

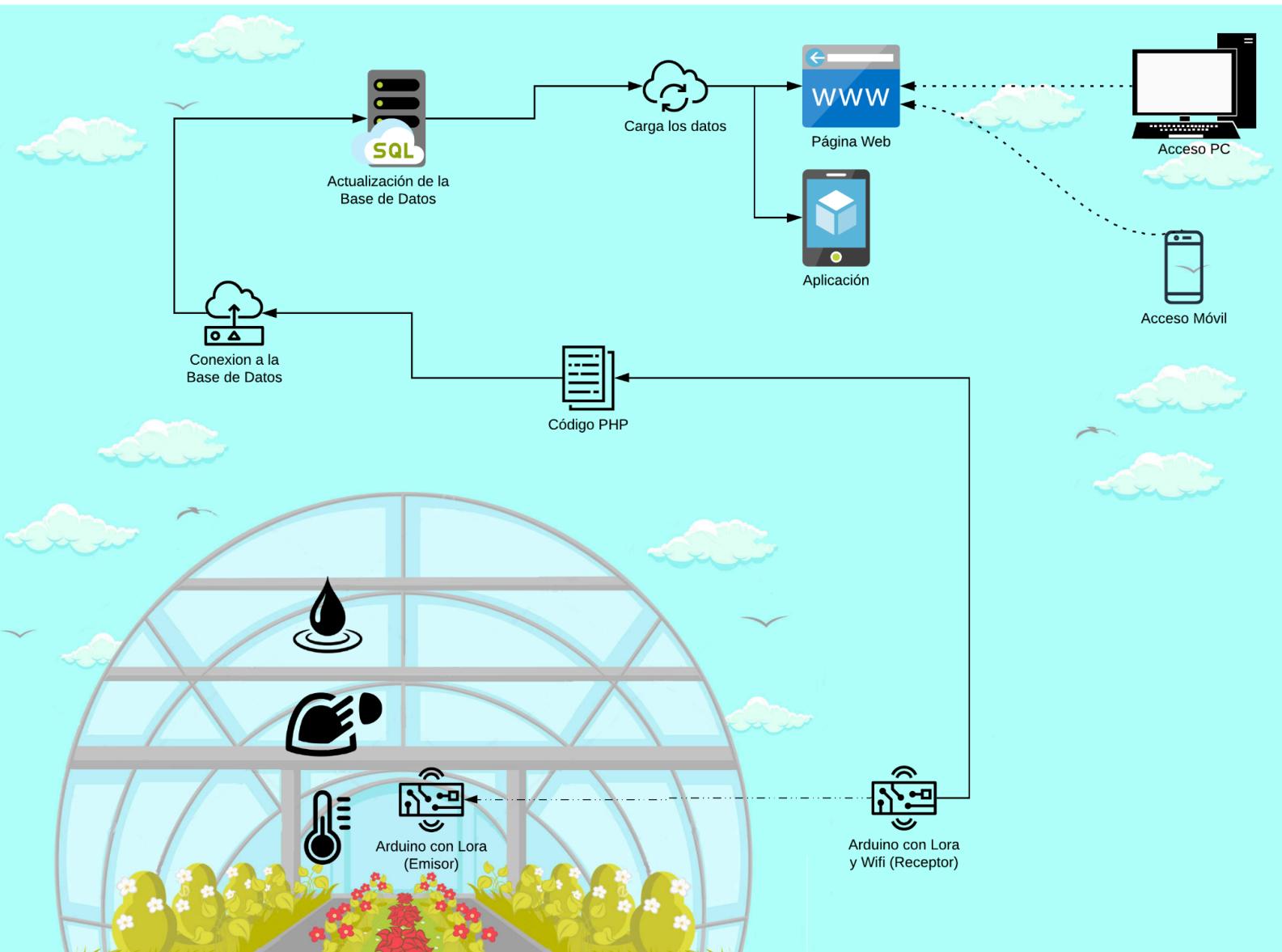
Arduino LaGranja

Índice

Índice	2
Esquema del Proyecto la Granja 4.0	4
Conceptos básicos de la tecnología LoRa	5
Conexión entre emisor y receptor ESP32 con LoRa usando Arduino-IDE	5
Instalación	5
Prueba de conexión y ejecución	7
Prueba de subida de código con ESP8266	8
Comprobamos que el programa se ejecuta correctamente	10
Modificamos el programa para que nos imprima cualquier cosa en el Monitor Serie	11
Actualizaciones inalámbricas (OTA) en la placa ESP32	12
Instalar la biblioteca AstynElegantOTA	12
Instalar las bibliotecas AsyncTCP y ESPAsyncWebServer	13
Resuelto el problema de subida de código	18
Resuelto el problema de subida de código II.	19
Implementación del código Arduino	20
Resultado de la conexión ESP32 receptor con LoRa	22
Muestras del código Arduino y PHP y del resultado en la Base de Datos	23
Probando a subir contenido a la Base de Datos	28
Creamos base de datos	28
Capturas de la ejecución del código y del resultado en la Base de Datos	31
Conexión de sensor de Agua	35
Interfaz del sensor de nivel de agua con Arduino	35
Resultado de la conexión del sensor de agua	36
Capturas de la ejecución del código y del resultado en la Base de Datos	37
Subir datos a la API	49
Capturas de la ejecución del código y del resultado en la API	49
Conexión de sensor de Temperatura y Humedad DHT11	64
Esquema de conexión	64
Librerías para añadir el sensor de temperatura	66
Capturas de la ejecución del código y del resultado en la API	67
Conexión del sensor de luz	76
Esquema de conexión	76
Capturas de la ejecución del código y del resultado en la API	78
Actuador	94
Esquema de conexión	94
Código para manejar el LED	95

Bibliografía	96
API de LoRa	96
ESP32 Hardware Design	96
Comenzando con ESP32	96
Instalación de la placa ESP32 en Arduino IDE	96
Para la placa ESP32 el driver es CP210x	96
ESP32 con LoRa usando Arduino IDE	96
Error al conectarse a ESP32	96
ESP8266	96
Para la placa ESP8266 el driver es CH341	96
Prueba de parpadeo con la placa ESP8266	96
Conectar sensor de agua	97
Cómo funciona el sensor de nivel de agua	97
Conectar sensor de temperatura	97
Conectar sensor de temperatura DHT11	97
Conectar sensor de luz	97
Explicación sensor de luz	97
Lectura de valores del sensor de luz	97
Conectar sensor bme280 de Temperature, Humidity, and Pressure	97
Otro ejemplo de estación meteorológica, usando pila	97
Estación meteorológica ESP32	97
Aplicación Fritzing para hacer los conexión de los sensores	97
Encender LEDs para el actuador	98
Cómo conectar una antena externa a ESP32	98

Esquema del Proyecto la Granja 4.0



Este esquema trata de explicar mediante una imagen como sería el resultado final del proyecto.

En primer lugar tendríamos un emisor en el invernadero y un receptor en el instituto de la Granja. El emisor recogerá la información de 3 sensores (temperatura, luz y agua). Esos datos tratarán de enviarse al receptor mediante LoRa.

Una vez el receptor haya recibido los datos de los sensores los subiría a la API haciendo un POST gracias a un archivo externo .php ya que está conectado a Internet mediante WiFi.

Conceptos básicos de la tecnología LoRa

- LoRa es una técnica de modulación de radio
- LoRa permite la comunicación a larga distancia de pequeñas cantidades de datos y requiere poca energía
- Puede utilizar LoRa en la comunicación punto a punto o en una red
- LoRa puede ser especialmente útil si desea monitorear sensores que no están cubiertos por su red Wi-Fi y que están separados por varios metros.

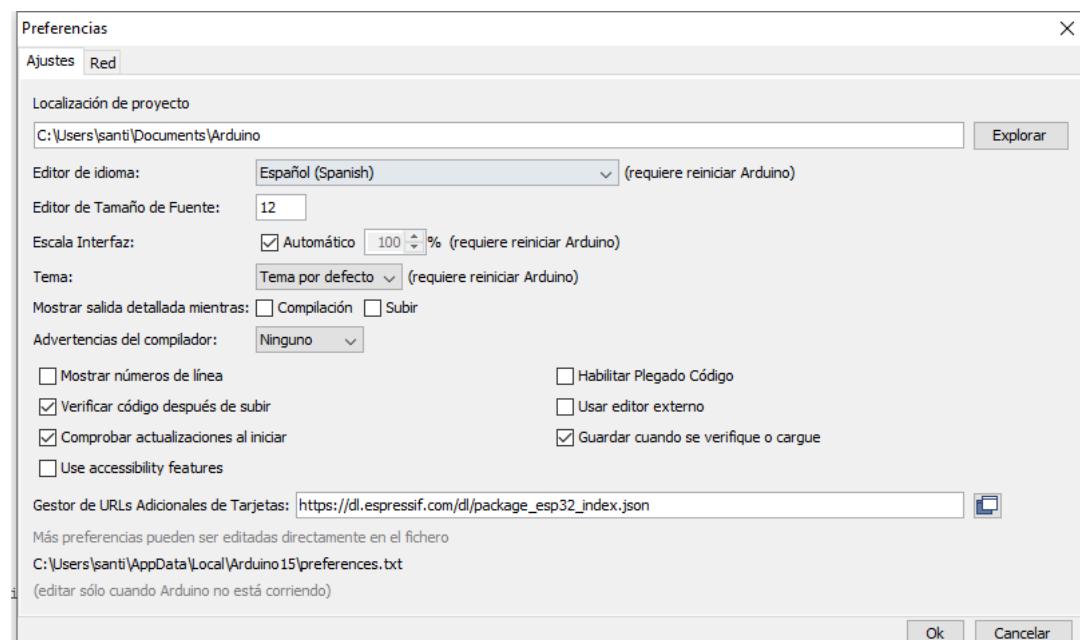
También le mostramos cómo construir un remitente LoRa y un receptor LoRa simples. Estos son solo ejemplos simples para comenzar con LoRa.

Conexión entre emisor y receptor ESP32 con LoRa usando Arduino-IDE

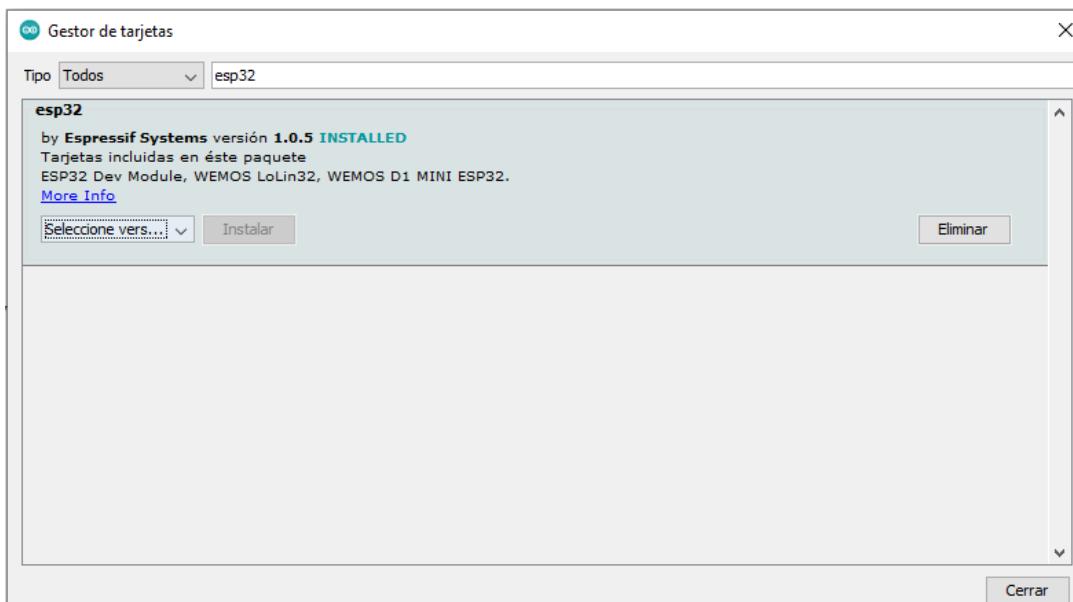
Instalación

En preferencias incluimos el siguiente link

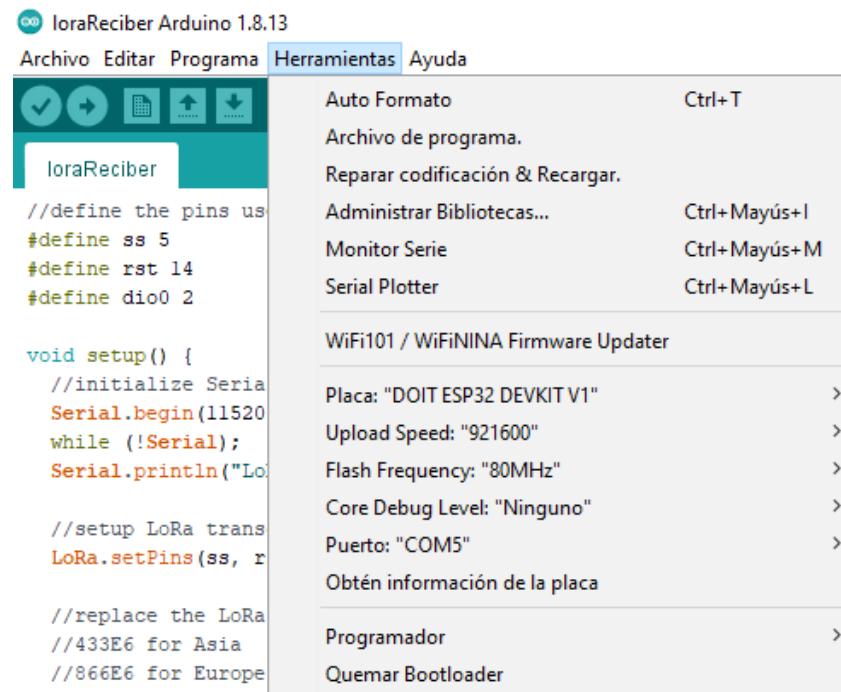
https://dl.espressif.com/dl/package_esp32_index.json



Herramientas > Placa > Gestor de placas...



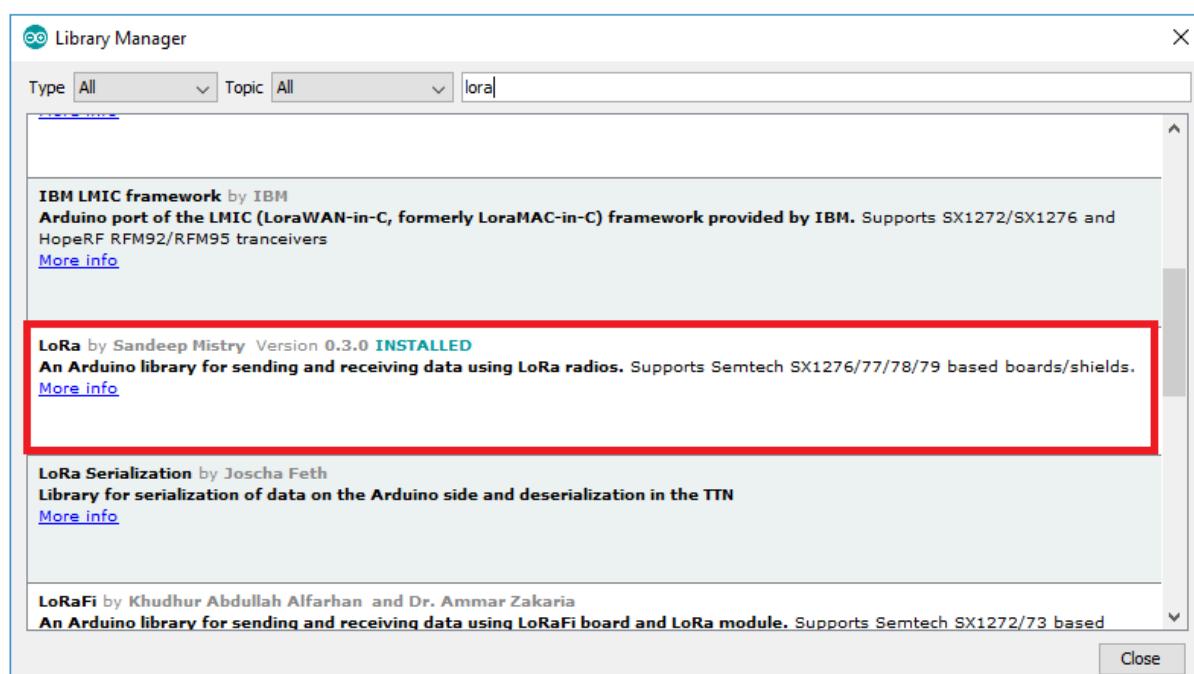
Seleccionamos nuestra placa “DOIT ESP32 DEVKIT V1” y el puerto que sea equivalente



Si no ve el puerto COM en su IDE de Arduino, debe instalar los controladores CP210x USB a UART Bridge VCP

Añadimos la librería LORA

Herramientas → administrar bibliotecas → buscamos la versión que aparece en rojo abajo.



Prueba de conexión y ejecución

Como en nuestro caso el código venía instalado solo tenemos que mostrar la conexión y ejecución de las tarjetas usando el monitor serie con cada placa seleccionando el puerto.

Pulsamos el botón de la placa "en", que significa "enable" para habilitarlo y que se conecten.

La conexión es instantánea, podemos comprobar que nos devuelve los mensajes.

```
COM5 - X Enviar
17:04:51.713 -> ets Jun 8 2016 00:22:57
17:04:51.713 ->
17:04:51.713 -> rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
17:04:51.713 -> configsip: 0, SPIWP:0xee
17:04:51.713 -> clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
17:04:51.713 -> mode:DIO, clock div:1
17:04:51.713 -> load:0x3fff0018,len:4
17:04:51.713 -> load:0x3fff001c,len:928
17:04:51.713 -> ho 0 tail 12 room 4
17:04:51.713 -> load:0x40078000,len:9280
17:04:51.761 -> load:0x40080400,len:5848
17:04:51.761 -> entry 0x40080698
17:04:51.854 -> LoRa Receiver
17:04:51.900 -> LoRa Initializing OK!
17:06:19.525 -> Received packet 'hello kkkkkk nnnnnn 1111 123456780' with RSSI -47
17:06:29.616 -> Received packet 'hello kkkkkk nnnnnn 1111 123456781' with RSSI -42
17:06:39.673 -> Received packet 'hello kkkkkk nnnnnn 1111 123456782' with RSSI -42
17:06:49.742 -> Received packet 'hello kkkkkk nnnnnn 1111 123456783' with RSSI -42
17:06:58.767 -> Received packet 'hello kkkkkk nnnnnn 1111 123456780' with RSSI -40
17:07:08.865 -> Received packet 'hello kkkkkk nnnnnn 1111 123456781' with RSSI -43
17:07:18.943 -> Received packet 'hello kkkkkk nnnnnn 1111 123456782' with RSSI -43
17:07:28.975 -> Received packet 'hello kkkkkk nnnnnn 1111 123456783' with RSSI -42
```

Autoscroll Mostrar marca temporal Ambos NL & CR 115200 baudio Limpiar salida

```
COM6 - X Enviar
17:06:49.662 -> Sending packet: 3
17:06:58.504 -> ets Jun 8 2016 00:22:57
17:06:58.504 ->
17:06:58.504 -> rst:0x1 (POWERON_RESET),boot:0x1b (SPI_FAST_FLASH_BOOT)
17:06:58.504 -> configsip: 0, SPIWP:0xee
17:06:58.504 -> clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
17:06:58.550 -> mode:DIO, clock div:1
17:06:58.550 -> load:0x3fff0018,len:4
17:06:58.550 -> load:0x3fff001c,len:928
17:06:58.550 -> ho 0 tail 12 room 4
17:06:58.550 -> load:0x40078000,len:9280
17:06:58.550 -> load:0x40080400,len:5848
17:06:58.550 -> entry 0x40080698
17:06:58.688 -> LoRa Sender
17:06:58.688 -> LoRa Initializing OK!
17:06:58.688 -> Sending packet: 0
17:07:08.788 -> Sending packet: 1
17:07:18.866 -> Sending packet: 2
17:07:28.899 -> Sending packet: 3
17:07:39.004 -> Sending packet: 4
17:07:49.084 -> Sending packet: 5
```

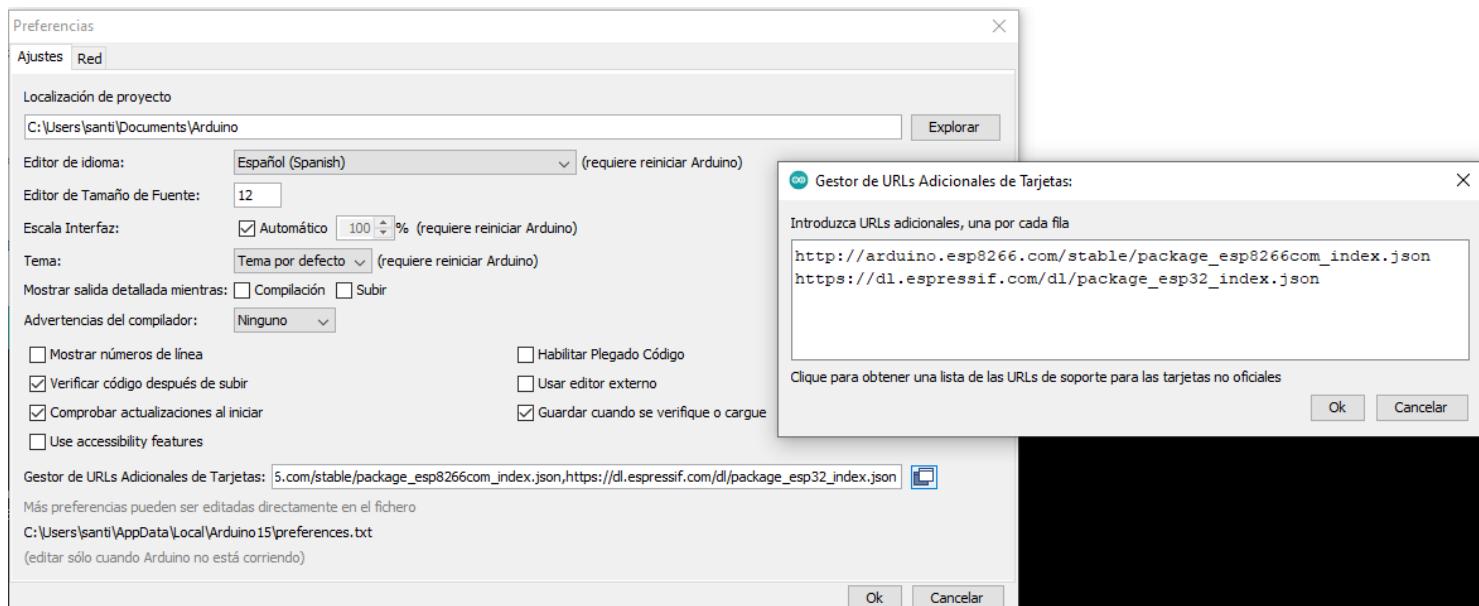
Autoscroll Mostrar marca temporal Ambos NL & CR 115200 baudio Limpiar salida

Prueba de subida de código con ESP8266

Hemos decidido hacer una prueba con la placa ESP8266 porque con la anterior placa, la ESP32, tenía código grabado anteriormente por un profesor. Entonces hemos decidido introducir un código de pruebas para probar la subida de código.

En primer lugar, insertamos éste nuevo URL adicionar para el gestor de tarjetas:
“http://arduino.esp8266.com/stable/package_esp8266com_index.json”

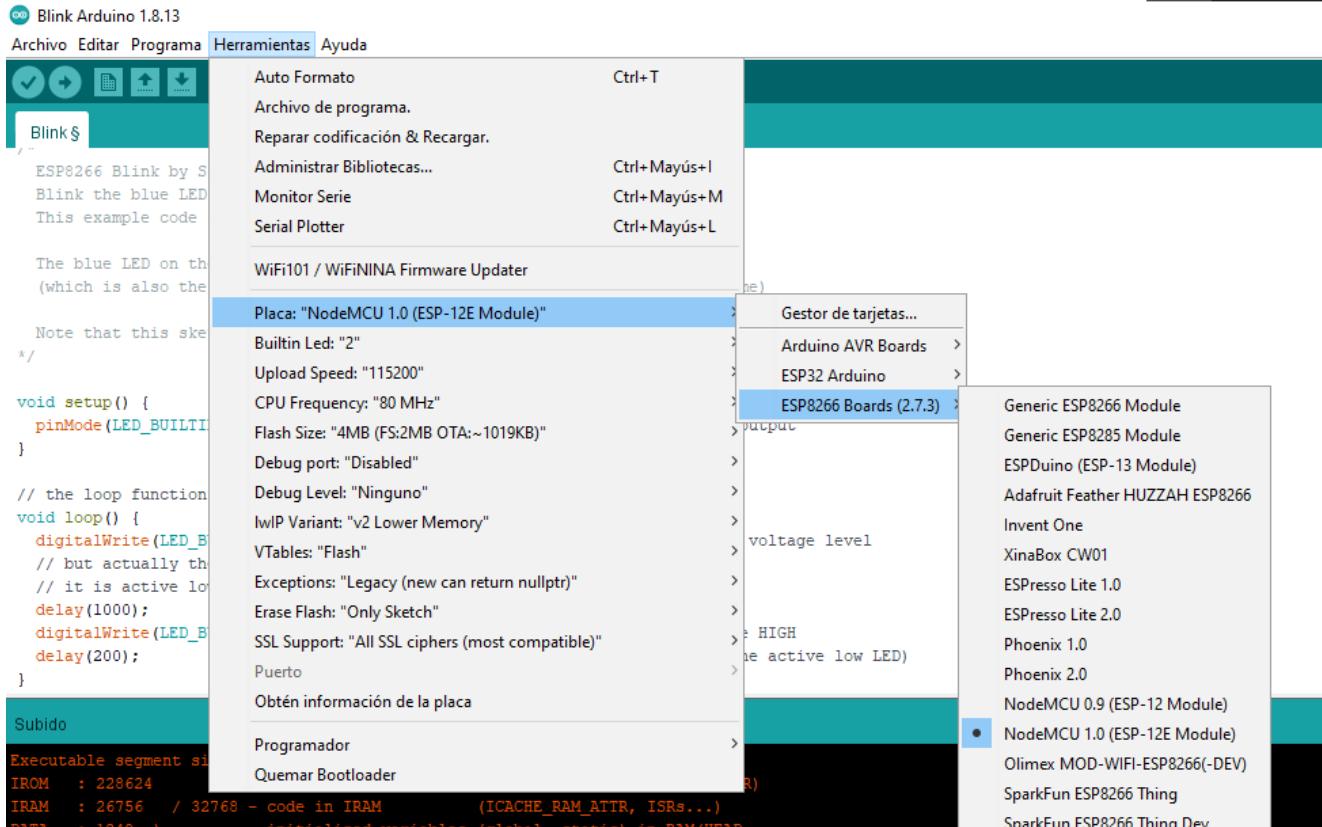
Es importante no eliminar la URL anterior correspondiente a la tarjeta ESP32 ya que si no lo incluimos, no nos reconocerá la tarjeta ESP32 que necesitaremos en otra ocasión.



Cuando hayamos incluido la URL accederemos a Herramientas > Placa y buscaremos el nombre “ESP8266” en el gestor de tarjetas y lo instalamos.

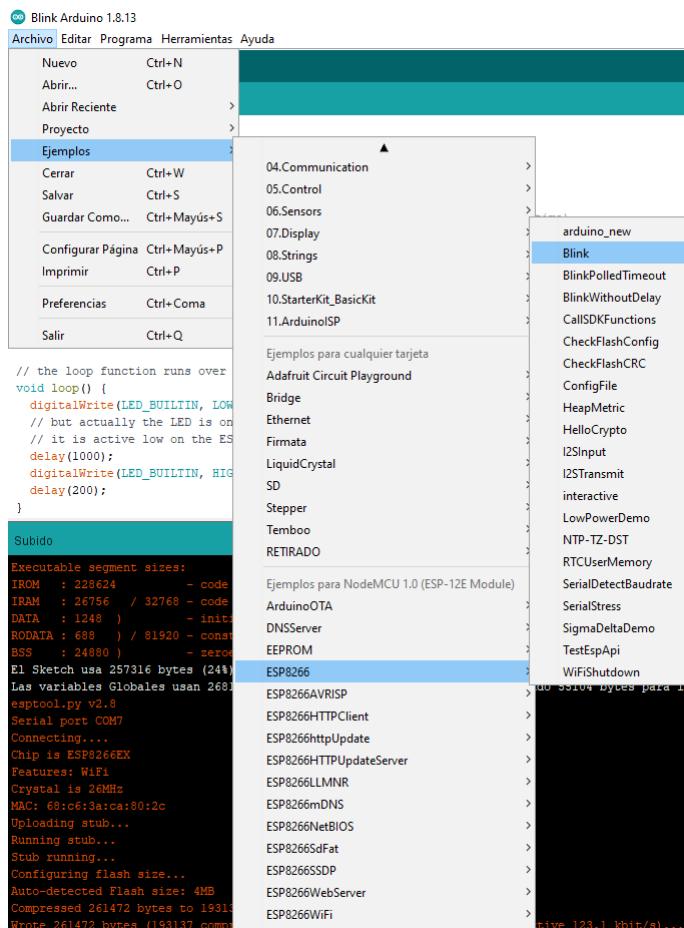


Cuando hayamos incluido la URL accederemos a Herramientas > Placa > ESP8266 y seleccionamos "NodeMCU 1.0 (ESP-12E Module)". También seleccionaremos el puerto correspondiente. Después de seleccionar la tarjeta adecuada, se autocompleta la selección de las demás configuraciones.



Una vez esté todo bien seleccionado accederemos a Archivos > Ejemplos > ESP8266 > Blink y ejecutaremos la prueba.

Esta es una simple prueba para hacer parpadear el led azul que tiene la placa.



Comprobamos que el programa se ejecuta correctamente

The screenshot shows the Arduino IDE interface with the following details:

- Top Bar:** Shows "Blink Arduino 1.8.13" and menu items: Archivo, Editar, Programa, Herramientas, Ayuda.
- Toolbar:** Includes icons for back, forward, file operations, and upload.
- Title Bar:** Displays "Blink \$".
- Code Area:** Contains the "Blink8266" sketch by Simon Peter. The code initializes the LED_BUILTIN pin as an output and uses a loop to alternate between LOW and HIGH states, with a two-second delay between each state change.
- Output Area:** Labeled "Subido", it displays the memory usage report and the upload log. The memory report shows:

Segment	Size	Description
IROM	228624	- code in flash (default or ICACHE_FLASH_ATTR)
IRAM	26756 / 32768	- code in IRAM (ICACHE_RAM_ATTR, ISRs...)
DATA	1248	- initialized variables (global, static) in RAM/HEAP
RODATA	688	- constants (global, static) in RAM/HEAP
BSS	24880	- zeroed variables (global, static) in RAM/HEAP

It also notes that the sketch uses 257316 bytes (24%) of program storage space, leaving 1044464 bytes available. The log shows the upload process starting on serial port COM7, identifying the chip as ESP8266EX, and successfully writing 261472 bytes at 193137 compressed to 193137 in 17.0 seconds at 123.1 kbit/s. A hash of the data was verified.
- Bottom Bar:** Shows the page number "23".

Modificamos el programa para que nos imprima cualquier cosa en el Monitor Serie

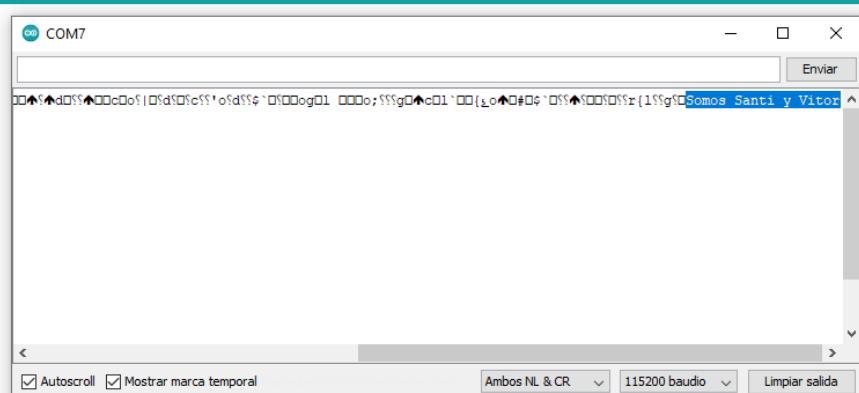
Blink Arduino 1.8.13

Archivo Editar Programa Herramientas Ayuda



Blink \$

```
/*  
 * ESP8266 Blink by Simon Peter  
 * Blink the blue LED on the ESP-01 module  
 * This example code is in the public domain  
  
The blue LED on the ESP-01 module is connected to GPIO1  
(which is also the TXD pin; so we cannot use Serial.print() at the same time)  
  
Note that this sketch uses LED_BUILTIN to find the pin with the internal LED  
*/  
  
void setup() {  
    pinMode(LED_BUILTIN, OUTPUT);      // Initialize the LED_BUILTIN pin as an output  
    // Inicio el Monitor Serie  
    Serial.begin(115200); // Baudios a los que va a funcionar  
    while (!Serial);  
    Serial.println("Somos Santi y Vitor"); // Imprimo el texto que quiero mostar  
}  
  
// the loop function runs over and over again forever  
void loop() {  
    digitalWrite(LED_BUILTIN, LOW); // Turn the LED on (Note that LOW is the voltage level  
    // but actually the LED is on; this is because  
    // it is active low on the ESP-01)
```



```
Executable segment sizes:  
IROM : 232968 - code in flash (default or ICACHE_FLASH_ATTR)  
IRAM : 27264 / 32768 - code in IRAM (ICACHE_RAM_ATTR, ISRs...)  
DATA : 1248 ) - initialized variables (global, static) in RAM/HEAP  
RODATA : 728 ) / 81920 - constants (global, static) in RAM/HEAP  
BSS : 24944 ) - zeroed variables (global, static) in RAM/HEAP
```

El Sketch usa 262208 bytes (25%) del espacio de almacenamiento de programa. El máximo es 1044464 bytes.

Las variables Globales usan 26920 bytes (32%) de la memoria dinámica, dejando 55000 bytes para las variables locales. El máximo es 81920 bytes.

```
esptool.py v2.8  
Serial port COM7  
Connecting...  
Chip is ESP8266EX
```

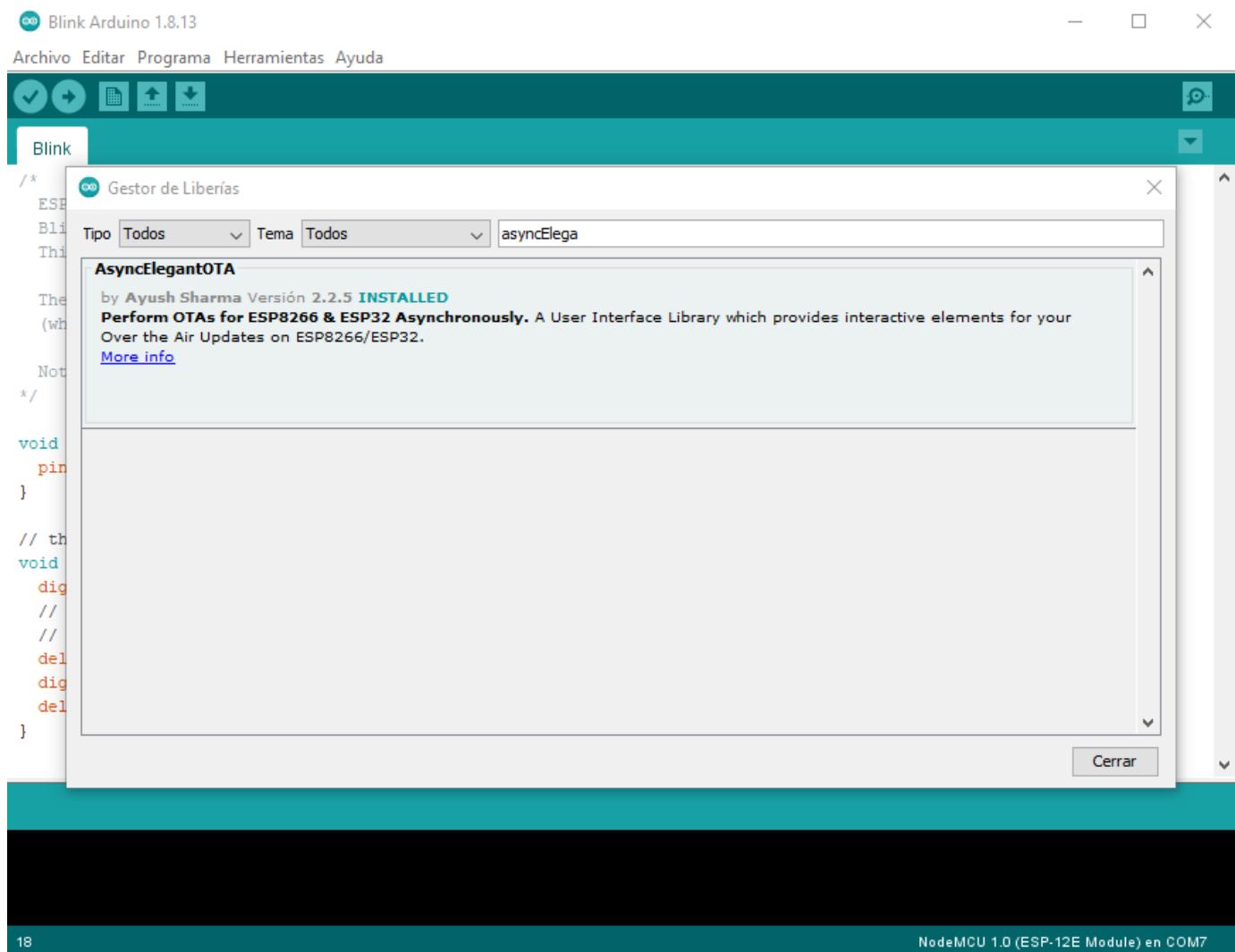
Actualizaciones inalámbricas (OTA) en la placa ESP32

Instalar la biblioteca AstynElegantOTA

Puede instalar la biblioteca AsyncElegantOTA utilizando Arduino Library Manager.

Accedemos a Programa > Incluir biblioteca > Administrar bibliotecas

Buscamos "AsyncElegantOTA" e instalamos.



Instalar las bibliotecas AsyncTCP y ESPAsyncWebServer

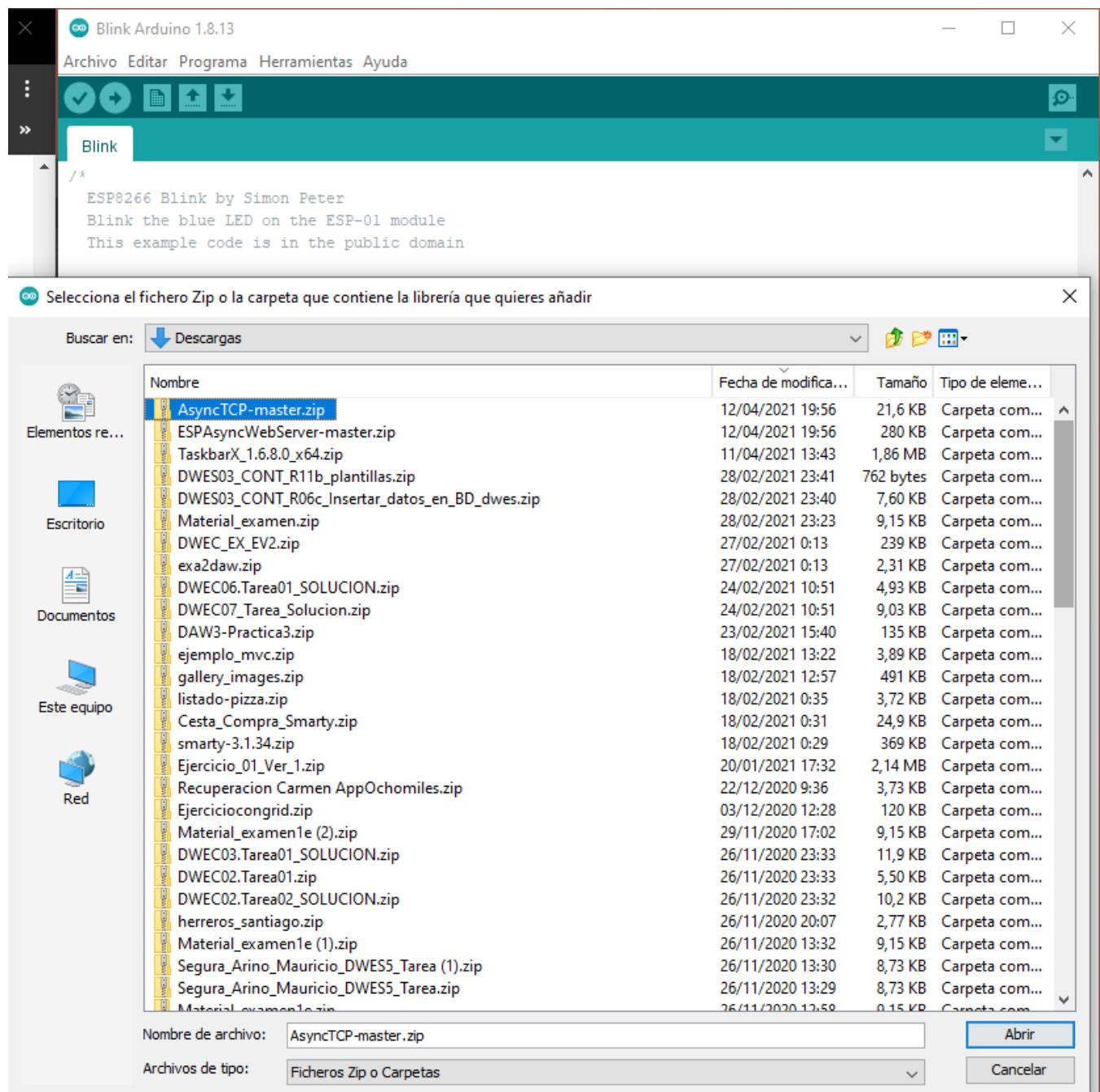
Después instalamos las bibliotecas AsyncTCP y ESPAsyncWebServer.

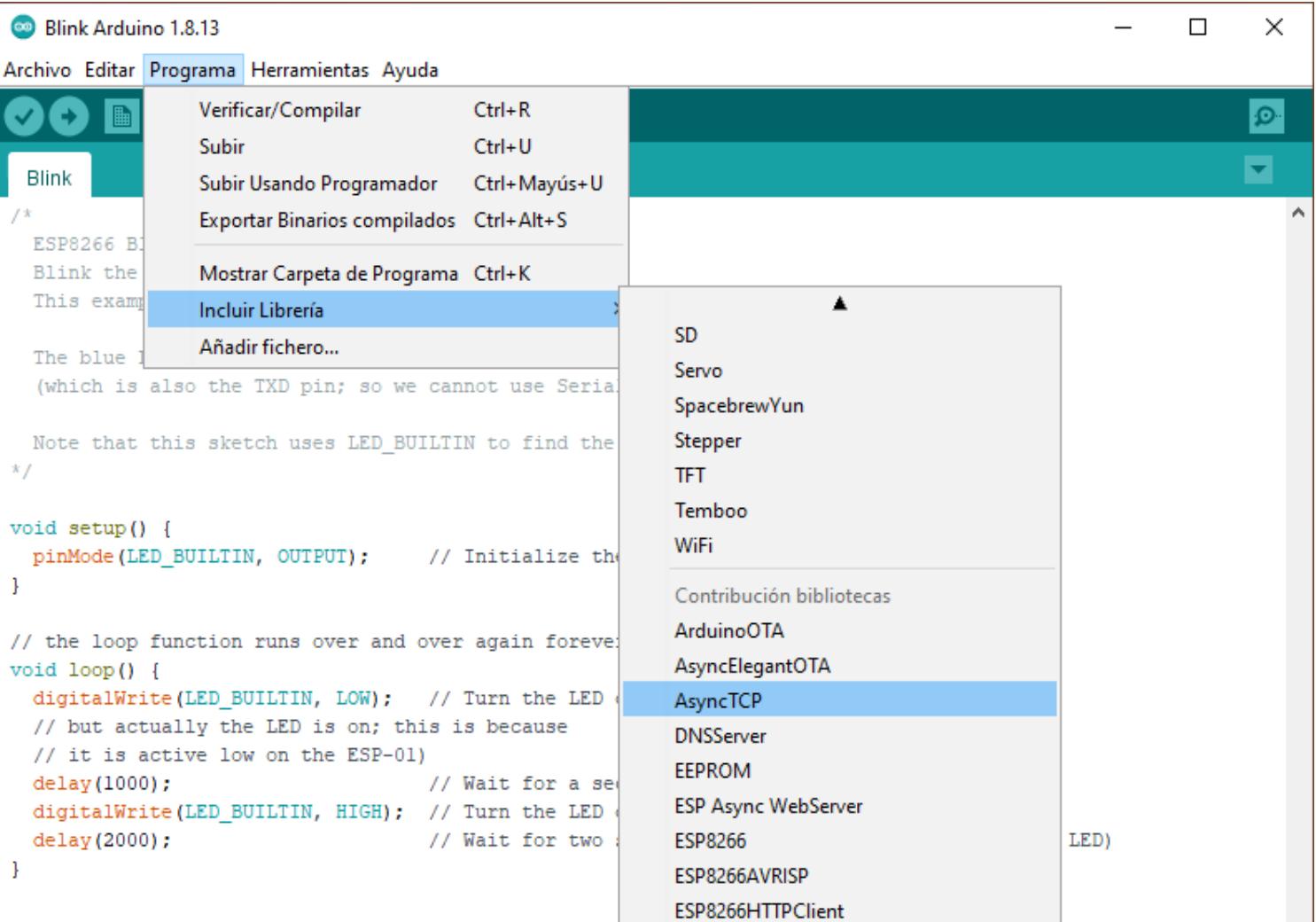
ESPAsyncWebServer

AsyncTCP

Estas bibliotecas no están disponibles para instalar a través del Administrador de bibliotecas de Arduino, por lo que hay que copiar los archivos de la biblioteca en la carpeta Bibliotecas de instalación de Arduino.

En IDE de Arduino, vamos a Programa > Incluir biblioteca > Agregar biblioteca .zip y seleccionamos las bibliotecas que acaba de descargar.





Copiamos el código y subimos el código a nuestra placa. Comprobamos su funcionamiento abriendo el monitor serie.

The screenshot shows the Arduino IDE interface. The top menu bar includes Archivo, Editar, Programa, Herramientas, and Ayuda. Below the menu is a toolbar with icons for upload, refresh, and other functions. The main window title is "simple_server". The code area contains the following C++ code:

```
#include <Arduino.h>
#include <WiFi.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include <AsyncElegantOTA.h>

const char* ssid = "MiFibra-962A";
const char* password = "travieso" ;

AsyncWebServer server(80);

void setup(void) {
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  Serial.println("");
}

// Wait for connection
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
}
Serial.println("");
Serial.print("Connected to ");
Serial.println(ssid);
Serial.print("IP address: ");
Serial.println(WiFi.localIP());

server.on("/", HTTP_GET, [] (AsyncWebServerRequest *request) {
  request->send(200, "text/plain", "Hi! I am ESP32.");
});

AsyncElegantOTA.begin(&server); // Start ElegantOTA
server.begin();
Serial.println("HTTP server started");


```

The serial monitor at the bottom displays the following output:

```
Writing at 0x00001000... (0%)
Writing at 0x00078000... (90%)
Writing at 0x0007c000... (93%)
Writing at 0x00080000... (96%)
Writing at 0x00084000... (100%)
Wrote 811632 bytes (485649 compressed) at 0x00010000 in 7.7 seconds (effective 839.2 kbit/s)...
Hash of data verified.
Compressed 3072 bytes to 128...
Writing at 0x00008000... (100%)
Wrote 3072 bytes (128 compressed) at 0x00008000 in 0.0 seconds (effective 1890.5 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

COM5

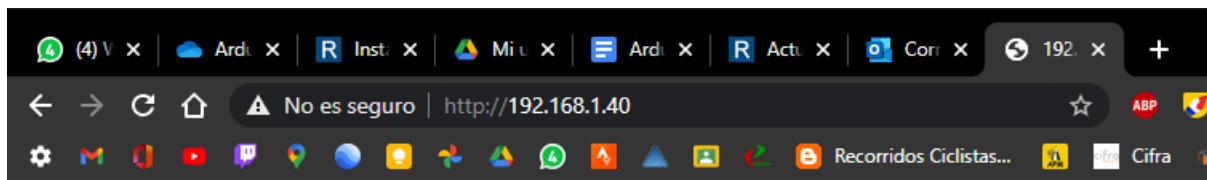
20:06:41.633 -> ets Jun 8 2016 00:22:57
20:06:41.633 ->
20:06:41.633 -> rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
20:06:41.633 -> configsip: 0, SPIWP:0xee
20:06:41.633 -> clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
20:06:41.633 -> mode:DIO, clock div:1
20:06:41.633 -> load:0x3fff0018,len:4
20:06:41.633 -> load:0x3fff001c,len:1216
20:06:41.633 -> ho 0 tail 12 room 4
20:06:41.633 -> load:0x40078000,len:10944
20:06:41.679 -> load:0x40080400,len:6388
20:06:41.679 -> entry 0x400806b4
20:06:42.045 ->
20:06:42.554 ->

20:06:45.041 -> Connected to MiFibra-962A
20:06:45.041 -> IP address: 192.168.1.40
20:06:45.088 -> HTTP server started

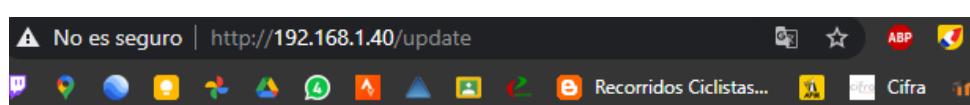
Autoscroll Mostrar marca temporal Ambos NL & CR 115200 baudio Limpiar salida

Comprobamos que funciona correctamente. Abrimos el navegador y escribimos la IP que nos aparece en el monitor serie.

Vemos el mensaje de bienvenida que está escrito en el código.



Ahora añadimos la dirección IP/"update". Debería cargarse la siguiente página web.



- Firmware
- Sistema de archivos

Seleccionar archivo

loraReceptorEmisor Arduino 1.8.13

Archivo Editar Programa Herramientas Ayuda

LORA.setSyncWord(0xFF);
 Serial.println("LoRa Initializing OK!");

void

//

int

if

11:45:54.019 -> ---- Se han recibido por LORA las siguientes parejas CLAVE => VALORLoRaData => Antonio 21
 11:45:54.019 -> ----
 11:45:54.019 -> bien :)
 11:45:54.019 -> Cerrando conexion
 11:46:54.039 -> Received packet 'Aquí va el loradatasentiiiiHolanda123' with RSSI -35
 11:46:54.039 -> s:
 11:46:54.039 -> XXXX Los parámetros son: LoRaData=Antonio 22
 11:46:54.039 -> Conectando a 192.168.1.111
 11:46:54.133 -> Requesting URL: /LaGranjaPruebaPHP/Servidor/prueba_arduino.phpHTTP/1.1 200 OK
 11:46:54.133 -> Date: Sat, 08 May 2021 09:46:54 GMT
 11:46:54.133 -> Server: Apache/2.4.46 (Win64) OpenSSL/1.1.1g PHP/7.4.10
 11:46:54.178 -> X-Powered-By: PHP/7.4.10
 11:46:54.178 -> Content-Length: 104
 11:46:54.178 -> Content-Type: text/html; charset=UTF-8
 11:46:55.154 ->
 11:46:55.154 -> ---- Se han recibido por LORA las siguientes parejas CLAVE => VALORLoRaData => Antonio 22
 11:46:55.154 -> ----
 11:46:55.154 ->
 11:46:55.154 -> bier :)
 11:46:55.154 -> Cerrando conexion
 11:47:46.713 -> ets Jun 8 2016 00:22:57
 11:47:46.713 ->
 11:47:46.713 -> rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
 count:0 configSip: 0, SPIWP:0xee

del Autoreload Mostrar marca temporal

Ambos NL & CR 115200 baudio Limpiar salida

tuninantonio Arduino 1.8.13

Archivo Editar Programa Herramientas Ayuda

Serial.println("LoRa Initializing OK!");

conectarWifi();

void loop()

{

// try to parse packet
 int packetSize = LoRa.parsePacket();
 if (packetSize) {
 // received a packet
 Serial.print("Received packet ");

// read packet
 while (LoRa.available()) {
 Serial.print("Aquí va el loradatasentiiii");
 String LoRaData = LoRa.readString();
 Serial.print(LoRaData);

// print RSSI of packet
 Serial.print(" with RSSI ");
 Serial.println(LoRa.packetRssi());

} else {
 Serial.print("ESTOY TRISTE :(");

// Preparar datos
 String LoRaData = "Antonio ";
 LoRaData = LoRaData + numero;
 String parametros = "LoRaData=" + LoRaData;
 Serial.println("El valor de LoRaData es: " + numero);
 Serial.println("XXXX Los parámetros son: " + parametros);

Subido

Writing at 0x000070000... (100 %)
 Wrote 654016 bytes (400681 compressed) at 0x00010000 in 6.9 seconds (effective 759.8 kbit/s)...
 Hash of data verified.
 Compressed 3072 bytes to 128...
 Writing at 0x00008000... (100 %)
 Wrote 3072 bytes (128 compressed) at 0x00008000 in 0.0 seconds (effective 2730.7 kbit/s)...
 Hash of data verified.

Leaving...
 Hard resetting via RTS pin...

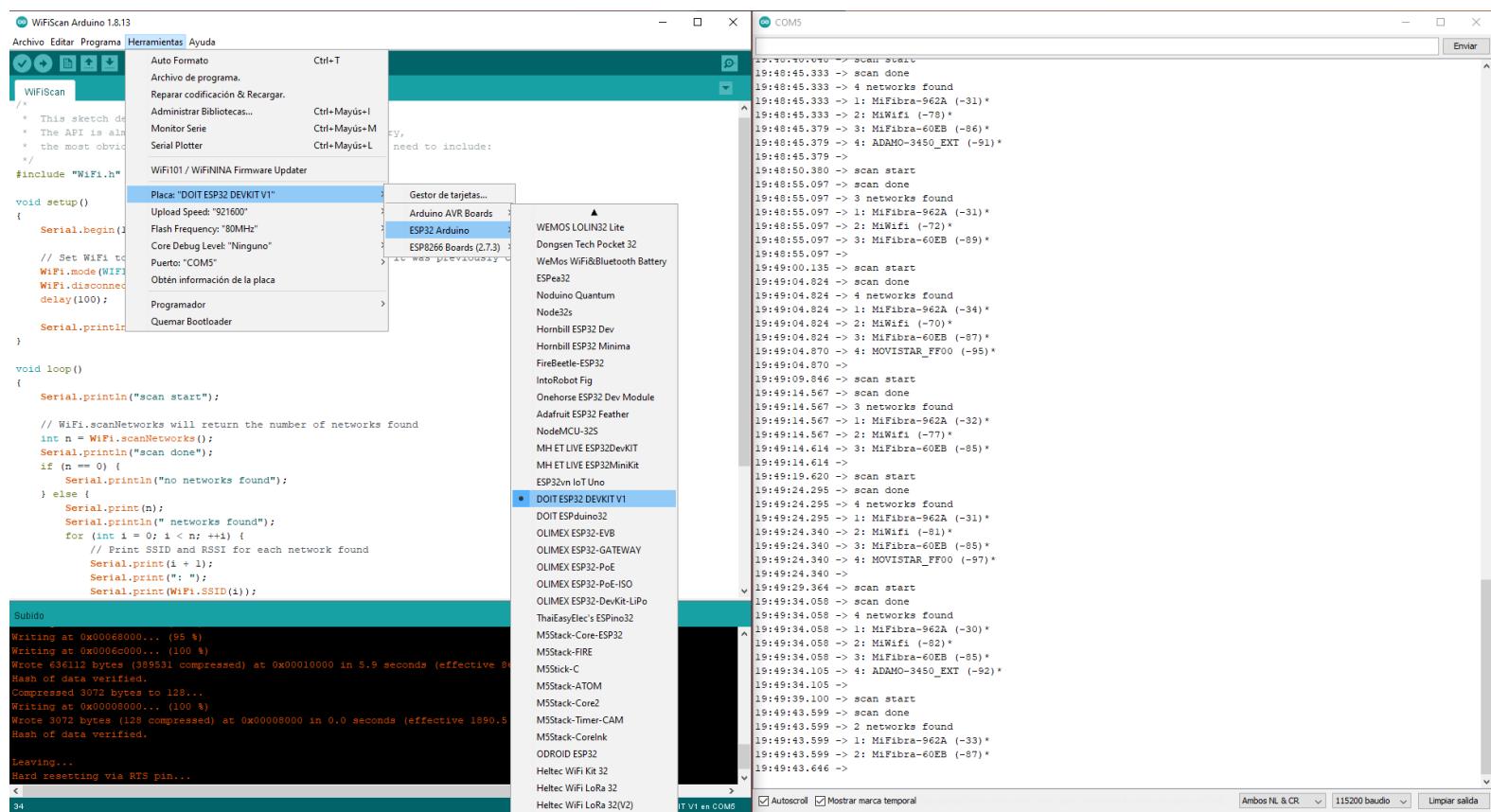
DOIT ESP32 DEVKIT V1.80MHz 921600 None en COM3 58

Resuelto el problema de subida de código

Una vez seleccionados el nombre de placa y el por COM correctos, siga estos pasos:

- Mantén presionado el botón " BOOT " de la placa ESP32.
- Presiona el botón " Cargar " en el IDE de Arduino para cargar el boceto.
- Después de que vea el mensaje " Conectando.... "En su Arduino IDE, suelta el dedo del botón " BOOT ".
- Después de eso, deberías ver el mensaje " Carga finalizada ".

El ESP32 debería tener el nuevo boceto en ejecución. Presiona el botón " ENABLE " para reiniciar el ESP32 y ejecutar el nuevo boceto cargado.



Resuelto el problema de subida de código II.

El problema que nos da es por culpa del cable DIO0: GPIO 2 parece ser. No influye que sea emisor o receptor. Si no le conectas, la conexión se realiza con éxito aparentemente.

The image shows two side-by-side screenshots of the Arduino IDE. Both windows have the title 'loraRecibir' and version 1.8.13.

Left Window (Code):

```
#include <LoRa.h>
#include <LoRa.h>

//***** Modified from the examples of the Arduino LoRa library
// More resources: https://randomnerdtutorials.com
***** /  

#include <SPI.h>
#include <LoRa.h>

#define pins used by the transceiver
#define ss 5
#define rst 14
#define dio0 2

void setup() {
  //Initialize Serial Monitor
  Serial.begin(115200);
  while (!Serial) {
    Serial.println("LoRa Receiver");
  }
  //setup LoRa transceiver module
  LoRa.setPins(ss, rst, dio0);

  //replace the LoRa.begin(--E-) argument
  //433E6 for Asia
  //866E6 for Europe
  //915E6 for North America
  while (!LoRa.begin(866E6)) {
    Serial.println(".");
    delay(500);
  }
  // Change sync word (0xF3) to match the
  // The sync word assures you don't get LoRa messages from other LoRa transceivers
  // ranges from 0-0xFF
  LoRa.setSyncWord(0xF3);
  Serial.println("LoRa Initializing OK!");
}

// The sync word (0xF3) to match the receiver
// The sync word assures you don't get LoRa messages from other LoRa transceivers
// ranges from 0-0xFF
LoRa.setSyncWord(0xF3);
Serial.println("LoRa Initializing OK!");
```

Right Window (Serial Monitor):

```
loraRecibir Arduino 1.8.13
Archivo Editar Programa Herramientas Ayuda
loraRecibir
#include <LoRa.h>
#include <LoRa.h>

//***** Modified from the examples of the Arduino LoRa library
// More resources: https://randomnerdtutorials.com
***** /  

#include <SPI.h>
#include <LoRa.h>

#define pins used by the transceiver
#define ss 5
#define rst 14
#define dio0 2

void setup() {
  initialize Serial Monitor
  Serial.begin(115200);
  while (!Serial) {
    Serial.println("LoRa Receiver");
  }
  //setup LoRa transceiver module
  .setPins(ss, rst, dio0);

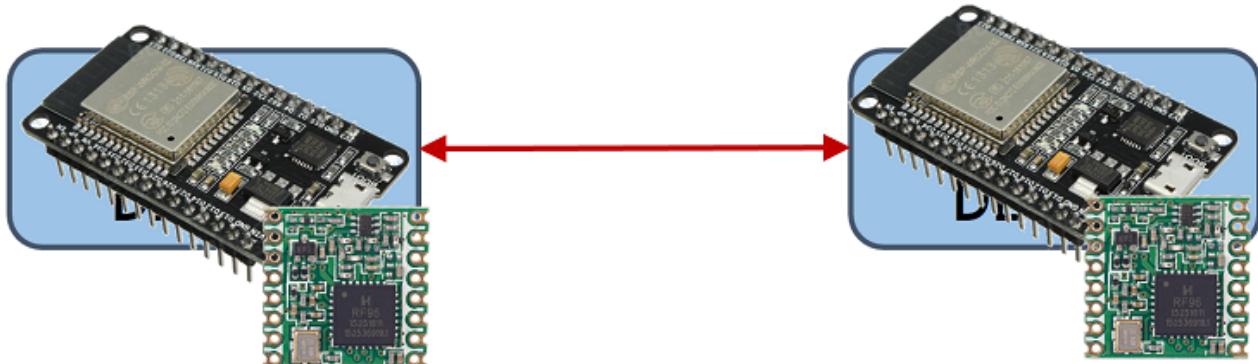
  place the LoRa.begin(--E-) argument with your location's frequency
  E6 for Asia
  E6 for Europe
  E6 for North America
  E (!LoRa.begin(866E6)) {
    Serial.println(".");
    delay(500);
  }

  Change sync word (0xF3) to match the receiver
  // The sync word assures you don't get LoRa messages from other LoRa transceivers
  // ranges from 0-0xFF
  LoRa.setSyncWord(0xF3);
  Serial.println("LoRa Initializing OK!");
```

The serial monitor window shows a series of received packets labeled 'Hello' followed by a timestamp and RSSI value. The first few lines of the log are:

```
21:26:48.010 -> LoRa Initializing OK!
21:26:53.622 -> Received packet 'Hello 2' with RSSI -91
21:27:01.309 -> Received packet 'Hello 0' with RSSI -87
21:27:11.341 -> Received packet 'Hello 1' with RSSI -86
21:27:16.498 -> Received packet 'Hello 2' with RSSI -91
21:27:21.371 -> Received packet 'Hello 3' with RSSI -87
21:27:31.434 -> Received packet 'Hello 3' with RSSI -84
21:27:34.842 -> Received packet 'Hello 4' with RSSI -85
21:27:41.450 -> Received packet 'Hello 4' with RSSI -86
21:27:51.499 -> Received packet 'Hello 5' with RSSI -91
21:28:01.516 -> Received packet 'Hello 6' with RSSI -84
21:28:11.549 -> Received packet 'Hello 7' with RSSI -87
21:28:12.769 -> Received packet 'Hello 8' with RSSI -83
21:28:21.615 -> Received packet 'Hello 9' with RSSI -84
21:28:41.618 -> Received packet 'Hello 10' with RSSI -84
21:28:51.682 -> Received packet 'Hello 11' with RSSI -81
21:29:01.700 -> Received packet 'Hello 12' with RSSI -70
21:29:11.763 -> Received packet 'Hello 13' with RSSI -42
21:29:21.781 -> Received packet 'Hello 14' with RSSI -43
21:30:27.543 -> Received packet 'Hello 15' with RSSI -43
```

Implementación del código Arduino



En este apartado se explica como se ha realizado la conexión entre ambos arduinos con lora y como subimos a una base de datos la información que se transmite entre ambos.

El código que únicamente realiza la conexión Lora, no sube nada a la base de datos, se puede encontrar en “conexionLora.zip”

En primer lugar, hemos creado una estructura de pruebas para trabajar desde casa con un arduino emisor al que hemos conectado un transceptor Lora y otro arduino receptor con su correspondiente transceptor Lora.

El Arduino emisor contiene un código que se encarga de conectarse con el receptor y enviarle un mensaje (en este caso, como es una prueba envía un simple saludo)*. Si recibe un mensaje del receptor lo que hará será mostrarlo en el Monitor Serie.

El Arduino receptor contiene un código que se encarga de conectarse al emisor y a la red Wifi asignada (en este caso, como es una prueba se conecta a la red Wifi de mi casa). Cuando se conecta al emisor y ha recibido el mensaje lo muestra en el Monitor Serie y envía un mensaje al emisor (otro saludo).

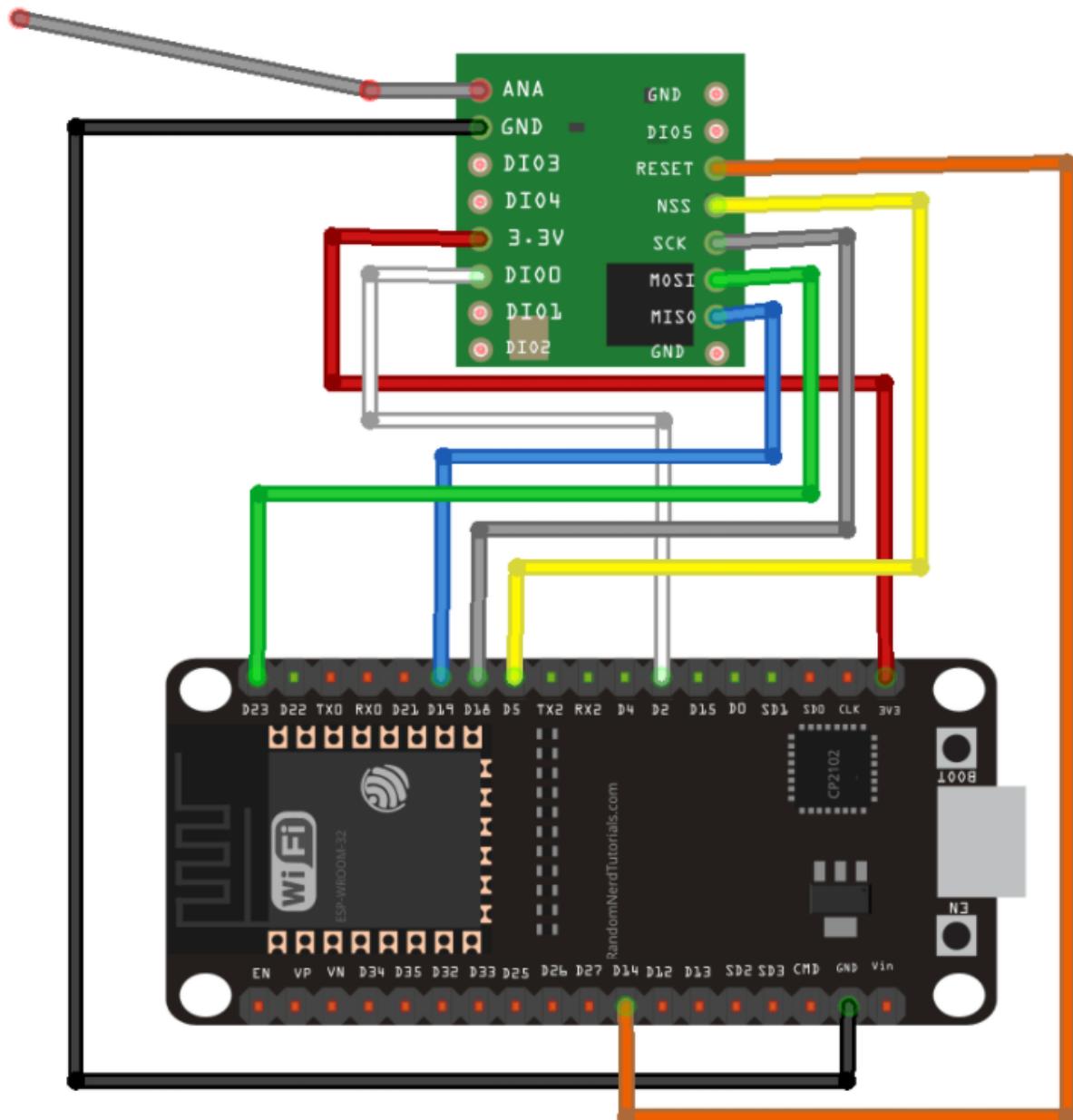
Cuando se ha conectado a la red Wifi y ha recibido el mensaje, hace una conexión con la base de datos usando un archivo .php y guarda en ella los saludos del emisor.

*El mensaje de saludo le hemos llamado [LoRaData](#)

De este modo tenemos un emisor encargado de emitir y recibir datos y un receptor encargado de recibir y enviar datos al emisor y subir los datos recibidos a una base de datos de pruebas.

Resultado de la conexión ESP32 receptor con LoRa

**Ésta configuración de cables es igual que la del emisor.
Está conectado directamente a LoRa.**



Resultado de la conexión ESP32 receptor con LoRa

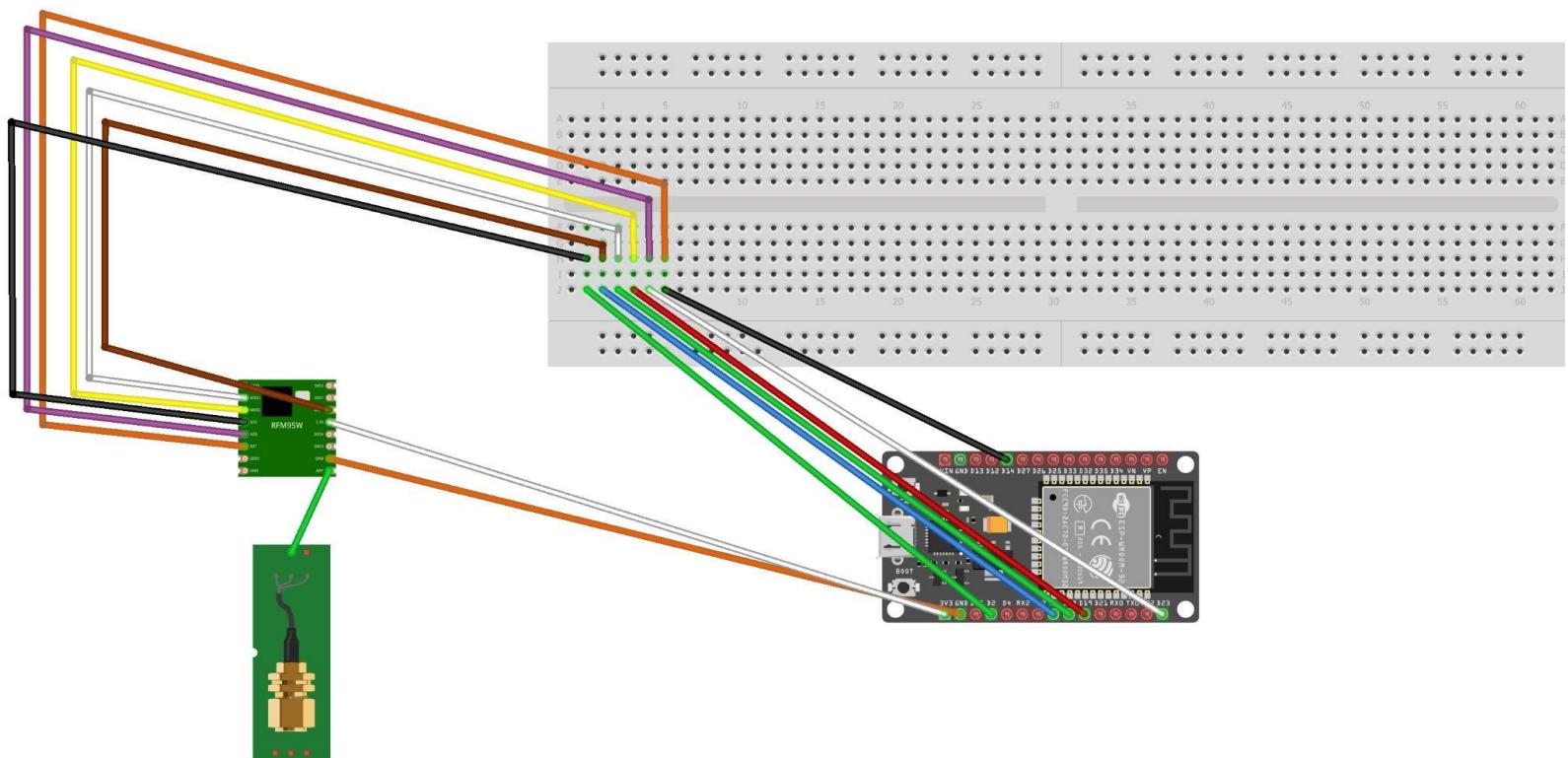
Ésta configuración de cables es igual que la del emisor.

En este caso, LoRa, está conectado a la Protoboard.

Como aclaración, en este caso los cables 3v3 y GND van conectados directamente a LoRa, ya que de esta manera es la única que forma de que nos deje subir código.

El funcionamiento es el mismo, es simplemente otra manera de conectarlo. Haciéndolo de esta manera, podremos conectar diferentes tipos de sensores como veremos a continuación

*El color de los cables es indiferente.



Muestras del código Arduino y PHP y del resultado en la Base de Datos

Código del Emisor

```
#include <SPI.h>
#include <LoRa.h>

//define the pins used by the transceiver module
#define ss 5
#define rst 14
#define dio0 2

int counter = 0;
void setup() {
    //initialize Serial Monitor
    Serial.begin(115200);
    while (!Serial);
    Serial.println("LoRa Sender");

    //setup LoRa transceiver module
    LoRa.setPins(ss, rst, dio0);

    while (!LoRa.begin(866E6)) {
        Serial.println(".");
        delay(500);
    }

    LoRa.setSyncWord(0xF3);
    Serial.println("LoRa Initializing OK!");
}

void loop() {
    Serial.print("Sending packet: ");
    Serial.println(counter);

    //Send LoRa packet to receiver
    LoRa.beginPacket();
    LoRa.print("hello ");
    LoRa.print(counter);
    LoRa.endPacket();
    int packetSize = LoRa.parsePacket();
    if (packetSize) {
        // received a packet
    }
}
```

```
Serial.print("Received packet '") ;

// read packet
while (LoRa.available()) {
    String LoRaData = LoRa.readString();
    Serial.print(LoRaData);
}

// print RSSI of packet
Serial.print("' with RSSI ");
Serial.println(LoRa.packetRssi());
}

counter++;
delay(2000);
}
```

Código del Receptor

```
#include <LoRa.h>
#include <WiFi.h>
#include "WiFi.h"
#include <SPI.h>

//define the pins used by the transceiver module
#define ss 5
#define rst 14
#define dio0 2

const char* wifi_ssid      = "MiWifi";
const char* wifi_password  = "942543459";
const char* host           = "192.168.1.111";
int numero = 0;
int counter = 0;

void setup() {
    conectarWifi(); //Llamamos a la función conectarWifi()
    //initialize Serial Monitor
    Serial.begin(115200);
    while (!Serial);
    Serial.println("LoRa Receiver");

    //setup LoRa transceiver module
    LoRa.setPins(ss, rst, dio0);

    while (!LoRa.begin(866E6)) {
        Serial.println(".");
        delay(500);
    }
    LoRa.setSyncWord(0xF3);
    Serial.println("LoRa Initializing OK!");
}

void loop() {
    // try to parse packet
    int packetSize = LoRa.parsePacket();

    if (packetSize) {
        Serial.print("PacketSize: ");
        Serial.println(packetSize);
```

```

// received a packet
Serial.print("Received packet ''");
String LoRaData = "estonorecibionada";

// read packet
while (LoRa.available()) {
    LoRaData = LoRa.readString();
    Serial.print(LoRaData);
}

// print RSSI of packet
Serial.print(' with RSSI ');
Serial.println(LoRa.packetRssi());
Serial.print("Sending packet: ");
Serial.println(counter);

//Send LoRa packet to receiver
LoRa.beginPacket();
LoRa.print("hello ");
LoRa.print(counter);
LoRa.endPacket();

counter++;
delay(2000);

String parametros = "&LoRaData=" + LoRaData;
enviarPOST(parametros);
}

}

void enviarPOST(String parametros){
String url = "/LaGranjaPruebaPHP/Servidor/prueba_arduino.php";
Serial.print("Conectando a ");
Serial.println("127.0.0.1");

WiFiClient client;
const int httpPort = 8080; //8080
// strhost.toCharArray(host, 49);
if (!client.connect(host, httpPort))
{
    Serial.println("conexión fallida");
    return;
}

```

```
Serial.print("Requesting URL: ");
Serial.print(url);

client.println("POST " + url + " HTTP/1.1");
client.println("Host: 192.168.1.111");      // + host);
client.println("Cache-Control: no-cache");
client.println("Content-Type: application/x-www-form-urlencoded");
client.print("Content-Length: ");
client.println(parametros.length());
client.println();
client.println(parametros);

unsigned long timeout = millis();
while (client.available() == 0){
    if (millis() - timeout > 5000) {
        Serial.println(">>>Client Timeout !");
        client.stop();
        return;
    }
}
while (client.available()) {
    String line = client.readStringUntil('\r');
    Serial.print(line);
}

Serial.println();
Serial.println("Cerrando conexion");
}

//Estas son las funciones que se utilizan
void conectarWifi(){
    Serial.begin(115200);
    Serial.println();
    while (!Serial);
    Serial.println("Prueba de POST contra el servidor desde Arduino");
    Serial.print("Conectando a ");
    Serial.println(wifi_ssid);

    WiFi.begin(wifi_ssid, wifi_password);

    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
```

```

        Serial.println(".");
    }

    Serial.println();
    Serial.println("Conectado al WiFi");
    Serial.println("Direccion IP:  ");
    Serial.println(WiFi.localIP());
}

```

Probando a subir contenido a la Base de Datos

Creamos base de datos

Creamos una base de datos de pruebas a la que subimos datos manualmente. Con ella lo que conseguiremos es probar la conexión LoRa, empezando por mandar simples mensajes y más adelante subir datos de diferentes sensores antes de conectarnos a la API.

The screenshot shows the phpMyAdmin interface with the following details:

- Server:** 127.0.0.1
- Database:** prueba_arduino
- Structure Tab:** Active tab
- Tables:** 1 tabla (prueba_arduino)
- Table Data:**

Filas	Tipo	Cotejamiento	Tamaño	Residuo a depurar
238	InnoDB	utf8mb4_general_ci	16.0 KB	-
238	InnoDB	utf8mb4_general_ci	16.0 KB	0 B
- Operations Tab:** Available
- Privileges Tab:** Available
- Routines Tab:** Available
- Events Tab:** Available
- Left Sidebar:** Shows available databases: Nueva, dwec, dwes, information_schema, mysql, performance_schema, phpmyadmin, pizzería, prueba_arduino, test, tienda.

Código del archivo .php encargado de hacer la conexión y cargar la información a la base de datos de pruebas.

```
<?php

// Es mejor tener los datos de conexión por separado, así es más fácil
cambiarlos

$servidor = "localhost";
$basedatos = "prueba_arduino";
$usuario = "arduinousu";
$password = "arduino123";

try {
    // Pruebo a hacer la conexión antes de nada
    $dwes = new PDO("mysql:host=" . $servidor . ";dbname=" .
$basedatos, $usuario, $password);
    $dwes->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $p) {
    echo "<p>Error al conectar" . $p->getMessage() . "</p>";
    exit();
}

// Este foreach muestra todos los parámetros pasados por POST
echo "----- Se han recibido por LORA las siguientes parejas CLAVE
=> VALOR ";
foreach ($_POST as $clave => $valor) {
    echo ($clave . " => " . $valor . " ----- \n\n");
}

// Cojo unos valores según la fecha y la hora, para que los datos que
introducimos en la BBDD no sean siempre iguales
$LoRaData = "defecto " . Date(" d/m/y - h:m:s");      // Valor por
defecto de LoRaData, por si no viene definido
$numero = Date("hms");                                // Valor para el
número, que es la hora minuto y segundo seguidos

//Primero compruebo si se ha pasado por POST el valor para LoRaData
if (isset($_POST['LoRaData'])) {
    // Si se ha definido lo cojo.
    $LoRaData = $_POST['LoRaData'];
}
echo "Valor final de LoRaData: " . $LoRaData . " \n";

try {
```

```
// Genero la consulta y la lanzo
$consulta = "INSERT INTO tabla_arduino (mensaje, numero) VALUES ('"
. $LoRaData . "','" . $numero . ")";
$dwes->exec($consulta);
} catch (Exception $ex) {
    throw new Exception("No se ha podido recuperar la lista: " + $ex);
} finally {
    // Tanto si funciona como si no elimino la conexión
    $dwes = null;
}

echo "bien :)" ;
```

Capturas de la ejecución del código y del resultado en la Base de Datos

Captura de la ejecución del **código emisor**. Como podemos ver en esta imagen, el emisor envía un paquete al emisor e imprime el que recibe del receptor en el Monitor Serie.

The screenshot shows the Arduino IDE interface with two main windows: the code editor and the Serial Monitor.

Code Editor (loraEmisorReceptor Arduino 1.8.13):

```
loraEmisorReceptor Arduino 1.8.13
Archivo Editar Programa Herramientas Ayuda
loraEmisorReceptor$ //433E6 for Asia
//866E6 for Europe
//915E6 for North America
while (!LoRa.begin(866E6)) {
    Serial.println(".");
    delay(500);
}
// Change sync word (0xF3) to match the receiver
// The sync word assures you don't get LoRa ranges from 0-0xFF
LoRa.setSyncWord(0xF3);
Serial.println("LoRa Initializing OK!");
}

void loop() {
    Serial.print("Sending packet: ");
    Serial.println(counter);

    //Send LoRa packet to receiver
    LoRa.beginPacket();
    LoRa.print("hello ");
    LoRa.print(counter);
    LoRa.endPacket();
    int packetSize = LoRa.parsePacket();
    if (packetSize) {
        // received a packet
        Serial.print("Received packet ''");

        // read packet
        while (LoRa.available()) [REDACTED]
```

Serial Monitor (COM6):

```
21:17:52.454 -> ets Jun 8 2016 00:22:57
21:17:52.454 ->
21:17:52.454 -> rst:0x1 (POWERON_RESET),boot:0x1b (SPI_FAST_FLASH_BOOT)
21:17:52.501 -> configsip: 0, SPIWP:0xee
21:17:52.501 -> clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
21:17:52.501 -> mode:DIO, clock div:1
21:17:52.501 -> load:0x3fff0018,len:4
21:17:52.501 -> load:0x3fff001c,len:1044
21:17:52.501 -> load:0x40078000,len:10124
21:17:52.501 -> load:0x40080400,len:5856
21:17:52.501 -> entry 0x400806a8
21:17:52.595 -> LoRa Sender
21:17:52.641 -> LoRa Initializing OK!
21:17:52.641 -> Sending packet: 0
21:17:54.652 -> Sending packet: 1
21:17:56.712 -> Sending packet: 2
21:17:58.723 -> Sending packet: 3
21:18:00.786 -> Sending packet: 4
21:18:02.926 -> Sending packet: 5
21:18:04.846 -> Sending packet: 6
21:18:04.846 -> Received packet 'hello 6' with RSSI -21
21:18:06.859 -> Sending packet: 7
21:18:08.873 -> Sending packet: 8
21:18:08.922 -> Received packet 'hello 8' with RSSI -20
21:18:10.929 -> Sending packet: 9
21:18:12.947 -> Sending packet: 10
21:18:12.993 -> Received packet 'hello 1' with RSSI -20
21:18:15.006 -> Sending packet: 11
21:18:17.023 -> Sending packet: 12
21:18:17.071 -> Received packet 'hello 1' with RSSI -21
21:18:19.038 -> Sending packet: 13
21:18:21.095 -> Sending packet: 14
21:18:21.142 -> Received packet 'hello 1' with RSSI -20
```

Captura de la ejecución del **código receptor**. Como podemos ver en esta imagen, el receptor recibe el mensaje del emisor y lo muestra en el Monitor Serie. Posteriormente envía un paquete al emisor y nos avisa de que lo ha enviado. Se conecta y nos avisa de que está cargado el mensaje recibido en la base de datos y cierra la conexión cuando ha terminado.

```
COM5
Enviar
21:36:37.659 -> ets Jun 8 2016 00:22:57
21:36:37.659 ->
21:36:37.659 -> rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
21:36:37.659 -> configsip: 0, SPIWP:0xee
21:36:37.659 -> clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
21:36:37.659 -> mode:DIO, clock div:1
21:36:37.659 -> load:0x3fff0018,len:4
21:36:37.659 -> load:0x3fff001c,len:1044
21:36:37.659 -> load:0x40078000,len:10124
21:36:37.706 -> load:0x40080400,len:5856
21:36:37.706 -> entry 0x400806a8
21:36:37.940 ->
21:36:37.940 -> Prueba de POST contra el servidor desde Arduino
21:36:37.940 -> Conectando a MiWifi
21:36:38.550 -> .
21:36:39.064 -> .
21:36:39.580 -> .
21:36:40.044 -> .
21:36:40.559 -> .
21:36:41.074 -> .LoRa Receiver
21:36:41.310 -> LoRa Initializing OK!
21:36:42.295 -> PacketSize: 7
21:36:42.295 -> Received packet 'hello 3' with RSSI -22
21:36:42.295 -> Sending packet: 0
21:36:44.311 -> Conectando a 127.0.0.1
21:36:44.357 -> Requesting URL: /LaGranjaPruebaPHP/Servidor/prueba_arduino.phpHTTP/1.1 200 OK
21:36:44.403 -> Date: Wed, 19 May 2021 19:36:44 GMT
21:36:44.403 -> Server: Apache/2.4.46 (Win64) OpenSSL/1.1.1g PHP/7.4.10
21:36:44.403 -> X-Powered-By: PHP/7.4.10
21:36:44.450 -> Content-Length: 150
21:36:44.450 -> Content-Type: text/html; charset=UTF-8
21:36:45.428 ->
21:36:45.428 -> ----- Se han recibido por LORA las siguientes parejas CLAVE => VALOR LoRaData => hello 3 -----
21:36:45.428 ->
21:36:45.428 -> Valor final de LoRaData: hello 3
21:36:45.428 -> bien :)
21:36:45.428 -> Cerrando conexion
21:36:46.363 -> PacketSize: 7
```

The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** loraEmisorReceptor Arduino 1.8.13
- Menu Bar:** Archivo Editar Programa Herramientas Ayuda
- Toolbar:** Includes icons for Open, Save, Print, and others.
- Code Editor:** The main area contains C++ code for an LoRa transceiver. The code includes headers for SPI and I2C, defines pins for the transceiver module, initializes serial communication, and sets up LoRa parameters. It then enters a loop where it repeatedly sends packets (labeled 5 through 10) and prints the transmission time for each. The code ends with a cleanup section for serial and LoRa.
- Serial Monitor:** A window titled "COM6" is open, showing the transmitted packets and their times. The output is as follows:
 - 20:44:24.024 -> Sending packet: 5
 - 20:44:26.071 -> Sending packet: 6
 - 20:44:28.111 -> Sending packet: 7
 - 20:44:30.119 -> Sending packet: 8
 - 20:44:32.176 -> Sending packet: 9
 - 20:44:34.171 -> Sending packet: 10
- Serial Port Selection:** The port is set to COM6.
- Send Button:** An "Enviar" button is located in the top right corner of the Serial Monitor window.
- Status Bar:** Shows checkboxes for "Autoscroll" and "Mostrar marca temporal", and dropdowns for "Ambos NL & CR", "115200 baudio", and "Limpia salida".
- Page Footer:** 1 - 14 and DOIT ESP32 DEVKIT V1 en CO

The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** loraRecibirEmisorAbueno Arduino 1.8.13
- Menu Bar:** Archivo Editar Programa Herramientas Ayuda
- Toolbar:** Includes icons for Open, Save, Print, and others.
- Code Editor:** Displays C++ code for LoRa communication and file writing. The code includes functions like `enviarPOST(LoRaData)`, `Serial.println("Sending packet: ");`, and `Serial.println(counter);`. It also handles LORA frame reception and file writing to memory.
- Serial Monitor:** Shows the serial port selection as COM3 and the current baud rate as 115200. It displays the received data from the LoRa module, such as "Valor final de LoRaData: hello_6".
- Status Bar:** Shows the message "Writing at 0x00006C000... (96 %)" and "Writing at 0x000070000... (100 %)". It also indicates compressed data transfer rates (760.6 kbit/s) and verified data hashes.
- Bottom Buttons:** Includes checkboxes for "Autoscroll" and "Mostrar marca temporal", and buttons for "Ambos NL & CR", "115200 baudio", and "Limpiar salida".

Captura de la **base de datos**. Esta base de datos es una simple prueba con 2 campos, un campo mensaje, el cual es el mensaje que recibe el receptor y el campo número, el cual contiene la fecha.

phpMyAdmin

Servidor: 127.0.0.1 » Base de datos: prueba_arduino » Tabla: tabla_arduino

Examinar Estructura SQL Buscar Insertar

```
SELECT * FROM `tabla_arduino`
```

<< < 194 > Número de filas: 25 Filtrar filas: Buscar en es

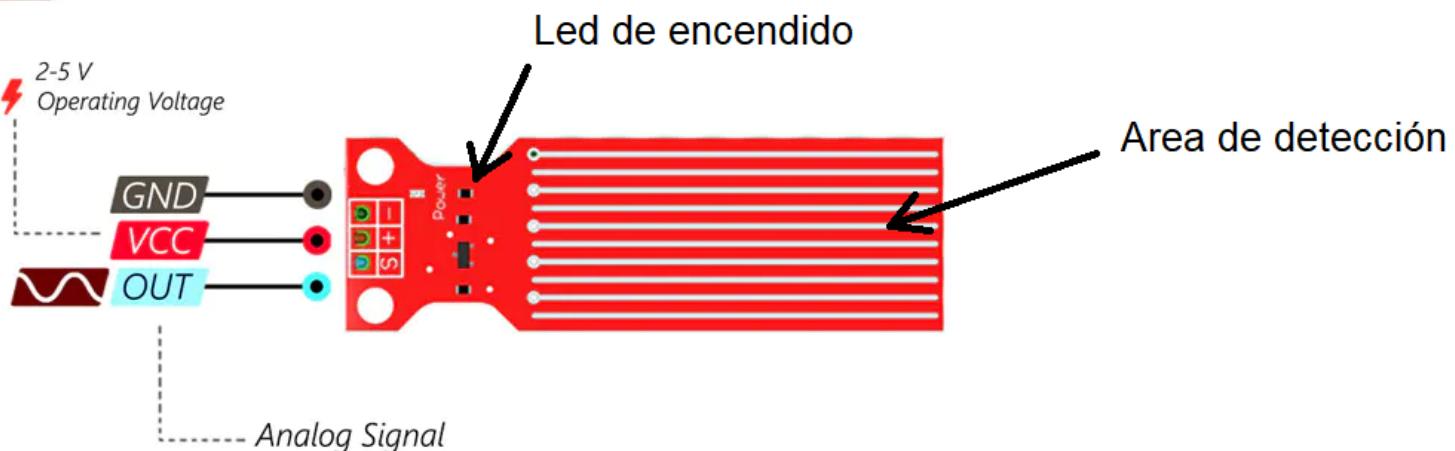
+ Opciones

mensaje	numero
hello 303	90555
hello 305	90559
hello 307	90503
hello 309	90507
hello 311	90511
hello 315	90519
hello 317	90523
hello 319	90527
hello 3	90538
hello 5	90542
hello 7	90546
hello 9	90550
hello 11	90554
hello 13	90558
hello 15	90502
hello 17	90506

Conexión de sensor de Agua

Hemos conectado un sensor de agua para hacer pruebas con el dato que nos devuelva y subirlo a la base de datos. Éste ha sido el procedimiento.

En primer lugar, el sensor consta de lo siguiente:



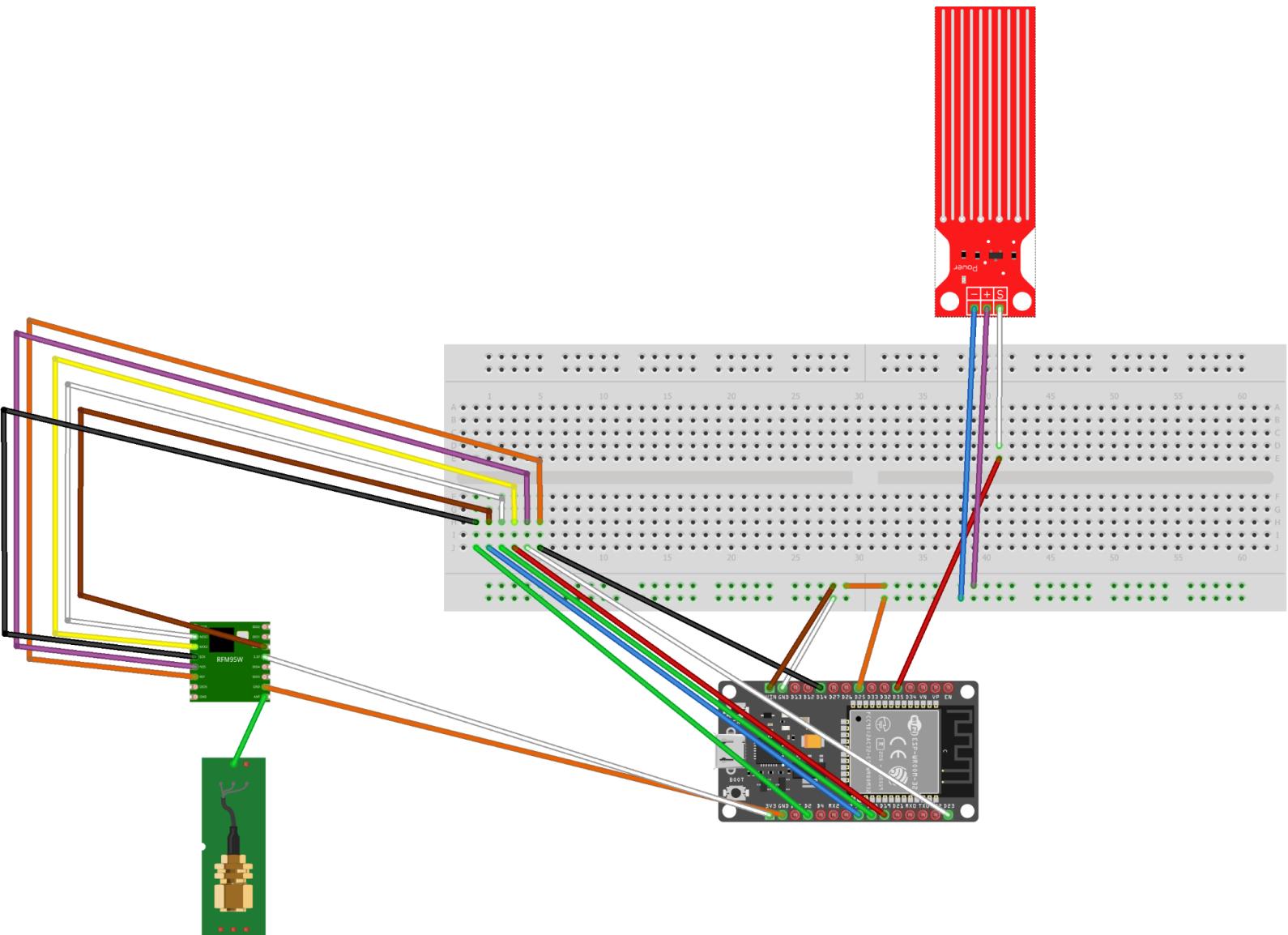
Este sensor de nivel de agua tiene 3 pines. 2 de ellos son para alimentación (+), que se conectan al + 5V, y tierra (-), que se conectan al terminal de tierra del Arduino. El otro pin (S), es el pin de salida analógica.

Conectamos el pin(S) al pin(GPIO 35) que es un pin de entrada analógica de ESP32, después el pin(+) a la entrada de 2-5v de ESP32(VIN), por último conectamos el pin(-) al pin(GND) de ESP32.

Interfaz del sensor de nivel de agua con Arduino

No necesita ninguna biblioteca específica para usar este sensor. Simplemente leemos el valor analógico del pin de salida y calculamos la cantidad de agua echando unas gotas encima de él para comprobar su correcto funcionamiento.

Resultado de la conexión del sensor de agua



En esta captura podemos ver la conexión del ESP32 emisor con el del sensor de agua junto a LoRa

Capturas de la ejecución del código y del resultado en la Base de Datos

El código utilizado se puede encontrar en el zip “29052021.zip”

Código del archivo Receptor encargado ahora también de interpretar los datos del sensor de agua. En primer lugar, leemos el string y almacenamos los datos recibidos, después tenemos la función indexOf que se encarga de localizar los valores. También utilizamos el substring para extraer los caracteres de un punto a otro, primero de la posición 0 a la 1 y después de la 2 al resto de posiciones. Por último guardamos los datos en las variables loraData y waterSensor dentro de la variable parámetros para después hacer el POST con parámetros. Este dispositivo cumple también la función de emisor enviando mensajes.

```
#include <LoRa.h>
#include <WiFi.h>
#include "WiFi.h"

/*****************
 Modified from the examples of the Arduino LoRa library
 More resources: https://randomnerdtutorials.com
*****/

#include <SPI.h>

//define the pins used by the transceiver module
#define ss 5
#define rst 14
#define dio0 2

const char* wifi_ssid = "MiWifi";
const char* wifi_password = "942543459";
unsigned long previousMillis = 0;
unsigned long interval = 30000;
const char* host = "192.168.1.111";
int numero = 0;
int counter = 0;
String loraMessage;
String LoRaData;
String waterSensor;

void setup() {

    conectarWifi(); //Llamamos a la función conectarWifi()
```

```
//initialize Serial Monitor
Serial.begin(115200);
while (!Serial);
Serial.println("LoRa Receiver");

//setup LoRa transceiver module
LoRa.setPins(ss, rst, dio0);

//replace the LoRa.begin(---E-) argument with your location's
frequency
//433E6 for Asia
//866E6 for Europe
//915E6 for North America
while (!LoRa.begin(866E6)) {
    Serial.println(".");
    delay(500);
}
// Change sync word (0xF3) to match the receiver
// The sync word assures you don't get LoRa messages from other LoRa
transceivers
// ranges from 0-0xFF
LoRa.setSyncWord(0xF3);
Serial.println("LoRa Initializing OK!");
}

void loop() {

unsigned long currentMillis = millis();
// if WiFi is down, try reconnecting every CHECK_WIFI_TIME seconds
if ((WiFi.status() != WL_CONNECTED) && (currentMillis -
previousMillis >= interval)) {
    Serial.print(millis());
    Serial.println("Reconnecting to WiFi...");
    WiFi.disconnect();
    // WiFi.reconnect();
    WiFi.begin(wifi_ssid, wifi_password);
    previousMillis = currentMillis;
}

// try to parse packet
int packetSize = LoRa.parsePacket();
```

```

if (packetSize) {
    Serial.print("PacketSize: ");
    Serial.println(packetSize);
    // received a packet
    Serial.print("Received packet ''");

    String datosLoraRecibidos = "estonorecibionada";

    // read packet
    while (LoRa.available()) {
        datosLoraRecibidos = LoRa.readString();
        Serial.print(datosLoraRecibidos);

        // Get readingID, temperature and soil moisture
        int pos1 = datosLoraRecibidos.indexOf('/');
        // int pos2 = datosLoraRecibidos.indexOf('&');
        // int pos3 = datosLoraRecibidos.indexOf('#');

        //     temperature = datosLoraRecibidos.substring(pos1 +1, pos2);
        //     humidity = datosLoraRecibidos.substring(pos2+1, pos3);
        //     pressure = datosLoraRecibidos.substring(pos3+1,
        datosLoraRecibidos.length());

        LoRaData = datosLoraRecibidos.substring(0, pos1);
        waterSensor = datosLoraRecibidos.substring(pos1 + 1,
        datosLoraRecibidos.length());
    }

    // print RSSI of packet
    Serial.print("' with RSSI ");
    Serial.println(LoRa.packetRssi());

    Serial.print("Sending packet: ");
    Serial.println(counter);

    //Send LoRa packet to receiver
    LoRa.beginPacket();
    LoRa.print("hello ");
    LoRa.print(counter);
    LoRa.endPacket();

    counter++;
}

```

```
delay(2000);

String parametros = "&LoRaData=" + LoRaData + "&waterSensor=" +
waterSensor;
enviarPOST(parametros);
}

}

void enviarPOST(String parametros)
{
String url = "/LaGranjaPruebaPHP/Servidor/prueba_arduino.php";
Serial.print("Conectando a ");
Serial.println("127.0.0.1");

WiFiClient client;
const int httpPort = 8080; //8080
// strhost.toCharArray(host, 49);
if (!client.connect(host, httpPort))
{
    Serial.println("conexión fallida");
    return;
}

Serial.print("Requesting URL: ");
Serial.print(url);

// client.print(String("POST ") + url + " HTTP/1.1" + "\r\n" +
"Host: " + host + "\r\n" + "Connection: close\r\n\r\n");
client.println("POST " + url + " HTTP/1.1");
client.println("Host: 192.168.1.111"); // + host);
client.println("Cache-Control: no-cache");
client.println("Content-Type: application/x-www-form-urlencoded");
client.print("Content-Length: ");
client.println(parametros.length());
client.println();
client.println(parametros);
// Ejemplo línea de parámetros:
// LoRaData=123&idDispositivo=23&idMedida=34&nombre=Antonio

unsigned long timeout = millis();
while (client.available() == 0)
{
    if (millis() - timeout > 5000)
```

```
{  
    Serial.println(">>>Client Timeout !");  
    client.stop();  
    return;  
}  
}  
while (client.available())  
{  
    String line = client.readStringUntil('\r');  
    Serial.print(line);  
}  
  
Serial.println();  
Serial.println("Cerrando conexion");  
}  
//Estas son las funciones que se utilizan  
void conectarWifi()  
{  
    Serial.begin(115200);  
    Serial.println();  
    while (!Serial);  
    Serial.println("Prueba de POST contra el servidor desde Arduino");  
    Serial.print("Conectando a ");  
    Serial.println(wifi_ssid);  
  
    WiFi.begin(wifi_ssid, wifi_password);  
  
    while (WiFi.status() != WL_CONNECTED)  
    {  
        delay(500);  
        Serial.println(".");  
    }  
  
    Serial.println();  
    Serial.println("Conectado al WiFi");  
    Serial.println("Direccion IP: ");  
    Serial.println(WiFi.localIP());  
}
```

Captura de la ejecución del **código receptor**. Como podemos ver en esta imagen, imprimimos los datos del sensor de agua en el Monitor Serie y subimos ese datos a la base de datos. También imprimimos el valor de los sensores y el tipo de variable que es para controlar al 100% los resultados

```
13:54:42.662 -> racketsize. 0
13:54:42.662 -> Received packet '2012/0' with RSSI -23
13:54:42.662 -> Sending packet: 1001
13:54:44.104 -> ets Jun  8 2016 00:22:57
13:54:44.104 ->
13:54:44.104 -> rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
13:54:44.104 -> configsip: 0, SPIWP:0xee
13:54:44.104 -> clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
13:54:44.104 -> mode:DIO, clock div:1
13:54:44.104 -> load:0x3fff0018,len:4
13:54:44.104 -> load:0x3fff001c,len:1044
13:54:44.104 -> load:0x40078000,len:10124
13:54:44.151 -> load:0x40080400,len:5856
13:54:44.151 -> entry 0x400806a8
13:54:44.384 ->
13:54:44.384 -> Prueba de POST contra el servidor desde Arduino
13:54:44.384 -> Conectando a MiWifi
13:54:45.125 -> .
13:54:45.589 -> .
13:54:46.102 -> .
13:54:46.613 -> .
13:54:47.081 -> .
13:54:47.590 -> .
13:54:48.097 -> .
13:54:48.610 -> .
13:54:49.123 -> .
13:54:49.585 -> .
13:54:50.095 -> .
13:54:50.604 -> .
13:54:51.119 -> .
13:54:51.583 -> .
13:54:52.092 -> .
13:54:52.603 -> .
13:54:53.115 -> .
13:54:53.629 -> .LoRa Receiver
13:54:53.860 -> LoRa Initializing OK!
13:54:55.808 -> PacketSize: 6
13:54:55.808 -> Received packet '6/1154' with RSSI -24
13:54:55.808 -> Sending packet: 0
13:54:57.804 -> Conectando a 127.0.0.1
13:54:57.850 -> Requesting URL: /LaGranjaPruebaPHP/Servidor/prueba_arduino.phpHTTP/1.1 200 OK
13:54:57.989 -> Date: Sat, 29 May 2021 11:54:57 GMT
13:54:57.989 -> Server: Apache/2.4.46 (Win64) OpenSSL/1.1.1g PHP/7.4.10
13:54:57.989 -> X-Powered-By: PHP/7.4.10
13:54:57.989 -> Content-Length: 225
13:54:57.989 -> Content-Type: text/html; charset=UTF-8
13:54:59.013 ->
13:54:59.013 -> ----- Se han recibido por LORA las siguientes parejas CLAVE => VALOR LoRaData
13:54:59.013 ->
13:54:59.013 -> waterSensor => 1154 -----
13:54:59.013 ->
13:54:59.013 -> Valor final de LoRaData: 6
13:54:59.013 -> Valor final de waterSensor: 1154 y es de tipo string
13:54:59.013 -> bien :)
13:54:59.013 -> Cerrando conexion
<
```

Código emisor. Este código es el encargado de leer y enviar al receptor, tanto el mensaje de saludo como el dato del sensor de agua que le hemos conectado.

En primer lugar declaramos el pin que vamos a usar y después, al ser una entrada analógica, leemos sus valores con la función `analogRead()`, indicando en el paréntesis el número del pin que estamos leyendo. Lo guardamos en la variable `waterSensor`.

Después en el Loop guardamos tanto el saludo como los valores de `waterSensor` en la variable `LoRaMessage`. Separamos las variables con un carácter especial para después en el receptor poder separarlo y extraer los valores.

También cumple la función receptora, recibiendo los mensajes que manda el receptor.

```
#include <SPI.h>
#include <LoRa.h>

//define the pins used by the transceiver module
#define ss 5
#define rst 14
#define dio0 2

int counter = 0;
String LoRaMessage = "";

int waterSensor = 0;      // variable para almacenar valor leido
// set pin numbers
const int buttonPin = 35; // the number of the pushbutton pin

void setup() {
    //initialize Serial Monitor
    Serial.begin(115200);
    while (!Serial);
    Serial.println("LoRa Sender");

    //setup LoRa transceiver module
    LoRa.setPins(ss, rst, dio0);

    //replace the LoRa.begin(---E-) argument with your location's
frequency
    //433E6 for Asia
    //866E6 for Europe
    //915E6 for North America
    while (!LoRa.begin(866E6)) {
        Serial.println(".");
        delay(500);
```

```

        }

        // Change sync word (0xF3) to match the receiver
        // The sync word assures you don't get LoRa messages from other LoRa
transceivers
        // ranges from 0-0xFF
        LoRa.setSyncWord(0xF3);
        Serial.println("LoRa Initializing OK!");
    }

void loop() {
    Serial.print("Sending packet: ");
    Serial.println(counter);

    // read the state of the pushbutton value
    // waterSensor = digitalRead(buttonPin);
    waterSensor = analogRead(buttonPin);
    Serial.print("Sending waterSensor: ");
    Serial.println(waterSensor);

    // LoRaMessage = String(id) + "/" + String(temperatura) + "&" +
String(luz) + "#" + String(humedad);

    LoRaMessage = String(counter) + "/" + String(waterSensor);

    //Send LoRa packet to receiver
    LoRa.beginPacket();

    LoRa.print(LoRaMessage);

    LoRa.endPacket();

    int packetSize = LoRa.parsePacket();
    if (packetSize) {
        // received a packet
        Serial.print("Received packet '");

        // read packet
        while (LoRa.available()) {
            String LoRaData = LoRa.readString();

```

```

        Serial.print(LoRaData);
    }

    // print RSSI of packet
    Serial.print("' with RSSI ");
    Serial.println(LoRa.packetRssi());
}

counter++;
delay(2000);
}

```

Captura del código **emisor**. Como podemos observar, envía de waterSensor.

The screenshot shows a Windows-style terminal window titled "COM6". The window has a header bar with standard window controls (minimize, maximize, close) and a toolbar with a "Enviar" button. The main area displays a log of LoRaWAN transmissions. The log starts with system initialization messages like "rst:0x1 (POWERON_RESET),boot:0x1b (SPI_FAST_FLASH_BOOT)" and "configsip: 0, SPIWP:0xee". It then enters a loop where it repeatedly sends a "waterSensor" value. The transmitted values are: 0, 0, 781, 2, 926, 3, 1136, 4, 1280, 5, 1042, 6, 1154, 7, 1211, and 1123. At the end of the log, it receives a packet with the ID '7/12110' and RSSI -22. The bottom of the window includes standard terminal controls: "Autoscroll" (unchecked), "Mostrar marca temporal" (checked), "Ambos NL & CR" (dropdown), "115200 baudio" (dropdown), and "Limpiar salida" (button).

```

13:54:42.633 -> Sending packet: 2012
13:54:42.633 -> Sending waterSensor: 0
13:54:43.420 -> ets Jun  8 2016 00:22:57
13:54:43.420 ->
13:54:43.420 -> rst:0x1 (POWERON_RESET),boot:0x1b (SPI_FAST_FLASH_BOOT)
13:54:43.420 -> configsip: 0, SPIWP:0xee
13:54:43.420 -> clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
13:54:43.420 -> mode:DIO, clock div:1
13:54:43.420 -> load:0x3fff0018,len:4
13:54:43.420 -> load:0x3fff001c,len:1044
13:54:43.420 -> load:0x40078000,len:10124
13:54:43.420 -> load:0x40080400,len:5856
13:54:43.420 -> entry 0x400806a8
13:54:43.560 -> LoRa Sender
13:54:43.560 -> LoRa Initializing OK!
13:54:43.560 -> Sending packet: 0
13:54:43.560 -> Sending waterSensor: 0
13:54:45.604 -> Sending packet: 1
13:54:45.604 -> Sending waterSensor: 781
13:54:47.652 -> Sending packet: 2
13:54:47.652 -> Sending waterSensor: 926
13:54:49.647 -> Sending packet: 3
13:54:49.647 -> Sending waterSensor: 1136
13:54:51.690 -> Sending packet: 4
13:54:51.690 -> Sending waterSensor: 1280
13:54:53.736 -> Sending packet: 5
13:54:53.736 -> Sending waterSensor: 1042
13:54:55.777 -> Sending packet: 6
13:54:55.777 -> Sending waterSensor: 1154
13:54:57.773 -> Sending packet: 7
13:54:57.819 -> Sending waterSensor: 1211
13:54:57.819 -> Received packet '7/12110' with RSSI -22
13:54:59.814 -> Sending packet: 8
13:54:59.814 -> Sending waterSensor: 1123
13:54:59.814 -> Received packet '7/12110' with RSSI -22

```

Código del archivo .php se encarga de recibir los parámetros del POST (LoraData y waterSensor) y subirlos a la base de datos, además de hacer la conexión y cargar la información a la base de datos de pruebas.

```
<?php

// Es mejor tener los datos de conexión por separado, así es más fácil
cambiarlos

$servidor = "localhost";
$basedatos = "prueba_arduino";
$usuario = "arduinousu";
$password = "arduino123";

try {
    // Pruebo a hacer la conexión antes de nada
    $dwes = new PDO("mysql:host=" . $servidor . ";dbname=" .
$basedatos, $usuario, $password);
    $dwes->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $p) {
    echo "<p>Error al conectar" . $p->getMessage() . "</p>";
    exit();
}

// Este foreach muestra todos los parámetros pasados por POST
echo "----- Se han recibido por LORA las siguientes parejas CLAVE
=> VALOR ";
foreach ($_POST as $clave => $valor) {
    echo ($clave . " => " . $valor . " ----- \n\n");
}

// Cojo unos valores según la fecha y la hora, para que los datos que
introducimos en la BBDD no sean siempre iguales
$LoRaData = "defecto " . Date(" d/m/y - h:m:s");      // Valor por
defecto de LoRaData, por si no viene definido
$numero = Date("hms");                                // Valor para el
número, que es la hora minuto y segundo seguidos

//Primero compruebo si se ha pasado por POST el valor para LoRaData
if (isset($_POST['LoRaData'])) {
    // Si se ha definido lo cojo.
    $LoRaData = $_POST['LoRaData'];
}
echo "Valor final de LoRaData: " . $LoRaData . " \n";
```

```
// //Primero compruebo si se ha pasado por POST el valor para
waterSensor
if (isset($_POST['waterSensor'])) {
    // Si se ha definido lo cojo.
    $waterSensor = $_POST['waterSensor'];
}

echo "Valor final de waterSensor: " . $waterSensor." y es de tipo ";
echo gettype($waterSensor), "\n";

try {
    // Genero la consulta y la lanzo
    $consulta = "INSERT INTO tabla_arduino (mensaje, numero,
waterSensor) VALUES ('" . $LoRaData . "',$numero,'" . $waterSensor .
"')";
    $dwes->exec($consulta);
} catch (Exception $ex) {
    throw new Exception("No se ha podido recuperar la lista: " + $ex);
} finally {
    // Tanto si funciona como si no elimino la conexión
    $dwes = null;
}

echo "bien :);
```

Captura de la **base de datos**. Como podemos observar, aparece el loraData y waterSensor cargados correctamente en la base de datos.

phpMyAdmin

Servidor: 127.0.0.1 » Base de datos: prueba_arduino » Tabla: tabla_arduino

Examinar Estructura SQL Buscar Insertar E

Reciente Favoritas

Nueva dwec dwes information_schema mysql performance_schema phpmyadmin pizzaeria prueba_arduino Nueva tabla_arduino test tienda

250 < > >> Número de filas: 25 Filtrar filas: Bu:

+ Opciones

mensaje	numero	waterSensor
hello 1614	40547	0
hello 1619	40557	0
hello 1623	40505	0
hello 1627	40513	0
hello 1633	40525	0
hello 1636	40531	0
hello 1640	40540	0
hello 1644	40548	0
hello 1649	40558	0
hello 1653	40506	0
hello 1658	40518	0
hello 6	20506	0
hello 13	20520	2441
hello 17	20528	1952
hello 20	20534	1936
hello 25	20544	1857
hello 32	20559	2097
hello 38	20511	0
hello 42	20519	0
hello 45	20525	0
hello 50	20535	0
hello 54	20543	0
hello 58	20551	0
hello 61	20558	0
hello 69	20514	0

Subir datos a la API

A continuación subiremos a la API los datos que recibe el receptor.

Capturas de la ejecución del código y del resultado en la API

El código utilizado se puede encontrar en el zip “15062021.zip”

Código del archivo Emisor. En primer lugar, declaramos todos los posibles tipos de medidas y todos los sensores, tanto de agua, como temperatura, como luz aunque en este caso solo vamos a usar el de Agua.

Especificamos el pin que va a usar el sensor para después leer los valores.

Una vez declarado almacenaremos dentro de la variable LoRaMessage el idEmisor, datoSensor3 y tipoMedidaSensor3. Separamos los valores con caracteres especiales para después poder separarlos y reconocer qué significa cada valor.

Después enviamos LoRaMessage por Loray terminamos recibiendo el mensaje del receptor.

- **idEmisor.** Es el identificador del ESP32 emisor. Como en el proyecto final, se supone que habrá varios emisores, gracias a idEmisor podremos identificar cual es cual.
- **datoSensor3.** Es el dato que recoge el Sensor de Agua.
- **tipoMedidaSensor3.** Es el número que identifica el tipo de medida que es. El valor del Agua es el 3 en la base de datos, entonces el tipo de medida será el 3. El 1 es temperatura y el 2 equivale a la luz.

```
#include <SPI.h>
#include <LoRa.h>

//define the pins used by the transceiver module
#define ss 5
#define rst 14
#define dio0 2

int counter = 0;
String LoRaMessage = "";
int idEmisor = 10;

int tipoMedidaSensor4 = 0;
int tipoMedidaSensor3 = 3;
```

```
int tipoMedidaSensor2 = 2;
int tipoMedidaSensor1 = 1;

int datoSensor4 = 0;      // variable para almacenar valor leido de
int datoSensor3 = 0;      // variable para almacenar valor leido de Agua
int datoSensor2 = 0;      // variable para almacenar valor leido de Luz
int datoSensor1 = 0;      // variable para almacenar valor leido de
Temperatura

const int LED = 12;    // Declaro Pin 12 de ESP32 donde está conectado el
LED
// set pin numbers
const int buttonPin = 35; // the number of the pushbutton pin

void setup() {
    //initialize Serial Monitor
    Serial.begin(115200);
    while (!Serial);

    //setup LoRa transceiver module
    LoRa.setPins(ss, rst, dio0);
    pinMode(LED, OUTPUT);

    //replace the LoRa.begin(---E--) argument with your location's
frequency
    //433E6 for Asia
    //866E6 for Europe
    //915E6 for North America
    while (!LoRa.begin(866E6)) {
        Serial.println(".");
        delay(500);
    }
    // Change sync word (0xF3) to match the receiver
    // The sync word assures you don't get LoRa messages from other LoRa
transceivers
    // ranges from 0-0xFF
    LoRa.setSyncWord(0xF3);
    Serial.println("LoRa Initializing OK!");
}

void loop() {

    Serial.print("Sending packet: ");

```

```
Serial.println(counter);

// read the state of the pushbutton value
// datoSensor3 = digitalRead(buttonPin);
datoSensor3 = analogRead(buttonPin);
Serial.print("idEmisor: ");
Serial.println(idEmisor);

Serial.print("Sending datoSensor3: ");
Serial.println(datoSensor3);

LoRaMessage = String(idEmisor) + "/" + String(datoSensor3) + "&" +
String(tipoMedidaSensor3);

//Send LoRa packet to receiver
LoRa.beginPacket();

LoRa.print(LoRaMessage);

LoRa.endPacket();

int packetSize = LoRa.parsePacket();
if (packetSize) {
    // received a packet
    Serial.print("Received packet '");

    // read packet
    while (LoRa.available()) {
        String LoRaData = LoRa.readString();
        Serial.print(LoRaData);
    }

    // print RSSI of packet
    Serial.print("' with RSSI ");
    Serial.println(LoRa.packetRssi());
}

counter++;
delay(10000);
}
```

Captura de la ejecución del **código emisor**. Como podemos ver en esta imagen, imprimimos los datos del sensor de agua en el Monitor Serie.

```
11:18:32.429 -> ets Jun  8 2016 00:22:57
11:18:32.429 ->
11:18:32.429 -> rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
11:18:32.429 -> configsip: 0, SPIWP:0xee
11:18:32.475 -> clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
11:18:32.475 -> mode:DIO, clock div:1
11:18:32.475 -> load:0x3fff0018,len:4
11:18:32.475 -> load:0x3fff001c,len:1044
11:18:32.475 -> load:0x40078000,len:10124
11:18:32.475 -> load:0x40080400,len:5856
11:18:32.475 -> entry 0x400806a8
11:18:32.613 -> LoRa Initializing OK!
11:18:32.613 -> Sending packet: 0
11:18:32.613 -> idEmisor: 10
11:18:32.613 -> Sending datoSensor3: 137
11:18:42.662 -> Sending packet: 1
11:18:42.662 -> idEmisor: 10
11:18:42.662 -> Sending datoSensor3: 160
11:18:42.662 -> Received packet '10/160s' with RSSI -56
11:18:52.697 -> Sending packet: 2
11:18:52.697 -> idEmisor: 10
11:18:52.697 -> Sending datoSensor3: 146
11:19:02.733 -> Sending packet: 3
11:19:02.733 -> idEmisor: 10
11:19:02.733 -> Sending datoSensor3: 167
11:19:02.733 -> Received packet '10/167s' with RSSI -32
11:19:12.761 -> Sending packet: 4
11:19:12.761 -> idEmisor: 10
11:19:12.761 -> Sending datoSensor3: 154
11:19:22.801 -> Sending packet: 5
11:19:22.801 -> idEmisor: 10
11:19:22.801 -> Sending datoSensor3: 168
11:19:22.848 -> Received packet '10/168s' with RSSI -32
11:19:32.844 -> Sending packet: 6
11:19:32.844 -> idEmisor: 10
```

Autoscroll Mostrar marca temporal Ambos NL & CR 115200 baudio Limpiar salida

Código del archivo Receptor encargado ahora también de interpretar los datos del sensor de agua.

En primer lugar, leemos el string datosLoraRecibidos y almacenamos los datos recibidos, después tenemos la función indexOf que se encarga de localizar los valores. También utilizamos el substring para extraer los caracteres de un punto a otro, primero de la posición 0 a la 1 y después de la 2 al resto de posiciones.

Por último guardamos los datos en las variables idEmisor, datoSensor3 y tipoMedidaSensor3 dentro de la variable parámetros para después hacer el POST con parámetros.

Este dispositivo cumple también la función de emisor enviando mensajes.

```
#include <LoRa.h>
#include <WiFi.h>
#include "WiFi.h"

*****
Modified from the examples of the Arduino LoRa library
More resources: https://randomnerdtutorials.com
*****/

#include <SPI.h>

//define the pins used by the transceiver module
#define ss 5
#define rst 14
#define dio0 2

const char* wifi_ssid = "MiWifi";
const char* wifi_password = "942543459";
unsigned long previousMillis = 0;
unsigned long interval = 30000;
const char* host = "192.168.1.111";
int numero = 0;
int counter = 0;
String loraMessage;
String LoRaData;

String datoSensor3;
String datoSensor2;
String datoSensor1;
String datoSensor4;
```

```
String idEmisor;

String tipoMedidaSensor3;
String tipoMedidaSensor1;
String tipoMedidaSensor2;
String tipoMedidaSensor4;

void setup() {

    conectarWifi(); //Llamamos a la función conectarWifi()

    //initialize Serial Monitor
    Serial.begin(115200);
    while (!Serial);
    Serial.println("LoRa Receiver");

    //setup LoRa transceiver module
    LoRa.setPins(ss, rst, dio0);

    //replace the LoRa.begin(---E--) argument with your location's
frequency
    //433E6 for Asia
    //866E6 for Europe
    //915E6 for North America
    while (!LoRa.begin(866E6)) {
        Serial.println(".");
        delay(500);
    }
    // Change sync word (0xF3) to match the receiver
    // The sync word assures you don't get LoRa messages from other LoRa
transceivers
    // ranges from 0-0xFF
    LoRa.setSyncWord(0xF3);
    Serial.println("LoRa Initializing OK!");
}

void loop() {

    unsigned long currentMillis = millis();
    // if WiFi is down, try reconnecting every CHECK_WIFI_TIME seconds
    if ((WiFi.status() != WL_CONNECTED) && (currentMillis -
previousMillis >= interval)) {
```

```

Serial.print(millis());
Serial.println("Reconnecting to WiFi... ");
WiFi.disconnect();
// WiFi.reconnect();
WiFi.begin(wifi_ssid, wifi_password);
previousMillis = currentMillis;
}

// try to parse packet
int packetSize = LoRa.parsePacket();

if (packetSize) {
    Serial.print("PacketSize: ");
    Serial.println(packetSize);
    // received a packet
    Serial.print("Received packet '");

String datosLoraRecibidos = "estonorecibionada";

// read packet
while (LoRa.available()) {
    datosLoraRecibidos = LoRa.readString();
    Serial.print(datosLoraRecibidos);

    // Get readingID, temperature and soil moisture
    int pos1 = datosLoraRecibidos.indexOf('/');
    int pos2 = datosLoraRecibidos.indexOf('&');
    // int pos3 = datosLoraRecibidos.indexOf('#');

    //     temperature = datosLoraRecibidos.substring(pos1 +1, pos2);
    //     humidity = datosLoraRecibidos.substring(pos2+1, pos3);
    //     pressure = datosLoraRecibidos.substring(pos3+1,
datosLoraRecibidos.length());

    // LoRaData = datosLoraRecibidos.substring(0, pos1);
    idEmisor = datosLoraRecibidos.substring(0, pos1);
    datoSensor3 = datosLoraRecibidos.substring(pos1 + 1, pos2);
    // datoSensor3 = datosLoraRecibidos.substring(pos1 + 1,
datosLoraRecibidos.length());
    tipoMedidaSensor3 = datosLoraRecibidos.substring(pos2 + 1,
datosLoraRecibidos.length());
}

```

```

}

// print RSSI of packet
Serial.print(" with RSSI ");
Serial.println(LoRa.packetRssi());

Serial.print("Sending packet: ");
Serial.println(counter);

// Preparo el actuador

//Send LoRa packet to receiver
LoRa.beginPacket();
LoRa.print("hello ");
LoRa.print(counter);
LoRa.endPacket();

counter++;

delay(2000);

// String parametros = "&LoRaData=" + LoRaData + "&datoSensor3=" +
datoSensor3;

String parametros = "&idEmisor=" + idEmisor + "&datoSensor1=" +
datoSensor1 + "&tipoMedidaSensor1=" + tipoMedidaSensor1+ +
"&datoSensor2=" + datoSensor2 + "&tipoMedidaSensor2=" +
tipoMedidaSensor2 + "&datoSensor3=" + datoSensor3 +
"&tipoMedidaSensor3=" + tipoMedidaSensor3+ "&datoSensor4=" +
datoSensor4 + "&tipoMedidaSensor4=" + tipoMedidaSensor4;

enviarPOST(parametros);

}

}

void enviarPOST(String parametros)
{
    String url = "/LaGranjaPruebaPHP/Servidor/prueba_arduino.php";
    Serial.print("Conectando a ");
    Serial.println("127.0.0.1");

    WiFiClient client;
    const int httpPort = 8080; //8080
    // strhost.toCharArray(host, 49);
}

```

```
if (!client.connect(host, httpPort))
{
    Serial.println("conexión fallida");
    return;
}

Serial.print("Requesting URL: ");
Serial.print(url);

// client.print(String("POST ") + url + " HTTP/1.1" + "\r\n" +
"Host: " + host + "\r\n" + "Connection: close\r\n\r\n");
client.println("POST " + url + " HTTP/1.1");
client.println("Host: 192.168.1.111");      // + host);
client.println("Cache-Control: no-cache");
client.println("Content-Type: application/x-www-form-urlencoded");
client.print("Content-Length: ");
client.println(parametros.length());
client.println();
client.println(parametros);
// Ejemplo linea de parámetros:
// LoRaData=123&idDispositivo=23&idMedida=34&nomnbre=Antonio

unsigned long timeout = millis();
while (client.available() == 0)
{
    if (millis() - timeout > 5000)
    {
        Serial.println(">>>Client Timeout !");
        client.stop();
        return;
    }
}
while (client.available())
{
    String line = client.readStringUntil('\r');
    Serial.print(line);
}

Serial.println();
Serial.println("Cerrando conexión");
}

//Estas son las funciones que se utilizan
void conectarWifi()
```

```
{  
    Serial.begin(115200);  
    Serial.println();  
    while (!Serial);  
    Serial.println("Prueba de POST contra el servidor desde Arduino");  
    Serial.print("Conectando a ");  
    Serial.println(wifi_ssid);  
  
    WiFi.begin(wifi_ssid, wifi_password);  
  
    while (WiFi.status() != WL_CONNECTED)  
    {  
        delay(500);  
        Serial.println(".");  
    }  
  
    Serial.println();  
    Serial.println("Conectado al WiFi");  
    Serial.println("Direccion IP: ");  
    Serial.println(WiFi.localIP());  
}
```

Captura de la ejecución del **código receptor**. Recibimos únicamente el dato del sensor de agua y lo subimos a la API.

COM1 Envir

```
13:51:52.200 -> 1st.0x1 (POWERON_RESET),0x00.0x10 (JTAG_FAST_BOOT)
13:51:52.200 -> configSip: 0, SPIWP:0xee
13:51:52.200 -> clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
13:51:52.200 -> mode:DIO, clock div:1
13:51:52.200 -> load:0x3fff0018,len:4
13:51:52.200 -> load:0x3fff001c,len:1044
13:51:52.200 -> load:0x40078000,len:10124
13:51:52.200 -> load:0x40080400,len:5856
13:51:52.200 -> entry 0x400806a8
13:51:52.480 ->
13:51:52.480 -> Prueba de POST contra el servidor desde Arduino
13:51:52.480 -> Conectando a MiWifi
13:51:53.088 -> .
13:51:53.602 -> .
13:51:54.118 -> .
13:51:54.586 -> .
13:51:55.102 -> .
13:51:55.615 -> .
13:51:56.084 -> .
13:51:56.596 -> .LoRa Receiver
13:51:56.876 -> LoRa Initializing OK!
13:52:01.605 -> PacketSize: 8
13:52:01.605 -> Received packet '10/206&3' with RSSI -37
13:52:01.605 -> Sending packet: 0
13:52:03.615 -> Conectando a 127.0.0.1
13:52:03.661 -> Requesting URL: /LaGranjaPruebaPHP/Servidor/prueba_arduino.phpHTTP/1.1 200 OK
13:52:03.846 -> Date: Tue, 15 Jun 2021 11:52:03 GMT
13:52:03.846 -> Server: Apache/2.4.46 (Win64) OpenSSL/1.1.1g PHP/7.4.10
13:52:03.846 -> X-Powered-By: PHP/7.4.10
13:52:03.846 -> Content-Length: 495
13:52:03.846 -> Content-Type: text/html; charset=UTF-8
13:52:04.827 ->
13:52:04.827 -> ----- Se han recibido por LORA las siguientes parejas CLAVE => VALOR idEmisor
13:52:04.875 ->
13:52:04.875 -> datoSensor1 => -----
13:52:04.875 ->
13:52:04.875 -> tipoMedidaSensor1 => -----
13:52:04.875 ->
13:52:04.875 -> datoSensor2 => -----
13:52:04.875 ->
13:52:04.875 -> tipoMedidaSensor2 => -----
13:52:04.875 ->
13:52:04.875 -> datoSensor3 => -----
13:52:04.875 ->
13:52:04.875 -> tipoMedidaSensor3 => -----
13:52:04.875 ->
13:52:04.875 -> datoSensor4 => -----
13:52:04.875 ->
13:52:04.875 -> tipoMedidaSensor4 => -----
13:52:04.875 ->
13:52:04.875 -> Valor final de idEmisor:
13:52:04.875 -> Valor final de waterSensor: y es de tipo string
13:52:04.875 -> Valor final de tipoMedidaWater: y es de tipo string
13:52:04.875 ->
13:52:04.875 -> Cerrando conexion
```

Código del archivo .php. En primer lugar realizamos la conexión con la API y después empezamos a indentificar y reconocer los valores recibidos, tanto los datos de los sensores como idEmisor como el tipo de medida que es. Finalmente sustituimos los datos por los datos de \$POST usando foreach. También imprimimos el valor de los sensores y el tipo de variable que es para controlar al 100% los resultados

```
<?php

$arrayDatos = array();
// Es mejor tener los datos de conexión por separado, así es más fácil
cambiarlos
echo "----- Se han recibido por LORA las siguientes parejas CLAVE
=> VALOR ";
foreach ($_POST as $clave => $valor) {
    echo ($clave . " => " . $valor . " ----- \n\n");
}

// // Cojo unos valores según la fecha y la hora, para que los datos
que introducimos en la BBDD no sean siempre iguales
// $LoRaData = "defecto " . Date(" d/m/y - h:m:s"); // Valor por
defecto de LoRaData, por si no viene definido
// $numero = Date("hms"); // Valor para el
número, que es la hora minuto y segundo seguidos

//Primero compruebo si se ha pasado por POST el valor para LoRaData
if (isset($_POST['idEmisor'])) {
    // Si se ha definido lo cojo.
    $idEmisor = $_POST['idEmisor'];
}
echo "Valor final de idEmisor: " . $idEmisor . " \n";

// //Primero compruebo si se ha pasado por POST el valor para
waterSensor

if (isset($_POST['datoSensor1'])) {
    // Si se ha definido lo cojo.
    $datoSensor1 = $_POST['datoSensor1'];
}
```

```
if (isset($_POST['tipoMedidaSensor1'])) {  
    // Si se ha definido lo cojo.  
    $tipoMedidaSensor1 = $_POST['tipoMedidaSensor1'];  
    $arrayDatos += [$datoSensor1 => $tipoMedidaSensor1];  
}  
  
if (isset($_POST['datoSensor2'])) {  
    // Si se ha definido lo cojo.  
    $datoSensor2 = $_POST['datoSensor2'];  
}  
  
if (isset($_POST['tipoMedidaSensor2'])) {  
    // Si se ha definido lo cojo.  
    $tipoMedidaSensor2 = $_POST['tipoMedidaSensor2'];  
    $arrayDatos += [$datoSensor2 => $tipoMedidaSensor2];  
}  
  
if (isset($_POST['datoSensor3'])) {  
    // Si se ha definido lo cojo.  
    $datoSensor3 = $_POST['datoSensor3'];  
}  
  
if (isset($_POST['tipoMedidaSensor3'])) {  
    // Si se ha definido lo cojo.  
    $tipoMedidaSensor3 = $_POST['tipoMedidaSensor3'];  
    $arrayDatos += [$datoSensor3 => $tipoMedidaSensor3];  
}  
  
if (isset($_POST['datoSensor4'])) {  
    // Si se ha definido lo cojo.  
    $datoSensor4 = $_POST['datoSensor4'];  
}  
  
if (isset($_POST['tipoMedidaSensor4'])) {  
    // Si se ha definido lo cojo.  
    $tipoMedidaSensor4 = $_POST['tipoMedidaSensor4'];  
    $arrayDatos += [$datoSensor4 => $tipoMedidaSensor4];  
}  
echo "Valor final de waterSensor: " . $datoSensor3." y es de tipo ";  
echo gettype($datoSensor3), "\n";  
  
echo "Valor final de tipoMedidaWater: " . $tipoMedidaSensor3." y es de tipo ";  
echo gettype($tipoMedidaSensor3), "\n";
```

```
try{
    $url = 'http://lagranja40.agllabs.es/api/dispositivos/setMedida';

    $ch = curl_init($url);

    foreach ($arrayDatos as $key => $value) {
        $data = array(
            //Sustituir los datos por los datos de $POST
            'id' => ".$idEmisor.",
            'medida' => ".$key.",
            'tipoMedida' => ".$value."
        );
        $payload = json_encode($data);

        curl_setopt($ch, CURLOPT_POSTFIELDS, $payload);

        curl_setopt($ch, CURLOPT_HTTPHEADER,
array('Content-Type:application/json'));

        curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);

        $result = curl_exec($ch);
    }
    curl_close($ch);

}catch(Exception $e){
    echo $e;
}
```

Captura de la API. Explicación de la captura de la API.

Como podemos ver, en el campo “valor” aparecen 0 y diferentes valores que indican la cantidad de agua medida por el sensor.

✓ Mostrando filas 0 - 24 (total de 998, La consulta tardó 0.0002 segundos.)

```
SELECT * FROM `registro` WHERE `e_dispositivo` = 10
```

1 > >> | Número de filas: 25 | Filtrar filas: Buscar en esta tabla | Ordenar según

← Opciones →

		id_registro	e_dispositivo	fh	valor	e_tipoMedida
<input type="checkbox"/>	Editar Copiar Borrar	601	10	2021-05-31 21:03:02	0	3
<input type="checkbox"/>	Editar Copiar Borrar	602	10	2021-05-31 21:03:12	0	3
<input type="checkbox"/>	Editar Copiar Borrar	603	10	2021-05-31 21:03:22	0	3
<input type="checkbox"/>	Editar Copiar Borrar	604	10	2021-05-31 21:03:32	0	3
<input type="checkbox"/>	Editar Copiar Borrar	605	10	2021-05-31 21:03:42	0	3
<input type="checkbox"/>	Editar Copiar Borrar	606	10	2021-05-31 21:03:52	0	3
<input type="checkbox"/>	Editar Copiar Borrar	607	10	2021-05-31 21:04:02	0	3
<input type="checkbox"/>	Editar Copiar Borrar	608	10	2021-05-31 21:04:24	0	3
<input type="checkbox"/>	Editar Copiar Borrar	609	10	2021-05-31 21:04:34	0	3
<input type="checkbox"/>	Editar Copiar Borrar	610	10	2021-05-31 21:04:44	0	3
<input type="checkbox"/>	Editar Copiar Borrar	611	10	2021-05-31 21:04:55	16	3
<input type="checkbox"/>	Editar Copiar Borrar	612	10	2021-05-31 21:05:05	0	3
<input type="checkbox"/>	Editar Copiar Borrar	613	10	2021-05-31 21:05:15	22	3
<input type="checkbox"/>	Editar Copiar Borrar	614	10	2021-05-31 21:05:25	0	3
<input type="checkbox"/>	Editar Copiar Borrar	615	10	2021-05-31 21:05:35	16	3
<input type="checkbox"/>	Editar Copiar Borrar	616	10	2021-05-31 21:05:45	0	3
<input type="checkbox"/>	Editar Copiar Borrar	617	10	2021-05-31 21:05:55	0	3
<input type="checkbox"/>	Editar Copiar Borrar	618	10	2021-05-31 21:06:05	27	3
<input type="checkbox"/>	Editar Copiar Borrar	619	10	2021-05-31 21:06:15	0	3
<input type="checkbox"/>	Editar Copiar Borrar	620	10	2021-05-31 21:06:25	0	3
<input type="checkbox"/>	Editar Copiar Borrar	621	10	2021-05-31 21:06:35	0	3
<input type="checkbox"/>	Editar Copiar Borrar	622	10	2021-05-31 21:06:45	6	3
<input type="checkbox"/>	Editar Copiar Borrar	623	10	2021-05-31 21:06:55	0	3
<input type="checkbox"/>	Editar Copiar Borrar	624	10	2021-05-31 21:07:05	0	3
<input type="checkbox"/>	Editar Copiar Borrar	625	10	2021-05-31 21:07:15	28	3

Conexión de sensor de Temperatura y Humedad DHT11

Hemos conectado un sensor de temperatura DHT11 para hacer pruebas con el dato que nos devuelva y subirlo a la base de datos. Estas son las características del sensor:

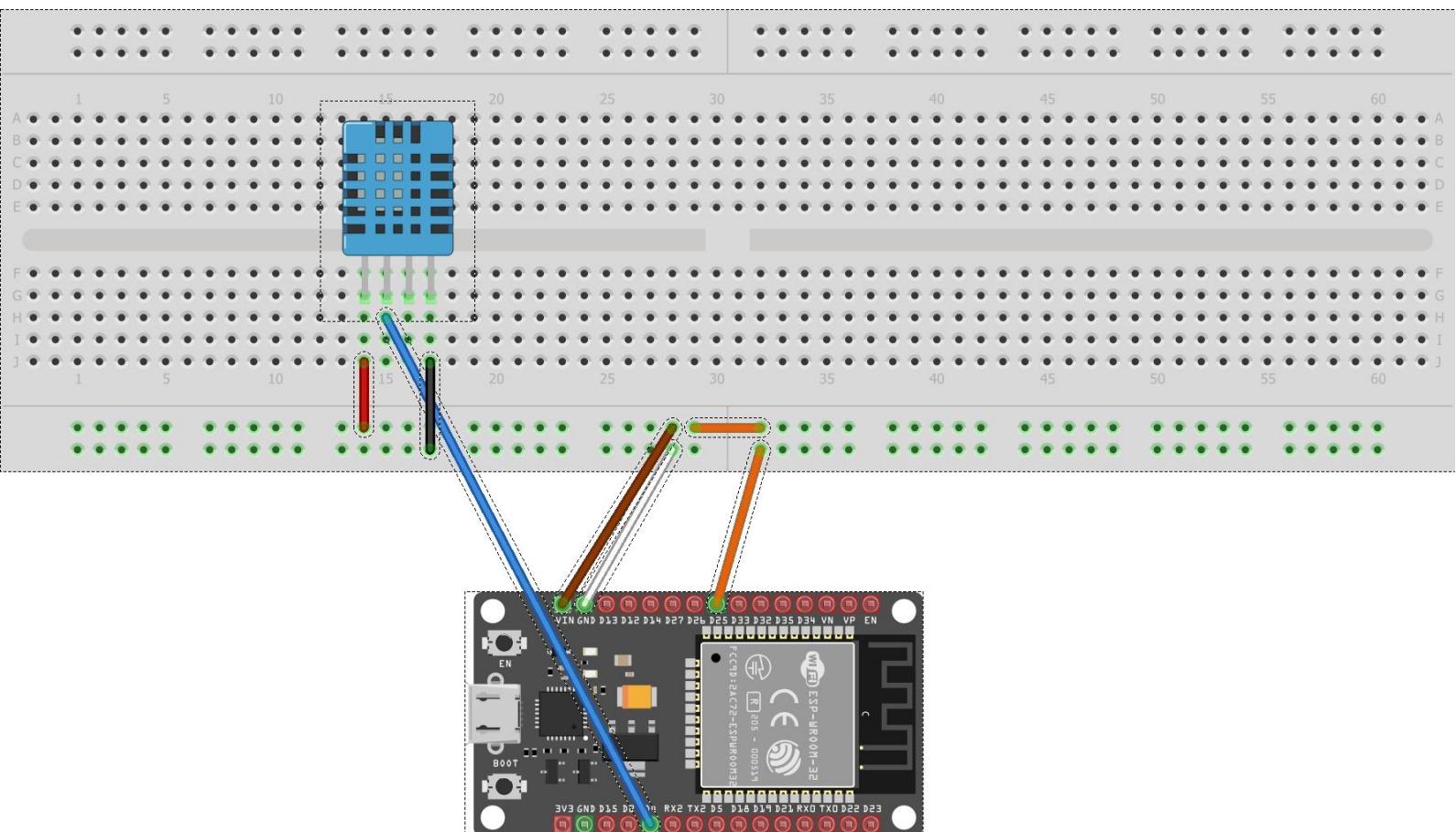
- Rango de temperatura: 0 to 50 °C +/-2 °C
- Rango de humedad: 20 to 90% +/-5%
- Tensión de funcionamiento: 3 – 5.5 V DC
- Suministro de corriente: 0.5 – 2.5 mA
- Tiempo de muestreo: 1 second

Este ha sido el procedimiento a sido el siguiente para subir enviar el dato y subirlo a la base de datos:

Resistencia de 10k ohmios

Esquema de conexión

En primer lugar, este sería el esquema de conexiones que utilizaremos.



El material utilizado es:

- 3 cables, sin importar el color.
- Una Resistencia de 10k ohmios(en nuestro caso no la hemos usado).
- Sensor de temperatura DH11.

En internet hay ejemplos que usan esta resistencia pero en nuestro caso no hemos utilizado ninguna.

Especificaciones de la resistencia:

Cantidad de bandas

4 Bandas 5 Bandas 6 Bandas

Parámetros de la resistencia

Salida

1.^a banda de color

Rojo 2 ▾

2.^a banda de color

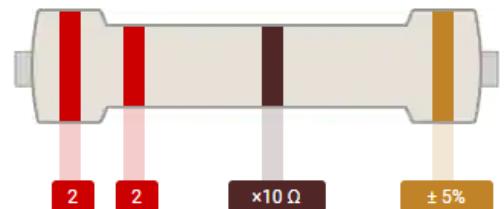
Rojo 2 ▾

Multiplicador

Marrón ×10 Ω ▾

Tolerancia

Oro ± 5% ▾

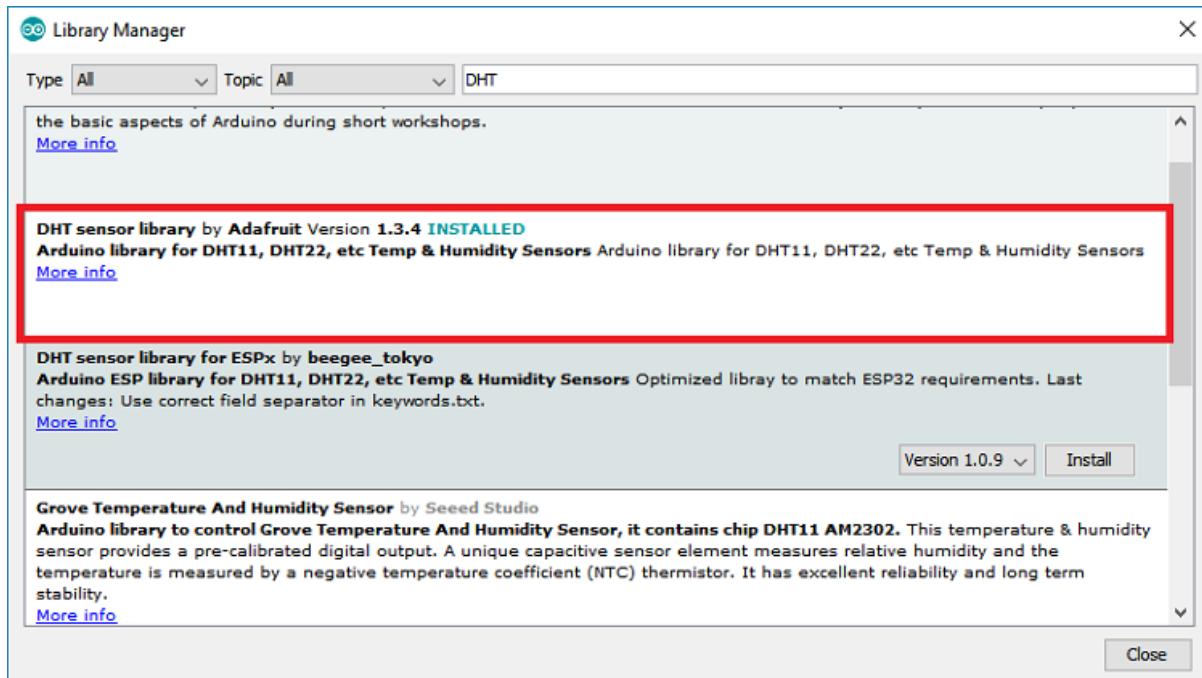


Valor de la resistencia:

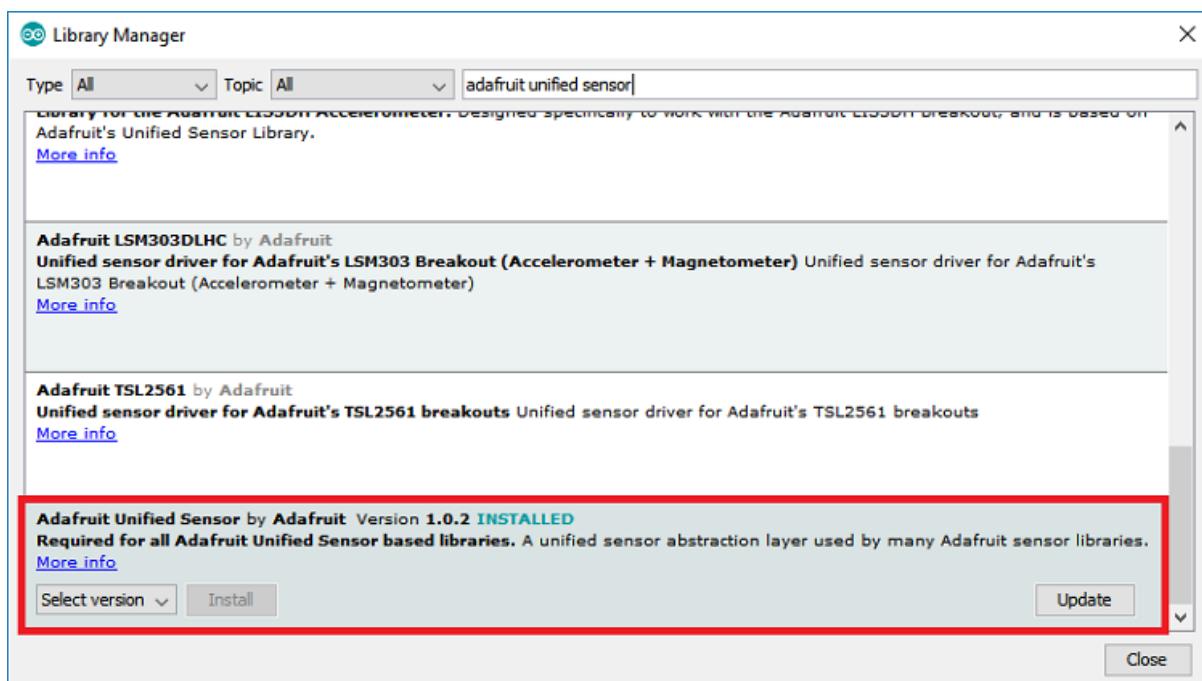
220 Ohms 5%

Librerías para añadir el sensor de temperatura

Vamos a instalar la librería para el DHT11. Para ello hacemos click en Programa - Incluir librería - Administrar Biblioteca e , instalamos "DHT".



Después de instalar la biblioteca DHT de Adafruit, instalamos "Adafruit Unified Sensor".



Capturas de la ejecución del código y del resultado en la API

El código utilizado se puede encontrar en el zip “16062021.zip”

Código del archivo Emisor. Éste código hace lo mismo que el 15062021.zip donde subiamos a la base de datos los valores recogidos por el sensor de Agua pero ahora lo que hacemos es conectar el sensor de temperatura y humedad DHT11 y mostrar los datos en el Monitor Serie.

Por cuestión de tiempo no hemos podido completar el código para que suba el valor de temperatura a la API aunque no debería llevar mucho tiempo ya que tan solo tendríamos que enviar el valor recogido por el sensor en LoRaMessage y en el receptor separar los valor igual que lo hacemos con waterSensor.

```
#include <SPI.h>
#include <LoRa.h>
#include "DHT.h"
#define DHTPIN 4
#define DHTTYPE DHT11

// Inicializamos el sensor DHT.
DHT dht(DHTPIN, DHTTYPE);

//define the pins used by the transceiver module
#define ss 5
#define rst 14
#define dio0 2

int counter = 0;
String LoRaMessage = "";
int idEmisor = 10;

int tipoMedidaSensor4 = 0;
int tipoMedidaSensor3 = 3;
int tipoMedidaSensor2 = 2;
int tipoMedidaSensor1 = 1;

int datoSensor4 = 0;      // variable para almacenar valor leido de
int datoSensor3 = 0;      // variable para almacenar valor leido de Agua
int datoSensor2 = 0;      // variable para almacenar valor leido de Luz
int datoSensor1 = 0;      // variable para almacenar valor leido de
Temperatura
```

```
const int LED = 12; // Declaro Pin 12 de ESP32 donde está conectado el LED

// set pin numbers
const int buttonPin = 35; // the number of the pushbutton pin

void setup() {
    //initialize Serial Monitor
    Serial.begin(115200);

    while (!Serial);

    //setup LoRa transceiver module
    LoRa.setPins(ss, rst, dio0);
    pinMode(LED, OUTPUT);
    Serial.println(F("DHTxx test!"));

    dht.begin();

    //replace the LoRa.begin(---E-) argument with your location's frequency
    //433E6 for Asia
    //866E6 for Europe
    //915E6 for North America
    while (!LoRa.begin(866E6)) {
        Serial.println(".");
        delay(500);
    }
    // Change sync word (0xF3) to match the receiver
    // The sync word assures you don't get LoRa messages from other LoRa transceivers
    // ranges from 0-0xFF
    LoRa.setSyncWord(0xF3);
    Serial.println("LoRa Initializing OK!");
}

void loop() {
    delay(2000); //Es un sensor lento, por lo que hay que darle tiempo.
    float h = dht.readHumidity();
    float t = dht.readTemperature();

    if (isnan(h) || isnan(t)) {
        Serial.println(F("Failed to read from DHT sensor!"));
    }
}
```

```
    return;
}

Serial.print(F("Humedad: "));
Serial.print(h);
Serial.print(F("% Temperatura: "));
Serial.print(t);
Serial.println(F("°C"));

Serial.print("Sending packet: ");
Serial.println(counter);

// read the state of the pushbutton value
// datoSensor3 = digitalRead(buttonPin);
datoSensor3 = analogRead(buttonPin);
Serial.print("idEmisor: ");
Serial.println(idEmisor);

Serial.print("Sending datoSensor3: ");
Serial.println(datoSensor3);

LoRaMessage = String(idEmisor) + "/" + String(datoSensor3) + "&" +
String(tipoMedidaSensor3);

//Send LoRa packet to receiver
LoRa.beginPacket();

LoRa.print(LoRaMessage);

LoRa.endPacket();

int packetSize = LoRa.parsePacket();
if (packetSize) {
    // received a packet
    Serial.print("Received packet ''");

    // read packet
    while (LoRa.available()) {
        String LoRaData = LoRa.readString();
        Serial.print(LoRaData);
    }
}
```

```

    // print RSSI of packet
    Serial.print("' with RSSI ");
    Serial.println(LoRa.packetRssi());
}

counter++;
delay(10000);
}

```

Código del archivo Receptor. Éste código es el mismo que el de 15062021.zip. Solo sube el valor recibido por waterSensor ya que el nuevo valor de temperatura no se le enviamos al receptor y por ende, no puede subir nada.

```

#include <LoRa.h>
#include <WiFi.h>
#include "WiFi.h"

/**********
Modified from the examples of the Arduino LoRa library
More resources: https://randomnerdtutorials.com
*****/

#include <SPI.h>

//define the pins used by the transceiver module
#define ss 5
#define rst 14
#define dio0 2

const char* wifi_ssid = "MiWifi";
const char* wifi_password = "942543459";
unsigned long previousMillis = 0;
unsigned long interval = 30000;
const char* host = "192.168.1.111";
int numero = 0;
int counter = 0;
String loraMessage;
String LoRaData;

String datoSensor3;
String datoSensor2;

```

```
String datoSensor1;
String datoSensor4;

String idEmisor;

String tipoMedidaSensor3;
String tipoMedidaSensor1;
String tipoMedidaSensor2;
String tipoMedidaSensor4;

void setup() {

    conectarWifi(); //Llamamos a la función conectarWifi()

    //initialize Serial Monitor
    Serial.begin(115200);
    while (!Serial);
    Serial.println("LoRa Receiver");

    //setup LoRa transceiver module
    LoRa.setPins(ss, rst, dio0);

    //replace the LoRa.begin(---E--) argument with your location's
frequency
    //433E6 for Asia
    //866E6 for Europe
    //915E6 for North America
    while (!LoRa.begin(866E6)) {
        Serial.println(".");
        delay(500);
    }
    // Change sync word (0xF3) to match the receiver
    // The sync word assures you don't get LoRa messages from other LoRa
transceivers
    // ranges from 0-0xFF
    LoRa.setSyncWord(0xF3);
    Serial.println("LoRa Initializing OK!");
}

void loop() {

    unsigned long currentMillis = millis();
    // if WiFi is down, try reconnecting every CHECK_WIFI_TIME seconds
```

```

if ((WiFi.status() != WL_CONNECTED) && (currentMillis - previousMillis >= interval)) {
    Serial.print(millis());
    Serial.println("Reconnecting to WiFi..."); 
    WiFi.disconnect();
    // WiFi.reconnect();
    WiFi.begin(wifi_ssid, wifi_password);
    previousMillis = currentMillis;
}

// try to parse packet
int packetSize = LoRa.parsePacket();

if (packetSize) {
    Serial.print("PacketSize: ");
    Serial.println(packetSize);
    // received a packet
    Serial.print("Received packet ''");

    String datosLoraRecibidos = "estonorecibionada";

    // read packet
    while (LoRa.available()) {
        datosLoraRecibidos = LoRa.readString();
        Serial.print(datosLoraRecibidos);

        // Get readingID, temperature and soil moisture
        int pos1 = datosLoraRecibidos.indexOf('/');
        int pos2 = datosLoraRecibidos.indexOf('&');
        // int pos3 = datosLoraRecibidos.indexOf('#');

        //     temperature = datosLoraRecibidos.substring(pos1 +1, pos2);
        //     humidity = datosLoraRecibidos.substring(pos2+1, pos3);
        //     pressure = datosLoraRecibidos.substring(pos3+1,
        datosLoraRecibidos.length());

        // LoRaData = datosLoraRecibidos.substring(0, pos1);
        idEmisor = datosLoraRecibidos.substring(0, pos1);
        datoSensor3 = datosLoraRecibidos.substring(pos1 + 1, pos2);
        // datoSensor3 = datosLoraRecibidos.substring(pos1 + 1,
        datosLoraRecibidos.length());
}

```

```

        tipoMedidaSensor3 = datosLoraRecibidos.substring(pos2 + 1,
datosLoraRecibidos.length());
    }

    // print RSSI of packet
    Serial.print(" with RSSI ");
    Serial.println(LoRa.packetRssi());

    Serial.print("Sending packet: ");
    Serial.println(counter);

// Preparo el actuador

    //Send LoRa packet to receiver
    LoRa.beginPacket();
    LoRa.print("hello ");
    LoRa.print(counter);
    LoRa.endPacket();

    counter++;

    delay(2000);

    // String parametros = "&LoRaData=" + LoRaData + "&datoSensor3=" +
datoSensor3;

    String parametros = "&idEmisor=" + idEmisor + "&datoSensor1=" +
datoSensor1 + "&tipoMedidaSensor1=" + tipoMedidaSensor1+ +
"&datoSensor2=" + datoSensor2 + "&tipoMedidaSensor2=" +
tipoMedidaSensor2 + "&datoSensor3=" + datoSensor3 +
"&tipoMedidaSensor3=" + tipoMedidaSensor3+ "&datoSensor4=" +
datoSensor4 + "&tipoMedidaSensor4=" + tipoMedidaSensor4;

    enviarPOST(parametros);
}

}

void enviarPOST(String parametros)
{
    String url = "/LaGranjaPruebaPHP/Servidor/prueba_arduino.php";
    Serial.print("Conectando a ");
    Serial.println("127.0.0.1");

    WiFiClient client;
}

```

```
const int httpPort = 8080; //8080
// strhost.toCharArray(host, 49);
if (!client.connect(host, httpPort))
{
    Serial.println("conexión fallida");
    return;
}

Serial.print("Requesting URL: ");
Serial.print(url);

// client.print(String("POST ") + url + " HTTP/1.1" + "\r\n" +
"Host: " + host + "\r\n" + "Connection: close\r\n\r\n");
client.println("POST " + url + " HTTP/1.1");
client.println("Host: 192.168.1.111");      // + host);
client.println("Cache-Control: no-cache");
client.println("Content-Type: application/x-www-form-urlencoded");
client.print("Content-Length: ");
client.println(parametros.length());
client.println();
client.println(parametros);
// Ejemplo linea de parámetros:
// LoRaData=123&idDispositivo=23&idMedida=34&nomnbre=Antonio

unsigned long timeout = millis();
while (client.available() == 0)
{
    if (millis() - timeout > 5000)
    {
        Serial.println(">>>Client Timeout !");
        client.stop();
        return;
    }
}
while (client.available())
{
    String line = client.readStringUntil('\r');
    Serial.print(line);
}

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

```
//Estas son las funciones que se utilizan
void conectarWifi()
{
    Serial.begin(115200);
    Serial.println();
    while (!Serial);
    Serial.println("Prueba de POST contra el servidor desde Arduino");
    Serial.print("Conectando a ");
    Serial.println(wifi_ssid);

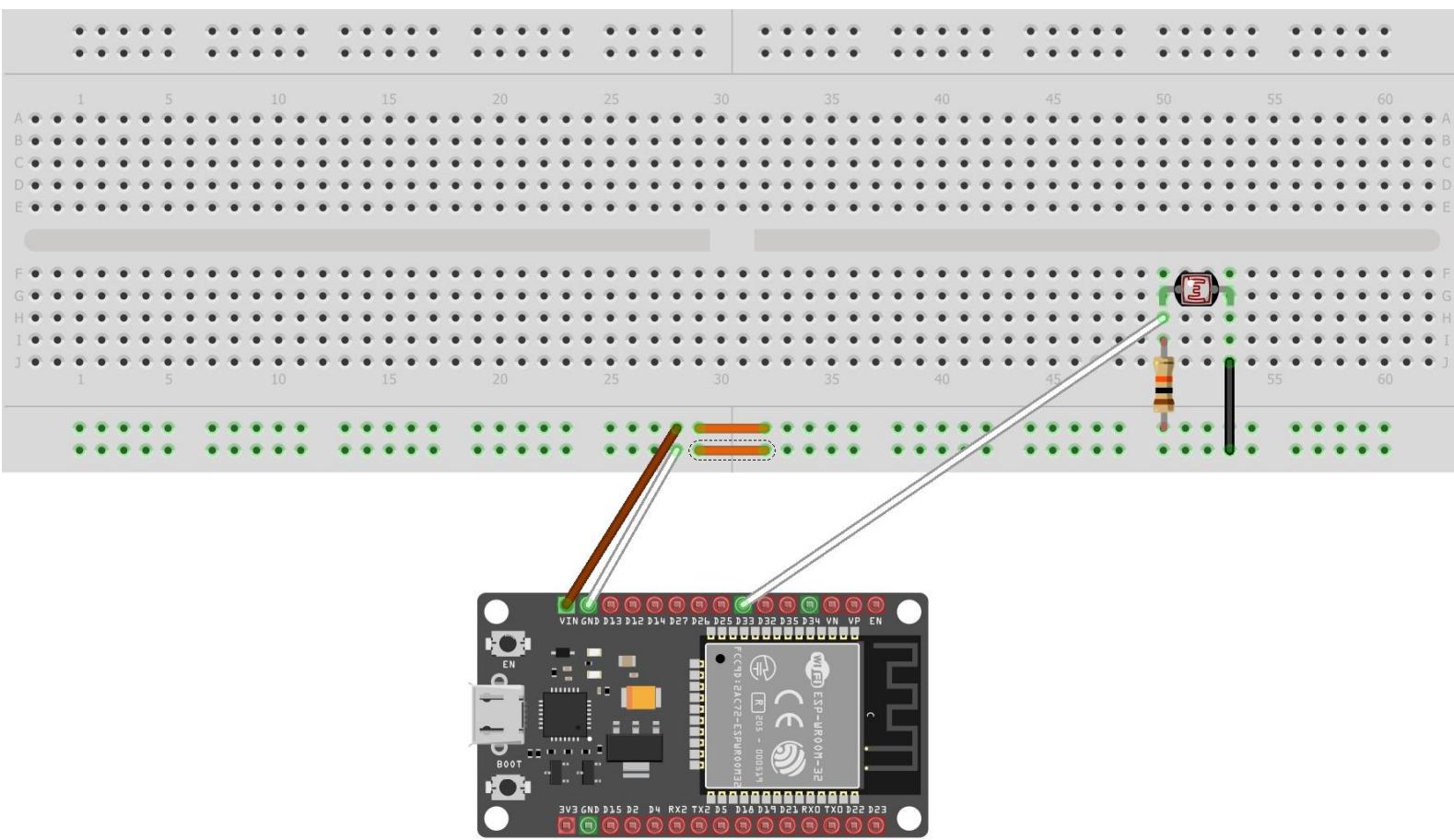
    WiFi.begin(wifi_ssid, wifi_password);

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

    Serial.println();
    Serial.println("Conectado al WiFi");
    Serial.println("Direccion IP:  ");
    Serial.println(WiFi.localIP());
}
```

Conexión del sensor de luz

Esquema de conexión



Este sensor de luz tiene 2 pines.

Un pin se usa para la lectura de datos, en este caso irá conectado al pin(GPIO 33) que es un pin de entrada analógica de ESP32. Esto llevará una resistencia de 10k ohmios a la corriente de + 5V pin(VIN).

El otro (tierra) irá conectado al pin(GND) de ESP32.

El material utilizado es:

- 3 cables, sin importar el color. Uno de los cables que conectamos es el GPIO 33
- Una Resistencia de 10k ohmios.
- Sensor de luz.

Especificaciones de la resistencia:

Cantidad de bandas

4 Bandas 5 Bandas 6 Bandas

Parámetros de la resistencia

1.ª banda de color

Marrón 1 ▾

2.ª banda de color

Negro 0 ▾

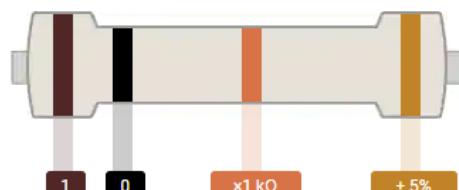
Multiplicador

Naranja $\times 1\text{ k}\Omega$ ▾

Tolerancia

Oro $\pm 5\%$ ▾

Salida



Valor de la resistencia:

10k Ohms 5%

Capturas de la ejecución del código y del resultado en la API

El código utilizado se puede encontrar en el zip “17062021.zip”

Código del archivo Emisor. Este código se encarga de leer los datos de los sensores de temperatura, luz y agua. Los mostramos en el Monitor Serie y se los manda al receptor usando LoRaMessage para que después los suba a la API.

```
// Incluyo las librerias
#include <SPI.h>
#include <LoRa.h>
#include "DHT.h"
#define DHTPIN 4
#define DHTTYPE DHT11

// Inicializamos el sensor DHT.
DHT dht(DHTPIN, DHTTYPE);

const int ldr = 33;// LDR (Sensor de luz o Fotoresistor)

//define the pins used by the transceiver module
#define ss 5
#define rst 14
#define dio0 2

int counter = 0;
String LoRaMessage = "";
int idEmisor = 10;

int tipoMedidaSensor4 = 0;
int tipoMedidaSensor3 = 3;
int tipoMedidaSensor2 = 2;
int tipoMedidaSensor1 = 1;

int datoSensor4 = 0;      // variable para almacenar valor leido de
int datoSensor3 = 0;      // variable para almacenar valor leido de Agua
int datoSensor2 = 0;      // variable para almacenar valor leido de Luz
int datoSensor1 = 0;      // variable para almacenar valor leido de
Temperatura

const int LED = 12; // Declaro Pin 12 de ESP32 donde está conectado el
LED
// set pin numbers
```

```
const int buttonPin = 35; // Declaro Pin 35 de ESP32 donde está
conectado el LED

void setup() {
    //initialize Serial Monitor
    Serial.begin(115200);

    while (!Serial);

    //setup LoRa transceiver module
    LoRa.setPins(ss, rst, dio0);
    pinMode(LED, OUTPUT);

    dht.begin();

    //replace the LoRa.begin(---E--) argument with your location's
frequency
    //433E6 for Asia
    //866E6 for Europe
    //915E6 for North America
    while (!LoRa.begin(866E6)) {
        Serial.println(".");
        delay(500);
    }
    // Change sync word (0xF3) to match the receiver
    // The sync word assures you don't get LoRa messages from other LoRa
transceivers
    // ranges from 0-0xFF
    LoRa.setSyncWord(0xF3);
    Serial.println("LoRa Initializing OK!");
}

void loop() {
    delay(2000); //Es un sensor lento, por lo que hay que darle tiempo.
Funciona cada 1s
    float h = dht.readHumidity(); //Leo la humedad y la guardo en la
variable h
    float t = dht.readTemperature(); //Leo la temperatura y la guardo en
la variable t. Los valores están en Celsius de forma predeterminada
pero se pueden pasar a Fahrenheit
```

```

if (isnan(h) || isnan(t)) { // Si devuelve nan aviso de que ha
fallado la lectura de datos
    Serial.println(F("Fallo en la lectura de los datos del sensor
DHT!"));
    return;
}

// Imprimo los datos de humedad y temperatura en el Monitor Serie para
comprobar que funciona correctamente
Serial.print(F("Humedad: "));
Serial.print(h);
Serial.print(F("% Temperatura: "));
Serial.print(t);
Serial.println(F("°C"));

Serial.print("Sending packet: ");
Serial.println(counter);

// Leo el valor del pin35 y lo guardo en la varibale datoSensor3
datoSensor3 = analogRead(buttonPin);
Serial.print("idEmisor: ");
Serial.println(idEmisor);

Serial.print("datoSensor3: ");
Serial.println(datoSensor3);

datoSensor1 = t;
Serial.print("datoSensor1: ");
Serial.println(datoSensor1);

datoSensor2 = analogRead(ldr);

Serial.print("datoSensor2: ");
Serial.println(datoSensor2);

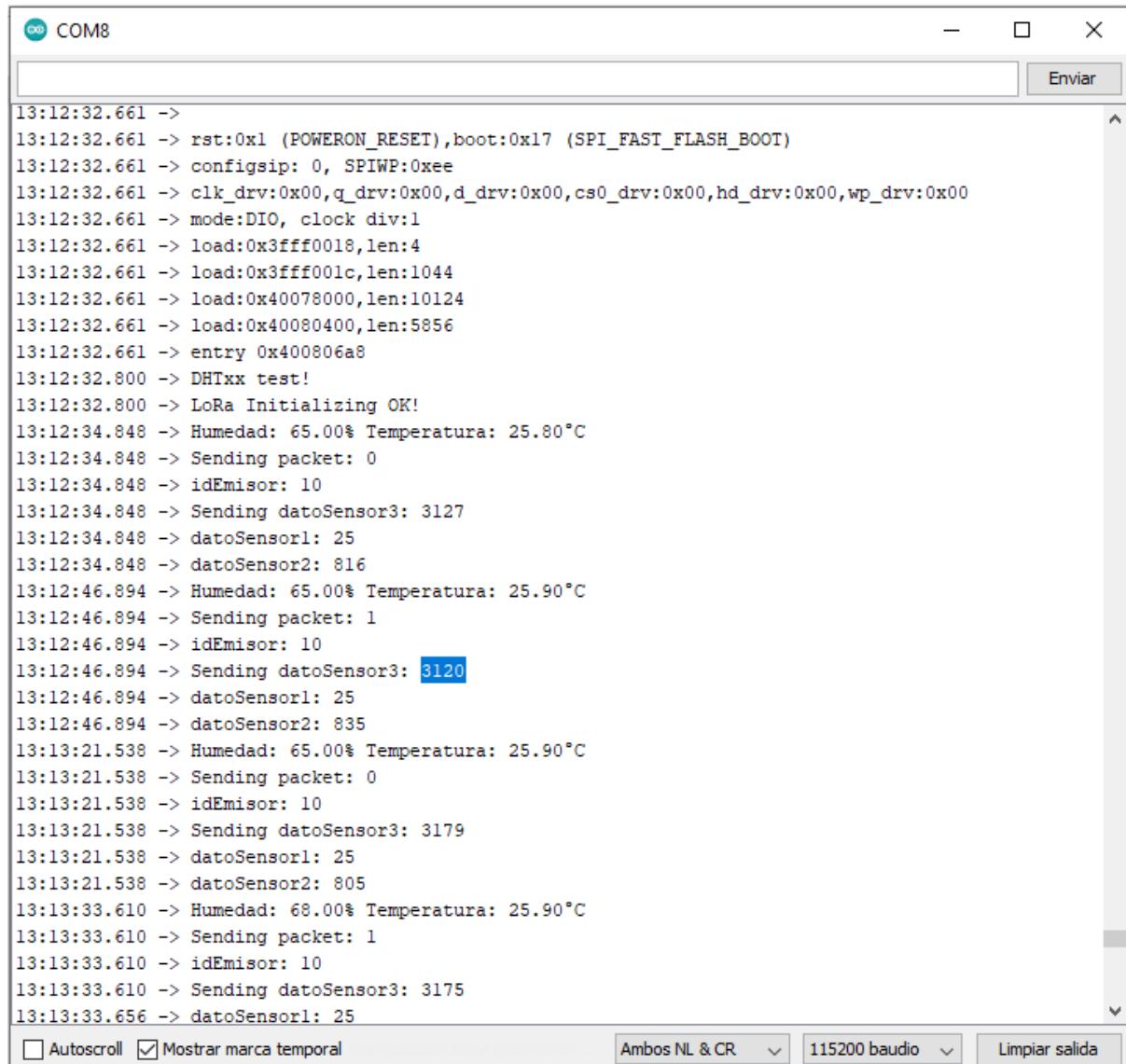
//Guardo todas los datos de los sensores en la variable LoRaMessage
para poder enviarlo por LoRa
LoRaMessage = String(idEmisor) + "/" + String(datoSensor1) + "&" +
String(tipoMedidaSensor1) + "#" + String(datoSensor2) + "@" +
String(tipoMedidaSensor2) + "$" + String(datoSensor3) + "^" +
String(tipoMedidaSensor3);

```

```
//Send LoRa packet to receiver
LoRa.beginPacket();
LoRa.print(LoRaMessage);
LoRa.endPacket();

//Recibimos el mensaje del receptor y lo imprimimos en el Monitor
Serie
int packetSize = LoRa.parsePacket();
if (packetSize) {
    // received a packet
    Serial.print("Received packet '");
    // read packet
    while (LoRa.available()) {
        String datosLoraRecibidos = LoRa.readString();
        Serial.print(datosLoraRecibidos);
    }
    // print RSSI of packet
    Serial.print("' with RSSI ");
    Serial.println(LoRa.packetRssi());
}
counter++;
delay(10000);
}
```

Captura de la ejecución del **código emisor**. Como podemos ver en esta imagen, imprimimos los datos del sensor de agua en el Monitor Serie.



```
13:12:32.661 ->
13:12:32.661 -> rst:0x1 (POWERON_RESET),boot:0x17 (SPI_FAST_FLASH_BOOT)
13:12:32.661 -> configSip: 0, SPIWP:0xee
13:12:32.661 -> clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
13:12:32.661 -> mode:DIO, clock div:1
13:12:32.661 -> load:0x3fff0018,len:4
13:12:32.661 -> load:0x3fff001c,len:1044
13:12:32.661 -> load:0x40078000,len:10124
13:12:32.661 -> load:0x40080400,len:5856
13:12:32.661 -> entry 0x400806a8
13:12:32.800 -> DHTxx test!
13:12:32.800 -> LoRa Initializing OK!
13:12:34.848 -> Humedad: 65.00% Temperatura: 25.80°C
13:12:34.848 -> Sending packet: 0
13:12:34.848 -> idEmisor: 10
13:12:34.848 -> Sending datoSensor3: 3127
13:12:34.848 -> datoSensor1: 25
13:12:34.848 -> datoSensor2: 816
13:12:46.894 -> Humedad: 65.00% Temperatura: 25.90°C
13:12:46.894 -> Sending packet: 1
13:12:46.894 -> idEmisor: 10
13:12:46.894 -> Sending datoSensor3: 3120
13:12:46.894 -> datoSensor1: 25
13:12:46.894 -> datoSensor2: 835
13:13:21.538 -> Humedad: 65.00% Temperatura: 25.90°C
13:13:21.538 -> Sending packet: 0
13:13:21.538 -> idEmisor: 10
13:13:21.538 -> Sending datoSensor3: 3179
13:13:21.538 -> datoSensor1: 25
13:13:21.538 -> datoSensor2: 805
13:13:33.610 -> Humedad: 68.00% Temperatura: 25.90°C
13:13:33.610 -> Sending packet: 1
13:13:33.610 -> idEmisor: 10
13:13:33.610 -> Sending datoSensor3: 3175
13:13:33.656 -> datoSensor1: 25
```

Autoscroll Mostrar marca temporal Ambos NL & CR 115200 baudio Limpiar salida

Código del archivo Receptor encargado de recibir los valores de los sensores de temperatura, luz y humedad y subirlos a la API.

En primer lugar, leemos el string datosLoraRecibidos y almacenamos los datos recibidos, después tenemos la función indexOf que se encarga de localizar los valores. También utilizamos el substring para extraer los caracteres de un punto a otro, primero de la posición 0 a la 1, 2 al 3, 4 al 5... y por último de la 6 al resto de posiciones.

Por último guardamos los datos en las variables idEmisor, datoSensor1, datoSensor2, datoSensor3 y tipoMedidaSensor1, tipoMedidaSensor2 y tipoMedidaSensor3 dentro de la variable parámetros para después hacer el POST con parámetros.

Este dispositivo cumple también la función de emisor enviando mensajes.

```
#include <LoRa.h>
#include <WiFi.h>
#include "WiFi.h"

/*
Modified from the examples of the Arduino LoRa library
More resources: https://randomnerdtutorials.com
*/

#include <SPI.h>

//define the pins used by the transceiver module
#define ss 5
#define rst 14
#define dio0 2

const char* wifi_ssid = "MiWifi";
const char* wifi_password = "942543459";
unsigned long previousMillis = 0;
unsigned long interval = 30000;
const char* host = "192.168.1.111";
int numero = 0;
int counter = 0;
String loraMessage;
// String LoRaData;

String datoSensor3;
String datoSensor2;
```

```
String datoSensor1;
String datoSensor4;

String idEmisor;

String tipoMedidaSensor3;
String tipoMedidaSensor1;
String tipoMedidaSensor2;
String tipoMedidaSensor4;

void setup() {

    conectarWifi(); //Llamamos a la función conectarWifi()

    //initialize Serial Monitor
    Serial.begin(115200);
    while (!Serial);
    Serial.println("LoRa Receiver");

    //setup LoRa transceiver module
    LoRa.setPins(ss, rst, dio0);

    //replace the LoRa.begin(---E-) argument with your location's
frequency
    //433E6 for Asia
    //866E6 for Europe
    //915E6 for North America
    while (!LoRa.begin(866E6)) {
        Serial.println(".");
        delay(500);
    }
    // Change sync word (0xF3) to match the receiver
    // The sync word assures you don't get LoRa messages from other LoRa
transceivers
    // ranges from 0-0xFF
    LoRa.setSyncWord(0xF3);
    Serial.println("LoRa Initializing OK!");
}

void loop() {

    unsigned long currentMillis = millis();
    // if WiFi is down, try reconnecting every CHECK_WIFI_TIME seconds
```

```

if ((WiFi.status() != WL_CONNECTED) && (currentMillis - previousMillis >= interval)) {
    Serial.print(millis());
    Serial.println("Reconnecting to WiFi...");
    WiFi.disconnect();
    // WiFi.reconnect();
    WiFi.begin(wifi_ssid, wifi_password);
    previousMillis = currentMillis;
}

// try to parse packet
int packetSize = LoRa.parsePacket();

if (packetSize) {
    Serial.print("PacketSize: ");
    Serial.println(packetSize);
    // received a packet
    Serial.print("Received packet ''");

    // String datosLoraRecibidos = "estonorecibionada";

    // read packet
    while (LoRa.available()) {
        String datosLoraRecibidos = LoRa.readString();
        Serial.print(datosLoraRecibidos);

        // Get readingID, temperature and soil moisture
        int pos1 = datosLoraRecibidos.indexOf('/');
        int pos2 = datosLoraRecibidos.indexOf('&');
        int pos3 = datosLoraRecibidos.indexOf('#');
        int pos4 = datosLoraRecibidos.indexOf('@');
        int pos5 = datosLoraRecibidos.indexOf('$');
        int pos6 = datosLoraRecibidos.indexOf('^');

        //     temperature = datosLoraRecibidos.substring(pos1 +1, pos2);
        //     humidity = datosLoraRecibidos.substring(pos2+1, pos3);
        //     pressure = datosLoraRecibidos.substring(pos3+1,
datosLoraRecibidos.length()));

        // LoRaData = datosLoraRecibidos.substring(0, pos1);
        idEmisor = datosLoraRecibidos.substring(0, pos1);
}

```

```

datoSensor1 = datosLoraRecibidos.substring(pos1 + 1, pos2);
tipoMedidaSensor1 = datosLoraRecibidos.substring(pos2 + 1, pos3);
datoSensor2 = datosLoraRecibidos.substring(pos3 + 1, pos4);
tipoMedidaSensor2 = datosLoraRecibidos.substring(pos4 + 1, pos5);
datoSensor3 = datosLoraRecibidos.substring(pos5 + 1, pos6);
tipoMedidaSensor3 = datosLoraRecibidos.substring(pos6 + 1,
datosLoraRecibidos.length());
}

// print RSSI of packet
Serial.print(" with RSSI ");
Serial.println(LoRa.packetRssi());

Serial.print("Sending packet: ");
Serial.println(counter);

//Send LoRa packet to receiver
LoRa.beginPacket();
LoRa.print("hello ");
LoRa.print(counter);
LoRa.endPacket();

counter++;

delay(2000);

// String parametros = "&LoRaData=" + LoRaData + "&datoSensor3=" +
datoSensor3;
String parametros = "&idEmisor=" + idEmisor + "&datoSensor1=" +
datoSensor1 + "&tipoMedidaSensor1=" + tipoMedidaSensor1 + +
"&datoSensor2=" + datoSensor2 + "&tipoMedidaSensor2=" +
tipoMedidaSensor2 + "&datoSensor3=" + datoSensor3 +
"&tipoMedidaSensor3=" + tipoMedidaSensor3 + "&datoSensor4=" +
datoSensor4 + "&tipoMedidaSensor4=" + tipoMedidaSensor4;

enviarPOST(parametros);
}

void enviarPOST(String parametros)
{
String url = "/LaGranjaPruebaPHP/Servidor/prueba_arduino.php";
Serial.print("Conectando a ");

```

```
Serial.println("127.0.0.1");

WiFiClient client;
const int httpPort = 8080; //8080
// strhost.toCharArray(host, 49);
if (!client.connect(host, httpPort))
{
    Serial.println("conexión fallida");
    return;
}

Serial.print("Requesting URL: ");
Serial.print(url);

// client.print(String("POST ") + url + " HTTP/1.1" + "\r\n" +
"Host: " + host + "\r\n" + "Connection: close\r\n\r\n");
client.println("POST " + url + " HTTP/1.1");
client.println("Host: 192.168.1.111"); // + host);
client.println("Cache-Control: no-cache");
client.println("Content-Type: application/x-www-form-urlencoded");
client.print("Content-Length: ");
client.println(parametros.length());
client.println();
client.println(parametros);
// Ejemplo línea de parámetros:
// LoRaData=123&idDispositivo=23&idMedida=34&nomnbre=Antonio

unsigned long timeout = millis();
while (client.available() == 0)
{
    if (millis() - timeout > 5000)
    {
        Serial.println(">>>Client Timeout !");
        client.stop();
        return;
    }
}
while (client.available())
{
    String line = client.readStringUntil('\r');
    Serial.print(line);
}
```

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

//Estas son las funciones que se utilizan
void conectarWifi()
{
    Serial.begin(115200);
    Serial.println();
    while (!Serial);
    Serial.println("Prueba de POST contra el servidor desde Arduino");
    Serial.print("Conectando a ");
    Serial.println(wifi_ssid);

    WiFi.begin(wifi_ssid, wifi_password);

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

    Serial.println();
    Serial.println("Conectado al WiFi");
    Serial.println("Direccion IP: ");
    Serial.println(WiFi.localIP());
}
```

Captura de la ejecución del **código receptor**. Como podemos ver en esta imagen, el receptor recibe el mensaje del emisor y lo muestra en el Monitor Serie. Hace el POST a la API usando pruebaarduino.php.

```
13:12:42.881 -> .
13:12:43.351 -> .LoRa Receiver
13:12:43.631 -> LoRa Initializing OK!
13:12:46.941 -> PacketSize: 20
13:12:46.941 -> Received packet '10/25&1#835@2$3120^3' with RSSI -50
13:12:46.941 -> Sending packet: 0
13:12:48.995 -> Conectando a 127.0.0.1
13:12:48.995 -> Requesting URL: /LaGranjaPruebaPHP/Servidor/prueba_arduino.phpHTTP/1.1 200 OK
13:12:49.691 -> Date: Thu, 17 Jun 2021 11:12:49 GMT
13:12:49.691 -> Server: Apache/2.4.46 (Win64) OpenSSL/1.1.1g PHP/7.4.10
13:12:49.691 -> X-Powered-By: PHP/7.4.10
13:12:49.691 -> Content-Length: 557
13:12:49.691 -> Content-Type: text/html; charset=UTF-8
13:12:50.673 ->
13:12:50.673 -> ----- Se han recibido por LORA las siguientes parejas CLAVE => VALOR idEmisor
13:12:50.673 ->
13:12:50.673 -> datoSensor1 => 25 -----
13:12:50.673 ->
13:12:50.673 -> tipoMedidaSensor1 => 1 -----
13:12:50.719 ->
13:12:50.719 -> datoSensor2 => 835 -----
13:12:50.719 ->
13:12:50.719 -> tipoMedidaSensor2 => 2 -----
13:12:50.719 ->
13:12:50.719 -> datoSensor3 => 3120 -----
13:12:50.719 ->
13:12:50.719 -> tipoMedidaSensor3 => 3 -----
13:12:50.719 ->
13:12:50.719 -> datoSensor4 => -----
13:12:50.719 ->
13:12:50.719 -> tipoMedidaSensor4 => -----
13:12:50.719 ->
13:12:50.719 -> Valor final de idEmisor: 10
13:12:50.719 -> Valor final de Temperatura: 25 y es de tipo string
13:12:50.719 -> Valor final de Luz: 835 y es de tipo string
13:12:50.719 -> Valor final de waterSensor: 3120 y es de tipo string
13:12:50.719 ->
13:12:50.719 -> Cerrando conexion
```

Autoscroll Mostrar marca temporal Ambos NL & CR 115200 baudio Limpiar salida

Código del archivo .php. En primer lugar realizamos la conexión con la API y después empezamos a identificar y reconocer los valores recibidos, tanto los datos de los sensores como idEmisor como el tipo de medida que es. Finalmente sustituimos los datos por los datos de \$POST usando foreach. También imprimimos el valor de los sensores y el tipo de variable que es para controlar al 100% los resultados.

```
<?php

$arrayDatos = array();
// Es mejor tener los datos de conexión por separado, así es más fácil
cambiarlos
echo "----- Se han recibido por LORA las siguientes parejas CLAVE
=> VALOR ";
foreach ($_POST as $clave => $valor) {
    echo ($clave . " => " . $valor . " ----- \n\n");
}

// // Cojo unos valores según la fecha y la hora, para que los datos
que introducimos en la BBDD no sean siempre iguales
// $LoRaData = "defecto " . Date("d/m/y - h:m:s"); // Valor por
defecto de LoRaData, por si no viene definido
// $numero = Date("hms"); // Valor para el
número, que es la hora minuto y segundo seguidos

//Primero compruebo si se ha pasado por POST el valor para LoRaData
if (isset($_POST['idEmisor'])) {
    // Si se ha definido lo cojo.
    $idEmisor = $_POST['idEmisor'];
}
echo "Valor final de idEmisor: " . $idEmisor . " \n";

// //Primero compruebo si se ha pasado por POST el valor para
waterSensor

if (isset($_POST['datoSensor1'])) {
    // Si se ha definido lo cojo.
    $datoSensor1 = $_POST['datoSensor1'];
}

if (isset($_POST['tipoMedidaSensor1'])) {
    // Si se ha definido lo cojo.
```

```
$tipoMedidaSensor1 = $_POST['tipoMedidaSensor1'];
$arrayDatos += [$datoSensor1 => $tipoMedidaSensor1];
}

if (isset($_POST['datoSensor2'])) {
    // Si se ha definido lo cojo.
    $datoSensor2 = $_POST['datoSensor2'];
}

if (isset($_POST['tipoMedidaSensor2'])) {
    // Si se ha definido lo cojo.
    $tipoMedidaSensor2 = $_POST['tipoMedidaSensor2'];
    $arrayDatos += [$datoSensor2 => $tipoMedidaSensor2];

}if (isset($_POST['datoSensor3'])) {
    // Si se ha definido lo cojo.
    $datoSensor3 = $_POST['datoSensor3'];
}

if (isset($_POST['tipoMedidaSensor3'])) {
    // Si se ha definido lo cojo.
    $tipoMedidaSensor3 = $_POST['tipoMedidaSensor3'];
    $arrayDatos += [$datoSensor3 => $tipoMedidaSensor3];

}if (isset($_POST['datoSensor4'])) {
    // Si se ha definido lo cojo.
    $datoSensor4 = $_POST['datoSensor4'];
}

if (isset($_POST['tipoMedidaSensor4'])) {
    // Si se ha definido lo cojo.
    $tipoMedidaSensor4 = $_POST['tipoMedidaSensor4'];
    $arrayDatos += [$datoSensor4 => $tipoMedidaSensor4];
}

echo "Valor final de Temperatura: " . $datoSensor1." y es de tipo ";
echo gettype($datoSensor1), "\n";

echo "Valor final de Luz: " . $datoSensor2." y es de tipo ";
echo gettype($datoSensor2), "\n";

echo "Valor final de waterSensor: " . $datoSensor3." y es de tipo ";
echo gettype($datoSensor3), "\n";
```

```
try{
    $url = 'http://lagranja40.agllabs.es/api/dispositivos/setMedida';

    $ch = curl_init($url);

    foreach ($arrayDatos as $key => $value) {
        $data = array(
            //Sustituir los datos por los datos de $POST
            'id' => ".$idEmisor.",
            'medida' => ".$key.",
            'tipoMedida' => ".$value."
        );
        $payload = json_encode($data);

        curl_setopt($ch, CURLOPT_POSTFIELDS, $payload);

        curl_setopt($ch, CURLOPT_HTTPHEADER,
array('Content-Type:application/json'));

        curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);

        $result = curl_exec($ch);
    }
    curl_close($ch);

}catch(Exception $e){
    echo $e;
}
```

Captura de la API. Como podemos ver, aparecen los valores de los 3 sensores (temperatura, luz y agua) introducidos en la API además del idEmisor(10).

✓ Mostrando filas 0 - 249 (total de 1000, La consulta tardó 0.0005 segundos.) [id_registro: 3716... - 2217...]

```
SELECT * FROM `registro` WHERE `e_dispositivo` = 10 ORDER BY `registro`.`id_registro` DESC
```

	1	>	>>	Número de filas:	250	Filtrar filas:	Buscar en esta tabla	Ordenar según la clav
+ Opciones								
<input type="checkbox"/>		Editar		Copiar		Borrar	3716	10 2021-06-17 12:21:51 0 3
<input type="checkbox"/>		Editar		Copiar		Borrar	3715	10 2021-06-17 12:21:51 57 2
<input type="checkbox"/>		Editar		Copiar		Borrar	3714	10 2021-06-17 12:21:51 25 1
<input type="checkbox"/>		Editar		Copiar		Borrar	3712	10 2021-06-17 12:19:50 0 3
<input type="checkbox"/>		Editar		Copiar		Borrar	3711	10 2021-06-17 12:19:50 1803 2
<input type="checkbox"/>		Editar		Copiar		Borrar	3710	10 2021-06-17 12:19:50 25 1
<input type="checkbox"/>		Editar		Copiar		Borrar	3708	10 2021-06-17 12:17:53 0 3
<input type="checkbox"/>		Editar		Copiar		Borrar	3707	10 2021-06-17 12:17:53 1801 2
<input type="checkbox"/>		Editar		Copiar		Borrar	3706	10 2021-06-17 12:17:53 25 1
<input type="checkbox"/>		Editar		Copiar		Borrar	3704	10 2021-06-17 12:08:19 6 3
<input type="checkbox"/>		Editar		Copiar		Borrar	3703	10 2021-06-17 12:08:19 1712 2
<input type="checkbox"/>		Editar		Copiar		Borrar	3702	10 2021-06-17 12:08:19 25 1
<input type="checkbox"/>		Editar		Copiar		Borrar	3701	10 2021-06-17 11:39:51 0 3
<input type="checkbox"/>		Editar		Copiar		Borrar	3699	10 2021-06-17 11:39:51 25 1
<input type="checkbox"/>		Editar		Copiar		Borrar	3696	10 2021-06-17 11:23:34 0 3
<input type="checkbox"/>		Editar		Copiar		Borrar	3694	10 2021-06-17 11:20:59 0 3
<input type="checkbox"/>		Editar		Copiar		Borrar	3692	10 2021-06-17 11:14:30 0 3

Actuador

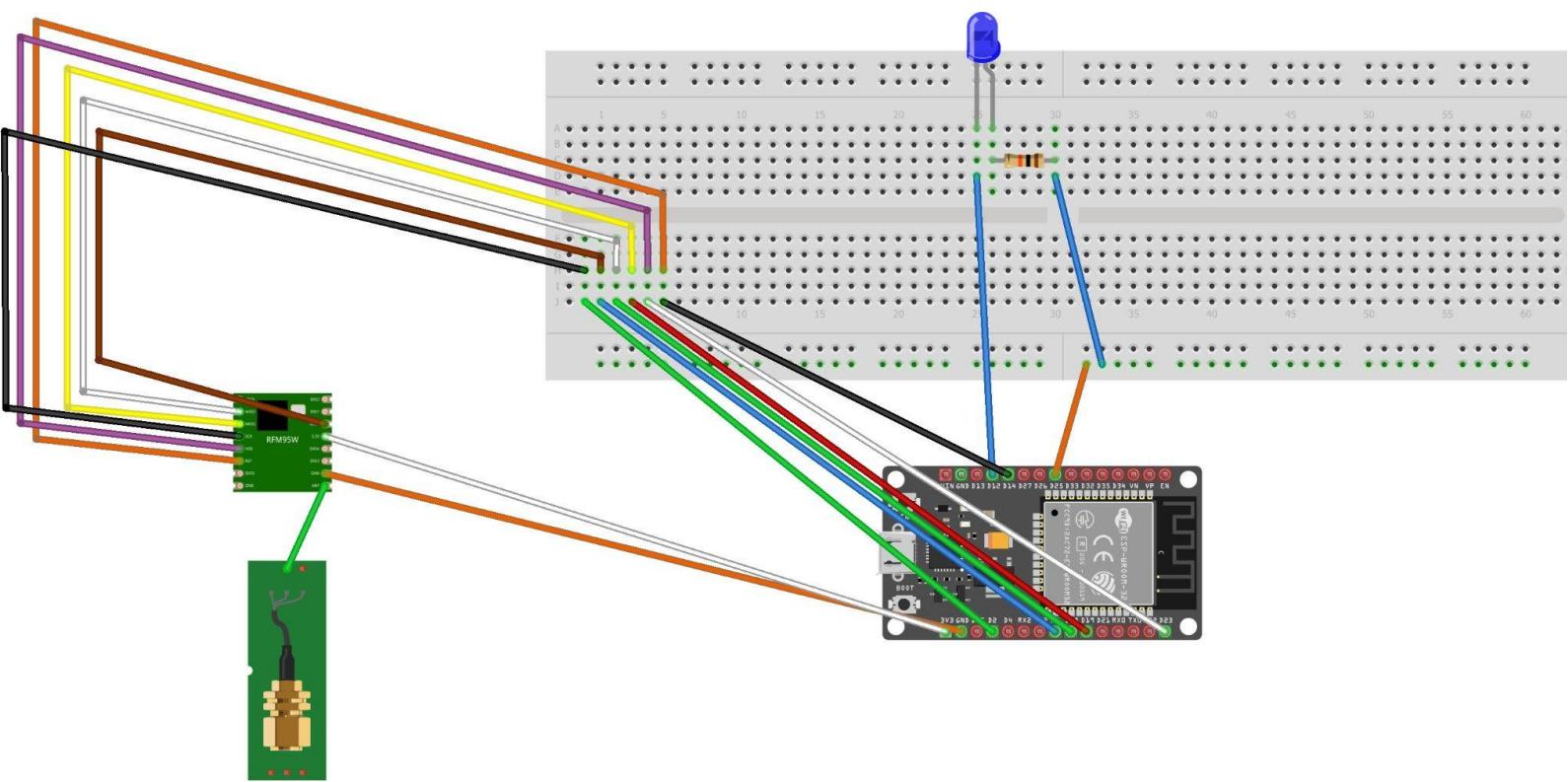
La idea principal del actuador es que cuando el receptor inserta los datos en la API, dicho receptor envíe un mensaje al emisor con las variables id, acción, cantidad.

- id. Identificador del actuador
- acción. Nos especificará con el valor 100 o 150 si se enciende o no. El valor 100 corresponde a encender el LED y 150 apagar el LED.
- cantidad

Primero programaremos un pequeño código para manejar el LED encendiendo y apagando. Para ello necesitamos lo siguiente:

Esquema de conexión

En primer lugar, este sería el esquema de conexiones que utilizaremos.



El material utilizado es:

- Una LED, sin importar el color.
- 2 cables, sin importar el color.
- Una Resistencia de 10k ohmios.

Especificaciones de la resistencia:

Cantidad de bandas

4 Bandas 5 Bandas 6 Bandas

Parámetros de la resistencia

1.ª banda de color

Rojo 2 ▾

2.ª banda de color

Rojo 2 ▾

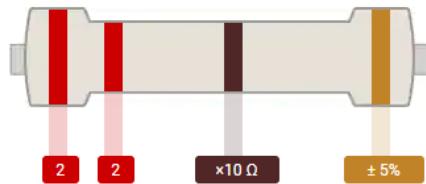
Multiplicador

Marrón $\times 10 \Omega$ ▾

Tolerancia

Oro $\pm 5\%$ ▾

Salida



Valor de la resistencia:

220 Ohms 5%

Conectaremos un cable al ánodo (lado positivo), que tiene una pata más larga y se conectará a un pin GPIO 12, 13 y 14 aunque se puede usar cualquier otro GPIO adecuado.

Conectaremos un cable al cátodo (lado negativo), que tiene una pata más corta y se conectará al pin GND.

Para conectar un LED debemos respetar la polaridad, de lo contrario el dispositivo no funcionará.

Código para manejar el LED

Utilizaremos en primer lugar este código para comprobar que el LED se enciende y se apaga correctamente. Irá insertado en la primera parte del Loop Más adelante, programaremos un código para que cumpla la función de actuador.

```
void loop() {  
  
    digitalWrite(LED, HIGH);  
    delay(1000);  
    digitalWrite(LED, LOW);  
    delay(1000);  
}
```

Bibliografía

API de LoRa

<https://github.com/sandeepmistry/arduino-LoRa/blob/master/API.md>

ESP32 Hardware Design

https://www.espressif.com/sites/default/files/documentation/esp32_hardware_design_guidelines_en.pdf

Comenzando con ESP32

<https://circuitdigest.com/microcontroller-projects/getting-started-with-esp32-with-arduino-ide>

Instalación de la placa ESP32 en Arduino IDE

<https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/>

Para la placa ESP32 el driver es CP210x

<https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers>

ESP32 con LoRa usando Arduino IDE

<https://randomnerdtutorials.com/esp32-lora-rfm95-transceiver-arduino-ide/>

Error al conectarse a ESP32

<https://randomnerdtutorials.com/solved-failed-to-connect-to-esp32-timed-out-waiting-for-packet-header/>

ESP8266

<https://www.luisllamas.es/programar-esp8266-con-el-ide-de-arduino/>

Para la placa ESP8266 el driver es CH341

<http://kio4.com/arduino/1inicio.htm>

Prueba de parpadeo con la placa ESP8266

http://kio4.com/arduino/57modulowifi_7.htm

Actualizaciones de ESP32 OTA (Over-the-Air)

<https://randomnerdtutorials.com/esp32-ota-over-the-air-arduino/>

Referencia de distribución de pines ESP32

<https://randomnerdtutorials.com/esp32-pinout-reference-gpios/>

Entradas y salidas digitales ESP32

<https://randomnerdtutorials.com/esp32-digital-inputs-outputs-arduino/>

Conectar sensor de agua

<https://create.arduino.cc/projecthub/omer-beden/basic-water-sensor-190c7a>

Cómo funciona el sensor de nivel de agua

<https://create.arduino.cc/projecthub/electropeak/make-a-liquid-level-indicator-with-arduino-596bd3>

Conectar sensor de temperatura

<https://randomnerdtutorials.com/esp32-dht11-dht22-temperature-humidity-sensor-arduino-ide/>

Conectar sensor de temperatura DHT11

<https://www.juanjobeunza.com/esp32-dht11/>

Conectar sensor de luz

<https://randomnerdtutorials.com/esp32-weather-station-pcb/>

Explicación sensor de luz

<https://www.arduino.cc/documents/datasheets/LDR-VT90N2.pdf>

Lectura de valores del sensor de luz

<http://www.esp32learning.com/code/esp32-and-ldr-example.php>

Conectar sensor bme280 de Temperature, Humidity, and Pressure

<https://randomnerdtutorials.com/esp32-bme280-arduino-ide-pressure-temperature-humidity/>

Otro ejemplo de estación meteorológica, usando pila

<https://how2electronics.com/lora-based-wireless-weather-station-with-arduino-esp32/>

Estación meteorológica ESP32

<https://randomnerdtutorials.com/esp32-weather-station-pcb/>

Aplicación Fritzing para hacer los conexión de los sensores

<https://www.filehorse.com/es/descargar-fritzing-32/>

Encender LEDs para el actuador

<https://randomnerdtutorials.com/esp32-web-server-websocket-sliders/>

Cómo conectar una antena externa a ESP32

<https://randomnerdtutorials.com/esp32-cam-connect-external-antenna/>