MEMORIA TÉCNICA

HOLA HUERTO!

Proyecto practico en marco curso Programacion Placas roboticas IOT.

Realizado por:
Enrique Lorenzo.

Muy agradecido por su indispensable colaboracion a:

Maribel Ribes Rosell
David Serrano Moreno
Joan Masdemont Fontas
Asi como a todos los creadores opensource que facilitan sus creaciones de manera abierta.

INDICE

Memoria descriptiva. _____ *pg 3.*

Lista de materiales necesarios. _____ *pg 4.*

Codigo usado. _____ *pg 5.*

Diagrama de circuito electrico. _____ *pg 11.*

Fuentes bibliogr@fia. _____ *pg 12.*

Memoria descriptiva.

El objetivo del proyecto es la monitorizacion de las condiciones ambientales de temperatura y humedad en tiempo real, de un huerto situado a 170 km de distancia.
Para ello programaremos el microcontrolador ESP8266 01, usando un Arduino uno y un ordenador personal, para que obtenga datos de un sensor de temperatura / humedad y utilizando la conexión accesible en la casa, envie estos datos a una pagina web desde donde estos seran accesibles.

Descripcion del esp8266 01:

Características ESP8266EX:

- MCU 32-bit 80 MHz (programable)
- Comunicación serie 8N1 (115200 baudios)
- Frecuencia de trabajo: 2.4GHz (2400 – 2484 MHz)
- WiFi: 802.11 b/g/n
- Wifi Direct (P2P), soft-AP, LNA, WPA/WPA2,…
- Potencia salida: +19.5dBm (802.11b)
- Consumo<1.0mW(standby),240mA(max)

Existen 2 versiones que se distinguen por el color de la placa: la primera, azul, dispone de 512 kB, y la segunda, negra, dispone de 1024 kB. Ambas disponen de 2 pines GPIO y un interfaz de comunicación serie que permite configurarlo mediante comandos AT.

El ESP01 incluye un microcontrolador mucho más potente que el propio Arduino, de hecho es posible programar este módulo y utilizarlo de forma independiente.

Antes de realizar las conexiones hay que considerar lo siguiente:

- El ESP8266 funciona a 3.3V y no tiene entradas tolerantes 5V, por lo que se necesita realizar una **conversión de nivel** para comunicarse con un microcontrolador a 5V como Arduino. También se puede utilizar una tarjeta adaptadora que algunos vendedores suministran junto al ESP-01S
- Haciendo un uso intensivo del Wifi, el consumo puede superar los 200 mA, por tanto debe tener una **alimentación independiente**; no obstante para nuestros sketchs de ejemplo usamos la alimentación de Arduino (max 50mA)

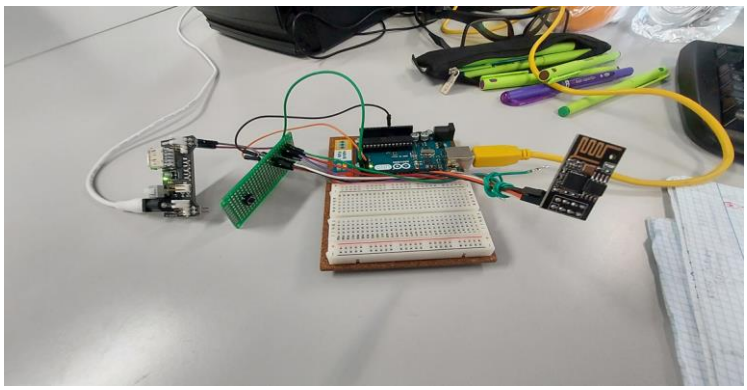
Memoria descriptiva.

Para programar el esp8266 producimos un programador siguiendo las indicaciones de tutoriales online.

El proceso de grabación se resume en los pasos siguientes.

Este modelo tal vez sea el más molesto de conectar: para que el modulo arranque en modo “grabar” conectar los pines VCC (3.3V) y CH_PD o EN(a 3.3V) y GND con GND de alimentacion, RX y TX directamente con los Pin RX y TX de arduino esto establece el modo comunicacion par programar.
El pin IoO del esp01 permanecera alto(1) durante el compilado y hasta el momento de inicio de carga entonces debe estar bajo (0) hasta que el sketch este cargado. {En nuestro grabador mantener pulsado boton, pero en cuanto el programa empieza a cargarse soltar el boton en nuestro gravador.}
Para la conexión modo “trabajo”,cuando se necesita abrir comunicación serial entre
el esp01 y el PC vamos a conectar el RX y el TX a los Pines TX y RX de arduino (es decir cruzados), con esto se establece la comunicación serial con el PC.
Para programar se necesita que el ESP8266 arranque con el pin GPIO0 en estado LOW, de esta forma el ESP entra a modo de programación. (mantener en low durante proceso compilacion, Arduino.
[La conexión del módulo *ESP8266-01* **con la placa** *Arduino* **es algo compleja, ya que el cableado necesario para cargar** *software* **y el necesario para ejecutarlo es algo diferente**. **Para cargar** *software* **al módulo** *WIFI* **utilizaremos los pines** *RX* **y** *TX* **de la placa** *Arduino*, **mientras que en funcionamiento normal, para enviar variables desde la placa *Arduino* al módulo *WIFI* utilizaremos la librería *SoftwareSerial.h* y los pines *D10* y *D11]*.**

[El módulo *ESP8266-01* **cuenta con dos modos de funcionamiento en función de el estado del pin** *GPIO0 (esp01)* **(0 o 1), ‘modo programación’ o ‘***UART’***.y ‘modo ejecutar código’.]**



La imagen muestra el grabador creado.

Listado de materiales necesarios.

Para el gravador de esp8266:

Alimentacion externa estable 3,3 v cc.	1 und
Arduino uno.	1 und
Esp 8266.	1 und
Placa de prototipado pequeña.	1 und
Boton Push N/O.	1 und
Cables jumper H-M.	9 und
Cables jumper M-M.	3 und

Para la constitucion del elemento:

Esp 8266.	1 und
Sensor DHT11.	1 und
Placa propotipado pequeña.	1 und
Alimentacion externa estable 3,3 v cc.	1 und
Cable:	
rojo, negro, blanco.	1 und
Caja contenedora.	1Und

Codigo usado:

Titulo del archivo ***.ino (codigo ejemplo esp8266_wiffi_serverconDHT11_1.0)*** obtenido en internet de formato open source. Modificado y personalizado.

Librerias necesarias incluidas:

<WiFi.h>
<Arduino.h>
<ESP8266WiFi.h>
<Hash.h>
<ESPAsyncTCP.h>
<ESPAsyncWebServer.h>
<Adafruit_Sensor.h>
 <DHT.h>

EL CODIGO.

//(1.5)

```
#ifdef ESP32
#include <WiFi.h>
#endif
#include <Arduino.h>
#include <ESP8266WiFi.h>
#include <Hash.h>
#include <ESPAsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include <Adafruit_Sensor.h>
#include <DHT.h>

// Replace with your network credentials
const char* ssid = "###########";
const char* password = "###########";

#define DHTPIN 2     // Digital pin D1 connected to the DHT sensor

// Uncomment the type of sensor in use:
#define DHTTYPE    DHT11     // DHT 11
//#define DHTTYPE    DHT22     // DHT 22 (AM2302)
//#define DHTTYPE    DHT21     // DHT 21 (AM2301)

DHT dht(DHTPIN, DHTTYPE);

// current temperature & humidity, updated in loop()
float t = 0.0;
float h = 0.0;

// Create AsyncWebServer object on port 80
```

```
AsyncWebServer server(80);
```

EL CODIGO.

//(2.5)

```
// Generally, you should use "unsigned long" for variables that hold time
// The value will quickly become too large for an int to store
unsigned long previousMillis = 0;    // will store last time DHT was updated

// Updates DHT readings every 10 seconds
const long interval = 10000;

const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE HTML><html>
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.7.2/css/all.css" integrity="sha384-
fnmOCqbTlWIlj8LyTjo7mOUStjsKC4pOpQbqyi7RrhN7udi9RwhKkMHpvLbHG9Sr"
crossorigin="anonymous">
  <style>
    html {
     font-family: Arial;
     display: inline-block;
     margin: 0px auto;
     text-align: center;
    }
    h2 { font-size: 3.0rem; }
    p { font-size: 3.0rem; }
    .units { font-size: 1.2rem; }
    .dht-labels{
      font-size: 1.5rem;
      vertical-align:middle;
      padding-bottom: 15px;
    }
  </style>
</head>
<body>
  <h2>ESP8266 DHT Server</h2>
  <p>
    <i class="fas fa-thermometer-half" style="color:#059e8a;"></i>
    <span class="dht-labels">Temperature</span>
    <span id="temperature">%TEMPERATURE%</span>
    <sup class="units">°C</sup>
  </p>
  <p>
    <i class="fas fa-tint" style="color:#00add6;"></i>
    <span class="dht-labels">Humidity</span>
    <span id="humidity">%HUMIDITY%</span>
    <sup class="units">%</sup>
  </p>
  <p>
      <script src="https://apps.elfsight.com/p/platform.js" defer></script>
<div class="elfsight-app-65e091b0-d33c-4191-81f3-be77c921660a"></div>
    </P>
  <p>
    <i class="fab fa-youtube" style="font-size:1.0rem;color:red;"></i>
```

EL CODIGO.

//**(3.5)**

```
    <span style="font-size:1.0rem;">Subscribe to </span>
    <a href="https://www.youtube.com/channel/UC49xSqiQ6gBrxUMQ9zvzO6A" target="_blank"
style="font-size:1.0rem;">The IoT Projects YouTube Channel</a>
  </P>
</body>
<script>
setInterval(function ( ) {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("temperature").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "/temperature", true);
  xhttp.send();
}, 10000 ) ;

setInterval(function ( ) {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("humidity").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "/humidity", true);
  xhttp.send();
}, 10000 ) ;
</script>
</html>)rawliteral";

// Replaces placeholder with DHT values
String processor(const String& var){
  //Serial.println(var);
  if(var == "TEMPERATURE"){
    return String(t);
  }
  else if(var == "HUMIDITY"){
    return String(h);
  }
  return String();
}

void setup(){
  // Serial port for debugging purposes
  Serial.begin(115200);
  dht.begin();
  IPAddress ip(10, 199, 160, 145);
  IPAddress dns(8, 8, 8, 8);
  IPAddress gateway(10, 199, 160, 254);
  IPAddress subnet(255, 255, 255, 0);
```

```
  WiFi.config(ip, dns, gateway, subnet);
```

EL CODIGO.

//(4.5)

```
 // Connect to Wi-Fi
  WiFi.begin(ssid, password);
  Serial.println("Connecting to WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.println(".");
  }

  // Print ESP8266 Local IP Address
  Serial.println(WiFi.localIP());

  // Route for root / web page
  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/html", index_html, processor);
  });
  server.on("/temperature", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", String(t).c_str());
  });
  server.on("/humidity", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", String(h).c_str());
  });

  // Start server
  server.begin();
  delay(500);

}

void loop(){

  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval) {
    // save the last time you updated the DHT values
    previousMillis = currentMillis;
    // Read temperature as Celsius (the default)
    float newT = dht.readTemperature();
    // Read temperature as Fahrenheit (isFahrenheit = true)
    //float newT = dht.readTemperature(true);
    // if temperature read failed, don't change t value
    if (isnan(newT)) {
      Serial.println("Failed to read from DHT sensor!");
    }
    else {
      t = newT;
      Serial.println(t);
    }
    // Read Humidity
    float newH = dht.readHumidity();
    // if humidity read failed, don't change h value
    if (isnan(newH)) {
```

```
    Serial.println("Failed to read from DHT sensor!");
```

EL CODIGO.

//(5.5)

```
  }
   else {
     h = newH;
     Serial.println(h);
   }
```

CONCLUSIONES.

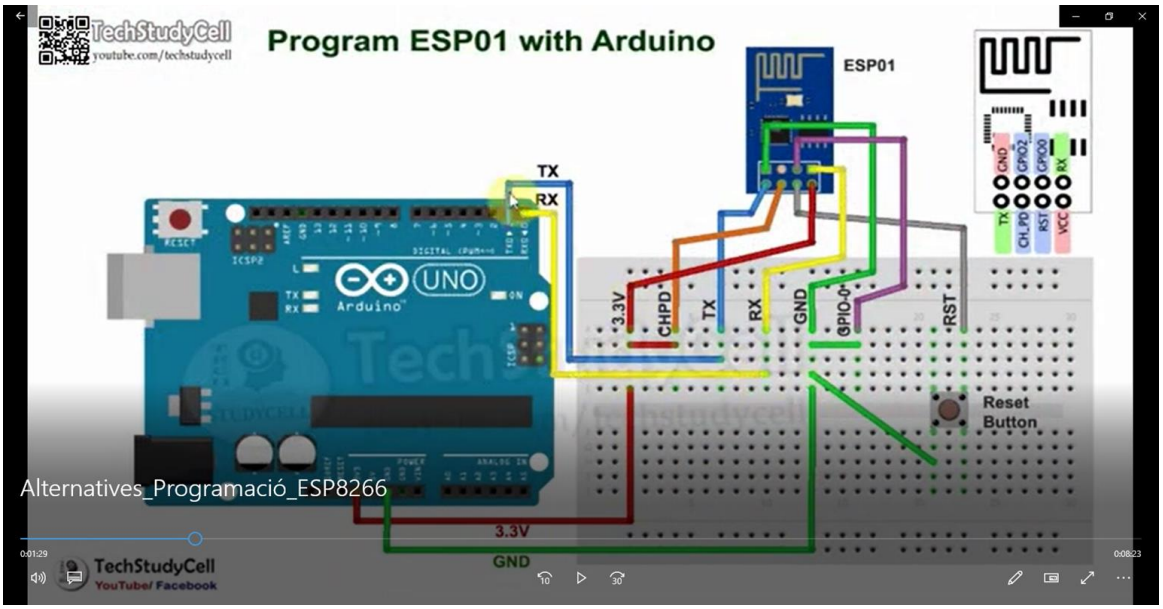
El grabador debe de re hacerse, seria conbeniente el uso de un interruptor electronico como un optoacoplador para regular el estado de el pin esp01 GPIO Io0, de una forma mas eficaz, el uso de un boton mecanico resta mucha precision en el cambio de estado y provoca muchos fallos de carga.

No he podido recrear el proyecto en casa ni en el lugar donde esta destinado, por lo que ahun tengo que resolver algunos problemas, para poder implementar libremente el proyecto.

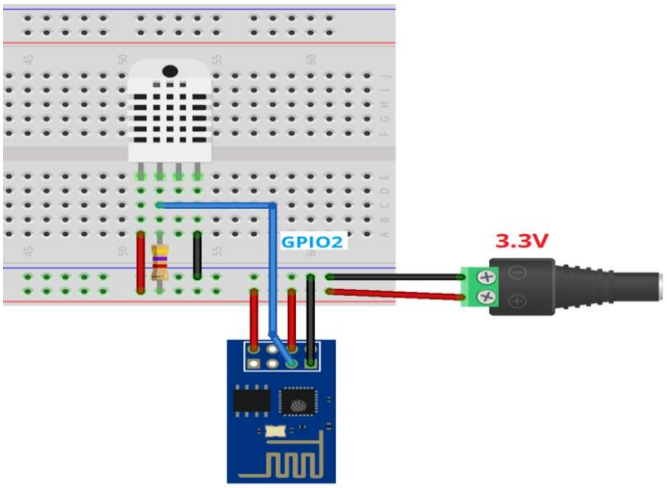
Por otro lado este proyecto es susceptible de ser desarrollado incorporando camaras, aumentando numero y tipo de sensores/ actuadores, creando una red de varios esp01 (modo master/slave).

Planos electricos.

Circuito del gravador.



Circuito del detector emisor.



BILIOGR@FIA

http://www.playbyte.es/electronica/arduino/esp-01s-modulo-wifi-basado-en-esp8266/

https://naylampmechatronics.com/blog/56_usando-esp8266-con-el-ide-de-arduino.html

https://arduino.org

https://randomnerdtutorials.com

https://lluisllamas.es

https://github.com

https://annefou.github.io/IoT_introduction/02-ESP8266/index.html

https://techstudycell.com