EE/CE1100 Introduction to Electrical and

Computer Engineering

Semester Project Lab Report

Laboratory Team Members: Bharathy Babu and Lecture Section #1100.002, Ethan Hull # CE-1100.001, and Ahmad Nabizadah Lecture Section #1100.001

Instructor: James M. Florence

Date: December 9, 2022

Abstract

For this project we created a Robotic Arm and utilized online resources to print 3D parts. We used pre-existing knowledge to create radio communication. With this knowledge we created a robotic arm that can be controlled wirelessly through radio communication using potentiometers. We accomplished this by implementing our TinkerCAD design and performing extensive troubleshooting. At the end we successfully built the robotic arm and controlled it using the potentiometers.

Introduction

In this project we wanted to create a robotic arm that uses radio communication. The motivation for this project was because we wanted to create a project that would test our understanding of electronics and programming. We also wanted to build something new, challenge ourselves, and learn along the way. In this project, we implemented radio communication and servo motors to control a 3D printed robotic arm. The reason why we used a radio communication device than an IR communication device was that a radio communication device can send precise data (specific mapped values of the potentiometers/servos) in comparison to IR which can only receive 0 (off) and 1 (on) wave. Servos as well is connected to a digital PWM pins on the Arduino which can control the servo to move to specific degrees.

Some special design features of this project are that it will include a 3D printed materials in order to create the robotic arm and it will also use a PCB board for the connection of the servos on the robotic arm to connect easily to the receiving radio communication device.

The expected result of this project is to create a fully functional robotic arm with radio communication that can more in sync with the potentiometers.

Design and Analysis

Materials:

The materials used for this lab report are 3D printer(type), 3D printing material, 3 standard servo (type), 3 micro servos (type), male to female wires, screws, nuts, screwdriver, rubber band, 6 5V regulator, diode, breadboard, 5 potentiometers, 8 V – 10 V battery, wires, 2 Arduino nanos, radio module transmitter, and radio module receiver.

Robot Arm Design and Analysis:

The robotic arm design was based on previous designs and used grad CAD in order to create an efficient and working robotic arm. The previous designs of the robotic arm were a 3-axis robotic arm that was created by 3 micro servos and wood that was jig sawed. This design was very flawed as the robotic arm couldn't pick anything up because it was too light.

3-axis robotic arm with 3 micro servos

Figure 1: Previous Arm design

We decided that we needed a bigger servo in order to pick up objects. So, we checked grab cad website to find any CAD models that we can use for our robotic arm. We found two CAD models that we could use, one was by Robotic Arm Magnibot by HOW TO MECHATRONICS and another was by David Fernandez.

Robotic Arm Design 1

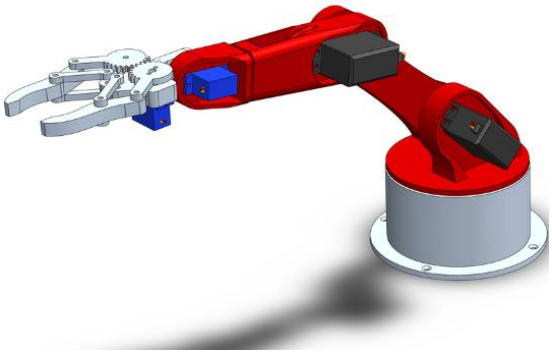

Figure 2: Magnibot model by HOW TO MECHATRONICS in grab cad

This robotic arm uses 3 standard servos and 3 micro servos, both types of servos can move to 0 degrees to 180 degrees. The advantage of this design is that you can pick objects up at any angle. Furthermore, this design will be easy to make with 3D printing and assembly is also easy as well. One con of this design was that the load that the second motor would have to carry might be too much and create a short circuit if the load is too much for it.

Robotic Arm Design 2

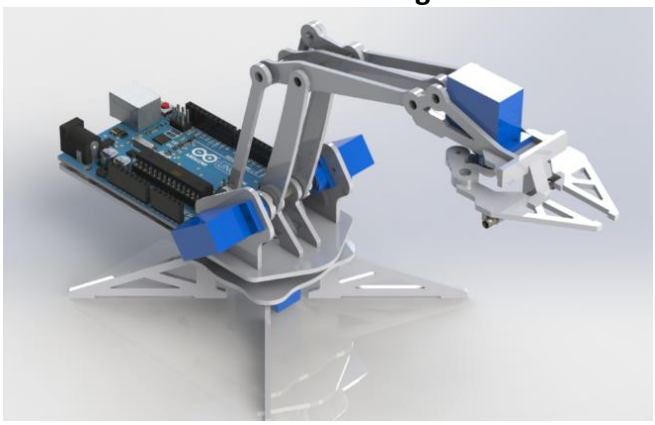

Figure 3: Robotic Arm by David Fernandez

This robotic arm uses 4 standard servos and many special parts for assembly, this makes the assembly hard. Which can be difficult for the engineer, another con for this design is that it can only pick objects up in Parrell with the arm. Unlike the first robotic Arm design. A positive thing about this design is it only uses 4 servos unlike the Robotic Arm Design 1 which uses 6 servos.

Chosen Design:

We decided to go with Robotic Arm Design 1 as shown in figure 2 because it was easy to 3D print and assemble with the available time. A way we decided to remove the con from the first design was that we would add 5-volt regulators with the servos, and we would add a rubber band to easy the load to the second servo on the 1st Robotic Arm Design.

Remote Control/Transmitter Circuit Design and Analysis:

The transmitter circuit for our Robot Arm is made from an Arduino Nano, a large breadboard, 5 potentiometers, and 2 8-pin headers for the radio. The radio will transmit whatever input we make on the potentiometers to the receiver circuit. As shown in Figure 5 the headers are connected to 3.3V, GND and digital pins 13, 12, 11, 8, and 7 on the Arduino. The potentiometers are connected to A0 to A4. Each potentiometer controls a certain part of the Robot arm.

Transmitter Circuit Design

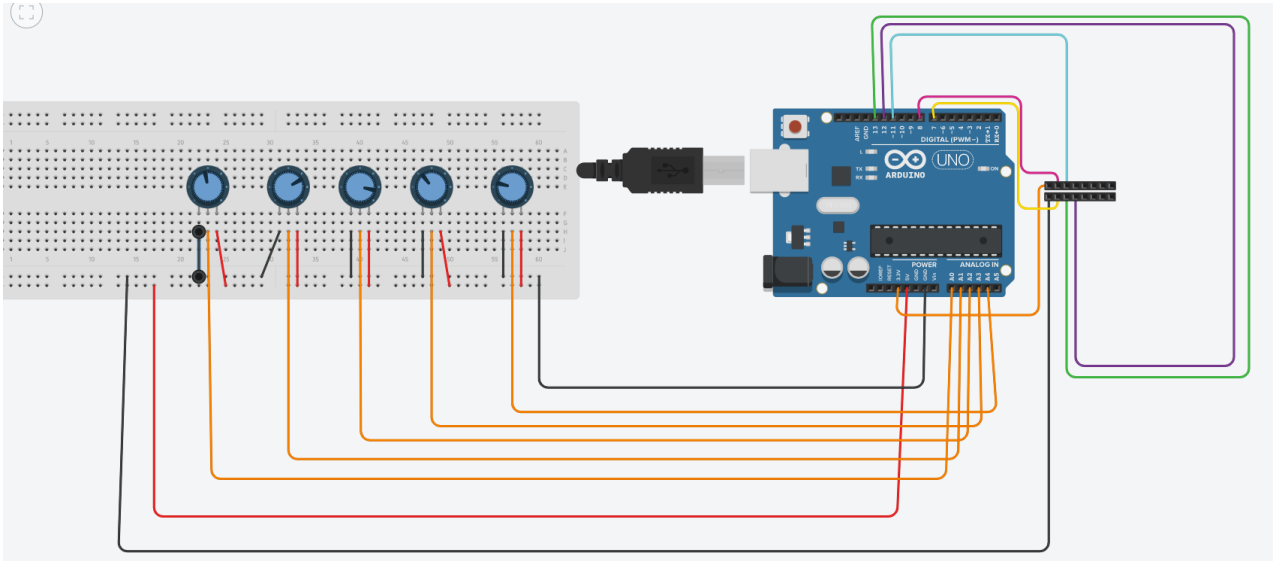

Figure 5: Transmitter Circuit of the Robot arm modeled in TinkerCAD

First in the program we install the libraries for the radio module which is nRF24L0.h, RF24.h, and SPI.h. These libraries are important for the program to work. Then we create a data structure to store the values of each potentiometer separately. We then state that value at A0 to A4 (which are connected to potentiometer) are equal to the data.pot0 - data.pot4. This means whenever the potentiometer value changes it also changes the data.pot0 - data.pot4 depending on which potentiometer is changing. Then we map the values of the potentiometer to 0 to 180. This is important because the potentiometer originally stores the values from 0 to 1023 and that needs to change because the servo values can only store values from 0 to 180. Then finally the radio transmitter will send the servo values to the receiver program.

Receiver Circuit Desing and Analysis:

The receiver circuit of our Robot Arm consisted of an Arduino Nano, a 5V regulator for each of the six servo-motors, a protective diode, an NRF24L01 radio module, and eight 1.2V NiMH batteries connected in series to power the board and the motors as shown in Figure 6. To movement of the robot arm is controlled using three regular servo motors (black) and three micro servo motors (blue).

Although our TinkerCAD model shows an Arduino uno, we used an Arduino nano to reduce the size of our circuit. TinkerCAD did not have an Arduino nano model, so we used the Arduino uno and pretended as if it was a nano. To power each motor, we used six 5-volt regulators, one for each motor. The reason we did not connect all motors to the Arduino's 5V pin is because of the current limitations. The 5V pin of the Arduino can only supply a maximum of 500mA, which is far less than what we need. It is not even enough to supply one of the micro servos. The datasheet of the micro servo's states a maximum stall current of 750 mA, which the Arduino 5V can't support. The regular size servos have an even higher stall current at about 1.5 Amps. So, to overcome this challenge, we used six 5V 1.5 Amps max regulators, one for each servo motor. Each servo should have enough current supply.

Receiver Circuit Design

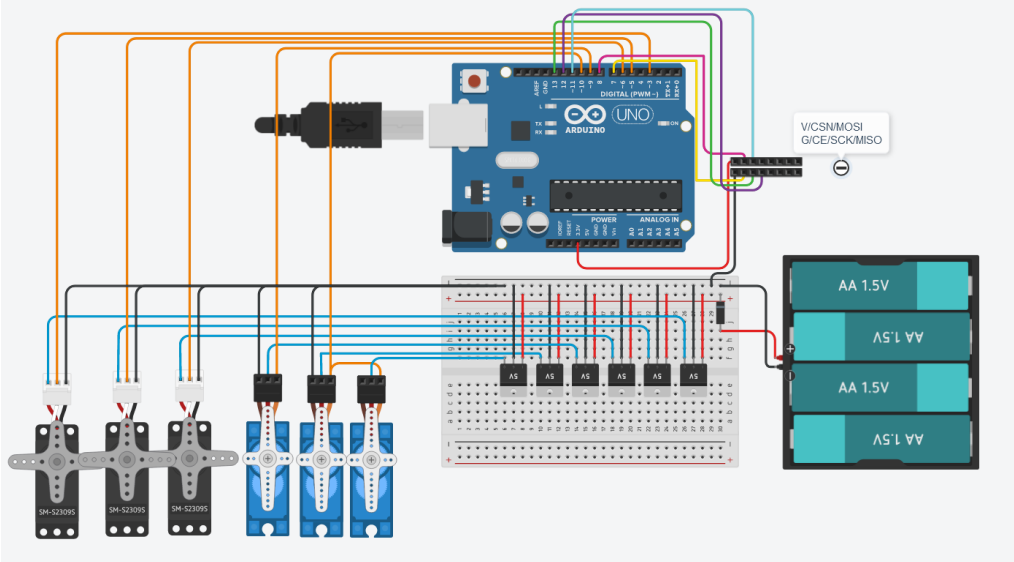

Figure 6: Receiver circuit of the Robot arm Modeled in TinkerCAD

All six regulators are connected in parallel to the power supply. The first two pins are the input and ground connecting to the battery, and the third pin is the 5V output connecting to the 5V pin (red) of the servo motors. The brown/black pin of the servo motors are connected to ground, and the yellow/white pins are the signal pins connected to PWM pins of the Arduino: digital pin 3, 5, 6, 9, and 10. As shown in Figure 6, we have 6 servo motors but only 5 PWM pins available on the Arduino. The 6th PWM pin (Pin 11) is used by the radio module. We could've used an Arduino mega, which has plenty of PWM pins. However, we did not have the time and resources, so we connected the last two micro servos (used for the claw of the robot) to the same PWM pins, which makes them synchronous.

The power supply consists of eight rechargeable 1.2V NiMH batteries connected in series, which supplies a voltage of around 11 volts when fully charged. However, I learned that voltage regulators build a lot of heat if the difference between Vin and Vout is high, it also becomes inefficient. So, we changed the power supply from eight to six batteries connected in series. This gave us a voltage of a little over 8V when fully charged, which is sufficient for the voltage regulators. The batteries have a discharge current capability of 2C, which should be able to safely supply up to 4 Amps. We don't expect the robot to draw more than a total of 1.5-2Amps of continues current, so we have a good safety margin. The battery is connected to a diode which protects the electronic components in case the battery is reverse-biased.

We decided to implement wireless communication between the receiver and transmitter circuits. So, we used an NRF24L01 wireless radio module. It is a popular transceiver module that is used with Arduinos. It uses SPI communication and only requires connections to 7 pins of the Arduino: ground, 3.3V, digital pins 7, 8, 11, 12, and 13. We used the NRF24 library to establish communication and transmit/receive data. The transmitter circuit reads the value of 5 potentiometers that are used for controlling the motors, maps it between 0-180 (rotation range of the servo motors) and sends it to the receiver in a data package. The receiver receives the data package and writes each potentiometer values to their corresponding PWM pins of the Arduino, which will rotate the motors accordingly.

Results

We were able to successfully implement our design and build the transmitter, receiver, and the robot arm. We 3D printed the model for our robot. Figure 7 shows all parts of the robot 3D print and assembled. The servo motors hold the arm together and are used for rotation of the joints.

3D Robotic Arm with Servos

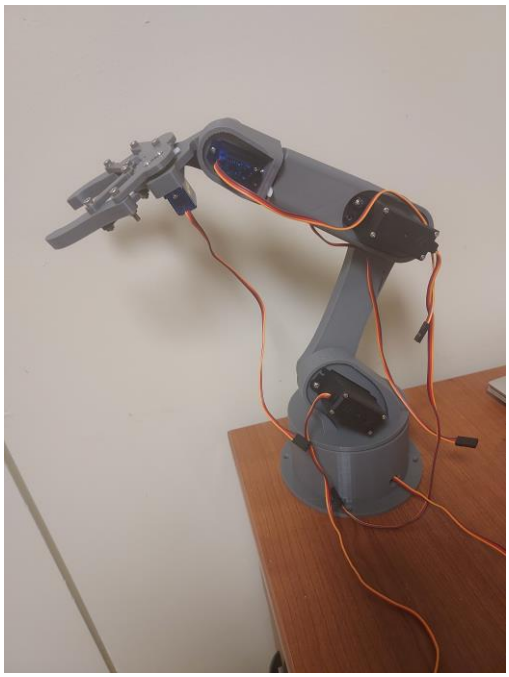

Figure 7: Finished assembling the 3D parts of the robotic with the Servos

Figures 7.1 and 7.2 show the front and the back of the receiver circuit. It consists of six three-pin male headers, one for each servo motor. The pins aligned in a zig-zag shape to save space and prevent misconnections. The battery connects to the two-pin male header located at the top right corner. These figures show the special characteristics of this project as it included a PCB which connected to the robotic arm.

Receiver Circuit Front

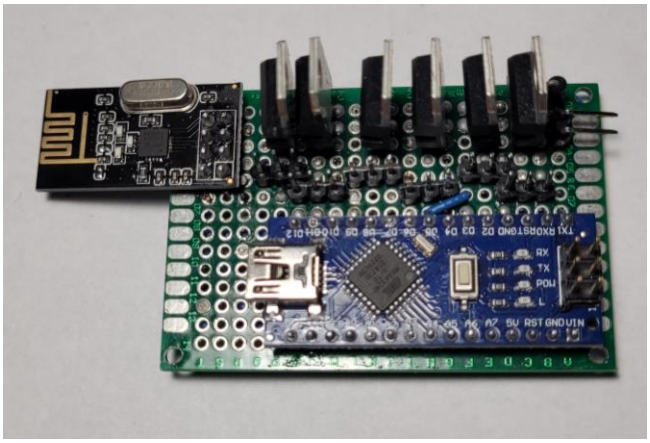

Figure 7.1: Receiver circuit front side.

Receiver Circuit Back

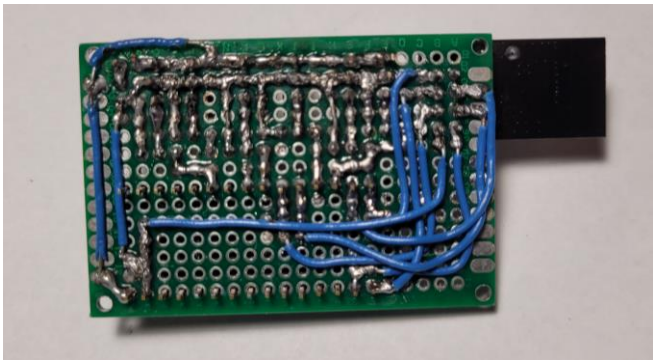

Figure 7.2: Receiver circuit backside.

Figure 8 shows the robot powered on and connected the battery. The robot was demonstrated in the lab on Wednesday December 7th.

Robot Arm Energized

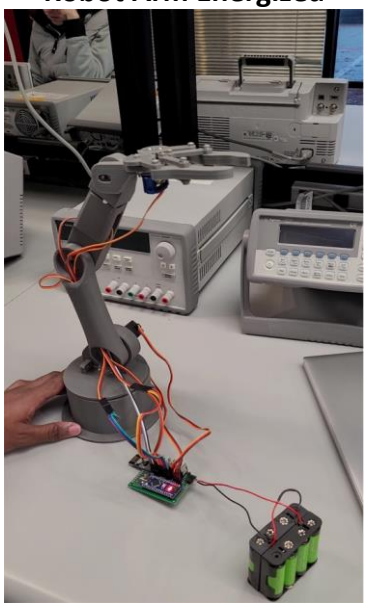

Figure 8: Robot arm is connected to the receiver circuit, powered up, and demonstrated in the lab.

Another special component of this project was the radio communication devices that were used in Figure 9 and 7.1 and 7.2. These radio communication devices were able to send analog signals to the receiver for the servo to move to a position by the sent analog signal by the transmitter which is controlled by the potentiometers.

Figure 9 shows the transmitter circuit which consists of 5 potentiometers and a radio module connected to the Arduino. We powered the transmitter using a USB cable connected to our Laptop.

Transmitter with potentiometers

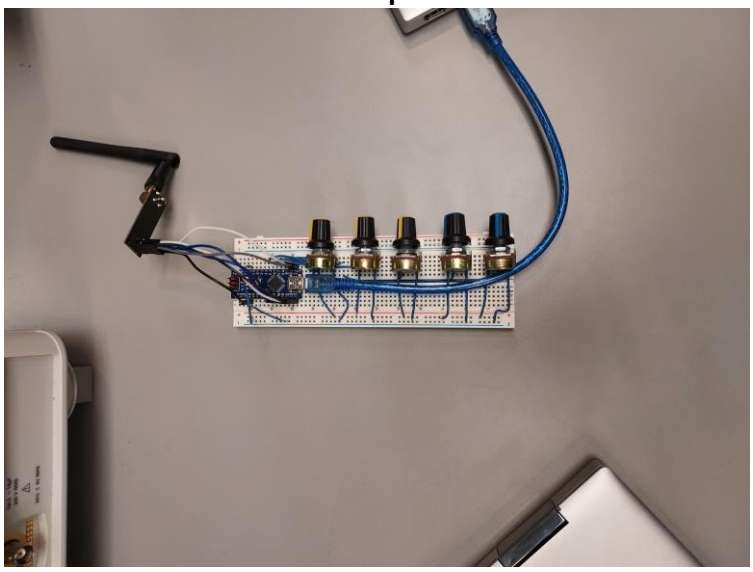

Figure 9: The transmitter is connected to the potentiometers which will send the mapped values to the robotic arm

However, it is noteworthy that we encountered lots of problems while building the robot arm. Our first issue was communication issue. The receiver and transmitter circuits were able to communicate without issue for as long as the servo motors are not connected. As soon as we connected the motors, communication would stop. We suspected a low battery voltage. We took our robot the UTD makerspace and checked the battery voltage, and it turned out to be very low. It turned out that connecting the servos dropped the voltage even lower, to the point where it was not sufficient for the Arduino and the radio module. We used a DC voltage generator to power the robot and perform our tests. The problem we encountered next was very rough motor movements. The servo motors were stuttering and shaking quite a lot. The movement was not smooth at all. After some troubleshooting, we figured out the current limit on the DC voltage generator was 250mA, which is not enough to power the servo motors. We increased the current limit to 1.5Ams (max) and the servos worked without issue.

Next, we encountered intermittent communication stops and interruptions. We noticed that the jumper wires we were using were loose and would cause noise/interruptions. We replaced the jumper wires with aluminum wires (small in diameter) and cut them to the right lengths as shown in Figure 9. This gave our circuit much more stability.

We also experienced a problem with the 5-volt regulators being burnt out specifically for the second servo. This was because the amount of load on the servo was very high meaning that the amount of current that passed through the 5V regulator is high which can burn the 5V regulator which will then create an open circuit. Another reason why the 5-volt regulator burnt was that the voltage drop from 10V to 5V can cause a lot of strain on the regulator which then can also burn up the 5V regulator. In result, a solution for the next time we do this project is that we could reduce the input voltage to the 5V regulators.

Discussion and Conclusions

We were able to successfully create a working robotic arm that was able to move in every direction. However, we did have difficulty in picking up objects because the last two servos were connected synchronously due to Arduino nano/uno not having enough PWM pins. Some changes would be to use an Arduino Mega so we could have more PWM pins that we can work with. Another addition would be attaching the arm to a base by screwing it down to a wooden board so there would be support. We can also use better voltage regulators that can supply a larger amount of current without overheating. In addition, we can add extra sensors to the robot arm and program it to pick objects from one point to another. We can also control the robotic arm by a smartphone or more user-friendly devices than a potentiometer.

References/Appendices

Appendices:

Transmitter TinkerCAD model: Ethan Hull
Receiver TinkerCAD Model: Ahmad Nabizadah
Receiver Circuit: Ahmad Nabizadah
Transmitter Circuit: Ethan Hull & Bharathy Babu
Robotic Arm: Bharathy Babu
Troubleshooting: Ahmad Nabizadah & Bharathy Babu
Arduino Program: Ahmad Nabizadah & Bharathy Babu

Abstract - Everyone
Introduction - Everyone
Design and Anayses - Everyone
Results - Everyone
Discussion and Conclusions- Everyone

https://grabcad.com/library/robot-arm-43

https://grabcad.com/library/robotic-arm-magnibot-1

https://howtomechatronics.com/tutorials/arduino/diy-arduino-robot-arm-with-smartphone-control/

Transmitter Code:

```
//Libraries for Radio communication
#include <nRF24L01.h>
#include <RF24.h>
#include <SPI.h>

//Radio module CE pin (D7) and CSN pin (D8)
RF24 radio(7, 8); // RF24 CE, CSN

//Address of the transmitter. Same as the reciver
const byte address[6] = "01010";

//Structure to store the potentiometer values.
struct Data_Package {
  byte pot0;
  byte pot1;
  byte pot2;
  byte pot3;
  byte pot4;
  };

// Create a data_package structure named "data"
Data_Package data;


void setup() {
  //Begin radio communication.
  radio.begin();
  radio.openWritingPipe(address);    //Set the address of the writing(transmitting) pipe. Same address as the receiver reading pipe
  radio.setPALevel(RF24_PA_MIN);    //Range set to minimum for max stability.
  radio.stopListening();          //StopListening sets the modules as transmitter

  //Define data package values with an initial value.
  data.pot0 = 0;
  data.pot1 = 0;
  data.pot2 = 0;
  data.pot3 = 0;
  data.pot4 = 0;

}
```

```
void loop() {
  //Read and store each potentiometer's value.
  data.pot0 = read_A0();
  data.pot1 = read_A1();
  data.pot2 = read_A2();
  data.pot3 = read_A3();
  data.pot4 = read_A4();
  //Serial.println(data.pot4);

  //Transmit the data package.
  radio.write(&data, sizeof(Data_Package));
}

//**********************FUNCTION IMPLEMENTATION************************//

// Read potentiometer connected to the the analog pin AX
int read_A0(){
  int pot_value = analogRead(A0);
  // Map the numbers from 0, 1023 to 0, 255 beccause thats what a byte can hold
  pot_value = map(pot_value, 0, 1023, 0, 180);
  return pot_value;
  }

// Read potentiometer connected to the the analog pin AX
int read_A1(){
  int pot_value = analogRead(A1);
  // Map the numbers from 0, 1023 to 0, 255 beccause thats what a byte can hold
  pot_value = map(pot_value, 0, 1023, 0, 180);
  return pot_value;
  }

// Read potentiometer connected to the the analog pin AX
int read_A2(){
  int pot_value = analogRead(A2);
  // Map the numbers from 0, 1023 to 0, 255 beccause thats what a byte can hold
  pot_value = map(pot_value, 0, 1023, 0, 180);
  return pot_value;
  }

// Read potentiometer connected to the the analog pin AX
int read_A3(){
  int pot_value = analogRead(A3);
  // Map the numbers from 0, 1023 to 0, 255 beccause thats what a byte can hold
```

```
  pot_value = map(pot_value, 0, 1023, 0, 180);
  return pot_value;
  }

// Read potentiometer connected to the the analog pin AX
int read_A4(){
  int pot_value = analogRead(A4);
  // Map the numbers from 0, 1023 to 0, 255 beccause thats what a byte can hold
  pot_value = map(pot_value, 0, 1023, 0, 180);
  return pot_value;
  }
```

Receiver Code:

```
#include <nRF24L01.h>
#include <RF24.h>
#include <SPI.h>
#include<Servo.h>

const int noOfServos = 5; //We are using 5 PWM pins for servos.
// Create servo variables.
Servo myservo0;
Servo myservo1;
Servo myservo2;
Servo myservo3;
Servo myservo4;

//Radio module CE, and CSN pin. CE pin is digital 7, CSN is 8.
RF24 radio(7, 8); // RF24 CE, CSN

//Radio address
//The reciver and the transmitter must have the same address to communicate.
const byte address[6] = "01010";

//Structure to store the potentiomenter values.
struct Data_Package {
  byte pot0;
  byte pot1;
  byte pot2;
  byte pot3;
  byte pot4;
  };
```

```
// create a data package structre named 'data'
Data_Package data;

void setup() {

  //Begin radio communication.
  radio.begin();
  radio.openReadingPipe(0, address);  //Open a reading Pipe with the same address as the transmitters writing pipe
  radio.setPALevel(RF24_PA_MIN);        //Sets the range of the radio. We want it to be minimum for max stablility
  radio.startListening();            //Startlistening will define the radio module as a reciver.

  //Define data package values with an initial value.
  data.pot0 = 0;
  data.pot1 = 0;
  data.pot2 = 0;
  data.pot3 = 0;
  data.pot4 = 0;

  //Attach each servo to a PWM pin and define the range from 0 to 180 degrees.
  myservo0.attach(3, 0, 180);
  myservo1.attach(5, 0, 180);
  myservo2.attach(6, 0, 180);
  myservo3.attach(9, 0, 180);
  myservo4.attach(10, 0, 180);
}

void loop() {

  //Rcieves the data package and updates the potentiometer values.
  if (radio.available()){
    radio.read(&data, sizeof(data));
    }

  //Writes the potentiometer value into each servo.
  myservo0.write(data.pot0);
  myservo1.write(data.pot1);
  myservo2.write(data.pot2);
  myservo3.write(data.pot3);
  myservo4.write(data.pot4);

}
```