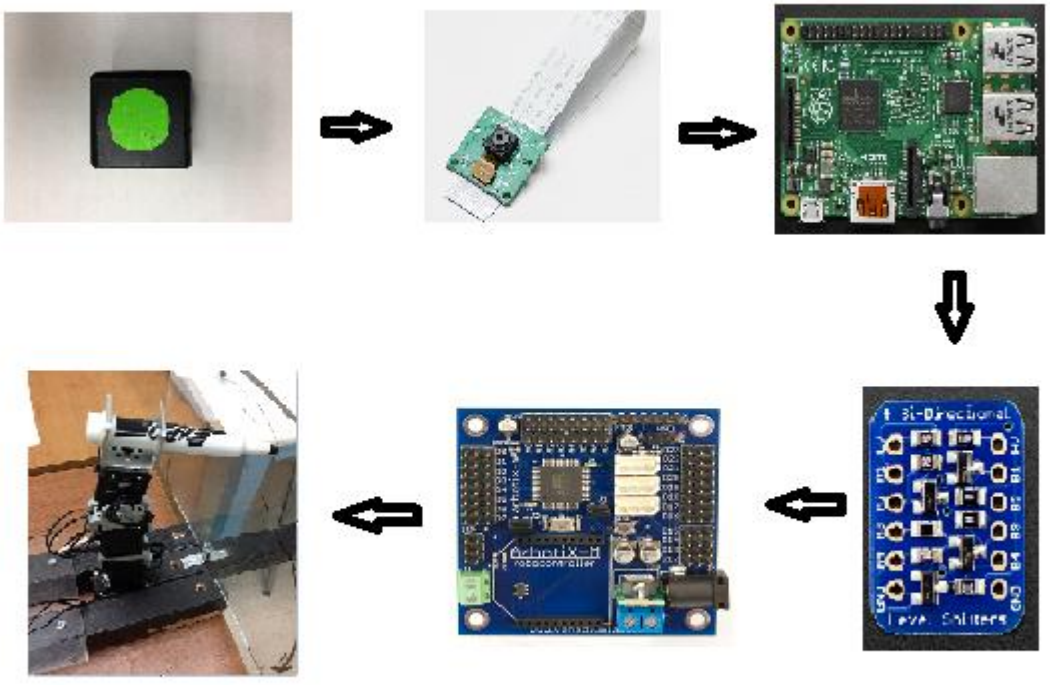
NYU | POLYTECHNIC SCHOOL OF ENGINEERING

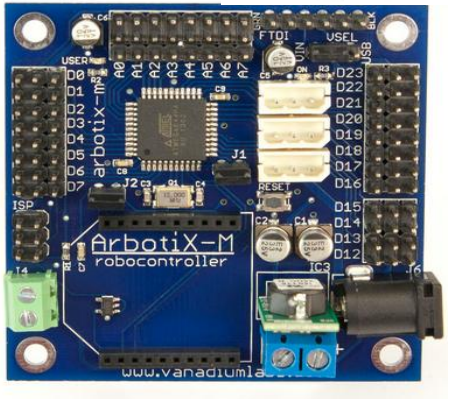
Vinci, the painter!

Sai Prasanth Krishnamoorthy
Mrudula Vaidya
Lalit Damodaran

- More than 100 million disabled
- Children with severe motor disabilities do not get to enjoy and express themselves properly
- The purpose of this project is to give an opportunity to the disabled to draw and sketch with ease using a robotic manipulator.



- **Robotic Arm:**
  4DOF RRRR Manipulator with Dynamixel Motors. A gripper is attached at the end of the manipulator to hold the pen.
- **Arbotix Robot Controller:**
  AVR-based robot controller used to actuate the dynamixel motors based on the joint angles



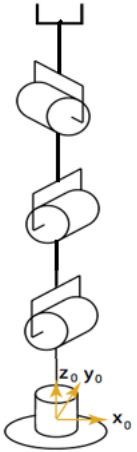
With $a_1 = 64mm$ and $a_2 = 107mm$, the DH table:

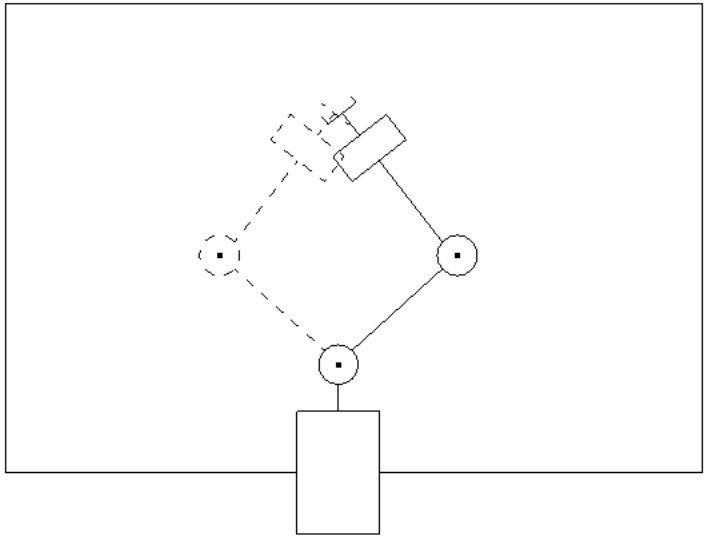
	α_i	a_i	d_i	θ_i
Link 2	0	a_1	0	θ_1
Link 3	0	a_2	0	θ_2

We get the forward kinematics equations:

$$x = a_2 \cos(\theta_1 + \theta_2) + a_1 \cos\theta_1$$
$$y = a_2 \sin(\theta_1 + \theta_2) + a_1 \sin\theta_1$$

The inverse kinematics(2)



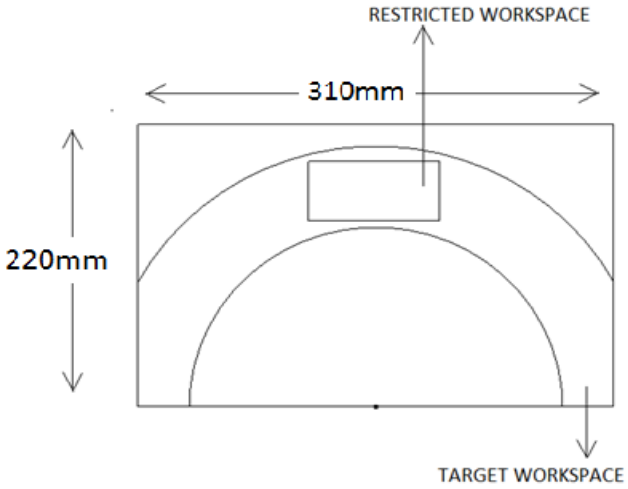


On performing inverse kinematics, we find:

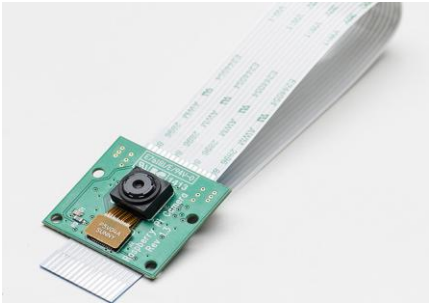
$$\boldsymbol{\theta_1} = \tan^{-1}\left(\frac{\boldsymbol{y}}{\boldsymbol{x}}\right) - \tan^{-1}\left(\frac{\boldsymbol{a_2}\sin\theta_2}{\boldsymbol{a_1} + \boldsymbol{a_2}\cos\theta_2}\right)$$

$$\boldsymbol{\theta_2} = \tan^{-1}\left(\pm\sqrt{\boldsymbol{1} - \left(\frac{\boldsymbol{x^2 + y^2 - a_1^2 - a_2^2}}{\boldsymbol{2a_1a_2}}\right)^2} \Big/ \frac{\boldsymbol{x^2 + y^2 - a_1^2 - a_2^2}}{\boldsymbol{2a_1a_2}}\right)$$

The workspace

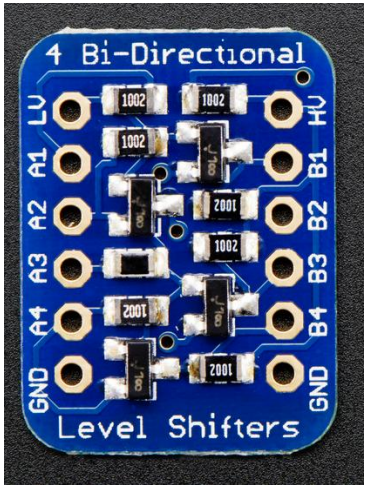


- **Raspberry Pi 2:**
  Uses OpenCV installed on the pi to track the green marker and perform the inverse kinematics based on the coordinates recorded.
- **Pi Camera:**
  Has an onboard sensor with 5MP resolution and fixed focus lens.



•**Logic-Level Convertor:**

The bi-directional convertor to convert between 3.3V and 5V when communicating serially between the Raspberry pi and Arbotix Board.



- **USART serial communication**
- **Using the ardupi library the pi programs are written as if writing an Arduino program.**

The Raspberry pi code (1/6)

```
#include<ctime>
#include<math.h>
#include<stdio.h>
#include <sstream>
#include <string>
#include<iostream>
#include<raspicam/raspicam_cv.h>
#include"opencv2/highgui/highgui.hpp"
#include"opencv2/core/core.hpp"
#include"opencv2/imgproc/imgproc.hpp"
#include "arduPi.h"

using namespace std;
using namespace cv;

//variables
int H_MIN = 30;
int H_MAX = 59;
int S_MIN = 162;
int S_MAX = 256;
int V_MIN = 100;
int V_MAX = 256;
```

```
//variables for kinematics and inverse
kinematics
float theta1;
float theta2;
float angle1;
float angle2;
double e;
double f;
int a = 64; //link l1 approx.
int b = 107; //link l2 approx.

//tracking variables
string messagex = "";
int nTrackCount = 0;
int nMaxPoints = 20;
vector <Point> tracePath;
vector< vector<Point> > finishedPath;
vector <Point> previousPath;
Scalar colorOfTrace(0,0,255); //red
Point pt;
int error1= 0;
Point current;
Point previous;
bool detected = false;
bool textDrawn = false;
int xAxis = 0;
int yAxis = 0;
int mappedX = 0;
int mappedY = 0;
```

```
//default capture width and height
const int FRAME_WIDTH = 1280;
const int FRAME_HEIGHT = 960;

//max number of objects to be detected in
frame
const int MAX_NUM_OBJECTS=50;

//minimum and maximum object area
const int MIN_OBJECT_AREA = 20*20;
const int MAX_OBJECT_AREA =
FRAME_HEIGHT*FRAME_WIDTH/1.5;

//names that will appear at the top of each
window
//const string windowName = "Original Image";
//const string windowName1 = "HSV Image";
//const string windowName2 = "Thresholded
Image";
//const string windowName3 = "After
Morphological Operations";
const string trackbarWindowName =
"Trackbars";

void on_trackbar( int, void* )
{//This function gets called whenever a
    // trackbar position is changed
}
```

The Raspberry pi code (2/6)

```
void createTrackbars(){
    //create window for trackbars
    namedWindow(trackbarWindowName,0);
    //create memory to store trackbar name on window
    char TrackbarName[50];
    sprintf( TrackbarName, "H_MIN", H_MIN);
    sprintf( TrackbarName, "H_MAX", H_MAX);
    sprintf( TrackbarName, "S_MIN", S_MIN);
    sprintf( TrackbarName, "S_MAX", S_MAX);
    sprintf( TrackbarName, "V_MIN", V_MIN);
    sprintf( TrackbarName, "V_MAX", V_MAX);
    //create trackbars and insert them into window
    //3 parameters are: the address of the variable that is
changing when the trackbar is moved(eg.H_LOW),
    //the max value the trackbar can move (eg. H_HIGH),
    //and the function that is called whenever the trackbar is
moved(eg. on_trackbar)
    //                                  ---->    ---->     ---->
    createTrackbar( "H_MIN", trackbarWindowName, &H_MIN, H_MAX,
on_trackbar );
    createTrackbar( "H_MAX", trackbarWindowName, &H_MAX, H_MAX,
on_trackbar );
    createTrackbar( "S_MIN", trackbarWindowName, &S_MIN, S_MAX,
on_trackbar );
    createTrackbar( "S_MAX", trackbarWindowName, &S_MAX, S_MAX,
on_trackbar );
    createTrackbar( "V_MIN", trackbarWindowName, &V_MIN, V_MAX,
on_trackbar );
    createTrackbar( "V_MAX", trackbarWindowName, &V_MAX, V_MAX,
on_trackbar );
}
```

```
string intToString(int number){
    std::stringstream ss;
    ss << number;
    return ss.str();
}
long mapVariable(long V, long in_min, long in_max, long
out_min, long out_max)
{
  return (V - in_min) * (out_max - out_min) / (in_max -
in_min) + out_min;
}
void drawObject(int x, int y, double areaOfImage,Mat &frame){
    //use some of the openCV drawing functions to draw
crosshairs
    //on your tracked image!
    //UPDATE:JUNE 18TH, 2013
    //added 'if' and 'else' statements to prevent
    //memory errors from writing off the screen (ie. (-25,-25)
is not within the window!)
    circle(frame,Point(x,y),20,Scalar(0,0,255),2);
    if(y-25>0)
    line(frame,Point(x,y),Point(x,y-25),Scalar(0,0,255),2);
    else line(frame,Point(x,y),Point(x,0),Scalar(0,0,255),2);
    if(y+25<FRAME_HEIGHT)
    line(frame,Point(x,y),Point(x,y+25),Scalar(0,0,255),2);
    else
line(frame,Point(x,y),Point(x,FRAME_HEIGHT),Scalar(0,0,255),2)
;
```

The Raspberry pi code (3/6)

```
if(x-25>0)
    line(frame,Point(x,y),Point(x-25,y),Scalar(0,0,255),2);
    else line(frame,Point(x,y),Point(0,y),Scalar(0,0,255),2);
    if(x+25<FRAME_WIDTH)
    line(frame,Point(x,y),Point(x+25,y),Scalar(0,0,255),2);
    else
line(frame,Point(x,y),Point(FRAME_WIDTH,y),Scalar(0,0,255),2);

putText(frame,intToString(x)+","+intToString(y)+","+intToString((i
nt)areaOfImage),Point(x,y+30),1,1,Scalar(0,0,255),2);

    //drawing line
   if(detected == false){
        tracePath.clear();
        tracePath.push_back(Point(x,y));
        detected = true;
    }
    else
    {
       tracePath.push_back(Point(x,y));
    }
}
void morphOps(Mat &thresh){
    //create structuring element that will be used to "dilate" and
"erode" image.
    //the element chosen here is a 3px by 3px rectangle
    Mat erodeElement = getStructuringElement(
MORPH_RECT,Size(3,3));
 //dilate with larger element so make sure object is nicely visible
    Mat dilateElement = getStructuringElement(
MORPH_RECT,Size(8,8));
    erode(thresh,thresh,erodeElement);
    dilate(thresh,thresh,dilateElement);
    dilate(thresh,thresh,dilateElement);
    erode(thresh,thresh,erodeElement);
}
void trackFilteredObject(int &x, int &y, Mat threshold, Mat
&cameraFeed){
    Mat temp;
    threshold.copyTo(temp);
    //these two vectors needed for output of findContours
    vector< vector<Point> > contours;
    vector<Vec4i> hierarchy;
    //find contours of filtered image using openCV findContours
function

findContours(temp,contours,hierarchy,CV_RETR_CCOMP,CV_CHAIN_APPROX_
SIMPLE );
    //use moments method to find our filtered object
    double refArea = 0;
    bool objectFound = false;
    if (hierarchy.size() > 0) {
        int numObjects = hierarchy.size();
        //if number of objects greater than MAX_NUM_OBJECTS we have
a noisy filter
```

The Raspberry pi code (4/6)

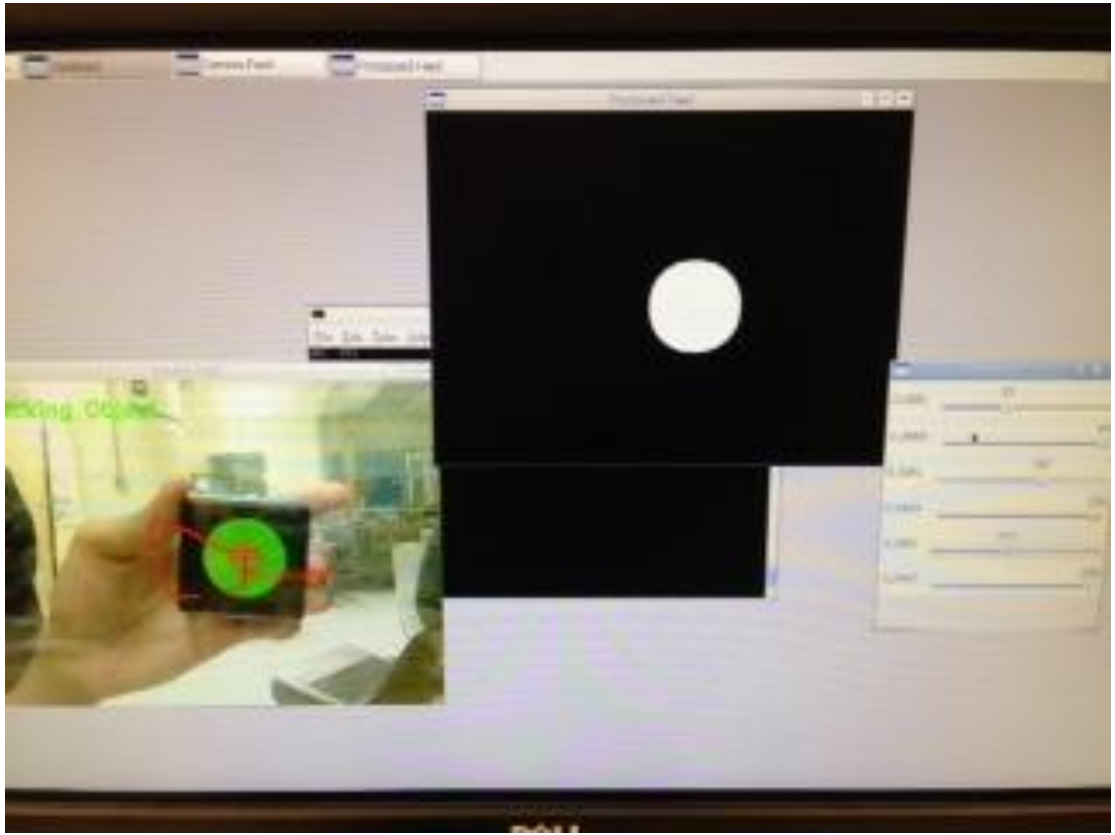
```
 if(numObjects<MAX_NUM_OBJECTS){
            for (int index = 0; index >= 0; index =
hierarchy[index][0]) {
                Moments moment = moments((cv::Mat)contours[index]);
                double area = moment.m00;
                //if the area is less than 20 px by 20px then it is
probably just noise
                //if the area is the same as the 3/2 of the image
size, probably just a bad filter
                //we only want the object with the largest area so we
safe a reference area each
                //iteration and compare it to the area in the next
iteration.
                if(area>MIN_OBJECT_AREA && area<MAX_OBJECT_AREA &&
area>refArea){
                    x = moment.m10/area;
                    y = moment.m01/area;
                    objectFound = true;
                    refArea = area;
                }else objectFound = false;
            }
            //let user know you found an object
            if((objectFound ==true)&& (refArea > 12000)){
                putText(cameraFeed,"Tracking
Object",Point(0,50),2,1,Scalar(0,255,0),2);
                //draw object location on screen
                drawObject(x,y,refArea,cameraFeed);
                //messagex << "@"<< x << "#" << y <<"$a" ;
                xAxis = x-339;
                yAxis = y-182;
                if(xAxis > 60)
                {
                    xAxis = 60;
                }
                else if(xAxis < -60){
                    xAxis = -60;
                }
                if(yAxis > 160)
                {
                    yAxis = 160;
                }
                else if(yAxis < 110){
                    yAxis = 110;
                }
                mappedX = mapVariable(xAxis,-60,60,60,-60);
                mappedY = mapVariable(yAxis,110,160,160,110);
                //inverse kinematics
                e = (pow(mappedX,2) + pow(mappedY,2) -
pow(a,2) - pow(b,2)) / (2*a*b) ;
                f = sqrt(1 - pow(e,2));
                theta2 = atan2(f,e);
                theta1 = atan2(mappedY,mappedX) -
atan2((b*sin(theta2)),(a+(b*cos(theta2))));
                angle1 = 3.41*(theta1*57.295+60);
                angle2 = 510 - 3.41*theta2*57.295;
                cout << intToString(angle1) << "   " <<
intToString(angle2) << endl;
                messagex =
"@"+intToString(angle1)+"#"+intToString(angle2)+"$a";
                Serial.print(messagex.c_str());
//
Serial.write(messagex.c_str(),messagex.size());
                //Serial.print("@-60#160$a");
            }
```

The Raspberry pi code (5/6)

```
else
        {
            detected = false;
           // textDrawn = true;
            if(!tracePath.empty())
            {
              finishedPath.push_back(tracePath);
              tracePath.clear();
            }
            xAxis = x-339;
            yAxis = y-182;
            if(xAxis > 60)
            {
                xAxis = 60;
            }
            else if(xAxis < -60){
                xAxis = -60;
            }
            if(yAxis > 160)
            {
                yAxis = 160;
            }
            else if(yAxis < 110){
                yAxis = 110;
            }
            //inverse kinematics
            mappedX = mapVariable(xAxis,-
60,60,60,-60);
 mappedY = mapVariable(yAxis,110,160,160,110);
                e = (pow(mappedX,2) + pow(mappedY,2) - pow(a,2) - pow(b,2)) / (2*a*b)
;
                f = sqrt(1 - pow(e,2));
                theta2 = atan2(f,e);
                theta1 = atan2(mappedY,mappedX) -
atan2((b*sin(theta2)),(a+(b*cos(theta2))));
                angle1 = 3.41*(theta1*57.295+60);
                angle2 = 510 - 3.41*theta2*57.295;
                cout << intToString(angle1) << "  " << intToString(angle2) << endl;
                messagex = "@"+intToString(angle1)+"#"+intToString(angle2)+"$b";
               // messagex << "@"<< x << "#" << y <<"$b" ;
                Serial.print(messagex.c_str());
                //Serial.write(messagex.c_str(),messagex.size());
                //Serial.print("@-60#160$b");
              //  tracePath.clear();
                //textDrawn = false;
            }
        }else putText(cameraFeed,"TOO MUCH NOISE! ADJUST
FILTER",Point(0,50),1,2,Scalar(0,0,255),2);
    }
}

void setup(){
  Serial.begin(9600);
 // cout >> "Serial Port Opened" >> endl;
}
```

The Raspberry pi code (6/6)

```
void loop()
{
   raspicam::RaspiCam_Cv Camera;
    //functions
    bool trackObjects = true;
    bool useMorphOps = true;
    int x=0, y=0;
    Mat image,resized,HSV,threshold;
    cout <<"Opening camera" << endl;
    if(!Camera.open()){
    cerr<<"Error opening camera" << endl;
    }
    createTrackbars();
    sleep(3);
    Camera.set(CV_CAP_PROP_FRAME_WIDTH,FRAME_WIDTH);
    Camera.set(CV_CAP_PROP_FRAME_HEIGHT,FRAME_HEIGHT);
    Camera.set(CV_CAP_PROP_BRIGHTNESS,50);
    Camera.set(CV_CAP_PROP_CONTRAST,50);
    namedWindow("Camera Feed", CV_WINDOW_AUTOSIZE);
    namedWindow("Processed Feed",CV_WINDOW_AUTOSIZE);
    while(waitKey(1) != 'q'){
     //grab new frame
 Camera.grab();
        Camera.retrieve(image);
        resize(image,resized,Size(),0.5,0.5,INTER_LINEAR);
        cvtColor(resized,HSV,CV_BGR2HSV);

inRange(HSV,Scalar(H_MIN,S_MIN,V_MIN),Scalar(H_MAX,S_MAX,V_MAX
),threshold);
 if(useMorphOps){
        morphOps(threshold);}
        if(trackObjects){
            trackFilteredObject(x,y,threshold,resized);}
        if(!finishedPath.empty())
         {
            for(int k = 0 ; k  < finishedPath.size() ; k++){
                previousPath = finishedPath[k];
             for( int iTrack = 1; iTrack < previousPath.size()-1 ; iTrack++){
                line(resized,previousPath[iTrack-
1],previousPath[iTrack],colorOfTrace,3);
             }
            }
         }
        if(!tracePath.empty()){
        for( int iTrack = 1; iTrack < tracePath.size()-1 ; iTrack++){
           line(resized,tracePath[iTrack-1],tracePath[iTrack],colorOfTrace,3);
        }
        }
        imshow("Camera Feed", resized);
        imshow("Processed Feed", threshold);
        waitKey(30);
    }
}

int main( int argc, char **argv){
    setup();
    while(1){
      loop();
    }
    return 0;
}
```



The Arduino code (1/3)

```
#include <SoftwareSerial.h>
#include <ax12.h>
#include <math.h>
#define setSpeed(id, spd)
(ax12SetRegister2(id, AX_GOAL_SPEED_L  ,
spd))


SoftwareSerial mySerial(24,25); // RX,
TX
boolean stateTransition = false;
char ch;
char Xc[5];
char Yc[5];
int t1 = 0;
int t2 = 0;
boolean pen = false;
int index = 0;
boolean val1 = false;
boolean val2 = false;
boolean val3 = false;
boolean val1Neg = false;
boolean val2Neg = false;

double a = 64;
double b = 107;
float theta1;
float theta2;
double e,f,g;
float pos1;
float pos2;
//test
int angle1,angle2;

void setup()
{
  // Open serial communications and wait for port to open:
  initialize();
  Serial.begin(9600);
  mySerial.begin(9600);
 Serial.println("Connected");  // wait for serial port to connect. Needed
for Leonardo only
  Serial.println("Goodnight moon!");
  // set the data rate for the SoftwareSerial port
  //Serial.begin(9600);
  Serial.println("Hello, world?");
}
```

The Arduino code (2/3)

```
void loop() // run over and over
{
  if (mySerial.available()){
    ch = mySerial.read();
   // Serial.print(ch);
    if(ch == '@')
    {
       val1 = true;
       val2 = false;
       val3 = false;
       index = 0;
    }
    else if(ch == '#')
    {
      val1 = false;
      val2 = true;
      val3 = false;
       index = 0;
    }
     else if(ch == '$')
    {
      val1 = false;
      val2 = false;
      val3 = true;
       index = 0;
    }
```

```
else{
      handleData();
    }
  }
  else{
  }
}
void handleData()
{
  if(val1 == true)
  {
   // Serial.println("In val 1");
   if(ch == '-')
   {
     val1Neg = true;
   }
   else{
    Xc[index] = ch;
    index++;
   }
  }
  else if(val2 == true)
  {
   // Serial.println("In val 2");
   if(ch == '-')
   {
     val2Neg = true;
   }
```

```
else{
   Yc[index] = ch;
   index++;
  }
 }
else if(val3 == true)
 {
  // Serial.println("In val 3");
   if(ch == 'a')
   {
    pen = true;
   }
   else if(ch == 'b')
   {
     pen = false;
   }
   if(val1Neg){

     t1 = -1 * atoi(Xc);
     val1Neg  = false;
   }
   else{
     t1 = atoi(Xc);
   }
   if(val2Neg){

     t2 = -1 * atoi(Yc);
     val2Neg = false;
   }
```

The Arduino code (3/3)

```
else{
     t2 = atoi(Yc);
    }
//t1 = atoi(Xc);
   //t2 = atoi(Yc);
   if(pen == true){
    Serial.print("Theta1 = ");
   Serial.print(t1);
   Serial.print(" Theta2 = ");
   Serial.print(t2);
   Serial.println(" Penned DOWN ");
   SetPosition(4,500);
   }
   else{
     Serial.print("Theta1 = ");
   Serial.print(t1);
   Serial.print(" Theta2 = ");
   Serial.print(t2);
   Serial.println(" Penned UP ");
   SetPosition(4,530);
   }
memset(Xc, 0, sizeof(Xc));
 memset(Yc, 0, sizeof(Yc));
 moveArm(t1,t2);
 }
else{
 }
}
```

```
void initialize()
{
  setSpeed(1,100);
  setSpeed(2,100);
  setSpeed(3,100);
  setSpeed(4,100);
   SetPosition(1,810); //set the
position of servo # 1 to '0'
  SetPosition(2,510);
  SetPosition(3,510);
  SetPosition(4,550);  //610
exetended
  setSpeed(1,200);
  setSpeed(2,200);
  setSpeed(3,200);
  setSpeed(4,400);
  delay(100);
}
```

```
void moveArm(double x , double y)
{
  //maps
  x =map(x,-60,60,60,-60);
  y =map(y,110,160,160,110);
  e = (sq(x)+ sq(y) - sq(a) - sq(b))/(2*a*b);
  f =  sqrt(1 - sq(e));
  theta2 = atan2(f , e );
  theta1 = atan2(y,x) -
atan2((b*sin(theta2)),(a+(b*cos(theta2))));
  pos1 = 3.41*(theta1*57.295+60);
  pos2 = 510 - 3.41*theta2*57.295;
  SetPosition(2,pos1);
  SetPosition(3,pos2);
}
```

THANK YOU