DATA ACQUISITION

Project Report

Movable Temperature Sensing Unit. MTSU

By Preetika Jain, Chaitra Bhaskarachary, and Gari Ciodaro Guerra

Dec 9, 2019

Abstract

This document explains the development of the *Movable Sensing Unit (MSU)* as the final project for the course *Data Acquisition* fall 2019. MSU is a wireless controlled car that is constantly streaming sensing data to the internet. The controlling unit uses acceleration measurements from the hand of the user to command the car to move in tree degrees freedoms, namely forward, backwards, and axis rotation (both counter and clock-wise). MSU applications will depend on the sensor connected to it, in the present project, we used it to measure temperature, therefore use cases where focal temperature sensing is required and ambient conditions are harmful human are ideal for its use.

Key Words: Micro controllers, Arduino, Acceleration sensor, Temperature sensor, XBee Modules, WIFI Module, PHP, SQL.

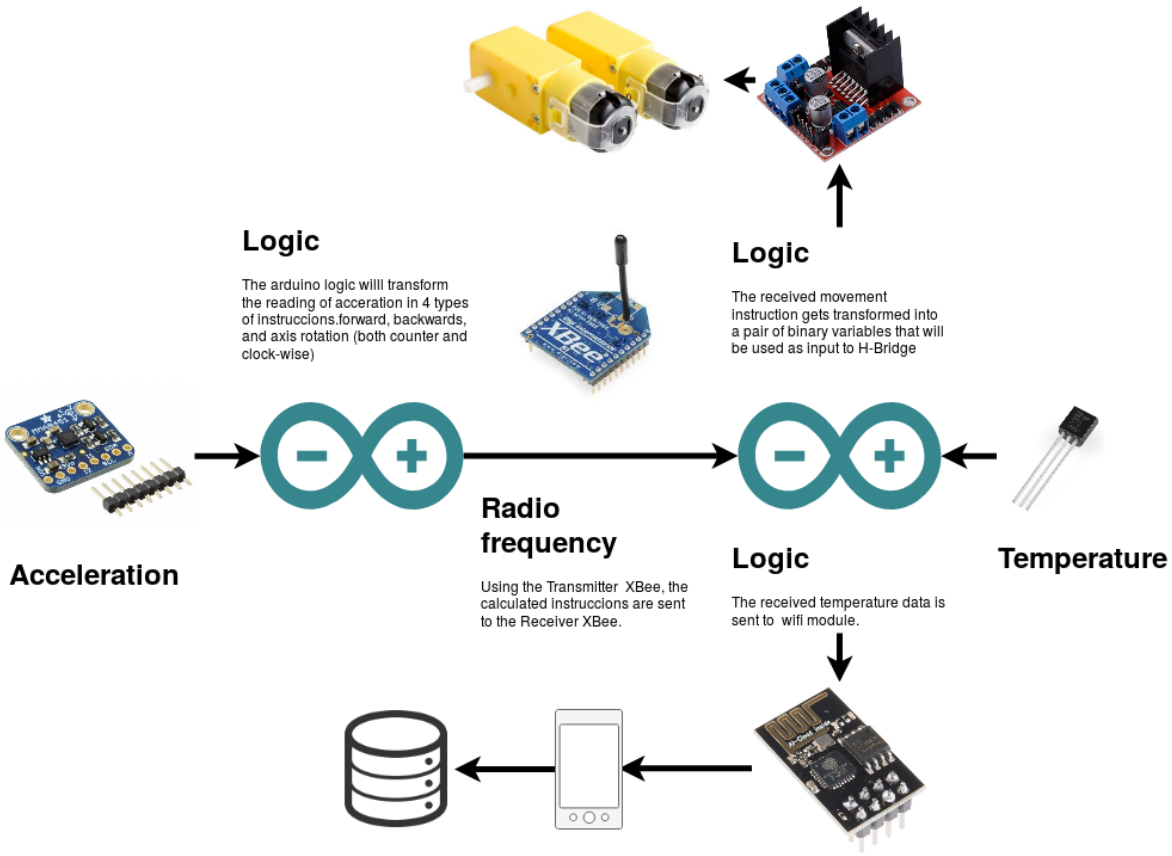

Figure 1: General diagram of MSU.

1 Introduction

In this era of technology, evolving domains such as Artificial Intelligence, Virtual and Augmented Reality, Machine Learning has brought Automation into light. Robot interaction plays an important role in human society due it is numerous advantages. Robot is a mechanical device designed in such a way that its functioning depends on the series of code which allows it to perform complex tasks automatically. A robot can be controlled in two ways. First, it can be controlled using an external device like remote and second, the controls are embedded within itself. Robots are divided into two categories, Autonomous and Semi-Autonomous or Human Controlled. Autonomous robots work automatically and makes its own decisions based on the code written and are not controlled by humans. On the other hand, as the name suggests Semi-Autonomous or Human controlled, are the robots controlled by human activities like Voice recognition, Face recognition, Motion Control, Touch recognition and others. One of the most common motion-controlled robots is Hand Gesture controlled Robot. This system plays an important role in human-robot interactions as the gestures made by humans are one of the most powerful way of communication. Some of the major areas of application includes remote surveillance, military, spying, to control the vehicles used by physically challenged people to move their wheelchairs etc.

Based on this idea of efficient hand recognition interaction with robot, the project is developed using two Funduino Uno Arduino which is a 8-bit microcontroller board, A 3-axis Accelerometer "MMA8451", used to find the direction of the triple axis (X,Y,Z). A Wireless connector used to transmit signals from transmitter to the receiver using two XBee modules. H-bridge, DC-Motors, Temperature sensor, WiFi module, Arduino IDE software to load the instructions to the Arduino microcontroller and Power Supply whose functioning and hardware descriptions are described in a Components section.

The sensor recognizes the hand gestures made, to move the robot car forward, backward, left and right and this bridging of interface between human and robot was implemented using two necessary elements called transmitter and receiver. Here, Transmitter refers to the human hand which send the data or signals to move the car in required directions and the receiver is the Robot Car. The working of the transmitter and the receiver systems are described in the later sections in this report.

2 Components

2.1 Arduino

Arduino is an open-source platform consisting of both hardware and software. Hardware component consists of circuit board referred as microcontroller and software being Arduino IDE, which is used to write and upload code to the microcontroller chip.

2.1.1 Hardware

Funduino Uno Arduino is a 8-bit microcontroller board which can be powered using a USB cable or with an external power supply. It is comprised of 14 digital input/output pins and 6 analog pins which can be used to connect various components like LED's, sensors, switches, etc.

2.1.2 Software

Arduino Integrated Development Environment(IDE) is used to connect to arduino hardware component by sending set of instructions via USB cable in C++ programming language to communicate with them. It mainly consists of two parts: editor and compiler. Editor is used to write the code and is called as sketch and the compiler is used to upload the code to the arduino microcontroller board. To allow serial communication on any of the pins on Uno SoftwareSerial library is used.

2.2 Accelerometer

Accelerometer is a device which is used for finding orientations in triple-axis(X,Y,Z) by measures acceleration or gravitational force. The accelerometer used here is MMA8451 miniature accelerometer with 14bit ADC resolution. It has built in orientation or tilt detection which helps us to identify if our hand is tilted forward, backward, left or right in our project. This sensor senses the axis of direction of movement of hand and the robotic car starts moving according to movement of the hand. It uses I2C bus for data transfer and it requires repeated-start I2C[5].

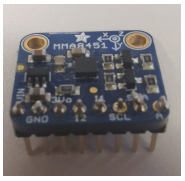

Figure 2: Accelerometer MMA8451

2.3 XBee S2C

XBee S2C module is designed for the exchange of data or wireless communication, which works on Zigbee mesh communication protocols. XBee modules are configured using X-CTU software developed by Digi International before setting up a mesh network to avoid failure[1]. It is better than Bluetooth device as it can transmit data for longer distance. We have used two XBee modules and configured them as transmitter and receiver in our project. The communication with arduino is facilitated by UART, the modules can be combined into the same personal area network(PAN) by configuring PAN ID to the same values. Each XBee has a unique 64-bits serial number on their module. We need to

configure the destination address of the transmitter to the receiver's serial number in order to send a message to a XBee receiver[2].

We have used Arduino pin 11 as its TX and pin 12 as RX, then UART data communication between Arduino and XBee module is connected as described in the table below :

XBee Shield	Arduino
5V	5V
GND	GND
RX (pin 0)	Pin 12 (SoftwareSerial pin RX)
TX (pin 1)	Pin 11 (SoftwareSerial pin TX)

Figure 3: Pin connection between XBee shield and Arduino

2.4 XBee Shield

Xbee Shield serves as an interface. The XBee module pins are too thick and cannot be plugged on to the breadboard so XBee is plugged to the XBee shield. This connection was inbuilt for us to use in our project. The Module connects to the sheet internally as described in the table below :

XBee	Arduino/XBee Shield
VCC (pin 1)	5V
GND (pin 10)	GND
DOUT (pin 2)	RX (pin 0)
DIN (pin3)	TX (pin 1)

Figure 4: Pin connection between XBee module and Arduino/XBee Shield

2.5 Temperature sensor

Temperature sensors are used to measure the temperature of a medium accurately and efficiently under a given set of requirements. It plays a significant role in various applications. In our project, we have used temperature sensor to measure and record the temperature at sensitive areas by sending the robotic car using our hand gestures to move the car in the required directions. These sensors are calibrated in Celsius, suitable for remote applications. While connecting a temperature sensor, one should

Figure 5: Temperature Sensor

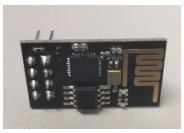

Figure 6: ESP8266 Wifi Module

take extra care about the polarity, else the sensor gets burn out. We can find the polarity by keeping the flat side facing towards us and pins facing down and the three terminals should be connected as :

- Left pin should be connected to 5V
- Right pin should be connected to Ground
- Middle pin receives the temperature signal and should be connected to A0 of the Arduino.

2.6 Wifi module

ESP8266 Wifi module is a standalone wireless transceiver which enables internet connectivity to embedded applications. It uses TCP/UDP communication protocol to connect to client or server Arduino communicates with wifi using UART with specific baud rate[4].

2.7 H-bridge

H-bridge is used to control the speed and direction of the two DC motors at the same time. The circuit consists of four switches to control the flow of power to the load. In order to reverse a motor, the power supply must be reversed, and this is accomplished using H-bridge. The typical usage of H-Bridge is in motor control applications.

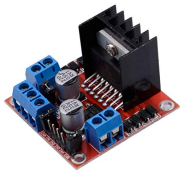

Figure 7: H-Bridge example

2.8 DC-Motors

Direct current (DC) Motors are one of the most important components used in our project. DC Motors uses direct electrical energy supplied by the battery to rotate the drive shaft. It has two terminals positive and negative. The motor rotates when these terminals are connected to a battery, if the terminals are switched, then motor rotates in the opposite direction. We have used two DC motors in our project.

2.9 Power supply

Power supply to the Arduino board can be done via USB connection or external power supply like battery or AC-to-DC adapter. The recommended range of power supply to board is 7-12 Volts. The

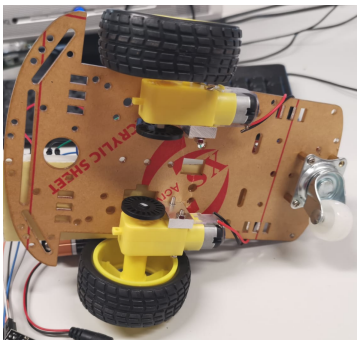

Figure 8: DC motors

Figure 9: 9v Battery

5V pin may supply less power and the board can be unstable and if more than 12V power is supplied, the voltage regulator may overheat and damage the board. The power pins are as follows :

- VIN : Input Voltage is supplied to the arduino through this pin.
- 5V : Outputs 5V from regulator to board. The board can be powered using DC power jack with 7-12 Volts, USB cable with 5V, or the VIN pin on board with 7-12 Volts.
- 3V3 : On board regulator generates a 3.3Volts
- GND : Pin which is used for grounding

The DC Motors used in our project are powered using 9V battery.

3 Connections

The Movable Sensing Unit (MSU) can be divided into tree main component: the arduino that sends instructions regarding movement to the vehicle(*The emitter*), the arduino that actually controls the vehicle and also senses the temperature(*The receiver*) and the data base that receives this information (*web interface*). Let us go through each of these components.

3.1 The Emitter

This component contains an arduino, the accelerometer, and the XBEE emitter. In code 1 we can see the logic inside this arduino. It sets the pin 11 and 12 as the emitter and the receiver of XBEE, these two channels has to be set up in the same manner in the other arduino (*The receiver*). Using the reading from the accelerometer decides whether to send instructions between still, forward-backward and rotations. It gives priority to forward-backward as can be seem in the last part of the code 1, where it will always send these instructions on that degree of freedom unless the y-acceleration goes outside some boundaries. The exact values for the *if* statement had to be calibrated manually using trial and error.

3.2 Receiver

This component contains an arduino, an H-bridge, XBEE receiver, two DC motors, one Wifi-module, Temperature-sensor and external power supply (9V battery).

3.2.1 XBEE receiver and H-bridge interaction

In code 2 we can see how the input from the receiver XBee module is transformed into four binary variables that serve as the input to the H-Bridge. The H-Bridge will decide based on the state of this four variables in which direction to spin each DC Motor; producing the desired direction of movement. When both motor spin in the same direction, we have forward or backward movement, when they spin in opposite directions, we have rotation.

3.2.2 Temperature sensor and Wifi module interaction

In code 3 we can see the initialization of the Wifi module. In the current setup it requires a Mobile phone (see 1) to serve as the Hotspot to get to the internet. Code 3 has to be run one time, since the credentials for the network will be stored in the internal memory of the wifi Module. Once the connection has been made, we proceed in reading and sending the data from the temperature sensor to the internet. We achieved this by using a HTTP POST request to the public server *garisplace.com*, specifically to the the end point */esppost.php*, an php file that will eventuality connect to the server database. Code 4 contains all this process. The principal loop can be observed in line 27. In line 83 we can see the most important auxiliary function, *sendRequest*, where we manually created the plain text of the HTTP protocol to send the data.

3.3 web interface

Once the POST request is sent to garisplace.com/esppost.php, the temperature gets deposit into a previously created data base. The php that handles the request in the server can be observed in code 5 where we first set up the parameter for the data base connection, that is, the DB name, the name

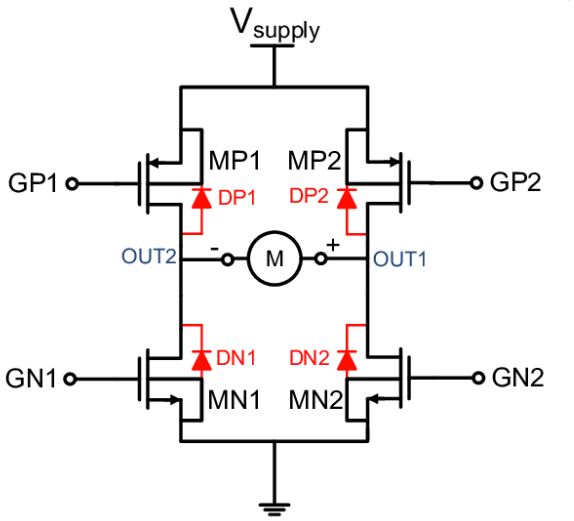

Figure 10: H-bridge Diagram taken from [3].

of the table, and the credentials to access it. The temperature value gets dumped in the php variable, that is later inserted into a *my sql insert statement*.

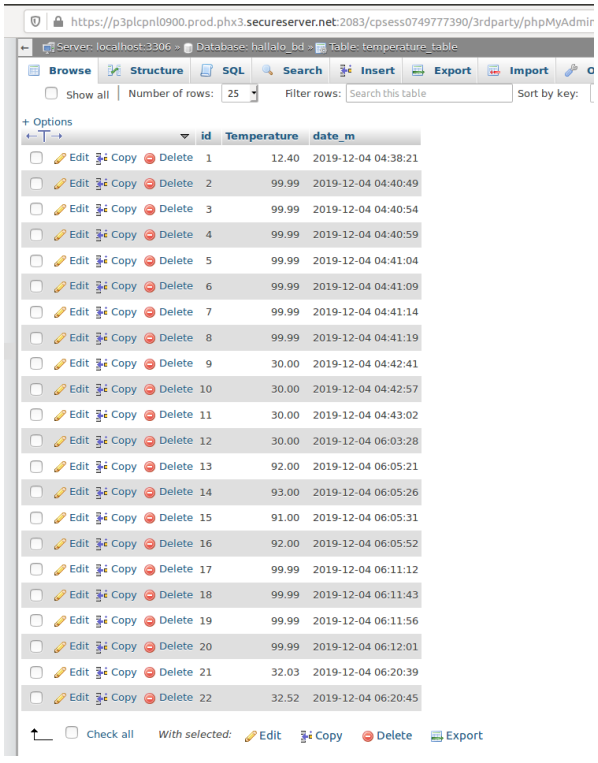

Figure 11: My sql data base.

4 Conclusion and Future Scope of Work

This project deals with the recording of the temperature data using a robotic car whose sense of direction is controlled by the hand gesture. The motive of this paper is to create a hand gestured controlled robot, which can detect the temperature of its surroundings and store the temperature data in the database. The initial step refers to the implementation of the transmitter, which is made using the Arduino, Xbee, Accelerometer, Power Supply. Second, creation of receiver is done using Arduino, Xbee, H-bridge, DC-motors, Power Supply, Temperature Sensor to sense the temperature of its surroundings and wifi module to transfer the temperature data to the database created. The respective codes are written and uploaded using Arduino IDE.

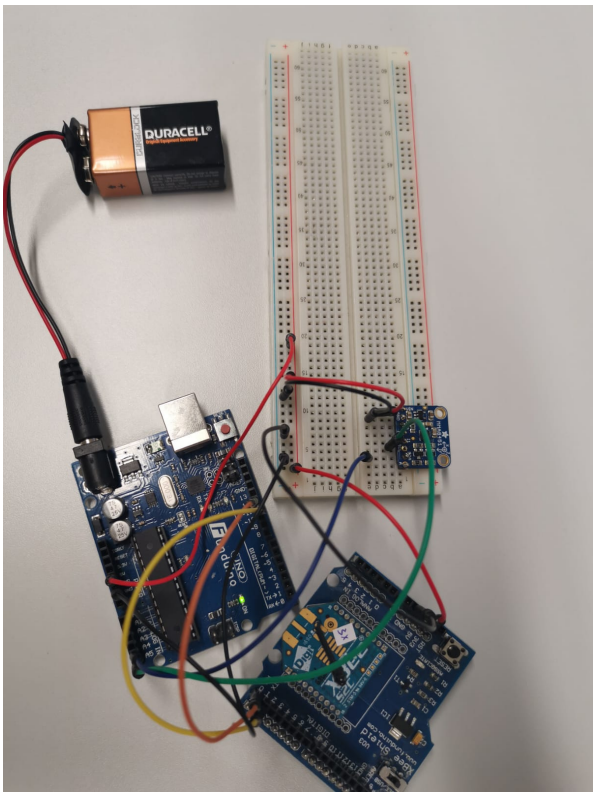

Figure 12: Full emitter

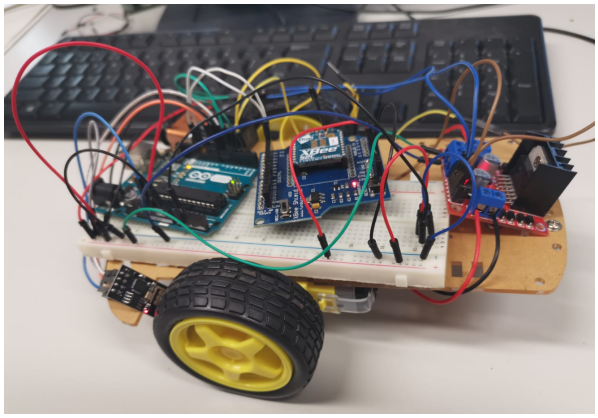

Figure 13: Full Receiver

Along with temperature sensor, we can also use Camera, Ultra Distance Sensor, Sound Sensor, Light Sensors to carry out the human computer interaction. This system could be used to send the robot to sensitive location and fetch the required data. The gestures could be captured, stored and could be used to train a model and apply machine learning to classify patterns could be considered as future scope of work.

Appendix A Codes.

Listing 1: **Emitter Arduino code.** Used to read acceleration values form the sensor, and send the calculated movement instruction to the vehicle using XBee.

```
#include <SoftwareSerial.h>
#include <Wire.h>
#include <Adafruit_MMA8451.h>
#include <Adafruit_Sensor.h>

Adafruit_MMA8451 mma = Adafruit_MMA8451();
char instruction_x= 's' ;
char instruction_y = 'k' ;

////wireless
short xBeeTx = 11;
short xBeeRx = 12;
SoftwareSerial xBeeSerial( xBeeRx , xBeeTx ) ;

void setup(void) {
  xBeeSerial.begin (9600) ;
  Serial.begin(9600);
  Serial.println("Adafruit MMA8451 test!");
  if (!mma.begin()) {
    Serial.println("Couldnt start");
    while (1);
  }
  Serial.println("MMA8451 found!");
  mma.setRange(MMA8451_RANGE_2_G);
  Serial.print("Range = "); Serial.print(2 << mma.getRange());
  Serial.println("G");
}

void loop() {
  // Read the 'raw' data in 14-bit counts
  mma.read();
  Serial.print("X:\t"); Serial.print(mma.x);
  Serial.print("\tY:\t"); Serial.print(mma.y);
  Serial.print("\tZ:\t"); Serial.print(mma.z);
  Serial.println();

  /* Get a new sensor event */
  sensors_event_t event;
  mma.getEvent(&event);

    // move forwards, negative value x
  if(mma.x<-1000){
    //F
    instruction_x='F';
    Serial.print(" instruction_x F \t");
   }

 else if (mma.x > -1000  && mma.x < 500){
    //digitalWrite(6,LOW);
    //digitalWrite(7,LOW);
    // s STOP_x
    instruction_x='s';
    Serial.print("instruction_x s \t");

    }
```

```
  else if (mma.x > 500) {
    // MOVING_BACKWARD
    instruction_x='B';
    Serial.print(" instruction_x B \t");
  }

  // move forwards, negative value y
  if(mma.y<-1000){
    // L  MOVING_LEFT
    instruction_y='L';
    Serial.print("instruction_y L \t");
   }

 else if (mma.y > -1000  && mma.y < 500){
  // STOP_y k
    instruction_y='k';
    Serial.print("instruction_y k \t");
    }
  else if (mma.y > 500) {
    // MOVING_RIGHT
    instruction_y='R';
    Serial.print("instruction_y R \t");
  }
  if(mma.y > -1000  && mma.y < 500){
    xBeeSerial.print(instruction_x) ;
    }
   else{
     xBeeSerial.print(instruction_y) ;
    }
}
```

Listing 2: **Receiver-control Arduino code.** Used to read the instructions sent by the transmitter to in term, control the H-bridge of the two DC Motors.

```
 # include <SoftwareSerial.h>

short pin_a_R = 2;
short pin_b_R = 3;
short pin_a_L = 4;
short pin_b_L = 5;
short xBeeTx = 11;
short xBeeRx = 12;
char xBeeSerial_read='s';

SoftwareSerial xBeeSerial(xBeeRx ,xBeeTx);
void setup(){
Serial.begin(9600) ;
xBeeSerial.begin(9600);
pinMode(pin_a_R,OUTPUT);
pinMode(pin_b_R,OUTPUT);
pinMode(pin_a_L,OUTPUT);
pinMode(pin_b_L,OUTPUT);
}

void loop(){
if (xBeeSerial.available()>0) {
  xBeeSerial_read = xBeeSerial.read();
  //Serial.print(instruction_x);
```

```
    if(xBeeSerial_read =='B'){
     digitalWrite (pin_a_R,HIGH);
     digitalWrite (pin_b_R,LOW);
     digitalWrite (pin_a_L,HIGH);
     digitalWrite (pin_b_L,LOW);
      }
    else if(xBeeSerial_read=='F'){
     digitalWrite (pin_a_R,LOW);
     digitalWrite (pin_b_R,HIGH);
     digitalWrite (pin_a_L,LOW);
     digitalWrite (pin_b_L,HIGH);
     }
    else if(xBeeSerial_read=='s'){
     digitalWrite (pin_a_R,LOW);
     digitalWrite (pin_b_R,LOW);
     digitalWrite (pin_a_L,LOW);
     digitalWrite (pin_b_L,LOW);
     }

  else if(xBeeSerial_read=='R'){
    digitalWrite (pin_a_R,HIGH);
    digitalWrite (pin_b_R,LOW);
    digitalWrite (pin_a_L,LOW);
    digitalWrite (pin_b_L,HIGH);
    }
  else if(xBeeSerial_read=='L'){
    digitalWrite (pin_a_R,LOW);
    digitalWrite (pin_b_R,HIGH);
    digitalWrite (pin_a_L,HIGH);
    digitalWrite (pin_b_L,LOW);
  }
  else if(xBeeSerial_read=='k'){
    digitalWrite (pin_a_R,LOW);
    digitalWrite (pin_b_R,LOW);
    digitalWrite (pin_a_L,LOW);
    digitalWrite (pin_b_L,LOW);
  }
  Serial.print(xBeeSerial_read);
  Serial.println();
  }
}
```

Listing 3: **Receiver. WIFI one time configuration.** Used to read to setup the network credential in order to get access to the internet. We connected to hotspot configured in a cell phone. The code used for configuration and usage of the WIFI module was extracted from here, thanks to Ralph Florent for a fantastic explanation.

```
#include <SoftwareSerial.h>

#define DEBUG true

short pinTx = 7; // use pin 7 as Arduino TX pin
short pinRx = 6; // use pin 6 as Arduino RX pin
short chpd = 8;

SoftwareSerial WIFISerial(pinRx, pinTx);

char LED = 13;
```

```
boolean LEDst = false;

void setup () {
  WIFISerial.begin(115200);
  Serial.begin(115200);
  pinMode(LED, OUTPUT);

  pinMode(chpd, OUTPUT);
  digitalWrite(chpd, LOW);
  delay(10);
  digitalWrite(chpd, HIGH);

  Serial.println("Initializing Wi-Fi Setup...");

  // send command to ESP
  sendWifiCommand("AT+RST\r\n", 1000, DEBUG); // reset module
  sendWifiCommand("AT+CWLAP\r\n", 2000, DEBUG); // List of access points
  sendWifiCommand("AT+CWQAP\r\n", 2000, DEBUG);
  // Set connection to Access Point
  sendWifiCommand("AT+CWJAP=\"gari_phone\",\"xxxxxx\"\r\n", 5000, DEBUG);
  sendWifiCommand("AT+CIPSTA=\"192.168.43.10\"\r\n", 2000, DEBUG); // Set static IP

  sendWifiCommand("AT+CWMODE=3\r\n", 1000, DEBUG);
  sendWifiCommand("AT+CIFSR\r\n", 1000, DEBUG); // Display IP address
}

void loop () {
//  while (WIFISerial.available() > 0) {
//    char a = WIFISerial.read();
//    Serial.println("Wi-Fi just received: " + a);
//    digitalWrite(LED, LEDst = !LEDst);
//  }
}

String sendWifiCommand(String command, const int timeout, boolean debug) {
  String response = "";
  long int time = millis();

  WIFISerial.print(command); // send the read character to the esp8266

  while ((time + timeout) > millis()) {
    while (WIFISerial.available()) {
      // The esp has data so display its output to the serial window
      char c = WIFISerial.read(); // read the next character.
      response += c;
    }
  }

  if (debug) {
    Serial.print(response);
  }
  return response;
}
```

Listing 4: **Receiver. WIFI and temperature sensor.** Used to read to read the temperature data from the sensor. Connect to a WIFI network, and send an HTTP POST request to the server, so that the measurements of temperature are stored in a database.

```
#include <SoftwareSerial.h>

#define DEBUG       true
#define LED         13
#define CH_PD       4
#define BAUDRATE    115200
#define PIN_TX      7
#define PIN_RX      6

#define WIFI_IP     "192.168.43.10"
#define SERVER_API  "garisplace.com"
#define SERVER_PORT "80"
#define ENDPOINT    "/esppost.php"

int pinTemp = A1;

SoftwareSerial WIFISerial(PIN_RX, PIN_TX);

void setup () {
  pinMode(LED, OUTPUT);
  digitalWrite(LED, HIGH);
  Serial.begin(BAUDRATE);

  setupWifi();
}

void loop () {
  int reading  = analogRead(pinTemp);    //Read the analog pin
  float voltage = reading * 5.0;
  voltage /= 1024.0;
  float temperatureC = (voltage - 0.5) * 100 ;
  String string_temp;
  string_temp = String(temperatureC);
  Serial.println("Temperature: " +string_temp);

   if (connect(0, SERVER_API)) {
    String query = "?temp="+string_temp;
    const int connectionId = 0;
    bool isSent = sendRequest("POST", ENDPOINT + query, "{}", connectionId, SERVER_API);
    if (isSent) {
      if (DEBUG) Serial.println("Request has been sent successfully!");
    } else {
      if (DEBUG) Serial.println("Request couldn't be sent...");
    }
    disconnect(0);

  } else {
    if (DEBUG) Serial.println("Connection to the server " + String(SERVER_API) + " failed!");
  }
}

bool connect(int connectionId, String server) {
  String cipStart = "AT+CIPSTART=" +
                                  String(connectionId)
                                  + ",\"TCP\",\"" + server + "\"," + SERVER_PORT;
  WIFISerial.println(cipStart);
  if (DEBUG) Serial.println(cipStart);

  delay(1000);

  if (WIFISerial.available())
```

```
    if (WIFISerial.find("OK"))
      return true;

  return false;
}
bool disconnect(int connectionId) {
  String closeCmd = "AT+CIPCLOSE=" + String(connectionId);
  WIFISerial.println(closeCmd);
  if (DEBUG) Serial.println(closeCmd);
  delay(1000);

  if (WIFISerial.available() > 0) {
    if (WIFISerial.find("CLOSED")) {
      clearBuffer();
      return true;
    }
  }
  return false;
}


bool sendRequest(String method, String resource, String queryParams, int connectionId, String server) {
  bool isPost = false;

  // Prepare command to send http request
  String cipCmd = "";
  cipCmd += method + " " + resource + " HTTP/1.1\r\n";
  cipCmd += "Host: " + server + ":" + SERVER_PORT + "\r\n";
  cipCmd += "Content-Type: application/json\r\n";

  if (method == "POST" && queryParams != NULL) {
    cipCmd += "Content-Length: " + String(queryParams.length()) + "\r\n";
    cipCmd += "Connection: close\r\n\r\n";
    cipCmd += queryParams; // add json query parameters to the request
    isPost = true;
  } else {
    cipCmd += "Connection: close\r\n\r\n";
  }

  // First, establish an HTTP connection/session
  String cipSend = "AT+CIPSEND=" + String(connectionId) + "," + cipCmd.length();
  WIFISerial.println(cipSend);
  if (DEBUG) Serial.println(cipSend);
  delay(1000); // wait until request is sent

  // Now try to send the HTTP request
  if (WIFISerial.available()) {
    if (WIFISerial.find("OK")) {
      if (isPost) {
        sendWifiCommand(cipCmd, 1000, DEBUG);
      } else {
        sendWifiCommand(cipCmd, 1000, DEBUG);
      }
      if (DEBUG) Serial.println(cipCmd);
      return true;
    }
  }

  return false;
}

void clearBuffer() {
```

```
  while (WIFISerial.available()) {
    char a = WIFISerial.read();
    if (DEBUG) Serial.write(a);
  }
}

void setupWifi() {
  WIFISerial.begin(BAUDRATE);

  pinMode(CH_PD, OUTPUT);
  digitalWrite(CH_PD, LOW);
  delay(10);
  digitalWrite(CH_PD, HIGH);

  if (DEBUG) Serial.println("Initializing Wi-Fi setup...");
  sendWifiCommand("AT+CIPSTA=\"" + String(WIFI_IP) + "\"\r\n", 2000, DEBUG);
  sendWifiCommand("AT+CIFSR\r\n", 3000, DEBUG); // display ip address
  sendWifiCommand("AT+CIPMUX=1\r\n", 1000, DEBUG); // configure for multiple connections
  sendWifiCommand("AT+SLEEP=0\r\n", 500, DEBUG); // disable sleep mode
  if (DEBUG) Serial.println("WIFI setup done!");
}

String sendWifiCommand(String command, const int timeout, boolean debug) {
  String response = "";
  long int time = millis();

  WIFISerial.print(command); // send the read character to the esp8266

  while ((time + timeout) > millis()) {
    while (WIFISerial.available()) {
      //if(WIFISerial.find("data")) digitalWrite(LED, LOW);
      // The esp has data so display its output to the serial window
      char c = WIFISerial.read(); // read the next character.
      response += c;
    }
  }

  if (debug) Serial.print(response);

  return response;
}
```

Listing 5: **PHP file on the server side (esppost.php).** Used to read the request sent by the WIFI module and write the temperature measurement to the data base.

```
<?php
define('DB_NAME','hallalo_bd');
define('DB_USER','gari');
define('DB_PASSWORD','XXXXX');
define('DB_HOST','localhost');
$link = mysql_connect(DB_HOST, DB_USER, DB_PASSWORD);
if (!$link){
        die('could not connect:' . mysql_error());
}
$db_selected = mysql_select_db(DB_NAME, $link);
if (!$db_selected){
die('Can\'t use' . DB_NAME . ': ' . mysql_error());
}
```

```
14  $Temp = $_REQUEST["temp"];
15  if(!empty($Temp))
16  {
17   if(!$insert = mysql_query("INSERT INTO `temperature_table` (Temperature)
18          VALUES ('$Temp')"))
19                  echo "Unable to register the temperature";
20          else{
21                  echo "The temperature has been registered";
22          }
23  }
24  mysql_close();
25  exit();
26  ?>
```

References

[1] Components101. *XBee S2C Module Pinout, Features & Datasheet.* 2019. URL: https://components101.com/wireless/xbee-s2c-module-pinout-datasheet (visited on 12/04/2019).

[2] Fangning Dr.Hu. *Data Acquisition and Sensor Network: Lab Manual Fall 2019.* 2019. URL: https://www.jacobs-university.de/directory/fhu (visited on 12/09/2019).

[3] Yasser Yousry Moursy. "A methodology for analysis and verification of the substrate noise coupling in HV/HT integrated circuits for automotive applications." PhD thesis. 2016.

[4] Electronicwings.com. (n.d.). *ESP8266 WiFi Module — Sensors & Modules.* 2019. URL: https://www.electronicwings.com/sensors-modules/esp8266-wifi-module (visited on 12/04/2019).

[5] Adafruit Learning System. *Adafruit MMA8451 Accelerometer Breakout.* 2019. URL: https://learn.adafruit.com/adafruit-mma8451-accelerometer-breakout?view=all (visited on 12/04/2019).