# **Source Code**

**Source code for motion of robotic arm**

**Version 1.0 for implementing automated and manual mode of the robotic arm**

#include <Servo.h>

Servo myservo[3]; //base turning

Servo myservopick;

#define baseArm A2

#define subArm A3

#define turningPoint A4

#define pick 5

#define pickServo 12

bool button = false;

bool pickStatus = false;

bool blockloop = false;

bool runState = false;

bool modeButton =false;

bool modeState = false;

int receivedValue = 60;

int state[3] = {120,120,60}; // define servo state in oreder to A B C

int destination[3] = {170,170,150}; // define servo destination in order to A B C

int progress[3] = {0,0,0};

int progress\_state = 0;

int read\_position[3]={0,0,0};

void setup() {

// put your setup code here, to run once:

myservo[0].attach(9);

myservo[1].attach(10);

myservo[2].attach(11);

myservopick.attach(pickServo);

pinMode(pick,OUTPUT);

pinMode(14,OUTPUT);

pinMode(15,OUTPUT);

pinMode(16,OUTPUT);

//pinMode(17,OUTPUT);

pinMode(18,INPUT);

digitalWrite(14,HIGH);

digitalWrite(15,HIGH);

digitalWrite(16,HIGH);

//digitalWrite(17,HIGH);

Serial.begin(9600);

start();

myservopick.write(120);

}

//String y;

void loop() {

stateManager();

picker(digitalRead(pick));

}

void start(){

progress\_state=1;

automated();

}

void manual(){

int turn = map(analogRead(turningPoint),1,280,60,150);

int sub = map(analogRead(subArm),1,260,80,170);

int base = map(analogRead(baseArm),510,730,80,170);

picker(digitalRead(pick));

if((base>80)&&(base<170)){

destination[0] = base;

}

if((sub>80)&&(sub<170)){

destination[1] = sub;

}

if((turn>60)&&(turn<150)){

destination[2] = turn;

}

stateManager();

progress\_state=1;

automated();

}

void automated(){

while(progress\_state!=0){

progress\_state=0;

for(int x=0;x<3;x++){

progress[x] = servoTurn(x);

myservo[x].write(state[x]);

progress\_state+=progress[x];

delay(10);

if(Serial.available()>1){

receivedValue = map(Serial.parseInt(),0,550,80,140);

}

picker(digitalRead(pick));

}

}

}

bool servoTurn(int x){

//myservo.write(pos);

if(state[x]!=destination[x]){

if(state[x]>destination[x]){

state[x]--;

}else{

state[x]++;

}

Serial.println(state[x]);

return true;

}

else return false;

}

void feedToUser(){

destination[0]=170;

destination[1]=100;

destination[2]=150;

progress\_state=1;

automated();

destination[0]=170;

destination[1]=150;

destination[2]=150;

progress\_state=1;

automated();

destination[0]=170;

destination[1]=150;

destination[2]=receivedValue;

progress\_state=1;

automated();

destination[0]=120;

destination[1]=110;

destination[2]=receivedValue;

progress\_state=1;

automated();

destination[0]=120;

destination[1]=150;

destination[2]=150;

progress\_state=1;

automated();

}

void picker(bool x){

delay(10);

if(x && !blockloop){

pickStatus = !pickStatus;

blockloop = true;

}else if(!x && blockloop ){

blockloop = false;

}

if(pickStatus){

myservopick.write(50);

}else{

myservopick.write(120);

}

}

void stateManager(){

modeButton = digitalRead(18);

Serial.println(modeButton);

if(modeButton && !modeState){

modeState = true;

runState = !runState;

}else if(!modeButton && modeState){

modeState = false;

}

if(runState){

feedToUser();

}else{

manual();

}

}

**Version 2.0 for addition of Image Processing**

#include <Servo.h>

Servo myservo[3]; //base turning

Servo myservopick;

#define baseArm A2

#define subArm A3

#define turningPoint A4

#define pick 5

#define pickServo 12

bool button = false;

bool pickStatus = false;

bool blockloop = false;

bool runState = false;

bool modeButton =false;

bool modeState = false;

int receivedValue = 60;

int state[3] = {120,120,60}; // define servo state in oreder to A B C

int destination[3] = {170,170,150}; // define servo destination in order to A B C

int progress[3] = {0,0,0};

int progress\_state = 0;

int read\_position[3]={0,0,0};

void setup() {

// put your setup code here, to run once:

myservo[0].attach(9);

myservo[1].attach(10);

myservo[2].attach(11);

myservopick.attach(pickServo);

pinMode(pick,OUTPUT);

pinMode(14,OUTPUT);

pinMode(15,OUTPUT);

pinMode(16,OUTPUT);

//pinMode(17,OUTPUT);

pinMode(18,INPUT);

digitalWrite(14,HIGH);

digitalWrite(15,HIGH);

digitalWrite(16,HIGH);

//digitalWrite(17,HIGH);

Serial.begin(9600);

start();

myservopick.write(120);

}

//String y;

void loop() {

stateManager();

if(Serial.available()>1){

receivedValue = map(Serial.parseInt(),0,550,80,140);

}

picker(digitalRead(pick));

}

void start(){

progress\_state=1;

automated();

}

void manual(){

int turn = map(analogRead(turningPoint),1,280,60,150);

int sub = map(analogRead(subArm),1,260,80,170);

int base = map(analogRead(baseArm),510,730,80,170);

picker(digitalRead(pick));

if((base>80)&&(base<170)){

destination[0] = base;

}

if((sub>80)&&(sub<170)){

destination[1] = sub;

}

if((turn>60)&&(turn<150)){

destination[2] = turn;

}

stateManager();

progress\_state=1;

automated();

}

void automated(){

while(progress\_state!=0){

progress\_state=0;

for(int x=0;x<3;x++){

progress[x] = servoTurn(x);

myservo[x].write(state[x]);

progress\_state+=progress[x];

delay(10);

if(Serial.available()>1){

receivedValue = map(Serial.parseInt(),0,550,80,140);

}

picker(digitalRead(pick));

}

}

}

bool servoTurn(int x){

//myservo.write(pos);

if(state[x]!=destination[x]){

if(state[x]>destination[x]){

state[x]--;

}else{

state[x]++;

}

Serial.println(state[x]);

return true;

}

else return false;

}

void feedToUser(){

destination[0]=170;

destination[1]=100;

destination[2]=150;

progress\_state=1;

automated();

destination[0]=170;

destination[1]=150;

destination[2]=150;

progress\_state=1;

automated();

destination[0]=170;

destination[1]=150;

destination[2]=receivedValue;

progress\_state=1;

automated();

destination[0]=120;

destination[1]=110;

destination[2]=receivedValue;

progress\_state=1;

automated();

destination[0]=120;

destination[1]=150;

destination[2]=150;

progress\_state=1;

automated();

}

void picker(bool x){

delay(10);

if(x && !blockloop){

pickStatus = !pickStatus;

blockloop = true;

}else if(!x && blockloop ){

blockloop = false;

}

if(pickStatus){

myservopick.write(50);

}else{

myservopick.write(120);

}

}

void stateManager(){

modeButton = digitalRead(18);

Serial.println(modeButton);

if(modeButton && !modeState){

modeState = true;

runState = !runState;

}else if(!modeButton && modeState){

modeState = false;

}

if(runState){

feedToUser();

}else{

manual();

}

}

**Testing the Boundary condition of the arm**

#include <Servo.h>

Servo myservo[3];

Servo myservopick;

#define baseArm A2

#define subArm A3

#define turningPoint A4

#define pick 5

#define pickServo 12

bool button = false;

bool pickStatus = false;

bool blockloop = false;

bool runState = false;

bool modeButton =false;

bool modeState = false;

int receivedValue = 60;

int state[3] = {120,120,60}; // define servo state in oreder to A B C

int destination[3] = {170,170,150}; // define servo destination in order to A B C

int progress[3] = {0,0,0};

int progress\_state = 0;

int read\_position[3]={0,0,0};

void setup() {

myservo[0].attach(9);

myservo[1].attach(10);

myservo[2].attach(11);

myservopick.attach(pickServo);

pinMode(pick,OUTPUT);

}

void loop() {

if(Serial.available()>1){

receivedValue = map(Serial.parseInt(),0,550,80,140);

String data = Serial.readString();// data input format => xx,xx,xx in order to x y z positions

read\_position[0] = data.substring(0,2).toInt(); //x

read\_position[1] = data.substring(3,5).toInt(); //y

read\_position[2] = data.substring(6,8).toInt(); //z

destination[0] = read\_position[0]\*10;

destination[1] = read\_position[1]\*10;

destination[2] = read\_position[2]\*10;

progress\_state=1;

automated();

Serial.println(read\_position[2]+1);

}

}

void automated(){

while(progress\_state!=0){

progress\_state=0;

for(int x=0;x<3;x++){

progress[x] = servoTurn(x);

myservo[x].write(state[x]);

progress\_state+=progress[x];

delay(10);

if(Serial.available()>1){

receivedValue = map(Serial.parseInt(),0,550,80,140);

}

picker(digitalRead(pick));

}

}

}

bool servoTurn(int x){

//myservo.write(pos);

if(state[x]!=destination[x]){

if(state[x]>destination[x]){

state[x]--;

}else{

state[x]++;

}

Serial.println(state[x]);

return true;

}

else return false;

}

void picker(bool x){

delay(10);

if(x && !blockloop){

pickStatus = !pickStatus;

blockloop = true;

}else if(!x && blockloop ){

blockloop = false;

}

if(pickStatus){

myservopick.write(50);

}else{

myservopick.write(120);

}

}

**For Image Processing using Raspberry Pi 3**

from picamera.array import PiRGBArray

from picamera import PiCamera

import numpy as np

import cv2

camera = PiCamera()

camera.resolution = (640, 480)

camera.framerate = 30

rawCapture = PiRGBArray(camera, size=(640, 480))

faceCascade = cv2.CascadeClassifier('haarcascade\_smile.xml')

for frame in camera.capture\_continuous(rawCapture, format="bgr", use\_video\_port=True):

frame = frame.array

#frame = cv2.flip(frame, 1) # Flip camera vertically

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

faces = faceCascade.detectMultiScale(gray, scaleFactor=1.2, minNeighbors=5, minSize=(20, 20))

for (x,y,w,h) in faces:

cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),2)

cv2.imshow('video',frame)

rawCapture.truncate(0)

if cv2.waitKey(1) & 0xFF == ord('q'):

break

cv2.destroyAllWindows()