Robotics Project

3 DOF Robot Arm

Team members:

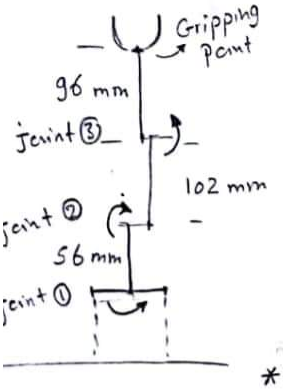
- Mohamed Medhat Ghareeb
- Mohamed Selim Sayid
- Mahmoud Tarik Shosha
- Mostafa Mahmoud Marzouk
- Mostafa Adel Mohamed

This document explains a procedure for getting models of robot kinematics that are appropriate for robot control design. The procedure consists of the following steps:

1- derivation of robot kinematic models and establishing correctness of their structures
2- experimental estimation of the model parameters
3- model validation

Motors calculations:

Robotics Project



* Gripping Point

$\therefore F * 0.035 = T$

$= 0.13$

$F = 3.7\ N$ → Assuming $\mu = 0.2$

max Weight gripper can hold $= 3.7 * 0.2 = 0.74\ N$

$= \boxed{74}$ gram Force

- ABS 40 MPa
- PLA 65 MPa
- motor torque
  3.2 Kg.cm
  ≈ 0.32 N.m

* joint ③ $0.74 * 0.096 = 0.071\ N.m$

$\boxed{T_3 = 0.71\ Kg.cm}$

neglecting links inertia

* joint ② (# weight of 3.2 Kg servo)

$T_2 = (0.102 + 0.096) * 0.74 + 0.36 * (0.102)$

$= 0.183\ N.m = \boxed{1.8}\ Kg\ cm$

* joint ① → the Load on this joint as (Torque) is due to acceleration in movements

So assuming all links are horizontal $l = 0.102 + 0.096 = 0.198\ m$

With $m = (36 * 2 + 74) = 150$ gram

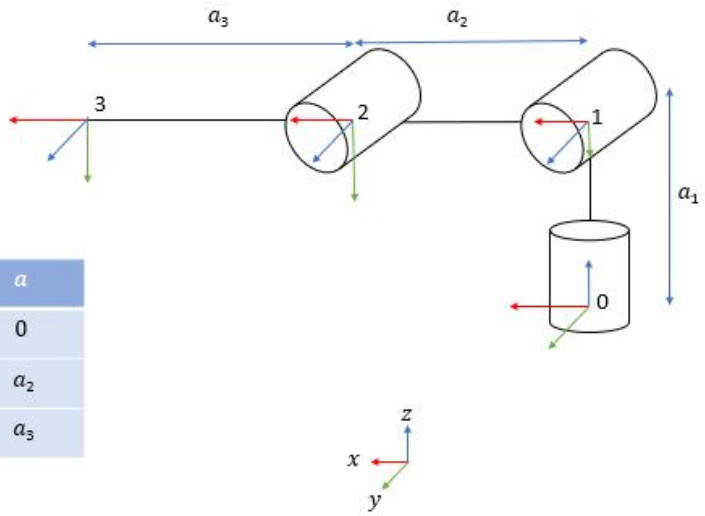
$\therefore T = 3.2\ Kg.cm = 0.32\ N.m \Rightarrow 0.32 = 0.198 * 0.15 * a$

$a_{max} = 10.77\ m/s^2$

⇒ Final Notes

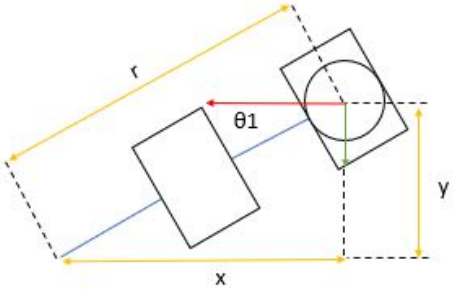
* Fixations of motors & Load on shafts by pending

DH parameter:



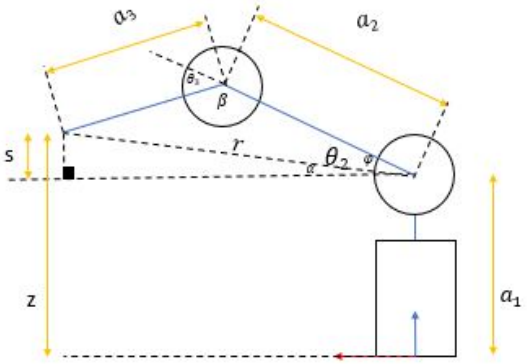
	θ	α	d	a
1	θ_1	$-pi/2$	a_1	0
2	θ_2	0	0	a_2
3	θ_3	0	0	a_3

Inverse Kinematics:



$$r = \sqrt{x^2 + y^2}$$

$$\theta 1 = \tan^{-1}\frac{y}{x}$$



$$s = z - a_1$$

$$\alpha = \sin^{-1}\frac{s}{r}\ ,\ \theta_2 = \alpha + \varphi,\ \theta_3 = \pi - \beta$$

$$a_3{}^2 = a_2{}^2 + r^2 - 2a_2 r\cos\varphi$$

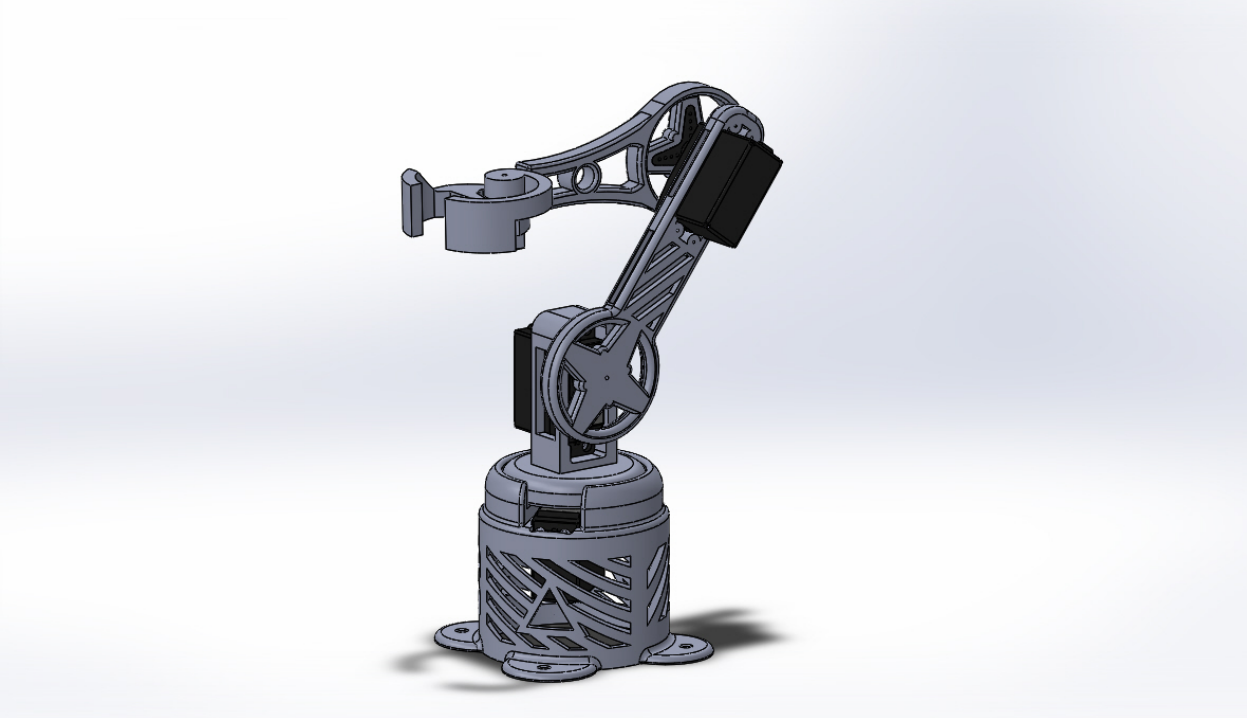
$$\varphi = \cos^{-1}\frac{a_2{}^2 + r^2 - a_3{}^2}{2a_2 r}$$

$$\theta_2 = \alpha + \cos^{-1}\frac{a_2{}^2 + r^2 - a_3{}^2}{2a_2 r}$$

$$r^2 = a_2{}^2 + a_3{}^2 - 2a_2 a_3\cos\beta$$

$$\theta_3 = \pi - \cos^{-1}\frac{a_2{}^2 + a_3{}^2 - r^2}{2a_2 a_3}$$

CAD model:



The Arduino code for controlling the robot arm using FK and IK by a joystick with a simulation of its moves in V-rep using ROS for IK:

1- Forward Kinematics With PS4 Controller :

```
#include <PS4USB.h>
#ifdef dobogusinclude
#include <spi4teensy3.h>
#endif
#include <SPI.h>

USB Usb;
PS4USB PS4(&Usb);
```

```
bool printAngle, printTouch;
uint8_t oldL2Value, oldR2Value;

int ix_axis_left;
int ix_left;
int iy_axis_left;
int iy_left;
int ix_axis_right;
int ix_right;

int g_angle = 90 ;
int b_angle = 90 ;
int s_angle = 90 ;
int e_angle = 90 ;

int vrep_b_angle;
int vrep_s_angle;
int vrep_e_angle;
int vrep_g_angle;

#include <VarSpeedServo.h>

 VarSpeedServo servo1, servo2, servo3, servo4;


void setup() {
  //Serial.begin(115200);
  Serial.begin(115200);
```

```
#if !defined(__MIPSEL__)
  while (!Serial); // Wait for serial port to connect
#endif
  if (Usb.Init() == -1) {
    //Serial.print(F("\r\nOSC did not start"));
    while (1); // Halt




  }
 // Serial.print(F("\r\nPS4 USB Library Started"));
  servo1.attach(4);
  servo2.attach(5);
  servo3.attach(6);
  servo4.attach(7);

}

void loop() {
     //V-REP joint values
        vrep_b_angle = map ( b_angle , 10, 180, -45, 125);
        vrep_s_angle = map ( s_angle, 10, 180, -100, 70);
        vrep_e_angle = map ( e_angle, 0, 180, 10, -170);
        vrep_g_angle = map ( g_angle, 80, 170, -20, 80);

    //serial_senddata();

  Usb.Task();
```

```
  if (PS4.connected()) {




// base servo control by left x axis pad on PS4 Controller


    if (PS4.getAnalogHat(LeftHatX) && PS4.getAnalogHat(LeftHatX) < 100) {
    // Serial.print(F("\r\nLeft_X: "));
    // Serial.print(PS4.getAnalogHat(LeftHatX));
    // Serial.print(F("\r\nbase angle: "));
    // Serial.print(b_angle);
     ix_axis_left = map (PS4.getAnalogHat(LeftHatX), 0, 100, 30, 0);


     b_angle++;
     delay(40);
      if ( b_angle >= 10 || b_angle <= 180 ){
            b_angle++;
            }
      if ( b_angle < 10) {
            b_angle = 10;
            }
      if (b_angle > 180) {
           b_angle = 180;
            }



     servo1.write(b_angle, ix_axis_left);
```

```
    serial_senddata();
    }

  if (PS4.getAnalogHat(LeftHatX) && PS4.getAnalogHat(LeftHatX) > 150){
    // Serial.print(F("\r\nLeft_X: "));
    // Serial.print(PS4.getAnalogHat(LeftHatX));
    // Serial.print(F("\r\nbase angle: "));
    // Serial.print(b_angle);
     ix_left = map(PS4.getAnalogHat(LeftHatX), 150, 255, 0, 30);

    b_angle--;
     delay(40);
    if ( b_angle >= 10 || b_angle <= 180 ){
          b_angle--;
          }
    if ( b_angle < 10) {
          b_angle = 10;
          }
    if (b_angle > 180) {
         b_angle = 180;
         }
     servo1.write(b_angle, ix_left);
     serial_senddata();
      }

// shoulder servo control by left Y axis PS4 controller

    if (PS4.getAnalogHat(LeftHatY) && PS4.getAnalogHat(LeftHatY) < 100) { // These are the only analog
buttons on the PS4 controller
```

```
// Serial.print(F("\r\nLeft_Y: "));
// Serial.print(PS4.getAnalogHat(LeftHatY));
// Serial.print(F("\r\nshoulder angle: "));
// Serial.print(s_angle);
 iy_axis_left = map (PS4.getAnalogHat(LeftHatY), 0, 100, 30, 0);
 s_angle++;
 delay(40);
  if ( s_angle >= 10 || s_angle <= 180 ){
        s_angle++;
        }
  if ( s_angle < 10) {
        s_angle = 10;
        }
  if (s_angle > 180) {
       s_angle = 180;
        }
 servo2.write(s_angle, iy_axis_left);
 serial_senddata();
 }

if (PS4.getAnalogHat(LeftHatY) && PS4.getAnalogHat(LeftHatY) > 150){
  // Serial.print(F("\r\nLeft_Y: "));
  // Serial.print(PS4.getAnalogHat(LeftHatY));
  // Serial.print(F("\r\nshoulder angle: "));
  // Serial.print(s_angle);
   iy_left = map (PS4.getAnalogHat(LeftHatY), 150, 225, 0, 30);

   s_angle--;
   delay(40);
```

```
    if ( s_angle >= 10 || s_angle <= 180 ){
        s_angle--;
        }
    if ( s_angle < 10) {
        s_angle = 10;
        }
    if (s_angle > 180) {
       s_angle = 180;
        }
     servo2.write(s_angle, iy_left);
     serial_senddata();
      }



// ELBOW SERVO CONTROL BY PS4 RIGHT X AXIS  on PS4 controller



   if (PS4.getAnalogHat(RightHatY) && PS4.getAnalogHat(RightHatY) < 100) { // These are the only
analog buttons on the PS4 controller
    // Serial.print(F("\r\nRight_X: "));
   //  Serial.print(PS4.getAnalogHat(RightHatX));
    // Serial.print(F("\r\nelbow angle: "));
    // Serial.print(e_angle);
     ix_axis_right = map (PS4.getAnalogHat(RightHatY), 0, 100, 30, 0);

    // Serial.print(F("\r\nRight_X inverse: "));
   //  Serial.print(ix_axis_right);
    e_angle++;
     delay(40);
```

```
  if ( e_angle >= 0 || e_angle <= 180 ){
        e_angle++;
        }
  if ( e_angle < 0) {
        e_angle = 0;
        }
  if (e_angle > 180) {
       e_angle = 180;
        }
  servo3.write(e_angle, ix_axis_right);
  serial_senddata();
  }

if (PS4.getAnalogHat(RightHatY) && PS4.getAnalogHat(RightHatY) > 150){
  ////  Serial.print(F("\r\nRight_X: "));
  //  Serial.print(PS4.getAnalogHat(RightHatX));
  //  Serial.print(F("\r\nelbow angle: "));
 //   Serial.print(e_angle);
    ix_right = map ( PS4.getAnalogHat(RightHatY), 150, 255, 0, 30);

    e_angle--;
     delay(40);
  if ( e_angle >= 0 || e_angle <= 180 ){
        e_angle--;
        }
  if ( e_angle < 0) {
        e_angle = 0;
        }
  if (e_angle > 180) {
```

```
          e_angle = 180;
           }
       servo3.write(e_angle,  ix_right);
       serial_senddata();
        }

// gribber control by R2 & L2 analog buttons on PS4 controller
   if (PS4.getAnalogButton(L2)) { // These are the only analog buttons on the PS4 controller
     //Serial.print(F("\r\nL2: "));
    // Serial.print(PS4.getAnalogButton(L2));
    // Serial.print(F("\r\ngribber angle: "));
    // Serial.print(g_angle);
     g_angle++;
    delay(40);
     if ( g_angle >= 80 || g_angle <= 175 ){
           g_angle++;
           }
     if ( g_angle < 80) {
           g_angle = 80;
           }
     if (g_angle > 175) {
          g_angle = 175;
           }


    servo4.write(g_angle, PS4.getAnalogButton(L2));
    serial_senddata();
    }
```

```
    if (PS4.getAnalogButton(R2)){
       //Serial.print(F("\r\nR2: "));
      // Serial.print(PS4.getAnalogButton(R2));
      // Serial.print(F("\r\ngribber angle: "));
      // Serial.print(g_angle);



       g_angle--;
       delay(40);
      if ( g_angle >= 80 || g_angle <= 175 ){
            g_angle--;
            }
      if ( g_angle < 80) {
            g_angle = 80;
            }
      if (g_angle > 175) {
          g_angle = 175;
            }
       servo4.write(g_angle, PS4.getAnalogButton(R2));
       serial_senddata();
        }
if (PS4.getButtonClick(CIRCLE)) {
  Position_1();
    Serial.print(35);
    Serial.print(",");
    Serial.print(-20);
    Serial.print(",");
    Serial.print(-80);
    Serial.print(",");
```

```
    Serial.println(-20);
    delay(40);
}
if (PS4.getButtonClick(CROSS)) {
  Position_2();
    Serial.print(-45);
    Serial.print(",");
    Serial.print(-100);
    Serial.print(",");
    Serial.print(-170);
    Serial.print(",");
    Serial.println(80);
    delay(40);
}
  if (PS4.getButtonClick(SQUARE)) {
  Position_3();
    Serial.print(125);
    Serial.print(",");
    Serial.print(-100);
    Serial.print(",");
    Serial.print(10);
    Serial.print(",");
    Serial.println(47);
    delay(40);
}
if (PS4.getButtonClick(TRIANGLE)) {
  Position_4();
```

```
    Serial.print(50);
    Serial.print(",");
    Serial.print(-35);
    Serial.print(",");
    Serial.print(-155);
    Serial.print(",");
    Serial.println(-20);
    delay(40);
}
    }

     // delay(1000);

 }



 void serial_senddata(){

    Serial.print(vrep_b_angle);
    Serial.print(",");
    Serial.print(vrep_s_angle);
    Serial.print(",");
    Serial.print(vrep_e_angle);
    Serial.print(",");
    Serial.println(vrep_g_angle);

    delay(40);
```

```
    return;
  }

void Position_1(){
  servo1.write(90, 35);
  servo2.write(90, 35);
  servo3.write(90, 35);
  servo4.write(80, 35);

  servo1.wait();
  servo2.wait();
  servo3.wait();
  servo4.wait();


  delay(300);
}

void Position_2(){
  servo1.write(10, 35);
  servo2.write(10, 35);
  servo3.write(180, 35);
  servo4.write(170, 35);

  servo1.wait();
  servo2.wait();
  servo3.wait();
  servo4.wait();
```

```
  delay(300);



}

void Position_3(){
  servo1.write(180, 35);
  servo2.write(10, 35);
  servo3.write(0, 35);
  servo4.write(140, 35);

  servo1.wait();
  servo2.wait();
  servo3.wait();
  servo4.wait();


  delay(300);
}


void Position_4(){
  servo1.write(105, 35);
  servo2.write(75, 35);
  servo3.write(165, 35);
  servo4.write(80, 35);

  servo1.wait();
  servo2.wait();
```

```
  servo3.wait();
  servo4.wait();



  delay(300);
}
```

2- Inverse Kinematics Arduino Code :

```
#include <VarSpeedServo.h>
#include <math.h>

/* Arm dimensions( mm ) */
#define a1 55 //a1 = base to 2# joint length arm
#define a2 111 //a2 = length arm from joint 2 to 3
#define a3 140 //a3 = length arm from joint #3 to gripper



#define base_servo 4
/* Using Servo 3.2 Kg.cm  */
#define shoulder_servo 5
/* Using Servo 3.2 Kg.cm  */
#define elbow_servo 6
/* Using Micro servo 1.3 Kg.cm */
#define Gripper_servo 7
/*Using Micro servo 1.3 Kg.cm */

  float x_coord;          // X coordinate of the end point
  float y_coord;          // Y coordinate of the end point
  float z_coord;          // Z coordinate of the end point
  float gripper_angle;     //gripper angle


#define speed1 30
#define speed2 100
```

```
#define speed3 170



//Some Values to be used

#define pi 3.141592654


VarSpeedServo servo1,servo2,servo3,servo4;









void setup()

{

servo1.attach( base_servo, 544, 2400 ); //setting min and max values in microseconds,default min is 544, max is 2400

servo2.attach( shoulder_servo, 544, 2400 );

servo3.attach( elbow_servo, 544, 2400 );

servo4.attach( Gripper_servo, 544, 2400 );



//servos.start(); //Start the servo shield

servo_park();

delay( 2000 );

Serial.begin( 9600 );

Serial.println("Start");

}
```

```
void loop()
{
// test fixed positions

//set_arm ( 20, 20, 0, 90, 10);
//delay(1000);
//set_arm ( 20, 50, 0, 150, 10);
//delay(1000);
//set_arm ( 20, 70, 0, 150, 10);
//delay(3000);



if(Serial.available()>0){

 char data = Serial.read();
  switch(data) {

    case 'x' : x_coord++; //adjust x position up
    break;
    case 'u' : x_coord--; // adjust x position down
    break;

    case 'y' : y_coord++; // adjust y position up
    break;
    case 'v' : y_coord--; // adjust y position down
    break;
```

```
 case 'z' : z_coord++; // agjust z position up
 break;
 case 'w' : z_coord--; // adjust z postion down
 break;

 case 'r' : gripper_angle++; //adjust gripper wider
          // set boundries to the gripper
          if( gripper_angle > 150 ) {
          gripper_angle = 120;
         }
          break;
 case 's' : gripper_angle--; // adjust gripper narrower
         // set boundries to the gripper
          if( gripper_angle < 70 ) {
          gripper_angle = 170;
         }
         break;

  case 'm' : arm_park();
  break;

  case 'n' : servo_park();
  break;
  }

set_arm(x_coord, y_coord, z_coord, gripper_angle, speed2);

 }
```

```
}


// XYZ Positioning using the base, shoulder, elbow joints

void set_arm( float x, float y, float z, float gripper, int servospeed){


float theta1_r = atan2( x, y ); // in radian value
float theta1_d = ((theta1_r*180)/pi);

if(theta1_d >= 10 || theta1_d <= 180 ){

        theta1_d = theta1_d;
  }
  if( theta1_d <=9){
   theta1_d =10;
  }

  if ( theta1_d >= 181 ){
   theta1_d = 180;
  }

float r_distance = sqrt(( x * x ) + ( y * y ));


float s = z -a1 ; // length in mm

float alpha_r = asin(s/r_distance); // s must always be less than R distance
```

```
float alpha_d = ((alpha_r*180)/pi);



float theta2_r = alpha_r + acos((sq(a2)+sq(r_distance)-sq(a3))/(2*a2*r_distance));
float theta2_d = ((theta2_r*180)/pi);


 if(theta2_d >= 10 || theta2_d <= 180 ){

        theta2_d = theta2_d;
  }
  if( theta2_d <=9 ){
   theta2_d = 10;
  }

  if (theta2_d >= 181 ) {
   theta2_d = 180;
  }



float theta3_r = pi - acos((sq(a1)+sq(a3)-sq(r_distance))/(2*a2*a3));
float theta3_d = ((theta3_r*180)/pi);


if(theta3_d >= 0 || theta3_d <= 180 ){
```

```
        theta3_d = theta3_d;
  }
  if( theta3_d <=0 ){
   theta3_d = 0;
  }

  if( theta3_d > 180) {
   theta3_d = 180 ;
  }


servo1.write ( theta1_d, servospeed);
servo2.write ( theta2_d, servospeed);
servo3.write ( theta3_d, servospeed);
servo4.write ( gripper, servospeed);
servo1.wait();
servo2.wait();
servo3.wait();
servo4.wait();
Serial.print("T1:");
Serial.print(theta1_d);
Serial.print("          T2:");
Serial.print(theta2_d);
Serial.print("        T3:");
Serial.println(theta3_d);

Serial.print("X:");
Serial.print(x);
Serial.print("          Y:");
```

```
Serial.print(y);
Serial.print("        Z:");
Serial.println(z);

delay(100);  // delay to give the robot time to perform the required position

}



void servo_park(){                  // park position using servo angles
  servo1.write ( 150, speed1);
  //servo.setposition(base_servo ): (Angle, Speed)
  servo2.write ( 105, speed1);
  //servo.setposition( shoulder_servo )
  servo3.write ( 170, speed1);
  //servo.setposition( elbow_servo )
  servo4.write ( 80, speed1);
  //servo.setposition( Gripper_servo )
  servo1.wait();
  servo2.wait();
  servo3.wait();
  servo4.wait();
  delay(500);

  return;
}

void arm_park(){  // park position using XYZ Co-Ordinates
```

```
  set_arm(x_coord = 70, y_coord = 0, z_coord = 0, gripper_angle= 140, speed1);
  delay (500);
}




void zero_x()   //fixed y axis movement
{
for( double yaxis = 250.0; yaxis < 400.0; yaxis += 1 ) {
   Serial.print(" Y axis = ");
   Serial.println(yaxis);
   set_arm( 0, yaxis, 200.0, 0 ,10);
   delay( 10 );
   }
   delay(1000);
for( double yaxis = 400.0; yaxis > 250.0; yaxis -= 1 ) {
   Serial.print(" Y axis = ");
   Serial.println(yaxis);
   set_arm( 0, yaxis, 200.0, 0 ,10);
   delay( 10 );
   }
}
```

3- LUA Code On V_REP:

```
function sysCall_threadmain()

 sim.setThreadSwitchTiming(2) -- Default timing for automatic thread switching
 simDelegateChildScriptExecution()
  --defining the serial port number
 -- port=sim.getScriptSimulationParameter(sim.handle_self,'serialPortNumber')
  portNumber="\\\\.\\COM8"
  --could be defined as followed
  --portNumber=[[\\.\COM21]]
  baudrate=115200
  serial=sim.serialOpen(portNumber,baudrate)



  jointHandles={-1,-1,-1,-1}

     jointHandles[1]=sim.getObjectHandle('first_link')
     jointHandles[2]=sim.getObjectHandle('second_link')
     jointHandles[3]=sim.getObjectHandle('Third_link')
     jointHandles[4]=sim.getObjectHandle('Gripper_link')


 -- Set-up some of the RML vectors:
  vel=120
  accel=40
```

```
 jerk=80
 currentVel={0,0,0,0}
 currentAccel={0,0,0,0}
 maxVel={vel*math.pi/180,vel*math.pi/180,vel*math.pi/180,vel*math.pi/180}
 maxAccel={accel*math.pi/180,accel*math.pi/180,accel*math.pi/180,accel*math.pi/180}
 maxJerk={jerk*math.pi/180,jerk*math.pi/180,jerk*math.pi/180,jerk*math.pi/180}
 targetVel={0,0,0,0}
 --sim.serialSend(serial,'D')
while true do

 --read a full line
 str=sim.serialRead(serial,250,true,'\n',0)

  if str~= nil then
     print(str)
    --sending next reading request
    -- sim.serialSend(serial,'D')
     local token
       val={}
       cpt=0
       --extracting the values in str separated by a ,
       for token in string.gmatch( str, "[^,]+") do
         cpt=cpt+1
         val[cpt]=tonumber(token)
       end

  end
  targetPos={val[1]*math.pi/180,val[2]*math.pi/180,val[3]*math.pi/180,val[4]*math.pi/180}
```

```
    sim.rmlMoveToJointPositions(jointHandles,-
1,currentVel,currentAccel,maxVel,maxAccel,maxJerk,targetPos,targetVel)


    -- targetPos2={-90*math.pi/180,45*math.pi/180,-45*math.pi/180,135*math.pi/180}
    -- sim.rmlMoveToJointPositions(jointHandles,-
1,currentVel,currentAccel,maxVel,maxAccel,maxJerk,targetPos2,targetVel)


    -- targetPos3={0,0,0,0,0,0}
    --sim.rmlMoveToJointPositions(jointHandles,-
1,currentVel,currentAccel,maxVel,maxAccel,maxJerk,targetPos3,targetVel)
    sim.switchThread()


  end
end
```