Securing Hackable IoT Enabled LTE Devices

(S.H.I.E.L.D)

SYNDICATE MEMBERS

Muhammad Abdullah Farooq

Umar Ahmad Khan

Saif Ur Rehman

Abdul Samad

SUPERVISOR

Dr. Mir Yasir Umair

Submitted to the faculty of Department of Electrical Engineering,
Military College of Signals, National University of Sciences and Technology,
in partial fulfillment for the requirements of B.E Degree in Electrical Engineering

(July), 2020

CERTIFICATE OF CORRECTIONS & APPROVAL

Certified that work contained in this thesis titled **"Securing Hackable IoT Enabled LTE Devices***"*, carried out by "NC Muhammad Abdullah Farooq, NC Umar Ahmad Khan, NC Saif Ur Rehman and NC Abdul Samad" under the supervision of "Asst Prof Dr Mir Yasir Umair" partial fulfillment of Degree of Bachelors of Electrical Engineering, in Military College of Signals, National University of Sciences and Technology, Islamabad during the academic year 2019-2020 is correct and approved. The material that has been used from other sources has been properly acknowledged / referred.

Approved by

Supervisor

_____

_____

Date: _____

DECLARATION

No portion of work presented in this thesis has been submitted in support of another award or qualification in either this institute or anywhere else.

Plagiarism Certificate (Turnitin Report)

This thesis has been checked for Plagiarism. Turnitin report endorsed by Supervisor is attached.

Signature of Student

Muhammad Abdullah Farooq

Registration Number

Signature of Student

Umar Ahmad Khan

Registration Number

Signature of Student

Saif Ur Rehman

Registration Number

Signature of Student

Abdul Samad

Registration Number

Signature of Supervisor

Acknowledgements

We are thankful to our Creator Allah Subhana-Watala to have guided us throughout this work at every step and for every new thought which your setup in my mind to improve it. Indeed, we could have done nothing without Your priceless help and guidance. Whosoever helped me throughout the course of my thesis, whether my parents or any other individual was Your will, so indeed none be worthy of praise but You.

We are profusely thankful to our beloved parents who raised us when we were not capable of walking and continued to support us in every department of my life.

We would also like to express special thanks to our supervisor Dr. Mir Yasir Umair for his constant guidance which helped us in this endeavor and also for Mobile Communication Systems course which he has taught us. We can safely say that we haven't learned any other engineering subject in such depth than the ones which he has taught.

Finally, we would like to express our gratitude to ASC Zeeshan Ali, his drill machine and many others who have rendered valuable assistance to our study.

DEDICATION

Dedicated to our exceptional parents and adored siblings whose tremendous support and cooperation led us to this wonderful accomplishment.

ABSTRACT

In this modern era of Global World, the importance of Internet and Telecommunication increases a lot. Everyday business deals, diplomatic relations, health monitoring devices, weather stations, smart home appliances, goods monitoring appliances and so on, all are using Internet services. We are living in the world where intruders might be present. They can get access to our network and monitor our communication.

So, confidentiality and integrity of data might be lost. Currently, Wi-Fi communication is very much secured as compared to GSM/LTE, but this is not moveable. Wi-Fi communication has some limitations, it can't be used in mobile devices i.e. weather stations, transport and etc. LTE communication is secured but the link between sender and receiver is not secure. We are making an IoT device which will be secure and provide authentication as well. This device will use LTE SIM-7000E module, Wi-Fi ESP-8266 module, Temperature sensor BMP-280, Arduino, Google Sheets, PCB and AES-128-bit encryption. BMP-280 sensor will measure the temperature and send it to Google Sheets, where AES-128-bit encryption will be applied. After this data will be sent to cloud services (Google Drive) using LTE communication and end user can monitor it on the Android App. As the data is encrypted, hacker/intruder can't view the data in the form of plain text. Even if the cloud services get compromised your data will remain secure. In this way you can monitor your data anywhere in the world without worrying about security and privacy.

Table of Contents

Table of Figures

CHAPTER 1: INTRODUCTION

CHAPTER 1: INTRODUCTION TO IOT

1.1 Overview

SHIELD communicates data to user with Wi-Fi and LTE as mode of communication while keeping data safe from any unauthorized person who may intercept the data. Our IoT device receives data from its sensors and encrypts it before transmitting to cloud and the cloud receives the data and re transmits the data to android application and windows GUI for the user to view his data.

1.2 Problem Statement

IoT enabled equipment is being used worldwide. They are used as monitors**;** control systems and many other devices are applications of Internet of Things. The drawback of some of these devices is that they are not mobile and some devices which are mobile can be moved only within a small radius of a few meters and data transmitted by these devices does not insure confidentially and integrity. In the wrong hands this data can be used to bring harm to a person or to a corporation or even sold to data centers. SHIELD has been created to overcome this problem in the future to come which is a future dominated by IoT.

1.3 What is IoT?

IoT stands for Internet of things. In simple words IoT is the way of connecting many devices to the Internet and to other devices and all such devices interact with each other to collect and share data. An IoT system consists of sensors/devices which communicate with each other and to the cloud through connectivity (which can be wired or wireless). Once the data gets to the cloud, software's (or Web App) processes it and then might decide to perform an action, such as sending an alert or automatically adjusting the sensors/devices without the need for a human to monitor the process. Automated surveillance, smart transportation, Artificial Intelligence enhanced energy management systems, automated irrigation systems and environmental monitoring all are

examples of internet of things applications for smart cities of the future. According to recent statistics round about 20 billion IoT devices are being used around the world today and the total number of devices will increase to 50 billion in 2025. And IoT market value is expected to reach 6.2 trillion dollars by 2025.

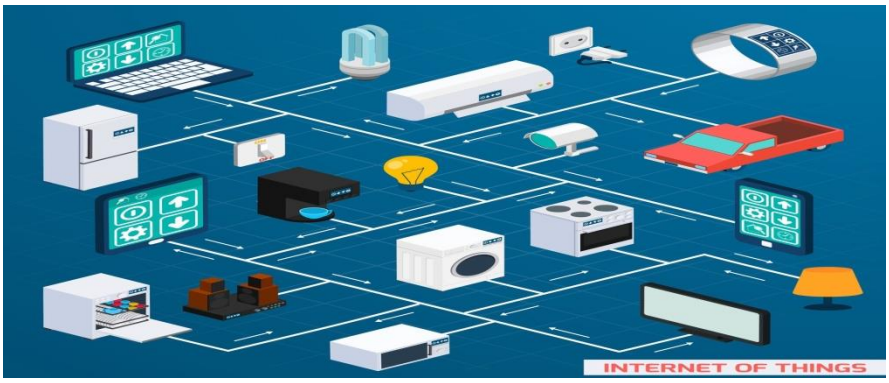

Figure 1: *IoT Basic Structure*

1.4 Why IoT?

IoT provides a lot of benefits in every field.

- IoT is an opportunity for data scientists to gather large amounts of data easily
- IoT will provide promising career opportunities in the future
- IoT has the feature of real time monitoring of data
- IoT provides connectivity from domestic level to industrial level

1.5 Hardware and Protocols

IoT infrastructure in the ambient of IoT devices can be divided into the two classifications

1. General Devices
2. Sensing Devices

General devices do the processing and establishing connectivity for the platforms they are connected to which can be by a wired network or wireless interfaces. They are the main components for data collection and information processing and parsing.

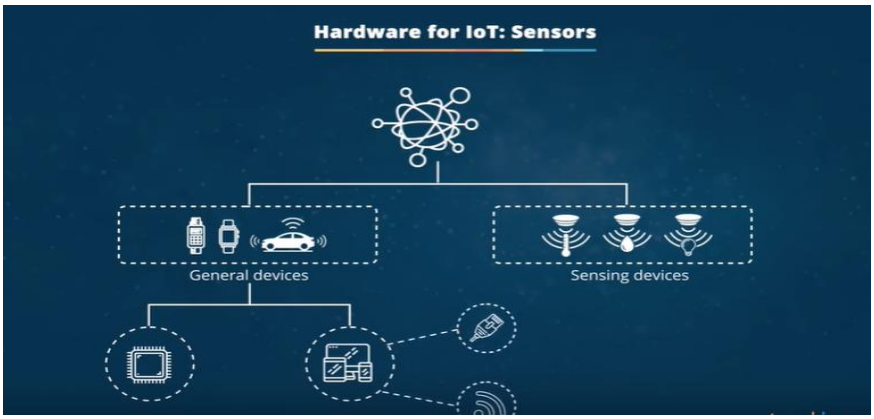

Figure 2:*Hardware_Sensors*

Aside from sensors, actuators are another very important part of an IoT device that performs similar functions with different capabilities; they work as an interface between sensors and embedded processing modules. They collect various information like humidity, light intensity this information is computed using edge layer which typically assists between the cloud and the sensor. They are the layers that store fragmented transfer of information; at the end backend server of the cloud processes this information from both the sensors and actuators for the major components of the IoT device. Sensors can measure temperature, humidity, light intensity and other key parameters of environment.

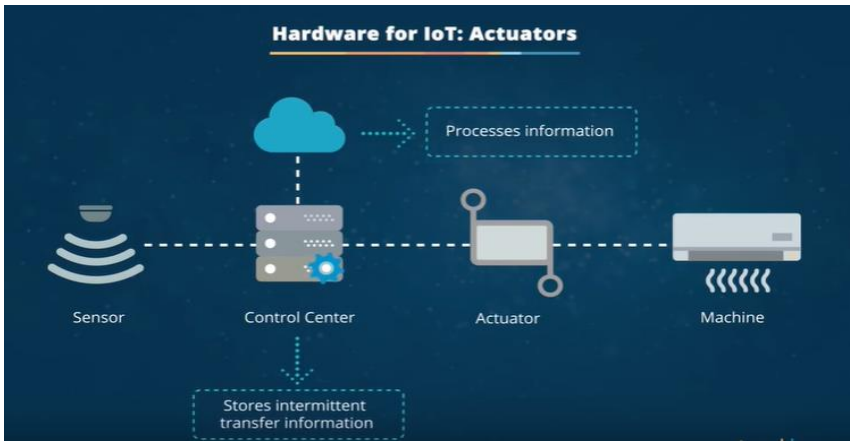

Figure 3: *Hardware_Actuators*

1.6 Device Management Platforms

The second most important part of IoT is its Device Management Platforms or DMP's. The DMP's as they are shortly called are the platforms through which these devices interact with an application layer through network gateways. DMP's come with various functionalities which include operating system upgrade, security patching and analysis, metrics and diagnostics report and alert mechanisms for triggers.

1.7 Communication Protocols and Standards

Various Communication protocols are available to set up an IoT.

- **Satellite**

  This enables a cell phone to wirelessly communicate through an antenna situated within 10 to 15 miles. They have a stable and universal connection but it is very slow.

- **Wi-Fi**

  Wi-Fi is based on 2.6 GHz and 5GHz of operating frequency; it provides internet access

within a certain radius and is an affordable and easily available option. It has well protected security protocols.

- **Radiofrequency**

  It is one of the simplest forms of communication systems. For example, a Zigbee is an example of such system, which use low power RF radio. It consumes low energy which is very advantageous. It is relatively simple to configure for small scale implementation.

- **RFID**

  Radio-frequency identification or commonly known as RFID uses wireless electromagnetic fields to identify sensors, objects and authentication of personnel. It does not require power.

- **Bluetooth**

  It is useful for short term and short distance data exchange. It is present in every Smartphone and has sensors but it very slow and is not recommended.

- **Near Field Communication**

  NFC uses electromagnetic induction and loop antennas. It comes with encryption as low speed communication and is used for short range data exchange.

1.8 Future of IoT Connectivity

IoT will get a boost with new communication technologies being developed like 5G.The low latency and higher bandwidth of 5G will decrease response time of IoT devices ensuring faster communication and results. 5G and IoT are intertwined technologies both can be sliced and diced for predictive analytics and real-time business decisions.

1.9 IoT Device Architecture

There are four basic layers in an IoT device.

- The base layer of IoT devices. This includes all the sensory and computative components like sensors with the ability to sense, compute and connect other devices.
- The second layer which is the IoT entryway or accumulation layer. This layer essentially totals information from different sensors.
- The third layer depends on cloud it's called handling motor or occasion preparing layer. It has various calculations and information handling components. They are at last shown on a dashboard. This layer essentially forms the information acquired from the sensor layer.
- The last layer is known as the application layer or API the board layer. It goes about as an interface between outsider application and foundation. The whole scene bolstered by gadget directors and character and access administrators which are valuable for security of the design.

1.10 Security Standards of IoT

Many Individuals and consortiums have taken part to create and set the standard of IoT so that it covers the entire network and is in connection with endpoint systems. It brings a structural approach with security controls in place. Most likely the most demanding and essential requirement for the widespread realization of many IoT solutions is security. Security in IoT has an exceptionally wide scope in at least four dimensions. In terms of security scope, it includes rarely addressed tasks such as trusted sensing, computation, communication, privacy, and digital forgetting. It also asks for new and better techniques for the protection assets of IoT which include hardware, software, and data that considers the possibility of physical access to IoT devices. Sensors and actuators are common components of IoT devices and pose several unique security challenges including the integrity of physical signals and actuating events. Finally, during processing of data many man in the middle attacks can occur.

1.11 Best Practices of IoT Security

There are many leading trends in IoT security.

- Hardware should be tampered proof.
- They should be tested for rigorously.
- Specific data protection algorithms.
- Must be updated on firmware.
- The network components must be authentic with encryption.

Other security measures include privacy protection for sensitive information and other regulations like safe harbor statements.

1.12 Working Principle

The Working Principle of SHIELD has been divided into the following units:

- Sensors
- Processing Unit
- Cloud
- Android Application and Windows GUI

1.12.1 Sensors

The sensors are the source of user data which can be temperature, pressure, humidity sensor to a smart thermostat which will provide data for transmission

1.12.2 Processing Unit

The processing unit is responsible for acquiring the data from sensors, formatting data for transmission, establishment of link for transmission and transmitting the data. For this purpose,

we are using ESP8266 which are Wi-Fi modules with onboard Microcontroller Units.

1.12.3 Cloud

The cloud is responsible for receiving, storing and retransmitting the data. For this purpose, we are using Google Drive (a cloud by Google). For the reception and retransmission of we write scripts in Google Scripts which are written in Node Java Script.

1.12.4 Android Application

The Android Application is responsible for authenticating the user, receiving data from cloud and decrypting the data so it can Displayed on GUI for user readability. The android application has been created using Android Studio and it is scripted in Kotlin for Android.

1.12.5 Windows Application

The Windows application (Python GUI) is another platform for the user to access his or her data. The GUI registers and authenticates user and shows them their encrypted data first which they can decrypt if they want to. The GUI has been created in Visual Studio and scripted in Python 3.

1.13 OBJECTIVES

The primary objective of this project is to design a mobile IoT device which sends data to user while keeping it secure from any attackers. Furthermore, our objective is to minimize the cost of final product so it can me mass produced. The end goals or objectives can be broadly identified as:

1. Design and Develop IoT devices which work on both Wi-Fi and LTE based Mi-FI are battery sustained.
2. Microcontroller unit which processes and transmits data to cloud

3. Giving the device a body, which is strong but does not affect signal strength
4. Android application development for user end
5. Windows Application so that the user may be able to view the data on a desktop or a laptop

1.14 Scope of the Project:

This project has the scope to be deployed anywhere in the world, from a small cupboard to an airplane which has sensors on it. Smart Home, Connected Cars, Wearable's, Industry Internet of Things (IIoT), IoT in Healthcare, IoT in Agriculture, Smart Cities are some of the possible applications of our project. It is a time, cost and security efficient solution for IoT devices.

1.15 Organization of Document

This document is divided into the following main parts:

1. The first part includes the introduction and objectives of the project.
2. The second part is the summary of the literature review and previous work done on this project .
3. The third part includes the hardware and software description of the project.
4. The fourth part shows Plan for Implementation.
5. The fifth part will show working of each part.
6. The sixth and final part is Appendixes which is abbreviations used in the theses and the synopsis of this project.

CHAPTER 2: LITERATURE REVIEW

CHAPTER 2: LITERATURE REVIEW

2.1 Overview

The research work in this chapter is related to IoT (Internet of things), it has two parts. First, part includes the basic overview of IoT. In this we look about, what IoT is? How it works? What is the scope of IoT? Second, part of this chapter includes the security of IoT. In this we will look about security mechanism of IoT based on LTE and Wi-Fi.

Today's most of communication is based on Internet. Everyone wants to use smart devices for their daily business and normal life. Smart devices are well capable of doing work efficiently. But the major problem is the security of the data which these devices are sending and receiving. Some of the devices use Wi-Fi and others use GSM/LTE for communication. Previously, work is done on the security of data for the Wi-Fi communication. But Wi-Fi communication has certain limitations i.e. immobility. On the other hand, no work is done for the security of data which uses GSM/LTE. IoT is an exponentially growing field; almost 20 billion IoT devices are present today. But the security of the most of IoT devices never come into consideration, especially IoT based on LTE/GSM.

We are aiming to build a secure IoT device which will work in both Wi-Fi and LTE/GSM environment securely and effectively. The main purpose of this project is to provide security of data to every single customer using IoT appliances.

2.2. Literature Review

The following research paper was consulted to get guidance related to SHIELD:

- Roger Piqueras Jover, Ilona Murynets **"Connection-less communication of IoT devices over LTE mobile networks"** 2015 12th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)

2.3. Previous Work on this topic

As far as we have researched on this topic there has been no previous work on this topic in the past. So, it is the first project of its kind.

CHAPTER 3: DESIGN REQUIREMENTS

CHAPTER 3: DESIGN REQUIREMENTS

3.1. Project Hardware

The project utilizes the following hardware components:

3.1.1. ESP8266 NODEMCU

ESP8266 **NodeMCU** is an open-source microcontroller unit with integrated Wi-Fi and Bluetooth chipsets. Its firmware is ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module. The operating system or firmware of this module is written in the Lua Scripting Language.

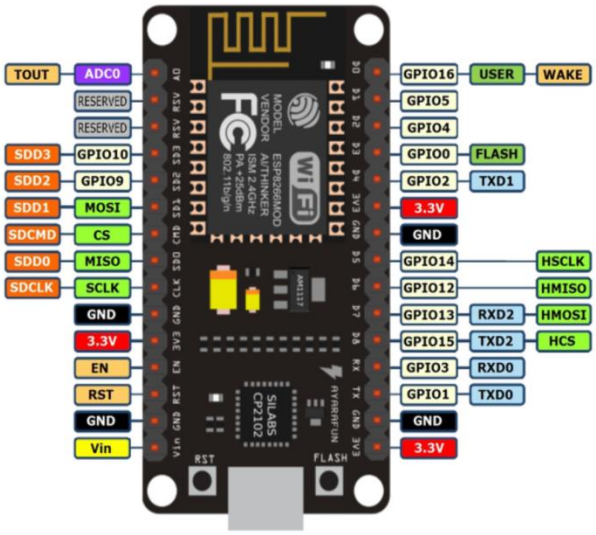

Figure 4: ESP8266 Module

3.1.2. BMP-280

The BMP280 is an absolute barometric pressure sensor, which is especially feasible for mobile applications. Its small size and low power usage enable it to be used in small circuits or mobile devices such as GPS modules or watches. The BMP280 is based on Bosch's proven piezo-resistive pressure sensor technology which gives it high efficiency and linearity as well as stability for a long period of time. Numerous device operation options guarantee for highest flexibility. The device is optimized in terms of power consumption, resolution and filter performance.

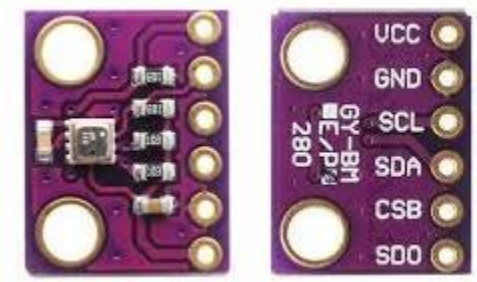

Figure 5: *BMP-280*

3.2. Project Software

The project required following development environments and software for this project:

3.2.1. Arduino IDE

The Arduino Integrated Development Environment (IDE) is a cross-platform application (for Windows, macOS, Linux) that is written in functions from C and C++. It is used to write and upload programs to Arduino compatible boards, but also, with the help of 3rd party cores, other vendor development boards.

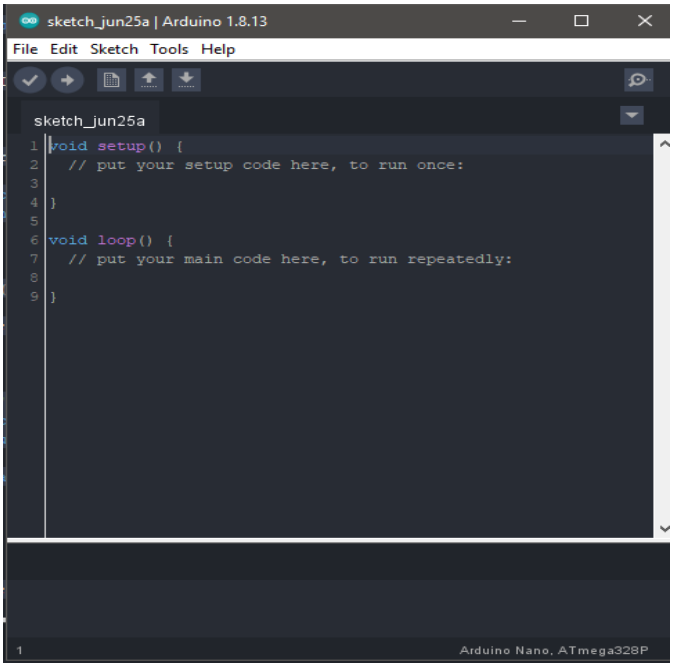

Figure 6: *Arduino IDE*

3.2.2. Android Studio

Android Studio is the official integrated development environment for Google's Android operating system, built on JetBrains 'IntelliJ IDEA software and designed specifically for Android development.

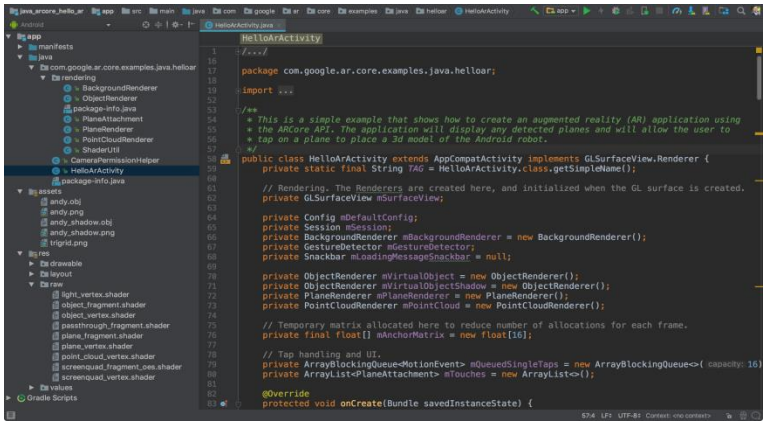

Figure 7: *Android Studio*

3.2.3. Visual Studio

Microsoft Visual Studio is an incorporated scripting environment from Microsoft. It is used for the development and testing of computer programs, websites, web applications, web administrations and versatile mobile applications (android and iOS)

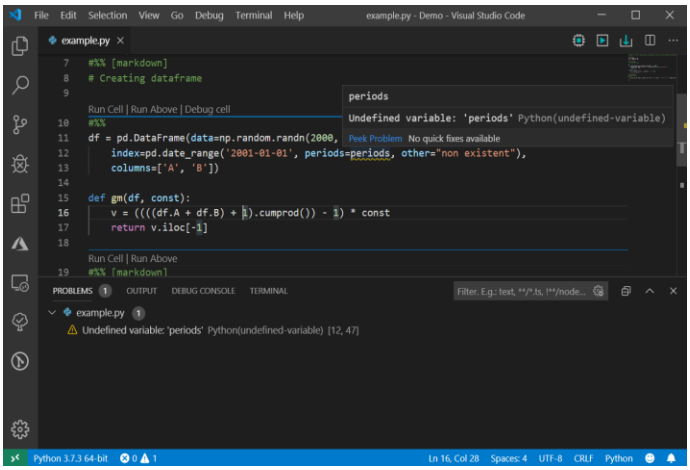

Figure 8: *Visual Studio*

3.2.4. Proteus

The Proteus Design Suite is a trademarked software tool suite used primarily for electronic design automation. The software is used mainly by electronic design engineers and technicians to create schematics and electronic prints for manufacturing printed circuit boards.

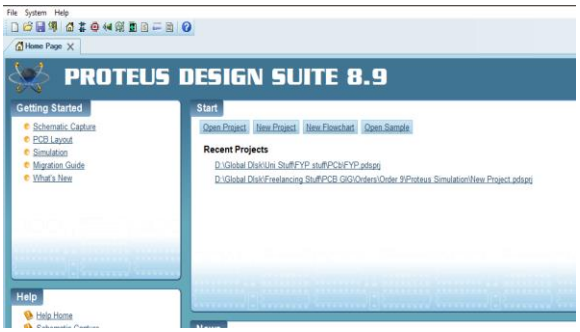

Figure 9: *Proteus Design Suite*

CHAPTER 4: PLAN FOR IMPLEMENTATION

CHAPTER 4: PLAN FOR IMPLEMENTATION

4.1. Project Timeline

The below picture illustrates the timeline we followed for our project

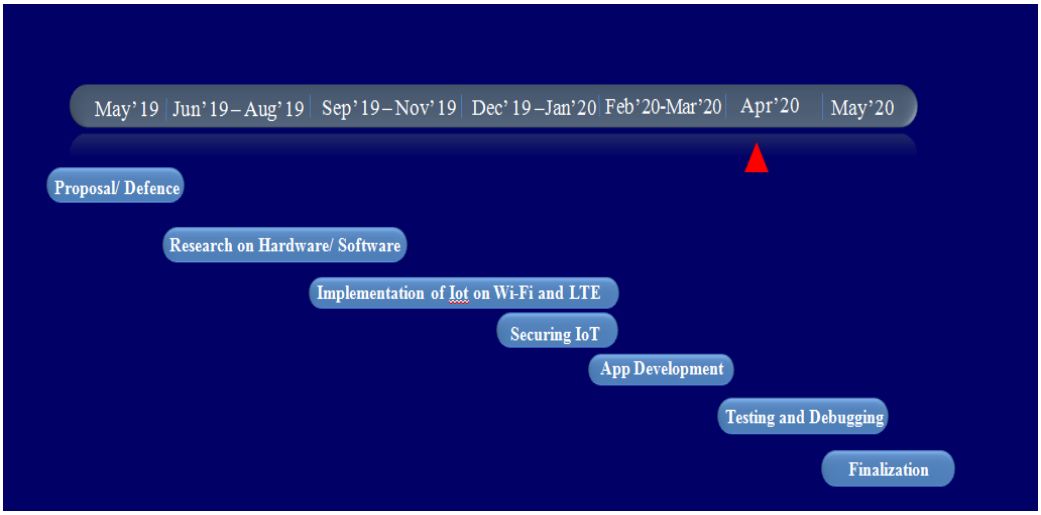

Figure 10: *Project Timeline*

4.2. Project Implementation

The implementation of the project follows the following steps:

4.2.1. IoT Device

The IoT device has been designed using BMP280 as sensor, ESP8266 as gateway and MCU for Wi-Fi based device. For its design we first integrated sensor with Microcontroller unit and gateway and then established the link between gateway and cloud. Then we applied encryption and other security measures to data being transmitted and to the link to ensure privacy. The codes which the microcontroller follows have been developed in Arduino IDE and optimized for maximum efficiency. The power delivery for the device has been optimized so it can run for a long time on a small battery.

4.2.2. Android Application

The android application was developed in Android Studio. The app first authenticates the user operating the application and then acquires data from cloud and process (decrypt and format) data for user readability

The user can login to the app using the credentials provided by us to ensure security and success in authentication. The credentials are also encrypted and stored on the data base. Attached below is a picture of the login screen of the Application.

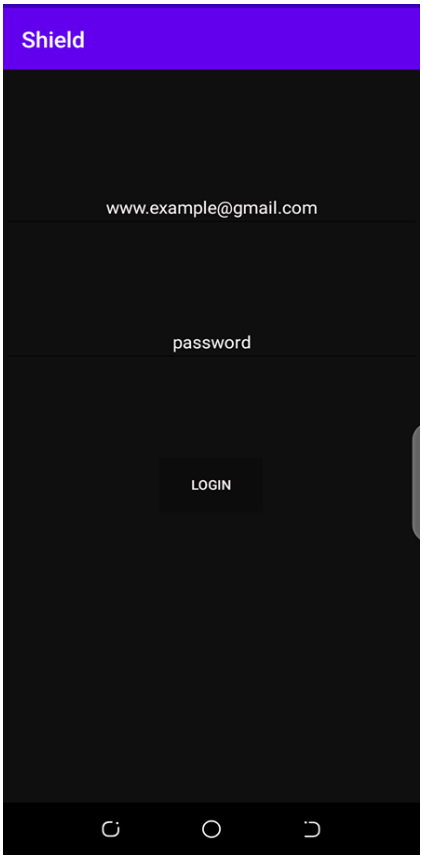

Figure 11: *Android Application*

4.2.3. Python GUI for Windows

We developed a GUI for user to access his or her data in Windows environment. It also decrypts the data received from Google Sheets but it has an interactive interface for user to first view his encrypted data and then press a button to decrypt the data. A picture of the login screen has been shared below.

Figure 12: *Python based GUI*

CHAPTER 5: WORKING

CHAPTER 5: WORKING

5.1. Microcontroller Working

5.1.1. Workflow

The project follows the following work flow.

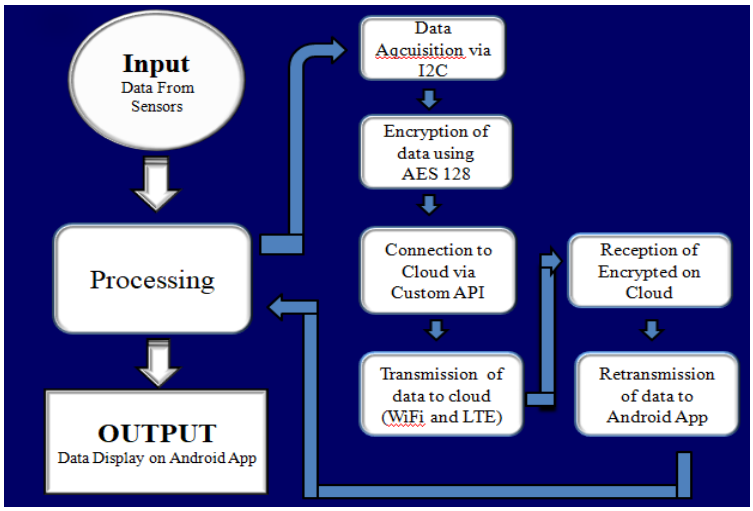

Figure 13: *Workflow*

The data acquisition is done through ESP 8266 which is integrated with BMP 280 sensors which collects the data of the environment and the ESP 8266 formats the data then it encrypts the data using the AES 128-bit Cipher and then parses the data for transmission and transmits the data to Google sheets.

5.1.2. ESP8266 code and working

The microcontroller is the most important part of an IoT device. Our IoT device follows the following code:

```
#include <ESP8266WiFi.h>

#include <WiFiClientSecure.h>

#include <Adafruit_BMP280.h>                                                    // library to use BMP280
objects and get direct values of parameters

#include "AESLib.h"



AESLib aesLib;

Adafruit_BMP280 BMP;

String plaintext = "HELLO WORLD!";

String t,a;

String readBuffer ;

char cleartext[256];

char ciphertext[512];



uint8_t mac[6] {0xA8, 0xD9, 0xB3, 0x0D, 0xAA, 0xCE};    //New Spoofed MAC address



const char* ssid = "Malik";    // name of your wifi network!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

const char* password = "123malik123";    // wifi pasword !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

IPAddress staticIP(192, 168, 100, 90); //ESP static ip

IPAddress gateway(192, 168, 100, 1);   //IP Address of your WiFi Router (Gateway)
```

```
IPAddress subnet(255, 255, 255, 0);  //Subnet mask

IPAddress dns(208, 67, 220,220);  //OPEN DNS which is provided by cisco and is the most secure

const char* host = "script.google.com";

const int httpsPort = 443;

// Use WiFiClientSecure class to create TLS connection

WiFiClientSecure client;

// SHA1 fingerprint of the certificate, don't care with your GAS service

const char* fingerprint = "46 B2 C3 44 9C 59 09 8B 01 B6 F8 BD 4C FB 00 74 91 2F EF F6";

String GAS_ID = "AKfycbyVBKIQAhLQ3UXT-DPgdUdNRUamfk0b0DRgTHfyo6jHmurYcaQu";    // Replace by your GAS service id           !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!




// AES Encryption Key

byte aes_key[] = { 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30 };



// General initialization vector (you must use your own IV's in production for full security!!!)

byte aes_iv[N_BLOCK] = { 11, 69, 75, 1, 99, 854, 712, 658, 0, 7, 9671, 111, 2, 3, 4, 71 };



// Generate IV (once)

void aes_init() {



aesLib.gen_iv(aes_iv);

// workaround for incorrect B64 functionality on first run...
```

```
Serial.println(encrypt(strdup(plaintext.c_str()), aes_iv));

}



String encrypt(char * msg, byte iv[]) {

int msgLen = strlen(msg);

Serial.println("message length = ");

Serial.println(msgLen);

char encrypted[4 * msgLen]; // AHA! needs to be large, 2x is not enough

aesLib.encrypt64(msg, encrypted, aes_key, iv);

Serial.print("encrypted = "); Serial.println(encrypted);

return String(encrypted);

}




void setup()

{



// initialize BMP280 sensor

bool status;

status = BMP.begin(0x76);

if (!status) {

Serial.println("Could not find a valid BMP280 sensor, check wiring!");          // checks if there is a
```

```
BMP 280 at the specified address it can be found using i2c scanner

while (1);}

Serial.begin(115200); //Serial

Serial.println("Changing Mac address");

Serial.print("OLD ESP8266 MAC: ");

Serial.println(WiFi.macAddress()); //This will read MAC Address of ESP



wifi_set_macaddr(0, const_cast<uint8*>(mac));   //This line changes MAC adderss of ESP8266



Serial.print("NEW ESP8266 MAC: ");

Serial.println(WiFi.macAddress());

//connecting to internet

Serial.print("connecting to ");

Serial.println(ssid);

WiFi.mode(WIFI_STA);

WiFi.begin(ssid, password);

while (WiFi.status() != WL_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.println("WiFi connected");

Serial.println("IP address: ");
```

```
Serial.println(WiFi.localIP());

Serial.println("new gateway is ") ;

Serial.println("dns is ") ;

Serial.println(dns) ;



Serial.println("aes_init()");

aes_init();

}



void wait(unsigned long milliseconds) {

unsigned long timeout = millis() + milliseconds;

while (millis() < timeout) {

yield();

}}

void loop()

{

float a = BMP.readAltitude(1012.025);

float t =BMP.readTemperature();



Serial.print("Temperature = ");

Serial.print(t);

Serial.print(" Altitude = ");

Serial.println(a);
```

```
sendData(t, a);



delay(30000);



}



// Function for Send data into Google Spreadsheet

void sendData(float tem, float alt)

{

String string_temperature =  String(tem);

t= string_temperature ;

String string_altitude =  String(alt);

a=string_altitude ;




byte enc_iv[N_BLOCK] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 }; // iv_block gets written to,
provide own fresh copy...

for(int i=0 ; i<2 ;i++){




if (i== 0) {
```

```
readBuffer = string_temperature;
Serial.println("INPUT:" + readBuffer);

sprintf(cleartext, "%s", readBuffer.c_str()); // must not exceed 255 bytes; may contain a newline

// Encrypt

string_temperature = encrypt(cleartext, enc_iv);
sprintf(ciphertext, "%s", string_temperature.c_str());
Serial.print("Ciphertext 1 : ");
Serial.println(string_temperature);  }
if (i== 1){
readBuffer = string_altitude;
Serial.println("INPUT:" + readBuffer);

sprintf(cleartext, "%s", readBuffer.c_str()); // must not exceed 255 bytes; may contain a newline

// Encrypt

string_altitude = encrypt(cleartext, enc_iv);
sprintf(ciphertext, "%s", string_altitude.c_str());
Serial.print("Ciphertext 2 : ");
```

```
Serial.println(string_altitude);  }

}

client.setInsecure();

Serial.print("connecting to ");

Serial.println(host);

if (!client.connect(host, httpsPort)) {

Serial.println("connection failed");

return;

}



if (client.verify(fingerprint, host)) {

Serial.println("certificate matches");

} else {

Serial.println("certificate doesn't match");

}



String JsonStr = "/macros/s/" + GAS_ID + "/exec?temperature=" + t + "&altitude=" + a  ;

Serial.print("requesting URL: ");

Serial.println(JsonStr);



client.print(String("GET ") + JsonStr + " HTTP/1.1\r\n" +

"Host: " + host + "\r\n" +

"User-Agent: BuildFailureDetectorESP8266\r\n" +
```

```
"Connection: close\r\n\r\n\r\n");



Serial.println("request sent");

while (client.connected()) {

String line = client.readStringUntil('\n');

if (line == "\r") {

Serial.println("headers received");

break;

}

}

String line = client.readStringUntil('\n');

if (line.startsWith("{\"state\":\"success\"")) {

Serial.println("esp8266/Arduino CI successfull!");

} else {

Serial.println("esp8266/Arduino CI has failed");

}

Serial.println("reply was:");

Serial.println("==========");

Serial.println(line);

Serial.println("==========");

Serial.println("closing connection");}
```

1. The microcontroller (esp8266) first connects to the provided Wi-FI connection with function *WiFI.connect(SSID,Password)* in this the SSID and password are the parameters required for it to be able to connect to a Wi-Fi outlet .

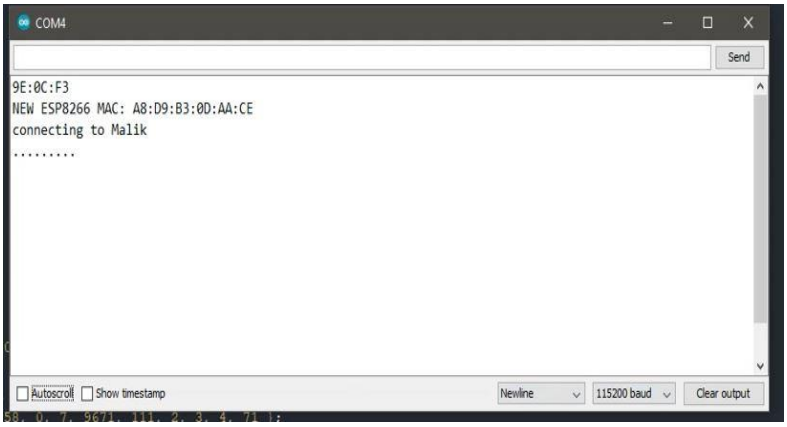

Figure 14: *ESP8266 in operation-1*

2. The controller the calls the *BMP.Begin(ox76)* function with 0x76 as the address of sensor on the I2C bus and it created a BMP object and checks its status to see if it is actually connected

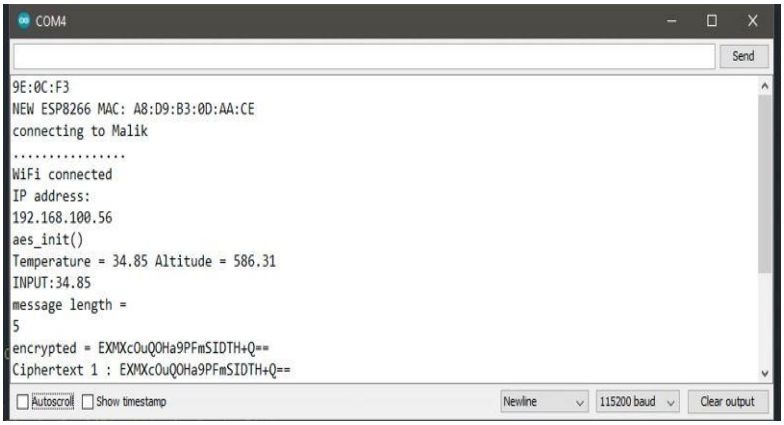

Figure 15: *ESP in operation-2*

3. The next is to call the *aes_init()* function which initializes the initialization vector and padding vector needed for AES encryption

4. *Void loop()* is the next function to be accessed and it runs in every iteration . The first part in this loop is to acquire data from BMP280 via I2C which is done through *bmp.Readtemperature () and bmp.Readaltitude ()* .

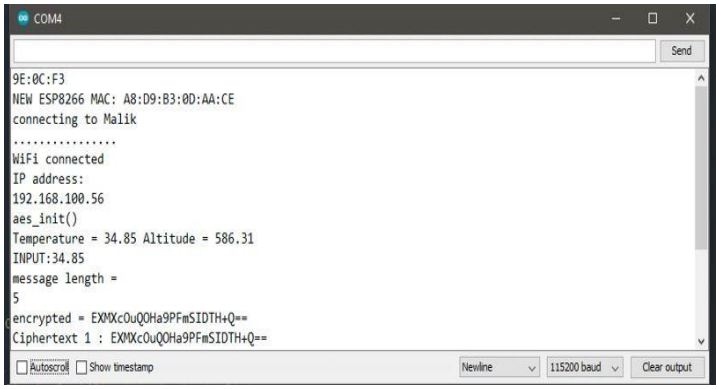

Figure 16: *ESP in operation-3*

5. *Void Send data (t,a)* is responsible for sending data to Google Script web App . But first it converts the data to String format. Then Calls the function *encrypt (key,iv,data)* to encrypt the data before transmission . It then matches the SHA 1 fingerprint of the

Google Script before establishing connection. Connection is established with *Client.Connect (host, port)* function. It receives the URL of the host and port number for the transmission. It then sends data to Web and App verifies response and then closes the connection.

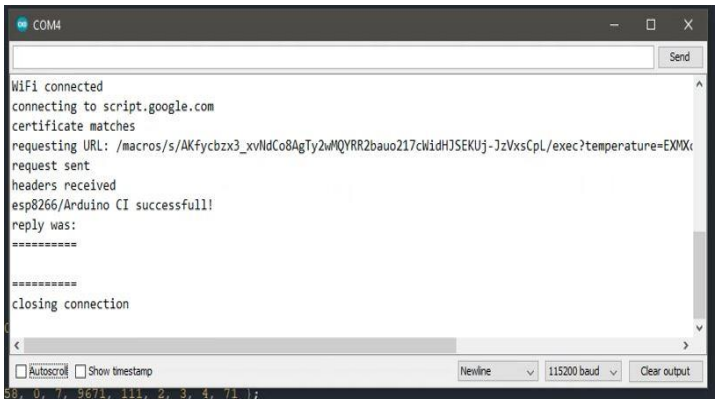

Figure 17: *ESP in operation-4*

5.2. Google Sheets Interface

For data to be transmitted to Google sheets, it is first received on a web applet created in Google script which is written in Java Script. The script separates the data and then transfers it to relevant column of Google Sheets with time stamp.

5.2.1. Google Script code

The Google script is shown below:

```
function doGet(e) {

  Logger.log( JSON.stringify(e) );  // view parameters

  var result = 'Ok'; // assume success

  if (e.parameter == 'undefined') {
```

```
  result = 'No Parameters';
}
else {



  var sheet_id = '1Jh5_ZmYZpgwleF0VPK42VHBm2rJI2xrMkT0aKCZdGLE';               // Spreadsheet ID
  var sheet = SpreadsheetApp.openById(sheet_id).getActiveSheet();              // get Active sheet
  var newRow = sheet.getLastRow() + 1;
  var rowData = [];
  rowData[0] = new Date();
              // Timestamp in column A
  for (var param in e.parameter) {
    Logger.log('In for loop, param=' + param);
    var value = stripQuotes(e.parameter[param]);
    Logger.log(param + ':' + e.parameter[param]);
    switch (param) {
       case 'temperature': //Parameter
        rowData[1] =value; //Value in column B
        result = 'Written on column B';
        break;
       case 'altitude': //Parameter
        rowData[2] = value; //Value in column C
        result += ' ,Written on column C';
```

```
          break;



      }

    }

    Logger.log(JSON.stringify(rowData));

    // Write new row below

    var newRange = sheet.getRange(newRow, 1, 1, rowData.length);

    newRange.setValues([rowData]);

  }

  // Return result of operation

  return ContentService.createTextOutput(result);

}

/**

* Remove leading and trailing single or double quotes

*/

function stripQuotes( value ) {

  return value.replace(/^["']|['"]$/g, "");

}
```

The encrypted data on the Google Sheet can viewed on the sheet too if you have access to the sheet and it has been shown below.

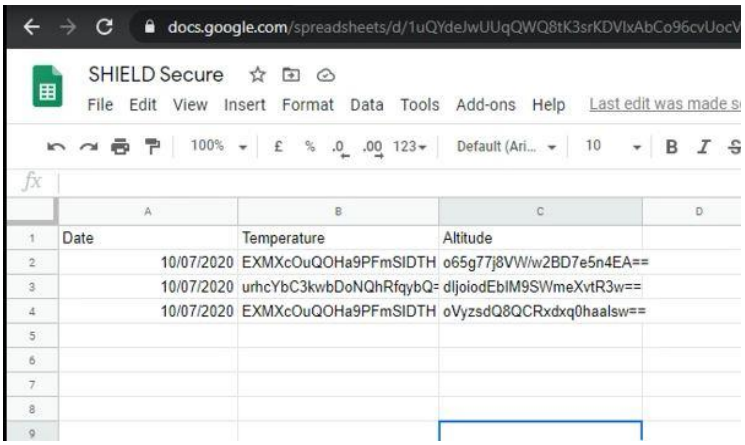

Figure 18: *Encypted data on Google Sheets*

5.2.2. Working of Code

The web app first receives the data then use switch () statement to see if relevant parameters are present if they are it transmits them to the attached Google sheet. The Google sheet is identified via its sheet ID extracted from its URL (Uniform Resource Locator). The data is transmitted to specified column. Each column is specified for one specific parameter.

5.3. Google Sheet to Android App Transmission

For data to be transmitted to android app it has to again be transmitted via a web app. Android App sends GET request to web app and after authentication it acquires data from Google Sheet and sends it to Android App.

5.3.1. Google Script code for transmission of data

```
Var ss= SpreadsheetApp.openByUrl("https://docs.google.com/spreadsheets/d/1Jp0ldvOYcEopaULFzAUkzrexR-mw8XdzmnPkdvGYX3I/edit#gid=0");

var sheet = ss.getSheetByName('Items'); // be very careful ... it is the sheet name .. so it should match

function doPost(e){

var action = e.parameter.action;

if(action == 'addItem'){

  return addItem(e);

}}

function addItem(e)

{

var date =  new Date();



var id  =  "Item"+sheet.getLastRow(); // Item1



var itemName = e.parameter.itemName;



var brand = e.parameter.brand;



sheet.appendRow([date,id,itemName,brand]);



   return ContentService.createTextOutput("Success").setMimeType(ContentService.MimeType.TEXT);
```

```
}
```

5.3.2. Working of code

The web first authenticates the Get request from Android App and after authentication it acquires data from Defined Google Sheet and transmits it to the Android App in JSON (Java Script Object Notation) format with data and time added with it.

5.4. Reception on Android App

The android app acquires data from Google Sheet to decrypt and display it. The code of the Android App has been shown below.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity"

android:orientation="vertical">

<FrameLayout

android:layout_width="match_parent"
android:layout_height="match_parent"
android:background="#000"
>
<LinearLayout
android:orientation="horizontal"
android:layout_marginTop="100dp"
android:layout_width="match_parent"
android:layout_height="wrap_content">

<Button
```

```
android:id="@+id/howBtn"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginLeft="60dp"
android:background="#0E0D0D"
android:text="How To"
android:textColor="#FFFFFF">

</Button>

<Button
android:id="@+id/showToBtn"
android:text="Show Data"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginLeft="95dp"
android:background="#131212"
android:textColor="#FFFFFF">

</Button>
</LinearLayout>

<LinearLayout
android:orientation="horizontal”

android:layout_marginTop="200dp"
android:layout_width="match_parent"
android:layout_height="wrap_content">

<Button
android:id="@+id/futureDatabtn"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginLeft="60dp"
android:background="#0E0D0D"


android:text="Future Work"
android:textColor="#FFFFFF">

</Button>

<Button
android:text="About"
android:id="@+id/aboutWorkBtn"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginLeft="95dp"
android:background="#131212"
android:textColor="#FFFFFF">

</Button>
</LinearLayout>
```

```
</FrameLayout>
</LinearLayout>


package com.example.shield


import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Button



class MainActivity : AppCompatActivity() {


override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
setContentView(R.layout.activity_main)


valbtn_show = findViewById(R.id.showToBtn) as Button
valbtn_about = findViewById(R.id.aboutbtn) as Button
valbtn_future = findViewById(R.id.futurebtn) as Button
valbtn_howto = findViewById(R.id.howBtn) as Button



btn_about.setOnClickListener{
valintent = Intent(this, About::class.java)
startActivity(intent)
}



btn_show.setOnClickListener{
valintent = Intent(this, Login::class.java)
startActivity(intent)
}


btn_future.setOnClickListener{
valintent = Intent(this, Future::class.java)
startActivity(intent)
}


btn_howto.setOnClickListener{
valintent = Intent(this, HowTo::class.java)
```

```
startActivity(intent)
}

}}
```

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayoutxmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:id="@+id/color"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:background="#100F0F"
android:orientation="vertical"
tools:context=".Display">

<TextView
android:layout_width="match_parent"
android:layout_height="80dp"
android:paddingTop="20dp"
android:text="Temperature"
android:textAlignment="center"
android:textColor="#FFFFFF"
android:textSize="40dp">
</TextView>

<TextView
android:id="@+id/temperature"
android:layout_width="match_parent"
android:layout_height="80dp"
android:layout_marginTop="50dp"
android:textAlignment="center"
android:textColor="#FDFDFD"
android:textColorHighlight="#FDFDFD">
</TextView>

<TextView
android:layout_width="match_parent"
android:layout_height="80dp"
android:paddingTop="20dp"
android:text="Altitude"
android:textAlignment="center"
android:textColor="#FDFBFB"
android:textColorHighlight="#FDFBFB"
android:textSize="50dp">
</TextView>

<TextView
android:id="@+id/altitude"
```

```
android:layout_width="match_parent"
android:layout_height="80dp"
android:layout_marginTop="50dp"
android:textAlignment="center"
android:textColor="#FBF8F8"
android:textColorHighlight="#FBF9F9">

                                    </TextView
import android.os.Bundle
import android.os.Handler
import android.os.Looper
import android.os.Message
import android.text.TextUtils.isEmpty
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity
importcom.google.api.client.extensions.android.http.AndroidHttp
import com.google.api.client.http.HttpTransport
import com.google.api.client.json.JsonFactory
import com.google.api.client.json.jackson2.JacksonFactory
import com.google.api.services.sheets.v4.Sheets
class Display : AppCompatActivity() {
private lateinit var data: String
valtemperature = mutableListOf<String>()
valaltitude = mutableListOf<String>()
private lateinit var temp:TextView
private lateinit var alti:TextView

override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
setContentView(R.layout.activity_display)
var i: Int = 0

while (i<res.size) {

temperature.add(res[i].replace("" + "[", "", true).replace("]", "", true))


i += 1


altitude.add(res[i].replace("" + "[", "", true).replace("]", "", true))


i +=1

}//loop ends

var j=0
```

```
while(j<2){

if (j==0){

 j += 1
threadMsg(temperature.toString()) }

else {

threadMsg(altitude.toString())

  j += 1}


} //while loop ends
```

Thread exception which may occur due to slow network or no internet issue

```
 } catch (t: Throwable) {

Log.i("Animation", "Thread  exception $t")}}


private fun threadMsg(msg: String?) {

                        if (msg != null && msg != "") {


valmsgObj: Message = handler.obtainMessage()
valb = Bundle()
b.putString("message", msg)


msgObj.setData(b)
handler.sendMessage(msgObj)
            }
                                                }
```

```
private valhandler = @SuppressLint("HandlerLeak")

object : Handler(Looper.getMainLooper()) {
override fun handleMessage(msg: Message){

valaResponse: String? = msg.data.getString("message")
if (null != aResponse) {
s=findViewById(R.id.temperature)
if(isEmpty(s.text)){
s.text=aResponse
                }
else {
alti = findViewById(R.id.altitude)
alti.text= aResponse
                }



              }
else {

Toast.makeText(baseContext, "Not Got Response From Server.", Toast.LENGTH_SHORT).show()
              }
            }
          }
      })

      background.start()


   }
}
```

It can be seen in the code above that the app first of all imports all the necessary libraries which are, then initialize variables for storing data and the objects are acquired from Google Sheets and the JSON parser function parses the objects so they can be read as string by the App.

Data is acquired from Google Sheet using Google Sheet API and it sends the encrypted data to the App. The App after formatting the data decrypts it so user can understand it easily.

An example of using this App has been elaborated below:

1. First the login screen. It will ask for a username and password and authenticate them when the user presses the Login button.

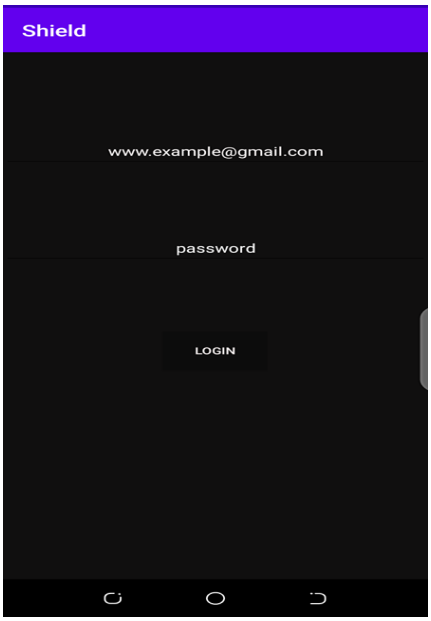

Figure 19: *Android Application in working-1*

2. Now once the user as logged in on the app he or she will see 4 more buttons. These are for “How to use app”, “Display Data”, “About” and “Future Work”.

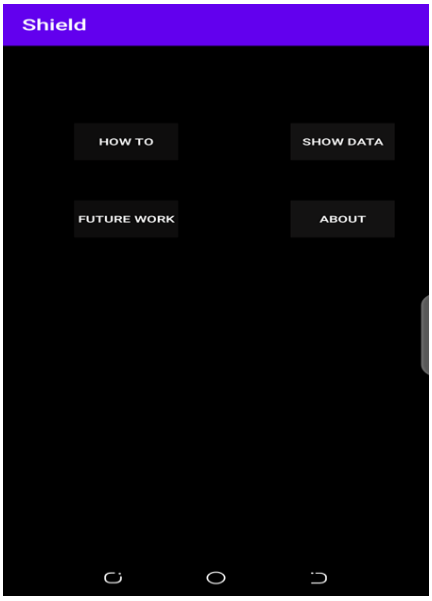

Figure 20: *Android Application in working-2*

3. Once user presses the “Show Data” button then his data will be acquired from Google Sheets using its API and it is parsed and decrypted and then shown to user .

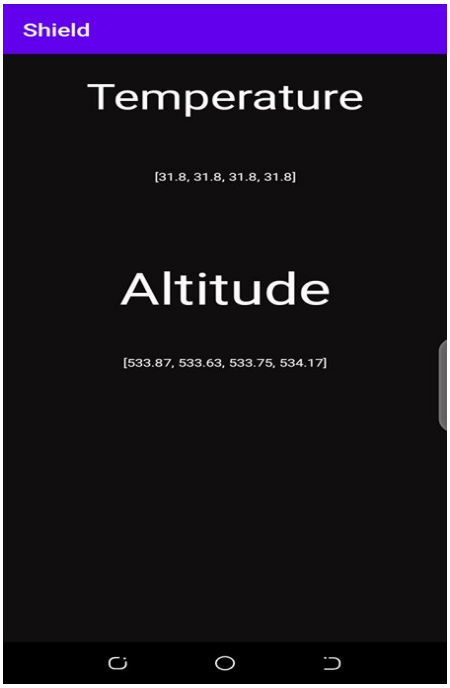

Figure 21: *Android Application in working-3*

5.5. Google Sheet to Python GUI

The GUI that displays the decrypted data was developed by using Tkinter library. Tkinter is a standard GUI library for Python. Python when combined with Tkinter offers a quick method of developing GUI applications. It provides an influential object-oriented interface to Tk GUI toolkit.

In the GUI what we are basically trying to do is that we have registered the users within our MariaDB database. Those registered users log in to the system and their private JSON files i.e. generated from Google Sheets, are authenticated and they see the encrypted data present at that sheet. A 'Decrypt' button decrypts all the encrypted data present on the sheets. That decrypted data is presented to the user so that he can see the data. The decrypted data in this case will only be visible to the user that was already present in the database and had a private JSON file generated at the time of installation.

The passwords in the MariaDB Database were firstly hashed in SHA256 format and then stored. This was done as a precaution so that if our database gets compromised the passwords will not be humanly readable (Parr, n.d.). The connection with the database was hardcoded into our GUI.

Our GUI will only be provided to our users so the hardcoding of the connection will not be an issue. New users will be registered into the database by using the Register button in our GUI.

The data of one sensor is uploaded on one sheet. We can add as many sensors as we want, but for every sensor a new JSON file will be required to be generated as the JSON file contains the authentication methods of the user. We will hardcode the new worksheet into the GUI provided to already existing users so that the user can access the new Google Sheet that is displaying the data of new sensor from now onwards. We are keeping the ability of registering new worksheets on our end so that if one of our users get compromised, they do not result into being a vulnerability and keeping the growth of the security breach limited. Already existing users are not aware of the sheets on which data is being uploaded so it's not a security vulnerability.

In order to minimize DDoS attacks, we have programmed our GUI in such a way that only registered users are able to access our Database. We have allowed certain IP addresses a connection with our database as in order to decrypt the data while using our GUI the user must be registered in our database. Currently we have implemented the database on the Localhost but as soon as we launch our product in the market, we will be implementing a proper database and encrypting all the data stored in the database. Encryption of the data is the sole reason we have selected MariaDB as our database.

Now, we discuss our code that was used to develop the GUI. Tkinter was used and its reason is discussed earlier. In order to make a connection with MariaDB Database we used *mysql.connector* library. In order to extract the data from Google Sheets we used *gspread, pandas* and *oauth2client.service_account*. *Gspread* is a python API for Google Sheets as it was used to access the data from the sheets. *Pandas* was used analyzing and manipulating the data present in the Google Sheets. *oauth2client.service_account* was used for authentication of the user so that only an authenticated user is allowed to access the sheets that are to be used for decryption (Dalvi, n.d.).

In order to hash our passwords that were to be stored in our database, *cryptohash* was used. We imported *SHA256* from *cryptohash* and hashed our passwords. We used *numpy* library in order to work with the large multidimensional Google Sheets. We accessed columns of the data and decrypted that data by a set of predefined and hardcoded keys and initialization vectors used for decryption. *OS* was used in order to interact with operating system and use OS dependent functionality.

In the *register* function we are creating a registration window in which the user will enter the details which he wants to register to our database. A *350x200* window was created. *Login* function creates a login window, in that window the user will enter his/her personal details in order to access and view the decrypted data.

Register_user function will successfully register the user in our database and upon successful registration a message will be displayed reading *Record inserted successfully into accounts table.*

Login_verify will verify that whether the user is present in the database or not. If the user is present in the database, the decrypted data will be accessible to the user.

Decrypt function will access the data and decrypt the whole column based on predefined and hardcoded key alongside with initialization vectors. It must be noted that we are using PSK so that the key may not be leaked during the key exchange that will take place as the hacker can have an access to our packets, If we are not sharing the key in those packets then the attacker can never guess what the key is, only our registered users that are in possession of our GUI will have access to the key. In order to change the key the user will have to contact our team and only our team will be able to change the key.

Display_list will display the data present on Google Sheets. *Decrypt* button will display the decrypted data to the user.

In case the user enters a wrong password or is non existing in our database control will be shifted to *password_not_recognised* function and *Invalid Password* data is displayed on the screen, control is shifted to *user_not_found* in case of nonexistence of the user in our database.

In the *main_account_screen* as the name suggests the main screen of our GUI is created with a custom background and two buttons for Login and Register.

For references the code has been attached below.

```
#import modules



from tkinter import *

from tkinter import ttk

import mysql.connector
```

```
from mysql.connector import Error

import mariadb_dyncol

import csv

import tkinter as tk

import os

from string import punctuation

from cryptohash import sha256

import numpy as np

from itertools import product

from Crypto.Cipher import AES

import Crypto.Cipher.AES

from binascii import hexlify, unhexlify

#data extraction part

from oauth2client.service_account import ServiceAccountCredentials

import gspread

import pandas as pd



global dec_temp, dec_list



# Designing window for registration
```

```
def register():
    global register_screen
    register_screen = Toplevel(main_screen)
    register_screen.title("Register")
    register_screen.geometry("300x250")


    global username
    global password
    global username_entry
    global password_entry
    username = StringVar()
    password = StringVar()


    Label(register_screen, text="Please enter details below", bg="blue").pack()
    Label(register_screen, text="").pack()
    username_lable = Label(register_screen, text="Username * ")
    username_lable.pack()
    username_entry = Entry(register_screen, textvariable=username)
    username_entry.pack()
```

```
    password_lable = Label(register_screen, text="Password * ")

    password_lable.pack()

    password_entry = Entry(register_screen, textvariable=password, show='*')

    password_entry.pack()

    Label(register_screen, text="").pack()

    Button(register_screen, text="Register", width=10,

           height=1, bg="blue", command=register_user).pack()




# Designing window for login



def login():

    global login_screen

    login_screen = Toplevel(main_screen)

    login_screen.title("Login")

    login_screen.geometry("300x250")

    Label(login_screen, text="Please enter details below to login").pack()

    Label(login_screen, text="").pack()



    global username_verify
```

```
    global password_verify



    username_verify = StringVar()

    password_verify = StringVar()



    global username_login_entry

    global password_login_entry



    Label(login_screen, text="Username * ").pack()

    username_login_entry = Entry(login_screen, textvariable=username_verify)

    username_login_entry.pack()

    Label(login_screen, text="").pack()

    Label(login_screen, text="Password * ").pack()

    password_login_entry = Entry(

        login_screen, textvariable=password_verify, show='*')

    password_login_entry.pack()

    Label(login_screen, text="").pack()

    Button(login_screen, text="Login", width=10,

           height=1, command=login_verify).pack()
```

```
# Implementing event on register button




def register_user():



    username_info = username.get()

    password_info = password.get()

    password_validator(password_info)



    #establishing connection with database

    try:

        connection = mysql.connector.connect(host='localhost',

                                             database='shield',

                                             user='root',

                                             password='saifoo 1730')

        cursor = connection.cursor()



        mySql_insert_query = """INSERT INTO accounts (username, password) VALUES
(%s,%s) """



        recordTuple = (username_info, sha256(password_info))
```

```
        cursor.execute(mySql_insert_query, recordTuple)

        connection.commit()

        print(cursor.rowcount, "Record inserted successfully into accounts table")

        cursor.close()



    except mysql.connector.Error as error:

        print("Failed to insert record into accounts table {}".format(error))



    finally:

        if (connection.is_connected()):

            connection.close()

            print("MySQL connection is closed")



    username_entry.delete(0, END)

    password_entry.delete(0, END)



    Label(register_screen, text="Registration Success",

          fg="green", font=("calibri", 11)).pack()



# Implementing event on login button
```

```
def login_verify():
    username1 = username_verify.get()
    password1 = password_verify.get()
    username_login_entry.delete(0, END)
    password_login_entry.delete(0, END)

    connection = mysql.connector.connect(host='localhost',
                                         database='shield',
                                         user='root',
                                         password='saifoo 1730')
    cursor = connection.cursor()

    # PREPARED STATEMENT
    sql = """SELECT * FROM `accounts`
          WHERE `username` = %s AND `password` =%s
       """

    # EXECUTE WITH PARAMS
```

```
    cursor.execute(sql, (username1, sha256(password1)))



    rud = cursor.fetchall()



    if rud:



        login_sucess()

    else:

        password_not_recognised()




def decrypt_lists(dec_temp, dec_list):

   key = [0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30,

          0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30]

   s1 = [11, 69, 75, 1, 99, 8, 12, 8, 0, 7, 9, 111, 2, 3, 4, 71]




    IV = [hex(item) for item in s1]



    # Getting length of list
```

```
    length = len(list_1)

    i = 0



# Iterating using while loop

    while i < length:

        plaintext = (list_1[i])

        cipher = AES.new(key, AES.MODE_CBC, IV)

        ciphertext = unhexlify(cipher.encrypt(plaintext))

        decipher = AES.new(key, AES.MODE_CBC, IV)

        plaintext = decipher.decrypt(unhexlify(ciphertext))

        print("plaintxt: ", hexlify(plaintext))

        if (plaintext == plaintext1):

            print("successful decryption")

        plaintext = hexlify(plaintext)

        print("\n Plaintext : ", plaintext)



        i += 1

   # Designing popup for login success
```

```
def display_list(list_date, list_0, list_1):

    win = tk.Tk()

    win.resizable(width=0, height=0)

    global tv

    tv = ttk.Treeview(win, selectmode='browse')

    tv.pack(side='left')



    vsb = ttk.Scrollbar(win, orient="vertical", command=tv.yview)

    vsb.pack(side='right', fill='y')



    tv.configure(yscrollcommand=vsb.set)



    tv['columns'] = ('ID', 'Altitude', 'Temprature',)

    tv['show'] = 'headings'

    tv.heading("#1", text='ID', anchor='w')

    tv.column("#1", anchor="w", width=80)



    tv.heading("#2", text='Altitude')

    tv.column("#2", anchor='center', width=120)
```

```
    tv.heading("#3", text='Temprature')

    tv.column("#3", anchor='center', width=90)



    for row in range(len(list_0)):

        tv.insert('', 'end', values=(list_date[row], list_0[row], list_1[row]))



    root = Tk()

    root.geometry('50x50')

    btn = Button(root, text='DECRYPT', bd='5',

                 command=decrypt_lists)

    btn.pack(side='top')



    root.mainloop()

    win.mainloop()




def display_list_decrypted(list_date, list_0, list_1):

    win = tk.Tk()

    win.resizable(width=0, height=0)

    global tv
```

```
    tv = ttk.Treeview(win, selectmode='browse')

    tv.pack(side='left')



    vsb = ttk.Scrollbar(win, orient="vertical", command=tv.yview)

    vsb.pack(side='right', fill='y')



    tv.configure(yscrollcommand=vsb.set)



    tv['columns'] = ('ID', 'Altitude', 'Temprature',)

    tv['show'] = 'headings'

    tv.heading("#1", text='ID', anchor='w')

    tv.column("#1", anchor="w", width=80)



    tv.heading("#2", text='Altitude')

    tv.column("#2", anchor='center', width=120)



    tv.heading("#3", text='Temprature')

    tv.column("#3", anchor='center', width=90)



    for row in range(len(list_0)):
```

```
        tv.insert('', 'end', values=(list_date[row], list_0[row], list_1[row]))



    root = Tk()

    root.geometry('50x50')

    btn = Button(root, text='REFRESH', bd='5',

                 command=login_success)

    btn.pack(side='top')



    root.mainloop()

    win.mainloop()




def login_sucess():

    data_extraction()

    global login_success_screen

    login_success_screen = Toplevel(login_screen)

    login_success_screen.title("Success")

    login_success_screen.geometry("150x100")

    Label(login_success_screen, text="Login Success").pack()

    Button(login_success_screen, text="OK",
```

```
            command=delete_login_success).pack()




def data_extraction():

    #data extraction part

    scope = [

        'https://www.googleapis.com/auth/spreadsheets',

    ]



    GOOGLE_KEY_FILE = 'Shield-EncryptedSheet-1d5300762160.json'



    credentials = ServiceAccountCredentials.from_json_keyfile_name(

        GOOGLE_KEY_FILE, scope)

    gc = gspread.authorize(credentials)



    wokbook_key = '1uQYdeJwUUqQWQ8tK3srKDVlxAbCo96cvUocVvKcdPwg'

    workbook = gc.open_by_key(wokbook_key)

    sheet = workbook.get_worksheet(0)



    date = sheet.col_values(1)
```

```
    temperature = sheet.col_values(2)

    altitude = sheet.col_values(3)



    display_list(date, temperature, altitude)



    os.system('pause')



# Designing popup for login invalid password




def password_not_recognised():

    global password_not_recog_screen

    password_not_recog_screen = Toplevel(login_screen)

    password_not_recog_screen.title("Success")

    password_not_recog_screen.geometry("150x100")

    Label(password_not_recog_screen, text="Invalid Password ").pack()

    Button(password_not_recog_screen, text="OK",

           command=delete_password_not_recognised).pack()
```

```
# Designing popup for user not found



def user_not_found():

    global user_not_found_screen

    user_not_found_screen = Toplevel(login_screen)

    user_not_found_screen.title("Success")

    user_not_found_screen.geometry("150x100")

    Label(user_not_found_screen, text="User Not Found").pack()

    Button(user_not_found_screen, text="OK",

           command=delete_user_not_found_screen).pack()



# Deleting popups




def delete_login_success():

    login_success_screen.destroy()




def delete_password_not_recognised():

    password_not_recog_screen.destroy()
```

```
def delete_user_not_found_screen():
    user_not_found_screen.destroy()



# Designing Main(first) window


def main_account_screen():
    global main_screen
    main_screen = Tk()
    Height = (main_screen.winfo_height())
    Width = (main_screen.winfo_width())
    canvas = Canvas(main_screen, height=Height, width=Width)
    canvas.pack()


    background_image = PhotoImage(file='shield-bg1.png')
    background_label = Label(main_screen, image=background_image)
    background_label.place(relwidth=1, relheight=1)
```

```
    main_screen.title("Account Login")



    Label(text="").pack()

    Button(text="Login", height="2", width="30", command=login).pack()

    Label(text="").pack()

    Button(text="Register", height="2", width="30", command=register).pack()



    main_screen.mainloop()




main_account_screen()
```

A step by step working of the GUI is shown below.

1. When you open the GUI it shows two buttons with a login and register

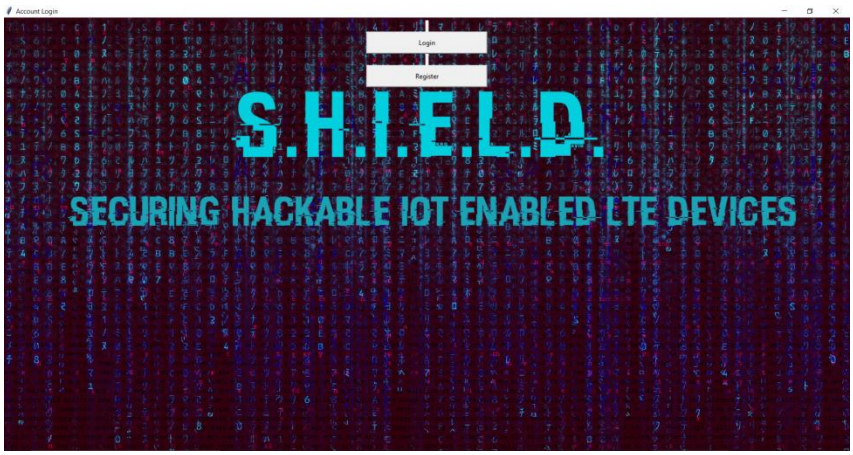

Figure 22: *GUI in function-1*

2. When you click on register it asks for your username and password

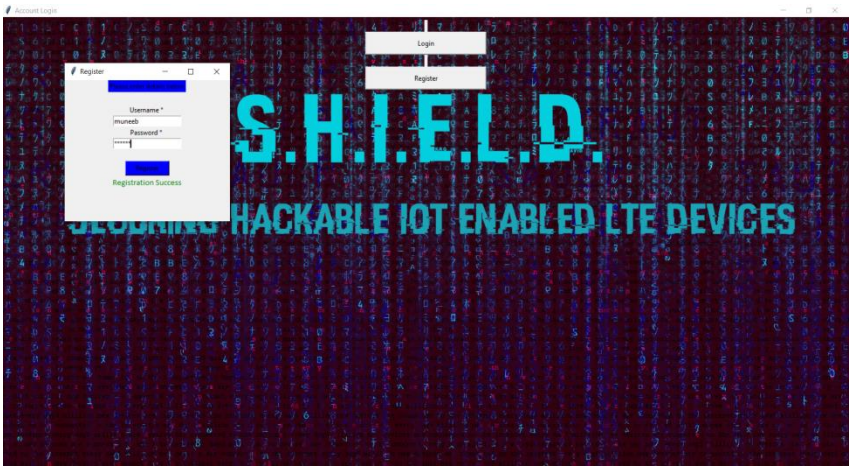

Figure 23:*GUI in function-2*

3. When you successfully register on the GUI your username and password is added to SQL database and the passwords are hashed and then stored in data base so even the database administrator does not have access to passwords.

shield.accounts: 6 rows total (approximately)

id	username	password
5	farooq	78f3147cd91db76f0b024f12cd0671c865d80617...
6	saif	f98c9d9340dc4bee35b4a18d79d0fba231d0d5b...
7	mir yasir	6d98614ec142aec10e6bd69b272bde40bc0717c...
8	imran	e378432fc68f9b707bd923e7968c1524c57e91ff...
9	lahore	feb81ca45117d135c1f6d86f24454e7527f5a50e...
10	muneeb	4a11836470651d13e1a29de644b6dbc2f8beada...

Figure 24:*MariaDB used in the project*

4. Now when you press login it will ask for your password and match it with the passwords in database. The hash of entered password is created and matched with the hash stored in database and it it authenticates then the data from the google sheet of this user is displayed.

Figure 25: *GUI in function-3*

5. Now the latest 10 rows of the google sheet are shown and as it can be seen it is in encrypted state. You can scroll through previous results if you want to see all the entries in the sheet.

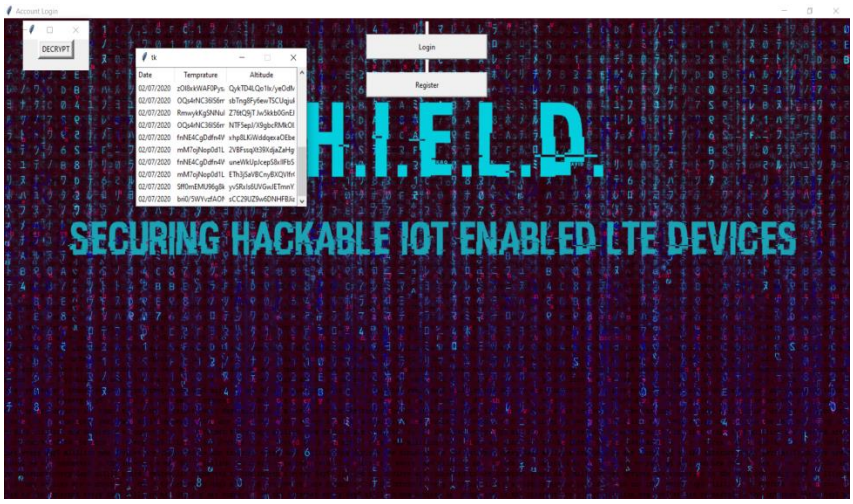

Figure 26:GUI in function-4

Now a button can be seen on the left top corner with “decrypt” written on it . When it is pressed the data of the user is decrypted and plain data can be seen in the picture below .

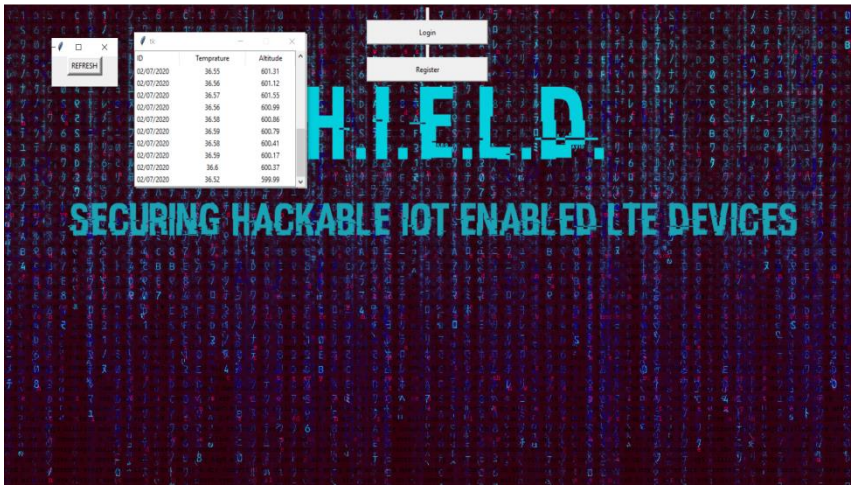

Figure 27:*GUI in function-5*

5.5. Printed Circuit Board Design

5.6Printed Circuit Board Design

5.6.1 Overview

PCB design is used to make a one-piece design of a circuit. It is used for professional design of motherboards. PCB core material is made of fiber glass. It can have from one to upto 10 layers . After that we make a mask of our circuit design and print it on a butter paper. This print out will be placed on the PCB board and we use FeCl3 for etching (to remove access copper) on the PCB board. At the end we place our circuit components according to the design on the PCB board and solder the circuit components. Final step is to test our PCB board we use DMM to check continuity of our cooper traces.

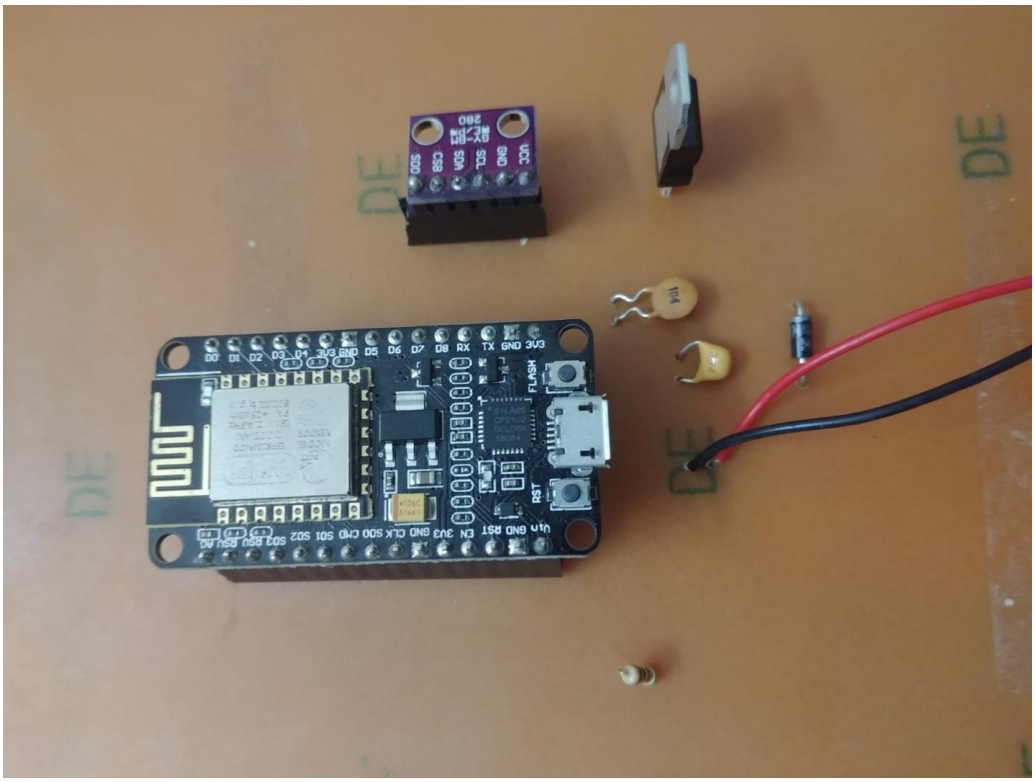

Figure 28: *SHIELD Hardware*

5.6.2 PCB Design Process

PCB designing process involves following steps:

5.6.2.1 Circuit Design

We make circuit design using our prior knowledge of Electronics Circuit Design. Our Hardware involves following components. SHIELD hardware operates at 9V input voltage. 5V output voltage is required for the output. We use 7805 regulator IC for voltage regulation. And 2 capacitors are used to stable the output voltage. ESP 8266 Wi-Fi module is used for Wi-Fi communication. BMP280 temperature and pressure sensor is used for temperature and pressure measurement.

5.6.2.2 PCB

After circuit design we simulate PCB using Proteus. Proteus does not have proper ESP package so we design ourselves 30 pins layout of ESP. BMP 280 sensor package is also not available in the Proteus we design BMP layout ourselves.

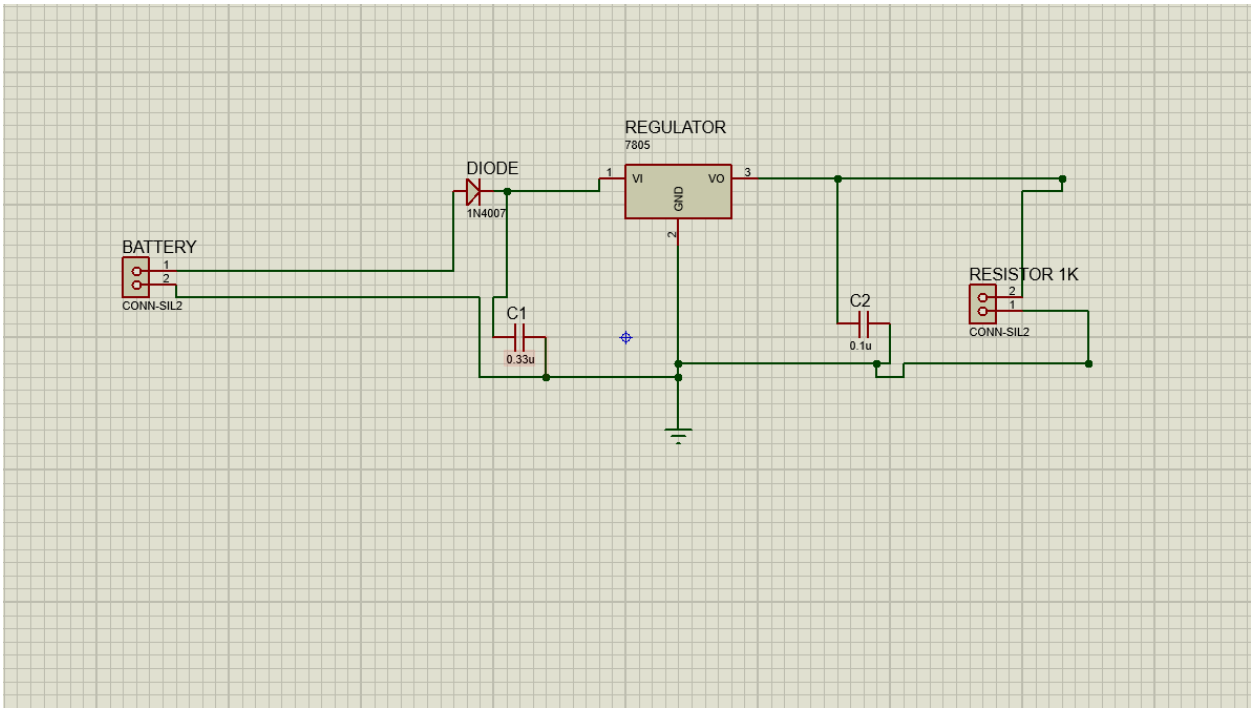

Figure 29: *Proteus Design*

5.6.2.3 3D Visualization

3d Visualization is the practice of using a computer to create imagery in order for the viewer to '**visualize**' a thing or things (usually a design concept) that does not yet exist.

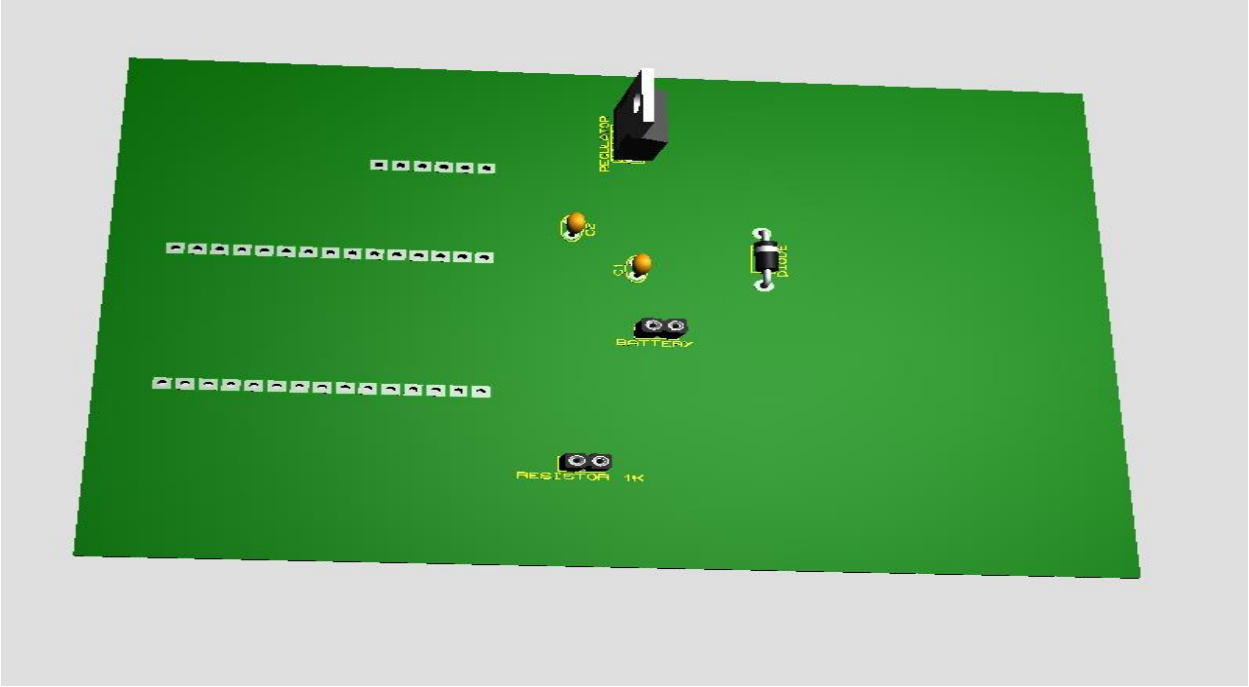

Figure 30: *3D Visualization of PCB(Top)*

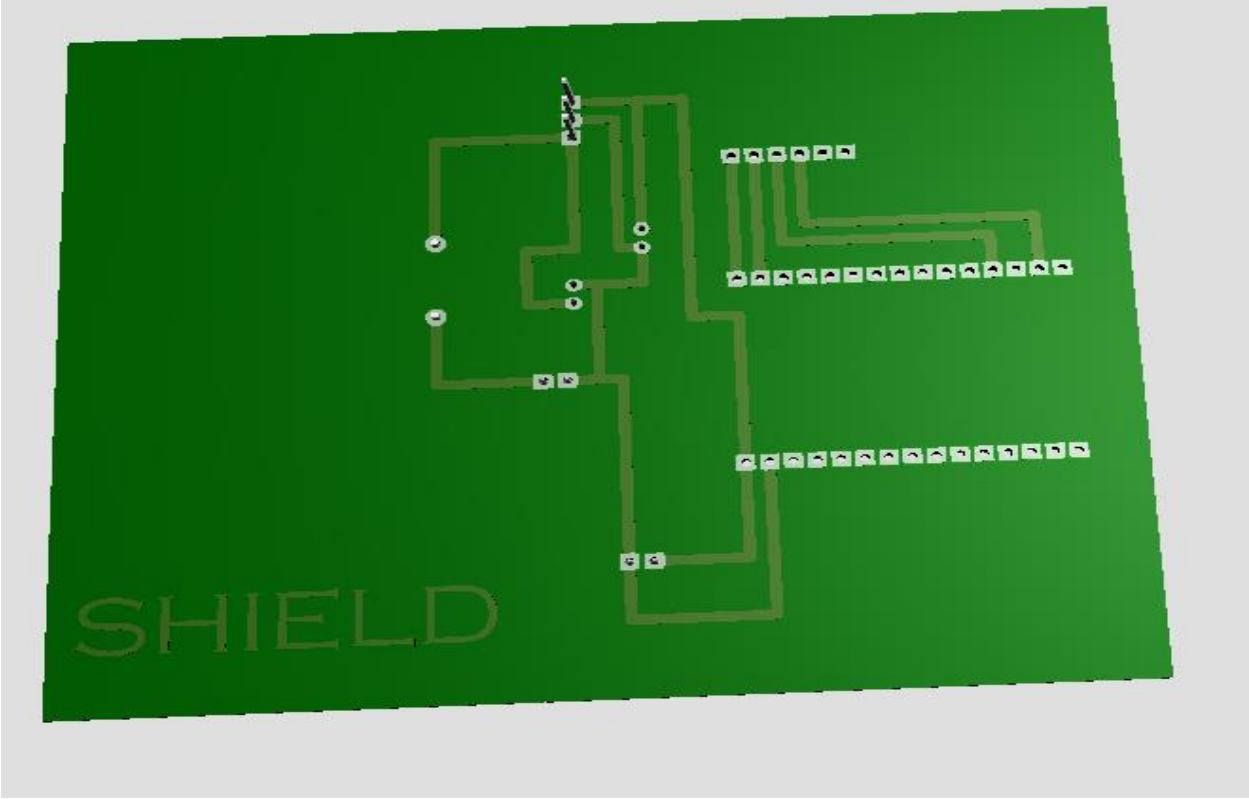

Figure 31:3D *Visualization of PCB (Bottom)*

5.6.2.4 Routing

Next, step is the routing of our design using Proteus. Routing is used to place components according our circuit requirements.

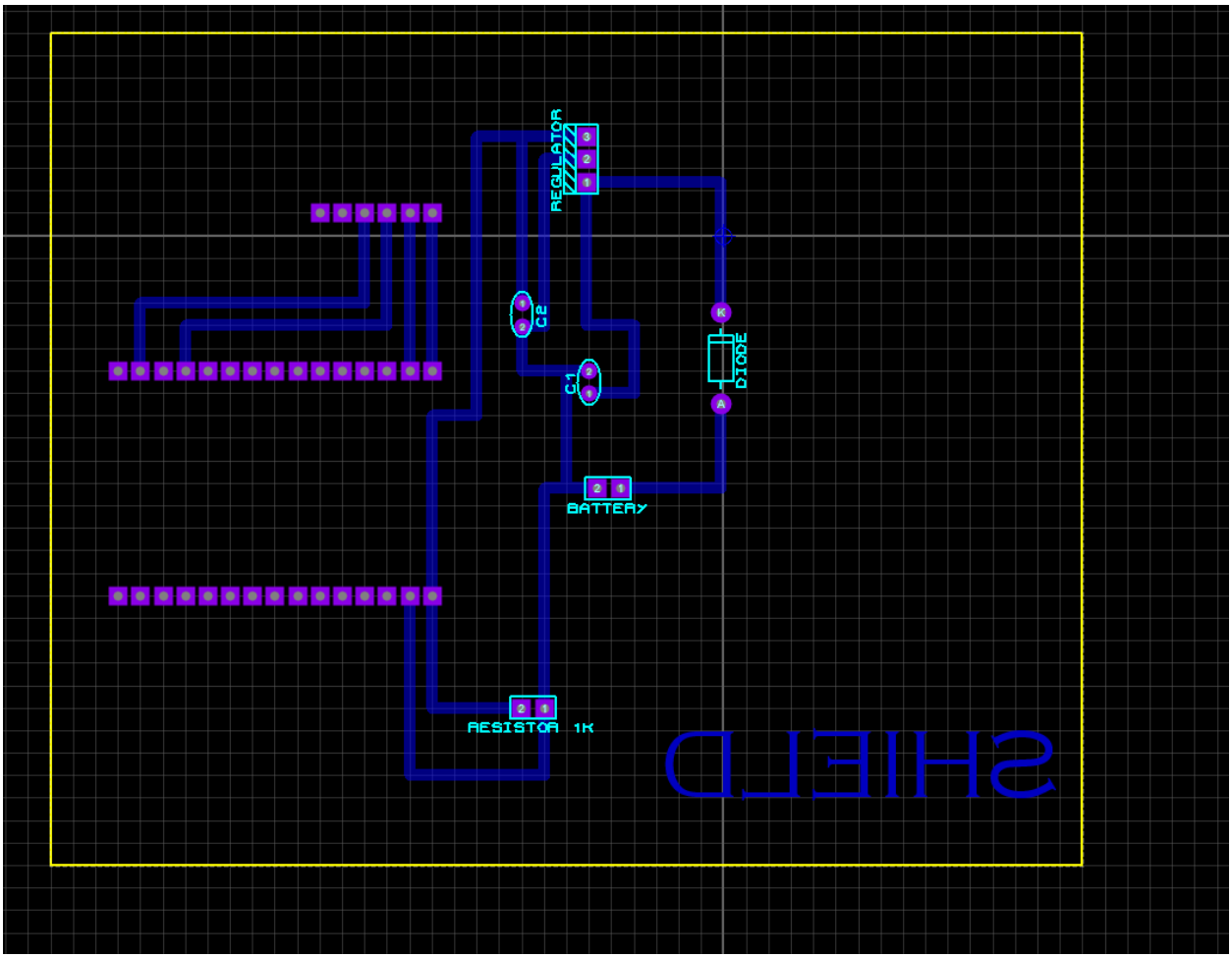

Figure 32: *PCB Routing*

5.6.2.5 Fabrication

After routing we performed fabrication. It has the following concepts:

1. First step is to take print out of routes on butter paper or chart.
2. Second step is to perform etching of the PCB.
3. Then we place components at their respective position and performed soldering.

5.6.2.6 Testing

Last step is to test our circuit board. DMM is used to test for connections of the board and finally, by supplying input voltage we test our design.

CHAPTER 6: FINISHING UP

CHAPTER 6: FINISHING UP

This chapter will focus on how SHIELD will affect the IoT market and what future improvements can be made to it.

6.1 Conclusion

It can be said with confidence that SHIELD will have a massive effect on the IoT market as no such device has been made in the past with focuses on the Security of the device.

SHIELD can be implemented from a single home to corporate buildings. In this modern era data is the most valuable asset of an individual or company. SHIELD was developed to insure that people can move towards automation without worrying about security.

6.2 Improvements for the Future

In the future SHIELD can be improved in every single field of automation that can exist. But some changes on the lower level that can be made in future are as follows:

6.2.1 Increase Access Interfaces

In the future we can increase the platforms on which we provide this service. To be exact many people use Linux and MacOS so we can create application for them too.

6.2.2 More Security

More authentication protocols can be added at the user end and a moveable data access device can be made so if user authenticates with his fingerprint or PIN, he can access data on that device.

APPENDICES

APPENDIX-A

SYNOPSIS

S.H.I.E.L.D(Securing Hackable IoT Enabled on LTE Devices)

Extended Title: Securing Hackable IoT Enabled on LTE Devices
Brief Description of the Project/ Thesis with Salient Specifications: IoT is a field of science that is growing exponentially and so is its demand all across the globe. The main problem with IoT is its vulnerabilities which can be used to exploit the network. Securing such vulnerabilities is to be achieved by using protection mechanisms and securing IoT that is being implemented on LTE.
Scope of Work: A secured link of IoT and LTE will provide assurance to the user about their privacy as the users' data will be secured via protection mechanisms. Through this project we wish to integrate our theoretical knowledge with practical experience to gain further insight and redefine our skills.
Academic Objectives: The project will give hands-on experience of the implementation of the courses studied so far like: • Computer and Communication Networks • Network Security • Microprocessor and Systems • Programming Techniques
Application/ End Goal Objectives: For any IoT application, it is crucial to protect data, user privacy, and safety. Breaches can harm production continuity and business processes, customers trust, and worst of all human health and life. If you are an IoT platform provider, you must provide your business-to-business (B2B) customers a secure infrastructure for the same reasons. Providing this security to the users is our main objective.
Previous Work Done on the Subject: The security of IoT has already been implemented on DSL based networks. We are aiming to secure the connection between IoT network and Cloud based on LTE to make it mobile and no such work has been done in the past.
Material Resources Required: The system will work based on a 32-bit microcontroller and controller-based Wi-Fi modules. Some common software environments will be used to test the vulnerability of our network.
Number of Students Required: 4 **Group Members:** • NC Abdul Samad • NC Muhammad Abdullah Farooq • NC Saif Ur Rehman • NC Umar Ahmad Khan
Special Skills Required: • Algorithm Analysis • Cloud Computing • Network Security • Programming

APPENDIX-B

Abbreviations List

AES	Advanced Encryption System
I2C	Inter Integrated Circuit
API	Application Programming Interface
IV	Initialization Vector
Wi-Fi	Wireless Fidelity
IoT	Internet of Things
IDE	Integrated Development Environment
PCB	Printed Circuit Board
GUI	Graphical User Interface
SHA	Secure Hash Algorithm

References

(n.d.). Retrieved from https://www.espressif.com/en/products/socs/esp8266/overview

(n.d.). Retrieved from https://www.python.org/about/

(n.d.). Retrieved from https://visualstudio.microsoft.com/vs/

(n.d.). Retrieved from https://developer.android.com/studio/preview

(n.d.). Retrieved from https://www.arduino.cc/en/main/software

(n.d.). Retrieved from https://www.labcenter.com/Proteus

(n.d.). Retrieved from https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT#:~:text=The%20internet%20of%20things%2C%20or,human%2Dto%2Dcomputer%20interaction.

(n.d.). Retrieved from https://www.adafruit.com/product/2651 bmp 280

(n.d.). Retrieved from https://www.electronicsforu.com/resources/learn-electronics/7805-ic-voltage-regulator

(n.d.). Retrieved from https://en.wikipedia.org/wiki/Advanced_Encryption_Standard

(n.d.). Retrieved from https://www.google.com/script/start/

(n.d.). Retrieved from https://www.google.com/sheets/about/

(n.d.). Retrieved from https://developers.google.com/sheets/api

(n.d.). Retrieved from https://en.wikipedia.org/wiki/Initialization_vector

(n.d.). Retrieved from https://developers.google.com/android/guides/client-auth

(n.d.). Retrieved from https://www.youtube.com/watch?v=7DZR5UaAM0E

(n.d.). Retrieved from https://www.youtube.com/watch?v=LlhmzVL5bm8

(n.d.). Retrieved from https://iiot-world.com/ics-security/cybersecurity/main-challenges-of-implementing-iot-security-standards/

(n.d.). Retrieved from https://www.zdnet.com/article/what-is-the-internet-of-things-everything-you-need-to-know-about-the-iot-right-now/

(n.d.). Retrieved from https://www.hindawi.com/journals/jece/2017/9324035/

Dalvi, P. (n.d.). Retrieved from https://medium.com/analytics-vidhya/how-to-read-and-write-data-to-google-spreadsheet-using-python-ebf54d51a72c

Parr, J. (n.d.). Retrieved from http://justinparrtech.com/JustinParr-Tech/the-importance-of-hashing-passwords/#:~:text=6.2.-,Advantages,A%20hash%20code%20is%20useless!