IOT-EDGE-SERVER BASED EMBEDDED SYSTEM FOR WIDE-RANGE HABITATS

by

Archit Gajjar, B.Tech.

THESIS

Presented to the Faculty of

The University of Houston-Clear Lake

In Partial Fulfillment

Of the Requirements

For the Degree

MASTER OF SCIENCE

in Computer Engineering

THE UNIVERSITY OF HOUSTON-CLEAR LAKE

MAY, 2019

IOT-EDGE-SERVER BASED EMBEDDED SYSTEM
FOR WIDE-RANGE HABITATS

by

Archit Gajjar

APPROVED BY

Xiaokun Yang, PhD, Chair

Hakduran Koc, PhD, Committee Member

Ishaq Unwala, PhD, Committee Member

APPROVED/RECEIVED BY THE COLLEGE OF SCIENCE AND ENGINEERING:

Said Bettayeb, PhD, Associate Dean

Ju H. Kim, PhD, Dean

Dedication

I would like to dedicate this thesis to my parents, family members, roommates, and professors. I would never have made it here without all of you.

Acknowledgments

First and foremost, I am grateful to my advisor, Dr. Xiaokun Yang, for being friendly, caring, supportive, and help in numerous ways. Without his support, I could not have done what I was able to do. He was very generous in sharing his experiences on electrical and computer engineering, academic life and beyond. He is not only my adviser but also, a friend inspiring me for the rest of my life.

Next, I would like to thank the members of my committee, Dr. Hakduran Koc and Dr. Ishaq Unwala for their support and suggestions in improving the quality of this dissertation. It is truly honored to have such great fantastic and knowledgeable professors serving as my committee members.

I would also like to thank all the lab mates and members at the Integrated Circut (IC) and Internet-of-Things (IoT) - IICOT Laboratory for creating an amazing working environment, and thank my friend, Yunxiang Zhang, for his assistance on work related to my research. I would also like to thank Shivang Dave for assisting in my research ventures.

Furthermore, I would also like to acknowledge the research support provided by the Department of Engineering at the University of Houston-Clear Lake.

Finally, I want to thank my father, Manish Gajjar, and mother, Shilpa Gajjar, for their unconditional love, support, faith, and encouragement. I am grateful to one of my roommates, Dharmik Nanavati, and his family members for being there for me during my ups and downs. Last but not least, I am appreciative to April Reed and Harold Schmoyer for their support.

ABSTRACT

IOT-EDGE-SERVER BASED EMBEDDED SYSTEM FOR WIDE-RANGE HABITATS

Archit Gajjar

University of Houston-Clear Lake, 2019

Thesis Chair: Xiaokun Yang, PhD

This thesis presents a hardware architecture, IoT-Edge-Server, of a diverse embedded system combining the applications of a smart city, smart building, or smart agricultural farm. First of all, we improve computation time by integrating the idea of edge computing on Raspberry Pi, CPU, and Field Programmable Gate Array (FPGA) which processes different algorithms. Second, the hardware processors are connected to a server which can manipulate the entire system and also possess storage capacity to save the systems important data and log files. More specifically, the hardware computes data from 1) a non-standardized Bluetooth Low Energy (BLE) Mesh System, and 2) a Security Monitoring System. The BLE Mesh System has one master and three slave devices while the Security Monitoring System has a Passive Infrared

Sensor (PIR) and a webcam to detect motion. Experimental results prove that using the phenomena of edge computing also known as fog computing demonstrates an improvement in computation speed and data privacy.

Although the results from the Raspberry Pi and general purpose CPU show drastic improvement, our expectations were more than that from the systems. To enhance it even further, we also propose third computing device which is a hardware accelerator. We present a synthesis of the well-known Viola-Jones face detection algorithm on Xilinx software and platform - Vivado and FPGA as Nexys 4 Artix-7 device, due to hardware accelerators' fast computation ability [35] [36]. Compared with the prior work on the Altera platform proposed in [22], our work reduces the slice count by 1018. Additionally, the power consumption of the implementation is 714 mW, including 15% as the static cost and 85% as the dynamic power dissipation. Furthermore, the design details of the components of the structure, such as the generation of an integral image, multiple pipelined classifiers, as well as the parallel processing, are discussed in this work, in order to provide a potential improvement for the future work. This thesis not only provides a successful synthesis of a face detection system on a hardware accelerator but also ignites intriguing ideas in terms of improvement aspects, such as approximating the design for finding an optimal energy-quality tradeoff corresponding to different applications as our future work.

Our vision is thus to create a smart building system as a case study, capable of sensing our surrounding environment, and more important, directing different types of sensor data to the optimal place, in terms of computing devices, for analysis and making decisions autonomous at the proximity of the network edge to improve data privacy, latency, and bandwidth usage.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I:

INTRODUCTION

Background & Related Work

In recent years, cloud computing's certain limitations or capabilities has led us to the new so-called paradigm, edge computing or also known as fog computing, sometimes [4] [5] [6] [12]. There are so many Internet of Things(IoT) devices out there, increasing in tremendous amount, in an ungovernable manner because of their prosperous usage. According to the research work of Cisco, there will be an estimated 50 billion IoT devices producing around 600 ZB data by the 2020 [46]. It is tedious work to upload such gigantic data to cloud servers, and process and retrieve back by systems when requested. This is where edge computing comes into the picture. In edge computing, you do not need to send the whole data packet on the cloud server, instead, data can be computed at the edge where it is being produced which is better in terms of efficiency and performance. For better understanding, a comparison between an edge computing and a traditional cloud computing systems in shown in Table 1.1.

In the last decade, the applications of embedded systems have boosted in the field of IoT. Typically, IoT systems have multiple sensors including computation devices scattered over an enormous area [54]. While the advent of the IoT solved many hitches, several inevitable problems were invited as well [3] [11]. The amalgamation of IoT and cloud requests facilitated the formation of edge computing; In which computing befalls at the network edge where there is no limitation of devices in terms of hardware type [14] [16]. In other words, the computing hardware device can be anything such as Raspberry Pi [7] [8], Field Programmable Gate Array (FPGA),

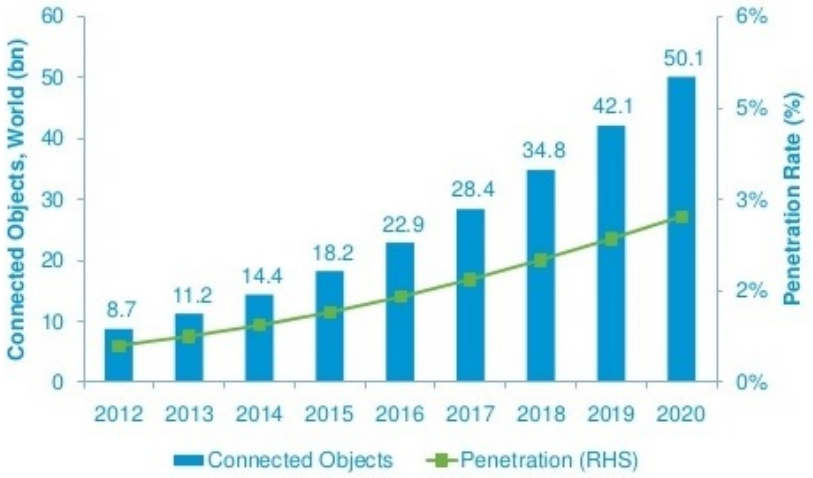

Figure 1.1: *Cisco's Prediction About IoT Devices [46]*

System-on-Chip (SoC), Application-Specific Integrated Circuit (ASIC), general purpose Central Processing Unit (CPU) or server [53].

Cloud Computing	***Parameters***	***Edge Computing***
Centralized	***Architecture***	Distributed
Far from the source	***Data Processing***	Adjacent to the source
High	***Latency***	Low
High	***Jittering***	Low
Low	***Data Privacy***	High
Global	***Accessibility***	Local
Limited	***Mobility***	Feasible
Few	***No. of Nodes***	Extremely High
Global	***Data Exploitation Risk***	Remains in Edge Network
Over the Internet	***Communication with Devices***	Local through Edge Node

Table 1.1: *Cloud Computing vs. Edge Computing*

With such a system, connectivity becomes primary issues [39] [57]. As far as communication is concerned, we have decided to use the IoT Mesh solution to establish an IoT network [39]. Although the IoT Mesh technology is yet to standardize, we have foreseen its abilities to be far better than ever imagined or experienced in terms

of power efficiency, connectivity, and confidentiality. We also anticipate that the IoT Mesh would not be the only way of communication in future rather a fusion of IoT Mesh with advanced standards such as Wi-Fi and Zigbee protocols to provide a better quality of communication [2].

Under the described idea, there are many research groups working on the edge / fog computing in the hunt for further exploration and improvement. The edge computing is, currently, one of the hot topics and with the trend of machine learning, scholars are combining two ideas in order to achieve desired goals which are mentioned above [16] [40]. In the field of edge computing, the major focus of the research work is on the software side which includes performance improvement, algorithm optimization, and increased efficiency with better task scheduling [17] [18] [19] [20] [37] [41] [42]. Comparatively, there is much less flow where scholars are working on the FPGA, digital circuit, embedded systems, and hardware architecture [15] [38] [41].

On that front, this literature proposes a promising architecture on IoT-Edge-Server based embedded systems with BLE Mesh system, surveillance system, a couple of processors and server. In a generic manner, we have two systems a) Bluetooth Low Energy (BLE) Mesh System and b) Surveillance System; These systems are fetching data, at one instance, to Intel i7-7700HQ CPU and to Raspberry Pi on the other instance. The entire system has, virtually, three layers 1) disposed systems, 2) edge computing network and 3) cloud computing network [43].

Due to the benefits of face/object detection/recognition [23] [24] [34], it plays an important role in our daily life. Although with such surveillance, privacy is compromised, the benefits overrule the denials of the system [55] [56]. On a second note, the application-specific computation on FPGA is usually higher efficiency than that on software like general purpose MCU or embedded systems, due to the hardware parallelism and specific design. Such a scenario has already been proved by Microsoft for

their search engine Bing [25] [26]. Briefly, they offloaded some complex computation on pure hardware platforms by reconfiguring traditional servers with multiple grids of FPGAs. In the end, the latency was reduced with a throughput of almost $\times 2.25$ [25].

The edge computing is boon to the IoT, but it also brings veil challenges which inspired us to propose a pioneering architecture. Nexys 4 FPGA is adopted in our work to implement the famous Viola-Jones face detection algorithm [27] [45], in order to find an optimal trade-off between quality and speed-energy constraints. The main contributions of this work are:

- To provide a robust architecture of IoT-Edge-Server embedded systems which can be implemented on extensively scattered environments such as agricultural farms, smart cities, or commercial / industrial buildings.

  Depending upon the application-specific task, we provide a system with an architecture which has two conceivable methods of computation 1) Traditional Cloud Computing and 2) Edge Computing.

- An implementation of two of the innumerable face recognition algorithms on Raspberry Pi and Intel i7-7700HQ CPU. There are few publications where they have performed a similar task on the comparably likewise hardware setting, but there is no mention of computation time or accuracy [9] [10] [13]. While our work not only delivers computation time and accuracy but also provides those data with an assortment of multiple face recognition algorithms, cameras, and computation devices as well.

- Improved Data Privacy - Edge computing enables computation at the edge nodes which requires personal data to be on the edge nodes rather than storing them on a cloud. In other words, data remains at the network edge which restricts data being hacked by anyone from the server.

- Improved Computation Speed - Performing any type of computation at any edge node prevents extreme back-and-forth of data to the cloud server and also improves the computation speed by reducing the latency. We provide benchmarks for computation time of cloud computing and edge computing for the same task to prove our hypothesis.
- We create a website (`www.open-ic.org`) to control the system using cloud computing. The website is also a platform for educational projects.
- We design communication commands for BLE Mesh fabricated boards which are reserved for only theses boards.
- We offer an implementation of the Viola-Jones algorithm with Very High Speed Integrated Circuit Hardware Description Language (VHDL) on Nexys 4 Artix-7 FPGA using Vivado v2017.4. This work is derived from the existing prototype which is a fully functional demo of face detection using Altera DE2-115 FPGA [22].
- To provide an alternative version of a synthesis of face detection demo with the implementation on Xilinx by using Vivado and Nexys 4 FPGA. Except for the data path of the Viola-Jones algorithm, we created memory blocks including RAMs and ROMs, 44-bit signed comparator, clock module, etc. Experimental results show that the slice count of our work on Xilinx is reduced by 1018 compared with the prior work on Altera in [22].
- The preliminary results, a demonstration of OV7670 camera-VGA monitor displaying, have been presented in our previous work [44]. In this study, not only do we synthesize the Viola-Jones design with VHDL, but we also evaluate the FPGA performance in terms of slice count and power consumption by using the estimation methodology in [28].

- The details of the implementation have been discussed in this work, in order to offer a potential of optimizations as our future work like [29] [47] [30]. The improvement of approximate designs on this platform will be coming soon: by providing the high-performance architectures [48] [49] [50] [51] and inaccurate designs on the computation components such as [21] [31], it is able to find an optimal energy-quality trade-off corresponding to different applications [32].

Structure Of The Dissertation

The rest of this dissertation is organized as follows. The Chapter 2 presents a proposed architecture and how it is different from traditional cloud computing systems. The Chapter 3 presents an implementation of the BLE Mesh systems with specific commands to communicate within sensors. The Chapter 4 provides working of the surveillance system with a brief introduction of face recognition. The Chapter 5 shows the architecture of the face detection system and explains each hardware part in detail. The Chapter 6 presents the implementation of a server on CPU and working. The Chapter 7 provides the experimental results collected from the working system and puts light on the improvement with preliminary results of a synthesis of the Viola-Jones face detection algorithm. Finally, in Chapter 8, we conclude this dissertation and discuss possible research directions in future work.

CHAPTER II:

PROPOSED WORK

Overview

In this section, we propose our unique architecture as shown in Fig. 2.1, we also provide various computation options. The computation options can be decided based on the requirements in terms of computation time and responsiveness.

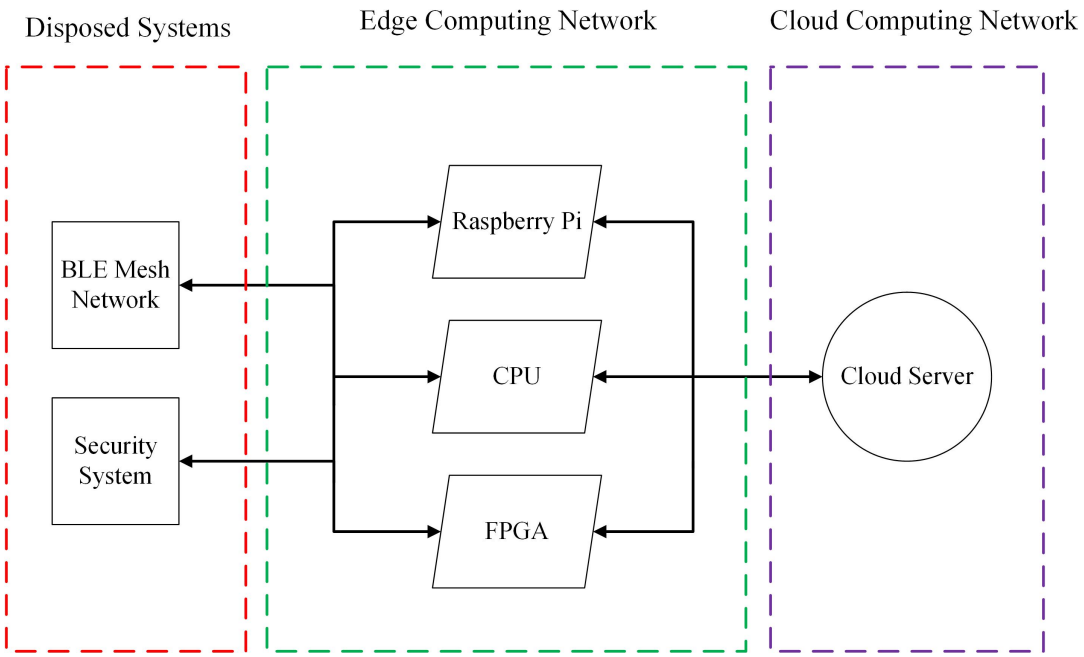

Figure 2.1: *Three Channel Computation Options*

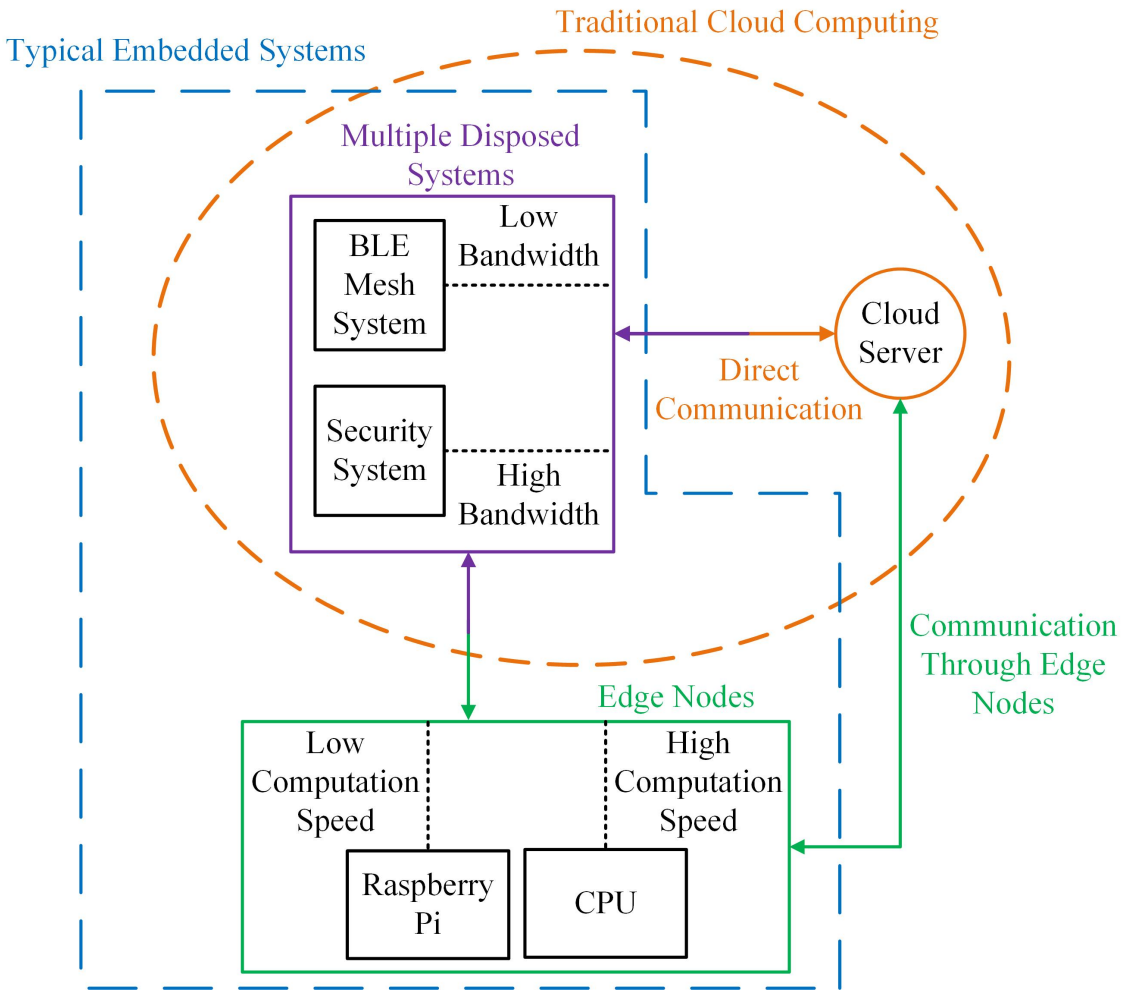

Figure 2.2: *Proposed Architecture*

Proposed Architecture

The Fig. 2.2 displays the proposed architecture of the IoT-Edge-Server based Embedded System. More specifically, the architecture consists mainly of three, virtual, layers 1) Disposed systems, 2) Edge computing network, and 3) Cloud computing.

Disposed Systems

In the Fig. 2.2, the disposed systems are labeled which are BLE Mesh network and security monitoring system. Generally, disposed systems are multiple integrated

systems to the edge computing network depending upon the applications and requirements. As mentioned earlier, for our proposed architecture and mainly to test our hypothesis we implemented BLE Mesh Network and Surveillance System.

The BLE Mesh network was a creation of four semi-customized boards using the APlix CSR 1020 modules which are suitable for low-power and restricted-complexity IoT applications. In this typical system, there is one master host connected to three slave boards; Each board possesses components to perform a designated task 1) Temperature & Humidity Sensor, 2) Light-Emitting Diode (LED), and 3) Motor.

On the other side, the surveillance system manifests a PIR sensor associated with a camera to detect and recognize face(s) in the milieu.

Edge Network

As shown in Fig. 2.2, the green-colored area consists of all the computation devices for the architecture. Such devices are connected to the disposed systems and cloud server which apparently creates an edge computing network. For time being, we neglect the orange-colored area to visualize a simple embedded system without any cloud server. For our system, we propose three types of edge nodes which includes FPGA, CPU, and Raspberry Pi. These devices will provide assistance in the improvement of latency for the task and sieving data which needs to be stored on the server eventually. The filtration of data includes removal of repetitive data, duplicated logs or any junk data.

Cloud Network

The entire orange-colored system, as displayed in the Fig. 2.2, is envisioned as a cloud computing network where all the data from the disposed systems can be transferred to the server for the computation to take further steps. When the computation is completed, the decision or data is sent back to the disposed systems. We would

like to emphasize that the orange-colored portion in Fig. 2.2 is a traditional cloud computing architecture where all the computing and data would be sent to cloud for any kind of processing. For the proposed research work, the server specifications are as follows: Intel(R) Core (TM) i3-7350K @ 4.20 GHz with 8 Gigabyte (GB) Random Access Memory (RAM).

Summary

This chapter presents a novel architecture which consists of mainly three, virtual, sub-systems. In other words, we offer a scalable architecture with three computing devices which perform the same task with different computation time based on the requirements.

CHAPTER III:
BLE MESH SYSTEM

Overview

The BLE Mesh system will sense the environment around placed sensors. The system consists of three devices and one host/server. The communication protocol is BLE to communicate between them. There are other possible protocols like Wi-Fi, ZigBee, etc. but the reason behind using BLE is that, it is the latest non-standardized version of Bluetooth, which enables research opportunities for researchers. Also, it works on low power, cost-effective and has more range compared to Bluetooth V4.2.

Implementation

In this section, we provide information about the working of the BLE Mesh system in terms of - 1) Sensor Network, 2) Physical Boards, 3) Communication Commands, and 4) Analysis & Feedback Algorithm.

1. Sensor Network

   As mentioned before, the BLE Mesh system has, currently, four self-designed boards accomplishing varied tasks creating the complete system on the foundation of the BLE low energy protocol. As shown in Fig. 3.3, using the mesh-topology method and abundant sensors, we can establish a large-scale environment. The Fig. 3.3, visually elucidates the arrangement of all the sensors in the giant setting. Moreover, the maximum distance between two boards to be able to communicate is around 57 feet. As shown in the Fig. 3.3, BLE Mesh host/server is only able to communicate with an actuator which are in the same group 2. Furthermore, the actuator works as a relay for group 1 and 2, in that manner BLE Mesh host/server is able to communicate with LED and tempera-

ture/humidity sensor boards. With the use of a relay in between two boards, the maximum-communication distance increases to 77 feet. Thus, a large network with numerous sensors can be deployed for wide-range habitats.

2. Physical Boards

   The BLE Mesh boards were a production of four fabricated boards using the APlix CSR1020 modules as mentioned in the Chapter 2. They consume low power and more suitable for less-complexity based applications. There is one main host connected with three other boards; Each board incorporates a specific task - temperature and humidity sensor, LED, and motor.

   The Fig. 3.1 presents all the manufactured using the APlix CSR1010 modules.

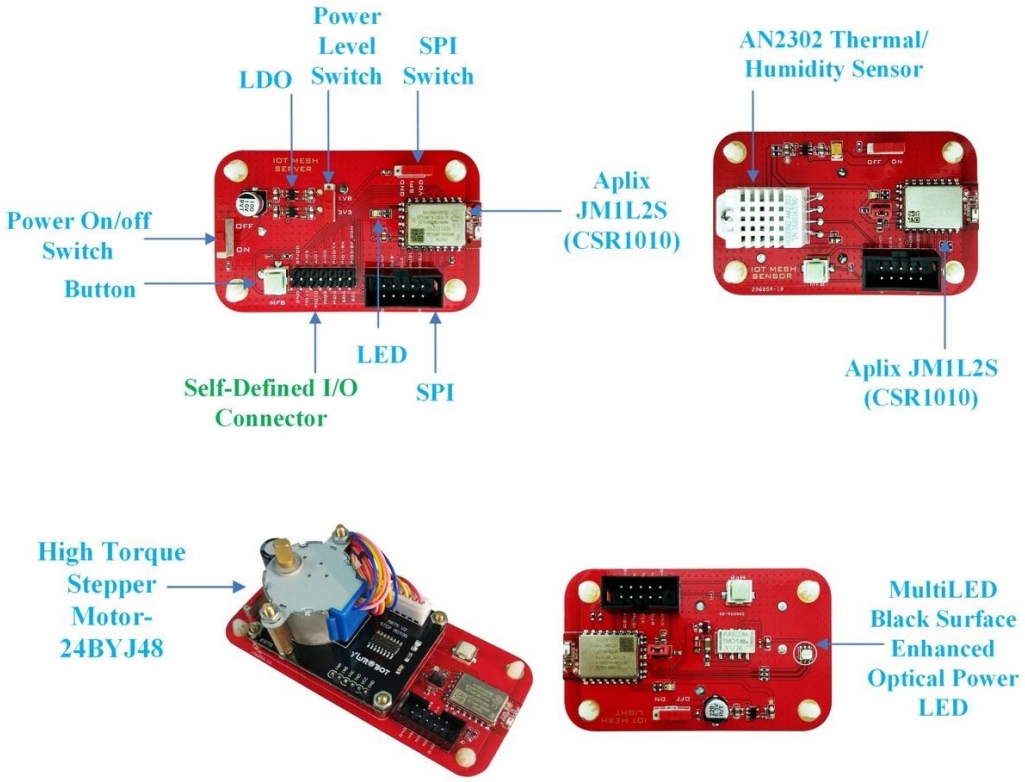

Figure 3.1: *BLE Mesh Boards v.1*

The Fig. 3.2 displays updated version, using the APlix CSR1020 modules, of the all boards which are more stable and efficient compared to the previous version.

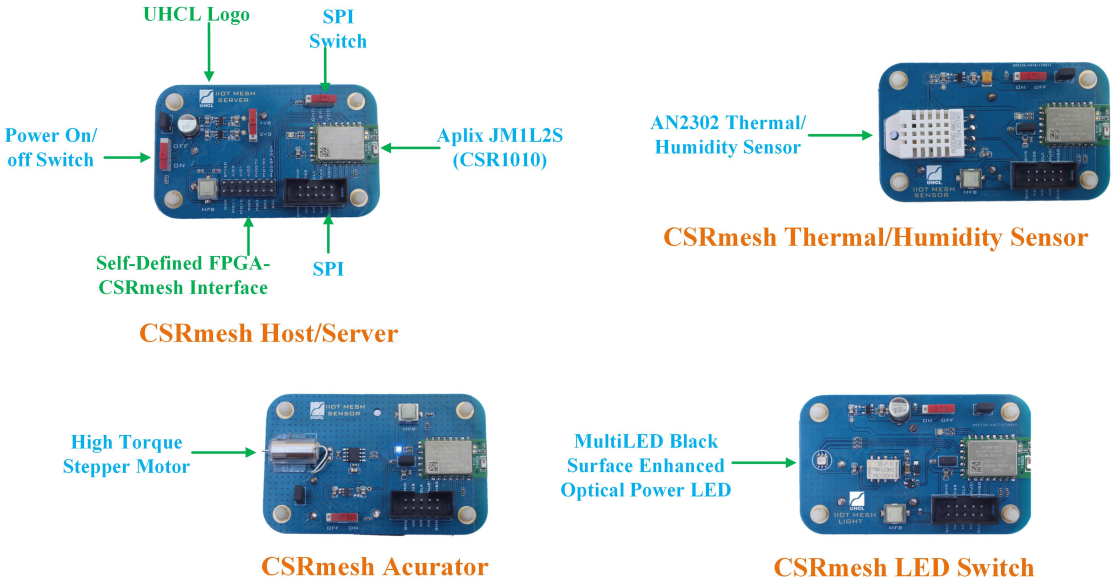

Figure 3.2: *BLE Mesh Boards v.2*

3. Communication Commands

Above mentioned boards communicate on specific commands with each other. Due to the self-designed boards, they are not, commercially, available in the market and also interact with each other on the private set of commands which are reserved for these typical boards only. Fig. 3.4 is a generic hexadecimal command which includes required data types with their pre-defined size, content, and description for better understanding.

Fig. 3.5 presents a particular hexadecimal command to turn on/off the LED.

Fig. 3.6 demonstrates a particular hexadecimal command to control the LED and the output is mixed-color of Red, Green, and Blue colors.

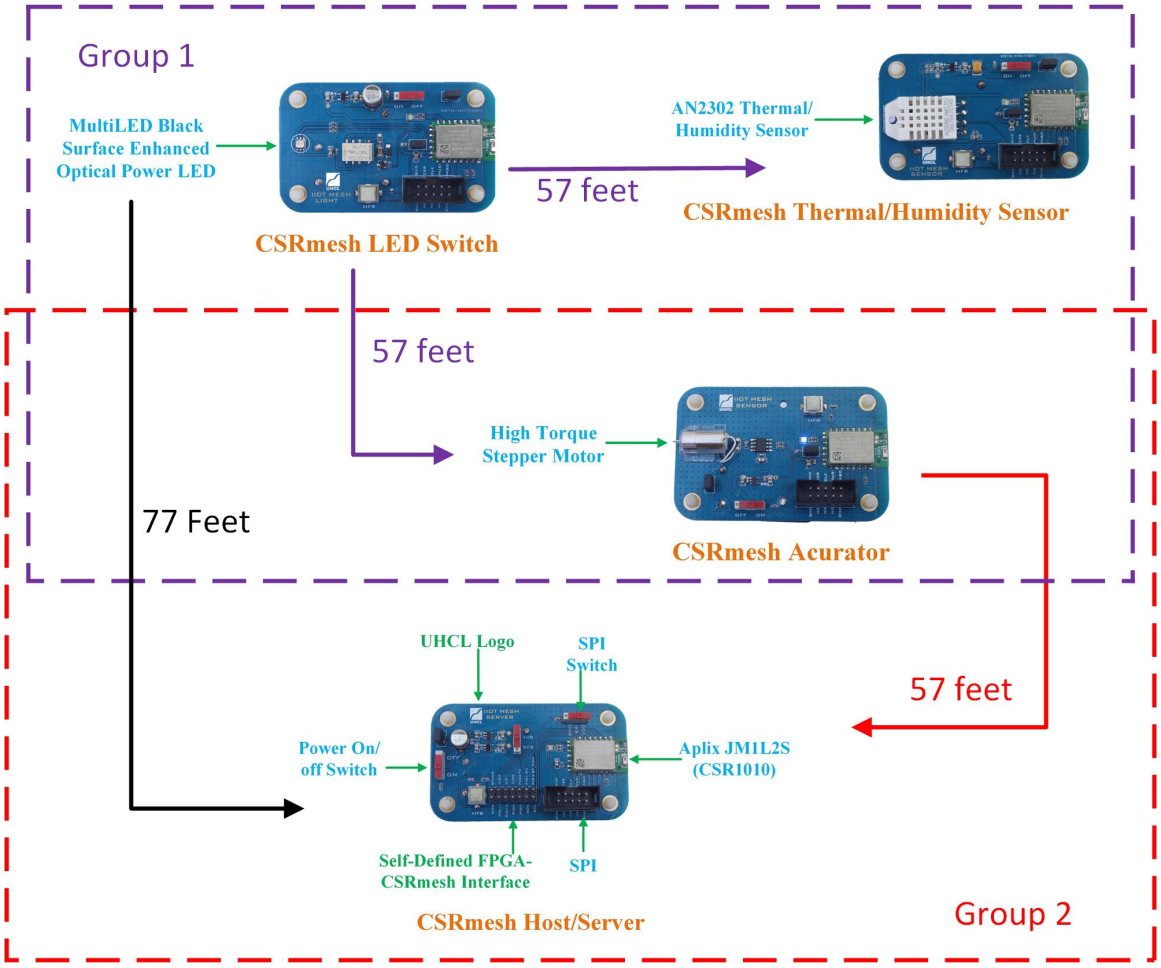

Figure 3.3: *Mesh Topology*

Fig. 3.7 is mainly used to check the network connection between the host and other devices. Note that, Device numbers 02, 03, and 04 allows to check network connection with LED, a thermal sensor, and motor, respectively.

Fig. 3.8 displays the command to control motor in which direction has two input values: 00 or 01 for clockwise and anti-clockwise rotation, respectively. The strength represents the speed of the motor which has a total of six input values: 00 stops the motor and 01 to 05 changes the speed of the motor from low to high.

Data Type	Size	Content	Description
Header	2 B	0xFA, 0xF5	Data start header
Size	1 B	0x00 to 0x20	No more than 32
Data	N B	0xFF... 0xFF	Data frame
Checker	1 B	0x00 to 0xFF	Addition checking bit, including size and data
Stop	2 B	0x0D, 0x0A	Stop tail

Figure 3.4: *Data Packet*

Header	Size	Type	On/Off	Checker	Stop
FA, F5	4	F2	00~01	XX	0D, 0A

Figure 3.5: *LED ON / OFF*

Fig. 3.9 provides ability to control the sensor. Here, 00 stops data receiving from the sensor and 01 activates data receiving which provides current temperature and humidity.

The command is shown in the Fig. 3.10, transmits current humidity and temperature value data to host which have 16 bits resolution for each, Most Significant Bit (MSB) to Least Significant Bit (LSB). The sensor data is 10 times of the real humidity. On the other side, the MSB is the sign bit for temperature where 1 and 0 represent negative and positive temperature, respectively. The other 15 bits are the value of temperature which is again 10 times of the real temperature.

An example of Humidity:

Humidity - 0000 FFFF

0000 0010 1001 0010 (Binary) = 0292 (Hex)

0292 (Hex) = 2*256+9*16+2 = 658

Real Humidity = 65.8% RH

Header	Size	Type	On/Off	Red	Green	Blue	Checker	Stop
FA, F5	7	F1	00~01	00~FF	00~FF	00~FF	XX	0D, 0A

Figure 3.6: *LED Control*

Header	Size	Type	Device No.	Checker	Stop
FA, F5	4	AB	02,03,04	XX	0D, 0A

Figure 3.7: *Mesh Network Check*

Temperature - 0000 FFFF

An example for Temperature:

0000 0001 0000 1101 (Binary) = 010D (Hex)

010D (Hex) = 1*256+0*16+13 = 269

Real Temperature = 26.9° C

1000 0000 0110 0101 (Binary) = 1065 (Hex)

1065 (Hex) = 0*256+6*16+5 = 101

Real Temperature = -10.1° C

4. Analysis & Feedback Algorithm

   As shown in Fig. 3.11, we set up the IoT Mesh network to test communication among host and devices using the CSRmesh protocol. In what follows, we propose a data analysis algorithm at the network edge in this work.

   Fig. 3.12 shows the algorithm part of the system, typically it describes the workflow of the algorithm.

   - Fetched data through sensors

     In the proposed system, the initial and primary step would be to gather all the information before any kind of analysis or process is done. First

Header	Size	Type	Direction	Strength	Checker	Stop
FA, F5	5	F4	00~01	00~05	XX	0D, 0A

Figure 3.8: *Motor Control*

Header	Size	Type	Sensing Data	Checker	Stop
FA, F5	4	F1	00~01	XX	0D, 0A

Figure 3.9: *Sensor Control*

of all, sensors will collect all the data containing information about light, humidity, temperature readings. The IoT Mesh host module will receive the data information, which works on the BLE technology.

- Data processing & Analysis

  Once data assembling is initiated, the analysis of data will be done. In this step, a different measurement will be carefully observed such as frequent data fluctuation, steady data information, garbage values, etc. Based on information database processing will be conducted. We have also planned to implement edge computing in this structure of the system. Typically, data processing is the primary part where all the decisions are being through which contemplated the application of edge computing. Here, processes will be introduced to one of the three categories of edge computing. They are created based on time sensitiveness of data.

  1) Most time-sensitive data are a priority and will be executed first by all means.

  2) A bit less sensitive data which can wait a bit and will be communicated to edge computation part and rest of the process will be handled over there.

Header	Size	Type	Humidity	Temperature	Checker	Stop
FA, F5	7	F3	0000~FFFF	0000~FFFF	XX	0D, 0A

Figure 3.10: *Sensor Data*

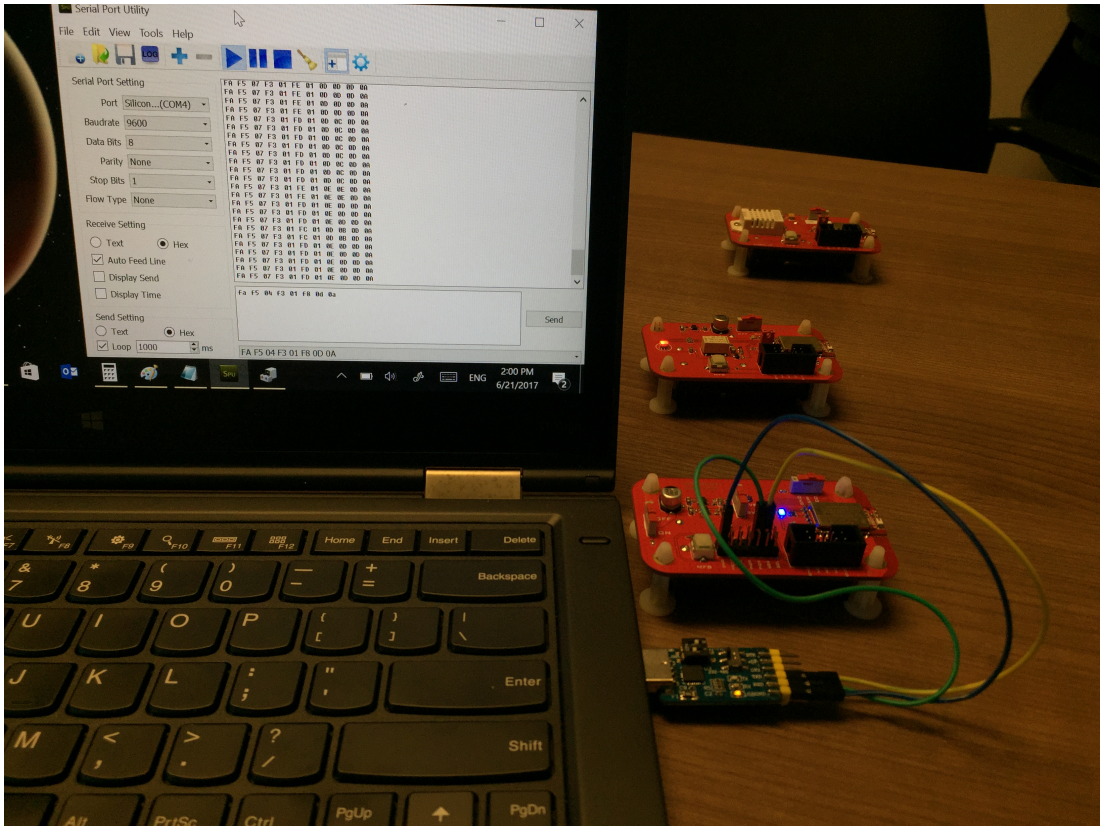

Figure 3.11: *Test: IoT Mesh Network [1]*

3) The last category will be the one where **data does not require to be processed immediately**, which can wait for some time, will be sent to the data server, where it will be stored. In the case, when such data is required, it will be processed followed by restoring the data to the data server after completion of the process.

- Autonomous Decision Execution

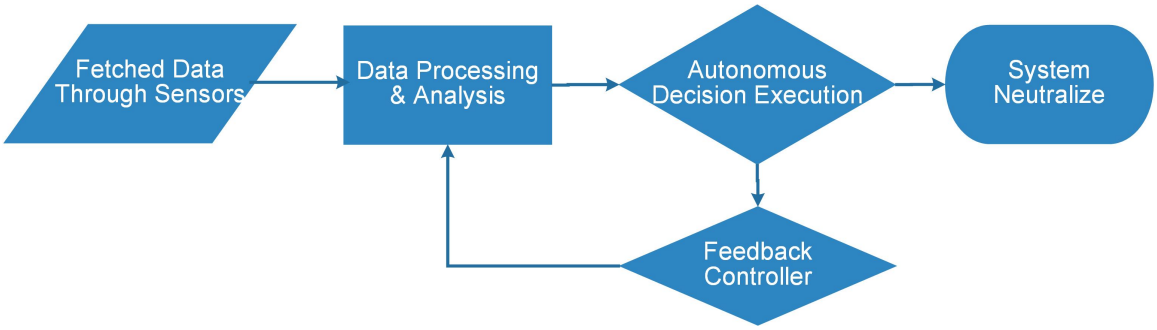

Figure 3.12: *Flow Diagram of the Algorithm [1]*

In this phase of the system, as the name suggests, data will be executed. When the temperature sensor detects heat or temperature rising in the surrounding atmosphere, a cooling system can be activated or a cooling system's temperature will be decreased if the system is already turned on. It will vise-versa too. If light is discovered through a light sensor, LED will be turned off or in the dark LED will be turned on. All performance will be completely independent of human interruption.

- Feedback Controller

  Feedback controller contains the data after the execution of previous processes and will be transferred again to the Data Processing & Analyzation phase to complete the concurrent and iterative process.

- System Neutralization

  System neutralization will be done only when the system no longer requires any kind of process execution. However, we do not put the whole system in hibernation, instead, data fetching and processing will always be done.

Summary

In this section, we show the working of the BLE Mesh system and how effective the system could be for large-scale environments using the traditional mesh-topology which also opens research directions for research scholars.

CHAPTER IV:
SURVEILLANCE SYSTEM

Overview

When such a giant system is deployed on the large environments, security becomes one of the main concerns [12]. Keeping security, a crucial factor of our system, we implemented an intelligent control system which can detect motion with a PIR sensor. Moreover, a detected motion enables the camera which captures an image on that instance of motion to detect and recognize a human face(s) in that typical image frame. If an unknown human is detected, the system creates an alert and notifies to the designated person via email. Upon the recognition of a known person from the database, there will be no alert generated.

Face recognition has become a very useful and essential part of the humans routine life. Face recognition simply means to identify, more precisely, verifying the input which either can be image or video, depending on the system requirements. To recognize the face, steps are divided into various steps like face detection, feature extraction, and face recognition. Fig. 4.1 shows the flow of face recognition process. It is an old concept and has grown in a huge manner. Yet, it still does not provide fully satisfactory results due to different factors in the recognition process like pose variation, feature occlusion, facial expressions, imaging conditions. It is still improving day by day and we have started getting better results because of better cameras

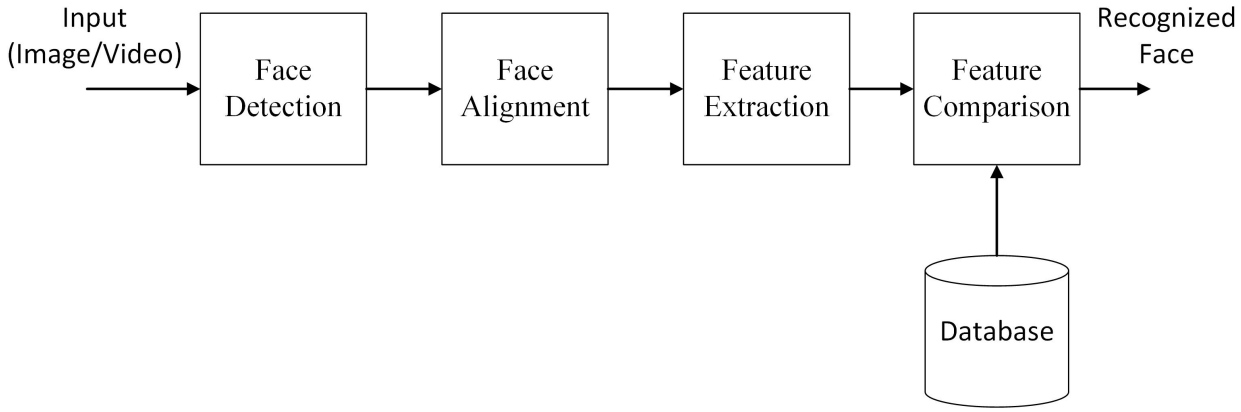

Figure 4.1: *Face Recognition Algorithm Flow*

compared to previous ones, improved approach to the problem, advanced algorithms, etc.

Face Detection

Face detection simply means to locate the face/faces in the given input which either be images or videos and it involves several challenges [24].

- Pose Variation - The ideal scenario would be one in which only frontal images are involved. Unlikely, this is not the usual case in the real case.
- Feature Occlusion - The presence of features like spectacles, beards, cap, etc. can be a hindrance in the process of detecting face(s).
- Facial Expression - Facial features may vary greatly due to different facial gestures.
- Imaging Conditions - Different cameras and ambient conditions can affect the quality of the image which can eventually, create problems in detecting faces.

Face detection is not an easy task and there are not globally accepted grouping criteria. There are different detection methods while keeping in the mind about the various scenarios mentioned below.

- Controlled Environment - It is most straightforward case. Photographs are taken under the controlled light, background, facial pose and angle, etc.
- Color Images - Typically, skin color is used to find faces. But the drawback would be if light conditions are weak, this does not work quite well [33].
- Images in Motion - Real-time videos give the chance to use motion detection to localize faces. Nowadays, most commercial systems must locate faces in videos. There continuing challenge is to achieve the best detecting results with the best performance.

Face Extraction

A human can recognize faces since we are 5 years old [58]. It seems to be an automated and dedicated process in our brains [58], though it is a much-debated issue [59]. However, it is much clear that we can recognize the people we know regardless of any facial occlusions. We can recognize a person even if he has grown beard. We can even recognize the persons very old photograph. Such tasks are trivial for us but they characterize a heck of challenges to computers. Feature extraction involves multiple steps, dimensional reduction, feature extraction, and feature selection.

Face Recognition

This is a part where face recognition is done based on the database which is created in the feature extraction part. After locating faces from face detection step, it is important to save some vital face features which will be useful in the face recognition process since they are useful to identify the faces.

There are plentiful algorithms available to recognize the face but for the sake of simplicity we chose two of them to test on our system 1) the Low Binary Pattern Histogram (LBPH) and 2) the Deep Metric Learning (DML).

- Low Binary Pattern Histogram

  The LBPH is one of the algorithms open-sourced in the Open-CV library [60]. Post-implementation, we noticed that LBPH provides lesser accuracy compared to the Deep Metric Learning algorithm but at the same time it also has lower computation time for the equal set-up. We implemented LBPH on Raspberry Pi 3 and Intel Intel(R) Core (TM) i7-7700HQ CPU @ 2.80GHz with 16 Gigabyte (GB) Random Access Memory (RAM). We also need to create and train the data-set in order to recognize the face(s) using the algorithm. Training of the data-set is performed every time changes occur in the data-set otherwise we do not need to train and directly run the algorithm to recognize face(s).

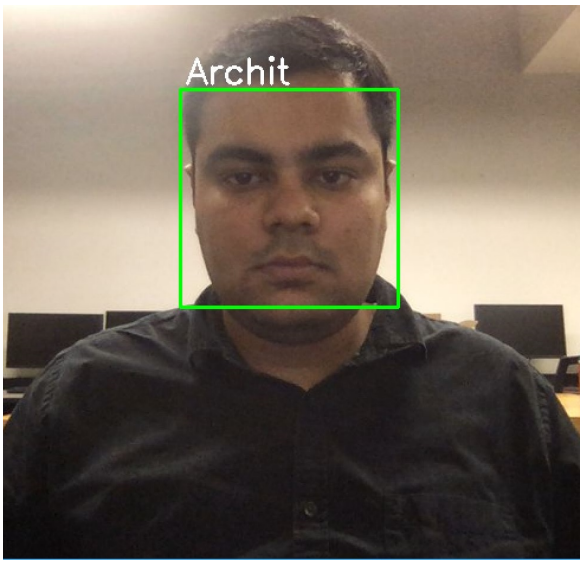

Figure 4.2: *LBPH Based Face Recognition [43]*

Fig. 4.2 is a screenshot of recognizing person (Archit). Due to the lower computation time, it was feasible to contrivance LBPH on the Raspberry Pi. For the LBPH, the lighting of the surrounding, distance of human from a camera, and many other perilous situations may affect the accuracy of recognizing a person from the given image frame.

- Deep Metric Learning

  The Deep Metric Learning (DML) is a face recognition algorithm from the dlib C++ library which is an open-source collection of machine learning applications and algorithms [61]. The DML delivers surprisingly higher accuracy but at the same time drains out the computation power of a processor. The requirement of this algorithm is so high that we were not able to implement such algorithm on the Raspberry Pi. The execution was successfully achieved on the Intel(R) Core (TM) i7-7700HQ CPU @ 2.80 GHz with 16 Gigabyte (GB) Random Access Memory (RAM).

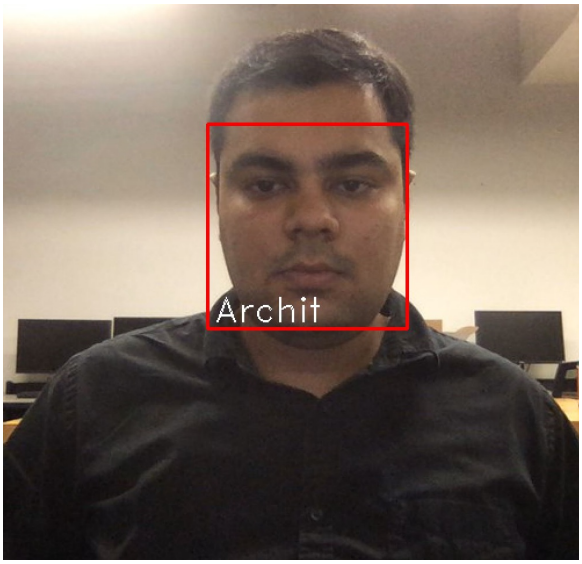

Figure 4.3: *Deep Metric Learning Based Face Recognition [43]*

Fig. 4.3 is a screenshot Deep Metric Learning based face recognition system on CPU identifying a human (Archit). There is no need to train data-set in Deep Metric Learning like LBPH.

Furthermore, on top of the face recognition systems, we have implemented an alert system which notifies the designated person if the recognized face(s) is not in the data-set. To accomplish such task, we instigated a script using

Simple Mail Transfer Protocol (SMTP) which will send an alert in the email to the authorities to initiate safekeeping regarded action. The benefit of such notification is that the nominated person can choose to take contemplating safety-action without physically being in the range of the system if needed.

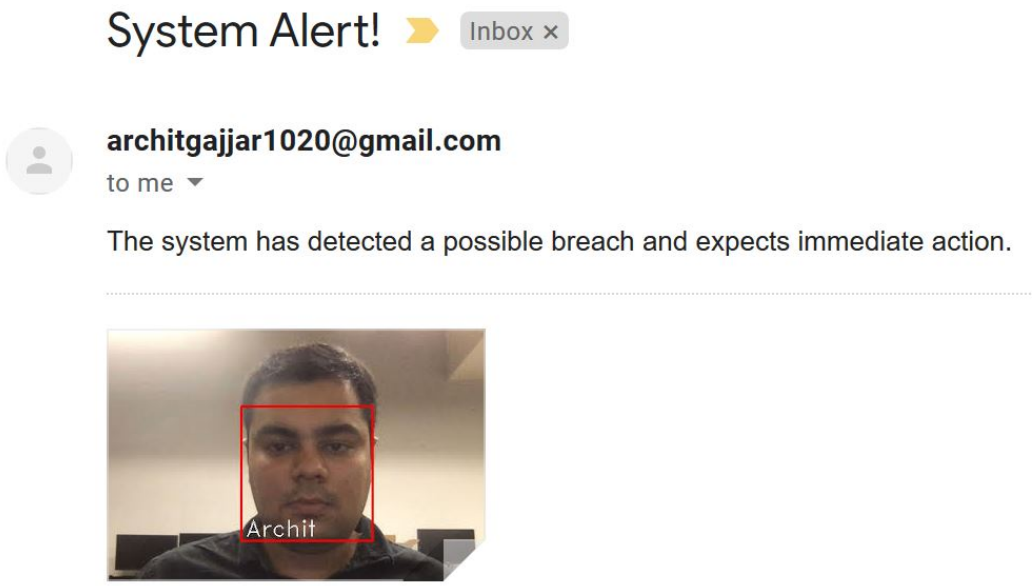

Figure 4.4: *Email Notification [43]*

Fig. 4.4 is a screenshot of a security alert from the system. Importantly, the provided screenshot is merely an example of alerting an authority. The actual system would only notify when the human face is detected and not recognized from the trained data-set(LBPH) or database(DML).

Summary

In this section, we present an overview of the face recognition process with low latency & lesser accuracy and high latency & higher accuracy by, a couple of face recognition algorithms, the LBPH and DML, respectively. Additionally, we also pro-

vide the alert system using SMTP protocol which is merely an example of notifying the user which could be more extended with iOS or android applications.

CHAPTER V:

FACE DETECTION ON FPGA

Overview

In this section, the design architecture on FPGA is introduced, and the details of the implementation are further explained as well.

Hardware Architecture

Before moving onto the design, let us explain about the architecture which would make the understanding of design much easier. As shown in Fig. 5.1, the design can be separated into three main components.

First, the OV7670 camera module consists of controller and capture units which are configured by the top module [44]. Second, where the Viola-Jones face detection algorithm is implemented. The output generated by the algorithm would be sent through the last but not the least part of the system, VGA interface, and monitor.

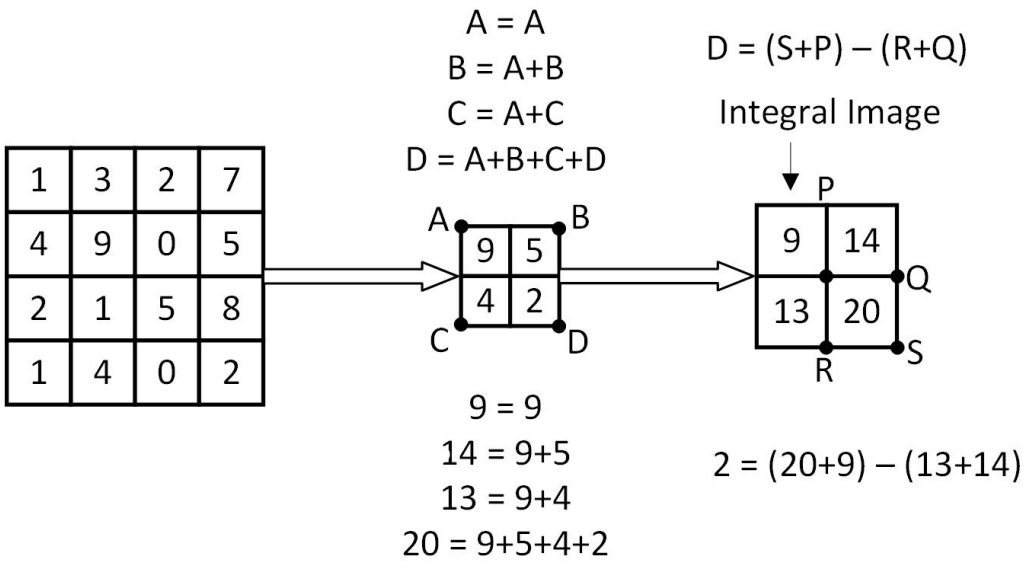

Figure 5.1: *Integral Image Example [45]*

Implementation & Design Flow

As per Fig. 5.1, the input image/video is being captured by OV7670 camera module which is a 12-bit RGB, 4-bit for each color [52]. Data captured by the camera is in 320×240 pixel resolution by default. Here, the camera is manipulated by the top module which contains different capture modes as required and capture button trigger to take a snapshot of the image. Later, that image is stored in the image buffer.

The stored data in the image buffer is used to generate an integral image which speeds up the processing time and efficiently produces the sum of the values, rather pixels in this case, in a rectangular subset. Note that, before producing integral image, input, a 12-bits RGB colored image is converted into an 8-bits grayscale image. Not only the integral image but integral image square is also generated in the followed up step. Each integral image possesses dual buffers containing data read through memory. Word size for integral image is 21-bits (16× word size) where else integral image square is 29-bits (16× word size). Addressed chunks enable routing of any data set (16× word size) to mux which is followed by a buffer.

Fig. 5.2 is an example of how integral image is calculated. First, we have 4×4 matrix, in our case which is a 12-bit colored image. That input/image is downgraded by 2 converting it into 2×2 matrix, for our case 8-bit colored image, eventually, converted into 8-bit grayscale image. Later, 2×2 is converted into integral image using generic formula provided in Fig. 5.2, along with an example. Once, an integral image is found, using those pixel values one can find any pixel values in 2×2 matrix using just 4 values of the integral image.

The method was introduced by Viola-Jones face detection algorithm [34]. To find an integral image, you need to take sum of all pixel values that are on the left and top

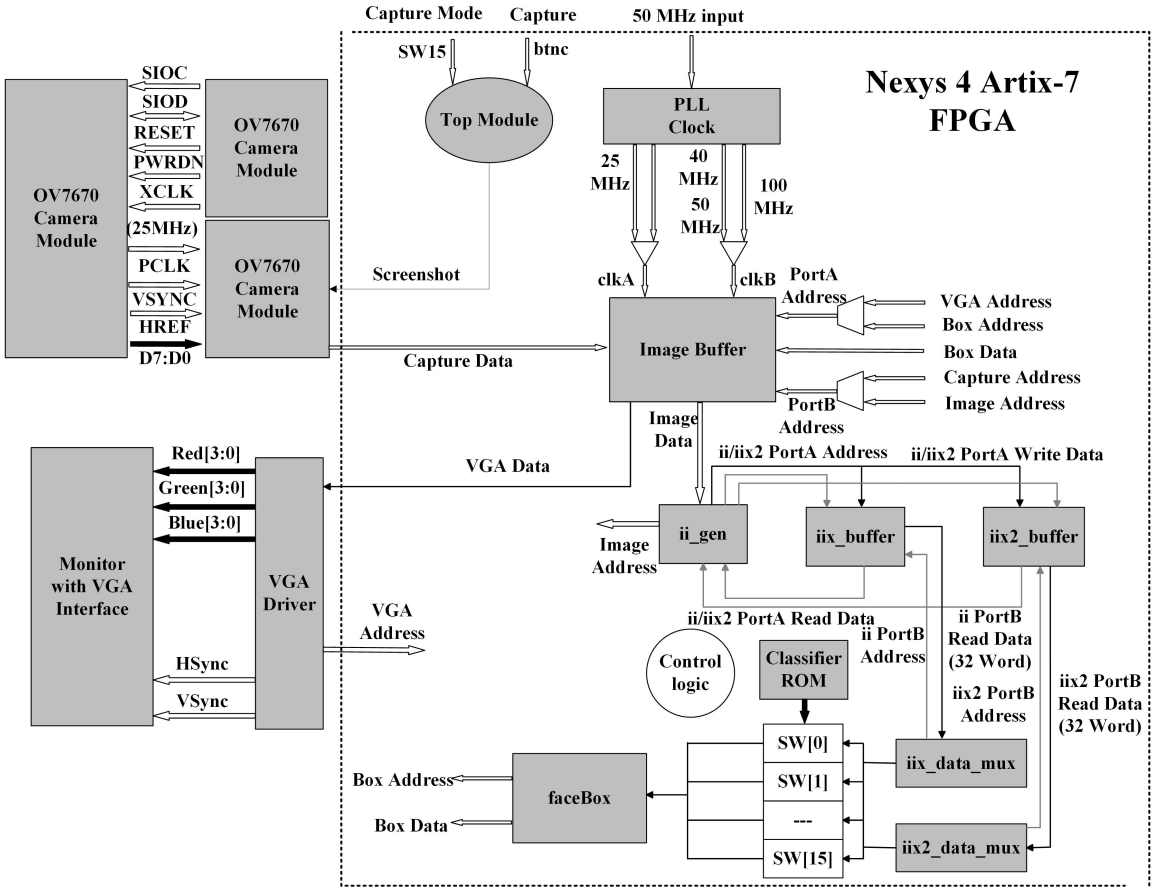

Figure 5.2: *Hardware Architecture [45]*

side. Considering the example from an image, to find a value for 2, you need to sum 9+5+4+2 which gives 20. Seeing, integral image, while using formula D = (S+P) (Q+R), one would be able to find pixel value at the same place from the original image. This saves a lot of processing cost and decreases the computation time of the system.

There is a submodule in the main algorithm which is `subwindow_top`. There are 16 sub windows processing in parallel on the data chunks provided from the addresses. Now, these 16 sub windows described as sw[0], sw[1], sw[2] up to sw[15] classifiers make a decision whether the provided image contains any face/faces. In more detail, theses classifiers create weak classifiers first. Weak classifiers would only be considered

as strong classifiers once they pass a weak threshold. Furthermore, strong classifiers are compared with a strong threshold value. If the value for the strong classifier is greater than the strong threshold value, the system consider that the image has face in it. More details about the threshold are explicitly mentioned in the Viola-Jones face detection algorithm [23]. Once we get the positive values from the strong classifiers or in other words those strong classifiers which are beyond the strong threshold values, faceBox will be fetched those data and draw rectangles around all the detected faces. This image is displayed on the monitor since data was fetched to image buffer from faceBox via VGA interface. The exhibited output is 12-bits RGB colored image with rectangles around the detected faces.

We employ Mentor Graphic ModelSim 10.4d as the simulator in our study. After simulation, we can obtain the waveforms (VCD) with switching activities of signals, IOs, and logics. And after the synthesis, the gate-level netlists (NCD) is able to be collected. These files are needed to analyze the power consumption using XPower Analyzer (XPA). Finally, Xilinx Vivado is used as the synthesis tool with the target device Nexys 4 Artix-7 FPGA [27].

Summary

In this section, we present overview of face recognition with applications by a couple of face recognition algorithms.

CHAPTER VI:
SERVER-SIDE EXECUTION

Overview

In this section, the literature demonstrates cloud-server set-up implementation for traditional cloud computing systems shown in Fig. 2.2 with the orange-colored circle.

Implementation

As shown in the Fig. 2.2, we set traditional cloud computing architecture to measure the computation time for each task such as recognition time for both LBPH and DML algorithms as well as BLE Mesh network commands. We create a server using Node.js, a JavaScript run-time environment to execute JavaScript, scripting on an Intel(R) Core (TM) i3-7350K @ 4.20 GHz with 8 Gigabyte (GB) Random Access Memory (RAM). In Fig. 6.1, steps to create a server on a processor are shown including all the project environments which must be followed by 'nodemon app' command where the 'app' is our project name.

As represented in Fig. 6.2, whenever web-request is triggered by task or user, an appropriate Node.js script runs to request data on the cloud from the disposed systems. In a follow up after that processed request, data is transmitted towards cloud from the disposed systems. Depending upon the requirements, the cloud server executes a task or performs computation and sends back data to the disposed systems, if needed. For this particular thesis work, the cloud server will be performing all the task or computation in a python scripting environment. Also, for any continuous

```
global.__basedir = __dirname;

const express = require('express');
const dotasksController = require('./controllers/dotasksController.js');

const app = express();

app.set('view engine','ejs');
app.use('/',express.static('./'));
app.use('/assets',express.static('./assets'));
app.use('/uploads',express.static('./uploads'));

dotasksController(app);

app.listen(2000);
console.log('Now listening on port 2000');
```

Figure 6.1: *Creating Server on Processor*

task, such a process could have back-and-forth data transmission from cloud to a disposed system or vice-versa.

To control the whole system, we create a simple web-page to perform all the tasks using Hypertext Markup Language (HTML) and Node.js. A screenshot from demo web-page is shown in the Fig. 6.3.

Fig. 6.4 is a screenshot from the web-page after a task is executed on the BLE Mesh system and it also shows the time of execution for a task on cloud computing.

Fig. 6.5 displays a screenshot of web-page when the database is stored on the cloud for LBPH algorithm based face recognition. The database stores 300 images each for every person.

As mentioned before, every-time we change the database, the algorithm is required to train the new database and for the status-update screenshot is shown in the Fig. 6.6.

Finally, after receiving the database and training them, the algorithm would be able to recognize the face(s) from the image/video frame. One of the successful

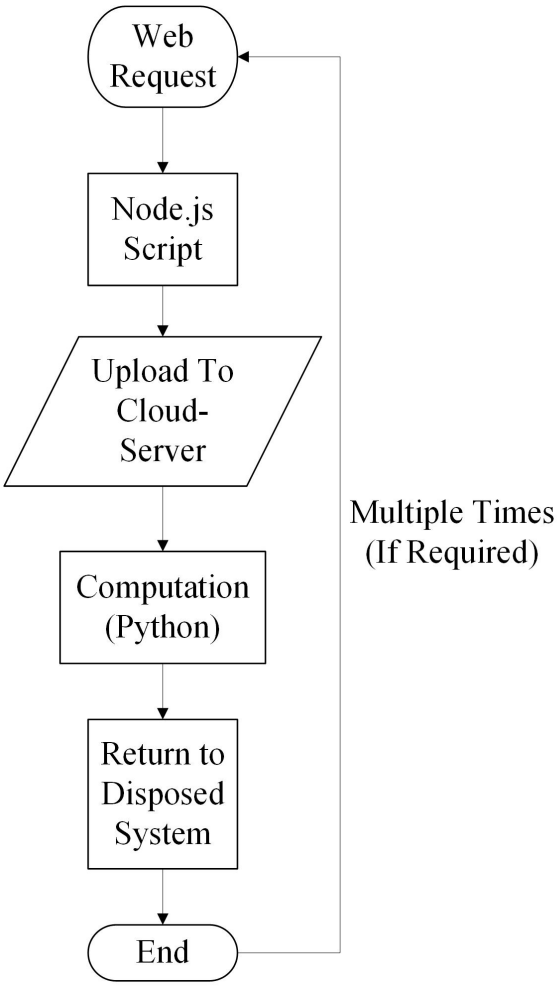

Figure 6.2: *Cloud Server Request Flow*

recognition images, we provide a screenshot with total computation time as well in the Fig. 6.7

In the Fig. 6.8, we share a screenshot of the face recognition system using DML algorithm including the total computation time.

Summary

In section, implementation of cloud server using Node.js scripts is shown including the screenshots taken from the working system. The images from the face recognition

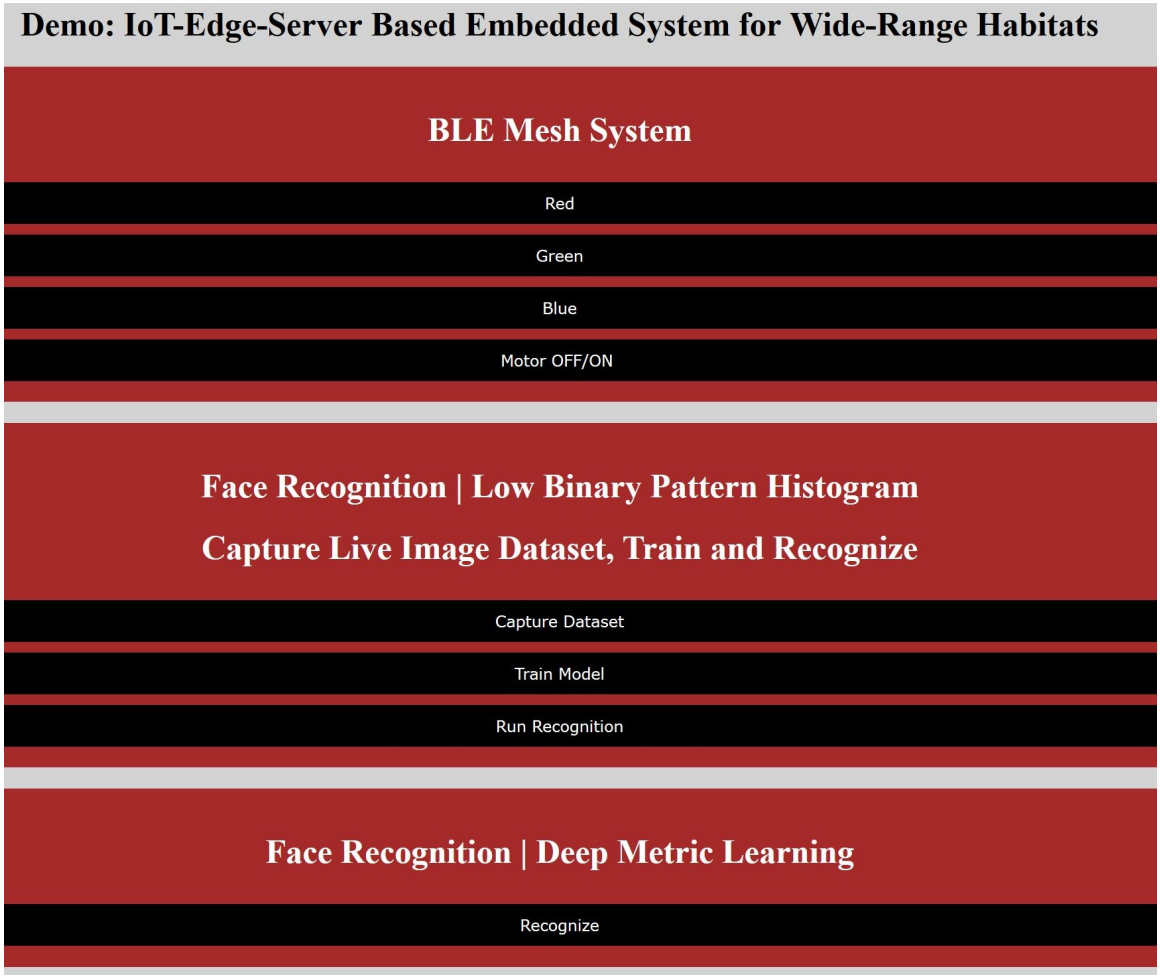

Figure 6.3: *Demo Webpage*

processes also denotes, with the reference from printed computation time, higher computation speed.

Figure 6.4: *BLE Mesh System - Task Execution*

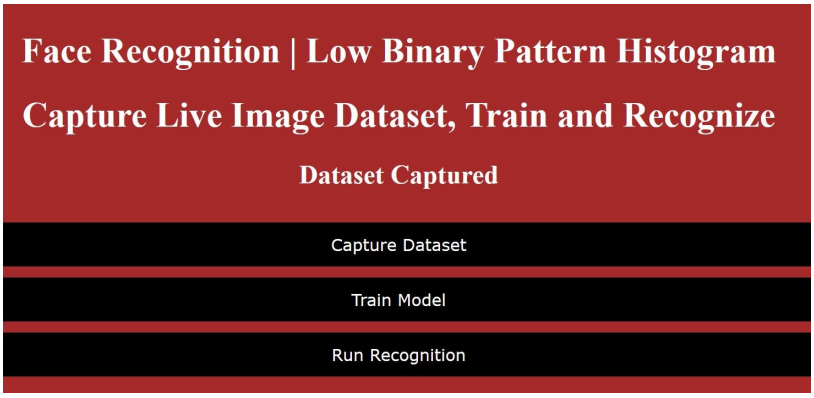

Figure 6.5: *Capture Data-set for LBPH*

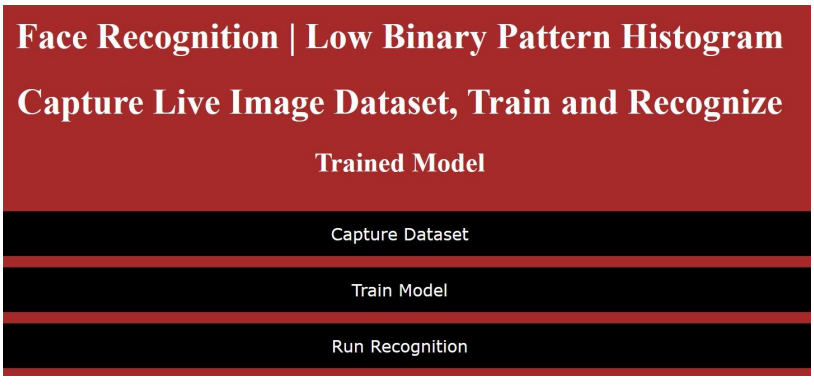

Figure 6.6: *Trained Data-set*

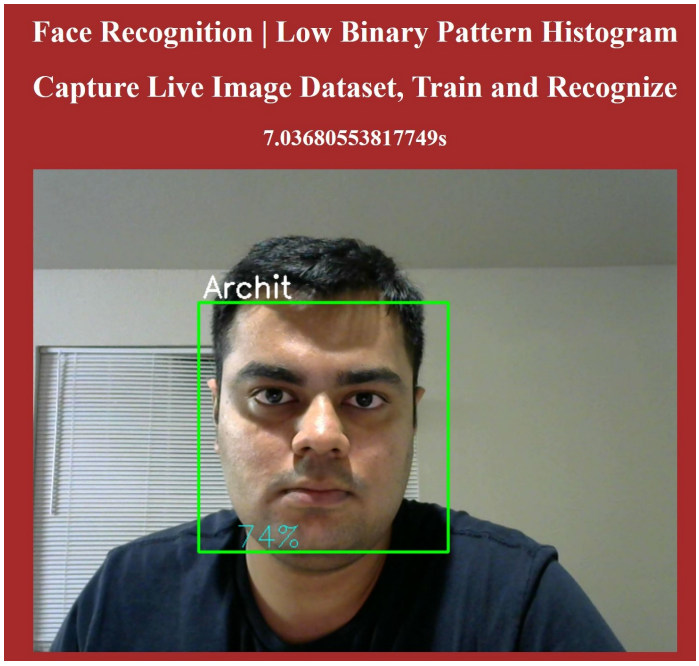

Figure 6.7: *Face Recognition using LBPH*

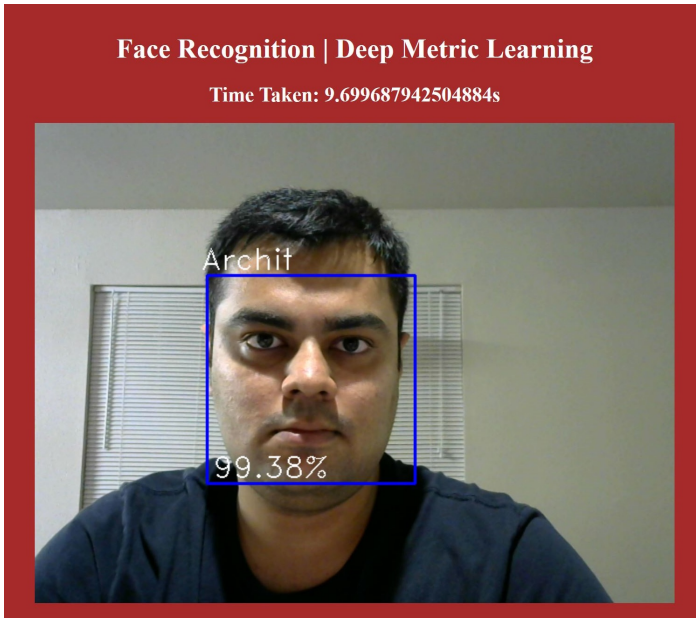

Figure 6.8: *Face Recognition using DML*

CHAPTER VII:
EXPERIMENTAL RESULTS

Overview

In this section, we display results from the conducted experiments on the system to bolster our hypothesis. We share the BLE Mesh system distance improvement with a node between them functioning as a relay. We provide accuracy results for the LBPH and DML algorithms on various computing devices including their performance results in terms of computation time. Additionally, this section also includes preliminary results from the synthesis of the face detection algorithm on FPGA which consists of resource costs on the FPGA and power consumption.

BLE Mesh System Distance and Computation Time

Table 7.1 represents the maximum distance between two Mesh network boards which is 57 feet. Interestingly, any board would not be able to communicate with other boards if the distance is greater than 57 feet, but it can be, definitely, increased up to 77 feet using other boards as a relay between those boards.

Maximum Distance	***No. of Node(s)***
57 feet	0
77 feet	1

Table 7.1: *Distance Results For Mesh Network [43]*

Table 7.2 represents computation time for a specific task on a couple of computation devices as well as a cloud server. While execution time on the Raspberry Pi and CPU for a specific same task is quite negligible, the execution time cloud server is terribly high even for such a low-data-rate task execution.

Device	***Task Execution Time(s)***
Raspberry Pi	≥ 0.07
CPU	≥ 0.06
Cloud Server	≥ 0.2

Table 7.2: *Task Execution on BLE Mesh System [43]*

Surveillance System Computation Time

Table 7.3 displays time taken, in seconds, by different devices along with both the recognizing methods for the edge computing as well as execution for cloud computing. The computation time utterly relies on the hardware and it is processing speed. Furthermore, the time taken by the cloud server is fairly high compared to any edge computation devices.

Device	***Recognition Time(s)***	***Method***
Raspberry Pi	0.41 - 0.45	LBPH
CPU	0.035 - 0.037	LBPH
CPU	0.391 - 0.398	DML
Cloud Server	≥ 6	LBPH
Cloud Server	≥ 9	DML

Table 7.3: *Face Recognition Performance For Edge/Cloud Computing [43]*

Surveillance System Accuracy

Table 7.4 is the accuracy of recognizing the face(s) on the different system with the same environment setting. The LBPH provides extremely lower accuracy compared to the DML method which consists potential raise to a certain percentage by training more data-sets and also replacing with better camera quality. Although there is a

plausible way to increase accuracy, we exceedingly doubt about LBPH's capability to imitate the accuracy of DML.

Devices	***Recognition Accuracy(%)***	***Webcam***	***Method***
Raspberry Pi	50 - 60 %	Logitech C270	LBPH
CPU	50 - 60 %	Logitech C270	LBPH
CPU	60 - 70 %	720p HD Laptop	LBPH
CPU	99.38 %	720p HD Laptop	DML
Cloud-Server	50 - 60 %	Logitech C270	LBPH
Cloud-Server	99.38 %	Logitech C270	DML

Table 7.4: *Face Recognition Accuracy [43]*

FPGA Resource Cost

Using Vivado 2017.4 and the constraint file with the Nexys 4 Artix-7 FPGA board, we synthesize the code. In our design, we use 32,309 Look Up Tables (LUTs) which is 50.96% of the LUTs available on the FPGA board. Compared to the implementation in [22] where the number is 33,327, our proposed work reduces the number of LUT by 1018. In addition, the D flip-flop cost in our work is 38130, which is 30.07% of the total resource provided. Moreover, we use 1 Phase-Locked Loop (PLL) out of 6 which is similar to the other design, and 35 DSPs employed as some complex computation components like variances and floating-point multiplications.

Since Nexys 4 Artix-7 FPGA does not have ADV7123 Digital to Analog Converter (DAC), used IO pins are quite less, which are 41 compared to [22] where their total number of pins is 56. Such change does not make a bigger difference in terms of quality of results which was reduced to 12-bit VGA output instead 24-bit VGA output, however, the reduced number of IOs is able to lower the switching activities of signals and logics, resulting in lower power consumption on FPGA development.

Resource	*Utilization*	*Available*	*Utilization (%)*
LUT	32309	63400	50.96
LUTRAM	644	19000	3.39
FF	38130	126800	30.07
BRAM	94	135	69.63
DSP	35	240	14.58
IO	41	210	19.52
MMCM	1	6	16.67

Table 7.5: *Summary of Resource Utilization [45]*

Fig. 7.1 depicts the resource cost as a percentage of the entire resource on the board, in such a way clear results are able to be described. The figure is generated by Vivado.

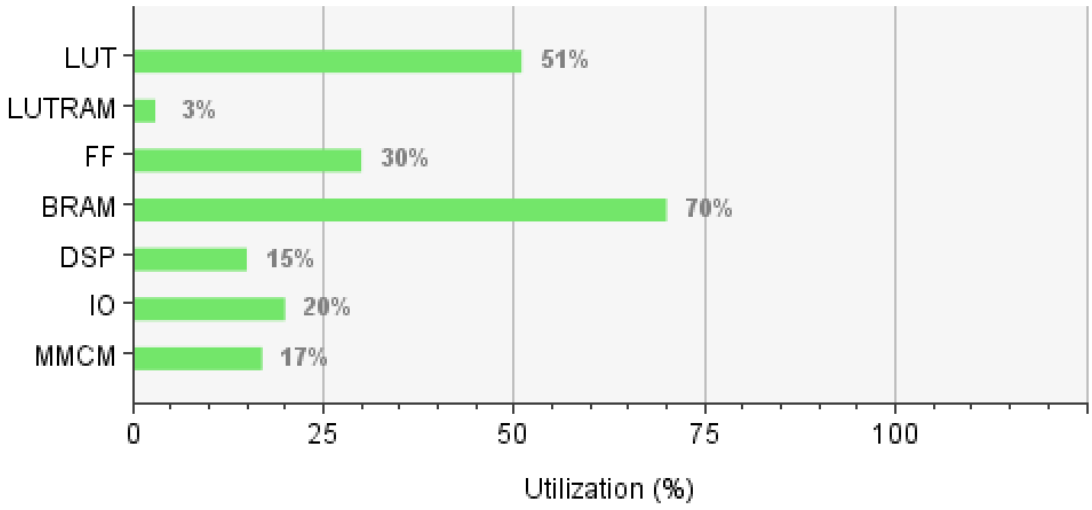

Figure 7.1: *Resource Utilization Graph [45]*

FPGA Power Consumption

Fig. 7.2 demonstrates the abbreviated and graphical breakdown of power estimation report generated by the power analyzer. Generally, the total power is dissipated

in terms of static power and dynamic power. It can be observed that the static power cost is 15% and the majority proportion of the total power is dissipated by the dynamic power. For our design, static power consumption is 104 mW and dynamic power cost is 610 mW with total power consumption of 714 mW. Since the power cost has not been estimated in [22], it is not able to compare the power dissipation with the prior work between the two different platforms.

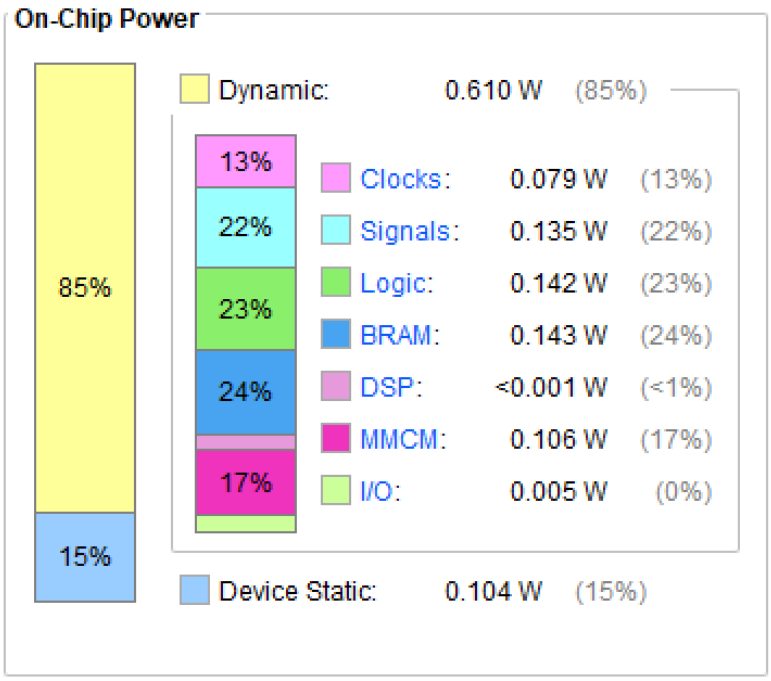

Figure 7.2: *Power Estimation [45]*

The dynamic power is mainly affected by the toggle rates of clocks, signals, logics, and IOs, etc. More specifically, the switching activities of signals, logics, and BRAMs cost 22%, 23%, and 24% of the dynamic power dissipation, respectively. And the remaining power consumption is mainly come from the mixed-mode clock manager module (MMCM) and the clock input, totally 30% of the dynamic power cost. Due to the small number of IO and DSP utilization, the power cost is very low (around 1%) compared to the aforementioned FPGA components.

Summary

The experimental results dispense a shred of concrete evidence to our proposed architecture. The BLE Mesh system plausibly, can be scattered on the wide-range habitats using nodes between two mesh boards. The results also show that for different computation requirements various computing devices can be implanted as the disposed systems. With a better camera, lighting surrounding, and hardware face recognition accuracy has a potential improvement.

CHAPTER VIII:
CONCLUSIONS AND FUTURE WORK

In this chapter, we summarize our contributions presented in this dissertation. We then discuss the possible directions for our future research work.

Conclusions

The proposed architecture including the prototype results presents favorable fall-out as far as wide-range habitats are concerned with plausibility to deploy long-range BLE Mesh system, scalable surveillance system, and an email alert system. A large-scale BLE Mesh system can be established using mesh-topology. A scalable surveillance system means using a different combination of face recognition algorithms, camera quality, and hardwares computation speed, one can use such a set-up depending on the requirements. With implementation of the proposed architecture, there are many application-specific large-scale sites can be benefited such as commercial buildings, agricultural farms, industrial factories, etc.

Future Work

The proposed architecture has many future research directions. The manufactured boards can be replaced with ASIC chips with research funding. As demonstrated in the proposed hardware architecture figure, all these computations can be done at logic level using FPGAs which can be, eventually, replaced with ASIC chips, too. An alert system upgrades are also possible where mobile applications could be created rather than web-page based control method. Since the proposed work has a lot of potential research opportunities in the emerging field of artificial intelligence, it is likely to create big impact to the design of such application-specific integrated circuit

for offering low-latency and inexpensive computations.

BIBLIOGRAPHY

[1] A. Gajjar, Y. Zhang, and X. Yang, "Demo Abstract: A Smart Building System Integrated with An Edge Computing Algorithm and IoT Mesh Networks," The Second ACM/IEEE Symposium on Edge Computing (SEC2017), Article No. 35, Oct. 2017

[2] X. Yang and X. He, "Establishing a BLE Mesh Network using Fabricated CSRmesh Devices," The 2nd ACM/IEEE Symposium on Edge Computing (SEC2017), Article No. 34, Oct. 2017.

[3] X. Yang , L. Wu, etc., "A Vision of Fog Systems with Integrating FPGAs and BLE Mesh Network," ournal of Communications (ISSN: 1796-2021), Vol. 14, No. 3, pp. 210-215, 2019. doi: 10.12720/jcm.14.3.210-215

[4] M. Wang, G. Zhang, C. Zhang, J. Zhang and C. Li, "An IoT-based appliance control system for smart homes," 2013 Fourth International Conference on Intelligent Control and Information Processing (ICICIP), Beijing, 2013, pp. 744-747. doi: 10.1109/ICICIP.2013.6568171

[5] A. Sajid, H. Abbas and K. Saleem, "Cloud-Assisted IoT-Based SCADA Systems Security: A Review of the State of the Art and Future Challenges," in IEEE Access, vol. 4, pp. 1375-1384, 2016. doi: 10.1109/ACCESS.2016.2549047

[6] H. Suo, J. Wan, C. Zou and J. Liu, "Security in the Internet of Things: A Review," 2012 International Conference on Computer Science and Electronics Engineering, Hangzhou, 2012, pp. 648-651. doi: 10.1109/ICCSEE.2012.373

[7] Gareth Halfacree and Eben Upton. 2012. Raspberry Pi User Guide (1st ed.). Wiley Publishing.

[8] Warren Gay. 2014. Raspberry Pi Hardware Reference (1st ed.). Apress, Berkely, CA, USA.

[9] M. Sahani, C. Nanda, A. K. Sahu and B. Pattnaik, "Web-based online embedded door access control and home security system based on face recognition," 2015 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2015], Nagercoil, 2015, pp. 1-6. doi: 10.1109/ICCPCT.2015.7159473

[10] A. Patil and M. Shukla, "Implementation of Classroom Attendance System Based on Face Recognition in Class," International Journal of Advances in Engineering & Technology, vol. 7, (3), pp. 974-979, 2014

[11] W. Shi, J. Cao, Q. Zhang, Y. Li and L. Xu, "Edge Computing: Vision and Challenges," in IEEE Internet of Things Journal, vol. 3, no. 5, pp. 637-646, Oct. 2016. doi: 10.1109/JIOT.2016.2579198

[12] Z. Zhang, M. C. Y. Cho, C. Wang, C. Hsu, C. Chen and S. Shieh, "IoT Security: Ongoing Challenges and Research Opportunities," 2014 IEEE 7th International Conference on Service-Oriented Computing and Applications, Matsue, 2014, pp. 230-234. doi: 10.1109/SOCA.2014.58

[13] I. Gupta, V. Patil, C. Kadam and S. Dumbre, "Face detection and recognition using Raspberry Pi," 2016 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE), Pune, 2016, pp. 83-86. doi: 10.1109/WIECON-ECE.2016.8009092

[14] S. K. Datta and C. Bonnet, "An edge computing architecture integrating virtual IoT devices," 2017 IEEE 6th Global Conference on Consumer Electronics (GCCE), Nagoya, 2017, pp 1-3. doi: 10.1109/GCCE.2017.8229253

[15] J. Grover and R. M. Garimella, "Reliable and Fault-Tolerant IoT-Edge Architecture," 2018 IEEE SENSORS, New Delhi, 2018, pp. 1-4. doi: 10.1109/ICSENS.2018.8589624

[16] H. El-Sayed et al., "Edge of Things: The Big Picture on the Integration of Edge, IoT and the Cloud in a Distributed Computing Environment," in IEEE Access, vol. 6, pp. 1706-1717, 2018. doi: 10.1109/ACCESS.2017.2780087

[17] Y. Li and S. Wang, "An Energy-Aware Edge Server Placement Algorithm in Mobile Edge Computing," 2018 IEEE International Conference on Edge Computing (EDGE), San Francisco, CA, 2018, pp. 66-73. doi: 10.1109/EDGE.2018.00016

[18] P. Ren, X. Qiao, J. Chen and S. Dustdar, "Mobile Edge Computing a Booster for the Practical Provisioning Approach of Web-Based Augmented Reality," 2018 IEEE/ACM Symposium on Edge Computing (SEC), Seattle, WA, 2018, pp. 349-350. doi: 10.1109/SEC.2018.00041

[19] J. Hochstetler, R. Padidela, Q. Chen, Q. Yang and S. Fu, "Embedded Deep Learning for Vehicular Edge Computing," 2018 IEEE/ACM Symposium on Edge Computing (SEC), Seattle, WA, 2018, pp. 341-343. doi: 10.1109/SEC.2018.00038

[20] F. Wei, S. Chen and W. Zou, "A greedy algorithm for task offloading in mobile edge computing system," in China Communications, vol. 15, no. 11, pp. 149-157, Nov. 2018. doi: 10.1109/CC.2018.8543056

[21] H. Badri, T. Bahreini, D. Grosu and K. Yang, "A Sample Average Approximation-Based Parallel Algorithm for Application Placement in Edge Computing Systems," 2018 IEEE International Conference on Cloud Engineering (IC2E), Orlando, FL, 2018, pp. 198-203. doi: 10.1109/IC2E.2018.00044

[22] P. Irgens, C. Bader, Theresa L, D. Saxena, C. Ababei, "An efficient and cost effective FPGA based implementation of the Viola-Jones face detection algorithm," HardwareX, (1), 2017, pp 68-75. https://doi.org/10.1016/j.ohx.2017.03.002

[23] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, Kauai, HI, USA, 2001, pp. I-I. doi: 10.1109/CVPR.2001.990517

[24] Ming-Hsuan Yang, D. J. Kriegman and N. Ahuja, "Detecting faces in images: a survey," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 1, pp. 34-58, Jan. 2002. doi: 10.1109/34.982883

[25] A. M. Caulfield et al., "A cloud-scale acceleration architecture," 2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), Taipei, 2016, pp. 1-13. doi: 10.1109/MICRO.2016.7783710

[26] A. Putnam et al., "A Reconfigurable Fabric for Accelerating Large-Scale Datacenter Services," in IEEE Micro, vol. 35, no. 3, pp. 10-22, May-June 2015. doi: 10.1109/MM.2015.42

[27] Nexys 4 Artix-7 FPGA Trainer Board, 2017. Available at `https://reference.digilentinc.com/_media/nexys4-ddr:nexys4ddr_rm.pdf`.

[28] X. Yang, N. Wu, and J. Andrian, "A Novel Bus Transfer Mode: Block Transfer and A Performance Evaluation Methodology," Elsevier, Integration, the VLSI Journal, Vol. 52, PP. 23-33, Jan. 2016. doi: 10.1016/j.vlsi.2015.07.012

[29] X. Yang, N. Wu, and J. Andrian, "Comparative Power Analysis of An Adaptive Bus Encoding Method on The MBUS Structure," Journal of VLSI Design, Vol. 2017, Article ID 4914301, PP. 1-7, May 2017. doi: 10.1155/2017/4914301

[30] X. Yang and J. Andrian, "A High Performance On-Chip Bus (MSBUS) Design and Verification," IEEE Trans. Very Large Scale Integr. Syst. (TVLSI), Vol. 23, Issue: 7, PP. 1350-1354, Sept. 2015. doi: 10.1109/TVLSI.2014.2334351

[31] Y. Zhang, X. Yang, and L. Wu, et al, "Hierarchical Synthesis of Approximate Multiplier Design for Field-Programmable Gate Arrays (FPGA)-CSRmesh System," Intl. Journal of Compt. Applications (IJCA), Vol. 180, No. 17 PP. 1-7, Feb. 2018. doi: 10.5120/ijca2018916380

[32] M. Fan, Q. Han, and X. Yang, "Energy Minimization for On-Line Real-Time Scheduling with Reliability Awareness," Elsevier Journal of Systems and Software (JSS), Vol. 127, PP. 168176, May 2017. doi: 10.1016/j.jss.2017.02.004

[33] S. Singh, D. Chauhan, M. Vatsa, R. Singh, "A Robust Skin Color Based Face Detection Algorithm," Tamkang Journal of Science and Engineering, Vol. 6, No. 4, PP. 227-234, 2003.

[34] P. Viola, M. Jones, "Robust Real-Time Face Detection," Intl. Journal of Computer Vision, Vol. 57, PP.137-154, May 2004.

[35] S. Rana, M. P. Deepu, S. Sivanantham and K. Sivasankaran, "Face detection system using FPGA," 2015 Online International Conference on Green Engineering and Technologies (IC-GET), Coimbatore, 2015, pp. 1-4. doi: 10.1109/GET.2015.7453793

[36] J. Matai, A. Irturk and R. Kastner, "Design and Implementation of an FPGA-Based Real-Time Face Recognition System," 2011 IEEE 19th Annual International Symposium on Field-Programmable Custom Computing Machines, Salt Lake City, UT, 2011, pp. 97-100. doi: 10.1109/FCCM.2011.53

[37] A. Mebrek, L. Merghem-Boulahia and M. Esseghir, "Efficient green solution for a balanced energy consumption and delay in the IoT-Fog-Cloud computing," 2017 IEEE 16th International Symposium on Network Computing and Applications (NCA), Cambridge, MA, 2017, pp. 1-4. doi: 10.1109/NCA.2017.8171359

[38] S. W. Kum, J. Moon and T. Lim, "Design of fog computing based IoT application architecture," 2017 IEEE 7th International Conference on Consumer Electronics - Berlin (ICCE-Berlin), Berlin, 2017, pp. 88-89. doi: 10.1109/ICCE-Berlin.2017.8210598

[39] P. Zenker, S. Krug, M. Binhack and J. Seitz, "Evaluation of BLE Mesh capabilities: A case study based on CSRMesh," 2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN), Vienna, 2016, pp. 790-795. doi: 10.1109/ICUFN.2016.7537146

[40] M. Alrowaily and Z. Lu, "Secure Edge Computing in IoT Systems: Review and Case Studies," 2018 IEEE/ACM Symposium on Edge Computing (SEC), Seattle, WA, 2018, pp. 440-444. doi: 10.1109/SEC.2018.00060

[41] S. Singh, "Optimize cloud computations using edge computing," 2017 International Conference on Big Data, IoT and Data Science (BID), Pune, 2017, pp. 49-53. doi: 10.1109/BID.2017.8336572

[42] A. Yousefpour, G. Ishigaki and J. P. Jue, "Fog Computing: Towards Minimizing Delay in the Internet of Things," 2017 IEEE International Con-

ference on Edge Computing (EDGE), Honolulu, HI, 2017, pp. 17-24. doi: 10.1109/IEEE.EDGE.2017.12

[43] A. Gajjar, X. Yang, H. Koc, et al., "Mesh-IoT Based System For Large-Scale Environment," 5th Annual Conf. on Computational Science & Computational Intelligence (CSCI2018), Accepted, In Press, Oct. 2018.

[44] Y. Zhang, X. Yang, A. Gajjar, etal., " Exploring Slice-Energy Saving on an Video Processing FPGA Platform with Approximate Computing," In Proceedings of the 2018 2nd International Conference on Algorithms, Computing and Systems (ICACS '18). ACM, New York, NY, USA, 138-143. doi: 10.1145/3242840.3242852

[45] A. Gajjar, X. Yang, et al., "An FPGA Synthesis of Face Detection Algorithm using HAAR Classifiers," Intl. Conference on Algorithms, Computing, and Systems (ICACS2018), PP.133-137, July 27-29, Beijing China, 2018. doi: 10.1145/3242840.3242851

[46] Cisco Public, "Cisco Global Cloud Index: Forecast and Methodology,2015-2020," White Paper, June, 2015. Available at `http://www.cisco.com/c/dam/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.pdf`.

[47] X. Yang and W. Wen, "Design of A Pre-Scheduled Data Bus (DBUS) for Advanced Encryption Standard (AES) Encrypted System-on-Chips (SoCs)," The 22nd Asia and South Pacific Design Automation Conference (ASP-DAC 2017), PP. 1-6, Chiba, Japan, February, 2017. doi: 10.1109/ASPDAC.2017.785837

[48] X. Yang, "A High Performance Advanced Encryption Standard (AES) Encrypted On-Chip Bus Architecture for Internet-of-Things (IoT) System-on-Chips (SoC)," Ph. D dissertation, Florida International University, 2016.

[49] X. Yang and JH Andrian, "Advanced bus architecture for aes-encrypted high-performance internet-of-things (iot) embedded systems," US Patent, App. 15/130, 298, 2017.

[50] X. Yang and J. Andrian, "A low-cost and high-performance embedded system architecture and an evaluation methodology," 2014 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), PP. 240–243, 2014.

[51] X. Yang, W. Wen, and M. Fan, "Improving AES Core Performance via An Advanced ASBUS Protocol," ACM Journal of Emerging Technologies in Computing (ACM JETC), 2017.

[52] X. Yang, Y. Zhang, and L. Wu, "A Scalable Image/Video Processing Platform with Open Source Design and Verification Environment," 20th Intl. Symposium on Quality Electronic Design (ISQED 2019) , Accepted, In Press, Dec. 2018.

[53] J. Thota, P. Vangali, and X. Yang, "Prototyping An Autonomous Eye-Controlled System (AECS) using Raspberry-Pi on Wheelchairs," International Journal of Computer Applications, Vol. 158, No. 8, PP. 1-7, 2017.

[54] P. Vangali and X Yang, "A Compression Algorithm Design and Simulation for Processing Large Volumes of Data from Wireless Sensor Networks," Vol. 7, No. 4, Communications on Applied Electronics (CAE), July 2017.

[55] H He, et al., "Dual long short-term memory networks for sub-character representation learning," Information Technology-New Generations, PP. 421-426, 2018.

[56] L. Nwosu, et al., "Deep convolutional neural network for facial expression recognition using facial parts," 2017 IEEE 15th Intl. Conf on Dependable, Autonomic and Secure Computing, PP. 1318-1321, Nov. 2017.

[57] J. Gopaluni, I. Unwala, J. Lu, and X. Yang, "Implementation of GUI for OpenThread," 5th Annual Conf. on Computational Science & Computational Intelligence (CSCI2018), Accepted, In Press, Oct. 2018.

[58] S. Bentin, T. Allison, A. Puce, E. Perez, G. McCarthy, "Electrophysiological Studies of Face Perception in Humans," Journal of Cognitive Neuroscience, 8:6, pp 551-565, 1998.

[59] R. Diamond, S. Carey, Why Faces Are and Are Not Special: An Effect of Expertise, Journal or Experimental Psychology: General, 1986, Vol. 115. No. 2, 107-117.

[60] G. Bradski, "The OpenCV Library," Dr. Dobbs Journal of Software Tools. `https://docs.opencv.org/3.4.3/`

[61] D. King., "Dlib-ml: A Machine Learning Toolkit," Journal of Machine Learning Research 10, pp. 1755-1758, 2009 `http://dlib.net/`

VITA

Archit Gajjar

2016	B.Tech., Information and Communication Technology Dhirubhai Ambani Institute of Information and Communication Technology Gujarat, India
2019	M.S., Computer Engineering University of Houston Clear Lake Houston, TX

PUBLICATIONS

A. Gajjar, S. Dave, and X. Yang, "IoT-Edge-Server Based Embedded System for Wide-Range Habitats," Journal of Systems Architecture, Work-in-Process, 2019.

A. Gajjar, Y. Zhang, and X. Yang, "Demo Abstract: A Smart Building System Integrated with An Edge Computing Algorithm and IoT Mesh Networks," The Second ACM/IEEE Symposium on Edge Computing (SEC2017), Article No. 35, Oct. 2017.

A. Gajjar, X. Yang, H. Koc, et al., "Mesh-IoT Based System For Large-Scale Environment," 5th Annual Conference on Computational Science & Computational Intelligence (CSCI2018), Accepted , In Press, 2018.

A. Gajjar, X. Yang, et al., "An FPGA Synthesis of Face Detection Algorithm using HAAR Classifiers," International Conference on Algorithms, Computing, and Systems (ICACS2018), PP.133-137, July 27-29, Beijing China, 2018.

K. Vaca, **A. Gajjar**, X. Yang, et al., "Real-Time Automatic Music Transcription (AMT) with Zync FPGA," IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Under Review, 2019.

X. Yang, L. Wu, **A. Gajjar**, et al., "A Vision of Fog Systems with Integrating FPGAs and BLE Mesh Network," Journal of Communications (ISSN: 1796-2021), Vol. 14, No. 3, pp. 210-215, 2019.

Y. Zhang, X. Yang, **A. Gajjar**, et al., "Exploring Slice-Energy Saving on An Video Processing FPGA Platform with Approximate Computing," International Conference on Algorithms, Computing, and Systems (ICACS2018), PP.138-143, July 27-29, Beijing China, 2018.

Y. Zhang, X. Yang, **A. Gajjar**, et al., "Hierarchical Synthesis of Approximate Multiplier Design for Field-Programmable Gate Arrays (FPGA)-CSRmesh System," International Journal of Computer Applications (IJCA), Vol. 180, No. 17 PP. 1-7, Feb. 2018.

A. Gajjar, X. Yang, et al., "Poster presentation - Mesh-IoT Based Smart and Secure Monitoring System for Wide-Range Territory," IEEE Innovation and Automation Conference, Gilruth Recreation Center, NASA-JSA, Houston, November, 2018.

A. Gajjar, X. Yang, "Poster presentation - A Wide Area IoT Mesh Network With Edge Computing," IEEE Innovation and Automation Conference, Gilruth Recreation Center, NASA., October 2017.

A. Gajjar, X. Yang, "Poster presentation - A Smart Home/Building System Integrated with An Edge Computing Algorithm and CSRmesh Networks," Houston Robotics and AI Day, July 2017.

X. Yang, Y. Zhang, **A. Gajjar**, H. Schmoyer, and N. Ly, "Learning-on-Chip: Facial Detection with Approximations of FPGA Computing," Houston Robotics and AI Day, UHCL, August 2018.