Lidar

Setup:

Hardware:
Laser Scanner RPLiDAR A1M8-R6

Firmware:
1.29

Troubleshooting:

Python package installation:

1.Browsing internet for libraries.
https://github.com/Hyun-je/pyrplidar/tree/master
https://github.com/Roboticia/RPLidar
I've decided to choose the package from Hyun-je because it was newer and it had tests.

2.Testing Hyun-je package.

output:

```
PyRPlidar Info : device is connected
Traceback (most recent call last):
  File "c:\Users\Komfig\Desktop\projects\lidar\main.py", line 10, in <module>
    samplerate = lidar.get_samplerate()
                 ^^^^^^^^^^^^^^^^^^^^^^
  File
"C:\Users\Komfig\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.12_qbz5
n2kfra8p0\LocalCache\Local\pypoetry\Cache\virtualenvs\lidar-m99jlUfZ-py3.12\Lib\s
ite-packages\pyrplidar.py", line 85, in get_samplerate
    discriptor = self.receive_discriptor()
                 ^^^^^^^^^^^^^^^^^^^^^^^^^
  File
"C:\Users\Komfig\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.12_qbz5
n2kfra8p0\LocalCache\Local\pypoetry\Cache\virtualenvs\lidar-m99jlUfZ-py3.12\Lib\s
ite-packages\pyrplidar.py", line 42, in receive_discriptor
    discriptor =
PyRPlidarResponse(self.lidar_serial.receive_data(RPLIDAR_DESCRIPTOR_LEN))

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File
"C:\Users\Komfig\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.12_qbz5
```

```
n2kfra8p0\LocalCache\Local\pypoetry\Cache\virtualenvs\lidar-m99jlUfZ-py3.12\Lib\s
ite-packages\pyrplidar_protocol.py", line 110, in __init__
    self.sync_byte1 = raw_bytes[0]
                      ~~~~~~~~~^^^
IndexError: index out of range
PyRPlidar Info : device is disconnected
```

3. Fixing the issue.
https://github.com/Hyun-je/pyrplidar/issues/1
Changed the Baudrate to 115200, however it still didn't work.

https://stackoverflow.com/questions/66867608/pyrplidar-indexerror-index-out-of-range
I've changed the library from pyrplidar to rplidar.

```
from rplidar import RPLidar
lidar = RPLidar('COM3')

info = lidar.get_info()
print(info)

health = lidar.get_health()
print(health)

for i, scan in enumerate(lidar.iter_scans()):
    print('%d: Got %d measures' % (i, len(scan)))
    if i > 10:
        break
lidar.stop()
lidar.stop_motor()
lidar.disconnect()


I've got the error:
{'model': 24, 'firmware': (1, 29), 'hardware': 7, 'serialnumber':
'93A9ED95C4E493CAA5E69EF002794B6E'}
('Good', 0)
Traceback (most recent call last):
  File "c:\Users\Komfig\Desktop\projects\lidar\main.py", line 10, in <module>
    for i, scan in enumerate(lidar.iter_scans()):
  File
"C:\Users\Komfig\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.12_qbz5
n2kfra8p0\LocalCache\Local\pypoetry\Cache\virtualenvs\lidar-m99jlUfZ-py3.12\Lib\s
ite-packages\rplidar.py", line 357, in iter_scans
    for new_scan, quality, angle, distance in iterator:
  File
"C:\Users\Komfig\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.12_qbz5
n2kfra8p0\LocalCache\Local\pypoetry\Cache\virtualenvs\lidar-m99jlUfZ-py3.12\Lib\s
ite-packages\rplidar.py", line 323, in iter_measurments
```

```
    raw = self._read_response(dsize)
          ^^^^^^^^^^^^^^^^^^^^^^^^^^
  File
"C:\Users\Komfig\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.12_qbz5
n2kfra8p0\LocalCache\Local\pypoetry\Cache\virtualenvs\lidar-m99jlUfZ-py3.12\Lib\s
ite-packages\rplidar.py", line 199, in _read_response
    raise RPLidarException('Wrong body size')
rplidar.RPLidarException: Wrong body size
```

https://github.com/adafruit/Adafruit_CircuitPython_RPLIDAR/issues/15

Reviewing the code I've added the port='COM3' and timeout=3 as the parameters

```
from rplidar import RPLidar
lidar = RPLidar(port='COM3',timeout=3)

info = lidar.get_info()
print(info)

health = lidar.get_health()
print(health)

for i, scan in enumerate(lidar.iter_scans()):
    print('%d: Got %d measures' % (i, len(scan)))
    if i > 10:
        break
lidar.stop()
lidar.stop_motor()
lidar.disconnect()
```

Running the code for the first time i've gotten the output:

```
Traceback (most recent call last):
  File "c:\Users\Komfig\Desktop\projects\lidar\main.py", line 4, in <module>
    info = lidar.get_info()
           ^^^^^^^^^^^^^^^^
  File
"C:\Users\Komfig\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.12_qbz5
n2kfra8p0\LocalCache\Local\pypoetry\Cache\virtualenvs\lidar-m99jlUfZ-py3.12\Lib\s
ite-packages\rplidar.py", line 211, in get_info
    dsize, is_single, dtype = self._read_descriptor()
                              ^^^^^^^^^^^^^^^^^^^^^^^
  File
"C:\Users\Komfig\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.12_qbz5
n2kfra8p0\LocalCache\Local\pypoetry\Cache\virtualenvs\lidar-m99jlUfZ-py3.12\Lib\s
ite-packages\rplidar.py", line 189, in _read_descriptor
    raise RPLidarException('Incorrect descriptor starting bytes')
rplidar.RPLidarException: Incorrect descriptor starting bytes
```

However, the second time I ran the code it ran correctly.

```
{'model': 24, 'firmware': (1, 29), 'hardware': 7, 'serialnumber':
'93A9ED95C4E493CAA5E69EF002794B6E'}
('Good', 0)
0: Got 127 measures
1: Got 165 measures
2: Got 169 measures
3: Got 167 measures
4: Got 171 measures
5: Got 172 measures
6: Got 164 measures
7: Got 171 measures
8: Got 169 measures
9: Got 173 measures
10: Got 163 measures
11: Got 162 measures
```

Depending on the position of the sensor the output was different.

Preparing script to read data from sensor:

Using pre-made script to get example of measurements:

https://github.com/SkoltechRobotics/rplidar/blob/master/examples/record_measurments.py
Using an example from the rplidar library, I have gotten the expected output, that returned angle of the measurement, as well as the distance from the lidar scanner in the .txt format.

```
#!/usr/bin/env python3
'''Records measurments to a given file. Usage example:
$ ./record_measurments.py out.txt'''
import sys
from rplidar import RPLidar
PORT_NAME = '/dev/cu.usbserial-0001'
def run(path):
    '''Main function'''
    lidar = RPLidar(PORT_NAME,timeout=3)
    outfile = open(path, 'w')
    try:
        print('Recording measurments... Press Crl+C to stop.')
        for measurment in lidar.iter_measurments():
            line = '\t'.join(str(v) for v in measurment)
            outfile.write(line + '\n')
    except KeyboardInterrupt:
        print('Stoping.')
    lidar.stop()
    lidar.disconnect()
    outfile.close()
if __name__ == '__main__':
    run("/Users/rousanter/Desktop/lidar/out.txt")
```

When I was trying to run the example on my Windows PC, poetry library stopped working, as it wouldn’t install numpy, returning errors as well as information that it was already installed, when in fact, it wasn’t. I couldn’t find the solution to the problem, so I switched environments to macOS and it ran without any issues. I will continue the project on this OS.

Parsing data:

I started by dividing the offline output from the .txt file into chunks, each made out of full 360° scans. I've collected all necessary information from rows in each chunk (angle and distance). All data collected is stored in a list that is made of lists of full scans. I also made a class, so the variables from measurements can be easily accessible.

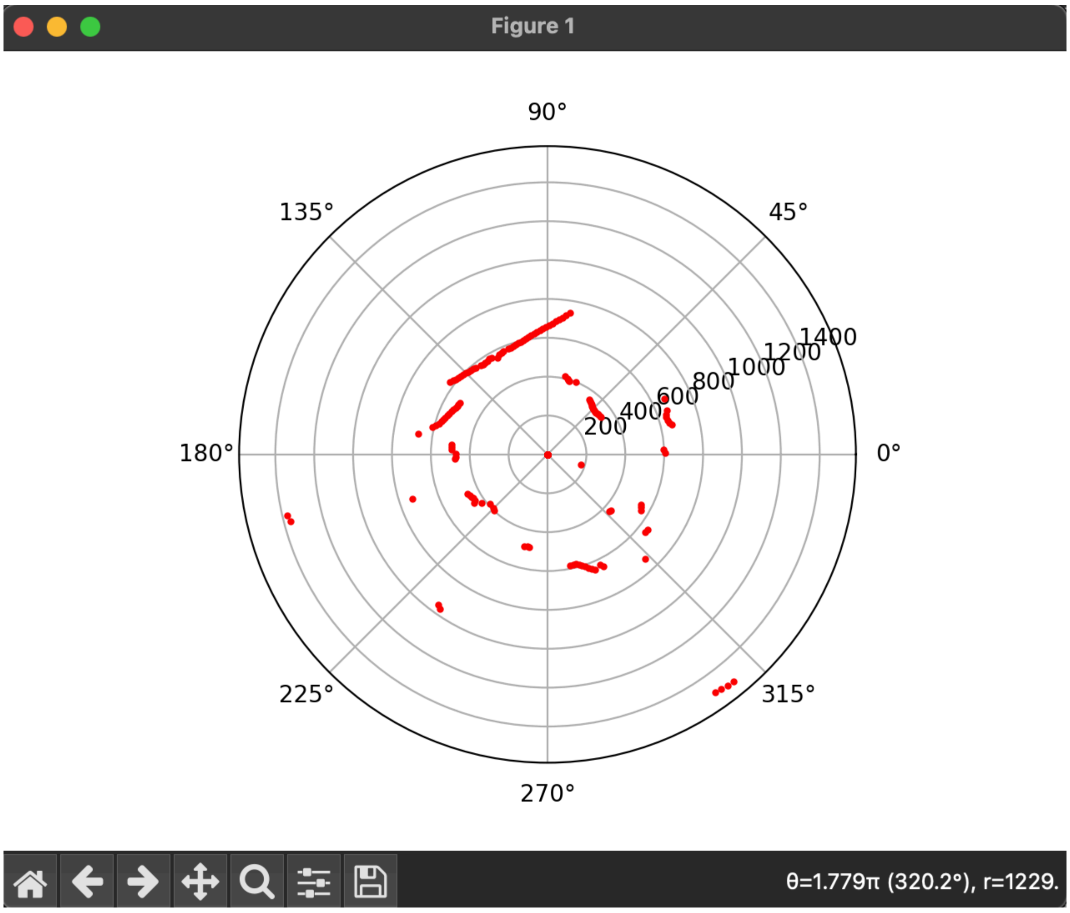
```python
class Measurement:
    def __init__(self, accuracy, angle, distance):
        self.accuracy = accuracy
        self.angle=angle
        self.distance=distance
def parse_row(row):
    my_var=row.strip("\n").split("\t")
    return Measurement(int(my_var[1]),float(my_var[2]),float(my_var[3]))
def parse_chunk(chunk):
    measurement_list=[]
    for row in chunk:
        measurement_list.append(parse_row(row))
    return measurement_list
def get_raw_data(path_to_file):
    f= open(path_to_file)
    all_raw_data=[]
    full_circle=[]
    for line in f:
        if "True" in line:
            all_raw_data.append(full_circle)
            full_circle=[]
            full_circle.append(line)
        else:
            if "False" in line:
                full_circle.append(line)
            else:
                break
    all_raw_data.pop(0)
    return all_raw_data
def get_data(path_to_file):
    raw_data_list=get_raw_data(path_to_file)
    data_list=[]
    for chunk in raw_data_list:
        data_list.append(parse_chunk(chunk))
    return data_list
```

Displaying collected data on matplotlib grid:

Using data parser, I was able to map each point on matplotlib canvas. It displays polar coordinates from offline data stored in out.txt file. Number_of_cycles variable shows the number of data collected from full rotation of the lidar sensor.

```python
from sensor_parser import get_data
import matplotlib.pyplot as plt
import numpy as np
import math
if __name__=="__main__":
    data = get_data("out.txt")
    a=[]
    b=[]
    number_of_cycle = 4
    print(type(data[0][0].accuracy))
    for i in range(0,len(data[number_of_cycle])):
        if data[number_of_cycle][i].accuracy ==0:
            continue
        else:
            a.append(math.radians((data[number_of_cycle][i].angle)))
            b.append(data[number_of_cycle][i].distance)
    fig, ax =plt.subplots(subplot_kw={'projection':'polar'})
    ax.plot(a,b, 'ro', markersize=2)
    plt.show()
```

Below is an example of the output:



Displaying real time data from sensor:

Based on the example from the rplidar library I’ve created an empty list, that collects data from 360° scan, that then is displayed on the grid, and the list is emptied. Each time the grid is updated.

```
from sensor_parser import Measurement
from rplidar import RPLidar
import matplotlib.pyplot as plt
import math
PORT_NAME = '/dev/cu.usbserial-0001'
def update_canvas(data, fig, line):
    parsed_data=list()
    for measurement in data:
        if measurement.accuracy==0:
            continue
        else:
            parsed_data.append(measurement)
    a=list()
    b=list()
    for i in range(len(parsed_data)):
        a.append(math.radians(parsed_data[i].angle))
        b.append(parsed_data[i].distance)
    print(a)
    print(b)
    line.set_data(a, b)
    fig.canvas.draw()
    fig.canvas.flush_events()

def run():
    '''Main function'''
    lidar = RPLidar(PORT_NAME,timeout=3)
    snapshot = list()
    plt.ion()
    fig, ax =plt.subplots(subplot_kw={'projection':'polar'})
    line, =ax.plot([0],[1200], 'ro', markersize=2, markeredgecolor="r")
    try:
        print('Recording measurments... Press Crl+C to stop.')
        counter = 1
        for measurment in lidar.iter_measurments():
            if measurment[0]==True:
                counter -= 1
                if counter == 0:
                    counter = 2
                    update_canvas(snapshot, fig, line)
                snapshot=list()
                snapshot.append(Measurement(accuracy=measurment[1],
angle=measurment[2], distance=measurment[3]))
            else:
                if measurment[0]==False:
                    snapshot.append(Measurement(accuracy=measurment[1],
angle=measurment[2], distance=measurment[3]))
    except KeyboardInterrupt:
        print('Stopped.')
    lidar.stop()
    lidar.disconnect()
if __name__ == '__main__':
    run()
```

Further development:

- Processing data from moving sensor
- Simultaneous Localization and Mapping