GNG 1103

Design Project User and Product Manual

Lights, Camera, Action

Submitted by:

Team Chem Eng., C1

Ty Kaloczi, 300213464

Sam Batu, 300170093

Colby Driscoll, 300175999

Kris Gandhi, 300071884

April 9th, 2022

University of Ottawa

Table of Contents

[Table of Contents ii](#_Toc100421763)

[List of Figures iv](#_Toc100421764)

[List of Tables v](#_Toc100421765)

[List of Acronyms and Glossary vi](#_Toc100421766)

[1 Introduction 1](#_Toc100421767)

[3.1 Setup Considerations 4](#_Toc100421768)

[Setup and Draw Function 9](#_Toc100421769)

[Screens 10](#_Toc100421770)

[Mouse Interaction 10](#_Toc100421771)

[Keyboard input usage 13](#_Toc100421772)

[1 4.3.1 Camera 15](#_Toc100421773)

[1 4.3.2 LED Lights 16](#_Toc100421774)

[1 4.3.3 Depth Sensor 19](#_Toc100421775)

[1 4.3.4 Compiled Code for Entire System 21](#_Toc100421776)

[6.1.1 BOM (Bill of Materials) 29](#_Toc100421777)

[6.1.2 Equipment list 29](#_Toc100421778)

[6.1.3 Instructions 29](#_Toc100421779)

[6.2.1 BOM (Bill of Materials) 30](#_Toc100421780)

[6.2.2 Equipment list 31](#_Toc100421781)

[BOM (Bill of Materials) 33](#_Toc100421782)

[Equipment list 34](#_Toc100421783)

[9. APPENDICES 38](#_Toc100421784)

[1 9.1 APPENDIX I: Design Files 38](#_Toc100421785)

List of Figures

*Figure 1. Final Prototype………………………………………………………………………………….2*

*Figure 2. Arduino IDE icon………………………………………………………………………………..4*

*Figure 3. Assembled Arm and End-Effector………………………………………………………….….5*

*Figure 4. Sample setup function…………………………………………………………………….…….9*

*Figure 5. Sample code for the MousePressed function showing button checks and screen flips..11*

*Figure 6. Sample code for the MouseDragged function that shows the coordinates being…...…11*

*Figure 7. Sample code for button check functions…………………………………………………….12*

*Figure 8. Sample Code for the function keyTyped()…………………………………………………..13*

*Figure 9. Depth Sensor & Light wiring setup………………………………………………………….14*

*Figure 10. Camera wiring schematic created by Indrek Luuk………………………………………15*

*Figure 11. The creation of variables……………………………………………………………………28*

*Figure 12. The Inverse Kinematics function that completes the calculation………………………28*

*Figure 13. The code block that checks angle value and sends it to the motors by looping through steps…………………………………………………………………………………………………………29*

*Figure 14. Sample variables to be created…………………………………………………………….30*

*Figure 15. End-Effector Casing…………………………………………………………………………31*

*Figure 16. End-Effector Casing Lid…………………………………………………………………….32*

List of Tables

[Table 1. Acronyms vii](#_Toc63505509)

[Table 2. Glossary vii](#_Toc63505510)

[Table 3. Referenced Documents 10](#_Toc63505511)

List of Acronyms and Glossary

Table 1. Acronyms

|  |  |
| --- | --- |
| **Acronym** | **Definition** |
| UI | User interface |
| IK | Inverse Kinematics |
| DOF | degree of freedom |
|  |  |
|  |  |

# Introduction

Group C1’s mission was to provide a creative inverse kinematics program to safely pilot a 3 DOF robotic arm that works cohesively with multiple end-effectors. We designed an end effector which can effectively carry out the scanning of surfaces that are to be painted, all of which being operated by a single low-skilled crewmember through a simple user interface. After reading this manual you will be well equipped to use our product the reader does not require technological training.

1. **Overview**

The client is a senior project engineer for the Naval Material Technology Management of the Department of National Defence of Canada. His objective is to develop a robot arm that will be able to perform 3 main functions of inspecting, clean and painting on surfaces on Halifax class warship. These functions will be used to carryout painting logos on the hull these ships. The purpose for investing time and money into creating a robot to do those tasks is to free up sailors to tend to other duties. A prototype is shown below which is made of 3D printed parts and uses an Arduino uno to operate.

The primary focus of our side of the projects was to come up with an inverse kinematics solution which allows a scaled down 3 DOF arm to achieve a specific position and orientation in space to carry out its intended functions. This will serve as a proof of concept, in addition we were tasked with creating 1 or more end-efforts, and simple user interface to interact with the robot arm. Our product is unique as we made/coded the UI from completely from scratch, not using MIT app developer etc, this allowed for a lot of flexibility despite it being a lot more difficult. The final product was worth time spent, as our arm can now operate in an autonomous state once a task is set by the operator using the GUI. The end-effector that we made is called the Light Camera end-effect that Scans working area and compute the path and approach needed, used for the inspection, it also has safety mechanism which uses a depth sensor and lights. Along with the topic of safety, our GUI also has a safety feature, we added stop button which completely halts the arm. In this project we used Arduino uno as the main communicator/ link. When coming up with this design, we focused on ensuring it was able in maneuver complicated spaces, that the end effectors could be easily removed and attached, easily piloted by one operator, and that it was safe to be around. ​Below is the final prototype.

A picture containing indoor

Description automatically generated

*Figure 1. Final Prototype*

* 1. **Cautions & Warnings**

Collision with the arm would be detrimental to the safety of anyone who is hit by the arm, and detrimental to the health of the arm itself. Waiver use and copy permissions must be obtained by Team Chem Eng for use of this goldmine product.

1. **Getting started**

# Setup Considerations

The first thing required for any user of this product is to download and install the required version of Arduino IDE based on their operating system from the following link, https://www.arduino.cc/en/software. Once the .zip file is opened, an ‘Arduino’ icon should appear on your desktop.

**A picture containing text, sign

Description automatically generated** *Figure 2. Arduino IDE icon*

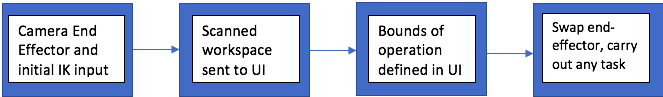
Once the Arduino IDE is installed, the code in 4.3.4 can be uploaded into the software.

1. **Configuration Considerations**

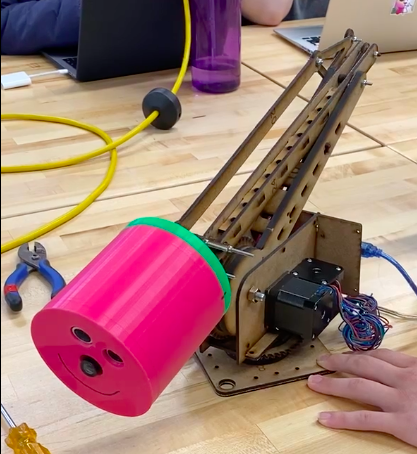
You will need the following not included tools for operation:

* Arduino board adaptor cable
* A computer or laptop with a USB drive
* 2.5mm Alan Key

The scanning system operates by initially attaching the camera end effector to the arm. Coordinates can be input to move the arm as required and take pictures using the camera in conjunction with the depth sensor to scan the room into a coordinate system from which a workspace can be selected with the UI. At this point the end effector can be swapped to whatever personalized end effector the customer has created to carry out whatever task they would like in the defined workspace.



The assembled equipment:



*Figure 3. Assembled Arm and End-Effector*

The equipment includes the constructed arm, and the end-effector containing lights, depth sensor, and camera. The stepper-motors in the arm communicate with the system through the Arduino Uno controller. The Camera and Depth sensor also communicate with the user interface through a separate arduino uno controller contained inside the end effector. The UI can then communicate back to the IK through the Arduino Uno.

1. **User Access Considerations**

Any user that will be controlling the arm through the UI will do it the same way. Once the IK code is uploaded to the arm then the user will be able to search for the picture taken by the camera end effector. Due to constraints of the UI interface, if a photo that cannot be found is searched for, the UI will crash. Once the picture is found and opened then the user can highlight a working area. When the working area is highlighted, the code will turn the pixel coordinates into real life coordinates to be input into the Ik code for the arm to go through all the necessary movements to carry out any tasks the user wishes the arm to complete.

1. **Accessing/setting-up the System**

The IK code needs to be uploaded to the arm. Once the UI has been opened everything is then operational.

* 1. **System Organization & Navigation**

The UI consists of a search bar (that will first be on the screen when the UI is opened) where the user can search for their photo of a desired area, as well as a drop down menu that includes a change image button and the emergency stop button. There is also a clear button to clear your highlighted area.

1. **Exiting the System**

Once the Arm has reached the necessary coordinates the motors will hold in place until the arm is turned off and guided back to the resting position. Then the UI and other codes can be exited out of safely.

1. **Using the System**
   1. **Inverse Kinematics**
      1. **Angle Calculating Function**

The function works by taking the three global variables for the angles that are created at the very top of the file and changing their values to the angles calculated. The method must be written as in figure 12. If the void is forgotten it will look to return a value which we are not. The method can be given any arbitrary name if the input of the function is (int x, int y, int z). These inputs are the desired coordinates for the arm to move to.

1. **Motor Control Code**

The block of code goes within the setup function. This will ensure that the code will only run until it has reached the designated coordinates before shutting itself off. Each pin that is in use must be initiated as an output pin using the command

pinMode(pinName, OUTPUT)

Once the pins are instantiated, they can be utilized. The direction of every motor will be set clockwise with the function

digitalWrite(Pin, HIGH)

The Angle Calculating function will be called

functionName(x, y, z),

this will give a value to the three angle variables a, b and c. The angles will then be checked as to whether they are positive or negative, if the value is negative then the direction of the motor will be set to counterclockwise with

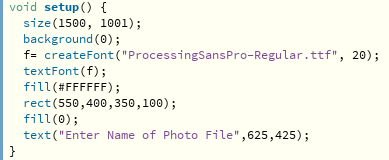
digitalWrite(Pin, LOW)

To move the motors, they will be looped through setting the motor to HIGH and then after a delay to LOW. The loop will be a For loop that will run from 0 to the angle/1.8. The angle needs to be divided by 1.8 because the motor has 200 steps per revolution which works out to 1.8 degrees per step.

* 1. **User Interface**

### Setup and Draw Function

The setup and draw functions are the main functions within the processing software. All code must be put into these functions or another function. The setup function works the same as the setup function of Arduino, it will run once and the draw function works the same as the loop function of Arduino, it will loop until the code is terminated. Within the setup function the variables needed for text will be created as well as the size of the display and the initial search display will be created.



*Figure 4. Sample setup function*

### Screens

The UI works by taking input from the user and switching through prebuilt screens this is done by checking for mouse input in a certain place and changing the screen if needed. This works by utilizing the block of code

switch(screen)

case “”;

break;

These lines will first instantiate screen as the variable that switches the screens when the value is equal to the case value. To create the screen the information will be put between the case: and break; command. All the code must be put in the draw function so that it is kept on the display until the switch is called and the display is overwritten.

### Mouse Interaction

The mouse interaction is tracked using two custom functions

MousePressed() and

MouseDragged()

The function MousePressed is used to check when the mouse is clicked, and it saves the place where the mouse was clicked. This tracking is used to find whether the user has clicked on a “button” so that the button can be utilized. The sample code for the MousePressed function can be seen in Figure 5. The function MouseDragged is used to find the coordinates where the mouse is released, and these are used with the coordinates from the MousePressed function to draw a rectangle on the display which then becomes the highlighted work area. Sample code for the MouseDragged function can be found in Figure 6. To check if the mouse coordinates were on a button the custom codes for buttons were created based on the screen that the display was on. The button checks depend on the screen because some buttons are only available on certain screens. This can be seen in figure 7.

Text

Description automatically generated

*Figure 5. Sample code for the MousePressed function showing button checks and screen flips*

A picture containing diagram

Description automatically generated

*Figure 6. Sample code for the MouseDragged function that shows the coordinates being saved*

Graphical user interface, text, application, email

Description automatically generated

*Figure 7. Sample code for button check functions*

### Keyboard input usage

For the keyboard to be used another custom function needed to be created called

keyTyped()

This function checked every time a key on the keyboard was pressed and checked what the key was if it was a standard key, it would be drawn onto the search bar and added to a String so that it could be searched for within the folder when the enter key was pressed. If the key pressed was a backspace the code would remove the last character from the String and would redraw the string onto the search bar. The sample code can be found in figure 8.

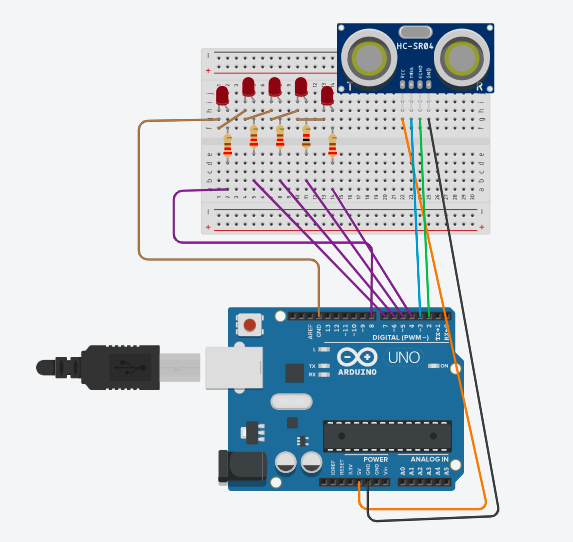
Text

Description automatically generated

*Figure 8. Sample Code for the function keyTyped()*

* 1. **End-Effector**

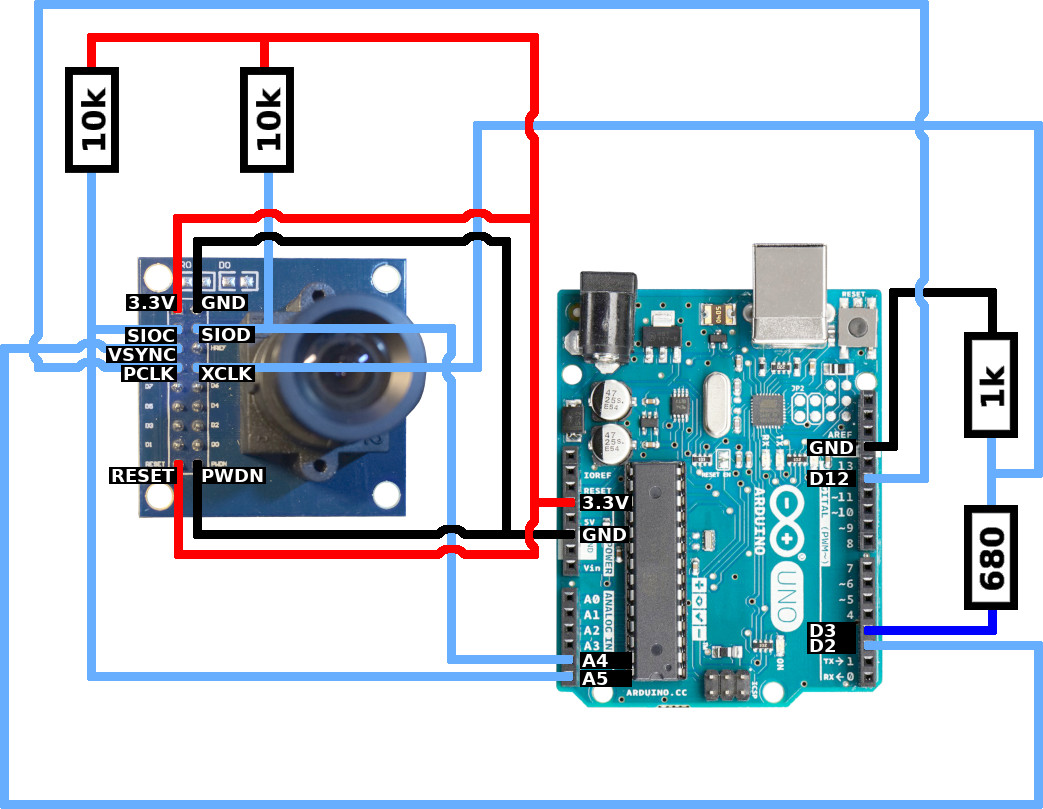
The light and camera end-effector attaches to the robot arm with the purpose of scanning and capturing photos of the surfaces on the ships that must sandblasted and painted. The end-effector has three subsystems, the camera itself, the LED lights used to illuminate the surfaces for the camera, and the depth sensor which is intended to signal when the end-effector gets too close to another object or surface. Detailed descriptions of these three subsystems can be found below. The following TinkerCAD schematic shows how to wire the Depth Sensor and lights:



*Figure 9. Depth Sensor & Light wiring setup*

# 4.3.1 Camera

The camera is attached to the face of the end effector and takes images that can be sent through the Arduino to the UI. Due to our lack of funds from the university we were unable to acquire a bluetooth module to enable this interaction between camera and UI. The following schematic created by Indrek Luuk shows how to wire the Arduino to the camera. He also has open source code available on GITHub to operate the camera which can be accessed here: <https://www.github.com/indrekluuk/LiveOV7670>



*Figure 10. Camera wiring schematic created by Indrek Luuk*

# 4.3.2 LED Lights

The LED lights are controlled by the Arduino and Triggered by readings on the depth sensor to alert anyone that is in the arm’s path of motion. The following code must be uploaded through the Arduino IDE for the lights to operate as required.

Define the LED’s to the corresponding pin:

int LED1 =8;

int LED2 =7;

int LED3 =6;

int LED4 =5;

int LED5 =4;

In the setup the following code must be present to set all the LED pins to outputs:

pinMode(8,OUTPUT);

pinMode(7,OUTPUT);

pinMode(6,OUTPUT);

pinMode(5,OUTPUT);

pinMode(4,OUTPUT);

In the Loop the following code must be implemented to control when they are active/inactive based on distance readings from the sensor.

{if (distance <15&& distance >10)

digitalWrite(LED1, HIGH);

if (distance <10&& distance >8)

digitalWrite(LED1, HIGH);

digitalWrite(LED2,HIGH);

if (distance <8&& distance >5)

digitalWrite(LED1,HIGH);

digitalWrite(LED2,HIGH);

digitalWrite(LED3,HIGH);

if (distance <5&& distance >4)

digitalWrite(LED1,HIGH);

digitalWrite(LED2,HIGH);

digitalWrite(LED3,HIGH);

digitalWrite(LED4,HIGH);

if (distance <4)

Serial.println("RESTRICT FORWARD MOTION"); //Send Signal to IK to stop moving forward HERE if Pankaj would buy us a bluetooth module.

digitalWrite(LED1,HIGH);

digitalWrite(LED2,LOW);

digitalWrite(LED3,HIGH);

digitalWrite(LED4,LOW);

digitalWrite(LED5,HIGH);

delayMicroseconds(250);

digitalWrite(LED1,LOW);

digitalWrite(LED2,HIGH);

digitalWrite(LED3,LOW);

digitalWrite(LED4,HIGH);

digitalWrite(LED5,LOW);

# 4.3.3 Depth Sensor

The Depth Sensor operates by sending ultrasonic waves through one speaker and has a receiver to measure how long it takes for the ultrasonic wave to reflect back to the depth sensor.

The following must be defined:

int TRIG =3;

int ECHO =2;

float duration;

float distance;

This defines the Trigger to its corresponding pin and the echo sensor to its corresponding pin on the Arduino board. Two variables must also be created, duration, which is the amount of time the echo sensor reads and distance which is calculated from the duration.

In the setup the following code must be present for operation of the depth sensor:

pinMode(TRIG,OUTPUT);

pinMode(ECHO,INPUT);

This defines the Trig Pin on the Arduino as an output and the Echo pin on the Arduino as an input.

In the loop this code must be implemented for the depth sensor to operate:

digitalWrite(TRIG, LOW);

delayMicroseconds(200);

digitalWrite(TRIG, HIGH);

delayMicroseconds(1000);

digitalWrite(TRIG, LOW);

duration = pulseIn(ECHO, HIGH);

//time --> distance

distance = duration\*0.034/2;

This sends the ultrasonic wave and measures the amount of time it will take to return and uses this time along with the speed of sound through standard air to calculate the distance in cm. If this were to be used in a different medium than standard air, the respective speed of sound can be swapped in place of 0.034 in the step, distance = duration\*0.034/2.

# 4.3.4 Compiled Code for Entire System

int LED1 =8;

int LED2 =7;

int LED3 =6;

int LED4 =5;

int LED5 =4;

int TRIG =3;

int ECHO =2;

float duration;

float distance;

void setup() {

pinMode(8,OUTPUT);

pinMode(7,OUTPUT);

pinMode(6,OUTPUT);

pinMode(5,OUTPUT);

pinMode(4,OUTPUT);

pinMode(TRIG,OUTPUT);

pinMode(ECHO,INPUT);

Serial.begin(9600);

Serial.println("Ultrasonic Sensor HC-SR04 With Arduino Uno");

}

void loop() {

//make some noise

digitalWrite(TRIG, LOW);

delayMicroseconds(200);

digitalWrite(TRIG, HIGH);

delayMicroseconds(1000);

digitalWrite(TRIG, LOW);

duration = pulseIn(ECHO, HIGH);

distance = duration\*0.034/2;

if(distance < 10)

{

Serial.println("Distance: ");

Serial.print(distance);

Serial.println("cm");

}

{

digitalWrite(TRIG, LOW);

delayMicroseconds(2);

digitalWrite(TRIG, HIGH);

delayMicroseconds(10);

digitalWrite(TRIG, LOW);

duration = pulseIn(ECHO, HIGH);

distance = duration\*0.034/2;

if(distance < 10)

{

Serial.println("Distance: ");

Serial.print(distance);

Serial.println("cm");

}

{if (distance <15&& distance >10)

digitalWrite(LED1, HIGH);

if (distance <10&& distance >8)

digitalWrite(LED1, HIGH);

digitalWrite(LED2,HIGH);

if (distance <8&& distance >5)

digitalWrite(LED1,HIGH);

digitalWrite(LED2,HIGH);

digitalWrite(LED3,HIGH);

if (distance <5&& distance >4)

digitalWrite(LED1,HIGH);

digitalWrite(LED2,HIGH);

digitalWrite(LED3,HIGH);

digitalWrite(LED4,HIGH);

if (distance <4)

Serial.println("RESTRICT FORWARD MOTION");

digitalWrite(LED1,HIGH);

digitalWrite(LED2,LOW);

digitalWrite(LED3,HIGH);

digitalWrite(LED4,LOW);

digitalWrite(LED5,HIGH);

delayMicroseconds(250);

digitalWrite(LED1,LOW);

digitalWrite(LED2,HIGH);

digitalWrite(LED3,LOW);

digitalWrite(LED4,HIGH);

digitalWrite(LED5,LOW);

}

}

}

1. **Troubleshooting & Support**

If you run into a problem, start by back tracking and re-tracking your step, the probable cause to most issues is that the wires are connected incorrectly. In the code, make sure the right pins are assigned.

* 1. **Error Messages or Behaviors**

There are no likely errors associated with the end-effector. Errors associated with the IK include inputs outside of the range of motion of the arm, it will be impossible to reach and result in the IK not moving the arm. UI errors could arise if the file searched for is not present on the computer, thus it is important to input exactly the name of the picture file taken by the camera to avoid the UI crashing. Corrective measures for these failures are making sure the inputs are within reach of the arm for IK, and ensuring the correct file is searched for in the UI.

* 1. **Special Considerations**

With respect to the end effector, if it is being used in a medium that is not air under standard conditions, the depth sensor accuracy could be optimized by adjusting the speed of sound to the speed of sound in the medium the end effector will be used in.

1. **Maintenance**

Our prototype is very low maintenance. Once it is assembled it is unlikely to fail. Small maintenance tweaks may be required with the code depending on when this product is purchased and changes to the Arduino IDE that may have been made since April 8th 2022. Joints on the arm may need to be replaced as wear and tear occurs, however, no maintenance has been required on the prototype thus far.

1. **Support**

The delegation of assistance and support will be left up to the customers of our product. Should technical support be required, this document should be thoroughly referenced, then and only then may contact be made to Colby Driscoll, Kris Gandhi, Ty Kaloczi, or Sam Batu for assistance however we would not like to be reached with questions regarding this project. Identified problems with the system should not be reported, if you are unsatisfied with the product full reimbursement will be made by our mentor Pankaj Kumar Rathi. Security incident handling is up to the customer of this product, however it is important to note that ZERO liability is in the hands of Team Chem Eng with respect to anything regarding this product once you have purchased it.

1. **Product Documentation**
   1. **Inverse Kinematics**

## BOM (Bill of Materials)

|  |  |  |
| --- | --- | --- |
| Software | Cost ($) | Link |
| Arduino IDE | 0 | <https://www.arduino.cc/en/software> |

### Equipment list

* Computer with Arduino IDE installed
* USB Connector

### Instructions

1. Create and instantiate all of the instance variables needed for operation of Inverse Kinematics and the arm. Figure 11 can be used as a reference.

Text, letter

Description automatically generated

*Figure 11. The creation of variables*

2. Create a function for the calculation of motor angles. Figure 12 can be used as a reference.

Text

Description automatically generated with low confidence

*Figure 12. The Inverse Kinematics function that completes the calculation*

3. Code within the setup() function so that the arm moves only once to ensure it only goes to the directed coordinates and not an endless loop. Figure 13 can be used as a reference.

Text

Description automatically generated with medium confidence

*Figure 13. The code block that checks angle value and sends it to the motors by looping through steps*

* 1. **User Interface**

## BOM (Bill of Materials)

|  |  |  |
| --- | --- | --- |
| Software | Cost ($) | Link |
| Processing IDE | 0 | <https://processing.org/download> |

### Equipment list

* Computer with Processing IDE installed

1. **Instructions**

1. The necessary variables must be instantiated as in figure 14

Text

Description automatically generated

*Figure 14. Sample variables to be created*

2. The setup function must be created and filled as in figure 4

3. The draw function must be created, and the screen functions must be built into it

4.All the custom functions must then be created from the sample codes provided in figures 5, 6, 7, 8

* 1. **End-Effector**

The code contained in section 4.3.4 must be uploaded and compiled in the Arduino IDE.

The camera, lights, and depth sensor must be wired by following the schematics provided in sections 4.3.1 and sections 6.11. The files used for the 3D printing are available with purchase of this product, and CAD sketches of the 3D printed parts are below.

A picture containing electronics

Description automatically generated  
*Figure 15. End-Effector Casing Face*

A picture containing icon

Description automatically generated  
*Figure 16. End-Effector Casing Lid*

Once everything is wired, the depth sensor and camera can be inserted into their respective holes, and the lights can be mounted wherever the user likes so long as they are wired as shown in 6.11. At this point the end effector can be sealed using screws and mounted to the arm.

### BOM (Bill of Materials)

|  |  |
| --- | --- |
| Material | Cost ($) |
| Plastics Housing | 0 |
| Ultrasonic sensor | 4 |
| Motor shield | 15 |
| Arduino Cam | 17.81 |
| LED lights | 0 |
| Wires | 0 |
| Resistors | 0 |
| Arduino | 0 |
| Breadboard | 0 |
| Screws | 0 |
| Total cost: | 36.81 |

### Equipment list

* Ultimaker 3D printer
* Ultimaker Cura
* Onshape CAD software
* Tinkercad
* Arduino IDE
* Drill Press
  1. **Testing & Validation**

Testing that was done included tests of the IK motion which were successful. Many trials were run to ensure that the robot arm would successfully move in all 3 degrees of freedom. Changes were made to the code during these tests and ultimately the arm was able to move in all 3 degrees of freedom after so many trials. Our data collected for this test operated on a “yes/no” basis where it was either working fully which resulted in “yes” or it was not working which resulted in “no”. After trials and modifications to the code, the results were “yes” with the code attached in this document.

Testing of the prototype included ensuring the depth sensor worked by placing it(while inside the end effector) 30cm away from a wall, measured with a ruler. As the end effector was moved towards and away from the wall it consistently output the correct distance from end effector to the wall with an uncertainty of +/- 0.6cm. The LED lights were also tested to make sure the wiring of them was correct, these tests were successfully marked by the LED’s illuminating.

Testing of the UI was done by taking untrained users who were unfamiliar with the product and allowing them to interact with the UI. These tests proved that the UI is functional and easy to maneuver.

1. **Conclusions and Recommendations for Future Work**

When wanting to make edits to the IK or the UI it is important to make another test file in case something changed or added will hinder the execution of the parts already working satisfactorily that do not need to be tinkered with. This slow implementation will also ensure that every part of the codes will work in tandem and will allow the programmer to notice and make optimizations quickly, easily and efficiently.

With more time the UI would have created a coordinate system using the highlighted work area. The coordinates would have been imported to the IK code which would then loop through them to ensure that the arm goes to every coordinate required. To make the transmission and process more streamlined a Bluetooth module could be implemented to not only send messages from UI to the Arm but from the end effector to the UI or Arm.

1. **Bibliography**

Github <https://www.github.com/indrekluuk/LiveOV7670>

<http://thor.angel-lm.com/>

<https://www.canada.ca/en/department-national-defence/services/procurement/halifax-class-modernization.html>

<https://www.google.com/search?q=how+to+use+arduino+camera+module+with+and+arduino+uno&rlz=1C1CHBF_enCA811CA811&oq=how+to+use+arduino+camera+module+with+and+ardu&aqs=chrome.2.69i57j33i160l3.14421j0j7&sourceid=chrome&ie=UTF-8#kpvalbx=_nTYzYsiGCsGztQbckaygCQ25>

9. **APPENDICES**

# 9.1 APPENDIX I: Design Files

Table 3. Referenced Documents

|  |  |  |
| --- | --- | --- |
| **Document Name** | **Document Location and/or URL** | **Issuance Date** |
| Deliverable B | Makerepo Repository | January 30 |
| Deliverable C. Group 1 | Makerepo Repository | February 6 |
| Deliverable D. Group 1 | Makerepo Repository | February 13 |
| Deliverable E. Group 1 | Makerepo Repository | February 20 |
| Deliverable F. Group 1 | Makerepo Repository | March 6 |
| Deliverable G. Group 1 | Makerepo Repository | March 13 |
| Deliverable H. Group 1 | Makerepo Repository | Mach 27 |
| Bill of Materials | Makerepo Repository | February 20 |
| Part Studio 2. Part 1 | Makerepo Repository | March 10 |
| Part Studio 2. Part 2 | Makerepo Repository | March 10 |

Makerepo link: <https://makerepo.com/ColbyDriscoll/1101.gng-1103-c1-group1-team-chem-eng>