# Arduino Remote Sensor Data Transmitter/Receiver

Alejandro de Haro, Daniel Jarne, Josep M. Palet

Universitat Politècnica de Catalunya

**Abstract**

This report covers the technical process of the implementation of a communication system between an analog sensor such as temperature TMP36 and a computer in order to plot real-time data using Matlab software. The communication channel is over radio frequency at 433MHz using two Arduino UNO boards, one as a transmitter and the other one as a receiver. Finally, a video has been recorded in order to show the results of this work. It can be found at `http://youtu.be/an6dajl-xX0`.

## 1 Introduction

This project was intended to take data from a remote sensor station, such as the one on board on airplanes, and process this data form determining flight conditions of the plane, such as its attitude, altitude, velocity and other aircraft parameters. Despite our original intentions, we had two main problems in the developing of our project:

- Money: a whole sensor station with temperature, pressure, humidity sensors, accelerometer and GPS (this was the first station we wanted

to make) was a lot more expensive than what we initially expected (the whole station costs are around 200). Our solution was to reduce the number of sensors and, as a consequence, the quantity of data to be processed.

- Shipping: mean shipping time for components where we bought them was about 7 days, but finally that time extended to a whole month, thing that made us start later than we had expected in an initial prediction of the project progress.

Finally, our project, named Arduino Remote Sensor Data Receiver, takes data from remote sensors for its processing. Its basic structure is the following: two Arduino boards connect themselves via radio frequencies; one of the boards, the emitter, has sensors attached, such as temperature and pressure sensors. The other board, the receiver, is connected to a computer, so that it establishes connection to the other board and actuates as an intermediate between sensors and the central computer.

## 1.1 Objective

The objective of this is to receive data from remote sensors, so the computer can use this data to study, for example, weather conditions where the sensors are placed. If sensors were on different parts of a plane, remote sensor communication would allow to have less physical connections, so that the central computer of the plane could retrieve data from sensors and modify different parameters of the plane in order to adapt its flight regime to external conditions, such as angle of attack, spoilers deflections, speed, or simply study passively weather evolution for preventing some potential dangerous situations, such as crossed wind, which can cause difficulties in the aircraft operations.

This data processing would also enable us to communicate with the on board computer, so that we could easily adapt our project to do bidirectional processes: Receive data from sensors, process it on the ground

using supporting software (such as MATLAB), and analyze this data to accurately determine what is really going on in the aircraft. Then, we would be able to build automatic controlling algorithms so that we would transmit back to the aircraft the instructions to regulate the aircraft behavioral in front of perturbations.

Building a solid scheme of sensors, Arduino controllers and having an adequate processing algorithm will enable us to apply our project to many aerospace systems.

For example, we could install it in atmospheric air balloons. With temperature and pressure data we could obtain the altitude at which the balloon is operating. Using also GPS data and acceleration data from adequate sensors we could track the balloon, and determine if it is being dragged by non-expected wind currents. If the balloon had actuators (such as small helix engines) we could use the data to control the balloon consequently.

# 2 Materials

The materials used for this project are summarized in the following table:

| **Quantity** | **Component** | **Main function** |
|---|---|---|
| x2 | Arduino UNO Rev 3 | Micro-controllers |
| x2 | Prototype Shield v5 | Solid interface to connect devices |
| x1 | Transmitter CDT-88 | Transmitter module |
| x1 | Receiver R03A | Receiver module |
| x1 | TMP36 | Temperature sensor |

Table 1: List of materials

## 2.1 Arduino UNO Rev 3

Arduino UNO is an AT-Mega 328-based micro-controller, which function is, basically, deal with different kind of input and output analogical or digital

data from external devices or from a computer through USB connection.

In this project, there is the need of using a micro-controller which connects, syncs and manages both sensor and emitter/receiver stations. Arduino UNO Rev. 3 brings enough processing power for this project needs: take data from a temperature sensor, convert it into a char sequence and send it through a pin which is connected to an emitter with an antenna attached to it. Also, in the receiver station, the Arduino board receives the char string codified in ASCII code, decodes it into an entire number (in this case, it is being broadcasted an entire value for temperature in degrees centigrade) and passes this information to the computer, in which the storage of data and its plotting is done.

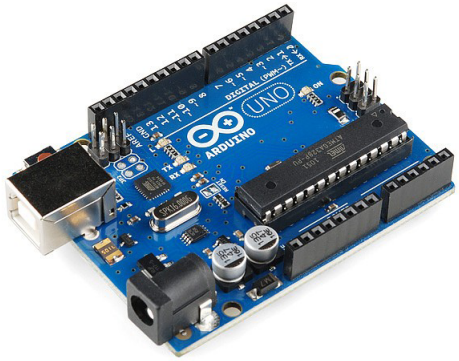


Figure 1: Arduino UNO Rev.3

## 2.2 Prototype Shield v5

The prototype shield v5 is an Arduino attachment which function is simply provide a protoboard connected to the Arduino system without external wires, which simplifies connections with external components not specifically designed for Arduino applications. Also, it provides more capability of

connecting more devices to Arduino board, which is the main reason why it is needed in this project: there has to be connected to Arduino the temperature sensor, and also the emitter/receiver module, which has to have attached also an antenna. All this connections are well managed with prototype shield v5.

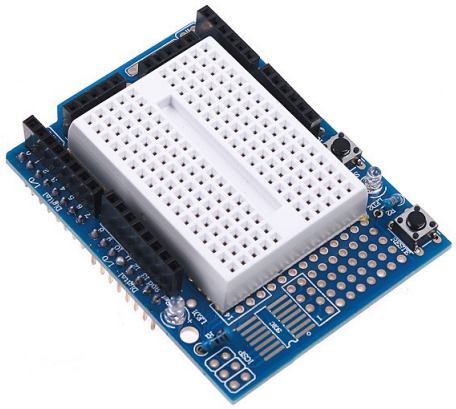

Figure 2: Prototype Shield v5

### 2.3 Transmitter CDT-88

Because of this kind of project, it is essential to have a transmission module. CDT-88 transmits information in the band of 433 MHz, which is reserved for license free communication devices, such as in this case. This transmitter takes data from Arduino and sends it through the antenna, codifying char elements into ASCII code for its transmission.

### 2.4 Receiver R03A

This receiver, calibrated specifically for receive analogic signals in the band of 433 MHz, transforms electric and magnetic fields emitted by its reciprocal

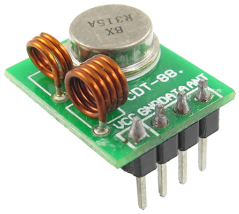

Figure 3: Transmitter module, CDT-88

in the other Arduino board into electrical signals that can be read and interpreted by Arduino.

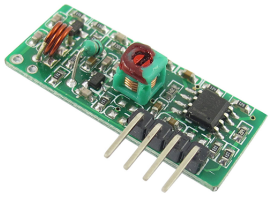

Figure 4: Receiver module, R03A

### 2.5 Temperature sensor TMP36

This temperature sensor has an accuracy of 1C with low temperatures in its measurement range, and 2C for high temperatures. This accuracy is enough for making a good approximation of at which altitude a plane is using ISA (International Standard Atmosphere) equations. Data made by this sensor is sent to Arduino board, which directly sends it to the emitter station for its real-time broadcast, in order to ensure an actualized lecture of the aircrafts altitude (in case it was on an aircraft).

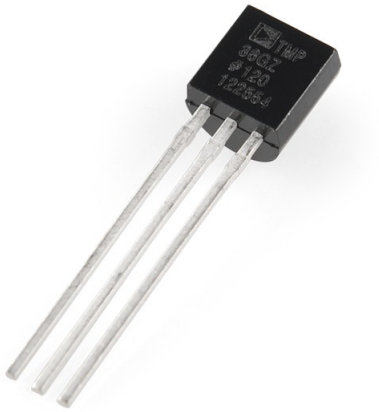


Figure 5: Temperature sensor, TMP36

# 3 Hardware connections

The project is composed by two different main parts: the transmitter module and the receiver module. Their description, also with their connection diagrams, are handled below.

## 3.1 Transmitter module

The transmitter module is an Arduino-powered sensor module with an antenna and an emitting module attached to it. In this case, the sensor is a TMP36, which measures ambient temperature in a range of -40C to 80C. The emitting module is a CDT-88, very similar to the more commonly-used in market WRL 8945. A protoboard keeps all components connected.

The connection diagram is shown in the Figure 6:

Color legend of wires is the following:

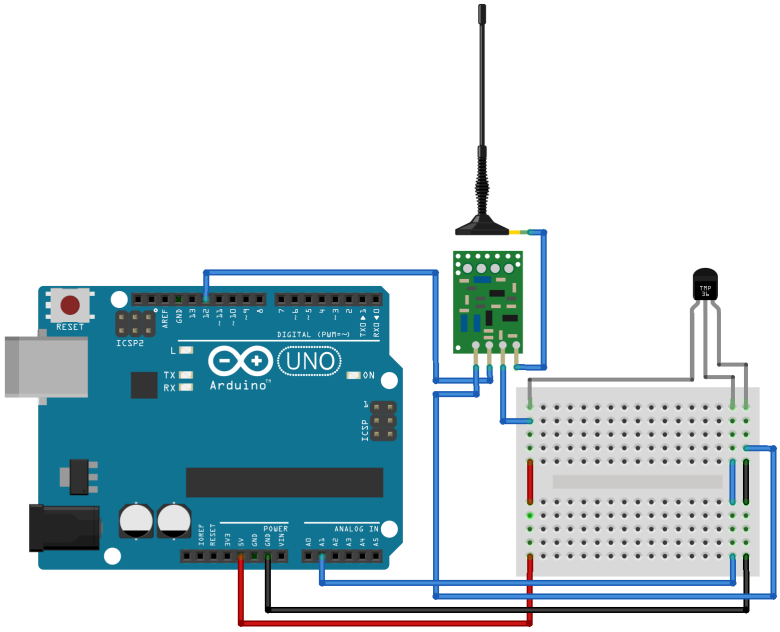

Figure 6: Connection scheme of components for the transmitter device

- **Red wires**: Connection to power source of +5V (5V pin) of Arduino board.
- **Black wires**: Connection to ground (GND pin) of Arduino board.
- **Brown wires**: Connection between the antenna and the other components.
- **Blue wires**: Connection between data ports (input and output).

The electrical connection scheme of this module is shown in Figure 7.

### 3.2 Receiver module

The receiver module consists in an antenna connected to a receiver, which is also connected to an Arduino. In this case, no protoboard is needed for managing the connections that must be made. The receiver module is

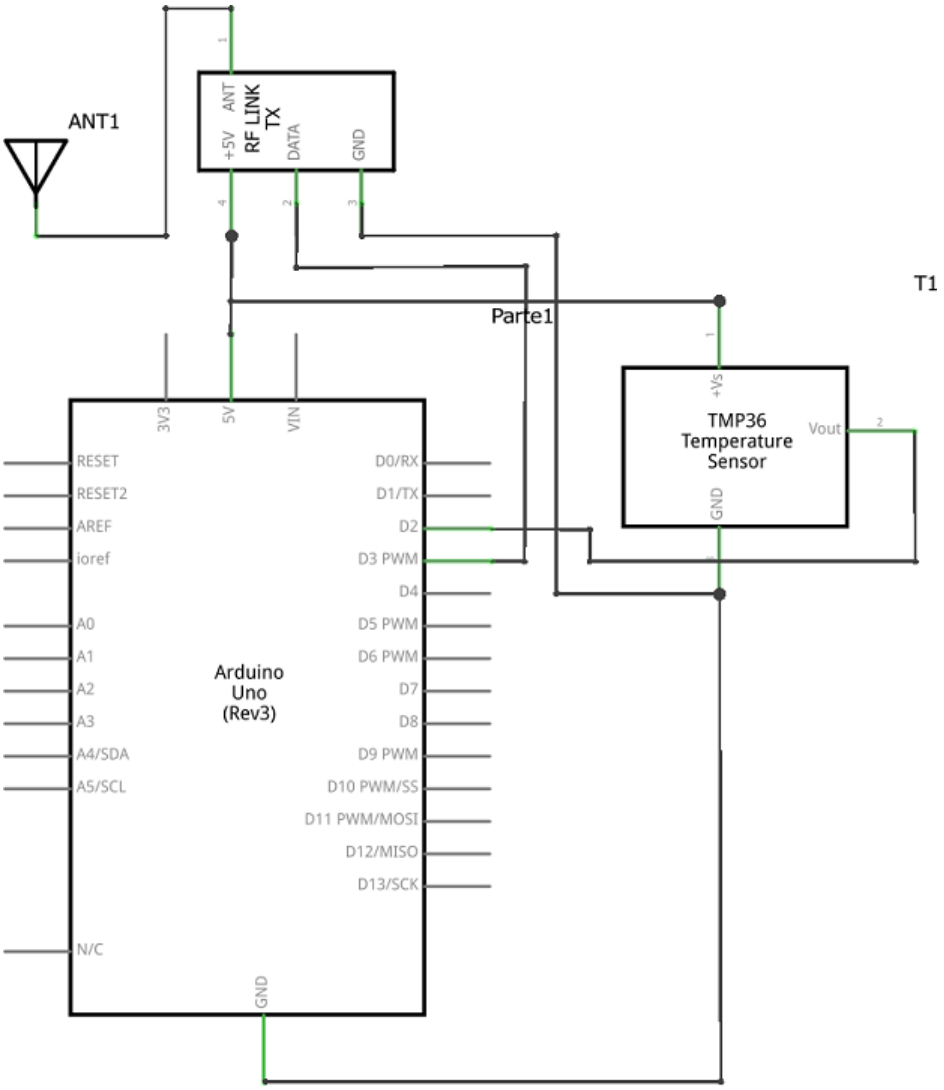


Figure 7: Electrical connection scheme for the transmitter device

a R03A, very similar to the WRL 8947, more commonly used in Arduino radio-frequency applications.

The connection diagram is shown in the Figure 8

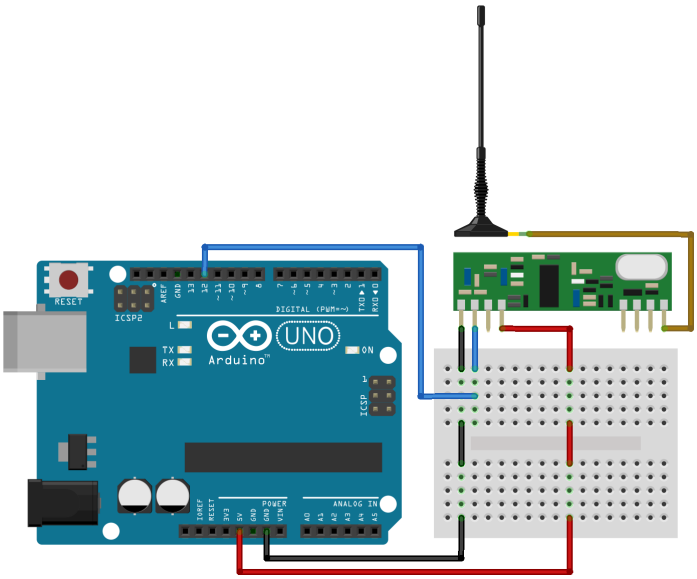


Figure 8: Connection scheme of components for the receiver device

Color legend of wires is the following:

- **Red wires**: Connection to power source of +5V (5V pin) of Arduino board.
- **Black wires**: Connection to ground (GND pin) of Arduino board.
- **Brown wires**: Connection between the antenna and the other components.
- **Blue wires**: Connection between data ports (input and output).

The electrical connection scheme of this module is shown in Figure 9

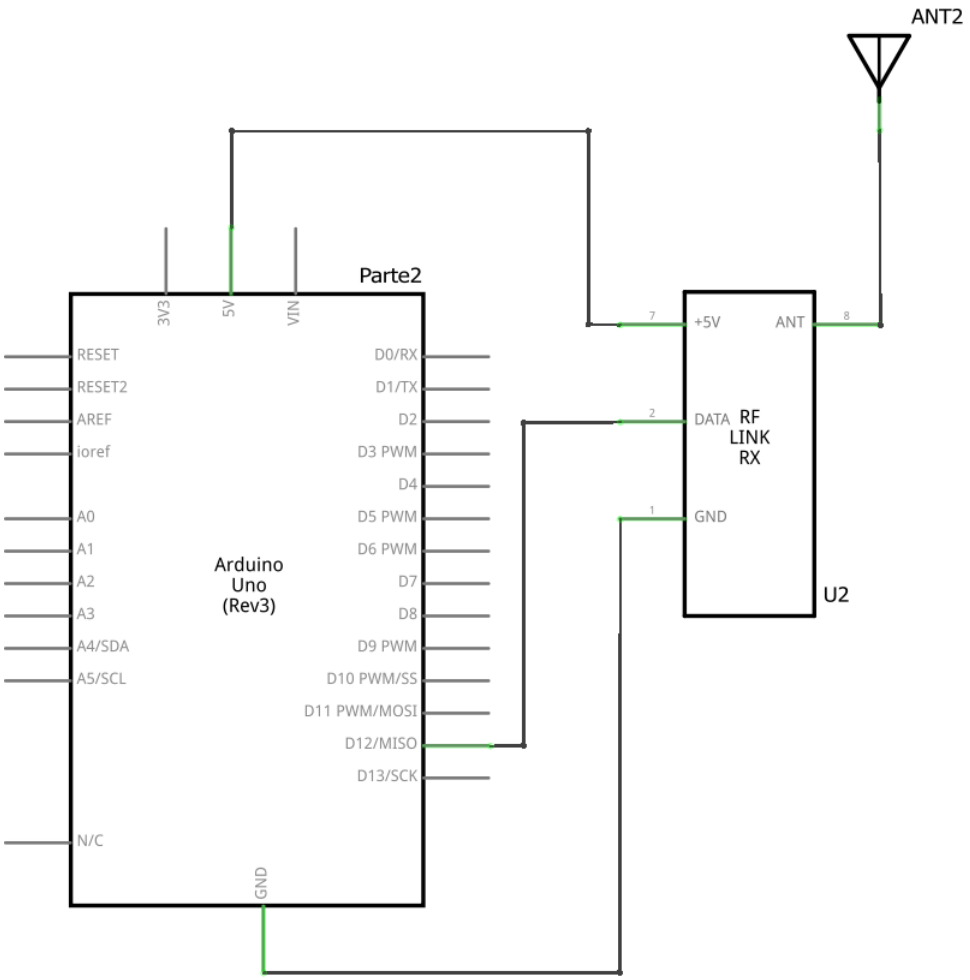


Figure 9: Electrical connection scheme for the receiver device

# 4 Device operation

As it is physically separated in two parts, the whole device operation has also to be separated in two distinguishable parts: the emitter and the receiver.

## 4.1 Transmitter

The TMP36 temperature sensor takes lecture of the temperature of air surrounding it and sends this data through the wires, which are connected to analog input pin 12 of Arduino board. The Arduino takes this data and reroutes it through the pin 13, which is connected to the data pin of the RF emitter. Once the data has arrived to the emitter, it is sent through the antenna in a wave which theoretical range is 40 m (the practical one has been measured in a 20 m range wave).

## 4.2 Receiver

The receiver module has attached the R03A, which has an inside-build antenna for receiving the 433 MHz band, in which the emitter sends the information of the TMP36 sensor. Once the antenna receives the information, it is sent through the pin 13 to the Arduino, and then the signal is decoded in the Arduino board, and sent to the computer through the USB port.

## 4.3 Computer

Once the computer gets the information from the Arduino receiver board, it sends it to MATLAB, which does a real-time plot of the data received, and also it calculates the altitude at which the aircraft (in case it was on it) is as described by ISA equations in real time. At the same time it does these two tasks, it also stores the received data for its further processing (if some other calculations have to be made). (Figure 10)

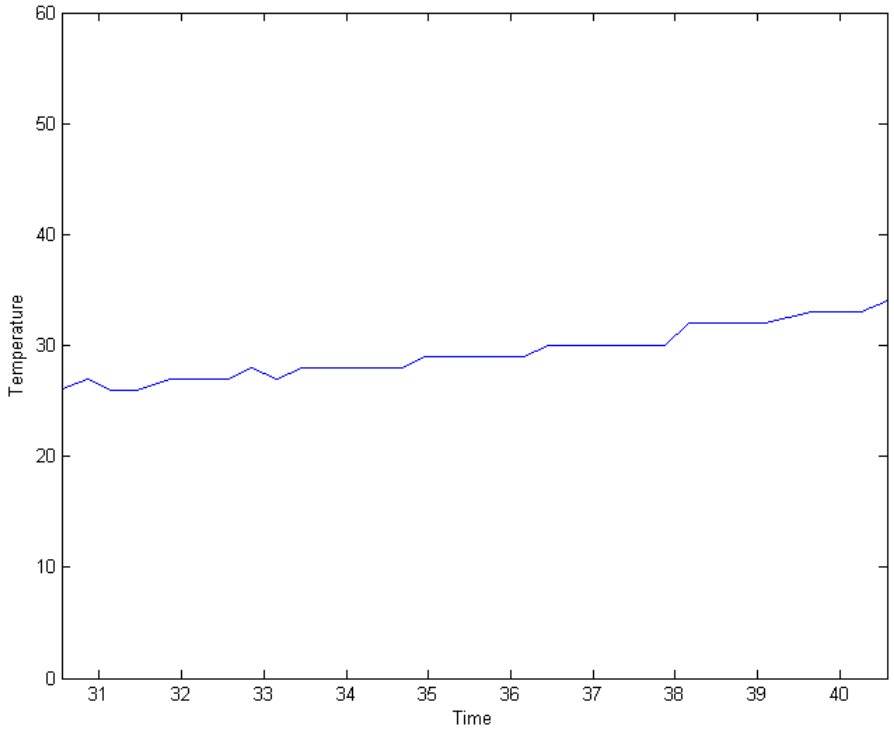


Figure 10: Screenshot of real-time temperature plotting with Matlab

# 5 Software

## 5.1 Transmitter

```
#include <VirtualWire.h>                          // |  RF transmitter library
void setup(){                                     // |
  Serial.begin(9600);                             // |
  pinMode(13,OUTPUT);                             // |  Port 13, LED to check
                                                  // |  data transmission
  pinMode(1,INPUT);                               // |  Analog input 1,
                                                  // |  TMP36 analog data pin
  vw_set_ptt_inverted(true);                      // |
  vw_set_tx_pin(12);                              // |  Emitter digital data pin
  vw_setup(4000);                                 // |  Data transfer velocity (bps)
}                                                 // |
void loop(){                                      // |
  int Temp=(analogRead(1)-105)*27/(156-105);      // |  Zero adjustment
                                                  // |  (105 --> 0 C; 156 --> 27 C)
  char Signal[24];                                // |  Variable which content will
                                                  // |  be sent through the antenna
  sprintf(Signal,"%i",Temp);                      // |  Variable filling
  vw_send((uint8_t *)Signal,strlen(Signal));      // |  Variable content sending
  Serial.print("Temperature: ");                  // |  Comprovation of data sent
                                                  // |  visualizing it by the
  Serial.print(Signal);                           // |  Serial console
  Serial.print(" C");                             // |
  Serial.println();                               // |
  vw_wait_tx();                                   // |  Pause for letting the emitter
                                                  // |  complete the transmission
  digitalWrite(13,1);                             // |  Turn on LED on port 13 for
                                                  // |  checking data transmission
  delay(10);                                      // |
}                                                 // |
```

## 5.2 Receiver

```

#include <VirtualWire.h>                  // |  RF receiver library
int num[2];                               // |
int num_int;                              // |
int num_int_ant;                          // |
int rang=3;                               // |
void setup(){                             // |
```

```
  Serial.begin(9600);                    // |
  vw_set_ptt_inverted(true);             // |  Required for DR3100
  vw_set_rx_pin(12);                     // |  Receiver digital data pin
  vw_setup(4000);                        // |  Data transfer velocity (bps)
  pinMode(13, OUTPUT);                   // |  Port 13, LED to check data reception
  vw_rx_start();                         // |  Start the receiver PLL running
}                                        // |
void loop(){                             // |
  num_int_ant=num_int;                   // |  Storage of previous reception
  uint8_t buf[VW_MAX_MESSAGE_LEN];       // |  Declaration of buffer variable
  uint8_t buflen=VW_MAX_MESSAGE_LEN;     // |  Maximum length of buffer variable
  if(vw_get_message(buf,&buflen)){       // |  Comprovation of data reception
    for(int i=0;i<buflen;i++){           // |
      num[i]=int(buf[i]);                // |  Storage of data received on the
      }                                  // |  buffer variable
  num_int=10*num[0]+num[1];              // |  Union of data received in a
  if(abs(num_int-num_int_ant)<rang){     // |  single variable
    for(int i=0;i<buflen;i++){           // |
      Serial.print(char(buf[i]));        // |  Transmission of data variable to
      }                                  // |  Serial Console
    Serial.println();                    // |
    }                                    // |
  }                                      // |
  delay(100);                            // |
}                                        // |
```

### 5.3 Computer

The function of the computer is to read the data sent by the receiver and plot a real-time graph of the sensor values located on the Transmitter.

This is being done by using a MATLAB script which gets the string data in the serial port of the computer connected to the Receiver by USB.

```
clear all; close all; clc;
numSec=5000;
t=[];
v=[];

%s1 = serial('/dev/cu.usbmodem411');            % define serial port
s1 = serial('COM8');
s1.BaudRate=9600;                % define baud rate
set(s1, 'terminator', 'LF');     % define the terminator for println
```

```
fopen(s1);

try                                 % use try catch to ensure fclose
                                    % signal the arduino to start collection
w=fscanf(s1,'%s');                  % must define the input %d or %s, etc.
if (w=='A')
    display(['Collecting data']);
    fprintf(s1,'%s\n','A');         % establishContact just wants
                                    % something in the buffer
end

i=0;
t0=tic;
while (i<numSec)
    if(~isnan(fscanf(s1, '%d')))
        i = i+1;

        t(i)=toc(t0);
        t(i)=t(i)-t(1);
        v(i)=fscanf(s1,'%d');       % must define the input %d or %s, etc.
        v(i) = v(i)-4;
        disp(v(i));
        set(gcf, 'color', 'white');
        drawnow;
        if i>20
            tprint=t(i-20:i);
            vprint=v(i-20:i);
            plot(tprint,vprint);
            axis([tprint(1) tprint(21) 0 60]);
        else
            plot(t,v);
            axis([0 6.8153 0 60]);
        end

        grid on;
        title('V Test');
        xlabel('Time');
        ylabel('Input');

    end
end

catch me
    disp(me);
    fclose(s1);
end
```

# 6 Conclusions

At the end, we were able to obtain a couple of interesting conclusions from the development of our project.

The first one is regarding the radio transmission and reception. With our system, we achieved a clear transmission up to a distance around 15 meters from the transmitter to the receptor. As it was mentioned before, we could not manage to get the best radio modules or antennas, but despite of this, we were able to build a homemade antenna for the emitter and the results were satisfying. A possible improvement for our project in this sense could be to set up a better quality set of antennas, and power them with an additional power source so that we could be able to cover larger distances.

Also, we could easily implement a system to simultaneously send the information from different sensors at the same time. This could be really useful if we wanted to set up an entire telemetry system on board of an aircraft.

Finally, it should also be mentioned the possibility of making the communication a bi-directional process. This could be done with the same software used for this project. The main challenge would be to mount on both Arduino boards the transmitter and receiver modules. Also, the software for each board should be slightly different, given that each board should simultaneously send information and receive the data sent from the other. This would enable us to implement other devices such as servos or small engines, using the information available.

These are all improvements for future developments, but all based on the work done on this project. It was our main goal to set up a platform for a telemetry system, so that we could base more sophisticated systems using what we developed here.

## References


[1] Arduino Language Reference http://arduino.cc/en/Reference/HomePage

[2] Virtualwire Library http://www.seeedstudio.com/depot/images/product/VirtualWire.rar

[3] Humidity sensor tutorial http://tallerarduino.com/2012/12/24/sensor-dht11-humedad-y-temperatura-con-arduino/

[4] Temperature sensor TMP36 tutorial http://www.instructables.com/id/RF-315433-MHz-Transmitter-receiver-Module-and-Ardu/step3/Arduino-Virtual-Wire-Library/

[5] RF 433MHz Transmitter and Receiver Module Tutorial http://www.instructables.com/id/RF-315433-MHz-Transmitter-receiver-Module-and-Ardu/?lang=es

[6] Examples of sending messages in RF with Arduino http://www.buildcircuit.com/how-to-use-rf-module-with-arduino/

[7] Arduino transmitter information https://www.sparkfun.com/products/10535

[8] Transmitter Datasheet http://embedded-lab.com/uploads/datasheets/Tx_KST-TX01.pdf

[9] TMP36 Datasheet http://www.analog.com/static/imported-files/data_sheets/TMP35_36_37.pdf

[10] TMP36 Examples http://tallerarduino.com/2013/04/27/sensor-de-temperatura-tmp36-y-arduino/