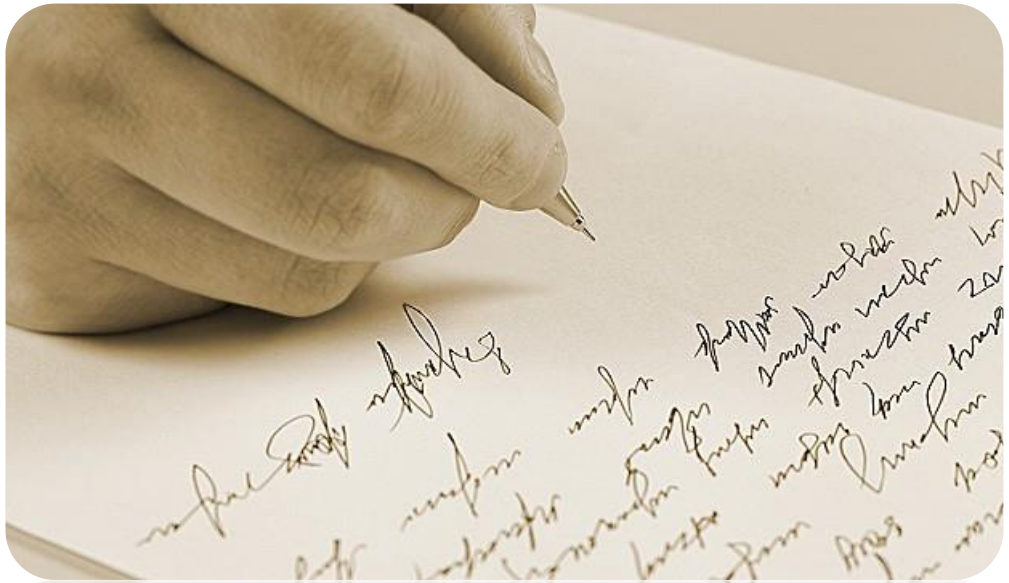
CATCH ME IF YOU CAN…

Advanced Mechatronics :

Final Project



Presented By:

Federico Gregori

Karim Chamaa

Presented to:
Dr. Vikram Kapila

Outline

- Introduction
- System Description
- Improvements
- Coding
- Components
- Technical Specifications
- Cost Analysis
- Future Improvements
- Conclusion

Introduction

- Design a writing and drawing machine capable of mimicking a paint or captured image.
- Goal is to implement the Raspberry Pi to provide on-board computational power
- Improve and modify the system in order to achieve better results

System Description

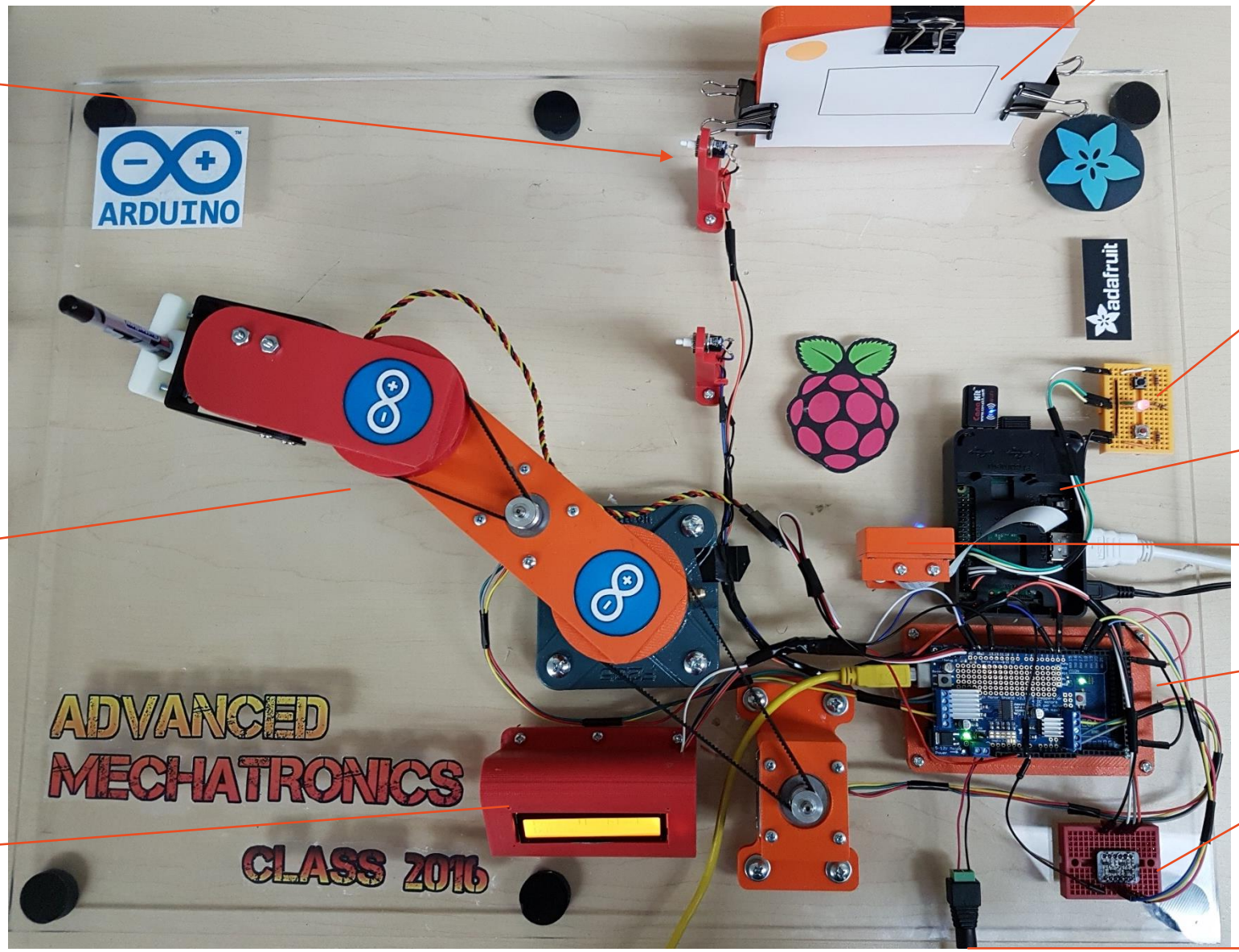


Image To Capture

Stoppers

Push Buttons

Raspberry Pi

Manipulator

Camera

Arduino Mega +
Stepper Driver

Logic Level Shifter

LCD

System Power

Improvements

Servo Motors	→	Steppers Motors + Belt + Stoppers
LabVIEW	→	Raspberry Pi
Paint Image	→	Paint Image + Camera Image
Data Manually Copied	→	Data Transmitted Serially
Need of Pc for Feedback	→	LCD for feedback

Result: Achieved a **Stable**, **Stand-Alone** and **Autonomous system**

Coding
Python Transmitter

STEP1  Import Packages, setup LED's and acquire choice (Camera or Paint Image)

```
import serial
import time
import picamera
import math
import array
import PIL
import numpy
from PIL import Image
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
buttonPin = 22
ledPin1 = 23
ledPin2 = 24
GPIO.setup(buttonPin,GPIO.IN,pull_up_down=GPIO.PUD_UP)
GPIO.setup(ledPin1,GPIO.OUT)
GPIO.setup(ledPin2,GPIO.OUT)

while True:
    GPIO.output(ledPin1,GPIO.HIGH)
    GPIO.output(ledPin2,GPIO.LOW)
    print("System started")
    while True:
        inputValue=GPIO.input(buttonPin)
        if(inputValue == False):
            print("Black button pressed")
            GPIO.output(ledPin1,GPIO.LOW)
            Selection='1'
            break
```

Coding
Python Transmitter

STEP2 Transforming Image into matrix form depending on choice selected

```
if Selection=='0':
    MatrixImage=numpy.asarray(Image.open('Paint.jpg').convert('L'))
elif Selection=='1':
    with picamera.PiCamera() as camera:
        camera.resolution=(2592,1944)
        camera.brightness=40
        camera.sharpness=100
        camera.start_preview()
        time.sleep(5)
        camera.capture('CameraImage.jpg')
        camera.stop_preview()

    MatrixImage=[[255 for x in range(210)] for y in range(90)]#Define Matrix 9(
    #scale the image to a smaller one by changing the basewidth 115- 86
    basewidth=80
    img=Image.open('CameraImage.jpg')
    img=img.crop((880,580,1820,1100))
    wpercent=(basewidth/float(img.size[0]))
    hsize=int((float(img.size[1])*float(wpercent)))
    img=img.resize((basewidth,hsize),PIL.Image.ANTIALIAS)
    img.save('ScaledCamera.jpg')
    #Scaled Iage to Matrix
    CameraImage=numpy.asarray(Image.open('ScaledCamera.jpg').convert('L'))
    #Add Values to the workspace side of Matrix
    for i in range(len(CameraImage)):#Scan Matrix up to down
        for j in range (len(CameraImage[0])):# Row outer loop and column inner
            MatrixImage[i+45][j+130]=CameraImage[i][j]
```

Coding
Python Transmitter

STEP3  Extracting points by setting a threshold

```
#Find X and Y not mapped
#Replace by indexes and get Mapped ValuesXN
XNotMapped=list()
YNotMapped=list()
XNotMappedGlobal=list()
YNotMappedGlobal=list()
for i in range(len(MatrixImage)):#Scan Matr
    for j in range (len(MatrixImage[0])):#
        if MatrixImage[i][j]<50:
            XNotM=i;
            YNotM=j;
            XNotMapped.append(XNotM)
            YNotMapped.append(YNotM)
            XNotMappedGlobal.append(XNotM)#
            YNotMappedGlobal.append(YNotM)
```

Coding
Python Transmitter

STEP4 → Mapping the points using an efficient algorithm

```
#Map x and Y
indexArray=list()
IndexArrayMapped=list()
LiftArray=list()
Newi=0 #Since I am not able to change indices of for loop
        # Python Start at 0
for i in range(len(XNotMapped)):
    for j in range(len(XNotMapped)):
        MinDis=math.sqrt(pow((XNotMapped[j]-XNotMapped[Newi]),2) + pow((YNotMapped[j]-YNotMapped[Newi]),2))
        indexArray.append(MinDis)
    if 1 in indexArray:
        index=indexArray.index(1)
        Lift=0
    elif math.sqrt(2) in indexArray:
        index=indexArray.index(math.sqrt(2))
        Lift=0
    else:
        if(i==len(XNotMapped)-1):
            Lift=1
            index=indexArray.index(math.sqrt(0))
        else:
            for k in range(len(indexArray)):
                if (indexArray[k]<100 and indexArray[k]>0):
                    index=k
                    Lift=1

    XNotMapped[Newi]=10000
    YNotMapped[Newi]=10000
    IndexArrayMapped.append(Newi)
    LiftArray.append(Lift)
    Newi=index
    indexArray=list()#Need to empty it otherwise it append
```

Coding
Python Transmitter

STEP5  Applying Inverse Kinematic Equations

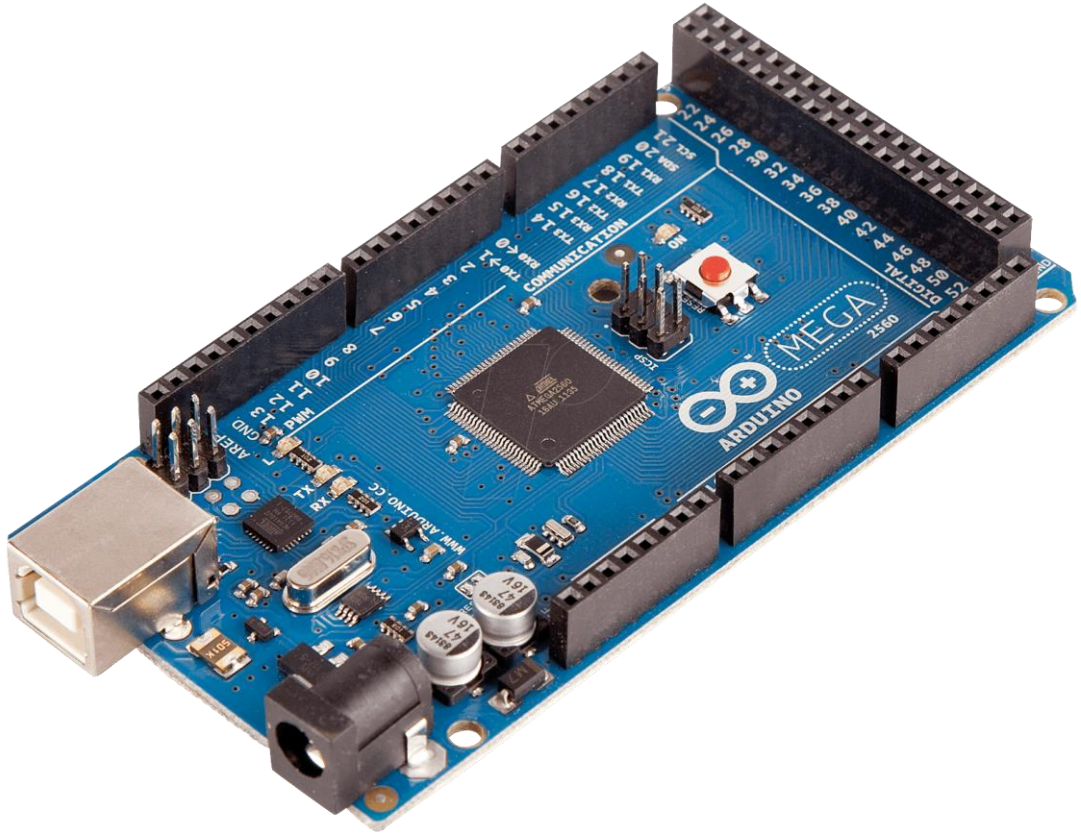
```
#Replace by indexes and get Mapped Values
XMapped= list()
YMapped=list()
for i in range(len(IndexArrayMapped)):
    XMapped.append(XNotMappedGlobal[IndexArrayMapped[i]])
    YMapped.append(YNotMappedGlobal[IndexArrayMapped[i]])
print("XMapped=",XMapped)
print("YMapped=",YMapped)

#Apply Inverse Kinematics Equations
Angle2= list()
Angle1=list()
Step1Array=list()
Step2Array=list()
for i in range(len(XMapped)):
    D=(pow(XMapped[i],2)+pow(YMapped[i],2)-pow(132,2)-pow(100,2))/(2*132*100)
    Ang2=math.atan2(math.sqrt(1-pow(D,2)),D)
    Angle2.append(math.degrees(Ang2))
    A=100*math.sin(Ang2)
    B=132+(100*math.cos(Ang2))
    Ang1=math.atan2(YMapped[i],XMapped[i])-math.atan2(A,B)
    Angle1.append(math.degrees(Ang1))
    Step1=int(round((math.degrees(Ang1)*470)/9))
    Step2=int(round((math.degrees(Ang2)*470)/9))
    Step1Array.append(Step1)
    Step2Array.append(Step2)
```

Coding
Python Transmitter

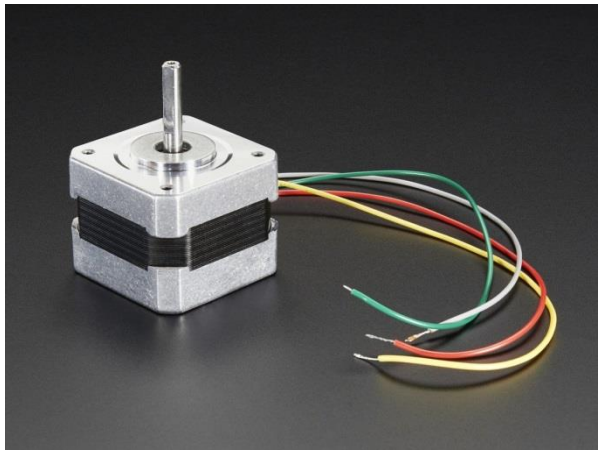
STEP6  Transmitting data serially to Arduino

```
GPIO.output(ledPin2,GPIO.HIGH)
print("Computation done")
#Serial Part
ser=serial.Serial('/dev/ttyAMA0',9600)
startmsg=str(b'r')
ser.write(bytes(startmsg,encoding="ascii"))
while True:

      if ser.read(1) == b'S':
         GPIO.output(ledPin2,GPIO.LOW)
         print("Red button pressed")
         break

for i in range(len(Step1Array)):
    message1=str(Step1Array[i]+1000)
    message2=str(Step2Array[i]+1000)
    message3=str(LiftArray[i])
    ser.write(bytes(message1,encoding="ascii"))
    ser.write(bytes(message2,encoding="ascii"))
    ser.write(bytes(message3,encoding="ascii"))
    #time.sleep(0.5)
    while True:
         if ser.read(1) == b'S':
            break
```

Coding Arduino

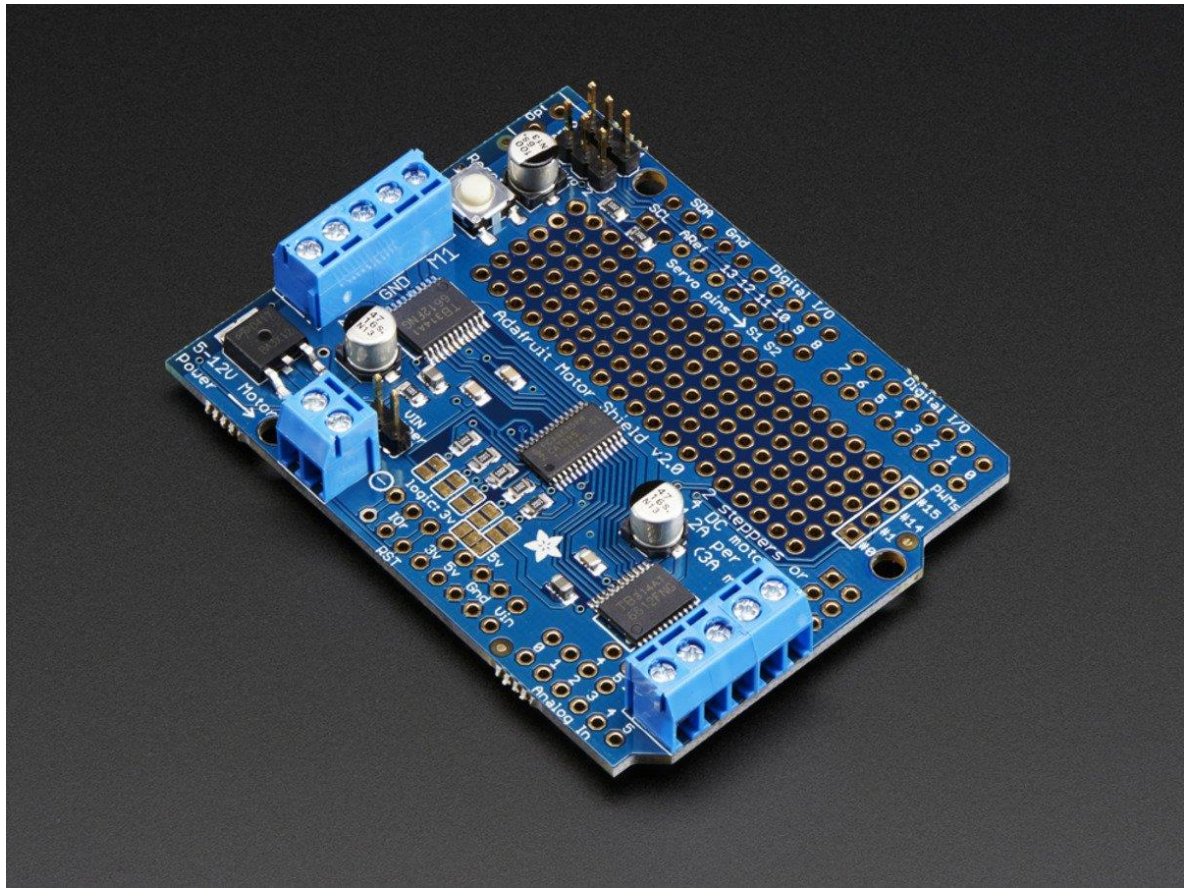


Components



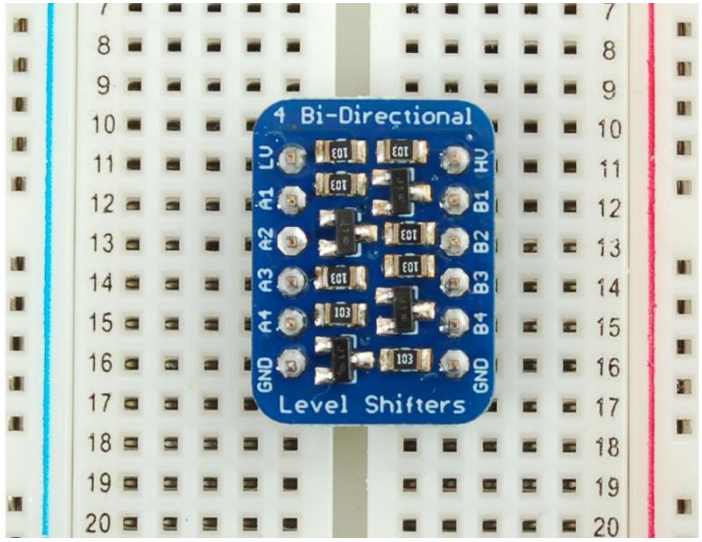
- Stepper motors by Adafruit
- 350 mA , 12v
- 200 steps/revolution
- Up to 18,800 steps/revolution with gear reduction and microsteps function.

Components



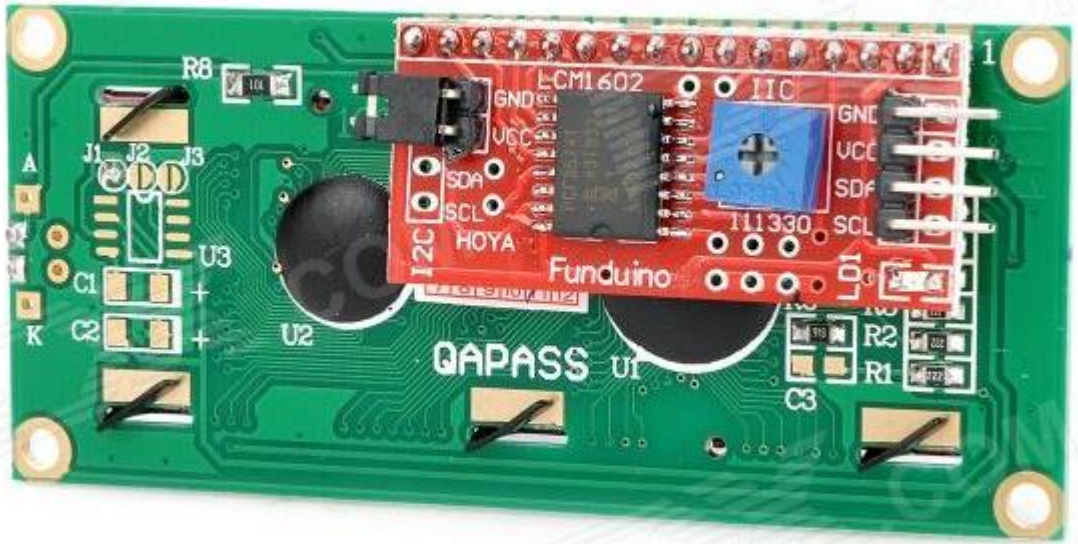
- Adafruit Motorshield V2
- Up to 2 steppers and 2 servos working together
- Addressable I2C communication

Components



- 4 Bi-Directional Adafruit Logic Level Shifter
- Allows communication between R-Pi and Arduino

Components



- I2C LCD by Geeetech
- Default I2C address 0x27
- Orange backlit

Components

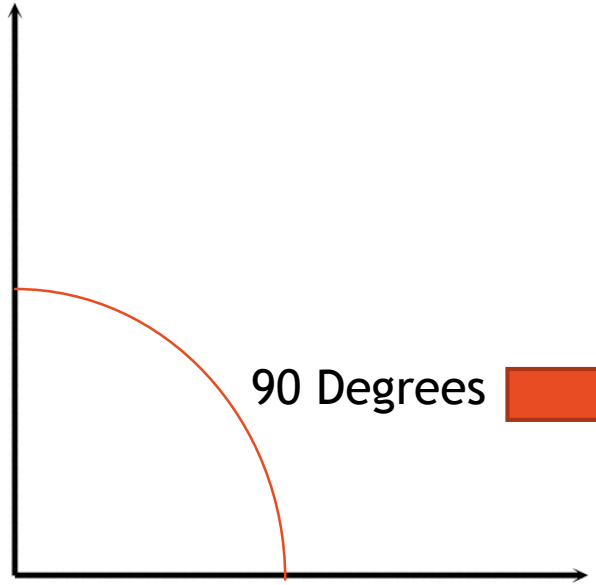
- Push buttons

- Bicolor Led

User control and feedback

Technical Specifications
System Speed and Number of Angles

Gear Ratio 1:6

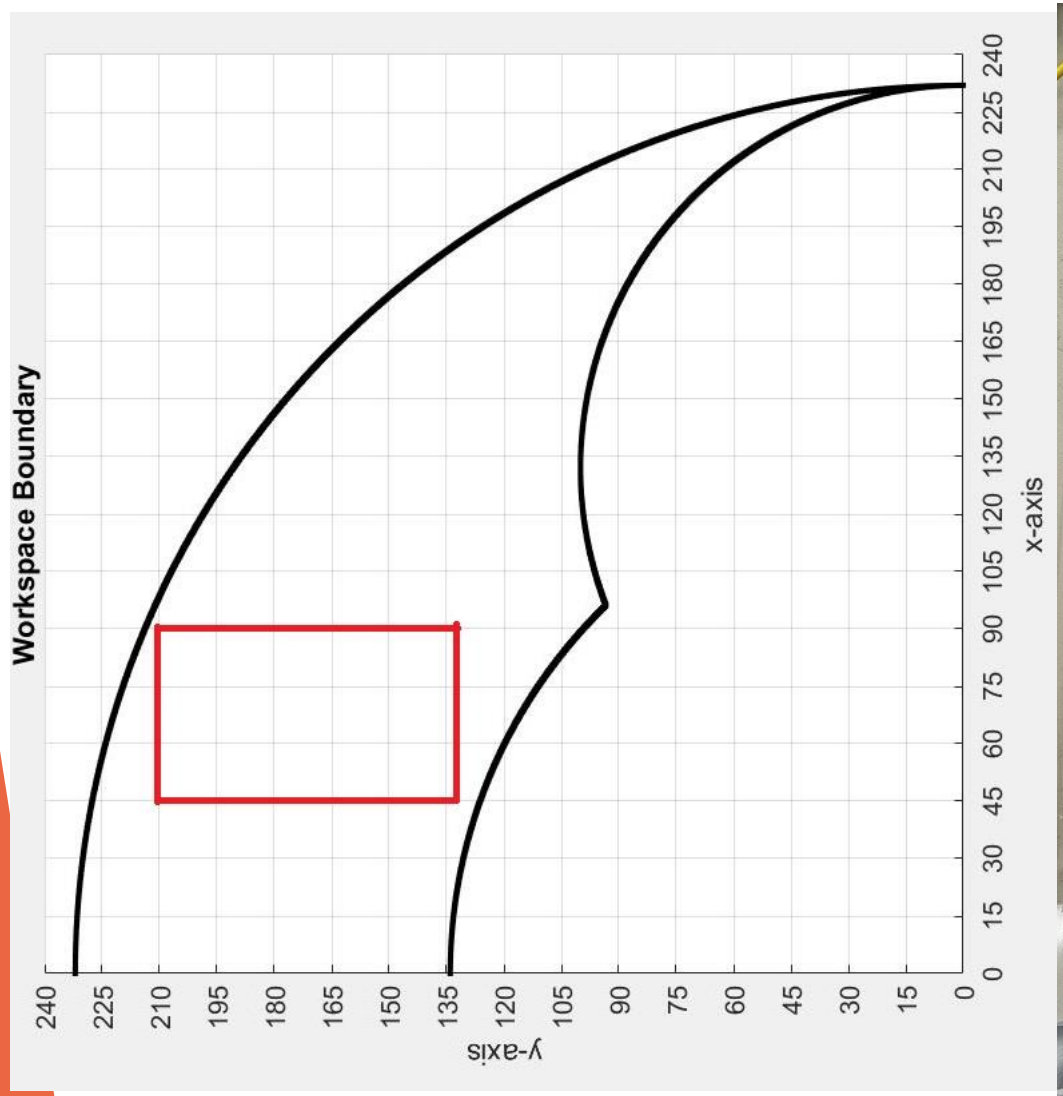


90 Degrees → 47000Steps

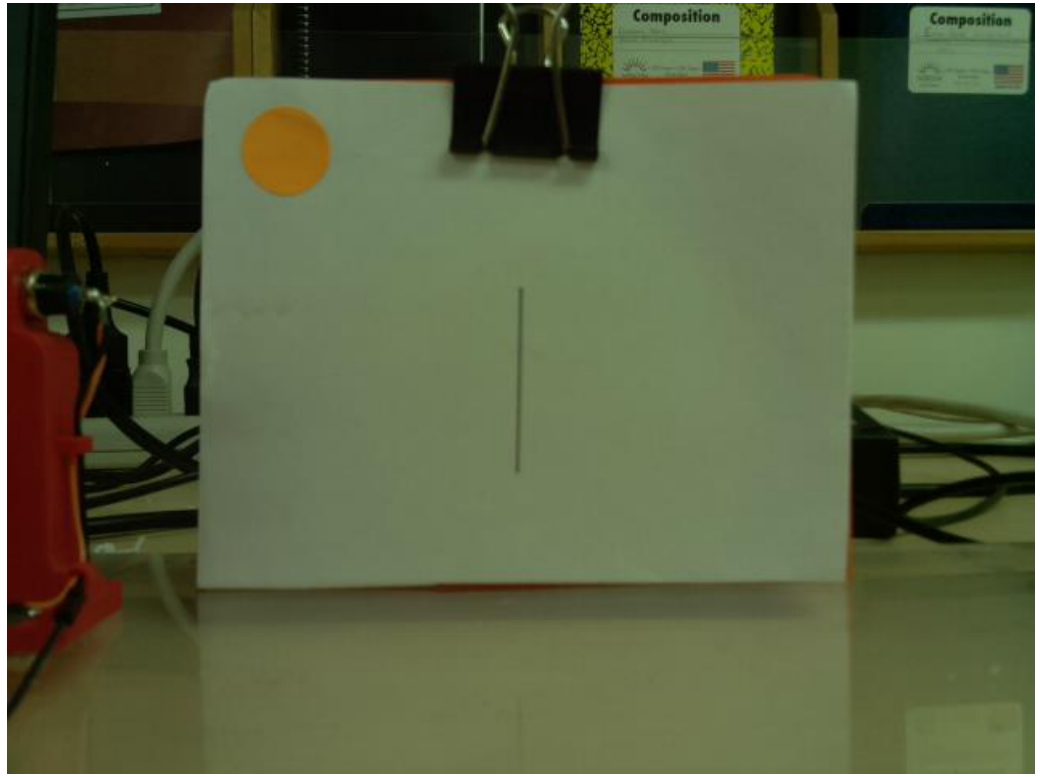
Perimeter:160mm
Draw Time :3 minutes

0.88 mm/second

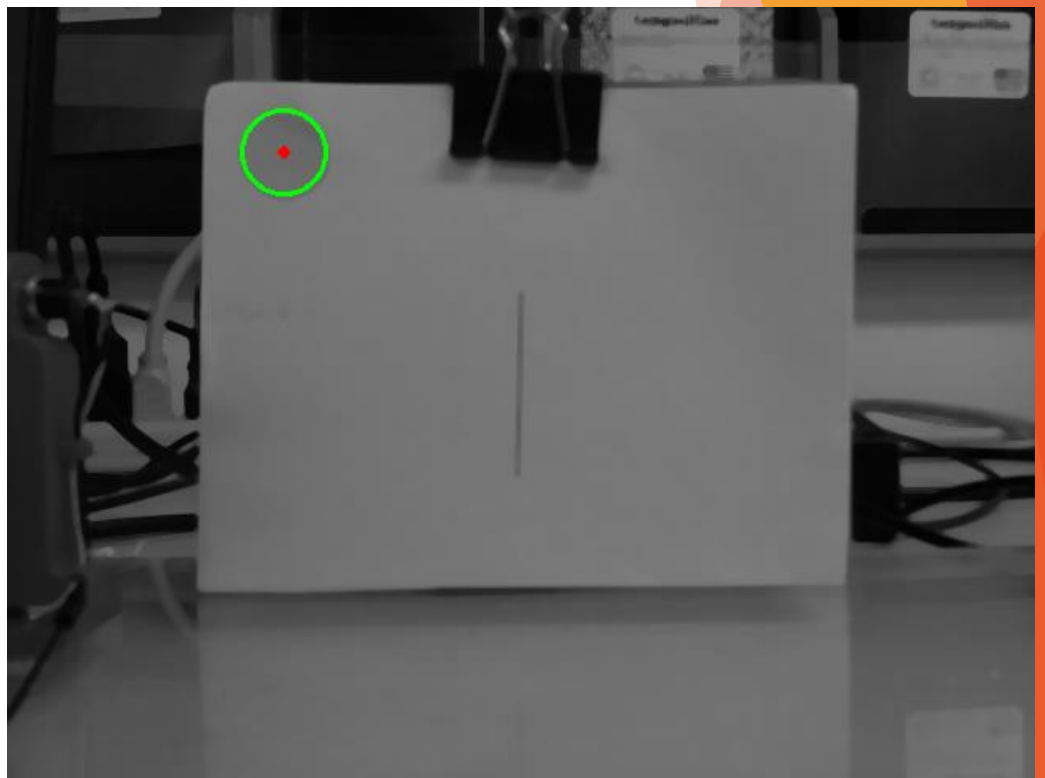
Technical Specifications
Workspace Area



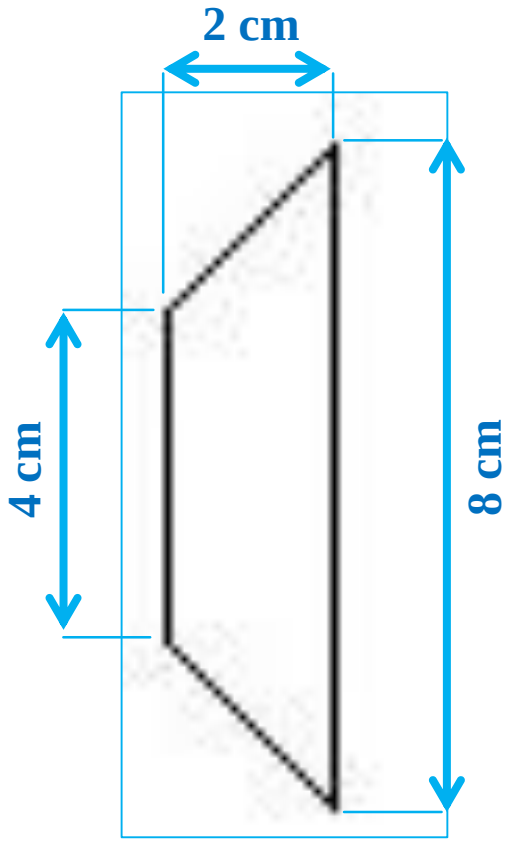
Technical Specifications
Captured Image Scale

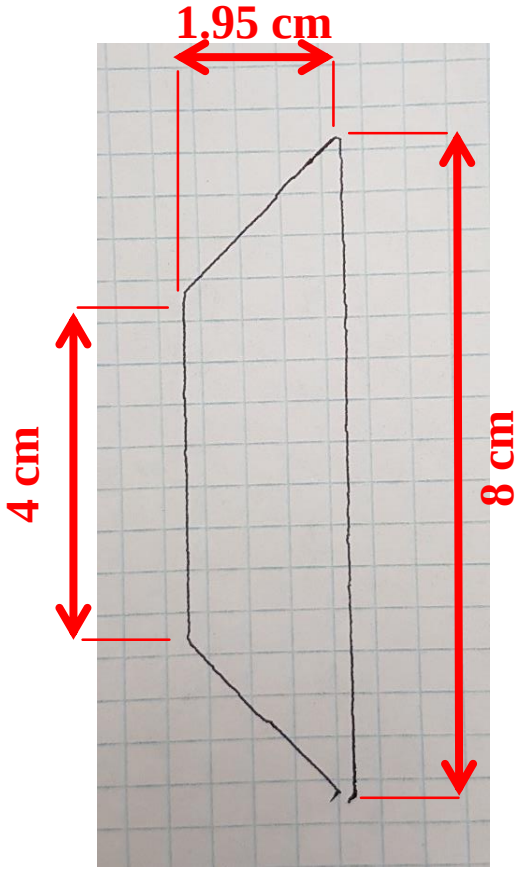


SCALE
1Pixel=1mm



Technical Specifications
Accuracy





- The error evaluated is 0.8%*

$$\epsilon = \frac{1}{3}\left(\frac{|l_1 - l_1^*|}{l_1^*} + \frac{|l_2 - l_2^*|}{l_2^*} + \frac{|l_3 - l_3^*|}{l_3^*}\right)$$

* The tecnique used is the mean value of the relative error of the three mesuraments.

Technical Specifications Accuracy

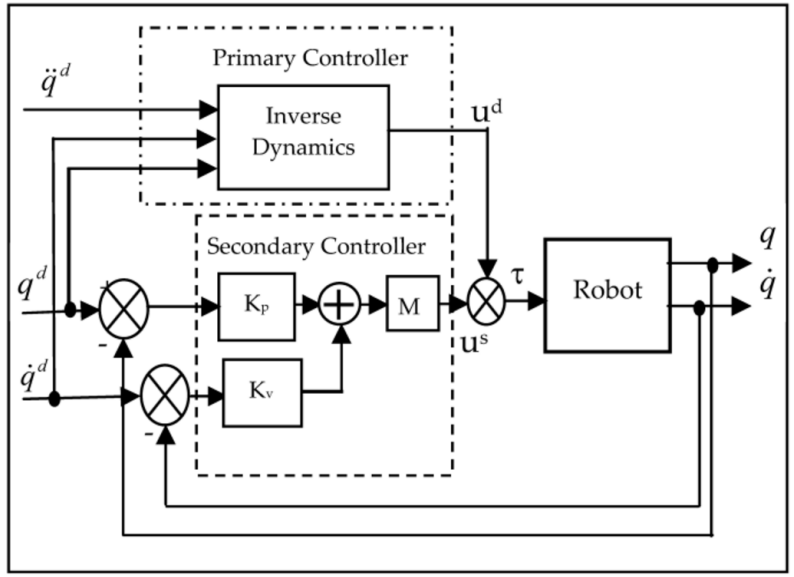
Board	Number of Processors	Motors	Function	%Error
Arduino	2	Servo	LabVIEW Programming	1.2
Arduino	1	Servo	writeMicroseconds()	1.6
Propeller	2	Servo	Servo_angle	5.8
Propeller	1	Servo	Servo_angle	7.8
Propeller	1	Servo	Pulse_out	2
Propeller	2	Servo	Pulse_out	1.2
Arduino+Pi	1	Steppers	AccelMotor Libraty	0.8

Cost Analysis

Materials	Quantity Usage	Unit of Measure	Unit Cost	Usage Cost
Plexiglas	1	Each	24	24$
Raspberry Pi	1	Each	35$	35$
Steppers + Board	2	Each	25$	50$
Printing Parts	2	Each	25$	50$
Arduino Mega	1	Each	30$	30$
Servo	1	Each	15$	15$
Voltage Converter	1	Each	12$	12$
LCD	2	Each	15$	15$
Others	1	Each	25$	25$
		Prototype Total Cost=		256$

Future Improvements

- **Path Planning**: Fitting trajectories (example: Cubic or sinusoidal) between desired joint variables at discrete points in time.
- **Control**: Designing an inverse proportional controller or PD in order to minimize the error over time. A combination of encoders and tachometers must be used in order to provide feedback.



Conclusions

- We achieved better results by replacing the servo motors by stepper motors since the range of angle and torque increases.
- We were able to design a stand alone system by the help of raspberry pi and eliminated the need of LabVIEW.
- To achieve better results more efficient algorithms and controllers should be used

Thank You

Questions ?