ESCUELA DE
CIENCIAS MATEMÁTICAS Y
TECNOLOGÍA INFORMÁTICA

Wireless Network

LAB9: ESP8266 WiFi Module based Projects

Emil Vega Gualán

Objective

1. Describe the basic principles of radio communications.
2. Understand the usage of WiFi serial communication module to set the control parameters.

Discussion of fundamentals

Wi-Fi (Short for Wireless Fidelity) is a wireless technology that uses radio frequency to transmit data through the air. Wi-Fi transmits data in the frequency band of 2.4 GHz. Range of Wi-Fi technology is 40-300 feet.

Overview of the ESP8266 WiFi Module

The ESP8266 is a really useful, cheap WiFi module for controlling devices over the Internet. It can work with a micro-controller like the Arduino or it can be programmed to work on its own. ESP8266 delivers highly integrated WiFi SoC solution to meet users continuous demands for efficient power usage, compact design and reliable performance in the Internet of Things industry. With the complete and self-contained Wi-Fi networking capabilities, ESP8266 can perform either as a standalone application or as the slave to a multipoint host. When ESP8266 hosts the application, it promptly boots up from the flash. The integrated highspeed cache helps to increase the system performance and optimize the system memory. ESP8266 integrates antenna switches, RF balun, power amplifier, low noise receive amplifier, filters and power management modules. Besides the Wi-Fi functionalities, ESP8266 also integrates an enhanced version of Tensilicas L106 Diamond series 32-bit processor and on-chip SRAM. The ESP8266 comes with factory installed firmware allowing you to control it with standard AT commands. You can also easily create and upload your own code and this makes it hugely powerful and flexible

Pin Description of ESP8266 Module

The ESP8266 module consists of 8 pins and these pins are VCC, GND, TX, RX, RST, CH PD, GPIO0 and GPIO2. The following image shows the pin diagram of the ESP-01 Module.

- VCC: It is the power pin through which 3.3V is supplied.

- GND: It is the ground pin.
- TX: This pin is used to transmit serial data to other devices.
- RX: The RX pin is used to receive serial data from other devices.
- RST: It is the Reset Pin and it is an active LOW Pin. (ESP8266 will reset if the RST pin receives LOW signal)
- CH EN: This is the chip enable pin and it is an active HIGH Pin. It is usually connected to 3.3V.
- GPIO0: The GPIO0 (General Purpose I/O) Pin has dual functions one for normal GPIO Operation and other for enabling the Programming Mode of ESP8266.
- GPIO2: This is GPIO Pin.

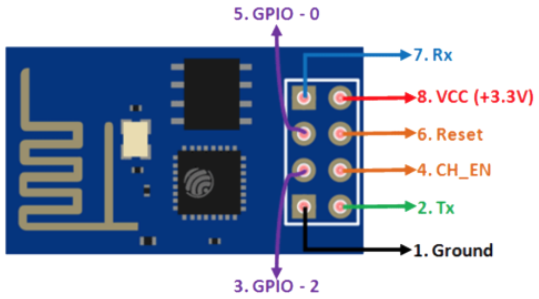


Figure 1: ESP8266 WiFi Module

Notes of ESP8266 Module

First and foremost, the ESP8266 Module works on 3.3V Power Supply and anything greater than that, like 5V for example, will kill the SoC. So, the VCC Pin and CH EN Pin of ESP8266 Module are connected to a 3.3V Supply. Next important thing to remember is that the ESP8266 WiFi Module has two modes of operation: Programming Mode and Normal Mode. In Programming Mode, you can upload the program or firmware to the ESP8266 Module and in Normal Mode, the uploaded program or firmware will run normally. In order to enable the Programming Mode, the GPIO0 pin must be connected to GND. Finally, ESP8266 works in two modes: Station (STA) and Access Point (AP). In short, AP mode allows it to create its own network and have other devices (your phone) connect to it and STA mode allows the ESP8266 to connect to a Wi-Fi network (one created by your wireless router). So, an important feature about the ESP8266 is that it can function as a client or as an access point or even both. In figure 2, we can see a summary of all the variations of the operation modes of the module.

(a) Station

(b) Soft access point

(c) Client

(d) Server

(e) Client secure

(f) Station + Soft Access Point

Figure 2: ESP8266 operating in different modes

Wi-Fi Key Features

- 802.11 b/g/n support
- 802.11n support (2.4 GHz), up to 72.2 Mbps
- Support Infrastructure BSS Station mode/SoftAP mode/Promiscuous mode

Applications

- Home appliances
- Home automation
- Smart plugs and lights
- Industrial wireless control
- Baby monitors
- IP cameras
- Sensor networks

- Wearable electronics
- Wi-Fi location-aware devices
- Security ID tags
- Wi-Fi position system beacons

DHT11 Sensor

DHT11 Temperature & Humidity Sensor features a temperature & humidity sensor complex with a calibrated digital signal output. By using the exclusive digital-signal-acquisition technique and temperature & humidity sensing technology, it ensures high reliability and excellent long-term stability. This sensor includes a resistive-type humidity measurement component and an NTC (Negative Temperature Coefficient → the resistance decreases as the temperature increases) temperature measurement component, and connects to a highperformance 8-bit microcontroller, offering excellent quality, fast response, antiinterference ability and cost-effectiveness.

The pins in the DHT11 are:

- GND: connection to ground
- DATA: data transmission
- VCC: Power

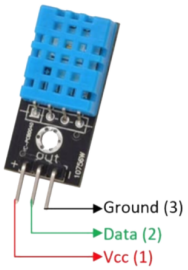


Figure 3: ESP8266 WiFi Module

Exercise 1. Program the ESP8266 using the Arduino IDE - Library ESP8266

There are several ways to program the ESP8266 module. For example, the AT commands, the arduino code, microPython, espruino, ESPBasic. In this lab, we are going to use the arduino IDE (C ++) because its libraries are compatible with ESP8266 and this allows us to program with ease. If you want to review in detail all the libraries compatible with ESP8266, check this link.

Installation of the library 8266

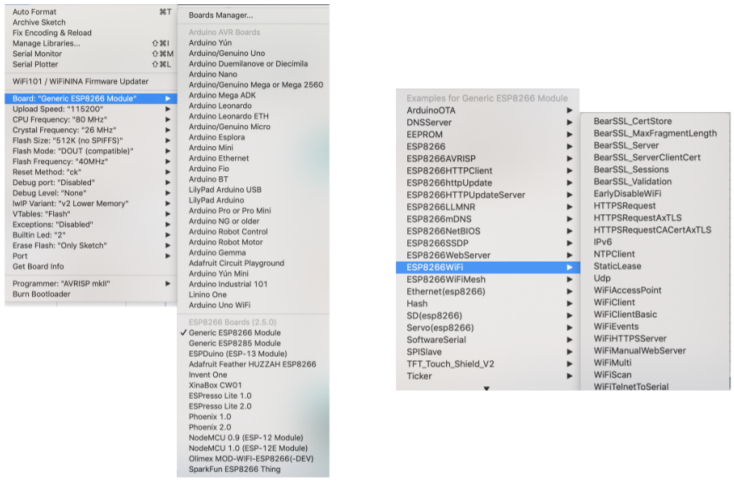
1. Open Installing with Boards Manager. Starting with 1.6.4, Arduino allows installation of third-party platform packages using Boards Manager. Check that your current upstream Arduino IDE at the 1.8.7 level or later.

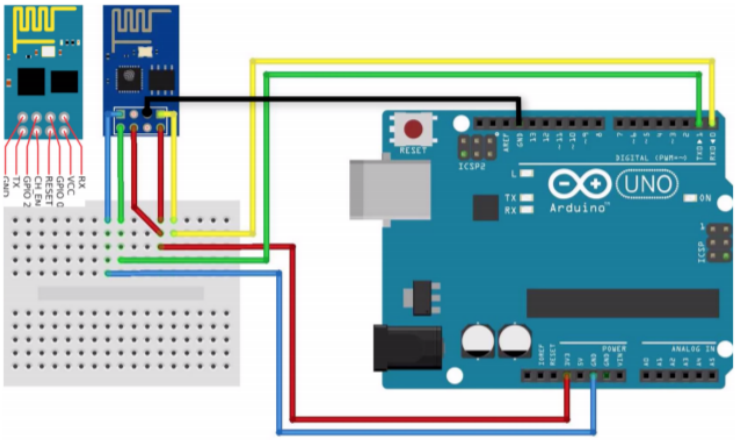
2. Start Arduino and open Preferences window. Enter `http://arduino.esp8266.com/stable/package_esp8266com_index.json` into Additional Board Manager URLs field. You can add multiple URLs, separating them with commas.

3. Open Boards Manager from Tools > Board menu and install esp8266 platform (and dont forget to select your ESP8266 board from Tools > Board menu after installation).

Programming the Firmware of the ESP8266 with Arduino IDE

Before making the connection make sure to unplug the pic from Arduino UNO. Make following circuit.



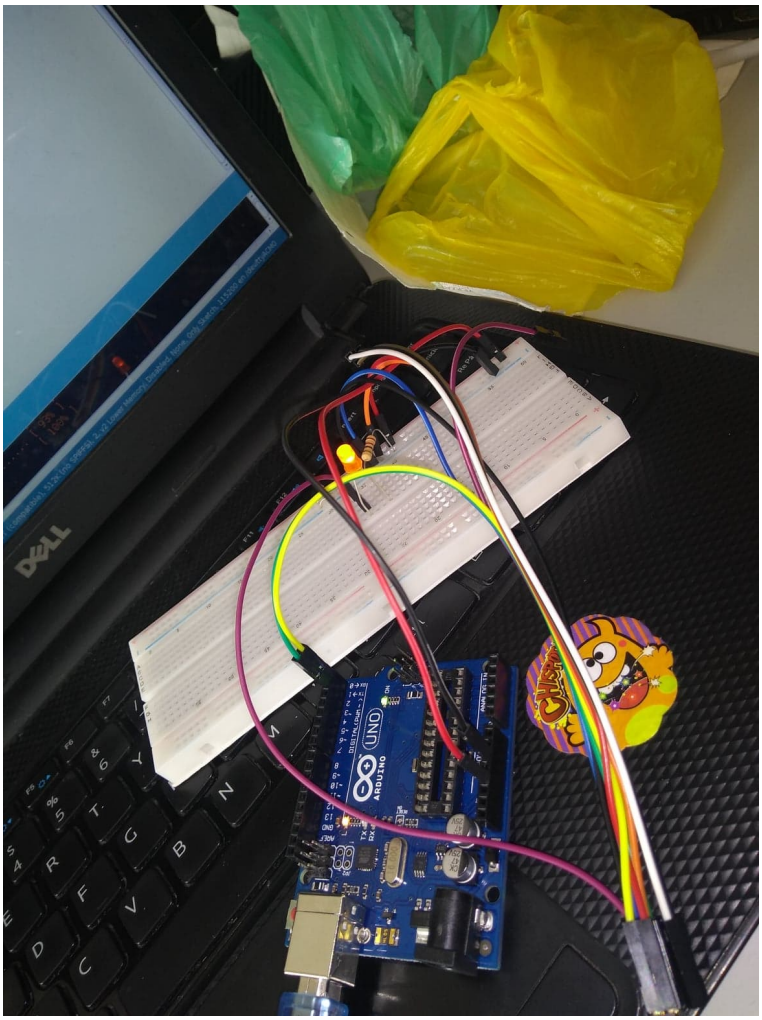


Programming

Open up the Arduino IDE and load the WiFiScan code from Examples > ESP8266WiFi. This sketch demonstrates how to scan WiFi networks:

Code

```
#include "ESP8266WiFi.h"

void setup() {
  Serial.begin(115200);
  // Set WiFi to station mode and disconnect from an AP if it was previously connected
  WiFi.mode(WIFI_STA);
  WiFi.disconnect();
  delay(100);
  Serial.println("Setup done");
}

void loop() {
  Serial.println("scan start");
  // WiFi.scanNetworks will return the number of networks found
  int n = WiFi.scanNetworks();
  Serial.println("scan done");
  if (n == 0) {
    Serial.println("no networks found");
  } else {
    Serial.print(n);
    Serial.println("networks found");
    for (int i = 0; i < n; ++i) {
      // Print SSID and RSSI for each network found
      Serial.print(i + 1);
      Serial.print(": ");
      Serial.print(WiFi.SSID(i));
      Serial.print(" (");
      Serial.print(WiFi.RSSI(i));
      Serial.print(")");
      Serial.println((WiFi.encryptionType(i) == ENC_TYPE_NONE) ? " " : "");
      delay(10);
    }
  }
  Serial.println("");
  // Wait a bit before scanning again
  delay(5000);
}
```



Exercise 2: Using the GPIO pins of the ESP8266

This exercise demonstrates a way to use as an output of the GPIO2 pin to an LED that will flash every second.

Building Circuit

Before making the connection make sure to unplug the power source from Arduino UNO. Make following circuit. Note that the GIOI0 is connected to ground so that the firmware recording is possible. Once the firmware is loaded, we can disconnect it so that it works correctly.

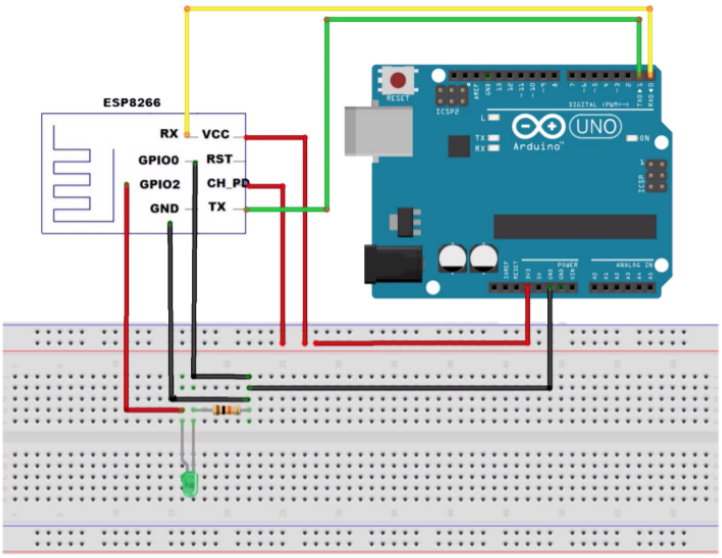


Figure 4: Assembly Scheme 2

Programming

Open up the Arduino IDE and write the following code into a new sketch

Code

```
#include "ESP8266WiFi.h"

void setup() {
  // put your setup code here, to run once:
  pinMode(2, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite (2, HIGH);
  delay(1000);
  digitalWrite (2, LOW);
  delay(1000);
}
```

Exercise 3. Station Mode Operation - Connect to a WIFI network

In this exercise, the station mode will be used to connect to a Wi-Fi network generated by an access point.

Programming

Open up the Arduino IDE and write the following code into a new sketch:

Code

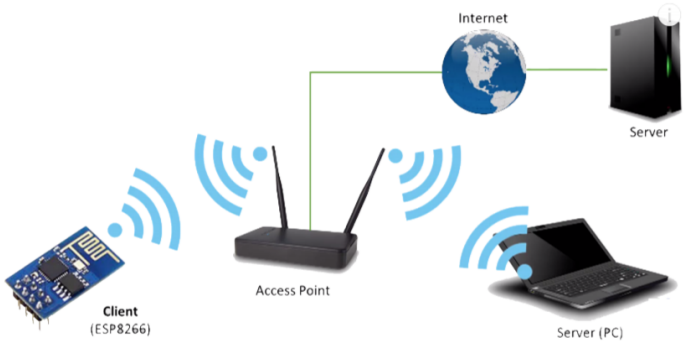
```
// Include the libs
#include <ESP8266WiFi.h>

//Parameters WIFI
const char* ssid = "XFORCE";
const char* password = "RonnychaUzumaqui";

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  delay(20);

  Serial.println ("ready");
  // Connect to the WIFI
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED){
  delay(500);
  Serial.print(".") ;
}

Serial.println ("");
Serial.println (" WiFi  connected!");
Serial.println (WiFi.localIP());
}

void loop() {
}
```

Output Result

Once you have successfully set up the circuit connections and your arduino code has been uploaded, you can connect a power source to your arduino board. Remember to check the port, click Tools -¿ Port and select the COM port which your WiFi module is connected. See the result on Serial Monitor

```
load 0x4010f000, len 1384, room 16 Wi-Fi connected!
192.168.0.107
```

Figure 5: Serial Monitor



Exercise 4. Operation in Client Mode - Perform requests to a server

In the exercise, we will connect to a server www.example.com. First, we connect to the access point using the station mode and then we will use the wifi client class to make the connection through the Get method through the internet and reach the server and it will return an answer:

Programming

Open up the Arduino IDE and write the following code into a new sketch:

Code

```
// 'Include the libs'
#include <ESP8266WiFi.h>

//'Parameters WIFI'
const char* ssid = "xforce";
const char* password = "dome1234";

const char* host = "www.google.com";

void setup()
{
  Serial.begin(115200);
  Serial.println();

  Serial.printf("Connecting to %s ", ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println(" connected");
}

void loop()
{
  // 'Client that will connect to the Host'
  WiFiClient client;

  Serial.printf("\n[Connecting to %s ... ", host);
  // 'Try to connect'
  if (client.connect(host, 80))
  {
    Serial.println("connected]");

    Serial.println("[Sending a request]");
    client.print(String("GET /") + " HTTP/1.1\r\n" +
                 "Host: " + host + "\r\n" +
                 "Connection: close\r\n" +
                 "\r\n"
                 );

    Serial.println("[Response:]");
    // 'While the connection lasts'
    while (client.connected())
    {
      // If there are available data
      if (client.available())
      {
        String line = client.readStringUntil('\n');
        Serial.println(line);
      }
    }
    // 'Once the server sends all the required data it is disconnected and the program continues'
    client.stop();
    Serial.println("\n[Disconnected]");
  }
  else
  {
    Serial.println("connection failed !]");
    client.stop();
  }
  delay(5000);
}
```

Output Result

Once you have successfully set up the circuit connections and your arduino code has been uploaded, you can connect a power source to your arduino board. Remember to check the port, click Tools -¿ Port and select the COM port which your WiFi module is connected. See the result on Serial Monitor

```
load 0x4010f000, len 1384, room 16
Connecting to TVCABLE_REY .......... connected

[Connecting to www.example.com ... connected]
[Sending a request]
[Response:]
HTTP/1.1 200 OK
Cache-Control: max-age=604800
Content-Type: text/html; charset=UTF-8
Date: Thu, 18 Apr 2019 08:07:09 GMT
Etag: "1541025663+gzip+ident"
Expires: Thu, 25 Apr 2019 08:07:09 GMT
Last-Modified: Fri, 09 Aug 2013 23:54:35 GMT
Server: ECS (agb/52B2)
Vary: Accept-Encoding
X-Cache: HIT
Content-Length: 1270
Connection: close

<!doctype html>
<html>
<head>
    <title>Example Domain</title>

    <meta charset="utf-8" />
    <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />

[Disconnected]
```

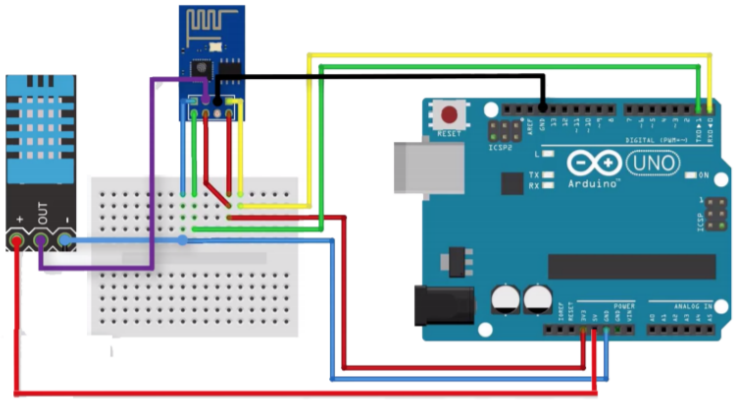
Figure 6: Serial Monitor

Exercise 5. Temperature and Humidity - DHT Library.

This exercise demonstrates a way to use as an output of the GPIO2 pin to an LED that will flash every second.

Building Circuit

Before making the connection make sure to unplug the power source from Arduino UNO. Make following circuit. Note that the GIOI0 is connected to ground so that the firmware recording is possible. Once the firmware is loaded, we can disconnect it so that it works correctly



Programming

Open up the Arduino IDE and write the following code into a new sketch:

Code

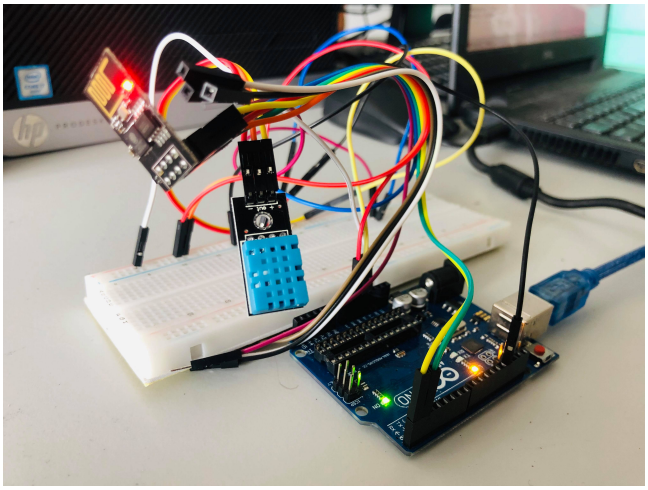
```
#include <ESP8266WiFi.h>
#include <DHT.h>

/*** Sensor model ***/
#define DHTTYPE DHT11 //'DHT21, DHT22'

/*** Pin GPIO2***/
#define DHTPIN 2 // GPIO2

DHT dht(DHTPIN, DHTTYPE, 27); // 'DHT11 works fine for ESP8266 threshold => MHZ CPU'

/*** Variables for Humidity and Temperature ****/
float temperature; // 'double'
float humidity;

void setup()
{
  Serial.begin(115200);
  dht.begin();

}

void loop(){
   temperature = dht.readTemperature();
   humidity = dht.readHumidity();
   Serial.println("------------------------------");
   Serial.print("Temperature: ");
   Serial.print(temperature);
   Serial.println(" C ");
   Serial.print("Humidity: ");
   Serial.print(humidity, 4);
   Serial.println("%");
   Serial.println("------------------------------");
   Serial.println();

   delay(3000);
} // EOF loop()
```

Code

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

// 'Sensor model'
#define DHTTYPE DHT11 //'DHT21, DHT22'

// 'Pin GPIO2'
#define DHTPIN 2 // GPIO2

DHT dht(DHTPIN, DHTTYPE, 27); // 'DHT11 works fine for ESP8266 threshold => MHZ CPU'

// 'Variables for Humidity and Temperature'
float temperature; // 'double'
float humidity;

void setup(){
  Serial.begin(115200);
  dht.begin();
}

void loop(){
  temperature = dht.readTemperature();
  humidity = dht.readHumidity();
  Serial.println("----------------------------");
  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.println(" C ");
  Serial.print("Humidity: ");
  Serial.print(humidity, 4);
  Serial.println("%");
  Serial.println("----------------------------");
  Serial.println() ;
  delay(3000);
} // 'EOF loop()'
```

Output Result

Once you have successfully set up the circuit connections and your arduino code has been uploaded, you can connect a power source to your arduino board. Remember to check the port, click $Tools- > Port$ and select the COM port which your WiFi module is connected. See the result on Serial Monitor.



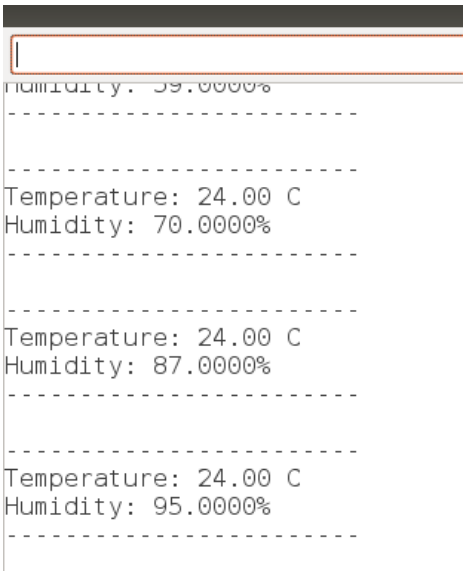


Figure 7: Serial Monitor

Exercise 6. Read Temperature, Humidity and Display at the Web

In this exercise, we will use XAMPP as a suite of Web development tools, created by Apache Friends, makes it easy to run PHP (Personal Home Pages) scripts locally on your computer. For this, you must:

1. Install, start and test XAMPP
2. Create a folder /esp8266/ in /htdocs
3. Create index.php and copy in the folder: /htdocs/esp8266/

Programming the index.php file

Open up an PhP editor and write the following code into a new file:

Code

```
<?php
   //'Create a text  file  to save the data sent by ESP8266'
   if (!  file_exists  ("miTemp&Hum.txt")){
      // 'If the file  does not exist , we create it'
       file_put_contents  ("miTemp&Hum.txt" ,"0.0\r\n0.0");
   }

   // 'If we receive Data with the GET Method, we process it'
   if ( isset  ($_GET['Temp']) && isset($_GET['Hum'])){
      $var3 = $_GET['Temp'];
      $var4 = $_GET['Hum'];
      $fileContent = $var3 . "\r\n" . $var4;
       $fileSave = file_put_contents ("miTemp&Hum.txt", $fileContent);
   }

   // 'We read the data in the file  to save them in variables'
    $fileStr = file_get_contents ("miTemp&Hum.txt");
   $pos1 = strpos($fileStr , "\r\n");
   $var1 = substr($fileStr , 0, $pos1);
   $var2 = substr($fileStr , $pos1 + 1);
?>

<!DOCTYPE html>
<html lang="es">
<head>
   <meta charset=" utf8 ">
   <meta name="viewport" content="width=devicewidth,  initialscale =1.0">
   <meta  httpequiv ="refresh" content="15">
   <title>Exercise6</title>
</head>
   <style>
      h1 {
         color : antiquewhite;
          backgroundcolor : dodgerblue;
           textalign  : center ;
      }
   </style>
<body>
   <header>
      <h1>Welcome to UEYT</h1>
   </header>

   <section>
      <h3>Temperature: <?php echo $var1; ?></h3>
      <h3>Humedad: <?php echo $var2; ?></h3>
   </section>
</body>
</html>
```

Programming ESP8266

Open up the Arduino IDE and write the following code into a new sketch:

Code

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

//'Parameters WIFI'
const char* ssid = "iLuis";
const char* password = "12345678";
const char* host = "172.20.10.5"; //'localhost    IP    from your PC'

// 'Sensor model'
#define DHTTYPE DHT11 //DHT21, DHT22

// 'Pin GPIO2'
#define DHTPIN 2 // GPIO2
DHT dht(DHTPIN, DHTTYPE, 27);

// 'Variables for Humidity and Temperature'
float temperature;
float humidity;

void setup(){
  Serial.begin(115200);
  Serial.println () ;
  dht.begin();
  Serial.printf ("Connecting to %s ", ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED){
    delay(500);
    Serial.print(".");
  }
  Serial.println(" connected");
}

void loop()
{
  WiFiClient client;
  Serial.printf ("\n[Connecting to %s ... ", host);
  if ( client.connect(host, 80))
  {
    Serial.println("connected]");
    temperature = dht.readTemperature();
    humidity = dht.readHumidity();
    Serial.println ("[Sending a request]");
    client.print(String("GET /esp8266/?Temp=") + temperature + "&Hum=" + humidity +
          "HTTP/1.1\r\n" + "Host: " + host + "\r\n" +"Connection: close\r\n" +"\r\n") ;
    Serial.println ("[Response:]");
    while (client.connected())
    {
      if ( client.available ())
      {
        String line = client .readStringUntil('\r\n');
        Serial.println( line ) ;
      }
    }
    client.stop() ;
    Serial.println("\n[Disconnected]");
  }
  else
  {
    Serial.println("connection failed !] ");
    client.stop() ;
  }
  delay(5000);
}
```

Output Result

Once you have successfully set up the circuit connections and your arduino code has been uploaded, you can connect a power source to your arduino board. Remember to check the port, click Tools -¿ Port and select the COM port which your WiFi module is connected. See the result on the bowser online http://localhost/esp8266/

Figure 8: Web Browser http://localhost/esp8266/

Exercise 7. Humidity & Temperature Monitoring using DHT11 & 8266 on ThingSpeak

In the exercise you must use Thingspeak as cloud service provider and DHT11 to measure temperature and humidity. Thingspeak is the easiest way to get Arduino devices connected to ThingSpeak IoT services.

Code

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

//'Parameters WIFI'
const char* ssid = "iLuis";
const char* password = "12345678";
const char* host = "172.20.10.5"; //'localhost    IP    from your PC'

// 'Sensor model'
#define DHTTYPE DHT11 //DHT21, DHT22

// 'Pin GPIO2'
#define DHTPIN 2 // GPIO2
DHT dht(DHTPIN, DHTTYPE, 27);

// 'Variables for Humidity and Temperature'
float temperature;
float humidity;

void setup(){
  Serial.begin(115200);
  Serial.println () ;
  dht.begin();
  Serial.printf ("Connecting to %s ", ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED){
    delay(500);
    Serial.print(".");
  }
  Serial.println(" connected");
}

void loop()
{
  WiFiClient client;
  Serial.printf ("\n[Connecting to %s ... ", host);
  if ( client.connect(host, 80))
  {
    Serial.println("connected]");
    temperature = dht.readTemperature();
    humidity = dht.readHumidity();
    Serial.println ("[Sending a request]");
    client.print(String("GET /esp8266/?Temp=") + temperature + "&Hum=" + humidity +
          "HTTP/1.1\r\n" + "Host: " + host + "\r\n" +"Connection: close\r\n" +"\r\n") ;
    Serial.println ("[Response:]");
    while (client.connected())
    {
      if ( client.available ())
      {
        String line = client .readStringUntil('\r\n');
        Serial.println( line ) ;
      }
    }
    client.stop() ;
    Serial.println("\n[Disconnected]");
  }
  else
  {
    Serial.println("connection failed !] ");
    client.stop() ;
  }
  delay(5000);
}
```

Exercise 8. Humidity & Temperature Monitoring using DHT11 & 8266 on Blynk app

In this exercise using an esp8266, to show the temperature and humidity DHT11 sensor on your Smartphone or tablet. The ESP8266 collects the temperature and humidity from DHT11 sensor and sends it to Blynk app every second.

Code

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

//'Parameters WIFI'
const char* ssid = "iLuis";
const char* password = "12345678";
const char* host = "172.20.10.5"; //'localhost    IP    from your PC'

// 'Sensor model'
#define DHTTYPE DHT11 //DHT21, DHT22

// 'Pin GPIO2'
#define DHTPIN 2 // GPIO2
DHT dht(DHTPIN, DHTTYPE, 27);

// 'Variables for Humidity and Temperature'
float temperature;
float humidity;

void setup(){
  Serial.begin(115200);
  Serial.println () ;
  dht.begin();
  Serial.printf ("Connecting to %s ", ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED){
    delay(500);
    Serial.print(".");
  }
  Serial.println(" connected");
}

void loop()
{
  WiFiClient client;
  Serial.printf ("\n[Connecting to %s ... ", host);
  if ( client.connect(host, 80))
  {
    Serial.println("connected]");
    temperature = dht.readTemperature();
    humidity = dht.readHumidity();
    Serial.println ("[Sending a request]");
    client.print(String("GET /esp8266/?Temp=") + temperature + "&Hum=" + humidity +
          "HTTP/1.1\r\n" + "Host: " + host + "\r\n" +"Connection: close\r\n" +"\r\n") ;
    Serial.println ("[Response:]");
    while (client.connected())
    {
      if ( client.available ())
      {
        String line = client .readStringUntil('\r\n');
        Serial.println( line ) ;
      }
    }
    client.stop() ;
    Serial.println("\n[Disconnected]");
  }
  else
  {
    Serial.println("connection failed !] ");
    client.stop() ;
  }
  delay(5000);
}
```