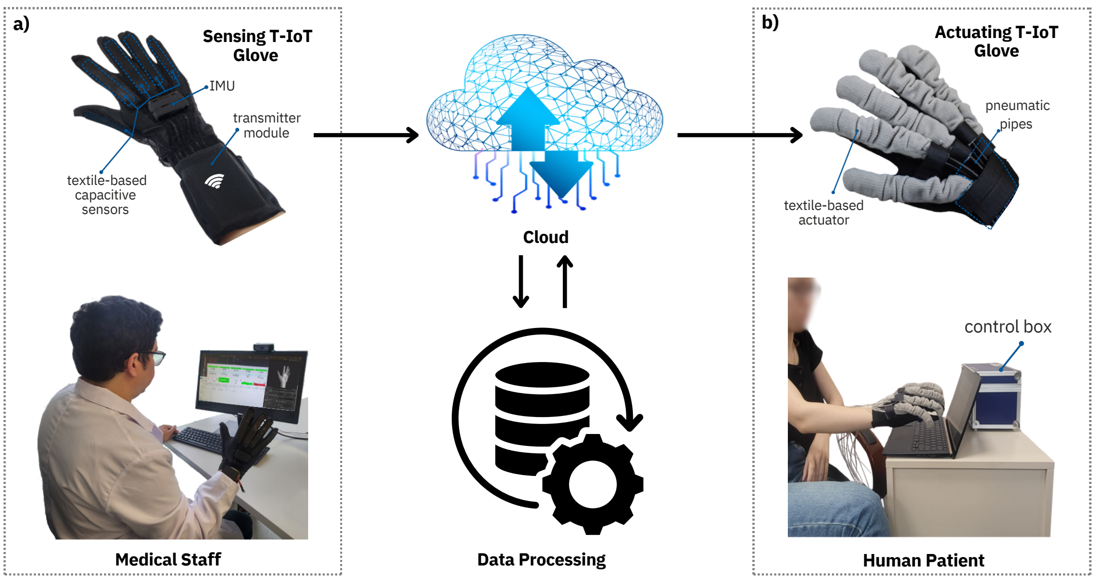


Figure 5.1 : Telerehabilitation over the cloud with the medical staff wearing a sensing T-IoT glove and the human patient wearing an actuating T-IoT glove for telerehabilitation: a) Sensing T-IoT glove, b) Actuating T-IoT glove.

5.1.1 Design of the sensing T-IoT glove

The sensing T-IoT glove consists of five capacitive-based textile sensors that are highly stretchable, making them fit the human hand. These highly sensitive capacitive sensors consist of two layers of knitted fabric coated with silver nanoparticles (Shieldex Medtex-130, V Technical Textiles Inc., USA), which are utilized as upper and lower electrodes. Additionally, a silicon layer (Ecoflex 00-30, SmoothOn Inc., USA) works as the dielectric material between them.

The change in capacitance in capacitive sensors is typically due to variations in the size of the sensor and can be utilized for detecting joint movements [278]. Figure 5.1(a) illustrates the sensing T-IoT glove, its sensors on the fingers, and the transmitter module on the wrist area. Silicon is cast on a conductive fabric with the desired thickness according to the required application, which also affects the performance of the sensor in terms of sensitivity and working range. The manufacturing methodology of these capacitive sensors, along with their working principle and characterizations are explained in detail in our previous work [47].

After cutting the electrodes into the desired shapes using a laser cutting machine, particularly along the sides of the fingers, the silicon was cast. Following the silicon casting, these layers were merged and left to cure in an oven. Once shaped, the sensors were carefully positioned over the fingers of the glove. Connections with the Beetle BLE (DFRobot, Shanghai, China) development board and MPR121 capacitive sensor controller (Adafruit, New York, USA) were established using Thermoplastic PolyUrethane (TPU) coated conductive yarns. This approach ensures that the flexibility of the sensing T-IoT glove is not compromised and prevents the yarns from getting into contact with each other, thus minimizing the risk of short circuits and reducing parasitic capacitance.

5.1.2 Design of the actuating T-IoT glove

The key components of the proposed actuating Textile-based IoT glove are illustrated in Figure 5.1(b). The actuating T-IoT glove consists of textile-based knitted actuators capable of extending and bending upon applied force. The creation of such anisotropy is made possible by arranging the dorsal and plantar surfaces of the produced shells of the actuators to lead to localized expansion of fabric and actuator bending motion. This arrangement of knit loops and courses per centimeter in the computerized knitting machine (SHIMA SEIKI) controls needle yarn carriers simultaneously to create desired patterns. Another promising advantage of using computerized knitting during the production of actuators is eliminating the burdensome, time-consuming steps of producing actuators via the cut-and-sew approach, which requires manual assembly.

The machine we employed allows for controlled mass production and reduces variation in products caused by manual assembly, meeting the demands of the health sector. Within these actuators, TPU sheets (Stretchlon 200, Fiber Glast) are utilized to create flexion and extension air pouches sealed by welding (impulse sealer PCS 300, Brother). The actuators are then mounted onto a base and worn using Velcro finger cuffs. Each bladder was manufactured by laser cutting two identical rectangles measuring 17×2.5 cm. The pneumatic system has been designed to provide forces to the actuators that will lead to their inflation, enabling the required action through fingers that are detected, processed, and conveyed via the sensing T-IoT glove and cloud computing. Eventually, the actuating T-IoT glove provides safer interactions with patients, thanks to the inherent lightweight and compliant nature of textile products. As these components communicate with each other via the Internet using IoT protocols, it ultimately causes the patient's hand to close and open during the therapy session. Detailed information about the material and methodology can be found in [8].

5.2 Data Processing

The proposed system consists of three main components: the sensing T-IoT glove, the actuating T-IoT glove, and the cloud. Figure 5.2 illustrates the sensor and control signal transmission architecture of the proposed system. The exercise movements performed by the medical staff. The sensing T-IoT glove captures the medical staff's finger movements via the capacitive sensors and transfers these data to a computer via Bluetooth, and then to the cloud in real time via the Internet. The cloud system processes the sensor data from the sensing T-IoT glove using signal processing and machine learning techniques to generate appropriate control signals for the actuating T-IoT glove. In addition to computing services, it also establishes a communication infrastructure for data transfer between the sensing and actuating T-IoT gloves. The actuating T-IoT glove worn by the patient is responsible for executing the exercise movements received from the medical staff.

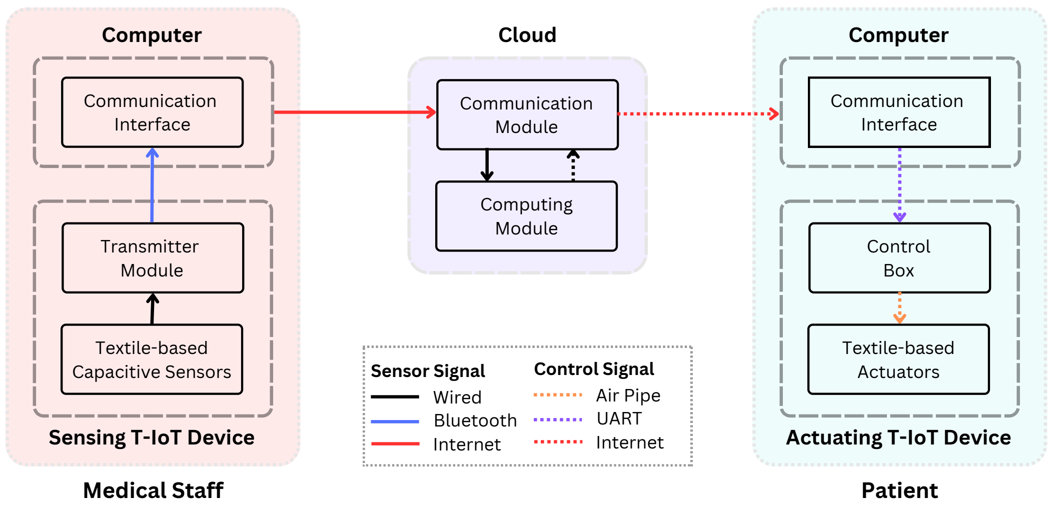


Figure 5.2 : System architecture of sensor and control signals transmission.

5.2.1 Data processing in the sensing T-IoT glove

The sensing T-IoT glove consists of five textile-based capacitive strain sensors and a transmitter module. The strain sensors are placed on the fingers. Depending on the movement of the fingers, there is a change in capacitance due to the mechanical alteration in the length and thickness of the sensor. Because of this mechanical effect, the capacitance increases when a finger is in a closed position and decreases when it is in an open position.

The transmitter module consists of a Beetle BLE, an MPR121 capacitive sensor controller, and a 3.7 V LiPo battery. The MPR121 capacitive sensor controller is used to measure the capacitance of the textile-based capacitive strain sensors. One end of each sensor is connected together and attached to the ground of the module, while the other ends are connected to the electrode inputs. The module charges the sensors by providing the configured current value to each sensor individually for the assigned duration. The charge stored in the capacitance sensor is calculated as follows:

$$Q = I \cdot T, \tag{5.1}$$

where Q is the stored charge, I is the assigned current value, and T is the charging time. The stored charge creates a voltage on the capacitive sensor, which is calculated as follows:

$$Q = C \cdot V, \tag{5.2}$$

where C is the capacitance value of the sensor, and V is the voltage value resulting from the stored charge. At the end of the charging period, the capacitive sensor controller measures this voltage value via electrodes accessed through I2C. The values of I and T are determined during the configuration stage, and since the value of V can be obtained from the module, the capacitance value of the sensor is calculated as follows:

$$I \cdot T = C \cdot V \tag{5.3}$$

$$C = \frac{I \cdot T}{V} \tag{5.4}$$

The minimum capacitance and supply voltage of the sensors are used to configure the values of I and T. Using Equation (5.4), $I \cdot T$ can be determined as follows:

$$I \cdot T < C_{min} \cdot V_{dd}, \tag{5.5}$$

where C_{min} is the minimum capacitance value, and V_{dd} is the supply voltage of the capacitive sensor controller. Additionally, the capacitive sensor controller incorporates a 2-level filter structure to denoise the capacitance measurements.

The Beetle BLE, which is an Arduino-based development board, collects the voltage values generated by the strain on the sensors in all five fingers at a sampling frequency of 50 Hz using the I2C protocol. These collected data are transmitted in real-time to a computer via Bluetooth. In the computer environment, a developed interface receives the data and transfers it to the cloud via a TCP socket connection established with the cloud.

5.2.2 Data processing in the cloud architecture

The cloud architecture has two primary tasks: communication and computation. In the communication task, data from finger sensors sent by medical staff is received via a socket connection. These data are added to the buffer associated with the user. Subsequently, through feature extraction, normalization, and machine learning, the raw sensor data for each finger is transformed into four control states: “Opening”, “Open”, “Closing”, and “Close”. These control signals are then transmitted in real-time to the actuating T-IoT glove on the patient site via a socket connection.

The communication module in the cloud serves as a socket server that connects the patient and the medical staff. Users and medical staff connect to the system using tokens created specifically for them. These tokens contain the role and meeting information of the users, which is used to match them accordingly. Information from the medical staff's sensing T-IoT glove is processed and sent as control signals to the patient's actuating T-IoT glove.

Additionally, the communication module transfers sensor data from the medical staff to the computation module. The computation module can be configured within the same cloud device or across different cloud devices. To minimize network latency, it is recommended to have the computation module within the same device.

The computation module is responsible for processing the sensor data transmitted by the communication module. As soon as a medical staff connects to the system, the computation module creates an object for the therapy service. This object contains a buffer for storing real-time data for each finger, calibration parameters specific to the fingers, and machine learning models. The buffer is used to extract time series features from the real-time incoming data.

To use machine learning, raw data, as well as the first, second, and third derivatives of the data for each finger, are extracted. These features are then normalized using a standard scaler [279]. To determine the parameters of the standard scaler, the medical staff is expected to perform several consequent grasp movements for calibration when connected to the system. After this period, the standard deviation (SD) and mean values of features for each finger are recorded for the standard scaler, and the same calibration parameters are used throughout the operation.

The extracted features are converted into control signals using pre-trained classification models for each finger. In this study, models were trained using machine learning methods such as Logistic Regression (LR), Decision Tree (DT), K-Nearest Neighbors (KNN), Multilayer Perceptron (MLP), and XG-Boost (XGB). The grid search method was used for hyperparameter tuning to improve the performance of the models. The trained models are stored in the cloud and are loaded into their respective objects each time a new medical staff connects to the system.

The real-time predictions of finger states made by the machine learning models are transmitted to the communication module, to be further sent to the patient in real time via a socket connection. The use of socket connections in the communication module establishes a seamless end-to-end data bridge between sensing and actuating T-IoT devices. This prevents the need to repeatedly establish connections for each data transmission, as required in systems utilizing SOAP or REST, avoiding repeated handshaking operations.

Furthermore, the actuating T-IoT glove application does not need to continuously query the server to check if a control signal has been generated. When a signal is generated, it is directly transmitted to the device by the server.

5.2.3 Control of the actuating T-IoT glove

The actuating T-IoT glove operates on the principle of fluidic drive. Pressurized air created by an air pump reaches the bladders through valves. There are two bladders for each finger. When air is supplied to the upper bladder, the actuator bends; when air is supplied to the lower bladder, the actuator extends. When pressurized air is given to one bladder in a finger, the air release valve of the other bladder opens, and the air inside is expelled due to the pressure created by the inflation of the other bladder. The working principle and control scheme of the valves are addressed in detail in [8].

The client application on the patient's computer facilitates the connection between the cloud and the actuating T-IoT glove. The control signal generated by the cloud is sent to the patient's computer. The client application transmits the control signals to the microcontroller of the actuating T-IoT glove via the serial port. The microcontroller initiates the task of opening or closing the respective finger whenever there is a change in the current finger state.

To detect the opening and closing actions of the fingers, the microcontroller is equipped with a pressure sensor for each bladder. Before starting rehabilitation, the open and closed positions for each finger are set via the client screen, and the threshold values for the bladders are determined. When a bladder reaches its threshold value in the opening or closing state, the movement is considered complete. The microcontroller controls each finger individually. If multiple fingers are moving simultaneously, the pressure in each finger is controlled separately. If one finger completes its movement, pressurization for that finger stops while the others continue their movements.

5.3 Experiments

In this section, the test conditions for evaluating the proposed telerehabilitation system are described. Details about the characterization of T-IoT gloves, data collection and labeling, demographic information of the test subjects, as well as the experimental setup, experimental scenarios, and evaluation criteria are provided in detail.

5.3.1 Characterization of T-IoT gloves

In this study, a Sensing T-IoT Glove was used to detect the finger movements of medical staff, while an Actuating T-IoT Glove was employed to move the patients' fingers. Unlike previous studies, the characterization of sensors and actuators is associated with finger motion, since in this study the focus is on finger movement. During the testing of both gloves, finger movements were detected using a label-based image processing technique.

In the tests, the index finger was used. Labels were placed on the Distal Phalanx (DP), Proximal InterPhalangeal joint (PIP), and MetaCarpoPhalangeal joint (MCP) of the index finger, as well as on the thumb side as shown in Figure 5.3(a). By using image processing techniques, the positions of these points were determined. The angle formed by the DP and MCP points at the PIP point was identified and symbolized as α.

The bending angle of the index finger (β) was defined as the angle of the arc formed by the bending of the finger (Figure 5.3(b)) and calculated as follows.

$$\beta = 360 - 2 \times \alpha \tag{5.6}$$

The bending angle ranges between 0° and 360°. A bending angle of 0° corresponds to a fully straight finger, also referred to as the "Open" position. An angle of 360° indicates full flexion or the "Close" position. However, physically achieving a complete 360° closure is impossible, as it would require the two labels to overlap.

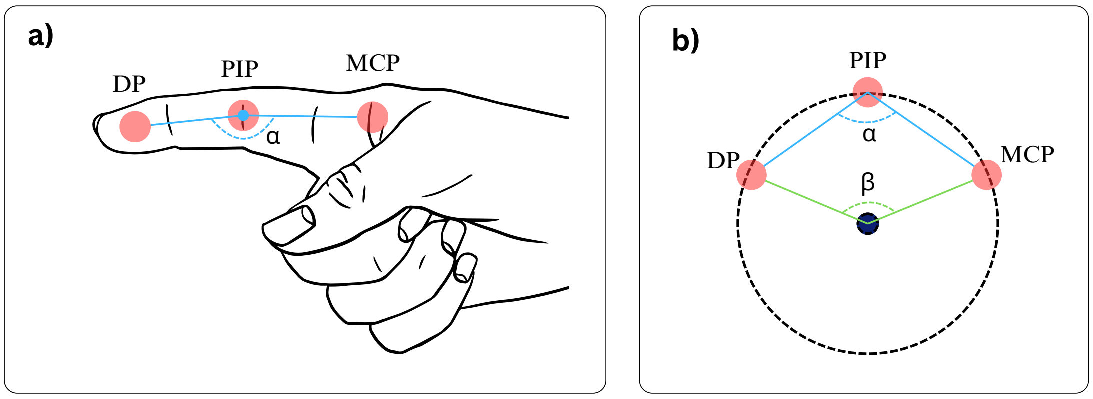


Figure 5.3 : Label Configuration for Bending Angle Calculation of Finger Movement: (a) Label Placement on the Index Finger, (b) Bending Arc Representation of Label Positions.

In the finger bending angle experiments of both sensing and actuating T-IoT gloves, an 8-cycle test was conducted following 4 cycles of preliminary tests, which all involve consequent flexion and extension movements.

5.3.2 Dataset and labelling

A machine learning-based control technique is applied for controlling the actuating T-IoT glove using the data from the sensing T-IoT glove. Machine learning is used to detect the states of the fingers, and the actuating T-IoT glove moves according to these states. To utilize machine learning techniques, appropriate data collected under suitable conditions is necessary. In this study, data is collected from 12 test subjects with an average age of 27.3 years, average anthropometrics hand data [280] of hand length of 17.1±1.3 cm, and hand breadth of 7.9±0.5 cm. Demographic information of the test subjects is provided in Table 5.1. Eight of the test subjects are female and four are male. Excluding the data collected for calibration in the dataset, an average of 161 minutes of data per finger has been used for machine learning. This study was approved by the Ethics Committee of Istanbul Technical University Human Medical and Engineering Research (SM-INAREK-2021-03). Written informed consent was obtained from all participants prior to data collection.

A procedure for data collection was laid down with medical experts the sensing T-IoT glove shown in Figure 5.4(a) was worn to the right hand. At the beginning of the test, subjects were instructed to open and close their hand as shown in Figure 5.4(b) several

Table 5.1 : Test Subject Demographic Information

Test Subject	**Age**	**Sex**	**Height (cm)**	**Weight (kg)**	**Anthropometrics Hand Data**	
					Hand Length (cm)	**Hand Breadth (cm)**
1	32	Male	172	92	19.2	8.7
2	25	Female	157	53	15.4	7.5
3	28	Female	167	50	17.5	7.5
4	31	Male	173	70	18.0	8.5
5	26	Female	164	55	18.5	8.0
6	31	Female	161	60	15.0	7.5
7	22	Male	181	67	17.0	8.5
8	22	Male	183	62	17.0	8.0
9	23	Female	167	69	18.0	8.5
10	40	Female	168	65	17.0	7.0
11	21	Female	163	69	16.0	8.0
12	27	Female	158	51	16.0	7.5
Average	27.3	F: 8	167.8	63.6	17.1	7.9
SD	5.5	M:4	8.2	11.5	1.3	0.5

times. This data was used for the calibration of normalization parameters specific to each test subject. All data, including calibration data, was saved into separate files during experiments.

After the calibration stage, subjects were asked to perform different movements. The experimenter shows the movement to the test subject at the beginning of each new movement. After the experimenter confirms that the test subject has correctly performed the movement, the recording of the test is initiated. Each movement was repeated sequentially 30 times. Subjects were instructed to wait around one second in the "Close" and "Open" states of the movements. They were allowed to perform movements at any speed they favored. During the movements, subjects were also asked to label active fingers using buttons. A five-second break was given to the subject before each new movement.

First, the test subjects performed a fist gesture, fully opening and closing the hand as depicted in Figure 5.4(b). Afterwards, they individually opened and closed each finger, as shown in Figure 5.4(c). Following them, they opened the thumb while closing and opening the other four fingers 30 times (Figure 5.4(d.I)). Finally, they

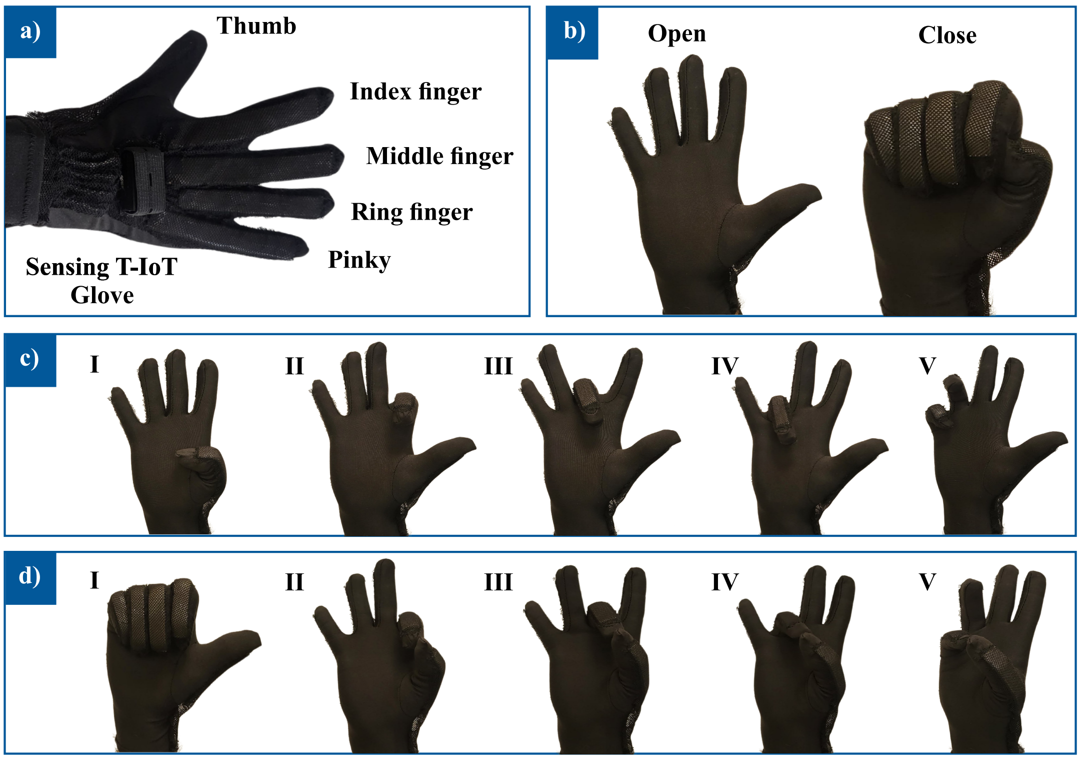


Figure 5.4 : a) Sensing T-IoT glove and its fingers. b) Sensing T-IoT glove calibration and fist movement: open hand and close hand. c) Single finger closing and opening: I. Thumb, II. Index, III. Middle, IV. Ring, V. Pinkie. d) Motions with a combination of fingers closing and opening: I. Open thumb and other fingers close, II. Cylindrical pinch with thumb and index, III. Cylindrical pinch with thumb and middle, IV. Cylindrical pinch with thumb and ring, V. Cylindrical pinch with thumb and pinkie.

sequentially touched the rest of the fingers with the thumb in cylindrical pinch movements, completing the data collection process as shown in Figure 5.4(d.II-V). The gestures performed by the sensing T-IoT glove and their corresponding gestures executed by the actuating T-IoT glove are depicted in Figure 5.5. Figure 5.5(a,b) presents the "Open" and "Close" states of the T-IoT gloves, which are critical for grasping and releasing motions. Figure 5.5(c,d) illustrates the necessary individual finger movements essential for single-finger exercises.

A GUI is designed for the experimenter as shown in Figure 5.6. This GUI is organized into several rows and columns, each displaying specific information and controls. In the first row, instantaneous analog values from the finger capacitive sensors are shown. The second row contains five columns: the first three columns display the X, Y, and Z gyroscope values of the glove, which are used to calculate its orientation, although these values are not utilized in this study. The fourth column in this row indicates the

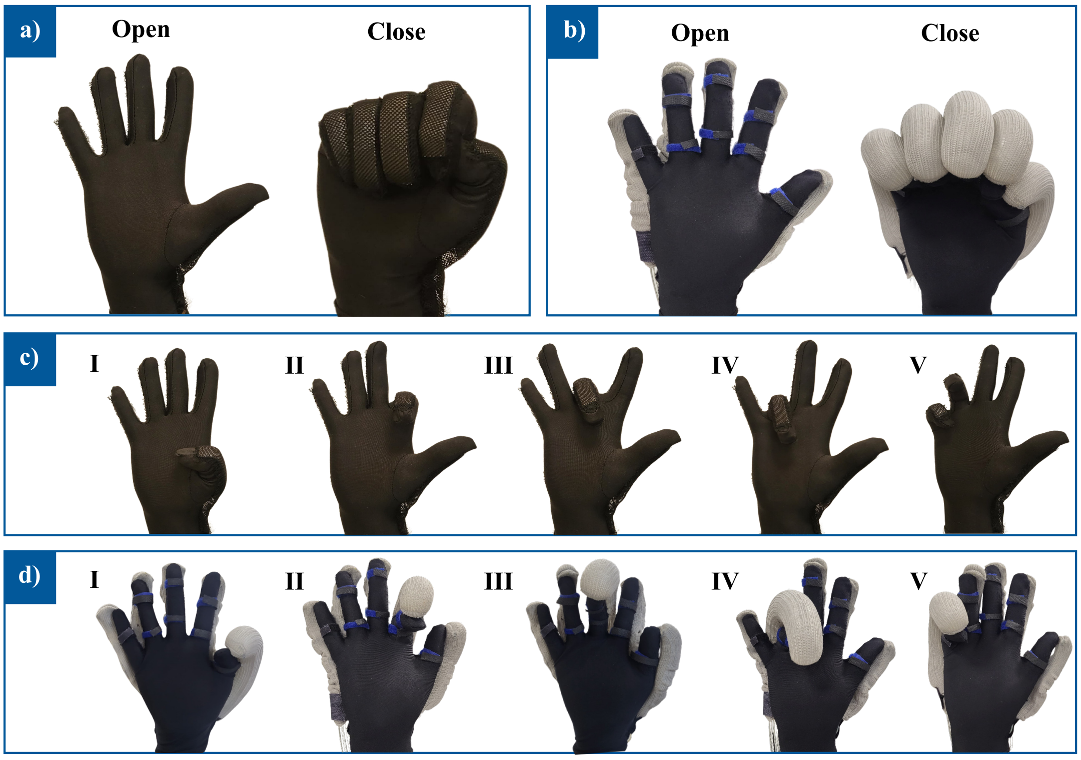
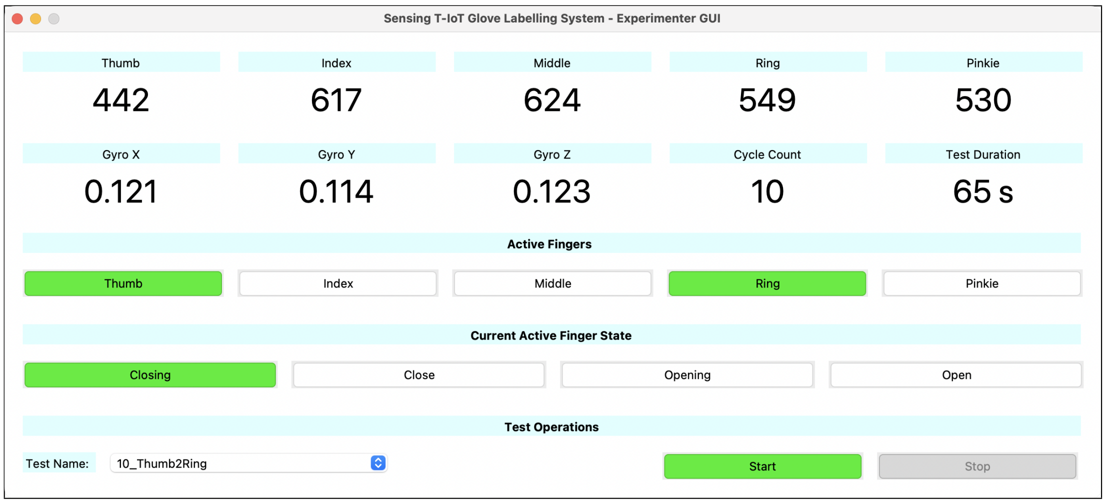


Figure 5.5 : a) Sensing T-IoT glove calibration and fist movement: open hand and close hand. b) Actuating T-IoT glove calibration and fist movement: open hand and close hand. c) Sensing T-IoT Glove single finger closing and opening: I. Thumb, II. Index, III. Middle, IV. Ring, V. Pinkie. d) Actuating T-IoT Glove single finger closing and opening: I. Thumb, II. Index, III. Middle, IV. Ring, V. Pinkie.


Sensing T-IoT Glove Labelling System - Experimenter GUI

Thumb	Index	Middle	Ring	Pinkie
442	617	624	549	530
Gyro X	Gyro Y	Gyro Z	Cycle Count	Test Duration
0.121	0.114	0.123	10	65 s

Active Fingers: Thumb, Ring

Current Active Finger State: Closing

Test Name: 10_Thumb2Ring

Figure 5.6 : Experimenter Graphical User Interface of the Sensing T-IoT Glove Data Acquisition and Labelling System.

cycle count of the current test, incrementing each time the state changes from “Open” to “Closing”. The fifth column shows the duration of the test.

The third row illustrates the active or moving status of the test fingers—Thumb, Index, Middle, Ring, and Pinkie—using green to represent active fingers and white for passive ones, arranged according to the test name. Finally, the fifth row is dedicated to test operations, featuring the test name on the left, the Test Start Button in the middle, and the Test Stop Button on the right.

Data from the glove is received in real-time through a Bluetooth module connected to the computer. The GUI communicates with this module via the Serial Port. Although data from the glove is sampled at a frequency of 50 Hz, the GUI updates the displayed data at a screen refresh rate of 10 Hz to improve performance and ensure the changes can be effectively tracked. The test subject’s name is entered through a configuration file. The test is performed by the subject is selected from the “test name” combo box, and active fingers are automatically chosen based on the selected test. The filename for each test is automatically generated according to the selected test name. Data collected during the tests is temporarily stored in a buffer and written to the file in bulk every 5 seconds.

Additionally, in the developed Web interface, sensor data from the 5 fingers and the user’s labels are displayed in real-time (Figure 5.7). The GUI sends the data it receives from the Bluetooth module in real-time to the web interface via WebSocket. The client program of the web interface instantly transfers the incoming data to charts. The experimenter monitors the data collection process, identifies any issues, and notes errors. After the experiments, the collected data are labeled manually as “Opening”, “Open”, “Closing”, and “Close”.

5.3.3 Experimental setup

In the proposed system, real-time data from the sensing T-IoT glove worn by the medical staff is converted into control signals for the actuating T-IoT glove worn by the patient. This setup provides telerehabilitation infrastructure for medical staff-patient interactions at separate locations.

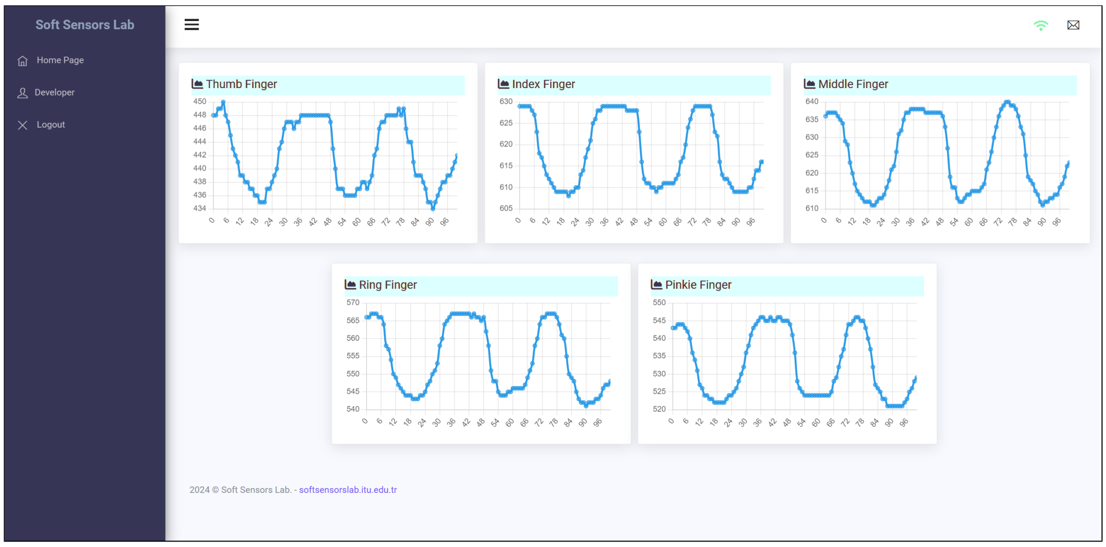


Figure 5.7 : T-IoT Glove Finger Sensor Virtualization Interface.

Performance testing of the developed system utilized a traffic generator and cloud devices as follows.

① **Traffic generator and receiver:** Dell Inspiron 15 5000 (Intel Core i7-8550U CPU @1.80GHz, 32GB DDR4 RAM, 240GB SSD, and Windows 11 Enterprise 64-bit). Software: Node.js v18.15.0 and Python 3.9.13.
② **Cloud device:** Digital Ocean VPS (2x Dedicated Premium Intel CPU, 8GB RAM, 50GB SSD, Cloud Location: Frankfurt/Germany).

Traffic generator and receiver ① replicates the clients of the sensing and actuating T-IoT gloves. To prevent synchronization issues that may arise from using different devices, tests were conducted solely on one traffic generator.

Cloud device ② provides computational services and communication infrastructure to IoT devices. It processes sensor data coming from the medical staff's application, converts it into control signals, and then sends the processed data to the patient's application. In this way, it acts as a bridge between the patient and the medical staff.

During the experiments, both applications are initiated simultaneously and connected to the cloud. At this stage, the medical staff's application informs the cloud about the machine learning model it intends to use during the connection. The cloud application loads the registered model, creates necessary buffers for sensor data, and generates objects associated with them.

After the preparations are completed the medical staff's application receives a notification. Subsequently, the medical staff's application starts to send the previously recorded data sampled at a frequency of 50 Hz. This procedure is repeated similarly for 11 different movements for each test subject. Tests for all machine learning models are repeated using the same procedure. The reason for using recorded data in the performance tests of the proposed system is to ensure that the tests for all models are conducted under the same conditions.

For the evaluation of machine learning performance, the data was randomly split into 75% training and 25% testing sets. The same training and testing sets were used across all machine learning models. Hyperparameter tuning was performed using the grid search method. The parameters of the classifiers are given in Table 5.2, and the rest of the parameters are set as default.

Table 5.2 : Model Parameters of The Classifiers.

Classifier	Parameter	Value
LR	Inverse of regularization strength	100
DT	Minimum samples required to split as internal node	8
KNN	Number of neighbors	20
MLP	Activation function for the hidden layer	tanh
RF	The number of trees in the forest	20
XGB	Number of gradient boosted trees	25
	Maximum tree depth for base learners	18

5.3.4 Evaluation criteria

Accuracy analysis has been conducted for the performance of the T-IoT devices and trained models. Criteria such as time characteristics, cloud assessing resource usage, and network bandwidth usage have been used to examine the performance of the cloud computing and communication modules. The definitions for the time characteristics are provided in Section 4.6.3.2, resource usage in Section 4.6.3.3, and network bandwidth usage in Section 4.6.3.4.

5.3.4.1 Accuracy criteria

Accuracy, recall, precision, and F1 score metrics were used to evaluate the performance of the trained models in this study. The following equation was used for calculating accuracy:

$$accuracy(y,\hat{y}) = \frac{1}{N}\sum_{i=0}^{N} 1(\hat{y}_i = y_i), \tag{5.7}$$

where y_i is actual label of the i-th sample, $\hat{y}_i$ is predicted label of the i-th sample, N is sample numbers.

The recall, precision, and F1 score metrics were calculated as follows:

$$P_c = \frac{TP_c}{TP_c + FP_c}, \tag{5.8}$$

$$R_c = \frac{TP_c}{FN_c + TP_c}, \tag{5.9}$$

$$F1_c = \frac{P_c \cdot R_c}{P_c + R_c}, \tag{5.10}$$

where P_c is precision score for class c, R_c is recall score for class c, and $F1_c$ is F1 score for class c. TP_c is true positive prediction of class c, FP_c is false positive prediction of class c, and FN_c is false negative prediction of class c. The final scores for precision, recall, and F1-score, as well as the average performance of all models, were calculated using the macro-average method as follows:

$$MacroScore = \frac{\sum_{c=0}^{M} Score_c}{M}, \tag{5.11}$$

where $Score_c$ is the score of the c-th class and M is the number of the classes.

5.4 Results

In the first two sections, the performance of the sensing T-IoT glove and the actuating T-IoT glove with respect to bending angle is examined. Then, the performance analysis of the sensing T-IoT glove was conducted by examining the accuracy of models trained using different machine-learning methods on the created dataset. Subsequently, the trained models were integrated into the cloud, and their end-to-end operational performance was evaluated. In these tests, parameters such as time performance, resource usage, and network bandwidth usage were examined to analyze the impact of different machine learning models on cloud computing. Additionally, using the FogETex framework, various concurrency, and inter-process communication techniques were tested on the worker device. Furthermore, a multi-worker structure was also tested with this application.

5.4.1 Characterization of sensing T-IoT gloves

In this test, the test subject was asked to open and close their finger at different speeds. The average closing time of the finger was 1.54 seconds, while the opening took 1.88 seconds in average. Including the average waiting times between each cycle, one complete cycle lasted approximately 6.06 seconds.

The bending angle-capacitance graph for the index finger of the Sensing T-IoT Glove is shown in Figure 5.8. While the angle values range from 0.1° to 240.4°, the capacitance values vary between 147.6 pF and 160.1 pF. The closing movement produced a more

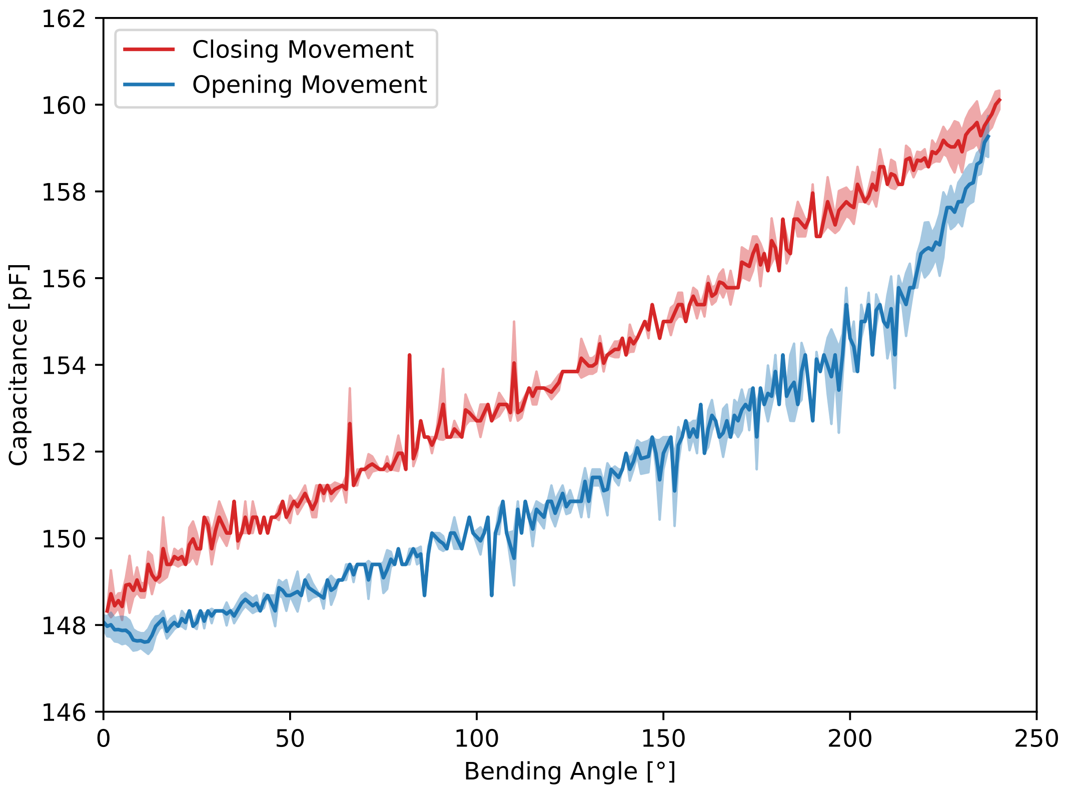


Figure 5.8 : Capacitance Change of Textile-based Sensor During Index Finger Movement.

linear result, whereas the opening movement resulted in a more curved pattern and had lower capacitance values compared to the closing movement. This discrepancy is thought to be due to the recovery behavior of the textile structure.

5.4.2 Characterization of actuating T-IoT gloves

In this test, the test subject holds his/her finger without applying any force, and the flexion bladder is pressurized up to 150 kPa for the flexion movement, while the extension bladder is used during the extension movement. When one bladder is pressurized, the exhaust valve of the other bladder is opened, allowing air to escape into the atmosphere.

The flexion movement took an average of 5.92 seconds, while the extension movement lasted 3.45 seconds. A waiting time of 1 second was applied between movements, resulting in an average cycle duration of 11.37 seconds. The actuator was operated slower than the sensing glove to minimize the risk of potential harm to the patient whereas the system is capable of operating at a faster rate.

Figure 5.9 shows the angle change of the index finger in response to pressure applied by the Actuating T-IoT Glove. The shared pressure data pertains to the flexion bladder, which is directly related to the flexion movement. The results show that the flexion movement exhibited a more linear response to the applied pressure, while the extension movement responded more slowly to pressure reduction. The pressure values ranged from 102.8 kPa (approximately atmospheric pressure) to 150.0 kPa. Correspondingly, the angle values varied between 32.8° and 185.0°. It was observed that the finger's bending and extension movements had a slightly smaller range compared to a healthy person. This limitation is expected, as the Actuating T-IoT Glove contains exoskeletal actuators, which naturally restrict movement to some extent.

5.4.3 Results on accuracy

For the control of the actuating T-IoT glove, data was collected from 12 test subjects using the sensing T-IoT glove, with approximately 161 minutes of data for each finger. During data collection, the test subjects labeled the data as “Opening”, “Open”, “Closing”, and “Close”. The dataset collected from test subjects is divided into training

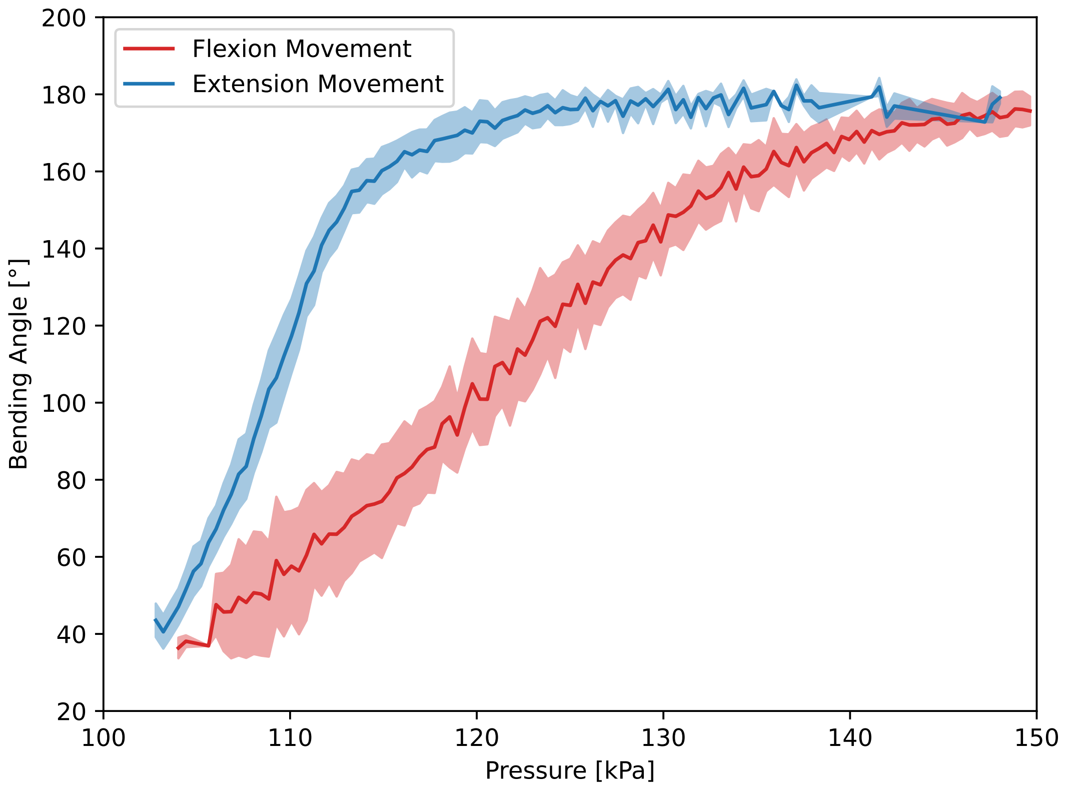


Figure 5.9 : Change of Index Finger Bending Angle During Flexion and Extension Movement of Textile-based Actuator.

(75%) and test (25%) sets. Separate models were trained for each finger using different machine learning methods. Table 5.3 presents the results of different machine learning methods trained for each finger based on the F1 score metric. Table 5.4 shows the average scores of the machine learning methods for all fingers.

When the scores of models trained with different machine learning methods were examined, the Random Forest (RF) method emerged as the most successful across all fingers. The RF models achieved a performance of 91.53 on the thumb and above 94.50 on the other fingers. The lower performance on the thumb is thought to be due to the thumb traveling a shorter distance than other fingers while closing, those moving more quickly, especially in actions involving multiple fingers moving simultaneously.

When compared with other methods based on F1 score, the XGBoost (XGB) and Decision Tree (DT) models achieved performances close to that of the RF models, with differences of 0.10% for XGB and 0.50% for DT. K-Nearest Neighbors (KNN) showed a 1.77% lower performance, Multi-layer Perceptron (MLP) showed a 4.45% lower performance, and Logistic Regression (LR) showed a 7.32% lower performance.

Table 5.3 : F1 Scores of Individual Finger Models

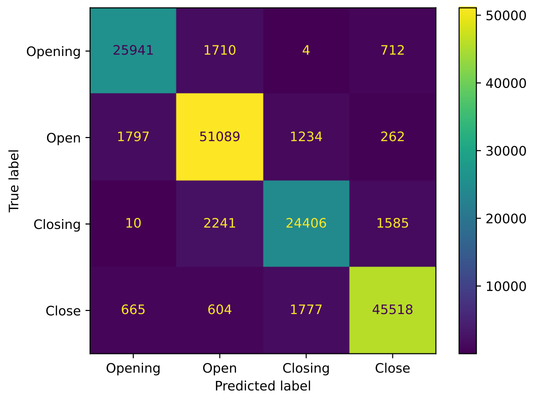
Classifier	Thumb	Index	Middle	Ring	Pinkie
LR	79.05	87.97	89.45	88.66	88.01
DT	90.90	94.13	94.06	94.09	94.08
KNN	88.38	92.43	92.57	92.56	92.48
MLP	83.63	90.67	91.50	91.02	90.69
RF	**91.53**	**94.56**	**94.51**	**94.52**	**94.64**
XGB	91.36	94.39	94.46	94.43	94.60

Table 5.4 : Average Performance of All Models

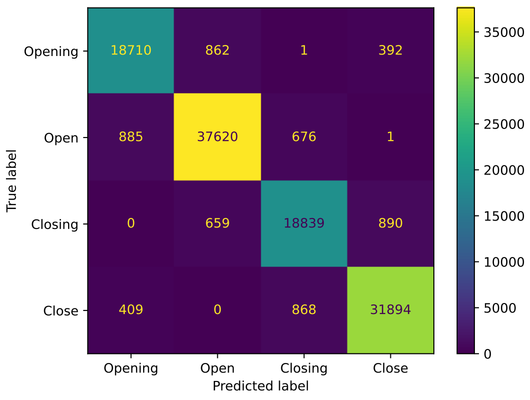
Classifier	Accuracy	Recall	Precision	F1Score
LR	87.32	85.89	87.65	86.63
DT	93.93	93.49	93.42	93.45
KNN	92.24	91.86	91.52	91.68
MLP	90.01	89.62	89.43	89.50
RF	**94.40**	**93.93**	**93.98**	**93.95**
XGB	94.30	93.85	93.85	93.85

It was observed that tree-based models are more successful in solving the problem with the collected dataset.

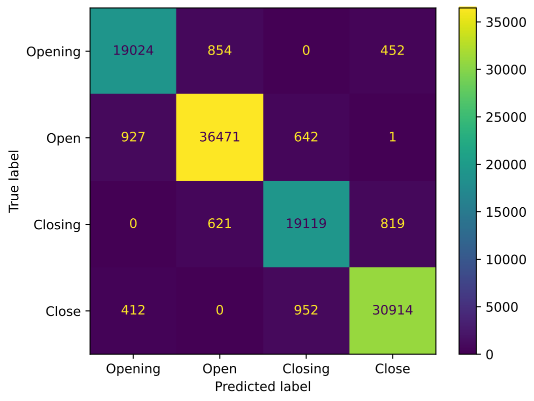
Figure 5.10 shows the confusion matrices for the most successful model results, which belong to the RF models. Figures 5.10(a)-(e) present the confusion matrices for models trained separately for each finger. Figure 5.10(f) provides the combined results of models trained for each finger using the RF method. The confusion matrices for models trained using other classifiers are provided in Figures A.1-A.5 (Appendix B). When the confusion matrices are examined, it is evident that the majority of the data, as expected, lies on the diagonal of the matrices. The mispredictions are generally concentrated in states that transition into each other. In the dataset, the states progress cyclically as "Open", "Closing", "Close", "Opening", and "Open". Given that transitions between states posed similar challenges during human labeling, it is understandable that the models also struggle with these predictions. Misclassifications are minimal for labels with distinct states in between, such as "Closing" to "Opening" or "Open" to "Close".



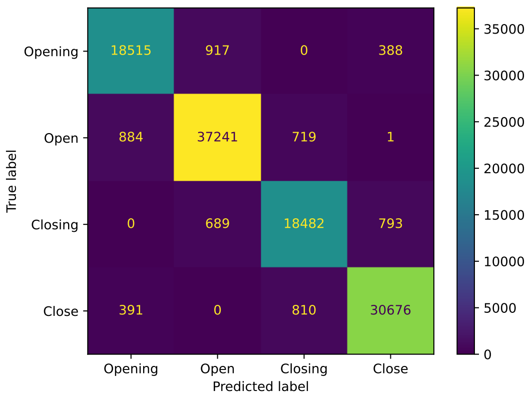
(a) Thumb Finger.



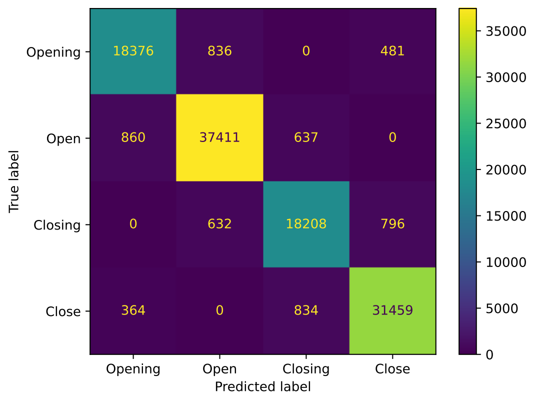
(b) Index Finger.



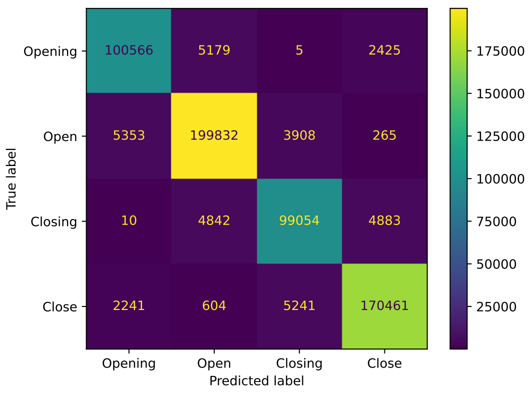
(c) Middle Finger.



(d) Ring Finger.



(e) Pinkie Finger.



(f) Overall Results.

Figure 5.10 : Confusion Matrices of Random Forest Classifier for Different Fingers.

5.4.4 Results on time performance

The developed cloud system provides real-time computation and data communication between medical staff and their patients. Therefore, the timing performance of the system is of significant importance. To measure the system's timing performance, the following metrics were used: arbitration time, latency, queuing delay, execution time, jitter, and total response time.

Figure 5.11 shows the preparation time of the system for different classifiers when a medical staff initiates a therapy request. The models for the classifiers are trained and saved to the system's computing module when it is first started. When a new medical staff starts a session, the saved model is loaded, and the necessary preliminary preparations for signal processing and feature extraction are made. Once these preparations are completed, the system notifies the medical staff and the patient that it is ready through the client program.

In the system, except for the type of trained model, all procedures such as signal processing and feature extraction are the same for all models. Therefore, classifiers with larger model sizes have longer arbitration times due to the length of file reading

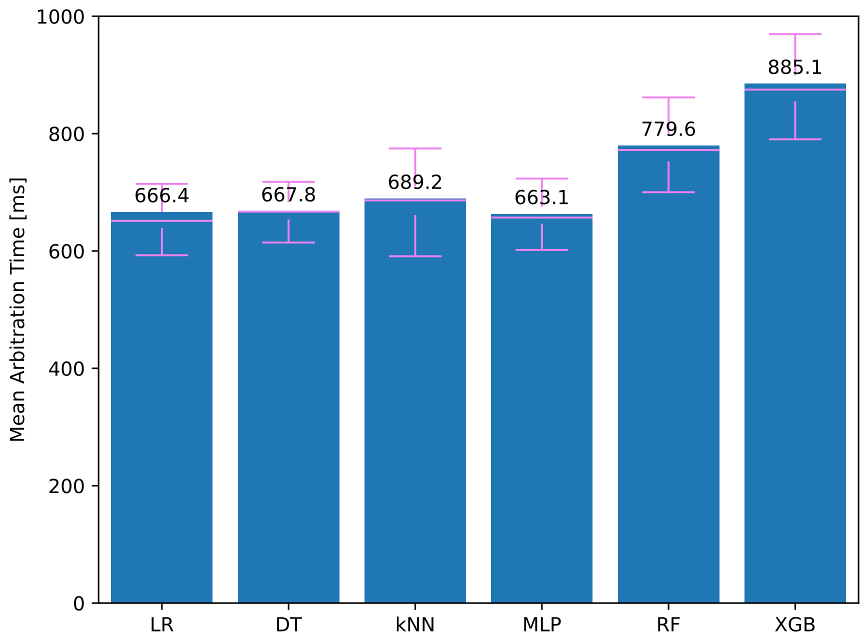


Figure 5.11 : Arbitration time for different classifiers.

operations. The lowest average arbitration time was observed in tests using MLP at 613.1 ms, while the highest was observed using XGB at 885.1 ms. Models with multiple tree structures also had high arbitration times. Additionally, the system prepared with kNN showed high arbitration times because, instead of using a mathematical equation, the entire training set is loaded, and predictions are made based on the proximity to the training elements. When examining the results, even the longest time is under one second, so the arbitration times for all models are quite low. When a medical staff connects to the system, he/she can start receiving computation services from their dedicated system within one second.

Figure 5.12 shows the latency scores for therapies using different classifiers. The lowest average latencies are from KNN and LR, with scores of 23.0 ms and 23.3 ms, respectively. The other classifiers have latencies above 23.5 ms. Although the other models have slightly worse results, they can still send their data to the computation module with a delay of up to 25 ms.

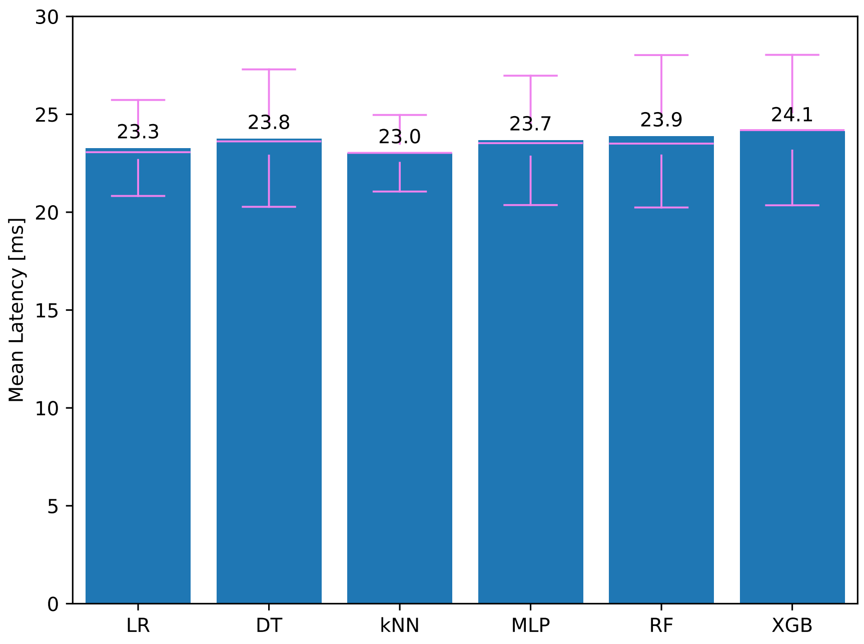


Figure 5.12 : Latency for different classifiers.

Figure 5.13 presents the average queuing delays for therapies using different classifiers. All models have very low and similar queuing delays of 0.2 ms, starting the data processing promptly.

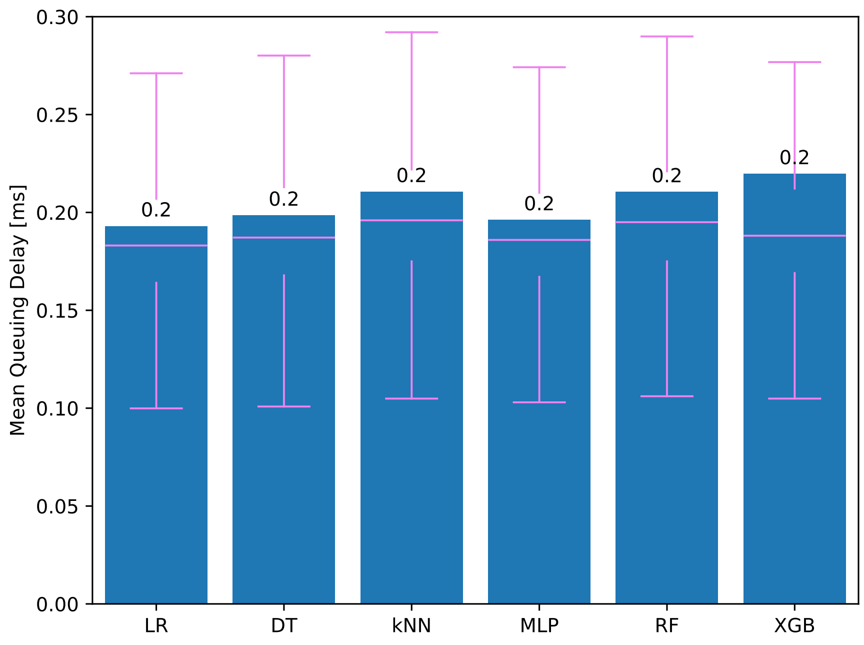


Figure 5.13 : Queuing delay for different classifiers.

Figure 5.14 shows the time taken to process incoming sensor data and make state predictions for each finger. The lower the execution time, the more therapies the system can handle simultaneously. Therefore, execution time is one of the most important metrics for the system. According to the results, LR and DT therapies have the lowest execution time with scores of 0.9 and 1.0 ms, respectively.

Figure 5.15 displays the average response times for therapies using different classifiers. The best scores are similar to previous results, with LR and DT models achieving 47.5 ms and 48.4 ms, respectively. The highest response times are observed with kNN, RF, and XGB. Overall, all models have an average delay of around 50 ms, which means they can provide real-time services. However, for faster response times, it is recommended to use models trained with LR and DT methods.

Figure 5.16 illustrates the jitter, or the variation in response time, for models trained with different classifiers. The lowest jitter is observed in models trained with LR at 2.0 ms, followed by DT and KNN at 2.3 ms. The highest jitter is seen in models trained with RF and XGB at 3.0 ms. High jitter can negatively affect synchronization between commands, potentially causing therapeutic movements to be shorter or longer

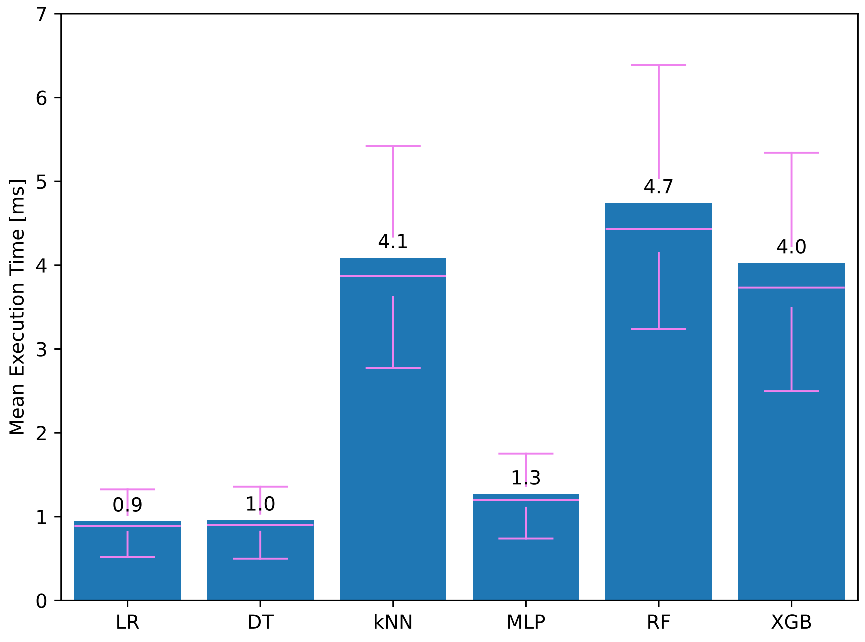


Figure 5.14 : Execution time for different classifiers.

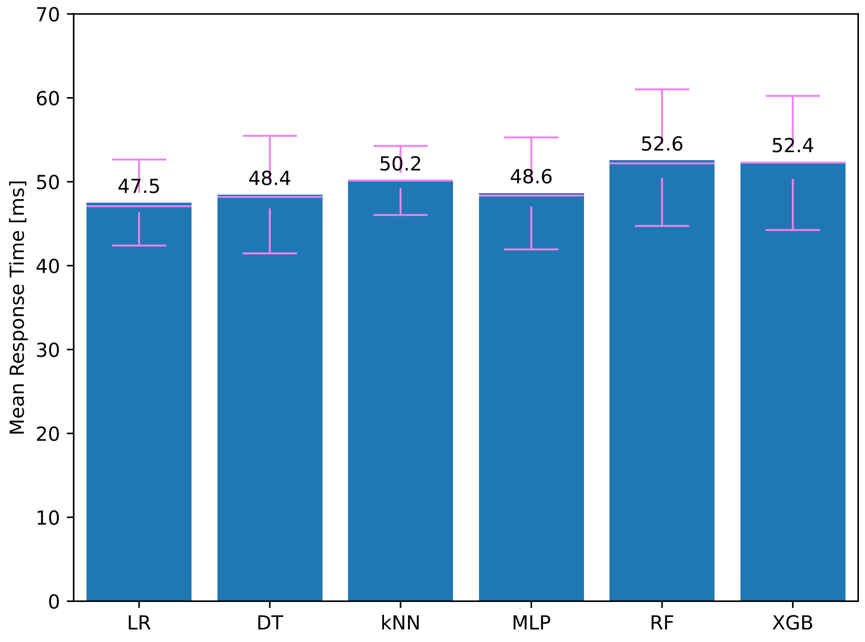


Figure 5.15 : Total response time for different classifiers.

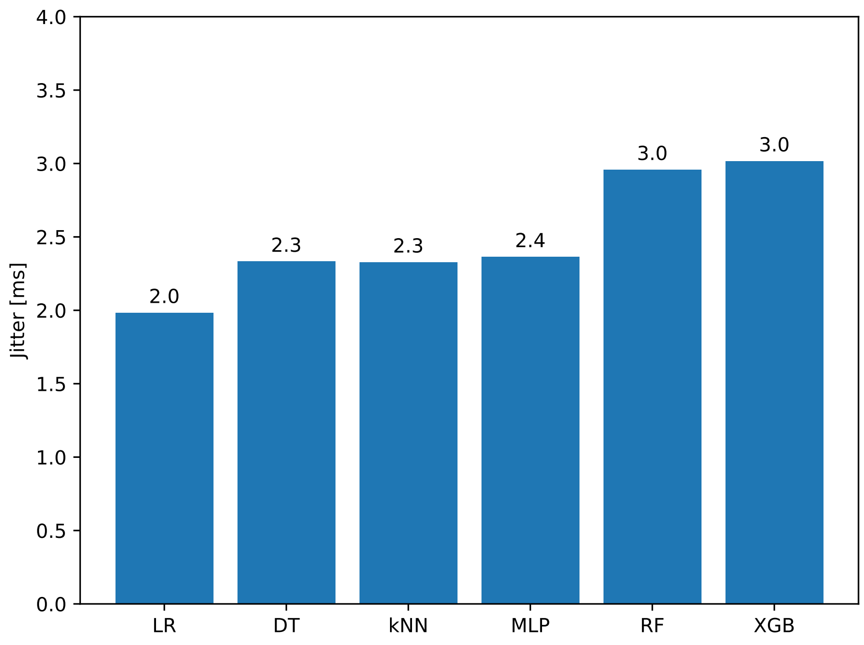


Figure 5.16 : Jitter for different classifiers.

than intended. However, since the obtained jitter values are relatively short compared to hand movements, they can be considered negligible.

5.4.5 Results on resource usage

In computational systems, resource utilization directly affects various time metrics such as response time, queuing delay, and execution time. The higher the resource usage, the longer the system will take to respond to new processes. Additionally, lower CPU consumption indicates that the system can serve more users and also results in lower cloud costs.

Figure 5.17 shows the resource consumption during therapy for models developed with different classifiers. The lowest CPU usage is observed with LR and DT models at 6.9% and 7.0%, respectively. For memory usage, the lowest values for LR and MLP are both 8.4%, while DT has a slightly higher minimum at 8.8%. The highest CPU usage is 20.8% and the highest memory usage is 14.2%, both for XGB. Given the lower CPU and RAM usage, models trained with LR, DT, and MLP are recommended as they can serve more users and are more cost-effective.

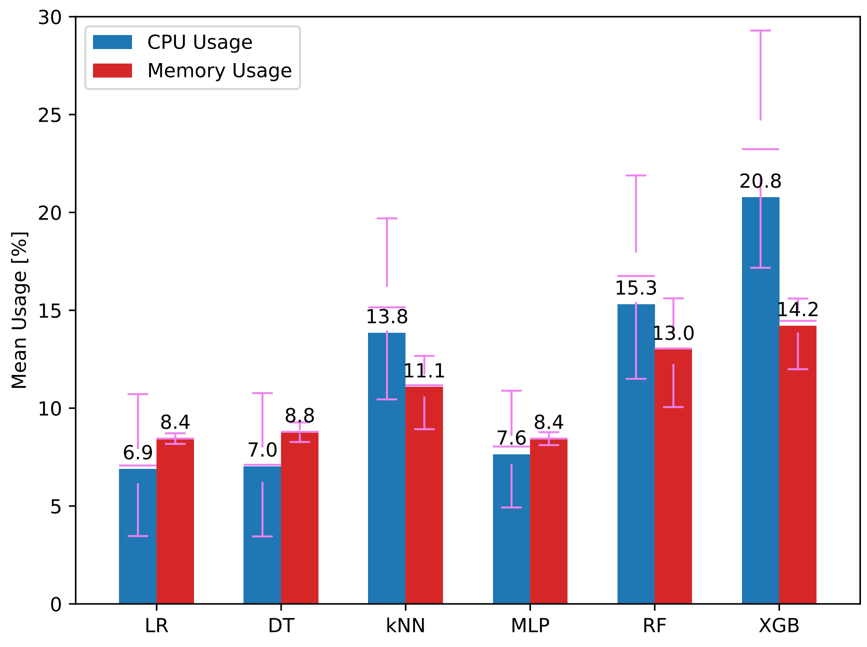


Figure 5.17 : CPU and memory usage for different classifiers.

5.4.6 Results on network bandwidth usage

Network bandwidth usage affects network traffic and can negatively impact network latency. However, in this study, since the incoming and outgoing data are the same for each model, similar network bandwidth usage is expected. Figure 5.18 shows the network bandwidth usage during therapy for models trained with different classifiers. The lowest value is 138.9 kbps for models trained with XGB, while the highest value is 140.7 kbps for models trained with RF. The difference between the lowest and highest values is observed to be 1.30%, with similar values across all models. Considering that cloud system network infrastructures operate at Gbps levels, the network bandwidth usage obtained is quite low.

5.4.7 Results on concurrency control techniques

In this section, the concurrency control techniques explained in Section 3.4 were utilized to enhance the performance of the FogETex framework. The studies in this section were conducted using the mock client in the Wi-Fi Test Bed provided in Section 4.6.1. Each result was obtained through six repetitions, each containing

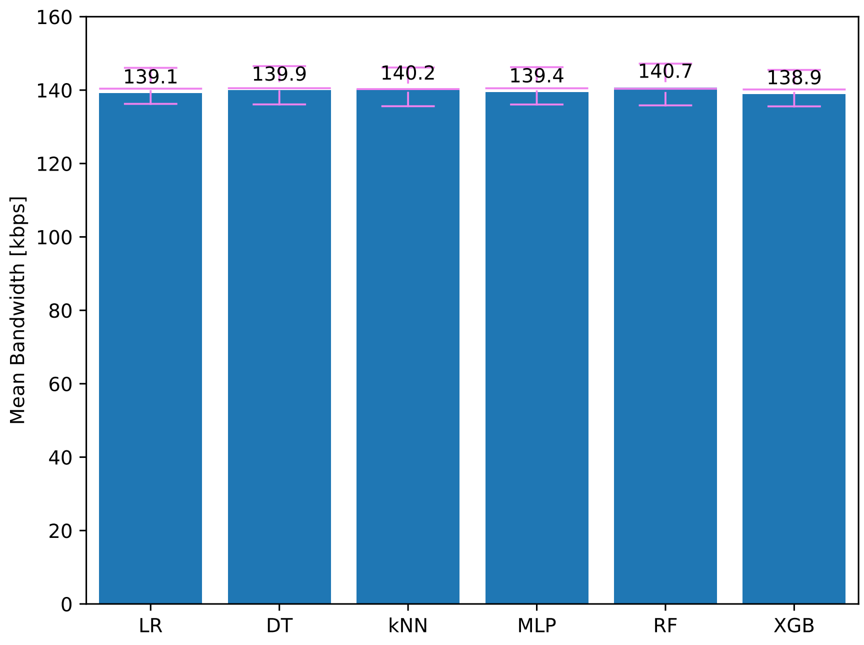


Figure 5.18 : Network bandwidth for different classifiers.

identical data for a total duration of 3 minutes. The model used in the tests was developed with the decision tree method, which demonstrated enhanced accuracy, resource usage, execution time, and response time compared to others. The limit of how many users the devices can serve is defined as the maximum number of users that maintain an average latency of 50 ms or less for worker devices and 80 ms or less for cloud devices. Beyond these average response time values, the devices fail to provide real-time service, and it has been observed that the response time continuously increases with successive requests.

Figure 5.19 presents the mean response time of the worker device using the multi-threaded data processing method. The WebSocket IPC method was employed as the IPC mechanism. In this test, stress tests were conducted with up to 6 threads to determine the optimal number of threads. The number 6 was chosen because the processor of the worker device has 4 cores, and using a number slightly above the core count allows for more comprehensive stress testing. The configuration with a single thread also served as the single-threaded data processing method.

The tests showed that the single-thread configuration delivered the highest performance and could serve up to 5 users. While other configurations could serve

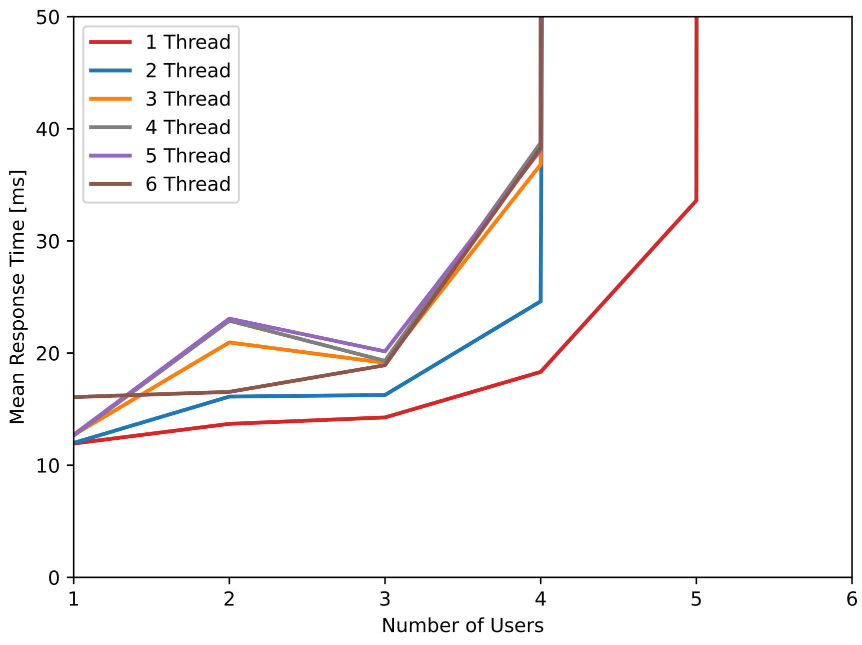


Figure 5.19 : Stress test results of multi-threaded data processing via WebSocket IPC.

up to 4 users, the 2-thread configuration demonstrated the lowest response time. However, the multi-threaded structure performed worse due to the Global Interpreter Lock (GIL) mechanism in Python. Since other processor cores were not actively utilized, increasing the number of threads limited performance.

Figure 5.20 presents the mean response time of the worker device using the multi-process data processing method. Similar to the multi-threaded data processing method, the WebSocket IPC method was employed in this test. Here, different configurations with up to 6 processes were evaluated, as the worker device's 4-core processor guided the choice of a slightly higher count. Since the single-process configuration uses the same method as the single-threaded configuration from the previous test, the same result was used.

While the single-process configuration could serve up to 5 users, the multi-process configurations were capable of serving up to 6 users. Although the results were fairly similar, the 2-process and 3-process configurations performed worse compared to the 4-process and 5-process configurations. When the number of processes was increased to 6, performance slightly decreased. The 4-process setup demonstrated marginally better results with a lower response time than 5-process communication. This indicates

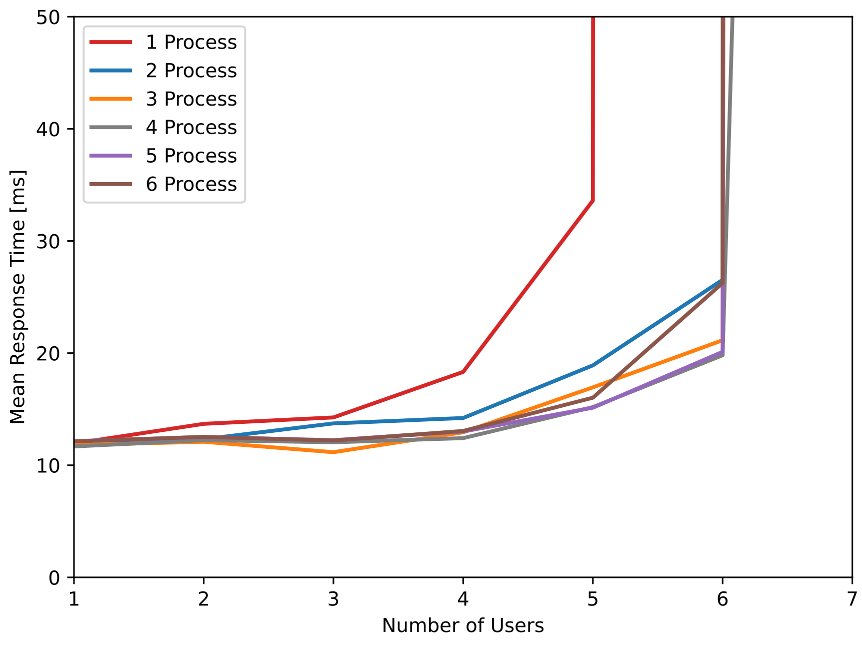


Figure 5.20 : Stress test results of multi-process data processing via WebSocket IPC.

that the 4-process configuration effectively utilized all the cores of the worker device's 4-core processor, leading to improved performance.

Figure 5.21 shows the mean response time of the stress test conducted using a RESTful API for inter-process communication between the socket server and the computation module. Since the multi-process concurrency method demonstrated better performance than the multi-threaded method, this test was conducted using multi-process operations. Each computation process included its own RESTful server. In the tests conducted with up to 6 users, it was observed that the single-process configuration could serve up to 3 users, while the other configurations were capable of serving up to 4 users. Among these, the 4-process configuration exhibited the best performance compared to the others.

Figure 5.22 presents the results of the multi-process stress test conducted using FIFO, another IPC method. In this test, an input and an output FIFO were created for each process. The test results indicate that the 2-process configuration outperformed the single-process configuration. However, increasing the number of processes beyond 2 negatively impacted performance. The single-process configuration was able to serve up to 8 users, while the 2-process configuration could serve up to 10 users.

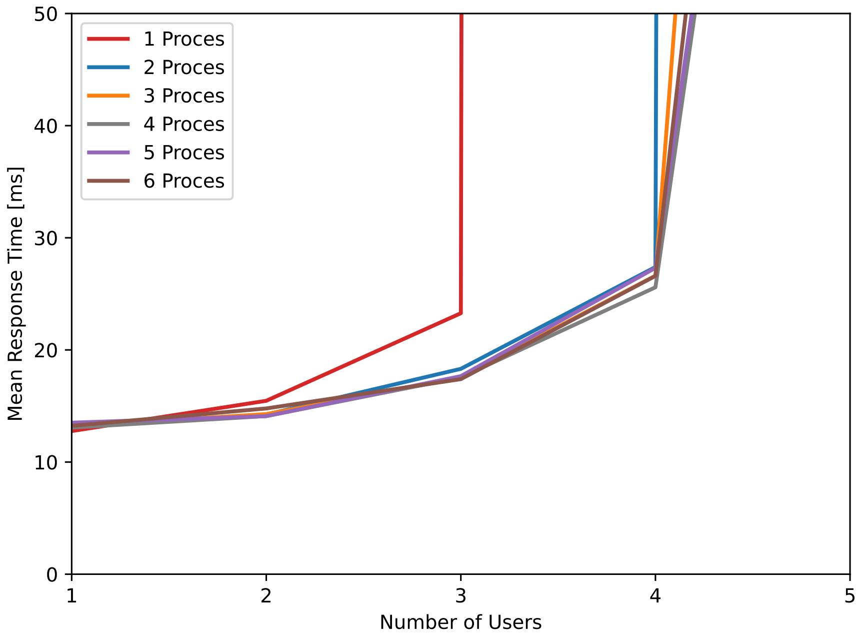


Figure 5.21 : Stress test results of multi-process data processing via RESTful IPC.

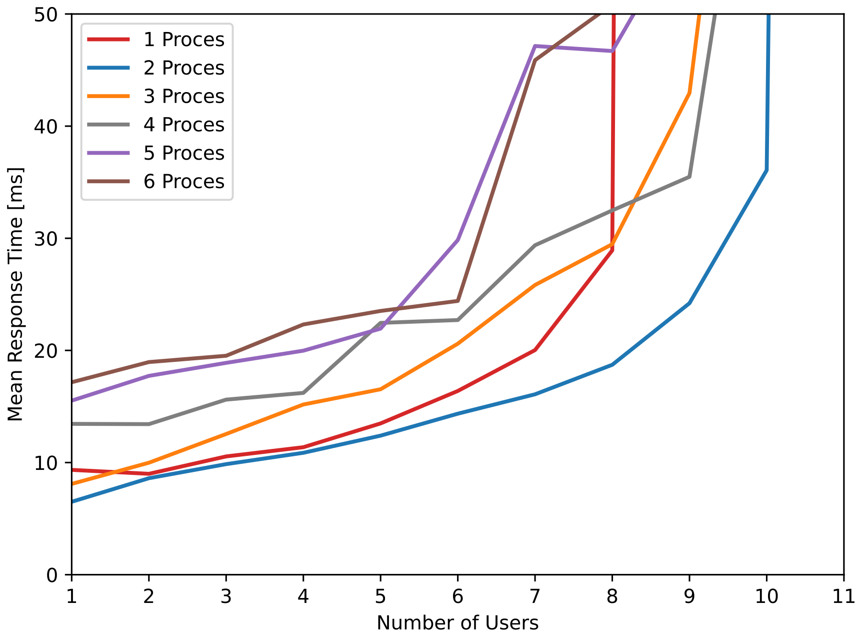


Figure 5.22 : Stress test results of multi-process data processing via FIFO IPC.

Figure 5.23 compares the best configurations of the different concurrency and IPC methods proposed for the FogETex framework. A 2-thread configuration was selected for the multi-thread concurrency method. For the multi-process concurrency method, 4 processes were chosen for both WebSocket and RESTful methods, while 2 processes were selected for the FIFO method. The test results indicate that the RESTful and multi-thread methods can serve up to 4 users. The single-thread method can handle up to 5 users, and the multi-process method can support up to 6 users. Among all, the FIFO IPC method demonstrated the highest capacity, serving up to 10 users.

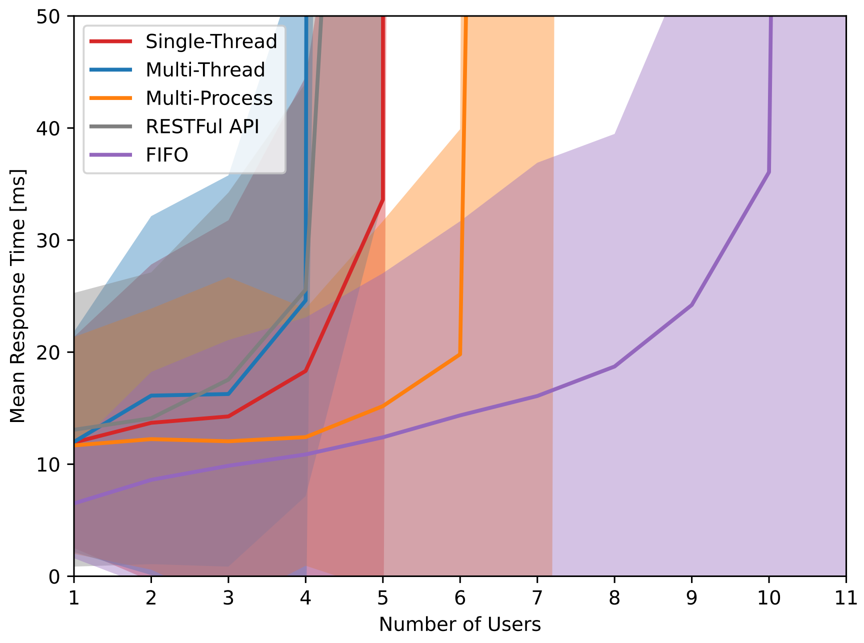


Figure 5.23 : Performance comparison of various concurrency control techniques.

Figure 5.24 presents the stress test results for a fog node with multiple workers. In this test, the 2-process configuration using the FIFO IPC method, which demonstrated the best performance in the previous test, was employed. The results show that, based on mean response time, a fog node with 1 worker can serve up to 10 users, 2 workers can handle up to 22 users, and a system with 3 workers can serve up to 26 users. It was observed that the cloud system could serve up to 23 users. Therefore, to build a system with performance comparable to the cloud, it was determined that at least a 3-worker fog node is required.

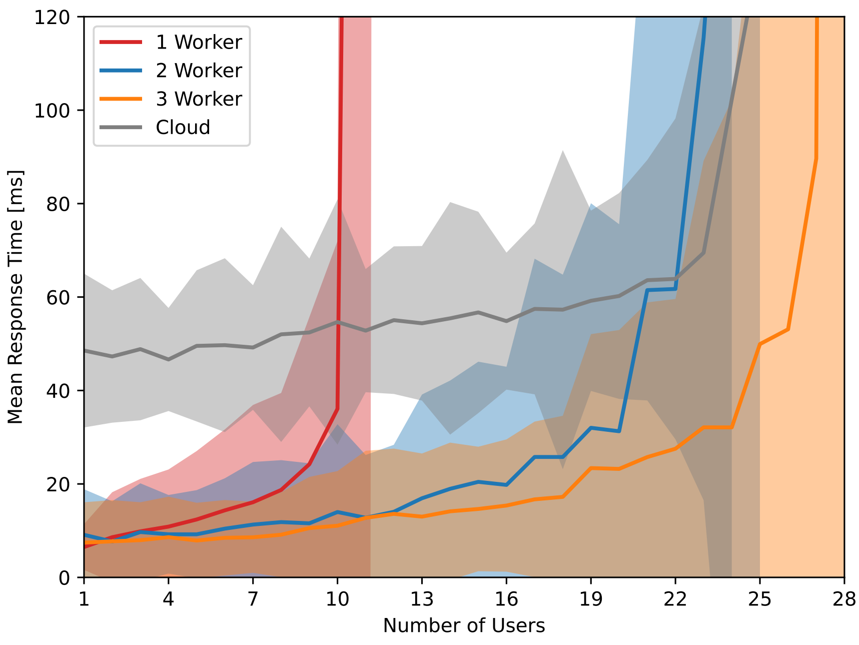


Figure 5.24 : Performance comparison of multi-worker and cloud.

5.4.8 Discussion

When examining the results of the developed system and trained models, RF is observed to be the most successful in terms of accuracy. However, it performs poorly in metrics such as arbitration time, latency, execution time, total response time, jitter, and CPU and Memory usage. Models trained with LR and DT methods have shown significantly better results in these metrics compared to RF, particularly in CPU usage, where they consume about half as many resources. It is estimated that a system using these methods could potentially serve approximately twice as many users as one using RF.

However, when examining the test results in terms of accuracy, there is a 7.32% difference in F1 score between models trained with LR and those trained with RF. This significant difference in prediction performance makes LR-trained models less preferable compared to RF-trained models. On the other hand, models trained with DT show only a 0.50% difference in F1 score and overall scores compared to RF models. Thus, by sacrificing just 0.50% in accuracy, the system can achieve lower response times and reduced resource consumption, which allows for serving more users and lowering resource costs as a trade-off.

Upon examining the accuracy scores of the proposed system, an F1 score of 93.95 was observed. The reasons for mispredictions in the system include the inability of machine learning methods to fully distinguish transitions between states, variations in the anthropometric hand data of test subjects, and hand tremors during movements. The lower accuracy score for the thumb, compared to other fingers, is attributed to its shorter movement distance, leading to less variation in sensor data.

In the system's use for therapy, it is anticipated that medical staff will observe the patient's hand movements in real-time through a third-party video conferencing system. Since the proposed system is not intended for use in critical scenarios such as surgery, it can be argued that the system is more tolerant of trade-offs between latency and accuracy.

5.5 Conclusion

This study has demonstrated the efficacy of a telerehabilitation strategy by utilizing cloud computing-based IoT devices such as textile-based sensing and actuating gloves for patients with hand impairments. The system developed is introduced into the field of remote healthcare to complement conventional therapy, aiming to overcome location hindrances, thereby enabling continuous improvement regardless of location.

The application of advanced machine learning algorithms for interpreting real-time hand movements, captured by the sensing T-IoT glove and processed via cloud technology, enables the control of the actuating T-IoT glove. The proposed system performed an average response time of 48.4 milliseconds and an average accuracy of 93.45%.

This study demonstrates that the FogETex framework has been successful in multi-sensor applications. Additionally, as each developed model is integrated into the system as a separate application, the framework has shown the capability to host and run multiple applications simultaneously. Although the FogETex framework was primarily designed for fog computing, its cross-platform support and the ability of all nodes to provide services enable it to operate using only cloud resources, as exemplified in these applications.

On the other hand, the fog system was tested with different concurrency and IPC methods for this application. The multi-process concurrency method, combined with the FIFO IPC technique, demonstrated the best performance. It was observed that the system could serve up to 10 users with 1 worker, 22 users with 2 workers, 26 users with 3 workers, and 23 users in the cloud configuration.

Since our aim was to eliminate injury risks that might occur during conventional robotic therapy applications, due to the lightweight, soft attributes of textile materials, the system provides compliant and safe interactions. We believe that our approach will enhance the outcomes of hand therapy aimed at restoring neuromuscular function, thereby increasing the quality of life for people. Moreover, the system architecture developed for cloud computing can be implemented not only in IoT systems for rehabilitation but also in other applications such as remote medical surgeries or virtual/augmented reality applications.

6. CONCLUSION

In this thesis, a fog computing-based framework for e-textile applications, named FogETex, is proposed. The FogETex framework is designed to meet the computational requirements of low-power microcontrollers used in e-textile products, addressing concerns related to comfort. The framework is tailored to meet the needs of e-textile applications, enabling both indoor and outdoor operations. This allows T-IoT devices to operate without mobility restrictions. A WebSocket connection is used for bidirectional communication between the worker and user, enabling the transfer of tens of data points per second without the need for a new connection. Single requests, such as device allocation, are managed via HTTP RESTful API connections. The framework has been developed using NodeJS for communication operations and Python for data preprocessing and machine learning services. Additionally, the developed user interface allows system administrators to monitor devices in real-time.

Although this thesis is primarily based on a fog computing architecture, many e-textile applications have been developed to enhance the architecture and analyze the requirements of e-textile applications. The primary studies include the gait phase recognition system and the hand motion recognition system. Additionally, the signals processed by the FogETex framework have been integrated with textile-based soft robotic systems to provide benefits to humans. For instance, exoskeleton gloves were developed for individuals with muscle weakness. Lastly, a review study identified the security and privacy requirements for e-textile applications, providing a roadmap for the development of the framework. All these studies contributed to the creation of features within the modules of the FogETex system, while also being unified under the FogETex framework as an umbrella.

The applications developed were selected based on their suitability for testing the system's characteristics. The first case scenario was the gait phase recognition system, which was used to test the system with a single sensor. This system, equipped with a deep learning-based machine learning model, challenged the FogETex system in terms of computation. Meanwhile, the system's time characteristics, resource usage, and network bandwidth usage metrics were measured across different experimental scenarios and devices. The FogETex framework was tested in an ideal environment with a mock client, and under real-life conditions with an actual client. To cover both indoor and outdoor applications, tests were conducted on Wi-Fi with LAN and LTE with WAN. In Wi-Fi tests, results for worker, broker, and cloud devices were analyzed, whereas in LTE tests, only worker and cloud devices were examined. The broker device was not separately analyzed in LTE tests, as it acted as a proxy to transfer data from the user to the worker. The tests showed that in Wi-Fi tests, the worker device with a mock client achieved a latency of 10.5 ms, while with an actual client, it showed a latency of 22.3 ms. In LTE tests, the worker device had an average response time of 54.8 ms with the actual client and 88.0 ms with another actual client. As expected, the worker device performed better in its category.

In the gait phase recognition system's stress test, it was observed that the worker device could serve up to 6 users in both LTE and Wi-Fi testbeds. While the broker could serve up to 18 users and the cloud up to 14 users, the worker device was found to be more advantageous in multi-worker scenarios and in terms of price-performance comparison. When comparing the FogETex system to other works in the literature, it was noted that the only similar system is HealthFog. In the same case scenario, the FogETex system outperformed in latency, execution time, response time, and working frequency. Thus, the FogETex system successfully passed single-sensor and multi-user tests, showing its advantages over competitors.

In another case scenario, the assistive soft robotic glove control system was used. In this scenario, the FogETex system was tested with multi-sensor inputs. Since the developed application was cloud computing-based, it was observed that the framework worked successfully across different computing devices. Each machine learning model developed was integrated as a separate application into the system, allowing the framework to run different applications within the same infrastructure. Additionally, for the first time, soft robotics and e-textile applications were combined under the cloud computing framework. Furthermore, it was observed that the FogETex framework could integrate sensing and actuating devices from different locations. In the tests conducted, the average response time was observed to be 48.4 ms, while the average accuracy was calculated to be 93.45%. On the other hand, the fog system was tested with the application using different concurrency and IPC methods. The multi-process concurrency method, combined with the FIFO IPC technique, demonstrated the best performance. It was observed that the system could serve up to 10 users with 1 worker, up to 22 users with 2 workers, up to 26 users with 3 workers, and up to 23 users with the cloud configuration.

The developed test cases have shown that the FogETex framework performs in a real-time and robust manner across different scenarios, demonstrating its effectiveness for e-textile applications. Furthermore, by providing the necessary computational services for e-textile products, it has the potential to integrate all e-textile products under one umbrella. Although the system was primarily developed for e-textile applications, being designed as a PaaS model means there is no barrier to its use in other IoT applications.

The future works of this thesis include the integration of various sensor applications into the proposed framework. In addition to textile-based solutions, the FogETex framework can be utilized to enhance the performance of non-textile sensor and actuator systems. Addressing one of the primary challenges in e-textile applications—connection issues arising from transitions between rigid and flexible structures—could further strengthen the reliability of the framework.

Furthermore, enhancements to the FogETex framework can be explored to address critical battery conditions or instances of incomplete sensor data. In such scenarios, T-IoT devices could switch off their Bluetooth connections to conserve energy and store data in their onboard memory. Additionally, applications involving sensors operating at different frequencies can be developed, as not all sensors generate data at the same rate. This flexibility would broaden the scope of the framework's applicability.

The current study validated the FogETex framework through system-level testing. Future work should include real-world user testing to assess potential latency or performance discrepancies across systems. For instance, the gait phase recognition system could be integrated with foot-drop actuators, enabling it to function as a controller. Clinical trials involving patients are planned for this integration, particularly in conjunction with the assistive soft robotic glove project. These trials aim to evaluate the system's effectiveness in rehabilitation and assistive scenarios.

REFERENCES

[1] **Du, K.**, **Lin, R.**, **Yin, L.**, **Ho, J.S.**, **Wang, J. and Lim, C.T.** (2022). Electronic textiles for energy, sensing, and communication, *iScience*, *25*(5), 104174.

[2] **Sanchez, V.**, **Walsh, C.J. and Wood, R.J.** (2021). Textile Technology for Soft Robotic and Autonomous Garments, *Advanced Functional Materials*, *31*(6), 2008278.

[3] **Zaman, S.u.**, **Tao, X.**, **Cochrane, C. and Koncar, V.** (2022). Smart E-Textile Systems: A Review for Healthcare Applications, *Electronics*, *11*(1), 99.

[4] **Cherston, J. and Paradiso, J.A.** (2019). SpaceSkin: development of aerospace-grade electronic textile for simultaneous protection and high velocity impact characterization, ***J.P. Lynch**, **H. Huang**, **H. Sohn and K.W. Wang**, editors, Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems 2019*, volume10970, International Society for Optics and Photonics, SPIE, p.109700J.

[5] **Buechley, L. and Eisenberg, M.** (2009). Fabric PCBs, electronic sequins, and socket buttons: techniques for e-textile craft, *Personal and Ubiquitous Computing*, *13*, 133–150.

[6] **Miyada, D. and Jing, L.** (2021). Detection of Hand Strength Distribution with E-Textile-Based Tactile Glove for Peach Harvesting, ***C. Stephanidis**, **M. Antona and S. Ntoa**, editors, HCI International 2021 - Posters*, Springer International Publishing, Cham, pp.366–372.

[7] **Yilmaz, A.F.**, **Ozlem, K.**, **Celebi, M.F.**, **Taherkhani, B.**, **Kalaoglu, F.**, **Atalay, A.T.**, **Ince, G. and Atalay, O.** (2024). Design and Scalable Fast Fabrication of Biaxial Fabric Pouch Motors for Soft Robotic Artificial Muscle Applications, *Advanced Intelligent Systems*, *6*(8), 2300888.

[8] **Elmoughni, H.M.**, **Yilmaz, A.F.**, **Ozlem, K.**, **Khalilbayli, F.**, **Cappello, L.**, **Tuncay Atalay, A.**, **Ince, G. and Atalay, O.** (2021). Machine-Knitted Seamless Pneumatic Actuators for Soft Robotics: Design, Fabrication, and Characterization, *Actuators*, *10*(5), 94.

[9] **Payra, S.**, **Wicaksono, I.**, **Cherston, J.**, **Honnet, C.**, **Sumini, V. and Paradiso, J.A.** (2021). Feeling Through Spacesuits: Application of Space-Resilient E-Textiles to Enable Haptic Feedback on Pressurized Extravehicular Suits, *2021 IEEE Aerospace Conference (50100)*, pp.1–12.

[10] **Hartman, K.**, **Westecott, E.**, **Colpitts-Campbell, I.**, **Robinson Faber, J.**, **Shao, Y.**, **Luginbuhl, C.**, **Prior, O. and Laroia, M.** (2021). Textile Game Controllers: Exploring Affordances of E-Textile Techniques as Applied to Alternative Game Controllers, *Proceedings of the Fifteenth International Conference on Tangible, Embedded, and Embodied Interaction*, TEI '21, Association for Computing Machinery, pp.1–14.

[11] **Gumus, C.**, **Ozlem, K.**, **Khalilbayli, F.**, **Erzurumluoglu, O.F.**, **Ince, G.**, **Atalay, O. and Atalay, A.T.** (2022). Textile-based pressure sensor arrays: A novel scalable manufacturing technique, *Micro and Nano Engineering*, *15*, 100140.

[12] **Mauriello, M.**, **Gubbels, M. and Froehlich, J.E.** (2014). Social Fabric Fitness: The Design and Evaluation of Wearable E-Textile Displays to Support Group Running, *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, Association for Computing Machinery, pp.2833–2842.

[13] **Glanc-Gostkiewicz, M. and Harris, N.** (2017). A Textile Based Polypyrrole Chloride Sensor for Agricultural Use, *Proceedings*, *1*(4), 430.

[14] **Kara, E. and Cagiltay, K.** (2023). Using E-textiles to Design and Develop Educational Games for Preschool-aged Children, *Educational Technology & Society*, *26*(2), pp. 19–35.

[15] **Anne Schwarz, Lieva Van Langenhove, P.G. and Deguillemont, D.** (2010). A roadmap on smart textiles, *Textile Progress*, *42*(2), 99–180.

[16] **Elmoughni, H.M.**, **Atalay, O.**, **Ozlem, K. and Menon, A.K.** (2022). Thermoelectric Clothing for Body Heat Harvesting and Personal Cooling: Design and Fabrication of a Textile-Integrated Flexible and Vertical Device, *Energy Technology*, *10*(10), 2200528.

[17] **Li, X. and Sun, Y.** (2017). WearETE: A Scalable Wearable E-Textile Triboelectric Energy Harvesting System for Human Motion Scavenging, *Sensors*, *17*(11), 2649.

[18] **Hossain, I.Z.**, **Khan, A. and Hossain, G.** (2022). A Piezoelectric Smart Textile for Energy Harvesting and Wearable Self-Powered Sensors, *Energies*, *15*(15), 5541.

[19] **Ali, I.**, **Dulal, M.**, **Karim, N. and Afroj, S.** (2024). 2D Material-Based Wearable Energy Harvesting Textiles: A Review, *Small Structures*, *5*(1), 2300282.

[20] **Gupta, P.**, **Saini, D.K.**, **Rawat, P. and Zia, K.** (2023). *Bio-Inspired Optimization in Fog and Edge Computing Environments: Principles, Algorithms, and Systems*, CRC Press.

[21] **Chen, C.**, **Zhang, P.**, **Zhang, H.**, **Dai, J.**, **Yi, Y.**, **Zhang, H. and Zhang, Y.** (2020). Deep learning on computational-resource-limited platforms: a survey, *Mobile Information Systems*, *2020*, 1–19.

[22] **Pazar, A.**, **Khalilbayli, F.**, **Ozlem, K.**, **Yilmaz, A.F.**, **Atalay, A.T.**, **Atalay, O. and İnce, G.** (2022). Gait Phase Recognition using Textile-based Sensor, *7th International Conference on Computer Science and Engineering (UBMK)*, pp.1–6.

[23] **Louis, M. and John, S.** (1942). *Electrically conductive fabric*, US Patent 2,274,840.

[24] **Seth, A.** (1943). *Electrically conductive fabric*, US Patent 2,327,756.

[25] **William, W.** (1949). *Electrically conductive fabric*, US Patent 2,473,183.

[26] **Sanders, J.** (1974). *Electrically-conductive textile fiber*, US Patent 3,823,035.

[27] **Paton, G.A.**, **Nichols, S.M. and Sanders, J.H.** (1977). *Integral, electrically-conductive textile filament*, uS Patent 4,045,949.

[28] **Özlem, K.**, **Kuyucu, M.K.**, **Bahtiyar, Ş. and İnce, G.** (2019). Security and Privacy Issues for E-textile Applications, *2019 4th International Conference on Computer Science and Engineering (UBMK)*, IEEE, pp.102–107.

[29] **Castano, L.M. and Flatau, A.B.** (2014). Smart fabric sensors and e-textile technologies: a review, *Smart Materials and Structures*, *23*(5), 053001.

[30] **Farringdon, J.** (2001). Wearable electronics and clothing from Philips and Levi, *Technical Textiles International*, *10*(8), 22–24.

[31] **Paradiso, R.**, **Belloc, C.**, **Loriga, G. and Taccini, N.** (2005). Wearable healthcare systems, new frontiers of e-textile, *Studies in health technology and informatics*, *117*, 9–16.

[32] **Van Langenhove, L.** (2007). *Smart textiles for medicine and healthcare: materials, systems and applications*, Elsevier.

[33] **Winterhalter, C.A.**, **Teverovsky, J.**, **Wilson, P.**, **Slade, J.**, **Horowitz, W.**, **Tierney, E. and Sharma, V.** (2005). Development of electronic textiles to support networks, communications, and medical applications in future US Military protective clothing systems, *IEEE Transactions on Information Technology in Biomedicine*, *9*(3), 402–406.

[34] **Cherston, J. and Paradiso, J.A.** (2019). SpaceSkin: development of aerospace-grade electronic textile for simultaneous protection and high velocity impact characterization, *Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems 2019*, volume10970, International Society for Optics and Photonics, p.109700J.

[35] **Carpi, F. and De Rossi, D.** (2005). Electroactive polymer-based devices for e-textiles in biomedicine, *IEEE transactions on Information Technology in biomedicine*, *9*(3), 295–318.

[36] **Aigner, R.**, **Pointner, A.**, **Preindl, T.**, **Parzer, P. and Haller, M.** (2020). Embroidered resistive pressure sensors: A novel approach for textile interfaces, *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pp.1–13.

[37] **Pizarro, F.**, **Villavicencio, P.**, **Yunge, D.**, **Rodríguez, M.**, **Hermosilla, G. and Leiva, A.** (2018). Easy-to-build textile pressure sensor, *Sensors*, *18*(4), 1190.

[38] **Xu, W.**, **Huang, M.C.**, **Amini, N.**, **He, L. and Sarrafzadeh, M.** (2013). ecushion: A textile pressure sensor array design and calibration for sitting posture analysis, *IEEE Sensors Journal*, *13*(10), 3926–3934.

[39] **Shu, L.**, **Hua, T.**, **Wang, Y.**, **Li, Q.**, **Feng, D.D. and Tao, X.** (2010). In-shoe plantar pressure measurement and analysis system based on fabric pressure sensing array, *IEEE Transactions on information technology in biomedicine*, *14*(3), 767–775.

[40] **Heo, J.S.**, **Shishavan, H.H.**, **Soleymanpour, R.**, **Kim, J. and Kim, I.** (2019). Textile-based stretchable and flexible glove sensor for monitoring upper extremity prosthesis functions, *IEEE Sensors Journal*, *20*(4), 1754–1760.

[41] **Shyr, T.W.**, **Shie, J.W.**, **Jiang, C.H. and Li, J.J.** (2014). A textile-based wearable sensing device designed for monitoring the flexion angle of elbow and knee movements, *Sensors*, *14*(3), 4050–4059.

[42] **Mattmann, C.**, **Amft, O.**, **Harms, H.**, **Troster, G. and Clemens, F.** (2007). Recognizing upper body postures using textile strain sensors, *2007 11th IEEE international symposium on wearable computers*, IEEE, pp.29–36.

[43] **Atalay, O.**, **Kennon, W.R. and Demirok, E.** (2014). Weft-knitted strain sensor for monitoring respiratory rate and its electro-mechanical modeling, *IEEE Sensors Journal*, *15*(1), 110–122.

[44] **Ozlem, K.**, **Atalay, O.**, **Atalay, A. and Ince, G.** (2019). Textile Based Sensing System for Lower Limb Motion Monitoring, ***L. Masia, S. Micera, M. Akay and J.L. Pons**, editors, Converging Clinical and Engineering Research on Neurorehabilitation III*, Springer International Publishing, Cham, pp.395–399.

[45] **Soukup, R.**, **Hamacek, A.**, **Mracek, L. and Reboun, J.** (2014). Textile based temperature and humidity sensor elements for healthcare applications, *Proceedings of the 2014 37th international spring seminar on electronics technology*, IEEE, pp.407–411.

[46] **Husain, M.D. and Kennon, R.** (2013). Preliminary investigations into the development of textile based temperature sensor for healthcare applications, *Fibers*, *1*(1), 2–10.

[47] **Atalay, A.**, **Sanchez, V.**, **Atalay, O.**, **Vogt, D.M.**, **Haufe, F.**, **Wood, R.J. and Walsh, C.J.** (2017). Batch fabrication of customizable silicone-textile composite capacitive strain sensors for human motion tracking, *Advanced Materials Technologies*, *2*(9), 1700136.

[48] **Meyer, J.**, **Lukowicz, P. and Troster, G.** (2006). Textile pressure sensor for muscle activity and motion detection, *2006 10th IEEE International Symposium on Wearable Computers*, IEEE, pp.69–72.

[49] **Yang, C.**, **Yang, T.**, **Wu, C.**, **Hung, S.**, **Liao, M.**, **Su, M. and Hsieh, H.** (2014). Textile-based capacitive sensor for a wireless wearable breath monitoring system, *2014 IEEE International Conference on Consumer Electronics (ICCE)*, IEEE, pp.232–233.

[50] **Özlem, K.**, (2018). Textile Based Sensing System For Leg Motion Monitoring, Master's thesis, Istanbul Technical University, Graduate School Of Science Engineering and Technology, advisor: Asst. Prof. Dr. Gökhan İNCE.

[51] **Teodorescu, M. and Teodorescu, H.N.** (2020). Capacitive Interdigital Sensors for Flexible Enclosures and Wearables, *2020 International Conference on Applied Electronics (AE)*, pp.1–6.

[52] **Atalay, O.** (2018). Textile-Based, Interdigital, Capacitive, Soft-Strain Sensor for Wearable Applications, *Materials*, *11*(5), 768.

[53] **Martínez-Estrada, M.**, **Ventura, H.**, **Gil, I. and Fernández-García, R.** (2023). A Full Textile Capacitive Woven Sensor, *Advanced Materials Technologies*, *8*(1), 2200284.

[54] **Yilmaz, A.F.**, **Ahmed, I.A.K.**, **Gumus, C.**, **Ozlem, K.**, **Cetin, M.S.**, **Atalay, A.T.**, **Ince, G. and Atalay, O.** (2024). Highly Stretchable Textile Knitted Interdigital Sensor for Wearable Technology Applications, *Advanced Sensor Research*, *3*(2), 2300121.

[55] **Zhang, Q.**, **Wang, Y.L.**, **Xia, Y.**, **Zhang, P.F.**, **Kirk, T.V. and Chen, X.D.** (2019). Textile-only capacitive sensors for facile fabric integration without compromise of wearability, *Advanced Materials Technologies*, *4*(10), 1900485.

[56] **Kuyucu, C.F.**, **Ayvaz, U.**, **Özlem, K.**, **Atalay, A.**, **Atalay, O. and İnce, G.** (2019). Comparative Assessment of Knee Motion Monitoring Technologies, *2019 4th International Conference on Computer Science and Engineering (UBMK)*, pp.155–160.

[57] **Ayvaz, U.**, **Elmoughni, H.**, **Atalay, A.**, **Atalay, Ö. and Ince, G.** (2020). Real-Time Human Activity Recognition Using Textile-Based Sensors, *EAI International Conference on Body Area Networks*, Springer, pp.168–183.

[58] **Sevinc, H.**, **Ayvaz, U.**, **Ozlem, K.**, **Elmoughni, H.**, **Atalay, A.**, **Atalay, O. and Ince, G.** (2020). Step Length Estimation Using Sensor Fusion, *2020 IEEE International Conference on Flexible and Printable Sensors and Systems (FLEPS)*, pp.1–4.

[59] **Rao, S. and Carter, S.** (2012). Regional plantar pressure during walking, stair ascent and descent, *Gait & posture*, *36*(2), 265–270.

[60] **Valtonen, M.**, **Maentausta, J. and Vanhala, J.** (2009). Tiletrack: Capacitive human tracking using floor tiles, *2009 IEEE international conference on pervasive computing and communications*, IEEE, pp.1–10.

[61] **Singh, G.**, **Nelson, A.**, **Robucci, R.**, **Patel, C. and Banerjee, N.** (2015). Inviz: Low-power personalized gesture recognition using wearable textile capacitive sensor arrays, *2015 IEEE international conference on pervasive computing and communications (PerCom)*, IEEE, pp.198–206.

[62] **Baldwin, R.**, **Bobovych, S.**, **Robucci, R.**, **Patel, C. and Banerjee, N.** (2015). Gait analysis for fall prediction using hierarchical textile-based capacitive sensor arrays, *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, pp.1293–1298.

[63] **Koo, H.R.**, **Lee, Y.J.**, **Gi, S.**, **Khang, S.**, **Lee, J.H.**, **Lee, J.H.**, **Lim, M.G.**, **Park, H.J. and Lee, J.W.** (2014). The effect of textile-based inductive coil sensor positions for heart rate monitoring, *Journal of medical systems*, *38*(2), 1–12.

[64] **Tavassolian, M.**, **Cuthbert, T.J.**, **Napier, C.**, **Peng, J. and Menon, C.** (2020). Textile-Based Inductive Soft Strain Sensors for Fast Frequency Movement and Their Application in Wearable Devices Measuring Multiaxial Hip Joint Angles during Running, *Advanced Intelligent Systems*, *2*(4), 1900165.

[65] **García Patiño, A.**, **Khoshnam, M. and Menon, C.** (2020). Wearable device to monitor back movements using an inductive textile sensor, *Sensors*, *20*(3), 905.

[66] **Pola, T. and Vanhala, J.** (2007). Textile electrodes in ECG measurement, *2007 3rd International Conference on Intelligent Sensors, Sensor Networks and Information*, IEEE, pp.635–639.

[67] **Pani, D.**, **Dessì, A.**, **Saenz-Cogollo, J.F.**, **Barabino, G.**, **Fraboni, B. and Bonfiglio, A.** (2015). Fully textile, PEDOT: PSS based electrodes for wearable ECG monitoring systems, *IEEE Transactions on Biomedical Engineering*, *63*(3), 540–549.

[68] **Zhou, Y.**, **Ding, X.**, **Zhang, J.**, **Duan, Y.**, **Hu, J. and Yang, X.** (2014). Fabrication of conductive fabric as textile electrode for ECG monitoring, *Fibers and Polymers*, *15*(11), 2260–2264.

[69] **Finni, T.**, **Hu, M.**, **Kettunen, P.**, **Vilavuo, T. and Cheng, S.** (2007). Measurement of EMG activity with textile electrodes embedded into clothing, *Physiological measurement*, *28*(11), 1405.

[70] **Zhang, H.**, **Tian, L.**, **Zhang, L. and Li, G.** (2013). Using textile electrode EMG for prosthetic movement identification in transradial amputees, *2013 IEEE International Conference on Body Sensor Networks*, IEEE, pp.1–5.

[71] **Löfhede, J.**, **Seoane, F. and Thordstein, M.** (2010). Soft textile electrodes for EEG monitoring, *Proceedings of the 10th IEEE International Conference on Information Technology and Applications in Biomedicine*, IEEE, pp.1–4.

[72] **Löfhede, J.**, **Seoane, F. and Thordstein, M.** (2012). Textile electrodes for EEG recording—A pilot study, *Sensors*, *12*(12), 16907–16919.

[73] **Paket, E.**, **Ozlem, K.**, **Elmoughni, H.**, **Atalay, A.**, **Atalay, O. and Ince, G.** (2020). ECG Monitoring System Using Textile Electrodes, *2020 28th Signal Processing and Communications Applications Conference (SIU)*, IEEE, pp.1–4.

[74] **Persson, N.K.**, **Martinez, J.G.**, **Zhong, Y.**, **Maziz, A. and Jager, E.W.H.** (2018). Actuating Textiles: Next Generation of Smart Textiles, *Advanced Materials Technologies*, *3*(10), 1700397.

[75] **Mahadevan, K.**, **Stoltzfus, A.**, **Dealey, S. and Granberry, R.** (2023). 3D knit pneumatic actuators for wearable haptic displays, *Extreme Mechanics Letters*, *65*, 102102.

[76] **Zannat, A.**, **Uddin, M.N.**, **Mahmud, S.T.**, **Prithu, P.S.S. and Mia, R.** (2023). Review: Textile-based soft robotics for physically challenged individuals, *Journal of Materials Science*, *58*(31), 12491–12536.

[77] **Thalman, C. and Artemiadis, P.** (2020). A review of soft wearable robots that provide active assistance: Trends, common actuation methods, fabrication, and applications, *Wearable Technologies*, *1*, e3.

[78] **Xiloyannis, M.**, **Cappello, L.**, **Binh, K.D.**, **Antuvan, C.W. and Masia, L.** (2017). Preliminary design and control of a soft exosuit for assisting elbow movements and hand grasping in activities of daily living, *Journal of Rehabilitation and Assistive Technologies Engineering*, *4*, 2055668316680315.

[79] **Xiloyannis, M.**, **Chiaradia, D.**, **Frisoli, A. and Masia, L.** (2019). Physiological and kinematic effects of a soft exosuit on arm movements, *Journal of NeuroEngineering and Rehabilitation*, *16*(1), 29.

[80] **Schmidt, K.**, **Duarte, J.E.**, **Grimmer, M.**, **Sancho-Puchades, A.**, **Wei, H.**, **Easthope, C.S. and Riener, R.** (2017). The Myosuit: Bi-articular Anti-gravity Exosuit That Reduces Hip Extensor Activity in Sitting Transfers, *Frontiers in Neurorobotics*, *11*, 57.

[81] **Asbeck, A.T.**, **Schmidt, K. and Walsh, C.J.** (2015). Soft exosuit for hip assistance, *Robotics and Autonomous Systems*, *73*, 102–110.

[82] **Ding, Y.**, **Kim, M.**, **Kuindersma, S. and Walsh, C.J.** (2018). Human-in-the-loop optimization of hip assistance with a soft exosuit during walking, *Science Robotics*, *3*(15), eaar5438.

[83] **Samper-Escudero, J.L.**, **Giménez-Fernandez, A.**, **Sánchez-Urán, M.A. and Ferre, M.** (2020). A Cable-Driven Exosuit for Upper Limb Flexion Based on Fibres Compliance, *IEEE Access*, *8*, 153297–153310.

[84] **Popov, D.**, **Gaponov, I. and Ryu, J.H.** (2017). Portable Exoskeleton Glove With Soft Structure for Hand Assistance in Activities of Daily Living, *IEEE/ASME Transactions on Mechatronics*, *22*(2), 865–875.

[85] **Rognon, C.**, **Ramachandran, V.**, **Wu, A.R.**, **Ijspeert, A.J. and Floreano, D.** (2019). Haptic Feedback Perception and Learning With Cable-Driven Guidance in Exosuit Teleoperation of a Simulated Drone, *IEEE Transactions on Haptics*, *12*(3), 375–385.

[86] **Chen, Y.**, **Yang, Y.**, **Li, M.**, **Chen, E.**, **Mu, W.**, **Fisher, R. and Yin, R.** (2021). Wearable Actuators: An Overview, *Textiles*, *1*(2), 283–321.

[87] **Thalman, C.M.**, **Baye-Wallace, L. and Lee, H.** (2021). A Soft Robotic Hip Exosuit (SR-HExo) to Assist Hip Flexion and Extension during Human Locomotion, *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp.5060–5066.

[88] **Belforte, G.**, **Eula, G.**, **Ivanov, A.**, **Raparelli, T. and Sirolli, S.** (2018). Presentation of textile pneumatic muscle prototypes applied in an upper limb active suit experimental model, *The Journal of The Textile Institute*, *109*(6), 757–766.

[89] **Sridar, S.**, **Veale, A.J.**, **Sartori, M. and van der Kooij, H.** (2023). Exploiting a Simple Asymmetric Pleating Method to Realize a Textile Based Bending Actuator, *IEEE Robotics and Automation Letters*, *8*(3), 1794–1801.

[90] **Sanchez, V.**, **Mahadevan, K.**, **Ohlson, G.**, **Graule, M.A.**, **Yuen, M.C.**, **Teeple, C.B.**, **Weaver, J.C.**, **McCann, J.**, **Bertoldi, K. and Wood, R.J.** (2023). 3D Knitting for Pneumatic Soft Robotics, *Advanced Functional Materials*, *33*(26), 2212541.

[91] **O'Neill, C.T.**, **McCann, C.M.**, **Hohimer, C.J.**, **Bertoldi, K. and Walsh, C.J.** (2022). Unfolding Textile-Based Pneumatic Actuators for Wearable Applications, *Soft Robotics*, *9*(1), 163–172.

[92] **Ge, L.**, **Chen, F.**, **Wang, D.**, **Zhang, Y.**, **Han, D.**, **Wang, T. and Gu, G.** (2020). Design, Modeling, and Evaluation of Fabric-Based Pneumatic Actuators for Soft Wearable Assistive Gloves, *Soft Robotics*, *7*(5), 583–596.

[93] **Yilmaz, A.F.**, **Khalilbayli, F.**, **Ozlem, K.**, **Elmoughni, H.M.**, **Kalaoglu, F.**, **Atalay, A.T.**, **Ince, G. and Atalay, O.** (2022). Effect of Segment Types on Characterization of Soft Sensing Textile Actuators for Soft Wearable Robots, *Biomimetics*, *7*(4), 249.

[94] **Suulker, C.**, **Skach, S. and Althoefer, K.** (2022). Soft Robotic Fabric Actuator With Elastic Bands for High Force and Bending Performance in Hand Exoskeletons, *IEEE Robotics and Automation Letters*, *7*(4), 10621–10627.

[95] **Yilmaz, A.F.**, **Ozlem, K.**, **Khalilbayli, F.**, **Celebi, M.F.**, **Kalaoglu, F.**, **Atalay, A.T.**, **Ince, G. and Atalay, O.** (2024). Resistive Self-Sensing Controllable Fabric-Based Actuator: A Novel Approach to Creating Anisotropy, *Advanced Sensor Research*, *3*(7), 2300108.

[96] **Cappello, L.**, **Galloway, K.C.**, **Sanan, S.**, **Wagner, D.A.**, **Granberry, R.**, **Engelhardt, S.**, **Haufe, F.L.**, **Peisner, J.D. and Walsh, C.J.** (2018). Exploiting Textile Mechanical Anisotropy for Fabric-Based Pneumatic Actuators, *Soft Robotics*, *5*(5), 662–674.

[97] **Luo, Y.**, **Wu, K.**, **Spielberg, A.**, **Foshey, M.**, **Rus, D.**, **Palacios, T. and Matusik, W.** (2022). Digital Fabrication of Pneumatic Actuators with Integrated Sensing by Machine Knitting, *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, CHI '22, Association for Computing Machinery, New York, NY, USA, pp.1–13.

[98] **Fang, J.**, **Yuan, J.**, **Wang, M.**, **Xiao, L.**, **Yang, J.**, **Lin, Z.**, **Xu, P. and Hou, L.** (2020). Novel Accordion-Inspired Foldable Pneumatic Actuators for Knee Assistive Devices, *Soft Robotics*, *7*(1), 95–108.

[99] **Nassour, J. and Hamker, F.** (2019). Enfolded Textile Actuator for Soft Wearable Robots, *2019 IEEE International Conference on Cyborg and Bionic Systems (CBS)*, pp.60–65.

[100] **Kim, J.H.H.**, **Patil, S.D.**, **Matson, S.**, **Conroy, M. and Kao, C.H.L.** (2022). KnitSkin: Machine-Knitted Scaled Skin for Locomotion, *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, CHI '22, Association for Computing Machinery, New York, NY, USA, pp.1–15.

[101] **Ma, J.**, **Chen, D.**, **Liu, Z.**, **Wei, J.**, **Zhang, X.**, **Zeng, Z. and Jiang, Y.** (2023). All-Fabric Bi-directional Actuators for Multi-joint Assistance of Upper Limb, *Journal of Bionic Engineering*, *20*(6), 2661–2669.

[102] **Eschen, K.**, **Granberry, R.**, **Holschuh, B. and Abel, J.** (2020). Amplifying and Leveraging Generated Force Upon Heating and Cooling in SMA Knitted Actuators, *ACS Applied Materials & Interfaces*, *12*(48), 54155–54167.

[103] **Asar, A.**, **Irfan, M.**, **Khan, K.**, **Zaki, W. and Umer, R.** (2022). Self-sensing shape memory polymer composites reinforced with functional textiles, *Composites Science and Technology*, *221*, 109219.

[104] **Lee, J.A.**, **Li, N.**, **Haines, C.S.**, **Kim, K.J.**, **Lepró, X.**, **Ovalle-Robles, R.**, **Kim, S.J. and Baughman, R.H.** (2017). Electrochemically Powered, Energy-Conserving Carbon Nanotube Artificial Muscles, *Advanced Materials*, *29*(31), 1700870.

[105] **Jang, Y.**, **Kim, S.M.**, **Spinks, G.M. and Kim, S.J.** (2020). Carbon Nanotube Yarn for Fiber-Shaped Electrical Sensors, Actuators, and Energy Storage for Smart Systems, *Advanced Materials*, *32*(5), 1902670.

[106] **Aziz, S.**, **Martinez, J.G.**, **Foroughi, J.**, **Spinks, G.M. and Jager, E.W.H.** (2020). Artificial Muscles from Hybrid Carbon Nanotube-Polypyrrole-Coated Twisted and Coiled Yarns, *Macromolecular Materials and Engineering*, *305*(11), 2000421.

[107] **Arora, S.**, **Ghosh, T. and Muth, J.** (2007). Dielectric elastomer based prototype fiber actuators, *Sensors and Actuators A: Physical*, *136*(1), 321–328.

[108] **Jia, T.**, **Wang, Y.**, **Dou, Y.**, **Li, Y.**, **Jung de Andrade, M.**, **Wang, R.**, **Fang, S.**, **Li, J.**, **Yu, Z.**, **Qiao, R.**, **Liu, Z.**, **Cheng, Y.**, **Su, Y.**, **Minary-Jolandan, M.**, **Baughman, R.H.**, **Qian, D. and Liu, Z.** (2019). Moisture Sensitive Smart Yarns and Textiles from Self-Balanced Silk Fiber Muscles, *Advanced Functional Materials*, *29*(18), 1808241.

[109] **Wu, J.**, **Jiang, W.**, **Gu, M.**, **Sun, F.**, **Han, C. and Gong, H.** (2023). Flexible Actuators with Hygroscopic Adaptability for Smart Wearables and Soft Grippers, *ACS Applied Materials & Interfaces*, *15*(51), 59989–60001.

[110] **Zhao, H.**, **Qi, X.**, **Ma, Y.**, **Sun, X.**, **Liu, X.**, **Zhang, X.**, **Tian, M. and Qu, L.** (2021). Wearable Sunlight-Triggered Bimorph Textile Actuators, *Nano Letters*, *21*(19), 8126–8134.

[111] **Kim, S.**, **Gu, S. and Kim, J.** (2022). Variable Shape and Stiffness Feedback System for VR Gloves Using SMA Textile Actuator, *Fibers and Polymers*, *23*(3), 836–842.

[112] **Shin, J.**, **Han, Y.J.**, **Lee, J.H. and Han, M.W.** (2023). Shape Memory Alloys in Textile Platform: Smart Textile-Composite Actuator and Its Application to Soft Grippers, *Sensors*, *23*(3), 1518.

[113] **Kim, C.**, **Kim, G.**, **Lee, Y.**, **Lee, G.**, **Han, S.**, **Kang, D.**, **Koo, S.H. and Koh, J.s.** (2020). Shape memory alloy actuator-embedded smart clothes for ankle assistance, *Smart Materials and Structures*, *29*(5), 055003.

[114] **Park, S.J. and Park, C.H.** (2019). Suit-type Wearable Robot Powered by Shape-memory-alloy-based Fabric Muscle, *Scientific Reports*, *9*(1), 9157.

[115] **Wang, W.**, **Yao, L.**, **Cheng, C.Y.**, **Zhang, T.**, **Atsumi, H.**, **Wang, L.**, **Wang, G.**, **Anilionyte, O.**, **Steiner, H.**, **Ou, J.**, **Zhou, K.**, **Wawrousek, C.**, **Petrecca, K.**, **Belcher, A.M.**, **Karnik, R.**, **Zhao, X.**, **Wang, D.I.C. and Ishii, H.** (2017). Harnessing the hygroscopic and biofluorescent behaviors of genetically tractable microbial cells to design biohybrid wearables, *Science Advances*, *3*(5), e1601984.

[116] **Hoffmann, R., Brodowski, H., Steinhage, A. and Grzegorzek, M.** (2021). Detecting walking challenges in gait patterns using a capacitive sensor floor and recurrent neural networks, *Sensors*, *21*(4), 1086.

[117] **Ren, K., Chen, Z., Ling, Y. and Zhao, J.** (2022). Recognition of freezing of gait in Parkinson's disease based on combined wearable sensors, *BMC neurology*, *22*(1), 1–13.

[118] **Khan, M.H., Farid, M.S. and Grzegorzek, M.** (2020). A non-linear view transformations model for cross-view gait recognition, *Neurocomputing*, *402*, 100–111.

[119] **Muñoz, B., Castaño-Pino, Y.J., Paredes, J.D.A. and Navarro, A.** (2018). Automated gait analysis using a Kinect camera and wavelets, *2018 IEEE 20th International Conference on e-Health Networking, Applications and Services (Healthcom)*, IEEE, pp.1–5.

[120] **Rezaei, A., Ejupi, A., Gholami, M., Ferrone, A. and Menon, C.** (2018). Preliminary investigation of textile-based strain sensors for the detection of human gait phases using machine learning, *2018 7th IEEE International Conference on Biomedical Robotics and Biomechatronics (Biorob)*, IEEE, pp.563–568.

[121] **Aqueveque, P., Pastene, F., Osorio, R., Saavedra, F., Pinto, D., Ortega-Bastidas, P. and Gomez, B.** (2020). A novel capacitive step sensor to trigger stimulation on functional electrical stimulators devices for Drop Foot, *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *28*(12), 3083–3088.

[122] **Washabaugh, E.P., Kalyanaraman, T., Adamczyk, P.G., Claflin, E.S. and Krishnan, C.** (2017). Validity and repeatability of inertial measurement units for measuring gait parameters, *Gait & posture*, *55*, 87–93.

[123] **Sigcha, L., Costa, N., Pavón, I., Costa, S., Arezes, P., López, J.M. and De Arcas, G.** (2020). Deep learning approaches for detecting freezing of gait in Parkinson's disease patients through on-body acceleration sensors, *Sensors*, *20*(7), 1895.

[124] **Qiu, S., Wang, Z., Zhao, H., Qin, K., Li, Z. and Hu, H.** (2018). Inertial/magnetic sensors based pedestrian dead reckoning by means of multi-sensor fusion, *Information Fusion*, *39*, 108–119.

[125] **Nakamoto, H., Yamaji, T., Hirata, I., Ootaka, H. and Kobayashi, F.** (2018). Joint angle measurement by stretchable strain sensor, *Journal of Ambient Intelligence and Humanized Computing*, *14*(11), 1–6.

[126] **Di Nardo, F., Morbidoni, C., Cucchiarelli, A. and Fioretti, S.** (2020). Recognition of Gait Phases with a Single Knee Electrogoniometer: A Deep Learning Approach, *Electronics*, *9*(2), 355.

[127] **Ding, Z.**, **Yang, C.**, **Xing, K.**, **Ma, X.**, **Yang, K.**, **Guo, H.**, **Yi, C. and Jiang, F.** (2018). The real time gait phase detection based on long short-term memory, *2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*, IEEE, pp.33–38.

[128] **Gadaleta, M.**, **Cisotto, G.**, **Rossi, M.**, **Rehman, R.Z.U.**, **Rochester, L. and Del Din, S.** (2019). Deep learning techniques for improving digital gait segmentation, *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, IEEE, pp.1834–1837.

[129] **Phan, D.**, **Nguyen, N.**, **Pathirana, P.N.**, **Horne, M.**, **Power, L. and Szmulewicz, D.** (2019). A random forest approach for quantifying gait ataxia with truncal and peripheral measurements using multiple wearable sensors, *IEEE Sensors Journal*, *20*(2), 723–734.

[130] **Shetty, S. and Rao, Y.** (2016). SVM based machine learning approach to identify Parkinson's disease using gait analysis, *2016 International Conference on Inventive Computation Technologies (ICICT)*, volume 2, IEEE, pp.1–5.

[131] **Yan, C.**, **Zhang, B. and Coenen, F.** (2015). Multi-attributes gait identification by convolutional neural networks, *2015 8th International Congress on Image and Signal Processing (CISP)*, IEEE, pp.642–647.

[132] **Amirpour, E.**, **Fesharakifard, R.**, **Ghafarirad, H.**, **Rezaei, S.M.**, **Saboukhi, A.**, **Savabi, M. and Gorji, M.R.** (2022). A novel hand exoskeleton to enhance fingers motion for tele-operation of a robot gripper with force feedback, *Mechatronics*, *81*, 102695.

[133] **Peperoni, E.**, **Capitani, S.L.**, **Fiumalbi, T.**, **Capotorti, E.**, **Baldoni, A.**, **Dell'Agnello, F.**, **Creatini, I.**, **Taglione, E.**, **Vitiello, N.**, **Trigili, E. and Crea, S.** (2023). Self-Aligning Finger Exoskeleton for the Mobilization of the Metacarpophalangeal Joint, *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *31*, 884–894.

[134] **Haarman, C.J.W.**, **Hekman, E.E.G.**, **Rietman, J.S. and Van Der Kooij, H.** (2023). Mechanical Design and Feasibility of a Finger Exoskeleton to Support Finger Extension of Severely Affected Stroke Patients, *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *31*, 1268–1276.

[135] **Sun, N.**, **Li, G. and Cheng, L.** (2021). Design and Validation of a Self-Aligning Index Finger Exoskeleton for Post-Stroke Rehabilitation, *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *29*, 1513–1523.

[136] **Burns, M.K. and Vinjamuri, R.** (2020). Design of a Soft Glove-Based Robotic Hand Exoskeleton with Embedded Synergies, Springer International Publishing, Cham, pp.71–87.

[137] **Meyer-Heim, A. and van Hedel, H.J.** (2013). Robot-Assisted and Computer-Enhanced Therapies for Children with Cerebral Palsy: Current State and Clinical Implementation, *Seminars in Pediatric Neurology*, *20*(2), 139–145.

[138] **Sanders, Q.**, **Chan, V.**, **Augsburger, R.**, **Cramer, S.C.**, **Reinkensmeyer, D.J. and Do, A.H.** (2020). Feasibility of Wearable Sensing for In-Home Finger Rehabilitation Early After Stroke, *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *28*(6), 1363–1372.

[139] **Park, S.**, **Fraser, M.**, **Weber, L.M.**, **Meeker, C.**, **Bishop, L.**, **Geller, D.**, **Stein, J. and Ciocarlie, M.** (2020). User-Driven Functional Movement Training With a Wearable Hand Robot After Stroke, *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *28*(10), 2265–2275.

[140] **Gu, W.**, **Yan, S.**, **Xiong, J.**, **Li, Y.**, **Zhang, Q.**, **Li, K.**, **Hou, C. and Wang, H.** (2023). Wireless smart gloves with ultra-stable and all-recyclable liquid metal-based sensing fibers for hand gesture recognition, *Chemical Engineering Journal*, *460*, 141777.

[141] **Chu, M.**, **Cui, Z.**, **Zhang, A.**, **Yao, J.**, **Tang, C.**, **Fu, Z.**, **Nathan, A. and Gao, S.** (2022). Multisensory Fusion, Haptic, and Visual Feedback Teleoperation System Under IoT Framework, *IEEE Internet of Things Journal*, *9*(20), 19717–19727.

[142] **Yu, F.**, **Chen, Z.**, **Jiang, M.**, **Tian, Z.**, **Peng, T. and Hu, X.** (2023). Smart Clothing System With Multiple Sensors Based on Digital Twin Technology, *IEEE Internet of Things Journal*, *10*(7), 6377–6387.

[143] **Polygerinos, P.**, **Wang, Z.**, **Galloway, K.C.**, **Wood, R.J. and Walsh, C.J.** (2015). Soft robotic glove for combined assistance and at-home rehabilitation, *Robotics and Autonomous Systems*, *73*, 135–143.

[144] **Câmara Gradim, L.C.**, **Archanjo José, M.**, **Marinho Cezar da Cruz, D. and de Deus Lopes, R.** (2020). IoT Services and Applications in Rehabilitation: An Interdisciplinary and Meta-Analysis Review, *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *28*(9), 2043–2052.

[145] **Nuckols, R.W.**, **Lee, S.**, **Swaminathan, K.**, **Orzel, D.**, **Howe, R.D. and Walsh, C.J.** (2021). Individualization of exosuit assistance based on measured muscle dynamics during versatile walking, *Science Robotics*, *6*(60), eabj1362.

[146] **Gasser, B.W.**, **Martínez, A.**, **Sasso-Lance, E.**, **Kandilakis, C.**, **Durrough, C.M. and Goldfarb, M.** (2020). Preliminary Assessment of a Hand and Arm Exoskeleton for Enabling Bimanual Tasks for Individuals With Hemiparesis, *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *28*(10), 2214–2223.

[147] **Zeng, H.**, **Yu, W.**, **Chen, D.**, **Hu, X.**, **Zhang, D. and Song, A.** (2022). Exploring Biomimetic Stiffness Modulation and Wearable Finger Haptics for Improving Myoelectric Control of Virtual Hand, *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *30*, 1601–1611.

[148] **M, S.**, **Venusamy, K.**, **S, S.**, **S, S. and O, N.K.** (2023). A Comprehensive Review of Haptic Gloves: Advances, Challenges, and Future Directions, *2023 Second International Conference on Electronics and Renewable Systems (ICEARS)*, pp.227–233.

[149] **Salman, F.**, **Cui, Y.**, **Imran, Z.**, **Liu, F.**, **Wang, L. and Wu, W.** (2020). A Wireless-controlled 3D printed Robotic Hand Motion System with Flex Force Sensors, *Sensors and Actuators A: Physical*, *309*, 112004.

[150] **Yap, H.K.**, **Khin, P.M.**, **Koh, T.H.**, **Sun, Y.**, **Liang, X.**, **Lim, J.H. and Yeow, C.H.** (2017). A Fully Fabric-Based Bidirectional Soft Robotic Glove for Assistance and Rehabilitation of Hand Impaired Patients, *IEEE Robotics and Automation Letters*, *2*(3), 1383–1390.

[151] **Soekadar, S.R.**, **Witkowski, M.**, **Gómez, C.**, **Opisso, E.**, **Medina, J.**, **Cortese, M.**, **Cempini, M.**, **Carrozza, M.C.**, **Cohen, L.G.**, **Birbaumer, N. and Vitiello, N.** (2016). Hybrid EEG/EOG-based brain/neural hand exoskeleton restores fully independent daily living activities after quadriplegia, *Science Robotics*, *1*(1), eaag3296.

[152] **Dunaway, S.**, **Dezsi, D.B.**, **Perkins, J.**, **Tran, D. and Naft, J.** (2017). Case Report on the Use of a Custom Myoelectric Elbow–Wrist–Hand Orthosis for the Remediation of Upper Extremity Paresis and Loss of Function in Chronic Stroke, *Military Medicine*, *182*(7), e1963–e1968.

[153] **Secciani, N.**, **Topini, A.**, **Ridolfi, A.**, **Meli, E. and Allotta, B.** (2020). A Novel Point-in-Polygon-Based sEMG Classifier for Hand Exoskeleton Systems, *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *28*(12), 3158–3166.

[154] **Tran, P.**, **Jeong, S.**, **Wolf, S.L. and Desai, J.P.** (2020). Patient-Specific, Voice-Controlled, Robotic FLEXotendon Glove-II System for Spinal Cord Injury, *IEEE Robotics and Automation Letters*, *5*(2), 898–905.

[155] **Hazubski, S.**, **Hoppe, H. and Otte, A.** (2020). Non-contact visual control of personalized hand prostheses/exoskeletons by tracking using augmented reality glasses, *3D printing in medicine*, *6*, 6.

[156] **Chen, W.**, **Li, G.**, **Li, N.**, **Wang, W.**, **Yu, P.**, **Wang, R.**, **Xue, X.**, **Zhao, X. and Liu, L.** (2023). Restoring Voluntary Bimanual Activities of Patients With Chronic Hemiparesis Through a Foot-Controlled Hand/Forearm Exoskeleton, *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *31*, 769–778.

[157] **Hampali, S.**, **Rad, M.**, **Oberweger, M. and Lepetit, V.** (2020). HOnnotate: A Method for 3D Annotation of Hand and Object Poses, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, Washington.

[158] **Yuan, S.**, **Ye, Q.**, **Stenger, B.**, **Jain, S. and Kim, T.K.** (2017). BigHand2.2M Benchmark: Hand Pose Dataset and State of the Art Analysis, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, Hawaii.

[159] **Lee, Y.**, **Kim, M.**, **Lee, Y.**, **Kwon, J.**, **Park, Y.L. and Lee, D.** (2019). Wearable Finger Tracking and Cutaneous Haptic Interface with Soft Sensors for Multi-Fingered Virtual Manipulation, *IEEE/ASME Transactions on Mechatronics*, *24*(1), 67–77.

[160] **Sun, H.**, **Kuchenbecker, K.J. and Martius, G.** (2022). A soft thumb-sized vision-based sensor with accurate all-round force perception, *Nature Machine Intelligence*, *4*(2), 135–145.

[161] **Armagan, A.**, **Garcia-Hernando, G.**, **Baek, S.**, **Hampali, S.**, **Rad, M.**, **Zhang, Z.**, **Xie, S.**, **Chen, M.**, **Zhang, B.**, **Xiong, F.**, **Xiao, Y.**, **Cao, Z.**, **Yuan, J.**, **Ren, P.**, **Huang, W.**, **Sun, H.**, **Hrúz, M.**, **Kanis, J.**, **Krňoul, Z.**, **Wan, Q.**, **Li, S.**, **Yang, L.**, **Lee, D.**, **Yao, A.**, **Zhou, W.**, **Mei, S.**, **Liu, Y.**, **Spurr, A.**, **Iqbal, U.**, **Molchanov, P.**, **Weinzaepfel, P.**, **Brégier, R.**, **Rogez, G.**, **Lepetit, V. and Kim, T.K.** (2020). Measuring Generalisation to Unseen Viewpoints, Articulations, Shapes and Objects for 3D Hand Pose Estimation Under Hand-Object Interaction, ***A. Vedaldi, H. Bischof, T. Brox and J.M. Frahm**, editors, Computer Vision – ECCV 2020*, Springer International Publishing, Cham, pp.85–101.

[162] **Baldi, T.L.**, **Scheggi, S.**, **Meli, L.**, **Mohammadi, M. and Prattichizzo, D.** (2017). GESTO: A Glove for Enhanced Sensing and Touching Based on Inertial and Magnetic Sensors for Hand Tracking and Cutaneous Feedback, *IEEE Transactions on Human-Machine Systems*, *47*(6), 1066–1076.

[163] **Thuruthel, T.G.**, **Shih, B.**, **Laschi, C. and Tolley, M.T.** (2019). Soft robot perception using embedded soft sensors and recurrent neural networks, *Science Robotics*, *4*(26), eaav1488.

[164] **Milea, P.**, **Dascalu, M.**, **Opris, C.**, **Franti, E.**, **Dumitrache, M. and Stoica, C.I.** (2016). Using pressure sensors for motion detection and actuation of remote manipulation devices, *Romanian journal of information science and technology*, *19*(4), 321–330.

[165] **Massari, L.**, **Fransvea, G.**, **D'Abbraccio, J.**, **Filosa, M.**, **Terruso, G.**, **Aliperta, A.**, **D'Alesio, G.**, **Zaltieri, M.**, **Schena, E.**, **Palermo, E.** *et al.* (2022). Functional mimicry of Ruffini receptors with fibre Bragg gratings and deep neural networks enables a bio-inspired large-area tactile-sensitive skin, *Nature Machine Intelligence*, *4*(5), 425–435.

[166] **Luo, Y.**, **Wang, Z.**, **Wang, J.**, **Xiao, X.**, **Li, Q.**, **Ding, W. and Fu, H.** (2021). Triboelectric bending sensor based smart glove towards intuitive multi-dimensional human-machine interfaces, *Nano Energy*, *89*, 106330.

[167] **Ling Li, Shuo Jiang, P.B.S. and Gu, G.** (2018). SkinGest: artificial skin for gesture recognition via filmy stretchable strain sensors*, *Advanced Robotics*, *32*(21), 1112–1121.

[168] **Saypulaev, G.R.**, **Merkuryev, I.V.**, **Saypulaev, M.R.**, **Shestakov, V.K.**, **Glazkov, N.V. and Andreev, D.R.** (2023). A Review of Robotic Gloves Applied for Remote Control in Various Systems, *2023 5th International Youth Conference on Radio Electronics, Electrical and Power Engineering (REEPE)*, volume 5, pp.1–6.

[169] **Haghshenas-Jaryani, M.**, **Patterson, R.M.**, **Bugnariu, N. and Wijesundara, M.B.** (2020). A pilot study on the design and validation of a hybrid exoskeleton robotic device for hand rehabilitation, *Journal of Hand Therapy*, *33*(2), 198–208.

[170] **Sarajchi, M.**, **Al-Hares, M.K. and Sirlantzis, K.** (2021). Wearable Lower-Limb Exoskeleton for Children With Cerebral Palsy: A Systematic Review of Mechanical Design, Actuation Type, Control Strategy, and Clinical Evaluation, *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *29*, 2695–2720.

[171] **Hu, D.**, **Giorgio-Serchi, F.**, **Zhang, S. and Yang, Y.** (2023). Stretchable e-skin and transformer enable high-resolution morphological reconstruction for soft robots, *Nature Machine Intelligence*, *5*(3), 261–272.

[172] **Kim, T.**, **Lee, S.**, **Hong, T.**, **Shin, G.**, **Kim, T. and Park, Y.L.** (2020). Heterogeneous sensing in a multifunctional soft sensor for human-robot interfaces, *Science Robotics*, *5*(49), eabc6878.

[173] **Maeder-York, P.**, **Clites, T.**, **Boggs, E.**, **Neff, R.**, **Polygerinos, P.**, **Holland, D.**, **Stirling, L.**, **Galloway, K.**, **Wee, C. and Walsh, C.** (2014). Biologically Inspired Soft Robot for Thumb Rehabilitation, *Journal of Medical Devices*, *8*(2), 020933.

[174] **Lai, J.**, **Song, A.**, **Wang, J.**, **Lu, Y.**, **Wu, T.**, **Li, H.**, **Xu, B. and Wei, X.** (2023). A Novel Soft Glove Utilizing Honeycomb Pneumatic Actuators (HPAs) for Assisting Activities of Daily Living, *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *31*, 3223–3233.

[175] **Correia, C.**, **Nuckols, K.**, **Wagner, D.**, **Zhou, Y.M.**, **Clarke, M.**, **Orzel, D.**, **Solinsky, R.**, **Paganoni, S. and Walsh, C.J.** (2020). Improving Grasp Function After Spinal Cord Injury With a Soft Robotic Glove, *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *28*(6), 1407–1415.

[176] **Rountree, D. and Castrillo, I.** (2013). *The basics of cloud computing: Understanding the fundamentals of cloud computing in theory and practice*, Newnes.

[177] **Buyya, R.**, **Vecchiola, C. and Selvi, S.T.** (2013). *Mastering cloud computing: foundations and applications programming*, Newnes.

[178] **Zhang, Q.**, **Cheng, L. and Boutaba, R.** (2010). Cloud computing: state-of-the-art and research challenges, *Journal of internet services and applications*, *1*, 7–18.

[179] **Armbrust, M.**, **Fox, A.**, **Griffith, R.**, **Joseph, A.D.**, **Katz, R.**, **Konwinski, A.**, **Lee, G.**, **Patterson, D.**, **Rabkin, A.**, **Stoica, I.** *et al.* (2010). A view of cloud computing, *Communications of the ACM*, *53*(4), 50–58.

[180] **Mell, P. and Grance, T.** (2011). *The NIST Definition of Cloud Computing*.

[181] **Ranjan, R.** (2014). Streaming Big Data Processing in Datacenter Clouds, *IEEE Cloud Computing*, *1*(1), 78–83.

[182] **Bernstein, D.**, **Ludvigson, E.**, **Sankar, K.**, **Diamond, S. and Morrow, M.** (2009). Blueprint for the intercloud-protocols and formats for cloud computing interoperability, *2009 fourth international conference on Internet and web applications and services*, IEEE, pp.328–336.

[183] **Hsiao, H.C.**, **Chung, H.Y.**, **Shen, H. and Chao, Y.C.** (2012). Load rebalancing for distributed file systems in clouds, *IEEE transactions on parallel and distributed systems*, *24*(5), 951–962.

[184] **Dean, J. and Ghemawat, S.** (2008). MapReduce: simplified data processing on large clusters, *Communications of the ACM*, *51*(1), 107–113.

[185] **Kandukuri, B.R.**, **V., R.P. and Rakshit, A.** (2009). Cloud Security Issues, *2009 IEEE International Conference on Services Computing*, pp.517–520.

[186] **Vogels, W.** (2009). Eventually consistent, *Communications of the ACM*, *52*(1), 40–44.

[187] **Satyanarayanan, M.** (2017). The Emergence of Edge Computing, *Computer*, *50*(1), 30–39.

[188] **Takabi, H.**, **Joshi, J.B. and Ahn, G.J.** (2010). Security and Privacy Challenges in Cloud Computing Environments, *IEEE Security & Privacy*, *8*(6), 24–31.

[189] **Subashini, S. and Kavitha, V.** (2011). A survey on security issues in service delivery models of cloud computing, *Journal of Network and Computer Applications*, *34*(1), 1–11.

[190] **Vaquero, L.M.**, **Rodero-Merino, L.**, **Caceres, J. and Lindner, M.** (2009). A break in the clouds: towards a cloud definition, *SIGCOMM Computer Communication Review*, *39*(1), 50–55.

[191] **Berisha, B.**, **Mëziu, E. and Shabani, I.** (2022). Big data analytics in Cloud computing: an overview, *Journal of Cloud Computing*, *11*(1), 24.

[192] **Kim, H.**, **Kim, J.**, **Kim, Y.**, **Kim, I. and Kim, K.J.** (2019). Design of network threat detection and classification based on machine learning on cloud computing, *Cluster Computing*, *22*, 2341–2350.

[193] **Belgaum, M.R.**, **Alansari, Z.**, **Musa, S.**, **Alam, M.M. and Mazliham, M.** (2021). Role of artificial intelligence in cloud computing, IoT and SDN: Reliability and scalability issues, *International Journal of Electrical and Computer Engineering*, *11*(5), 4458.

[194] **Rajabion, L.**, **Shaltooki, A.A.**, **Taghikhah, M.**, **Ghasemi, A. and Badfar, A.** (2019). Healthcare big data processing mechanisms: The role of cloud computing, *International Journal of Information Management*, *49*, 271–289.

[195] **Zhang, P.**, **Yu, K.**, **Yu, J.J. and Khan, S.U.** (2018). QuantCloud: Big Data Infrastructure for Quantitative Finance on the Cloud, *IEEE Transactions on Big Data*, *4*(3), 368–380.

[196] **Haroun, A.**, **Mostefaoui, A. and Dessables, F.** (2017). A Big Data Architecture for Automotive Applications: PSA Group Deployment Experience, *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pp.921–928.

[197] **Zeng, Y.**, **Ouyang, S.**, **Zhu, T. and Li, C.** (2022). E-Commerce Network Security Based on Big Data in Cloud Computing Environment, *Mobile Information Systems*, *2022*(1), 9935244.

[198] **Chen, J.**, **Li, K.**, **Rong, H.**, **Bilal, K.**, **Yang, N. and Li, K.** (2018). A disease diagnosis and treatment recommendation system based on big data mining and cloud computing, *Information Sciences*, *435*, 124–149.

[199] **Lahoura, V.**, **Singh, H.**, **Aggarwal, A.**, **Sharma, B.**, **Mohammed, M.A.**, **Damaševičius, R.**, **Kadry, S. and Cengiz, K.** (2021). Cloud Computing-Based Framework for Breast Cancer Diagnosis Using Extreme Learning Machine, *Diagnostics*, *11*(2), 241.

[200] **Cui, Q.**, **Wang, Y.**, **Chen, K.C.**, **Ni, W.**, **Lin, I.C.**, **Tao, X. and Zhang, P.** (2019). Big Data Analytics and Network Calculus Enabling Intelligent Management of Autonomous Vehicles in a Smart City, *IEEE Internet of Things Journal*, *6*(2), 2021–2034.

[201] **Ye, Z. and Ying, R.** (2024). An AI-aware Orchestration Framework for Cloud-based LLM Workloads, *2024 IEEE 10th International Conference on Edge Computing and Scalable Cloud (EdgeCom)*, pp.22–24.

[202] **Stergiou, C.**, **Psannis, K.E.**, **Kim, B.G. and Gupta, B.** (2018). Secure integration of IoT and Cloud Computing, *Future Generation Computer Systems*, *78*, 964–975.

[203] **Celesti, A.**, **Lay-Ekuakille, A.**, **Wan, J.**, **Fazio, M.**, **Celesti, F.**, **Romano, A.**, **Bramanti, P. and Villari, M.** (2020). Information management in IoT cloud-based tele-rehabilitation as a service for smart cities: Comparison of NoSQL approaches, *Measurement*, *151*, 107218.

[204] **Tahir, A.**, **Chen, F.**, **Khan, H.U.**, **Ming, Z.**, **Ahmad, A.**, **Nazir, S. and Shafiq, M.** (2020). A Systematic Review on Cloud Storage Mechanisms Concerning e-Healthcare Systems, *Sensors*, *20*(18), 5392.

[205] **Tan, M. and Su, X.** (2011). Media cloud: When media revolution meets rise of cloud computing, *Proceedings of 2011 IEEE 6th International Symposium on Service Oriented System (SOSE)*, pp.251–261.

[206] **Gadea, C.**, **Solomon, B.**, **Ionescu, B. and Ionescu, D.** (2011). A Collaborative Cloud-Based Multimedia Sharing Platform for Social Networking Environments, *2011 Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN)*, pp.1–6.

[207] **Chard, K.**, **Caton, S.**, **Rana, O. and Bubendorfer, K.** (2010). Social Cloud: Cloud Computing in Social Networks, *2010 IEEE 3rd International Conference on Cloud Computing*, pp.99–106.

[208] **Shea, R.**, **Liu, J.**, **Ngai, E.C.H. and Cui, Y.** (2013). Cloud gaming: architecture and performance, *IEEE Network*, *27*(4), 16–21.

[209] **Wang, Q.**, **Xu, K.**, **Izard, R.**, **Kribbs, B.**, **Porter, J.**, **Wang, K.C.**, **Prakash, A. and Ramanathan, P.** (2014). GENI Cinema: An SDN-Assisted Scalable Live Video Streaming Service, *2014 IEEE 22nd International Conference on Network Protocols*, pp.529–532.

[210] **Lee, J.H.**, **Wishkoski, R.**, **Aase, L.**, **Meas, P. and Hubbles, C.** (2017). Understanding users of cloud music services: Selection factors, management and access behavior, and perceptions, *Journal of the Association for Information Science and Technology*, *68*(5), 1186–1200.

[211] **Baldassarre, M.T.**, **Caivano, D.**, **Dimauro, G.**, **Gentile, E. and Visaggio, G.** (2018). Cloud Computing for Education: A Systematic Mapping Study, *IEEE Transactions on Education*, *61*(3), 234–244.

[212] **Han, H. and Trimi, S.** (2022). Cloud Computing-based Higher Education Platforms during the COVID-19 Pandemic, IC4E '22, Association for Computing Machinery, New York, NY, USA, pp.83–89.

[213] **Shatalova, E.P. and Huseynov, R.M.** (2021). Cloud Technologies in Banking, Springer International Publishing, Cham, pp.41–48.

[214] **Qian, L.**, **Luo, Z.**, **Du, Y. and Guo, L.** (2009). Cloud computing: An overview, *IEEE International Conference on Cloud Computing*, Springer, pp.626–631.

[215] **Shi, W.**, **Cao, J.**, **Zhang, Q.**, **Li, Y. and Xu, L.** (2016). Edge computing: Vision and challenges, *IEEE internet of things journal*, *3*(5), 637–646.

[216] **Yu, W.**, **Liang, F.**, **He, X.**, **Hatcher, W.G.**, **Lu, C.**, **Lin, J. and Yang, X.** (2017). A survey on the edge computing for the Internet of Things, *IEEE access*, *6*, 6900–6919.

[217] **Ananthanarayanan, G.**, **Bahl, P.**, **Bodík, P.**, **Chintalapudi, K.**, **Philipose, M.**, **Ravindranath, L. and Sinha, S.** (2017). Real-time video analytics: The killer app for edge computing, *computer*, *50*(10), 58–67.

[218] **Barthélemy, J.**, **Verstaevel, N.**, **Forehead, H. and Perez, P.** (2019). Edge-computing video analytics for real-time traffic monitoring in a smart city, *Sensors*, *19*(9), 2048.

[219] **Wang, J.**, **Feng, Z.**, **Chen, Z.**, **George, S.**, **Bala, M.**, **Pillai, P.**, **Yang, S.W. and Satyanarayanan, M.** (2018). Bandwidth-efficient live video analytics for drones via edge computing, *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, IEEE, pp.159–173.

[220] **Zhang, Q.**, **Sun, H.**, **Wu, X. and Zhong, H.** (2019). Edge video analytics for public safety: A review, *Proceedings of the IEEE*, *107*(8), 1675–1696.

[221] **Hu, L.**, **Miao, Y.**, **Wu, G.**, **Hassan, M.M. and Humar, I.** (2019). iRobot-Factory: An intelligent robot factory based on cognitive manufacturing and edge computing, *Future Generation Computer Systems*, *90*, 569–577.

[222] **Chen, B.**, **Wan, J.**, **Celesti, A.**, **Li, D.**, **Abbas, H. and Zhang, Q.** (2018). Edge computing in IoT-based manufacturing, *IEEE Communications Magazine*, *56*(9), 103–109.

[223] **Wang, H.**, **Gong, J.**, **Zhuang, Y.**, **Shen, H. and Lach, J.** (2017). Healthedge: Task scheduling for edge computing with health emergency and human behavior consideration in smart homes, *2017 IEEE International Conference on Big Data (Big Data)*, IEEE, pp.1213–1222.

[224] **Oueida, S.**, **Kotb, Y.**, **Aloqaily, M.**, **Jararweh, Y. and Baker, T.** (2018). An edge computing based smart healthcare framework for resource management, *Sensors*, *18*(12), 4307.

[225] **Sodhro, A.H.**, **Luo, Z.**, **Sangaiah, A.K. and Baik, S.W.** (2019). Mobile edge computing based QoS optimization in medical healthcare applications, *International Journal of Information Management*, *45*, 308–318.

[226] **Dong, P.**, **Ning, Z.**, **Obaidat, M.S.**, **Jiang, X.**, **Guo, Y.**, **Hu, X.**, **Hu, B. and Sadoun, B.** (2020). Edge computing based healthcare systems: Enabling decentralized health monitoring in Internet of medical Things, *IEEE Network*, *34*(5), 254–261.

[227] **Luo, H.**, **Cai, H.**, **Yu, H.**, **Sun, Y.**, **Bi, Z. and Jiang, L.** (2019). A short-term energy prediction system based on edge computing for smart city, *Future Generation Computer Systems*, *101*, 444–457.

[228] **Petrovic, N. and Kocic, D.** (2019). Adopting linear optimization to support autonomous vehicles in smart city, *2019 27th Telecommunications Forum (TELFOR)*, IEEE, pp.1–4.

[229] **Zhou, S. and Zhang, L.** (2018). Smart home electricity demand forecasting system based on edge computing, *2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS)*, IEEE, pp.164–167.

[230] **Sahni, Y.**, **Cao, J. and Yang, L.** (2018). Data-aware task allocation for achieving low latency in collaborative edge computing, *IEEE Internet of Things Journal*, *6*(2), 3512–3524.

[231] **Tran, T.X.**, **Hajisami, A.**, **Pandey, P. and Pompili, D.** (2017). Collaborative mobile edge computing in 5G networks: New paradigms, scenarios, and challenges, *IEEE Communications Magazine*, *55*(4), 54–61.

[232] **Bonomi, F.**, **Milito, R.**, **Zhu, J. and Addepalli, S.** (2012). Fog Computing and Its Role in the Internet of Things, *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, Association for Computing Machinery, pp.13–16.

[233] **Minh, Q.T.**, **Tran, C.M.**, **Le, T.A.**, **Nguyen, B.T.**, **Tran, T.M. and Balan, R.K.** (2018). FogFly: A Traffic Light Optimization Solution Based on Fog Computing, *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers*, UbiComp '18, Association for Computing Machinery, pp.1130–1139.

[234] **Tang, C.**, **Xia, S.**, **Zhu, C. and Wei, X.** (2019). Phase Timing Optimization for Smart Traffic Control Based on Fog Computing, *IEEE Access*, *7*, 84217–84228.

[235] **Jang, H.C. and Lin, T.K.** (2018). Traffic-Aware Traffic Signal Control Framework Based on SDN and Cloud-Fog Computing, *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, pp.1–5.

[236] **Serdaroglu, K.C.**, **Baydere, c.**, **Saovapakhiran, B. and Charnsripinyo, C.** (2023). Location Aware Fog Computing Based Air Quality Monitoring System, *2023 International Conference on Smart Applications, Communications and Networking (SmartNets)*, pp.1–6.

[237] **Aliyu, F.**, **Abdeen, M.A.R.**, **Sheltami, T.**, **Alfraidi, T. and Ahmed, M.H.** (2023). Fog computing-assisted path planning for smart shopping, *Multimedia Tools and Applications*, *82*, 38827–38852.

[238] **Talaat, F.M. and ZainEldin, H.** (2023). An improved fire detection approach based on YOLO-v8 for smart cities, *Neural Computing and Applications*, *35*(28), 20939–20954.

[239] **Sodhro, A.H.**, **Sodhro, G.H.**, **Guizani, M.**, **Pirbhulal, S. and Boukerche, A.** (2020). AI-Enabled Reliable Channel Modeling Architecture for Fog Computing Vehicular Networks, *IEEE Wireless Communications*, *27*(2), 14–21.

[240] **Zhang, Y.**, **Zhang, H.**, **Long, K.**, **Zheng, Q. and Xie, X.** (2018). Software-Defined and Fog-Computing-Based Next Generation Vehicular Networks, *IEEE Communications Magazine*, *56*(9), 34–41.

[241] **Ning, Z.**, **Huang, J. and Wang, X.** (2019). Vehicular Fog Computing: Enabling Real-Time Traffic Management for Smart Cities, *IEEE Wireless Communications*, *26*(1), 87–93.

[242] **Wang, H.**, **Gong, Y.**, **Ding, Y.**, **Tang, S. and Wang, Y.** (2023). Privacy-Preserving Data Aggregation with Dynamic Billing in Fog-Based Smart Grid, *Applied Sciences*, *13*(2), 748.

[243] **Jaiswal, R.**, **Davidrajuh, R. and Wondimagegnehu, S.M.** (2021). Fog Computing for Efficient Predictive Analysis in Smart Grids, *Proceedings of the International Conference on Artificial Intelligence and Its Applications*, icARTi '21, Association for Computing Machinery, pp.1 – 6.

[244] **Forcan, M. and Maksimović, M.** (2020). Cloud-Fog-based approach for Smart Grid monitoring, *Simulation Modelling Practice and Theory*, *101*, 101988.

[245] **Li, Z.**, **Liu, Y.**, **Xin, R.**, **Gao, L.**, **Ding, X. and Hu, Y.** (2019). A Dynamic Game Model for Resource Allocation in Fog Computing for Ubiquitous Smart Grid, *2019 28th Wireless and Optical Communications Conference (WOCC)*, pp.1–5.

[246] **Silva, F.A.**, **Fé, I.**, **Brito, C.**, **Araújo, G.**, **Feitosa, L.**, **Choi, E.**, **Min, D. and Nguyen, T.A.** (2022). Supporting availability evaluation of a smart building monitoring system aided by fog computing, *Electronics Letters*, *58*(12), 471–473.

[247] **Bhatia, M.** (2020). Fog Computing-inspired Smart Home Framework for Predictive Veterinary Healthcare, *Microprocessors and Microsystems*, *78*, 103227.

[248] **Gill, S.S.**, **Garraghan, P. and Buyya, R.** (2019). ROUTER: Fog enabled cloud based intelligent resource management approach for smart home IoT devices, *Journal of Systems and Software*, *154*, 125–138.

[249] **Hassen, H.B.**, **Dghais, W. and Hamdi, B.** (2019). An E-health system for monitoring elderly health based on Internet of Things and Fog computing, *Health Information Science and Systems*, *7*(1), 24.

[250] **Kamruzzaman, M.**, **Alanazi, S.**, **Alruwaili, M.**, **Alrashdi, I.**, **Alhwaiti, Y. and Alshammari, N.** (2022). Fuzzy-assisted machine learning framework for the fog-computing system in remote healthcare monitoring, *Measurement*, *195*, 111085.

[251] **Arunkumar, P.M.**, **Masud, M.**, **Aljahdali, S. and Abouhawwash, M.** (2023). Healthcare Monitoring Using Ensemble Classifiers in Fog Computing Framework, *Computer Systems Science and Engineering*, *45*(2), 2265–2280.

[252] **Almas, A.**, **Iqbal, W.**, **Altaf, A.**, **Saleem, K.**, **Mussiraliyeva, S. and Iqbal, M.W.** (2023). Context-Based Adaptive Fog Computing Trust Solution for Time-Critical Smart Healthcare Systems, *IEEE Internet of Things Journal*, *10*(12), 10575–10586.

[253] **Beri, R.**, **Dubey, M.K.**, **Gehlot, A.**, **Singh, R.**, **Abd-Elnaby, M. and Singh, A.** (2021). A novel fog-computing-assisted architecture of E-healthcare system for pregnant women, *The Journal of Supercomputing*, *78*(6), 7591–7615.

[254] **Klonoff, D.C.** (2017). Fog Computing and Edge Computing Architectures for Processing Data From Diabetes Devices Connected to the Medical Internet of Things, *Journal of Diabetes Science and Technology*, *11*(4), 647–652.

[255] **Monteiro, A.**, **Dubey, H.**, **Mahler, L.**, **Yang, Q. and Mankodiya, K.** (2016). Fit: A Fog Computing Device for Speech Tele-Treatments, *2016 IEEE International Conference on Smart Computing (SMARTCOMP)*, pp.1–3.

[256] **Paul, A.**, **Pinjari, H.**, **Hong, W.H.**, **Seo, H.C. and Rho, S.** (2018). Fog Computing-Based IoT for Health Monitoring System, *Journal of Sensors*, *2018*(1), 1386470.

[257] **Mani, N.**, **Singh, A. and Nimmagadda, S.L.** (2020). An IoT Guided Healthcare Monitoring System for Managing Real-Time Notifications by Fog Computing Services, *Procedia Computer Science*, *167*, 850–859.

[258] **Moghadas, E.**, **Rezazadeh, J. and Farahbakhsh, R.** (2020). An IoT patient monitoring based on fog computing and data mining: Cardiac arrhythmia usecase, *Internet of Things*, *11*, 100251.

[259] **Jeevan Kharel, H.T.R. and Shin, S.Y.** (2019). Fog Computing-Based Smart Health Monitoring System Deploying LoRa Wireless Communication, *IETE Technical Review*, *36*(1), 69–82.

[260] **Ijaz, M.**, **Li, G.**, **Wang, H.**, **El-Sherbeeny, A.M.**, **Moro Awelisah, Y.**, **Lin, L.**, **Koubaa, A. and Noor, A.** (2020). Intelligent Fog-Enabled Smart Healthcare System for Wearable Physiological Parameter Detection, *Electronics*, *9*(12), 2015.

[261] **Tuli, S.**, **Mahmud, R.**, **Tuli, S. and Buyya, R.** (2019). FogBus: A Blockchain-based Lightweight Framework for Edge and Fog Computing, *Journal of Systems and Software*, *154*, 22–36.

[262] **Tuli, S.**, **Basumatary, N.**, **Gill, S.S.**, **Kahani, M.**, **Arya, R.C.**, **Wander, G.S. and Buyya, R.** (2020). HealthFog: An ensemble deep learning based Smart Healthcare System for Automatic Diagnosis of Heart Diseases in integrated IoT and fog computing environments, *Future Generation Computer Systems*, *104*, 187–200.

[263] **Medina, J.**, **Espinilla, M.**, **Zafra, D.**, **Martínez, L. and Nugent, C.** (2017). Fuzzy Fog Computing: A Linguistic Approach for Knowledge Inference in Wearable Devices, ***S.F. Ochoa**, **P. Singh and J. Bravo**, editors, Ubiquitous Computing and Ambient Intelligence*, Springer International Publishing, Cham, pp.473–485.

[264] **Constant, N.**, **Borthakur, D.**, **Abtahi, M.**, **Dubey, H. and Mankodiya, K.** (2017). Fog-Assisted wIoT: A Smart Fog Gateway for End-to-End Analytics in Wearable Internet of Things, *CoRR*, *abs/1701.08680*, `1701.08680`.

[265] *Resisitve Flex Sensors - Spectra Symbol*, `https://www.spectrasymbol.com/resistive-flex-sensors`, [Accessed 03-01-2024].

[266] **Wu, W.**, **Pirbhulal, S.**, **Sangaiah, A.K.**, **Mukhopadhyay, S.C. and Li, G.** (2018). Optimization of signal quality over comfortability of textile electrodes for ECG monitoring in fog computing based medical applications, *Future Generation Computer Systems*, *86*, 515–526.

[267] **Veness, C.** *Calculate distance and bearing between two Latitude/Longitude points using haversine formula in JavaScript*, `https://www.movable-type.co.uk/scripts/latlong.html`, [Accessed 13-09-2023].

[268] **Coffey, J.** (2017). *Latency in optical fiber systems*, `https://www.commscope.com/globalassets/digizuite/2799-latency-in-optical-fiber-systems-wp-111432-en.pdf`.

[269] **Gamma, E.** (1995). *Design patterns*, Pearson Education India.

[270] **Ruys, W.**, **Lee, H.**, **You, B.**, **Talati, S.**, **Park, J.**, **Almgren-Bell, J.**, **Yan, Y.**, **Fernando, M.**, **Biros, G.**, **Erez, M.**, **Burtscher, M.**, **Rossbach, C.J.**, **Pingali, K. and Gligoric, M.** (2024). A Deep Dive into Task-Based Parallelism in Python, *2024 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pp.1147–1149.

[271] **Sethi, D.**, **Bharti, S. and Prakash, C.** (2022). A comprehensive survey on gait analysis: History, parameters, approaches, pose estimation, and future work, *Artificial Intelligence in Medicine*, *129*, 102314.

[272] **Han, Y.C.**, **Wong, K.I. and Murray, I.** (2019). Gait Phase Detection for Normal and Abnormal Gaits Using IMU, *IEEE Sensors Journal*, *19*(9), 3439–3448.

[273] **Romijnders, R.**, **Warmerdam, E.**, **Hansen, C.**, **Welzel, J.**, **Schmidt, G. and Maetzler, W.** (2021). Validation of IMU-based gait event detection during curved walking and turning in older adults and parkinson's disease patients, *Journal of NeuroEngineering and Rehabilitation*, *18*(1), 28.

[274] **Negi, S.**, **Sharma, S. and Sharma, N.** (2021). FSR and IMU sensors-based human gait phase detection and its correlation with EMG signal for different terrain walk, *Sensor Review*, *41*(3), 235–245.

[275] **Gujarathi, T. and Bhole, K.** (2019). GAIT ANALYSIS USING IMU SENSOR, *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pp.1–5.

[276] **Yu, Y.**, **Si, X.**, **Hu, C. and Zhang, J.** (2019). A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures, *Neural Computation*, *31*(7), 1235–1270.

[277] **Wang, K.**, **Kong, S.**, **Chen, X. and Zhao, M.** (2024). Edge Computing Empowered Smart Healthcare: Monitoring and Diagnosis with Deep Learning Methods, *Journal of Grid Computing*, *22*(1), 30.

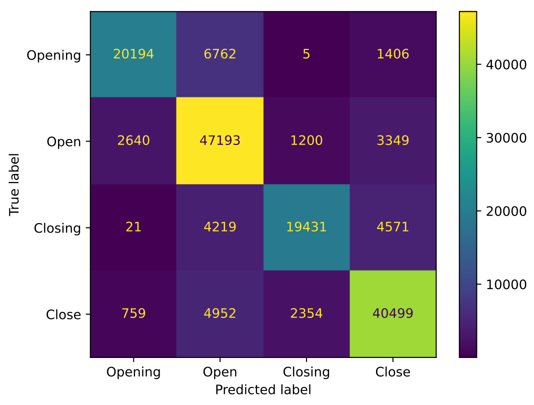
[278] **Ozlem, K.**, **Atalay, A.T.**, **Atalay, O. and Ince, G.** (2024). FogETex: Fog Computing Framework for Electronic Textile Applications, *IEEE Internet of Things Journal*, Early Access.

[279] *StandardScaler - scikit-learn.org*, `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html`, [Accessed 25-08-2024].

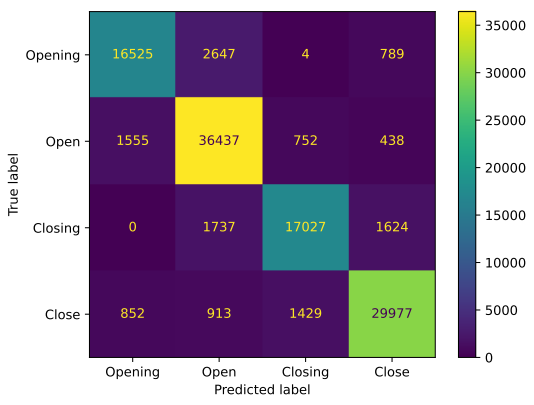
[280] **Milcic, D.**, **Vucina, A. and Bosnjak, M.** (2023). Ergonomic Design of the Hand Saw Handle, ***I. Salopek Čubrić, G. Čubrić, K. Jambrošić, T. Jurčević Lulić and D. Sumpor****, editors, Proceedings of the 9th International Ergonomics Conference*, Springer Nature Switzerland, Cham, pp.231–239.

APPENDICES

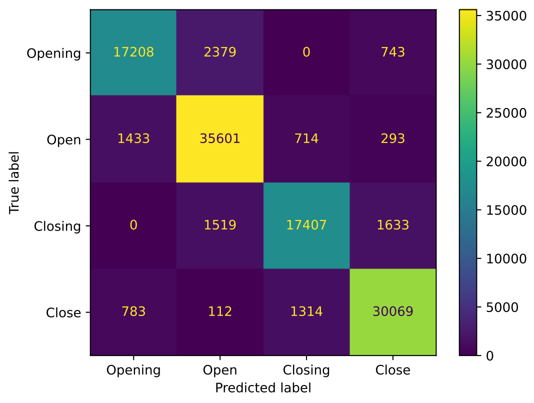
APPENDIX A : Assistive Soft Robotic Glove Control Supporting Information



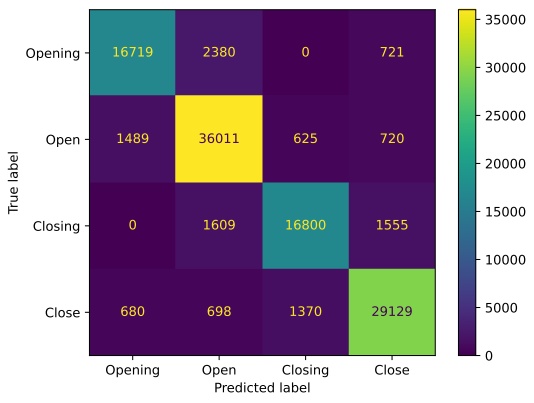
(a) Thumb Finger.



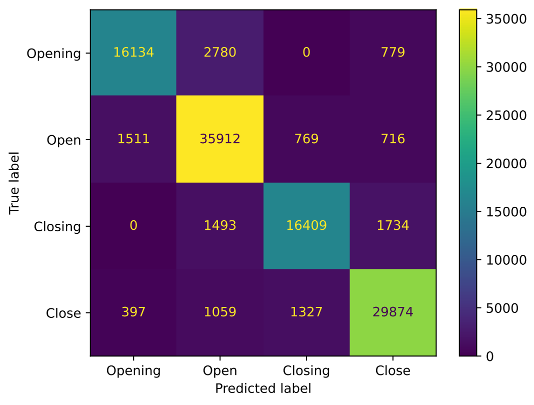
(b) Index Finger.



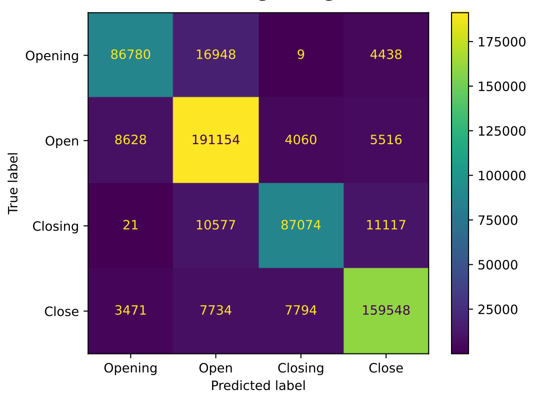
(c) Middle Finger.



(d) Ring Finger.

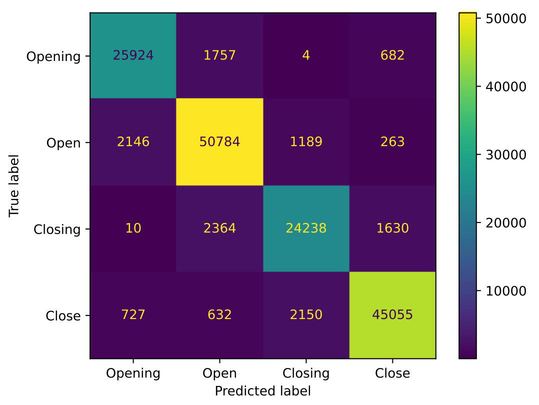


(e) Pinkie Finger.

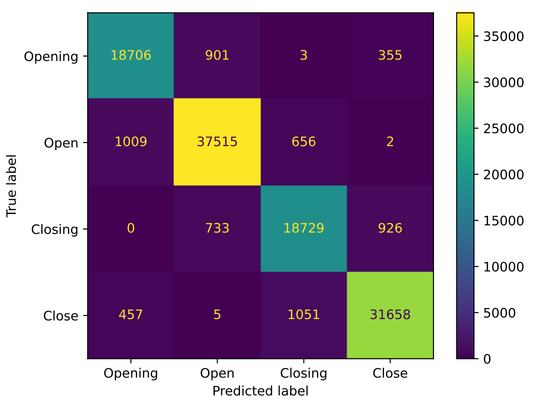


(f) Overall Results.

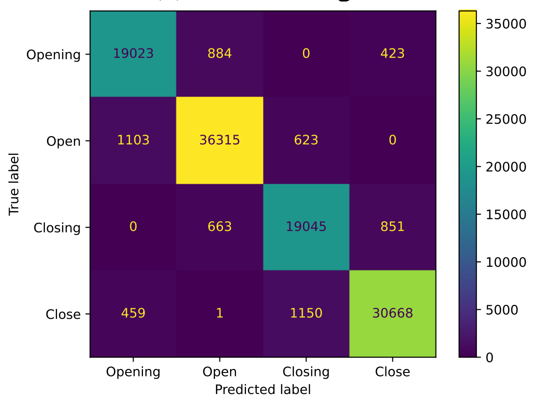
Figure A.1 : Confusion Matrices of Logistic Regression Classifier for Different Fingers.



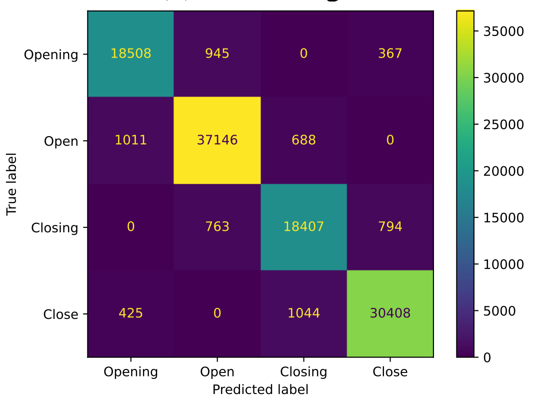
(a) Thumb Finger.



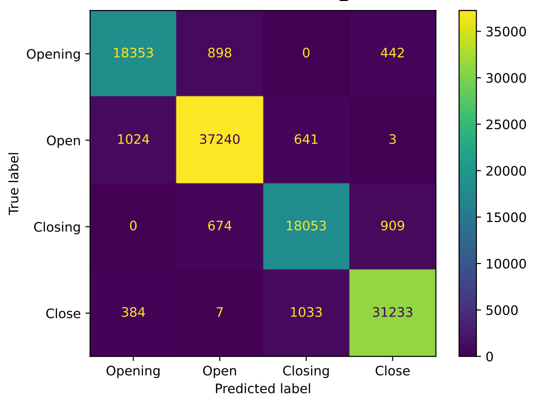
(b) Index Finger.



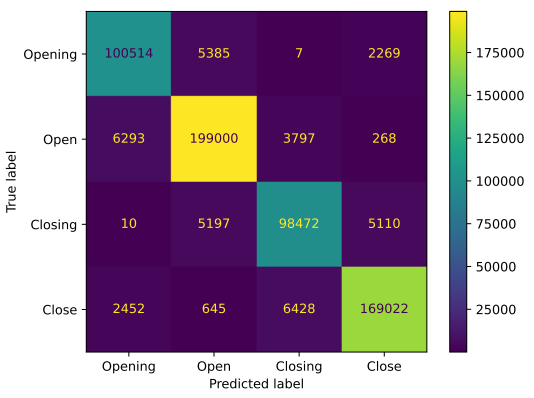
(c) Middle Finger.



(d) Ring Finger.

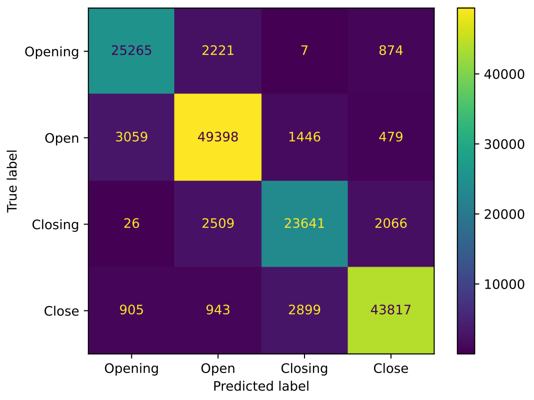


(e) Pinkie Finger.

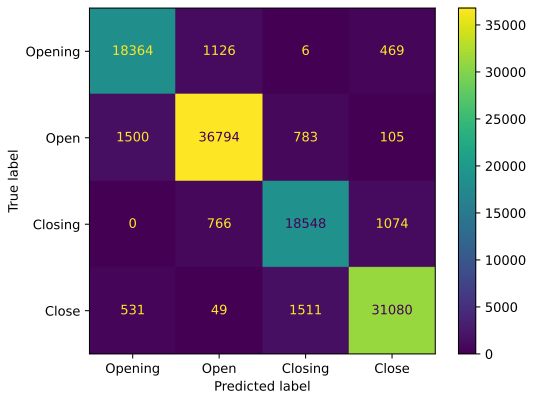


(f) Overall Results.

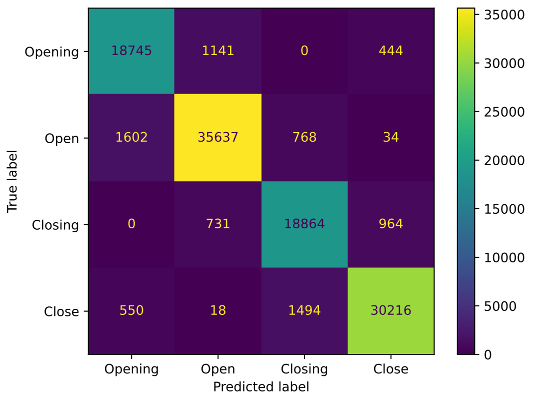
Figure A.2 : Confusion Matrices of Decision Tree Classifier for Different Fingers.



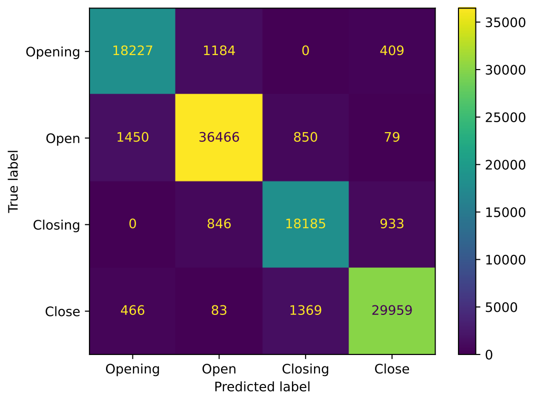
(a) Thumb Finger.



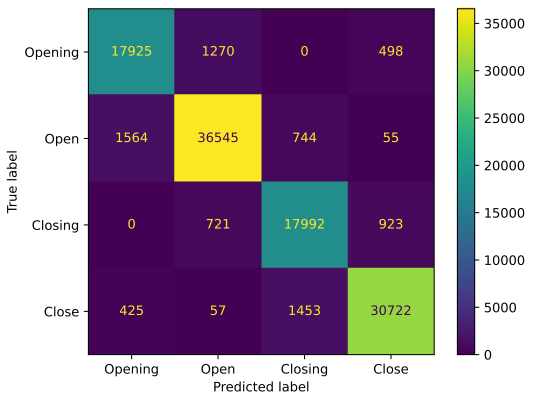
(b) Index Finger.



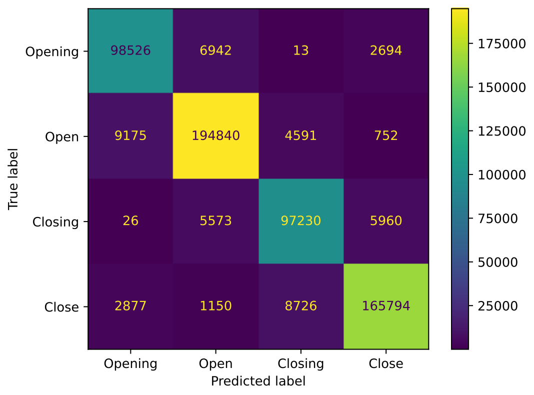
(c) Middle Finger.



(d) Ring Finger.

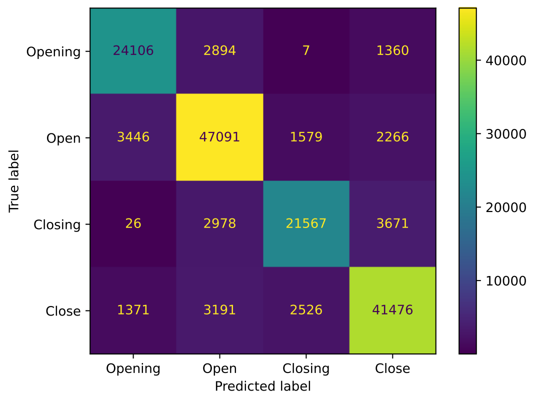


(e) Pinkie Finger.

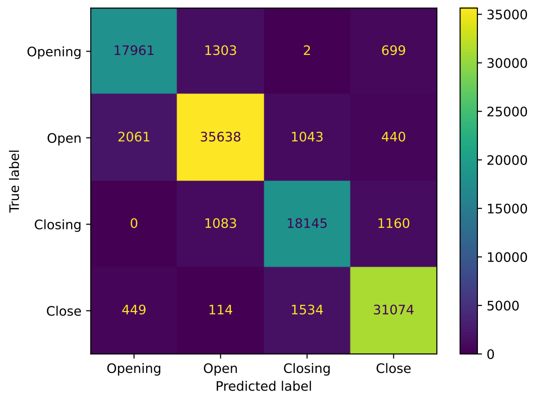


(f) Overall Results.

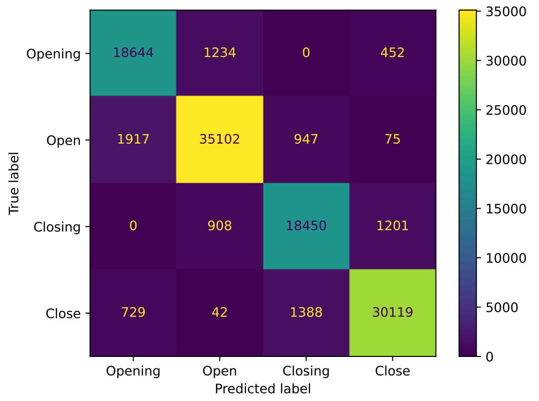
Figure A.3 : Confusion Matrices of K-Nearest Neighbors Classifier for Different Fingers.



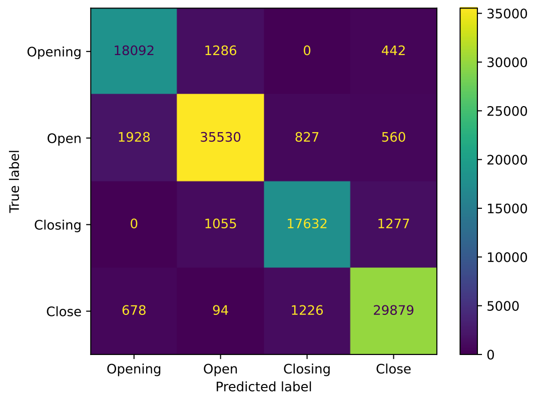
(a) Thumb Finger.



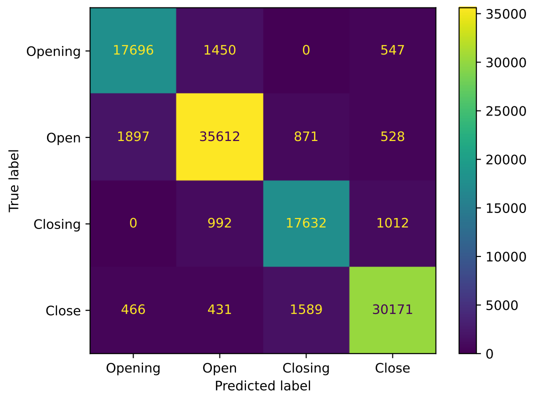
(b) Index Finger.



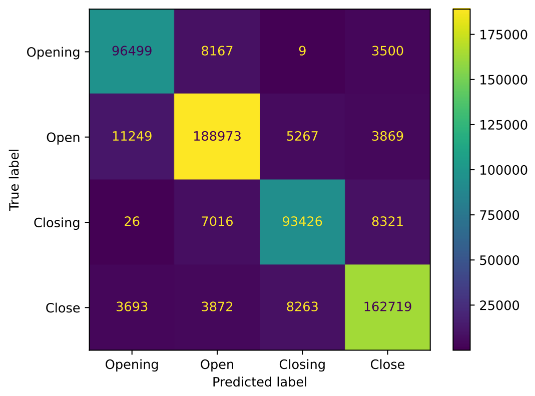
(c) Middle Finger.



(d) Ring Finger.

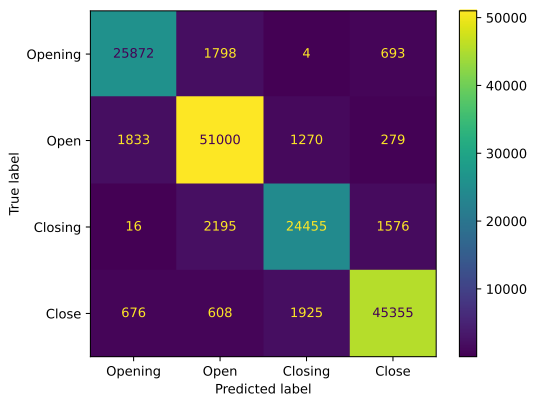


(e) Pinkie Finger.

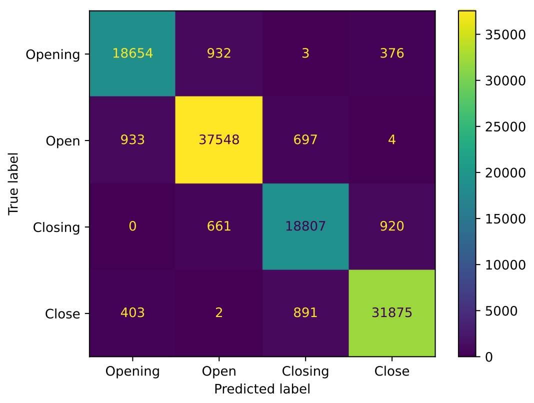


(f) Overall Results.

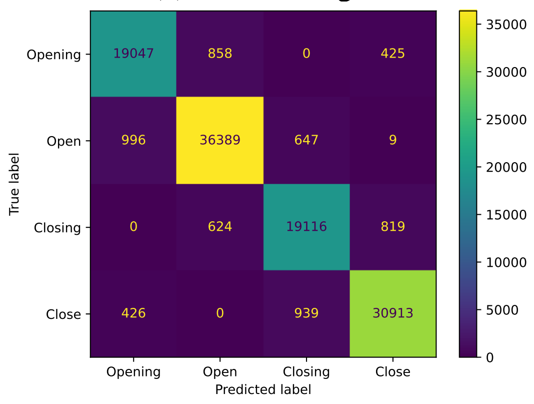
Figure A.4 : Confusion Matrices of Multi-layer Perceptron Classifier for Different Fingers.



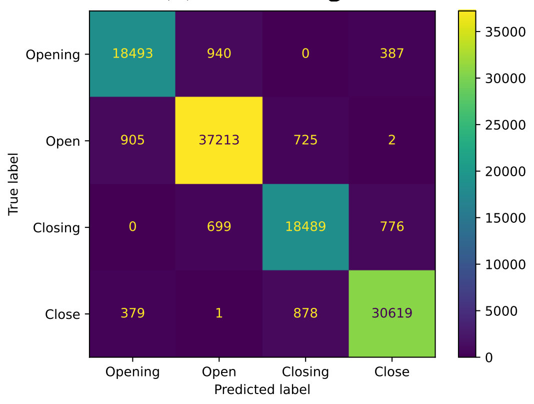
(a) Thumb Finger.



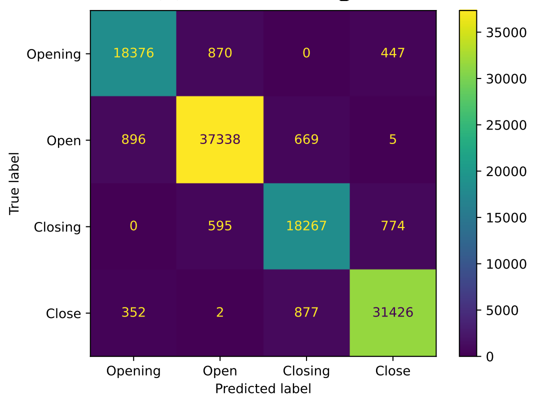
(b) Index Finger.



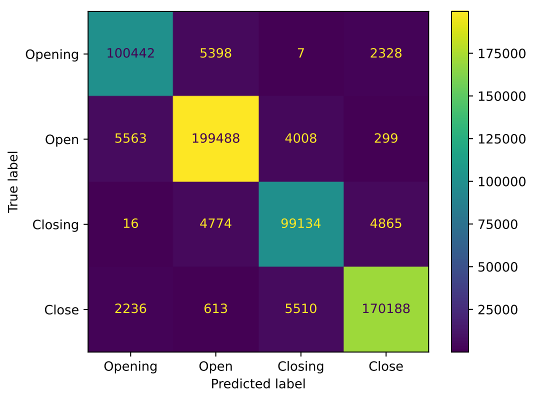
(c) Middle Finger.



(d) Ring Finger.



(e) Pinkie Finger.



(f) Overall Results.

Figure A.5 : Confusion Matrices of XGBoost Classifier for Different Fingers.

CURRICULUM VITAE

Name SURNAME: Kadir ÖZLEM

EDUCATION:

- **M.Sc.:** 2018, Istanbul Technical University, Graduate School of Science Engineering and Technology, Computer Engineering
- **B.Sc.:** 2016, Istanbul University, Engineering Faculty, Computer Engineering (Double Major)
- **B.Sc.:** 2016, Istanbul University, Engineering Faculty, Electrical-Electronics Engineering
- **B.Sc.:** 2016, Anadolu University, Faculty of Business Administration, Business Administration

PROFESSIONAL EXPERIENCE AND REWARDS:

- 2018-... Research Assistant on Computer Engineering Department, Istanbul Technical University
- 2016-2018 Pinnera - Software Engineer

PUBLICATIONS, PRESENTATIONS AND PATENTS ON THE THESIS:

- **Ozlem, K.**, Gumus, C., Yilmaz, A.F., Atalay, A. T., Atalay, O., Ince, G. Cloud-based Control System with Sensing and Actuating Textile-based IoT Gloves for Telerehabilitation Applications. *Advanced Intelligent Systems,* doi: 10.1002/aisy.202400894, to appear.
- **Ozlem, K.**, Atalay, A. T., Atalay, O., Ince, G. FogETex: Fog Computing Framework for Electronic Textile Applications. *IEEE Internet of Things Journal,* doi: 10.1109/JIOT.2024.3490981, to appear.
- Erzurumluoglu, O.F., **Ozlem, K.**, Atalay, A. T., Atalay, O., Ince, G. Fog Computing-based Real-Time Emotion Recognition using Physiological Signals, *Proceedings of International Conference on Advanced Communication Technology (ICACT2025)*, to appear.
- **Özlem, K.**, Kuyucu, M. K., Bahtiyar, Ş., İnce, G. (2019, September). Security and Privacy Issues for E-textile Applications. *2019 4th International Conference on Computer Science and Engineering (UBMK)*, September 11-15, 2019, Samsun, Turkey, pp. 102-107.

OTHER PUBLICATIONS, PRESENTATIONS AND PATENTS:

- Celik, I., Cetin, M.S., **Ozlem, K.**, Atalay, O., Atalay, A.T., Ince, G. (2024). Gesture Recognition on Textile-Based Pressure Sensor Array. 2024 *5th International Conference in Electronic Engineering, Information Technology & Education (EEITE)*, May 29-31, 2024, Chania, Greece, 2024, pp. 1-5.

- Yilmaz, A.F., **Ozlem, K.**, Celebi, M.F., Taherkhani, B., Kalaoglu, F., Atalay, A., Ince G., Atalay, O. (2024), Design and Scalable Fast Fabrication of Biaxial Fabric Pouch Motors for Soft Robotic Artificial Muscle Applications, *Advanced Intelligent Systems*, 6(8), 2300888.

- Atalay, O., **Ozlem, K.**, Gumus, G., Ahmed, I.A.K., Yilmaz, A.F., Celebi, M.F., Cetin, M.S., Taherkhani, B., Atalay, A.T., Ince, G. (2024), Thermally Driven 3D Seamless Textile Actuators for Soft Robotic Applications, *Advanced Intelligent Systems*, 6(11), 2400133.

- Yunculer, I., Al-Azzawi, N. Ayvaz, U., Cetin, M. S., **Ozlem, K.**, Atalay, A. T., Ince, G., Atalay, O. (2024). A Seamless T-shirt Design with Textile-based ECG Electrodes and Posture Monitoring Sensors, *Association of Universities for Textiles Conference (AUTEX 2024)*, June 17-9, 2024, Liberec, Czech Republic.

- Erzurumluoglu, O.F., **Ozlem, K.**, Tunc, H., Gumus, C., Khalilbayli, F., Buyukaslan, A., Yilmaz, H., Tuncay Atalay, A., Atalay, O., Ince, G. (2024). A Pressure Monitoring for Scoliosis Braces using Textile-based Pressure Sensor Arrays. *2023 12th HCist - International Conference on Health and Social Care Information Systems and Technologies, Procedia Computer Science*, November 08-10, 2023, Porto, Portugal, 239, 1409-1416.

- Yilmaz, A. F., **Ozlem, K.**, Khalilbayli, F., Celebi, M. F., Kalaoglu, F., Atalay, A. T., Ince, G., Atalay, O. (2023). Resistive Self-Sensing Controllable Fabric-Based Actuator: A Novel Approach to Creating Anisotropy. *Advanced Sensor Research,* 3(7), 2300108.

- Ayvaz, U., **Ozlem, K.**, Yilmaz, A. F., Atalay, A. T., Atalay, O., Ince, G. (2024). Real-time Stride Length Estimation using Textile-Based Capacitive Soft Strain Sensors. *IEEE Transactions on Instrumentation and Measurement,* 73, pp. 1-11.

- Yilmaz, A. F., Ahmed, I. A. K., Gumus, C., **Ozlem, K.**, Cetin, M. S., Atalay, A. T., Ince, G., Atalay, O. (2023). Highly Stretchable Textile Knitted Interdigital Sensor for Wearable Technology Applications. *Advanced Sensor Research,* 3(2), 2300121.

- Gumus, C., **Ozlem, K.**, Khalilbayli, F., Atalay, A. T., Onal, E., Ince, G., Atalay, O., Erzurumluoglu, O. F. (2023). Textile-based large-area pressure sensing arrays (Tekstil Tabanlı Geniş Alanlı Basınç Algılama Dizileri). Patent Number: TR2022/004079 and WO2023177380A1.

- Pazar, A., Khalilbayli, F., **Ozlem, K.**, Yilmaz, A. F., Atalay, A. T., Atalay, O., İnce, G. (2022). Gait Phase Recognition using Textile-based Sensor. *2022 7th International Conference on Computer Science and Engineering (UBMK)*, September 14-16, 2022, Diyarbakir, Turkey, pp. 1-6.

- Elmoughni, H. M., Atalay, O., **Ozlem, K.**, Menon, A. K. (2022). Thermoelectric Clothing for Body Heat Harvesting and Personal Cooling: Design and Fabrication of a Textile-Integrated Flexible and Vertical Device. *Energy Technology*, 10(10), 2200528.

- Yilmaz, A. F., Khalilbayli, F., **Ozlem, K.**, Elmoughni, H. M., Kalaoglu, F., Atalay, A. T., Ince, G., Atalay, O. (2022). Effect of Segment Types on Characterization of Soft Sensing Textile Actuators for Soft Wearable Robots. *Biomimetics,* 7(4), 249.

- Yilmaz, A. F., **Ozlem, K.**, Khalilbayli, F., Elmoughni, H. M., Atalay, A. T., Ince, G., Atalay, O. (2022). Actuators for soft robotic applications (Tekstil Tabanlı Geniş Alanlı Basınç Algılama Dizileri). Patent Number: TR2021/007340 and WO2023177380A1.

- Yilmaz, A. F., **Ozlem, K.**, Cetin, M. S., Atalay, A. T., Ince, G., Atalay, O. (2022). Knitted Interdigital Capacitive Strain Sensor for Wearable Applications, *Association of Universities for Textiles Conference (AUTEX 2022)*, June 7-10, 2022, Lodz, Poland.

- Gumus, C., **Ozlem, K.**, Khalilbayli, F., Erzurumluoglu, O. F., Ince, G., Atalay, O., Atalay, A. T. (2022). Textile-based pressure sensor arrays: A novel scalable manufacturing technique. *Micro and Nano Engineering*, 15, 100140.

- Yilmaz, A. F., **Ozlem, K.**, Elmoughni, H., Cappello, L., Atalay, A. T., Ince, G., Atalay, O. (2022). A Textile-based, Sensorized Pneumatic Actuator for Soft-robotics Applications, *Association of Universities for Textiles Conference (AUTEX 2021)*, September 5-9, 2021, Guimarães, Portugal.

- Elmoughni, H. M., Yilmaz, A. F., **Ozlem, K.**, Khalilbayli, F., Cappello, L., Atalay, A. T., Ince, G., Atalay, O. (2021). Machine-Knitted Seamless Pneumatic Actuators for Soft Robotics: Design, Fabrication, and Characterization. *Actuators*, 10(5), 94.

- Paket, E., **Ozlem, K.**, Elmoughni, H., Atalay, A., Atalay, O., Ince, G. (2020). ECG Monitoring System Using Textile Electrodes. *2020 28th Signal Processing and Communications Applications Conference (SIU)*, October 05-07, 2020, Gaziantep, Turkey, pp. 1-4.

- Sevinc, H., Ayvaz, U., **Ozlem, K.**, Elmoughni, H., Atalay, A., Atalay, O., Ince, G. (2020). Step Length Estimation Using Sensor Fusion. *2020 IEEE International Conference on Flexible and Printable Sensors and Systems (FLEPS)*, August 16-19, 2020, Manchester, United Kingdom, pp. 1-4.

- Kuyucu, C. F., Ayvaz, U., **Özlem, K.**, Atalay, A., Atalay, Ö., İnce, G. (2019, September). Comparative Assessment of Knee Motion Monitoring Technologies. *2019 4th International Conference on Computer Science and Engineering (UBMK)*, September 11-15, 2019, Samsun, Turkey, pp. 155-160.

- **Ozlem, K.**, Atalay, O., Atalay, A. and Ince, G. (2019). Textile Based Sensing System for Lower Limb Motion Monitoring, **L. Masia, S. Micera, M. Akay and J.L. Pons**, *editors*, *Converging Clinical and Engineering Research on Neurorehabilitation III*, Springer International Publishing, Cham, pp.395–399.