University of Bahrain
Deanship of Graduate Studies
and Scientific Research

جامعة البحرين
عمادة الدراسات العليا والبحث

Final Project

AC Power Meter

Prepared by: Ghadeer

Contents

Table of Figures:

Introduction:

The Smart Grid, considered as the next generation power grid, is an advanced bi-directional flow of electricity and data that involves the usage of Internet of Things (IoT). The system is self-healing, sustainable, resilient and adaptive with foresight for prediction under various uncertainties to create a broadly distributed automated power supply network [1-2].

IoT devices are utilized in many parts of SG such as, smart grid energy monitoring system, to monitor and control grid statistics for reliable and efficient delivery of power. Also, it can be used to assist consumers and electric power companies in better managing their use and lowering billing costs [3].

Smart meters collect data on energy use and display a complete picture of energy usage in the home, including loads and predicted costs. In this project, we will implement a simple AC power meter using microcontroller esp8266 and acs712 current sensor to measure the current and calculate the power.

Main Components:

ACS712 Hall Effect-Based Linear Current Sensor

ACS712 is a hall-effect linear current sensor IC. It can be used for both AC and DC currents measurements. To operate it requires a 5V supply voltage and it will produce an analog voltage output proportional to the measurement current being sensed [4].

SELECTION GUIDE

Part Number	Packing*	T_A (°C)	Optimized Range, I_P (A)	Sensitivity, Sens (Typ) (mV/A)
ACS712ELCTR-05B-T	Tape and reel, 3000 pieces/reel	–40 to 85	±5	185
ACS712ELCTR-20A-T	Tape and reel, 3000 pieces/reel	–40 to 85	±20	100
ACS712ELCTR-30A-T	Tape and reel, 3000 pieces/reel	–40 to 85	±30	66

Figure 1 ACS712 Selection Guide [4]

Board: ESP8266 NodeMCU WiFi Development Board

NodeMCU is a Firmware distribution that uses the Lua scripting language on Expressif ESP8622 Wi-Fi microcontroller, as well as an open-source hardware development kit built around the ESP-12E WiFi module that itself includes the ESP8266 core processor. It has a CP2102 single chip USB to UART bridge IC for programming and debugging. And also, it can be powered via its micro USB port [4].

NodeMCU Dev Kit shown in the below figure has an onboard register divider network which provides 1.0V from 3.3V to the ADC pin of ESP8266. Hence, we can use a 0–3.3V range for ADC input voltage for below NodeMCU Dev Kit. Since ADC is of 10-bit resolution, it will give 0-1023 value range for ADC input voltage 0-3.3V on Dev Kit [7].

Structure:

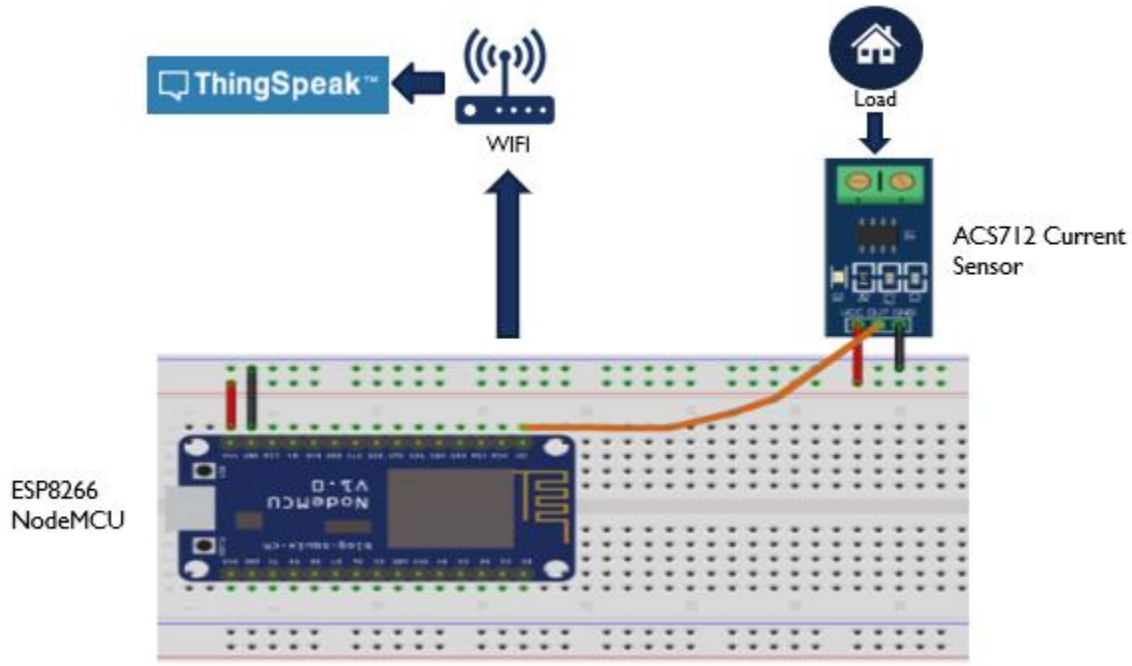

Figure 2 Board and sensor connection [4] & overall connection

Steps:

1. Adding the board

a) Adding esp8266 in Arduino preferences

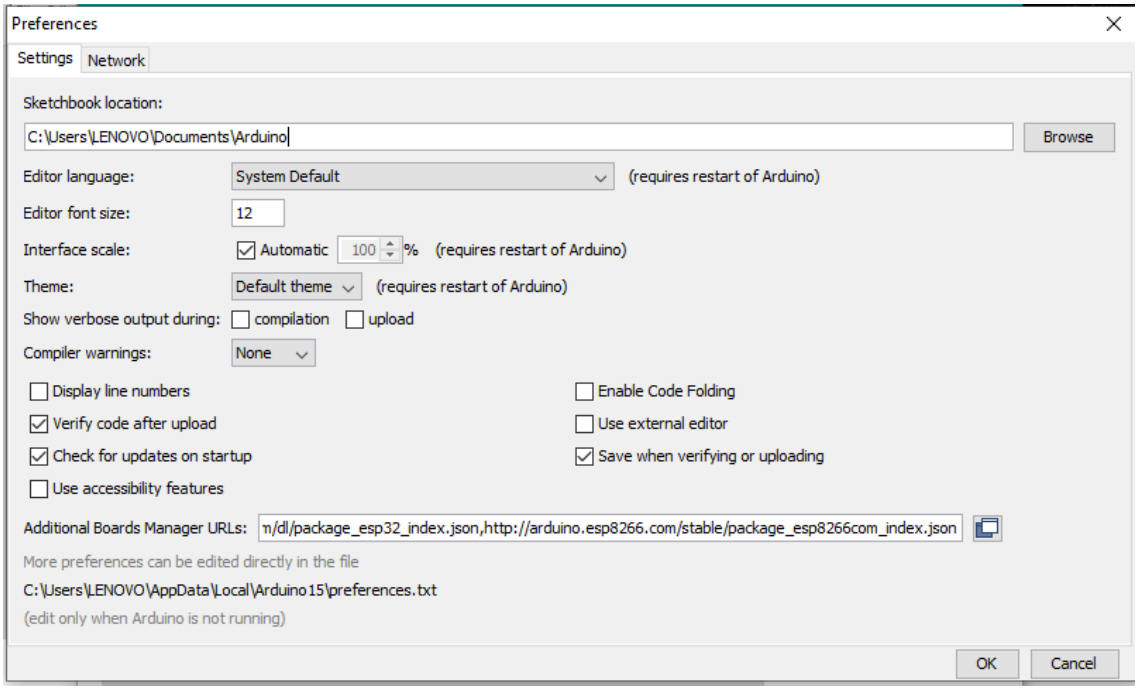

Figure 3 Arduino Preferences

b) Installing the board using “Boards Manager” by searching for “8266”, then click on “Install” button.

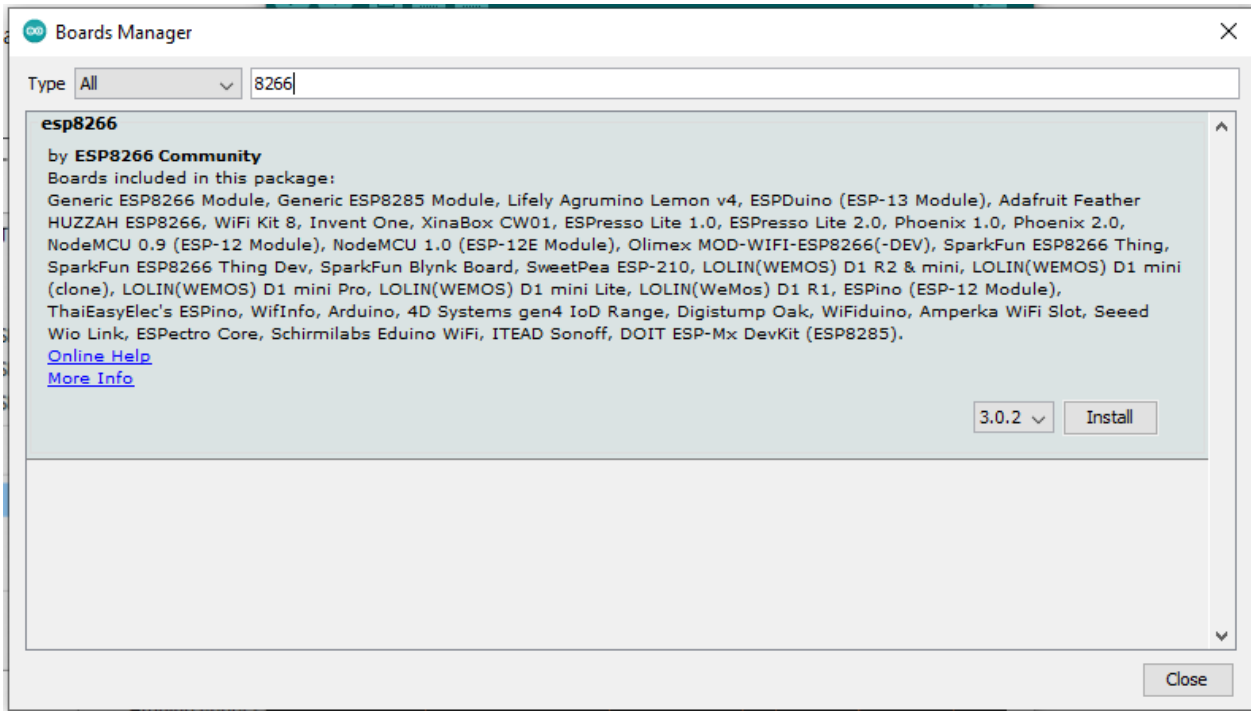

Figure 4 Arduino Board Manager

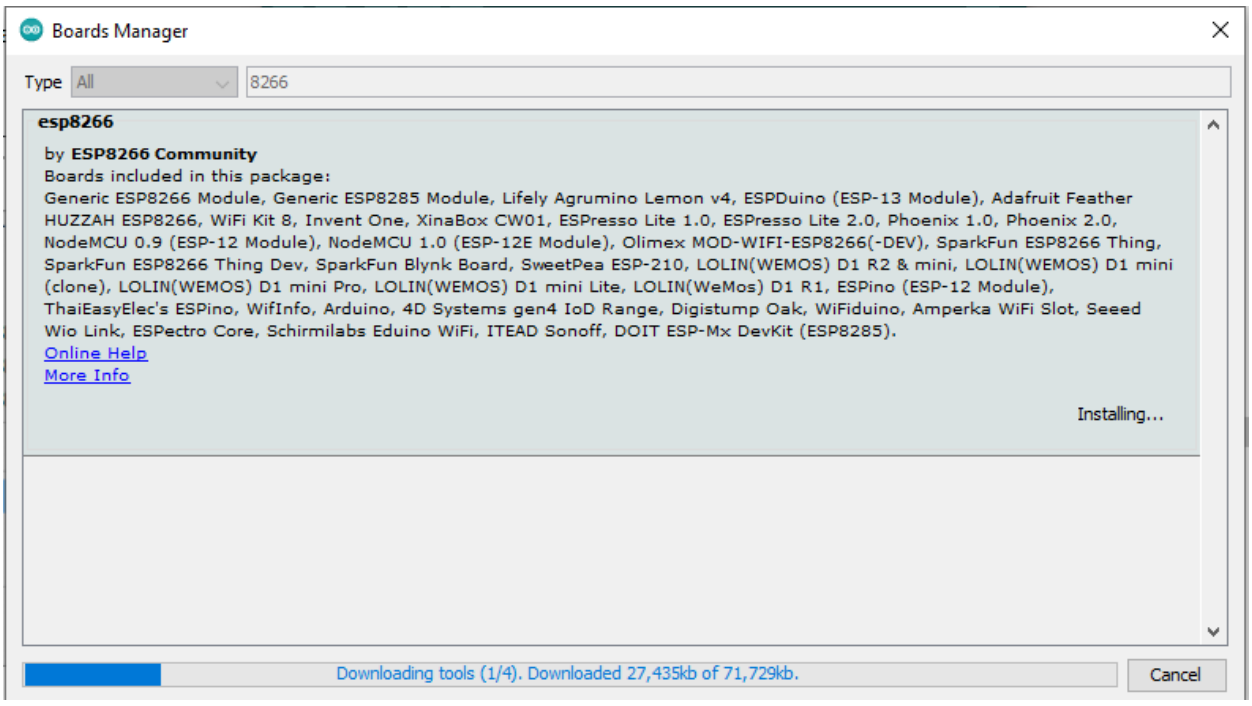

Figure 5 Arduino Board Manager – ESP8266 Package Installation

a) Select the board by going to Tools> Board> NodeMCU 1.0 > ESP8266 Boards

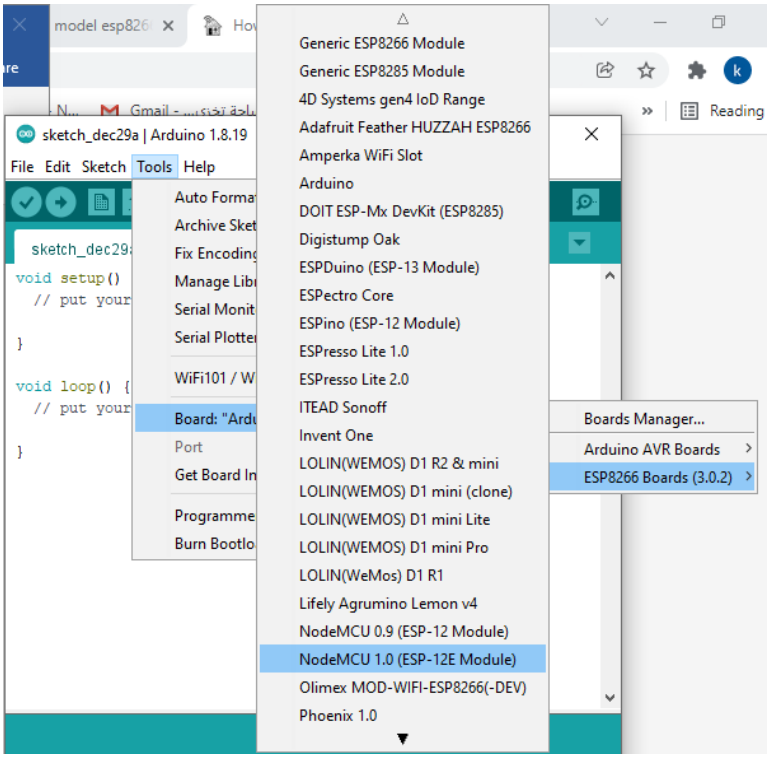

Figure 6 Board Selection

2. Analog Read without a load

The first code will read the analog output voltage of the ACS712 current sensor without any load. If no current passes through the current sensor the analog output voltage of the ACS712 should be 2.5V. The NodeMCU ADC will correlate the analog output voltage 2.5V with an ADC value, which turns out to be approx. 776 [4].

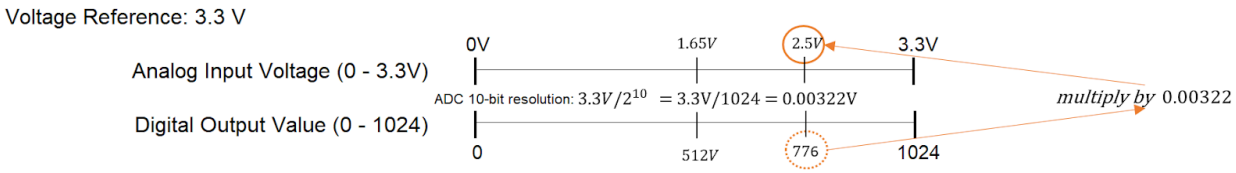

Figure 7 Voltage Reference [4]

Code:

```
/*
* AnalogRead Sketch using ACS712 and NodeMCU ESP8266 ESP-12E module
* by Alex Roman
*/
```

```
/*--------VARIABLES--------*/

/*--------NodeMCU--------*/

#define PIN A0

float resolution  = 3.3 / 1024;                // Input Voltage Range is 1V to 3.3V

                                        // ESP8266 ADC resolution is 10-bit. 2^10 = 1024

void setup()

{

 Serial.begin(115200);                      // Initialize Serial communication

 pinMode(PIN, INPUT);                        // Set pin A0 as read.

}

void loop() {

 Serial.print("analogRead = ");

 Serial.println(analogRead(PIN));              // Function to read from pin A0

 Serial.print("Voltage (when zero current) = ");

 Serial.print(analogRead(PIN)* resolution, 3);   // ADC value multp. by resolution will give your the corresponding voltage value.

 Serial.println(" V");

 delay(1000);

}
```

Output:

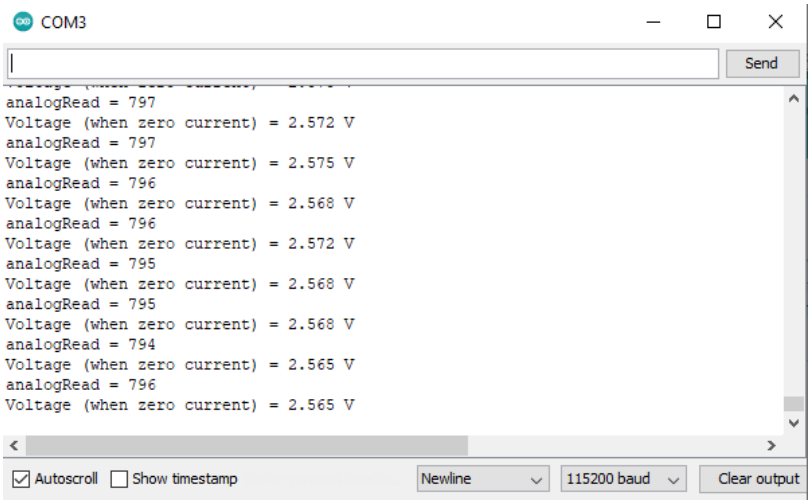

Figure 8 Serial Monitor

3. Installing ThingSpeak library

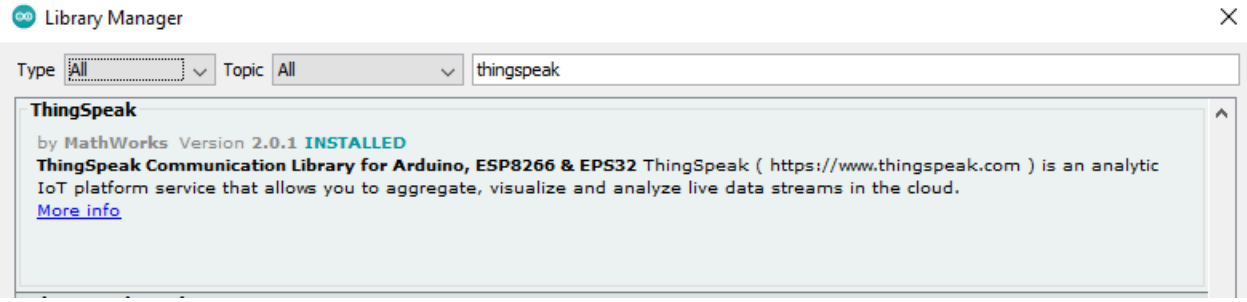

Figure 9 Library Manager

4. Create an account in ThingSpeak

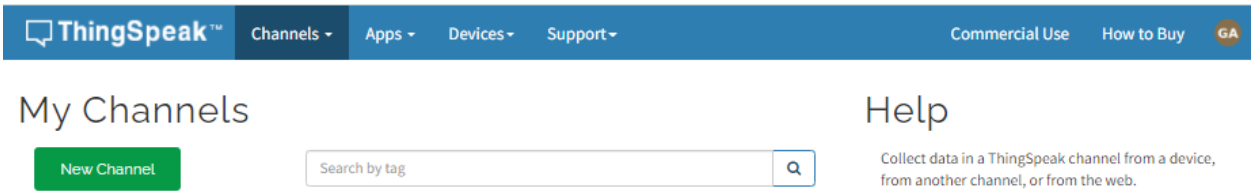

Figure 10 ThingSpeak Channel page

5. Create new channel and add the fields

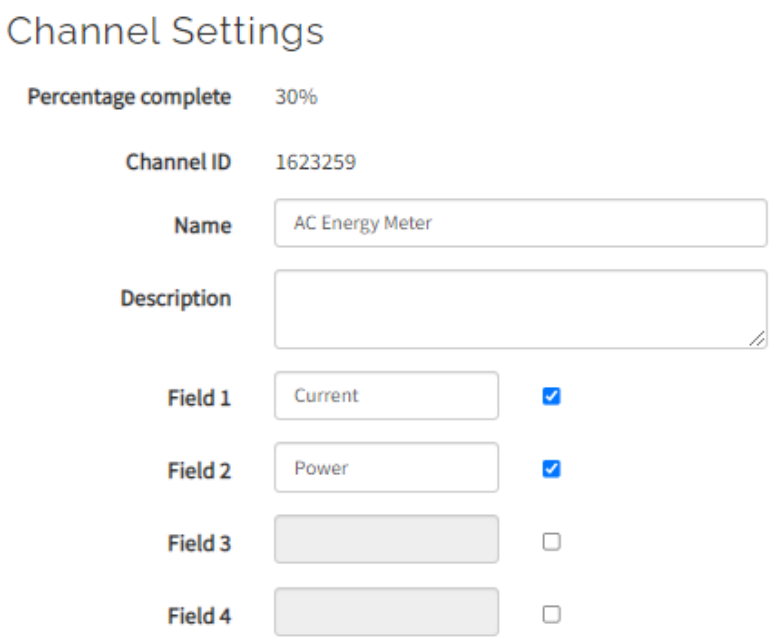

Figure 11 Channel Settings

6. Open Notepad, cerate a new file and save it as Secrets.h
7. Add WIFI network name & password in Secrets.h file

```
// Use this file to store all of the private credentials
// and connection details
```

#define SECRET_SSID "MySSID" // replace MySSID with your WiFi network name
#define SECRET_PASS "MyPassword" // replace MyPassword with your WiFi password

8. From API keys tab, copy “Channel ID” and “Key” and put it in Secrets.h file

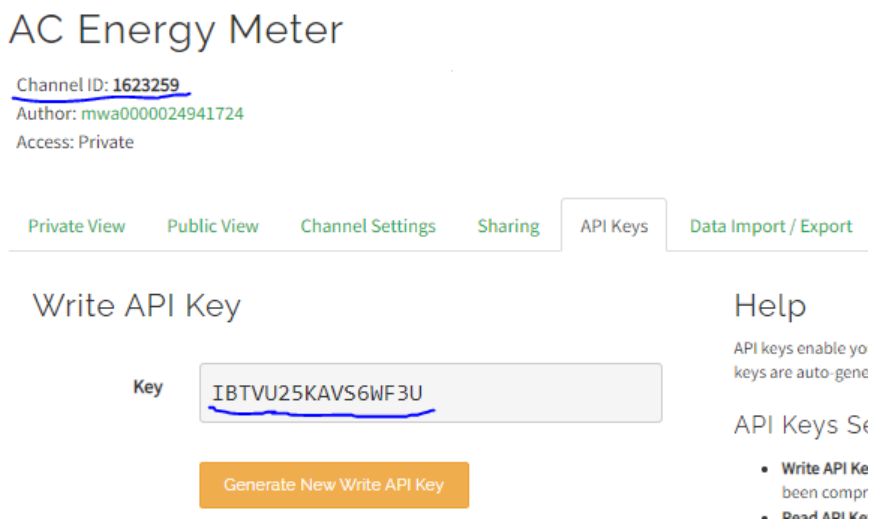

Figure 12 API Key Tab

9. Add WIFI network name & password in Secrets.h file
#define SECRET_CH_ID 0000000 // replace 0000000 with your channel number
#define SECRET_WRITE_APIKEY "XYZ" // replace XYZ with your channel write API Key

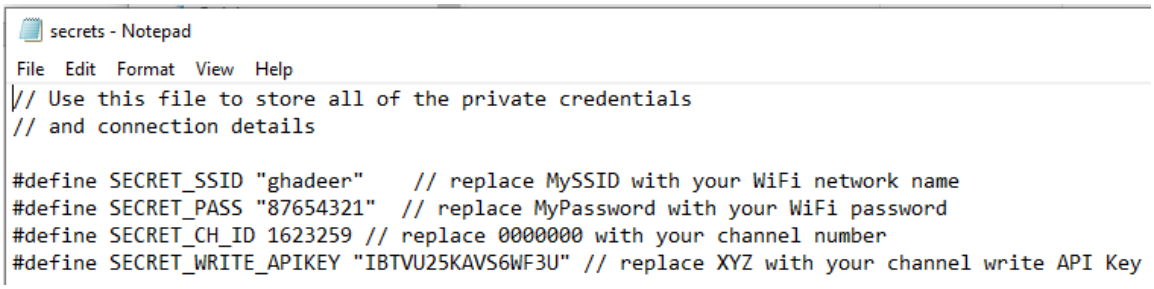

Figure 13 secrets.h file

10. Add widget to show the measured values

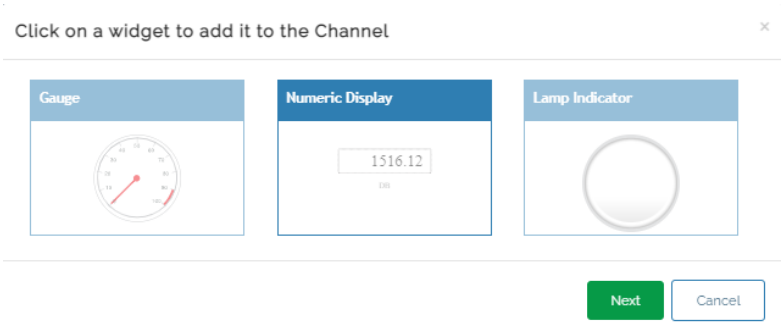

Figure 14 Add Widget popup

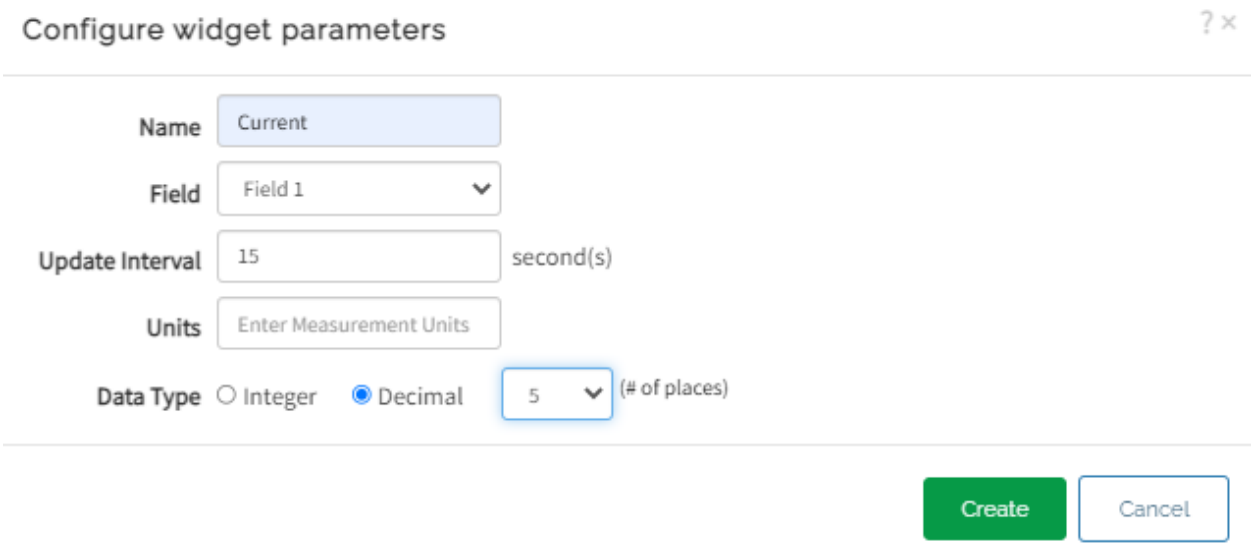

Figure 15 Configure Widget Parameters Popup

11. The root mean square (RMS or rms) of a periodic waveform: Vrms/Irms

The r.m.s. value, also known as the effective value, is the amount of heat an ac current produces across a resistance compared with a steady current when passed through the same resistance at the same period of time. We are going to modify the first code to obtain the effective voltage and current values, i.e. Vrms and Irms.

Final code:

```
/*
 * Vrms & Irms values from analogRead A0 using ACS712 and NodeMCU ESP8266 ESP-12E module.
 * Plus WiFi connection to an access point.
 * And ThingSpeak IoT platform Analysis access.
 * code constructed with reference to Alex Roman & https://www.youtube.com/watch?v=AK4Lp1Ko0l8 [4] [6]
 */
/*--------LIBRARIES--------*/
#include <ESP8266WiFi.h>                    // Need to add the ESP8266WiFi.h library
#include "secrets.h"                        // Also add the secrets.h file (The one saved in the same folder location as this scrip).
#include "ThingSpeak.h"
/*--------VARIABLES--------*/
/*--------NodeMCU--------*/
```

```
#define PIN A0

float resolution  = 3.3 / 1024;                 // Input Voltage Range is 1V to 3.3V

                                        // ESP8266 ADC resolution is 10-bit. 2^10 = 1024

uint32_t period = 1000000 / 50;                 // One period of a 50Hz periodic waveform

uint32_t t_start = 0;

// setup

float zero_ADC_Value = 0;

// loop

int cnt = 0;

float ADC = 0, Vrms = 0, Current = 0, Q = 0.000, c = 0;

float sensitivity = 0.66;                       // 185 mV/A, 100 mV/A and 0.66 mV/A for ±5A, ±20A and ±30A
current range respectively.

/*--------WiFi--------*/

char ssid[] = SECRET_SSID;                      // your network SSID (name)

char pass[] = SECRET_PASS;                      // your network password

int keyIndex = 0;                               // your network key Index number (needed only for WEP)

WiFiClient  client;                             // Object

/*--------ThingSpeak--------*/

unsigned long myChannelNumber = SECRET_CH_ID;       // your ThingSpeak Channel ID

const char * myWriteAPIKey = SECRET_WRITE_APIKEY;   // your ThingSpeak Channel API key

String myStatus = "";

void setup()

{

 //Serial.begin(115200);                         // Initialize Serial communication

 pinMode(PIN, INPUT);                           // Set pin A0 as read.

 /*--------WiFi--------*/

 //Serial.println();

 WiFi.begin(ssid, pass);                // Initializes the WiFi library's network settings

 /*--------NodeMCU--------*/
```

```
 t_start = micros();
 uint32_t ADC_SUM = 0, n = 0;
 while(micros() - t_start < period) {
  ADC_SUM += analogRead(PIN);
  n++;
 }
 zero_ADC_Value = ADC_SUM / n;          // The avg analog value when no current pass through the ACS712 sensor
 /*--------ThingSpeak--------*/
 ThingSpeak.begin(client);                      // Initialize ThingSpeak
}
void loop() {
 /*----Vrms & Irms Calculation----*/
 t_start = micros();
 uint32_t ADC_Dif = 0, ADC_SUM = 0, m = 0;
 while(micros() - t_start < period) {          // Defining one period of the waveform. Bahrain frequency(f) is 50Hz. Period = 1/f = 0.02 sec
  ADC_Dif = zero_ADC_Value - analogRead(PIN);   // To start from 0V we need to subtracting our initial value when no current passes through the current sensor, (i.e. 750 or 2.5V).
  ADC_SUM += ADC_Dif * ADC_Dif;               // SUM of the square
  m++;                                  // counter to be used for avg.
 }
 ADC = sqrt(ADC_SUM / m);                    // The root-mean-square ADC value.
 Vrms = ADC * resolution ;                  // The root-mean-square analog voltage value.
 Current = (Vrms  / sensitivity) - Q;       // The root-mean-square analog current value. Note: Q
ThingSpeak.setField(1, Current);                // Current
ThingSpeak.setField(2, Current*230);             // Power= current * Bahrain voltage
ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);
 delay(1000);
}
```

Results:

Connected to mobile hotspot:

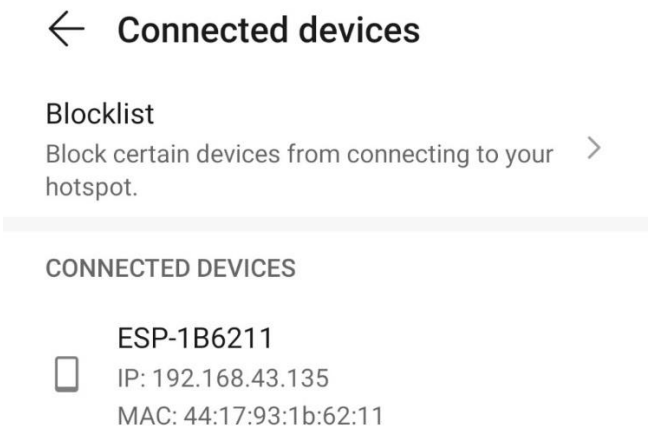

Figure 16 Personal Hotspot Connected Devices

First, doing some testing using 9 volt battery & a diode

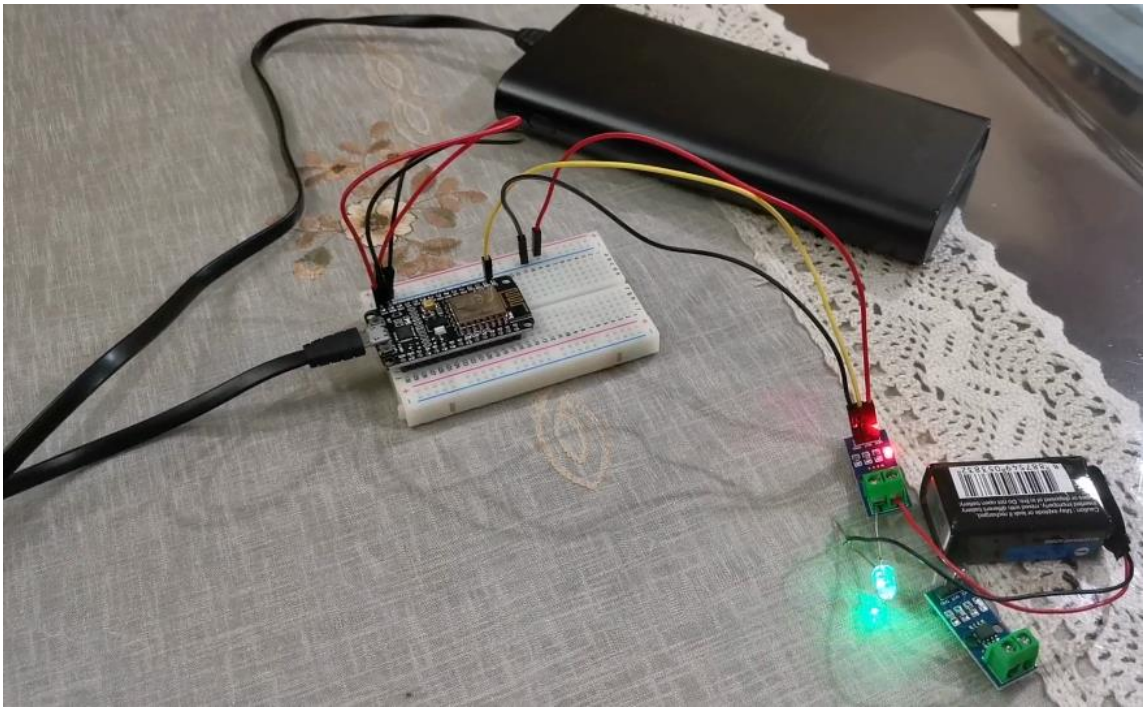

Figure 17 DC Connection

Diodes wattage: 0.1-1 Watt

Battery voltage: 9 volt

Current: power/voltage= 1/9= 0.111 Amp which is similar to the current that got measures

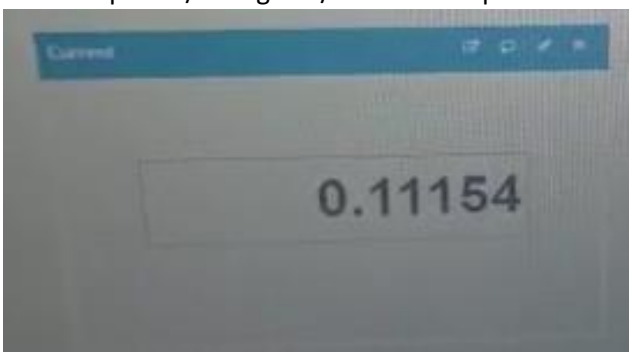

Figure 18 ThingSpeak Showing Current Reading

Then, tried it with a load connected to AC power

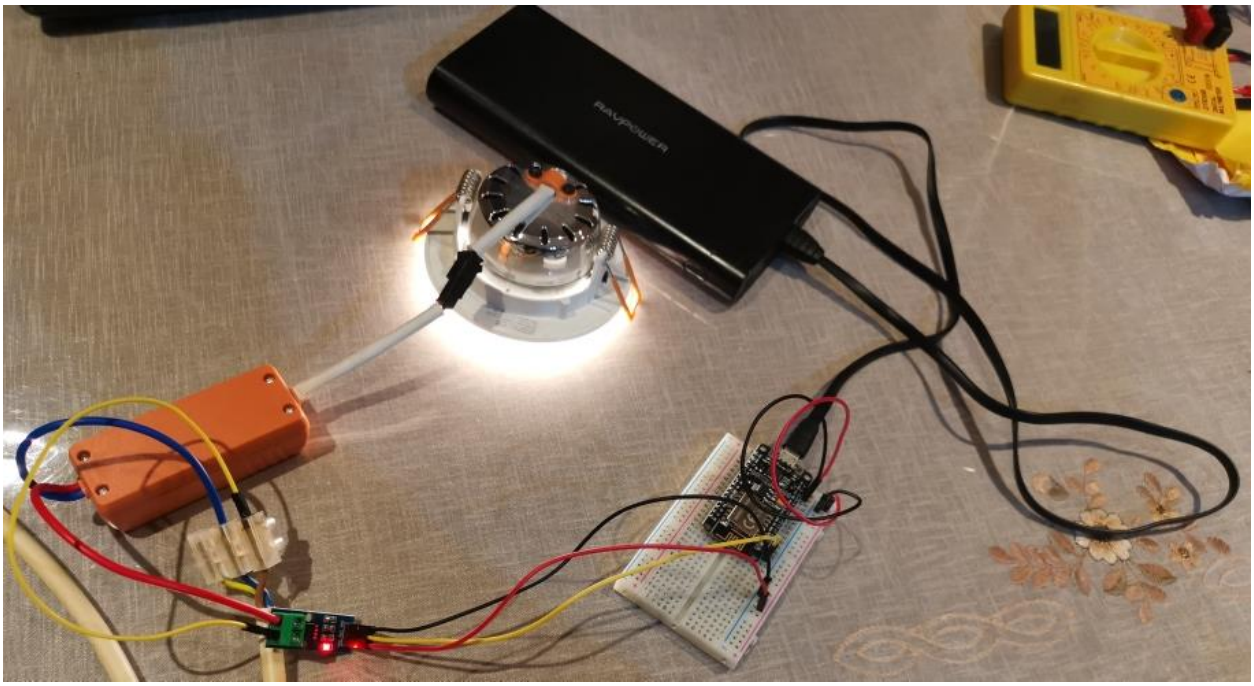

Figure 19 AC Connection

Light specifications:

Light wattage: 5 Watt

Figure 20 Light Specification

Bahrain voltage (approximately): 230 volt

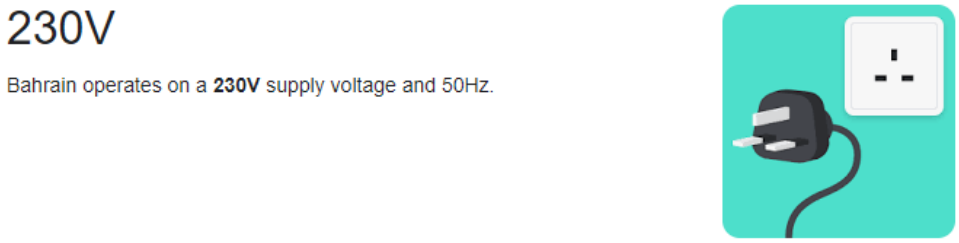

Figure 21 Bahrain Voltage & Frequency [8]

Current: power/voltage= 5/230= 0.0217 Amp which is kind of close to the current that got measures

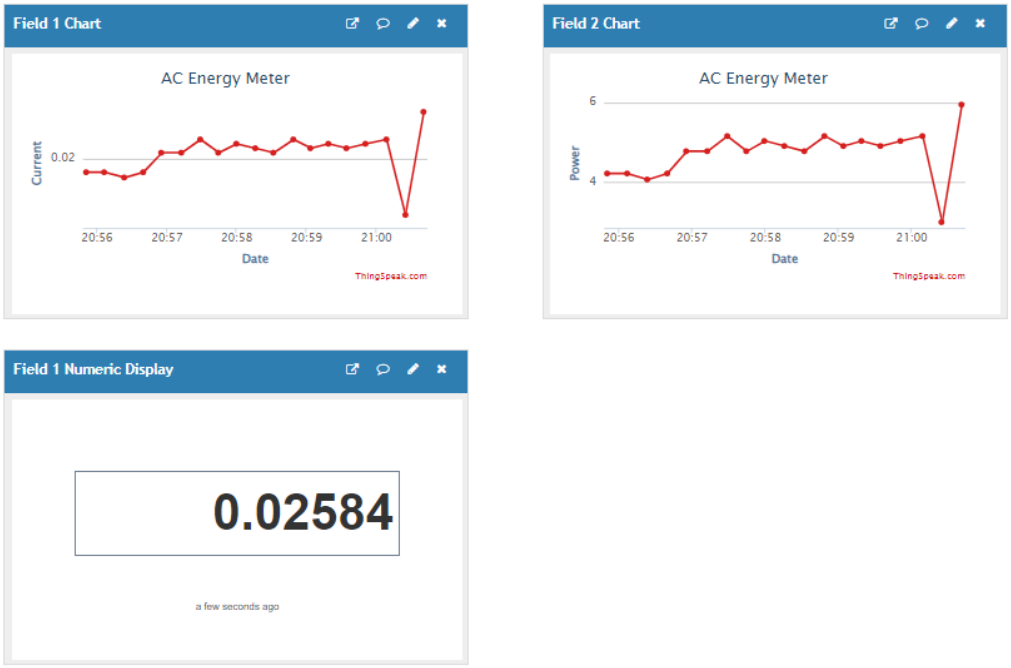

Figure 22 ThingSpeak Showing Readings

Implementation using ACS712 library

Implementation using ACS712 library

1. Install ACS712 library

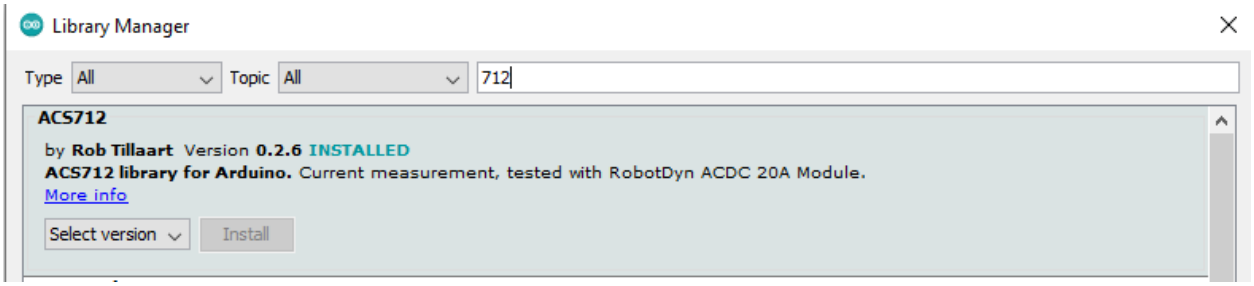

Figure 23 Install ACS712 Library from Library Manager

Code:

```
//  Code constructed with reference to Alex Roman & https://www.youtube.com/watch?v=AK4Lp1Ko0l8 & Rob Tillaart from library examples https://github.com/RobTillaart/ACS712 [4-6]

/*--------LIBRARIES--------*/

#include <ESP8266WiFi.h>                    // Add the ESP8266WiFi.h library

#include "secrets.h"                        // Add the secrets.h

#include "ThingSpeak.h"

#include "ACS712.h"

// ESP8266 has 3.3 volt with a max ADC value of 1023 steps

// ACS712 5A  uses 185 mV per A

// ACS712 20A uses 100 mV per A

// ACS712 30A uses  66 mV per A

ACS712  ACS(A0, 3.3, 1023, 66);

/*--------WiFi--------*/

char ssid[] = SECRET_SSID;                  // your network SSID (name)

char pass[] = SECRET_PASS;                  // your network password

int keyIndex = 0;                           // your network key Index number (needed only for WEP)

WiFiClient  client;                         // Object

/*--------ThingSpeak--------*/

unsigned long myChannelNumber = SECRET_CH_ID;      // your ThingSpeak Channel ID

const char * myWriteAPIKey = SECRET_WRITE_APIKEY;   // your ThingSpeak Channel API key

void setup()

{

 /*--------WiFi--------*/

 WiFi.begin(ssid, pass);              // Initializes the WiFi library's network settings and provides the current status.

 /*--------ThingSpeak--------*/

 ThingSpeak.begin(client);                  // Initialize ThingSpeak

}
```

```
void loop()
{
 float mA = ACS.mA_AC();
 ThingSpeak.setField(1, mA);                  // Current
 ThingSpeak.setField(2, mA/1000*230);              // Power= current * Bahrain voltage
 ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);
 delay(1000);
}
```

Output:

Note: the current unit is mA and power in Watt

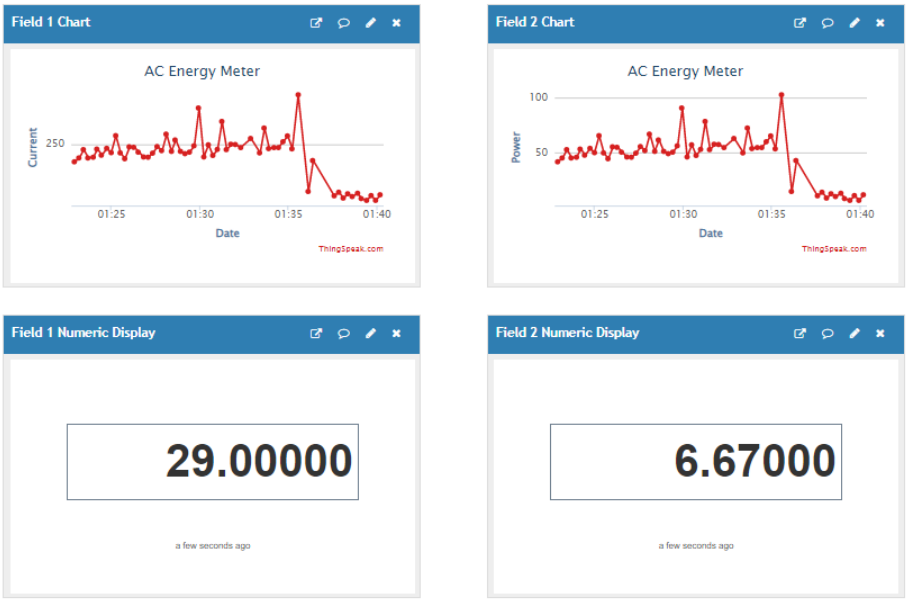

Figure 24 ThingSpeak Showing Current & Power Reading

Challenges:

1. Time limit
2. Delay in component delivery
3. It's been a while since I did a project using a microcontroller
4. First time working with esp8826, the current sensor and AC voltage in this type of project (burned wires,..).

Conclusion:

In this project, an AC energy meter was implemented using microcontroller esp8266 and acs712 current sensor to measure the current. Then, real-time data is going to be gathered and sent to 'ThingSpeak'

platform, which was used to visualized, analyze and monitor power usage in real time. This can assist consumers in better managing their use and lowering billing costs or even utilities can be given access to it to monitor consumers usage and behavior which could help in planning.

References:

1. G. Dileep, “A survey on Smart Grid Technologies and Applications,” Renewable Energy, vol. 146, pp. 2589–2625, 2020.
2. X. Fang, S. Misra, G. Xue, and D. Yang, “Smart Grid — the new and improved Power Grid: A survey,” IEEE Communications Surveys & Tutorials, vol. 14, no. 4, pp. 944–980, 2012.
3. F. Khan, M. A. Siddiqui, A. U. Rehman, J. Khan, M. T. Asad, and A. Asad, “IOT based power monitoring system for Smart Grid Applications,” 2020 International Conference on Engineering and Emerging Technologies (ICEET), 2020.
4. A. Roman, “AC/DC current measurement using ACS712 sensor and NODEMCU ESP-12E micro-controller.,” AC/DC current measurement using ACS712 sensor and NodeMCU ESP-12E micro-controller., 04-Oct-2021. [Online]. Available: https://www.romn.io/2020/05/a-simple-approach-to-measure-acdc.html. [Accessed: 02-Jan-2022].
5. R. Tillaart, “RobTillaart/ACS712: Arduino Library for ACS current sensor - 5A, 20A, 30A,” GitHub. [Online]. Available: https://github.com/RobTillaart/ACS712. [Accessed: 02-Jan-2022].
6. Paul, “How to use ThingSpeak with Arduino,” YouTube, 27-Feb-2017. [Online]. Available: https://www.youtube.com/watch?v=AK4Lp1Ko0l8. [Accessed: 02-Jan-2022].
7. “Nodemcu ADC with Arduino Ide: Nodemcu,” ElectronicWings. [Online]. Available: https://www.electronicwings.com/nodemcu/nodemcu-adc-with-arduino-ide#:~:text=Introduction,voltage%20from%20an%20external%20device. [Accessed: 02-Jan-2022].
8. “Travel Adaptor for Bahrain,” Electrical Safety First. [Online]. Available: https://www.electricalsafetyfirst.org.uk/guidance/advice-for-you/when-travelling/travel-adaptor-for-bahrain/#:~:text=Bahrain%20operates%20on%20a%20230V%20supply%20voltage%20and%2050Hz. [Accessed: 09-Jan-2022].