ABSTRACT

A collision avoidance system is designed by expertise to reduce the severity of an accident especially in cars. Nowadays, as reported by World Health Organization (WHO), about 1.25 million people die every year due to road traffic crashes. If there are no corrective steps taken, the number is expected to increase. Most of the collision avoidance system is utilizing sensors such as radar, camera and laser range finder.

For this project, a single laser range finder (LRF) and a single embedded platform, called Raspberry Pi (also known as Raspi) are used for the project development. The function of LRF sensor is to determine the distance between any objects with the sensor as the reference point based on the data obtained from it. Meanwhile, the Raspi platform will work as a tool to do the computational task when the LRF sends the data obtained in real-time and display the results to the user.

To perform the detection and tracking using LRF, single scan from it is taken at each time step. The obtained scan will go through few steps of process to obtain the position of the target object. The flow of tracking process consists of background subtraction, clustering and data association, and tracking.

ACKNOWLEDGEMENT

I would like to address my utmost gratitude to my supervisor, Dr Patrick Sebastian, without whom this project would not have seen the light of the day. Since the project was a mere idea, his support and guidance have always shed the light on the path to be taken, and helped me to overcome the difficulties and obstacles I came to face.

I am also forever thankful to my parents for their love and support, and my family and friends for always being by my side. Finally, I would also like to extend my warmest regards to Universiti Teknologi PETRONAS (UTP) for providing me with all the tools and help I needed during this project.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Background of Study

Cars crashes are not the new issues nowadays. It happens in every place around the world. A research has been done by a non-profit organization regarding the car crashes phenomenon. From their findings, according to the World Health Organization (WHO), the most leading cause of death among young people aged 15 to 29 years old is road traffic injuries. In addition to the facts, as the results of car crashes about 1.25 million people die annually.

The most top causes of car crashes are driver distraction. Distraction driving can be defined as form of activities that could divert a person’s attention away from the main task of driving which including texting or using smartphones, and talking to the passengers. [3] Hence, align with significant increases of car crashes cases due to distracted driving by the driver, there is a needs to develop a system that could detects the obstacles such cars, pedestrians and motorcyclists on the road thus alarming the driver with any possible threats.

In this project, the LRF was used to develop obstacles avoidance system. LRF use laser beams to measure the distance between the obstacles and its current position, for the experiment purpose and indoor used, a range of 4 meters of LRF was used.[4] The working principle of LRF was easy to understand. As it projects the laser beam to the obstacles, the time taken for the laser beam to reflected back by the object towards the receiver is used to measure the distance. [5]

For the embedded platform, Raspberry Pi (Raspi) platform was chosen in this project. It is popular among the people as small in size, low power consumption, low prices and most suitable for complex projects as compared to Arduino platform. [6]

1.2 Problem Statement

According to the World Health Organization (WHO), the road accidents majorly from car crashes are expected to increase by year of 2030 and become one of the top seventh causes of death globally. [1] This causes by driver's inattention and distraction during driving.

The current pre-crash alerting systems provide great help to drivers to avoid any possible events of car crashes and car collisions. Most of these systems depend on sensors and LRF is one of them. In fact, LRF is relatively low price but it has great accuracy for the measurement which in turns become as an alternative for low-budget and affordable collision avoidance system. [7]

1.3 Objectives and Scope of study

The objectives and scope of study for this project are as follows:

1.3.1 Objectives

1- To develop algorithm for detection and tracking multiple targets by using laser range finder.

2- To visualize tracked objects data on screen display.

1.3.2 Scope of Study

In order to run the project, Raspberry Pi platform was used. The platform run uses Jessie, a Debian based Linux operating systems. [8] The laser range finder, Hokuyo URG-04LX-UG01, was used to scan and measure the distance of object to the sensor device. [9] The data gather from LRF was used to process and display on the monitor screen. From the data obtained, the user will be notified the distance between the object and a warning message to alert the user if the upcoming objects are approaching closer.

CHAPTER 2

LITERATURE REVIEW

Throughout this chapter, a literature review about the project will be presented.

2.1 Characteristics of Laser Range Finder

Laser rangefinder (LRF) is a device that is used to measure the depth or the distance to the projected object with a default distances. The fundamental working principal of LRF is almost similar to radar and solar. But, the only difference is LRF use laser instead of electromagnetic waves.

It emits the laser beam and then measures the time taken for beam to fly and return back. In addition, since the speed of the beam is already predefined, thus the distance to the obstacles can be measured. [5]

The good side of LRF is it can detect object or obstacles for longer distance as compared to radar. This advantage makes it superior to be used as obstacle detection system in cars. However, it also has some drawbacks for instance: limited view range and sometimes in the scan area there are holes due to some part of beams not being reflected. [5]

Furthermore, it also needs some filtration for the bad data returned beams in order to get right distance measurement of object ahead. [5]

From Figure 1 show the measurement range in degree taken by a LRF.

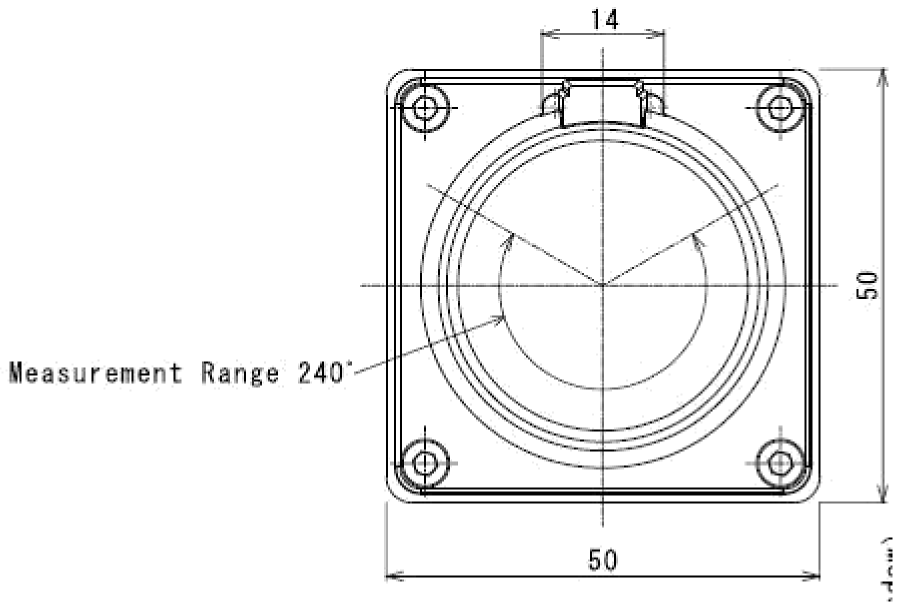


FIGURE 1 Measurement range of LRF.

The LRF starts the scanning process in anticlockwise direction. The area zone which out of 240 degree scanning boundary consider as dead zone. No data will collect and receive by the LRF.

In Figure 2 shows sensor's measurement details. Measurements points in LRF are called Steps. Each section has different value measurement point. [10]

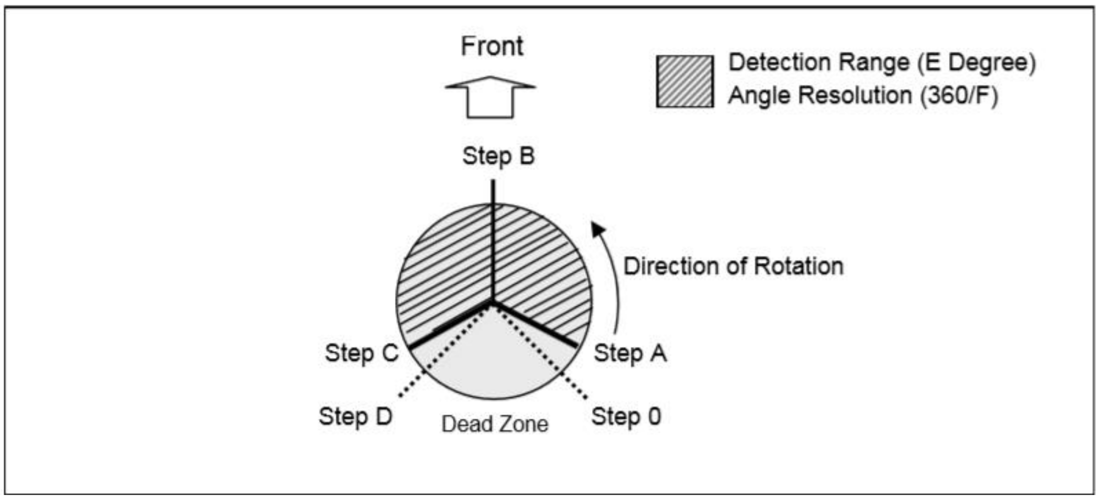


FIGURE 2 Measurement parameters.

Table 1 shows the measurement parameters of URG-04LX-01 which is the sensor’s model used in the project.

TABLE 1 Measurement parameters of URG-40LX-01

Parameter	Description	Value
Step 0	1st measurement point	0
Step A	Initial measurement step of detection range	44
Step B	Sensor front step	384
Step C	End point of detection range	725
Step D	Last measurement point	768
E	Detection range	239.77 degree
F	Slit division	1024

2.2 Time of Flight (TOF) and Speed of Objects

On this project, LIDAR (light detection and ranging) is used as optical remote sensing technology. The basic principle of LIDAR is infrared laser pulses will be send out by emitter and the photodiode will search and detect the reflected light. Meanwhile, the distance of the objects or targets are determine based on time of flight (TOF) of infrared laser pulses emitted and reflected light back to the receiver. For speed of the other objects, it can be done by comparing the relativity speed of objects with the speed of light. [11]
There are certain things too need to be taking account about measurement of the object's distance. They are including the peak power of the laser, the laser beam divergence, and target reflectivity, sensitivity of the detector as well as optics and air transmittance. On top of that, the pulse width of the laser, speed and accuracy of Analog-to-Digital Converter (ADC) used will affect the accuracy of TOF measurement too. [12]

2.3 Detection of Multiple Targets

The Kalman filtering method is a handy tool for detection and tracking of both moving and stationary objects in a forward- facing laser scan. [13] This method is the most common techniques of object tracking. As for example, [15] , [16] and [17] all use set of independent Kalman filter variants to recognize the location of moving targets.

The work of [15] use white noise acceleration velocity models meanwhile [16] modelled the moving objects with constant linear acceleration and constant rational velocity. By referring to the detection and tracking of moving objects (DATMO) system [14], it breaks down into three modules which are scan segmentation, object classification based on suitable voting scheme of objects properties and object tracking using Kalman filter for better tracking performance in the system.

2.4 Raspberry Pi versus Arduino platform

Arduino and Raspberry Pi are popular among the inventors and IT hobbyist. Both platforms have one common similarity which is small in size. However, they have differences. An Arduino is a microcontroller and easy to use. A Raspberry Pi is more complicated than Arduino due to its ability to execute multiple programs since it is a general-purpose computer. Hence, when it comes to do multiple tasks or full-fledged computer, Raspberry Pi is the best suit.

In Table 2, the general comparison between Raspberry Pi platform and Arduino platform in term of theirs hardware characteristics, software and price are presented.

TABLE 2 Comparison between Arduino Uno and Raspberry Pi 2B+

	Devices	
Platform	Arduino	Raspberry Pi
Variant	Uno	Model 2 B+
Software		
Operating system	-	Linux, RISC OS
Dev.Environments /Toolkits	Arduino IDE, Eclipse	Eclipse, QEMU, Scratchbox, OpenEmbedded
Programming language	Wiring-based (C++)	Python, C
Architecture	8bit	32 bit
Hardware		
Processor	ATMEGA328	BCM2835(ARM) Corttex - A7
Speed	16Mhz	900Mhz
RAM	2Kbyte	1GB
ROM	32Kbyte	SD
I/O (various protocols)	14	8
ADC	6	Internally used
USB	-	4 x 2.0
Audio	-	Stereo out, In w/ USB mic
Video	-	HDMI, NSTC or PAL
Misc.	Many shields available for added capability	SD, 10/100 Ethernet, JTAG
Cost		
*(*** 1USD = MYR 4.41***)*	$29.95 @ MYR131.93	$35.00 @ MYR154.18

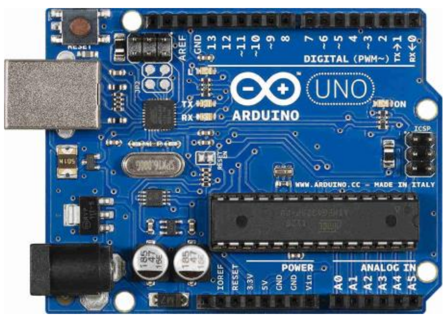


FIGURE 3 Arduino Uno

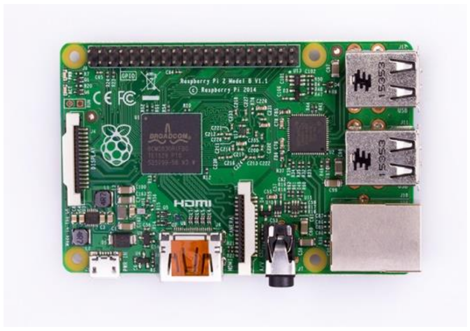


FIGURE 4 Raspberry Pi 2 B+

CHAPTER 3

METHODOLOGY

On Chapter 3, it will present the methodology that was used and show the general steps needed to accomplish the project goals.

3.1 Overall methodology

The summary of project methodology can be summarized as below:

a) **Problem Identification**

   Research and literature review been done to achieve better understanding on the project requirements. Furthermore, some deep analysis also been done on other projects that have done similar works.

b) **Software and Tools Selection**

   The selection of tools and software are based on the input obtained from the first stage.

c) **Development Phase**

   At this stage, the Laser Range Finder is interfacing with Raspberry Pi platform. The goal is to obtain distance measurement data.

d) **Testing and Optimizing the Performance of the System**

   In this final step, the system was tested to verify the results obtained. The main parameter that been consider is in term of validity of data transferred.

3.2 Project Flow Chart

Step 01: Data Acquisition

The Laser Range Finder (LRF) starts capturing the data from the environment.

Step 02: Data Processing

As the LRF capturing the data, the data obtained will be filtered to avoid false signal. Therefore, only truth data will be used and thus reduced the data errors.

Step 03: Classification of Possible Collision

The category of collisions was specified based on the reading obtain from LRF.

Step 04: Prompt notification to driver

The results are displayed to the driver. On the monitor screen, the distance of the objects from car was shown and if the objects are closer approaching the car, warning message will notified to alert the driver.

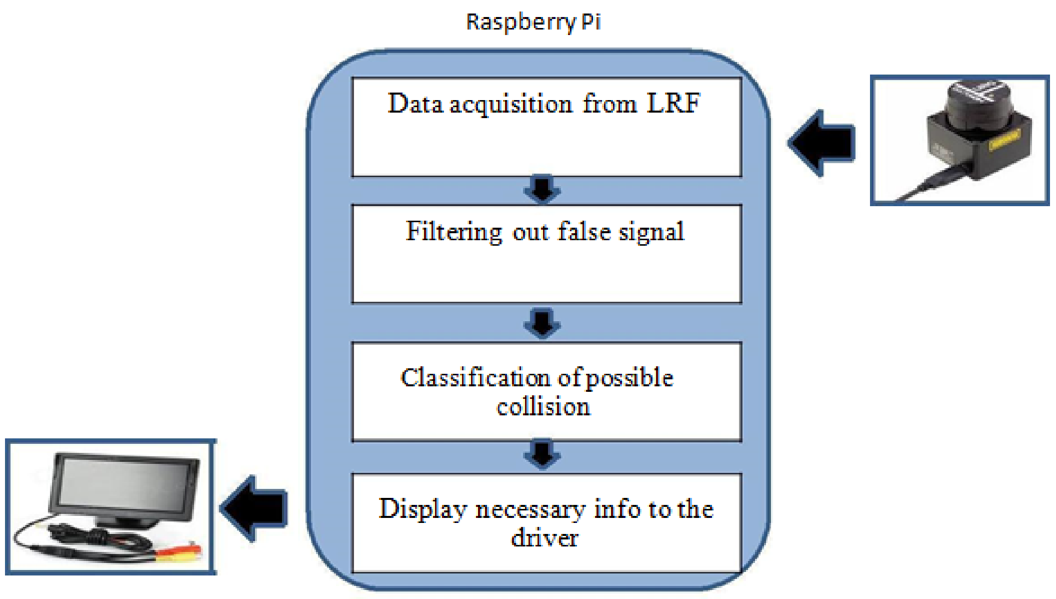

Raspberry Pi
Data acquisition from LRF
Filtering out false signal
Classification of possible collision
Display necessary info to the driver

FIGURE 5 Program Flow

3.3 Tools and Software

Below are list of tools and software used during project development period:

TABLE 3 Equipments used for the project

Bil	Name of equipment	Quantity (Unit)
1	Raspberry Pi B+ Model Board	1
2	Hokuyo URG Laser Range Finder	1
3	URG C Library	-
4	4.3” NTSC TFT LCD Color Monitor	1

3.4 Key Milestones

Below are the summary of some important milestones for project’s development:

- ✓ Both Raspberry Pi platform and LRF are interface and acquire the necessary measurement data.
- ✓ Development of GUI algorithm code to display the data towards driver on screen monitor.

3.5 Gant Chart

The following tables are the timeline for the project development period in final year project 1 (FYP1) and final year project 2 (FYP2) respectively.

TABLE 4 Gantt chart for FYP1

Task /Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Task 1	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
Task 2	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦						
Task 3					🟦	🟦	🟦	🟦						
Task 4					🟦	🟦	🟦	🟦						
Task 5				🟦	🟦	🟦								
Task 6						●								
Task 7							🟦	🟦						
Task 8									●					
Task 9										🟦	🟦	🟦	🟦	
Task 10										🟦	🟦	🟦	🟦	●

Note: The symbol ● indicates the important milestones.

Tasks:

- ✓ **Task 01:** Literature review and preliminary research.
- ✓ **Task 02:** Identify the software and tools required for the project work.
- ✓ **Task 03:** Setup Raspberry Pi and testing simple program code on it.
- ✓ **Task 04:** Configure and setup Laser Rangefinder on PC.
- ✓ **Task 05:** Write extended proposal.
- ✓ **Task 06:** Submission of extended proposal.
- ✓ **Task 07:** Preparation for proposal defence presentation.
- ✓ **Task 08:** Proposal defence presentation.
- ✓ **Task 09:** Write Interim report draft.
- ✓ **Task 10:** Submission for Interim report.

TABLE 5 Gantt chart for FYP2

Task /Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Task 01	🟦	🟦	🟦												
Task 02				🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦			
Task 03										●					
Task 04				🟦	🟦	🟦	🟦	●							
Task 05													●		
Task 06														●	
Task 07														●	
Task 08												🟦	🟦	🟦	
Task 09															●

Tasks:

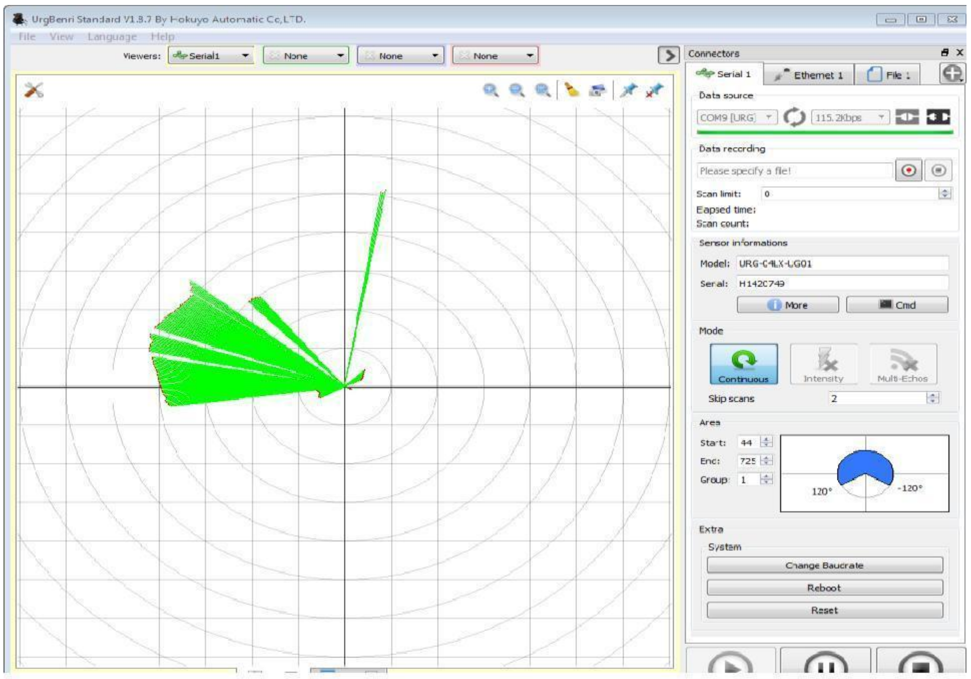
- ✓ **Task 01:** Setting up the interface LRF and Raspberry Pi.
- ✓ **Task 02:** Testing and benchmarking the integration of the system.
- ✓ **Task 03:** Poster presentation for Pre-Sedex.
- ✓ **Task 04:** Progress report s' preparation and submission.
- ✓ **Task 05:** Submission of draft final report.
- ✓ **Task 06:** Submission of dissertation.
- ✓ **Task 07:** Submission of technical paper.
- ✓ **Task 08:** Preparation for final viva.
- ✓ **Task 09:** Viva.

CHAPTER 4
RESULTS AND DISCUSSION

4.1 Laser Range Finder (LRF) interface with Raspberry Pi

LRF was interface with the Raspberry Pi to obtain the reading data. It scan environment with coverage of 240 degree angle. In addition, LRF gives 682 readings with angle resolution 0.36 degree. It also can cover distance up to 4000mm which equivalent to 4.0 meters.

To ensure the sensor is well function, it was tested using the Hokuyo URG viewer application in windows platform. Once it confirmed to well-functioning, the LRF will connect with Raspberry Pi. Figure 6 show the output obtained when a LRF start scanning process.


UrgBenri Standard V1.8.7 By Hokuyo Automatic Co,LTD.

Viewers: Serial1, None, None, None

Data source: COM9 [URG], 115.2Kbps

Data recording: Please specify a file!

Scan limit: 0

Eapsed time:

Scan count:

Sensor informations

Model: URG-C4LX-LG01

Serial: H142C749

Mode: Continuous

Skip scans: 2

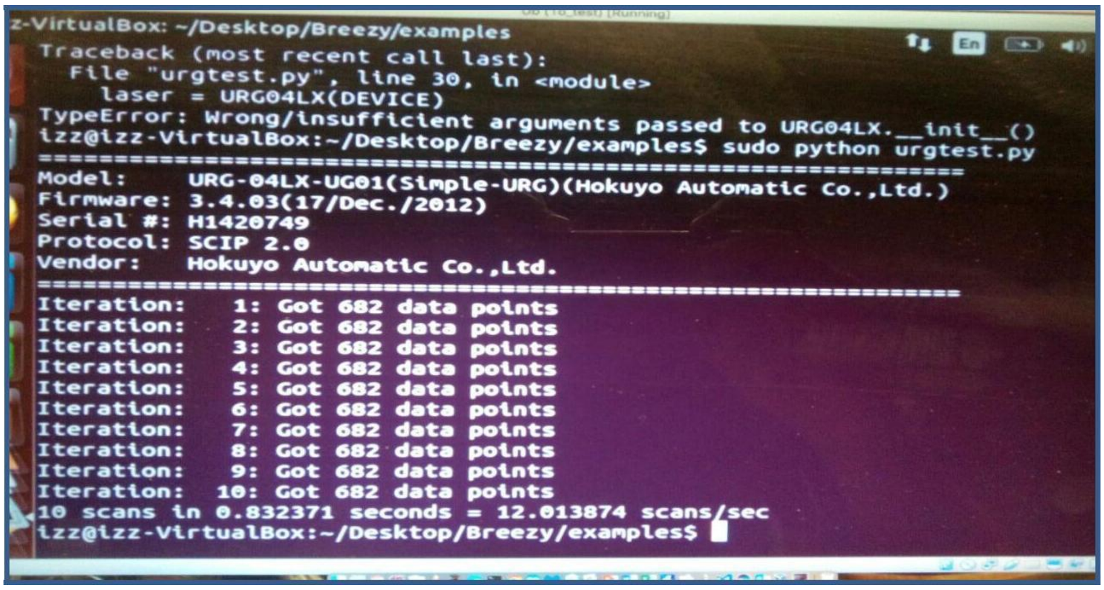
Area

Start: 44

End: 725

Group: 1

FIGURE 6 LRF Testing On URG



```
z-VirtualBox: ~/Desktop/Breezy/examples
  Traceback (most recent call last):
    File "urgtest.py", line 30, in <module>
      laser = URG04LX(DEVICE)
  TypeError: Wrong/insufficient arguments passed to URG04LX.__init__()
  izz@izz-VirtualBox:~/Desktop/Breezy/examples$ sudo python urgtest.py
  ==============================================================
  Model:     URG-04LX-UG01(Simple-URG)(Hokuyo Automatic Co.,Ltd.)
  Firmware:  3.4.03(17/Dec./2012)
  Serial #:  H1420749
  Protocol:  SCIP 2.0
  Vendor:    Hokuyo Automatic Co.,Ltd.
  ==============================================================
  Iteration:   1: Got 682 data points
  Iteration:   2: Got 682 data points
  Iteration:   3: Got 682 data points
  Iteration:   4: Got 682 data points
  Iteration:   5: Got 682 data points
  Iteration:   6: Got 682 data points
  Iteration:   7: Got 682 data points
  Iteration:   8: Got 682 data points
  Iteration:   9: Got 682 data points
  Iteration:  10: Got 682 data points
  10 scans in 0.832371 seconds = 12.013874 scans/sec
  izz@izz-VirtualBox:~/Desktop/Breezy/examples$
```

FIGURE 7   Data Points Scanning Checking From LRF

From Figure 7 above, when a LRF was connected with a Raspberry Pi, a check and balance procedure been done to ensure during real-time scanning, the LRF got maximum amount of data point. In this case, 10 iterations was done in the algorithm code and in each iteration got maximum of 682 data points. Those data points would be used to plot the data in radar like form. Also been noted, the LRF can do approximately 12 scans in 1 second after 10 iterations.

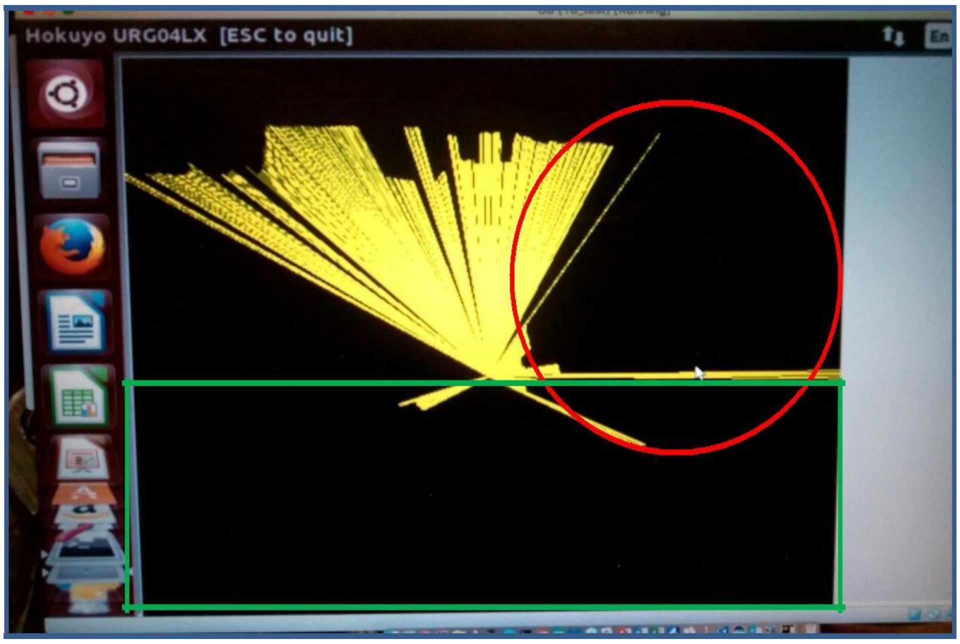


FIGURE 8 Plotted Data Points from LRF

From Figure 8 , noticed that some part of data missing during the real-time scanning. The yellow lines indicate that, no objects present in range of detection. The big black hole (in red circle) indicate that object present near to the LRF sensor. The green rectangle shape in the Figure 8 indicate that , the dead zone of measurement range. The full algorithm can be refer in Appendix section under file name *urgplot.py*

CHAPTER 5
CONCLUSION AND RECOMMENDATION

5.1 Conclusion

This paper proposed a method of detecting and tracking multiple targets by using laser rangefinder. Few challenges arose as to complete the project. Those problems are false data or missing data during the scanning and data retrieved by the LRF and the accuracy to locate the object in real-time.

Besides that, with proper construction of laser range algorithm and visualization algorithm for multiple target tracking in Raspberry Pi microcontroller, it can be a very powerful device to assist especially driver in vehicle by alerting them with the upcoming objects or obstacles in front of them.

5.2 Recommendation

As for future works, it can be improving more on processing data speed of device during real time and taking into account the ways to make detection and tracking scheme efficient during bad weather conditions.

REFERENCES

[1] World Health Organization. (2015, October): Road Traffic Injuries [Online]. Available: http://www/who.int/mediacentre/factsheets/fs358/en/

[2] Law Offices of Michael Pines. (2015) Top 25 Causes of car Accidents [Online]. Available: https://seriousaccidents.com/legal-advice/top-causes-of-car-accidents/

[3] National Highway Traffic Safety Administration. (2015) U Drive U Text U Pay [Online]. Available: http://www.distraction.gov/stats-research-laws/facts-and-statistics.html

[4] Y.Saito, T.Azumiy, N.Nishio and S.Kato, "Sensors Fusion of a Camera and a LRF on GPU for obstacle avoidance".

[5] S.J. Cacciola, "Fusion of Laser Range Finding and Computer Vision Data," 2007.

[6] "Comparison between Arduino and Raspberry Pi," [Online]. Available: http://circuitdigest.com/article/arduino-vs-raspberryp-pi-difference-between-the-two [Accessed 20 July 2016]

[7] BR^|^Scaron;CIC, Drazen, Takeshi SASAKI, and Hideki HASHIMOTO. "Tracking Of Humans And Robots Using Laser Range Finders". SICE Journal of Control, Measurement, and System Integration 1.5 (2008): 383-392. Web.

[8] Wikipedia.org, "Raspberry_pi," [Online]. Available : https://en.wikipedia.org/wiki/Raspberry_Pi [Accessed 29 July 2016]

[9] Wikipedia.org, "Laser RangeFinder," [Online]. Available : https://en.wikipedia.org/wiki/Laser_rangefinder [Accessed 29 July 2016]

[10] Communication Protocol Specification For SCIP2.0 Standard. 1st ed. Hokuyo Automatic Co, Ltd, 2017. Print..

[11] Texas Instrument. (2011). LIDAR System Design for Automotive/Industrial/Military Applications [Online]. Available : www.ti.com/lit/pdf/snaa123

[12] First Sensor. (2012, January). LIDAR For Automotive Applications [Online]. Available: www.sensortest.de/ausstellerbereich/upload/.../12021_Lidar_for_automotive_12.pd

[13] A. Mendes, L. Bento, and U. Nunes, "Multi-target detection and tracking with a laser scanner," IEEE Intelligent Vehicles Symposium, 2004

[14] M. Burke.(2016 , Feb). Laser-Based Target Tracking using Principal Component Descriptors.[Online]. Available: https://www.researchgate.net/publication

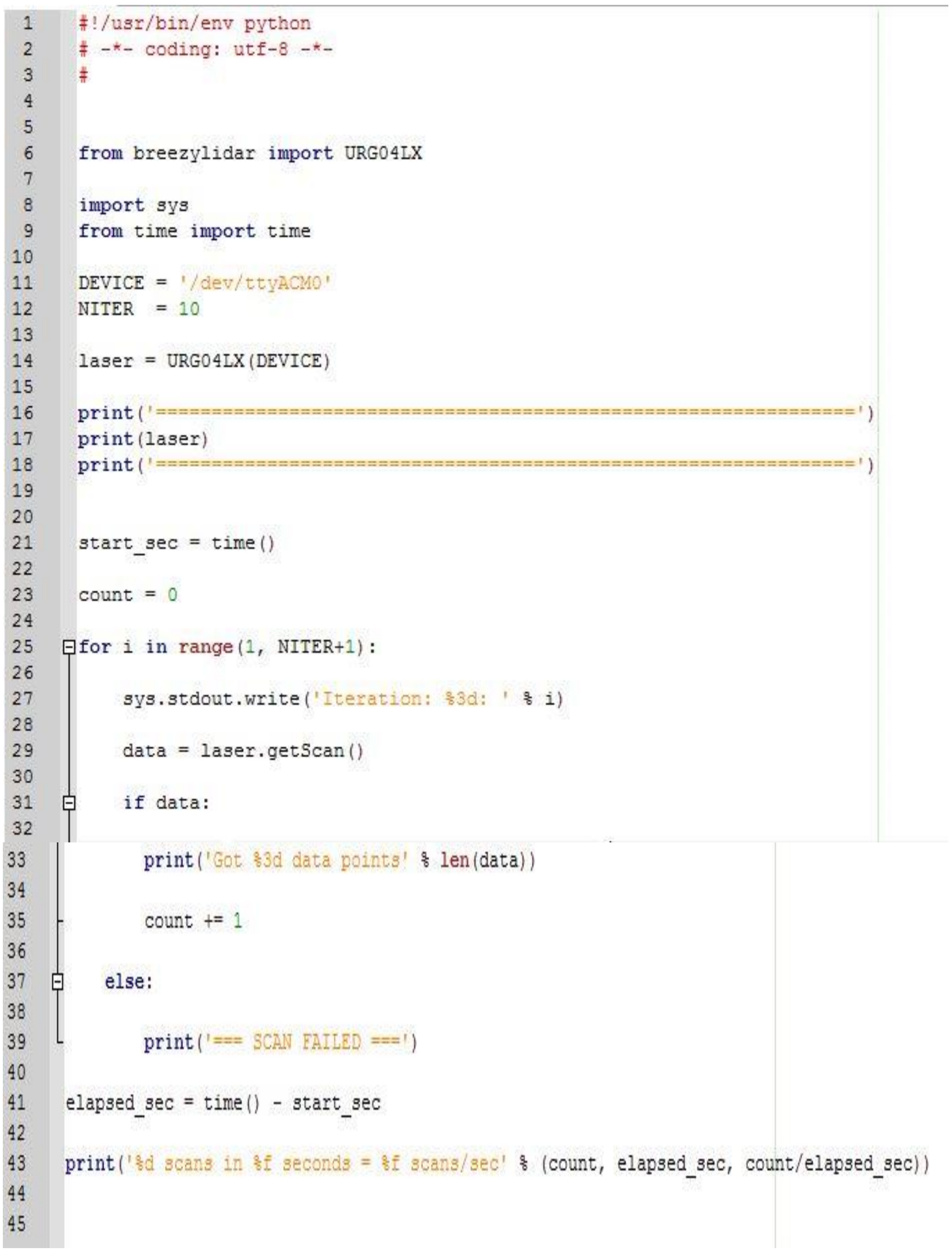
[15] A. Mendes, L. Bento, and U. Nunes, “Multi-target detection and tracking with a laser scanner,” in Intelligent Vehicles Symposium, 2004 IEEE, 17 2004, pp. 796 – 801

[16] A. Mendes and U. Nunes, “Situation-based multi-target detection and tracking with laser scanner in outdoor semi-structured environment,” in Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on, vol. 1, 28 2004, pp. 88 - 93.

[17] R. MacLachlan and C. Mertz, “Tracking of moving objects from a moving vehicle using a scanning laser rangefinder,” in Intelligent Transportation Systems Conference, 2006. ITSC ’06. IEEE, 17-20 2006, pp. 301 - 306

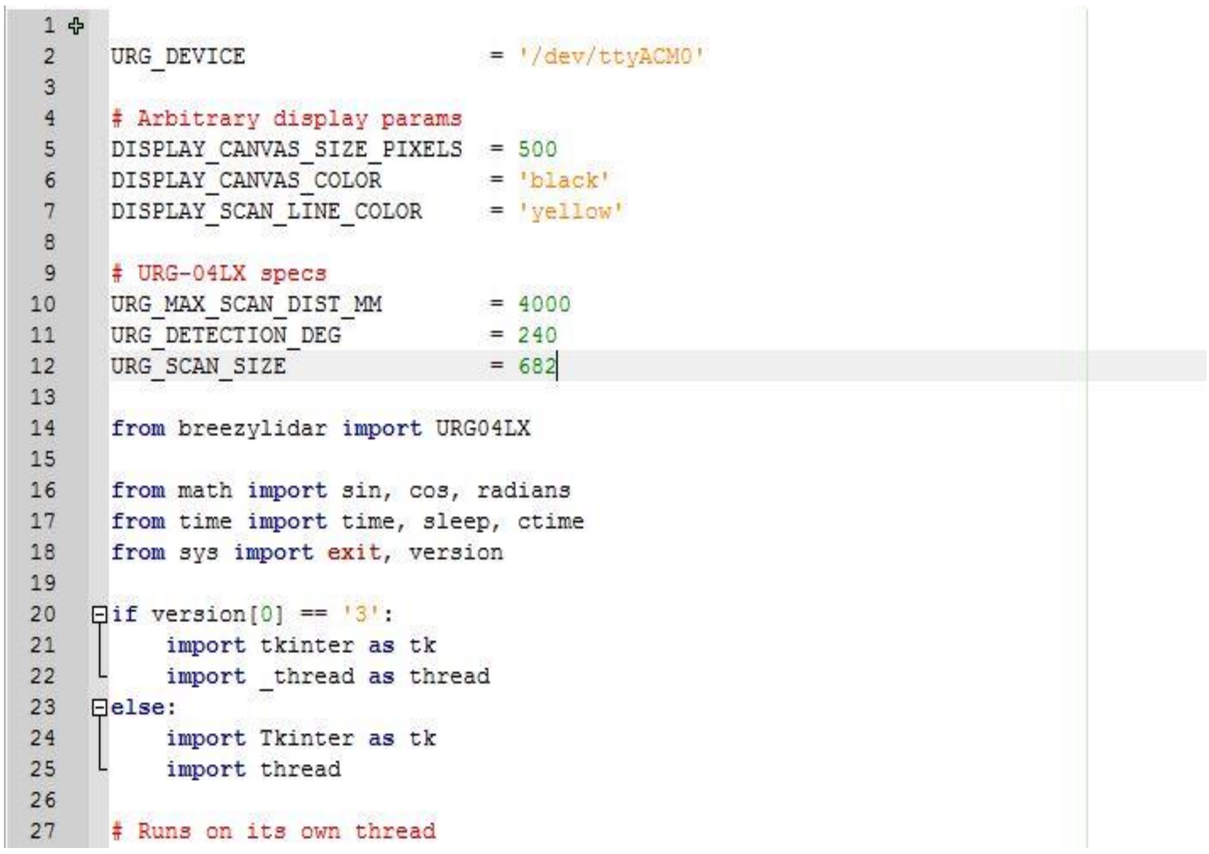
APPENDICES

a. urgtest.py

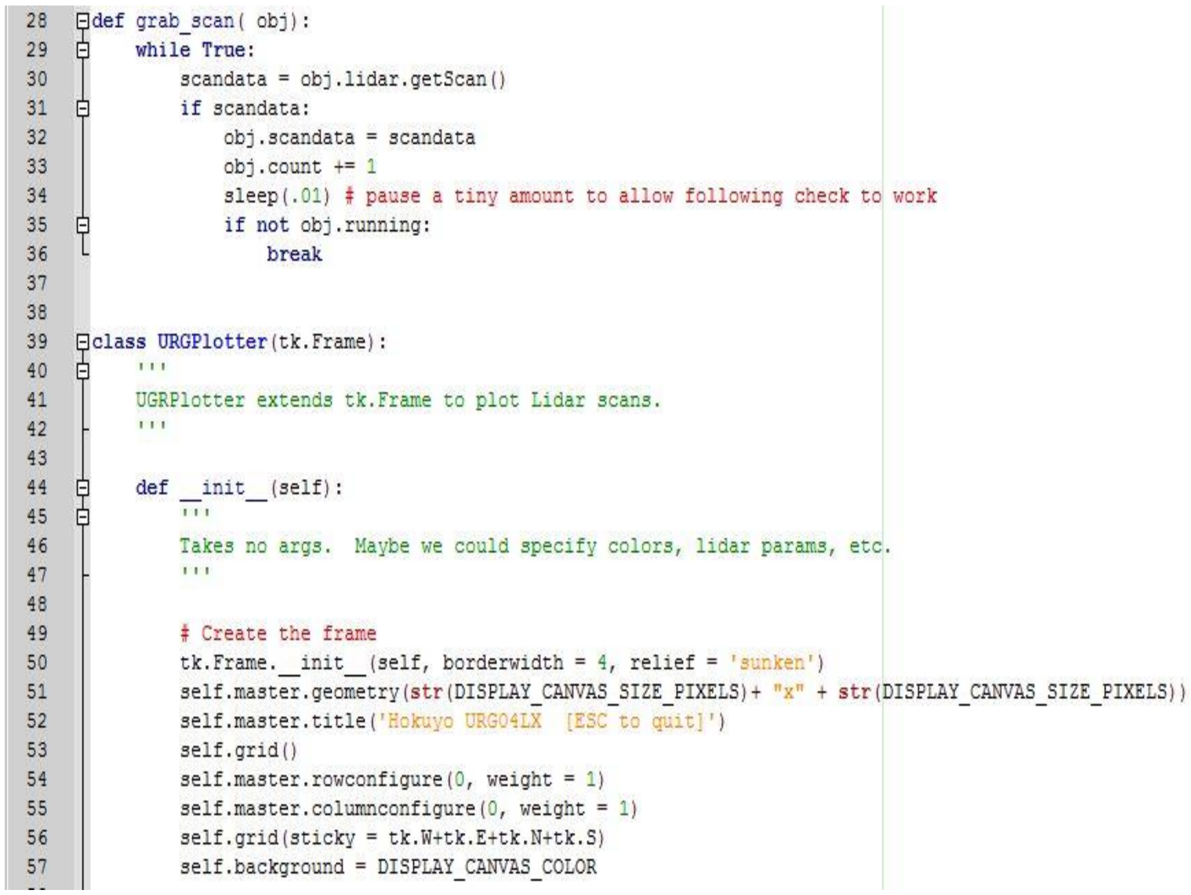


```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#


from breezylidar import URG04LX

import sys
from time import time

DEVICE = '/dev/ttyACM0'
NITER  = 10

laser = URG04LX(DEVICE)

print('===================================================================')
print(laser)
print('===================================================================')


start_sec = time()

count = 0

for i in range(1, NITER+1):

    sys.stdout.write('Iteration: %3d: ' % i)

    data = laser.getScan()

    if data:

        print('Got %3d data points' % len(data))

        count += 1

    else:

        print('=== SCAN FAILED ===')

elapsed_sec = time() - start_sec

print('%d scans in %f seconds = %f scans/sec' % (count, elapsed_sec, count/elapsed_sec))
```

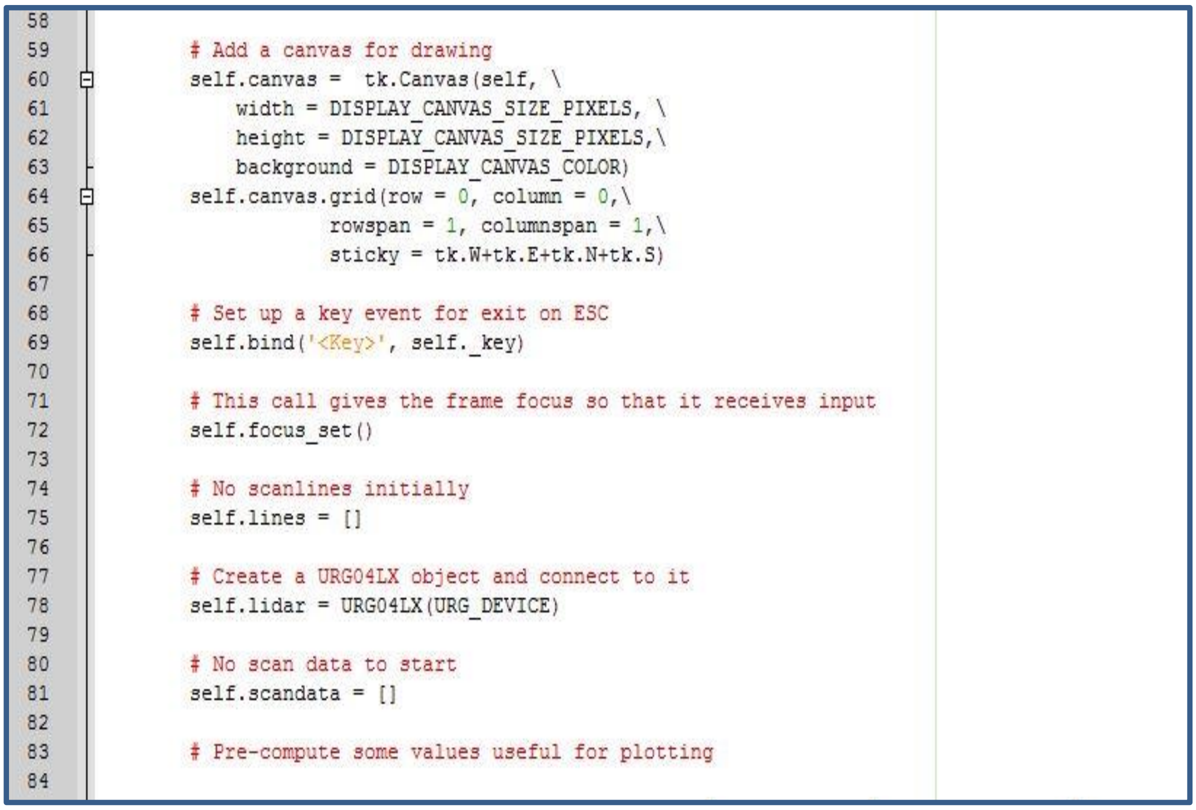
b. urgplot.py



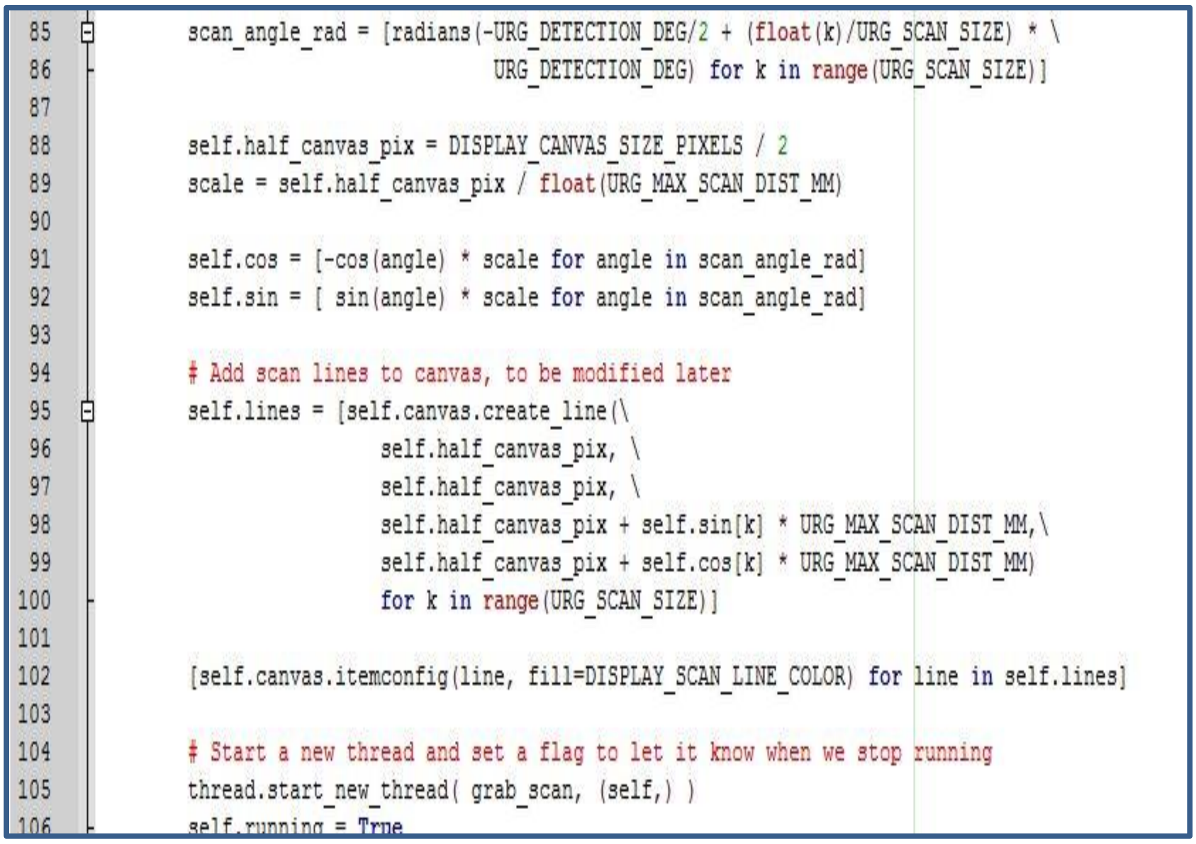
```python

URG_DEVICE                  = '/dev/ttyACM0'

# Arbitrary display params
DISPLAY_CANVAS_SIZE_PIXELS  = 500
DISPLAY_CANVAS_COLOR        = 'black'
DISPLAY_SCAN_LINE_COLOR     = 'yellow'

# URG-04LX specs
URG_MAX_SCAN_DIST_MM        = 4000
URG_DETECTION_DEG           = 240
URG_SCAN_SIZE               = 682

from breezylidar import URG04LX

from math import sin, cos, radians
from time import time, sleep, ctime
from sys import exit, version

if version[0] == '3':
    import tkinter as tk
    import _thread as thread
else:
    import Tkinter as tk
    import thread

# Runs on its own thread
```



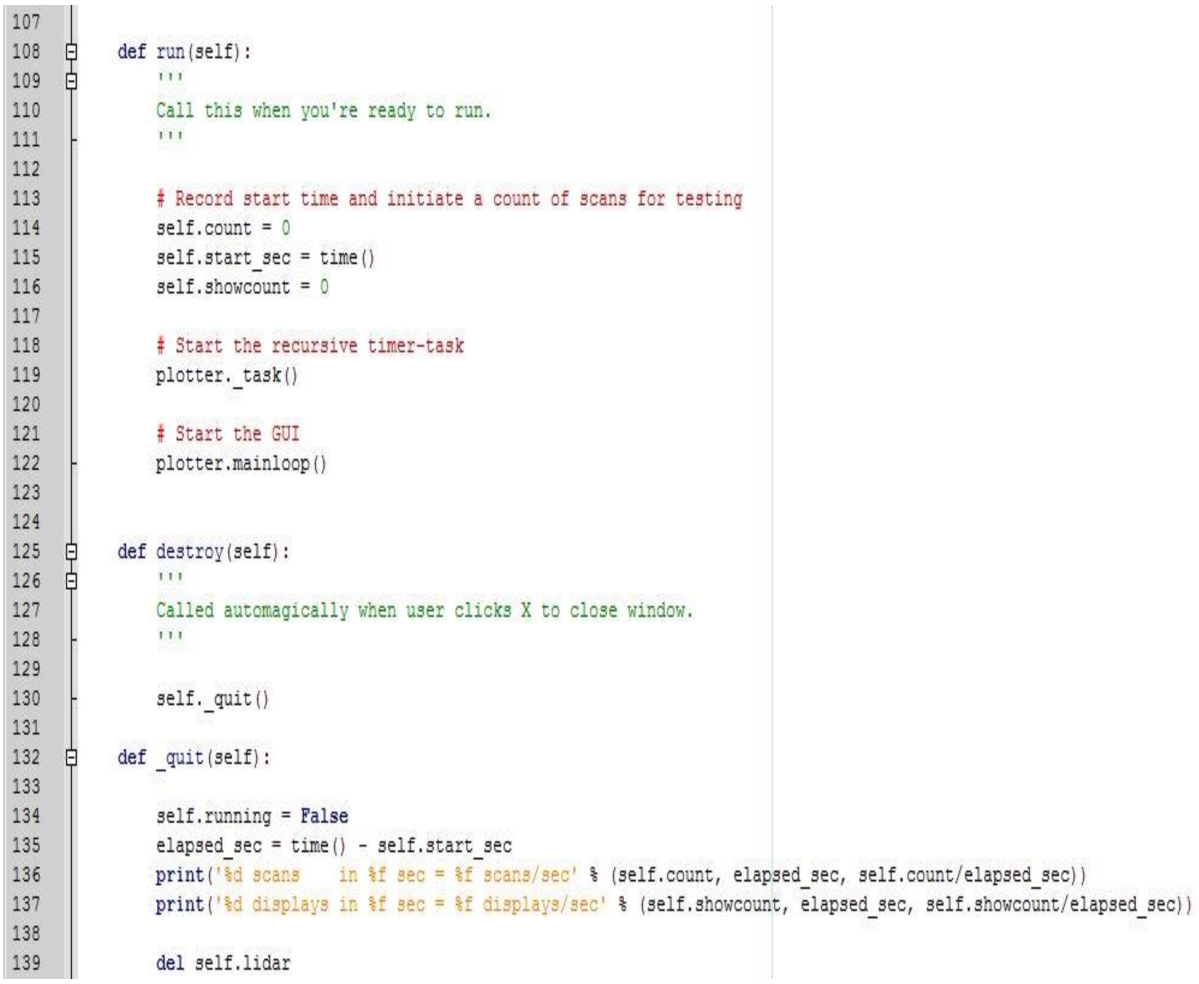
```python
def grab_scan( obj):
    while True:
        scandata = obj.lidar.getScan()
        if scandata:
            obj.scandata = scandata
            obj.count += 1
            sleep(.01) # pause a tiny amount to allow following check to work
            if not obj.running:
                break


class URGPlotter(tk.Frame):
    '''
    UGRPlotter extends tk.Frame to plot Lidar scans.
    '''

    def __init__(self):
        '''
        Takes no args.  Maybe we could specify colors, lidar params, etc.
        '''

        # Create the frame
        tk.Frame.__init__(self, borderwidth = 4, relief = 'sunken')
        self.master.geometry(str(DISPLAY_CANVAS_SIZE_PIXELS)+ "x" + str(DISPLAY_CANVAS_SIZE_PIXELS))
        self.master.title('Hokuyo URG04LX  [ESC to quit]')
        self.grid()
        self.master.rowconfigure(0, weight = 1)
        self.master.columnconfigure(0, weight = 1)
        self.grid(sticky = tk.W+tk.E+tk.N+tk.S)
        self.background = DISPLAY_CANVAS_COLOR
```



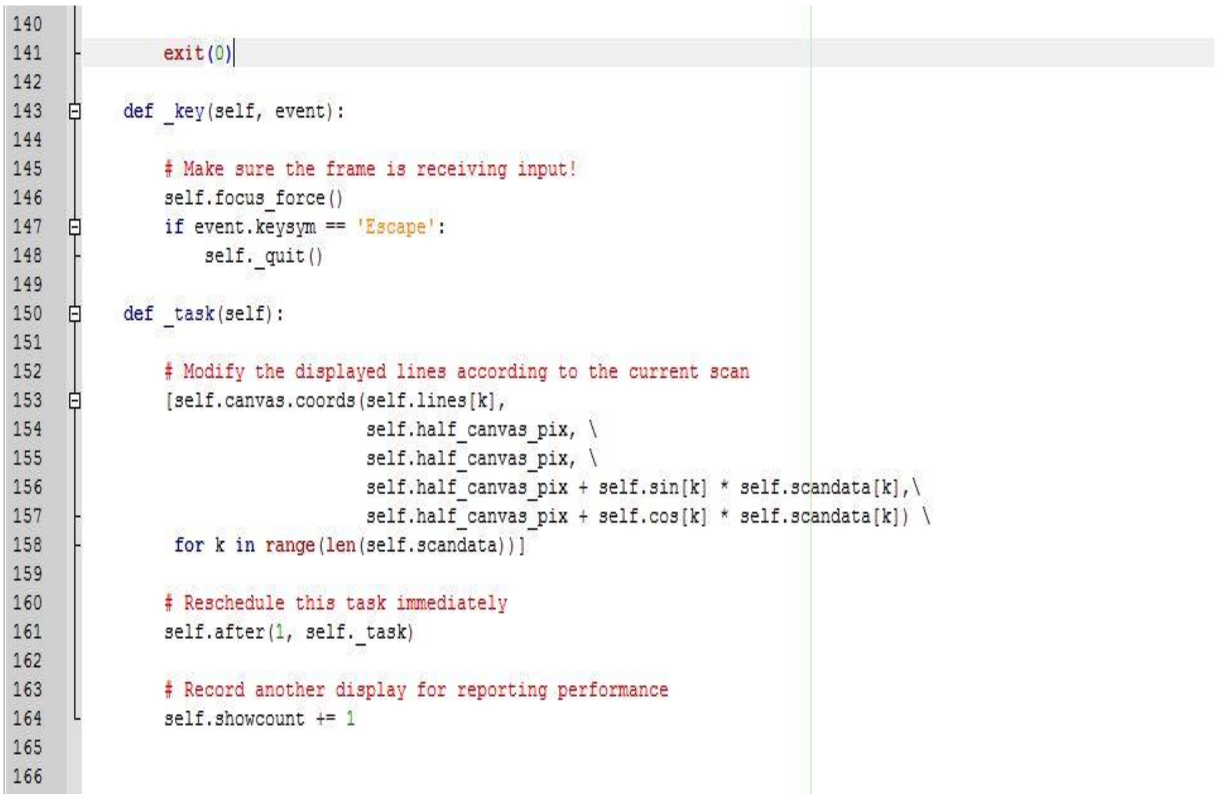
```python
        # Add a canvas for drawing
        self.canvas =  tk.Canvas(self, \
            width = DISPLAY_CANVAS_SIZE_PIXELS, \
            height = DISPLAY_CANVAS_SIZE_PIXELS,\
            background = DISPLAY_CANVAS_COLOR)
        self.canvas.grid(row = 0, column = 0,\
                    rowspan = 1, columnspan = 1,\
                    sticky = tk.W+tk.E+tk.N+tk.S)

        # Set up a key event for exit on ESC
        self.bind('<Key>', self._key)

        # This call gives the frame focus so that it receives input
        self.focus_set()

        # No scanlines initially
        self.lines = []

        # Create a URG04LX object and connect to it
        self.lidar = URG04LX(URG_DEVICE)

        # No scan data to start
        self.scandata = []

        # Pre-compute some values useful for plotting
```



```python
        scan_angle_rad = [radians(-URG_DETECTION_DEG/2 + (float(k)/URG_SCAN_SIZE) * \
                                 URG_DETECTION_DEG) for k in range(URG_SCAN_SIZE)]

        self.half_canvas_pix = DISPLAY_CANVAS_SIZE_PIXELS / 2
        scale = self.half_canvas_pix / float(URG_MAX_SCAN_DIST_MM)

        self.cos = [-cos(angle) * scale for angle in scan_angle_rad]
        self.sin = [ sin(angle) * scale for angle in scan_angle_rad]

        # Add scan lines to canvas, to be modified later
        self.lines = [self.canvas.create_line(\
                        self.half_canvas_pix, \
                        self.half_canvas_pix, \
                        self.half_canvas_pix + self.sin[k] * URG_MAX_SCAN_DIST_MM,\
                        self.half_canvas_pix + self.cos[k] * URG_MAX_SCAN_DIST_MM)
                        for k in range(URG_SCAN_SIZE)]

        [self.canvas.itemconfig(line, fill=DISPLAY_SCAN_LINE_COLOR) for line in self.lines]

        # Start a new thread and set a flag to let it know when we stop running
        thread.start_new_thread( grab_scan, (self,) )
        self.running = True
```

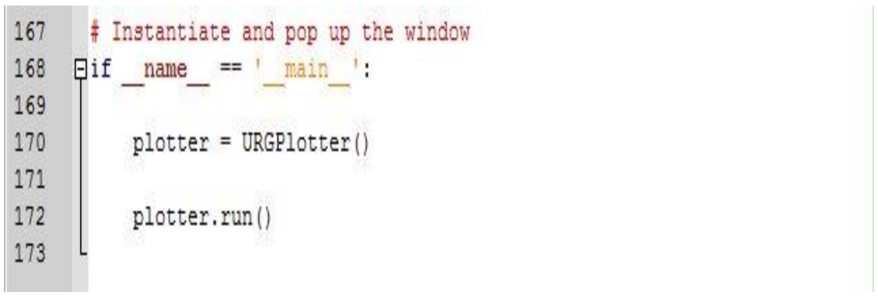


```python

    def run(self):
        '''
        Call this when you're ready to run.
        '''

        # Record start time and initiate a count of scans for testing
        self.count = 0
        self.start_sec = time()
        self.showcount = 0

        # Start the recursive timer-task
        plotter._task()

        # Start the GUI
        plotter.mainloop()


    def destroy(self):
        '''
        Called automagically when user clicks X to close window.
        '''

        self._quit()

    def _quit(self):

        self.running = False
        elapsed_sec = time() - self.start_sec
        print('%d scans    in %f sec = %f scans/sec' % (self.count, elapsed_sec, self.count/elapsed_sec))
        print('%d displays in %f sec = %f displays/sec' % (self.showcount, elapsed_sec, self.showcount/elapsed_sec))

        del self.lidar
```



```python

        exit(0)

    def _key(self, event):

        # Make sure the frame is receiving input!
        self.focus_force()
        if event.keysym == 'Escape':
            self._quit()

    def _task(self):

        # Modify the displayed lines according to the current scan
        [self.canvas.coords(self.lines[k],
                            self.half_canvas_pix, \
                            self.half_canvas_pix, \
                            self.half_canvas_pix + self.sin[k] * self.scandata[k],\
                            self.half_canvas_pix + self.cos[k] * self.scandata[k]) \
         for k in range(len(self.scandata))]

        # Reschedule this task immediately
        self.after(1, self._task)

        # Record another display for reporting performance
        self.showcount += 1


```

23



```
# Instantiate and pop up the window
if __name__ == '__main__':

    plotter = URGPlotter()

    plotter.run()

```