

بسم الله الرحمن الرحيم



Palestine Polytechnic University

College of IT and Computer Engineering

Department of Computer Engineering

Accurate Low-Cost Earthquake Detection and Alarm System Based on WSN, ESP32 and
Machine Learning

Team Members

Mariam Hasanat

Lama Ayaydah

Supervisor

Dr. Eng. Hani Salah

June 2024

Certification and Anti-Plagiarism Declaration

This is to declare that the graduation project produced under the supervision of Dr. Hani Salah having the title “Accurate Low-Cost Earthquake Detection and Alarm System Based on WSN, ESP32 and Machine Learning” was prepared by Mariam Hasanat and Lama Ayaydeh in partial fulfillment of the requirements for the Bachelor degree of Computer Systems Engineering.

The content of the project is original, and all referenced parts are cited properly and have been used to support the content. We certify that we will not commit any plagiarism, cheating, or any other academic integrity violation. We will be responsible and liable for any consequence if a violation of this declaration is proven.

Date:

Graduation project group's student(s):

Mariam Hasanat

Lama Ayaydeh

Signature:

Signature:

ABSTRACT

Earthquakes can occur suddenly and unexpectedly, causing widespread damage and loss of life. For example, the 2023 Syria-Turkey earthquake resulted in the death of over 50,000 people. Primary Wave (P-wave) and Secondary Wave (S-wave) are two types of seismic waves generated by earthquakes. P-wave is faster than S-wave, However, S-waves can do more damage than P-waves, and P-waves are used to detect the earthquake. Early earthquake warnings can help reduce these damages by giving people time to protect their lives. This leads us to propose a highly accurate, cost-effective, and accessible earthquake early warning system.

The system utilizes a wireless sensor network to collect data from the ground using acceleration sensor ADXL335 and pass it to the ESP32 microcontroller to detect and predict earthquakes using a logistic regression machine learning model. The system also includes an alert mechanism using buzzers capable of notifying users of a near earthquake and providing audible alerts on mobile applications to increase speed and allow users sufficient time to take safety measures. Finally, the system uses a shaking table that significantly simulates earthquake conditions for evaluation and testing purposes.

After picking 6000 earthquake and not-earthquake acceleration samples for x, y, and z from the shake table, these samples are divided and used to calculate the mean and standard deviation, each 100 samples together will give one vector of x, y, and z standard deviation values, and number of samples after calculating stand deviation is 43 per second, these sample will be used later to train and test the logistic regression model, that gives us a very high accuracy of 95.3%, which mean predict the earthquake in 0.5-1 sec. In the WSN, we implement star topology of two nodes, the 100% confidence of an earthquake happening depends on both nodes' values of alert, and the time arrival of alert to the mobile application depends on the internet latency, the total time needed from detecting the earthquake to mobile alerting is 7.5-12.5sec. mobile application is designed in a way that achieves easy and simple user interaction.

ACKNOWLEDGEMENT

In the name of Allah, Most Gracious, Most Merciful, for giving us the strength and knowledge to complete this project.

We would like to express our heartfelt thanks to our parents for their unconditional love and support, which has been invaluable to us throughout our lives. We know that words cannot fully convey our gratitude, but we hope that our actions and achievements will reflect the depth of our appreciation.

We also want to express our gratitude to our families and friends, who have provided us with endless encouragement and support, especially during the completion of this project. We could not have done it without their guidance and motivation.

Finally, we would like to thank our supervisor, Dr. Hani Salah, for his valuable help and advice throughout the project. His expertise and guidance were essential to our success. We would also like to thank Eng. Wael Al-takrouri and Dr. Hashem Tamimi for their expertise and assistance, which has been essential to the success of this project. We are truly grateful for their support and encouragement, and we hope to continue learning from their wisdom and expertise in the future.

TABLE OF CONTENT

Abstract	3
Acknowledgement	4
Table of Content	5
List of Tables	7
List of Figures	7
Chapter 1 Introduction	8
1.1 Motivation and Importance	9
1.2. Problem Statement	9
1.3. Project Objectives	9
1.4. Gantt Chart	10
Chapter 2 Background and Literature Review	11
2.1. Background	12
2.1.1. Earthquakes	12
2.1.2. P-wave and S-wave	12
2.1.3. Wireless Sensor Networks (WSNs)	13
2.1.4. Machine Learning	17
2.1.5. ESP32 Microcontroller	18
2.1.6. Seismometers	18
2.1.7. ADXL335	18
2.1.8. MPU6050	19
2.1.9. HMC5883	19
2.1.12. Buzzer	20
2.1.13 MIT App Inventor	20
2.1.14 Flutter	20
2.1.15 Grafana	21
2.1.15 Google Firebase	21
2.2. Literature Review	22
Chapter 3 System Design	23
3.1. High-Level System Description and General Block Diagram	24
3.1.1. Data collection	24
3.1.2. Data Processing	25
3.1.3. Earthquake Detection	25

3.1.4. User Alert	25
3.2. Design Options.....	26
3.2.1. Hardware Components	26
3.2.2. Software Components	29
Chapter 4 Implementation.....	31
4.1 Hardware Implementation	32
4.1.1. Shaking table.....	32
4.1.2. Hardware System Connections	33
4.1.3 Evaluation Setup	33
4.2 Software Implementation	34
4.2.1 Logistic Regression Earthquake Detection Model	34
Collect the Data.....	34
Data Preparation	34
Dataset Partitioning	34
Coding	35
4.2.2 WSN Implementation	36
4.2.3 Firebase Database	37
Connecting ESP32 with Firebase.....	38
Code	38
Connecting ESP32 with MIT App Inventor using Firebase	40
4.2.4 Mobile App Implementation	41
4.2.5 Web Application Implementation	42
WEB Implementation.....	42
WEB Deployment.....	45
Connect WEB App with Firestore.....	45
Chapter 5 Evaluation	46
5.1 Results and Discussion	47
Chapter 6 Summary and Future Works	49
6.1 Summary	50
6.2 Future Work.....	50
References	52
Appendix A- Schematic diagram	55
Appendix B- ESP32 Diagram	56
Appendix C – ANN Results.....	57

LIST OF TABLES

<i>Table 1: Specifications of the selected sensors.....</i>	<i>28</i>
---	-----------

LIST OF FIGURES

<i>Figure 1 - Gantt Chart.....</i>	<i>10</i>
<i>Figure 2 - Earthquake with P-wave and S-wave arrivals recorded in Mongolia and observed at 307 km distance (Adapted from [10])</i>	<i>13</i>
<i>Figure 3 - Overview of WSN component Implementation for Earthquake Detection</i>	<i>14</i>
<i>Figure 4 - Ad-hoc Topology</i>	<i>14</i>
<i>Figure 5 - Star Topology</i>	<i>15</i>
<i>Figure 6 - Mesh Topology</i>	<i>15</i>
<i>Figure 7 - Tree (Hierarchical) Topology</i>	<i>16</i>
<i>Figure 8 - Cluster Topology</i>	<i>16</i>
<i>Figure 9 - Hybrid Topology</i>	<i>17</i>
<i>Figure 10 - ESP32 Microcontroller</i>	<i>18</i>
<i>Figure 11 - ADXL335 Accelerometer Sensor</i>	<i>19</i>
<i>Figure 12 - MPU6050 Gyroscope Sensor</i>	<i>19</i>
<i>Figure 13 - HMC5883 Sensor</i>	<i>20</i>
<i>Figure 14 - Buzzer Device</i>	<i>20</i>
<i>Figure 15 - High-Level System Description</i>	<i>24</i>
<i>Figure 16 - Data Collection Block Diagram</i>	<i>25</i>
<i>Figure 17 - Data Processing</i>	<i>25</i>
<i>Figure 18 - User Alert</i>	<i>26</i>
<i>Figure 19 - Shaking Table</i>	<i>32</i>
<i>Figure 20 - Hardware Connections System</i>	<i>33</i>
<i>Figure 21 - WSN IMPLEMENTATION</i>	<i>37</i>
<i>Figure 22 - FIREBASE Real-Time DATABASE</i>	<i>37</i>
<i>Figure 23 - MIT App Inventor Viewer Screen</i>	<i>40</i>
<i>Figure 24 - MIT APP INVENTOR Code Blocks SCREEN</i>	<i>41</i>
<i>Figure 25 - Introduction Screen</i>	<i>41</i>
<i>Figure 26 - MACHINE LEARNING ACCURACY</i>	<i>47</i>
<i>Figure 27 - Confusion Matrix</i>	<i>57</i>
<i>Figure 28 - ROC Curves</i>	<i>57</i>
<i>Figure 29 - Performance Plot</i>	<i>58</i>

CHAPTER 1

INTRODUCTION

This chapter is structured into four sections: First, Section 1.1 describes the motivations and importance of the project. Next, Section 1.2 outlines the problem statement. After that, Section 1.3 describe the project objectives. Finally, Section 1.4 has the time line plan and Gantt chart.

1.1 MOTIVATION AND IMPORTANCE

Earthquakes are one of the most destructive natural disasters that occur beyond human control and pose a serious threat to human lives and properties. According to the World Health Organization, more than 750,000 people died due to earthquakes and related hazards between 1998-2017 [1]. Moreover, earthquakes can cause severe damage to infrastructure, the environment, and the economy, resulting in huge losses and recovery costs. Therefore, there is a significant need for effective and reliable earthquake detection and alarm systems that can provide timely and accurate warnings to the public and authorities, and enable appropriate actions and responses to mitigate the impact of earthquakes.

Predicting earthquakes is described as challengeable task, and based on a lot of scientific studies focusing on earthquake prediction (e.g., see [2], [3], [4]).

1.2. PROBLEM STATEMENT

Existing earthquake detection and alarm systems have one or more of the following limitations and drawbacks: First, they rely on expensive and sophisticated seismometers that are not widely available or accessible, especially in developing countries or remote areas. Second, even its sensitivity is high, but it is not available in our country due to the occupation restrictions. Third, they lack user-friendly interfaces or features, which can hinder the communication and comprehension of the alerts.

1.3. PROJECT OBJECTIVES

- **Accurate Seismic Detection:** the system should achieve accuracy in seismic detection.
- **Low Cost:** the system should be low cost; hence it will be available then other people can buy it.
- **Early and Real-time Warning:** The system should be fast in predict the earthquake, to achieve that the speed of prediction should be about 10sec to let the alert access the people in the right time.

- **Easy and Simple User Interaction:** One of the system parts is to have a user interface in order to allow users to interact with the system easily and display data and charts.

1.4. GANTT CHART

The Gantt chart as shown in Figure 1 outlines the timeline for a system's design and development, starting with planning and ending with a conclusion. It serves as a visual tool for project management, aiding in understanding the sequence of tasks and keeping teams organized and on track.

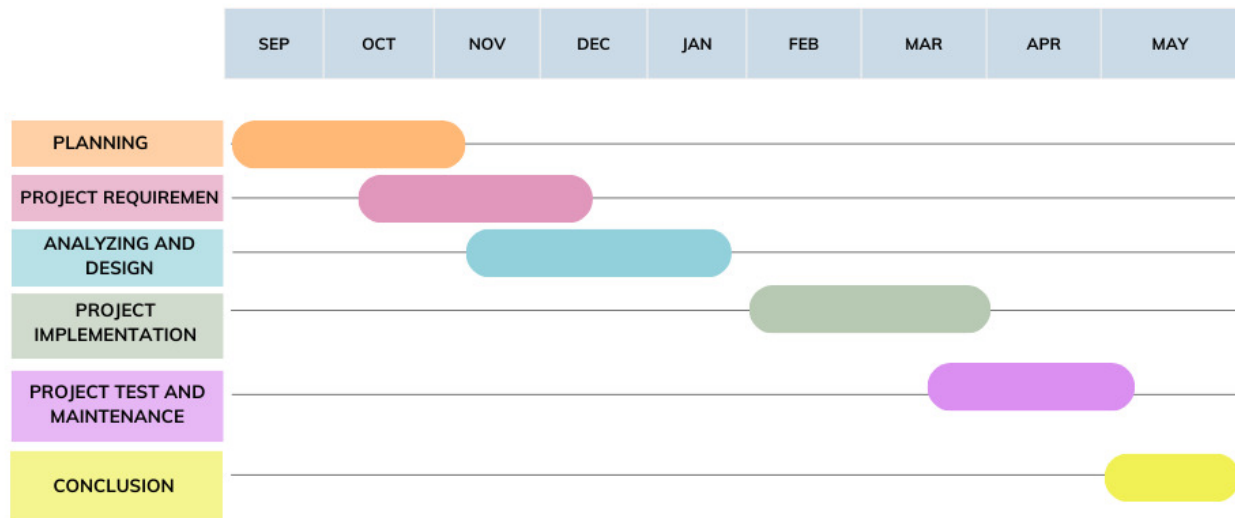


FIGURE 1 - GANTT CHART

CHAPTER 2

BACKGROUND AND LITERATURE REVIEW

This chapter is structured into two sections: First, Section 2.1 describes the background of the project. And, Section 2.2 describes literature Review and related works.

2.1. BACKGROUND

2.1.1. EARTHQUAKES

Earthquakes are natural phenomena that occur when the earth's tectonic plates move and release energy. They can cause severe damage to buildings, infrastructure, and human lives, especially in densely populated areas. Therefore, it is important to detect and monitor earthquakes, and to provide early warning and alert to the affected population. According to the United States Geological Survey (USGS), there are about 500,000 detectable earthquakes in the world each year, of which about 100,000 can be felt, and about 100 cause damage [5]. Some of the most devastating earthquakes in history include the 2010 Haiti earthquake, which killed over 316,000 people and displaced over 1.3 million people [6], the 2011 Japan earthquake and tsunami, which killed over 18,000 people and triggered a nuclear meltdown [7], and the 2023 Turkey and northern Syria which killed in Turkey over 45 089, and 6000 deaths in northern Syria [8].

2.1.2. P-WAVE AND S-WAVE

Primary Wave (P-wave) and Secondary Wave (S-wave) are two types of seismic waves that are generated by earthquakes. P-wave is a compression wave that travels faster than S-wave, which is a transverse wave and it has speed 4 to 8 km/sec.

S-waves are slower than P-waves roughly 1.7 times as shown in Figure 2. P-waves are used to detect the earthquake. S-wave has a lower frequency than P-wave, where S-wave has a frequency 1Hz and P-wave ranges from 1-10Hz. However, S-waves can do more damage than P-waves [9]. The amplitude and frequency of P-wave and S-wave can be used to estimate the magnitude and intensity of the earthquake. By detecting and measuring P-waves, the earthquake detection and alarm system can provide useful information and warnings to the users.

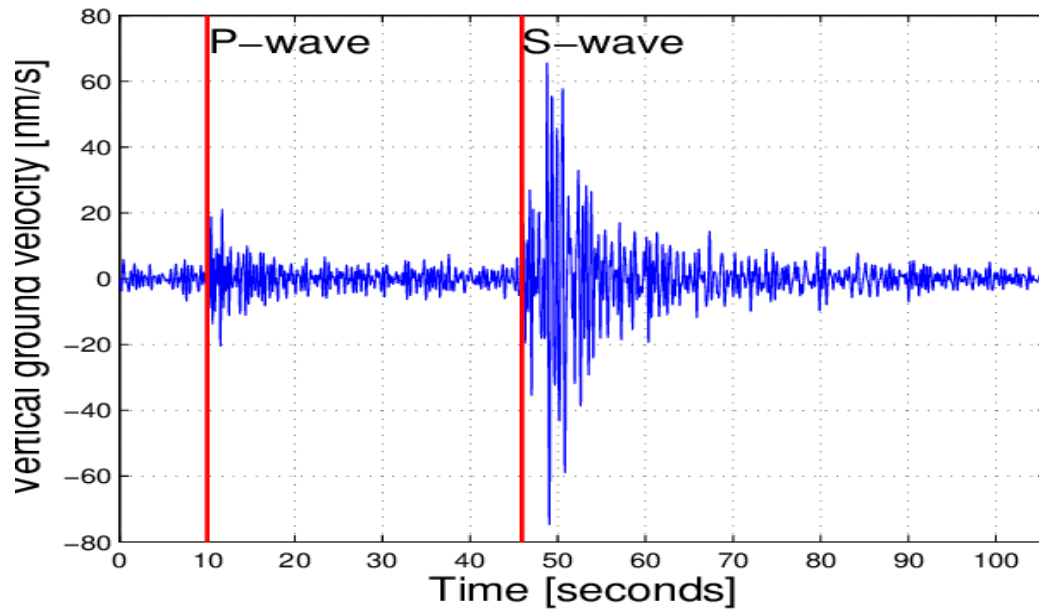


FIGURE 2 - EARTHQUAKE WITH P-WAVE AND S-WAVE ARRIVALS RECORDED IN MONGOLIA AND OBSERVED AT 307 KM DISTANCE (ADAPTED FROM [10])

2.1.3. WIRELESS SENSOR NETWORKS (WSNs)

WSNs consist of a network of small, battery-powered sensors that can be deployed in a wide area. These sensors can detect changes in the environment, such as ground movement, and send this data to a central processing unit [11], Figure 3 shows an overview of WSN components. The main components of a WSN are:

- **Sensors:** that measure the environmental variables, such as temperature, humidity, vibration, pressure, or motion. They convert the sensor signals into electrical signals that can be transmitted wirelessly.
- **Nodes:** These are the devices that receive the data from the sensors and send it to the base station or the gateway. They consist of a microcontroller, a transceiver, an external memory, and a power source.
- **Base Station or Gateway:** This device bridges the WSN and the external network, such as the internet or a cloud service.

- **Application Software:** This is the software that processes the data collected by the WSN and provides useful information or actions based on the data. It can be hosted on a computer, a smartphone, or a cloud server [12].

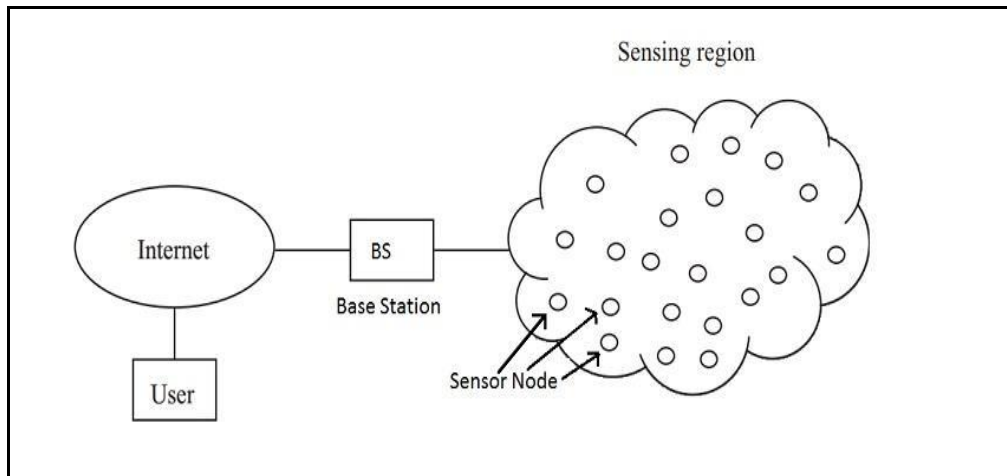


FIGURE 3 - OVERVIEW OF WSN COMPONENT IMPLEMENTATION FOR EARTHQUAKE DETECTION

Wireless Sensor Networks Topologies

Wireless Sensor Networks (WSNs) utilize various topologies to organize sensor nodes, each with its advantages and use cases. Here are some common WSN topologies.

Ad-hoc Topology

In an ad-hoc approach, each sensor node communicates directly with its neighbors without a central entity [13], as shown in Figure 4.

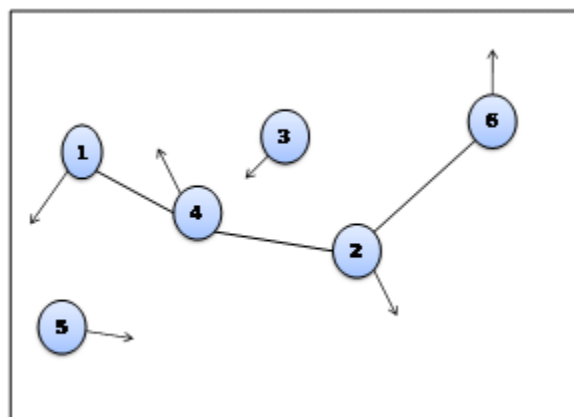


FIGURE 4 - AD-HOC TOPOLOGY

Star Topology

In a star topology, each sensor node directly communicates with a central base station as shown in Figure 5, [13].

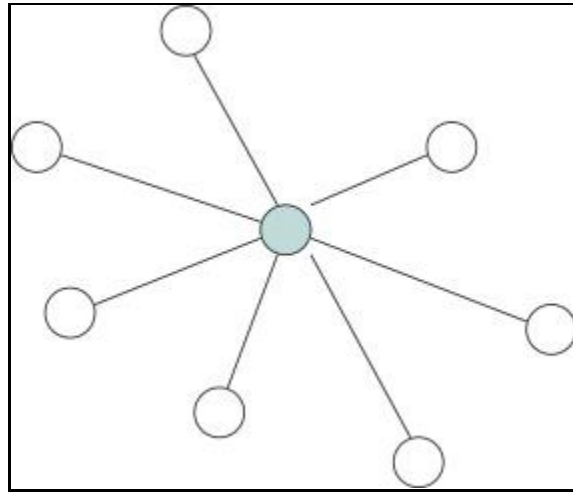


FIGURE 5 - STAR TOPOLOGY

Mesh Topology

Mesh topology allows each sensor node to communicate with multiple other nodes, creating multiple pathways for data transmission as shown in Figure 6.

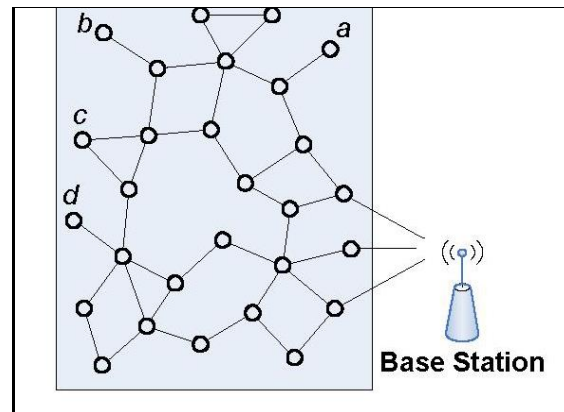


FIGURE 6 - MESH TOPOLOGY

Tree (Hierarchical) Topology

In tree topology, as shown in Figure 7, sensor nodes are organized hierarchically with parent-child relationships. Data is transmitted from leaf nodes through parent nodes to the base station [13].

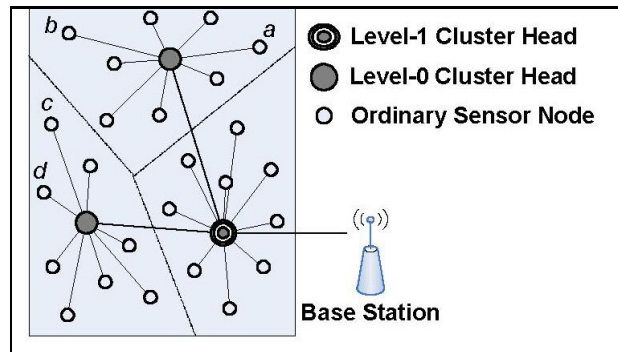


FIGURE 7 - TREE (HIERARCHICAL) TOPOLOGY

Cluster Topology

Cluster topology groups sensor nodes into clusters, each led by a cluster head that aggregates data and communicates with the base station as shown in Figure 8, [13].

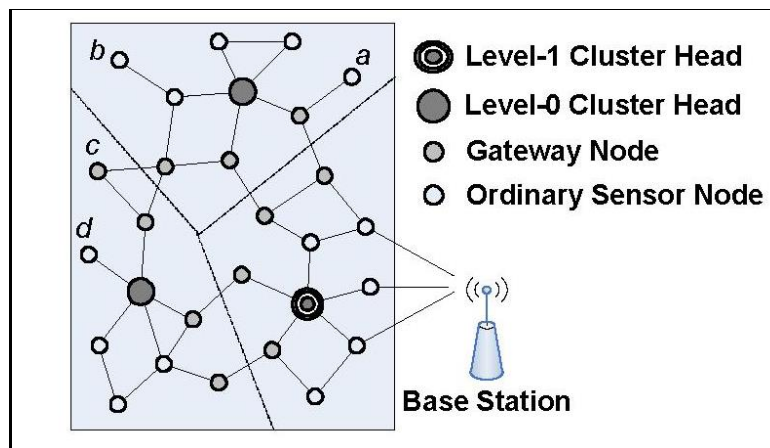


FIGURE 8 - CLUSTER TOPOLOGY

Hybrid Topology

Hybrid topology combines elements of various topologies, such as star and mesh as shown in Figure 9, to balance their strengths and weaknesses [13].

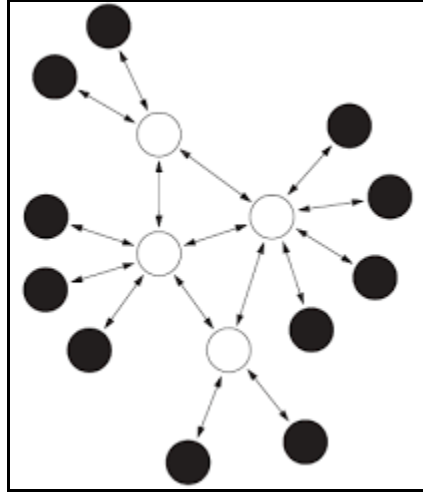


FIGURE 9 - HYBRID TOPOLOGY

2.1.4. MACHINE LEARNING

Machine Learning is a field of study in artificial intelligence that uses algorithms trained on data sets to create models that enable machines to perform tasks without explicit instructions. Machine learning can be used for various purposes, such as image recognition, natural language processing, data analysis, and more [4]. There are many types of machine learning models, but here we discussed the commonly used models:

- ✓ Logistic regression: a simple and interpretable model that can perform binary classification based on some features, such as magnitude, depth, location, etc. However, logistic regression may not be able to capture the complex patterns. For example, a study by [14] compared logistic regression with other machine learning models.
- ✓ ANN: is a powerful and flexible model that can learn from large and high-dimensional data, and can approximate any nonlinear function. ANN can also be combined with other techniques, such as convolutional neural networks (CNN), recurrent neural networks (RNN), and long short-term memory (LSTM).
- ✓ K-means clustering: is an unsupervised learning model that can group the data into clusters based on some similarity or distance measure.

2.1.5. ESP32 MICROCONTROLLER

The ESP32 microcontroller as shown in Figure 10, is a powerful, versatile, and low-power chip that can handle Wi-Fi, Bluetooth, and Bluetooth LE connectivity, as well as a range of peripherals and sensors. It has two CPU cores that can run up to 240 MHz, and a low-power coprocessor that can perform tasks without waking up the CPU. It also has an external flash and SRAM interface, and supports antenna diversity and adaptive frequency hopping. The ESP32 is suitable for many applications such as earthquake detection systems, voice encoding, music streaming, MP3 decoding, and low-power sensor networks [15].

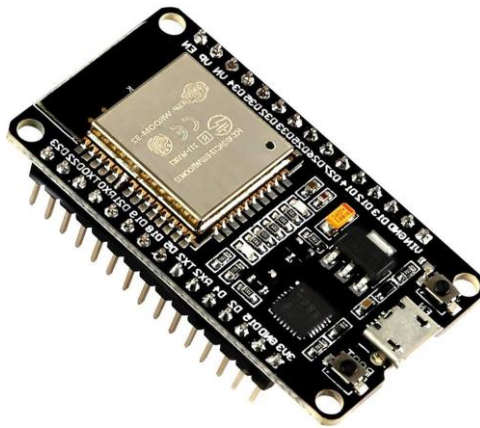


FIGURE 10 - ESP32 MICROCONTROLLER

2.1.6. SEISMOMETERS

A seismometer is a device that measures the ground motion caused by earthquakes, volcanic eruptions, or other seismic events. It consists of a mass suspended from a spring or a pendulum, and a sensor that detects the relative displacement of the mass and the frame. A seismometer can record the amplitude, frequency, and duration of the seismic waves, and provide information about the location, magnitude, and type of the source [16].

2.1.7. ADXL335

The ADXL335 sensor as shown in Figure 11, measures the ground motion acceleration. It has a full scale of $\pm 2g/\pm 4g/\pm 8g/\pm 16g$ and a data rate of 1 Hz to 5.3 kHz [17].

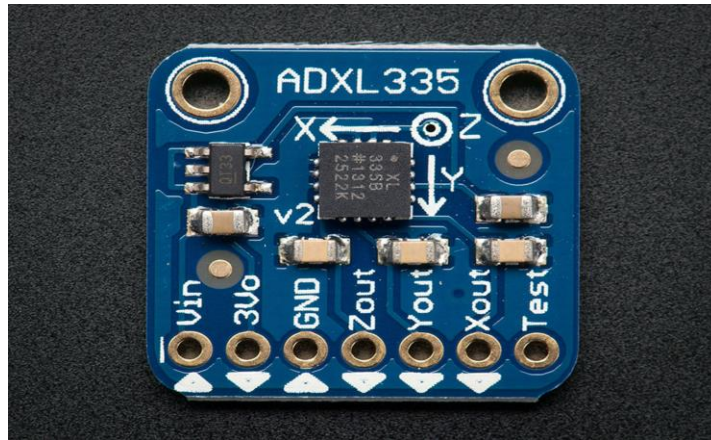


FIGURE 11 - ADXL335 ACCELEROMETER SENSOR

2.1.8. MPU6050

The MPU6050 gyroscope sensor as shown in Figure 12, measures ground motion rotational velocity during earthquakes with a selectable range of ± 250 , ± 500 , ± 1000 , or ± 2000 degrees per second [18].

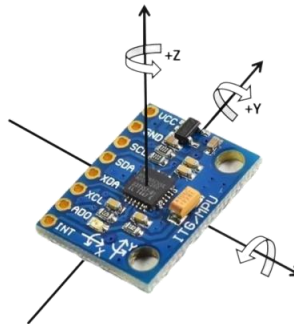


FIGURE 12 - MPU6050 GYROSCOPE SENSOR

2.1.9. HMC5883

The HMC5883 sensor as shown in Figure 13, measures the magnetic field in three axes, making it suitable for detecting magnetic north in compass and navigation applications [19].

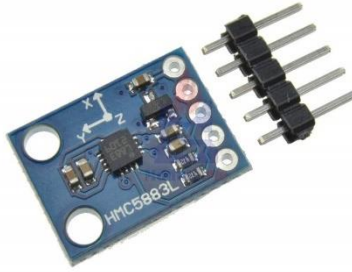


FIGURE 13 - HMC5883 SENSOR

2.1.12. BUZZER

The hardware component makes a sound when electricity flows through it. A buzzer as shown in Figure 14, can have different tones, frequencies, and volumes, depending on its design and specifications [20].

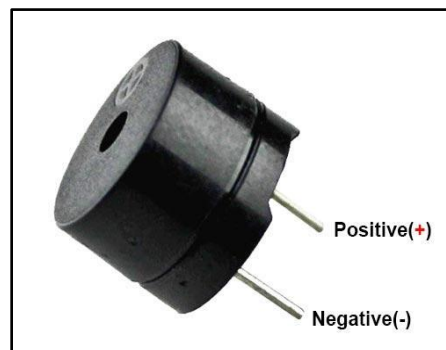


FIGURE 14 - BUZZER DEVICE

2.1.13 MIT APP INVENTOR

MIT App Inventor is a high-level block-based visual programming language, originally built by Google and now maintained by the Massachusetts Institute of Technology. It allows newcomers to create computer applications for two operating systems: Android and iOS, which, as of 25 September 2023, is in beta testing [21].

2.1.14 FLUTTER

Flutter is an open source framework by Google for building beautiful, natively compiled, multi-platform applications from a single codebase [22].

2.1.15 GRAFANA

Grafana is a multi-platform open-source analytics and interactive visualization web application. It can produce charts, graphs, and alerts for the web when connected to supported data sources [23].

2.1.15 GOOGLE FIREBASE

Firebase is a mobile and web application development platform created by Google [24]. It provides a suite of tools and services to help developers build, deploy, and manage their applications. It supports many services like:

- 1. Database (Cloud Firestore):** This serves as the central data storage for your application. You can store and retrieve structured data in documents and collections, making it flexible for various use cases.
- 2. Hosting:** This allows you to host the frontend code (HTML, CSS, JavaScript) of your application. Firebase Hosting provides a secure and scalable platform for serving your app content to users.
- 3. Cloud Functions:** These are server-side code snippets that execute in response to events triggered by your application or other Firebase services. You can use Cloud Functions to handle tasks like user authentication logic, data validation, or sending notifications based on changes in the database.
- 4. Realtime Database:** This provides a way to store and synchronize data in real-time across clients connected to your application. This is ideal for building features that require immediate updates, like chat applications or collaborative dashboards.

2.2. LITERATURE REVIEW

The idea of earthquake prediction was not new, there have been many studies and research in this field for several years, and there are many projects similar to this project. For example, [25] designed a system for natural disaster management, where they used a network of wireless sensors to monitor disasters. They used a rain sensor and machine learning to train the system on a set of data to improve the system's performance. This system is accurate and gives early and real-time alerts, but this system focused more on the flood disaster than on the earthquake problem the cost is high as well and it is not supported easy and simple user interaction.

As for [11], they designed a network of wireless sensors to monitor earthquakes using vibration and frequency sensors, and the earthquake was detected based on physical equations. These data are displayed on a website linked to the system which adds point to simple user interaction. This system is characterized by its accuracy, but what this system lacks is a mechanism to alert the user as well as low costs.

In [3], they designed a warning system of earthquakes and used a distinctive set of low-cost elements such as the ATmega328p processor and an accelerometer through a wireless network, but this project needs advanced mechanisms to achieve high accuracy, early and real-time warnings as well as a simple user interaction system.

In [26], They designed a high-quality system that depends on a network of sensors using accelerometers as the main sensors. This network connects to the smart home via IoT to alert the user through devices. Earthquakes are detected by training a machine learning model using a neural network. This project is characterized by high accuracy, but it lacks user-interaction methods as well as high cost.

As for [2], their research was talking about employing the CNN machine learning model and the Internet of Things in early warning of earthquakes, where they used the sensors existing within the mobile devices and linked them with a machine learning model trained to detect earthquakes, but it lacks to simple user interaction as well as it is a high-cost system.

Our system is designed in a way to achieve all the key features, the system will use machine learning model and WSN to achieve high accuracy with early and real time warning system, sensors and esp32 add extra point to the advantage of low costs, mobile app and web app will add easy and simple user interaction experience.

CHAPTER 3

SYSTEM DESIGN

This chapter describes the overall design of the system and the integration of its components. section 3.1 describes high-level system description and general block diagram; section 3.2 describes design options

3.1. HIGH-LEVEL SYSTEM DESCRIPTION AND GENERAL BLOCK DIAGRAM

The system involves four key steps, as can be seen in Figure 15. First, related data is collected through a wireless network of acceleration sensors. Next, process data in a microcontroller. Then, storing data in database, the database will be connected with mobile application and send data about detecting earthquakes into mobile. A machine learning model will detect the earthquake. In the event of an earthquake, a loud alarm and the mobile app will give warnings and tell the people how to stay safe.

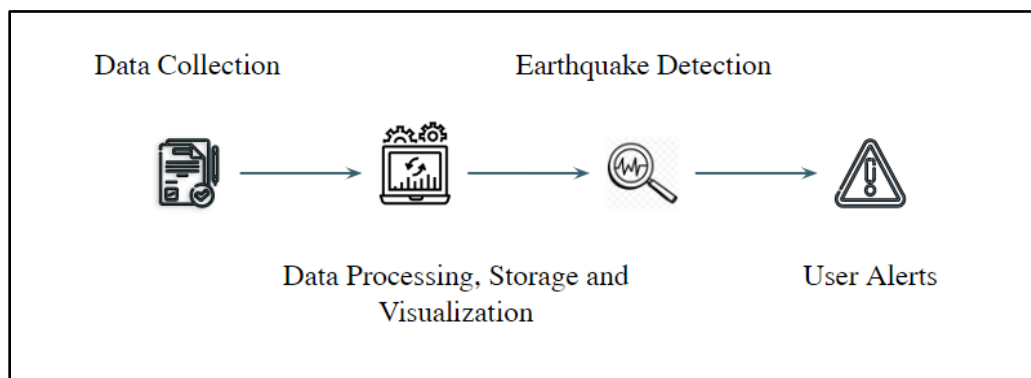


FIGURE 15 - HIGH-LEVEL SYSTEM DESCRIPTION

3.1.1. DATA COLLECTION

The system as shown in Figure 16, will collect the data using sensors like accelerometer, in this stage the system compounds a WSN, each sensor forms a node in the WSN.

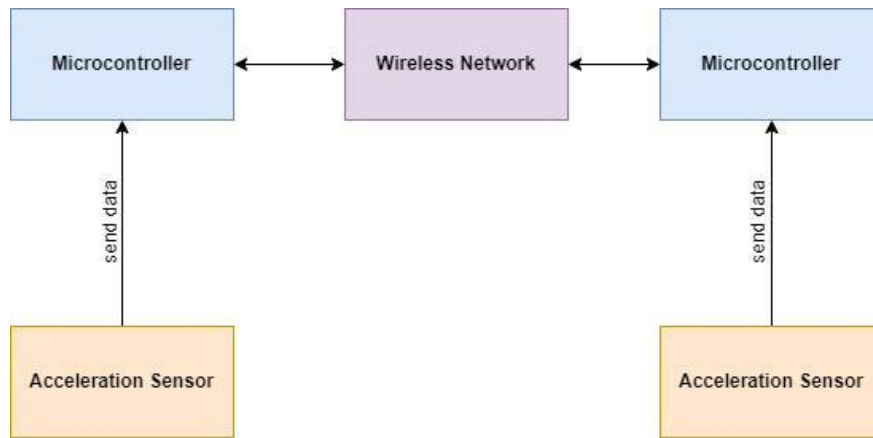


FIGURE 16 - DATA COLLECTION BLOCK DIAGRAM

3.1.2. DATA PROCESSING

In this stage one the esp'32 microcontroller will process the data and apply all arithmetic and ML model equations needed, before uploading the data to the Firebase, then the data on the Firebase will be sent to the WEB app reveal it using charts, Figure 17 shows a high-level description of this process.



FIGURE 17 - DATA PROCESSING

3.1.3. EARTHQUAKE DETECTION

Logistic regression machine learning is responsible for this task, which will be on the esp32, and will process data simultaneously after being collected from the accelerometer sensors.

3.1.4. USER ALERT

The alert consists of two major methods, hardware alert using a buzzer and software alert using a mobile application as shown in Figure 18, the alert will happen after the ML model detects the earthquake.

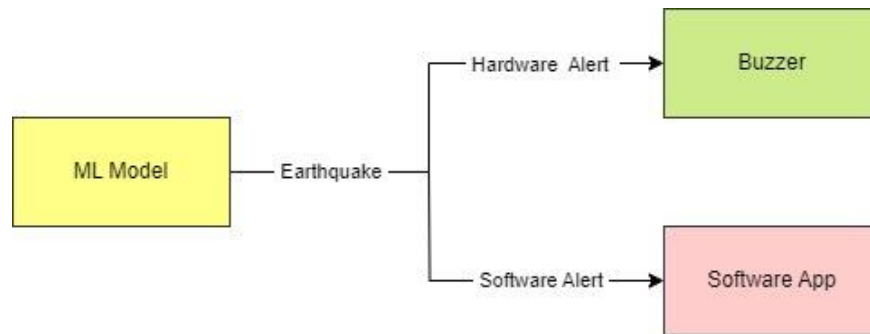


FIGURE 18 – USER ALERT

3.2. DESIGN OPTIONS

Through the comparison of available components and the rating of various hardware and software options, the goal is to identify the components that best match the project's requirements and objectives.

3.2.1. HARDWARE COMPONENTS

Our system consists of several hardware components that are essential for the earthquake detection. The main hardware components are:

ESP32 Microcontroller

We selected it in our project because it is distinguished from other microcontrollers such as Arduino and Raspberry Pi in terms of several points:

- **Connectivity:** The ESP32 microcontroller has the most connectivity options, as it supports Wi-Fi, Bluetooth, and Bluetooth LE, as well as SPI, I2C, UART, ADC, PWM, and PIO. The Arduino ATmega328 microcontroller has the least connectivity options, as it only supports USB, SPI, I2C, and UART and doesn't support Wi-Fi, or Bluetooth.
- **Power consumption:** The ESP32 microcontroller has the lowest power consumption, and supports various power modes like low-power and sleep modes. However, the Raspberry Pi microcontroller has the highest power consumption, as it requires a 5V power supply and does not have a low-power mode.

- **Real-time response:** real-time response can be based on the following criteria:
 - ✓ **Clock speed:** The ESP32 microcontroller has the highest clock speed, as it can run up to 240 MHz, compared to the Raspberry Pi microcontroller's 133 MHz, and the Arduino ATmega328's 16 MHz [27], [28], [29].
 - ✓ **Interrupt latency:** The ESP32 microcontroller has the lowest interrupt latency, as it can respond to an interrupt within 3.2 μ s, compared to the Arduino ATmega328's 4.5 μ s, and the Raspberry Pi microcontroller's 10 μ s [27], [28], [29].
 - ✓ **Operating system:** The ESP32 microcontroller and the Arduino ATmega328 microcontroller do not use an operating system, which means they have more direct control over the hardware and less interference from other processes. The Raspberry Pi microcontroller uses a real-time operating system (RTOS) called FreeRTOS, which is designed to provide and support concurrent and multitasking [28].
 - ✓ **Programming language:** The ESP32 microcontroller and the Arduino ATmega328 microcontroller use C/C++, which is a low-level, compiled, and fast language that allows direct access to the hardware and memory. The Raspberry Pi microcontroller uses MicroPython, which is a high-level, interpreted, and easy language [27], [28], [29].

Finally, we find that an ESP32 microcontroller meets our project's requirements in terms of speed, accuracy, and effectiveness in detecting earthquakes.

Earthquake Detection Sensor

Many sensors can detect earthquakes, but we need sensors that combine accuracy and reliability requirements at a reasonable and inexpensive cost. Therefore, we will compare the Seismometer [16], ADXL335[17], HMC5883[19], and MPU6050 [18] sensors that we found suitable for detecting earthquakes and they are compatible with the ESP32 microcontroller and the wireless sensor network as shown in Table 1:

TABLE 1: SPECIFICATIONS OF THE SELECTED SENSORS

features	Seismometer	ADXL335	HMC5883	MPU6050
Noise	Low	Low	Low	Low
Data output	Analog or Digital	Analog	Digital	Digital
Sensitivity	High sensitivity to detect small ground movements	Adjustable, but typically around 350 mV/g	Adjustable, typically around 2.5 mV/Gauss	Programmable
Availability	No	Yes	Yes	Yes
Max output	Variable	3.6 V	3.6 V	3.6 V
Price (\$)	144.76 ^[30]	9 ^[31]	6.69 ^[32]	8 ^[33]

Finally, we will choose ADXL335 sensor because it works well with the ESP32 and the wireless network, and it has a low noise and high resolution. It helps us to collect and process the data for the earthquake system.

Buzzer

Hardware alerting using a buzzer when an earthquake is detected.

Shaking Table

We will be using a shaking table for testing and training our systems in order to simulate earthquakes, and it also does not exist here in our country so we will build it by ourself.

3.2.2. SOFTWARE COMPONENTS

This section will describe mobile app, data visualization, data storage and machine learning model in a comparison, to reveal why we decide it the better options.

Mobile app

Our app is a smartphone software that notifies users of earthquake occurrences and gives them alerts. We have two options Flutter and MIT app Inventor platforms for building our app, In the end, we use app-inventor due to it is easier than Flutter because you do not need to deal with code just drag and drop your blocks, since the time we have to build the application is limited.

Database

We use a database for storing earthquake data so we have two options Firebase and InfluxDB, we use Firebase because InfluxDB's free version is a local database on PC and the cloud services are not free, so we use Firebase in our project to store real-time data about detecting and visualizing earthquakes data [34].

Visualization

To visualize earthquakes data to the users we have two choices, Grafana and build WEB Application. In order to create interactive charts that display sensor readings dynamically, we choose create WEB app, to easily connected with Firebase.

Machine Learning

The goal is to make this operation as accurate and fast as possible. In this section, we categorize the main developed ML algorithms as the following:

- **ANN** can achieve high accuracy and recall in earthquake detection and phase picking, as well as earthquake prediction. However, ANN may also have some drawbacks, such as high computational cost, overfitting, and sensitivity to hyperparameters and initialization.
- **K-means clustering** can be useful for exploratory analysis and anomaly detection of the seismic data, as well as for reducing the dimensionality and noise of the data. However, k-means clustering may not be suitable for earthquake detection and phase

picking, as it does not provide any labels or probabilities for the data, and it may not be able to handle the outliers and imbalanced data.

The best model depends on the specific problem and data, so to load a light code to an esp32 microcontroller, hence achieving the speed in performance we will choose logistic regression.

Wireless Sensors Network

In section 2.1.3. we discussed different topologies of WSN, and here we made a comparison to choose the most suitable one for our project:

- ✓ **Ad-hoc Topology:** it is easy to implement, but it leads to increased communication overhead, lack of a global network view, and higher energy consumption due to frequent interactions.
- ✓ **Star Topology:** This setup simplifies network management and minimizes power consumption in individual nodes, but the base station becomes a single point of failure
- ✓ **Mesh Topology:** This enhances reliability and scalability but increases complexity and energy consumption.
- ✓ **Tree (Hierarchical) Topology:** efficiently managing large-scale networks but potentially creating bottlenecks
- ✓ **Cluster Topology:** This reduces communication overhead but may overload cluster heads and create uneven energy consumption
- ✓ **Hybrid Topology:** It offers flexibility and scalability but increases design and management complexity

In the end, we chose the star topology type since we have just two esp32 microprocessors and two accelerometers, which we will connect via the Firebase database.

CHAPTER

4

IMPLEMENTATION

This chapter describes the overall implementation of the system and the integration of its components. Firstly Section 4.1 describes Hardware Implementation, Section 4.2 describes machine learning implementation, Section 4.3 describes wireless sensor network, and finally Section 4.4 mobile app implementation.

4.1 HARDWARE IMPLEMENTATION

4.1.1. SHAKING TABLE

Was built to simulate the waves caused by earthquakes. Four medium-tension springs and two woods plates make up the configuration. The springs are strongly fixed between the plates at the corners of the table's 5 cm length as shown in Figure 19. Earthquake-like vibrations can be produced by moving the table.



FIGURE 19 - SHAKING TABLE

4.1.2. HARDWARE SYSTEM CONNECTIONS

Connecting ESP32 microcontroller with ADXL335 and buzzer as show in Figure 20, to run the ESP32, follow the steps in [35], The first pin of buzzer which is the negative pin is connected to ESP32 GND, and second pin of buzzer is connected to pin number 32 in ESP32, and for ADXL335 connections, this sensor has 5 pins, VCC and GND will be connected to ESP32 VCC and GND pins, others three pins which are x, y and z, will be connected respectively with pins 33, 34 and 35 in ESP32. The second node has the same connection.

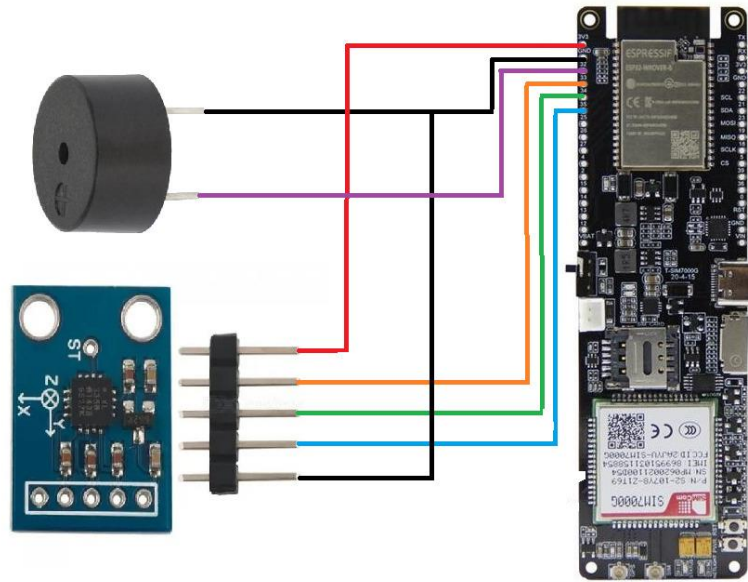


FIGURE 20 - HARDWARE CONNECTIONS SYSTEM

4.1.3 EVALUATION SETUP

In this section, we discuss the experimental setup, datasets, performance measures, experiments, and evaluation of hardware and software systems. A shake table generates accurate earthquake motions for experiments. We used a shake table which we created to generate earthquake motions for testing. The purpose of these tests is: to assess the accuracy of the sensors, test the low-cost sensor's performance, and collect data for testing the machine learning models. The shake table generates earthquake motion in three dimensions. We perform shake-table experiments by

simulating two virtual earthquakes, we depend on [26] to choose the recommended acceleration sensors for collecting data process.

4.2 SOFTWARE IMPLEMENTATION

4.2.1 LOGISTIC REGRESSION EARTHQUAKE DETECTION MODEL

In this project, we aim to develop a logistic regression model to detect earthquakes based on accelerometer sensors data. The goal is to classify data samples into two categories: "Earthquake" or "Not-Earthquake."

COLLECT THE DATA

The dataset used for this earthquake prediction model consists of acceleration readings captured along three axes (x, y, and z). For each data point, 100 samples of accelerations are collected along these axes. The standard deviation for each axis (x, y, and z) is then calculated. These standard deviation values are used as features in the model. Each data point is labeled as either indicating an earthquake or not.

DATA PREPARATION

To prepare the data for training, 43 acceleration samples are collected per second. The standard deviation is calculated for each axis using 100 samples, which represents a time interval of 0.023 seconds. This means that the time required to collect a single sample before collecting the next sample is less than 0.00023 seconds.

A total of 6,000 seismic and non-seismic samples were collected. To collect non-seismic samples, the sensors and table were fixed to simulate a normal situation and data was collected. Then, the earthquake scenario was simulated to collect the seismic data.

DATASET PARTITIONING

The dataset is partitioned into training and testing sets to evaluate the model's performance. Typically, the data is split such that 80% is used for training and 20% for testing. This ensures that the model can generalize well to unseen data.

CODING

1. Training Phase

```
// Define a structure for holding a single data point (x, y, z accelerations and label)

struct DataPoint {

    double x, y, z;

    int label; // 1 for earthquake, 0 for non-earthquake

};

// Define a structure for holding the logistic regression parameters (weights)

struct Parameters {

    double w0; // Bias term

    double w1; // Coefficient for x

    double w2; // Coefficient for y

    double w3; // Coefficient for z

};

// Sigmoid function

double sigmoid(double z) {

    return 1.0 / (1.0 + exp(-z));

}

// Hypothesis function

double hypothesis(const Parameters& theta, double x, double y, double z) {

    return sigmoid(theta.w0 + (theta.w1 * x) + (theta.w2 * y) + (theta.w3 * z));

}

// Function to predict labels

int predictLabel(const Parameters& theta, double x, double y, double z, double threshold = 0.5) {

    double prediction = hypothesis(theta, x, y, z);
```

```
return (prediction >= threshold) ? 1 : 0;
}
```

2. Testing Phase

```
// Define a structure for holding the logistic regression parameters (weights)
struct Parameters { //,,
    double w0 = -2.25237; // Bias term
    double w1 = 0.334201; // Coefficient for x
    double w2 = 0.0475398; // Coefficient for y
    double w3 = 0.319727; // Coefficient for z
};
```

The full code on the link: <https://github.com/MariamHasanat/GP-Arduino-Code.git>

4.2.2 WSN IMPLEMENTATION

In our project, we implement WSN star topology type as shown in Figure 21. First, we connect esp32 microcontroller with firebase and then connecting mobile app with firebase to take alert from it.

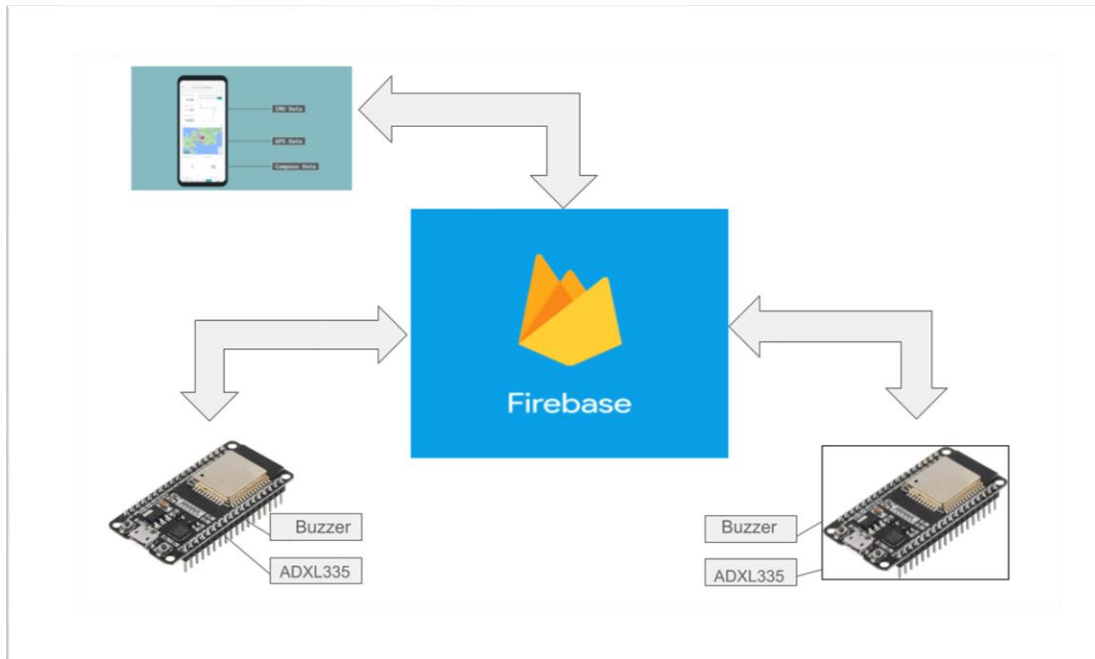


FIGURE 21 - WSN IMPLEMENTATION

4.2.3 FIREBASE DATABASE

Create new project and chose real-time database and take URL link and API key for this project and save it as shown in Figure 22.

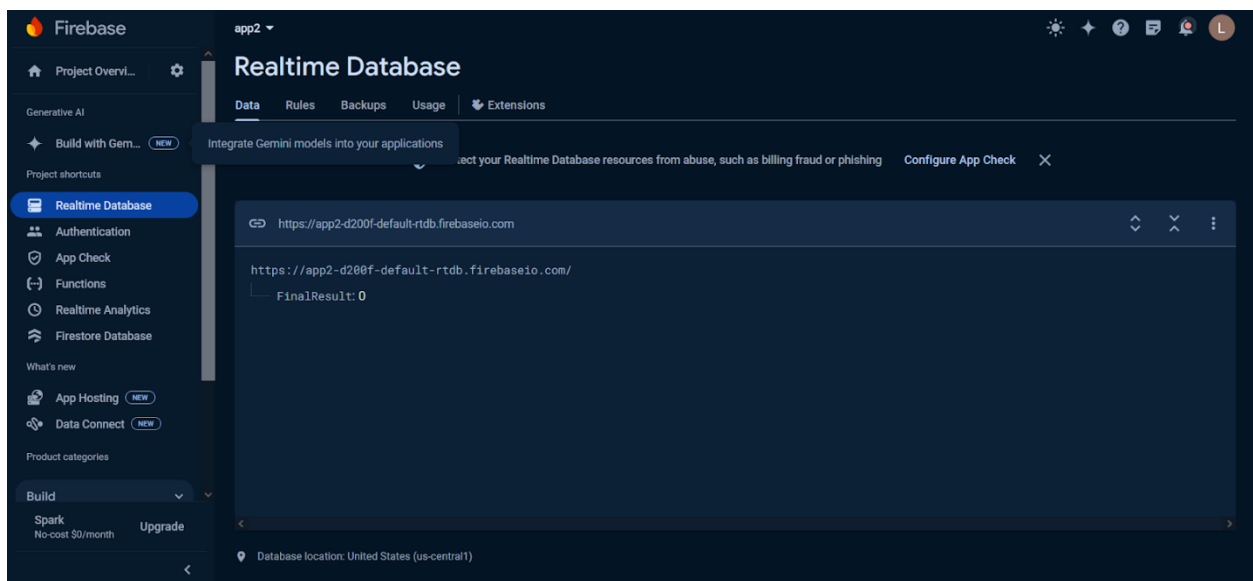


FIGURE 22 - FIREBASE REAL-TIME DATABASE

CONNECTING ESP32 WITH FIREBASE

To connect an ESP32 to Firebase, we use Firebase ESP32 library in the Arduino IDE, and install it then use the code to allow to send and receive data in real-time.

CODE

```
#include <Arduino.h>

#include <WiFi.h>

#include <FirebaseESP32.h>

#include <addons/TokenHelper.h>

#include <addons/RTDBHelper.h>

#define WIFI_SSID "WIFI_SSID"

#define WIFI_PASSWORD "WIFI_PASSWORD"

#define API_KEY "API Key"

#define DATABASE_URL "RTDB URL"

#define USER_EMAIL " Your email on firebase "

#define USER_PASSWORD "1password"

FirebaseData fbdo;

FirebaseAuth auth;

FirebaseConfig config;

unsigned long sendDataPrevMillis = 0;

unsigned long count = 0;

void setup()

{

  Serial.begin(9600);

  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);

  Serial.print("Connecting to Wi-Fi");
```

```

while (WiFi.status() != WL_CONNECTED)

{
Serial.print(".");

delay(300);

}

Serial.println();

Serial.print("Connected with IP: ");

Serial.println(WiFi.localIP());

Serial.println();

Serial.printf("Firebase Client v%s\n\n", FIREBASE_CLIENT_VERSION);

config.api_key = API_KEY;

auth.user.email = USER_EMAIL;

auth.user.password = USER_PASSWORD;

config.database_url = DATABASE_URL;

config.token_status_callback = tokenStatusCallback;

Firebase.reconnectNetwork(true);

fbdo.setBSSLBufferSize(4096 , 1024 );

Firebase.begin(&config, &auth);

Firebase.setDoubleDigits(5);

}

void loop()

{

if (Firebase.ready() && (millis() - sendDataPrevMillis > 15000 || sendDataPrevMillis == 0))

{

```

```
sendDataPrevMillis = millis();
```

```
Serial.printf("Set X ... %s\n", Firebase.setInt(fbdo, F("/x"), x_data) ? "ok" : fbdo.errorReason().c_str()); // writ  
on database
```

```
Serial.printf("Get X ... %s\n", Firebase.getInt(fbdo, F("/x")) ? String(fbdo.to<int>()).c_str() :  
fbdo.errorReason().c_str()); // read from database
```

```
}
```

CONNECTING ESP32 WITH MIT APP INVENTOR USING FIREBASE

First of all, create a new project in MIT App Inventor and design the user interface as shown in Figure 23. Second, Add the FirebaseDB component from experimental list, and set the Firebase URL and token in the properties. Finally, Use the Blocks for reading and writing data to Firebase as shown in Figure 24.

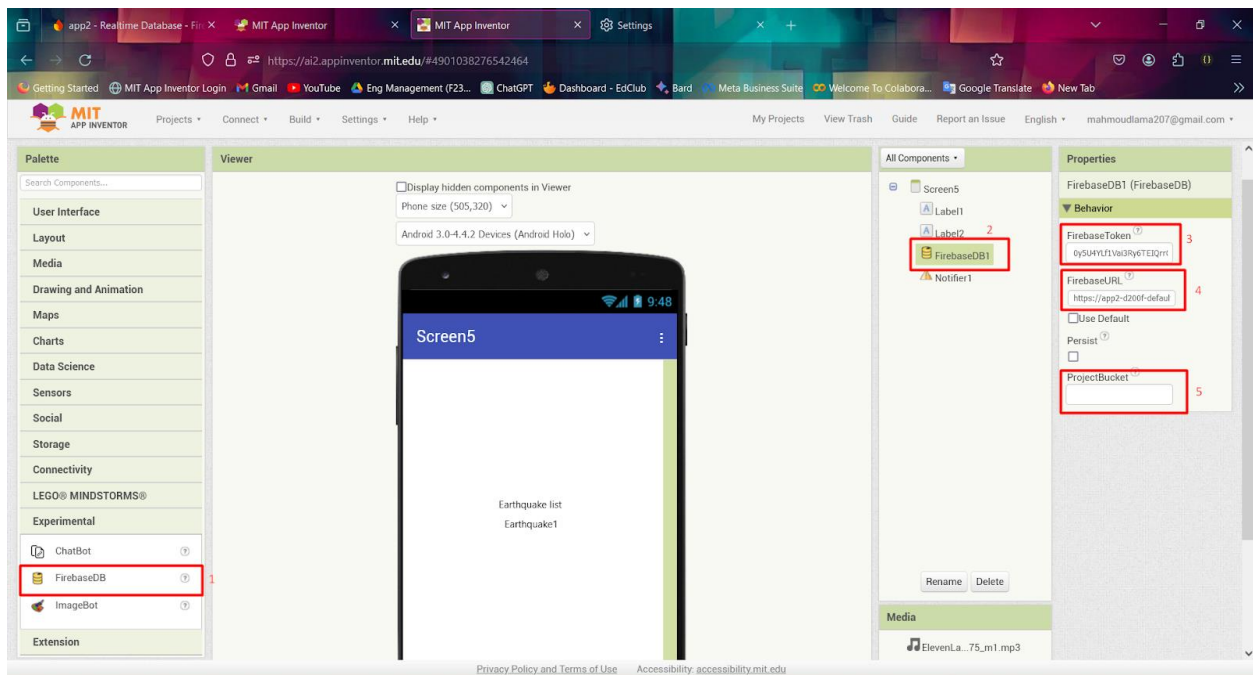


FIGURE 23 - MIT APP INVENTOR VIEWER SCREEN

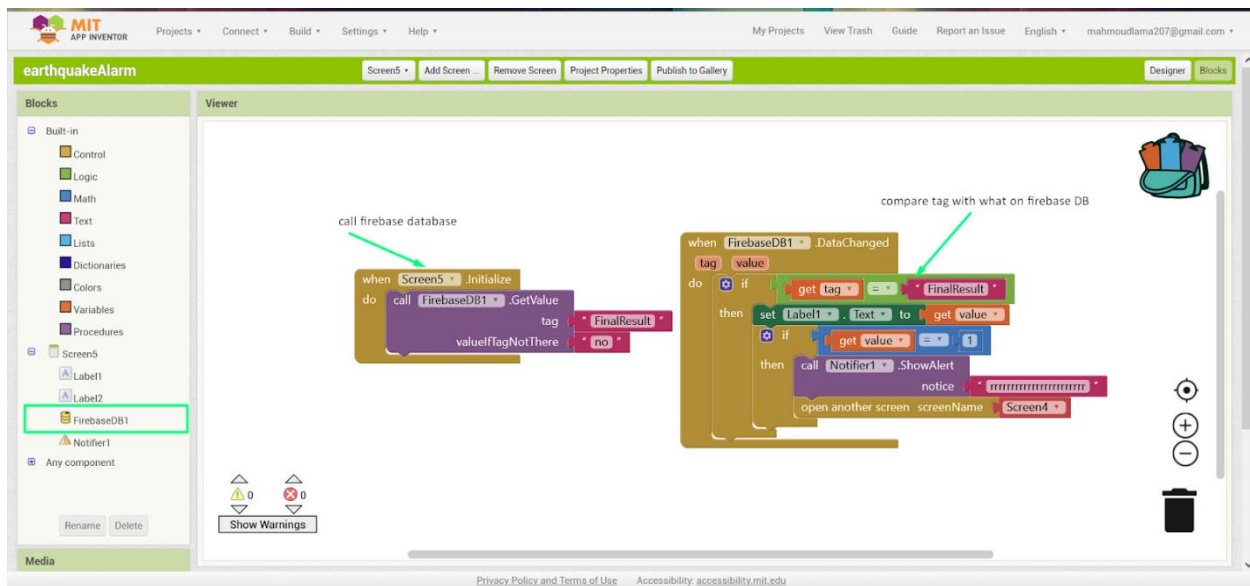


FIGURE 24 - MIT APP INVENTOR CODE BLOCKS SCREEN

4.2.4 MOBILE APP IMPLEMENTATION

Have four screens and each screen have different design and own blocks.

Introduction Screen: The introduction screen is where users interact with the system for the initial time as shown in Figure 25.



FIGURE 25 - INTRODUCTION SCREEN

Map Screen: The map screen displays the user's location.

Safety Instructions Screen: This screen displays earthquake safety measures.

Triggering Voice Notifications: When an earthquake is detected, a mobile app can trigger an aloud alert.

4.2.5 WEB APPLICATION IMPLEMENTATION

In this section, we will develop a web application to visualize the accelerometer data using the Chart.js library. The application will consist of three files: HTML, CSS, and JavaScript. These files will be deployed and connected to an Firestore database which will be connected to ESP32 microcontroller to collect the data, store it and send it to the WEB application.

WEB IMPLEMENTATION

HTML

The HTML file sets up the structure of the web page and includes the Chart.js library.

```
<!-- Complete project details: https://randomnerdtutorials.com/esp32-plot-
readings-charts-multiple/ -->

<!DOCTYPE html>
<html>
  <head>
    <title>ESP IOT DASHBOARD</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="icon" type="image/png" href="favicon.png">
    <link rel="stylesheet" type="text/css" href="style.css">
    <script src="https://code.highcharts.com/highcharts.js"></script>
  </head>
  <body>
    <div class="topnav">
      <h1>ESP WEB SERVER CHARTS</h1>
    </div>
    <div class="content">
      <div class="card-grid">
        <div class="card">
          <p class="card-title">Acceleration Chart - Sensor 1</p>
          <div id="chart-acceleration" class="chart-container"></div>
        </div>
      </div>
    </div>
  </body>
</html>
```

```

</div>
<div class="content">
  <div class="card-grid">
    <div class="card">
      <p class="card-title">Acceleration Chart - Sensor 2</p>
      <div id="chart-acceleration1" class="chart-container"></div>
    </div>
  </div>
</div>
<script src="script.js"></script>
</body>
</html>

```

CSS

The CSS file styles the web page.

```

/* Complete project details: https://randomnerdtutorials.com/esp32-plot-readings-charts-multiple/ */

html {
  font-family: Arial, Helvetica, sans-serif;
  display: inline-block;
  text-align: center;
}

.card {
  background-color: white;
  box-shadow: 2px 2px 12px 1px rgba(140,140,140,.5);
}

.card-title {
  font-size: 1.2rem;
  font-weight: bold;
  color: #034078
}

.chart-container {
  padding-right: 5%;
  padding-left: 5%;
}
// .....

```

JavaScript

The JavaScript file fetches the data from the Firestore database, processes it, and visualizes it using Chart.js.

```
// Complete project details: https://randomnerdtutorials.com/esp32-plot-readings-charts-multiple/

// Get current sensor readings when the page loads
window.addEventListener('load', getReadings);

// Create Acceleration Chart
var chartT = new Highcharts.Chart({ .....});

var chartA = new Highcharts.Chart({.....});

//Plot acceleration in the acceleration chart
function plotAcceleration(jsonValue) {

    var keys = Object.keys(jsonValue);
    console.log(keys);
    console.log(keys.length);

    for (var i = 0; i < keys.length; i++){
        var x = (new Date()).getTime();
        console.log(x);
        const key = keys[i];
        var y = Number(jsonValue[key]);
        console.log(y);

        if(chartA.series[i].data.length > 40) {
            chartA.series[i].addPoint([x, y], true, true, true);
        } else {
            chartA.series[i].addPoint([x, y], true, false, true);
        }
    }
}

// Function to get current readings on the webpage when it loads for the first time
function getReadings(){
    var xhr = new XMLHttpRequest();
    xhr.onreadystatechange = function() {
```

```
if (this.readyState == 4 && this.status == 200) {  
    var myObj = JSON.parse(this.responseText);  
    console.log(myObj);  
    plotAcceleration(myObj);  
}  
};  
xhr.open("GET", "/readings", true);  
xhr.send();  
}
```

Show the Full Code on the following link:

https://github.com/MariamHasanat/Data_Visualization.git

WEB DEPLOYMENT

Deploy web application using Netlify, to get an URL, which will be used in mobile application to open and show the web page on click the button, to display the charts.

The following link is the deployed WEB app: <https://earthquake-gp.netlify.app/>

CONNECT WEB APP WITH FIRESTORE

As mentioned in the document [36], to connect the web app with firestore we should apply the following steps:

- Create a Cloud Firestore database.
- Set up your development environment.
- Prototype and test with Firebase Local Emulator Suite.
- Initialize Cloud Firestore.
- Add data.
- Read data.
- Secure your data.

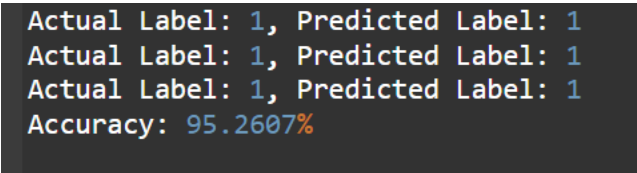
CHAPTER

5

EVALUATION

5.1 RESULTS AND DISCUSSION

Model testing is a critical step in measuring the accuracy of the forecasting process. However, it is very important to test the model on data that is completely different from the data it was trained on. Therefore, after collecting the necessary 6,000 seismic and non-seismic samples, we split the data into 80% for training the system and 20% for testing the system. The testing process resulted in an accuracy rate of 95.3% as shown in Figure 26, which is a very high percentage. When the system was installed and tested by simulating an earthquake using a checkerboard, the system sounded the alarm within 0.5 seconds and the result was displayed directly on the buzzer.



```
Actual Label: 1, Predicted Label: 1
Actual Label: 1, Predicted Label: 1
Actual Label: 1, Predicted Label: 1
Accuracy: 95.2607%
```

FIGURE 26 - MACHINE LEARNING ACCURACY

However, the time for the alarm to reach the phone was affected by internet speed, as the system took 4-7 seconds to send data from the console to Firebase, and it took about 3-5 seconds for the alarm to arrive. Access Firebase to phone. Therefore, the total time required for an alarm is 12.5 seconds in the worst case and 7.5 seconds in the best case.

How will the system achieve 100% confidence in confirming the presence of an earthquake, and the validity of the warning? This will be done by verifying the data coming from the WSN, as its presence serves the goal of warning accuracy, as the confidence percentage will be distributed among the nodes connected to the network. As a further explanation: If there are two nodes, the confidence percentage will be 100% distributed among these two so that the confidence percentage for each one becomes 50%, so if both of them indicate that there is an earthquake, the percentage of the earthquake is 100% and therefore the warning is certain. If there are 10 nodes, the percentage of each one will be 10%. If an alert comes from all the nodes that there is an earthquake, then the percentage of confidence is much greater because Confirmation of the existence of the earthquake involved 10 different sites.

As for the best place to install the system, due to the presence of supports to stabilize the buildings, the p-wave warning wave will not affect or appear in the building, and therefore rocks are considered the best place to install the system.

When using the Global Positioning System (GPS) integrated with the ESP32 controller to determine its location and show it on the map that appears to the user on the mobile phone application, it requires a lot of time to determine the location, and it must also be placed in open places such as a courtyard, and this affects the work of machine learning. In detecting the presence of earthquakes, thus reducing the accuracy and speed of the system, and for this reason, we relied on the GPS on the phone to determine the location on the map on the application, and this saves time for the benefit of the machine learning system in detecting the earthquake.

CHAPTER 6

SUMMARY

AND

FUTURE WORKS

6.1 SUMMARY

This project focused on achieving features of high accuracy, low cost, early and real-time alert as well as simple and easy user interaction in case of earthquake disaster by developing an alarm system using Wireless Sensor Networks (WSN), ESP32 microcontrollers, and machine learning techniques and alert the user with a mobile application as well as the sound system using the buzzer. The system leverages the ESP32's capabilities to collect seismic data, perform data aggregation, and execute machine learning algorithms directly on the microcontroller. This machine learning processing enhances the system's ability to accurately detect seismic activities and distinguish them from other environmental noises. The use of a Firebase database enabled seamless data synchronization and communication between multiple ESP32 devices and a mobile application.

6.2 FUTURE WORK

In future iterations of our project, we aim to expand its capabilities and enhance its impact by integrating several advanced features.

Power Backup: Power backup is a vital component of our earthquake detection and alarm system, as it ensures that the system remains functional in the event of a power outage. A power backup can consist of a battery, an uninterruptible power supply (UPS), or a generator that can provide enough power to run the system for a certain period. A power backup can also help protect the system from power surges or fluctuations that can damage the hardware or corrupt the data. Having a power backup can increase the reliability and resilience of our system, and prevent the loss of valuable information or alerts during an emergency.

Integrated Systems for Community-wide Enhancement: make our system for an area not just for one home. This means that you can deploy multiple sensor nodes in different locations, such as buildings, bridges, and roads, to cover a larger and denser area of the seismic activity. This can improve the accuracy and reliability of our system, and provide more benefits and convenience to more end-users. To do this, we need to use the wireless sensor network as the communication method between the sensor nodes and the central server, using the Wi-Fi and LoRa protocols. We

also need to design and implement the system architecture and the system requirements for the area-based system.

Smart Home: Integrating our system with smart home features, such as smart lights, smart locks, smart speakers, and smart cameras. This can make our system more functional and user-friendly. For example, you can use smart lights to flash or change color, smart locks to unlock the doors, smart speakers to broadcast the messages, and smart cameras to monitor and record the conditions. To do this, we need to use IoT protocols and platforms, such as MQTT, ZigBee, and Blynk, and design and implement the user interface and the user experience.

Monitoring System: Where we will benefit from the testimonies of other people where we will supply the system and do a look at the nearby areas.

System Security: System security is an important aspect of any project that involves data transmission and communication over a network. For our project, we may want to protect the data collected by the sensors from unauthorized access, modification, or deletion, ensuring the integrity and authenticity of the data sent from the sensors to the control unit and vice versa. encrypt the data and use secure protocols for communication between the devices and the server, preventing or mitigating the impact of cyberattacks, such as denial-of-service, spoofing, or tampering, on the system performance and functionality, design and implement a backup and recovery plan in case of system failure or damage.

Earthquake Simulation Software Program: The critical need for earthquake simulation has driven the development of innovative software programs that can accurately model seismic events. These programs offer a versatile platform for testing various applications, such as: building and infrastructure resilience, emergency response planning, public education, and awareness. Crucially, these programs bridge accessibility gaps for regions facing restrictions on physical earthquake simulation equipment.

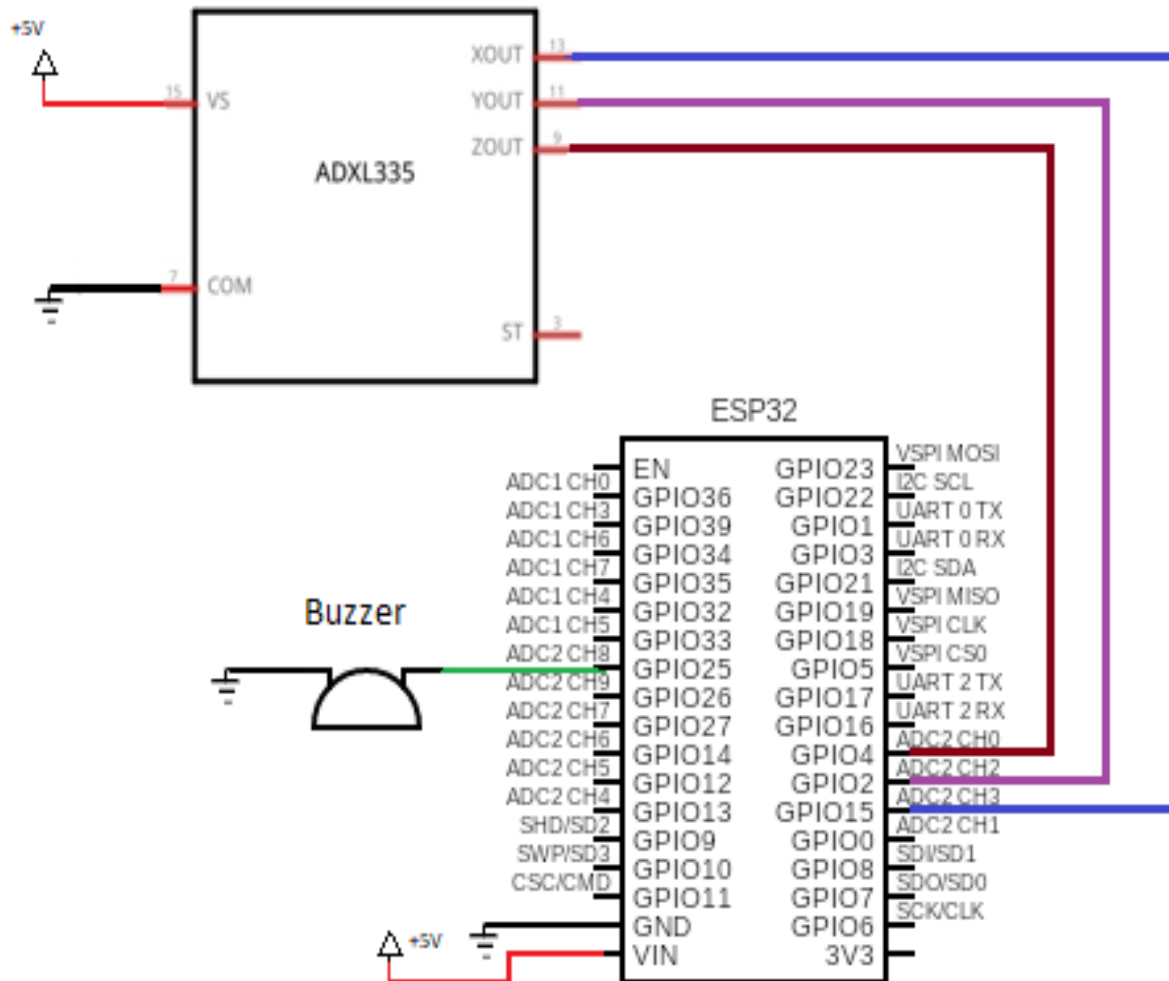
REFERENCES

- [1] "Earthquakes," WORLD HEALTH ORGANIZATION (WHO). Accessed: Jan. 19, 2024. [Online]. Available: https://www.who.int/HEALTH-TOPICS/EARTHQUAKES#TAB=TAB_1
- [2] M. S. Abdalzaher *et al.*, "On the urgent detection and alarm system (UrEDAS)," *Bull. Seismol. Soc. Am.*, vol. 13, no. 1, pp. 1–12, 2023, doi: 10.1785/0120030133.
- [3] S. Kalita *et al.*, "Wireless Earthquake Alarm System using ATmega328p, ADXL335 and XBee S2," *Int. J. Eng. Trends Technol.*, vol. 12, no. 3, pp. 105–129, 2023, doi: 10.14445/22315381/ijett-v12p227.
- [4] S. M. Mousavi, Y. Sheng, W. Zhu, and G. C. Beroza, "Stanford EArthquake Dataset (STEAD): A Global Data Set of Seismic Signals for AI," *IEEE Access*, vol. 7, pp. 179464–179476, 2019, doi: 10.1109/ACCESS.2019.2947848.
- [5] P. HAEUSSLER, "COOL EARTHQUAKE FACTS | U.S. GEOLOGICAL SURVEY," USGS.GOV. [Online]. Available: <https://www.usgs.gov/PROGRAMS/EARTHQUAKE-HAZARDS/COOL-EARTHQUAKE-FACTS>
- [6] M. NESIRKY and J. V. NKOLO, "2010 HAITI EARTHQUAKE," WIKIPEDIA.
- [7] "NoOn This Day: 2011 Tohoku Earthquake and Tsunami," NATIONAL CENTERS FOR ENVIRONMENTAL INFORMATION. [Online]. Available: <https://www.ncei.noaa.gov/news/day-2011-japan-earthquake-and-tsunami>
- [8] E. N. Cinar, A. Abbara, and E. Yilmaz, "Earthquakes in Turkey and Syria-collaboration is needed to mitigate longer terms risks to health," *BMJ*, vol. 380, no. March, p. p559, 2023, doi: 10.1136/bmj.p559.
- [9] Y. Sherki, N. Gaikwad, J. Chandle, and A. Kulkarni, "Design of real time sensor system for detection and processing of seismic waves for earthquake early warning system," *Proc. 2015 IEEE Int. Conf. Power Adv. Control Eng. ICPACE 2015*, pp. 285–289, 2015, doi: 10.1109/ICPACE.2015.7274959.
- [10] P. B. Quang, P. Gaillard, and Y. Cano, "Association of array processing and statistical modelling for seismic event monitoring," *2015 23rd Eur. Signal Process. Conf. EUSIPCO 2015*, pp. 1945–1949, 2015, doi: 10.1109/EUSIPCO.2015.7362723.
- [11] I. Sitanggang, J. A. I. Damanik, F. Hutabarat, and A. Sagala, "Implementation of Wireless Sensor Network (WSN) for Earthquake Detection," *Elkha*, vol. 14, no. 2, p. 102, 2022, doi: 10.26418/elkha.v14i2.56146.
- [12] S. JAIN, "WIRELESS SENSOR NETWORK (WSN)," GEEKSFORGEEKS.
- [13] A. Shrestha and L. Xing, "A performance comparison of different topologies for wireless sensor networks," *2007 IEEE Conf. Technol. Homel. Secur. Enhancing Crit. Infrastruct. Dependability*, pp. 280–285, 2007, doi: 10.1109/THS.2007.370059.
- [14] K. Wheeling, "Comparing Machine Learning Models for Earthquake Detection." [Online]. Available: <https://eos.org/RESEARCH-SPOTLIGHTS/COMPARING-MACHINE-LEARNING-MODELS-FOR-EARTHQUAKE-DETECTION>
- [15] "ESP32 DATASHEET AVAILABLE ONLINE." [Online]. Available: https://www.espressif.com/SITES/DEFAULT/FILES/DOCUMENTATION/ESP32-WROOM-32_DATASHEET_EN.PDF
- [16] "SEISMOMETER DATASHEET AVAILABLE ONLINE." [Online]. Available: <http://www.geoinstr.com/DS-KS1.PDF>

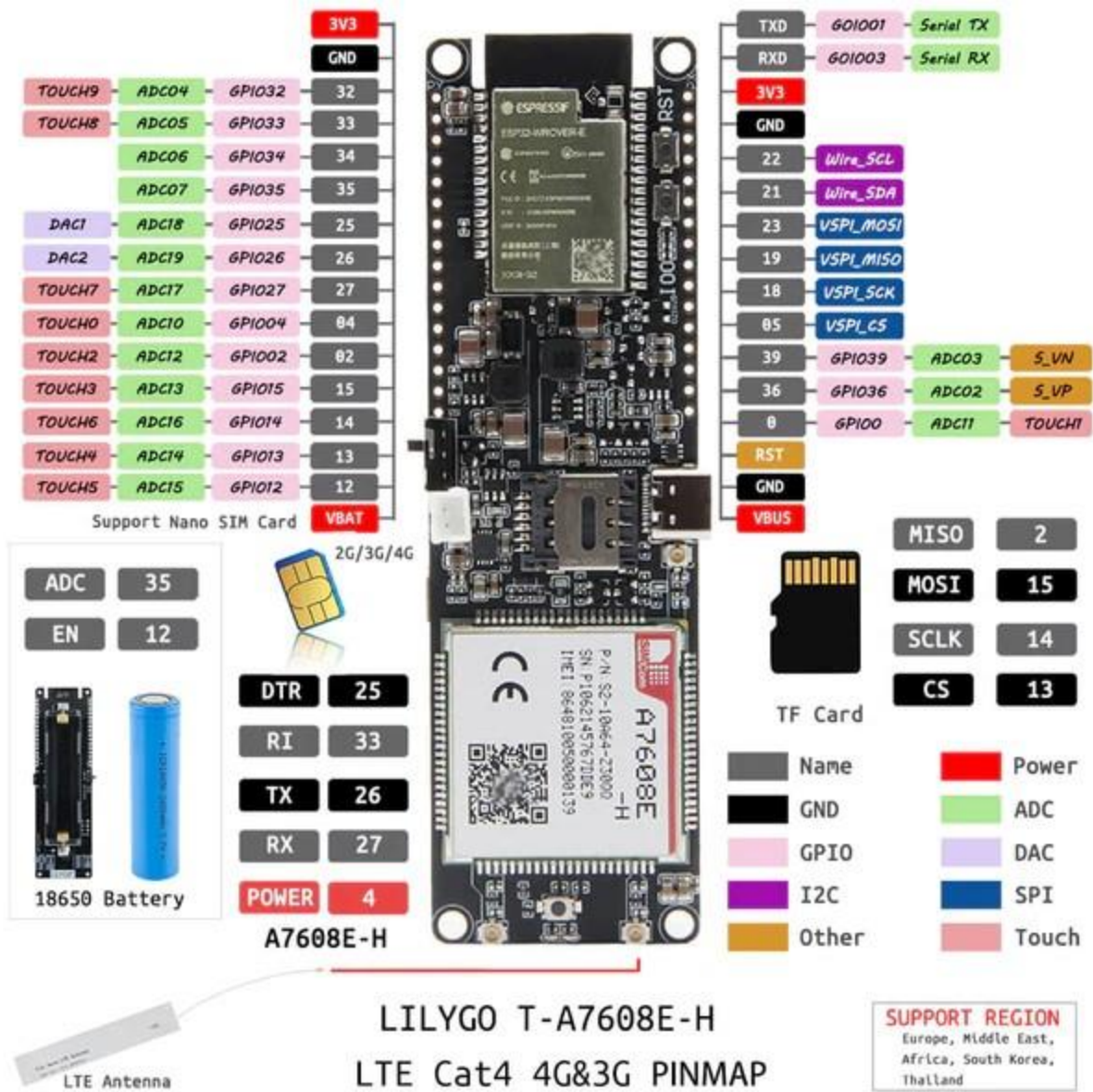
- [17] “ADXL335 DATASHEET AVAILABLE ONLINE.” [Online]. Available: <https://www.analog.com/media/en/technical-documentation/data-sheets/ADXL335.pdf>
- [18] “MPU-6000 and MPU-6050 Product Specification.” [Online]. Available: <https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf>
- [19] “HMC5883L DATASHEET AVAILABLE ONLINE.” [Online]. Available: <https://www.farnell.com/DATASHEETS/1683374.PDF>
- [20] “BUZZER,” WIKIPEDIA. Accessed: Jan. 21, 2024. [Online]. Available: <https://en.wikipedia.org/WIKI/BUZZER>
- [21] “MIT App Inventor.” [Online]. Available: https://en.wikipedia.org/wiki/MIT_App_Inventor
- [22] “Flutter.” [Online]. Available: <https://flutter.dev/>
- [23] “Grafana.” [Online]. Available: <https://en.wikipedia.org/wiki/Grafana>
- [24] “Firebase Services.” [Online]. Available: <https://firebase.google.com/>
- [25] R. Joshi, A. Shinde, S. Kelkar, and M. Deore, “Towards Efficient Disaster Management: Role of Machine Learning, Deep Learning and WSN Technologies,” *Int. J. Intell. Syst. Appl. Eng.*, vol. 12, no. 6s, pp. 98–111, 2024.
- [26] J. Lee, I. Khan, S. Choi, and Y. W. Kwon, “A smart iot device for detecting and responding to earthquakes,” *Electron.*, vol. 8, no. 12, pp. 1–19, 2019, doi: 10.3390/electronics8121546.
- [27] “ATMEGA328/P DATASHEET SUMMARY,” OCTOPART. [Online]. Available: <https://datasheet.octopart.com/ATMEGA328P-MU-MICROCHIP-DATASHEET-65729177.PDF>
- [28] The Raspberry Pi Foundation, “Raspberry Pi 4 Model B – Raspberry Pi,” The Raspberry Pi Foundation. [Online]. Available: <https://www.raspberrypi.com/products/raspberry-pi-4-model-b/%0Ahttps://www.raspberrypi.org>
- [29] “ESP32 SERIES DATASHEET. (N.D.). ESPRESSIF SYSTEMS.” [Online]. Available: https://www.espressif.com/SITES/DEFAULT/FILES/DOCUMENTATION/ESP32_DATASHEET_EN.PDF
- [30] “(N.D.). AMAZON.” Accessed: Jan. 20, 2024. [Online]. Available: <https://www.amazon.com/ANNCUS-SM-24-GEOPHONE-ELEMENT-SENSOR/DP/B095N9K3D2>
- [31] “SparkFun Triple Axis Accelerometer Breakout - ADXL335 - SEN-09269,” SparkFun Electronics. Accessed: Jan. 21, 2024. [Online]. Available: <https://www.sparkfun.com/products/9269>
- [32] “HILETGO GY-271 QMC5883L 3-5V IIC TRIPLE AXIS COMPASS MAGNETIC SENSOR MODULE ELECTRONIC COMPASS MODULE COMPATIBLE HMC5883L.” [Online]. Available: <https://www.amazon.com/HiLetgo-GY-271-QMC5883L-Compass-Magnetometer/dp/B008V9S64E>
- [33] “Gyroscope sensor (e.g., MPU6050) datasheet available online.” [Online]. Available: <https://invensense.tdk.com/wp-content/uploads/Small, Low Power, 3-Axis ±3 g Accelerometer ADXL3352015/02/MPU-6000-Datasheet1.pdf>
- [34] “Firebase.” [Online]. Available: <https://firebase.google.com/>
- [35] “Getting Started with LILYGO T-SIM7000G ESP32 (LTE, GPRS, and GPS),” Random Nerd Tutorials. [Online]. Available: <https://randomnerdtutorials.com/lilygo-t-sim7000g-esp32-lte-gprs-gps/>
- [36] “Get started with Cloud Firestore.” [Online]. Available:

<https://firebase.google.com/docs/firestore/quickstart?hl=en>

APPENDIX A- SCHEMATIC DIAGRAM



APPENDIX B- ESP32 DIAGRAM



APPENDIX C – ANN RESULTS

We try to use ANN wizard in MATLAB, and the division of data was 70% training, 15% testing and 15% validation, and it gives the following results, as shown in Figures 27, 28, 29 we show the confusion matrix, ROC curves, and performance plot respectively.

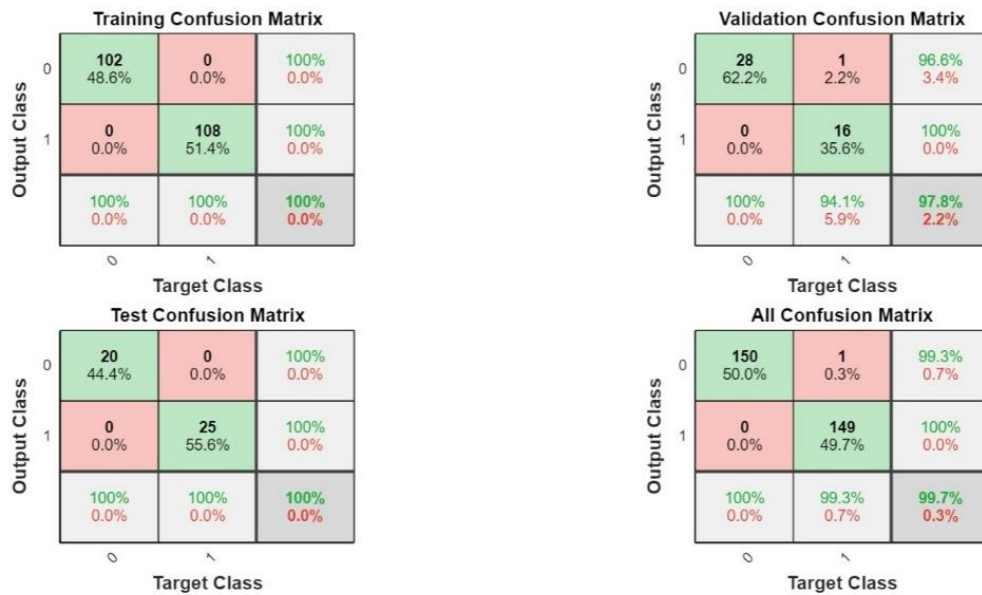


FIGURE 27 - CONFUSION MATRIX

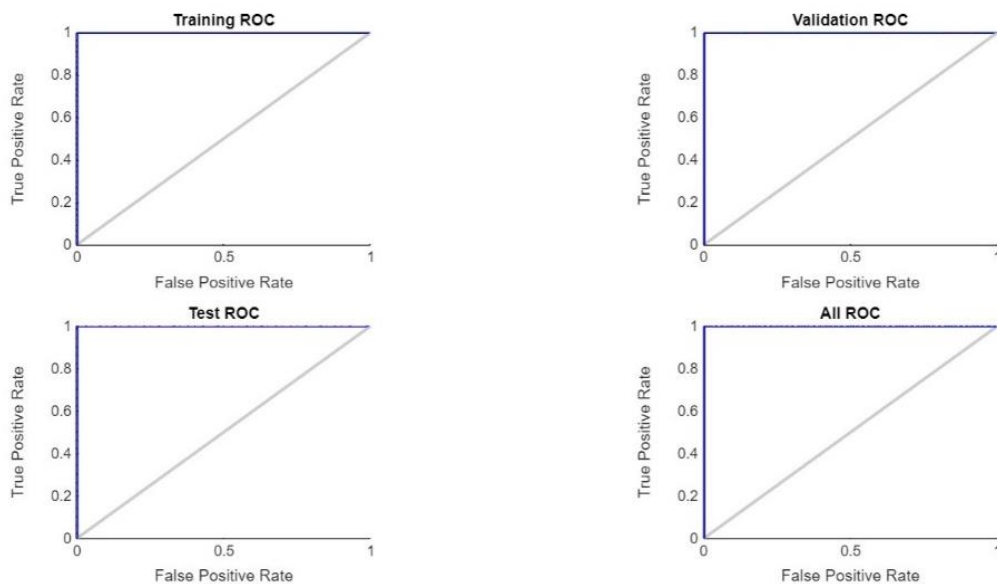


FIGURE 28 - ROC CURVES

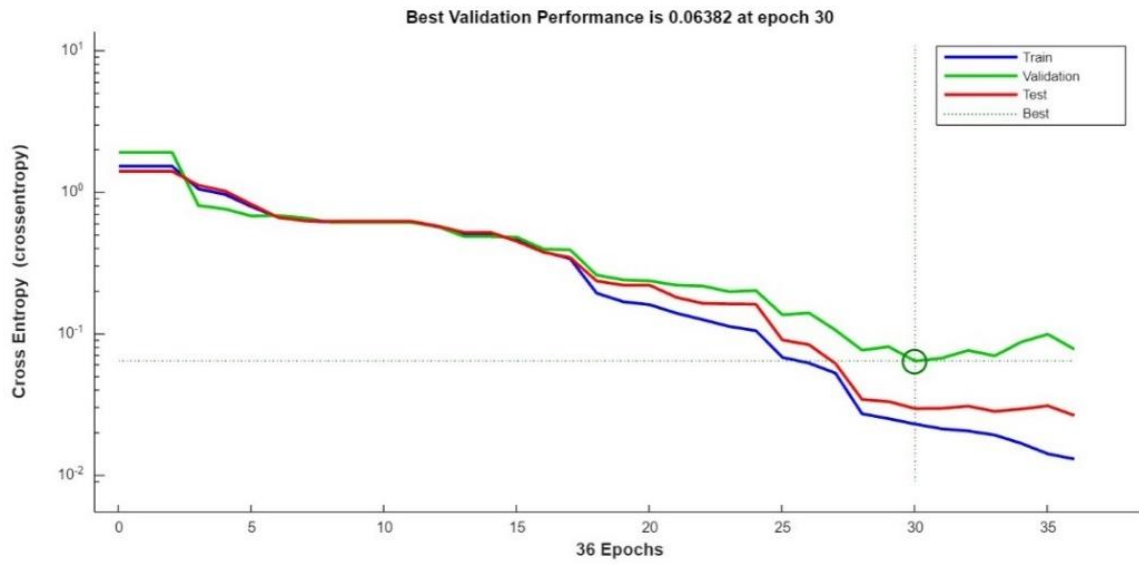


FIGURE 29 - PERFORMANCE PLOT