# Pole Climbing Robot

## Advanced Mechatronics
**Arduino Mini Project**

**Presented by**

Abhidipta Mallik
Skender Alickolli
Sonia Mary Chacko

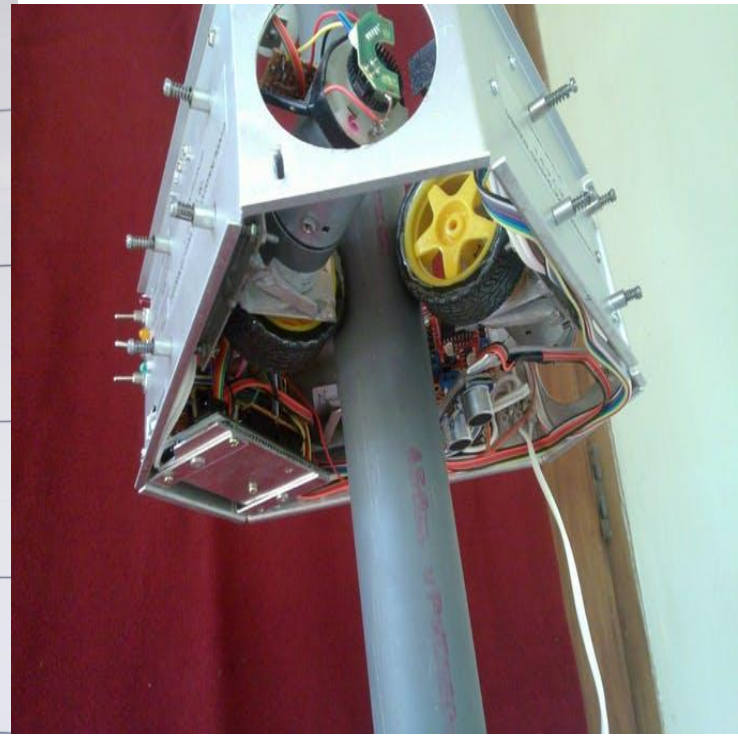
# Applications of Pole Climbing Robot

- Fruit harvesting
- Painting poles
- Cleaning lamp post
- Accessing wind turbines or street light for inspection
- Surveillance
- Fix Telecom cables

# Tree/Pole Climbing Robots

RiSE V1 and V3 Climbing Robot

Serpentine Climbing robot



VEPCRo –
Vertical External Pole Climbing Robot

# Challenges

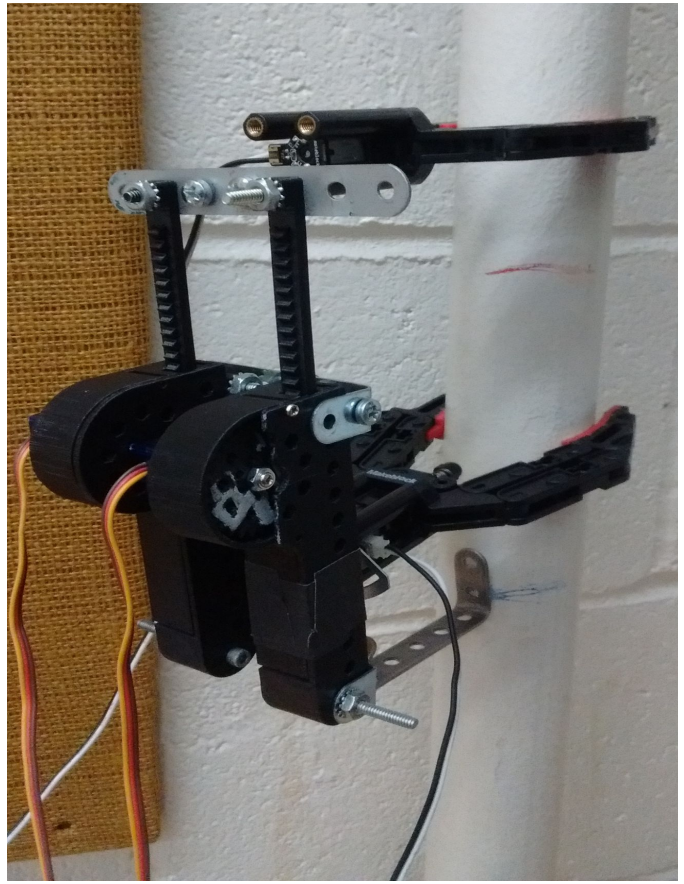
- Enough torque to overcome gravity
- Keep the structure as simple as possible
- Finding a good linear actuator

Using a gripper that keeps the Robot attached to the pole.

Finding a good Linear actuator

# Project approach

- Simple design with two grippers and two linear actuators
- Grippers are fixed at the upper and lower side of the linear actuator
- The climbing up and down movement is done by the linear actuator.
- Reduce the weight of the robot by using 3D printed lightweight linear actuators



# Mechanical design

Figure: Inspired from the tree climbing technique

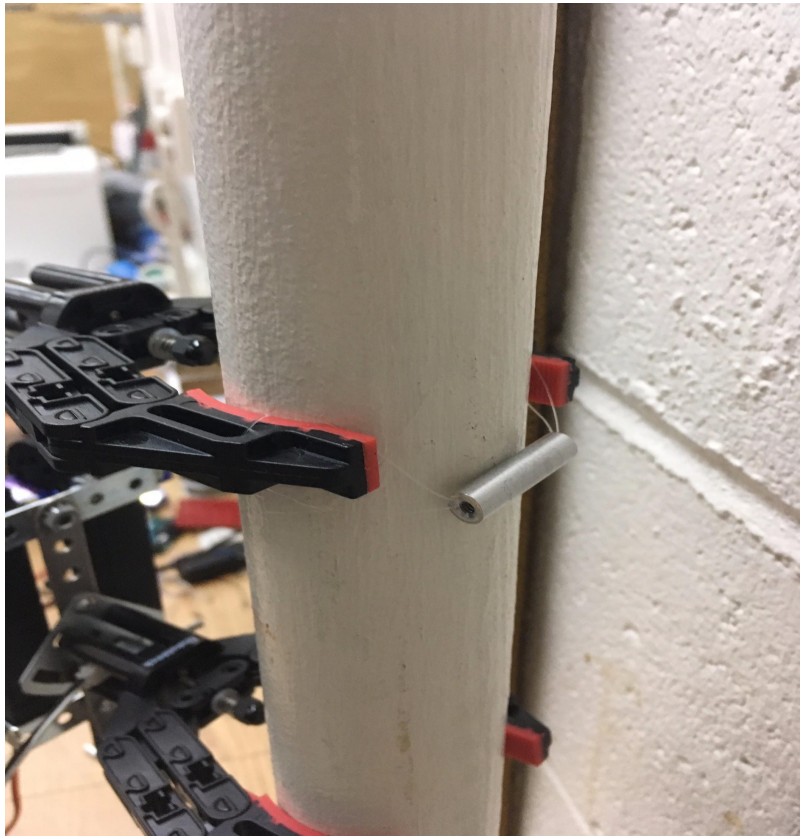


Figure: Extra support for the climber gripper

Fig: Horizontal support for the robot structure

# Linear actuator : 3D printed linear actuator

3D printed linear actuator

Rack and pinion mechanism

Actuator with micro servo

**Rotation to translation**
0 to 120 degree ----> 0 to 54mm

Each actuator can lift up to 200g with 6 V supply

# Main components

Arduino Uno
DC motors (2 No. )
Servo motors (2 No.)
L293D IC
9V battery (2 No.)


Arduino Uno


**MakeBlock robot gripper**

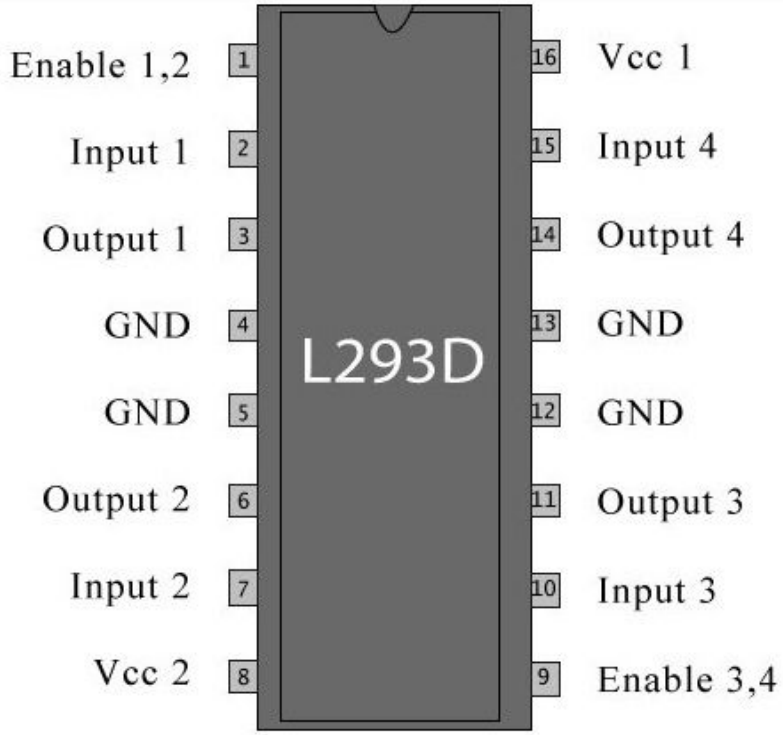
- Light-weight PVC material (68 g)
- Opening width: 67mm(3”)
- Anti-slippery material added on the inner side of two fingers
- Grip items up to 1.5 Kg
- N20 screw motor

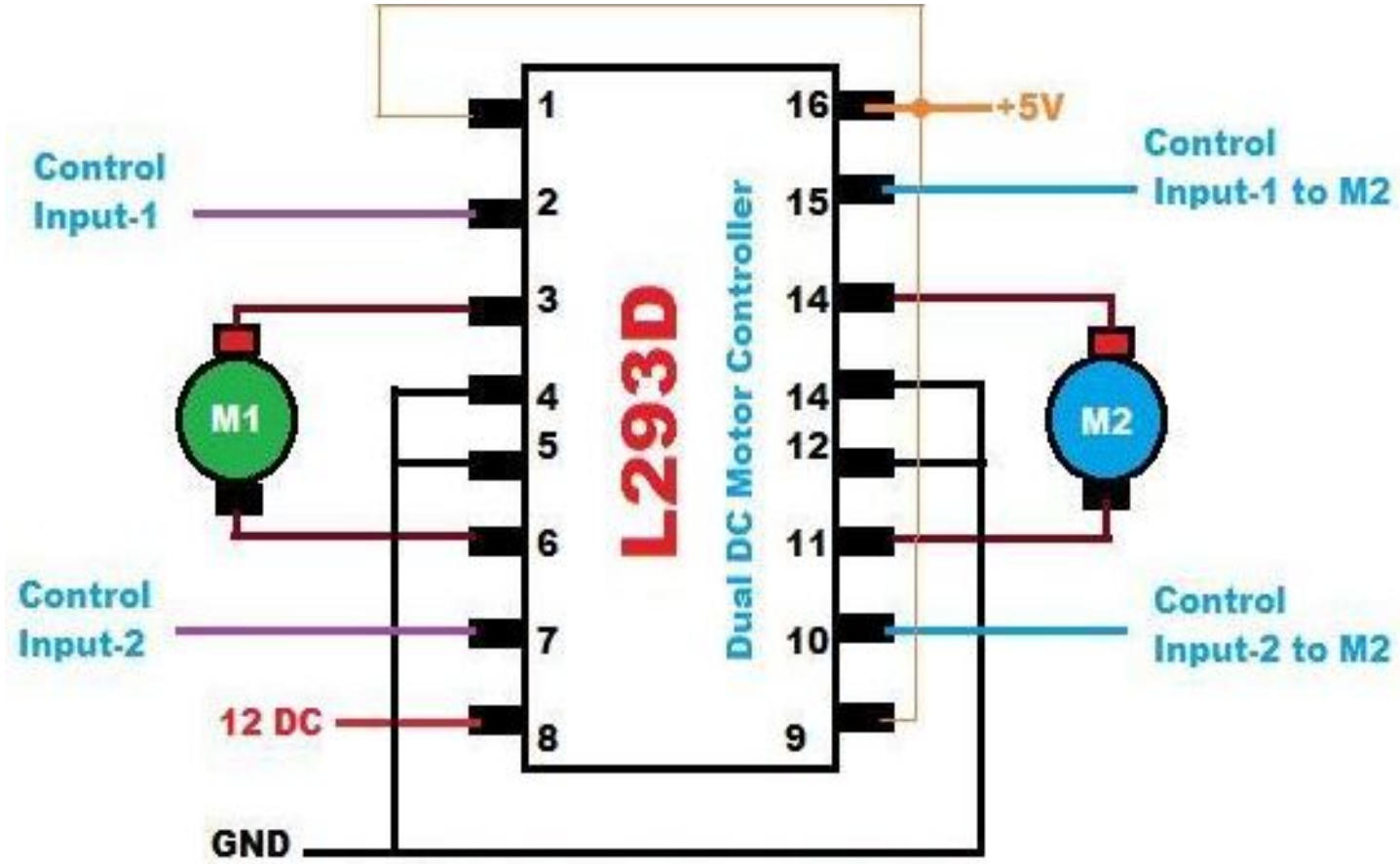
Voltage: 12V
Operating current: ≤110mA
Speed: 600±10%RPM


**Sg92R Servo motor**

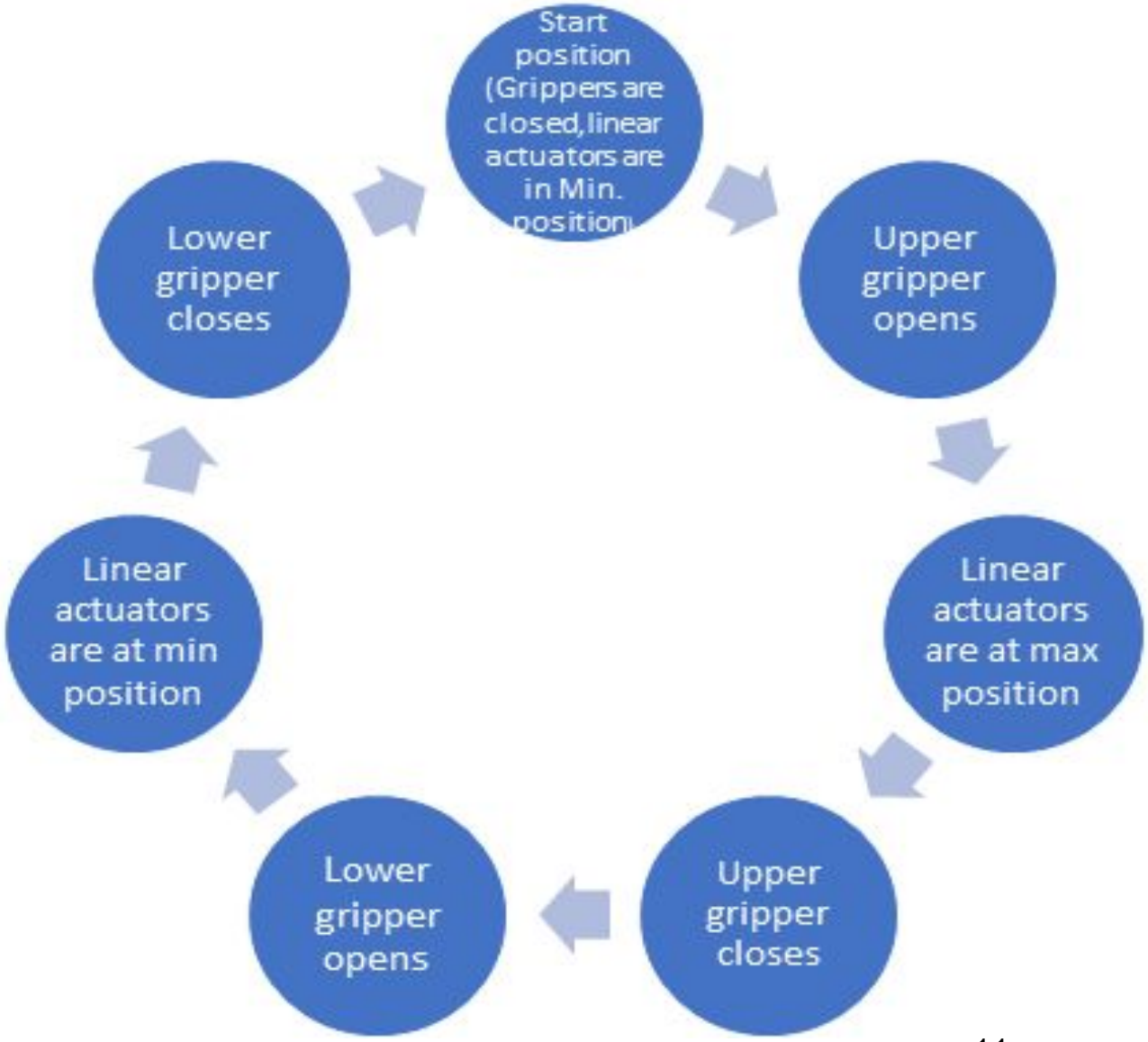
Weight: 9g
Stall torque: 2.5kgcm(4.8v)
Operating voltage: 4.8v

# DC motor driver circuit

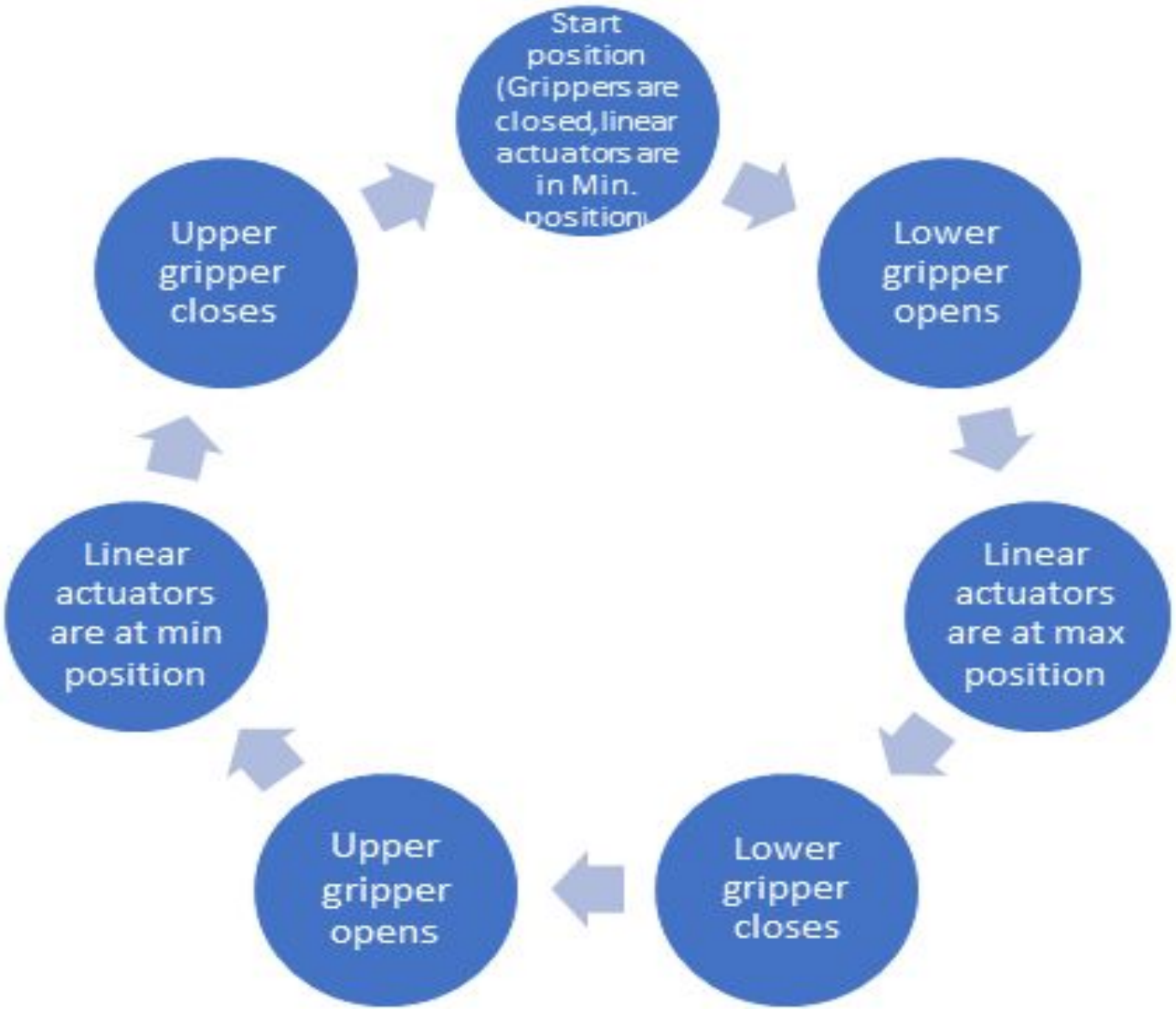


# Climbing up sequences



# Climbing down sequences



# Problems faced

- When we tested with one linear actuator we found that one actuator is not enough to carry the load of Arduino, battery, PCB, grippers
- We used two such actuators by placing it side by side
- Both the actuators should be synchronized to do up and down movement.
- Need extra support for the body (horizontal support metal piece) and the gripper (String and roller attached on the gripper opening)

# Prototype Version 1: Test and evaluation results

- The servo motors are not strong enough to carry the load without a support!
- Need to redesign the linear actuator for using high power servo motors.
- Need stronger gripping mechanism. Need to add one more gripper for more stability
- Climbing at a very slow pace.

# Future Design Improvements

- Design a prototype at big size, which can be used for pole with different diameters and high payloads capability
- Replace the grippers with a better gripping mechanism ( Ex: Soft robot gripper or more number of grippers to climb poles of any texture and diameter)
- Remotely control the robot from a smart device.

# Budget

| Components | Price |
|---|---|
| Arduino Uno | $ 24 |
| Servo motors (2 No.) | $ 14 |
| DC motors with gripper (2 No.) | $ 54 |
| 3D printing (2 No.) | $ 10 |
| IC chip and socket | $ 5 |
| Total | $ 107 |

Thank You!

Questions?

# Arduino Code

```
#include <Servo.h>
Servo myservo1;
Servo myservo2;
int val=0;
#define EnableUM 10  // Enable Pin for motor 1
#define EnableLM 11  // Enable Pin for motor 2

#define UM1 8  // Control pin 1 for Upper motor
#define UM2 9  // Control pin 2 for Upper motor
#define LM1 12  // Control pin 1 for Lower motor
#define LM2 13  // Control pin 2 for Lower motor

void setup() {
   myservo2.attach(3);
   myservo1.attach(5);
   pinMode(EnableUM, OUTPUT);
   pinMode(EnableLM, OUTPUT);
   pinMode(UM1, OUTPUT);
   pinMode(UM2, OUTPUT);
   pinMode(LM1, OUTPUT);
   pinMode(LM2, OUTPUT);

//Enabling the motors for gripper
  digitalWrite(EnableUM, HIGH);
  digitalWrite(EnableLM, HIGH);
 Serial.begin(9600);
}
```

```
void loop()

{

intial_state();

if(Serial.available()>0) {    // if data is available to read

 val = Serial.read();     // read it and store it in 'val'

  if( val == '1' )  { // Val=1 is for climbing up

    // Step1 :
    Serial.println(" Opens the upper gripper" );
    UM_open();
    delay(1000);
    motor_stop();
    delay(2000);

    // Step2 :
     Serial.println("Linear actuator at max position");
     myservo1.write(120);
     myservo2.write(120);
     delay(2000);

    //Step 3:
    Serial.println("Upper gripper closes");
    UM_close ();
    delay(1000);
    motor_stop();
    delay(2000);
```

```
// Step 4:
      Serial.println(" Opens the lower gripper" );
      LM_open();
      delay(1000);
      motor_stop();
      delay(2000);
     // Step 5:
      Serial.println("Linear actuator at Min position");
       myservo1.write(0);
       myservo2.write(0);
       delay(2000);


      // step 6:
      Serial.println("Lower gripper closes");
      LM_close ();
      delay(1000);
      motor_stop();
      delay(2000);
   }
if( val == '2' )  {    //  Val=2 is for climbing down

      // Step1 :
      Serial.println(" Opens the Lower gripper" );
      LM_open();
      delay(1000);
      motor_stop();
      delay(2000);
```

```
// Step2 :
  Serial.println("Linear actuator at max position");
  myservo1.write(120);
  myservo2.write(120);
  delay(2000);
//Step 3:
 Serial.println("Lower gripper closes");
 LM_close ();
 delay(1000);
 motor_stop();
 delay(2000);
  // Step 4:
 Serial.println(" Opens the Upper gripper" );
 UM_open();
 delay(1000);
 motor_stop();
 delay(2000);
// Step 5:
 Serial.println("Linear actuator at Min position");
  myservo1.write(0);
  myservo2.write(0);
  delay(2000);

 // step 6:
 Serial.println("Upper gripper closes");
 LM_close ();
 delay(1000);
 motor_stop();
 delay(2000);  } } }
```

```
// Functions

void intial_state()
{
  Serial.println("At the reset state");
  myservo1.write(0);
  myservo2.write(0);
  UM_close ();
  LM_close ();
  delay(2000);

 }

void UM_open()
{
digitalWrite(UM1, LOW);
digitalWrite(UM2, HIGH);
delay(25);
}
void LM_open()
{
digitalWrite(LM1, LOW);
digitalWrite(LM2, HIGH);
delay(25);
}
void UM_close()
{
digitalWrite(UM1, HIGH);
digitalWrite(UM2, LOW);
delay(25);
}

void LM_close()
{
digitalWrite(LM1, HIGH);
digitalWrite(LM2, LOW);
delay(25);
}

void motor_stop()
{
 digitalWrite(UM1, LOW);
 digitalWrite(UM2, LOW);
  digitalWrite(LM1, LOW);
 digitalWrite(LM2, LOW);
 delay(25);
}
```