SLAM ROBOT

UV LED DISINFECTION ROBOT

A Design Project Report

Presented to the School of Electrical and Computer Engineering of Cornell

Submitted By

Yanrui Wang (yw2226)

MEng Field Advisor: Joseph Skovira

MEng Outside Advisor: Vlad Protasenko

Abstract

Project Title: SLAM robot, a branch from UV LED Disinfection Robot

Author: Yanrui Wang (yw2226)

Abstract:

The M. Eng project, UV Disinfection Robot, created an autonomous robot for disinfection of the underside of furniture in a room. Applications of this project would be in the health and food industry where the risk of infection or contamination is particularly important. To achieve efficient disinfection, I did this SLAM, simultaneous localization and mapping, robot to build a 2D map for any disinfected rooms.

For this project, I utilized the iRobot Create 2 interface. To control iRobot Create 2, I used Raspberry Pi 3b+ as the controller. In order to build a 2D map, I added a rplidar sensor to compute distance from robot to around objects. In addition, due to consideration of swarm robots in the future, everything was implemented on ROS, Robot Operating System. Moreover, the PC running ROS was used to generate a real-time 2D map to display simultaneously.

Suffice to say, I used a PC to send control commands to Raspberry Pi 3b+ by Wifi. Meanwhile, Raspberry Pi 3b+ sends rplidar sensor data to PC in order to build a 2D map.

<u>Keywords:</u> robot, Roomba, SLAM (simultaneous localization and mapping), 2D map

Design Problems

To disinfect undersides of tables efficiently, we hope to design efficient motion planning algorithms. This heavily relies on a map of any environment. If we are able to build a 2D map of disinfected rooms, the robots could disinfect undersides of tables more efficiently. This is a good start for swarm robots in the future design since all the robots could share maps on cloud.

Requirement Summary:

- Autonomous throughout entire deployment lifetime
- Easy use by someone unfamiliar with project hardware and software
- Clean wires for non-professional people to use
- Robot easy to move or transport
- Compact mechanical design that can fit onto the Create2's base
- Accurate sensor data to detect distance from robot to around objects
- Odometry information to calibrate mapping contour
- SLAM algorithm to generate real time 2D map and robot localization

Progress Summary (In Detail)

1. I installed ROS Kinetic Kame (Robert Operating System) on my laptop (ubuntu 16.04).
    a. I initially installed ROS on ubuntu 18.04. However, when I tried to compile files of hector mapping on melodic (melodic is ROS for ubuntu 18.04), there was an error. Based on what I did, it's better to use ubuntu 16.04 instead of 18.04 for SLAM.
    b. ROS installation tutorial (for ubuntu 16.04): http://wiki.ros.org/kinetic/Installation/Ubuntu

2. I installed ROS (Robert Operating System) on Raspberry Pi 3b+ (ubuntu 18.04).
    a. Raspberry Pi 3b+ is used to gather rplidar data and send data to PC. If you want to do SLAM on local Raspberry Pi 3b+, you must install ubuntu 16.04 on Raspberry Pi 3b+.

b. The file for ubuntu 16.04 could be downloaded from ubuntu website.

c. ROS installation tutorial (for ubuntu 16.04 and ubuntu 18.04) could be found on ROS wiki.

3. I tested the RPLIDAR A1 sensor on my laptop and Raspberry Pi 3b+. It could be outputted distance data by meters in 4Hz successfully from 0 degrees to 360 degrees. To successfully compile files on Raspberry Pi 3, we must increase swap area. I increased it to 2GB.

    a. *RPLIDAR A1 sensor*

        i. RPLIDAR emits modulated infrared laser signal and the laser signal is then reflected by the object to be detected.

        ii. There is a spinning motor on rplidar A. During rotating,   a 360 degree scan of the current environment will be performed.

        iii. The system measures distance data in more than 2000 times per second and with high resolution distance output (<1% of the distance).

        iv. Specs of RPLIDAR A1

Item	Unit	Value
Distance Range	meter	0.15 - 6
Distance Resolution	mm	<1% of the resolution
Scan Rate	Hz	4 - 10

        v. RPLIDAR A1 uses serial port (UART) as the communication interface. There is only one USB cable to be connected. (See on System Diagram on next section)

        vi. Follow the following two github pages carefully to download RPLIDAR code and run the code on ROS.
        https://github.com/slamtec/rplidar_ros
        https://github.com/robopeak/rplidar_ros/wiki

   b. If you compile the files on Raspberry Pi 3b+. The system would be crashed since the swap area is too small, only approximately 100M. So we need to increase the swamp area to 2GB on Raspberry Pi 3b+.

4. Considering the communication between my laptop and Raspberry Pi, I tested communication between them by Wifi. To communicate between two machines on ROS, we need some editing on .bashrc file and /etc/ssh/sshd_config file on both PC and Raspberry Pi 3b+.

   Important Note:

   10.132.0.225 is the IP address of your laptop. (replaced with your IP address)
   10.148.10.56 is the IP address of your RPi-3. (replaced with your IP address)

   a. On Raspberry Pi 3b+:
      i. *sudo service ssh status*
      ii. *sudo service ssh reload*
      iii. *sudo service ssh start*

      Note: Make Sure ssh service is active (You might get some errors. Google it and install some packages. It's easy to be solved.)

      iv. *gedit /etc/ssh/sshd_config*
         1. **In your sshd_config file, add some lines at the bottom of the file.**

            # Port 22

            # PermitRootLogin no

            # AllowUsers yanray47@10.132.0.225

      Note: yanray47 is replaced with your username of ubuntu 16.04. 10.132.0.225 is the IP address for your laptop.

      v. *gedit .bashrc*
         1. **In your sshd_config file, add some lines at the bottom of the file.**

export ROS_MASTER_URI=http://10.132.0.225:11311/   *(master PC   IP address)* *export ROS_HOSTNAME=10.148.10.56* *(slave RPi3 IP address)*

export ROS_IP=10.148.10.56 *(slave RPi3 IP address)*

b. On your laptop:

   i. *sudo service ssh status*

   ii. *sudo service ssh reload*

   iii. *sudo service ssh start*

   Note: Make Sure ssh service is active (You might get some errors. Google it and install some packages. It's easy to be solved.)

c. *gedit /etc/ssh/sshd_config*

   i. **In your sshd_config file, add some lines at the bottom of the file.**

```
# PermitRootLogin no
# AllowUsers slam_robot@10.148.10.56
```

   Note: slam_robot is replaced with your username of Raspberry Pi 3b+. 10.148.10.56 is the IP address for your Raspberry Pi 3b+.

d. *gedit .bashrc*

   i. **In your sshd_config file, add some lines at the bottom of the file.**

export ROS_MASTER_URI=http://localhost:11311/

export ROS_HOSTNAME=10.132.0.255 *(master PC IP address)*

Finally my laptop (ubuntu 16.04) running ROS is able to transmit data with RPi3 (ubuntu 18.04) running ROS by Wifi.

I have two test cases.

a. First test case is on the laptop, running a . On RPi-3, running a *turtle_teleop_key* node to send control commands by Wifi. The turtle on the laptop would be moved.

   i. In the first terminal of your laptop. (Typing following command)

      1. *roscore*

ii. In the second terminal of your laptop, in the catkin_make folder, (this depends on where you install your catkin_make folder).

   1. *source devel/setup.bash*
   2. *rosrun turtlesim turtlesim_node*

iii. In the first terminal of your RPi-3, in the catkin_make folder, (this depends on where you install your catkin_make folder).

   1. *source devel/setup.bash*
   2. *rosrun turtlesim turtle_teleop_key*

Note, at this point, you're able to control the turtle on your laptop by your keyboard connected with your RPi-3. (You could switch the node running on your laptop and RPi-3.)

b. The second case is ROS on the laptop running a listener node, then ROS on RPi3 running a talker node. The listener could receive the message sent by the talker. **Note, before you run this test case, you must download the code on my Github in the appendix section, add the downloaded folder into your catkin_make/src folder.**

   i. In the first terminal of your laptop. (Typing following command)
      1. *roscore*
   ii. In the second terminal of your laptop, in the catkin_make folder, (this depends on where you install your catkin_make folder).
      1. *source devel/setup.bash*
      2. *roslaunch beginner_tutorials talker.py*
   iii. In the first terminal of your RPi-3, in the catkin_make folder, (this depends on where you install your catkin_make folder).
      1. *source devel/setup.bash*
      2. *roslaunch beginner_tutorials listener.py*

Note, in listener.py, comment everything about *serial.write and serial*.

Note, at this point, you're able to send a number on your laptop to your RPi-3. (You could switch the node running on your laptop and RPi-3.)

5. For clean wires design, I used a USB cable to send serial commands to iRobot Create directly instead of using *3.3V to 5V logic converter*.
    a. How to connect everything together is in next section (System Diagram)
6. To mount RPLIDAR A1 sensor firmly on iRobot Create 2, I designed three acrylic structures to mount robotic arm and lidar sensor firmly on the robot. (This SLAM project only tests the connection with RPLIDAR A1 sensor)
    a. The bottom layer, on Fig. 1, is used to attach acrylic firmly with iRobot Create 2, which generates possibilities to add more layers on top of the iRobot Create. The width of the laser cut figure is 291mm.
    b. The second layer, on Fig. 2, is used to put battery and Raspberry Pi. To present clean wires, I used acrylic to design a battery case. The upper holes are for battery case. The lower four holes are used to connect Raspberry Pi 3b+. The width of the laser cut figure is 291mm.
    c. The third layer, on Fig. 3, is used to mount a rplidar sensor. The width of the laser cut figure is 291mm.
    d. The battery case, on Fig. 4, is a cuboid design to store battery inside it. Fig. 5 is a reference figure showing how to mount a battery case.
    Notes: (You must specify the width if you cut acrylic in Cornell RPL. All the circles are #4 screw size. Approximately 15mm #4 screw length is good to connect the battery case. In addition, the middle layer and top layer are switchable. If you mount a robotic arm on iRobot Create 2, you must put top layer in the second layer.)
    e. The final robot appearance is on Fig. 6.

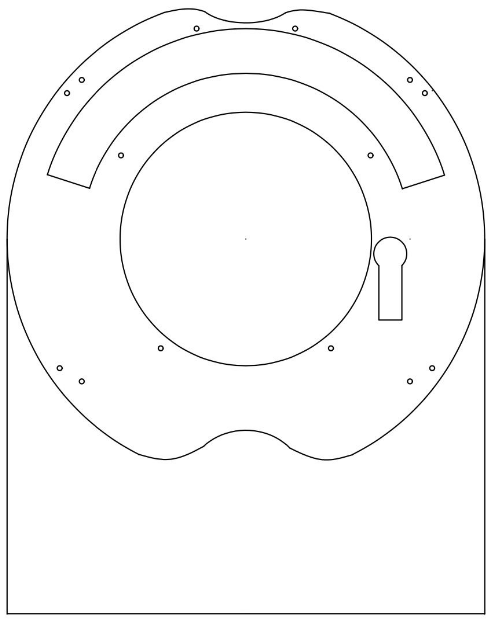


Fig. 1 bottom layer

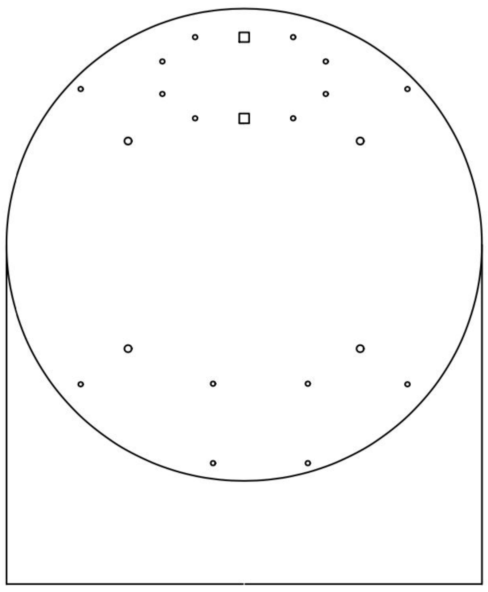


Fig. 2 middle layer

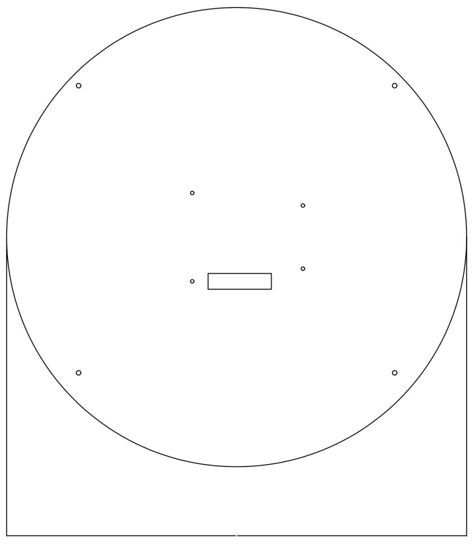


Fig. 3 top layer

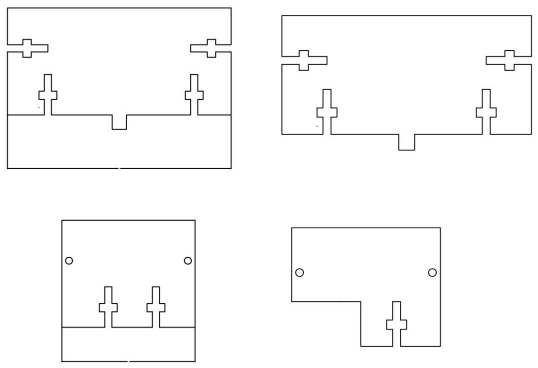


Fig. 4 battery case design

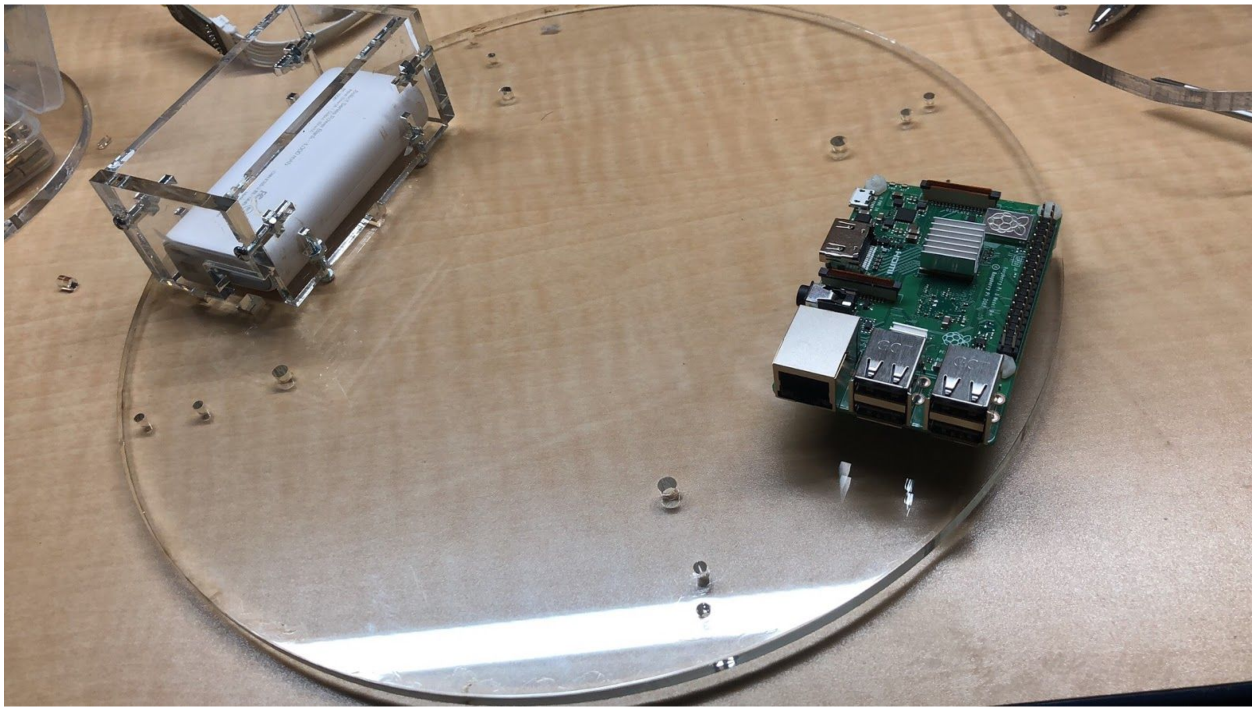


Fig. 5 battery case connection

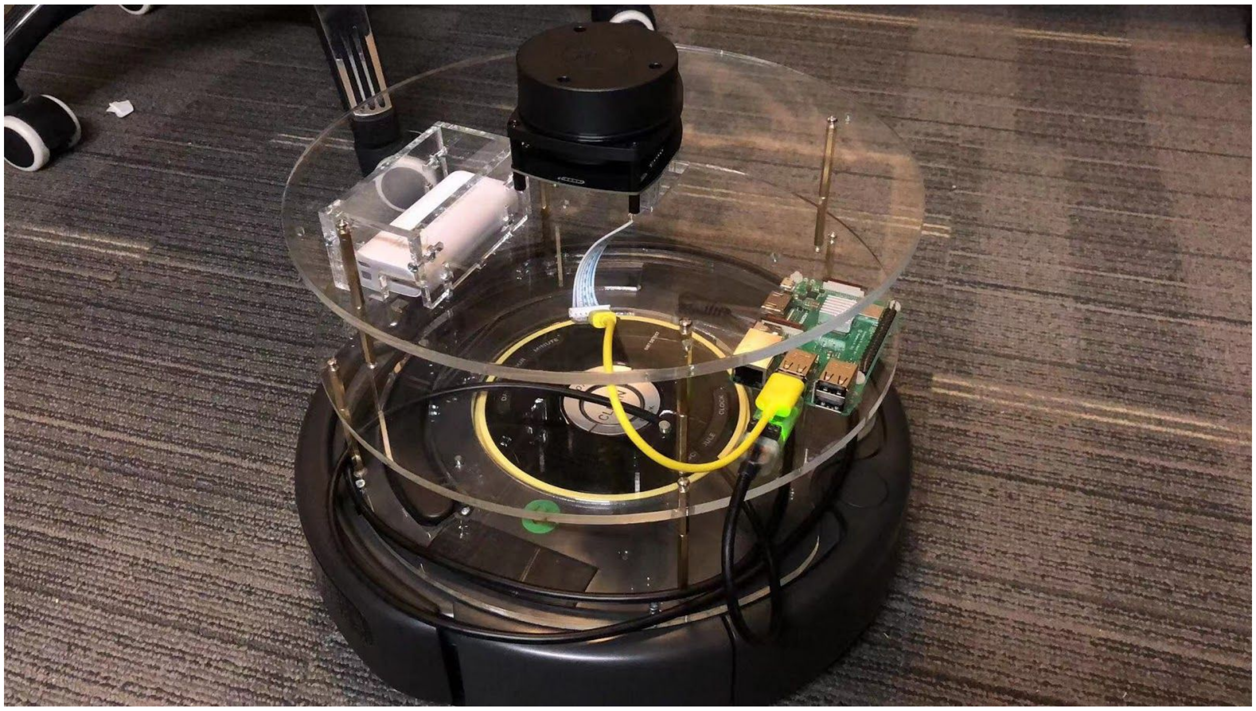


Fig. 6 robot appearance

7. Before building a map, we also need to get accurate odometry information. Since iRobot Create has built in encoders, the building in algorithms are able to output odometry information. You could refer to iRobot Create 2 to grab odometry information. I wrote a simple test case to control robot movement by distance and angle instead of PWM mode. I only checked documentation, but I haven't tested odometry information since hector mapping doesn't need odometry information.
8. I have tested hector mapping on ROS. It showed an initial map displayed on ROS RVIZ in real-time. See on Fig. 7. The black lines are objects boundary or walls. The grey areas are free space. The other areas are an unknown environment. The three colorful lines are the pose (x, y theta) of the robot. Based on my observation, the map is not accurate enough. Since we need to control robots to move in each time step. The other doable mapping method is gmapping on ROS. There are few bugs that I haven't finished yet. Gmapping requires odometry information.

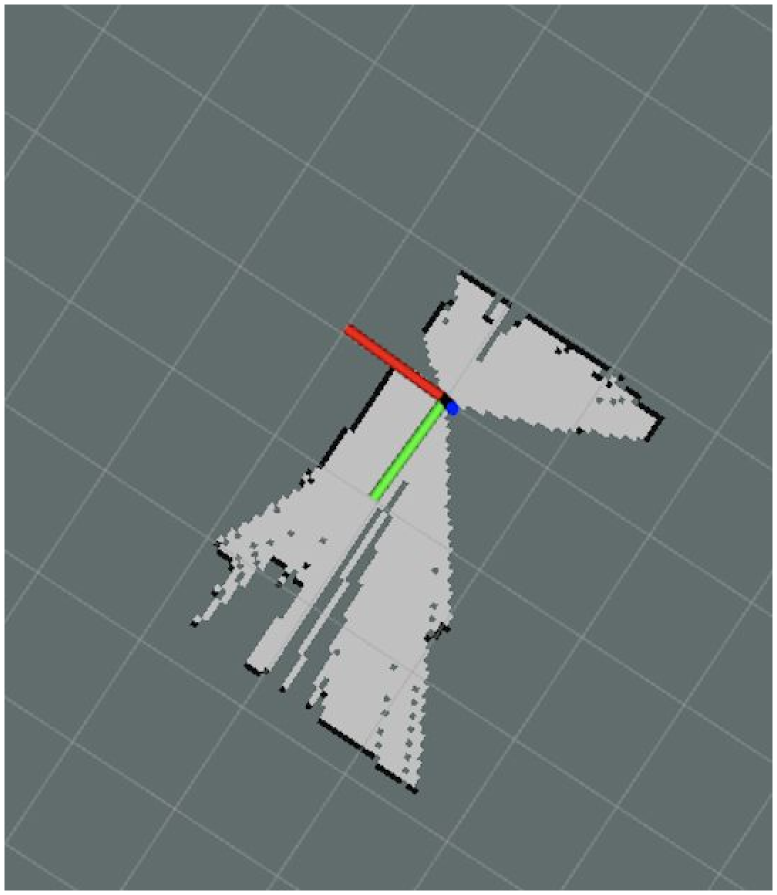


Fig. 7 initial map display

9. The final two test cases are building a lab for Professor Skovira's lab and the hallway between Professor Skovira's lab and Professor land's lab.

   Note: Assumption, please make sure you have configured everything above without any error, especially Wifi configuration.

   Put the robot on the ground.

   a. First test case is on the laptop, running a turtlesim_node node. On RPi-3, running a *turtle_teleop_key* node to send control commands by Wifi. The turtle on the laptop would be moved.

      iv. In the first terminal of your laptop. (Typing following command)
         1. *roscore*

      v. In the second terminal of your laptop, in the catkin_make folder, (this depends on where you install your catkin_make folder).
         1. *source devel/setup.bash*
         2. *roslaunch hector_slam_launch hector_ugv.launch*

      vi. In the third terminal of your laptop, in the catkin_make folder, (this depends on where you install your catkin_make folder).
         1. *source devel/setup.bash*
         2. *roslaunch beginner_tutorials talker.py*

      Note: To control the robot to move, you must enter 5 several different commands in your terminal.

      Enter 'w' to make the robot move forward. Enter 'd' to the robot turn right. Enter 'a' to make the robot turn left. Enter 's' to make the robot stop. Enter 'x' to make the robot move backward. Control + C is to quit the *talker* node. By the way, based on my test, you're not able to change the robot sharply, the might make pose estimation not accurate. If you must do that, changing the scan rate might be helpful, but I haven't tested on this point.

      Note: you need to configure your RPi-3 wifi first, then unplug in your HDMI cable, and use another laptop ssh your RPi-3.

vii. SSH login, in the first terminal of your RPi-3, in the catkin_make folder, (this depends on where you install your catkin_make folder).

   1. *source devel/setup.bash*
   2. *roslaunch rplidar_ros rplidar.launch*

viii. SSH login, in the first terminal of your RPi-3, in the catkin_make folder, (this depends on where you install your catkin_make folder).

   1. *source devel/setup.bash*
   2. *roslaunch beginner_tutorials listener.py*

At this point, what you need to do is control the robot by your PC, enter different driving commands, finally you're able to build a 2D real-time map, see on Fig. 8. It takes about 90 seconds to make a whole run to get the final map.

b. The second test case is the same process on your laptop and RPi-3. The only difference is put your robot in the hallway.

   Finally you're able to build a 2D real-time map, see on Fig. 9. It takes about 120 seconds to make a whole run to get the final map.

Hector_mapping estimates the pose of the robot based on scan rate of the sensors.

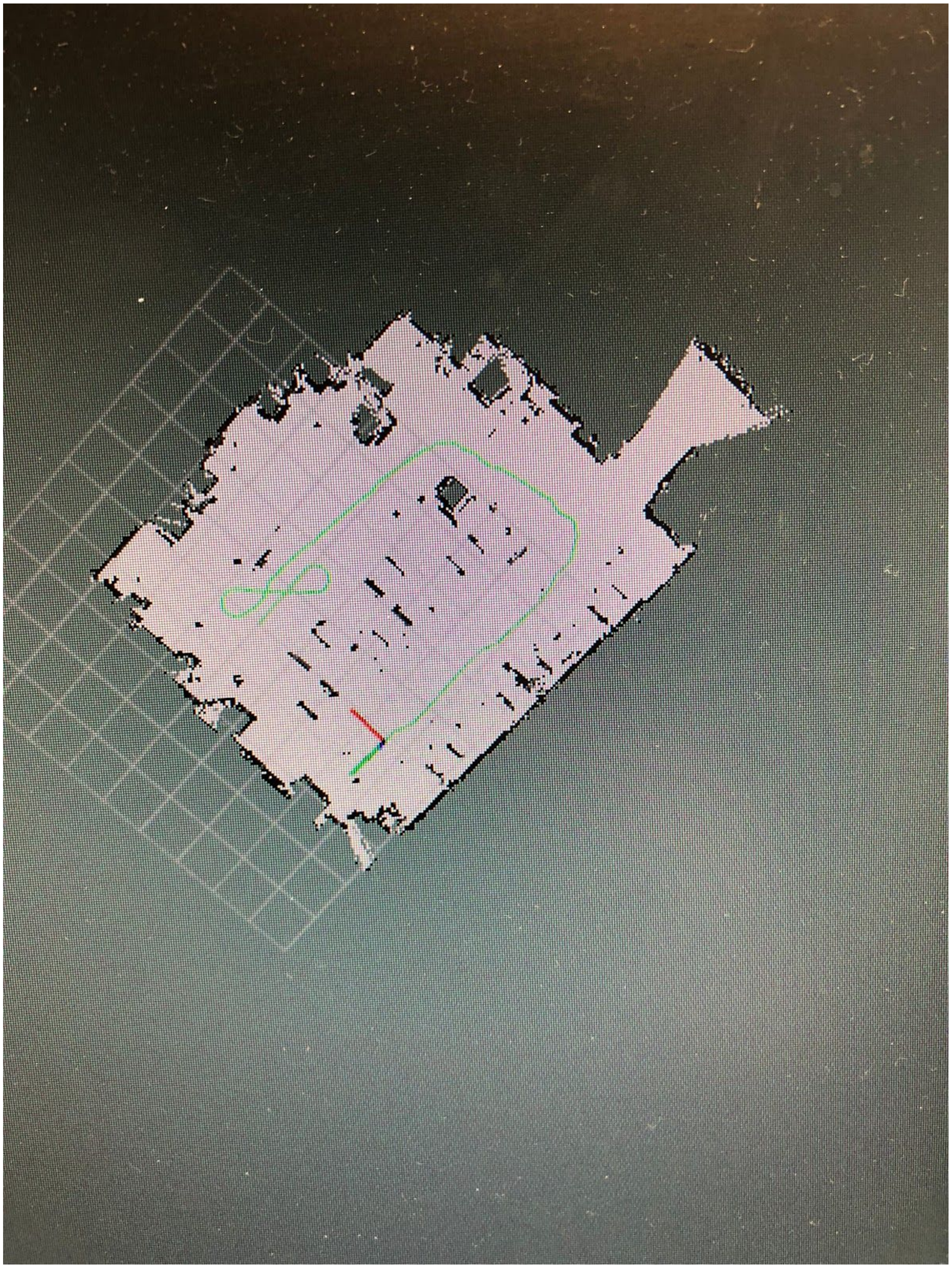
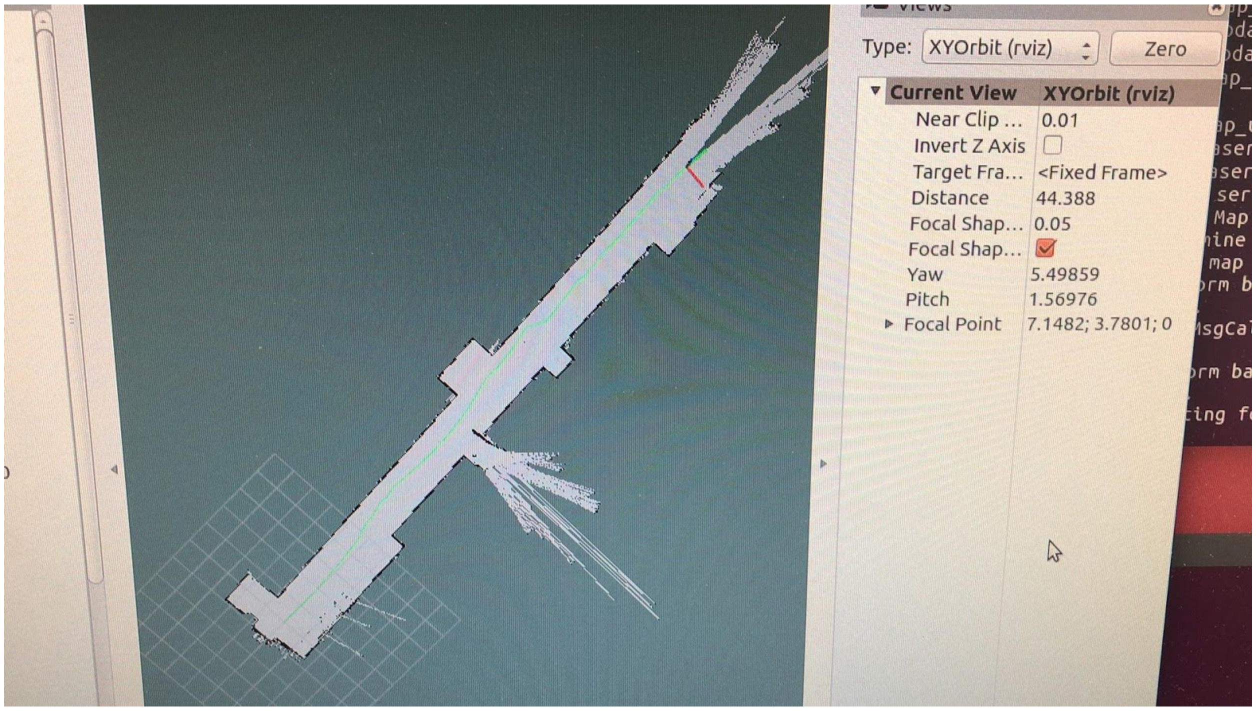


Fig. 8 map of Professor Skovira’s lab


Type: XYOrbit (rviz) Zero

Current View	XYOrbit (rviz)
Near Clip ...	0.01
Invert Z Axis	☐
Target Fra...	<Fixed Frame>
Distance	44.388
Focal Shap...	0.05
Focal Shap...	✓
Yaw	5.49859
Pitch	1.56976
Focal Point	7.1482; 3.7801; 0

Fig. 9 map of the hallway between Professor Skovira's lab and Professor land's lab.

<u>System Diagram</u>

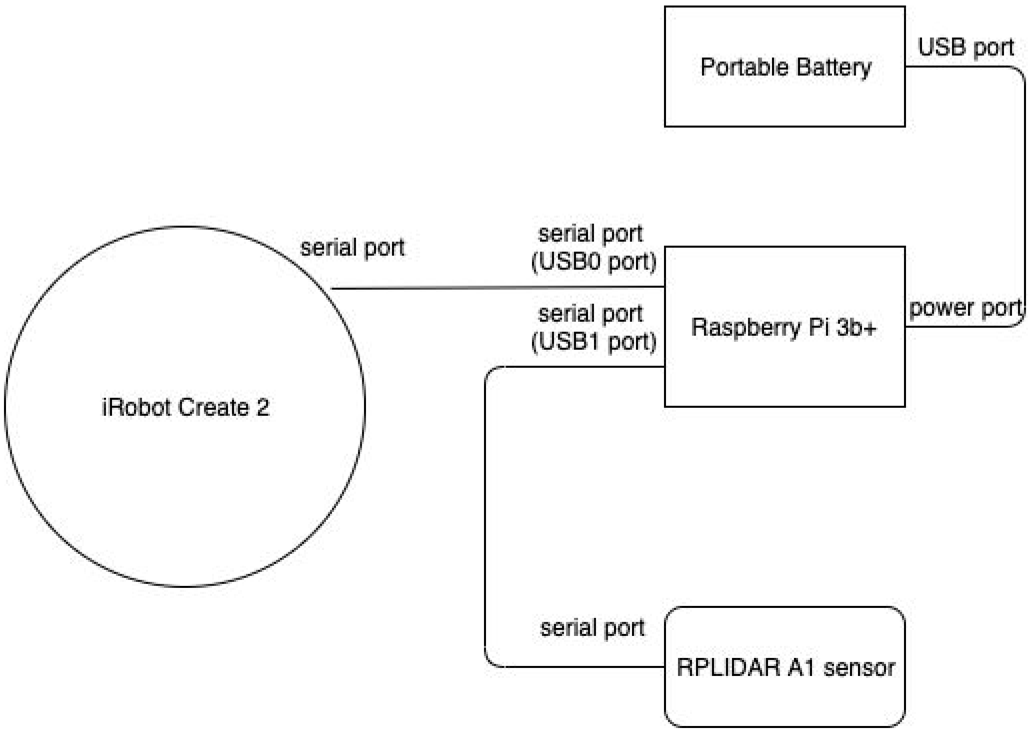


Fig. 10 system diagram of SLAM robot

On Fig. 10, it shows system wire connection. There is only cable for both iRobot Create 2 and RPLIDAR A1 sensor. I marked the USB port number on Raspberry Pi 3b+ with the USB0 port and USB1 port. The reason is that, in python code, RPLIDAR A1 sensor is defined with USB0 and iRobot Create 2 is defined with USB1. If the corresponding port is not matched. It would be an error when you run any of the code. You can only plug in the USB cable of RPLIDAR A1 sensor first, then check USB port number. If it's USB0, then you connect the MINI-DIN cable of iRobot Create2. If not, reboot Raspberry Pi and check the USB port number again. Raspberry Pi allocates USB0 for your first USB connection.

Note: use *ls -l /dev |grep ttyUSB* to list USB port numbers.
use *sudo chmod 666 /dev/ttyUSB0* to give the permission.
use *sudo chmod 666 /dev/ttyUSB1* to give the permission.

Future Work

To achieve more functionalities on iRobot Create, we need to add a controller to move the robot manually like an app instead of using a keyboard. Actually, to achieve autonomy, it's better to add motion planning algorithms in SLAM.

To achieve swarm robots of disinfection, all the maps should be uploaded to a cloud, then any robots could download a room map before disinfection. This map could be incorporated with motion planning algorithms in order to disinfect efficiently.

Another possibility would be high-level control. We could use LTL (linear temporal logic) to synthesize a controller in high-level based on different specifications. In this scenario, it would be a great experience for non-professional people to operate robots.

Contributions:

This is an individual project, I independently figured everything problem out by myself.

Appendix:

All code created for this project is available at this link: https://github.com/yanray/SLAM_robot

Bill of Materials

Material	Quantity	Price (USD)	Link
iRobot Create 2	1	199.99	
mini-Din cable	1	Included with iRobot	
RPi Kit (RPi, SD Card, charger)	1	Borrow from Professor Skovira	
Acrylic Plate	4 (12 x 12 inch)	Borrow from Professor Skovira	
Power Battery	1	Borrow from Professor Skovira	
RPLIDAR A1 sensor	1	99	https://www.amazon.com/RPLiDAR-A1M8-Degree-Laser-Scanner/dp/B07H7X3SFF/ref=sr_1_1?crid=F487EBXJMETP&keywords=rplidar+a1&qid=1576873558&sprefix=rplidaar+%2Caps%2C136&sr=8-1

In Total: the total budget including everything should be 300 - 400 USD dollars. Please be careful of everything.

References:

"iRobot Create 2 Open Interface (OI)" *iRobot,* iRobot. Jul. 19, 2018.
https://www.irobotweb.com/-/media/MainSite/Files/About/STEM/Create/2018-07-19_iRobot_Roomba_600_Open_Interface_Spec.pdf

"Raspberry Pi Tutorial" *iRobot*, iRobot. 2016.
https://www.irobotweb.com/-/media/MainSite/PDFs/About/STEM/Create/RaspberryPi_Tutorial.pdf

ROS wiki tutorial.
http://wiki.ros.org/ROS/Tutorials

SLAMTEC offical website
https://www.slamtec.com/en/Lidar/A1

SLAMTEC RPLIDAR A1 sensor Github page
https://github.com/slamtec/rplidar_sdk

RPLIDAR A1 datasheet
https://www.robotshop.com/media/files/pdf/rplidar-a1m8-360-degree-laser-scanner-development-kit-datasheet-1.pdf