APPENDIX A-1

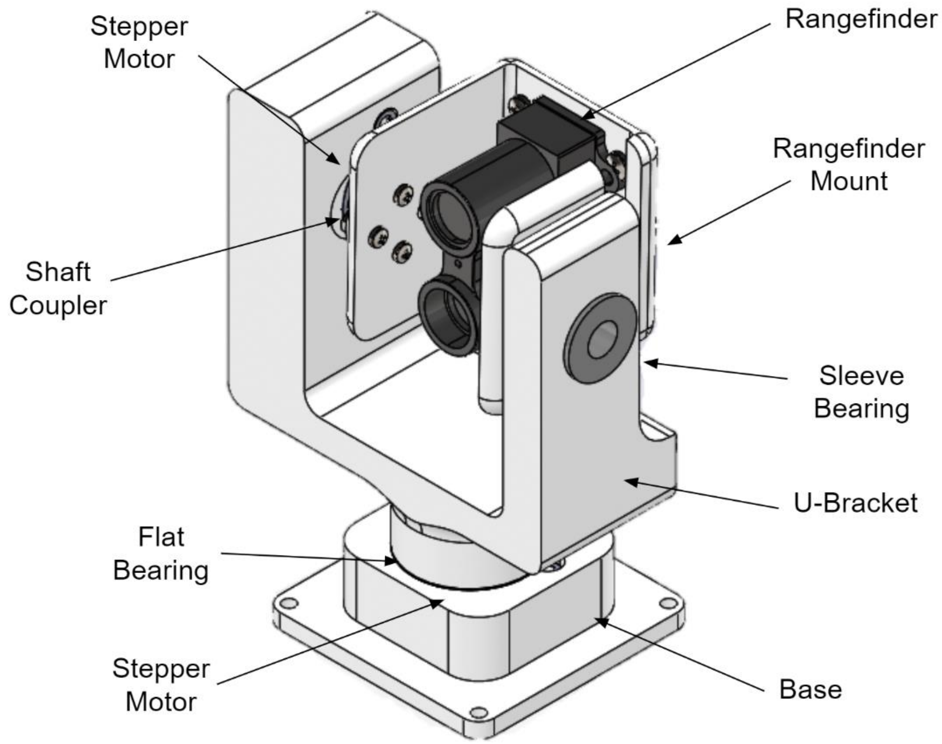

FIGURE A-1-1: ASSEMBLY OF LIDAR HOUSING SHOWING COMPLETE SYSTEM

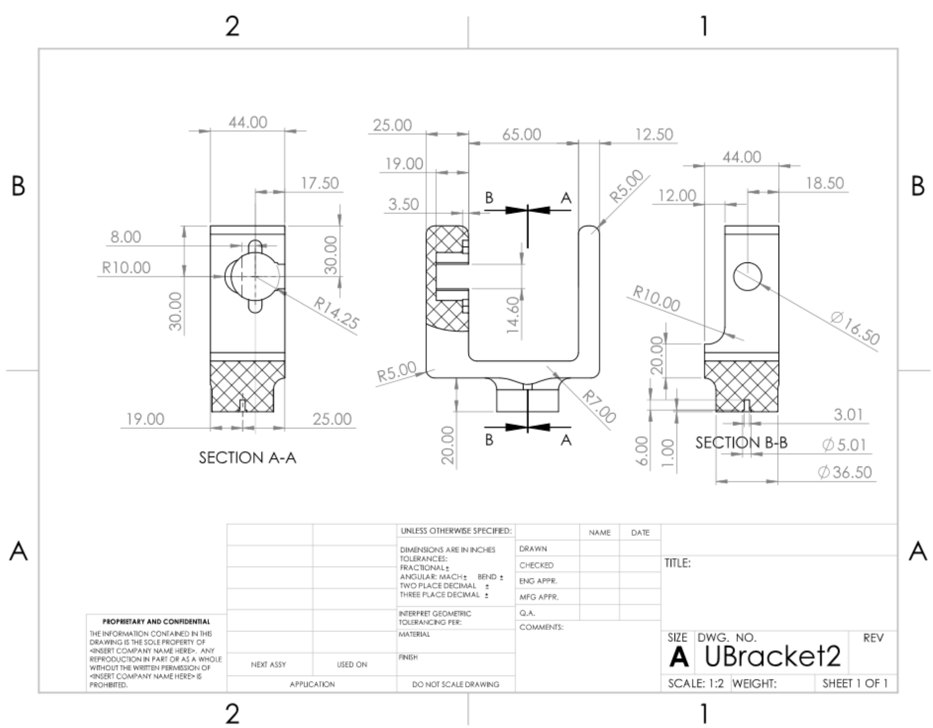

FIGURE A-1-2: U-BRACKET DRAWINGS (DIMENSIONS IN INCHES)

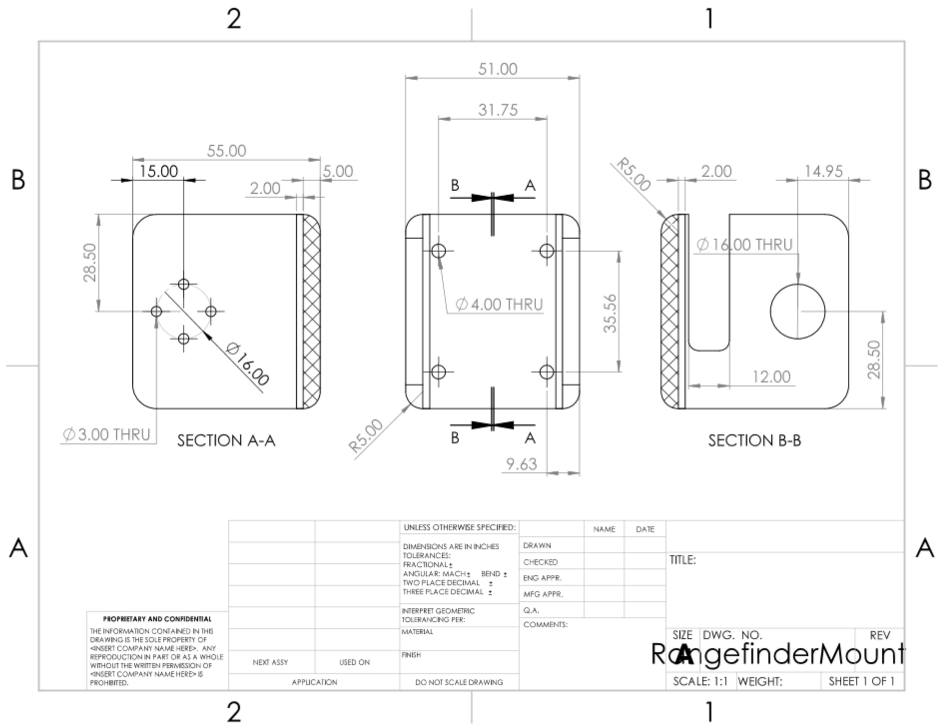

FIGURE A-1-3: RANGEFINDER MOUNT (DIMENSIONS IN INCHES)

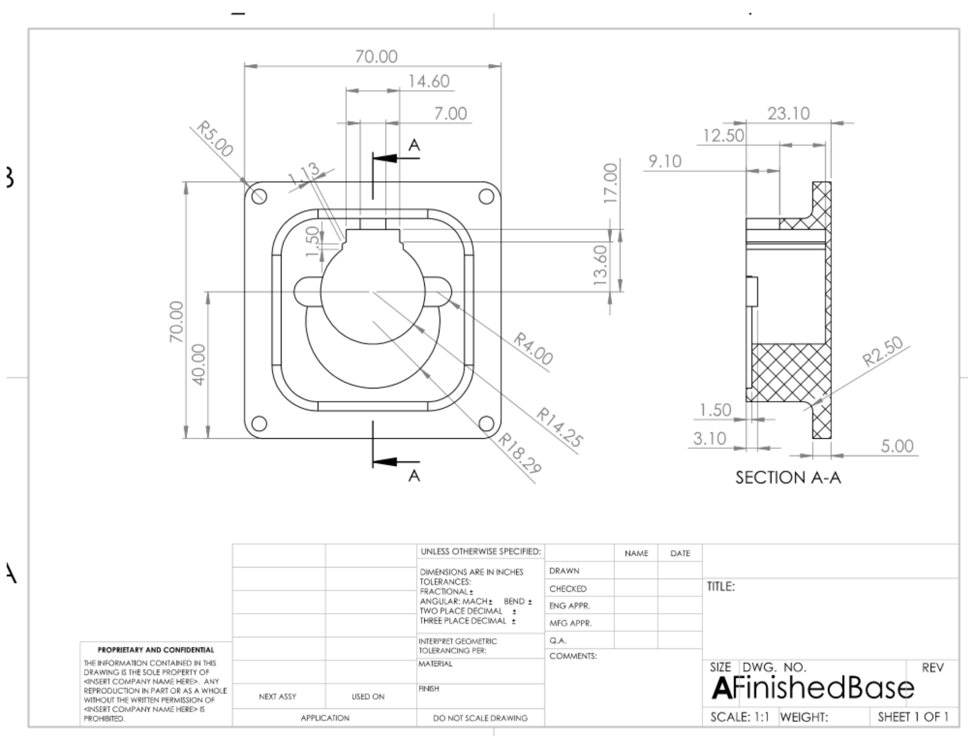

FIGURE A-1-4: BASE FOR LIDAR HOUSING (DIMENSIONS IN INCHES)

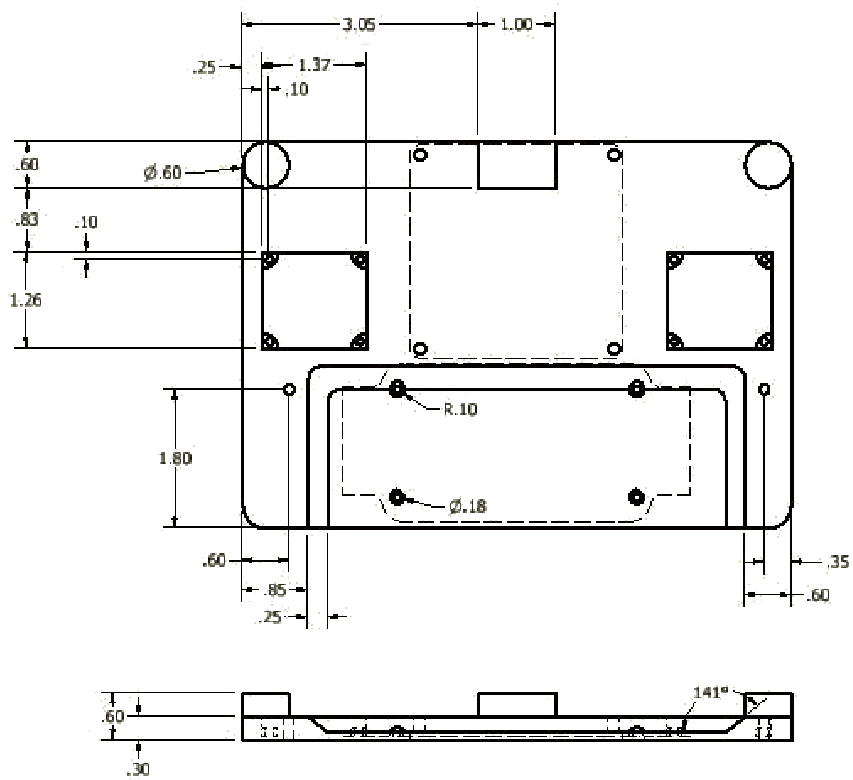

FIGURE A-1-5: MOUNTING PLATE DRAWING 1 OF 2 FOR SYSTEM COMPONENTS (DIMENSIONS IN INCHES)

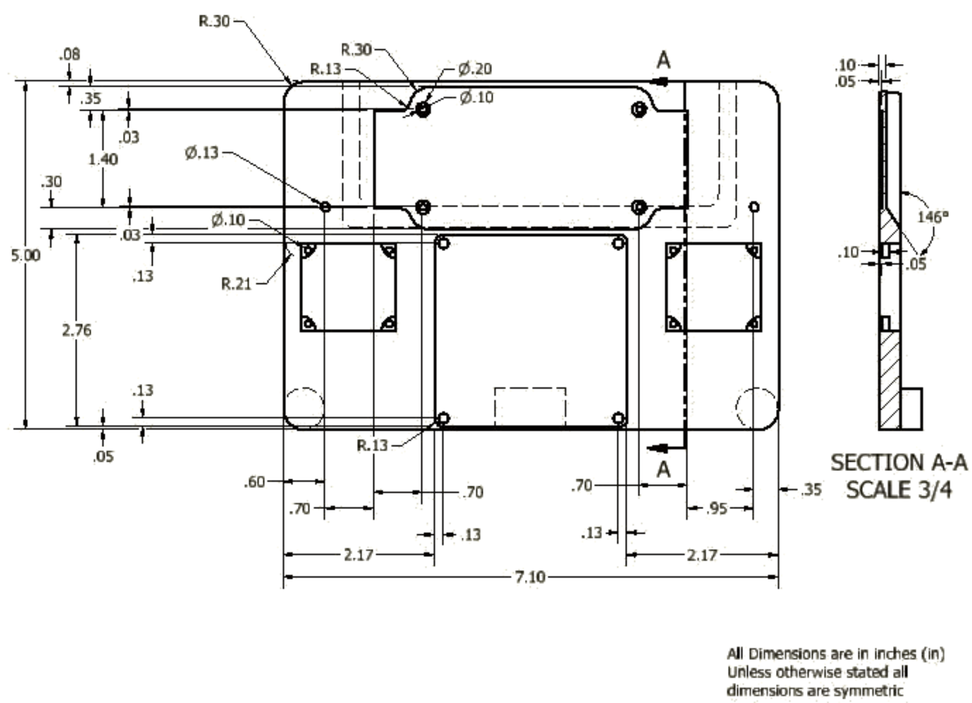

FIGURE A-1-6: MOUNTING PLATE DRAWING 2 OF 2 FOR SYSTEM COMPONENTS (DIMENSIONS IN INCHES)

APPENDIX A-2

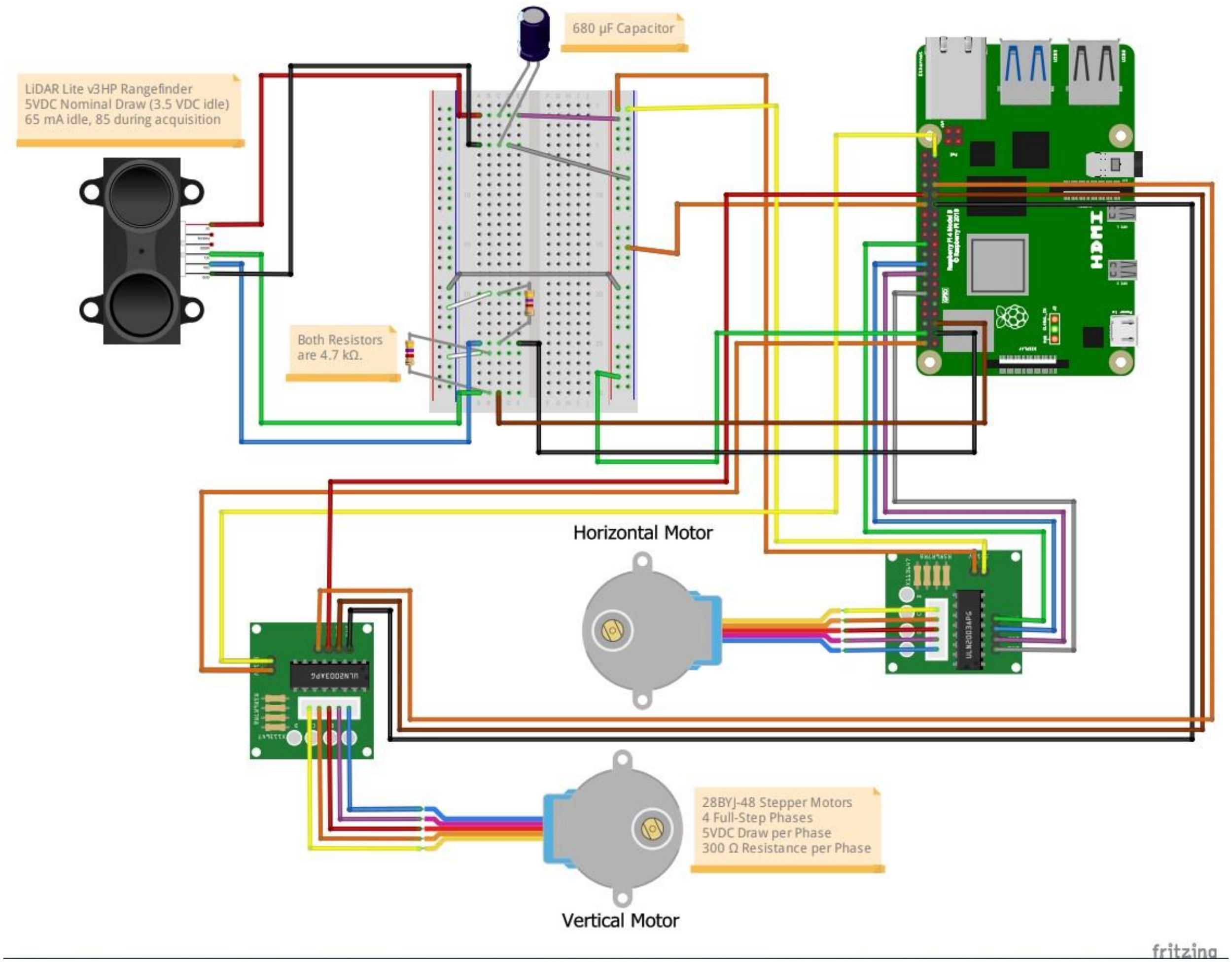

FIGURE A-2-1: WIRING SCHEMATIC FOR LIDAR SYSTEM

APPENDIX A-3: PYTHON CODE

```python
# Jarod Bennett. University of Kansas Mechanical Engineering.
# 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.
# 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.

# Creatinga function that can be called in the Graphical User Interface to run the code

def runCode():
    import timeit #importing the timeit python library to get a run time for the code
    start = timeit.default_timer() #starting the timer

    sample_data1= open("horizontal_angle.xyz", "w")
    sample_data1.close()
    sample_data2= open("vertical_angle.xyz", "w")
    sample_data2.close()
    sample_data3= open("distance.xyz", "w")
    sample_data3.close()
    sample_data4= open("output.xyz", "w")
    sample_data4.close()
    from lidar_lite import Lidar_Lite
    lidar = Lidar_Lite()
    import math
    connected = lidar.connect(1)
    import time
    import RPi.GPIO as GPIO
    GPIO.setwarnings(False)
    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.
    sample_data1= open("horizontal_angle.xyz", "w")
    sample_data1.close()
    sample_data2= open("vertical_angle.xyz", "w")
    sample_data2.close()
    sample_data3= open("distance.xyz", "w")
    sample_data3.close()
    sample_data4= open("output.xyz", "w")
    sample_data4.close()

    # Importing the lidar_lite python file that is provided by Garmin. Allows lidar.getdistance to record the distance later on.
    from lidar_lite import Lidar_Lite
    lidar = Lidar_Lite()
    import math # Uses trigonometry from the Python math library to get the distance in the x axis.
    connected = lidar.connect(1)
    import time # Importing the time python library to be able to speed up/slow down the motor. Also able to create pauses.
    import RPi.GPIO as GPIO #general-purpose input/output pins on the Raspberry Pi. Can now refer to it as just GPIO.
    GPIO.setwarnings(False) #gets rid of
    GPIO.setmode(GPIO.BOARD) # Set GPIO numbering mode from the pins.
    # Rotates the motor clockwise in the halfstep configuration.
    clockwise = [
        [1,0,0,1],
        [1,0,0,0],
        [1,1,0,0],
        [0,1,0,0],
        [0,1,1,0],
        [0,0,1,0],
        [0,0,1,1],
        [0,0,0,1]
    ]

    #  Rotates the motor counterclockwise in the halfstep configuration.
    counterclockwise = [
        [0,0,0,1],
        [0,0,1,1],
        [0,0,1,0],
        [0,1,1,0],
        [0,1,0,0],
        [1,1,0,0],
        [1,0,0,0],
        [1,0,0,1],
    ]

    # Moving the horizontal motor from its initial position to its starting position (15.5 degrees to the right).
    control_pins = [12,16,18,22] #horizontal motor
    for pin in control_pins:
      GPIO.setup(pin, GPIO.OUT)
      GPIO.output(pin, 0)

    clockwise # Tells the motor which way to rotate (this variable was declared at the beginning of code).
    for i in range(30): #this is for the degrees you want the motor to spin (512= 360 degrees)
      for halfstep in range(8):
        for pin in range(4):
          GPIO.output(control_pins[pin], clockwise[halfstep][pin])
        time.sleep(.005)

    # Moving the vertical motor from its initial position to its starting position (21.1 degrees up).
    control_pins = [29,31,32,33] #vertical motor
    for pin in control_pins:
          GPIO.setup(pin, GPIO.OUT)
          GPIO.output(pin, 0)

    clockwise # Tells the motor which way to rotate (this variable was declared at the beginning of code).
    for i in range(30): #The degrees you want the motor to rotate (512= 360 degrees, 30= 21.1 degrees).
        for halfstep in range(8):
            for pin in range(4):
```

```python
          GPIO.output(control_pins[pin], clockwise[halfstep][pin])
        time.sleep(.005)

# START OF LOOP
for j in range(60): # Determines how many sweeps the motor will do (Sweep = 15.5 degrees counterclockwise then 15.5 degrees clockwise).
                    # Also the degree range for the vertical motor (60*360 / 512 = 42.19 degrees) (+21 to -21 degrees))
    control_pins = [12,16,18,22] #horizontal motor
    for pin in control_pins:
        GPIO.setup(pin, GPIO.OUT)
        GPIO.output(pin, 0)
    for k in range(1,46): # Using the fullstepping configuration, it takes 44 steps to move 31 degrees
        counterclockwise # Moving the horizontal motor counterclockwise and recoding data
        for i in range(1):
            for halfstep in range(8):
                for pin in range(4):
                    GPIO.output(control_pins[pin], counterclockwise[halfstep][pin])
                time.sleep(.009)

            horizontal_angle= (23-k)*4*.1758 # Records the horizonal angle throughout the sweep. Goes from -15.5 to 15.5 degrees.
            print("horizontal angle = %s" % (horizontal_angle)) # Shows the real-time horizontal angle value in the Shell. Delete if it is not needed.
            sample_data1= open("horizontal_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
            with open("horizontal_angle.xyz", "a") as f:
                f.write(str(horizontal_angle)+ '\n') # Writes the angle value in a vertical list to the horizontal_angle.xyz file.

            vertical_angle= (30-j)*4*.1758 # Records the vertical angle. Goes from 21.1 to -21.1 degrees.
            print("vertical angle = %s" % (vertical_angle)) # Displays the real-time vertical angle value in the Shell. Delete if it is not needed.
            sample_data2= open("vertical_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
            with open("vertical_angle.xyz", "a") as f:
                f.write(str(vertical_angle)+ '\n') # Writes the vertical angle value in a vertical list to the vertical_angle.xyz file.

            distance = lidar.getDistance() # Uses the lidar.getDistance function from the lidar-lite.py file from Garmin
            real_distance= distance*math.cos(math.radians(vertical_angle))*math.cos(math.radians(horizontal_angle))
            print("Distance = %s" % (real_distance)) # Displays the real-time distance value in the Shell. Delete if it is not needed.
            sample_data1= open("distance.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
            with open("distance.xyz", "a") as f:
                f.write(str(real_distance)+ '\n') # Writes the distance value in a vertical list to the distance.xyz file.

    # Moving the motor clockwise back to the starting position. No data is recorded
    control_pins = [12,16,18,22] #horizontal motor
    for pin in control_pins:
        GPIO.setup(pin, GPIO.OUT)
        GPIO.output(pin, 0)
    for k in range(1,46):
        clockwise
        for i in range(1):
            for halfstep in range(8):
                for pin in range(4):
```

```python
            for halfstep in range(8):
                for pin in range(4):
                  GPIO.output(control_pins[pin], clockwise[halfstep][pin])
                time.sleep(.005)

    # Moves the vertical motor down .7 degrees going from +21.1 to -21.1
    control_pins = [29,31,32,33] #vertical motor
    for pin in control_pins:
        GPIO.setup(pin, GPIO.OUT)
        GPIO.output(pin, 0)
    counterclockwise
    for i in range(1): #this is for the degrees you want the motor to spin (512= 360 degrees)
        for halfstep in range(8):
            for pin in range(4):
                GPIO.output(control_pins[pin], counterclockwise[halfstep][pin])
            time.sleep(.009)
    time.sleep(.05)

with open('distance.xyz') as file1, open('horizontal_angle.xyz') as file2, open('vertical_angle.xyz') as file3:
    content1= [entry.strip() for entry in file1]
    content2= [entry.strip() for entry in file2]
    content3= [entry.strip() for entry in file3]

with open('output.xyz', 'w') as file:
    for entry1, entry2, entry3 in zip(content1, content2, content3):
        file.write(f'{entry1} {entry2} {entry3}\n')

# Moving the horizontal motor from its initial position to its starting position (15.5 degrees to the right).
control_pins = [12,16,18,22] #horizontal motor
for pin in control_pins:
  GPIO.setup(pin, GPIO.OUT)
  GPIO.output(pin, 0)

counterclockwise # Tells the motor which way to rotate (this variable was declared at the beginning of code).
for i in range(30): #this is for the degrees you want the motor to spin (512= 360 degrees)
  for halfstep in range(8):
    for pin in range(4):
      GPIO.output(control_pins[pin], counterclockwise[halfstep][pin])
    time.sleep(.005)

# Moving the vertical motor from its initial position to its starting position (21.1 degrees up).
control_pins = [29,31,32,33] #vertical motor
for pin in control_pins:
      GPIO.setup(pin, GPIO.OUT)
      GPIO.output(pin, 0)
clockwise # Tells the motor which way to rotate (this variable was declared at the beginning of code).
for i in range(30):
    for halfstep in range(8):
        for pin in range(4):
          GPIO.output(control_pins[pin], clockwise[halfstep][pin])
        time.sleep(.005)

GPIO.cleanup()

# Stops the timer and converts from seconds to minutes
stop = timeit.default_timer()
print("Execution time: %s minutes" % ((stop-start)/60))
```

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

```python
# Jarod Bennett. University of Kansas Mechanical Engineering.
# Medium accuracy code to produce a quality point cloud in about 11 minutes by running two 28byj-48 stepper motors with a Garmin Lidar-Lite v3HP using a Raspberry 4.
# 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.

# Creatinga function that can be called in the Graphical User Interface to run the code
def runCode2():
    import timeit #importing the timeit python library to get a run time for the code
    start = timeit.default_timer() #starting the timer

    # Opening four .xyz files to write data into. If there is previous data in the .xyz files, the "w" clears them.
    sample_data1= open("horizontal_angle.xyz", "w")
    sample_data1.close()
    sample_data2= open("vertical_angle.xyz", "w")
    sample_data2.close()
    sample_data3= open("distance.xyz", "w")
    sample_data3.close()
    sample_data4= open("output.xyz", "w")
    sample_data4.close()

    # Importing the lidar_lite python file that is provided by Garmin. Allows lidar.getdistance to record the distance later on.
    from lidar_lite import Lidar_Lite
    lidar = Lidar_Lite()
    import math # Uses trigonometry from the Python math library to get the distance in the x axis.
    connected = lidar.connect(1)
    import time # Importing the time python library to be able to speed up/slow down the motor. Also able to create pauses.
    import RPi.GPIO as GPIO #general-purpose input/output pins on the Raspberry Pi. Can now refer to it as just GPIO.
    GPIO.setwarnings(False) #gets rid of
    GPIO.setmode(GPIO.BOARD) # Set GPIO numbering mode from the pins.

    #  Rotates the motor clockwise in the fullstep configuration.
    clockwise = [
    [1,0,0,0],
    [0,1,0,0],
    [0,0,1,0],
    [0,0,0,1],
    ]

    #  Rotates the motor counterclockwise in the fullstep configuration.
    counterclockwise = [
    [0,0,0,1],
    [0,0,1,0],
    [0,1,0,0],
    [1,0,0,0],
    ]

    # Moving the horizontal motor from its initial position to its starting position (15.5 degrees to the right).
    control_pins = [12,16,18,22] #Pins the horizontal motor is connected into the rapsberry pi.
    for pin in control_pins:
      GPIO.setup(pin, GPIO.OUT)
      GPIO.output(pin, 0)

    clockwise # Tells the motor which way to rotate (this variable was declared at the beginning of code).
    for i in range(22): #The degrees you want the motor to rotate (512= 360 degrees, 22= 15.5 degrees).
      for halfstep in range(4):
        for pin in range(4):
          GPIO.output(control_pins[pin], clockwise[halfstep][pin])
        time.sleep(.005)

    # Moving the vertical motor from its initial position to its starting position (21.1 degrees up).
    control_pins = [29,31,32,33] #Pins the vertical motor is connected into the rapsberry pi.
    for pin in control_pins:
          GPIO.setup(pin, GPIO.OUT)
          GPIO.output(pin, 0)

    clockwise # Tells the motor which way to rotate (this variable was declared at the beginning of code).
    for i in range(30): #The degrees you want the motor to rotate (512= 360 degrees, 30= 21.1 degrees).
        for halfstep in range(4):
            for pin in range(4):
              GPIO.output(control_pins[pin], clockwise[halfstep][pin])
            time.sleep(.005)

    # Separating the full step counterclockwise to 4 individual full steps
    counterclockwise1 = [
        [0,0,0,1],
    ]

    counterclockwise2 = [
        [0,0,1,0],
    ]

    counterclockwise3 = [
        [0,1,0,0],
    ]

    counterclockwise4 = [
        [1,0,0,0],
    ]

    # START OF LOOP
    for j in range(60): # Determines how many sweeps the motor will do (Sweep = 15.5 degrees counterclockwise then 15.5 degrees clockwise).
                        # Also the degree range for the vertical motor (60*360 / 512 = 42.19 degrees) (+21 to -21 degrees))
        control_pins = [12,16,18,22] #horizontal motor
        for pin in control_pins:
```

```python
            GPIO.setup(pin, GPIO.OUT)
            GPIO.output(pin, 0)
        for k in range(1,46): # Using the fullstepping configuration, it takes 44 steps to move 31 degrees

            counterclockwise1 # Moving the first full step and recording data
            for i in range(1):
                for fullstep in range(1):
                    for pin in range(4):
                        GPIO.output(control_pins[pin], counterclockwise1[fullstep][pin])
                    time.sleep(.009)

                horizontal_angle1= (23-k)*4*.1758 # Records the horizonal angle throughout the sweep. Goes from -15.5 to 15.5 degrees.
                print("horizontal angle = %s" % (horizontal_angle1)) # Shows the real-time horizontal angle value in the Shell. Delete if it is not needed.
                sample_data1= open("horizontal_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
                with open("horizontal_angle.xyz", "a") as f:
                    f.write(str(horizontal_angle1)+ '\n') # Writes the angle value in a vertical list to the horizontal_angle.xyz file.

                vertical_angle= (30-j)*4*.1758 # Records the vertical angle. Goes from 21.1 to -21.1 degrees.
                print("vertical angle = %s" % (vertical_angle)) # Displays the real-time vertical angle value in the Shell. Delete if it is not needed.
                sample_data2= open("vertical_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
                with open("vertical_angle.xyz", "a") as f:
                    f.write(str(vertical_angle)+ '\n') # Writes the vertical angle value in a vertical list to the vertical_angle.xyz file.

                distance = lidar.getDistance() # Uses the lidar.getDistance function from the lidar-lite.py file from Garmin
                real_distance= distance*math.cos(math.radians(vertical_angle))*math.cos(math.radians(horizontal_angle1))
                print("Distance = %s" % (real_distance)) # Displays the real-time distance value in the Shell. Delete if it is not needed.
                sample_data1= open("distance.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
                with open("distance.xyz", "a") as f:
                    f.write(str(real_distance)+ '\n') # Writes the distance value in a vertical list to the distance.xyz file.

            counterclockwise2 # Moving the second full step and recording data
            for i in range(1):
                for fullstep in range(1):
                    for pin in range(4):
                        GPIO.output(control_pins[pin], counterclockwise2[fullstep][pin])
                    time.sleep(.009)

                horizontal_angle2= horizontal_angle1-.1758 # Records the horizonal angle throughout the sweep. Goes from -15.5 to 15.5 degrees.
                print("horizontal angle = %s" % (horizontal_angle2)) # Shows the real-time horizontal angle value in the Shell. Delete if it is not needed.
                sample_data1= open("horizontal_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
                with open("horizontal_angle.xyz", "a") as f:
                    f.write(str(horizontal_angle2)+ '\n') # Writes the angle value in a vertical list to the horizontal_angle.xyz file.

                vertical_angle= (30-j)*4*.1758 # Records the vertical angle. Goes from 21.1 to -21.1 degrees.
                print("vertical angle = %s" % (vertical angle)) # Displays the real-time vertical angle value in the Shell. Delete if it is not needed.
                print("vertical angle = %s" % (vertical_angle)) # Displays the real-time vertical angle value in the Shell. Delete if it is not needed.
                sample_data2= open("vertical_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
                with open("vertical_angle.xyz", "a") as f:
                    f.write(str(vertical_angle)+ '\n') # Writes the vertical angle value in a vertical list to the vertical_angle.xyz file.

                distance = lidar.getDistance() # Uses the lidar.getDistance function from the lidar-lite.py file from Garmin
                real_distance= distance*math.cos(math.radians(vertical_angle))*math.cos(math.radians(horizontal_angle2))
                print("Distance = %s" % (real_distance)) # Displays the real-time distance value in the Shell. Delete if it is not needed.
                sample_data1= open("distance.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
                with open("distance.xyz", "a") as f:
                    f.write(str(real_distance)+ '\n') # Writes the distance value in a vertical list to the distance.xyz file.

            counterclockwise3 # Moving the third full step and recording data
            for i in range(1):
                for fullstep in range(1):
                    for pin in range(4):
                        GPIO.output(control_pins[pin], counterclockwise3[fullstep][pin])
                    time.sleep(.009)

                horizontal_angle3= horizontal_angle2-.1758 # Records the horizonal angle throughout the sweep. Goes from -15.5 to 15.5 degrees.
                print("horizontal angle = %s" % (horizontal_angle3)) # Shows the real-time horizontal angle value in the Shell. Delete if it is not needed.
                sample_data1= open("horizontal_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
                with open("horizontal_angle.xyz", "a") as f:
                    f.write(str(horizontal_angle3)+ '\n') # Writes the angle value in a vertical list to the horizontal_angle.xyz file.

                vertical_angle= (30-j)*4*.1758 # Records the vertical angle. Goes from 21.1 to -21.1 degrees.
                print("vertical angle = %s" % (vertical_angle)) # Displays the real-time vertical angle value in the Shell. Delete if it is not needed.
                sample_data2= open("vertical_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
                with open("vertical_angle.xyz", "a") as f:
                    f.write(str(vertical_angle)+ '\n') # Writes the vertical angle value in a vertical list to the vertical_angle.xyz file.

                distance = lidar.getDistance() # Uses the lidar.getDistance function from the lidar-lite.py file from Garmin
                real_distance= distance*math.cos(math.radians(vertical_angle))*math.cos(math.radians(horizontal_angle3))
                print("Distance = %s" % (real_distance)) # Displays the real-time distance value in the Shell. Delete if it is not needed.
                sample_data1= open("distance.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
                with open("distance.xyz", "a") as f:
                    f.write(str(real_distance)+ '\n') # Writes the distance value in a vertical list to the distance.xyz file.

            counterclockwise4 # Moving the fourth full step and recording data
            for i in range(1):
                for fullstep in range(1):
                    for pin in range(4):
                        GPIO.output(control_pins[pin], counterclockwise4[fullstep][pin])
                    time.sleep(.009)
                horizontal_angle4= horizontal_angle3-.1758 # Records the horizonal angle throughout the sweep. Goes from -15.5 to 15.5 degrees.
                print("horizontal angle = %s" % (horizontal_angle4)) # Shows the real-time horizontal angle value in the Shell. Delete if it is not needed.
                sample_data1= open("horizontal_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
                with open("horizontal_angle.xyz", "a") as f:
                    f.write(str(horizontal_angle4)+ '\n') # Writes the angle value in a vertical list to the horizontal_angle.xyz file.

                vertical_angle= (30-j)*4*.1758 # Records the vertical angle. Goes from 21.1 to -21.1 degrees.
                print("vertical angle = %s" % (vertical_angle)) # Displays the real-time vertical angle value in the Shell. Delete if it is not needed.
                sample_data2= open("vertical_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
                with open("vertical_angle.xyz", "a") as f:
                    f.write(str(vertical_angle)+ '\n') # Writes the vertical angle value in a vertical list to the vertical_angle.xyz file.

                distance = lidar.getDistance() # Uses the lidar.getDistance function from the lidar-lite.py file from Garmin
                real_distance= distance*math.cos(math.radians(vertical_angle))*math.cos(math.radians(horizontal_angle4))
                print("Distance = %s" % (real_distance)) # Displays the real-time distance value in the Shell. Delete if it is not needed.
                sample_data1= open("distance.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
                with open("distance.xyz", "a") as f:
                    f.write(str(real_distance)+ '\n') # Writes the distance value in a vertical list to the distance.xyz file.

        # Moving the motor clockwise back to the starting position. No data is recorded
        control_pins = [12,16,18,22] #horizontal motor
        for pin in control_pins:
            GPIO.setup(pin, GPIO.OUT)
            GPIO.output(pin, 0)
        for k in range(1,46): # Using the fullstepping configuration, it takes 44 steps to move 31 degrees
            clockwise
            for i in range(1): # Moves 1 step (.7 degrees) a total of 44 times
                for fullstep in range(4):
                    for pin in range(4):
                      GPIO.output(control_pins[pin], clockwise[fullstep][pin])
                    time.sleep(.009)

        # Moves the vertical motor down .7 degrees going from +21.1 to -21.1
        control_pins = [29,31,32,33] #vertical motor
        for pin in control_pins:
            GPIO.setup(pin, GPIO.OUT)
            GPIO.output(pin, 0)
        counterclockwise
        for i in range(1): # Moves 1 step (.7 degrees) a total of 60 times (42.2 degrees)
            for fullstep in range(4):
                for pin in range(4):
                    GPIO.output(control_pins[pin], counterclockwise[fullstep][pin])
                time.sleep(.009)
```

```python
# This section converts the distance, vertical, and horizontal .xyz files into entry strips to be combined into one file
with open('distance.xyz') as file1, open('horizontal_angle.xyz') as file2, open('vertical_angle.xyz') as file3:
    content1= [entry.strip() for entry in file1]
    content2= [entry.strip() for entry in file2]
    content3= [entry.strip() for entry in file3]

# Produces a .xyz file with three columns:  distance, horizontal angle, vertical angle.
with open('output.xyz', 'w') as file:
    for entry1, entry2, entry3 in zip(content1, content2, content3):
        file.write(f'{entry1} {entry2} {entry3}\n')

# Moving the horizontal motor from its initial position to its starting position (15.5 degrees to the right).
control_pins = [12,16,18,22] #Pins the horizontal motor is connected into the rapsberry pi.
for pin in control_pins:
  GPIO.setup(pin, GPIO.OUT)
  GPIO.output(pin, 0)

counterclockwise # Tells the motor which way to rotate (this variable was declared at the beginning of code).
for i in range(22): #The degrees you want the motor to rotate (512= 360 degrees, 22= 15.5 degrees).
  for fullstep in range(4):
    for pin in range(4):
      GPIO.output(control_pins[pin], counterclockwise[fullstep][pin])
    time.sleep(.005)

# Moving the vertical motor from its initial position to its starting position (21.1 degrees up).
control_pins = [29,31,32,33] #Pins the vertical motor is connected into the rapsberry pi.
for pin in control_pins:
      GPIO.setup(pin, GPIO.OUT)
      GPIO.output(pin, 0)

clockwise # Tells the motor which way to rotate (this variable was declared at the beginning of code).
for i in range(30): #The degrees you want the motor to rotate (512= 360 degrees, 30= 21.1 degrees).
    for fullstep in range(4):
        for pin in range(4):
          GPIO.output(control_pins[pin], clockwise[fullstep][pin])
        time.sleep(.005)
GPIO.cleanup()

# Stops the timer and converts from seconds to minutes
stop = timeit.default_timer()
print("Execution time: %s minutes" % ((stop-start)/60))
```

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

```
# Jarod Bennett. University of Kansas Mechanical Engineering.
# 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.
# 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.

# Creatinga function that can be called in the Graphical User Interface to run the code
def runCode3():
    import timeit #importing the timeit python library to get a run time for the code
    start = timeit.default_timer() #starting the timer

    # Opening four .xyz files to write data into. If there is previous data in the .xyz files, the "w" clears them.
    sample_data1= open("horizontal_angle.xyz", "w")
    sample_data1.close()
    sample_data2= open("vertical_angle.xyz", "w")
    sample_data2.close()
    sample_data3= open("distance.xyz", "w")
    sample_data3.close()
    sample_data4= open("output.xyz", "w")
    sample_data4.close()

    # Importing the lidar_lite python file that is provided by Garmin. Allows lidar.getdistance to record the distance later on.
    from lidar_lite import Lidar_Lite
    lidar = Lidar_Lite()
    import math # Uses trigonometry from the Python math library to get the distance in the x axis.
    connected = lidar.connect(1)
    import time # Importing the time python library to be able to speed up/slow down the motor. Also able to create pauses.
    import RPi.GPIO as GPIO #general-purpose input/output pins on the Raspberry Pi. Can now refer to it as just GPIO.
    GPIO.setwarnings(False) #gets rid of
    GPIO.setmode(GPIO.BOARD) # Set GPIO numbering mode from the pins.

    # Rotates the motor clockwise in the halfstep configuration.
    clockwise = [
        [1,0,0,1],
        [1,0,0,0],
        [1,1,0,0],
        [0,1,0,0],
        [0,1,1,0],
        [0,0,1,0],
        [0,0,1,1],
        [0,0,0,1]
    ]

    #  Rotates the motor counterclockwise in the halfstep configuration.
    counterclockwise = [
        [0,0,0,1],
        [0,0,1,1],
        [0,0,1,0],
        [0,1,1,0],
        [0,1,0,0],
        [1,1,0,0],
        [1,0,0,0],
        [1,0,0,1],
    ]

    # Moving the horizontal motor from its initial position to its starting position (15.5 degrees to the right).
    control_pins = [12,16,18,22] #Pins the horizontal motor is connected into the rapsberry pi.
    for pin in control_pins:
      GPIO.setup(pin, GPIO.OUT)
      GPIO.output(pin, 0)

    clockwise # Tells the motor which way to rotate (this variable was declared at the beginning of code).
    for i in range(22): #The degrees you want the motor to rotate (512= 360 degrees, 22= 15.5 degrees).
      for halfstep in range(8):
        for pin in range(4):
          GPIO.output(control_pins[pin], clockwise[halfstep][pin])
        time.sleep(.005)

    # Moving the vertical motor from its initial position to its starting position (21.1 degrees up).
    control_pins = [29,31,32,33] #Pins the vertical motor is connected into the rapsberry pi.
    for pin in control_pins:
          GPIO.setup(pin, GPIO.OUT)
          GPIO.output(pin, 0)

    clockwise # Tells the motor which way to rotate (this variable was declared at the beginning of code).
    for i in range(30): #The degrees you want the motor to rotate (512= 360 degrees, 30= 21.1 degrees).
        for halfstep in range(8):
            for pin in range(4):
              GPIO.output(control_pins[pin], clockwise[halfstep][pin])
            time.sleep(.005)

    # Separating the half step counterclockwise to 8 individual half steps
    counterclockwise1 = [
        [0,0,0,1],
    ]

    counterclockwise2 = [
        [0,0,1,1],
    ]

    counterclockwise3 = [
        [0,0,1,0],
    ]

```

```python
counterclockwise4 = [
    [0,1,1,0],
]

counterclockwise5 = [
    [0,1,0,0],
]

counterclockwise6 = [
    [1,1,0,0],
]

counterclockwise7 = [
    [1,0,0,0],
]

counterclockwise8 = [
    [1,0,0,1],
]

# START OF LOOP
for j in range(60): # Determines how many sweeps the motor will do (Sweep = 15.5 degrees counterclockwise then 15.5 degrees clockwise).
                    # Also the degree range for the vertical motor (60*360 / 512 = 42.19 degrees) (+21 to -21 degrees))
    control_pins = [12,16,18,22] #horizontal motor
    for pin in control_pins:
        GPIO.setup(pin, GPIO.OUT)
        GPIO.output(pin, 0)
    for k in range(1,46): # Using the fullstepping configuration, it takes 45 steps to move 31 degrees (+15.5 to -15.5 dgrees)

        counterclockwise1 # Moving the first half step and recording data
        for i in range(1):
            for fullstep in range(1):
                for pin in range(4):
                    GPIO.output(control_pins[pin], counterclockwise1[fullstep][pin])
                time.sleep(.009)

            horizontal_angle1= (23-k)*8*.0879 # Records the horizonal angle throughout the sweep. Goes from -15.5 to 15.5 degrees.
            print("horizontal angle = %s" % (horizontal_angle1)) # Shows the real-time horizontal angle value in the Shell. Delete if it is not needed.
            sample_data1= open("horizontal_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
            with open("horizontal_angle.xyz", "a") as f:
                f.write(str(horizontal_angle1)+ '\n') # Writes the angle value in a vertical list to the horizontal_angle.xyz file.


            vertical_angle= (30-j)*4*.1758 # Records the vertical angle. Goes from 21.1 to -21.1 degrees.
            print("vertical angle = %s" % (vertical_angle)) # Displays the real-time vertical angle value in the Shell. Delete if it is not needed.
            sample_data2= open("vertical_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
            with open("vertical_angle.xyz", "a") as f:
                f.write(str(vertical_angle)+ '\n') # Writes the vertical angle value in a vertical list to the vertical_angle.xyz file.

            distance = lidar.getDistance() # Uses the lidar.getDistance function from the lidar-lite.py file from Garmin
            real_distance= distance*math.cos(math.radians(vertical_angle))*math.cos(math.radians(horizontal_angle1))
            print("Distance = %s" % (real_distance)) # Displays the real-time distance value in the Shell. Delete if it is not needed.
            sample_data1= open("distance.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
            with open("distance.xyz", "a") as f:
                f.write(str(real_distance)+ '\n') # Writes the distance value in a vertical list to the distance.xyz file.

        counterclockwise2 # Moving the second half step and recording data
        for i in range(1):
            for fullstep in range(1):
                for pin in range(4):
                    GPIO.output(control_pins[pin], counterclockwise2[fullstep][pin])
                time.sleep(.009)

            horizontal_angle2= horizontal_angle1-.0879 # Records the horizontal angle throughout the sweep. Goes from -15.5 to 15.5 degrees.
            print("horizontal angle = %s" % (horizontal_angle2)) # Shows the real-time horizontal angle value in the Shell. Delete if it is not needed.
            sample_data1= open("horizontal_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
            with open("horizontal_angle.xyz", "a") as f:
                f.write(str(horizontal_angle2)+ '\n') # Writes the angle value in a vertical list to the horizontal_angle.xyz file.

            vertical_angle= (30-j)*4*.1758 # Records the vertical angle. Goes from 21.1 to -21.1 degrees.
            print("vertical angle = %s" % (vertical_angle)) # Displays the real-time vertical angle value in the Shell. Delete if it is not needed.
            sample_data2= open("vertical_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
            with open("vertical_angle.xyz", "a") as f:
                f.write(str(vertical_angle)+ '\n') # Writes the vertical angle value in a vertical list to the vertical_angle.xyz file.

            distance = lidar.getDistance() # Uses the lidar.getDistance function from the lidar-lite.py file from Garmin
            real_distance= distance*math.cos(math.radians(vertical_angle))*math.cos(math.radians(horizontal_angle2))
            print("Distance = %s" % (real_distance)) # Displays the real-time distance value in the Shell. Delete if it is not needed.
            sample_data1= open("distance.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
            with open("distance.xyz", "a") as f:
                f.write(str(real_distance)+ '\n') # Writes the distance value in a vertical list to the distance.xyz file.

        counterclockwise3 # Moving the third half step and recording data
        for i in range(1):
            for fullstep in range(1):
                for pin in range(4):
                    GPIO.output(control_pins[pin], counterclockwise3[fullstep][pin])
                time.sleep(.009)

            horizontal_angle3= horizontal_angle2-.0879 # Records the horizontal angle throughout the sweep. Goes from -15.5 to 15.5 degrees.
            print("horizontal angle = %s" % (horizontal_angle3)) # Shows the real-time horizontal angle value in the Shell. Delete if it is not needed.
            sample_data1= open("horizontal_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
```

```python
        with open("horizontal_angle.xyz", "a") as f:
            f.write(str(horizontal_angle3)+ '\n') # Writes the angle value in a vertical list to the horizontal_angle.xyz file.

        vertical_angle= (30-j)*4*.1758 # Records the vertical angle. Goes from 21.1 to -21.1 degrees.
        print("vertical angle = %s" % (vertical_angle)) # Displays the real-time vertical angle value in the Shell. Delete if it is not needed.
        sample_data2= open("vertical_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
        with open("vertical_angle.xyz", "a") as f:
            f.write(str(vertical_angle)+ '\n') # Writes the vertical angle value in a vertical list to the vertical_angle.xyz file.

        distance = lidar.getDistance() # Uses the lidar.getDistance function from the lidar-lite.py file from Garmin
        real_distance= distance*math.cos(math.radians(vertical_angle))*math.cos(math.radians(horizontal_angle3))
        print("Distance = %s" % (real_distance)) # Displays the real-time distance value in the Shell. Delete if it is not needed.
        sample_data1= open("distance.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
        with open("distance.xyz", "a") as f:
            f.write(str(real_distance)+ '\n') # Writes the distance value in a vertical list to the distance.xyz file.

    counterclockwise4 # Moving the fourth half step and recording data
    for i in range(1):
        for fullstep in range(1):
            for pin in range(4):
                GPIO.output(control_pins[pin], counterclockwise4[fullstep][pin])
            time.sleep(.009)

        horizontal_angle4= horizontal_angle3-.0879 # Records the horizonal angle throughout the sweep. Goes from -15.5 to 15.5 degrees.
        print("horizontal angle = %s" % (horizontal_angle4)) # Shows the real-time horizontal angle value in the Shell. Delete if it is not needed.
        sample_data1= open("horizontal_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
        with open("horizontal_angle.xyz", "a") as f:
            f.write(str(horizontal_angle4)+ '\n') # Writes the angle value in a vertical list to the horizontal_angle.xyz file.

        vertical_angle= (30-j)*4*.1758 # Records the vertical angle. Goes from 21.1 to -21.1 degrees.
        print("vertical angle = %s" % (vertical_angle)) # Displays the real-time vertical angle value in the Shell. Delete if it is not needed.
        sample_data2= open("vertical_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
        with open("vertical_angle.xyz", "a") as f:
            f.write(str(vertical_angle)+ '\n') # Writes the vertical angle value in a vertical list to the vertical_angle.xyz file.

        distance = lidar.getDistance() # Uses the lidar.getDistance function from the lidar-lite.py file from Garmin
        real_distance= distance*math.cos(math.radians(vertical_angle))*math.cos(math.radians(horizontal_angle4))
        print("Distance = %s" % (real_distance)) # Displays the real-time distance value in the Shell. Delete if it is not needed.
        sample_data1= open("distance.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
        with open("distance.xyz", "a") as f:
            f.write(str(real_distance)+ '\n') # Writes the distance value in a vertical list to the distance.xyz file.

    counterclockwise5 # Moving the fifth half step and recording data
    for i in range(1):
        for fullstep in range(1):
            for pin in range(4):
                GPIO.output(control_pins[pin], counterclockwise3[fullstep][pin])
            time.sleep(.009)

        horizontal_angle5= horizontal_angle4-.0879 # Records the horizonal angle throughout the sweep. Goes from -15.5 to 15.5 degrees.
        print("horizontal angle = %s" % (horizontal_angle5)) # Shows the real-time horizontal angle value in the Shell. Delete if it is not needed.
        sample_data1= open("horizontal_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
        with open("horizontal_angle.xyz", "a") as f:
            f.write(str(horizontal_angle5)+ '\n') # Writes the angle value in a vertical list to the horizontal_angle.xyz file.

        vertical_angle= (30-j)*4*.1758 # Records the vertical angle. Goes from 21.1 to -21.1 degrees.
        print("vertical angle = %s" % (vertical_angle)) # Displays the real-time vertical angle value in the Shell. Delete if it is not needed.
        sample_data2= open("vertical_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
        with open("vertical_angle.xyz", "a") as f:
            f.write(str(vertical_angle)+ '\n') # Writes the vertical angle value in a vertical list to the vertical_angle.xyz file.

        distance = lidar.getDistance() # Uses the lidar.getDistance function from the lidar-lite.py file from Garmin
        real_distance= distance*math.cos(math.radians(vertical_angle))*math.cos(math.radians(horizontal_angle3))
        print("Distance = %s" % (real_distance)) # Displays the real-time distance value in the Shell. Delete if it is not needed.
        sample_data1= open("distance.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
        with open("distance.xyz", "a") as f:
            f.write(str(real_distance)+ '\n') # Writes the distance value in a vertical list to the distance.xyz file.

    counterclockwise6 # Moving the sixth half step and recording data
    for i in range(1):
        for fullstep in range(1):
            for pin in range(4):
                GPIO.output(control_pins[pin], counterclockwise6[fullstep][pin])
            time.sleep(.009)

        horizontal_angle6= horizontal_angle5-.0879 # Records the horizonal angle throughout the sweep. Goes from -15.5 to 15.5 degrees.
        print("horizontal angle = %s" % (horizontal_angle6)) # Shows the real-time horizontal angle value in the Shell. Delete if it is not needed.
        sample_data1= open("horizontal_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
        with open("horizontal_angle.xyz", "a") as f:
            f.write(str(horizontal_angle6)+ '\n') # Writes the angle value in a vertical list to the horizontal_angle.xyz file.

        vertical_angle= (30-j)*4*.1758 # Records the vertical angle. Goes from 21.1 to -21.1 degrees.
        print("vertical angle = %s" % (vertical_angle)) # Displays the real-time vertical angle value in the Shell. Delete if it is not needed.
        sample_data2= open("vertical_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
        with open("vertical_angle.xyz", "a") as f:
            f.write(str(vertical_angle)+ '\n') # Writes the vertical angle value in a vertical list to the vertical_angle.xyz file.

        distance = lidar.getDistance() # Uses the lidar.getDistance function from the lidar-lite.py file from Garmin
        real_distance= distance*math.cos(math.radians(vertical_angle))*math.cos(math.radians(horizontal_angle3))
        print("Distance = %s" % (real_distance)) # Displays the real-time distance value in the Shell. Delete if it is not needed.
```

```
            sample_data1= open("distance.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
            with open("distance.xyz", "a") as f:
                f.write(str(real_distance)+ '\n') # Writes the distance value in a vertical list to the distance.xyz file.

        counterclockwise7 # Moving the seventh half step and recording data
        for i in range(1):
            for fullstep in range(1):
                for pin in range(4):
                    GPIO.output(control_pins[pin], counterclockwise7[fullstep][pin])
                time.sleep(.009)

            horizontal_angle7= horizontal_angle6-.0879 # Records the horizonal angle throughout the sweep. Goes from -15.5 to 15.5 degrees.
            print("horizontal angle = %s" % (horizontal_angle7)) # Shows the real-time horizontal angle value in the Shell. Delete if it is not needed.
            sample_data1= open("horizontal_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
            with open("horizontal_angle.xyz", "a") as f:
                f.write(str(horizontal_angle7)+ '\n') # Writes the angle value in a vertical list to the horizontal_angle.xyz file.

            vertical_angle= (30-j)*4*.1758 # Records the vertical angle. Goes from 21.1 to -21.1 degrees.
            print("vertical angle = %s" % (vertical_angle)) # Displays the real-time vertical angle value in the Shell. Delete if it is not needed.
            sample_data2= open("vertical_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
            with open("vertical_angle.xyz", "a") as f:
                f.write(str(vertical_angle)+ '\n') # Writes the vertical angle value in a vertical list to the vertical_angle.xyz file.

            distance = lidar.getDistance() # Uses the lidar.getDistance function from the lidar-lite.py file from Garmin
            real_distance= distance*math.cos(math.radians(vertical_angle))*math.cos(math.radians(horizontal_angle3))
            print("Distance = %s" % (real_distance)) # Displays the real-time distance value in the Shell. Delete if it is not needed.
            sample_data1= open("distance.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
            with open("distance.xyz", "a") as f:
                f.write(str(real_distance)+ '\n') # Writes the distance value in a vertical list to the distance.xyz file.

        counterclockwise8 # Moving the eighth half step and recording data
        for i in range(1):
            for fullstep in range(1):
                for pin in range(4):
                    GPIO.output(control_pins[pin], counterclockwise8[fullstep][pin])
                time.sleep(.009)

            horizontal_angle8= horizontal_angle7-.0879 # Records the horizonal angle throughout the sweep. Goes from -15.5 to 15.5 degrees.
            print("horizontal angle = %s" % (horizontal_angle8)) # Shows the real-time horizontal angle value in the Shell. Delete if it is not needed.
            sample_data1= open("horizontal_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
            with open("horizontal_angle.xyz", "a") as f:
                f.write(str(horizontal_angle8)+ '\n') # Writes the angle value in a vertical list to the horizontal_angle.xyz file.

            vertical_angle= (30-j)*4*.1758 # Records the vertical angle. Goes from 21.1 to -21.1 degrees.
            print("vertical angle = %s" % (vertical_angle)) # Displays the real-time vertical angle value in the Shell. Delete if it is not needed.
            sample_data2= open("vertical_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
            sample_data2= open("vertical_angle.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
            with open("vertical_angle.xyz", "a") as f:
                f.write(str(vertical_angle)+ '\n') # Writes the vertical angle value in a vertical list to the vertical_angle.xyz file.

            distance = lidar.getDistance() # Uses the lidar.getDistance function from the lidar-lite.py file from Garmin
            real_distance= distance*math.cos(math.radians(vertical_angle))*math.cos(math.radians(horizontal_angle3))
            print("Distance = %s" % (real_distance)) # Displays the real-time distance value in the Shell. Delete if it is not needed.
            sample_data1= open("distance.xyz", "a") # Opens the blank .xyz file. The "a" appends the value to the end of the file.
            with open("distance.xyz", "a") as f:
                f.write(str(real_distance)+ '\n') # Writes the distance value in a vertical list to the distance.xyz file.

    # Moving the motor clockwise back to the starting position. No data is recorded
    control_pins = [12,16,18,22] #horizontal motor
    for pin in control_pins:
        GPIO.setup(pin, GPIO.OUT)
        GPIO.output(pin, 0)
    for k in range(1,46): # Using the fullstepping configuration, it takes 44 steps to move 31 degrees
        clockwise
        for i in range(1): # Moves 1 step (.7 degrees) a total of 44 times
            for halfstep in range(8):
                for pin in range(4):
                  GPIO.output(control_pins[pin], clockwise[halfstep][pin])
                time.sleep(.005)

    # Moves the vertical motor down .7 degrees going from +21.1 to -21.1
    control_pins = [29,31,32,33] #vertical motor
    for pin in control_pins:
        GPIO.setup(pin, GPIO.OUT)
        GPIO.output(pin, 0)
    counterclockwise
    for i in range(1): # Moves 1 step (.7 degrees) a total of 60 times (42.2 degrees)
        for halfstep in range(8):
            for pin in range(4):
                GPIO.output(control_pins[pin], counterclockwise[halfstep][pin])
            time.sleep(.009)

# This section converts the distance, vertical, and horizontal .xyz files into entry strips to be combined into one file
with open('distance.xyz') as file1, open('horizontal_angle.xyz') as file2, open('vertical_angle.xyz') as file3:
    content1= [entry.strip() for entry in file1]
    content2= [entry.strip() for entry in file2]
    content3= [entry.strip() for entry in file3]

# Produces a .xyz file with three columns:  distance, horizontal angle, vertical angle.
with open('output.xyz', 'w') as file:
    for entry1, entry2, entry3 in zip(content1, content2, content3):
        file.write(f'{entry1} {entry2} {entry3}\n')

# Moving the horizontal motor from its initial position to its starting position (15.5 degrees to the right).
control_pins = [12,16,18,22] #Pins the horizontal motor is connected into the rapsberry pi.
for pin in control_pins:
  GPIO.setup(pin, GPIO.OUT)
  GPIO.output(pin, 0)

counterclockwise # Tells the motor which way to rotate (this variable was declared at the beginning of code).
for i in range(22): #The degrees you want the motor to rotate (512= 360 degrees, 22= 15.5 degrees).
  for halfstep in range(8):
    for pin in range(4):
      GPIO.output(control_pins[pin], counterclockwise[halfstep][pin])
    time.sleep(.005)

# Moving the vertical motor from its initial position to its starting position (21.1 degrees up).
control_pins = [29,31,32,33] #Pins the vertical motor is connected into the rapsberry pi.
for pin in control_pins:
      GPIO.setup(pin, GPIO.OUT)
      GPIO.output(pin, 0)

clockwise # Tells the motor which way to rotate (this variable was declared at the beginning of code).
for i in range(30): #The degrees you want the motor to rotate (512= 360 degrees, 30= 21.1 degrees).
    for halfstep in range(8):
        for pin in range(4):
          GPIO.output(control_pins[pin], clockwise[halfstep][pin])
        time.sleep(.005)
GPIO.cleanup()

# Stops the timer and converts from seconds to minutes
stop = timeit.default_timer()
print("Execution time: %s minutes" % ((stop-start)/60))
```

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

APPENDIX A-4

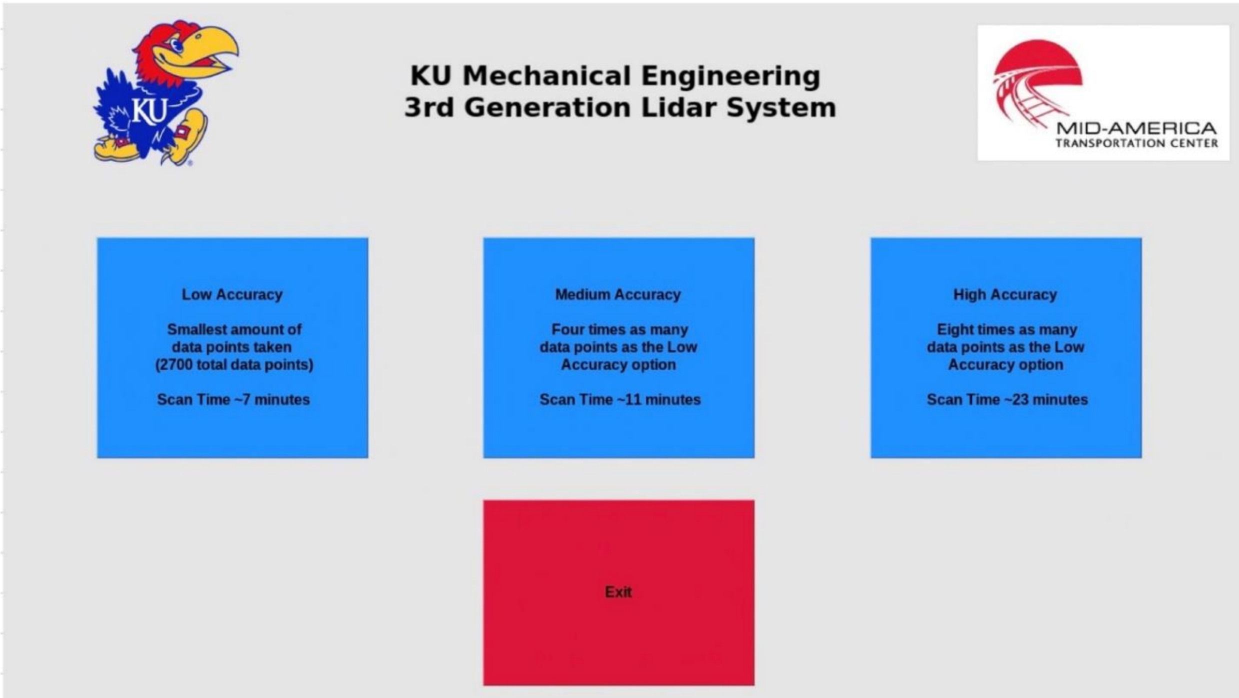

FIGURE A-4-1: SCREEN SHOT OF GRAPHICAL USER INTERFACE

```python
from tkinter import *
from Low import runCode
from Medium import runCode2
from High import runCode3
from PIL import Image, ImageTk
import lidar_lite

root = Tk()
root.wm_title("GUI")
root.configure(bg="#E5E5E5")

screen_width = root.winfo_screenwidth()
screen_height = root.winfo_screenheight()

root.attributes("-fullscreen", True)

def LowRes():
    runCode()

def btnExit():
        root.destroy()

def end_fullscreen(event):
        root.attributes("-fullscreen", False)

def MedRes():
    runCode2()

def HighRes():
    runCode3()

load= Image.open("jayhawk.png")
load = load.resize((round(screen_width*0.12),round(screen_width*0.12)), Image.ANTIALIAS)
render = ImageTk.PhotoImage(load)
img1 = Label(root, image=render, bg="#E5E5E5")
load2= Image.open("MATCimage.jpg")
load2 = load2.resize((round(screen_width*0.2025),round(screen_width*0.108)), Image.ANTIALIAS)
render2 = ImageTk.PhotoImage(load2)
img2 = Label(root, image=render2)



label_1 = Label(root, text="KU Mechanical Engineering\n 3rd Generation Lidar System", font="Verdana 20 bold",
                           fg="#000",
                           bg="#E5E5E5",
                           pady = round(screen_width*0.01),
                           padx = round(screen_height*0.01), justify='center')

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 = "#1E90FF",
                command=LowRes, height = round(screen_height*0.015), width=round(screen_width*0.022), font = "Arial 12 bold", justify='center', wraplength=180)

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 = "#1E90FF",
               command=MedRes, height = round(screen_height*0.015), width=round(screen_width*0.022), font = "Arial 12 bold", justify='center', wraplength=180)

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 = "#1E90FF",
                command=HighRes, height = round(screen_height*0.015), width=round(screen_width*0.022), font = "Arial 12 bold", justify='center', wraplength=180)

exitButton = Button(root, text="Exit", background = "#DC143C",
                command=btnExit, height = round(screen_height*0.0125), width=round(screen_width*0.022), font = "Arial 12 bold", justify='center', wraplength=300)


img1.place(x=round(screen_width*0.07), y=round(screen_height*0.02))
img2.place(x=round(screen_width*0.78), y=round(screen_height*0.03))
label_1.place(x=round(screen_width*0.20), y=round(screen_height*0.03))
lowButton.place(x=round(screen_width*0.074), y=round(screen_height*0.33))
mediumButton.place(x=round(screen_width*0.384), y=round(screen_height*0.33))
highButton.place(x=round(screen_width*0.694), y=round(screen_height*0.33))
exitButton.place(x=round(screen_width*0.384), y=round(screen_height*0.7))


root.bind("<Escape>", end_fullscreen)
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