Smart Open Lab

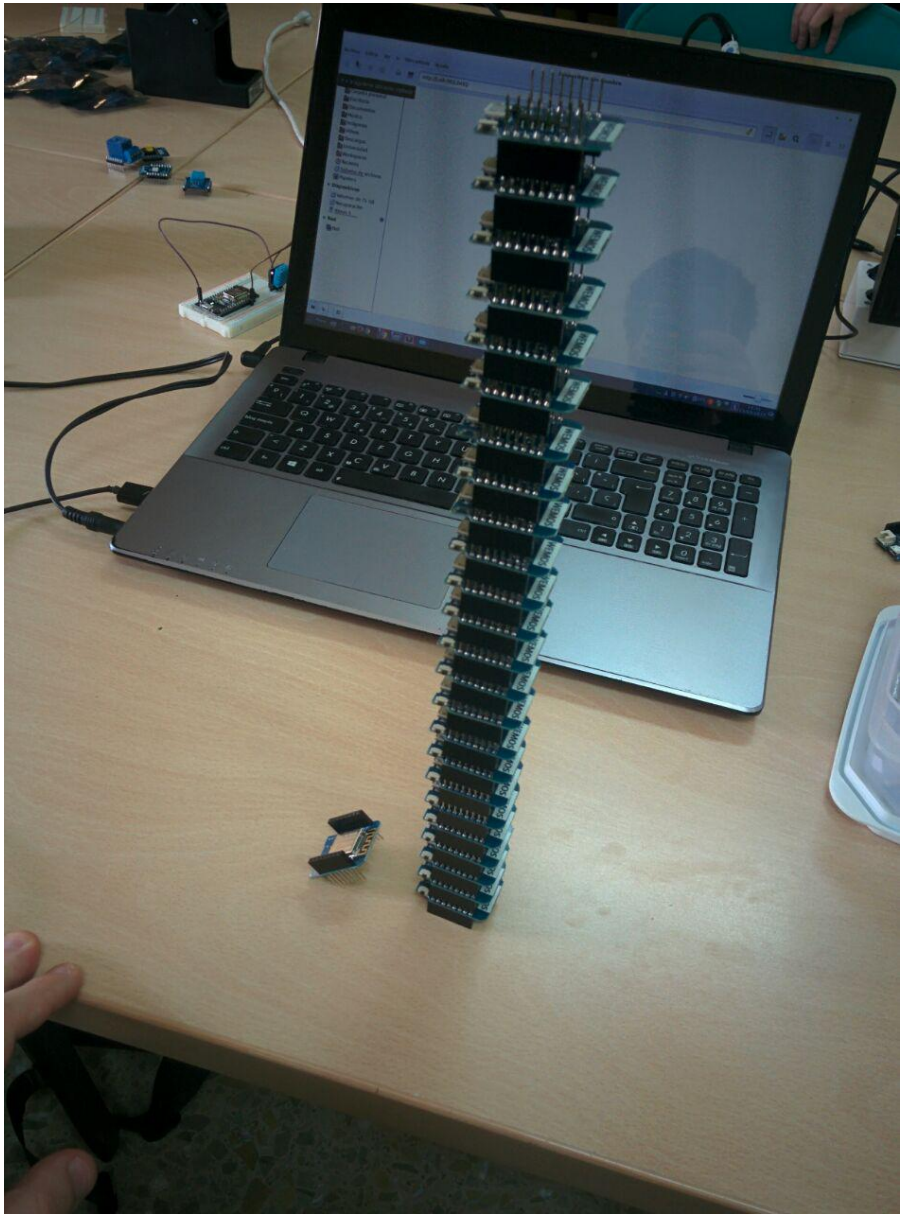
Curso Wemos (Arduino)

Internet of Things

#IoT

ARDUINO

Wemos es una placa que usa el chip ESP8266



- **Chip WiFi por unos 5€**
- **11 I/O pines**
  - Todos (excepto D0) soportan interrupciones, PWM, I2C y one-wire
- **1 entrada analógica**
- **Micro USB**
  - (alimentación y datos)
- **Compatible:**
  - Arduino
  - Nodemcu

Vamos a por la WiFi

- **Queremos hacer un programa que escaneé todas las redes WiFi disponibles, las muestre por consola y nos permita interactuar para seleccionar una, introducir la clave y conectarnos a una de ellas (obteniendo IP)**
- **#include "ESP8266WiFi.h"**
  - El objeto WiFi es quien tiene ahora toda la información sobre las redes
- **En el setup()**
  - Poner el módulo WiFi en modo station y desconectar de cualquier red a la que pudiese estar previamente conectado

```
WiFi.mode(WIFI_STA);
WiFi.disconnect();
```

Vamos a por la WiFi

EJ1

- **En el loop()**
  - Escaneando el entorno:

    int n = WiFi.scanNetworks(); // n redes detectadas

  - Con un bucle “for” podemos recorrer todas ellas

    Serial.print(WiFi.SSID(i)); //nombre de cada red

    Serial.print(WiFi.RSSI(i)); //potencia de cada red

  - Para saber si la red requiere password:

    if (WiFi.encryptionType(num) != ENC_TYPE_NONE)

Vamos a por la WiFi

- [Leer la entrada del usuario para seleccionar la red]

  num=Serial.readString().toInt();

- [Leer la entrada del usuario para introducir un password]

  while (!Serial.available()); //esperar hasta que haya algo tecleado en la consola

Vamos a por la WiFi

- En el loop()

```
WiFi.begin(WiFi.SSID(num).c_str(), password.c_str());
    //Si se conoce el user password simplemente WiFi.begin("user", "pass");
```

  - ¿Cómo sabemos cuando se ha logrado conectar?

```
while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
}
```

  - Imprimir la IP del ESP8266:

```
    Serial.println(WiFi.localIP());
```

Vamos a por la WiFi

EJ1

```
#include "ESP8266WiFi.h"
char num, caracter;
String password;

void setup() {
 Serial.begin(115200);
 WiFi.mode(WIFI_STA);
 WiFi.disconnect();
 delay(2000);
}

void loop() {
 Serial.println("Escaneando el entorno...");
 int n = WiFi.scanNetworks();
 if (n == 0)
  Serial.println("no hay redes en el rango de alcance");
 else{
  Serial.print(n);
  Serial.println(" redes en el rango del dispositivo");
  for (int i = 0; i < n; ++i)  {
   Serial.print(i);
   Serial.print(": ");
   Serial.print(WiFi.SSID(i));
   Serial.print(" (");
   Serial.print(WiFi.RSSI(i));
   Serial.print(")");
   Serial.println((WiFi.encryptionType(i) == ENC_TYPE_NONE)?" ":"*");
  }
```

Vamos a por la WiFi

EJ1

```
  Serial.print(n);
  Serial.println(": volver a escanear");


  while (!Serial.available());
  num = Serial.read() - 48;


  if (num!=n) {
   if (WiFi.encryptionType(num) !=
ENC_TYPE_NONE){
    Serial.println("introduce el password:
");
    while (!Serial.available());
    password = Serial.readString();
   }
```

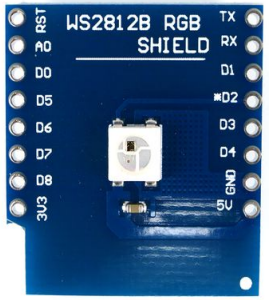
```
    Serial.print("Conectando a ");
    Serial.println(WiFi.SSID(num));
    WiFi.begin(WiFi.SSID(num).c_str(),
password.c_str());
    while (WiFi.status() != WL_CONNECTED) {
     delay(500);
     Serial.print(".");
    }
    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());


    while(1)
     yield();
   }
  }
}
```

Los ESP8266 funcionan también ¡como servers!

EJ2

- Vamos a encender remotamente una luz desde el móvil :-)

- Necesitamos la librería Adafruit Neopixel



- En la cabecera:

```
#define LED_GPIO D3
const char* ssid = "aqui el SSID";
const char* password = "aqui el password";
bool LED_estado = 0;
WiFiServer server(80); //objeto servidor en el puerto 80
```

- En el setup():

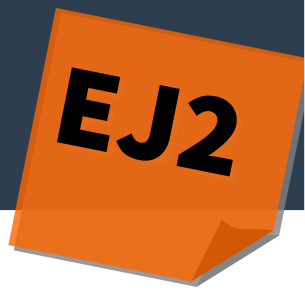
```
server.begin();
```

Los ESP8266 funcionan también ¡como servers!

EJ2

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

 WiFi.mode(WIFI_STA);
 WiFi.disconnect();
 delay(2000);

 pinMode(LED_Pin, OUTPUT);

 Serial.print("Conectando a ");
 Serial.println(ssid);
 WiFi.begin(ssid, password);

 while (WiFi.status() != WL_CONNECTED)
 {
  delay(500);
  Serial.print(".");
 }
 Serial.print("Conectado, la IP es: ");
 Serial.println(WiFi.localIP());

 server.begin();
 Serial.println("Servidor iniciado");
}
```

Los ESP8266 funcionan también ¡como servers!



- **En el loop():**
  - Si no se ha conectado un cliente reiniciar

```
WiFiClient client = server.available();
if (!client)
  return;
```

  - Esperar hasta que el cliente envíe algún dato

```
while(!client.available())
  yield();   //delay(1);
```

  - Lee la request del cliente

```
String request = client.readStringUntil('\r');
client.flush(); //vaciamos por seguridad
```

Los ESP8266 funcionan también ¡como servers!

EJ2

Completar! (falta el código para apagar)

- **En el loop():**
  - Request es un objeto String que contiene la URL completa, incluyendo parámetros. Buscamos si contiene la cadena "/LED/ON"

```
if (request.indexOf("/LED/ON") != -1){
   //hacer algo con esa URL
  //activar LED,...
}
```

- Parte "response" del servidor hacia el cliente, una página Web

```
client.println("HTTP/1.1 200 OK");
client.println("Content-Type: text/html");
client.println(""); //requerido
client.println("<!DOCTYPE HTML>");
client.println("<html>");

client.print("<p>LED ");
 if(LED_Estado)
  client.print("encendido</p>");
...
 client.println("<a
href="/LED/ON"><button>Encender</button><
/a>");
...
 client.println("</html>");
 Serial.println("Client disconnected");
```

Los ESP8266 funcionan también ¡como servers!

EJ2

```
#include <ESP8266WiFi.h>
#define LED_Pin D2
bool LED_Estado=0;

#include <Adafruit_NeoPixel.h>
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(1, LED_Pin,
NEO_GRB + NEO_KHZ800);

const char* ssid = "marinoAP";
const char* password = "eraseunavez";

WiFiServer server(80); //objeto servidor que
inicializaremos en el puerto 80

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

 pixels.begin();

 // Poner el módulo WiFi en modo station y
desconectar de cualquier red a la que pudiese
estar previamente conectado
 WiFi.mode(WIFI_STA);
 WiFi.disconnect();
 delay(2000);

 pinMode(LED_Pin, OUTPUT);

 //Conectar a la red WiFi
 Serial.println();
 Serial.print("Conectando a ");
 Serial.println(ssid);

 WiFi.begin(ssid, password);
```

Los ESP8266 funcionan también ¡como servers!

EJ2

```
 while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
 }
 Serial.print("Conectado, la IP del dispositivo es: ");
 Serial.println(WiFi.localIP());

 for(int i = 3; i>0; i--){
  pixels.setPixelColor(0, pixels.Color(0,150,0));
  pixels.show();
  delay(500);
  pixels.setPixelColor(0, pixels.Color(0,0,0));
  pixels.show();
  delay(500);
 }

 //Iniciar server
 server.begin();
 Serial.println("Servidor iniciado");
}
```

```
void loop() {
 //chequeamos si se ha conectado un cliente, en caso contrario terminar (se reiniciaría)
 WiFiClient client = server.available();
 if (!client)
  return;

 //Esperar hasta que el cliente envíe algún dato
 while(!client.available())
  yield();

 //Lee la request del cliente
 String request = client.readStringUntil('\r');
 client.flush(); //vacía por seguridad

 //Request en un objeto String que contiene la URL compelta, incluyendo parámetros. Buscamos si contien la cadena "/RELE=ON"
 if (request.indexOf("/LED/ON") != -1){
  pixels.setPixelColor(0, pixels.Color(150,0,150));
  //debería ser HIGH, pero funciona en modo inverso en algunas placas, includo la nodemcu
  pixels.show();
  LED_Estado=1;
  Serial.println("Encendido");
 }
```

Los ESP8266 funcionan también ¡como servers!

EJ2

```
else
 if (request.indexOf("/LED/OFF") != -1){
  pixels.setPixelColor(0, pixels.Color(0,0,0));
  pixels.show();
  LED_Estado=0;
  Serial.println("Apagado");
 }
 else
  if (request.indexOf("/LED/TOGGLE") != -1){
   if (LED_Estado==0)
    pixels.setPixelColor(0, pixels.Color(150,0,150));
   else
    pixels.setPixelColor(0, pixels.Color(0,0,0));
   pixels.show();
   LED_Estado=!LED_Estado;
   Serial.println("Conmutado");
  }
```

```
 client.println("HTTP/1.1 200 OK");
 client.println("Content-Type: text/html");
 client.println(""); //requerido
 client.println("<!DOCTYPE HTML>");
 client.println("<meta name='viewport' content='width=device-width, user-
scalable=no'>");
 client.println("<html>");

 client.print("<p>LED ");
 if(LED_Estado)
  client.print("encendido</p><br>");
 else
  client.print("apagado</p><br>");
 client.println("<a href=\"/LED/ON\"><button
style='width:100%;height:50px;background:#6C0;'>Encender</button></a>
");
 client.println("<a href=\"/LED/OFF\"><button
style='width:100%;height:50px;background:#C00;'>Apagar</button></a>");
 client.println("<a href=\"/LED/TOGGLE\"><button
style='width:100%;height:50px;background:#09F;'>Conmutar</button></a>
");
 client.println("</html>");
 delay(1);
 Serial.println("Cliente desconectado");
 Serial.println("");
}
```

Modos de ahorro de energía en ESP8266

	Modem-sleep	Light-sleep	Deep-sleep
Wi-Fi	OFF	OFF	OFF
System clock	ON	OFF	OFF
RTC	ON	ON	ON
CPU	ON	Pendiente	OFF
Corriente	~15 mA	~0.4 mA	~20 µA

Automáticamente activo cuando conectado a un AP y en modo STA WiFi.mode(WIFI_STA)

Automáticamente activo cuando conectado a Wi-Fi y la CPU sin uso. Se puede despertar con interrupciones

Solo se puede despertar reseteándolo (WDT o Pin RST)

ESP.deepSleep(sleepTimeuS);

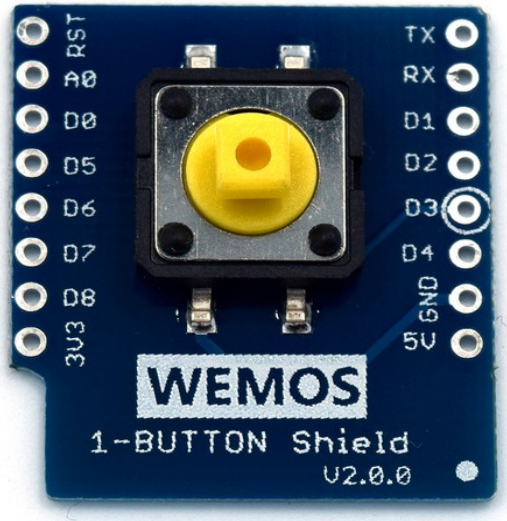
Enviar datos a Internet y manejando interrupciones para...

EJ3

- **… lograr que se registre en un servidor (thingspeak) cada pulsación de un botón**

- **Consejos:**
  - Las interrupciones deben ser lo más cortas posibles, si es posible simplemente “seteando” banderas
  - Para añadir interrupciones en arduino:

    attachInterrupt(GPIO_Pin, funcionAEjecutar, cuando);
    - //cuando = CHANGE, RISING, FALLING
    - Lógica inversa



Usad el modo de ahorro LIGHT para...

EJ3

- **… que el ESP8266 espere dormido y al presionar el botón se active de nuevo**

  while (condición) ← ¿¿¿¿¿¿ usando BUTTON_state ???????

    yield();

  - Para controlar las veces que se ejecuta loop(), probar a incluir:

    Serial.println("llamada control en loop()");

Usad el modo de ahorro LIGHT para...

EJ3

```
#include <ESP8266WiFiMulti.h>


// ThingSpeak Settings
const int channelID = 343024;
String writeAPIKey = "IIMUPX6LPZ8X0V1I";
const char* server = "api.thingspeak.com";
const int postingInterval = 20 * 1000; // post data every 20 seconds


ESP8266WiFiMulti WiFiMulti;
WiFiClient client;


bool BUTTON_state = false;
void button_pressed() {
 BUTTON_state = true;
}
```

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


 WiFiMulti.addAP("Orange-152A", "8EE529C4");
 WiFiMulti.addAP("marinoAP", "eraseunavez");
//tantos como queramos, conecta al mejor RSSI


 while(WiFiMulti.run() != WL_CONNECTED) {
  Serial.print(".");
  delay(500);
 }
 attachInterrupt(D3, button_pressed, FALLING); //CHANGE, RISING, FALLING Lógica inversa


 Serial.println("");
}
```

Usad el modo de ahorro LIGHT para...

EJ3

```
void loop() {
 if (BUTTON_state) {
  if (client.connect(server, 80)) {
   Serial.println("conectado al servidor");

   // API request body
   String body = "field1=1";

   client.print("POST /update HTTP/1.1\n");
   client.print("Host: api.thingspeak.com\n");
   client.print("Connection: close\n");
   client.print("X-THINGSPEAKAPIKEY: " +
writeAPIKey + "\n");
   client.print("Content-Type: application/x-
www-form-urlencoded\n");
   client.print("Content-Length: ");
   client.print(body.length());
   client.print("\n\n");
   client.print(body);
   client.print("\n\n");
  }
  if (!client.connected())
   client.stop();
 }
 BUTTON_state=false;

 while (!BUTTON_state)
  yield();

 Serial.println("botón pulsado");
}
```