UNIVERSITAT POLITÈCNICA DE VALÈNCIA

BIP DIG-SENSING

Webinar 4

Communications Technologies

Juan Luis Posadas Yagüe
José V. Benlloch-Dualde

UPV
COMPUTER ENG. DEPT. (DISCA)
Universitat Politècnica de València
{jposadas,jbenlloc}@disca.upv.es

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

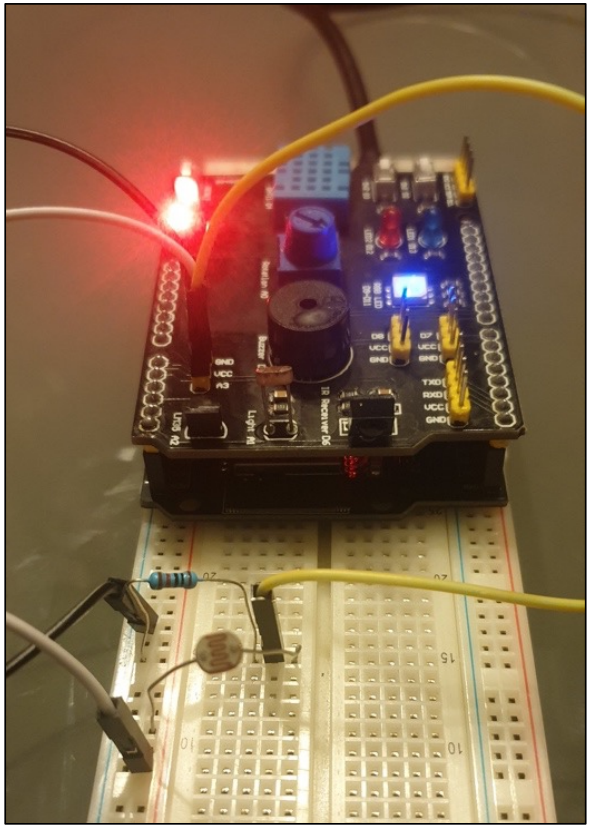
Welcome to the world of communications technologies

- ✓ The ESP32 device is a powerful microcontroller that can connect to a wide range of communication technologies, including **serial**, **Bluetooth**, and **Wi-Fi**

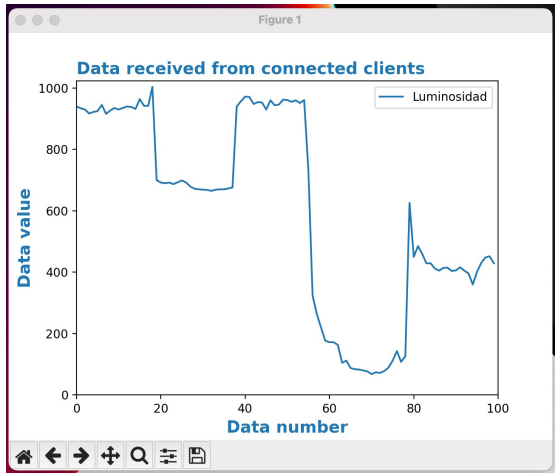
UPV
COMPUTER ENG. DEPT. (DISCA)
Universitat Politècnica de València
{jposadas,jbenlloc}@disca.upv.es

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Our goal is:



- ✓ To communicate ESP32 devices between them and with other devices such as a PC or a mobile phone
- ✓ We will be able to remotely send orders to the ESP32 to, for example, turn a light on or off
- ✓ The ESP32 will also be able to publish the information from its sensors on the internet so that it can be accessed from anywhere



UPV
COMPUTER ENG. DEPT. (DISCA)
Universitat Politècnica de València
{jposadas,jbenlloc}@disca.upv.es

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Some fundamental concepts...

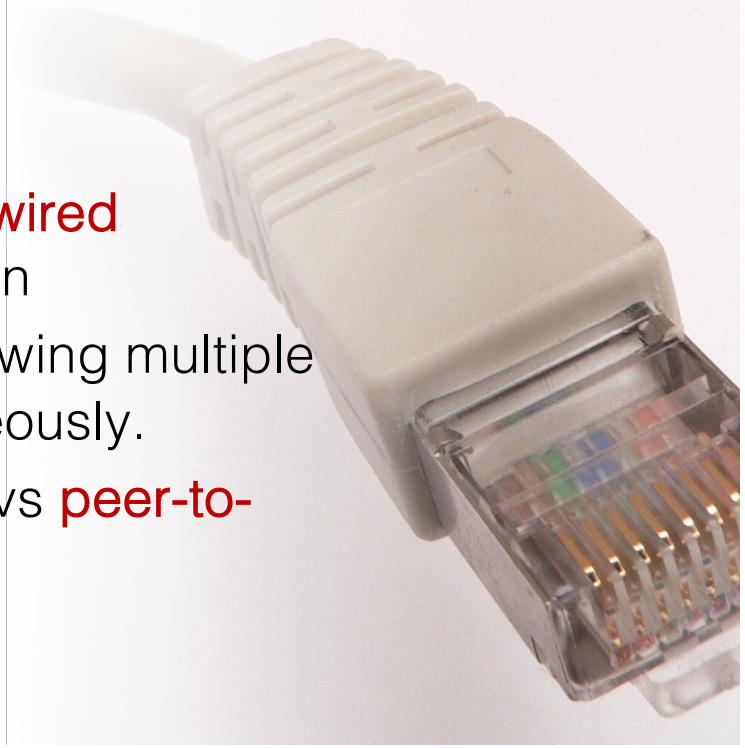
- ✓ Communications: transmission of signals using a common code between the sender and the receiver.

- ✓ The transmission speed is measured in changes of state (1 or 0) per second (hertz: Hz), or in bits per second (bps) or bauds
- ✓ Point-to-point or multipoint communications with wired or wireless connections.

UPV
COMPUTER ENG. DEPT. (DISCA)
Universitat Politècnica de València
{jposadas,jbenlloc}@disca.upv.es

Some fundamental concepts...

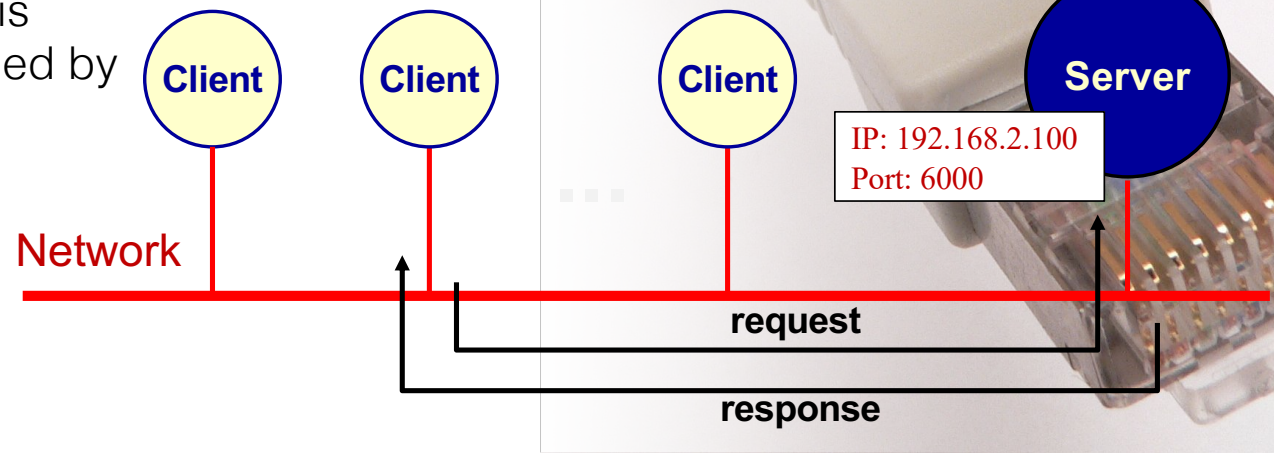
- ✓ Point-to-point communication:
  - Connection between only two devices, such as a wired serial connection (e.g. USB) or a wireless Bluetooth connection
  - Data is transmitted directly between the two devices without any intermediaries.
- ✓ Multipoint communication:
  - Connection of more than two devices, such as a wired Ethernet connection or a wireless Wi-Fi connection
  - Data is transmitted through a network or bus, allowing multiple devices to communicate between them simultaneously.
  - Models for exchanging information: client-server vs peer-to-peer (P2P)



UPV
COMPUTER ENG. DEPT. (DISCA)
Universitat Politècnica de València
{jposadas,jbenlloc}@disca.upv.es

Some fundamental concepts...

- ✓ The client-server model:
  - Clients request services or resources to a server
  - Servers provide the requested data or perform the requested tasks
  - Commonly used in web-based applications
  - TCP is the standard communications protocol used in the internet between clients and servers
    - Each device is identified by an IP address
    - Each server configures what is called a listening port, identified by a number
    - Clients connect with a server by specifying its IP address and port



UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UPV

COMPUTER ENG. DEPT. (DISCA)

Universitat Politècnica de València

{jposadas,jbenlloc}@disca.upv.es

Some fundamental concepts...

- ✓ The peer-to-peer (P2P) model:
  - Peers communicate between them by publishing/subscribing messages associated with topics
  - Any peer can publish information about any topic
  - Any peer can subscribe to any topic to automatically receive all the information published
  - MQTT is a communication protocol over TCP that allows the exchange of information according to the Publish/Subscribe model

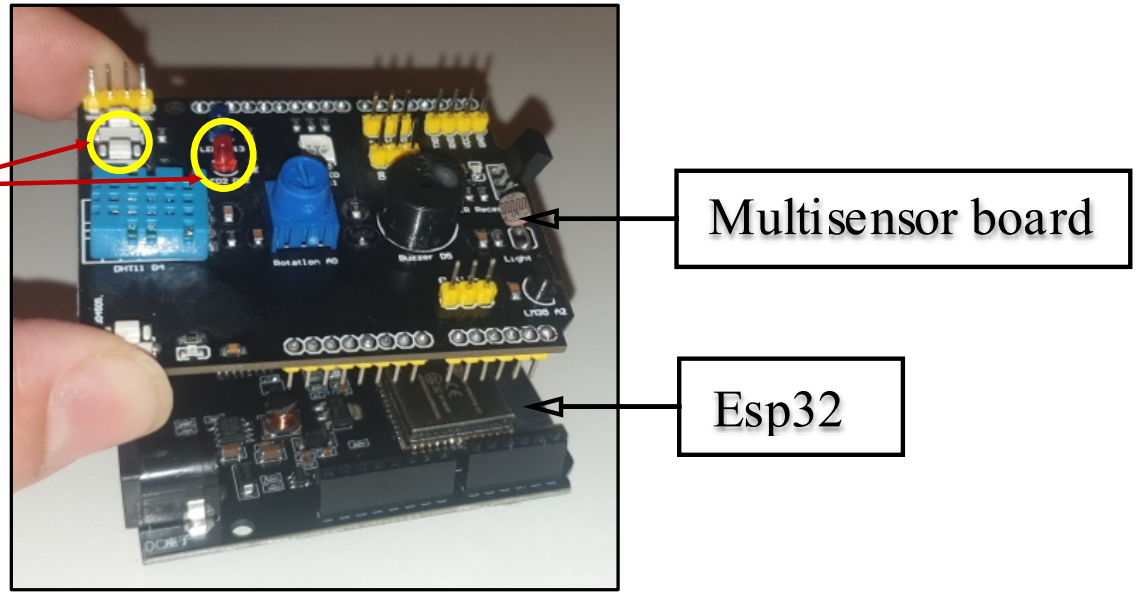
UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UPV

COMPUTER ENG. DEPT. (DISCA)
Universitat Politècnica de València
{jposadas,jbenlloc}@disca.upv.es

How are we going to test the different communications technologies?

- ✓ Starting from the circuit already tested that turns on an LED when a button is pressed
  - First webinar



UPV
COMPUTER ENG. DEPT. (DISCA)
Universitat Politècnica de València
{jposadas,jbenlloc}@disca.upv.es

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

How are we going to test the different communications technologies?

- ✓ Starting from the circuit already tested that turns on an LED when a button is pressed
  - First webinar

```
int pinLed = 19; //pin19(ESP32)=D12(board)
int pinButton = 25; //pin25(ESP32)=D3(board)
int buttonState;

void setup() {
  pinMode(pinLed, OUTPUT);
  digitalWrite(pinLed, LOW);
  pinMode(pinButton, INPUT);
}

void loop() {
  buttonState = digitalRead(pinButton);
  if (buttonState == 0) {
    digitalWrite(pinLed, HIGH);
  } else
    digitalWrite(pinLed, LOW);
  delay(100);
}
```

Important: when the button is pressed, digitalRead returns 0

file ***01.ESP32_ButtonAndLed.ino***

UPV
COMPUTER ENG. DEPT. (DISCA)
Universitat Politècnica de València
{jposadas,jbenlloc}@disca.upv.es

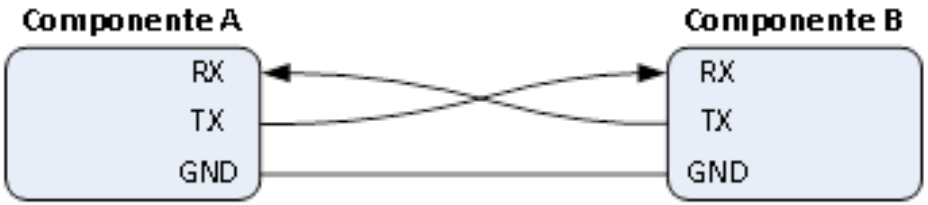
UNIVERSITAT POLITÈCNICA DE VALÈNCIA

How are we going to test the different communications technologies?

Now we will activate the LED of the ESP32
when we press the button of another different ESP32
or when we order it from the PC
or mobile phone
communicating in different ways.

UPV
COMPUTER ENG. DEPT. (DISCA)
Universitat Politècnica de València
{jposadas,jbenlloc}@disca.upv.es

Wired serial communications



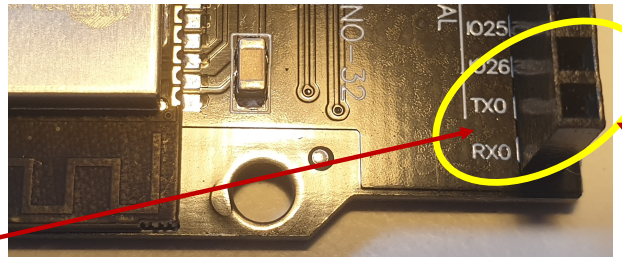
- ✓ We can easily establish a connection between two ESP32s or even between an ESP32 and a computer
- ✓ This is a great way to send and receive data in a simple and reliable way
- ✓ It involves using two wires, one for transmitting data (TX) and the other for receiving data (RX), in addition to a shared ground wire (GND)

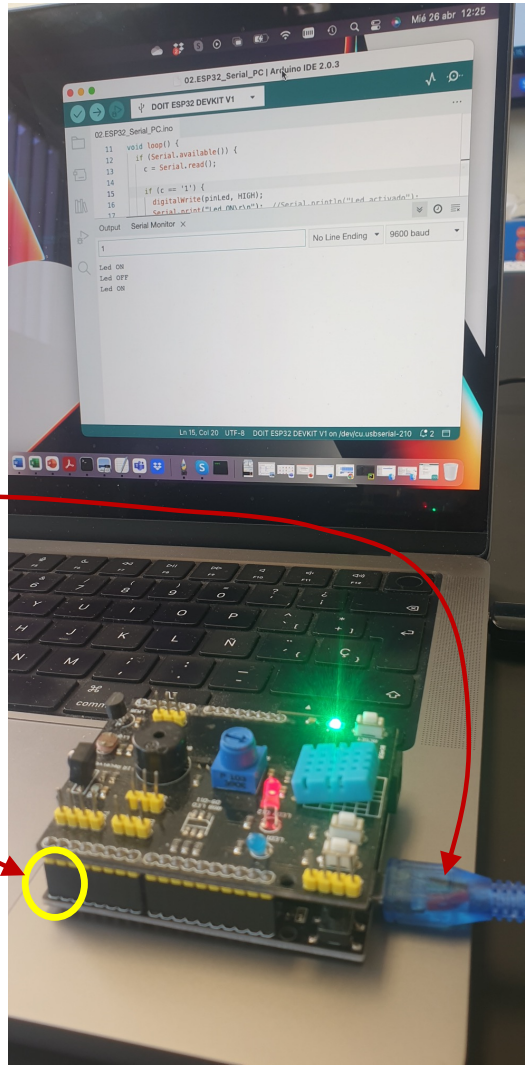
UPV
COMPUTER ENG. DEPT. (DISCA)
Universitat Politècnica de València
{jposadas,jbenlloc}@disca.upv.es

Wired serial communications

1. Serial communications between the ESP32 and the PC

- ✓ We have the "serial" library that provides functions to transmit and receive data over a physical serial connection made between a ESP32's serial port and a PC's USB port using a USB cable
  - *Serial.begin()*
  - *Serial.print()*
  - *Serial.read()*
- ✓ The "serial" library uses pins 0 and 1 internally for the USB connection



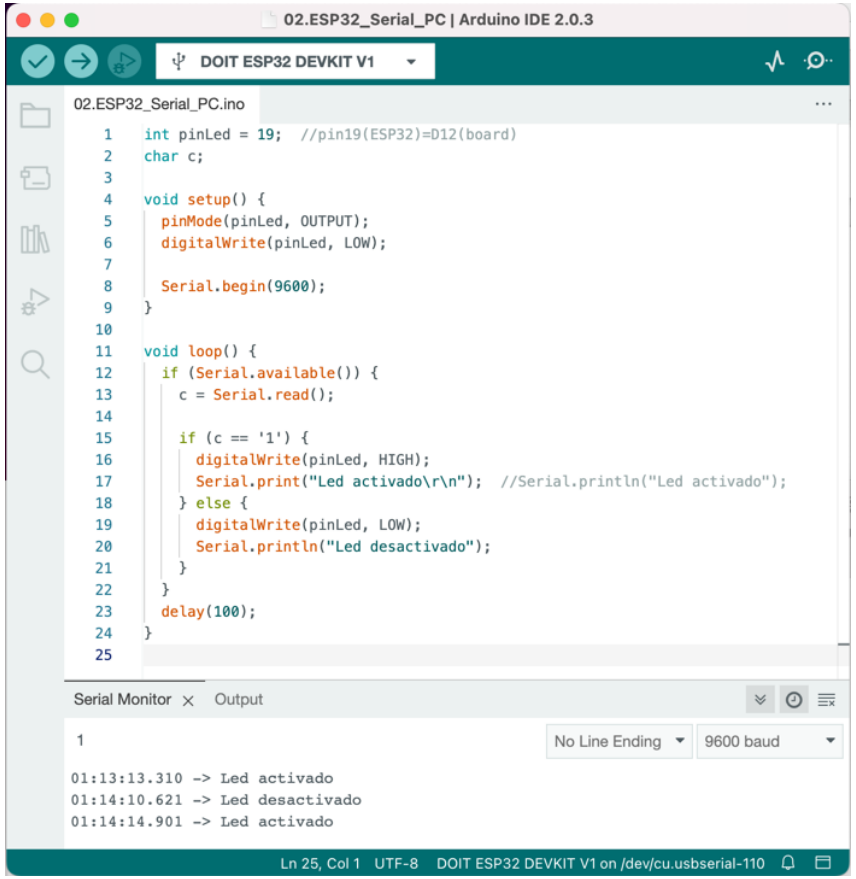


COMPUTER ENG. DEPT. (DISCA)
Universitat Politècnica de València
{jposadas,jbenlloc}@disca.upv.es

Wired serial communications

1. Serial communications between the ESP32 and the PC

- ✓ This code activates the LED when the ESP32 receives the value 1 from the keyboard instead of using a button, and deactivate the LED if it receives any other value
- ✓ It also displays the LED's status on the screen



file ***02.ESP32_Serial_PC.ino***

Important:

- Select: "**No Line Ending**" in order to send only one char (i.e. 1) without additional control chars (NL, CR)
- Select: the same baud rate (9600) configured with Serial.begin

UPV

COMPUTER ENG. DEPT. (DISCA)

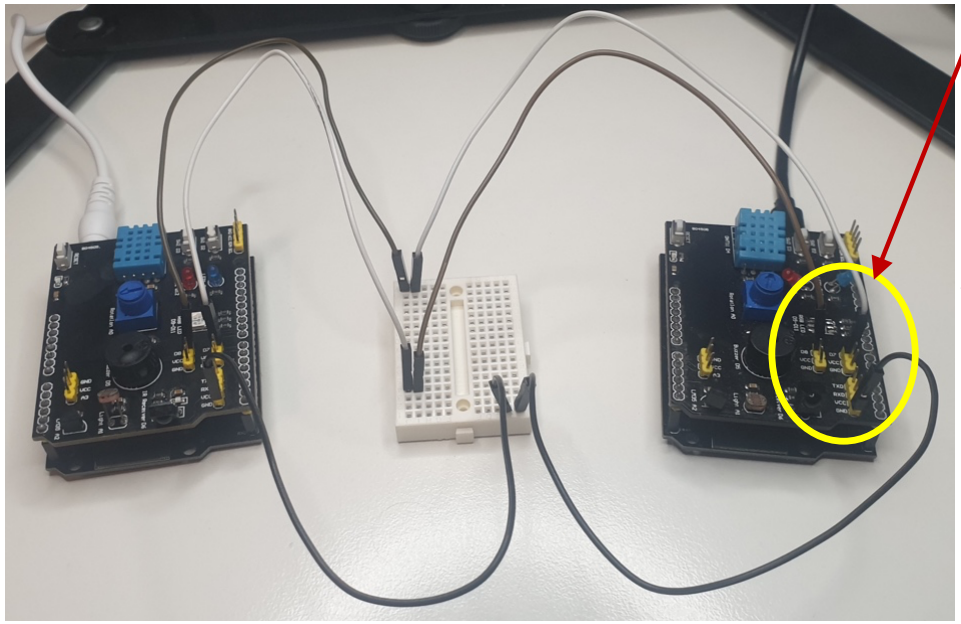
Universitat Politècnica de València

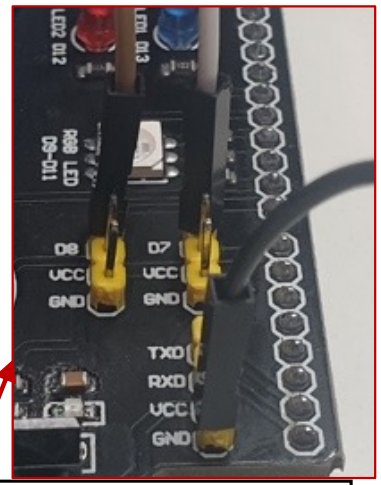
{jposadas,jbenlloc}@disca.upv.es

Wired serial communications

2. Serial communications between two ESP32 devices

- ✓ We can use the "Software Serial" library
- ✓ It allows you to create a virtual serial port on any pair of digital pins (different than 0 and 1) that can be used to communicate with other devices or other Arduino boards





Important:

- To share ground: both ESP32 have to be connected to the same ground. That is, it necessary to connect a wire between ground pins (black wire)
- Crossing wires: the brown wire (pin D8) to transmit from an ESP32 has to be connected to the white wire (pin D7) to receive from the other ESP32

UPV
COMPUTER ENG. DEPT. (DISCA)
Universitat Politècnica de València
{jposadas,jbenlloc}@disca.upv.es

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Wired serial communications

2. Serial communications between two ESP32 devices

```
//Sender
#include <SoftwareSerial.h>

SoftwareSerial Sender(14, 12);  //RX=pin14(ESP32)=D7(board)
                                //TX=pin12(ESP32)=D8(board)
int pinButton = 25;             //pin25(ESP32)=D3(board)
int buttonState;

void setup() {
  pinMode(pinButton, INPUT);
  Sender.begin(9600);
}

void loop() {
  buttonState = digitalRead(pinButton);
  if (buttonState == 0)
    Sender.write('1');
  else
    Sender.write('0');
  delay(100);
}
```

files ***03.ESP32_Serial_Receive_Button.ino*** and ***04.ESP32_Serial_Send_Button.ino***

```
// Receiver
#include <SoftwareSerial.h>

SoftwareSerial Receiver(14, 12);  //RX=pin14(ESP32)=D7(board)
                                  //TX=pin12(ESP32)=D8(board)
int pinLed = 19;                  //pin19(ESP32)=D12(board)
char c;

void setup() {
  pinMode(pinLed, OUTPUT);
  digitalWrite(pinLed, LOW);
  Serial.begin(9600);
  Receiver.begin(9600);
}

void loop() {
  if (Receiver.available()) {
    c = Receiver.read();
    if (c == '1') {
      digitalWrite(pinLed, HIGH);
      Serial.print("Led ON\r\n");  //Serial.println("Led ON");
    } else {
      digitalWrite(pinLed, LOW);
      Serial.println("Led OFF");
    }
  }
  delay(100);
}
```

UPV
COMPUTER ENG. DEPT. (DISCA)
Universitat Politècnica de València
{jposadas,jbenlloc}@disca.upv.es

Bluetooth communications

- ✓ Bluetooth communication takes things to the next level, allowing you to connect your ESP32 to other devices like smartphones and tablets.
- ✓ This opens up a whole new world of possibilities for remote control and monitoring of your projects.
- ✓ It allows for short-range data transmission between devices in a similar way to serial communication, but without the need for cables or physical connections
- ✓ "Bluetooth Serial" library
  - Slave Mode: it can be connected to by other Bluetooth devices
  - Master Mode: it can connect to other Bluetooth slaves
  - Once connected, ESP32 devices can send and receive data

UPV
COMPUTER ENG. DEPT. (DISCA)
Universitat Politècnica de València
{jposadas,jbenlloc}@disca.upv.es

Bluetooth communications

3. Bluetooth connections between two ESP32 devices

```
// Slave
#include "BluetoothSerial.h"

BluetoothSerial BT;
int pinLed = 19;  //pin19(ESP32)=D12(board)
char c;

void setup() {
  pinMode(pinLed, OUTPUT);
  digitalWrite(pinLed, LOW);
  BT.begin("My_ESP32_Bluetooth"); //Name and Slave configuration by default
  Serial.begin(9600);
  Serial.println("ESP32 Bluetooth is ready to pair");
}

void loop() {
  if (BT.available()) {
    c = BT.read();

    if (c == '1') {
      digitalWrite(pinLed, HIGH);
      Serial.print("Led ON\r\n");  //Serial.println("Led ON");
    } else {
      digitalWrite(pinLed, LOW);
      Serial.println("Led OFF");
    }
  }
  delay(100);
}
```

Important! Replace with your name

```
// Master
#include "BluetoothSerial.h"

BluetoothSerial BT;
String slaveName = "My_ESP32_Bluetooth";
int pinButton = 25;  //pin25(ESP32)=D3(board)
int buttonState;

void setup() {
  pinMode(pinButton, INPUT);

  BT.begin("ESP32_BT_Master", true);  //Name and Master configuration
  Serial.begin(9600);
  Serial.print("Connecting to Bluetooth slave ESP32...");

  while (!BT.connect(slaveName))
    Serial.print("...");
  Serial.println("...connected!");
}

void loop() {
  buttonState = digitalRead(pinButton);
  if (buttonState == 0)
    BT.write('1');
  else
    BT.write('0');
  delay(100);
}
```

Importat: ***true*** argument in begin function to master configuration

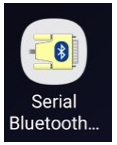
files ***05.ESP32_BT_Slave.ino*** and ***06.ESP32_BT_Master.ino***

UPV
COMPUTER ENG. DEPT. (DISCA)
Universitat Politècnica de València
{jposadas,jbenlloc}@disca.upv.es

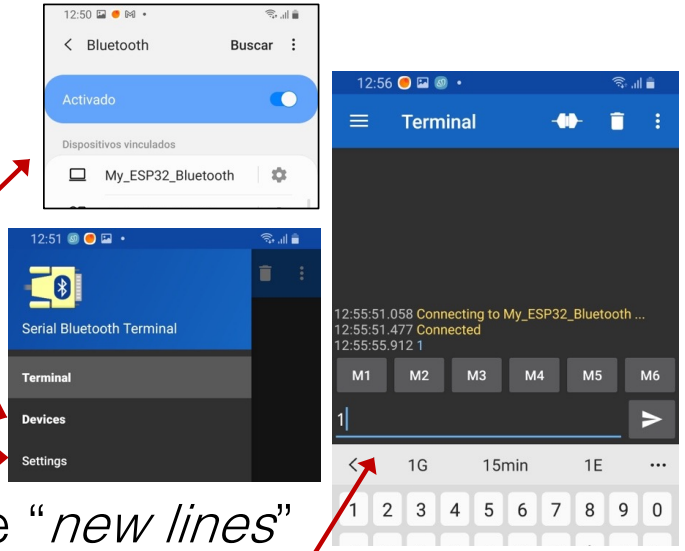
Bluetooth communications

4. Bluetooth connections between the ESP32 and a mobile phone

- ✓ We don't need to develop new code for this, as we will use the ESP32 with the compiled and uploaded slave code
- ✓ We will install a Serial connection program via Bluetooth on our mobile phone:
  - *Serial Bluetooth terminal*



- ✓ Pair the Slave ESP32 with your smartphone
- ✓ Configure the application:
  - "*Devices*" menu: connect to the Slave ESP32
  - "Settings" menu: configure the way to send **text data without** adding **control chars** to indicate "*new lines*"

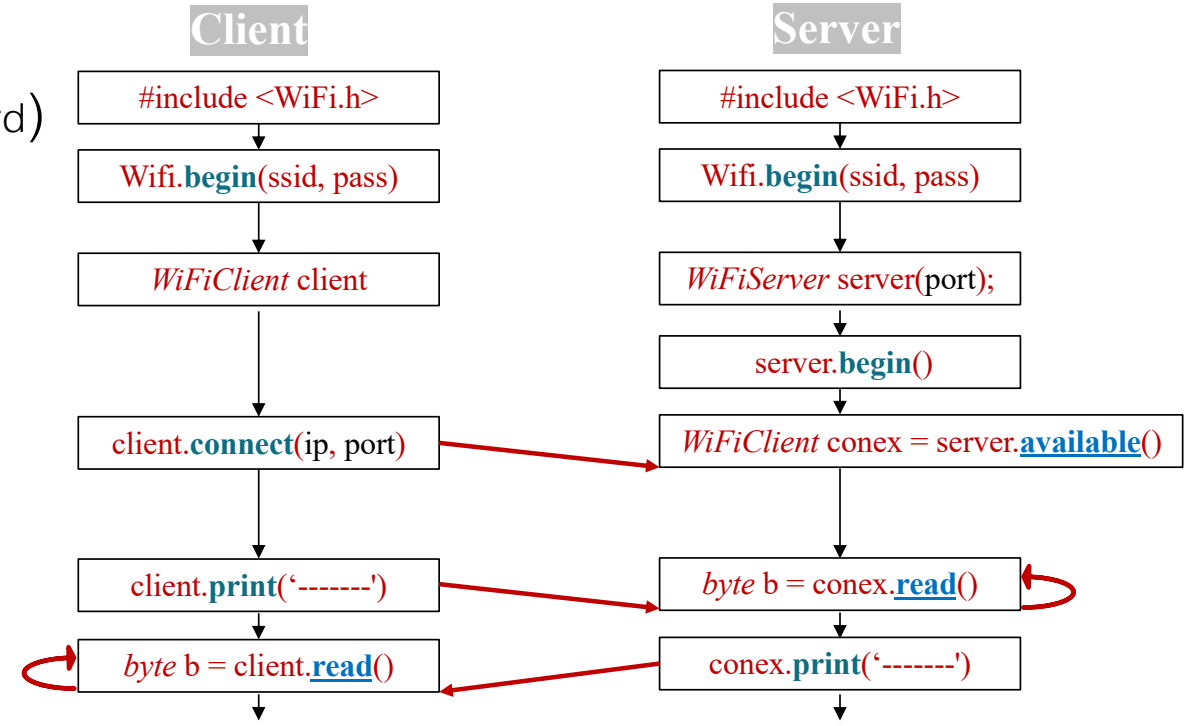


we can send messages (i.e.: characters '1' or '0') to the ESP32 to turn on or turn off the LED

UPV
COMPUTER ENG. DEPT. (DISCA)
Universitat Politècnica de València
{jposadas,jbenlloc}@disca.upv.es

Wi-Fi Communications

- ✓ With Wi-Fi, you can connect your ESP32 to the internet and create web-based applications, enabling you to control your project from anywhere in the world
- ✓ "Wifi" library
  - WiFi.begin(router_SSID, password)
  - WiFiClient
  - WiFiServer



UPV
COMPUTER ENG. DEPT. (DISCA)
Universitat Politècnica de València
{jposadas,jbenlloc}@disca.upv.es

Wi-Fi Communications

5. Wi-Fi connections between two ESP32 devices

07.ESP32_TCP_Server.ino

```
// TCP Server
#include <WiFi.h>

const char *ssid = "your_router_name";      // wifi router name
const char *pass = "your_router_password";  // wifi router password

WiFiServer server(6000);

int pinLed = 19;  //pin19(ESP32)=D12(board)
char c;

void setup() {
  pinMode(pinLed, OUTPUT);
  digitalWrite(pinLed, LOW);
  Serial.begin(115200);                    // initialize serial communication
  WiFi.mode(WIFI_STA);                     // set mode to wifi station
  WiFi.begin(ssid, pass);                  // connect to wifi router
  while (WiFi.status() != WL_CONNECTED) {  // check status of connection
    delay(500);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());  //print the Ip allocated by router
  delay(500);

  server.begin();
}
```

Important: Wifi.localIP() returns the current server IP allocated by router. We have to **update the client code with this value**.

```
void loop() {
  WiFiClient client = server.available();
  if (client) {
    c = client.read();
    if (c == '1') {
      digitalWrite(pinLed, HIGH);
      Serial.print("Led ON\r\n");  //Serial.println("Led ON");
    } else {
      digitalWrite(pinLed, LOW);
      Serial.println("Led OFF");
    }
    client.stop();
  }
  delay(100);
}
```

UPV
COMPUTER ENG. DEPT. (DISCA)
Universitat Politècnica de València
{jposadas,jbenlloc}@disca.upv.es

Wi-Fi Communications

5. Wi-Fi connections between two ESP32 devices

```
// TCP Client
#include <WiFi.h>

const char *ssid = "your_router_name";      // wifi router name
const char *pass = "your_router_password";  // wifi router password
const char *host = "192.168.1.143";     // server IP Address
const int port = 6000;                  // server port
WiFiClient client;
int pinButton = 25;  //pin25(ESP32)=D3(board)
int buttonState;

void setup() {
  pinMode(pinButton, INPUT);
  Serial.begin(115200);  // initialize serial communication
  WiFi.mode(WIFI_STA);                    // set mode to wifi station
  WiFi.begin(ssid, pass);                 // connect to wifi router
  while (WiFi.status() != WL_CONNECTED) {  // check status of connection
    delay(500);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");
  delay(500);
}
```

It is necessary to update with the real server IP and port

08.ESP32_TCP_Client.ino

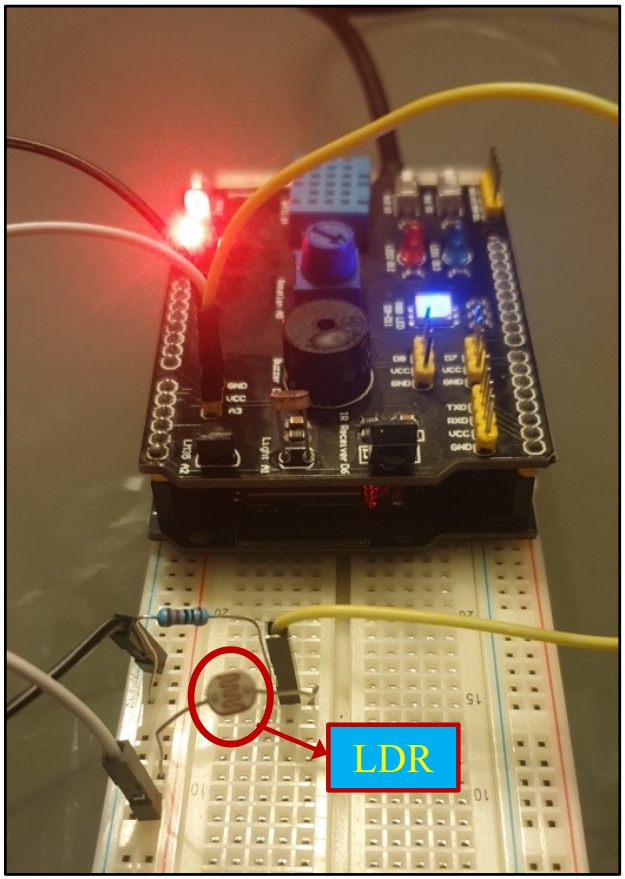
```
void loop() {
  if (client.connect(host, port)) {  // connect to server
    buttonState = digitalRead(pinButton);
    if (buttonState == 0)
      client.print('1');  //send data
    else
      client.print('0');  //send data
    client.stop();
  } else
    Serial.println("Error to connect to server");

  delay(100);
}
```

UPV
COMPUTER ENG. DEPT. (DISCA)
Universitat Politècnica de València
{jposadas,jbenlloc}@disca.upv.es

Wi-Fi Communications

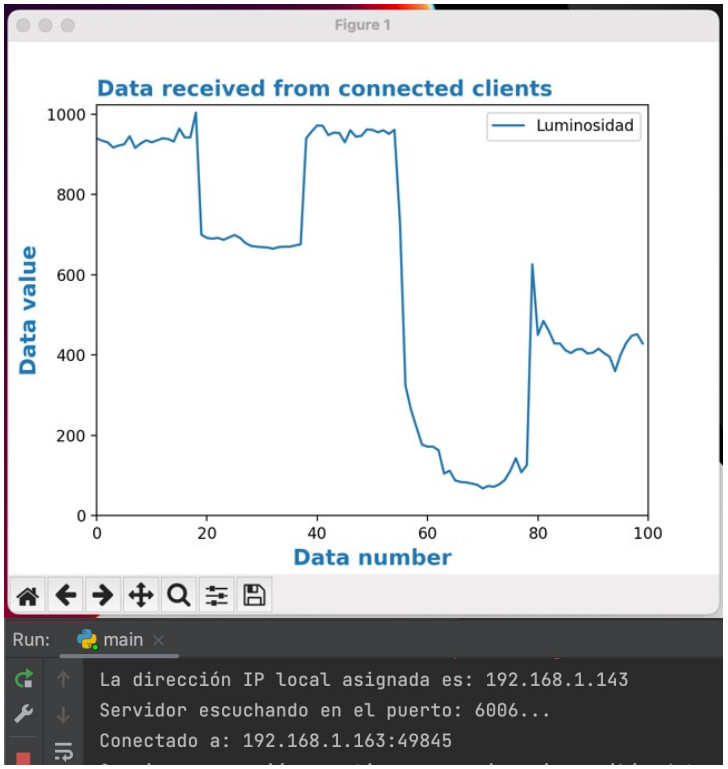
Optional advanced exercises



file ***09.ESP32_Extended_TCP_Client.ino***

6. Connecting to server

- ✓ ESP32 sends periodically the light level to a python server
- ✓ The server displays the data it in a real-time graph
- ✓ Data in JSON format



file ***TCPServer.py***

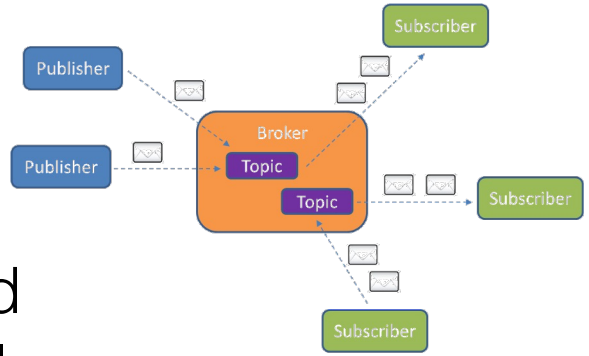
UPV
COMPUTER ENG. DEPT. (DISCA)
Universitat Politècnica de València
{jposadas,jbenlloc}@disca.upv.es

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Wi-Fi Communications

Optional advanced exercises

7. Working with MQTT



- ✓ MQTT is widely used in the Internet of Things (IoT)
- ✓ It is a lightweight messaging protocol over TCP
- ✓ It allows devices with limited processing power and bandwidth to communicate between them efficiently
- ✓ It allows the exchange of information according to the model called "Publish/Subscribe" based on topics
- ✓ It is necessary all devices called MQTT clients connect with a central broker or message-oriented middleware (MOM)
- ✓ An open-source implementation of the MQTT broker is Mosquitto

UPV
COMPUTER ENG. DEPT. (DISCA)
Universitat Politècnica de València
{jposadas,jbenlloc}@disca.upv.es

Wi-Fi Communications

Optional advanced exercises

7. Working with MQTT

file *11.ESP32_MQTT_Client.ino*

- ✓ "PubSubClient" library
  - Allows defining an MQTT publisher-subscriber client
- ✓ We will configure the ESP32 as an MQTT client to periodically publish the values of its sensors on the internet
- ✓ It will also be able to receive messages/commands sent from other MQTT clients
- ✓ We will connect with the external server/broker or MOM "*test.mosquitto.org*"
- ✓ The python MQTT client (*Python_MQTT_Client.py*) will display the data published by the ESP32

UPV
COMPUTER ENG. DEPT. (DISCA)
Universitat Politècnica de València
{jposadas,jbenlloc}@disca.upv.es

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

So whether you're building a robot, monitoring environmental conditions, or creating a home automation system, the ESP32's communication capabilities will make your project come alive

I encourage you to explore the power of ESP32 communications and see what amazing things you can create!

UPV

COMPUTER ENG. DEPT. (DISCA)

Universitat Politècnica de València

{jposadas,jbenlloc}@disca.upv.es

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

THANK YOU

UPV

COMPUTER ENG. DEPT. (DISCA)

Universitat Politècnica de València

{jposadas,jbenlloc}@disca.upv.es