Internet of Things Network to inhibit power theft in power distribution network



**Submitted By**

Muhammad Hamza

Submitted in partial fulfillment of the requirement for the degree of Bachelor of Science in Electronics

Department of Electronics Quaid-i-Azam University, Islamabad.

Certificate

This is to certify that this thesis submitted by Muhammad Rizwan and Muhammad Hamza is accepted in its present form by the department of Electronics Quaid-i-Azam University, Islamabad for the award degree of Bachelor of Science in Electronics (BS-Electronics). The project work is supervised by Dr. Muhammad Zia.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dr. Muhammad Zia.

Department of Electronics,

Quaid-i-Azam University, Islamabad.

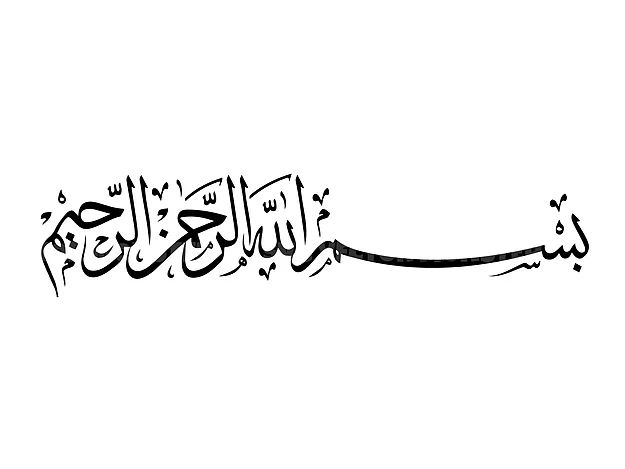
Submitted through:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dr. Qaisar Abbas Naqvi

Chairman Department of Electronics,

Quaid-i-Azam University, Islamabad.



ACKNOWLEDGMENT

First and foremost, we thank Allah Almighty the most merciful and the most gracious, who showered his blessings upon us throughout our lives. We pay our sincere gratitude to our supervisor Dr. Muhammad Zia. He has been a source of motivation throughout this degree and most importantly, during our final year project. He has always been an excellent leader and mentor. His encouragement made it possible to achieve the goal. His help and sympathetic attitude at every point during this project helped me to meet the deadline. We thank our university which is providing such a competitive platform to show our skills. Above all, we would like to thank our parents for struggling for us and bringing us up to this stage. Without their tireless efforts, endeavors supplicate, and praying, we would not be able to be at this stage of life. Finally, we pay a hat tip to each dignitary who helped us straightforwardly and in a roundabout way through this expedition.

Muhammad Rizwan

--------------------------

Muhammad Hamza

Abstract

Electricity theft is the major problem in Pakistan and technical solution does not exist to curb the power theft. Many projects have been initiated to eliminate the power theft and none has been matured to the marketable product. This project is a step in the right direction to provide solution for the power theft by integrating off the shelf electronics modules. The proposed design uses non-invasive current sensors interfaced with microcontroller to measure the flow of current. Thus, multiple current sensors interface with smart devices connected wirelessly for a network of internet of things (IoT). In the rural are, where WiFi network is not available, data from the current sensor is relayed using long range (LoRa) wireless devices. Data from the sensor network is stored at the cloud and accessed for the analysis in real-time fashion by mobile application. The project comprises of four components. They are sensor nodes, server, data base and application.

Contents

[Chapter 1 1](#_Toc111763678)

[Introduction 1](#_Toc111763679)

[1.1 Power distribution system and thieving mechanisms 2](#_Toc111763680)

[1.2 Internet of Things 4](#_Toc111763681)

[1.3 Communication 6](#_Toc111763682)

[1.3.1 Serial Peripheral Interface 7](#_Toc111763683)

[1.4 Wireless communication 8](#_Toc111763684)

[1.4.1 5G 9](#_Toc111763685)

[1.4.2 Star Network Topology 10](#_Toc111763686)

[1.5 Sensor 11](#_Toc111763687)

[1.5.1 Calibration 11](#_Toc111763688)

[1.6 Analog to digital converter (ADC) 11](#_Toc111763689)

[1.7 Microcontrollers 12](#_Toc111763690)

[Chapter 2 13](#_Toc111763691)

[Hardware 13](#_Toc111763692)

[2.1 Current Sensor 13](#_Toc111763693)

[2.1.1 Magnetic current sensor 13](#_Toc111763694)

[2.2 LoRa 14](#_Toc111763695)

[2.2.1 LoRa Spread Spectrum 14](#_Toc111763696)

[2.2.2 Network Trial 15](#_Toc111763697)

[2.2.3 Link Budget 15](#_Toc111763698)

[2.3 Microcontrollers 16](#_Toc111763699)

[2.3.1 Features [7] 16](#_Toc111763700)

[2.3.2 Pin Diagram 17](#_Toc111763701)

[Chapter 3 18](#_Toc111763702)

[Software 18](#_Toc111763703)

[3.1 C++/C – Programming language 18](#_Toc111763704)

[3.1.1 Applications 18](#_Toc111763705)

[3.2 Arduino IDE – Hardware Programming tool 18](#_Toc111763706)

[3.2.1 Programming Language 19](#_Toc111763707)

[3.2.2 Uploading 19](#_Toc111763708)

[3.2.3 Serial Monitor 20](#_Toc111763709)

[3.3 JavaScript – Web development Language 20](#_Toc111763710)

[3.3.1 JavaScript Development Tool 20](#_Toc111763711)

[Chapter 4 22](#_Toc111763712)

[Working Mechanism 22](#_Toc111763713)

[4.1 System Model 22](#_Toc111763714)

[4.2 Node 23](#_Toc111763715)

[4.2.1 Sensor-Controller Interfacing 23](#_Toc111763716)

[4.2.2 Controller Calculation 24](#_Toc111763717)

[4.2.3 Controller-Transmitter Interfacing 25](#_Toc111763718)

[4.3 Gateway/Server 26](#_Toc111763719)

[4.3.1 Controller-Receiver Interfacing 26](#_Toc111763720)

[4.3.2 Uploading 27](#_Toc111763721)

[4.4 Database 27](#_Toc111763722)

[4.5 Application (App) Designing 29](#_Toc111763723)

[4.6 Self-made System 30](#_Toc111763724)

[Results 32](#_Toc111763725)

[Advancements to the system 33](#_Toc111763726)

[Cost Analysis 34](#_Toc111763727)

[References 35](#_Toc111763728)

[Appendix A 37](#_Toc111763729)

[Appendix B 39](#_Toc111763730)

[Appendix C 43](#_Toc111763731)

[Appendix D 44](#_Toc111763732)

[Figure 1‑1 World economies losses billions of dollars per year by electricity theft 1](#_Toc111767007)

[Figure 1‑2 Power line losses in Pakistan 2](#_Toc111767008)

[Figure 1‑3 Electricity distribution network 3](#_Toc111767009)

[Figure 1‑4 Meter Bypassing 3](#_Toc111767010)

[Figure 1‑5 Hooking 4](#_Toc111767011)

[Figure 1‑6 Connected IoT Networks 5](#_Toc111767012)

[Figure 1‑7 Global distribution of IoT projects among America (USA, South America, and Canada), Europe, and APAC (Asia and Pacific region) 5](#_Toc111767013)

[Figure 1‑8 Global share of IoT projects across the world 6](#_Toc111767014)

[Figure 1‑9 Devices using 4 wire SPI 7](#_Toc111767015)

[Figure 1‑10 Wireless Communication system 8](#_Toc111767016)

[Figure 1‑11 5G-Wireless Communication Capabilities 9](#_Toc111767017)

[Figure 1‑12 Star Network Topology 10](#_Toc111767018)

[Figure 2‑1 Diagram of a Coreless Current Sensor 12](file:///C:\Users\rizwa\Downloads\BS-FYP%20Report-FinalZiaAug19.docx#_Toc111767019)

[Figure 2-2 Comparison between FSK and LoRa range and accuracy 14](#_Toc111767020)

[Figure 2‑3 Budget comparison for narrow band FSK and LoRa 15](#_Toc111767021)

[Figure 2‑4 diagram of ESP-32 16](file:///C:\Users\rizwa\Downloads\BS-FYP%20Report-FinalZiaAug19.docx#_Toc111767022)

[Figure 4‑1 Complete model of the project 21](#_Toc111767023)

[Figure 4‑2 Project Flowchart 22](#_Toc111767024)

[Figure 4‑3 Sensor-Controller interfacing 22](#_Toc111767025)

[Figure 4‑4 ESP-32 LoRa Interfacing 25](#_Toc111767026)

[Figure 4‑5 Server Database 27](#_Toc111767027)

[Figure 4‑6 App Database 27](#_Toc111767028)

[Figure 4‑7 a: Main Screen b: Login Screen c: Signup Screen 28](file:///C:\Users\rizwa\Downloads\BS-FYP%20Report-FinalZiaAug19.docx#_Toc111767029)

[Figure 4‑8 Self-made System 29](#_Toc111767030)

# Chapter 1

# Introduction

The recent report reveals that the world economy bears about $30 billion in electricity theft [1]. Figure 1-1 provides details of electricity theft in various countries. Employees of the electricity providers and consumers are the main culprits of power theft. Due to the power theft, power providers bear huge loss, which is eventually transferred to the customers. Consequently, average electricity cost increase. There is dire need to curtail uncontrolled power theft by new legislation and enforcement. Furthermore, available IoT technology can play pivotal role to detect the electricity theft in real-time fashion. In this project, we propose an IoT based solution in order to detect power theft and generate alert in real-time manner.

Instead of bringing culprits to the justice, power companies add power theft to the line-losses. As a result, Pakistan bears high line-losses up to 26%. The line-losses faced by various energy providers is shown in figure 1-2.

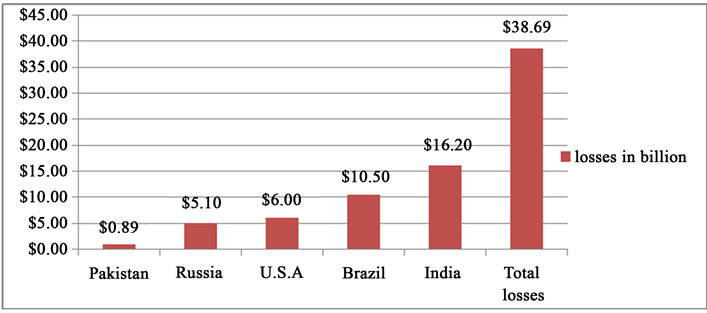


Figure 1‑1 World economies losses billions of dollars per year by electricity theft

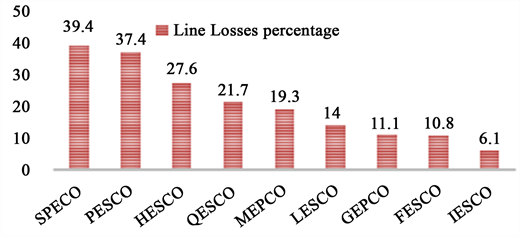


Figure 1‑2 Power line losses in Pakistan

## 1.1 Power distribution system and thieving mechanisms

The two main kinds of power theft techniques are meter tempering and power line tempering. In meter tempering, working principle of power meters is manipulated. In line tempering, connection to the electricity meter is by passed or by local earth. These types of power theft are widespread and easy for a lay man to use. As a result of electricity theft occurring at various levels, the nation experiences economic losses and electricity theft concerns. An electricity distribution model is presented in figure 1-3. Power theft methods are described in figure 1-4 and figure 1-5. IoT technology can be useful to detect and monitor power theft remotely and evidence of power theft can be recorded in cloud for forensic analysis.

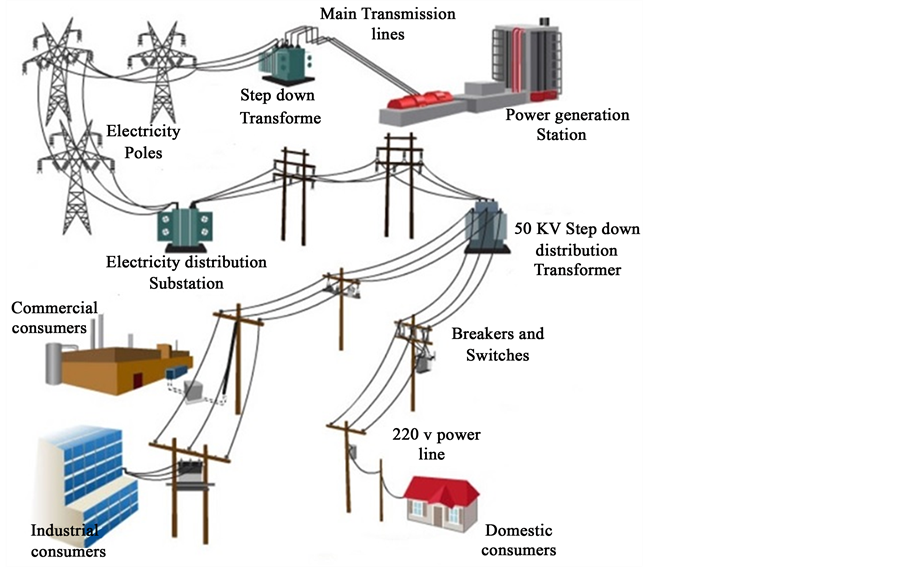


Figure 1‑3 Electricity distribution network

* Bypassing Electricity monitor.

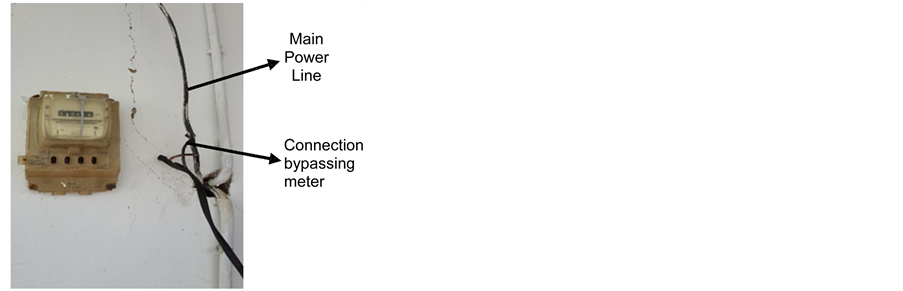


Figure 1‑4 Meter Bypassing

* Hooking Load Lines directly.

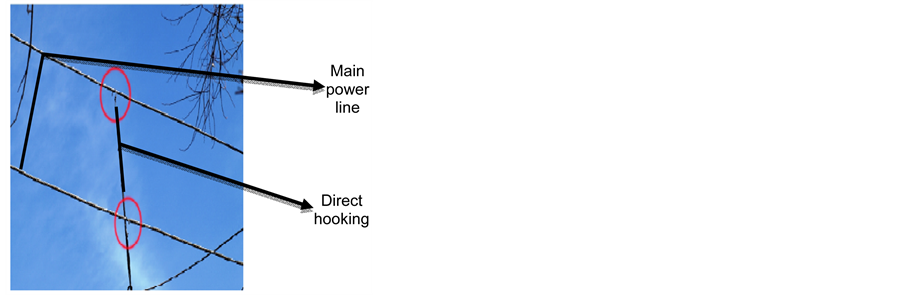


Figure 1‑5 Hooking

## 1.2 Internet of Things

IoT is the communication between devices (things) comprised of sensors, software, controllers, and machines connected over the internet as described in figure 1-6. The advent in this field has automated our homes and has also taken its place in the industrial world, especially in industry revolution 4. Industrial IoT refers to machine-to-machine communication to achieve wireless control and automation. This industrial IoT is referred as the fourth wave of the industrial revolution. The most common uses of industrial revolution 4 are as follows:

* Smart Industrial manufacturing
* Smart power grid stations
* Smart cities
* Digital supply chains

IoT has taken over the market in different forms and many countries are expressing a great deal of investment in the field. Figure 1-7 provides the data about the fact [2].

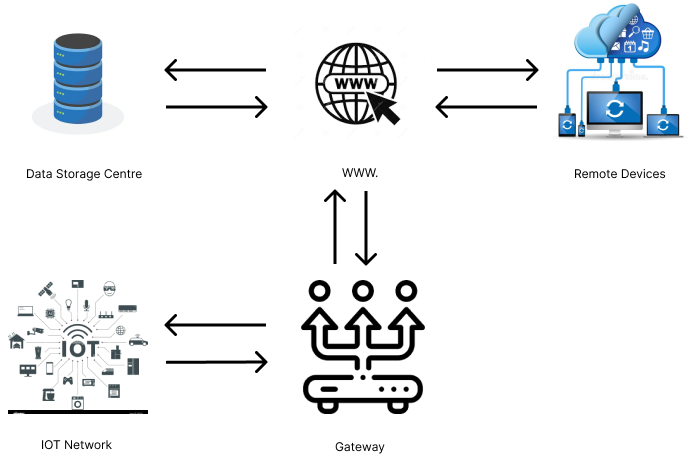


Figure 1‑6 Connected IoT Networks

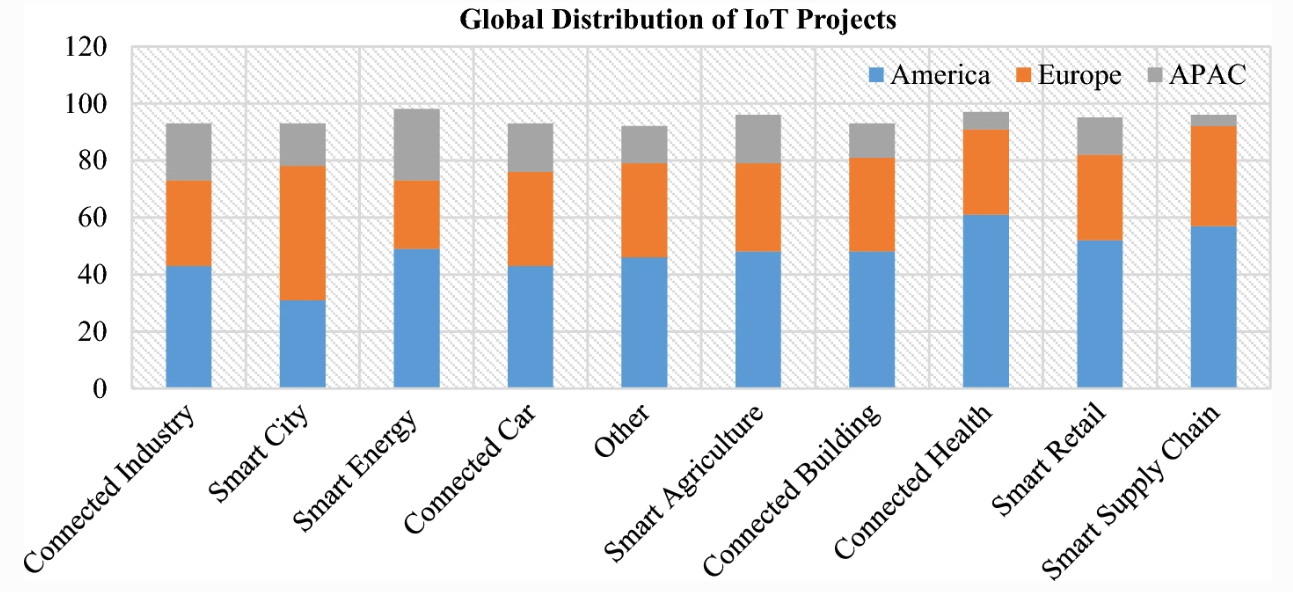


Figure 1‑7 Global distribution of IoT projects among America (USA, South America, and Canada), Europe, and APAC (Asia and Pacific region)

Figure 1-8 depicts the worldwide share in the IoT field.

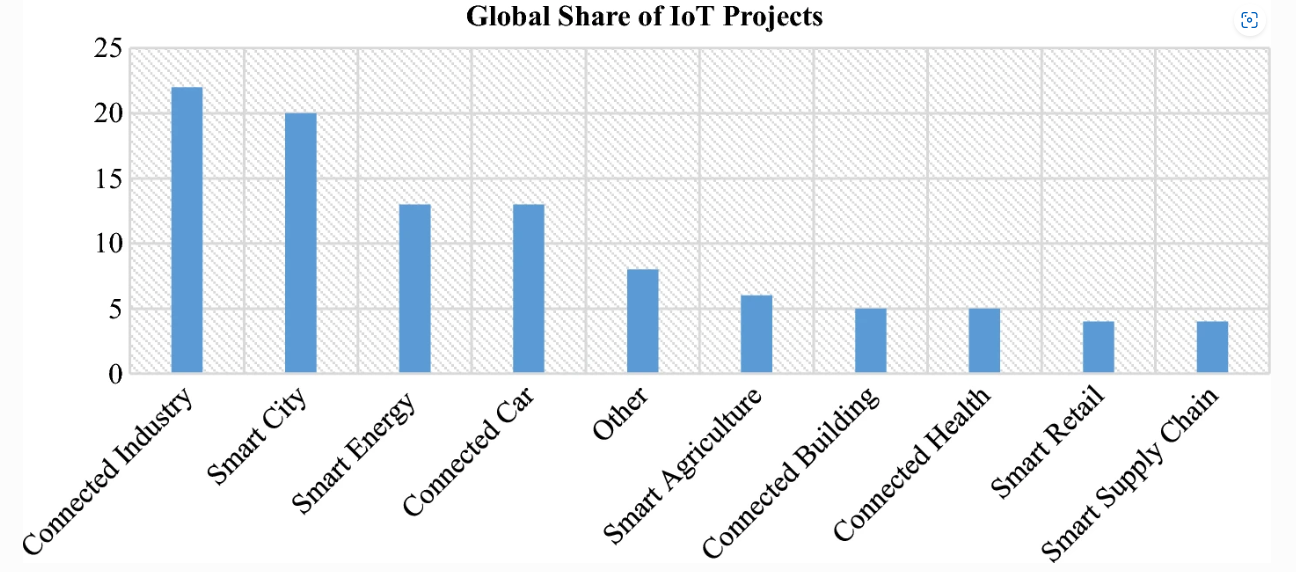


Figure 1‑8 Global share of IoT projects across the world

## 1.3 Communication

Communication is the exchange of information between devices called transmitters and receivers. Based on transceiver types, a communication system can be classified as follows:

* Simplex (transmitting device to the receiving device communication)
* half duplex (a device can be receiver and transmitter at a time)
* Full duplex (a device being transmitter and receiver at the same time)

Based on medium, communication systems can be categorized as

* Guided medium (wired channel)
* Unguided medium (wireless channel)

Billions of communication systems are connected over wireless medium from body area network to the mobile cellular network. The base stations are connected to the main switches over wired back-haul and front-haul network. The future 5G and 6G wireless technology supports IoT devices.

Communicating devices follow specified protocols for communication such as I2C and SPI (Serial Peripheral Interface). We elaborate SPI in this report, which is used in the project. In a communication network, data is relayed to other devices using a network topology such as mesh topology and star topology.

### 1.3.1 Serial Peripheral Interface

One of the most well-known and popular interfaces between microcontrollers and peripheral devices is the serial peripheral interface (SPI). SPI is a controller-node interface, which uses synchronous data exchange in full-duplex communication mode. Data synchronization between the transmitter and receiver is based upon the rising or the falling edges of the reference clock signal [3]. Both controller and node can simultaneously exchange their data. Note that 3-wire and 4-wire SPI interface are currently in use. A 4 wire SPI transceiver is shown in figure 1-9.



Figure 1‑9 Devices using 4 wire SPI

* Clock (SPI CLK, SCLK)
* Chip selection (CS)
* Controller-out, node-in (MOSI)
* Controller-in, node-out (MISO)

#### Data Transmission

In SPI protocol the controller first enables the CS signal, sends a clock signal, and chooses the node. To select the node, the controller is made to send a logic 0 on the CS bus, which is an active low signal. MISO and MOSI buses are used for full-duplex data communication. Data exchange happens simultaneously through buses MOSI/SDO as output and MISO/SDI as input. The data exchange is synchronized by the clock signal [3].

## 1.4 Wireless communication

The unguided medium does not involve any physical connection between communicating devices. Since the transmitting device spreads the signal, the unguided signal interacts with the physical objects, which leads to distortion and ultimately receiver encounters complex impulse responses of the channel. The impulse response affects the information encoded in the signal and the range of the transmitting device.

A wireless communication system can be categorized based on its range.

* Short range (Wi-Fi, Bluetooth, ZigBee, WWAN)
* Long range (LoRa, Sigfox, LTE-M, 5G)

The range of the devices can also be controlled by the transmitting device by changing the transmit power within the allowed power upper limit. The transmission power has established an inverse relationship between distance between the nodes and the data rate.

In wireless communication, to avoid any error caused by the channel response, a signal is passed through several different encoding systems working under assigned protocols to design a wirelessly transmittable signal. The complexity has been illustrated in figure 1-10. Since wireless communication plays an important role in IoT and a small amount of delay can cause lag in controlling and communication cannot be real-time. Thus, to cope with the problem certain advancements in the field have happened over the years. Thus, a communication through 5G network is the faster than predecessors.

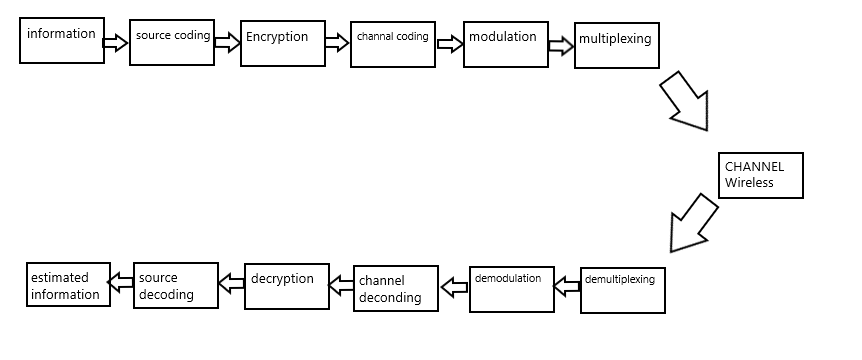


Figure 1‑10 Wireless Communication system

### 1.4.1 5G

With the advancement of wireless communication technology, the 5G standard has been developed standards, which has high data rate. The fifth generation of technology can connect every gadget, object machine, mobile, and computer in a single new type of network. The technology provides fast data rates, can compensate more end users and industries and has low latency and a better user experience.

The 5G network has a greater capacity to support new deployment strategies, enable new kinds of online experiences, and provide novel capabilities. Due to the fast speeds of 5G, high reliability, and low latency, the mobile system will enter into new domains. The new technology will affect every business bringing forth innovations in transportation, healthcare, farming, and more. Figure 1-11 depicts the 5G capabilities.

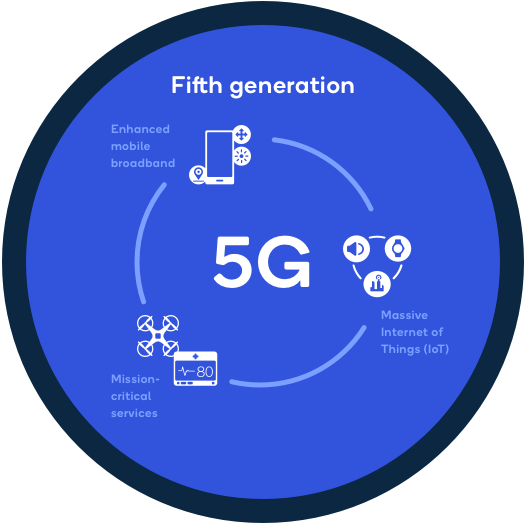


Figure 1‑11 5G-Wireless Communication Capabilities

### 1.4.2 Star Network Topology

The star network is the most popular and straightforward network topology. The central coordinator serves as the gateway for all network broadcasts as shown in figure 1-12. Network traffic is reduced with the use of star-network architecture. For the simplest communication in a star network, one requires 3 devices including 2 nodes and 1 gateway in a network with unrestricted connectivity. Nodes are also segregated from one another, which makes it simple to replace them. All network traffic may be inspected at one location thanks to centralization. The failure of the gateway causes the whole network non-functional which is a major drawback of this design.

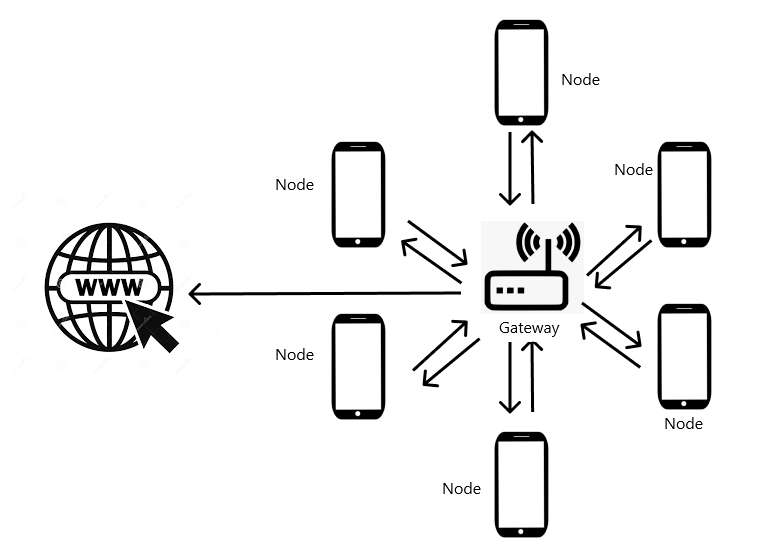


Figure 1‑12 Star Network Topology

## 1.5 Sensor

A sensor is a device responding to a physical stimulus such as heat, pressure, current, and voltage. All the electrical sensors respond in the form of voltage levels as output. The sensor reads the stimulus, generates an impulse response, and converts it into a voltage value which is fed into the voltmeter.

### 1.5.1 Calibration

Sensor establishes the arithmetic relation between two measuring units and calibration understands the arithmetic relation between the sensor outputs (voltage) to the physical stimulus values. This relation is sometimes provided in the sensor guide but some error remains still and some errors are introduced when the sensor is interfaced with the voltmeter, so eliminating those error also counts as calibration.

## 1.6 Analog to digital converter (ADC)

As the name specifies, ADC converts the voltage signal into digital values. It is a chip operating and for its functionality. ADC is supplied with a specified voltage value and the reference voltage. The conversion is based on the comparison between the reference voltage and the input voltage. The accuracy of ADC is dependent on the quantization levels and the difference between the maximum measurable voltage value and the minimum measurable voltage value called the measurable voltage. Quantization is slicing and leveling the measurable voltage and specifying a digital value to each level, the higher the quantization, the higher the accuracy.

## 1.7 Microcontrollers

The microcontroller is an integrated circuit designed to implement the specified task. It is comprised of a CPU, memory, and input/output peripherals. The basic operation a microcontroller follows is the data collected from the input peripherals, passing it to the processor. The processor performs the specified task on the data and then relays the processed data to the output peripherals.

The choice of the microcontroller's processor is based on the required functionality. The industry provides different options in terms of processors ranging from the simple 4-bit, 8-bit or 16-bit processors to more sophisticated 32-bit or 64-bit processors and their choice depends upon the project. Microcontrollers usually use non-volatile memory types such as programmable read-only memory (EPROM), [flash memory](https://www.techtarget.com/searchstorage/definition/flash-memory), electrically erasable programmable read-only memory (EEPROM), and some volatile memory types such as random access memory ([RAM](https://www.techtarget.com/searchstorage/definition/RAM-random-access-memory)) [4].

# Chapter 2

# Hardware

## 2.1 Current Sensor

We use non-invasive current sensor to measure the current at distribution point and also measure the return current. The technique used for theft detection is to measure the current flowing in the load lines because it’s the only factor of electrical-energy consumption controllable by the consumer. Current sensors are available off the today use a different technique for measurement, have different ranges depending on the technique used and provide accuracy dependent on the range and technique used but the suitable option for the project is the use of the current transformer since it’s a non-invasive device and can measure high current values.

### 2.1.1 Magnetic current sensor

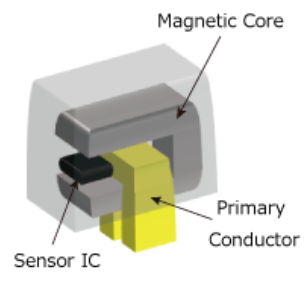
 The magnetic current sensor works on the basis that the current to be measured creates a magnetic field around the current path, which can then be detected by a sensor to determine the current's amount as shown in figure 2-1. The magnetic current sensor can be isolated internally [5]. That means there is no open circuitry that can create shocks. Certain drawbacks are associated with the use of current transformers as:

Figure 2‑1 Diagram of a Coreless Current Sensor

* The magnetic core potentially possesses hysteresis.
* The space is challenging because of the core (particularly because of its height).

## 2.2 LoRa

LoRa stands for "long range," which defines the physical layer of the communication network. It is based upon chirp spread spectrum technology (CSS). Lora was developed by Cycleo (patent 9647718-B2), a Grenoble, France-based business [6].

The internet architecture and communication protocol are defined by LoRaWAN. LoRa and LoRaWAN together form their new network having the capability of Low Power, Wide Area Network (LPWAN) with a networking protocol that focuses on Internet of Things (IoT) requirements like duplex communication, security, mobility, and other localization services. On the other hand, WAN (Wide area network) is intended to connect people or companies and allows the transmission of much more data than LoRa, while consuming more power. But LoRa is low power, low bit rate, and IoT enabled. The LoRaWAN data rate per channel is between 0.3 and 50 kbit/s.

### 2.2.1 LoRa Spread Spectrum

LoRa uses its chirp spread spectrum technique. It works by creating a chirp signal that continuously changes in frequency, LoRa modulation spreads the spectrum. The chirp's frequency bandwidth is equal to the signal's spectral bandwidth. The chirp signal is firstly modulated with the data signal to be sent after which it is chipped to a higher data rate [6]. We can define the modulation bit rate, Rb, as:

where:

* SF = spreading factor (7….to….12)
* BW = modulation bandwidth (Hz)

### 2.2.2 Network Trial

A data table along with the street-view picture is shown in figure 2-2 where data collected in Japan Shinjuku is used as an illustration of the of LoRa modulation performance relative to narrowband FSK [6].

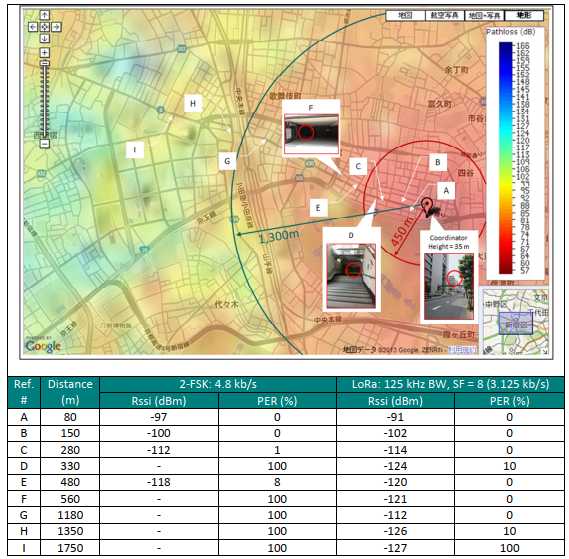


Figure 2-2 Comparison between FSK and LoRa range and accuracy

The LoRa and a narrowband FSK are both configured to transmit a small amount of payload at an output power of +13 dBm. It can be seen from the figure that even in populated areas the LoRa range is 3 times greater than the narrowband FSK.

### 2.2.3 Link Budget

The link budget in delta, shown in figure 2-3 below for a fixed transmitter output power and the comparison between the link budgets that may be obtained for each modulation scheme i.e., LoRa and narrowband FSK.

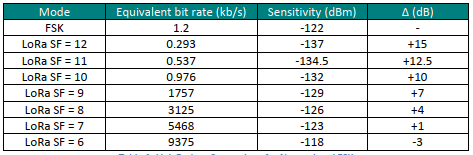


Figure 2‑3 Budget comparison for narrow band FSK and LoRa

It can be seen that LoRa modulation provides relatively comparable sensitivity to a traditional FSK while transmitting at more than 4 times greater data rate [6].

## 2.3 Microcontrollers

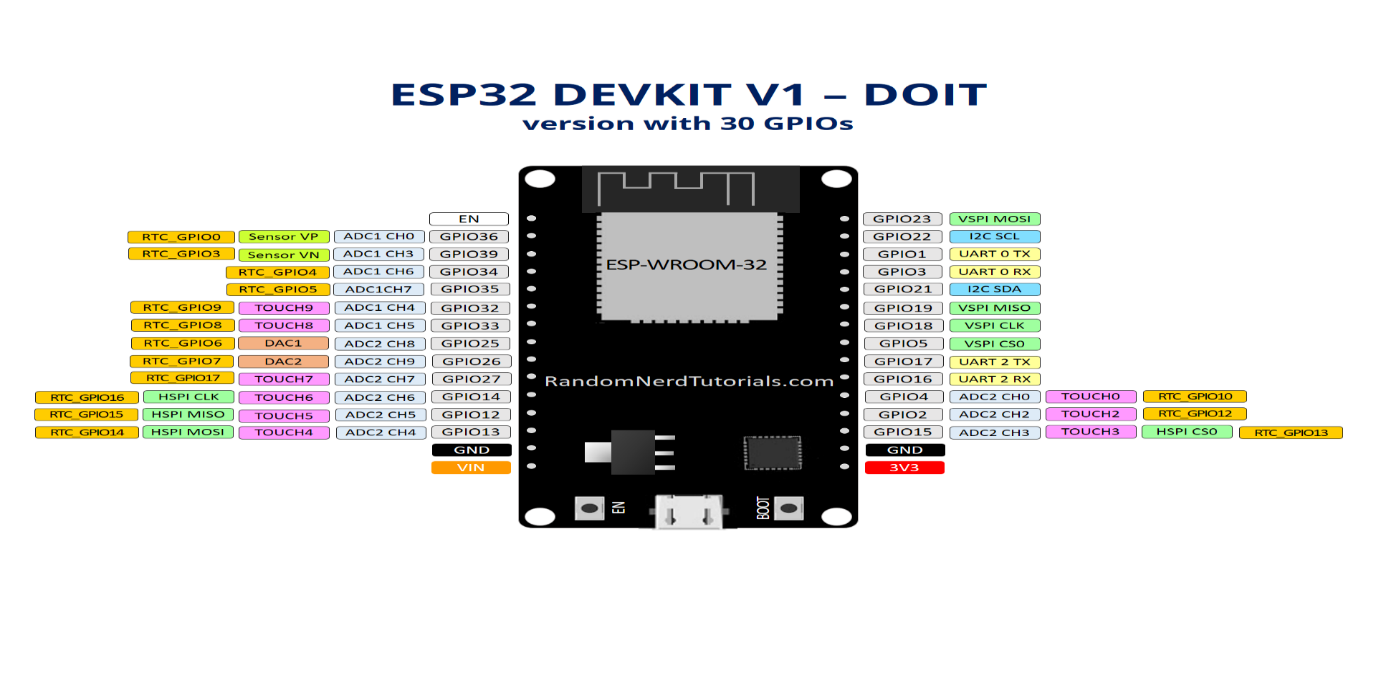
ESP-32, ESP-8266, Arduino Uno, and stm\_32 are the microcontrollers available in the market. Since our project has to use ADCs, WIFI, and LoRa and Arduino does not have a built \_in Wi-Fi, ESP-8266 has only one ADC, STM-32 is more powerful and expensive than ESP-32, But ESP-32 has 16 ADCs, built-in Wi-Fi and is a Lora compatible device and inexpensive than STM-32.

### 2.3.1 Features [7]

* 802.11 b/g/n
* TX/RX over Wi-Fi
* Station and SoftAP
* Bluetooth version 4.2
* Single/Dual-core 32-bit microprocessor(s)
* 4MB flash drive
* Internal 80 MHz oscillator with calibration
* 34 × programmable GPIOs
* 12-bit SAR ADC up to 18 channels
* 2 × 8-bit DAC
* 10 × touch sensors
* 4 × SPI
* Hall sensor
* Random Number Generator (RNG)

### 2.3.2 Pin Diagram

Figure 2‑4 diagram of ESP-32

Figure 2-4 shows the PIN schematics of ESP-32.

# Chapter 3

# Software

This chapter explains all the key details about the software used during the project. The word software is being used in abstract terms and will be explained in the chapter since everything discussed in the chapter is related to software.

## 3.1 C++/C – Programming language

C programming language was invented to create the UNIX operating system. C is the most popular computer language.

### 3.1.1 Applications

C was firstly used for system development tasks, especially for the operating system's program. Because C language codes execute almost as quickly as assembly language code, it was accepted as a system development language. Here are a few instances of how C is used [8]:

* Operating Systems
* Language Compilers
* Assemblers
* Text Editors
* Print Spoolers
* Network Drivers
* Modern Programs
* Databases
* Language Interpreters
* Utilities

## 3.2 Arduino IDE – Hardware Programming tool

The Arduino IDE is also known as Arduino Integrated Development Environment. It offers its features to other boards like the ESP-32 through its libraries in addition to a text editor for writing code, a message area, a text terminal, a toolbar with buttons for frequently used activities, and a variety of menus for Arduino. It connects to the Arduino hardware to upload applications and communicate with them [9]. To connect Arduino to the Computer Serial peripheral Interface is used SPI.

### 3.2.1 Programming Language

The Arduino Programming Language is a C++ foundation. The sketch is a program created using the Arduino programming language. A sketch is often saved with the .ino extension (from Arduino). Your whole code is confined to 2 core functions, which is the primary difference between this and "normal" C or C++. However, every Arduino program must have at least two functions.  The first is referred to as **setup (),** and the second is **loop ()** and you can have more too. While your application is running, the second is continuously called while the first is only called once at program startup. As the point of entry for a program, we do not have a main () method like you are accustomed to in C/C++ [10]. The IDE will verify that the final product is a valid C++ program after you compile your design and will essentially preprocess it to include the missing glue. The remaining code is standard C++, and all legitimate C++ code is also legitimate Arduino code. The fact that you can launch your program over numerous files, but they all need to be in the same folder, is one difference that might give you problems. If your software becomes really large, this limitation could be fatal, but at that point switching to a native C++ configuration will be simple and feasible.

The built-in libraries that are a component of the Arduino programming language enable simple integration with the features offered by the Arduino board.

### 3.2.2 Uploading

Before uploading the sketch firstly select the port by entering the menu Tools > Board and Tools > Port, in the submenu Port, select the port mentioned as COM1 or COM2).

The Arduino bootloader, a small program installed onto the microcontroller’s board, is used to upload the sketch. It helps by declining the need for any external hardware for uploading. The bootloader works for a brief time and when the microcontroller turns on it uploads the recent sketch onto it.

### 3.2.3 Serial Monitor

This displays the transmitted information from the board using a serial or USB port. To send any command to the board, input it in the menu and press enter. Choose the baud rate that corresponds to the data rate supplied by the board [11].

## 3.3 JavaScript – Web development Language

JavaScript is a programming language utilized to make complex web pages and advance their functionality. It is the third tier of common web technologies, the first two (HTML and CSS) [12].

For client-side programming the most used language is JavaScript. The script's code can only be used in wen if it has its link to an HTML document.

* It implies that a web page may not be a simple HTML document, but rather may contain a program that can interact with users, interact with browsers, and generate HTML content on the go.
* When a user performs a task on the web page, JavaScript runs, which compares all the entries that the task created to the stored entries and is sent to the web server if the entries are recognized.
* Use of JavaScript is used to create, and capture events that are initiated by the user, such as button clicks, link navigation, and other explicit or implicit activities.

### 3.3.1 JavaScript Development Tool

Java scripting can be done by using a basic text editor like Notepad. There is no need for a special compiler because it is an interpreted language used in a web browser.

Several businesses have developed JavaScripting tools. Some are listed here:

* **Apps script**: this editor lets you design custom google sheets for data storage.
* **Microsoft FrontPage**: Microsoft created the well-known HTML editor FrontPage. Web designers can create interactive websites with the help of the JavaScript capabilities that FrontPage also offers.

#### Apps Script

Google sheets allow the user to edit, control and use as the database by setting its working through the Google Apps Script. Custom menus, dialogues, and sidebars can be added to Google Sheets using Apps Script. Additionally, you can combine Sheets with other Google services like Calendar, Drive, and Gmail, as well as build your custom functions for Sheets. The majority of Google Sheets scripts use array manipulation to communicate with a spreadsheet's cells, rows, and columns.

# Chapter 4

# Working Mechanism

This chapter describes all the hardware and software implementation details step by step.

## 4.1 System Model

As shown in figure 4-1, a complete project model is given, where the sensor nodes read the value, nodes process the value, and sends it to the server using LoRa transceivers. Then the server/gateway uploads the data to the cloud, which can be monitored from the cloud directly or can be monitored through an android smartphone application. Figure 4-2 shows the flowchart.

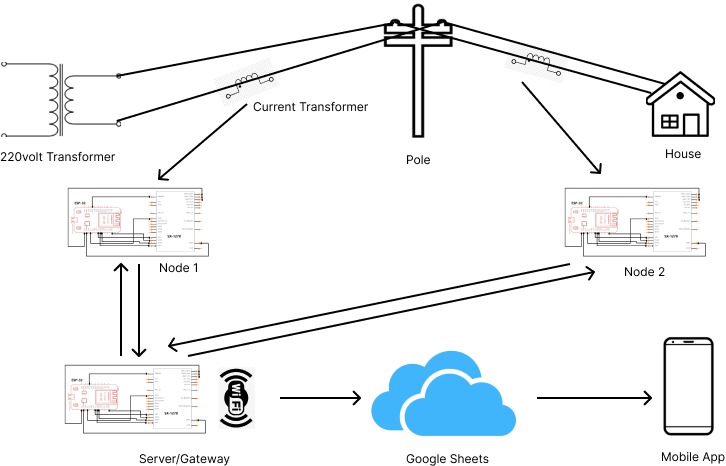


Figure 4‑1 Complete model of the project

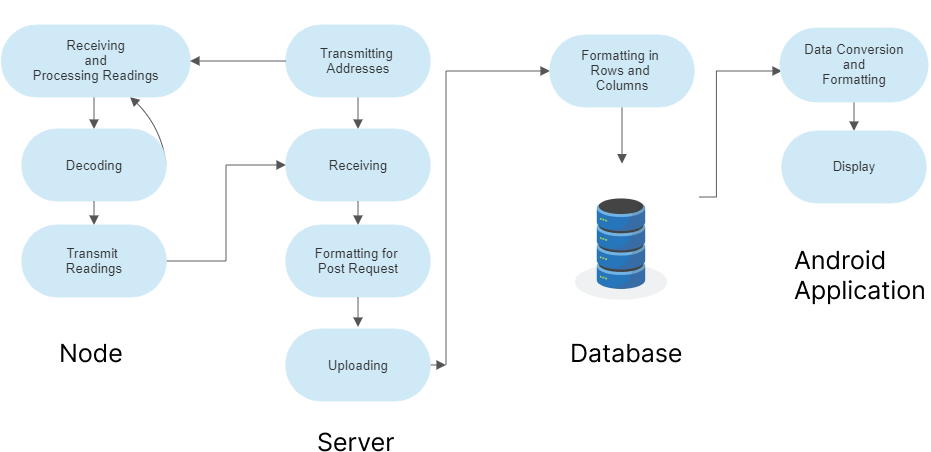


Figure 4‑2 Project Flowchart

## 4.2 Node

The project node is a device that monitors and senses the load lines using a Current transformer and relays it to the ESP-32 where the assigned algorithm calculates the current being transferred through the wire and creates a data packet in the form of a string. The string is then encoded with all the protocols assigned by LoRaWAN which is then relayed to the LoRa connected to the ESP-32 by the serial-peripheral interface. The LoRa on the node side only acts as a transmitter and sends the data in the 433MHZ band. Since the node is comprised of certain modules working together and their interfacing with each other is a major thing that is discussed in the coming topics.

### 4.2.1 Sensor-Controller Interfacing



Figure 4‑3 Sensor-Controller interfacing

#### Apparatus

* Current transformer 2 x 30A to 30mA converter
* ESP-32
* Resistor 100 ohm
* Resistor 2 x 10 Kohm
* Connecting Wires

#### Procedure

* The Current transformer is non-invasive sensor, which wraps around the load line or return line. The current sensor provides gain of 0.001 to the actual current.
* The current is then passed through the 100-ohm resistor called the bypass resistor to create the voltage according to ohm’s law.
* Since AC flows through the resistor, which means the voltage created across is also alternating and has positive and negative values. The measurable voltage range, that ESP-32 provides is 0 to 3.3 volts. So, this problem is coped with by providing DC offset.
* DC offset of 1.55 volt is being provided using the 3.3-volt output from the ESP-32 and using the voltage divider of two 10-K ohm resistors. Using the low value of the resistor can draw more than 12mA of current from ESP-32, which is not affordable.
* Now the voltage created from the current sensor followed by the DC offset is relayed into the ESP-32 ADC pin, where the signal processing and other calculations are performed.
* The same procedure is followed by other current transformer nodes in use by the project. Figure 4-3 shows the circuit diagram of the sensor interfacing.

### 4.2.2 Controller Calculation

When ESP-32 receives voltage value at the ADC pin, controller compares input level with the Vref which is normally 1100mV and is adjustable. This signal is applied from an outer source, and outputs a 12-bit value since the ADC being used is 12-bit. The ADC’s accuracy can be adjusted to the minimum level of 8-bit.

The library used for the current calculation is EmonLib.h (Energy monitoring). It is an open-source library made by the OpenEnergyMonitor company which has its own energy monitoring devices. Library also takes care of the calibration and requires a calibration value to adjust the readings. A known current source was taken and repeated trials were made to get the sensor calibrated with the controller.

The ADC pic and its calibration are done by the API, **elemon\_1.current(ADC\_pin, calibration-value)** where elemon\_1 is a class and current are the object. The API, **Irms = elemon\_1.calcIrms(no. of readings)** calculates the currentcalculates the RMS voltage and calculates from it, the RMS current value and returns the measured RMS current.

### 4.2.3 Controller-Transmitter Interfacing

The LoRa transmitter is connected to the Microcontroller using serial-peripheral interfacing, explained in the above chapter, and figure 4-4 explain the physical connection of the interface. When RMS current has been calculated by the controller it is then sent by LoRa and the library used in coding is **LoRa.h**. the function used in the sketch is, **LoRa.setPins(reset-pin, ss-pin, I/O),** which establishes an SPI connection between microcontroller and LoRa module and **LoRa.begin(433E6)** set the transceiver frequency to 433MHz. **LoRa.setSyncWord()** this function synchronizes the clocks of microcontroller and LoRa module. If the LoRa is not interfaced correctly it starts generating a signal using LED. After the successful interfacing, the data is compiled by the functions mentioned below.

* **LoRa.beginPacket()**
* **LoRa.print("microcontroller-ID")**
* **LoRa.print(Current-value)**
* **LoRa.print("#")**

And the packet is then sent by the function **LoRa.endPacket().** As every LoRa module spreads the signal everywhere around it in the same frequency band for that an ID is attached with the data.

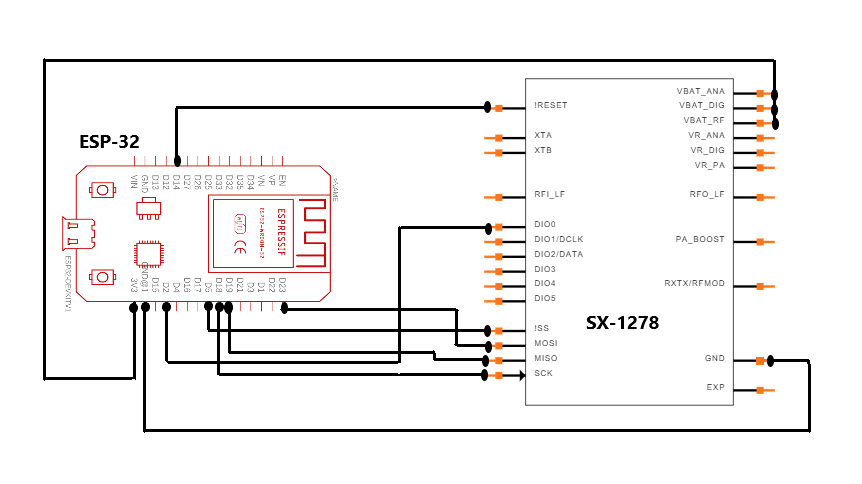


Figure 4‑4 ESP-32 LoRa Interfacing

## 4.3 Gateway/Server

In this project server is a device that receives the data packet sent by the node side LoRa module. The server Decrypts the packet by the following algorithm and then uploads it to the cloud database using Wi-Fi. The Server side is comprised of LoRa receiver and an ESP-32.

### 4.3.1 Controller-Receiver Interfacing

The hardware interfacing is the same as that of the node side. The LoRa is connected to the controller using SPI. But here the LoRa is set as receiver by using the library **LoRa.h**. the function used in the library is, **LoRa.setPins** which required a three-parameter for PIN assignment. The pins to be assigned are the reset pin, ss pin, and I/O pin. **LoRa.begin(433E6)** is a function that establishes an SPI connection between the microcontroller and LoRa module and set the transceiver frequency to 433MHz. **LoRa.setSyncWord()** this function synchronizes the clocks of microcontroller and LoRa module. A certain setup procedure is followed to set up the connection. If the setup is malfunctional it also produces an alert using LED.

After the connection has been established LoRa can receive data from any other LoRa using the function **LoRa.readString()** but the packet with the required information key can only be parsed. The data from the transmitter is decrypted by using the function **LoRaData.substring()** subtracting the data up to a certain length from the packet received and is encrypted again to be sent to the cloud database.

### 4.3.2 Uploading

The uploading is performed using the HTTP protocol by a microcontroller using the libraries **WiFi.h** and **HTTPClient.h**. WiFi.h lets the microcontroller connected to the specified internet connection. For that, the SSID and the Password are stored in the microcontroller for establishing the connection by the function **WiFi.begin(SSID, PASSWORD)** and **WiFi.status()!= WL\_CONNECTED** checks for the connection establishment.

After which Uploading is simply sending a get request to the database made by calling **http.begin(URL)** and **http.GET()**. The get request has its protocol i.e., making the suitable URL with all the information required by the website. And taking in the view that all the protocols are fulfilled when making that URL.

The URL contains:

* Google sheets link
* Google sheets ID
* The parameters

## 4.4 Database

Google Sheets stores the data and the data is formatted by the app script written in JavaScript as shown in figure 4-5. The sheet only gives access to the users with the specified key and monitors any get requests received. Apps Script executes the function **doGet()** whenever a user accesses an app or whenever a program sends the app an HTTP GET request (e). The e argument in this situation denotes an event parameter that may include details about any request parameters. In our case, the **e.parameter** is being used and it is a key/value pair object with the corresponding values for the request parameters. When there are several values for a parameter, just the first value is returned. Simple formatting sets the rows and columns and **new Date()** function assigns a date to every node data. When a user accesses server database through a mobile application his credentials are stored in the google firebase as shown in figure 4-6.

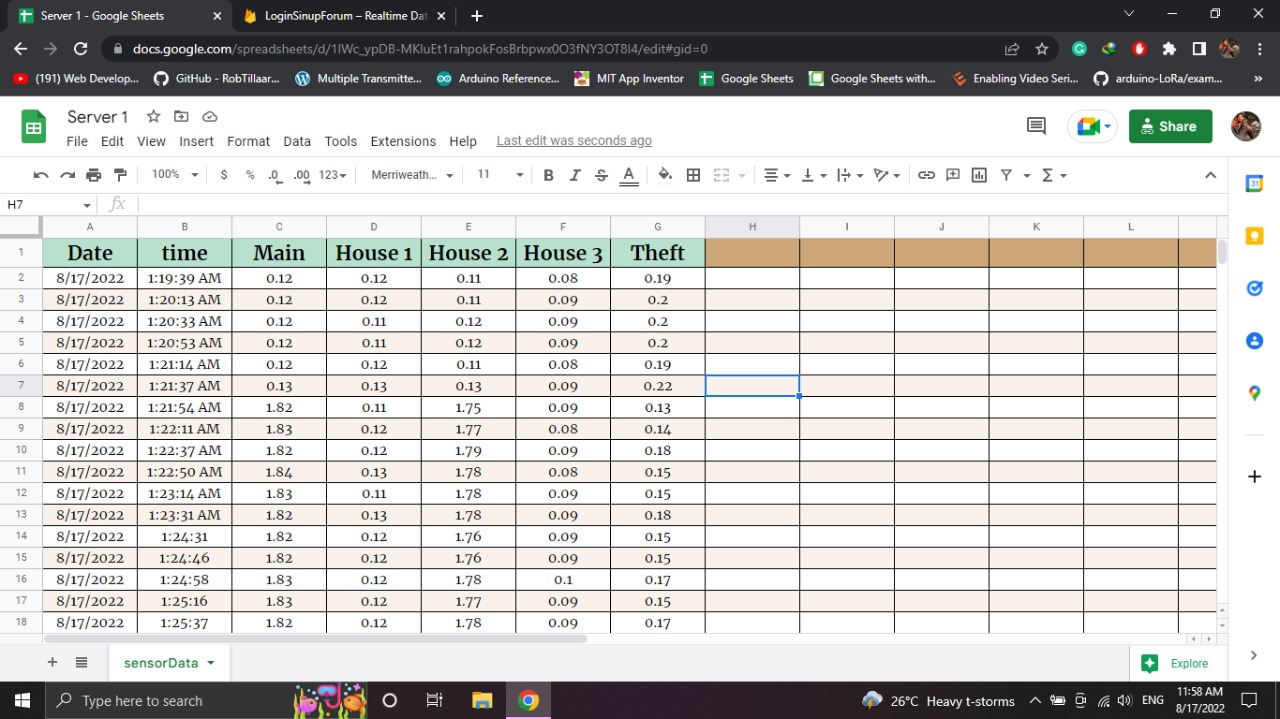


Figure 4‑5 Server Database

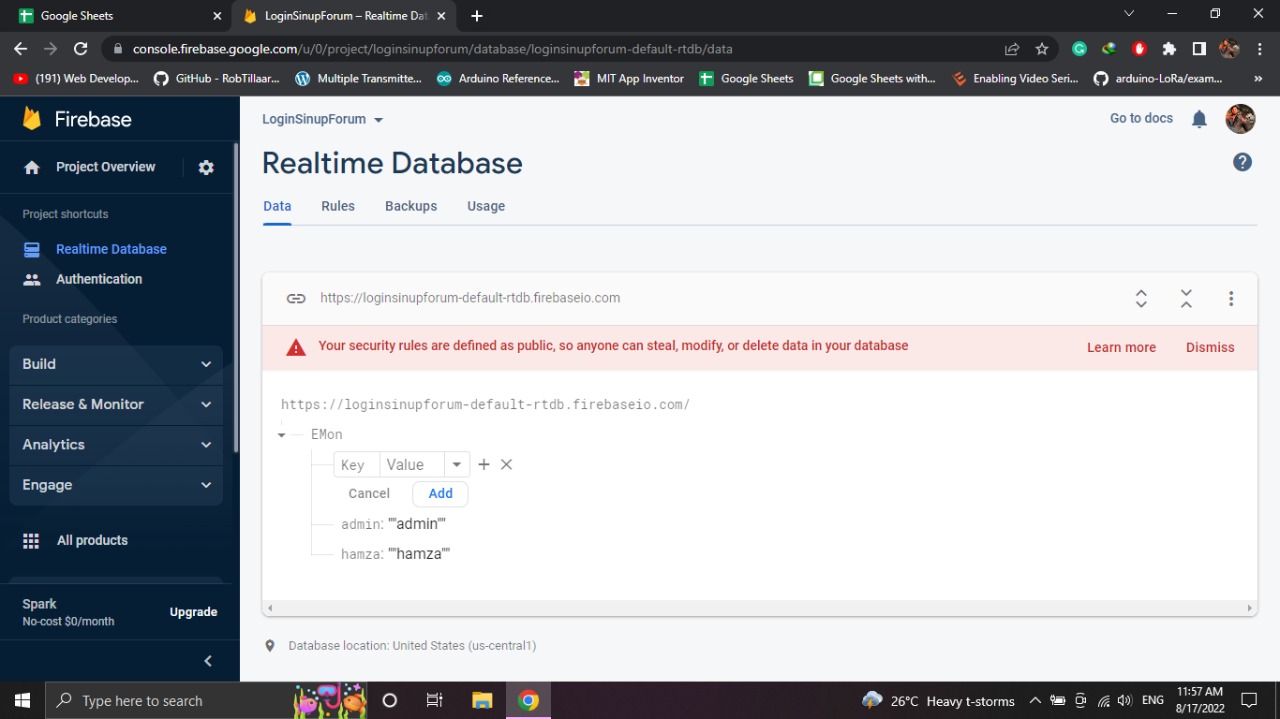


Figure 4‑6 App Database

## 4.5 Application (App) Designing

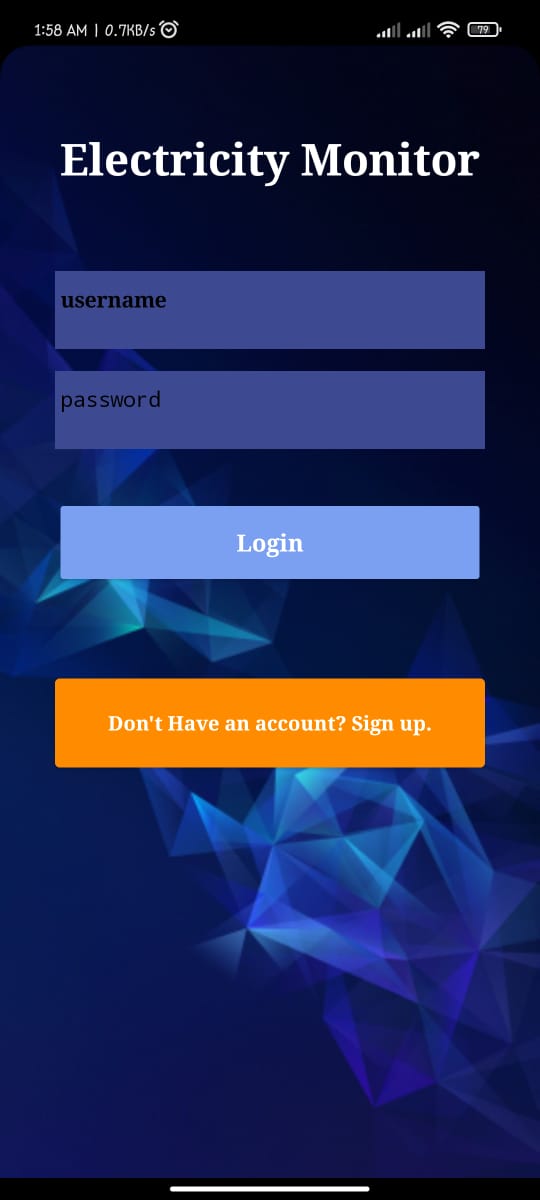
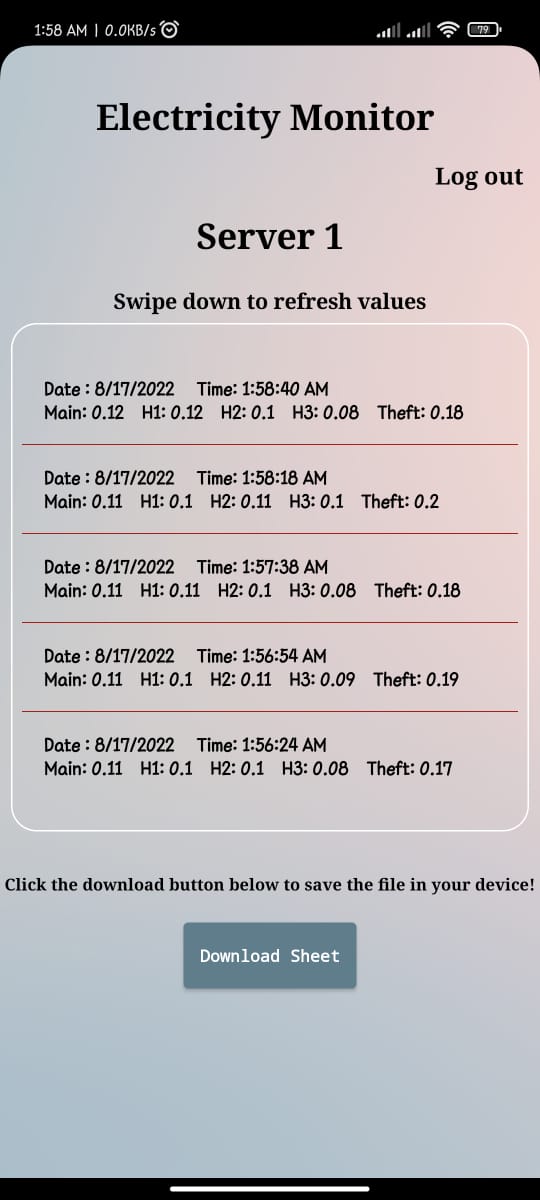
The app is designed with the help of a tool called MIT app inventor. The major function used in the app is **web.get ()** which sends the get request to the google sheet and fetches the data. After which the **webGotText ()** removes the header**,** andconverts the data into CSV table text in the form of a list. Secondly, its formatting function arranges the data in horizontal lists, and thirdly each element of the list is stored in a new vertical list and is deployed onto the screen as shown in figure 4-7a below.

The main screen also has a refresh button which can sync the app manually.

The log-in screen matches the data entered in the required fields with the data stored in the firebase. The screen is shown in figure 4-7b.

The sign-up screen collects the data from a new user and stores that data into the firebase for later matching usage. The screen is shown in figure 4-7c.

Figure 4‑7 a: Main Screen b: Login Screen c: Signup Screen



## 4.6 Self-made System

Figure 4-8 shows the self-made project.

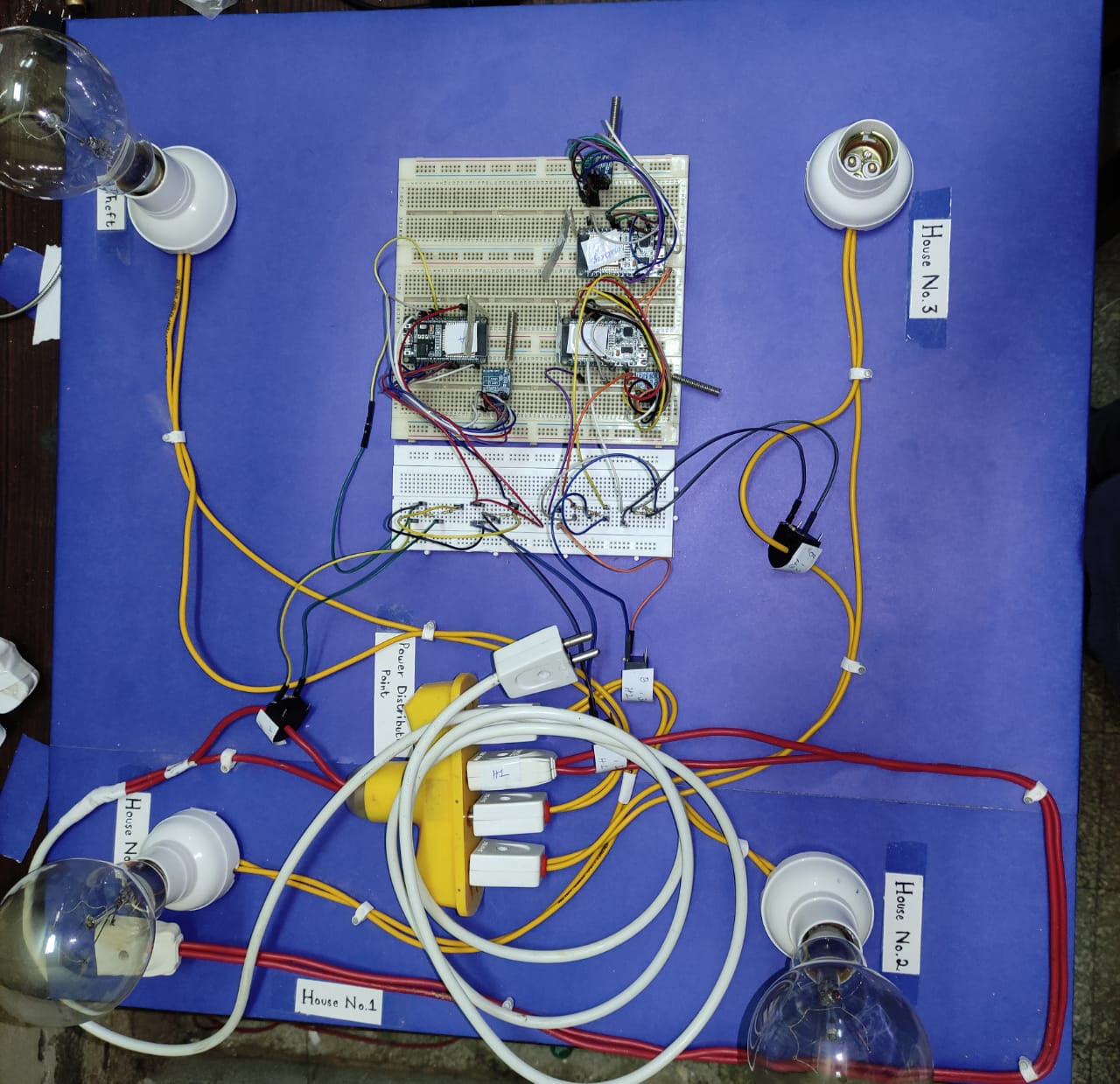


Figure 4‑8 Self-made System

# Results

Results show that if there is no current passing through the load lines the value that we receive is between 0.20 to 0.25 so it can be said that this much current is not readable by the nodes. Above this level, readings are almost in line with the readings taken by the multimeter with a maximum error of 1%. Data on the database is uploaded every 3 seconds from nodes. Figure A shows the data stored in the database.

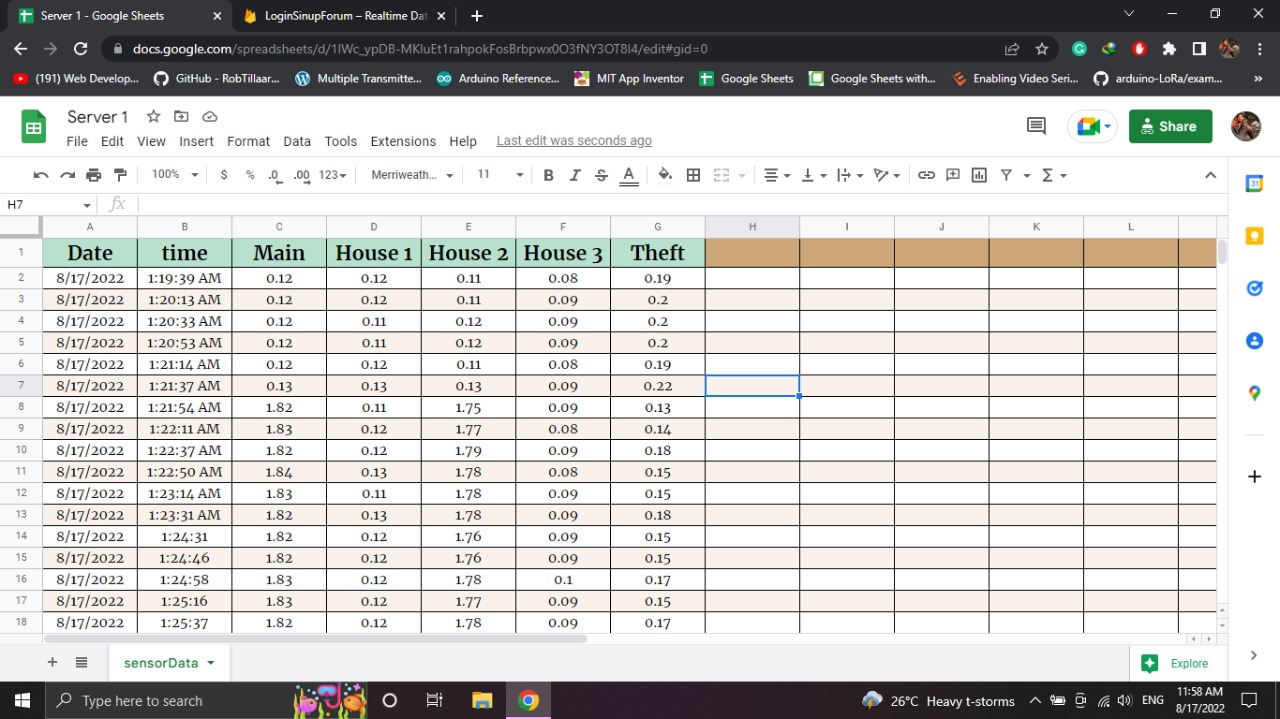


Figure A. results

# Conclusion

The project was aimed to monitor power theft using IoT network, which is accomplished and executed using microcontrollers, sensors, and LoRa successfully. Now it is concluded that use of this network compiles reliable results related to power theft. Work in this project can be used to develop multiple applications such as power monitoring at home and security. We can also monitor the power consumption and optimize electricity usage.

# Future Work

This work can be extended to develop many applications. For example, we can analyze logs from the proposed IoT network to control power consumption. The proposed IoT network can also be used to spot electricity circuit faults by placing current sensors in the wiring network.

# Cost Analysis

|  |  |  |
| --- | --- | --- |
| **Device** | **Quantity** | **Price per unit in Rs** |
| **Node** | **1** | **3560/-** |
| ESP-32 | 1 | 1200/- |
| LoRa (sx-1278) | 1 | 1450/- |
| Current transformer | 2 | 450/- |
| Resistors | 3 | 10/- |
| **Server/Gateway** | **1** | **2650/-** |
| ESP-32 | 1 | 1200/- |
| LoRa (SX-1278) | 1 | 1450/- |
| **Total Cost** | **1** | **6210/-** |

# References

1. Hussain, Z., Memon, S., Shah, R., Bhutto Z., and Aljawarneh, “Methods and Techniques of Electricity Thieving in Pakistan | *Journal of Power and Energy Engineering”*, *www.scirp.org*, 2022. [Online]. Available: <https://www.scirp.org/journal/paperinformation.aspx?paperid=70365> [Accessed: 10- Aug- 2022]
2. S. Kumar, P. Tiwari and M. Zymbler, "Internet of Things is a revolutionary approach for future technology enhancement: a review", 2022.
3. "Introduction to SPI Interface | Analog Devices", *Analog.com*, 2022. [Online]. Available: https://www.analog.com/en/analog-dialogue/articles/introduction-to-spi-interface.html. [Accessed: 14- Aug- 2022]
4. "What is a Microcontroller and How Does it Work?", *IoT Agenda*, 2022. [Online]. Available: https://www.techtarget.com/iotagenda/definition/microcontroller. [Accessed: 16- Aug- 2022]
5. C. Sensors and T. Sensors, "Types and Characteristics of Current Sensors | Tutorials | Current Sensors | Products | Asahi Kasei Microdevices (AKM)", *Tutorials | Current Sensors | Products | Asahi Kasei Microdevices (AKM)*, 2022. [Online]. Available: https://www.akm.com/eu/en/products/current-sensor/tutorial/types-characteristics/. [Accessed: 16- Aug- 2022]
6. "Lora - Wikipedia", *En.wikipedia.org*, 2022. [Online]. Available: https://en.wikipedia.org/wiki/Lora. [Accessed: 16- Aug- 2022]
7. ESP. 32, "An Overview", *espressif.com*, 2022. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32\_datasheet\_en.pdf. [Accessed: 16- Aug- 2022]
8. "C Tutorial", *Tutorialspoint.com*, 2022. [Online]. Available: https://www.tutorialspoint.com/cprogramming/index.htm. [Accessed: 16- Aug- 2022]
9. *Docs.arduino.cc*, 2022. [Online]. Available: https://docs.arduino.cc/software/ide-v1/tutorials/arduino-ide-v1-basics/. [Accessed: 16- Aug- 2022]
10. F. Copes, "Introduction to the Arduino Programming Language", *Flaviocopes.com*, 2022. [Online]. Available: https://flaviocopes.com/arduino-programming-language/. [Accessed: 16- Aug- 2022]
11. *Docs.arduino.cc*, 2022. [Online]. Available: https://docs.arduino.cc/software/ide-v1/tutorials/arduino-ide-v1-basics/. [Accessed: 16- Aug- 2022]
12. JavaScript - Overview", *Tutorialspoint.com*, 2022. [Online]. Available: https://www.tutorialspoint.com/javascript/javascript\_overview.htm. [Accessed: 16- Aug- 2022]

# Appendix A

Node

Appendix A contains the program uploaded in node A. which is the same for all other nodes.

/\*

LoRa Node A

Current value below 0.25 is not measurable.

\*/

// including libraries

#include <SPI.h>

#include <LoRa.h>

#include <EmonLib.h>

//pin configuration for LoRa SX-1278

#define nss 5

#define rst 14

#define dio0 2

//pins for sensors connected to Node A for current reading

#define H\_1 34

#define \_main 27

//taking two instances from the library for our sensor to get values

EnergyMonitor emon1**;**

EnergyMonitor emon2**;**

// Flag set by callback to perform read process in main loop

volatile bool doRead **=** **false;**

//assign addresses for LoRa Transmission for master node and Node A and two //varriablefor authentication request

byte MasterNode **=** 0xFF**;**

byte NodeA **=** 0xBB**;**

int recipient**;**

byte sender**;**

//defining varriable for currunt reading and incoming request from gateway

String currentReadings **=** ""**;**

String incoming **=** ""**;**

//variable store currunt readings

float house1**,**Main**;**

void setup**()**

**{**

LoRa**.**setPins**(**nss**,** rst**,** dio0**);**

**if** **(!**LoRa**.**begin**(**433E6**))**

**while** **(**1**);**

LoRa**.**onReceive**(**onReceive**);** // register the receive callback

LoRa**.**receive**();** // put the radio into receive mode

//instances setting up pins and calibration value

emon1**.**current**(**H\_1**,** 4.9**);**

emon2**.**current**(**\_main**,** 4.9**);**

**}**

void loop**()**

**{**

//Dummy varriables to get value from sensor for making the value stable

float dummy1 **=** emon1**.**calcIrms**(**1480**);**

float dummy2 **=** emon2**.**calcIrms**(**1480**);**

// parse for a packet, and call onReceive with the result:

**if** **(**doRead**)**

**{**

readMessage**();**

// Set flag back to false so next read will happen only after next ISR event

doRead **=** **false;**

LoRa**.**onReceive**(**onReceive**);**

LoRa**.**receive**();** //again put the radio in recieve mode

**}**

**}**

void onReceive**(**int packetSize**)**

**{**

**if** **(**packetSize **==** 0**)** // if there's no packet, return

**return;**

doRead **=** **true;**

**}**

void readMessage**()**

**{**

incoming **=** ""**;**

recipient **=** LoRa**.**read**();**

sender **=** LoRa**.**read**();**

**while** **(**LoRa**.**available**())**

incoming **+=** **(**char**)**LoRa**.**read**();**

**if** **(**recipient **!=** NodeA **&&** sender **!=** MasterNode**)**

**return;** // skip rest of function

int Val **=** incoming**.**toInt**();**

**if** **(**Val **==** 10**)** //check if the request is for the node

**{**

house1 **=** emon1**.**calcIrms**(**1480**);**

Main **=** emon2**.**calcIrms**(**1480**);**

currentReadings **=** house1**;**

currentReadings **+=** "/"**;**

currentReadings **+=** Main**;**

currentReadings **+=** "#"**;**

sendMessage**(**currentReadings**,** MasterNode**,** NodeA**);**

currentReadings **=** ""**;**

**}**

**else**

**return;**

**}**

void sendMessage**(**String outgoingSendingMessage**,** byte MasterNode**,** byte NodeA**)**

**{**

LoRa**.**beginPacket**();** // start packet

LoRa**.**write**(**MasterNode**);** // add destination address

LoRa**.**write**(**NodeA**);** // add sender address

LoRa**.**print**(**outgoingSendingMessage**);** // add payload

LoRa**.**endPacket**();** // finish packet and send it

**}**

# Appendix B

Server/Gateway

This appendix contains the code uploaded to the server/gateway.

/\*

Master LoRa Node(Gateway)

\*/

#include <WiFi.h>

#include <HTTPClient.h>

#include <SPI.h>

#include <LoRa.h>

#include <NTPClient.h>

#include <WiFiUDP.h>

WiFiUDP ntpUDP**;**

// Selecting Europe for getting real-time time stamps

NTPClient timeClient**(**ntpUDP**,** "de.pool.ntp.org"**,** 18000**,** 60000**);**

// Flag set by callback to perform read process in main loop

volatile bool doRead **=** **false;**

// flag set for conforming that gate has recived data from every node

boolean flag1 **=** **false,** flag2 **=** **false;**

//Setting up LoRa Pins

#define nss 5 //GPIO 5

#define rst 14 //GPIO 14

#define dio0 4 //GPIO 4

//Setting up WiFi connectivity.

const char**\*** ssid **=** "QAU"**;**

const char**\*** PASSWORD **=** "Abcd1dcbA2"**;**

//setting up credentials for dataBase which in our case google excel sheets

const char**\*** APP\_SERVER **=** "script.google.com"**;**

const char**\*** key **=** "AKfycbyk4DJzRuIarOcUs07uNzmrpvUvuUK\_EJCI6X\_2FRTf9FjlCZQUn9MM2PpLjagG-1qq8w"**;**

String serverTime **=** ""**;**

String distPnt **=** ""**;**

String path1 **=** ""**;**

String path2 **=** ""**;**

String path3 **=** ""**;**

float theft**;**

//assigning addresses to the gateway and senders

byte MasterNode **=** 0xFF**;**

byte NodeA **=** 0xBB**;**

byte NodeB **=** 0xCC**;**

//varriable for storing and for sending data to cloud

String incomingValues **=** ""**;**

String currentMain **=** ""**;**

String currentA **=** ""**;**

String currentB **=** ""**;**

String currentC **=** ""**;**

// Tracks the time since last event fired

unsigned long previousMillis **=** 0**;**

unsigned long int previoussecs **=** 0**;**

unsigned long int currentsecs **=** 0**;**

unsigned long currentMillis **=** 0**;**

int interval **=** 1 **;** // updated every 1 second

int Secs **=** 0**;**

void setup**()**

**{**

WiFi**.**begin**(**ssid**,** PASSWORD**);**

pinMode**(**2**,** OUTPUT**);**

**while** **(**WiFi**.**status**()** **!=** WL\_CONNECTED**)**

digitalWrite**(**2**,** HIGH**);** //if WiFi will not be connected LED will remain turn ON

timeClient**.**begin**();**

digitalWrite**(**2**,** LOW**);** //as the WiFi got connected LED will turn OFF

LoRa**.**setPins**(**nss**,** rst**,** dio0**);**

**if** **(!**LoRa**.**begin**(**433E6**))**

**while** **(**1**);**

// register the receive callback

LoRa**.**onReceive**(**onReceive**);**

// put the radio into receive mode

LoRa**.**receive**();**

**}**

void loop**()**

**{**

**if** **(**doRead**)**

**{**

readMessage**();**

// Set flag back to false so next read will happen only after next ISR event

doRead **=** **false;**

**}**

currentMillis **=** millis**();**

currentsecs **=** currentMillis **/** 1000**;**

**if** **((**unsigned long**)(**currentsecs **-** previoussecs**)** **>=** interval**)**

**{**

**if** **(** Secs **>=** 5 **)**

Secs **=** 1**;**

**if** **(**Secs **==** 2**)**

**{**

String message **=** "10"**;** //an identifier msg act as ID for security

//will send request to node A for values

sendMessage**(**message**,** MasterNode**,** NodeA**);**

LoRa**.**onReceive**(**onReceive**);**

LoRa**.**receive**();** //sets the Radio in recieve mode again

**}**

**if** **(**Secs **==** 4**)**

**{**

String message **=** "20"**;** //an identifier msg act as ID for security

//will send request to node B for values

sendMessage**(**message**,** MasterNode**,** NodeB**);**

LoRa**.**onReceive**(**onReceive**);**

LoRa**.**receive**();** //sets the Radio in recieve mode again

**}**

previoussecs **=** currentsecs**;**

Secs **=** Secs **+** 1**;**

**}**

timeClient**.**update**();**

serverTime **=** timeClient**.**getFormattedTime**();** //get current time

**}**

//send a signal to each node for requesting the data of sensors.

void sendMessage**(**String outgoingRequestForData**,** byte MasterNode**,** byte otherNode**)** **{**

LoRa**.**beginPacket**();** // start packet

LoRa**.**write**(**otherNode**);** // add destination address

LoRa**.**write**(**MasterNode**);** // add sender address

LoRa**.**print**(**outgoingRequestForData**);** // add payload

LoRa**.**endPacket**();** // finish packet and send it

**}**

//call back function will confirm that here's some data on the gateway to be recieved

void onReceive**(**int packetSize**)**

**{**

**if** **(**packetSize **==** 0**)** // if there's no packet, return

**return;**

doRead **=** **true;**

**}**

/\*

this function will read all the data and check if data is received from both node and prepare

the data to be uploaded to the sheets.

\*/

void readMessage**()**

**{**

// read packet header bytes:

int recipient **=** LoRa**.**read**();** // recipient address

byte sender **=** LoRa**.**read**();** // sender address

incomingValues **=** ""**;**

**while** **(**LoRa**.**available**())**

incomingValues **+=** **(**char**)**LoRa**.**read**();** //reading data message

**if** **(**sender **==** NodeA**)** //checking if data is from NodeA

**{**

int pos1 **=** incomingValues**.**indexOf**(**'/'**);**

int pos2 **=** incomingValues**.**indexOf**(**'#'**);**

path1 **=** incomingValues**.**substring**(**0**,** pos1**);**

distPnt **=** incomingValues**.**substring**(**pos1 **+** 1**,** pos2**);**

flag1 **=** **true;**

**}**

**if** **(**sender **==** NodeB**)** //checking if data is from NodeA

**{**

int pos1 **=** incomingValues**.**indexOf**(**'/'**);**

int pos2 **=** incomingValues**.**indexOf**(**'#'**);**

path2 **=** incomingValues**.**substring**(**0**,** pos1**);**

path3 **=** incomingValues**.**substring**(**pos1 **+** 1**,** pos2**);**

flag2 **=** **true;**

**}**

**if** **(**flag1 **==** **true** **&&** flag2 **==** **true)** //checking if data is recieved

**{**

theft **=** **(**path1**.**toFloat**()** **+** path2**.**toFloat**()** **+** path3**.**toFloat**())** **-** distPnt**.**toFloat**();**

accessToGoogleSheets**();**

flag1 **=** **false;**

flag2 **=** **false;**

**}**

**}**

void accessToGoogleSheets**()**

**{**

HTTPClient http**;**

String URL **=** "https://script.google.com/macros/s/"**;**

URL **+=** key**;**

URL **+=** "/exec?"**;**

URL **+=** "time="**;**

URL **+=** serverTime**;**

URL **+=** "&distributionPoint="**;**

URL **+=** distPnt**;**

URL **+=** "&path1="**;**

URL **+=** path1**;**

URL **+=** "&path2="**;**

URL **+=** path2**;**

URL **+=** "&path3="**;**

URL **+=** path3**;**

URL **+=** "&theft="**;**

URL **+=** theft**;**

http**.**begin**(**URL**);**

http**.**GET**();**

**}**

# Appendix C

Database

This appendix contains the formatting code for the google sheets. The code given is written in JavaScript.

function doGet(e) {

    let id = '1IWc\_ypDB-MKluEt1rahpokFosBrbpwx0O3fNY3OT8I4';

    let sheetName = 'sensorData';

    var result;

    // e.parameter has received GET parameters, i.e. current measurment from differnt sensors

    if (e.parameter == undefined)

      result = 'Parameter undefined';

    else {

            var sheet = SpreadsheetApp.openById(id).getSheetByName(sheetName);

            var newRow = sheet.getLastRow() + 1;  // get row number to be inserted

            var rowData = [];

            var d = new Date();

            rowData[0] = d.toLocaleDateString();

            rowData[1] = e.parameter.time;

            rowData[2] = e.parameter.distributionPoint;

            rowData[3] = e.parameter.path1;

            rowData[4] = e.parameter.path2;

            rowData[5] = e.parameter.path3;

            rowData[6] = e.parameter.theft;

            // 1 x rowData.length cells from (newRow, 1) cell are specified

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

            // insert data to the target cells

            newRange.setValues([rowData]);

            result =  'Ok';

            }

    return ContentService.createTextOutput(result);

}

# Appendix D

This appendix contains all the coding performed on kodular app platform in form of a .png file.

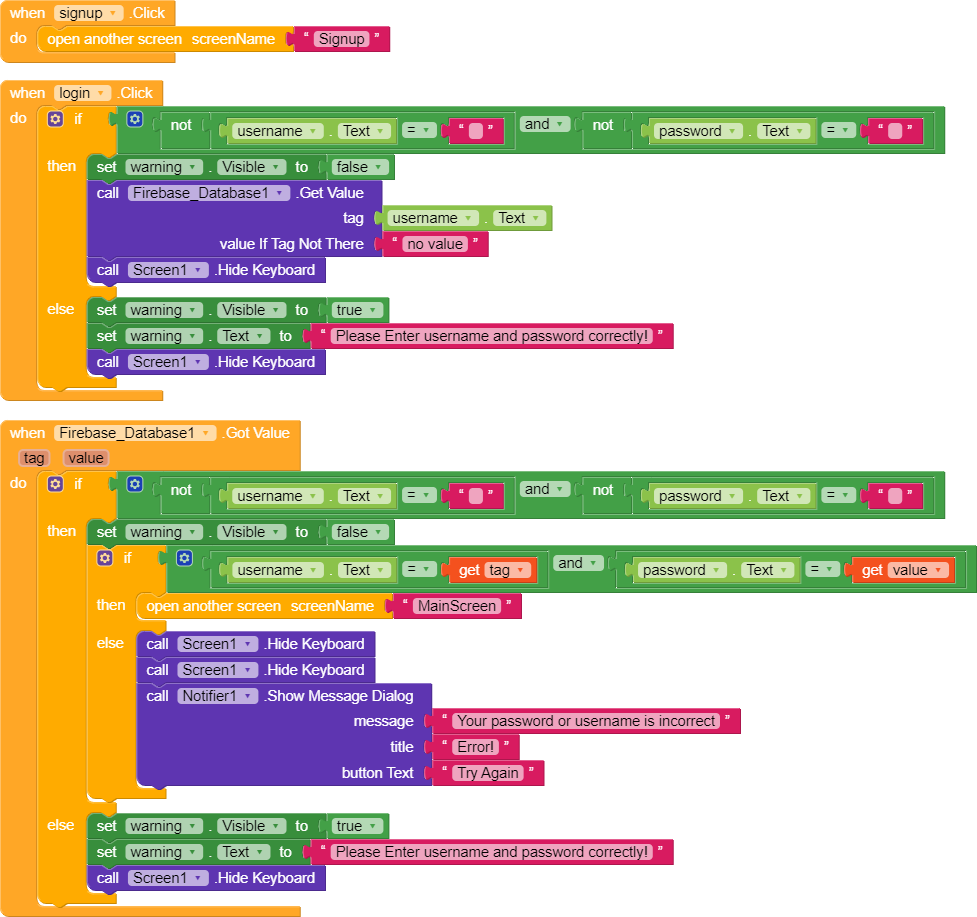


Figure B. Login screen

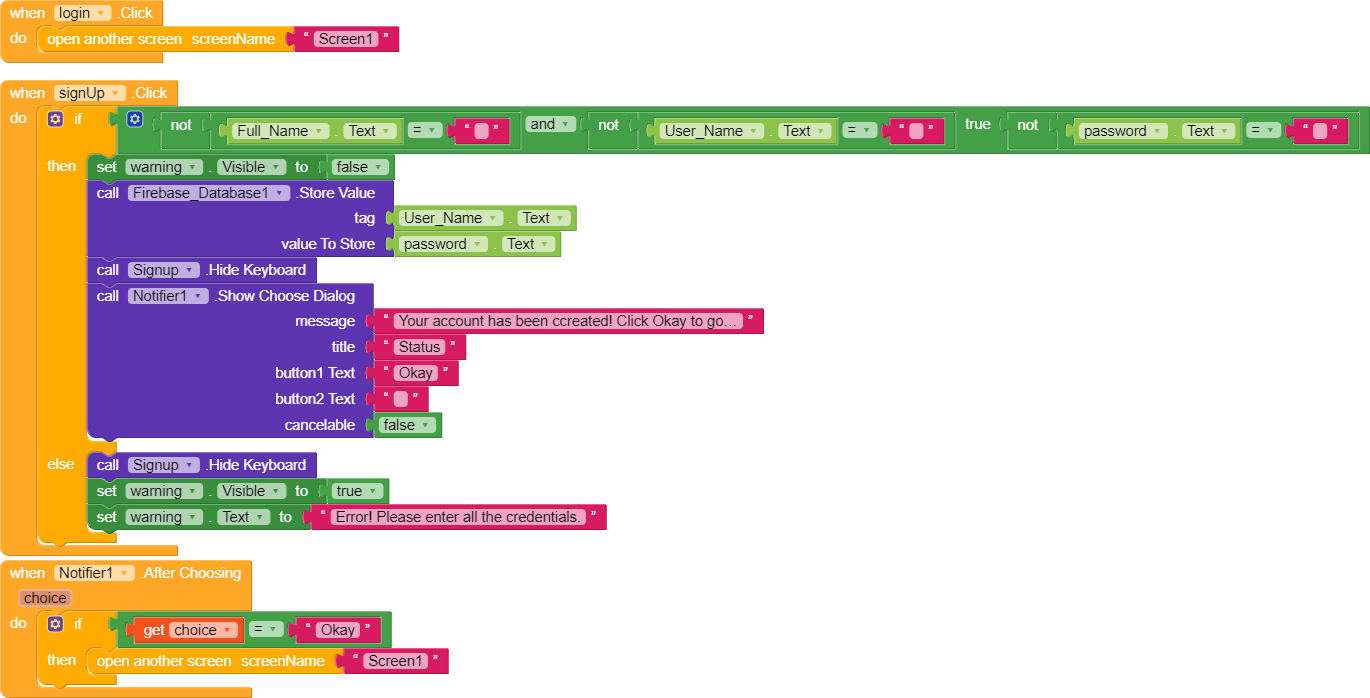


Figure C. Sign Up screen

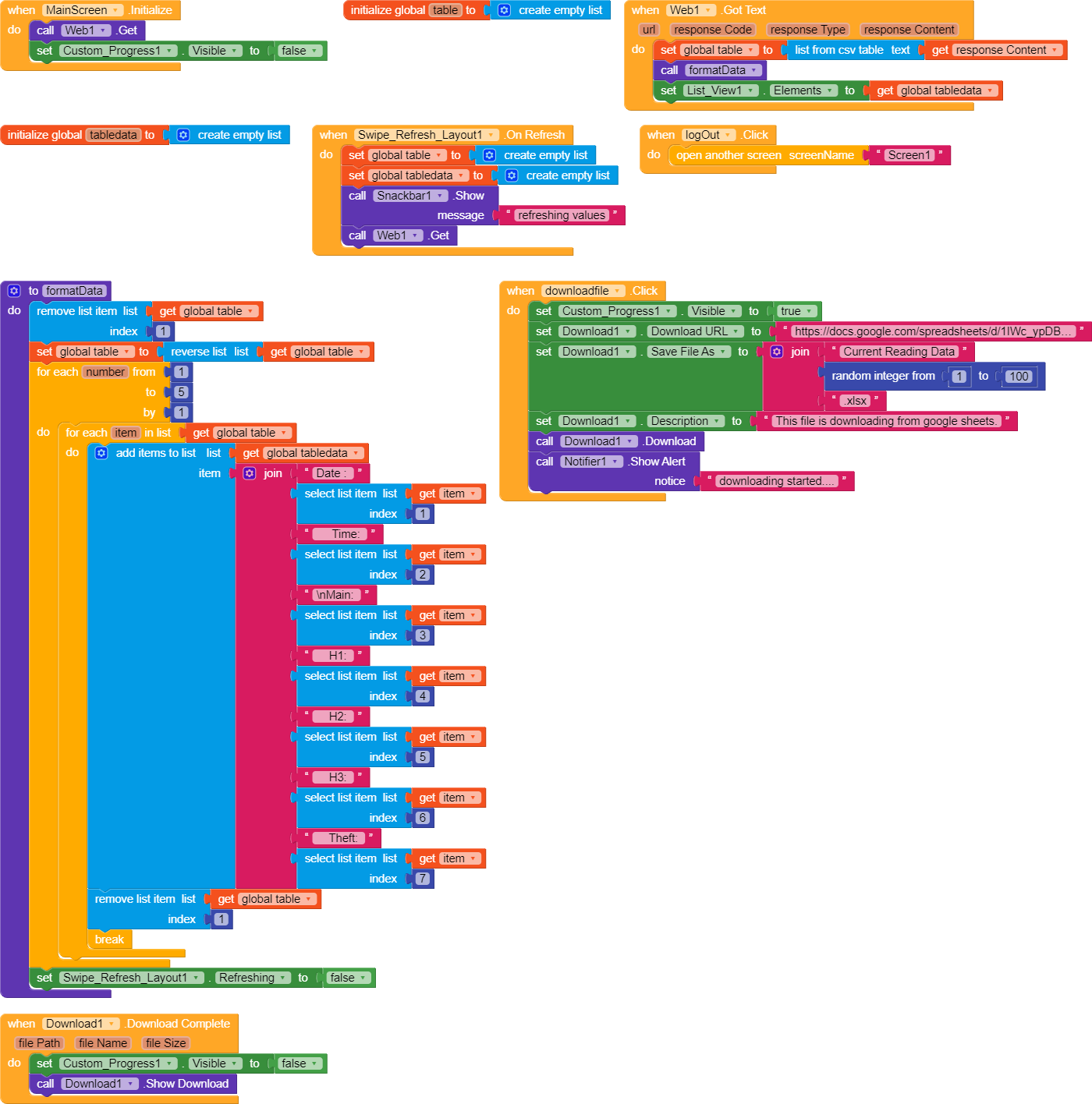


Figure D. Main screen