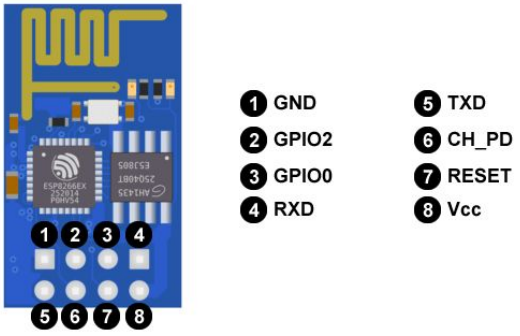
Universitat de Lleida

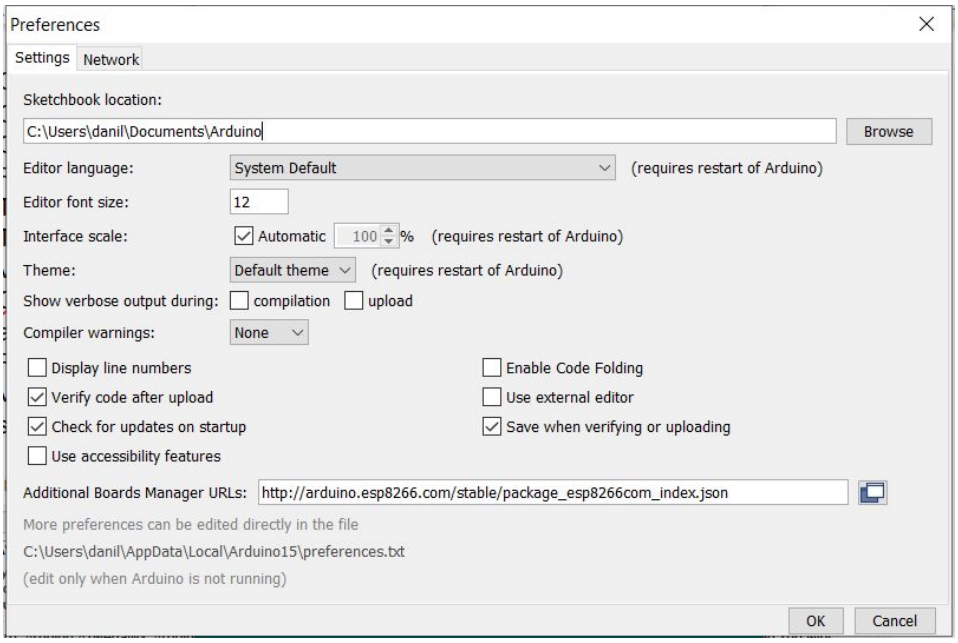
Arduino and ESP8266 integration (Data Receiver)
Embedded and Ubiquitous systems
Task part of the second sprint
UDL - MINF - 2021 - Team 1

Content:



1. GND = ground
2. GPIO2 general purpose I/O. Its digital pin 2.
3. GPIO0 general purpose I/O. Its digital pin 0.
4. RXD es el pin por donde se van a recibir los datos del puerto serie. Trabaja a 3,3 V. También se puede utilizar como pin digital GPIO: sería el número 3.
5. TXD es el pin por donde se van a transmitir los datos del puerto serie. Trabaja a 3,3 V. También se puede utilizar como pin digital GPIO: sería el número 1.
6. CH_PD pin para apagar y encender el ESP-01: si lo ponemos a 0 V (LOW) se apaga, y a 3,3 V (HIGH) se enciende.
7. **RESET** pin to reset ESP-01: if receives 0 V (LOW) it resets.
8. Vcc is where we power the ESP-01. Works in 3,3 V allows a maximum of 3,6 V. The supplied power must be greater than 200 mA.

Installing ESP-01 libraries on Arduino IDE:



http://arduino.esp8266.com/stable/package_esp8266com_index.json

Programming:

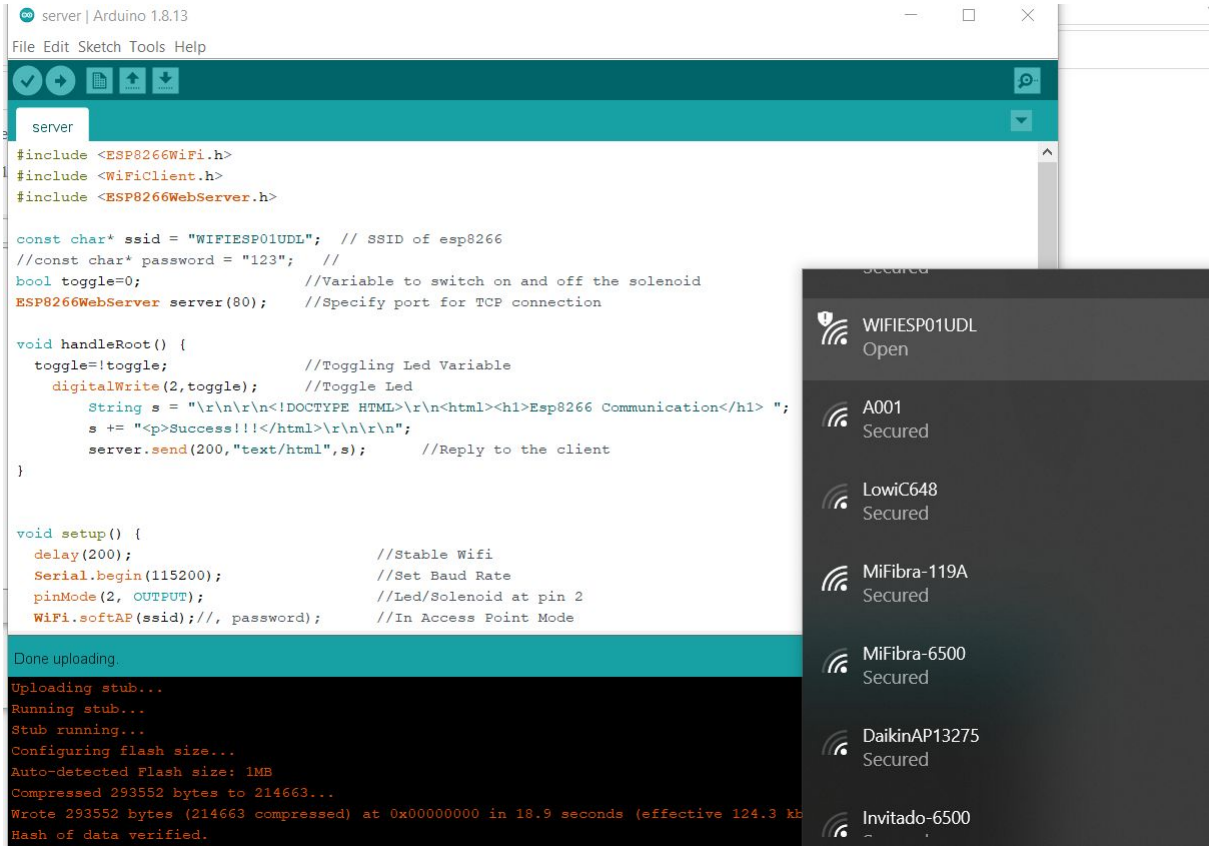
Since we have the USB to ESP-01 adapter (red board with USB), we can easily use that to program the ESP-01.

The idea here is to program (flash) every ESP-01 using this method, and then move it to the final implementation of the project

1. On the adapter, switch to the PROG mode (the other is UART)
2. Connect the adapter on computer and install its drivers:
   USB Adapter Driver: http://www.wch.cn/download/CH341SER_EXE.html
3. After that, you can connect your ESP-01 on it and plug on the computer
4. In the Arduino IDE go Tools > Board > ESP8266 boards and then select Generic ESP8266 Module.
5. Tools > Port > Select the port the USB is in (check in the devices page of windows)
6. Now you can Verify and then compile your code.
7. Here you have a simple code for blinking the led (blue) of the ESP8266:
   a. https://learn.sparkfun.com/tutorials/esp8266-thing-hookup-guide/example-sketch-blink

ESP-01 + DHT11: https://www.youtube.com/watch?v=of_g89PQQqU

Testing the Arduino as a Acess Point:



Setting it up to work as data receiver

1. The ESP-01 will communicate with the Arduino using serial connection, in this case, we are gonna to emulate this connection using the SoftwareSerial library.
2. That said, let's talk about the code:
    a. Arduino code (will need some minor changes for the final project):

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(2, 3); // RX, TX

// Variables to handle the data from the ESP
const byte numChars = 32;
char receivedChars[numChars];
char tempChars[numChars];     // temporary array for use when parsing

// variables to hold the parsed data
char message[numChars] = {0};
float floatTemp = 0.0;
float floatHum = 0.0;

boolean newData = false;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  mySerial.begin(115200);
  delay(5000);
}

// Enter data in this style <HelloWorld, 12, 24.7>

void loop() {
       recvWithStartEndMarkers();
       if (newData == true) {
       strcpy(tempChars, receivedChars);
              // this temporary copy is necessary to protect the original data
              //   because strtok() used in parseData() replaces the commas with \0
       parseData();
       showParsedData();
       newData = false;
       }
}

//============

void recvWithStartEndMarkers() {
       static boolean recvInProgress = false;
       static byte ndx = 0;
       char startMarker = '<';
       char endMarker = '>';
       char rc;

       while (mySerial.available() > 0 && newData == false) {
       rc = mySerial.read();
```

```
        if (recvInProgress == true) {
                if (rc != endMarker) {
                receivedChars[ndx] = rc;
                ndx++;
                if (ndx >= numChars) {
                        ndx = numChars - 1;
                }
                }
                else {
                receivedChars[ndx] = '\0'; // terminate the string
                recvInProgress = false;
                ndx = 0;
                newData = true;
                }
        }

        else if (rc == startMarker) {
                recvInProgress = true;
        }
        }
}

//============

void parseData() {     // split the data into its parts

        char * strtokIndx; // this is used by strtok() as an index

        strtokIndx = strtok(tempChars,",");   // get the first part - the string
        strcpy(message, strtokIndx); // copy it to messageFromPC

        strtokIndx = strtok(NULL, ","); // this continues where the previous call left off
        floatTemp = atof(strtokIndx); // convert this part to a float

        strtokIndx = strtok(NULL, ",");
        floatHum = atof(strtokIndx);  // convert this part to a float

}

//============

void showParsedData() {
        Serial.print("Message: ");
        Serial.println(message);
        Serial.print("Temperature: ");
        Serial.println(floatTemp);
        Serial.print("Humidity: ");
        Serial.println(floatHum);
}
```

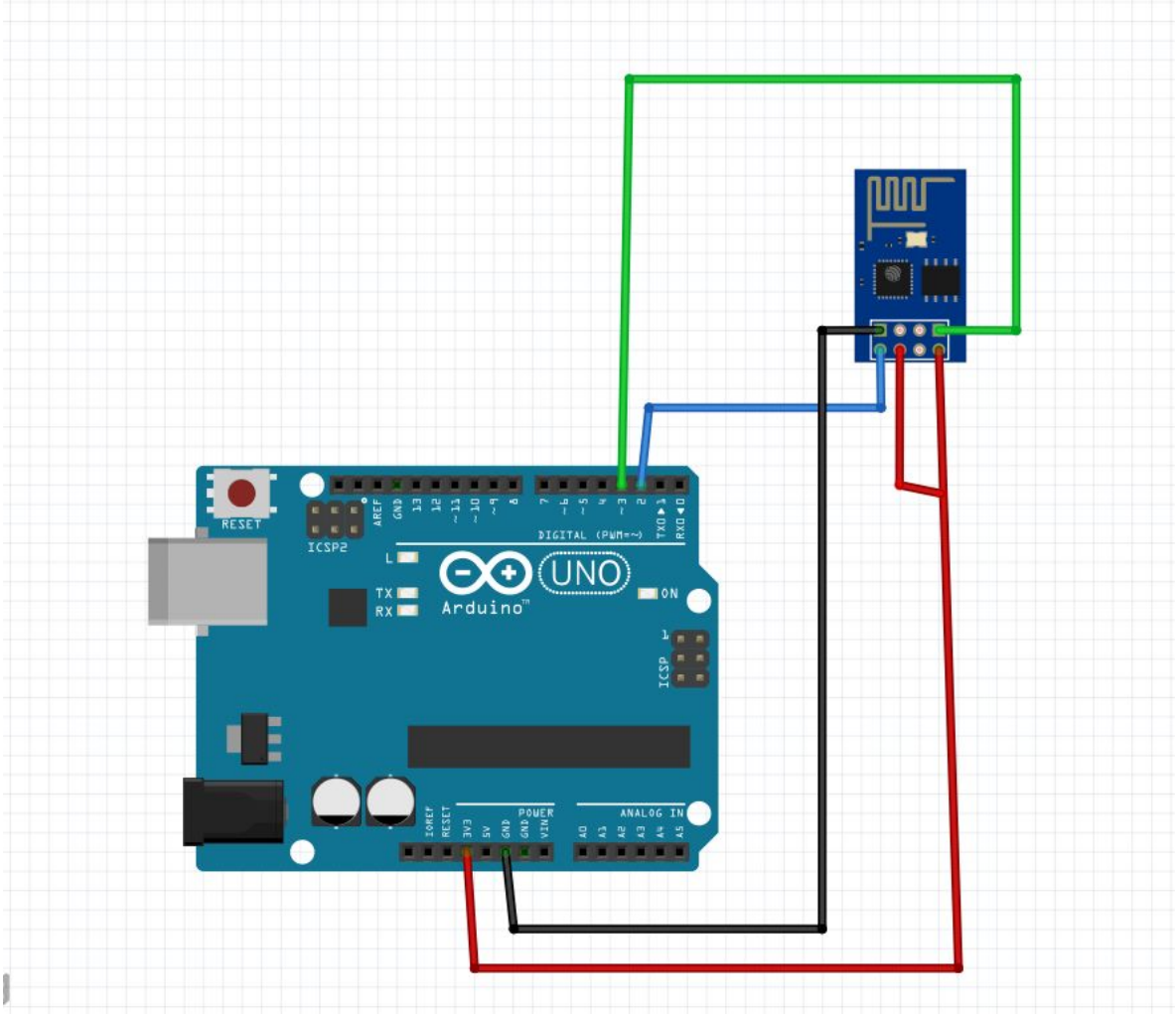
b. ESP-01 code (also pending of minor changes):

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

// Structure example to receive data
// Must match the sender structure
typedef struct struct_message {
        int id;
        float x;
        float y;
```

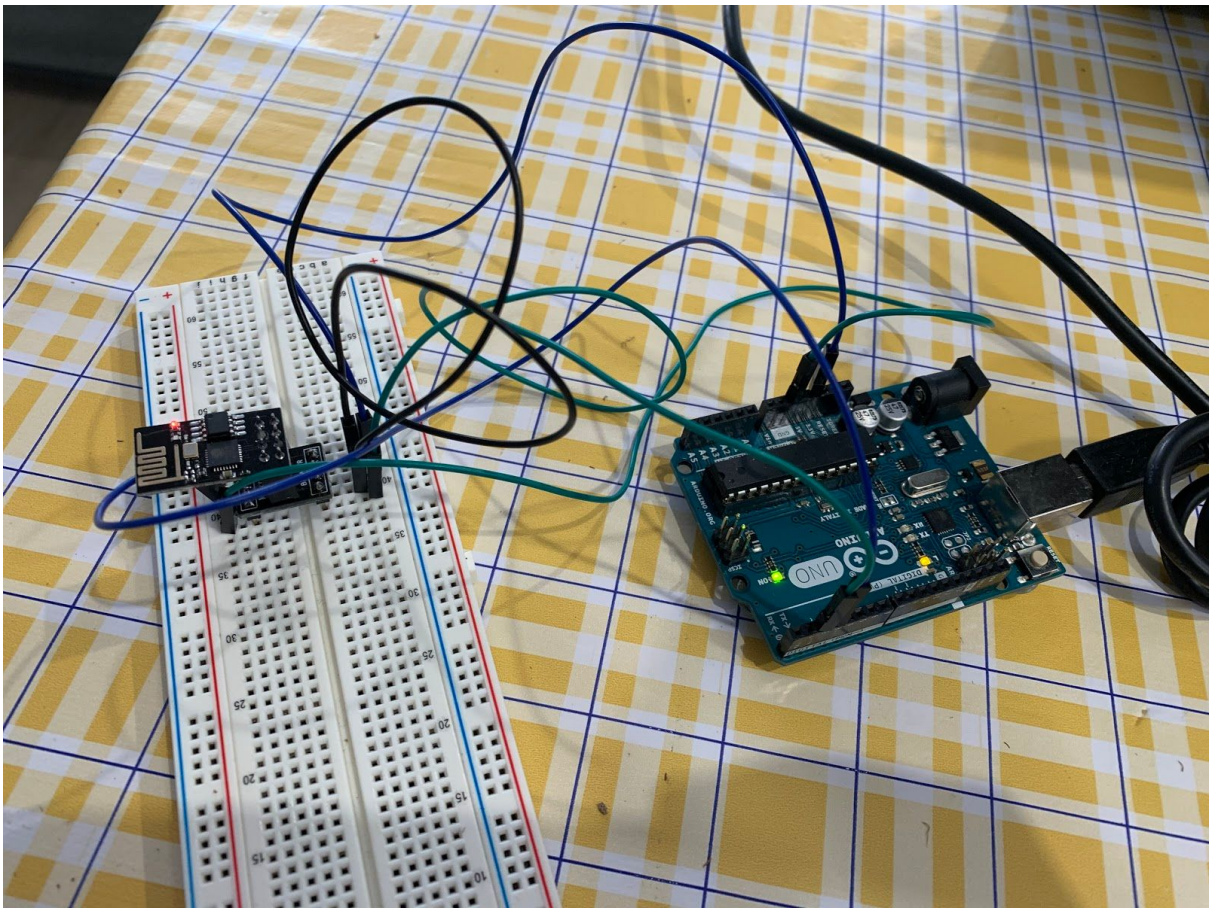
```
} struct_message;

// Create a struct_message called myData
struct_message myData;

// Create a structure to hold the readings from each board
struct_message board1;
struct_message board2;

// Create an array with all the structures
struct_message boardsStruct[2] = {board1, board2};

unsigned long lastTime = 0;
unsigned long timerDelay = 10000;

float board1Distance = 0.0;
float board2Temp = 0.0;
float board2Hum = 0.0;

// Callback function that will be executed when data is received
void OnDataRecv(uint8_t * mac_addr, uint8_t *incomingData, uint8_t len) {
  char macStr[18];
  //Serial.print("Packet received from: ");
  snprintf(macStr, sizeof(macStr), "%02x:%02x:%02x:%02x:%02x:%02x",
        mac_addr[0], mac_addr[1], mac_addr[2], mac_addr[3], mac_addr[4], mac_addr[5]);
  //Serial.println(macStr);
  memcpy(&myData, incomingData, sizeof(myData));
  //Serial.printf("Board ID %u: %u bytes\n", myData.id, len);
  // Update the structures with the new incoming data
  boardsStruct[myData.id-1].x = myData.x;
  boardsStruct[myData.id-1].y = myData.y;
  boardsStruct[myData.id-1].id = myData.id;
  //Serial.printf("x value: %f \n", boardsStruct[myData.id-1].x);
  //Serial.printf("y value: %f \n", boardsStruct[myData.id-1].y);
  //Serial.println();
}

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

  // Set device as a Wi-Fi Station
  WiFi.mode(WIFI_STA);
  WiFi.disconnect();

  // Init ESP-NOW
  if (esp_now_init() != 0) {
        Serial.println("Error initializing ESP-NOW");
        return;
  }

  // Once ESPNow is successfully Init, we will register for recv CB to
  // get recv packer info
  esp_now_set_self_role(ESP_NOW_ROLE_SLAVE);
  esp_now_register_recv_cb(OnDataRecv);
}

void loop(){
   // loop
  if ((millis() - lastTime) > timerDelay) {
        // Access the variables for each board

        board1Distance = boardsStruct[0].x;
        float test = boardsStruct[0].y;
```

```
        board2Temp = boardsStruct[1].x;
        board2Hum = boardsStruct[1].y;



        // Send the information to the arduino.
        Serial.print("<DataProducers,");
        Serial.print(board2Temp);
        Serial.print(",");
        Serial.print(board2Hum);
        Serial.print(">");
        //
        Serial.print(board1Distance);
        Serial.print(",");
        Serial.print(test);
        Serial.print("\n");

        lastTime = millis();
  }
}
```

3. Electrical connections:



4. Building the setup:



5. Testing
    a. In order to test it, we need to implement at least another ESP-01 as data sender using the ESPNow, in this case, the test was made using the Dataproducer 2(DHT11 sensor) providing the information via ESPNow.

```
COM3
:31:50.266 -> Humidity: 39.00
:32:00.274 -> Message: DataProducer1
:32:00.274 -> Temperature: 20.00
:32:00.274 -> Humidity: 39.00
:32:10.276 -> Message: DataProducer1
:32:10.276 -> Temperature: 19.00
:32:10.276 -> Humidity: 38.00
:32:20.250 -> Message: DataProducer1
:32:20.250 -> Temperature: 20.00
:32:20.250 -> Humidity: 38.00
:32:30.272 -> Message: DataProducer1
:32:30.272 -> Temperature: 20.00
:32:30.272 -> Humidity: 38.00
:32:40.268 -> Message: DataProducer1
:32:40.268 -> Temperature: 20.00
:32:40.268 -> Humidity: 38.00
:32:50.264 -> Message: DataProducer1
:32:50.264 -> Temperature: 20.00
:32:50.264 -> Humidity: 39.00
:33:00.274 -> Message: DataProducer1
:33:00.274 -> Temperature: 20.00
:33:00.274 -> Humidity: 39.00
:33:10.251 -> Message: DataProducer1
:33:10.251 -> Temperature: 20.00
:33:10.285 -> Humidity: 39.00
:33:20.271 -> Message: DataProducer1
:33:20.271 -> Temperature: 20.00
:33:20.271 -> Humidity: 39.00
Autoscroll  Show timestamp
```

```
ESP_NOW_Receiver_modified_ARDUINO.ino

//============

void parseData() {      // split the data into its parts

    char * strtokIndx; // this is used by strtok() as an index

    strtokIndx = strtok(tempChars,",");      // get the first part - the string
    strcpy(message, strtokIndx); // copy it to messageFromPC

    strtokIndx = strtok(NULL, ","); // this continues where the previous call left off
    floatTemp = atof(strtokIndx);     // convert this part to a float

    strtokIndx = strtok(NULL, ",");
    floatHum = atof(strtokIndx);     // convert this part to a float

}

//============

void showParsedData() {
    Serial.print("Message: ");
    Serial.println(message);
    Serial.print("Temperature: ");
    Serial.println(floatTemp);
    Serial.print("Humidity: ");
    Serial.println(floatHum);
}
```