Data Producer 1 development
Embedded and Ubiquitous systems
Task part of the second sprint
UDL - MINF - 2021 - Team 1

Reference guide: https://randomnerdtutorials.com/esp-now-many-to-one-esp8266-nodemcu/

1. The purpose of this document is to explain the process in developing the data producer 2.
   For this project we are using a library called ESPNOW that facilitates the communication between multiple ESP-01. In this case, we are gonna use a system of 1 receiver / multiple senders, better said: ESP-01 on Arduino is the receiver, and the 2 ESP-01 of the data producers are senders.
   a. That said, we have this setup:
   b. Arduino-ESP-01 slave (Receiver)
   c. Data producer 1 ESP-01 master (ESP+Ultrasonic sensor) Board #1 (Sender)
   d. Data producer 2 ESP-01 master (ESP+Temperature sensor) Board #2 (Sender)

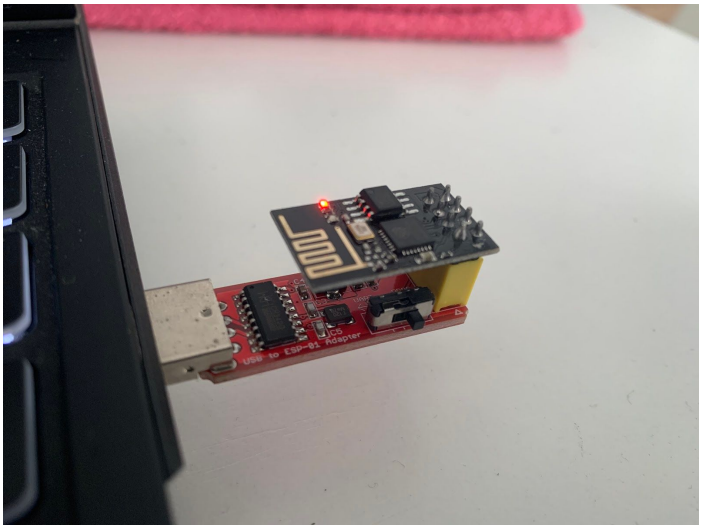
2. It's important to know in which ESP you are working, to adjust the code. So let's start on the Data Producer 2.
3. Open the Arduino IDE, select the board to Generic ESP 8266 Module and the correct port.
   The first step is to get the MAC Address of the receiver ESP-01, which means the ESP-01 that you are gonna use in the main part of the project (remember, the receiver), you need to execute this code on it to discover its Mac Address (See step 4 of how connect the ESP-01 to the computer to compile on it):

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

void setup(){
 Serial.begin(115200);
 Serial.println();
 Serial.print("ESP Board MAC Address:  ");

}

void loop(){
 Serial.println(WiFi.macAddress());
}
```

   Don't forget to disconnect the Adapter after compiling, change the switch to UART, and insert it again! Then your program should run.

Open the COM monitor on the Baud rate 115200 to see your mac address, and store it somewhere.
This part is essential because the library needs the mac address of the receiver to send data.

4. With this MAC Address in hand you can start programming the code into the ESP-01 of the Data Producer 1, for doing this we are using the ESP to USB adapter in the Switch in PROG:



5. Open the Arduino IDE, select the board to Generic ESP 8266 Module and to the correct port.
6. In the following code you will need to change this parameter with the MAC Address that you got from your receiver ESP-01:

```
// REPLACE WITH RECEIVER MAC Address (ESP-01 that is connected with the Arduino)
uint8_t broadcastAddress[] = {0xA4, 0xCF, 0x12, 0xBF, 0x15, 0xEE};
```

7. Code (also avaliable on Github):

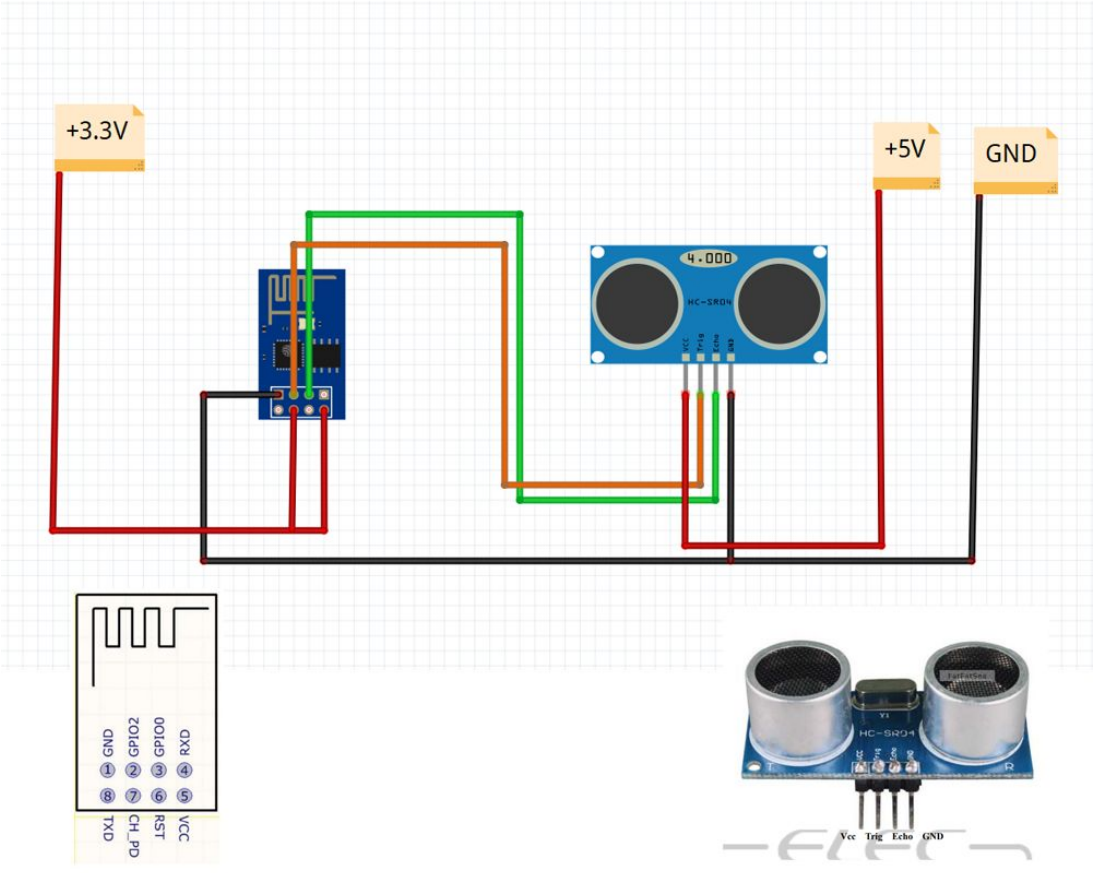
```
#include <ESP8266WiFi.h>
#include <espnow.h>

// REPLACE WITH RECEIVER MAC Address (ESP-01 thats connected with the Arduino)
uint8_t broadcastAddress[] = {0xA4, 0xCF, 0x12, 0xBF, 0x15, 0xEE};

// pins of the esp8266 connected to the sensor
#define trigPin 2  //GPIO2
#define echoPin 0  //GPIO0

// intialize the variables for storing the measurement
float timeElapsed;
float distance;

// Set your Board ID (ESP32 Sender #1 = BOARD_ID 1, ESP32 Sender #2 = BOARD_ID 2, etc)
#define BOARD_ID 1

// Structure to send data
```

```
// Must match the receiver structure
typedef struct struct_message {
    int id;
    float x;
    float y;
} struct_message;

// Create a struct_message called test to store variables to be sent
struct_message myData;

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

// Callback when data is sent
void OnDataSent(uint8_t *mac_addr, uint8_t sendStatus) {
 Serial.print("\r\nLast Packet Send Status: ");
 if (sendStatus == 0){
    Serial.println("Delivery success");
 }
 else{
    Serial.println("Delivery fail");
 }
}

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

 pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
 pinMode(echoPin, INPUT); // Sets the echoPin as an Input

 // 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;
 }
 // Set ESP-NOW role
 esp_now_set_self_role(ESP_NOW_ROLE_CONTROLLER);

 // Once ESPNow is successfully init, we will register for Send CB to
 // get the status of Trasnmitted packet
 esp_now_register_send_cb(OnDataSent);

 // Register peer
 esp_now_add_peer(broadcastAddress, ESP_NOW_ROLE_SLAVE, 1, NULL, 0);

}

void loop() {
 // do the measurements of the sensor
  digitalWrite(trigPin, LOW);  //para generar un pulso limpio ponemos a LOW 4us
  delayMicroseconds(4);

  digitalWrite(trigPin, HIGH);  //generamos Trigger (disparo) de 10us
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
```

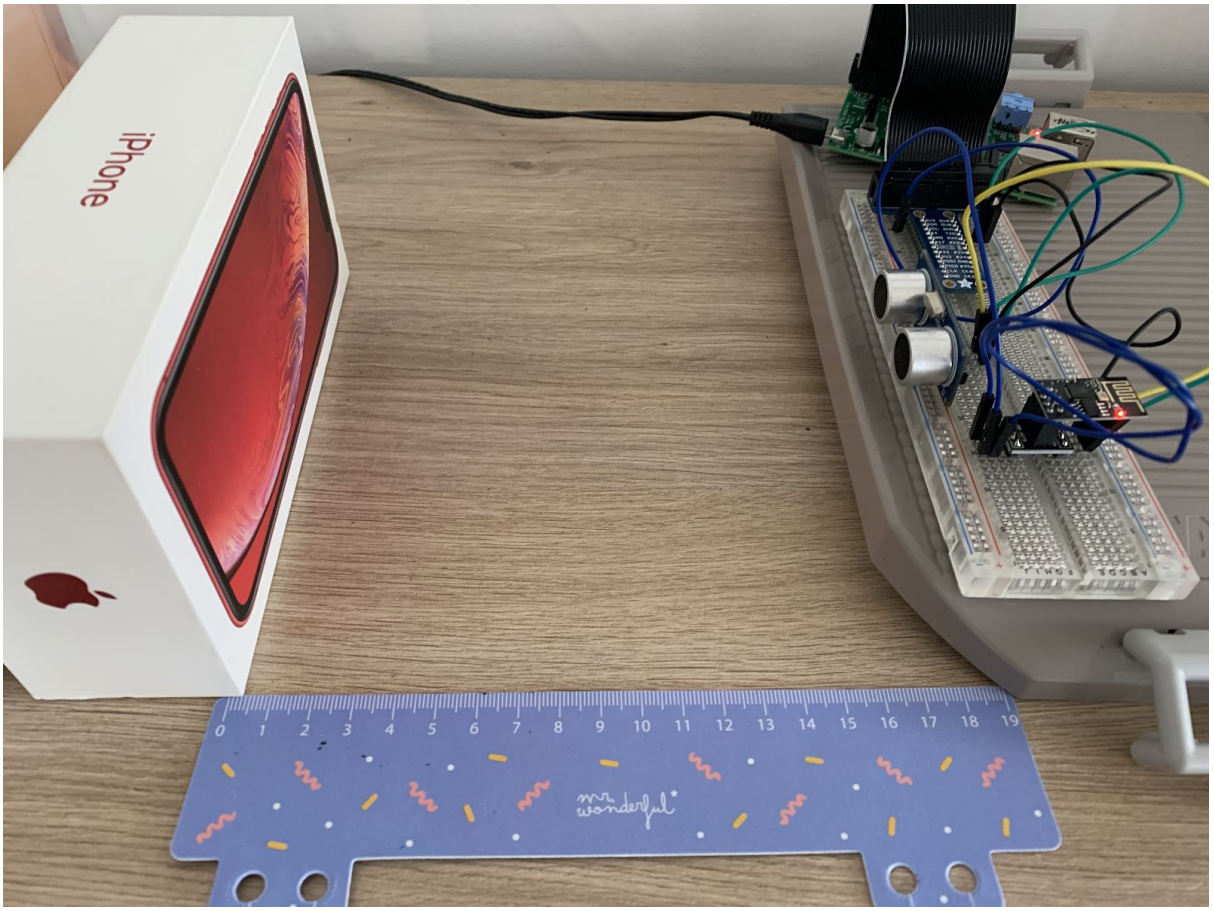
```
  timeElapsed = pulseIn(echoPin, HIGH);
  distance = timeElapsed/58.3;

  delay(1000);

 if ((millis() - lastTime) > timerDelay) {
              // Set values to send
              myData.id = BOARD_ID;
              myData.x = distance;
              myData.y = timeElapsed;

              // Send message via ESP-NOW
              esp_now_send(0, (uint8_t *) &myData, sizeof(myData));
              lastTime = millis();
  }
}
```

Now the code for your ESP-01 is done, the next step is to build the electrical connections with the sensor and power supply.

8. Connections



It should look like this (in this case i am using the raspberry pi as power supplly (5V, 3.3V and GND):



Unfortunately, in order to test it you will need to also setup the Receiver ESP-01 using the ESP-Now code to receive the measurements.