SHREYA KUMARI

Krshreya2402@gmail.com

IOT JUNE JULY BATCH

OUTSTANDING PROJECT 1

1.Control of a Robotic Arm:

Level 0: Control a Robotic arm of 6 DOF using the Analog input fed from a Potentiometer attached to it.

LEVEL 0:

Components required:

- Arduino board
- Robot arm (4 degree of freedom)
- 10K potentiometer
- Button x2
- 10K resistor x2
- Breadboard
- Jump wires
- Power adapter

Connections:

- Connect the Arduino VCC and GND to VCC and GND of the breadboard.
- Connect the power adapter +V and GND to VCC and GND of the breadboard.
- Connect the Power Adapter GND to GND of the Arduino on the breadboard.
- Fix the buttons on to breadboard.

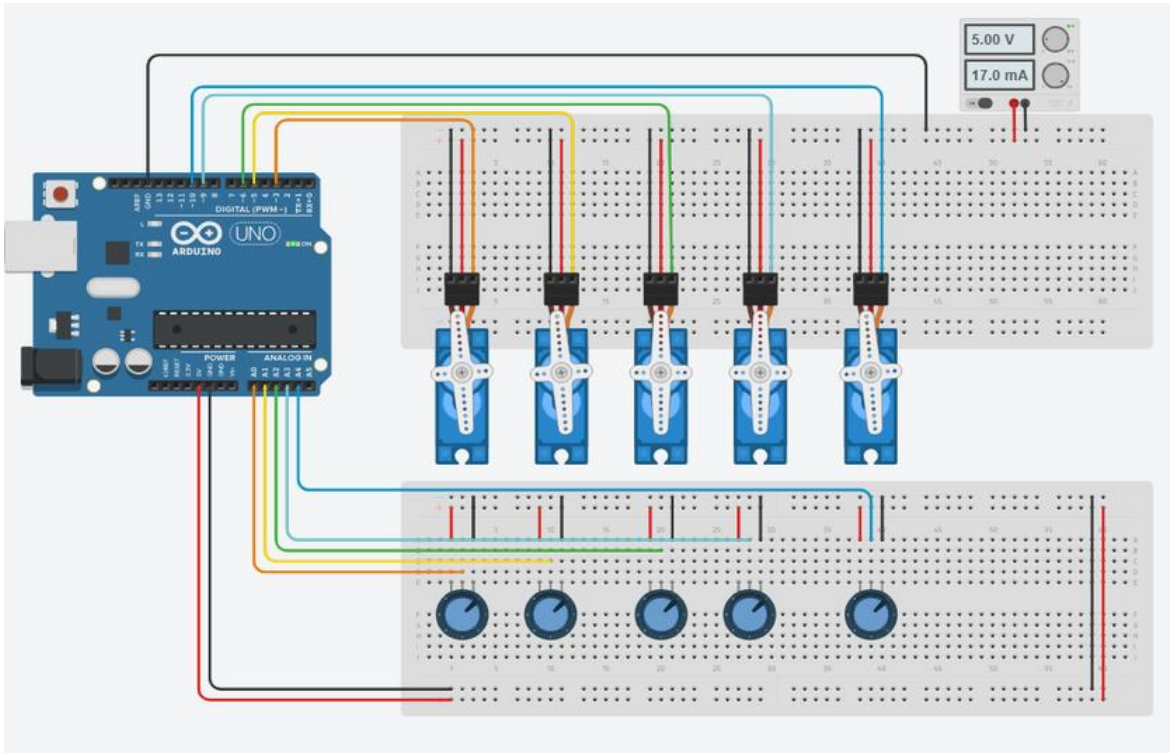
- Two buttons are used in the following circuit. One of the buttons is for recording the servo positions, and the other is for restarting recorded servo positions.
- The resistor connects to the one leg of the buttons (Other leg of the resistors connect to GND of the breadboard).
- Connect the other legs of the buttons to VCC (+5V) of the breadboard.
- Fix the potentiometers on to breadboard.
- First button's signal leg connects to digital 12 input of the Arduino.
- Second button's signal leg connect to digital 13 input of the Arduino.
- Signal (middle) leg of the potentiometers connects to analog A0 - A1 - A2 input of the Arduino respectively.
- One leg of the potentiometers connects to GND input of the breadboard.
- Other leg of the potentiometers connects to VCC (+5V) input of the breadboard. If you want, you can use other PWM inputs of the Arduino.
- The servos power and GND inputs connect to the +V and GND input of the power adapter on the breadboard.

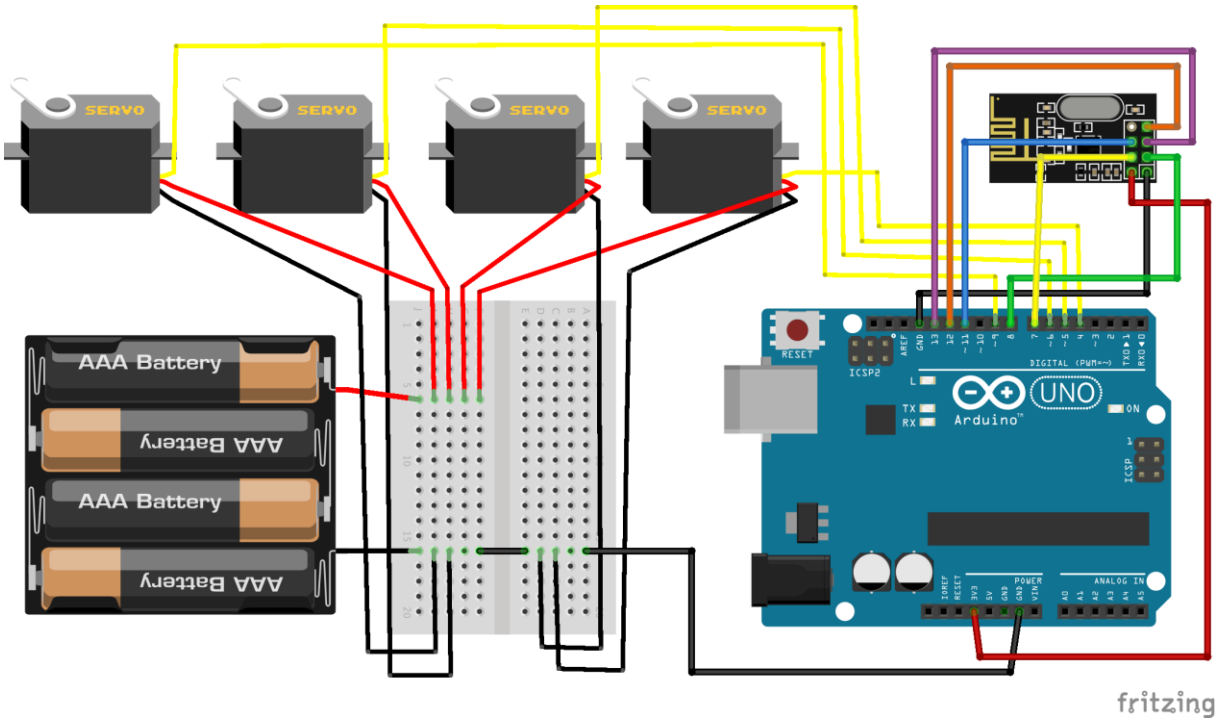
Servo connections:

- Orange Input - Signal Input
- Red Input - Power Input (VCC)
- Brown Input - Ground Input (GND)
- The Servo1 VCC and GND connect to the breadboard's VCC / GND inputs
- The Servo1 Signal connect to the Arduino Digital PWM 3
- The Servo2 VCC and GND connect to the breadboard's VCC / GND inputs
- The Servo2 Signal connect to the Arduino Digital PWM 5
- The Servo3 VCC and GND connect to the breadboard's VCC / GND inputs
- The Servo3 Signal connect to the Arduino Digital PWM 6
- The Servo4 VCC and GND connect to the breadboard's VCC / GND inputs

- The Servo4 Signal connect to the Arduino Digital PWM 9

Circuit diagram:





Code procedure:

- Add the Servo library
- Define Servos
- Define Potentiometers
- Variable to read the values from the analog pin (potentiometers)
- Attaches our servos on pins PWM 3-5-6-9 to the servos
- Reads the value of potentiometers (value between 0 and 1023)
- Scale it to use it with the servo (value between 0 and 180)
- Set the servo position according to the scaled value
- Upload the code

Code:

```
#include <Servo.h>

//Servo Objects
Servo Servo_0;
Servo Servo_1;
Servo Servo_2;
Servo Servo_3;
Servo Servo_4;

//Potentiometer Objects
int Pot_0;
int Pot_1;
int Pot_2;
int Pot_3;
int Pot_4;

//Variable to store Servo Position
int Servo_0_Pos;
int Servo_1_Pos;
int Servo_2_Pos;
int Servo_3_Pos;
int Servo_4_Pos;

//Variable to store Previous position values
int Prev_0_Pos;
```

```
int Prev_1_Pos;
int Prev_2_Pos;
int Prev_3_Pos;
int Prev_4_Pos;

//Variable to store Current position values
int Current_0_Pos;
int Current_1_Pos;
int Current_2_Pos;
int Current_3_Pos;
int Current_4_Pos;



int Servo_Position;       //Stores the angle
int Servo_Number;       //Stores no of servo

int Storage[600];      //Array to store data (Increasing array size will consume more memory)
int Index = 0;          // Array index starts from 0th position
char data = 0;         //variable to store data from serial input.

void setup()
{
  Serial.begin(9600);      //For Serial communication between Arduino and IDE.

  //Servo objects are attached to PWM pins.
  Servo_0.attach(3);
  Servo_1.attach(5);
  Servo_2.attach(6);
  Servo_3.attach(9);
  Servo_4.attach(10);

  //Servos are set to 100 position at initialization.
  Servo_0.write(100);
  Servo_1.write(100);
  Servo_2.write(100);
```

```
  Servo_3.write(100);
  Servo_4.write(100);

  Serial.println("Press 'R' to Record and 'P' to play");
}

void Map_Pot()
{
 /* The servos rotate at 180 degrees
    but to using it to limits is not
    a good idea as it makes the servos buzz continuously
    which is annoying so we limit the servo to move
    between: 1-179 */

   Poten_0 = analogRead(A0);              // Read input from poten and
store it in the Variable Pot_0.
   Servo_0_Pos = map(Poten_0, 0, 1023, 1, 179);      //Map servos as
per the value between 0 to 1023
   Servo_0.write(Servo_0_Pos);          //Move the servo to that position.

   Poten_1 = analogRead(A1);
   Servo_1_Pos = map(Poten_1, 0, 1023, 1, 179);
   Servo_1.write(Servo_1_Pos);

   Poten_2 = analogRead(A2);
   Servo_2_Pos = map(Poten_2, 0, 1023, 1, 179);
   Servo_2.write(Servo_2_Pos);

   Poten_3 = analogRead(A3);
   Servo_3_Pos = map(Poten_3, 0, 1023, 1, 179);
   Servo_3.write(Servo_3_Pos);

   Poten_4 = analogRead(A4);
   Servo_4_Pos  = map(Poten_4, 0, 1023 , 1, 179);
   Servo_4.write(Servo_4_Pos);
}
void loop()
```

```
{
  ana_Poten();      //Function call to read poten values

  while (Serial.available() > 0)
  {
    data = Serial.read();
    if (data == 'R')
    Serial.println("Recording Moves...");
    if (data == 'P')
    Serial.println("Playing Recorded Moves...");
  }

 if (data == 'R')      //If 'R' is entered, start recording.
 {

   //Store the values in a variable
   Prev_0_Pos = Servo_0_Pos;
   Prev_1_Pos = Servo_1_Pos;
   Prev_2_Pos = Servo_2_Pos;
   Prev_3_Pos = Servo_3_Pos;
   Prev_4_Pos = Servo_4_Pos;

   Ana_Poten();  // ana  function recalled for comparison

   if (abs(Prev_0_Pos == Servo_0_Pos))   // absolute value is obtained
by comparing
   {
     Servo_0.write(Servo_0_Pos);   // If values match servo is
repositioned
     if (Current_0_Pos != Servo_0_Pos)  // If values don't match
   {
       Storage[Index] = Servo_0_Pos + 0;   // Value is added to array
       Index++;   // Index value incremented by 1
     }
     Current_0_Pos = Servo_0_Pos;
   }
```

```
   /* Similarly the value comparison is done for all the servos, +100 is
added every for entry
       as a differential value. */

    if (abs(Prev_1_Pos == Servo_1_Pos))
    {
      Servo_1.write(Servo_1_Pos);
      if (Current_1_Pos != Servo_1_Pos)
      {
        Storage[Index] = Servo_1_Pos + 100;
        Index++;
    }
     Current_1_Pos = Servo_1_Pos;
   }


   if (abs(Prev_2_Pos == Servo_2_Pos))
   {
     Servo_2.write(Servo_2_Pos);
     if (Current_2_Pos != Servo_2_Pos)
     {
       Storage[Index] = Servo_2_Pos + 200;
       Index++;
     }
    Current_2_Pos = Servo_2_Pos;
   }


   if (abs(Prev_3_Pos == Servo_3_Pos))
   {
     Servo_3.write(Servo_3_Pos);
     if (Current_3_Pos != Servo_3_Pos)
     {
       Storage[Index] = Servo_3_Pos + 300;
       Index++;
     }
     Current_3_Pos = Servo_3_Pos;
```

```
   }
   if (abs(Prev_4_Pos == Servo_4_Pos))
   {
     Servo_4.write(Servo_4_Pos);
     if (Current_4_Pos != Servo_4_Pos)
     {
      Storage[Index] = Servo_4_Pos + 400;
      Index++;
     }
    Current_4_Pos = Servo_4_Pos;
  }

  /* Values are printed on serial monitor, '\t' is for displaying values in
tabular format */
  Serial.print(Servo_0_Pos);
  Serial.print(" \t ");
  Serial.print(Servo_1_Pos);
  Serial.print(" \t ");
  Serial.print(Servo_2_Pos);
  Serial.print(" \t ");
  Serial.print(Servo_3_Pos);
  Serial.print(" \t ");
  Serial.println(Servo_4_Pos);
  Serial.print ("Index = ");
  Serial.println(Index);
  delay(50);
  }

  if (data == 'P')  //IF 'P' is entered , Start playing recorded moves.
  {
    for (int i = 0; i < Index; i++) //Traverse the array using for loop
    {
      Servo_Number = Storage[i] / 100;  // Finds number of servo
      Servo_Position = Storage[i] % 100;  // Finds position of servo

      switch(Servo_Number)
      {
```

```
        case 0:
            Servo_0.write(Servo_Position);
            break;

        case 1:
            Servo_1.write(Servo_Position);
            break;

        case 2:
            Servo_2.write(Servo_Position);
            break;

        case 3:
            Servo_3.write(Servo_Position);
            break;

        case 4:
            Servo_4.write(Servo_Position);
            break;
       }
     delay(50);
   }
 }
}
```

The Logic of this code is fairly simple the values of potentiometers are stored in an array the records are then traversed using a for loop and the servos do the steps as per the values. First, we will declare all the necessary variables globally so we can use them throughout the program. Now we will write a setup function, where we set pins and their functions. This is the main function that executes first. Now we have to read the values of potentiometers using Analog Input pins and map them to control servos. For this we will define a function and name it **ana_Poten();** , you can name it anything you want it is a user defined function. Now we will write loop function. Once the code is ready, Now upload it to the Arduino board.

After uploading the code to the board successfully, Open 'Serial Monitor' you can find it in Tools option. When Serial monitor starts the Arduino will reset. Now you can control the robotic arm using the master arm. But nothing is being recorded. To start recording, enter 'R' in the monitor now you can perform the moves you wish to record. After the moves are done you have to enter 'P' in order to play the recorded moves. The servos will continue to perform the moves as long as the board is not reset.

Level 1: Make a Closed loop Control of a Robotic arm if the sensor attached to the gripper of a Robotic arm senses an Object in front of it.

If there is an Object → Arm base Joint Rotates

Components Required:

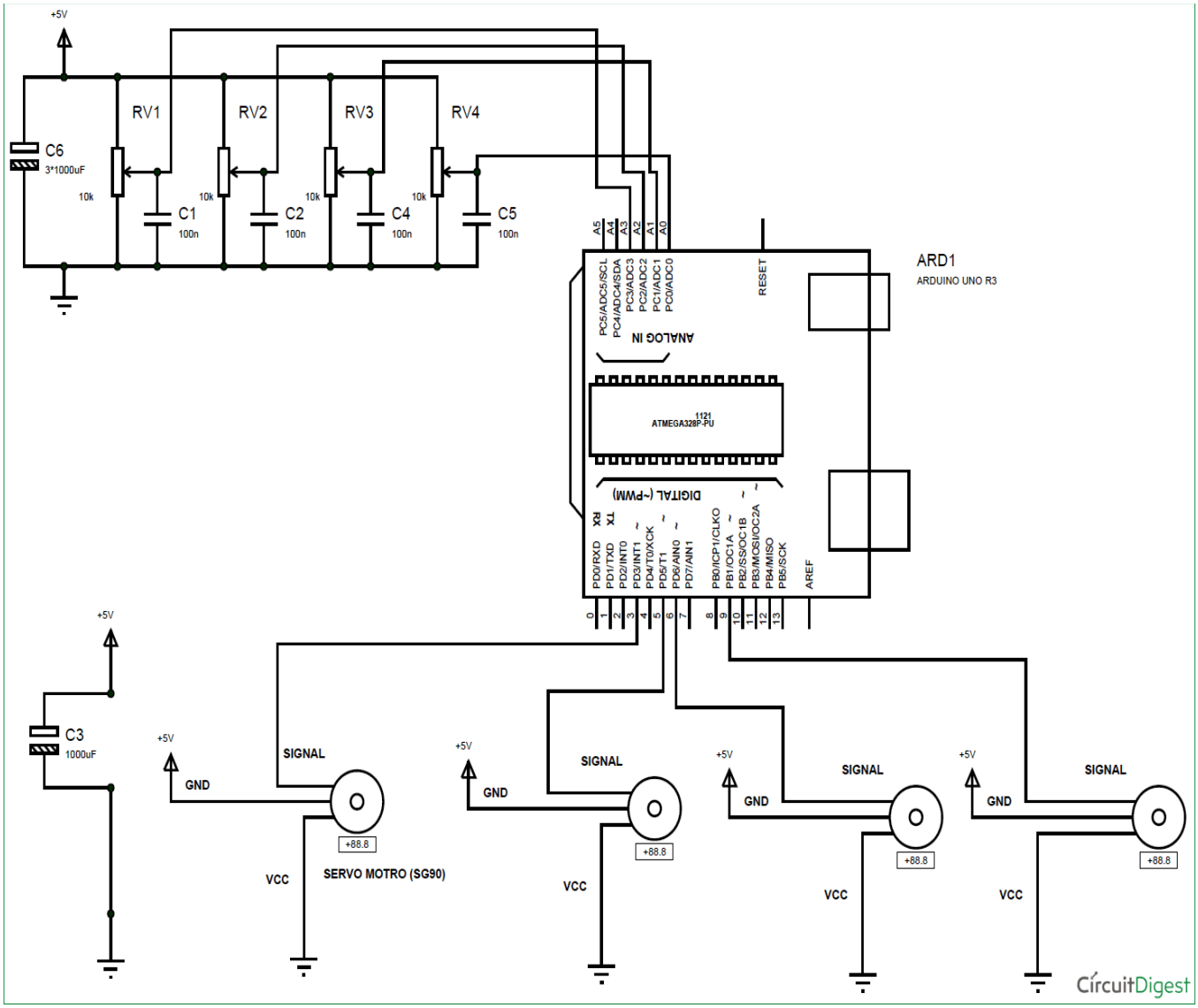
- Arduino Uno
- 1000uF Capacitor (4 pieces)
- 100nF Capacitor (4 pieces)
- Servo Motor (SG 90- four pieces)
- 10K pot- Variable Resistor (4 pieces)
- Power Supply (5v- preferably two)

Construction:

Take a flat and stable surface, like a table or a hard card board. Next place a servo motor in the middle and glue it in place. Make sure the degree of rotation is in the area. This servo acts as base of arm. Place a small piece of cardboard on top of first servo and then place the second servo on this piece of board and glue it in place. Take some cardboards and cut them into 3cm x 11cm pieces. Make sure the piece is not softened. Cut a rectangular hole at one end (leave 0.8cm from bottom) just enough to fit another servo and at another end fit the servo gear tightly with screws or by glue. Then fit the third servo in the first hole. Now cut another cardboard piece with lengths and glue another gear at the bottom of this piece. Now glue the fourth and the last servo at the edge of second piece. With this, two pieces together looks like. When we attach this setup to the base it should look like, it’s almost done. We

just need to make the hook to grab and pick the object like a robotic hand. For hook, cut another two pieces of card board of lengths 1cmx7cm & 4cmx5cm. Glue them together and stick final gear at the very edge. Mount this piece on top and with this we have done building our Robotic Arm. With this, our basic robotic arm design got completed and that's how we have built our low-cost robotic arm. Now connect the circuit in breadboard as per circuit diagram.

Circuit diagram:



The voltage across variable resistors is not completely linear; it will be a noisy one. So, to filter out this noise, capacitors are placed across each

resistor. Now we will feed the voltage provided by these variable resistor (voltage which represents position control) into ADC channels of Arduino. We are going to use four ADC channels of UNO from A0 to A3 for this. After the ADC initialization, we will have digital value of pots representing the position needed by user. We will take this value and match it with servo position.

Arduino has six ADC channels. We have used four for our **Robotic Arm**. The UNO ADC is of 10-bit resolution so the integer values ranging from 0-1023 (2^10=1024 values). This means that it will map input voltages between 0 and 5 volts into integer values between 0 and 1023. So, for every (5/1024= 4.9mV) per unit.

Arduino ADC channels have a default reference value of 5V. This means we can give a maximum input voltage of 5V for ADC conversion at any input channel. Since some sensors provide voltages from 0-2.5V, so with a 5V reference, we get lesser accuracy, so we have an instruction that enables us to change this reference value. So for changing the reference value we have *"analogReference();"*

As default we get the maximum board ADC resolution which is 10bits, this resolution can be changed by using instruction ("analogReadResolution(bits);").

In our Robotic hand circuit, we have left this reference voltage to the default, so we can read value from ADC channel by directly calling function *"analogRead(pin);",* here "pin" represents pin where we connected the analog signal, say we want to read "A0". The value from ADC can be stored into an integer as *int SENSORVALUE0 = analogRead(A0);*.

Now let's talk about the SERVO, the Arduino Uno has a feature which enables us to control the servo position by just giving the degree value. Say if we want the servo to be at 30, we can directly represent the value in the program. The SERVO header (*Servo.h*) file takes care of all the duty ratio calculations internally.

Here first statement represents the header file for controlling the SERVO MOTOR. Second statement is naming the servo; we leave it as *servo0*

as we are going to use four. Third statement states where the servo signal pin is connected; this must be a PWM pin. Here we are using PIN3 for first servo. Fourth statement gives commands for positioning servo motor in degrees. If it is given 30, the servo motor rotates 30 degrees.

Now, we have SG90 servo position from 0 to 180 and the ADC values are from 0-1023. We will use a special function which matches both values automatically.

Code:

```
#include <Servo.h>
Servo servo0;
Servo servo1;
Servo servo2;
Servo servo3;
int sensorvalue0;
int sensorvalue1;
int sensorvalue2;
int sensorvalue3;
void setup()
{
  pinMode(A0,INPUT);
  pinMode(3,OUTPUT);
  servo0.attach(3);

  pinMode(A1,INPUT);
  pinMode(5,OUTPUT);
  servo1.attach(5);

  pinMode(A2,INPUT);
  pinMode(6,OUTPUT);
  servo2.attach(6);

  pinMode(A3,INPUT);
  pinMode(9,OUTPUT);
```

```
  servo3.attach(9);
}

void loop()
{
  sensorvalue0 = analogRead(A0);
  sensorvalue0 = map(sensorvalue0, 0, 1023, 0, 180);
  servo0.write(sensorvalue0);
  sensorvalue1 = analogRead(A1);
  sensorvalue1 = map(sensorvalue1, 0, 1023, 0, 180);
  servo1.write(sensorvalue1);
  sensorvalue2 = analogRead(A2);
  sensorvalue2 = map(sensorvalue2, 0, 1023, 0, 180);
  servo2.write(sensorvalue2);
  sensorvalue3 = analogRead(A3);
  sensorvalue3 = map(sensorvalue3, 0, 1023, 0, 180);
  servo3.write(sensorvalue3);
}
```