15IT422E - Internet of Things

Semester - VI

Project Report

Title: Power Usage Monitoring

Name: G Harshith Mohan

Register No.: RA1611008010014

Department of Information Technology

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

S.R.M NAGAR, KATTANKULATHUR - 603 203

KANCHEEPURAM DISTRICT

LINKS TO GITHUB AND YOUTUBE:

YouTube:

https://www.youtube.com/watch?v=aIQ10rU7Qxw

Github:

https://github.com/harshithmohan/power-usage-monitoring

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to our IoT Professor Dr. Kayalvizhi Jayavel for being out there at every step of my course and guiding us all along the way to be capable of molding our ideas into smart projects and helping us enhance our skills.

I would also like to thank my parents, without whom I wouldn't be able to anything in any regard.

ABSTRACT

Lighting is one the most power-hungry component anywhere across globe both in commercial as well as domestic use. It is the biggest contributor in energy bills to the state exchanger for operating & maintaining the street light across the city or state therefore it becomes the choice for implementing latest energy efficient technologies. City administrator are open and key to adopt new products & technologies which will help them save energy & reduce their power bills significantly.

The light industry is already introducing & implementing energy efficient LED technologies which reduce energy consumption significantly. This would further cut down by the use of cutting-edge Internet of Things concept. We have developed IoT based wireless controller & software platform to control & manage streetlights or commercial facilities lights. We use platform state of the art and wireless mesh technology to convert these lights locally and then connect to the cloud via some Wi-Fi internet gateway. The data on the cloud platform helps the user or concerned authority to monitor, control & manage these assets in real time and cut down energy bills by efficient use through smart scheme usage or traffic pattern.

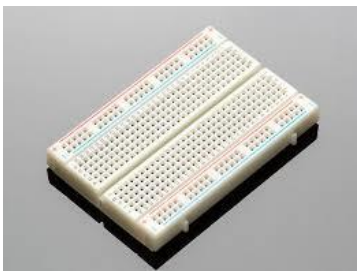
HARDWARE REQUIRED:

- NodeMCU (Rs.350)

- ACS712 Current Sensor (Rs.250)

- Jumper Wires **(**Rs.30)

- Breadboard (Rs.60)



SOFTWARE REQUIRED:

- Arduino IDE
- Adafruit IO
- Libraries: ESP8266, Adafruit_MQTT, Adafruit_MQTT_Client

TOTAL COST OF COMPONENTS: - ~Rs.700

- NodeMCU- Rs.350
- Current Sensor (ACS712) - Rs.250
- Jumper wires- Rs.30
- Breadboard – Rs.60

SYSTEM OVERVIEW

The main component of the setup is the NodeMCU ESP8266 module. All the other hardware components are connected to the NodeMCU. The microcontroller is programmed in Arduino IDE and uses the ACS712 (Current Sensor), is the only other hardware component. Adafruit MQTT Library has been added to the Arduino IDE.

The NodeMCU is powered using a micro USB cable. The ACS712 current sensor is directly connected to the NodeMCU. An AC circuit is connected the current sensor and whenever that circuit pulls current, the sensor calculates the voltage and sends it to the NodeMCU.

All the current RMS values are uploaded to the Adafruit MQTT feed and a graph is plotted with them.

The ACS712 current sensor sends the only the current value of voltage to the NodeMCU. We need to calculate the rest.

First, we get the maximum and minimum value of voltage over a period of time, for example, 1 second.

Then we calculate

Peak-to-Peak Voltage, $V_{pp} = V_{max} - V_{min}$

Peak Voltage, $V_p = {}^{V_{pp}}/_{2}$

RMS Voltage, $V_{rms} = {}^{V_p}/_{\sqrt{2}}$

Each ACS712 has a ${}^{mV}/_{A}$ value. It is 185, 100, 66 mv/A for 5A, 10A and 20A variant respectively.

To get RMS Current, we calculate

RMS Current, $I_{rms} = \frac{V_{rms} \times 1000}{mV\ per\ Ampere}$

This is the value we need.

Programming NodeMCU, setting up Adafruit Dashboard and sending values to Firebase:-

- Make all the necessary circuit connections.
- Import ESP8266WiFi.h, Adafruit MQTT, Adafruit MQTT Client Libraries to the Arduino IDE.
- Refer the Adafruit MQTT example and copy the functions to connect to WiFi and Adafruit
- Upload current sensor code to your NodeMCU.
- Setup your Adafruit dashboard, add a new graph and give it name.
- Once the code is running open COM3 port where all your readings are shown. Those readings are also updated to the Adafruit MQTT.

CODE

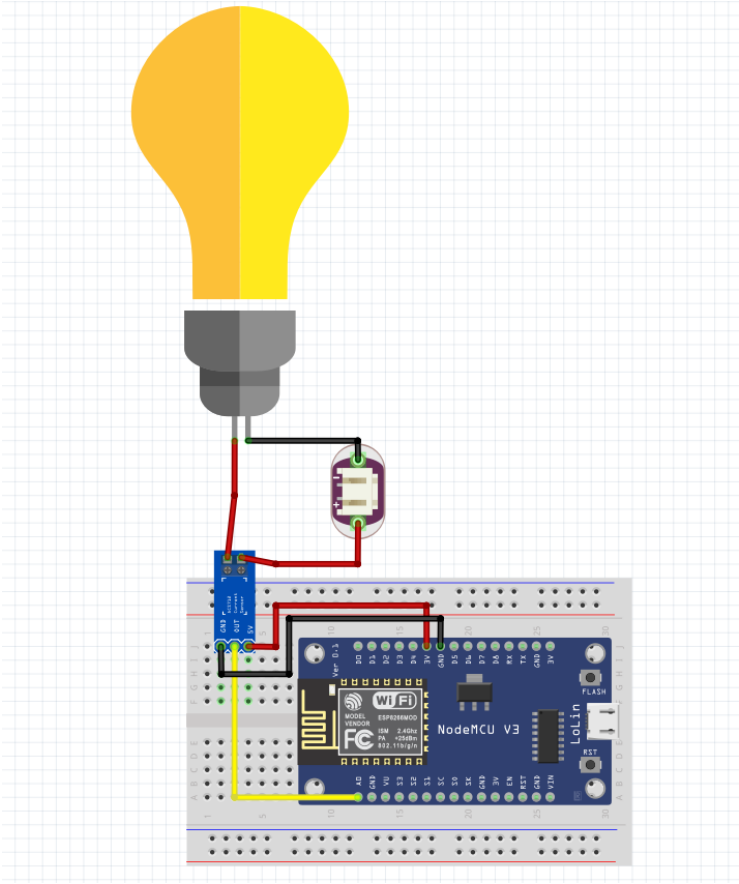
```
#include <ESP8266WiFi.h>
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"

#define WLAN_SSID       "PocoF1"
#define WLAN_PASS       "12345678"

#define AIO_SERVER          "io.adafruit.com"
#define AIO_SERVERPORT    1883                   // use 8883 for SSL
#define AIO_USERNAME        "mohan226"
#define AIO_KEY             "9dd4854894a844ebafae467ae17ac854"

WiFiClient client;
Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT,
AIO_USERNAME, AIO_KEY);
Adafruit_MQTT_Publish photocell = Adafruit_MQTT_Publish(&mqtt,
AIO_USERNAME "/feeds/current2");

const int sensorIn = A0;
int mVperAmp = 100; //185mV for 5A, 100mV for 10A, 66mV for 20A
double Vpp = 0;
double Vp = 0;
double Vrms = 0;
double Irms = 0;

void setup() {
   Serial.begin(9600);
   WiFi.begin(WLAN_SSID, WLAN_PASS);
   while(WiFi.status()!=WL_CONNECTED)
   {
      Serial.println("Connecting...");
      delay(1000);
   }
   Serial.println("Connected");
}

void loop() {
   MQTT_connect();
   Vpp = getVPP();
   Vp = Vpp/2.0;
   Vrms = Vp*0.707;
   Irms = ((Vrms * 1000)/mVperAmp) - 0.05;
```

```
    Serial.print(Irms);
    Serial.println(" Amps");
    if (! photocell.publish(Irms))
    {
       Serial.println("Failed");
    }
    else
    {
       Serial.println("OK!");
    }
    delay(2000);
}

double getVPP()
{
    double result;

    int readValue;
    int maxValue = 0;
    int minValue = 1024;

      uint32_t start_time = millis();
      while((millis()-start_time) < 1000) //sample for 1 Sec
      {
            readValue = analogRead(sensorIn);
            // see if you have a new maxValue
            if (readValue > maxValue)
            {
                  maxValue = readValue;
            }
            if (readValue < minValue)
            {
                  minValue = readValue;
            }
      }
      result = ((maxValue - minValue) * 5.0)/1024.0;
      return result;
  }

void MQTT_connect() {
    int8_t ret;

    if (mqtt.connected()) {
       return;
```

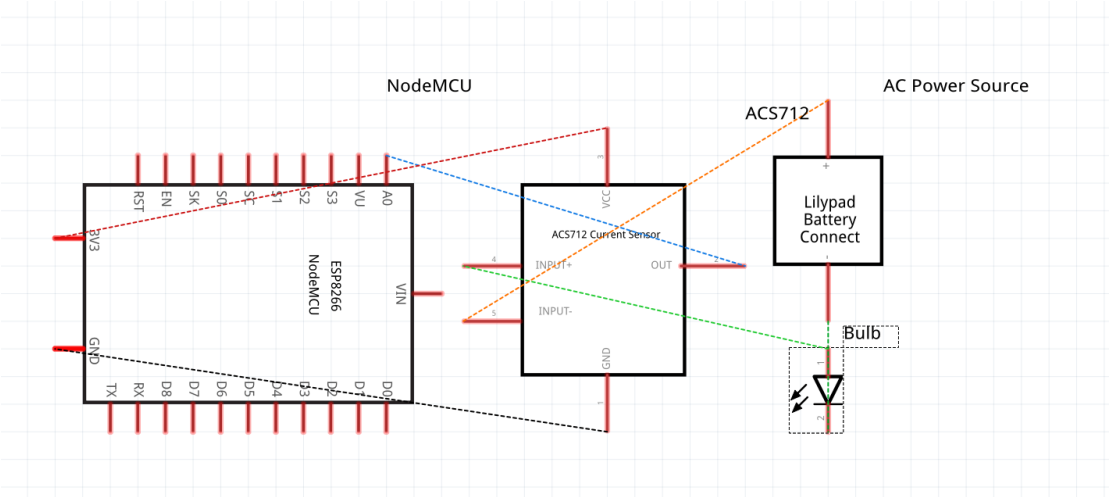
```
    }

    Serial.print("Connecting to MQTT... ");

    uint8_t retries = 3;
    while ((ret = mqtt.connect()) != 0) { // connect will return 0 for connected
           Serial.println(mqtt.connectErrorString(ret));
           Serial.println("Retrying MQTT connection in 5 seconds...");
           mqtt.disconnect();
           delay(5000);    // wait 5 seconds
           retries--;
           if (retries == 0) {
              // basically die and wait for WDT to reset me
              while (1);
           }
    }
    Serial.println("MQTT Connected!");
}
```

IMAGES

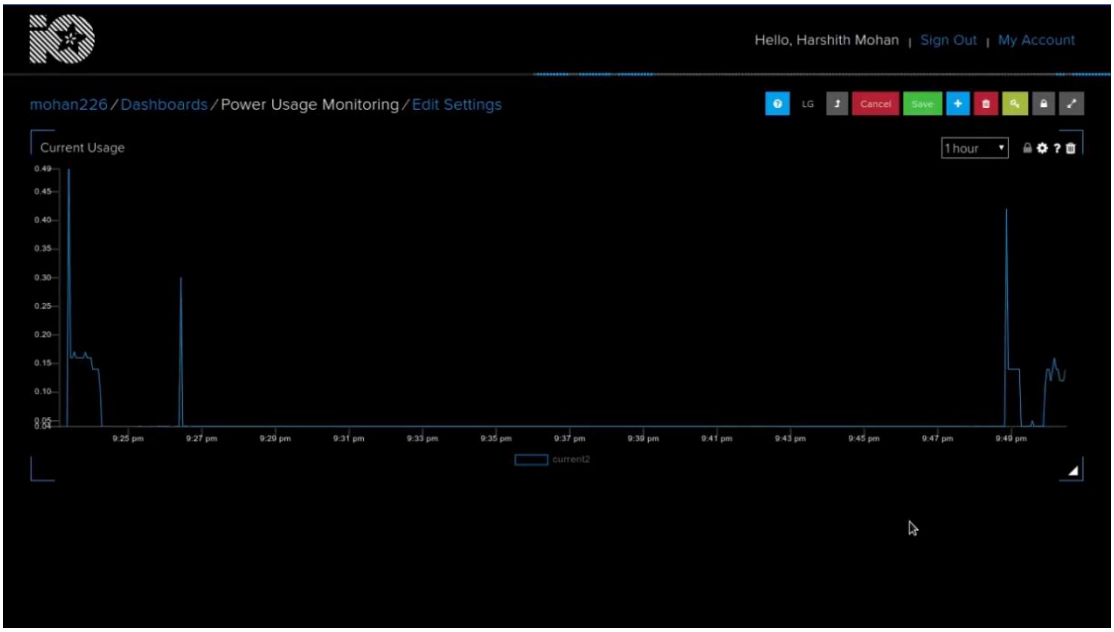
Circuit Diagram



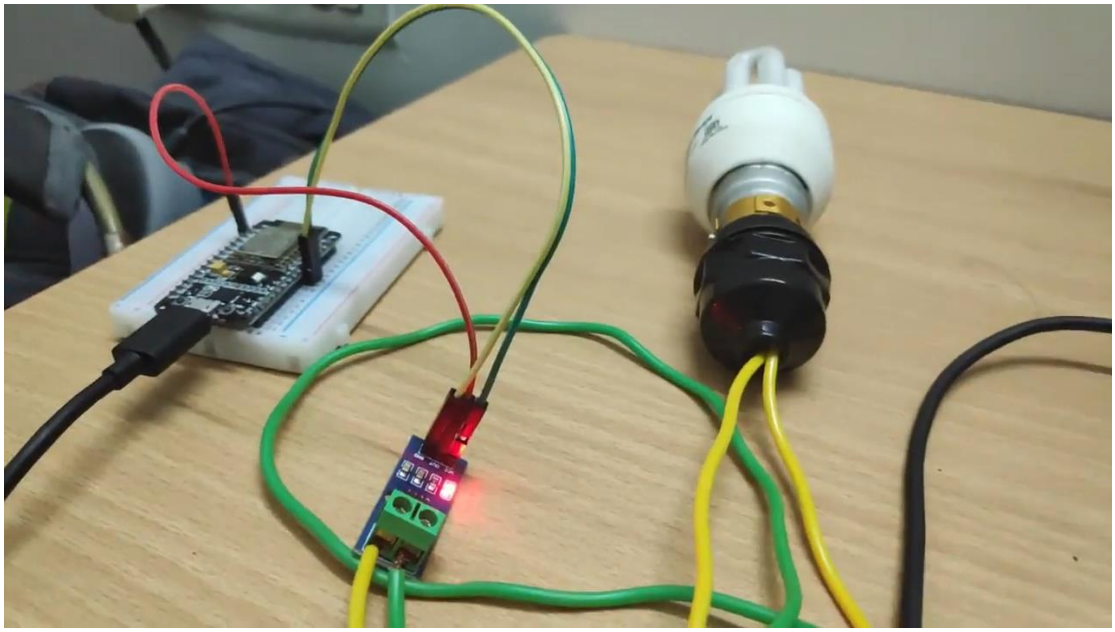
Schematic Diagram



Adafruit Dashboard



Implementation



Result:

Power Usage Monitoring System using ESP8266 NodeMCU and ACS712 Current Sensor has been successfully developed and implemented.