Prototype 3 and Customer Feedback

Shehryar Ali Memon

Oluwatamilore Ilupeju

Logan Jones

Saheel Ahmed

March 27, 2022

Abstract

This document is meant to portray our third prototype and analyze our inverse kinematics solutions for our end-effector. It also serves the purpose of defining any challenges we faced during testing and prototyping and makes clear our plan for our final product.

Table of Contents

Introduction

This document follows up on the last document, in which we defined our second prototype test plan and our stopping criteria. This document analyzes our final prototype by providing images of our final 3D print of our sander and the motor housing, the initial 3D prints of our paint gun, our final inverse kinematics solution, and our reworked GUI. It further defines our next steps for our future prototypes by virtue of another prototype test plan and makes known the software and hardware challenges we faced during prototyping. It also aims to clarify our design day readiness.

Prototyping test plan

Test ID	**Test Objectives**	**Description of prototypes and basic test method**	**Description of results to be recorded and how these results will be used**	**Estimated test duration and planned start date**
01	Paint gun 3D print	Final product of paint gun	Efficiency checked once printed	03/28/2022 5 hour(s)
02	Minimum Viable (MVP) Product test	Efficiency of whole system checked	Results will be improved upon, or MVP will be revisited	03/28/2022 - 03/30/2022 2 day(s)
03	Design Day readiness	Aesthetics and safety checked	Cleaning up of final details	03/30/2022 5 hour(s)

04	GUI	Using a simple GUI to control robotic arm	If instructions are properly followed and arm does what is required, python code will be reprogrammed if found not working	03/27/2022-03/30/2022 4 day(s)
06	IR sensor	Fitting the IR sensor at the desired position and final programming	Efficiency checked of code; final testing	03/28/2022 2 hour(s)
07	Ultrasound sensor	Fitting the ultrasound sensor at the desired position and final programming	Viability checked; final testing	03/28/2022 2 hour(s)
08	Inverse Kinematics solution	Programming solution to work manually rather than looped inputs	Once completed, program to work with GUI	03/28/2022 5 hour(s)

Prototype Analysis

Final Inverse Kinematics Solver

This is our final inverse kinematics program, coded on the Arduino IDE. Our initial solver allowed our robot to move, however it was jittery and not smooth as seen here: Movement due to initial IK solution. Our reworked solution allowed us to move smoothly (the delays are larger to make safe use of the arm) in 2 dof as seen here in Movement due to reworked IK solution; 2

dof. We were unable to come up with video evidence in time for the 3 dof solution, however our arm moved as expected.

```
   ESP32 PCA9685 Servo Control
    esp32-pca9685.ino
   Driving multiple servo motors with ESP32 and PCA9685 PWM module
   Use I2C Bus

   DroneBot Workshop 2020
    https://dronebotworkshop.com
*/

// Include Wire Library for I2C
#include <Wire.h>
#include  <InverseK.h>
// Include Adafruit PCA9685 Servo Library
#include  <Adafruit_PWMServoDriver.h>

volatile float L1;
volatile float L2;
// end effector

volatile float pi = 3.14159265359;
// Creat object to represent PCA9685 at default I2C address
Adafruit_PWMServoDriver  pca9685  =  Adafruit_PWMServoDriver(0x40);

// Define maximum and minimum number of "ticks" for the servo motors
// Range from 0 to 4095
// This determines the pulse width

#define SERVOMIN   80   // Minimum value
#define SERVOMAX   600   // Maximum value

// Define servo motor connections (expand as required)
#define SER0   0    //Servo Motor 0 on connector 0
#define SER1   1   //Servo Motor 1 on connector 12
#define SER2   2

// Variables for Servo Motor positions (expand as required)
int pwm0;
int pwm1;
int pwm2;
float x=1;
float y=1;
float z=1;
int i;
float  angle1;
float  angle2;
float  angle3;
double  rad_angle1;
float  rad_angle2;
```

```
float  rad_angle3;
float  preangle1=90;
float  preangle2=90;
float  preangle3=90;
void setup() {

  // Serial monitor setup
   Serial.begin(115200);

  // Print to monitor
   Serial.println("PCA9685 Servo Test");

  // Initialize PCA9685
   pca9685.begin();

  // Set PWM Frequency to 50Hz
   pca9685.setPWMFreq(50);
 Serial.println("Enter the length of first arm ");
  while(Serial.available()==0){}
  //L1=Serial.parseFloat();
  L1=6.5;
 Serial.println("Enter the length of second arm ");
  while(Serial.available()==0){}
  //L2=Serial.parseFloat();
  L2=8;
}

void loop() {
inverseKinematics();
}
void  inverseKinematics(){

    Serial.println("Enter the value x ");
        while(Serial.available()==0){}
        //x=Serial.parseFloat();

        Serial.println("Enter the value y ");
        while(Serial.available()==0){}
        //y=Serial.parseFloat();

        Serial.println("Enter the value z ");
        while(Serial.available()==0){}
        //z=Serial.parseFloat();

        rad_angle2 = acos((sq(z)+ sq(y) - sq(L1) - sq(L2)) / (2*L1*L2));
        rad_angle3 = acos((sq(L1) + sq(L2) - sq(y)- sq(z)) / (2*L1*L2));
        rad_angle1= (atan2(y , x) - atan2(L2*sin(rad_angle2),
L1*sin(rad_angle3)))/(L1*cos(rad_angle3)+L2*cos(rad_angle2));
       delay(1000);
```

```
      angle1= (rad_angle1*180)/pi;
      angle2= (rad_angle2*180)/pi;
      angle3= (rad_angle3*180)/pi;

  Serial.print("x is ");
   Serial.println(x);
  Serial.print("y is ");
   Serial.println(y);
  Serial.print("z is ");
   Serial.println(z);
  Serial.print("angle1 is  ");
   Serial.println(angle1);
  Serial.print("angle2 is ");
   Serial.println(angle2);
  Serial.print("angle3 is ");
   Serial.println(angle3);

   if(preangle1>angle1){
         angle1++;
          pwm0 = map(angle1, 0, 180, SERVOMIN, SERVOMAX);
          pca9685.setPWM(SER0, 0, pwm0);
          preangle1=angle1;
  }

    else if(preangle1<angle1){
       angle1--;
        pwm0 = map(angle1, 0, 180, SERVOMIN, SERVOMAX);
        pca9685.setPWM(SER0, 0, pwm0);
        preangle1=angle1;

  }

    if(preangle2>angle2){
     angle2++;
     pwm1 = map(angle2, 0, 180, SERVOMIN, SERVOMAX);
     pca9685.setPWM(SER1, 0, pwm1);
     preangle2=angle2;
   }

    else if(preangle2<angle2){
     angle2--;
     pwm1 = map(angle2, 0, 180, SERVOMIN, SERVOMAX);
     pca9685.setPWM(SER1, 0, pwm1);
     preangle2=angle2;
   }

    if(preangle3>angle3){
     angle3++;
```

```
    pwm2 = map(angle3, 0, 180, SERVOMIN, SERVOMAX);
    pca9685.setPWM(SER2, 0, pwm2);
    preangle3=angle3;

  }

   else if(preangle3<angle3){
    angle3--;
    pwm2 = map(angle3, 0, 180, SERVOMIN, SERVOMAX);
    pca9685.setPWM(SER2, 0, pwm2);
    preangle3=angle3;
  }
  x=x+0.1;
  y=y+0.1;
  z=z+0.1;
 }
```

We do not plan on changing the code to accommodate stepper motors, rather than servo motors.
Our next and final steps will be to do away with looped inputs, and integrating the use of our reworked GUI to introduce manual inputs to control the arm.

Graphical User Interface

We are unable to showcase any tangible GUI currently, as we took a last minute decision to change our plans and create a PC based application rather than a bluetooth application, solely because of the fact that we could not make the mobile GUI work (connect and then perform functions) on the robotic arm. We are currently working on developing our Python-based GUI, (or C language) and all our focus is based on this module as we have finished the IK solution and most of our 3D printing.

Note: We expect to be ready with a minimum viable GUI by Design Day.

Final Sander Prototype

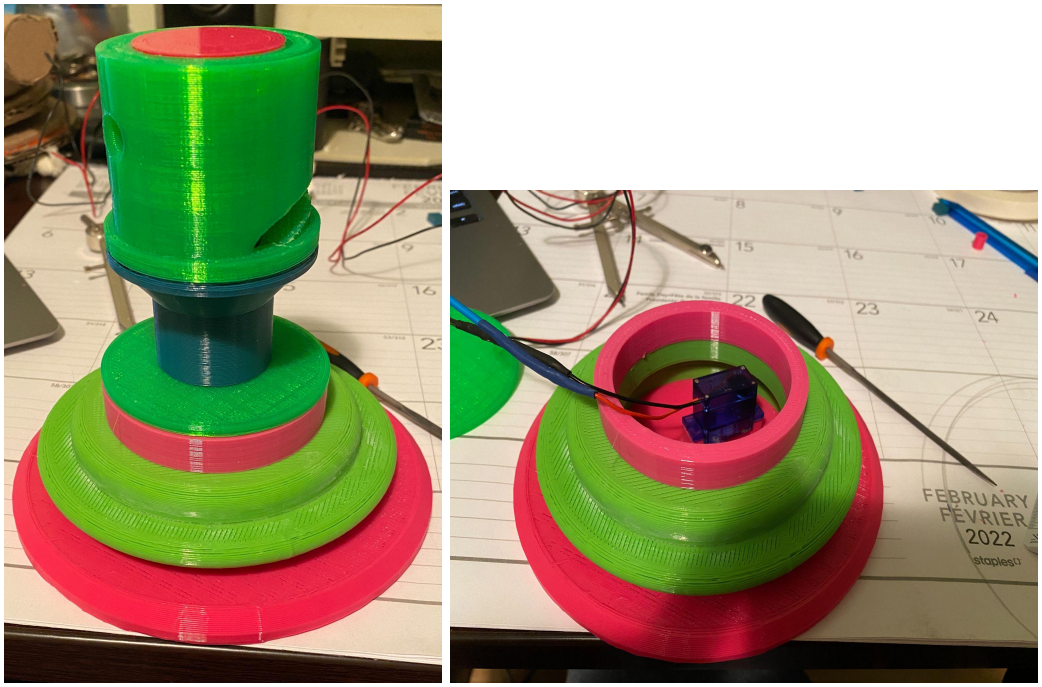

Figure: Final prototype of Sander

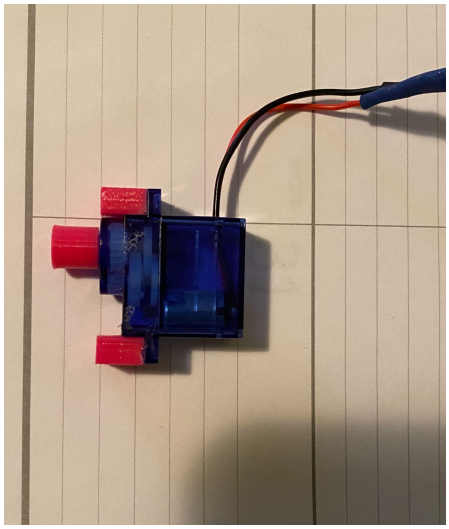

Figure: Motor; spindle and hooks attached

The sander can perform its intended task with a good degree of quality as the small motor can run at a higher RPM than expected. With the hardware components we have right now it would take a long time to sand rust from a steel hull. If budget allowed and we could purchase a higher quality motor then the sander would be able to properly complete its job. However, as a proof of concept, the prototype is perfectly adequate.

Throughout the design process, multiple prototypes were built but settled on our final one due to its decreased size, therefore, decreasing its weight and reducing the strain placed on the arm. Our original CAD design had a vacuum port attached to the sander. This is something we would have liked to include but it had to be cut from the design as the sander was too heavy with it attached. The area can still be cleaned afterward with a sailor and a vacuum, however, this will take more time.

The sander also has versatility; other pads can be attached to the sander such as buffing pads to polishing surfaces.

We were able to test that the motor could run while attached to the sanding disk, however, due to lack of time and budget we could not test its ability to properly sand rust or other contaminants from a surface.

Paint Gun 3D print prototype

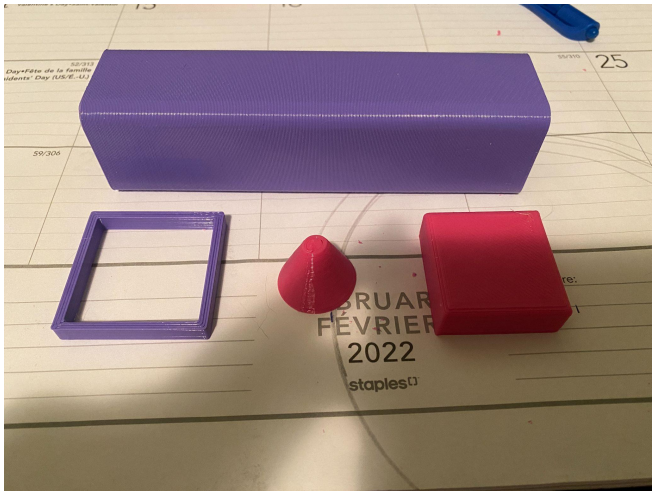

Figure: Nozzle, Stem and other proprietary parts

These are the parts of the 3D printed paint gun that are completed to date. Due to time constraints, we were not able to complete printing all the parts but the full model should be constructed by design day.

Safety

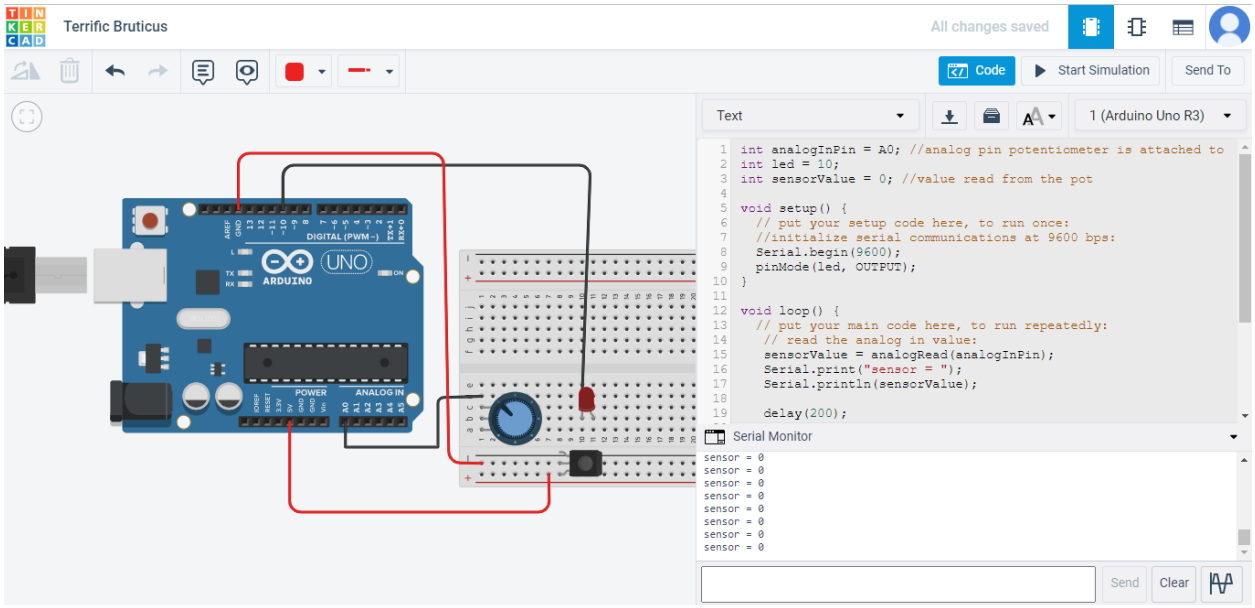

Figure: TinkerCAD demonstration

Obstruction Detection Software

```
int analogInPin = A0; //analog pin potentiometer is attached to
int led = 10;
int sensorValue = 0; //value read from the pot

void setup() {
  //initialize serial communications at 9600 bps:
  Serial.begin(9600);
  pinMode(led, OUTPUT);
}
void loop() {
   // read the analog in value:
   sensorValue = analogRead(analogInPin);
   Serial.print("sensor = ");
   Serial.println(sensorValue);
   delay(200);
   if(sensorValue<80) {
      digitalWrite(led,HIGH);
    }
    else {
    digitalWrite(led,LOW);
    }
}
```

Physical obstruction testing using the IR Sensor is yet to be conducted at this time because of primary focus on the IK solution and the Graphical User Interface (GUI).

However, we are confident in its effectiveness because it is a tried and tested method of obstruction detection. We have also done an analytical test using TinkerCAD to ensure the viability of the software.

If time permits, we plan on conducting a mini-test for the obstruction detection software and showcasing it on design day.

Ultrasound sensor

We also have a plan of installing an ultrasound sensor to our end effector because of the inability of the IR sensor to detect distance and depth. If time and budget constraints permit, then we will do so.

Challenges

Final Inverse Kinematics Solver

- Unexpected failure of the program recognizing the 3rd degree of freedom
- Program giving negative outputs i.e angle 1= -89 degrees
- Program making the robot repeatedly smack into the table
- Integrating the GUI to work with the Arduino IK solution
- Allowing the robot to stop in time

Paint Gun Prototype

- Limited time to print the paint gun as much of the time was spent on the sander
- Limited budget to buy adhesive

Graphical User Interface

- Understanding the Python language
- Finding a reliable Python IDE
- To connect the Python coded GUI to our Arduino IK solution
- Time constraints

Safety

- Inability to conduct an obstruction detection test using the Infrared Ray sensor and our software due to limited time.
- The unreliability of the IR sensor working alone
- Requiring a second sensor i.e Ultrasound sensor to complete make sure the robot is safe

Feedback

The feedback of our prototypes was mostly positive, however the failure of the GUI working as expected caused us to explore our contingency plan of developing a new GUI. We do fear that this GUI might not be developed properly in time, and may cause shortcomings on design day. For safety considerations, after the feedback from our class, we realized that the IR sensor needs to be coupled with an extra sensor to make effective safety decisions.

Conclusion

At this time, we have doubts about our design day readiness, however, we are going to try our best to work up a solution in time, as this was the exact reason why we had a contingency in place. Our hardware i.e our paint gun and sander, will be completed in time along with our safety software and hardware being implemented correctly. Regardless, the IK solution being completed allows us to devote our time to other requirements of the project.

Wrike updates: Wrike Snapshot