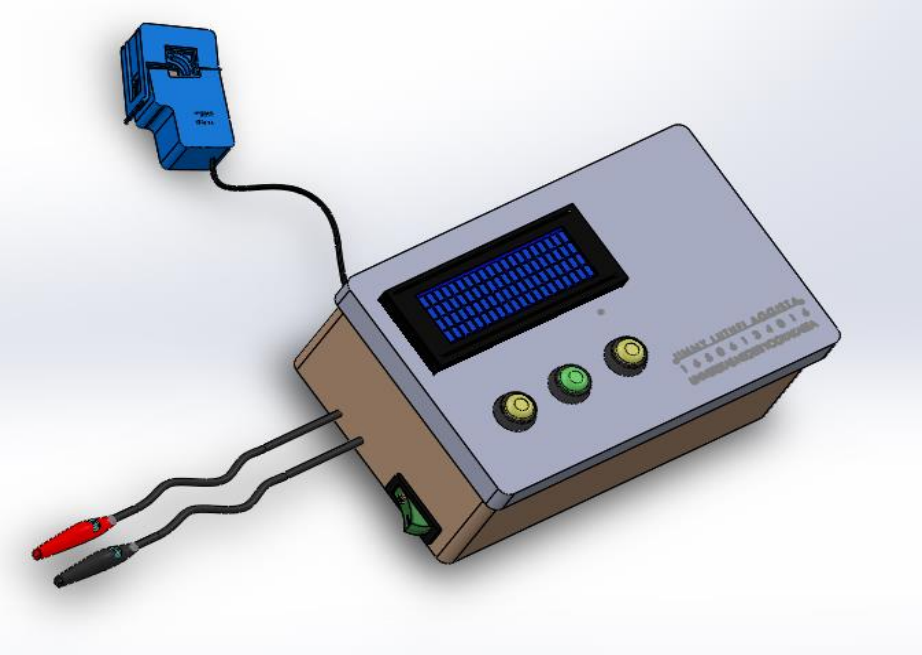
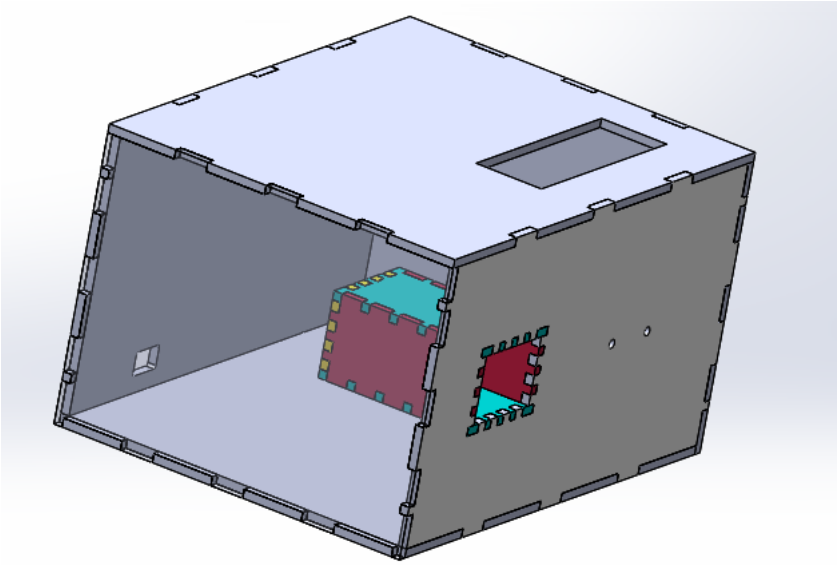
LAMPIRAN 1

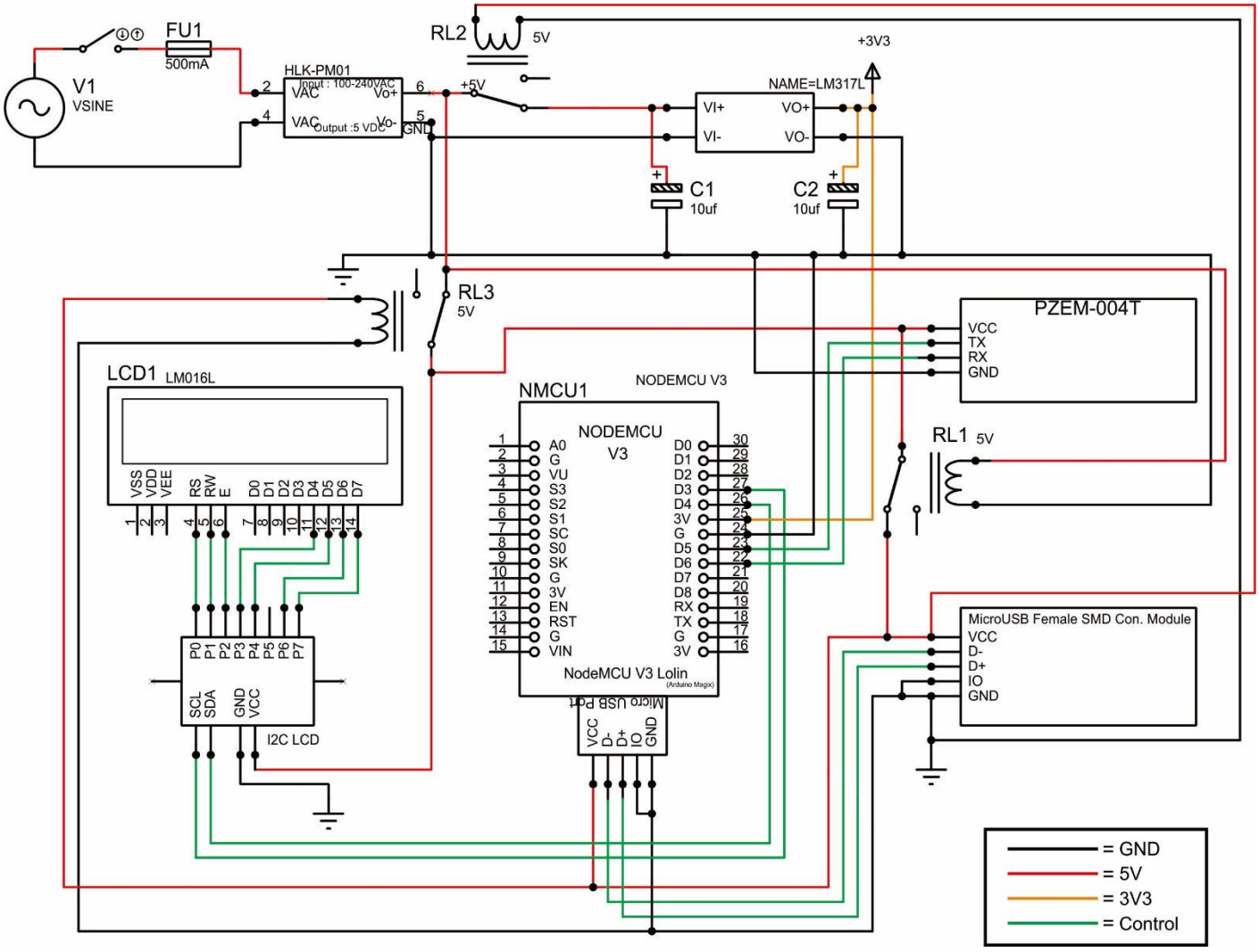
Desain 3D Alat





LAMPIRAN 2

Gambar Skema Rangkaian



LAMPIRAN 3

Source Code Alat

```
// Inisiasi Awal

#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <PZEM004T.h>
#include <Wire.h>
//#include <LCD.h>   // D4 SDA & D3 SCL //
#include <LiquidCrystal_I2C.h>

//inisiasi LCD DISPLAY
LiquidCrystal_I2C lcd(0x27, 16, 2);
//LiquidCrystal_I2C lcd(0x27 ,2,1,0,4,5,6,7,3, POSITIVE);

//inisiasi PZEM-004T
PZEM004T pzem(12,14);//nodemcu berarti D6 dan D5 sebagai RX, TX
IPAddress ip (192,168,1,1);

//Inisiasi variabel pengukuran
float voltage_blynk=0;
float current_blynk=0;
float power_blynk=0;
float energy_blynk=0;
float faktordaya_blynk=0;
```

```
// auth yang tertera pada apps BLYNK
char auth[] = "tlQr3t2YtBDNyfgbB0EDL9ESpGwR1UDq";

//SSID jaringan wifi yang digunakan
//Password jaringan tersebut
char ssid[] = "PASSION";
char pass[] = "Gangguru05";

BlynkTimer timer;
unsigned long lastMillis=0;

//inisiasi variabel cosphi
/*int pin = 15;
float pulsewidth=0;
float pf=0;
float phase=0;
float rads = 57.29577951;
float degree = 360;
float freq = 50;

void powerfactor ()
{
  pulsewidth = pulseIn (pin,HIGH);
  phase =( degree * freq * pulsewidth / 1000000);
  pf = cos(phase * 3.1415 / 180 );
  //lcd.setCursor (0,0);
 // lcd.print(phase);
```

```
 // lcd.setCursor(0,1);
  //lcd.print (pf);

  faktordaya_blynk=pf;
  Serial.print ("pulse="); Serial.print(pulsewidth);
  Serial.print("Cosphi= "); Serial.print(pf);
  Serial.print("Sudut= "); Serial.print(phase);
  //Serial.print("Time=");   Serial.print(lastMillis );
  delay (500);
}
*/
void mySensorDataSend()
{ float p = (pzem.power(ip)*0.94);
  float v = pzem.voltage(ip);
  float i = pzem.current(ip);
   /// Read meter PZEM
   {


    if(v < 0.0) v=0.0;
    {   voltage_blynk =v; } //V

  Serial.print("V= "); Serial.print(v); Serial.print(" Volt ");

  lcd.setCursor(0,0);
  lcd.print("V= ");
  lcd.setCursor(2,0);
```

```
lcd.print(v);
 }
 {

  if(i < 0.0 ) i=0.0;
  { current_blynk=i;    }
 Serial.print("I= ");   Serial.print(i); Serial.print(" Ampere ");

  lcd.setCursor(9,0);
  lcd.print("I= ");
  lcd.setCursor(11,0);
  lcd.print(i);
 }

 {

  if(p < 0.0) p=0.0;
  {power_blynk=(p);       } //kW

  Serial.print("P= ");   Serial.print(p); Serial.print(" Watt ");

  lcd.setCursor(9,1);
  lcd.print("P= ");
  lcd.setCursor(11,1);
  lcd.print(p);
 }
```

```
{
  float pf = (p/(v*i));
  /*((pzem.power(ip))/((pzem.voltage(ip))*(pzem.current(ip))));*/
 if ( pf > 1.0 )
{ ( pf = 1.0);  faktordaya_blynk=pf; }
 else if ( pf < 0.0 )
{( pf = 0.0); faktordaya_blynk=pf;  }
 else
 {
  //pf=((pzem.power(ip))/((pzem.voltage(ip))*(pzem.current(ip))));
  pf = (p/(v*i));
 }
lcd.setCursor(0,1);
lcd.print("Pf=");
lcd.setCursor(3,1);
 lcd.print(pf);
 Serial.print("PF= "); Serial.print(pf); Serial.print(" ");
faktordaya_blynk=pf;
}
{
 float e = pzem.energy(ip);
 if(e < 0.0) e=0.0;
 {  energy_blynk =e;  } ///kWh

 Serial.print("E= ");   Serial.println(e);
// lcd.setCursor(0,1);
// lcd.print("E=");
```

```
   // lcd.setCursor(2,1);
  // lcd.print(e);
   }
//delay(8000);
  //  lcd.print( " \n" );


  //Blynk.virtualWrite(V5,millis()/1000);


  //Publish data every 10 seconds (10000 milliseconds). Change this value to publish at
a different interval.
         if (millis() - lastMillis > 10000) {
          lastMillis = millis();
           //Serial.print("Time Elapsed Total = ");
           //Serial.println(lastMillis);
           Blynk.virtualWrite(V1, voltage_blynk);
           Blynk.virtualWrite(V2, current_blynk  );
           Blynk.virtualWrite(V3, power_blynk);
           Blynk.virtualWrite(V4, energy_blynk  );
           Blynk.virtualWrite(V5, lastMillis  );
           Blynk.virtualWrite(V6, faktordaya_blynk);
         }
}
void setup(){
  // Debug console
  Serial.begin(9600);
    pzem.setAddress(ip);
```

```
   Blynk.begin(auth, ssid, pass, "blynk-cloud.com",8442);
  timer.setInterval(10000L, mySensorDataSend);

    Wire.begin(2,0);
  lcd.init();   // initializing the LCD
  lcd.backlight(); // Enable or Turn On the backlight
}
void loop(){
 // powerfactor();
  Blynk.run();
  timer.run(); // Initiates BlynkTimer

}
```

LAMPIRAN 4

Datasheet PZEM-004T

AC digital display Multifunction Meter

Product Type: PZEM-004(V3.0)

A. Function

1. Ectrical parameter measurement function (voltage, current, active power, energy).
2. Overload alarm function (over power alarm threshold the power flash and the buzzer beeping to alarm).
3. Power alarm threshold preset function (can set power alarm threshold).
4. The reset function of energy key.
5. Store data when power off (store the accumulated energy before power off).
6. Bright red digital display function (display voltage, current, active power, energy).
7. Serial communication function (with TTL serial interface itself, can communicate with a variety of terminal through the pin board, read and set the parameters).

B. Front display and key

I. Display Interface

Display interface is formed by four bright red digital tubes, used to display the voltage, current, power, energy parameters.

II. Display Format

1. Power: Test Range: 0 ~ 22kW

   Within 0 ~ 10kW, the display format is 0.000 ~ 9.999;

   Within 10 ~ 22kW, the display format is 10.00 ~ 22.00.

2. Energy: Test Range: 0 ~ 9999kWh

   Within 0 ~ 10kWh, the display format is 0.000 ~ 9.999;

   Within 10 ~ 100kWh, the display format is 10.00 ~ 99.99;

   Within 100 ~ 1000kWh, the display format is 100.0 ~ 999.9;

   1000 ~ 9999kWh and above, the display format is 1000 ~ 9999.

3. Voltage: Test Range: 80 ~ 260VAC

   Display Format is 110.0 ~ 220.0.

4. Current: Test Range: 0 ~ 100A

   Display Format is 00.00 ~ 99.99.

III. Key

There is a key on the panel, it can be used to reset energy.

The method of reset energy: Long press the key for 5 seconds until the digital on energy display window flicker, then release the key. Short press the key again, then the energy data is cleared and quit the flickering state, now the reset operation is completed; if long press for 5 seconds again until no longer flicker, it means exit the reset state.

C. Wiring diagram

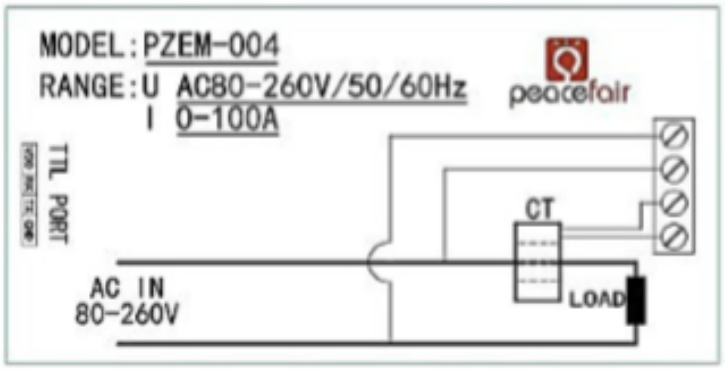


Figure 1 Wiring diagram

The wiring of this module is divided into two parts: the voltage and current test input terminal wiring and the serial communication wiring, as shown in Figure 1; according to the actual needs of the clients, with different TTL pin board to achieve communicate with different terminals.

D. Display Interface

The whole meter panel display window is formed by four windows, they are voltage, current, power and energy; the following are brief description of each parameter display:

1. Voltage Display

Measure and display the current power frequency grid voltage.

2. Current display

Measure and display the current load (appliances) current. There is supplementary instruction that the current test value is from the beginning of 10mA , but this module belongs to high power test equipment, if you care about the mA level current testing accuracy, it is not be recommended.

3. Energy display

Measure and display the current accumulative power consumption. There is supplementary instruction that the minimum unit of the energy metering is 0.001kWh,which means it begins to accumulate from 1Wh, relatively speaking, the resolution is rather high, for the low-power(within 100W)load test, you can observe the accumulative process rather intuitively.

4. Power display

Measure and display the current load power. There is supplementary instruction that the power test value is from the beginning of 0.001kW , which means it begins to test from 1W, but this module belongs to high power test equipment, if you have the requirement of the testing within 1W, it is not be recommended.

E. Serial communication

This module is equipped with TTL serial data communication interface, you can read and set the relevant parameters via the serial port; but if you want to communicate with a device which has USB or RS232 (such as computer), you need to be equipped with different TTL pin board (USB communication needs to be equipped with TTL to USB pin board; RS232 communication needs to be equipped with TTL to RS232 pin board), the specific connection type as shown in Figure 2. In the below table are the communication protocols of this module:

LAMPIRAN 5

Datasheet NodeMCU

NODEMCU LOLIN V3

Lolin NodeMCU V3 is an open source IoT platform. It uses the Lua scripting language. The eLua project is the basis of board, and built on the ESP8266 SDK 1.4. NodeMCU uses many open source projects, such as lua-cjson, and spiffs. The NodeMCU runs on the ESP8266 Wi-Fi SoC, and hardware which based on the ESP-12 module.

The Lolin NodeMCU V3 board ads USB/UART converter chip as well as decoupled LDO power supply. Also the board adds 2 miniature push buttons. The most important feature is that it breaks out all ESP8266 pins to board headers. The board headers are breadboard compatible 2.54 mm pitch headers.

The Lolin NodeMCU board uses the CH340G USB/UART converter chip. You will need to download and install the proper driver to get going with the development. You can find the drivers here:

NodeMCU CH340/CH340G Driver Download page (if not automatically recognised): Click Here

For MAC users please check this link .

Once you install the driver you can use the control panel to get the assigned serial COM port number. Connect to the LUA interpreter running on the ESP8266 via your favorite terminal emulator. The baud rate for most of the boards is 9600 baud (1 start bit, 8 data bits, no parity).

An easy way to communicate with the LUA interpreter on the ESP8266 is by using the ESP8266 LuaLoader. It allows you to perform simple tasks. For example, you can set the SSID and password for your wireless router so it can connect to your network via GUI. You can also read out or set the status of its GPIO ports. Get information like the IP address or the chipID, or upload files. Yet, you can as well try ESPlorer which has a more contemporary user interface. It also has syntax highlighting on LUA code.

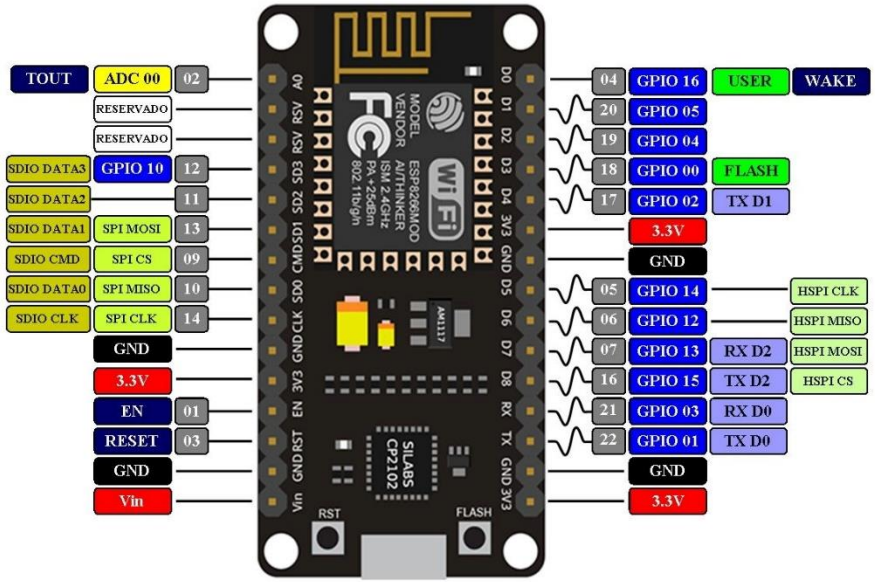
You can find some usage examples here to start your development activities.

Lolin NodeMCU Features

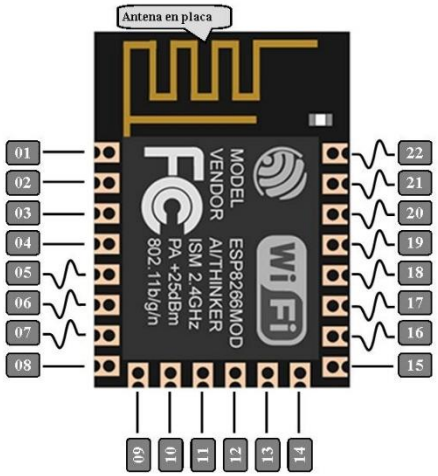
- Arduino-Like Hardware IO
- Code like Arduino, but interactively in Lua script
- Event-driven API for network applications, which facilitates developers writing code
- Integrates GPIO, PWM, IIC, 1-Wire and ADC all in one board
- 10 GPIO, every GPIO can be PWM, I2C, 1-wire
- 4M Flash Memory
- Built-in WiFi Antenna

PLACA NodeMCU 1.0 (V2)

PINOUT



ESP8266 12E



Vin — ALIMENTACIÓN EXTERNA (de 5V a 10V).

3.3 V — ALIMENTACIÓN INTERNA (desde la placa a dispositivos).

GND — TIERRA (GND *Ground*).

GPIO — PIN DE ENTRADA/SALIDA +3,3V (GPIO *General Purpose Input/Output*).
Entrada digital ——. Entrada analógica ∿. (Todas las salidas son digitales).

ADC — PIN DE SALIDA ANALÓGICA (el rango es entre +0V y +1V dividido en 1023 intervalos).

SPI — BUS SPI (*Serial Peripheral Interface*)

HSPI — BUS HSPI (*Hardware-Serial Peripheral Interface*).

SDIO — PINES PARA INICIO DEL ESP8266 DESDE UNA TARJETA SD.
Para activar el modo SDIO el pin GPIO 15 debe estar en tensión cuando se enciende la placa.

TX/RX — COMUNICACIÓN SERIE TX/RX.
Los pines GPIO01 y GPIO02 están conectados al puerto MicroUSB a través del conversor UART.

NOTAS:

- El voltaje de alimentación (Vin) debe estar comprendido entre 5 V y 10 V.

- La intensidad de **máxima** de salida a un pin es de 12 mA. No se debe demandar mas intensidad para no quemar el procesador. La intensidad de salida normal será de 6 mA.

- Para activar el modo de reposo *(sleep mode)*, unir los pines GPIO16 (D0) y RESET y poner el pin GPIO16 en tensión *(HIGH)*. Para reactivar *(wakeup)*, quitar la tensión en el pin GPIO16 *(LOW)*. El sistema se reiniciará.

- En *boot/reset/wakeup* (inicio/reseteo/reactivado), los pines GPIO00 (D3) **ó** GPIO15 (D8) **no** deben estar con tensión *(+3.3V)*. **Tampoco** el pin GPIO2 (D4) debe estar conectado a tierra *(+0.0V)*.

- Los pines GPIO01 (TX) y GPIO03 (RX) se utilizan en el puerto MicroUSB, por lo que no se deben utilizar simultáneamente con otro dispositivo ya que la conexión se interferiría.

- Los pines GPIO00 y GPIO02 **no** debe utilizarse para lectura *(input)*. El pin GPIO09 **no** debe utilizarse ni para lectura ni para escritura *(input/output)*.

- El pin GPIO02 (D4) controla el LED azul del ESP8266. Se enciende cuando no tiene tensión *(+0.0V)*.

- El pin GPIO16 (D0) controla el LED azul de la placa. Se enciende cuando no tiene tensión *(+0.0V)*. *(En la placa LoLin este LED no está disponible)*.

- Para flashear, en el caso de que la placa quede bloqueada, se debe conectar el pin GPIO00 (D3) a tierra, el MicroUSB con el ordenador y ejecutar el flasher.