Project Report
Smart Grid Systems

By-

Madhav M Anand
2018A8PS0028G

Apoorv Singh
2018A3PS0640G

Piyush Maheshwari
2018A8PS0447G

Anmol Mathur
2018A3PS0409G

For the Subject

Internet of Things

EEE F411

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

SEMESTER-II (2020-21)

Submitted to-

Prof. Anupama Karuppiah

ACKNOWLEDGEMENT

We would like to express our special thanks of gratitude to Prof. Anupama Karuppiah and prof. Sarang C Dhongdi who gave us this opportunity to do this wonderful project namely The Smart Grid Systems, it really helped us in exploring and understanding IoT systems very easily. We are Really thankful to the instructors.

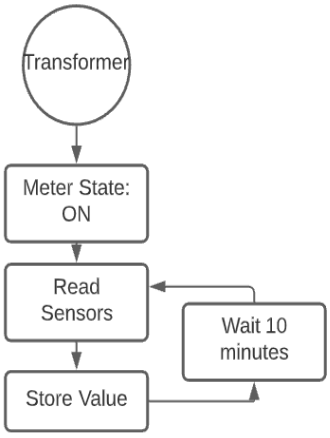
TABLE OF CONTENTS

PROJECT DESCRIPTION

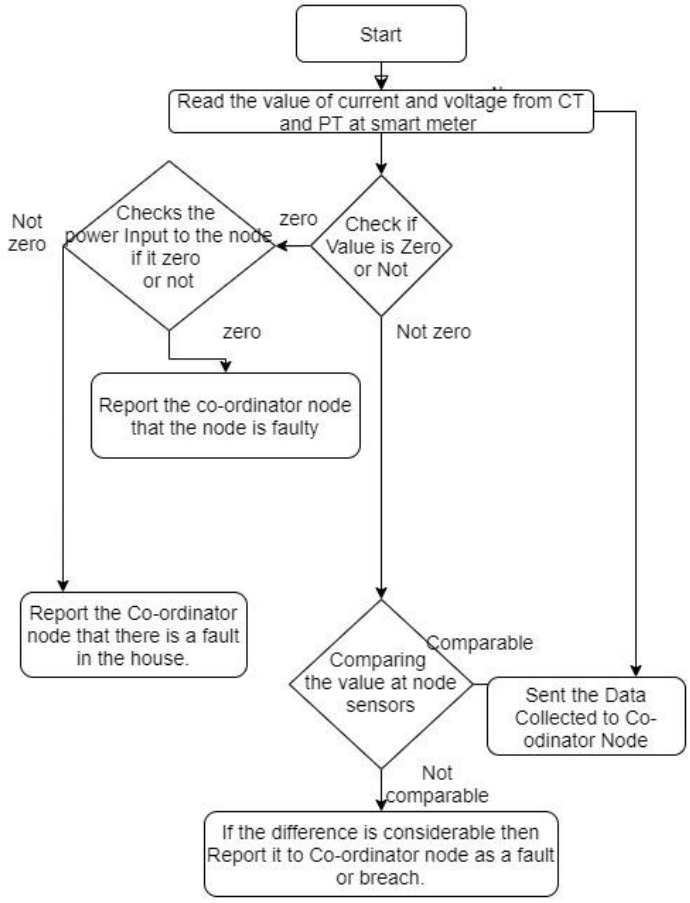
We have successfully attempted to construct a Prototype for a level 5 IoT system which can be applied to monitor Electric grid system Transformer health in real time. The Prototype uses a WiFi module with ESP8266 through which data is transmitted to the IoT Web application Thingspeak.com. Data sent to the Web application can be visualized graphically on a real time basis. The NodeMCU was programmed using Arduino IDE software. The DHT11 Sensor in our prototype is supposed to sense temperature and humidity levels in the transformer box. HC-SR04 ultrasonic sensor has the role to sense the Oil level depth inside the transformer. ACS712 is used to measure the current and power consumed. The Project needs to be further developed to realise. Features such as an alerting system and theft detection are still only in the process of being implemented.

PROPOSED IDEAL IOT SMART GRID SYSTEM

This is the ideal system architecture which was planned for the smart grid system. The proposed system has Transformer as a controller unit and Smart Meter as the end unit. The data collected from the Smart Meters will be received by the MCU at the transformer and minimal processing will be done ,then it will be sent to the servers for the data processing and analysis.

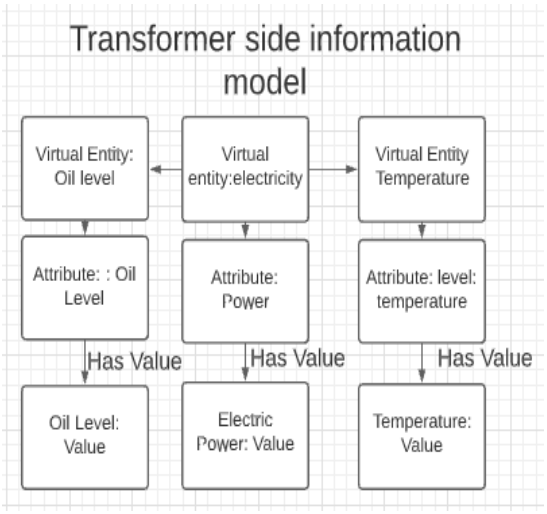


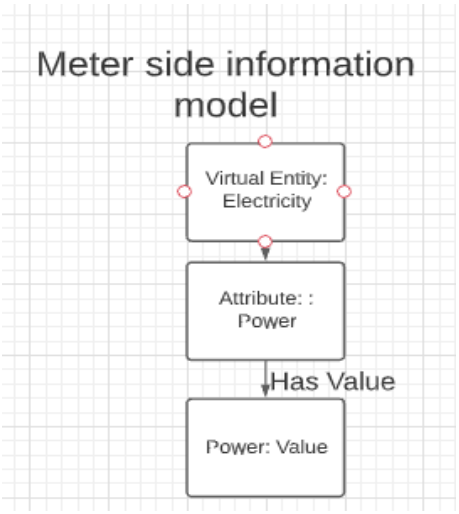
Transformers Sensors Flowchart



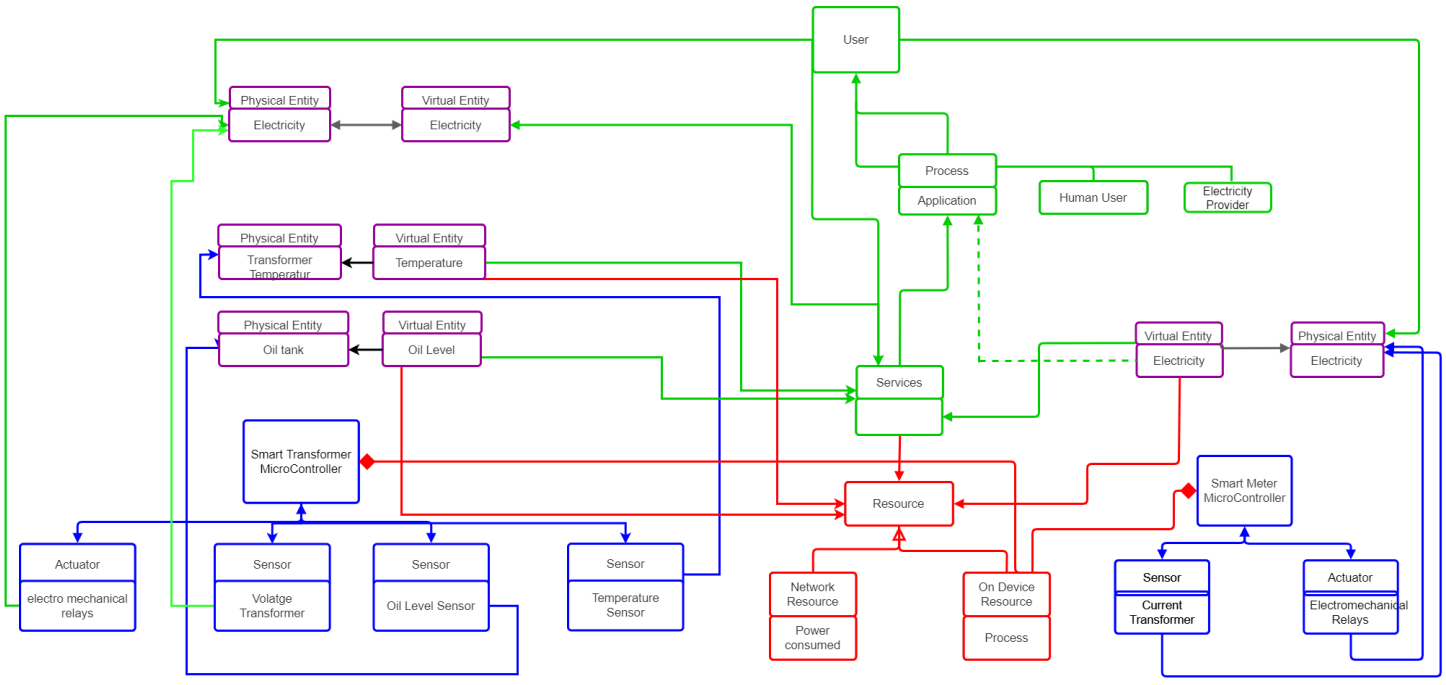
Prescribed algorithm Model for theft and fault detection

Information model specification for smart meter and transformer





Domain Model Specification



HARDWARE & SOFTWARE DESCRIPTION

Hardware Used:

1) Lolin Node MCU V3 (ESP8266 12E WiFi Module):

   The NodeMCU ESP8266 development board comes with the ESP-12E module containing ESP8266 chip having Tensilica Xtensa 32-bit LX106 RISC microprocessor. This microprocessor supports RTOS and operates at 80MHz to 160 MHz adjustable clock frequency. NodeMCU has 128 KB RAM and 4MB of Flash memory to store data and programs. Its high processing power with in-built Wi-Fi / Bluetooth and Deep Sleep Operating features make it ideal for IoT projects.

2) ACS712(30A) Current Sensor:

   The sensor gives precise current measurement for both AC and DC signals. Thick copper conductor and signal traces allows for survival of the device up to 5 times overcurrent conditions.

3) HC SR04 Ultrasonic Sensor:

   It uses SONAR to determine the distance of an object just like the bats do. It offers excellent non-contact range detection with high accuracy and stable readings in an easy-to-use package from 2 cm to 400 cm or 1” to 13 feet.

4) DHT11 Temperature and humidity sensor:

   It is a basic, ultra low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin (no analog input pins needed).

Software and Web Services Used:

1) Arduino IDE:

   The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino and Genuino hardware to upload programs and communicate with them.

2) Thingspeak:

   ThingSpeak provides instant visualizations of data posted by your devices to ThingSpeak. With the ability to execute MATLAB® code in ThingSpeak you can perform online analysis and processing of the data as it comes in. ThingSpeak is often used for prototyping and proof of concept IoT systems that require analytics.

ARDUINO IDE CODE

```
const int trigPin = D5;
const int echoPin = D6;
const int sensorIn = A0;
int mVperAmp = 66; // use 185 for 5A, 100 for 20A Module and 66 for 30A Module

#define PIN A0
double Voltage = 0;
double VRMS = 0;
double AmpsRMS = 0;
long duration;
int distance;

#include<ESP8266WiFi.h>
#include<DHT.h>
#include<ThingSpeak.h>

DHT dht(D1, DHT11);

WiFiClient client;

long myChannelNumber = 1363665;
const char myWriteAPIKey[] = "90USBIUE73NOJPNX";


void setup() {
  // put your setup code here, to run once:
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
  pinMode(A0, INPUT);
  Serial.begin(9600);
  WiFi.begin("S-431 Airtel", "abda@341");
  while(WiFi.status() != WL_CONNECTED)
```

```
  {
    delay(200);
    Serial.print("..");
  }
  Serial.println();
  Serial.println("NodeMCU is connected!");
  Serial.println(WiFi.localIP());
  dht.begin();
  ThingSpeak.begin(client);
}
float getVPP()
{
  float result;

  int readValue;             //value read from the sensor
  int maxValue = 0;          // store max value here
  int minValue = 1024;          // store min value here
  int itr=0;
   uint32_t start_time = millis();

   while((millis()-start_time) < 1000) //sample for 1 Sec
   {
      itr=itr+1;
       readValue = analogRead(sensorIn);
       // see if you have a new maxValue
       if (readValue > maxValue)
       {
         //  /record the maximum sensor value/
           maxValue = readValue;
       }
       if (readValue < minValue)
       {
          // /record the maximum sensor value/
           minValue = readValue;
```

```
    }
  }


  result = ((maxValue - minValue) *5)/1024.0;

  return result;
}

void loop() {
  //Current Sensor
   Voltage = getVPP();
 VRMS = (Voltage/2.0) *0.707; // sq root
 AmpsRMS = (VRMS * 1000*0.31468)/mVperAmp;
 float Wattage = (220*AmpsRMS);
 Serial.println(" Amps RMS ");
 Serial.print(Wattage);
 Serial.println(" Watt ");

  // put your main code here, to run repeatedly:
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  Serial.println("Temperature: " + (String) t);
  Serial.println("Humidity: " + (String) h);

  //sonic sensor
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);

  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
```

```
 // Reads the echoPin, returns the sound wave travel time in microseconds
 duration = pulseIn(echoPin, HIGH);

 // Calculating the distance
 distance= duration*0.034/2;
 // Prints the distance on the Serial Monitor
 Serial.print("Distance: ");
 Serial.println(distance);

 ThingSpeak.writeField(myChannelNumber, 1, t, myWriteAPIKey);
 ThingSpeak.writeField(myChannelNumber, 2, h, myWriteAPIKey);
 ThingSpeak.writeField(myChannelNumber, 3, distance, myWriteAPIKey);
 ThingSpeak.writeField(myChannelNumber, 4, (float) AmpsRMS, myWriteAPIKey);
 ThingSpeak.writeField(myChannelNumber, 5, Wattage, myWriteAPIKey);
delay(2000);
}
```

Formulas Used to Obtain the Data

For Current Sensor(ACS712)

The Current was run multiple times in 1 second(8000-9000 times) to record maximum and minimum values. This maximum and Minimum value will roughly mark the positive and negative amplitudes of the Provided A.C. Signal.

So 2*Vmax = (max - min)

Vmax = (max- min)/2

Vrms = Vmax/Sqrt(2)

AmpRms = Vrms*1000/mverAmp

mverAmp = 66 (Specific to the 30 Ampere version we are using ACS712T ELC-30A)

Wattage = 220*Vrms

Using these Formulae Current(AmpRms) and Power Consumed(Wattage) can be obtained.

For Ultrasonic Sensor(HCSR04)

The Ultrasonic sensor gives the time delay between the sending and the receiving Pulse.

Tdelay = 2*Distance Travelled/speed of sound

Distance travelled = Tdelay*Speed of sound/2

Tdelay is in microseconds and Distance obtained will be in cm so 10000 is further divided to it.

Speed of Sound is assumed to be 340m/s

Distance travelled = Tdelay*0.0340/2

PROCEDURE/ PLAN OF ACTION

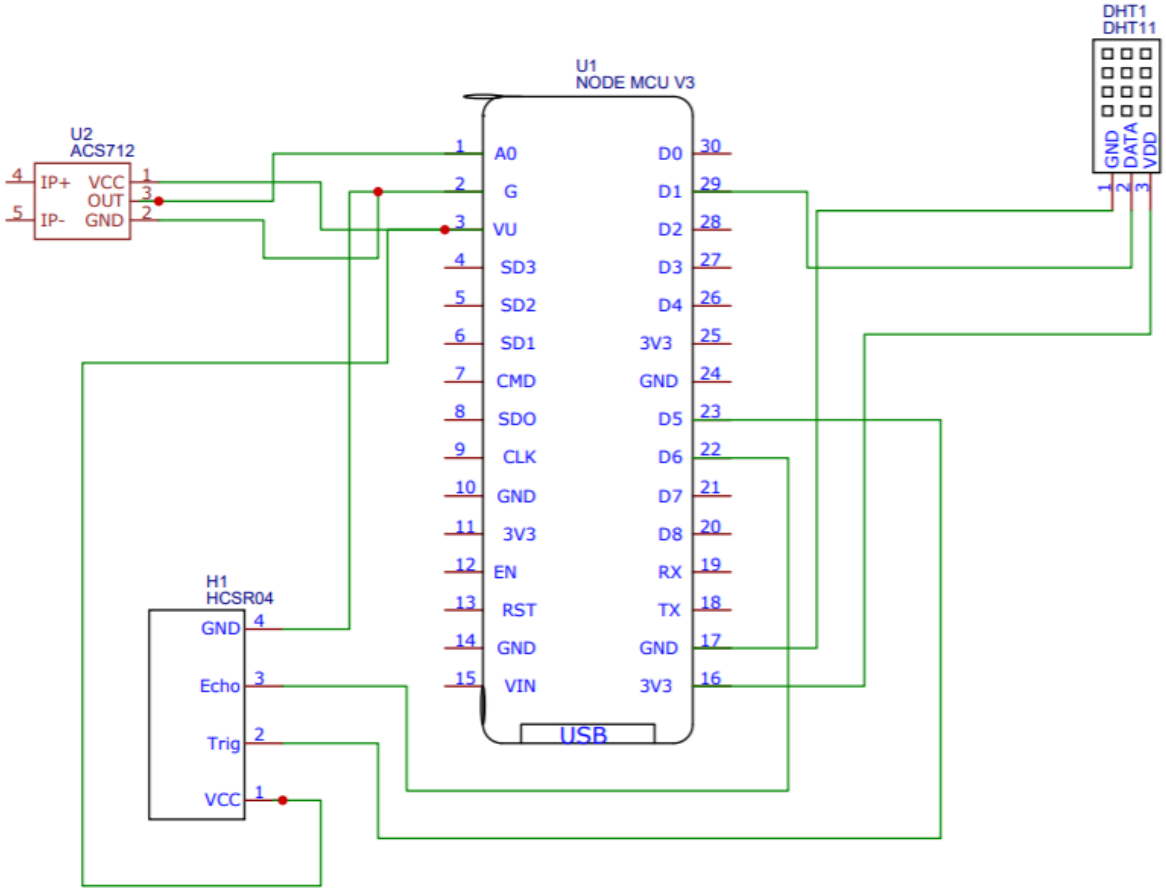
Before starting with our design project of an IoT prototype, Our group conducted a Survey of IoT systems which have been integrated with electric grid systems. From the survey, our group was able to gauge the hardware and software requirements for building a prototype.

The prototype was first designed virtually to get a rough idea of our project. Later, based on the most essential components required to successfully construct a working model electronic components were acquired. The components include the Lolin NodeMCU V3, DHT11, ACS712 and HC-SR04.

To Programme the NodeMCU(WiFi Module), We used Arduino IDE software version 1.8.3. Through the IDE we were able to send data directly to Thingspeak.com, interface various sensors and perform necessary calculations. The sensors and the NodeMCU were physically connected using wires on a breadboard. All sensors except the ACS712, whose output was fed into the ADC pin of the Lolin NodeMCU board, had their output fed to GPIO input pins. The NodeMCU was powered via USB connected to a computer through which its code files were also uploaded.

After all connections were made, the programme code was uploaded to the NodeMCU development board after which data from the sensors were updated on the Thingspeak.com Web application platform. Data is updated on Thingspeak every 20 seconds.

DESIGN SCHEMATIC



ACS712 Pinout:

VCC of ACS712 is connected to the VU (V USB) pin of the nodeMCU which gives +5V supply.

OUT pin of ACS712 is connected to the A0 (ADC) Pin of the NodeMCU. A0 pin is dedicated for Analog input in the Lolin NodeMCU V3 development board.

HCSR04 Pinout:

Echo pin produces a signal when the ultrasonic pulse returns after being reflected. This pin is connected to the D6 port of the microcontroller.

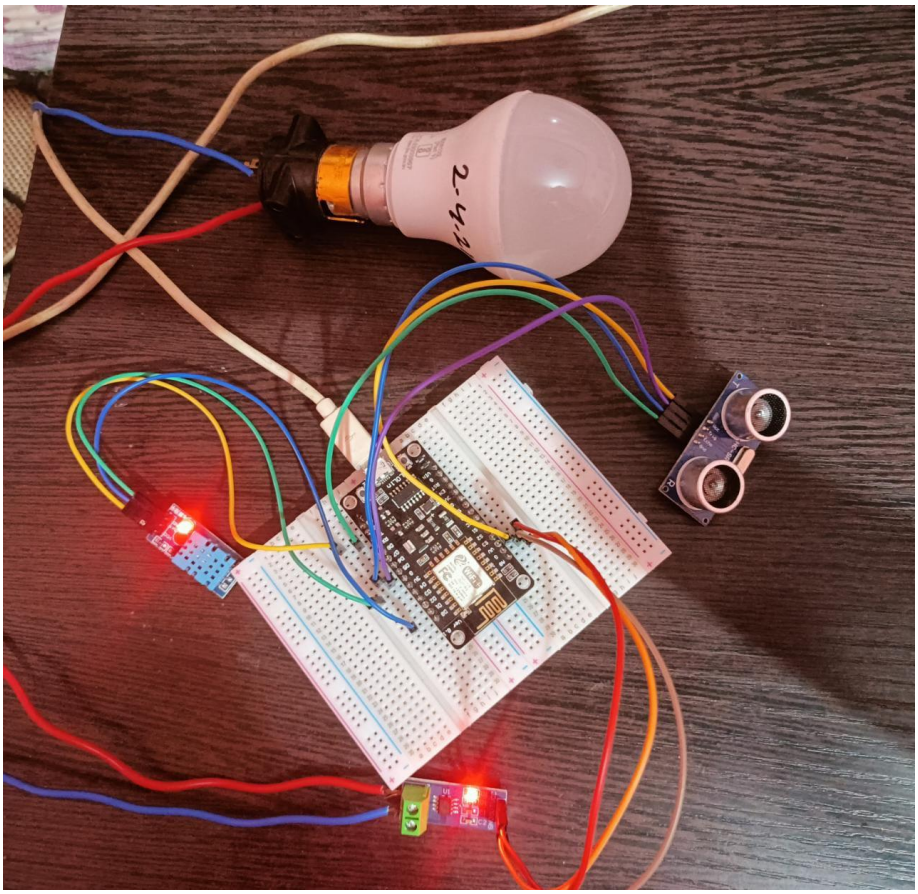
Trig pin is used to send the trigger for sending the ultrasonic pulse. The Trigger pin is connected to the D5 port of the microcontroller.

VCC of HCSR04 is connected to the VU (V USB) pin of the nodeMCU which gives +5V supply.

DHT11 Pinout:

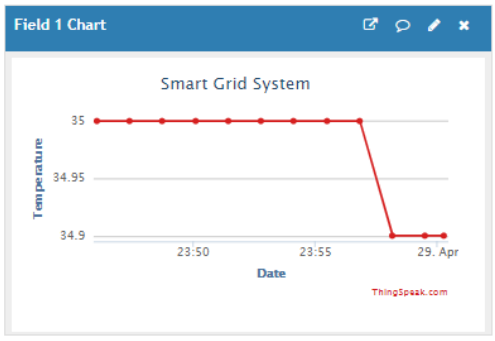
VDD Pin is connected to the 3V3 port of the NodeMCU which supplies 3.3V to the sensor.

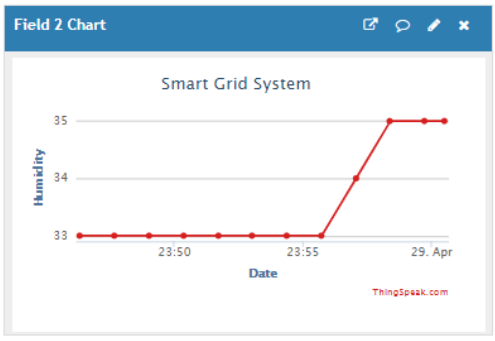
The DATA Pin sends the sensed value of temperature and humidity. This pin is connected to the D1 Data pin of the NodeMCU.

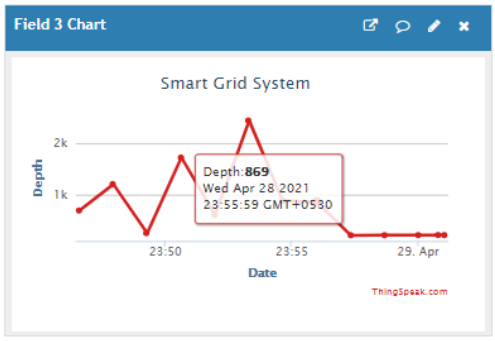


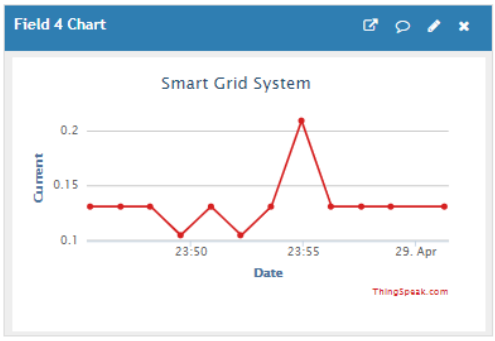
Hardware assembly

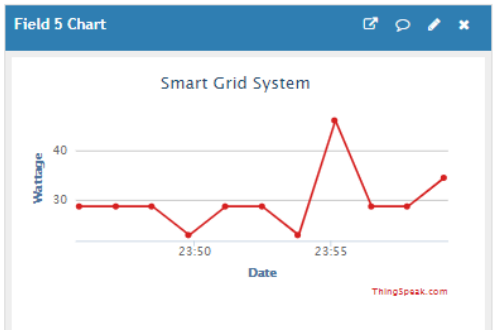
Data Visualisation using ThingSpeak.com











CONCLUSION

In conclusion, this report shows how an IoT system can be designed using sensors, Microprocessors and IoT web services. Our prototype implements basic functions of our ideal IoT system. Lolin NodeMCU V3 with ESP8266 was programmed and used with sensors such as DHT11 temperature and humidity sensor and HC-SR04 ultrasonic sensor can be used to monitor the health of Transformers in Electric grids.

Further Improvement like selecting or building a secure server which can be least prone to any hacking, can be implemented. As the current service which we are using can be prone to any malicious activity due to its very open nature. Anyone can edit the data using the Api key and Channel ID. An alert system can be implemented which will send the user an email notification if the value of the sensors exceed the threshold value. Thingspeak platform can be used for the above purpose. The inbuilt React App of thingSpeak can be used, which will trigger the email response if the data obtained from the matlab analysis crosses the threshold.

The Implementation of Smart grid IoT system can very efficiently save a lot of energy which gets wasted due to poor implementation, electricity theft during transmission. According to a survey by MDPI around $4.5 billion get wasted due to Electricity theft and around 20% of the Total electricity produced get wasted due to such activity. Any such implementation can effectively use countries' resources and money. This will also give a huge control and transparency to Consumers and Authorities to monitor their electricity consumption. At last but not the least, whenever the electricity goes down in our neighbourhood someone has to call the authorities to let them know the situation but an intelligent system can self detect fault and also sends relevant information to the authorities which can really help in increasing the system efficiency, even the transformers health is unnoticeable to them. This could save a lot of resources and help the transformer to last long, Hence, creating a sustainable and efficient system and beautifully describing the fact that ”Energy Saved is Energy Produced!!”.