ESCUELA DE
CIENCIAS MATEMÁTICAS Y
TECNOLOGÍA INFORMÁTICA

Wireless Network

Emil Vega-Gualán

LAB 8: Read data from the temperature sensor using bluetooth and arduino

Objective

1. Describe the basic principles of radio communications.
2. Understand the usage of Bluetooth serial communication module and AT command to set the control parameters.
3. Learn how use the Bluetooth module for controlling Arduino via Bluetooth communication.

Discussion of fundamentals

Temperature sensor module LM35

LM35 is a precision IC temperature sensor with its output proportional to the temperature (in o C). The sensor circuitry is sealed and therefore it is not subjected to oxidation and other processes. LM35 possess low self heating and does not cause more than 0.1 o C temperature rise in still air. The operating temperature range is from -55 o C to 150 o C. The output voltage varies by 10mV in response to every o C rise/fall in ambient temperature, i.e., its scale factor is 0.01V/ o C. The LM35 sensor consists of three pins in an encapsulation similar to that of a transistor. The first pin (number 1 in the image), will connect to VCC (+ 5V), the central pin (number 2) will be the sensor output, which will connect to one of the analog pins of our Arduino, and finally the third pin (number 3) will connect to GND (ground).

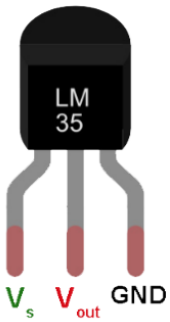


Figure 1: LM35 sensor

Exercise 1. LM35 temperature sensor in Arduino

We have opted for an LM35 sensor that detects temperatures from 55 o C to 150 o C, 1 o C equals 10mV and supports voltages between 4V and 30V. When we read an analog sensor with Arduino we do it through the analogRead function that gives us a value between 0 and 1023, 1024 possible values. If we have 0V at the entrance, it will return 0 and if we have 5V it will return 1023. From this information, we can obtain a mathematical formula that calculates the temperature based on the voltage that the LM35 provides us.

$$Temperature = Value * 1.1 * 100/1024 \tag{1}$$

Building Circuit

Before making the connection make sure to unplug the power source from Arduino UNO. Make following circuit.

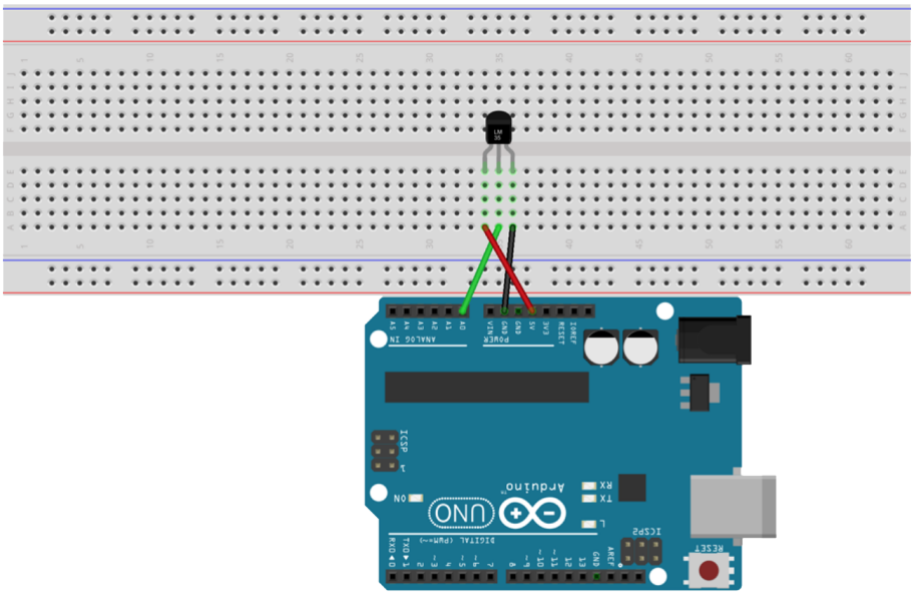


Figure 2: Assembly Scheme

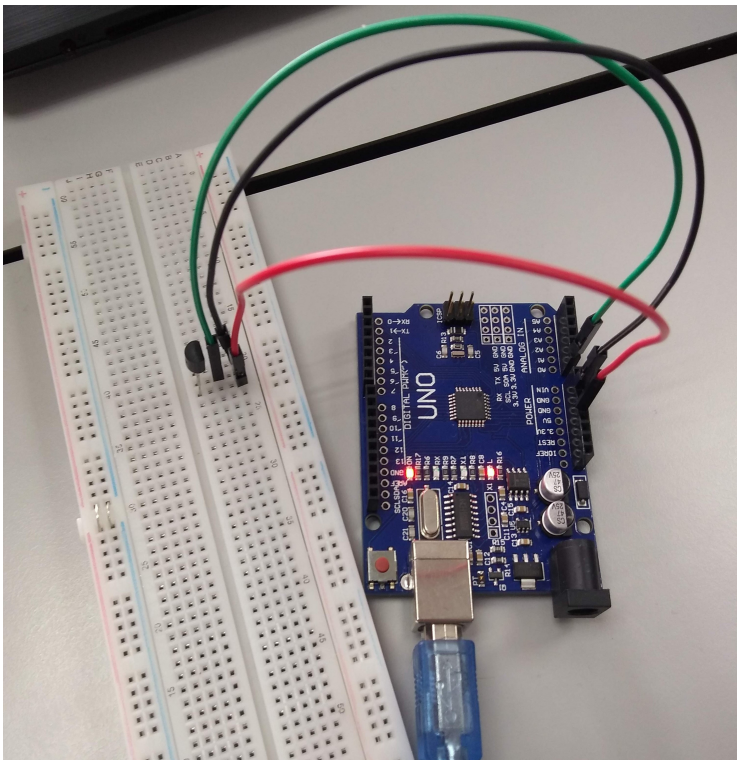


Figure 3: Implementation of the scheme

Programming

Open up the Arduino IDE and write the following code into a new sketch:

Code

```
// 'Declaration of global variables'
float tempC; // 'Variable to store the value obtained from the sensor (0 to 1023)'
int pinLM35 = 0; // 'Variable sensor input pin (A0)'

void setup () {
  // 'We change reference of the analog inputs'
  analogReference (INTERNAL);
  // 'We configure the serial port at 9600 bps'
  Serial.begin (9600);
}

void loop () {
  // 'With analogRead we read the sensor, remember that it is a value from 0 to 1023'
  tempC = analogRead (pinLM35);
  // 'We calculate the temperature with the formula'
  tempC = (1.1 * tempC * 100.0)/1024.0;
  // 'Send the data to the serial port'
  Serial.print (tempC);
  // 'Line break'
  Serial.print("\n");
  // 'We wait for a time to repeat the loop'
  delay (1000);
}
```

Output Result

See the result on Serial Monitor:

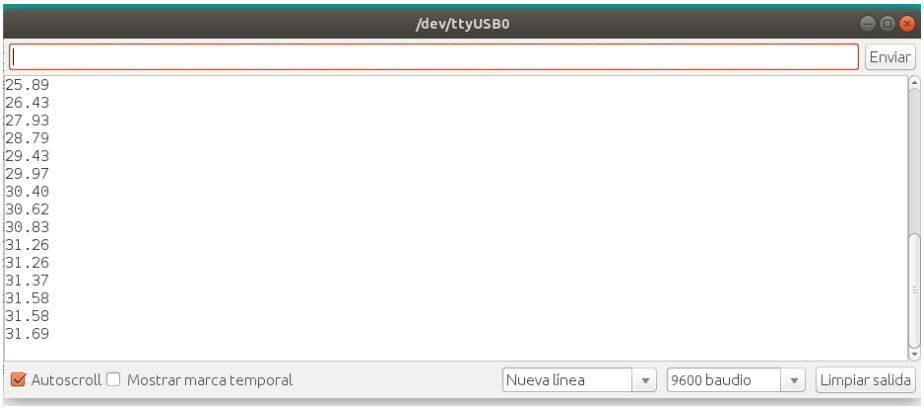


Figure 4: Serial Monitor

Exercise 2. Reading Sensor Data Using Bluetooth

This exercise demonstrates a way to make use of bluetooth for reading data in arduino projects. The aim of this exercise is to gather sensor data from a temperature sensor and transfer it to the PC using bluetooth communication. On the PC side, a simple arduino sketch is written, which will be uploaded onto the arduino board. We will then be able to see the sensor data and time on the serial monitor.

Building Circuit

Before making the connection make sure to unplug the power source from Arduino UNO. Make following circuit.

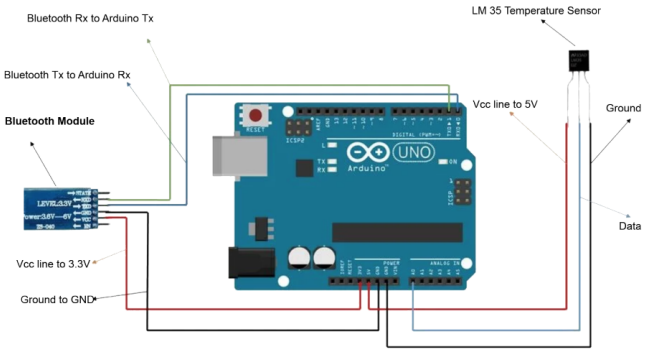


Figure 5: Assembly Scheme 2

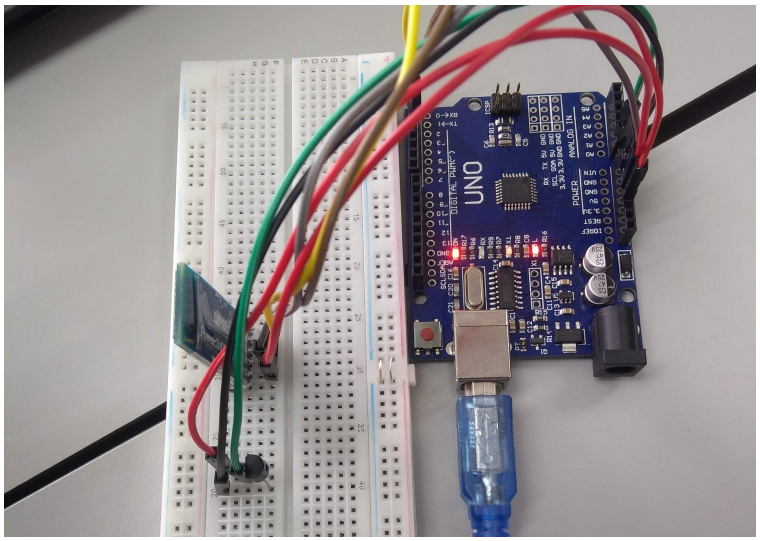


Figure 6: Implementation of the Scheme 2

Programming

Open up the Arduino IDE and write the following code into a new sketch:

Code

```
// 'Declaration of global variables'
int tempC; // 'variable to hold temperature sensor value'
long tm,t,d; // 'variables to record time in seconds'
int pinLM35 = 0;

void setup()
{
  Serial.begin(38400);
  pinMode(pinLM35,INPUT); // 'temperature sensor connected to analog 0'
  analogReference(INTERNAL);
}

void loop()
{
  tempC = analogRead(pinLM35); // 'analog reading temperature sensor values'
  tempC = (1.1 * tempC * 100.0)/1024.0;

  // 'required for converting time to seconds'
  tm = millis() ;
  t = tm/1000;
  d = tm%1000;

  Serial.flush () ;

  // 'printing time in seconds'
  Serial.print("time:") ;
  Serial.print(t) ;
  Serial.print(".") ;
  Serial.print(d);
  Serial.print("s\t");

  // 'printing temperature sensor values'
  Serial.print("temperature:");
  Serial.print(tempC);
  Serial.println ("C");
  delay(2000);  // 'delay of 2 seconds'
}
```

Output Result

Once you have successfully set up the circuit connections and your arduino code has been uploaded, you can connect a power source to your arduino board and disconnect the USB cable connecting your arduino board to the PC. Also, your bluetooth module should start blinking indicating that it is functioning properly and is ready to be paired with your PC. This will demonstrate that the Arduino board can remotely gather sensor data and tranfer it to the PC using bluetooth communication without being connected to it by a USB cable.

To see the sensor data on the serial monitor, go to your Arduino sketch, click Tools -¿ Port and select the COM port which your bluetooth module is connected. Once you have done this, you can open the serial monitor for that port; and the temperature sensor data should be dislayed along with the time. To view the data from zero seconds onwards, press the reset button on your arduino board.

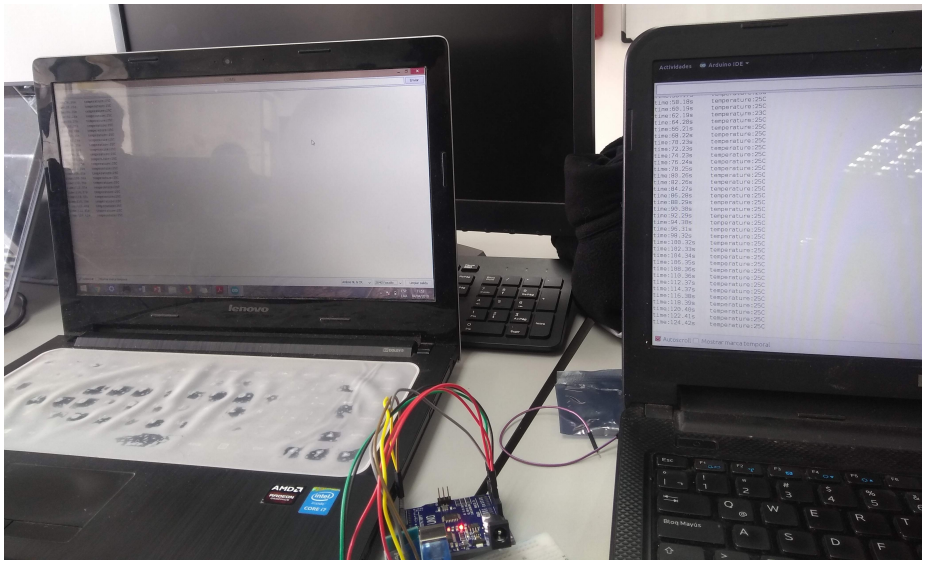


Figure 7: Same output in both computers because the Bluetooth transmission.

Exercise 3. Pairing two Bluetooth Modules and share data between them

From the lab 7, it has been already learned to configure control parameters of a Bluetooth Module. In this exercise, it will be demonstrated that how two Bluetooth modules can be paired together and allowed to share data between them. To do so, you have to configure one HC-05 as slave, and other HC-05 as master.

Steps for SLAVE:

1. Be sure to select correct Baudrate in your Terminal Emulator (Arduino IDE) for your module. There is no default baudrate. Different manufacturers set the baudrate on their HC-05s differently... 9600 and 38400 are good starting points.
2. Enter AT mode
3. Note the values of `AT+PSWD?`, `AT+NAME?` and `AT+ADDR?` (lets say `1111:22:123456`) commands. You might set the password to 1234 for easy remembering Issue `AT+PSWD=0000`...
4. Be sure to set the module to slave mode by issuing `AT+ROLE=0` command.

Steps for MASTER:

1. Be sure to select correct Baudrate in your Terminal Emulator (Arduino IDE) for your module. There is no default baudrate... Different manufacturers set the baudrate on their HC-05s differently... 9600 and 38400 are good starting points.
2. Enter AT mode
3. Reset the module to its original state by issuing `AT+ORGL` command.
4. Remove the previous parings(if any) by issuing `AT+RMAAD` command.
5. Set same password as slave by issuing `AT+PASSWD=0000` command.
6. Change the role to master; `AT+ROLE=1`
7. Set connecting mode to any address it sees; `AT+CMODE=1`
8. Initialize the SPP profile lib ; `AT+INIT`
9. Be sure that you see the slave HC-05 module, `AT+INQ` command will list the modules you see.
10. Be sure that you see address of the slave module(`1111:22:123456`).
11. Last of all enter `AT+LINK=1111,22,123456` (note the commas).

With these steps you should exchange data...

Building Circuit

For this, the same scheme from the exercise 1 in the lab 7 is used.

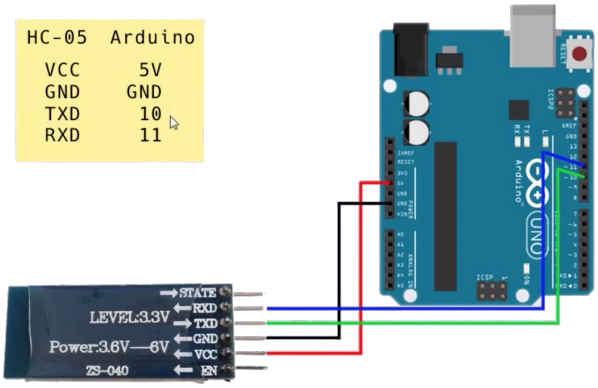


Figure 8: AT commands in Arduino

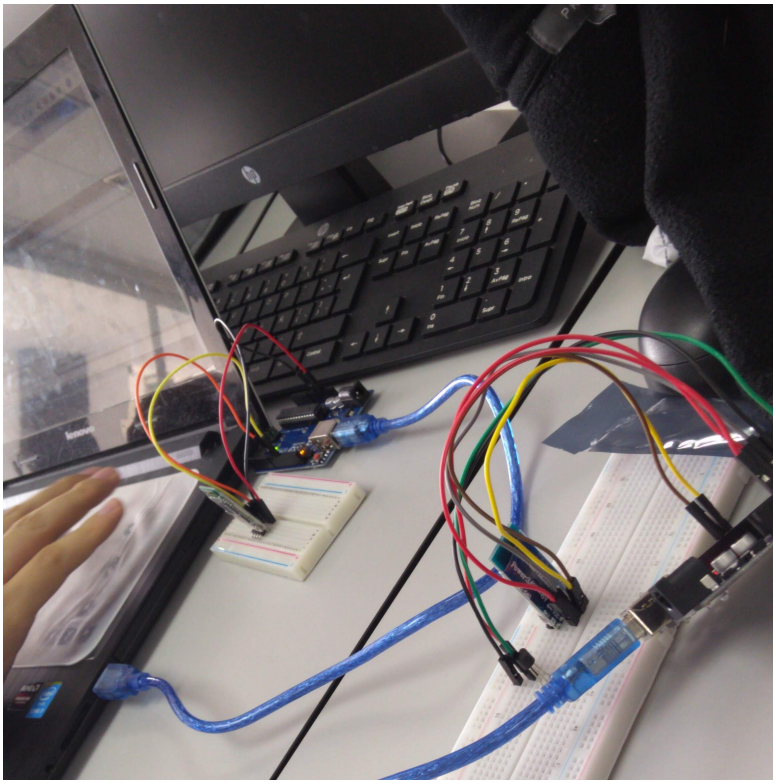


Figure 9: Implementation of the scheme in two computers to communication.

Programming

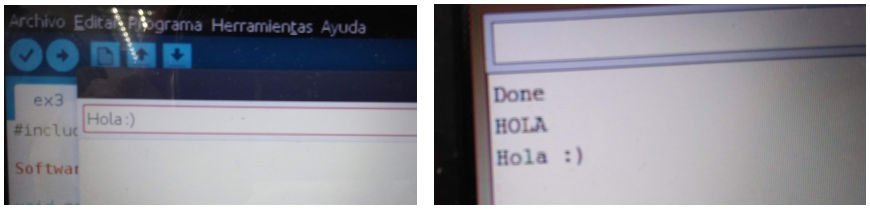
The programming is the same as in the Exercise 1 from lab 7. To configure the Bluetooth module.

Code

```
#include <SoftwareSerial.h> // 'set digital pins for serial communication'

SoftwareSerial wirelessNetworkYT(10, 11); // 'pin 10 as RX, pin 11 as TX'

void setup(){
  Serial.begin(9600); // ' serial monitor communication at 9600 bps'
  Serial.println("Done"); // 'write "Done" on the monitor'
  wirelessNetworkYT.begin(38400); // 'communication at 38400 bps'
}

void loop(){
  if (wirelessNetworkYT.available())      // 'information available from module'
    Serial.write(wirelessNetworkYT.read()); // 'read Bluetooth and send it to Arduino serial
        monitor'

  if ( Serial.available ())         // 'information available from the serial monitor'
    wirelessNetworkYT.write(Serial.read());   // 'read serial monitor and send to Bluetooth'
}
```

Output Result

The result was that a message was sent form one computer to another.



(a) Computer 1 (b) Computer 2

Figure 10: Message sent through Bluetooth module

Exercise 4. HC-05 Bluetooth Module As Master and Slave

Design a circuit that allows to turn on and off two LEDs through the Bluetooth module. In this exercise, both master and slave will be controlling LEDs by using buttons. Button A at master will be controlling LED3 and LED4 at slave whereas Button B at slave will control LED1 and LED2 at master.

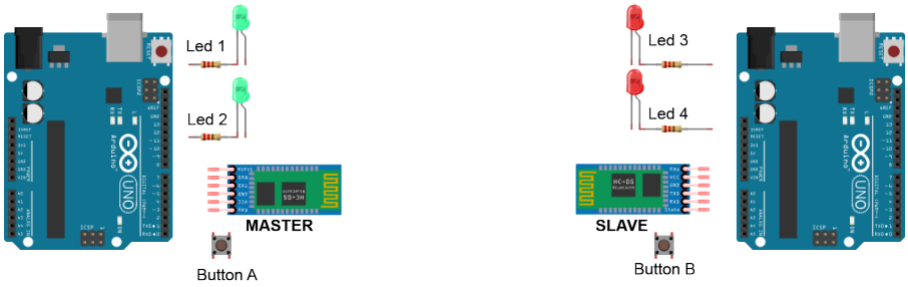


Figure 11: Assembly Scheme 3

Building Circuit

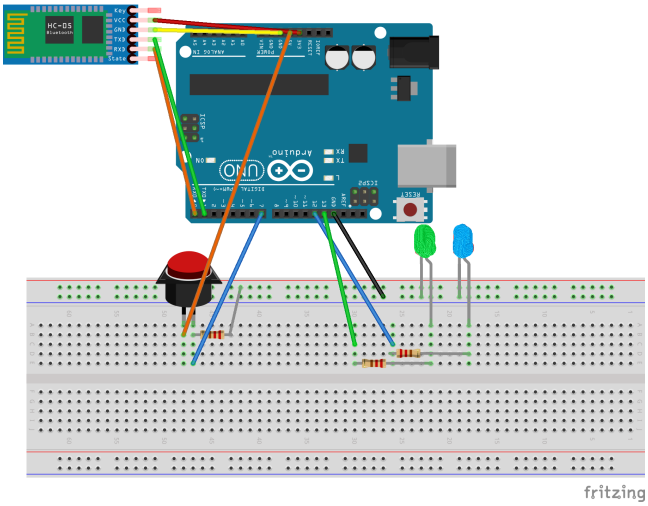


Figure 12: Scheme in firtzing

Programming

Code

```
#include<SoftwareSerial.h> // 'set digital pins for  serial  communication'

SoftwareSerial wirelessNetworkYT(10, 11); // 'pin 10 as TX, pin 11 as RX'

char DAT = 0;   // 'variable  to  store  received  character'
int  redLed = 2; // 'Yellow LED with digital pin number 5'
int  greenLed = 3; // 'Green LED with digital pin number 4'
bool stateL = false;
int  stateB = 0;
int  stateM=0;
int  button = 4;

void setup(){
  wirelessNetworkYT.begin(38400);// ' Serial communication between Arduino and the module at
       38400 bps'
  pinMode(redLed,OUTPUT); // 'pin 3 as output'
  pinMode(greenLed,OUTPUT); // 'pin 6 as output'
  pinMode(button,INPUT);
  digitalWrite(redLed, LOW); // turn off red LED
  digitalWrite(greenLed, LOW); // turn off green LED
  Serial.begin(9600);
  Serial.print("Done");
}

void loop(){
  if (wirelessNetworkYT.available()){ // ' if  there  is  information available  from module'
    DAT = wirelessNetworkYT.read(); // 'stores in DAT the character received from module'
    if( DAT == '2' ){// ' if the character received  is  number 2'
      DAT=0;
      if  (!stateL){
        stateL=true;
        digitalWrite(redLed, HIGH); // turn on red LED
        digitalWrite(greenLed, HIGH); // turn on green LED
      }else{
        stateL=false;
        digitalWrite(redLed, LOW); // turn off red LED
        digitalWrite(greenLed, LOW); // turn off green LED
      }
    }
  }
  stateB=digitalRead(button);
  if(stateB==HIGH){
        if(stateM==0){
            wirelessNetworkYT.write('2');
            stateM = 1 ;
        }
  }else {
      if  (stateM == 1) {
          stateM = 0;
        }
    }
}
```

Output Result

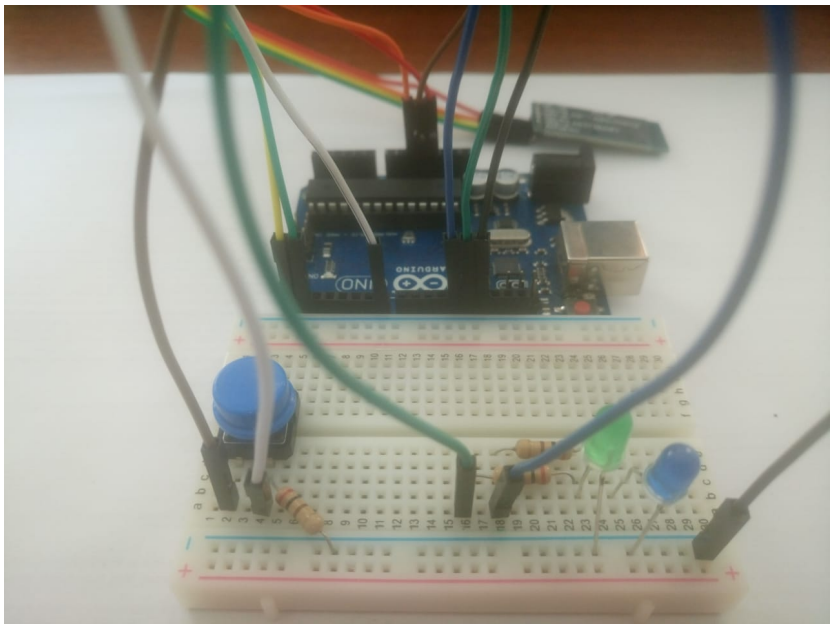


Figure 13: Implementation of Scheme 3

Exercise 5. Build Your Own Bluetooth project

Create your own bluetooth system. New approaches will be highly valued.

We created a control of LEDs using the cards of the RFID module sending a signal through Bluetooth to another arduino.

Building Circuit

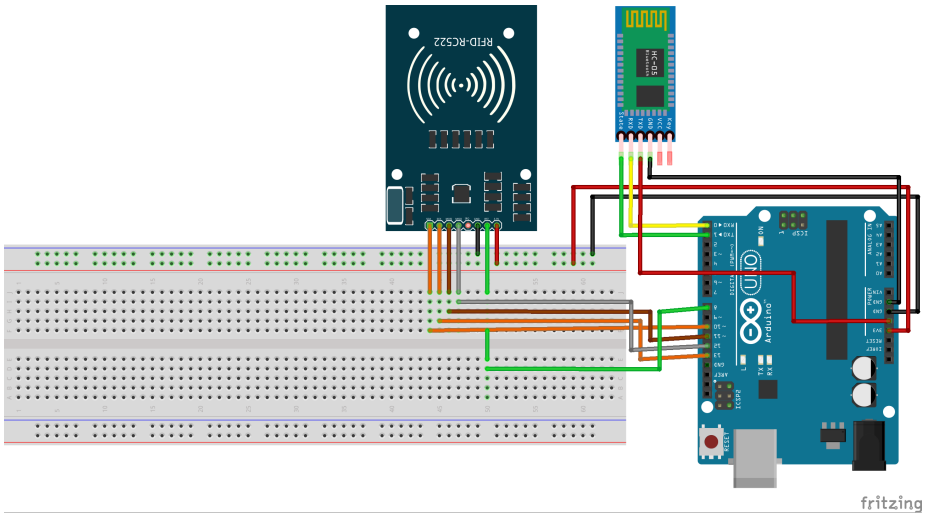


Figure 14: Assembly scheme of the transmitter

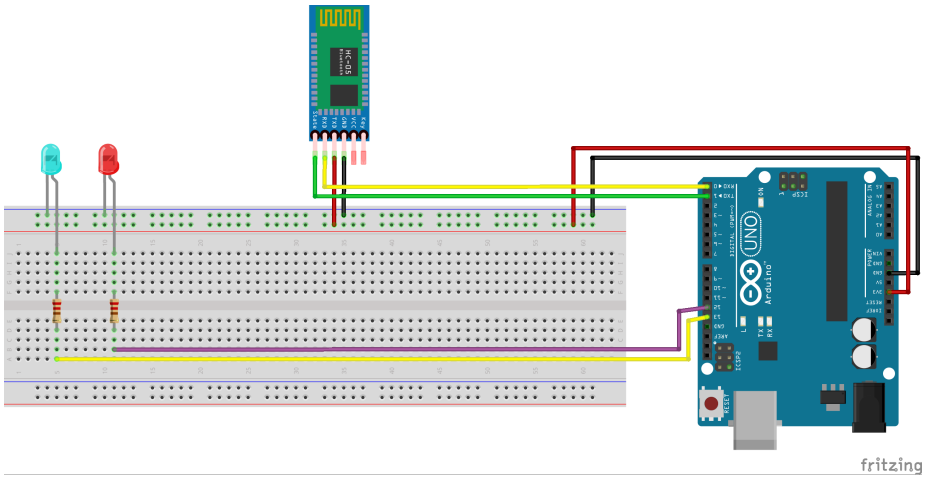


Figure 15: Assembly scheme of the receiver

Programming

The following code belongs to the transmitter device:

Code

```
#include <SPI.h>
#include <MFRC522.h>

#define RST_PIN         9           // 'Configurable, see typical pin layout above'
#define SS_PIN          10          // 'Configurable, see typical pin layout above'

MFRC522 mfrc522(SS_PIN, RST_PIN); // 'Create MFRC522 instance'

byte LecturaUID[4];
byte Usuario1[4]={0X86,0XFB,0X70,0X13};
byte Usuario2[4]={0X2C,0XEA,0XF1,0XC5};

void setup() {
  Serial.begin(9600);    // ' Initialize  serial  communications with the PC'
  SPI.begin();        // Init  'SPI bus'
  mfrc522.PCD_Init();   // 'Init  MFRC522'
  }

void loop() {
  Serial.flush();
  // 'Look for new cards'
  if ( ! mfrc522.PICC_IsNewCardPresent())
    return;

  // 'Select  one of the cards'
  if ( ! mfrc522.PICC_ReadCardSerial())
    return;

 // 'ALMACENA UID'
 for(byte i=0;i< mfrc522.uid.size;i++){
  LecturaUID[i]=mfrc522.uid.uidByte[i];
 }

 if(comparaUID(LecturaUID, Usuario1)){
  Serial.print("1");
 }else if(comparaUID(LecturaUID, Usuario2)){
  Serial.print("1");
 }else{
  Serial.print("0");
 }
    mfrc522.PICC_HaltA(); // 'DETIENE COM. CON CARD'
    delay(1000);
}

boolean comparaUID(byte lectura[], byte usuario[]){
  for(byte i=0;i< mfrc522.uid.size;i++){
  if(lectura[i]!=usuario[i])
    return(false);
  }
  return(true);
}
```

The following code belongs to the receiver device:

Code

```
// 'Declaration of global variables '
const int LED1 =13; // 'CORRECTO'
const int LED2 =12; // 'INCORRECTO'

int inByte = 0; // 'RECIBE ORDEN DE OTRO ARDUINO'

void setup(){
  Serial.begin(9600);
  pinMode(LED1,OUTPUT); // 'establecer que el pin digital es una seal de salida'
  pinMode(LED2,OUTPUT); // 'establecer que el pin digital es una seal de salida'
  digitalWrite(LED1,LOW);
  digitalWrite(LED2,LOW);
}

void loop(){
  Serial.flush();
  // 'RECIBE ORDEN DEL OTRO ARDUINO'
  if(Serial.available()>0){
    inByte=Serial.read();
    if(inByte=='1'){ // 'USUARIO ACEPTADO'
      digitalWrite(LED1,HIGH);
      delay(1000);
      digitalWrite(LED1,LOW);
    }
    else if(inByte=='0'){ // 'USUARIO DENEGADO'
      digitalWrite(LED2,HIGH);
      delay(1000);
      digitalWrite(LED2,LOW);
    }
  }
  delay(1000);
}
```

Output Result

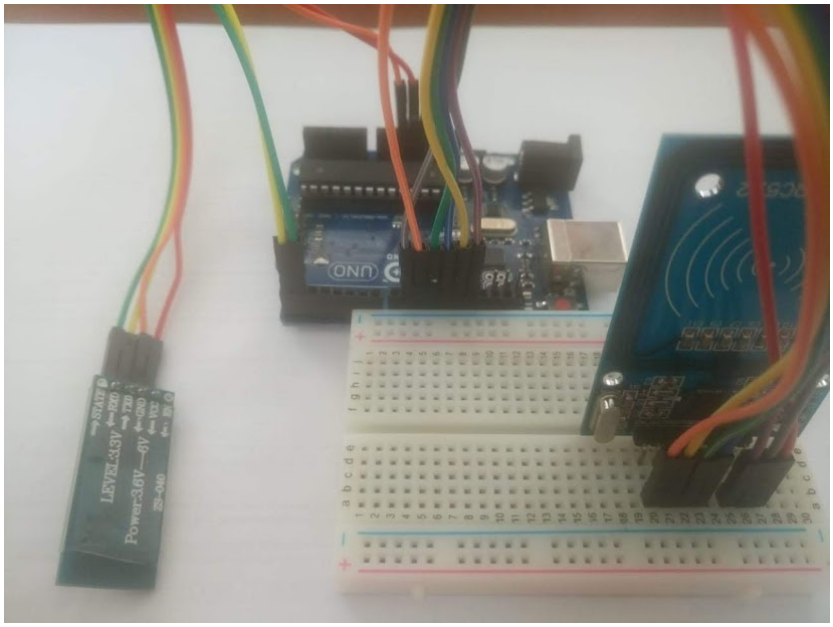


Figure 16: Implementation of the transmitter

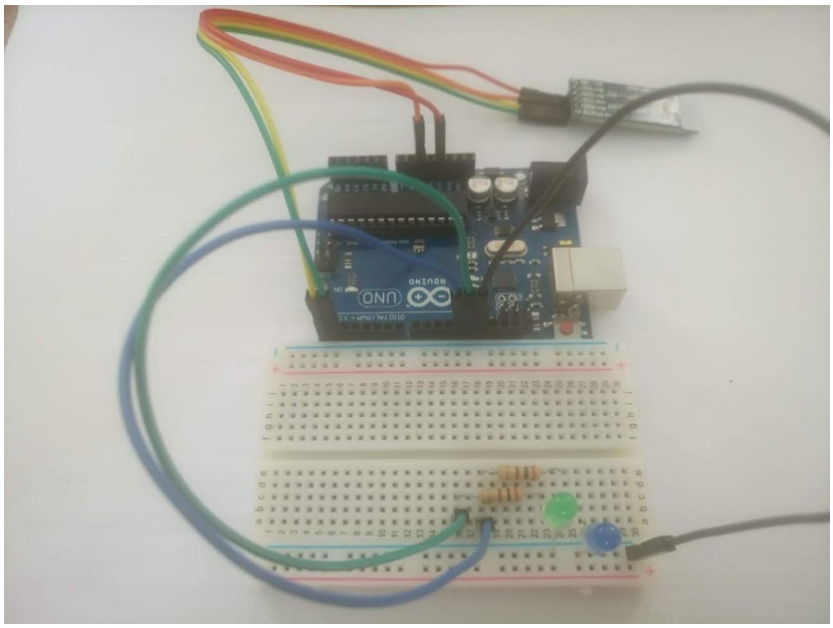


Figure 17: Implementation of the receiver