

APPENDIX A-1

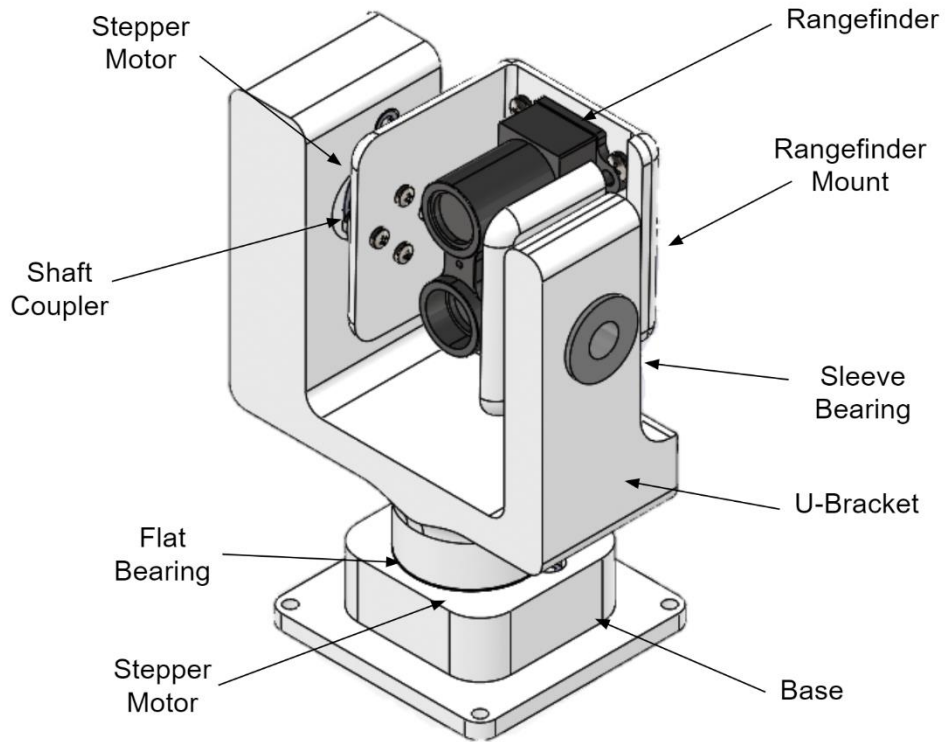


FIGURE A-1-1: ASSEMBLY OF LIDAR HOUSING SHOWING COMPLETE SYSTEM

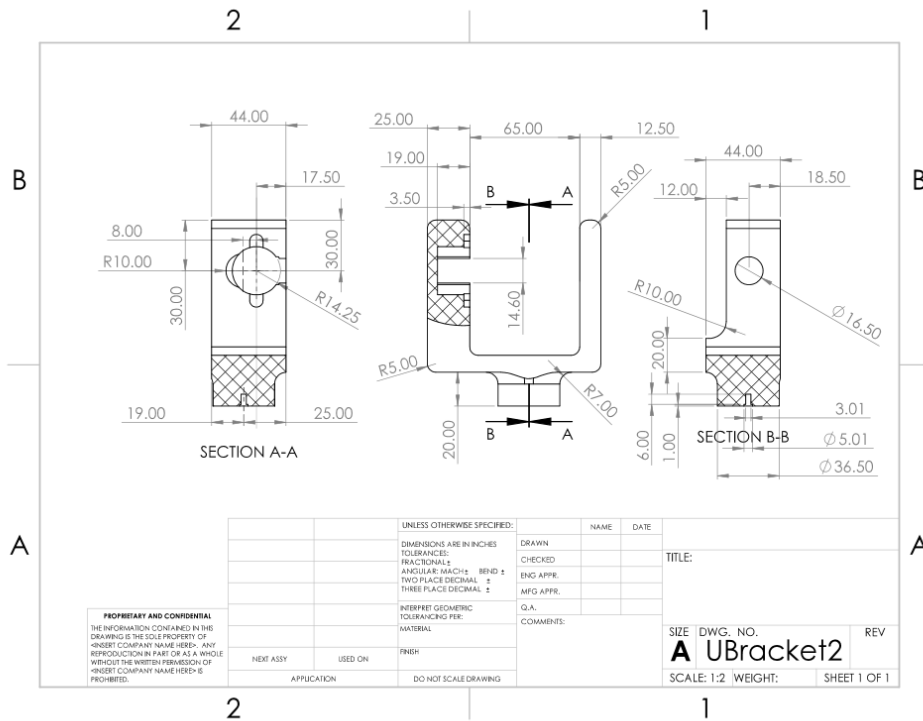


FIGURE A-1-2: U-BRACKET DRAWINGS (DIMENSIONS IN INCHES)

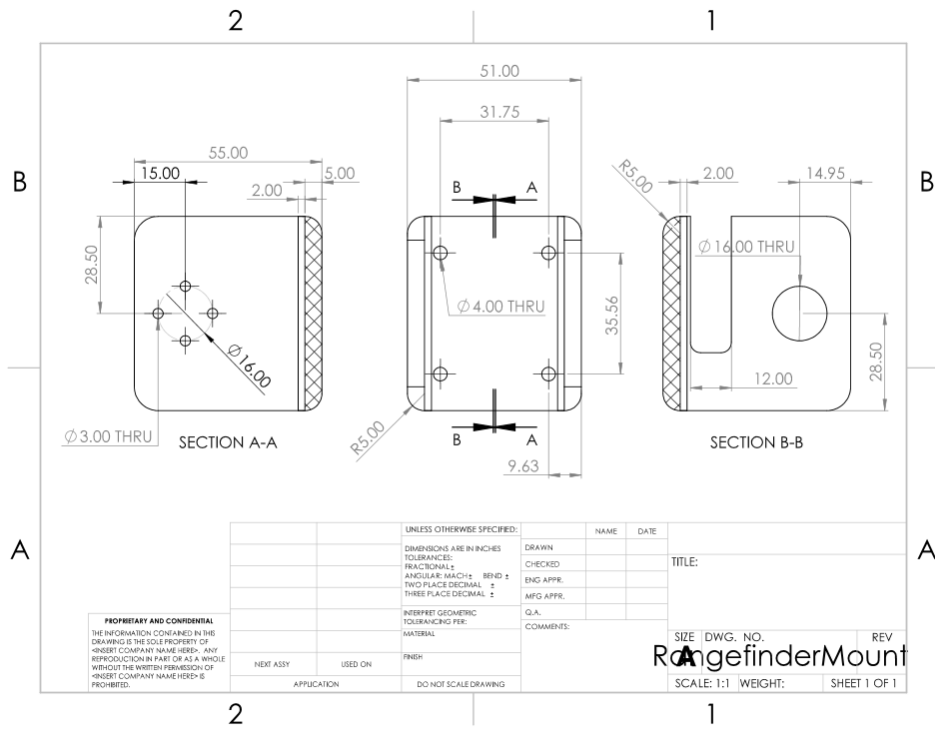


FIGURE A-1-3: RANGEFINDER MOUNT (DIMENSIONS IN INCHES)

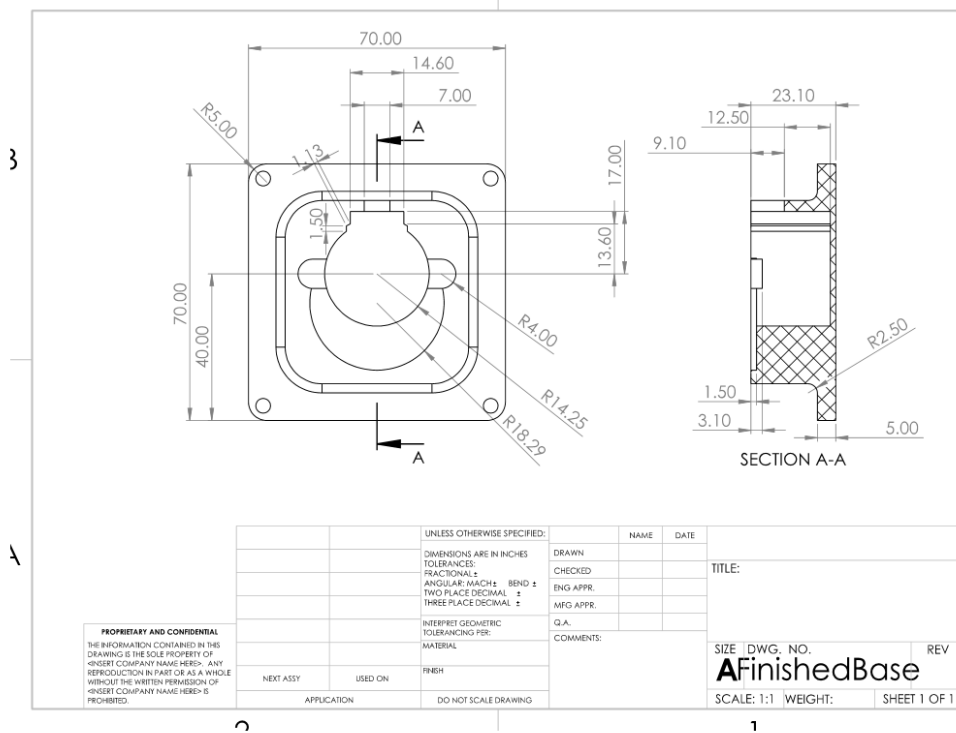


FIGURE A-1-4: BASE FOR LIDAR HOUSING (DIMENSIONS IN INCHES)

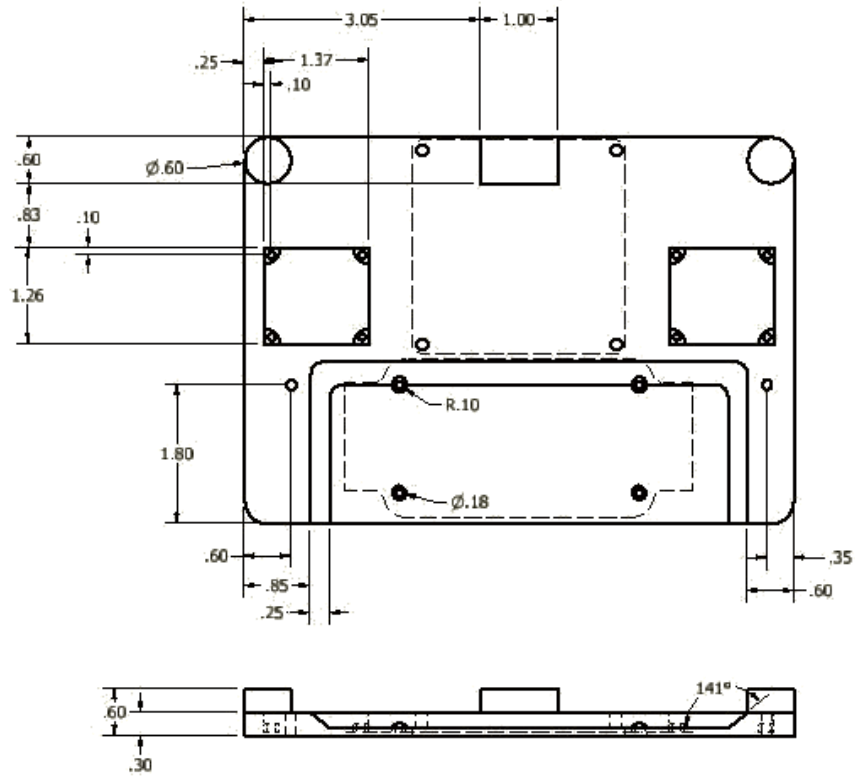
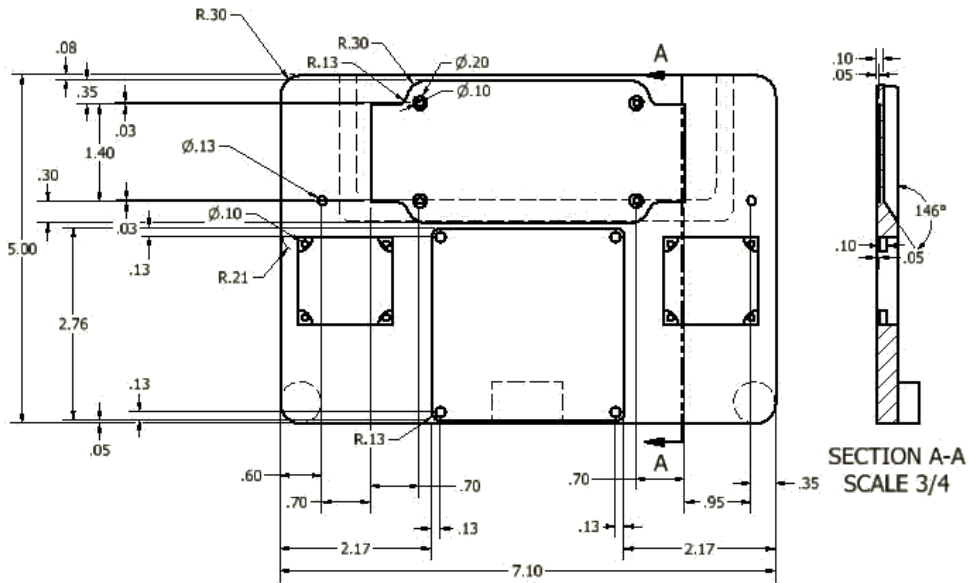


FIGURE A-1-5: MOUNTING PLATE DRAWING 1 OF 2 FOR SYSTEM COMPONENTS (DIMENSIONS IN INCHES)



All Dimensions are in inches (in)
 Unless otherwise stated all
 dimensions are symmetric

FIGURE A-1-6: MOUNTING PLATE DRAWING 2 OF 2 FOR SYSTEM COMPONENTS (DIMENSIONS IN INCHES)

APPENDIX A-2

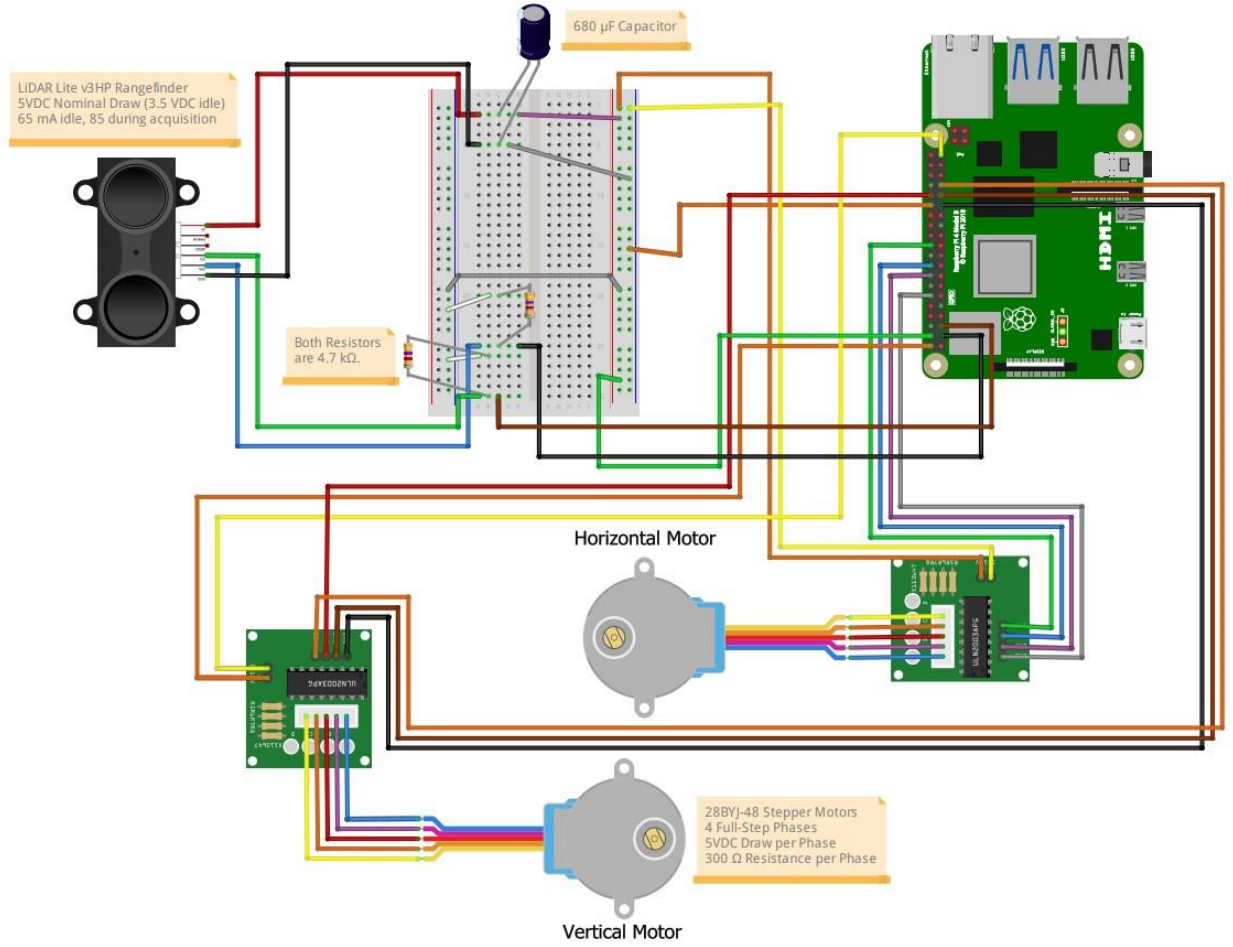


FIGURE A-2-1: WIRING SCHEMATIC FOR LIDAR SYSTEM

APPENDIX A-3: PYTHON CODE

```
1 # Jarod Bennett. University of Kansas Mechanical Engineering.
2 # Low accuracy code to quickly produce a point cloud in about 7 minutes by running two 28byj-48 stepper motors with a Garmin Lidar-Lite v3HP using a Raspberry 4.
3 # The final product is a .xyz file that can be read in Matlab to display a 3-D point cloud. The values are distance, horizontal angle, vertical angle.
4
5 # Creating a function that can be called in the Graphical User Interface to run the code
6
7 def runCode():
8     import timeit #importing the timeit python library to get a run time for the code
9     start = timeit.default_timer() #starting the timer
10
11     sample_data1= open("horizontal_angle.xyz", "w")
12     sample_data1.close()
13     sample_data2= open("vertical_angle.xyz", "w")
14     sample_data2.close()
15     sample_data3= open("distance.xyz", "w")
16     sample_data3.close()
17     sample_data4= open("output.xyz", "w")
18     sample_data4.close()
19     from lidar_lite import Lidar_Lite
20     lidar = Lidar_Lite()
21     import math
22     connected = lidar.connect(1)
23     import time
24     import RPi.GPIO as GPIO
25     GPIO.setwarnings(False)
26     GPIO.setmode(GPIO.BOARD) # Opening four .xyz files to write data into. If there is previous data in the .xyz files, the "w" clears them.
27     sample_data1= open("horizontal_angle.xyz", "w")
28     sample_data1.close()
29     sample_data2= open("vertical_angle.xyz", "w")
30     sample_data2.close()
31     sample_data3= open("distance.xyz", "w")
32     sample_data3.close()
33     sample_data4= open("output.xyz", "w")
34     sample_data4.close()
35
36     # Importing the lidar_lite python file that is provided by Garmin. Allows lidar.getdistance to record the distance later on.
37     from lidar_lite import Lidar_Lite
38     lidar = Lidar_Lite()
39     import math # Uses trigonometry from the Python math library to get the distance in the x axis.
40     connected = lidar.connect(1)
41     import time # Importing the time python library to be able to speed up/slow down the motor. Also able to create pauses.
42     import RPi.GPIO as GPIO #general-purpose input/output pins on the Raspberry Pi. Can now refer to it as just GPIO.
43     GPIO.setwarnings(False) #gets rid of
44     GPIO.setmode(GPIO.BOARD) # Set GPIO numbering mode from the pins.
45
46     # Rotates the motor clockwise in the halfstep configuration.
47     clockwise = [
48         [1,0,0,1],
49         [1,0,0,0],
50         [1,1,0,0],
51         [0,1,0,0],
52         [0,1,1,0],
53         [0,0,1,0],
54         [0,0,1,1],
55         [0,0,0,1]
56     ]
57
58     # Rotates the motor counterclockwise in the halfstep configuration.
59     counterclockwise = [
60         [0,0,0,1],
61         [0,0,1,1],
62         [0,0,1,0],
63         [0,1,1,0],
64         [0,1,0,0],
65         [1,1,0,0],
66         [1,0,0,0],
67         [1,0,0,1],
68     ]
69
70     # Moving the horizontal motor from its initial position to its starting position (15.5 degrees to the right).
71     control_pins = [12,16,18,22] #horizontal motor
72     for pin in control_pins:
73         GPIO.setup(pin, GPIO.OUT)
74         GPIO.output(pin, 0)
75
76     clockwise # Tells the motor which way to rotate (this variable was declared at the beginning of code).
77     for i in range(30): #this is for the degrees you want the motor to spin (512= 360 degrees)
78         for halfstep in range(8):
79             for pin in range(4):
80                 GPIO.output(control_pins[pin], clockwise[halfstep][pin])
81                 time.sleep(.005)
82
83     # Moving the vertical motor from its initial position to its starting position (21.1 degrees up).
84     control_pins = [29,31,32,33] #vertical motor
85     for pin in control_pins:
86         GPIO.setup(pin, GPIO.OUT)
87         GPIO.output(pin, 0)
88
89     clockwise # Tells the motor which way to rotate (this variable was declared at the beginning of code).
90     for i in range(30): #The degrees you want the motor to rotate (512= 360 degrees, 30= 21.1 degrees).
91         for halfstep in range(8):
92             for pin in range(4):
```

```

94     GPIO.output(control_pins[pin], clockwise[halfstep][pin])
95     time.sleep(.005)
96
97 # START OF LOOP
98 for j in range(60): # Determines how many sweeps the motor will do (Sweep = 15.5 degrees counterclockwise then 15.5 degrees clockwise).
99     # Also the degree range for the vertical motor (60*360 / 512 = 42.19 degrees) (+21 to -21 degrees)
100    control_pins = [12,16,18,22] #horizontal motor
101    for pin in control_pins:
102        GPIO.setup(pin, GPIO.OUT)
103        GPIO.output(pin, 0)
104    for k in range(1,46): # Using the fullstepping configuration, it takes 44 steps to move 31 degrees
105        counterclockwise # Moving the horizontal motor counterclockwise and recoding data
106        for i in range(1):
107            for halfstep in range(8):
108                for pin in range(4):
109                    GPIO.output(control_pins[pin], counterclockwise[halfstep][pin])
110                    time.sleep(.009)
111
112        horizontal_angle= (23-k)*4*.1758 # Records the horizontal angle throughout the sweep. Goes from -15.5 to 15.5 degrees.
113        print("horizontal angle = %s" % (horizontal_angle)) # Shows the real-time horizontal angle value in the Shell. Delete if it is not needed.
114        sample_data1= open("horizontal_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
115        with open("horizontal_angle.xyz", "a") as f:
116            f.write(str(horizontal_angle)+ '\n') # Writes the angle value in a vertical list to the horizontal_angle.xyz file.
117
118        vertical_angle= (30-j)*4*.1758 # Records the vertical angle. Goes from 21.1 to -21.1 degrees.
119        print("vertical angle = %s" % (vertical_angle)) # Displays the real-time vertical angle value in the Shell. Delete if it is not needed.
120        sample_data2= open("vertical_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
121        with open("vertical_angle.xyz", "a") as f:
122            f.write(str(vertical_angle)+ '\n') # Writes the vertical angle value in a vertical list to the vertical_angle.xyz file.
123
124        distance = lidar.getDistance() # Uses the lidar.getDistance function from the lidar-lite.py file from Garmin
125        real_distance= distance*math.cos(math.radians(vertical_angle))*math.cos(math.radians(horizontal_angle))
126        print("Distance = %s" % (real_distance)) # Displays the real-time distance value in the Shell. Delete if it is not needed.
127        sample_data1= open("distance.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
128        with open("distance.xyz", "a") as f:
129            f.write(str(real_distance)+ '\n') # Writes the distance value in a vertical list to the distance.xyz file.
130
131    # Moving the motor clockwise back to the starting position. No data is recorded
132    control_pins = [12,16,18,22] #horizontal motor
133    for pin in control_pins:
134        GPIO.setup(pin, GPIO.OUT)
135        GPIO.output(pin, 0)
136    for k in range(1,46):
137        clockwise
138        for i in range(1):
139            for halfstep in range(8):
140                for pin in range(4):
141                    GPIO.output(control_pins[pin], clockwise[halfstep][pin])
142                    time.sleep(.005)
143
144    # Moves the vertical motor down .7 degrees going from +21.1 to -21.1
145    control_pins = [29,31,32,33] #vertical motor
146    for pin in control_pins:
147        GPIO.setup(pin, GPIO.OUT)
148        GPIO.output(pin, 0)
149    counterclockwise
150    for i in range(1): #this is for the degrees you want the motor to spin (512= 360 degrees)
151        for halfstep in range(8):
152            for pin in range(4):
153                GPIO.output(control_pins[pin], counterclockwise[halfstep][pin])
154                time.sleep(.009)
155    time.sleep(.05)
156
157    with open('distance.xyz') as file1, open('horizontal_angle.xyz') as file2, open('vertical_angle.xyz') as file3:
158        content1= [entry.strip() for entry in file1]
159        content2= [entry.strip() for entry in file2]
160        content3= [entry.strip() for entry in file3]
161
162    with open('output.xyz', 'w') as file:
163        for entry1, entry2, entry3 in zip(content1, content2, content3):
164            file.write(f'{entry1} {entry2} {entry3}\n')
165
166    # Moving the horizontal motor from its initial position to its starting position (15.5 degrees to the right).
167    control_pins = [12,16,18,22] #horizontal motor
168    for pin in control_pins:
169        GPIO.setup(pin, GPIO.OUT)
170        GPIO.output(pin, 0)
171
172    counterclockwise # Tells the motor which way to rotate (this variable was declared at the beginning of code).
173    for i in range(30): #this is for the degrees you want the motor to spin (512= 360 degrees)
174        for halfstep in range(8):
175            for pin in range(4):
176                GPIO.output(control_pins[pin], counterclockwise[halfstep][pin])
177                time.sleep(.005)
178
179    # Moving the vertical motor from its initial position to its starting position (21.1 degrees up).
180    control_pins = [29,31,32,33] #vertical motor
181    for pin in control_pins:
182        GPIO.setup(pin, GPIO.OUT)
183        GPIO.output(pin, 0)
184
185    clockwise # Tells the motor which way to rotate (this variable was declared at the beginning of code).
186    for i in range(30):
187        for halfstep in range(8):
188            for pin in range(4):
189                GPIO.output(control_pins[pin], clockwise[halfstep][pin])
190                time.sleep(.005)
191
192    GPIO.cleanup()
193
194    # Stops the timer and converts from seconds to minutes
195    stop = timeit.default_timer()
196    print("Execution time: %s minutes" % ((stop-start)/60))
197

```

FIGURE A-3-1: COMMENTED LOW RESOLUTION PYTHON CODE

```

1  # Jarod Bennett. University of Kansas Mechanical Engineering.
2  # Medium accuracy code to produce a quality point cloud in about 11 minutes by running two 28hyj-48 stepper motors with a Garmin Lidar-Lite v3HP using a Raspberry 4.
3  # The final product is a .xyz file that can be read in Matlab to display a 3-D point cloud. The values are distance, horizontal angle, vertical angle.
4
5  # Creating a function that can be called in the Graphical User Interface to run the code
6  def runCode2():
7      import timeit #importing the timeit python library to get a run time for the code
8      start = timeit.default_timer() #starting the timer
9
10     # Opening four .xyz files to write data into. If there is previous data in the .xyz files, the "w" clears them.
11     sample_data1= open("horizontal_angle.xyz", "w")
12     sample_data1.close()
13     sample_data2= open("vertical_angle.xyz", "w")
14     sample_data2.close()
15     sample_data3= open("distance.xyz", "w")
16     sample_data3.close()
17     sample_data4= open("output.xyz", "w")
18     sample_data4.close()
19
20     # Importing the lidar_lite python file that is provided by Garmin. Allows lidar.getdistance to record the distance later on.
21     from lidar_lite import Lidar_Lite
22     lidar = Lidar_Lite()
23     import math # Uses trigonometry from the Python math library to get the distance in the x axis.
24     connected = lidar.connect(1)
25     import time # Importing the time python library to be able to speed up/slow down the motor. Also able to create pauses.
26     import RPi.GPIO as GPIO #general-purpose input/output pins on the Raspberry Pi. Can now refer to it as just GPIO.
27     GPIO.setwarnings(False) #gets rid of
28     GPIO.setmode(GPIO.BOARD) # Set GPIO numbering mode from the pins.
29
30     # Rotates the motor clockwise in the fullstep configuration.
31     clockwise = [
32         [1,0,0,0],
33         [0,1,0,0],
34         [0,0,1,0],
35         [0,0,0,1],
36     ]
37
38     # Rotates the motor counterclockwise in the fullstep configuration.
39     counterclockwise = [
40         [0,0,0,1],
41         [0,0,1,0],
42         [0,1,0,0],
43         [1,0,0,0],
44     ]
45
46     # Moving the horizontal motor from its initial position to its starting position (15.5 degrees to the right).
47     control_pins = [12,16,18,22] #Pins the horizontal motor is connected into the raspberry pi.
48     for pin in control_pins:
49         GPIO.setup(pin, GPIO.OUT)
50         GPIO.output(pin, 0)
51
52     clockwise # Tells the motor which way to rotate (this variable was declared at the beginning of code).
53     for i in range(22): #The degrees you want the motor to rotate (512= 360 degrees, 22= 15.5 degrees).
54         for halfstep in range(4):
55             for pin in range(4):
56                 GPIO.output(control_pins[pin], clockwise[halfstep][pin])
57                 time.sleep(.005)
58
59     # Moving the vertical motor from its initial position to its starting position (21.1 degrees up).
60     control_pins = [29,31,32,33] #Pins the vertical motor is connected into the raspberry pi.
61     for pin in control_pins:
62         GPIO.setup(pin, GPIO.OUT)
63         GPIO.output(pin, 0)
64
65     clockwise # Tells the motor which way to rotate (this variable was declared at the beginning of code).
66     for i in range(30): #The degrees you want the motor to rotate (512= 360 degrees, 30= 21.1 degrees).
67         for halfstep in range(4):
68             for pin in range(4):
69                 GPIO.output(control_pins[pin], clockwise[halfstep][pin])
70                 time.sleep(.005)
71
72     # Separating the full step counterclockwise to 4 individual full steps
73     counterclockwise1 = [
74         [0,0,0,1],
75     ]
76
77     counterclockwise2 = [
78         [0,0,1,0],
79     ]
80
81     counterclockwise3 = [
82         [0,1,0,0],
83     ]
84
85     counterclockwise4 = [
86         [1,0,0,0],
87     ]
88
89     # START OF LOOP
90     for j in range(60): # Determines how many sweeps the motor will do (Sweep = 15.5 degrees counterclockwise then 15.5 degrees clockwise).
91         # Also the degree range for the vertical motor (60*360 / 512 = 42.19 degrees) (+21 to -21 degrees)
92         control_pins = [12,16,18,22] #horizontal motor
93         for pin in control_pins:

```

```

94     GPIO.setup(pin, GPIO.OUT)
95     GPIO.output(pin, 0)
96     for k in range(1,46): # Using the fullstepping configuration, it takes 44 steps to move 31 degrees
97
98     counterclockwise1 # Moving the first full step and recording data
99     for i in range(1):
100         for fullstep in range(1):
101             for pin in range(4):
102                 GPIO.output(control_pins[pin], counterclockwise1[fullstep][pin])
103                 time.sleep(.009)
104
105         horizontal_angle1= (23-k)*4*.1758 # Records the horizontal angle throughout the sweep. Goes from -15.5 to 15.5 degrees.
106         print("horizontal angle = %s" % (horizontal_angle1)) # Shows the real-time horizontal angle value in the Shell. Delete if it is not needed.
107         sample_data1= open("horizontal_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
108         with open("horizontal_angle.xyz", "a") as f:
109             f.write(str(horizontal_angle1)+ '\n') # Writes the angle value in a vertical list to the horizontal_angle.xyz file.
110
111         vertical_angle= (30-j)*4*.1758 # Records the vertical angle. Goes from 21.1 to -21.1 degrees.
112         print("vertical angle = %s" % (vertical_angle)) # Displays the real-time vertical angle value in the Shell. Delete if it is not needed.
113         sample_data2= open("vertical_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
114         with open("vertical_angle.xyz", "a") as f:
115             f.write(str(vertical_angle)+ '\n') # Writes the vertical angle value in a vertical list to the vertical_angle.xyz file.
116
117         distance = lidar.getDistance() # Uses the lidar.getDistance function from the lidar-lite.py file from Garmin
118         real_distance= distance*math.cos(math.radians(vertical_angle))*math.cos(math.radians(horizontal_angle1))
119         print("Distance = %s" % (real_distance)) # Displays the real-time distance value in the Shell. Delete if it is not needed.
120         sample_data= open("distance.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
121         with open("distance.xyz", "a") as f:
122             f.write(str(real_distance)+ '\n') # Writes the distance value in a vertical list to the distance.xyz file.
123
124     counterclockwise2 # Moving the second full step and recording data
125     for i in range(1):
126         for fullstep in range(1):
127             for pin in range(4):
128                 GPIO.output(control_pins[pin], counterclockwise2[fullstep][pin])
129                 time.sleep(.009)
130
131         horizontal_angle2= horizontal_angle1-.1758 # Records the horizontal angle throughout the sweep. Goes from -15.5 to 15.5 degrees.
132         print("horizontal angle = %s" % (horizontal_angle2)) # Shows the real-time horizontal angle value in the Shell. Delete if it is not needed.
133         sample_data1= open("horizontal_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
134         with open("horizontal_angle.xyz", "a") as f:
135             f.write(str(horizontal_angle2)+ '\n') # Writes the angle value in a vertical list to the horizontal_angle.xyz file.
136
137         vertical_angle= (30-j)*4*.1758 # Records the vertical angle. Goes from 21.1 to -21.1 degrees.
138         print("vertical angle = %s" % (vertical_angle)) # Displays the real-time vertical angle value in the Shell. Delete if it is not needed.
139         sample_data2= open("vertical_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
140         with open("vertical_angle.xyz", "a") as f:
141             f.write(str(vertical_angle)+ '\n') # Writes the vertical angle value in a vertical list to the vertical_angle.xyz file.
142
143         distance = lidar.getDistance() # Uses the lidar.getDistance function from the lidar-lite.py file from Garmin
144         real_distance= distance*math.cos(math.radians(vertical_angle))*math.cos(math.radians(horizontal_angle2))
145         print("Distance = %s" % (real_distance)) # Displays the real-time distance value in the Shell. Delete if it is not needed.
146         sample_data= open("distance.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
147         with open("distance.xyz", "a") as f:
148             f.write(str(real_distance)+ '\n') # Writes the distance value in a vertical list to the distance.xyz file.
149
150     counterclockwise3 # Moving the third full step and recording data
151     for i in range(1):
152         for fullstep in range(1):
153             for pin in range(4):
154                 GPIO.output(control_pins[pin], counterclockwise3[fullstep][pin])
155                 time.sleep(.009)
156
157         horizontal_angle3= horizontal_angle2-.1758 # Records the horizontal angle throughout the sweep. Goes from -15.5 to 15.5 degrees.
158         print("horizontal angle = %s" % (horizontal_angle3)) # Shows the real-time horizontal angle value in the Shell. Delete if it is not needed.
159         sample_data1= open("horizontal_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
160         with open("horizontal_angle.xyz", "a") as f:
161             f.write(str(horizontal_angle3)+ '\n') # Writes the angle value in a vertical list to the horizontal_angle.xyz file.
162
163         vertical_angle= (30-j)*4*.1758 # Records the vertical angle. Goes from 21.1 to -21.1 degrees.
164         print("vertical angle = %s" % (vertical_angle)) # Displays the real-time vertical angle value in the Shell. Delete if it is not needed.
165         sample_data2= open("vertical_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
166         with open("vertical_angle.xyz", "a") as f:
167             f.write(str(vertical_angle)+ '\n') # Writes the vertical angle value in a vertical list to the vertical_angle.xyz file.
168
169         distance = lidar.getDistance() # Uses the lidar.getDistance function from the lidar-lite.py file from Garmin
170         real_distance= distance*math.cos(math.radians(vertical_angle))*math.cos(math.radians(horizontal_angle3))
171         print("Distance = %s" % (real_distance)) # Displays the real-time distance value in the Shell. Delete if it is not needed.
172         sample_data= open("distance.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
173         with open("distance.xyz", "a") as f:
174             f.write(str(real_distance)+ '\n') # Writes the distance value in a vertical list to the distance.xyz file.
175
176     counterclockwise4 # Moving the fourth full step and recording data
177     for i in range(1):
178         for fullstep in range(1):
179             for pin in range(4):
180                 GPIO.output(control_pins[pin], counterclockwise4[fullstep][pin])
181                 time.sleep(.009)
182
183         horizontal_angle4= horizontal_angle3-.1758 # Records the horizontal angle throughout the sweep. Goes from -15.5 to 15.5 degrees.
184         print("horizontal angle = %s" % (horizontal_angle4)) # Shows the real-time horizontal angle value in the Shell. Delete if it is not needed.
185         sample_data1= open("horizontal_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
186         with open("horizontal_angle.xyz", "a") as f:
187             f.write(str(horizontal_angle4)+ '\n') # Writes the angle value in a vertical list to the horizontal_angle.xyz file.
188
189         vertical_angle= (30-j)*4*.1758 # Records the vertical angle. Goes from 21.1 to -21.1 degrees.
190         print("vertical angle = %s" % (vertical_angle)) # Displays the real-time vertical angle value in the Shell. Delete if it is not needed.
191         sample_data2= open("vertical_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
192         with open("vertical_angle.xyz", "a") as f:
193             f.write(str(vertical_angle)+ '\n') # Writes the vertical angle value in a vertical list to the vertical_angle.xyz file.
194
195         distance = lidar.getDistance() # Uses the lidar.getDistance function from the lidar-lite.py file from Garmin
196         real_distance= distance*math.cos(math.radians(vertical_angle))*math.cos(math.radians(horizontal_angle4))
197         print("Distance = %s" % (real_distance)) # Displays the real-time distance value in the Shell. Delete if it is not needed.
198         sample_data= open("distance.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
199         with open("distance.xyz", "a") as f:
200             f.write(str(real_distance)+ '\n') # Writes the distance value in a vertical list to the distance.xyz file.
201
202     # Moving the motor clockwise back to the starting position. No data is recorded
203     control_pins = [12,16,18,22] #horizontal motor
204     for pin in control_pins:
205         GPIO.setup(pin, GPIO.OUT)
206         GPIO.output(pin, 0)
207     for k in range(1,46): # Using the fullstepping configuration, it takes 44 steps to move 31 degrees
208         clockwise
209         for i in range(1): # Moves 1 step (.7 degrees) a total of 44 times
210             for fullstep in range(4):
211                 for pin in range(4):
212                     GPIO.output(control_pins[pin], clockwise[fullstep][pin])
213                     time.sleep(.009)
214
215     # Moves the vertical motor down .7 degrees going from +21.1 to -21.1
216     control_pins = [29,31,32,33] #vertical motor
217     for pin in control_pins:
218         GPIO.setup(pin, GPIO.OUT)
219         GPIO.output(pin, 0)
220     counterclockwise
221     for i in range(1): # Moves 1 step (.7 degrees) a total of 60 times (42.2 degrees)
222         for fullstep in range(4):
223             for pin in range(4):
224                 GPIO.output(control_pins[pin], counterclockwise[fullstep][pin])
225                 time.sleep(.009)

```



```

227 # This section converts the distance, vertical, and horizontal .xyz files into entry strips to be combined into one file
228 with open('distance.xyz') as file1, open('horizontal_angle.xyz') as file2, open('vertical_angle.xyz') as file3:
229     content1= [entry.strip() for entry in file1]
230     content2= [entry.strip() for entry in file2]
231     content3= [entry.strip() for entry in file3]
232
233 # Produces a .xyz file with three columns: distance, horizontal angle, vertical angle.
234 with open('output.xyz', 'w') as file:
235     for entry1, entry2, entry3 in zip(content1, content2, content3):
236         file.write(f'{entry1} {entry2} {entry3}\n')
237
238 # Moving the horizontal motor from its initial position to its starting position (15.5 degrees to the right).
239 control_pins = [12,16,18,22] #Pins the horizontal motor is connected into the raspberry pi.
240 for pin in control_pins:
241     GPIO.setup(pin, GPIO.OUT)
242     GPIO.output(pin, 0)
243
244 counterclockwise # Tells the motor which way to rotate (this variable was declared at the beginning of code).
245 for i in range(22): #The degrees you want the motor to rotate (512= 360 degrees, 22= 15.5 degrees).
246     for fullstep in range(4):
247         for pin in range(4):
248             GPIO.output(control_pins[pin], counterclockwise[fullstep][pin])
249             time.sleep(.005)
250
251 # Moving the vertical motor from its initial position to its starting position (21.1 degrees up).
252 control_pins = [29,31,32,33] #Pins the vertical motor is connected into the raspberry pi.
253 for pin in control_pins:
254     GPIO.setup(pin, GPIO.OUT)
255     GPIO.output(pin, 0)
256
257 clockwise # Tells the motor which way to rotate (this variable was declared at the beginning of code).
258 for i in range(30): #The degrees you want the motor to rotate (512= 360 degrees, 30= 21.1 degrees).
259     for fullstep in range(4):
260         for pin in range(4):
261             GPIO.output(control_pins[pin], clockwise[fullstep][pin])
262             time.sleep(.005)
263 GPIO.cleanup()
264
265 # Stops the timer and converts from seconds to minutes
266 stop = timeit.default_timer()
267 print("Execution time: %s minutes" % ((stop-start)/60))
268

```

FIGURE A-3-2: COMMENTED MEDIUM RESOLUTION PYTHON CODE

```

1 # Jarod Bennett, University of Kansas Mechanical Engineering.
2 # High accuracy code to produce the best point cloud using two 28byj-48 stepper motors with a Garmin Lidar-Lite v3HP using a Raspberry 4.
3 # The final product is a .xyz file that can be read in Matlab to display a 3-D point cloud. The values are distance, horizontal angle, vertical angle.
4
5 # Creating a function that can be called in the Graphical User Interface to run the code
6 def runCode3():
7     import timeit #importing the timeit python library to get a run time for the code
8     start = timeit.default_timer() #starting the timer
9
10    # Opening four .xyz files to write data into. If there is previous data in the .xyz files, the "w" clears them.
11    sample_data1= open("horizontal_angle.xyz", "w")
12    sample_data1.close()
13    sample_data2= open("vertical_angle.xyz", "w")
14    sample_data2.close()
15    sample_data3= open("distance.xyz", "w")
16    sample_data3.close()
17    sample_data4= open("output.xyz", "w")
18    sample_data4.close()
19
20    # Importing the lidar_lite python file that is provided by Garmin. Allows lidar.getdistance to record the distance later on.
21    from lidar_lite import Lidar_Lite
22    lidar = Lidar_Lite()
23    import math # Uses trigonometry from the Python math library to get the distance in the x axis.
24    connected = lidar.connect(1)
25    import time # Importing the time python library to be able to speed up/slow down the motor. Also able to create pauses.
26    import RPi.GPIO as GPIO #general-purpose input/output pins on the Raspberry Pi. Can now refer to it as just GPIO.
27    GPIO.setwarnings(False) #gets rid of
28    GPIO.setmode(GPIO.BOARD) # Set GPIO numbering mode from the pins.
29
30    # Rotates the motor clockwise in the halfstep configuration.
31    clockwise = [
32        [1,0,0,1],
33        [1,0,0,0],
34        [1,1,0,0],
35        [0,1,0,0],
36        [0,1,1,0],
37        [0,0,1,0],
38        [0,0,1,1],
39        [0,0,0,1]
40    ]
41
42    # Rotates the motor counterclockwise in the halfstep configuration.
43    counterclockwise = [
44        [0,0,0,1],
45        [0,0,1,1],
46        [0,0,1,0],
47        [0,1,1,0],
48        [0,1,0,0],
49        [1,1,0,0],
50        [1,0,0,0],
51        [1,0,0,1],
52    ]
53
54    # Moving the horizontal motor from its initial position to its starting position (15.5 degrees to the right).
55    control_pins = [12,16,18,22] #Pins the horizontal motor is connected into the raspberry pi.
56    for pin in control_pins:
57        GPIO.setup(pin, GPIO.OUT)
58        GPIO.output(pin, 0)
59
60    clockwise # Tells the motor which way to rotate (this variable was declared at the beginning of code).
61    for i in range(22): #The degrees you want the motor to rotate (512= 360 degrees, 22= 15.5 degrees).
62        for halfstep in range(8):
63            for pin in range(4):
64                GPIO.output(control_pins[pin], clockwise[halfstep][pin])
65                time.sleep(.005)
66
67    # Moving the vertical motor from its initial position to its starting position (21.1 degrees up).
68    control_pins = [29,31,32,33] #Pins the vertical motor is connected into the raspberry pi.
69    for pin in control_pins:
70        GPIO.setup(pin, GPIO.OUT)
71        GPIO.output(pin, 0)
72
73    clockwise # Tells the motor which way to rotate (this variable was declared at the beginning of code).
74    for i in range(30): #The degrees you want the motor to rotate (512= 360 degrees, 30= 21.1 degrees).
75        for halfstep in range(8):
76            for pin in range(4):
77                GPIO.output(control_pins[pin], clockwise[halfstep][pin])
78                time.sleep(.005)
79
80    # Separating the half step counterclockwise to 8 individual half steps
81    counterclockwise1 = [
82        [0,0,0,1],
83    ]
84
85    counterclockwise2 = [
86        [0,0,1,1],
87    ]
88
89    counterclockwise3 = [
90        [0,0,1,0],
91    ]
92

```

```

93 counter-clockwise4 = [
94     [0,1,1,0],
95 ]
96
97 counter-clockwise5 = [
98     [0,1,0,0],
99 ]
100
101 counter-clockwise6 = [
102     [1,1,0,0],
103 ]
104
105 counter-clockwise7 = [
106     [1,0,0,0],
107 ]
108
109 counter-clockwise8 = [
110     [1,0,0,1],
111 ]
112
113 # START OF LOOP
114 for j in range(60): # Determines how many sweeps the motor will do (Sweep = 15.5 degrees counterclockwise then 15.5 degrees clockwise).
115     # Also the degree range for the vertical motor (60*360 / 512 = 42.19 degrees) (+21 to -21 degrees)
116     control_pins = [12,16,18,22] #horizontal motor
117     for pin in control_pins:
118         GPIO.setup(pin, GPIO.OUT)
119         GPIO.output(pin, 0)
120     for k in range(1,46): # Using the fullstepping configuration, it takes 45 steps to move 31 degrees (+15.5 to -15.5 degrees)
121
122         counter-clockwise1 # Moving the first half step and recording data
123         for i in range(1):
124             for fullstep in range(1):
125                 for pin in range(4):
126                     GPIO.output(control_pins[pin], counter-clockwise1[fullstep][pin])
127                     time.sleep(.009)
128
129                 horizontal_angle1= (23-k)*8*.0879 # Records the horizontal angle throughout the sweep. Goes from -15.5 to 15.5 degrees.
130                 print("horizontal angle = %s" % (horizontal_angle1)) # Shows the real-time horizontal angle value in the Shell. Delete if it is not needed.
131                 sample_data1= open("horizontal_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
132                 with open("horizontal_angle.xyz", "a") as f:
133                     f.write(str(horizontal_angle1)+ '\n') # Writes the angle value in a vertical list to the horizontal_angle.xyz file.
134
135                 vertical_angle1= (30-j)*4*.1758 # Records the vertical angle. Goes from 21.1 to -21.1 degrees.
136                 print("vertical angle = %s" % (vertical_angle1)) # Displays the real-time vertical angle value in the Shell. Delete if it is not needed.
137                 sample_data2= open("vertical_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
138                 with open("vertical_angle.xyz", "a") as f:
139                     f.write(str(vertical_angle1)+ '\n') # Writes the vertical angle value in a vertical list to the vertical_angle.xyz file.
140
141                 distance = lidar.getDistance() # Uses the lidar.getDistance function from the lidar-lite.py file from Garmin
142                 real_distance= distance*math.cos(math.radians(vertical_angle1))*math.cos(math.radians(horizontal_angle1))
143                 print("Distance = %s" % (real_distance)) # Displays the real-time distance value in the Shell. Delete if it is not needed.
144                 sample_data1= open("distance.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
145                 with open("distance.xyz", "a") as f:
146                     f.write(str(real_distance)+ '\n') # Writes the distance value in a vertical list to the distance.xyz file.
147
148         counter-clockwise2 # Moving the second half step and recording data
149         for i in range(1):
150             for fullstep in range(1):
151                 for pin in range(4):
152                     GPIO.output(control_pins[pin], counter-clockwise2[fullstep][pin])
153                     time.sleep(.009)
154
155                 horizontal_angle2= horizontal_angle1-.0879 # Records the horizontal angle throughout the sweep. Goes from -15.5 to 15.5 degrees.
156                 print("horizontal angle = %s" % (horizontal_angle2)) # Shows the real-time horizontal angle value in the Shell. Delete if it is not needed.
157                 sample_data1= open("horizontal_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
158                 with open("horizontal_angle.xyz", "a") as f:
159                     f.write(str(horizontal_angle2)+ '\n') # Writes the angle value in a vertical list to the horizontal_angle.xyz file.
160
161                 vertical_angle2= (30-j)*4*.1758 # Records the vertical angle. Goes from 21.1 to -21.1 degrees.
162                 print("vertical angle = %s" % (vertical_angle2)) # Displays the real-time vertical angle value in the Shell. Delete if it is not needed.
163                 sample_data2= open("vertical_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
164                 with open("vertical_angle.xyz", "a") as f:
165                     f.write(str(vertical_angle2)+ '\n') # Writes the vertical angle value in a vertical list to the vertical_angle.xyz file.
166
167                 distance = lidar.getDistance() # Uses the lidar.getDistance function from the lidar-lite.py file from Garmin
168                 real_distance= distance*math.cos(math.radians(vertical_angle2))*math.cos(math.radians(horizontal_angle2))
169                 print("Distance = %s" % (real_distance)) # Displays the real-time distance value in the Shell. Delete if it is not needed.
170                 sample_data1= open("distance.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
171                 with open("distance.xyz", "a") as f:
172                     f.write(str(real_distance)+ '\n') # Writes the distance value in a vertical list to the distance.xyz file.
173
174         counter-clockwise3 # Moving the third half step and recording data
175         for i in range(1):
176             for fullstep in range(1):
177                 for pin in range(4):
178                     GPIO.output(control_pins[pin], counter-clockwise3[fullstep][pin])
179                     time.sleep(.009)
180
181                 horizontal_angle3= horizontal_angle2-.0879 # Records the horizontal angle throughout the sweep. Goes from -15.5 to 15.5 degrees.
182                 print("horizontal angle = %s" % (horizontal_angle3)) # Shows the real-time horizontal angle value in the Shell. Delete if it is not needed.
183                 sample_data1= open("horizontal_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
184

```

```

185 with open("horizontal_angle.xyz", "a") as f:
186     f.write(str(horizontal_angle3)+'\n') # Writes the angle value in a vertical list to the horizontal_angle.xyz file.
187
188 vertical_angle= (30-j)*4*.1758 # Records the vertical angle. Goes from 21.1 to -21.1 degrees.
189 print("vertical angle = %s" % (vertical_angle)) # Displays the real-time vertical angle value in the Shell. Delete if it is not needed.
190 sample_data2= open("vertical_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
191 with open("vertical_angle.xyz", "a") as f:
192     f.write(str(vertical_angle)+'\n') # Writes the vertical angle value in a vertical list to the vertical_angle.xyz file.
193
194 distance = lidar.getDistance() # Uses the lidar.getDistance function from the lidar-lite.py file from Garmin
195 real_distance= distance*math.cos(math.radians(vertical_angle))*math.cos(math.radians(horizontal_angle3))
196 print("Distance = %s" % (real_distance)) # Displays the real-time distance value in the Shell. Delete if it is not needed.
197 sample_data1= open("distance.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
198 with open("distance.xyz", "a") as f:
199     f.write(str(real_distance)+'\n') # Writes the distance value in a vertical list to the distance.xyz file.
200
201 counterclockwise4 # Moving the fourth half step and recording data
202 for i in range(1):
203     for fullstep in range(1):
204         for pin in range(4):
205             GPIO.output(control_pins[pin], counterclockwise4[fullstep][pin])
206             time.sleep(.009)
207
208 horizontal_angle4= horizontal_angle3-.0879 # Records the horizontal angle throughout the sweep. Goes from -15.5 to 15.5 degrees.
209 print("horizontal angle = %s" % (horizontal_angle4)) # Shows the real-time horizontal angle value in the Shell. Delete if it is not needed.
210 sample_data1= open("horizontal_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
211 with open("horizontal_angle.xyz", "a") as f:
212     f.write(str(horizontal_angle4)+'\n') # Writes the angle value in a vertical list to the horizontal_angle.xyz file.
213
214 vertical_angle= (30-j)*4*.1758 # Records the vertical angle. Goes from 21.1 to -21.1 degrees.
215 print("vertical angle = %s" % (vertical_angle)) # Displays the real-time vertical angle value in the Shell. Delete if it is not needed.
216 sample_data2= open("vertical_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
217 with open("vertical_angle.xyz", "a") as f:
218     f.write(str(vertical_angle)+'\n') # Writes the vertical angle value in a vertical list to the vertical_angle.xyz file.
219
220 distance = lidar.getDistance() # Uses the lidar.getDistance function from the lidar-lite.py file from Garmin
221 real_distance= distance*math.cos(math.radians(vertical_angle))*math.cos(math.radians(horizontal_angle4))
222 print("Distance = %s" % (real_distance)) # Displays the real-time distance value in the Shell. Delete if it is not needed.
223 sample_data1= open("distance.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
224 with open("distance.xyz", "a") as f:
225     f.write(str(real_distance)+'\n') # Writes the distance value in a vertical list to the distance.xyz file.
226
227 counterclockwise5 # Moving the fifth half step and recording data
228 for i in range(1):
229     for fullstep in range(1):
230         for pin in range(4):
231             GPIO.output(control_pins[pin], counterclockwise5[fullstep][pin])
232             time.sleep(.009)
233
234 horizontal_angle5= horizontal_angle4-.0879 # Records the horizontal angle throughout the sweep. Goes from -15.5 to 15.5 degrees.
235 print("horizontal angle = %s" % (horizontal_angle5)) # Shows the real-time horizontal angle value in the Shell. Delete if it is not needed.
236 sample_data1= open("horizontal_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
237 with open("horizontal_angle.xyz", "a") as f:
238     f.write(str(horizontal_angle5)+'\n') # Writes the angle value in a vertical list to the horizontal_angle.xyz file.
239
240 vertical_angle= (30-j)*4*.1758 # Records the vertical angle. Goes from 21.1 to -21.1 degrees.
241 print("vertical angle = %s" % (vertical_angle)) # Displays the real-time vertical angle value in the Shell. Delete if it is not needed.
242 sample_data2= open("vertical_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
243 with open("vertical_angle.xyz", "a") as f:
244     f.write(str(vertical_angle)+'\n') # Writes the vertical angle value in a vertical list to the vertical_angle.xyz file.
245
246 distance = lidar.getDistance() # Uses the lidar.getDistance function from the lidar-lite.py file from Garmin
247 real_distance= distance*math.cos(math.radians(vertical_angle))*math.cos(math.radians(horizontal_angle5))
248 print("Distance = %s" % (real_distance)) # Displays the real-time distance value in the Shell. Delete if it is not needed.
249 sample_data1= open("distance.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
250 with open("distance.xyz", "a") as f:
251     f.write(str(real_distance)+'\n') # Writes the distance value in a vertical list to the distance.xyz file.
252
253 counterclockwise6 # Moving the sixth half step and recording data
254 for i in range(1):
255     for fullstep in range(1):
256         for pin in range(4):
257             GPIO.output(control_pins[pin], counterclockwise6[fullstep][pin])
258             time.sleep(.009)
259
260 horizontal_angle6= horizontal_angle5-.0879 # Records the horizontal angle throughout the sweep. Goes from -15.5 to 15.5 degrees.
261 print("horizontal angle = %s" % (horizontal_angle6)) # Shows the real-time horizontal angle value in the Shell. Delete if it is not needed.
262 sample_data1= open("horizontal_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
263 with open("horizontal_angle.xyz", "a") as f:
264     f.write(str(horizontal_angle6)+'\n') # Writes the angle value in a vertical list to the horizontal_angle.xyz file.
265
266 vertical_angle= (30-j)*4*.1758 # Records the vertical angle. Goes from 21.1 to -21.1 degrees.
267 print("vertical angle = %s" % (vertical_angle)) # Displays the real-time vertical angle value in the Shell. Delete if it is not needed.
268 sample_data2= open("vertical_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
269 with open("vertical_angle.xyz", "a") as f:
270     f.write(str(vertical_angle)+'\n') # Writes the vertical angle value in a vertical list to the vertical_angle.xyz file.
271
272 distance = lidar.getDistance() # Uses the lidar.getDistance function from the lidar-lite.py file from Garmin
273 real_distance= distance*math.cos(math.radians(vertical_angle))*math.cos(math.radians(horizontal_angle6))
274 print("Distance = %s" % (real_distance)) # Displays the real-time distance value in the Shell. Delete if it is not needed.

```

```

275 sample_data1= open("distance.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
276 with open("distance.xyz", "a") as f:
277     f.write(str(real_distance)+ '\n') # Writes the distance value in a vertical list to the distance.xyz file.
278
279 counterclockwise7 # Moving the seventh half step and recording data
280 for i in range(1):
281     for fullstep in range(1):
282         for pin in range(4):
283             GPIO.output(control_pins[pin], counterclockwise7[fullstep][pin])
284             time.sleep(.009)
285
286 horizontal_angle7= horizontal_angle6-.0879 # Records the horizontal angle throughout the sweep. Goes from -15.5 to 15.5 degrees.
287 print("horizontal angle = %s" % (horizontal_angle7)) # Shows the real-time horizontal angle value in the Shell. Delete if it is not needed.
288 sample_data1= open("horizontal_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
289 with open("horizontal_angle.xyz", "a") as f:
290     f.write(str(horizontal_angle7)+ '\n') # Writes the angle value in a vertical list to the horizontal_angle.xyz file.
291
292 vertical_angle= (30-j)*4*.1758 # Records the vertical angle. Goes from 21.1 to -21.1 degrees.
293 print("vertical angle = %s" % (vertical_angle)) # Displays the real-time vertical angle value in the Shell. Delete if it is not needed.
294 sample_data2= open("vertical_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
295 with open("vertical_angle.xyz", "a") as f:
296     f.write(str(vertical_angle)+ '\n') # Writes the vertical angle value in a vertical list to the vertical_angle.xyz file.
297
298 distance = lidar.getDistance() # Uses the lidar.getDistance function from the lidar-lite.py file from Garmin
299 real_distance= distance*math.cos(math.radians(vertical_angle))*math.cos(math.radians(horizontal_angle3))
300 print("Distance = %s" % (real_distance)) # Displays the real-time distance value in the Shell. Delete if it is not needed.
301 sample_data1= open("distance.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
302 with open("distance.xyz", "a") as f:
303     f.write(str(real_distance)+ '\n') # Writes the distance value in a vertical list to the distance.xyz file.
304
305 counterclockwise8 # Moving the eighth half step and recording data
306 for i in range(1):
307     for fullstep in range(1):
308         for pin in range(4):
309             GPIO.output(control_pins[pin], counterclockwise8[fullstep][pin])
310             time.sleep(.009)
311
312 horizontal_angle8= horizontal_angle7-.0879 # Records the horizontal angle throughout the sweep. Goes from -15.5 to 15.5 degrees.
313 print("horizontal angle = %s" % (horizontal_angle8)) # Shows the real-time horizontal angle value in the Shell. Delete if it is not needed.
314 sample_data1= open("horizontal_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
315 with open("horizontal_angle.xyz", "a") as f:
316     f.write(str(horizontal_angle8)+ '\n') # Writes the angle value in a vertical list to the horizontal_angle.xyz file.
317
318 vertical_angle= (30-j)*4*.1758 # Records the vertical angle. Goes from 21.1 to -21.1 degrees.
319 print("vertical angle = %s" % (vertical_angle)) # Displays the real-time vertical angle value in the Shell. Delete if it is not needed.
320 sample_data2= open("vertical_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
321 with open("vertical_angle.xyz", "a") as f:
322     f.write(str(vertical_angle)+ '\n') # Writes the vertical angle value in a vertical list to the vertical_angle.xyz file.
323
324 distance = lidar.getDistance() # Uses the lidar.getDistance function from the lidar-lite.py file from Garmin
325 real_distance= distance*math.cos(math.radians(vertical_angle))*math.cos(math.radians(horizontal_angle3))
326 print("Distance = %s" % (real_distance)) # Displays the real-time distance value in the Shell. Delete if it is not needed.
327 sample_data1= open("distance.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
328 with open("distance.xyz", "a") as f:
329     f.write(str(real_distance)+ '\n') # Writes the distance value in a vertical list to the distance.xyz file.
330
331 # Moving the motor clockwise back to the starting position. No data is recorded
332 control_pins = [12,16,18,22] #horizontal motor
333 for pin in control_pins:
334     GPIO.setup(pin, GPIO.OUT)
335     GPIO.output(pin, 0)
336 for k in range(1,46): # Using the fullstepping configuration, it takes 44 steps to move 31 degrees
337     clockwise
338     for i in range(1): # Moves 1 step (.7 degrees) a total of 44 times
339         for halfstep in range(8):
340             for pin in range(4):
341                 GPIO.output(control_pins[pin], clockwise[halfstep][pin])
342                 time.sleep(.005)
343
344 # Moves the vertical motor down .7 degrees going from +21.1 to -21.1
345 control_pins = [29,31,32,33] #vertical motor
346 for pin in control_pins:
347     GPIO.setup(pin, GPIO.OUT)
348     GPIO.output(pin, 0)
349 counterclockwise
350 for i in range(1): # Moves 1 step (.7 degrees) a total of 60 times (42.2 degrees)
351     for halfstep in range(8):
352         for pin in range(4):
353             GPIO.output(control_pins[pin], counterclockwise[halfstep][pin])
354             time.sleep(.009)
355
356 # This section converts the distance, vertical, and horizontal .xyz files into entry strips to be combined into one file
357 with open("distance.xyz") as file1, open("horizontal_angle.xyz") as file2, open("vertical_angle.xyz") as file3:
358     content1= [entry.strip() for entry in file1]
359     content2= [entry.strip() for entry in file2]
360     content3= [entry.strip() for entry in file3]
361
362 # Produces a .xyz file with three columns: distance, horizontal angle, vertical angle.
363 with open("output.xyz", "w") as file:
364     for entry1, entry2, entry3 in zip(content1, content2, content3):
365         file.write(f'{entry1} {entry2} {entry3}\n')
366
367 # Moving the horizontal motor from its initial position to its starting position (15.5 degrees to the right).
368 control_pins = [12,16,18,22] #Pins the horizontal motor is connected into the raspberry pi.
369 for pin in control_pins:
370     GPIO.setup(pin, GPIO.OUT)
371     GPIO.output(pin, 0)
372
373 counterclockwise # Tells the motor which way to rotate (this variable was declared at the beginning of code).
374 for i in range(22): #The degrees you want the motor to rotate (512= 360 degrees, 22= 15.5 degrees).
375     for halfstep in range(8):
376         for pin in range(4):
377             GPIO.output(control_pins[pin], counterclockwise[halfstep][pin])
378             time.sleep(.005)
379
380 # Moving the vertical motor from its initial position to its starting position (21.1 degrees up).
381 control_pins = [29,31,32,33] #Pins the vertical motor is connected into the raspberry pi.
382 for pin in control_pins:
383     GPIO.setup(pin, GPIO.OUT)
384     GPIO.output(pin, 0)
385
386 clockwise # Tells the motor which way to rotate (this variable was declared at the beginning of code).
387 for i in range(30): #The degrees you want the motor to rotate (512= 360 degrees, 30= 21.1 degrees).
388     for halfstep in range(8):
389         for pin in range(4):
390             GPIO.output(control_pins[pin], clockwise[halfstep][pin])
391             time.sleep(.005)
392 GPIO.cleanup()
393
394 # Stops the timer and converts from seconds to minutes
395 stop = timeit.default_timer()
396 print("Execution time: %s minutes" % ((stop-start)/60))

```

FIGURE A-3-3: COMMENTED HIGH RESOLUTION PYTHON CODE

APPENDIX A-4

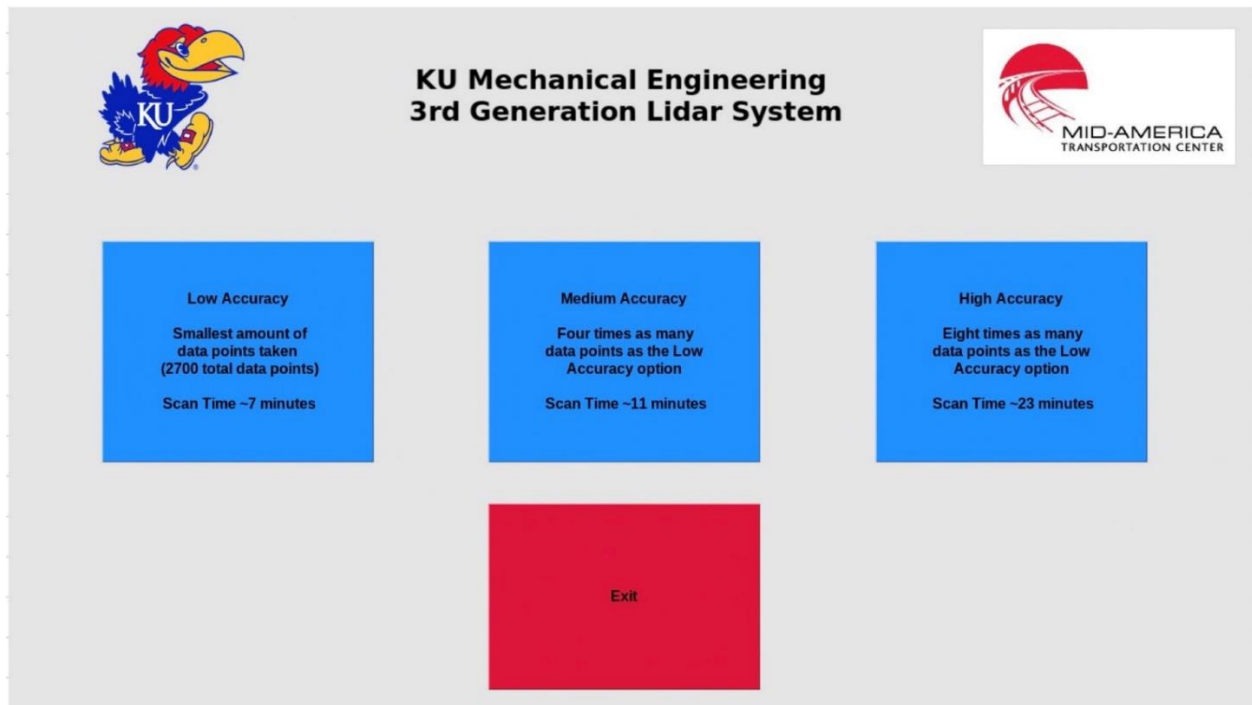


FIGURE A-4-1: SCREEN SHOT OF GRAPHICAL USER INTERFACE

```

1  from tkinter import *
2  from Low import runCode
3  from Medium import runCode2
4  from High import runCode3
5  from PIL import Image, ImageTk
6  import lidar_lite
7
8  root = Tk()
9  root.wm_title("GUI")
10 root.configure(bg="#E5E5E5")
11
12 screen_width = root.winfo_screenwidth()
13 screen_height = root.winfo_screenheight()
14
15 root.attributes("-fullscreen", True)
16
17 def LowRes():
18     runCode()
19
20 def btnExit():
21     root.destroy()
22
23 def end_fullscreen(event):
24     root.attributes("-fullscreen", False)
25
26 def MedRes():
27     runCode2()
28
29 def HighRes():
30     runCode3()
31
32 load= Image.open("jayhawk.png")
33 load = load.resize((round(screen_width*0.12), round(screen_width*0.12)), Image.ANTIALIAS)
34 render = ImageTk.PhotoImage(load)
35 img1 = Label(root, image=render, bg="#E5E5E5")
36 load2= Image.open("MATCimage.jpg")
37 load2 = load2.resize((round(screen_width*0.2025), round(screen_width*0.108)), Image.ANTIALIAS)
38 render2 = ImageTk.PhotoImage(load2)
39 img2 = Label(root, image=render2)
40
41
42
43
44 label_1 = Label(root, text="KU Mechanical Engineering\n 3rd Generation Lidar System", font="Verdana 20 bold",
45                 fg="#000",
46                 bg="#E5E5E5",
47                 pady = round(screen_height*0.01),
48                 padx = round(screen_height*0.01), justify='center')
49
50 lowButton = Button(root, text="Low Accuracy\n \n Smallest amount of data points taken\n (2700 total data points)\n \n Scan Time ~7 minutes", background = "#E990FF",
51                   command=LowRes, height = round(screen_height*0.015), width=round(screen_width*0.022), font = "Arial 12 bold", justify='center', wraplength=180)
52
53 mediumButton = Button(root, text="Medium Accuracy\n \n Four times as many data points as the Low Accuracy option\n \n Scan Time ~11 minutes", background = "#E990FF",
54                       command=MedRes, height = round(screen_height*0.015), width=round(screen_width*0.022), font = "Arial 12 bold", justify='center', wraplength=180)
55
56 highButton = Button(root, text="High Accuracy\n \n Eight times as many data points as the Low Accuracy option\n \n Scan Time ~23 minutes", background = "#E990FF",
57                    command=HighRes, height = round(screen_height*0.015), width=round(screen_width*0.022), font = "Arial 12 bold", justify='center', wraplength=180)
58
59 exitButton = Button(root, text="Exit", background = "#DC143C",
60                    command=btnExit, height = round(screen_height*0.0125), width=round(screen_width*0.022), font = "Arial 12 bold", justify='center', wraplength=300)
61
62
63 img1.place(x=round(screen_width*0.07), y=round(screen_height*0.02))
64 img2.place(x=round(screen_width*0.78), y=round(screen_height*0.03))
65 label_1.place(x=round(screen_width*0.20), y=round(screen_height*0.03))
66 lowButton.place(x=round(screen_width*0.074), y=round(screen_height*0.33))
67 mediumButton.place(x=round(screen_width*0.384), y=round(screen_height*0.33))
68 highButton.place(x=round(screen_width*0.694), y=round(screen_height*0.33))
69 exitButton.place(x=round(screen_width*0.384), y=round(screen_height*0.7))
70
71
72 root.bind("<Escape>", end_fullscreen)
73 root.mainloop()

```

FIGURE A-4-2: GRAPHICAL USER INTERFACE CODE

APPENDIX A-5: MATLAB CODE

```
clc; close all; clear;

[x y z]=xyzread('output.xyz'); %Read .xyz files and separates into
X,Y, and Z components.

for i = 1: length(x) %For loops allows getting rid of points,
shifting rows, and setting domain ranges.
    if x(i) > 350 %Setting Distance in which points at and after
will be deleted
        x(i) = NaN; %Nullifies the point
    end
end

for i = 1: length(x) %For loops allows getting rid of points,
shifting rows, and setting domain ranges.
    if x(i) < 5 %Setting Distance in which points at and after will
be deleted
        x(i) = NaN; %Nullifies the point
    end
end
```

Plotting 3D Scatter plot from data run

```
% 3-D View of plot
figure(1)
scatter3(x,y,z,25,x,'filled'); %Plots points into a 3D scatter.
set(gca, 'YDir', 'reverse'); %Reverses Y-axis Direction
xlabel('X') %Labeling axis
ylabel('Y')
zlabel('Z')
view(-135,35) %Sets orientation of the graph

colorbar %Allows for a color legends for distance
colormap(jet) %Creates a more detailed colorbar.

% 2-D View of plot
figure(2)
scatter3(x,y,z,25,x,'filled'); %Plots points into a 3D scatter.
set(gca, 'YDir', 'reverse'); %Reverses Y-axis Direction
xlabel('X') %Labeling axis
ylabel('Y')
zlabel('Z')
view(-90,-1) %Sets orientation of the graph
colorbar %Allows for a color legends for distance
colormap(jet) %Creates a more detailed colorbar.
```

Contourf Plot

Dr. Depcik came up with this to easily see fully colored image

```
% x => this is the distance data (typically z-direction for contour
plots)

% y => typically the x-direction for contour plots
% z => typically the y-direction for contour plots
Xc = y;
Yc = z;
Zc = x;
resX = 1000; % How many datapoints to plot in the contour X-
direction
resY = 1000; % How many datapoints to plot in the contour Y-
direction
resC = 10; % How many contour lines to show
Xi = linspace(min(Xc),max(Xc),resX); %generates resX # of points
equally spaced between min and max of (Xc)
Yi = linspace(min(Yc),max(Yc),resY); %generates resY # of points
equally spaced between min and max of (Yc)
Zg = griddata(Xc,Yc',Zc,Xi,Yi'); %fits a surface to the
scattered data in vectors Xc, Yc, and Zc

figure(3)
contourf(Xi,Yi',Zg,resC); % Creates 2D filled contour plot
xlabel('Y') % Labeling axis
ylabel('Z')
zlabel('X')
colorbar %Allows for a color legends for distance
colormap(jet) %Creates a more detailed colorbar.
```

Published with MATLAB® R2020a

FIGURE A-5-1: COMMENTED MATLAB CODE