Technical document - Transmission protocol - Final delivery

Ubiquitous and Embedded systems

UDL MINF 2021

Team 1

Document structure:

1. Data Producer 1 transmission protocol
2. Data Producer 2 transmission protocol
3. Arduino transmission protocol
4. Receiver ESP-01 transmission protocol
5. ChibiOS Raspberry Pi transmission protocol

1. Data Producer 1 transmission protocol

a. ESP-01 considerations:

i. We make the communications between the ESP's using the library espnow. It registers 2 senders and 1 receiver.

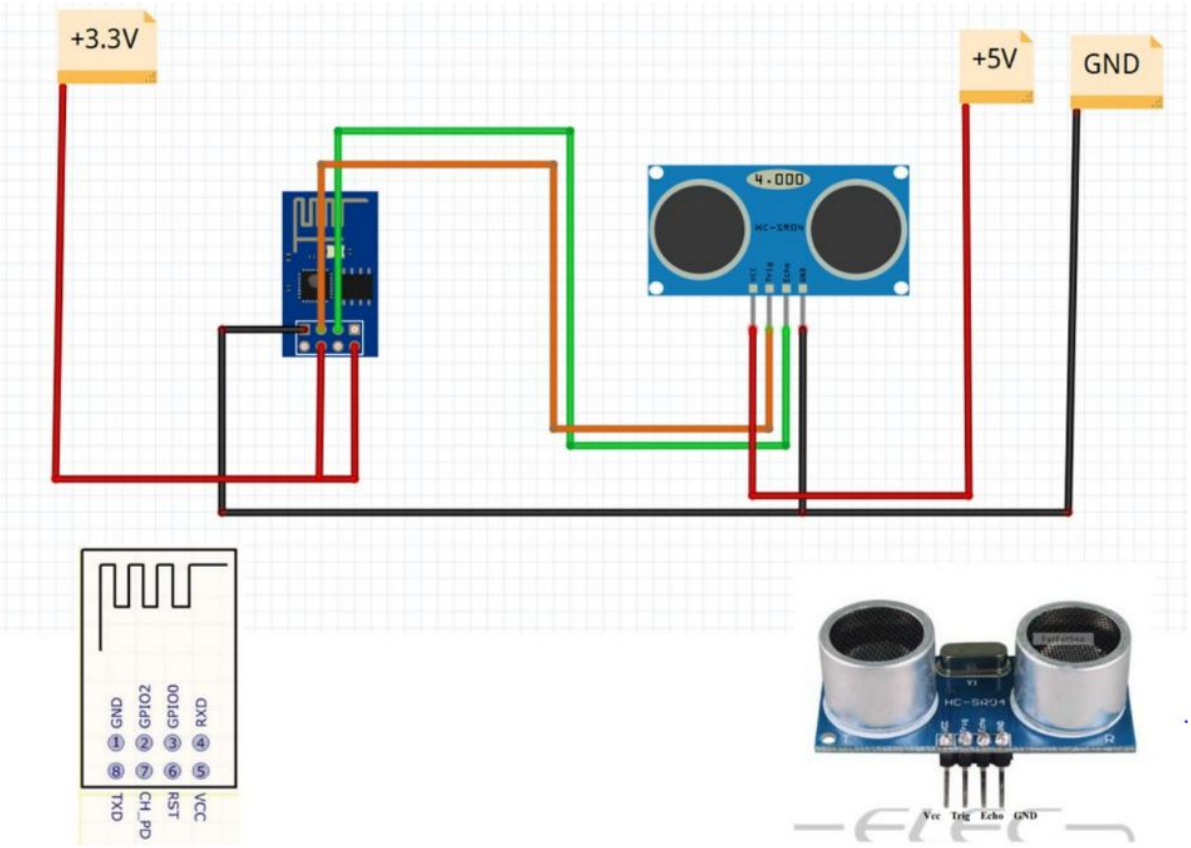
ii. The ESP reads the measures from the HCSR-04 sensor and then send this information to the receiver ESP, the format of the data is:

1. data.id = id of the board (1) int
2. data.x = distance, float
3. data.y = duration (dummy variable, not going to be used in the end), float

iii. Data is sent every 4 seconds

```
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();
  }
}
```

b. Schematics:



2. Data Producer 2 transmission protocol

a. ESP-01 considerations:

i. We make the communications between the ESP's using the library espnow. It registers 2 senders and 1 receiver.

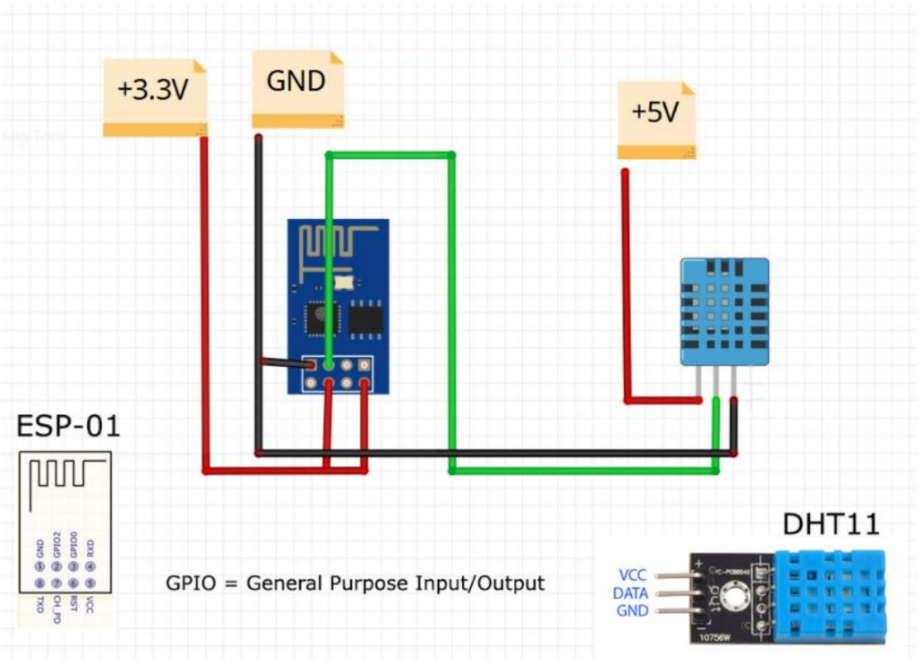
ii. The ESP reads the measures from the DHT11 sensor and then send this information to the receiver ESP, the format of the data is:

1. data.id = id of the board (2) int
2. data.x = temperature, float
3. data.y = humidity(dummy variable, not going to be used in the end), float

iii. Data is sent every time the data measured changes (in comparison with last value sent)

```
void loop() {
  Temperature = dht.readTemperature(); // Gets the values of the temperature
  Humidity = dht.readHumidity();

  if (Temperature != lastTemp && Humidity != lastHum ) {
    lastTemp = Temperature;
    lastHum = Humidity;

    // Set values to send
    myData.id = BOARD_ID;
    myData.x = lastTemp;
    myData.y = lastHum;

    // Send message via ESP-NOW
    esp_now_send(0, (uint8_t *) &myData, sizeof(myData));
  }
  delay(2000);
}
```

b. Schematics:



3. Arduino transmission protocols

a. Arduino considerations:

i. Arduino here have 2 roles, receive the data from the ESP-01 receiver and send that data to the Raspberry through the i2c connection.

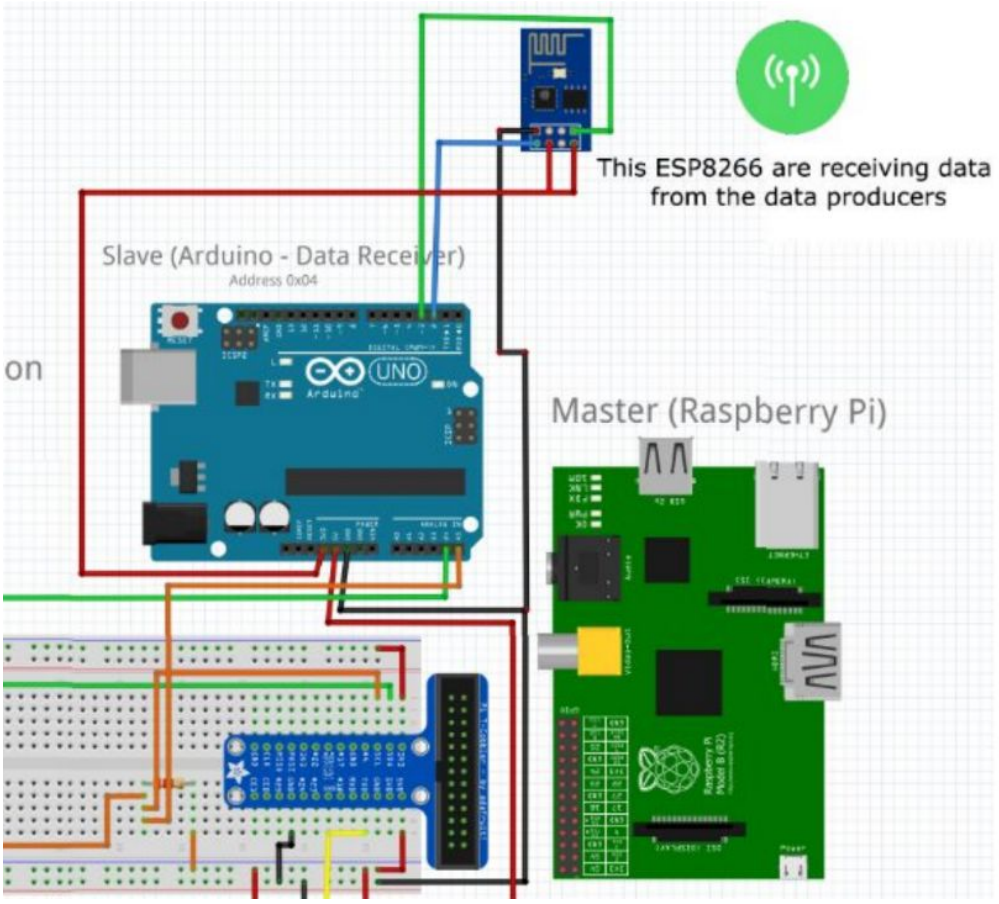
ii. The data is received from the ESP-01 through serial connection, using the softwareserial library.

iii. The data received is in this format:

1. **<Message,Distance,Duration,Temperature,Humidity>** char, float, float, float, float
2. The arduino receives this and then parses the data separating the variables, then the values are stored internally on the arduino.
3. This data is sent to the raspberry when requested through the i2c, using the function **Wire.onRequest(sendData_handler);**
4. The communication in i2c is made through pins A4 (SDA) and A5 (SDL)
5. The format of the data sent is 3 ints:

```
void sendData_handler (){

  sensorData[0] = lastTemp;
  sensorData[1] = lastHum;
  sensorData[2] = lastDistance;

  for (int i=0; i<3; i++) {
      Wire.write(sensorData[i]);  //data bytes are queued in local buffer
  }
}
```

b. Schematics:



4. ESP-01 receiver transmission protocols

a. ESP-01 considerations

i. This ESP-01 receives data periodically from the other 2 ESP's through the espnow connections, the MAC Address of this ESP is present on the code of the ESP senders in order for it to connect with each other.

ii. The data is stored and then sent to the arduino over the serial connection. (Pins TX and RX.

iii. Data is sent in this format whenever a new measure is received:

1. **<Message,Distance,Duration,Temperature,Humidity>** char, float, float, float, float

Receiver_ESP01

```
  //Serial.println(macStr);
  memcpy(&myData, incomingData, sizeof(myData));
  //Serial.printf("Board ID %u: %u bytes\n", myData.id, l
  // 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;

  board1Distance = int(boardsStruct[0].x);
  board1Time = int(boardsStruct[0].y);
  board2Temp = int(boardsStruct[1].x);
  board2Hum = int(boardsStruct[1].y);

  // send data to arduino
  Serial.print("<Data,");
  Serial.print(board1Distance);
  Serial.print(",");
  Serial.print(board1Time);
  Serial.print(",");
  Serial.print(board2Temp);
  Serial.print(",");
  Serial.print(board2Hum);
  Serial.print(">");
}
```

4. ChibiOS Raspberry receiver transmission protocols

b. Raaspberry considerations

i. The raspberry act as a master in the i2c connection
ii. The raspberry is connected to a lcd screen through serial connection
iii. It sends a request to the arduino about the measures, then the arduino responds as previously explained **(3 ints)**
iv. Then, temperature and humidity are used to display in the LCD Screen.
v. The distance, is calculated and then used to send a instruction to the PCF8754 through i2c, the format of the data sent is: **(u_int8) 0b01110000** for example

```
// Request values
status = chBSemWait(&smph);

if (status == 0)
{
  msg_t i2cMsg = i2cMasterReceiveTimeout(
      &I2C0, arduino_address, result, 3, MS2ST(1000));

  if (i2cMsg == 0x00)
  {
    temperature = result[0];
    humidity = result[1];
    distance = result[2];

    stackHandler();

    ledLevel = aux_counter;

    if (ledLevel > 8)
      ledLevel = 0;
  }

  chBSemSignal(&smph);
}
chThdSleepMilliseconds(6000);
```

```
if (ledLevel != lastLedLevel)
{
  status = chBSemWait(&smph);

  if (status == 0)
  {
    pinOut[0] = (uint8_t)handleMeasure(ledLevel);

    i2cMasterTransmitTimeout(&I2C0, pcf_address, pinOut, sizeof(pinOut), NULL, 0, MS2ST(2000));
    chThdSleepMilliseconds(10);

    lastLedLevel = ledLevel;

    chBSemSignal(&smph);
  }
}
chThdSleepMilliseconds(3000);
```