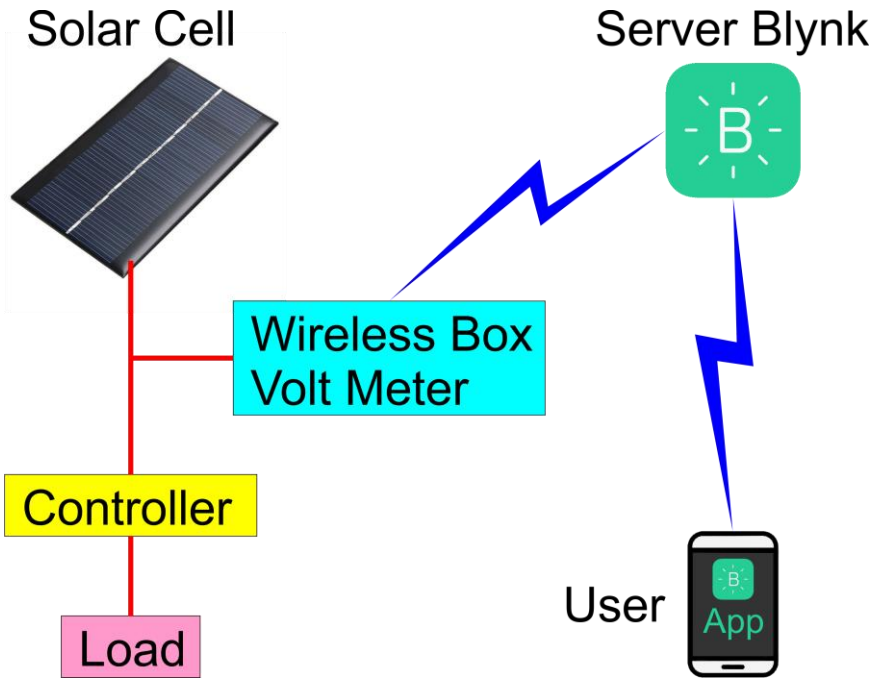
BAB III
METODOLOGI PENELITIAN

3.1 Deskripsi Sistem

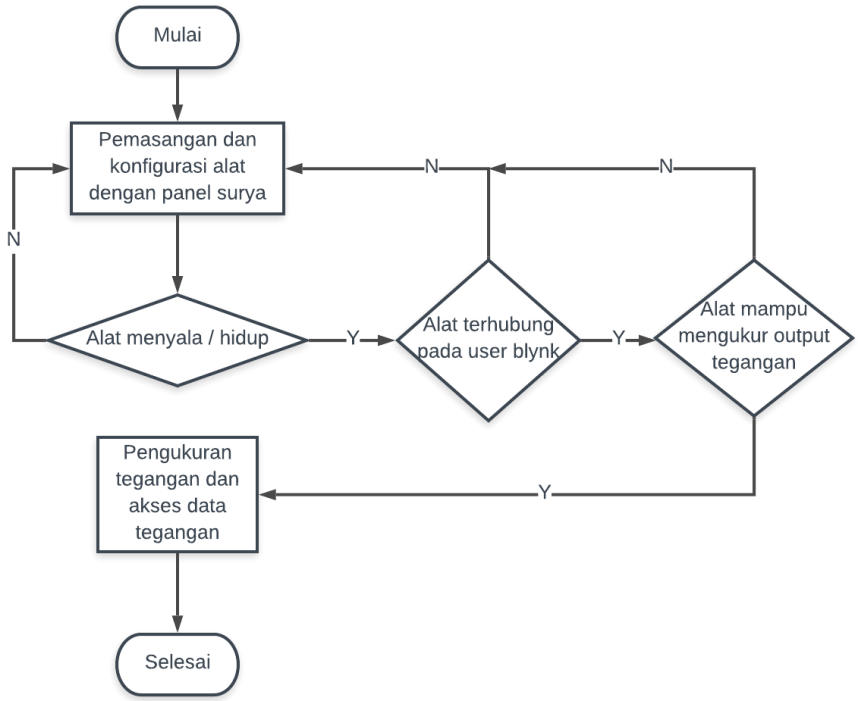
Alat tersebut kami sebut dengan *Wireless Box Volt Meter*. Prinsip dasarnya, menggunakan *NodeMCU ESP8266* sebagai mikrokontroler utama dengan mengatur jalannya program pengukuran dari *ADS1115* sebagai pengonversi sinyal analog ke digital. Selanjutnya dihubungkan dengan panel surya sebagai objek yg diukur. Data pengukuran ditransmisikan ke *server blynk* dan diteruskan ke *user* sehingga dapat diakses serta ditampilkan lewat aplikasi *blynk* pada *smartphone.* Berikut deskripsi sistem kerja Wireless Box Volt Meter ditunjukkan pada gambar 3.1.



Gambar 3.1 : Deskripsi sistem kerja Wireless Box Volt Meter

Untuk pemaparan lebih detail menggunakan blok diagram cara kerja sistem wireless box volt meter.

1. Pemasangan dan konfigurasi alat. Alat dipasang pada rangkaian dengan cara menghubungkan dengan sumber tegangan yang diperoleh dari baterai / tengangan PLN dengan adaptor AC-DC / tengangan keluaran dari panel surya yang telah disesuaikan oleh *controller*. Kemudian, input analog pada *ADS1115* dihubungkan pada kutub positif antara panel surya dengan *controller*. Dipasang juga *voltmeter* digital sebagai penanda dan pembanding pengukuran. Dapat ditambahkan input analog lainnya sesuai kebutuhan dan pengembangan.
2. Alat akan diuji terlebih dahulu dengan beberapa tahapan. Pertama, alat sudah aktif atau belum. Ditandai dengan nyala LED pada *NodeMCU*. Kedua, alat dapat terhubung dengan *user blynk* atau belum. Ditandai dengan status *connected* pada aplikasi *blynk*. Ketiga, alat mampu mengukur keluaran panel surya atau belum. Ditandai dengan perubahan nilai *gauge* pada aplikasi *blynk* dan *voltmeter* digital.
3. Jika seluruh tahapan sudah selesai, maka alat dapat beroperasi dengan benar dan mampu diakses oleh *user* dari mana saja kapan saja.

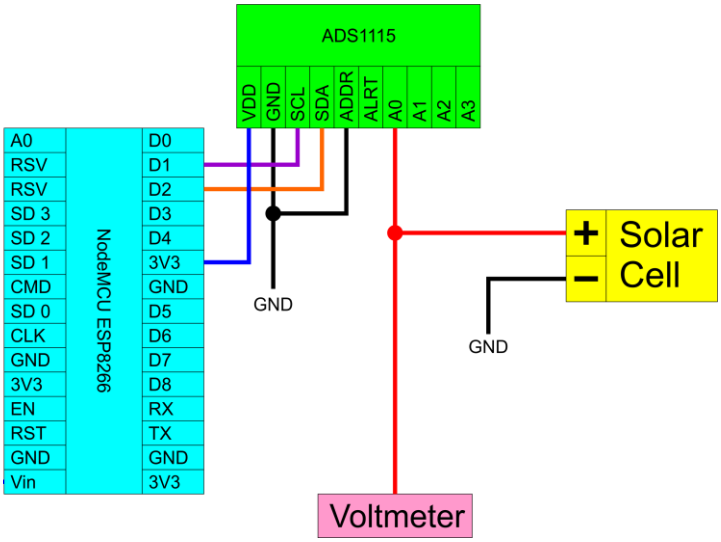


Gambar 3.2 : Cara kerja sistem *wireless box volt meter*

3.2 Pembuatan Perangkat Keras

3.2.1 Skematik

Untuk memudahkan dalam perancangan alat, digunakan skematik sebagai acuan dalam merangkai komponen-komponen yang tersedia. Berikut skematik yang digunakan ditunjukkan pada gambar 3.3.



Gambar 3.3 : Skematik dasar perancangan alat

3.2.2 Komponen Alat

Untuk merancang *prototype* dari alat tersebut, dibutuhkan beberapa komponen utama diantaranya:

1. *NodeMCU ESP8266*
2. Modul *ADC_ADS1115 4 Channel 16-bit*
3. Volt ampere meter digital

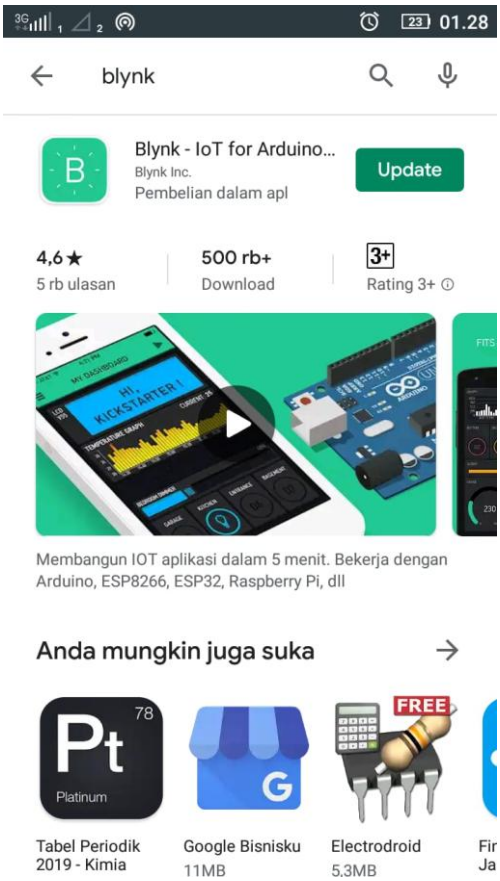
3.3 Pembuatan Perangkat Lunak

3.2.3 *Interface* pada *Blynk*

Untuk membuat *interface* pemantau tegangan pada aplikasi *blynk*, dijelaskan sebagai berikut (gambar pada lampiran).

1. Memasang aplikasi *Blynk*

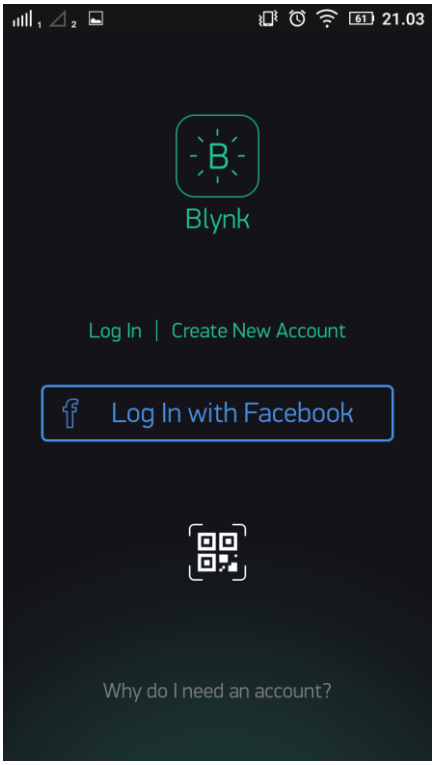
Pada versi *android*, dapat diunduh pada *google playstore* dengan kata kunci *blynk*. Ditunjukkan pada gambar 3.4.



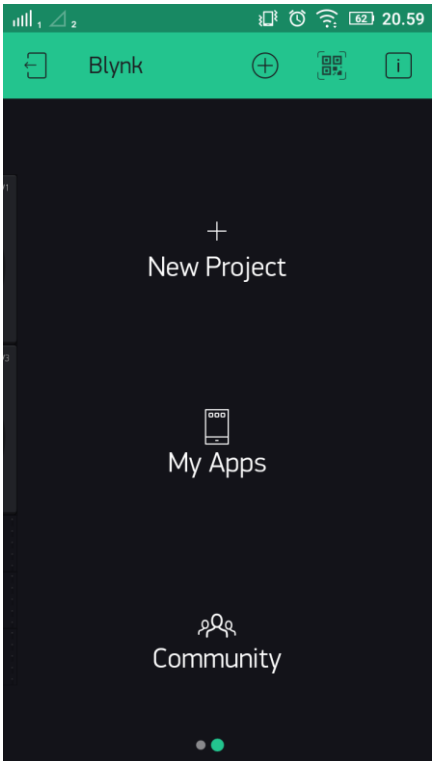
Gambar 3.4 : Aplikasi *blynk* dalam *playstore*

2. Membuat proyek baru

Setelah membuka *blynk* >> *Log In* >> *New Project* >> Sesuaikan nama proyek, *device* yang digunakan (*ESP8266*) >> *Create*. Ditunjukkan pada gambar 3.5 dan 3.6.



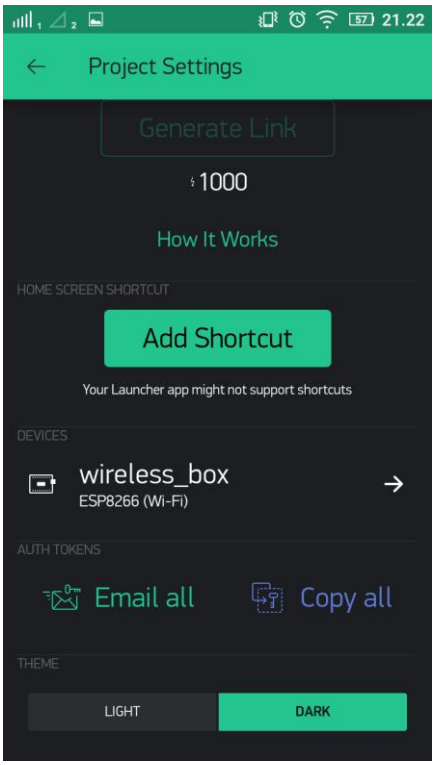
Gambar 3.5 : *Log In* pada *Blynk*



Gambar 3.6 : Tampilan awal aplikasi

3. Sesuaikan kode autentifikasi

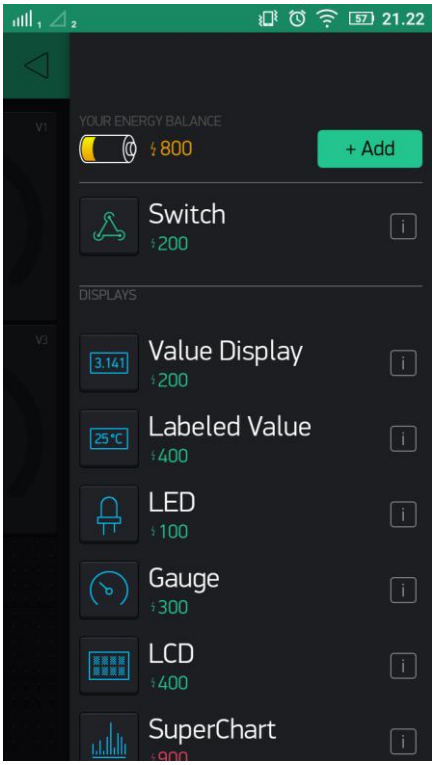
Pada *blynk*, kode autentifikasi menggunakan token khusus. Salin langsung atau kirim kode autentifikasi melalui *email*. Ditunjukkan pada gambar 3.7.



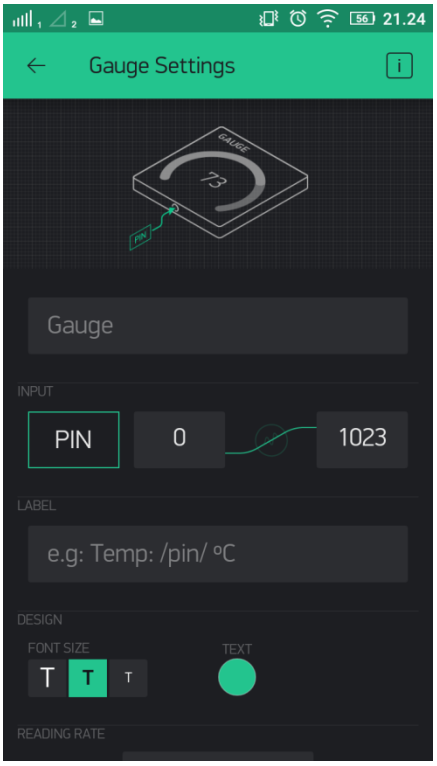
Gambar 3.7 : Tampilan pengaturan proyek *blynk*

4. Menyesuaikan *widget*

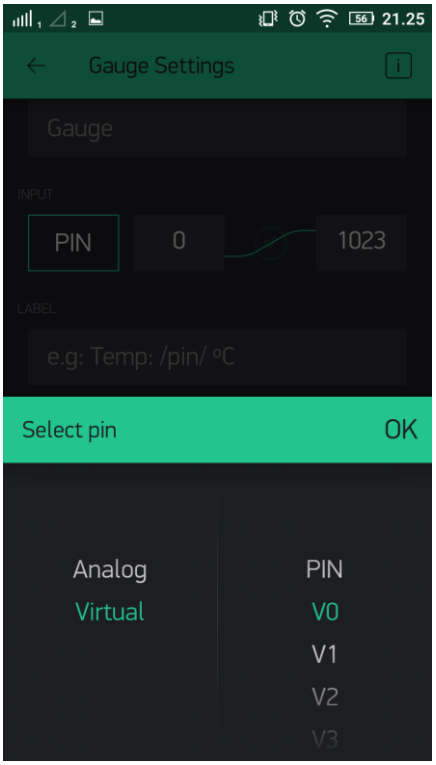
Tambahkan *widget* sesuai kebutuhan. Pada proyek ini, digunakan "*Gauge*" pada kelompok "*Display*". Klik *widget gauge* tersebut untuk menyesuaikan pengaturan >> pilih PIN yang digunakan (digunakan pin virtual. Pilih V0). Ditunjukkan pada gambar 3.8 hingga 3.11.



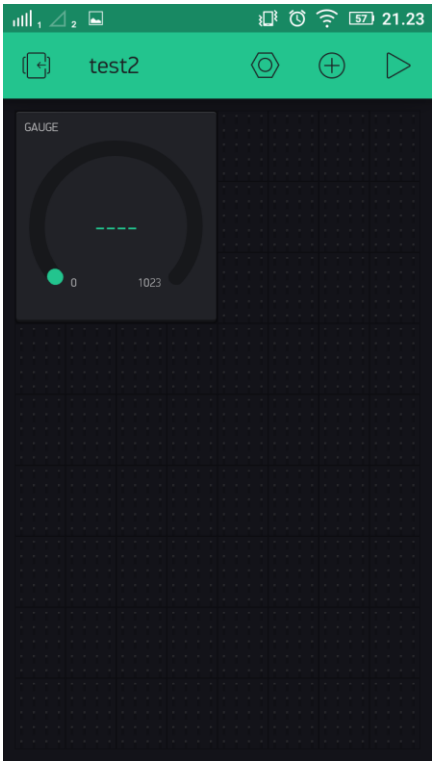
Gambar 3.8 : Variasi pilihan *widget*



Gambar 3.9 : Pengaturan *displays gauge*



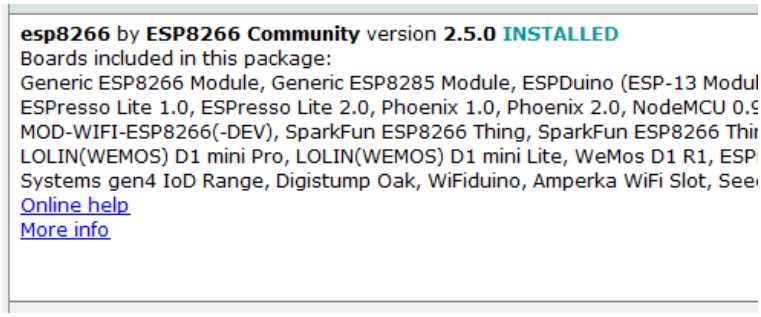
Gambar 3.10 : Pilihan pin pada *gauge widget*



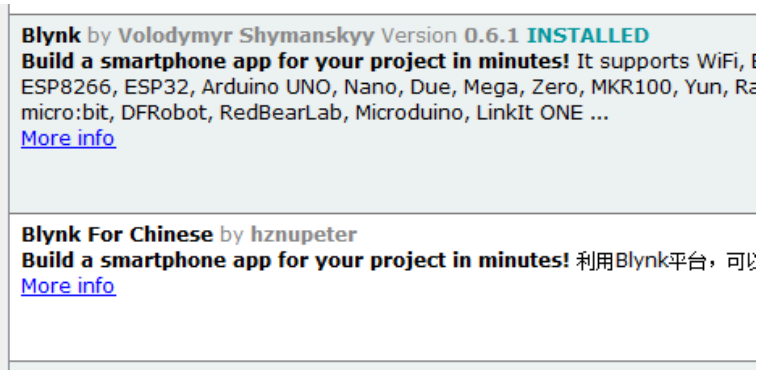
Gambar 3.11 : Tampilan awal proyek

3.2.4 **Pemrograman *ESP8266***

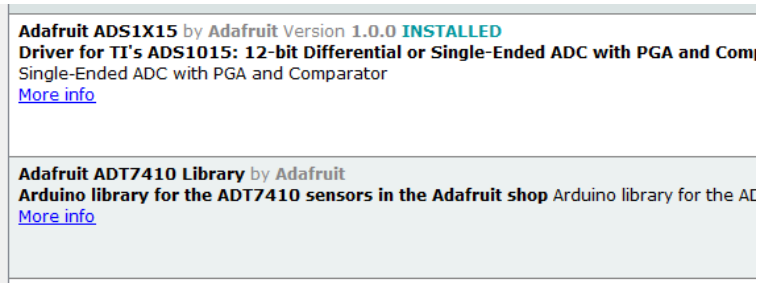
Pemrograman pada *ESP8266* menggunakan Arduino IDE dengan beberapa pengaturan tertentu. Diperlukan memperbarui *ESP8266 board* pada *board management* seperti gambar 3.12. Selanjutnya, menambahkan *blynk library* dan *ADS1115 library* pada *library management* seperti pada gambar 3.13 dan 3.14.



Gambar 3.12 : *ESP8266* pada *board manager*



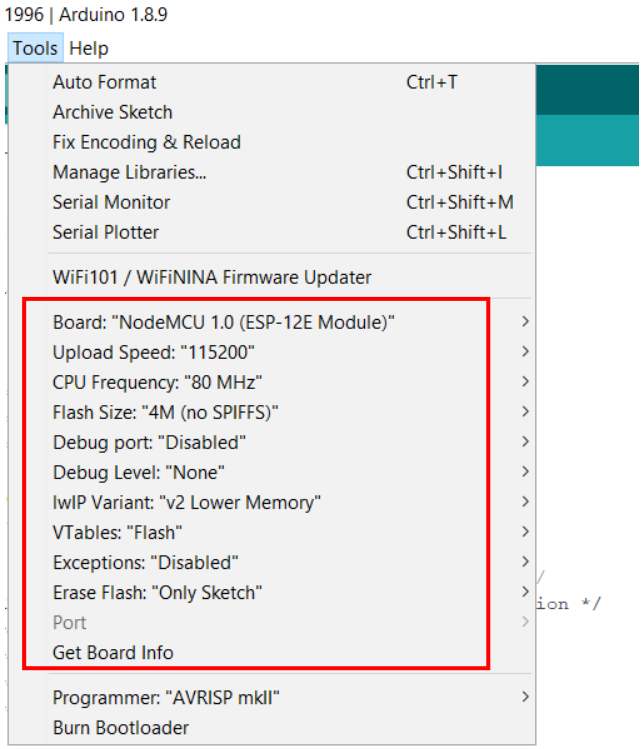
Gambar 3.13 : *Blynk* pada *library manager*



Gambar 3.14 : *Adafruit ADS1X15* pada *library manager*

Pada menu tools, diubah board sesuai kebutuhan yaitu *NodeMCU 1.0 (ESP-12E Module)* dan beberapa tambahan lainnya (pengaturan ke bawah

beberapa otomatis terpilih sesuai *board* yang digunakan, ditunjukkan pada gambar 3.15).



Gambar 3.15 : Pengaturan *board* arduino

Berikut adalah *script* program dasar yang digunakan pada *ESP8266* yang tehubung dengan *ADS1115* dan aplikasi *blynk*.

```
#include <ESP8266WiFi.h>
#include <BlynkSimpleESP8266.h>
#include <Wire.h>
#include <Adafruit_ADS1015.h>

char auth[] = "zbTHLAIPqLRgYLTQ0ntp-CAzjxddVdjV";
char ssid[] = "fauzi1996";
char pass[] = "1996fauzi";

bool isFirstConnect = true;
SimpleTimer timer;

Adafruit_ADS1115 ads;  /* Use this for the 16-bit version
```

```
*/
//Adafruit_ADS1015 ads;     /* Use this for the 12-bit
version */
float Voltage0 = 0.0;
float Voltage1 = 0.0;
float Voltage2 = 0.0;
float Voltage3 = 0.0;

void setup()
{
  Serial.begin(115200);
  Blynk.begin(auth, ssid, pass);
  int mytimeout = millis() / 1000;
  while (Blynk.connect() == false) { // try to connect to
server for 10 seconds
    if((millis() / 1000) > mytimeout + 8){ // try local
server if not connected within 9 seconds
       break;
    }
  }
  timer.setInterval(350L, ReadANAlOGS);
  timer.setInterval(30000L, reconnectBlynk);  // check
every 30s if still connected to server

  // The ADC input range (or gain) can be changed via the
following
  // functions, but be careful never to exceed VDD +0.3V
max, or to
  // exceed the upper and lower limits if you adjust the
input range!
  // Setting these values incorrectly may destroy your
ADC!
  //ADS1015  ADS1115
  // ads.setGain(GAIN_TWOTHIRDS); // 2/3x gain +/- 6.144V
 1 bit = 3mV      0.1875mV (default)
  ads.setGain(GAIN_ONE); // 1x gain   +/- 4.096V  1 bit =
2mV      0.125mV
```

```
  // ads.setGain(GAIN_TWO); // 2x gain +/- 2.048V  1 bit =
1mV      0.0625mV
  // ads.setGain(GAIN_FOUR); // 4x gain +/- 1.024V  1 bit
= 0.5mV    0.03125mV
  // ads.setGain(GAIN_EIGHT); // 8x gain +/- 0.512V  1 bit
= 0.25mV   0.015625mV
  // ads.setGain(GAIN_SIXTEEN); // 16x gain +/- 0.256V  1
bit = 0.125mV  0.0078125mV

  ads.begin();
}

BLYNK_CONNECTED() {
if (isFirstConnect) {
 Blynk.syncAll();
 Blynk.notify("ADS1115 STARTING!!!!");
isFirstConnect = false;
}
}

void ReadANAlOGS()
{

  int16_t adc0, adc1, adc2, adc3;

  adc0 = ads.readADC_SingleEnded(0);
  adc1 = ads.readADC_SingleEnded(1);
  adc2 = ads.readADC_SingleEnded(2);
  adc3 = ads.readADC_SingleEnded(3);

  Voltage0 = (adc0 * 0.125)/1000;
  Voltage1 = (adc1 * 0.125)/1000;
  Voltage2 = (adc2 * 0.125)/1000;
  Voltage3 = (adc3 * 0.125)/1000;

 //Serial.print("AIN0: "); Serial.println(Voltage0);
 //Serial.print("AIN1: "); Serial.println(Voltage1);
```

```
 //Serial.print("AIN2: "); Serial.println(Voltage2);
 //Serial.print("AIN3: "); Serial.println(Voltage3);
 //Serial.println(" ");


  Blynk.virtualWrite(V0, Voltage0);
  Blynk.virtualWrite(V1, Voltage1);
  Blynk.virtualWrite(V2, Voltage2);
  Blynk.virtualWrite(V3, Voltage3);
}

void reconnectBlynk() {
  if (!Blynk.connected()) {
    if(Blynk.connect()) {
     BLYNK_LOG("Reconnected");
    } else {
      BLYNK_LOG("Not reconnected");
    }
  }
}

void loop()
{

  if (Blynk.connected()) {
    Blynk.run();
  }
  timer.run();
}
```