United International University

Department of Electrical and Electronic Engineering

****

**Design & Demonstration of a Li-DAR Based Rescue Rover “Onnesha” for Assisting Rescue Mission**

*Authors Supervisor*

Md. Mehedi Hasan Abir (021181033) Dr. Raqibul Mostafa, Professor & Dean,

Farabi Gulandaz (021182026) School of Science & Engineering, UIU

Binta Mohammed Adib Zeta (021173013)

Tanzina Shahid Shupty (021211001)

Faiyaz Rayhan (021173023)

A capstone project in the Department of Electrical and Electronic Engineering presented

in partial fulfillment of the requirements for the Degree of

Bachelor of Science in Electrical and Electronic Engineering

United International University

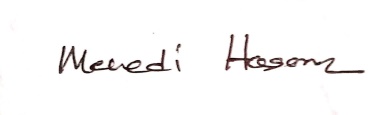
Dhaka, Bangladesh

[October, 2022]

**Declaration**

We, [Md. Mehedi Hasan Abir, Farabi Gulandaz, Binta Mohammed Adib Zeta, Tanzina Shahid Shupty , Faiyaz Rayhan], declare that this project Design & Demonstration of a Li-DAR Based Rescue Rover “Onnesha” for Assisting Rescue Missionand the work presented in it are our own. We confirm that:

* This work was done wholly or mainly while in candidature for a BSc degree at United International University.
* Where any part of this project has previously been submitted for a degree or any other qualification at United International University or any other institution, this has been clearly stated.
* Where we have consulted the published work of others, this is always clearly attributed.
* Where we have quoted from the work of others, the source is always given. With the exception of such quotations, this project is entirely our own work.
* We have acknowledged all main sources of help.
* Where the project is based on work done by ourselves, we have made it clear exactly what was done by others and what we have contributed by ourselves.



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Md. Mehedi Hasan Abir, ID: 021181033, Dept of EEE



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Farabi Gulandaz, ID: 021182026, Dept of EEE



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Binta Mohammed Adib Zeta, ID: 021173013, Dept of EEE



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tanzina Shahid Shupty, ID: 021211001, Dept of EEE



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Faiyaz Rayhan, ID: 021173023, Dept of EEE

**Certificate**

I do hereby declare that the works embodied in this project entitled **“Design &** **Demonstration of a Li-DAR Based Rescue Rover “Onnesha” for Assisting Rescue Mission”** is the outcome of an original work carried out by [Md. Mehedi Hasan Abir, Farabi Gulandaz, Binta Mohammed Adib Zeta, Tanzina Shahid Shupty, Faiyaz Rayhan under my supervision.

I further certify that this report meets the requirements and the standard for the degree of BSc in Electrical and Electronic Engineering.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dr. Raqibul Mostafa, Professor & Dean,

School of Science & Engineering, UIU

**Abstract**

In this project mapping, navigation and path planning of mobile robots in an indoor and outdoor rescue setting are investigated. The **“Design &** **Demonstration of a Li-DAR Based Rescue Rover “Onnesha” for Assisting Rescue Mission”** can be divided into three segments that work together. The rover and the control and processing unit must interact consistently for proper mapping and sensing. When the Rover is in use on the ground, each sensor has an external programmable microcontroller that sends data to the rover, which is then sent to the Control and Processing Unit through a Wi-Fi module. The user transmits an instruction to the Rover via the Wi-Fi module from the control unit. All incoming and outgoing data is processed by the processing unit.

Acknowledgement

We would like to show our gratitude to Dr. Raqibul Mostafa sir, our mentor, who helped us with this project. Additionally, we want to express our gratitude to our friends and family for their encouragement and valuable suggestions mostly on study. We want to thank the technical and support employees from the University, for their assistance. We also want to express our sincere gratitude to our teachers, who assisted us in completing our project. Also the financial help and research facilities will be provided by United International University.

Table of Content

1 [**Introduction** **1**](#_Toc78283172)

1.1 Introduction.............................................................................................................1

1.2 Objectives................................................................................................................1

1.3 Problem Statement..................................................................................................2

1.4 Block Diagram........................................................................................................2

1.5 Design Specification...............................................................................................3

2 [**Background and Literature Review**........................................................................... **4**](#_Toc78283173)

3 [**Methodology** **6**](#_Toc78283179)

3.1 Overview...................................................................................................................6

3.2 Proposed Rover.........................................................................................................7

3.3 Methodological Connections....................................................................................8

3.3.1 ESP 32 Wrover Module..................................................................................8

3.3.2 Li-DAR to TCP/IP data transfer...................................................................11

3.3.2.1 Li-DAR............................................................................................11

3.3.2.2 Li-DAR to TCP/IP...........................................................................13

3.3.2.3 Li-DAR with Servo motor...............................................................15

3.3.3 Mapping and Navigation with Li-DAR.........................................................16

3.3.4 Rover Control.................................................................................................17

3.3.5 RF Communication........................................................................................18

3.3.6 Control & Processing Unit.............................................................................19

4 [**Results**](#_Toc78283180) **20**

4.1 Mapping and navigation.........................................................................................20

4.2 Live video streaming...............................................................................................21

4.3 Cost Analysis..........................................................................................................21

4.4 Future Scope...........................................................................................................22

5 [**Evidence of Complex Engineering Problem and PO Mapping, Gantt Chart** **24**](#_Toc78283181)

[5.1 Complex of Engineering Problem](#_Toc78283182) 24

[5.2 PO Mapping](#_Toc78283183) 25

5.3 Gantt Chart...............................................................................................................27

6 [**Conclusion** **28**](#_Toc78283184)

[**References** **29**](#_Toc78283185)

[**Appendix**  **30**](#_Toc78283187)

LIST OF TABLES

[Table 1: Overall design specification of the project.](#_Toc78283188) 3

[Table 2: Cost analysis of the project](#_Toc78283189) 21

[Table 3: Program Outcome](#_Toc78283189) 25

## LIST OF FIGURES

[Figure 1: Block Diagram of the Rover](#_Toc78283190) 2

[Figure 2: Block Diagram of Li-DAR Based Rescue Rover “Onnesha”](#_Toc78283191) 6

[Figure 3: Block Diagram of “Onnesha” with modules & sensors](#_Toc78283190) 6

[Figure 4: The rover ”Onnesha”](#_Toc78283190) 7

[Figure 5: ESP 32 Wrover Module](#_Toc78283190)8

[Figure 6: Schematics of ESP32 ¬Wrover](#_Toc78283190) 8

[Figure 7: Connection Diagram ESP32 – FTDI programmer serial port](#_Toc78283190) 9

[Figure 8: IP address of the network in serial port](#_Toc78283190) 9

[Figure 9: Live video streaming display](#_Toc78283190) 10

[Figure 10: TF Mini Li-DAR connection and pin diagram](#_Toc78283190) 12

[Figure 11: Figure 3.11: Output data from TF mini Li-DAR](#_Toc78283190) 13

[Figure 12: Figure3.12: Arduino Uno with TF mini Li-DAR](#_Toc78283190) 13

[Figure 13: Arduino uno to ESP8266 Node MCU setup](#_Toc78283190) 14

[Figure 14 Virtual Com port:](#_Toc78283190) 14

[Figure 15: Output data from Li-DAR and servo motor](#_Toc78283190) 15

[Figure 16: Graphical User Interface (GUI) of the RADAR for 0 to 180 degrees](#_Toc78283190) 17

[Figure 17: BTS motor driver](#_Toc78283190) 18

[Figure 18: RemoteXY coding Platform](#_Toc78283190) 18

[Figure 19: Block diagram of the Radio frequency communication block](#_Toc78283190) 19

[Figure 20: Real life environment](#_Toc78283190) 20

[Figure 21:Mapped environment](#_Toc78283190) 20

[Figure 22:Live video of surroundings](#_Toc78283190) 21

[Figure 23: 3D mapping on sea using 3D Li-DAR](#_Toc78283190) 22

1 Introduction

# 1.1 Introduction

A mobile robot can be developed in a variety of ways and utilized for a variety of purposes, including security, medical and surgical applications, industrial tasks, and rescue work, among others. One of them is navigation and mapping, which is critical for rescue efforts in a perilous environment. The suggested project’s goal is to aid rescue operations by using a live video streaming server to map and sense persons or other living entities. A control and processing module, a Li-DAR module, an ESP 8266 Wi-Fi module, an ESP 32 camera module, and an audio transceiver are all included in this article. The rover’s position is estimated using Li-DAR and a gyroscope, and a map of the surroundings is created at the same time to estimate path planning. Positioning and path planning can be performed in any rescue situation using Li-DAR Gyroscope synchronization [1]. Object detection, often known as human body detection, is the most major element of any rescue operation in the event of a natural or man-made disaster. This rover is used to detect objects in a certain area. For object detection, this rover utilizes Li-DAR technology in combination with an ESP 32 camera module. The rover uses Li-DAR and a gyroscope to survey the surroundings and calculate the distance between the object and the rover. All of the sensors, including Li-DAR and ESP 32 camera module, will collect and send data to the control and processing unit. All of the data collected by the rover’s sensors will be wirelessly sent to the control and processing module through Wi-Fi. The rover’s incoming and outgoing data will be processed simultaneously by the control and processing module. Humans will operate the control and monitoring system, which will be software-based. The proposed project is versatile enough to assist in Geographical Survey as well as other disciplines of research, regardless of the fact that the rover is designed for rescuing purposes. Additional features for the rover can be added thanks to the development of portable and compact IoT devices.

**1.2 Objectives**

Explosions, floods, storms, earthquakes, tornadoes, fires, and many more unfortunate man-made or natural occurrences are to blame for human fatalities and physical harm. Mobile robots can navigate hazardous locations for rescue operations and carry out intelligent tasks including localization, mapping, and navigation. When human mobility is unsafe or impractical, it can aid emergency responders. Additionally, this rover has the ability to navigate its surroundings and carry out intelligent tasks on its own, opening up a wide range of realistic applications.

1 Introduction

**1.3 Problem Statement**

Rescue operation during a major natural disaster such as an earthquake or an unintentional man-made disaster of similar nature is of utmost importance to save lives and help trapped victims. From the perspective of Bangladesh these situations are legitimate threats and of grave concern as they involve people’s life, life-long injuries and traumas. It’s been 8 years but we still remember the collapse of Rana Plaza on 24th of April 2013. The complex housed five garment factories, and tragically took the lives of at least 1,132 people, and more than 2,500 were injured. We also heard about the burning of Tazreen Fashions factory which occurred just five months before than the Rana plaza tragedy. And, recently on the 4th of June 2022, a fire broke out which led to explosions at a container depot in Chittagong where at least 41 people died, and more than 450 were injured. The frequency and lethal ferocity of such disastrous events created an urgency for any possible help that could assist the rescue situation. A mobile rover is able to get into these dire places where people cannot. It can map the environment, detect living/non-living essential objects and can find safe routes in and out of the place. The rover is also equipped with a camera module and an audio system which enables live video stream and audio communication between the victim and rescuer. And, all the features are connected wirelessly between the Rover and the user’s control unit over the local network. Although, it has been designed to assist rescue operations, still the rover is flexible enough to assist in Geographical Survey along other fields of research.

**1.4 Block Diagram**

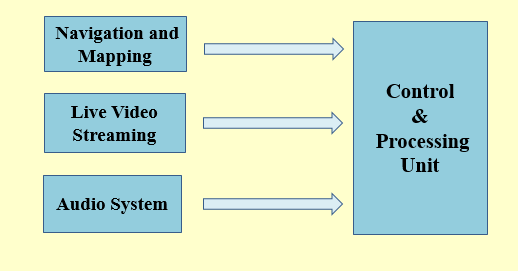


Figure 1.1: Block Diagram of the Rover

1 Introduction

**1.5 Design Specification**

Table 1.1: Overall design specification of the project.

|  |  |
| --- | --- |
| Design Specification | |
| Function | The function of the rover is to collect distance data from the surroundings and map the environment also stream live video and build half duplex communication between victim and rescue team. |
| Target Community |  |
| Material | We need Li-DAR for mapping, live video camera module, microcontroller, Wi-Fi module ect. |
| Size | The size of the rover is compact to reach such a place where human can’t reach. |
| Power Source | Here we use a 9V battery for powering the rover. |
| Cost | It costs arround Tk 16000. |

Chapter 2

# Background and Literature Review

Several novel techniques and algorithms have been introduced in recent studies. For each new sensor, the majority of these techniques and algorithms repeat localization and incremental mapping to create maps. If a map is easily available, navigation and mapping are also simple. The algorithms have already been shown to work for these localizations in a few papers. Here our focus is on Li-DAR detection along with a real-time video streaming server to detect humans or any other living beings. This paper presents an algorithm that combines ideas of 2D Li-DAR-based navigation and mapping as well as detecting objects or the human bodies with real-time video coverage. We present a monitoring robot with real-time video streaming, audio transfer, and the capacity to avoid obstacles in the process in this study. The system will be built up in such a way that the footage will be streamed live to the person in charge of the robot. The rover was controlled through a web server to include wireless communication into the proposed technique where we used the NodeMCU and ESP Module. The works and projects by researchers around the world studied for our project are provided here:

* In 2020, some researchers have implemented 2D Li-DAR based SLAM 90 and path planning indoor rescue mobile robots where they have used SLAM algorithm. They focused mainly on SLAM algorithms combined with path planning. [1]
* In 2018, four researchers have designed and implemented ROS based indoor positioning and navigation system of mobile robots where they focused on positioning using ROS operating system. [2]
* A report publish in 2016, real time loop closure in 2D Li-DAR SLAM is studied. Where they successfully achieve real time loop closure by using branch-and-bound approach for computing 2D Li-DAR data. [3]
* Simultaneous map building and localization in a mobile robot is a significant problem. To build an accurate map, a work has been done in 1991 to minimize the difficulty of precise mapping by a mobile robot. They used multiple sonar sensors so that the environment can be precisely learned and mapped. [4]
* In July 2004, a simultaneous localization and mapping algorithm is proposed for an autonomous vehicle is studied and they used Li-DAR sensor for detecting the environment. By using Kalman filter they corrected the map and the vehicle position as well. [5]
* Based on multi sensor fusion, robot mapping and navigation is studied in 2021. They mainly focused on the improvement of low accuracy of 2D grid map. They used 2D Li-DAR, Jetson TX2 development board, combining Bayesian estimation data fusion method to improve the mapping accuracy. [6]
* In 2021, a paper studied comparison among three algorithms (GMapping, Hector-SLAM, Google Cartographer) used for 2D mapping and localization in an indoor environment. [7]

Chapter 3

# Methodology

# 3.1 Overview

At the beginning a quick overview of the block diagram will help us to understand the system and the steps we have followed to improvised the system.

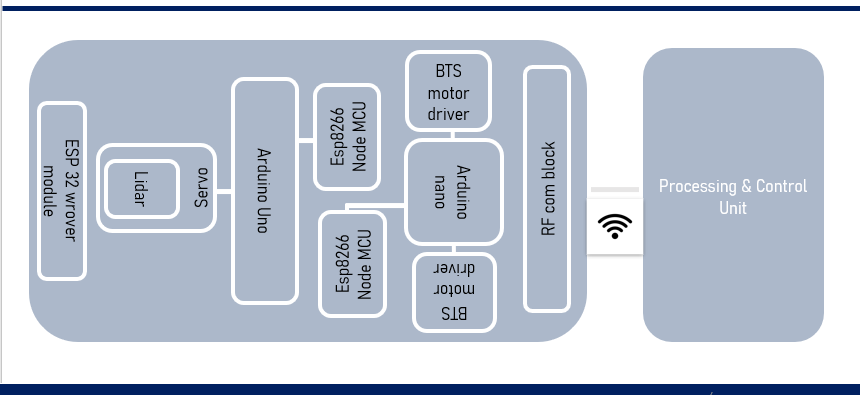


Figure 3.1: Block Diagram of Li-DAR Based Rescue Rover “Onnesha”

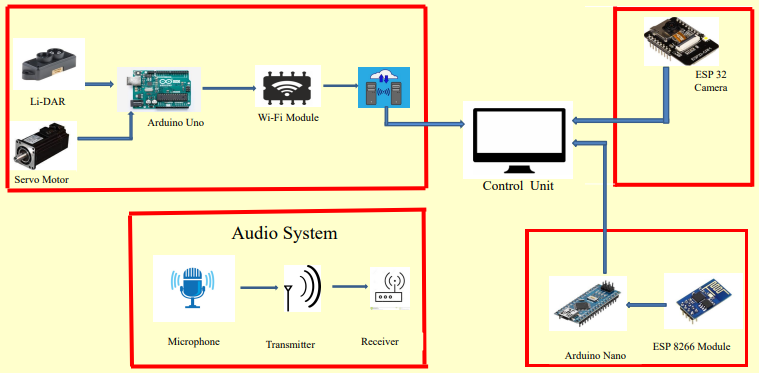
****

Figure 3.2: Block Diagram of “Onnesha” with modules & sensors

3 Methodology

Through the block diagram we can observe five major blocks for Li-DAR Based Rescue Rover “Onnesha” are as follows,

On Board Rover integrated blocks:

1. ESP32 Wrover module with camera.
2. Lidar to TCP/IP data transfer block through Arduino uno & esp8266 Node MCU.
3. Environment mapping using Li-DAR
4. Rover control block with Arduino nano and esp8266.
5. Radio Frequency communication Block.
6. Processing & Control Unit Block (Outside the rover)

Firstly, we will discuss about individual operations briefly including test data analysis for those blocks where necessary then we will integrate the whole system at the end of the discussion.

**3.2 Our Proposed Rover with Specifications:**

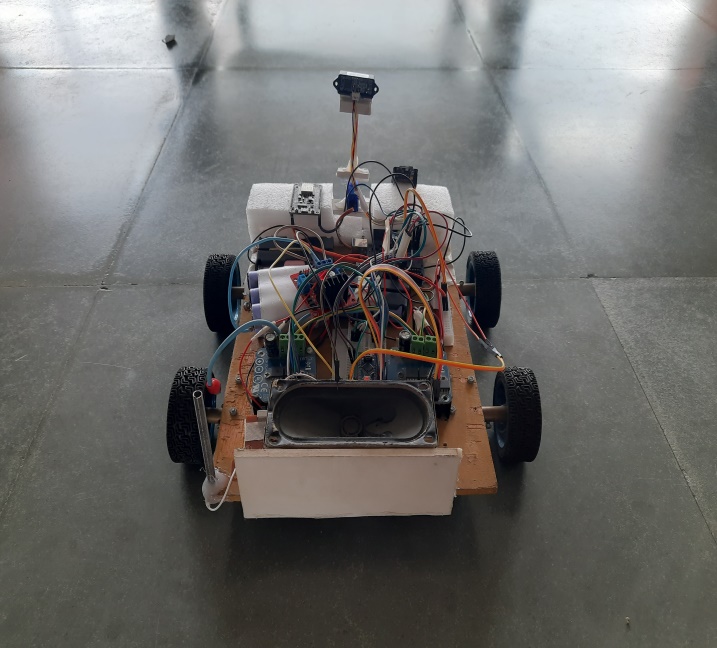
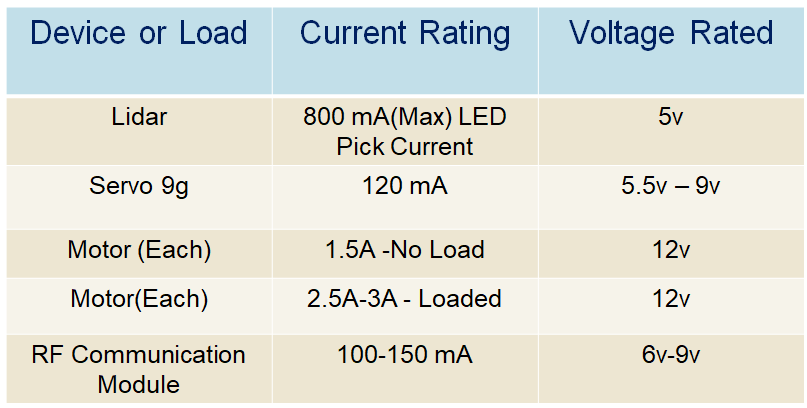


Figure 3.3: The rover ”Onnesha”

****

3 Methodology

* 1. **Methodological Connections**

**3.3.1 ESP32 Wrover Module with camera:**

Integrated camera module with ESP32 Wrover has protocols of 802.11 b/g/n (802.11n up to 150 Mbps) Standard including central frequency range for the operating channel in between 2412 ~ 2484 MHz with on board build in antenna as well as gives the opportunity to extend the range of the module through external antenna connector.



Figure3.4: ESP 32 Wrover Module

Like Arduino & other AVR microcontrollers, ESP32 Wrover module can also be codded through Arduino IDE. It comes with a package of code with supporting header file to initialize the program as GPIO (General Purpose Input Output) pins from the esp32 cheap also needed to be initialize along with the code to make the camera operational.

As we are not going to give a deep dive into the manufacturing process or the working principals of the esp32. So, a quick overview of the esp32 GPIOs Schematic will be enough for us to proceed on words as codes including header files will be added in the appendix section.

3 Methodology

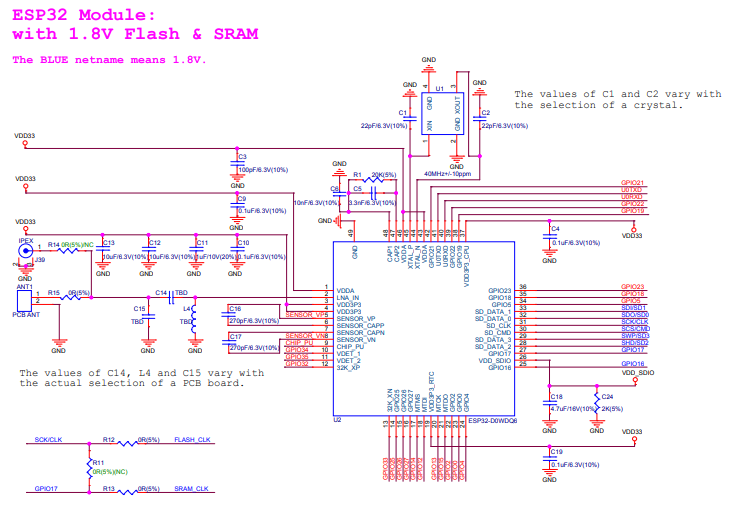


Figure3.5: Schematics of ESP32 ­Wrover.

Getting ready with the codes we need to make changes in SSID & Password, as ESP32 cam module will be connect through Wi-Fi router to the pc. Coeds in Arduino would like similar to this,

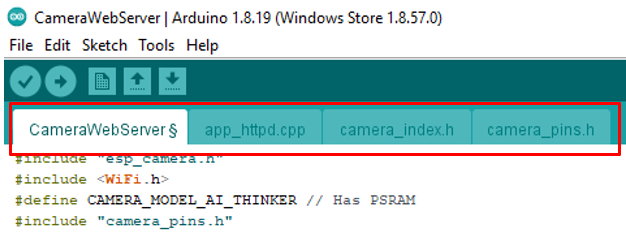


Figure3.6: Arduino IDE code for ESP32 including header file. (Codes in Appendix)

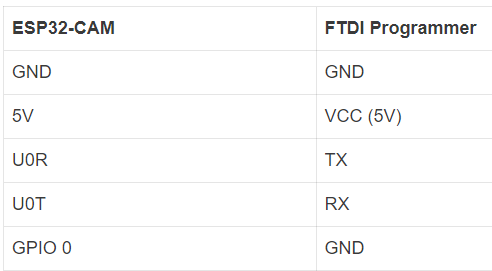
Where to upload the code the connection diagram will be look like this,



Figure3.7: Connection Diagram ESP32 – FTDI programmer serial port.

3 Methodology

Pin Setup



* We need to go to Tools > Board and select AI-Thinker ESP32-CAM. Moreover, we must have the ESP32 add-on installed
* Then to Tools > Port and select the COM port the ESP32-CAM is connected to.

At the end now if we open our serial port associates with the FTDI, we will get to see the IP address connected with the respective network as follows, <http://192.168.1.91>

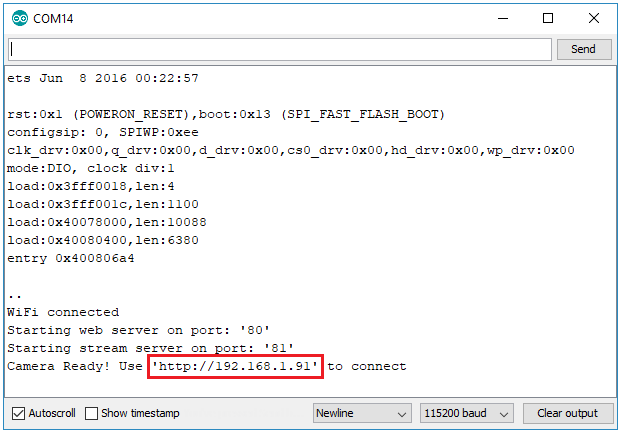


Figure 3.8: IP address of the network in serial port

3 Methodology

Finally, we are all set to go. After opening any browser connected to the same network if we browse the IP address we have gotten from the serial pot, the video will be started stimming.

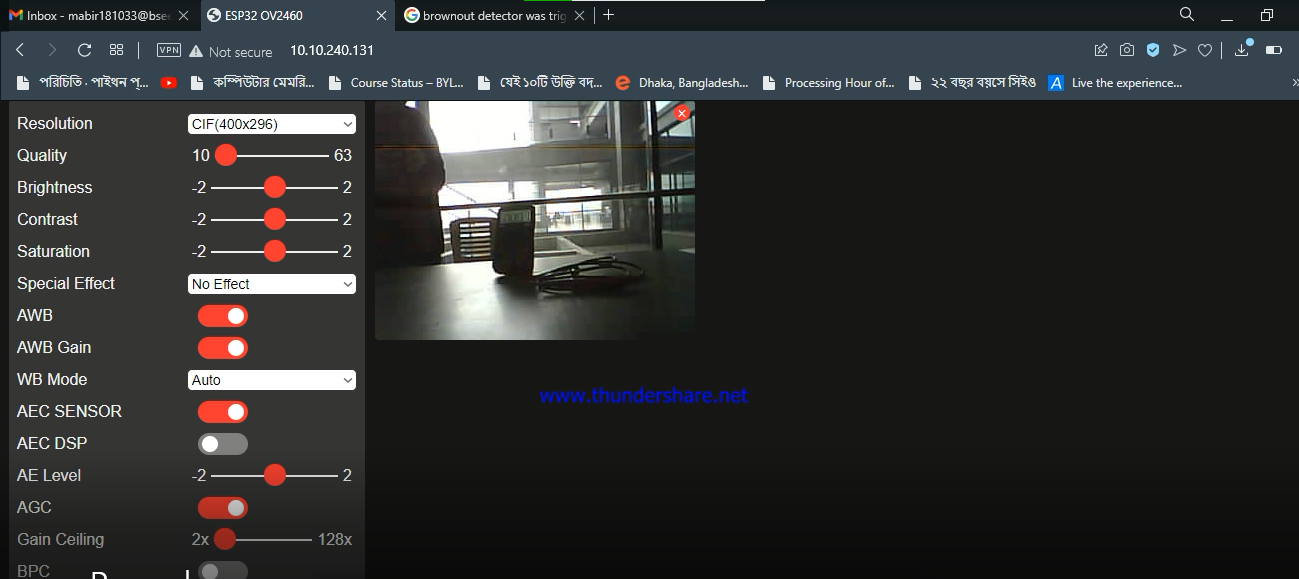
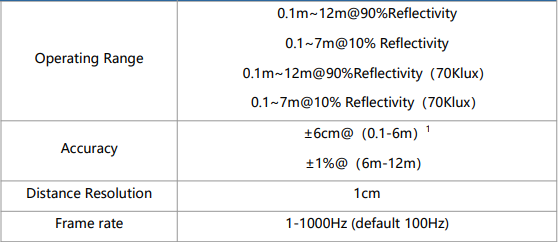


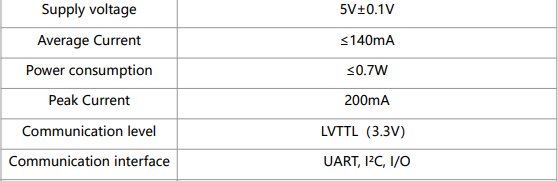
Figure 3.9: Live video streaming display

* + 1. **Lidar to TCP/IP data transfer block through Arduino uno & esp8266 Node MCU**
       1. **Li-DAR**

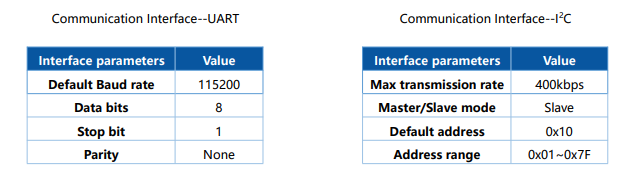
Li-DAR in abbreviation it is light detection and ranging, here we are using TF mini Li-DAR which is a single point time of flight lidar, it sends laser light if the object is in its range, it captures the reflected light to measure distance.it use 850nm Wave length with 2⸰ FOV.

For our project we are using TF mini lidar from **Benewake**, some keys specifications are as follows,





**Communication Interface:**



TF mini Li-DAR connection and pin diagram:

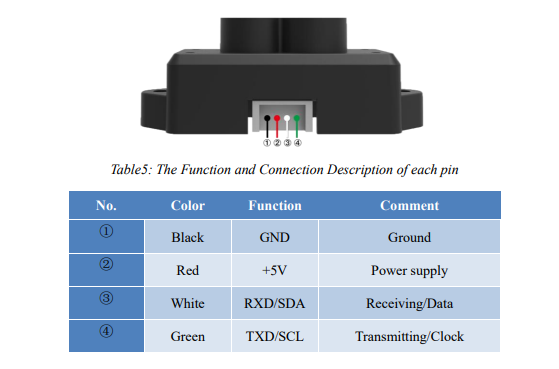
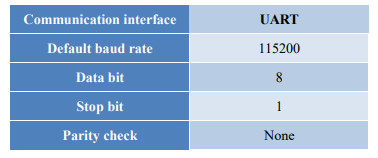


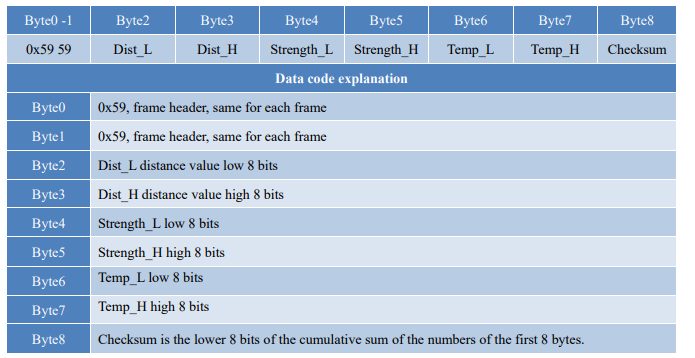
Figure 3.10: TF Mini Li-DAR connection and pin diagram

TF mini lidar communication Specification for UART protocol:



3 Methodology

Data is sent to in Hexadecimal format in default baud rate of 115200 for 8 bits data to interpolate data from the bits,



To get the distance data we need to code and burn that to Arduino with specified bits Finally after coding at Arduino IDE we can have output,

Specified bits to interpolate data to distance along with signal strength,

After uploading code to Arduino uno for the desired serial port with lidar now if we have to open up the serial monitor and select the baud rate 115200 and serial monitor will communicate with lidar through Arduino to display the Distance & signal strength,

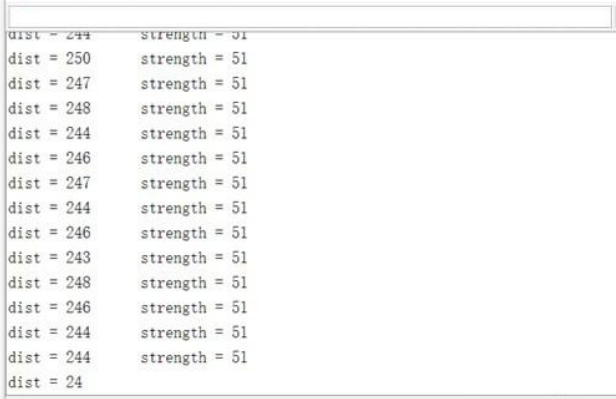


Figure 3.11: Output data from TF mini Li-DAR

3 Methodology

**3.3.2.2 Li-DAR to TCP/IP Data Transfer**

Now we are all set to transfer the data to the esp8266 Node MCU before proceeding the connection diagram of TF mini lidar with Arduino

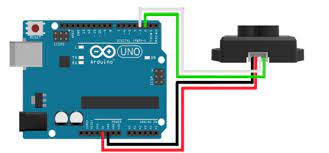


Figure3.12: Arduino Uno with TF mini Li-DAR

We could have used Lidar with esp8266 but as baud rate of esp8266 is 9600 and baud rate of lidar is 115200 so we had to switch to Arduino uno, Esp8266 Tx & Rx need to connect with Arduino’s Rx & Tx respectively, Tx-Rx & Rx-Tx as we will be using hardware serial port to communicate.

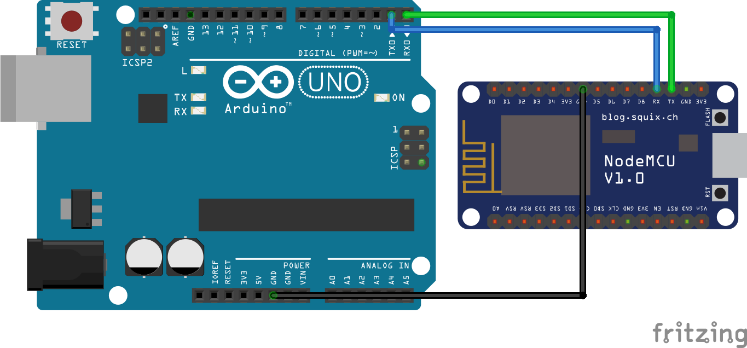


Figure3.13: Arduino uno to ESP8266 Node MCU setup

After doing that we have codded our esp8266 to read data from the serial port along with a TCP/IP protocol to communicate. Code to generate TCP/IP and to print data which we are reading from serial port is as follow

We need to add a buffer here as well as because data coming from Arduino hardware serial port is not a complete package rather it transmits per bit.

3 Methodology

After that as before we did if now the pc and the esp8266 is connected with the same network it will give an IP address as well as port to get the client connected now if the client is connected it will start transferring data.

The key reason behind choosing TCP/IP protocol is that it is asynchronous so we don’t need to knock every time to transmit the data whenever the data will be available the transmission starts. As reading data and to work with it from TCP/IP is not that easy we have created virtual comport from TCP/IP to read and work with that information in code.

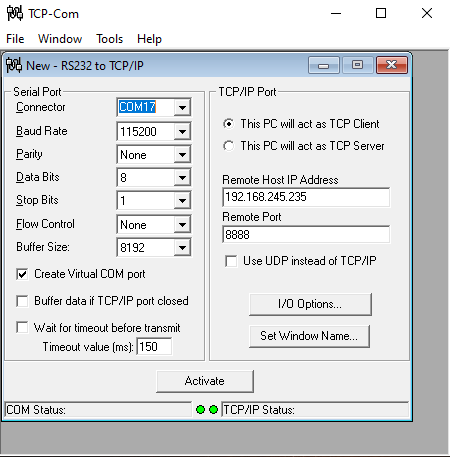


Figure 3.14: Virtual Com port

Selecting the baud rate and giving respective IP address and port we need to select a comport now if we activate the TCP-Com it will create a virtual comport to our pc.

Now again we can have access to data from our Arduino serial monitor by selecting the virtual comport number as created before.

* + - 1. **Integration of Li-DAR and Servo motor:**

The Rover must map its frontal environment for navigation. In order to do that, the Lidar must move on its own axis to pick up data for plotting. So, a servo motor is used and programmed to rotate from 0 to 180 degrees. Lidar is set up on top of the Servo Motor in such a way that its wires connected to Arduino does not get tangled. The diagram below shows the particular hardware setup of Lidar on a Servo Motor connected to an Arduino.

The integration of the Lidar and Servo on hardware is complete. And after uploading the code to the Arduino Uno, open the available serial port. When we open the serial monitor and select the baud rate 115200, the serial monitor will communicate with the system

through Arduino Uno. The Serial Monitor shows the distance the Lidar is picking up at each degree when the servo motor is rotating from 0 to 180 degrees.



Figure 3.15: Output data from Li-DAR and servo motor

* + 1. **Li-DAR Mapping and Navigation**

Before we transport the data from the above created TCP-Com i.e. virtual com port to Python, it is important to understand the nature of the data. Let’s recall the hardware setup, where the Lidar was set-up on a servo motor which rotates from 0 to 180 degrees. The incoming data in the TCP/IP is a distance that the Lidar is picking up at a certain degree. So, we have a pair of data (a degree, a distance) at every instant. This stream of data has to be accessed from the virtual com port and plotted in a suitable environment.

Many tools and software are available for plotting coordinates data, many gave appreciable results, but we aspired to create the perfect mapping environment. Therefore, we proceeded with Python. Python offers a special Python package which is especially designed to build maps of environments. PyCharm is the best IDE for compiling Python programs of this specific kind, that is, where a graphical map has to be created.

3 Methodology

A step by step explanation of how PyCharm creates the mapping environment is discussed below.

Firstly, a Graphical User Interface (GUI) of the RADAR for 0 to 180 degrees is created using a Python library called ‘matplotlib’. The ‘matplotlib’ is a comprehensive library for creating static, animated, and interactive visualizations in Python. The RADAR GUI in by PyCharm is shown below.

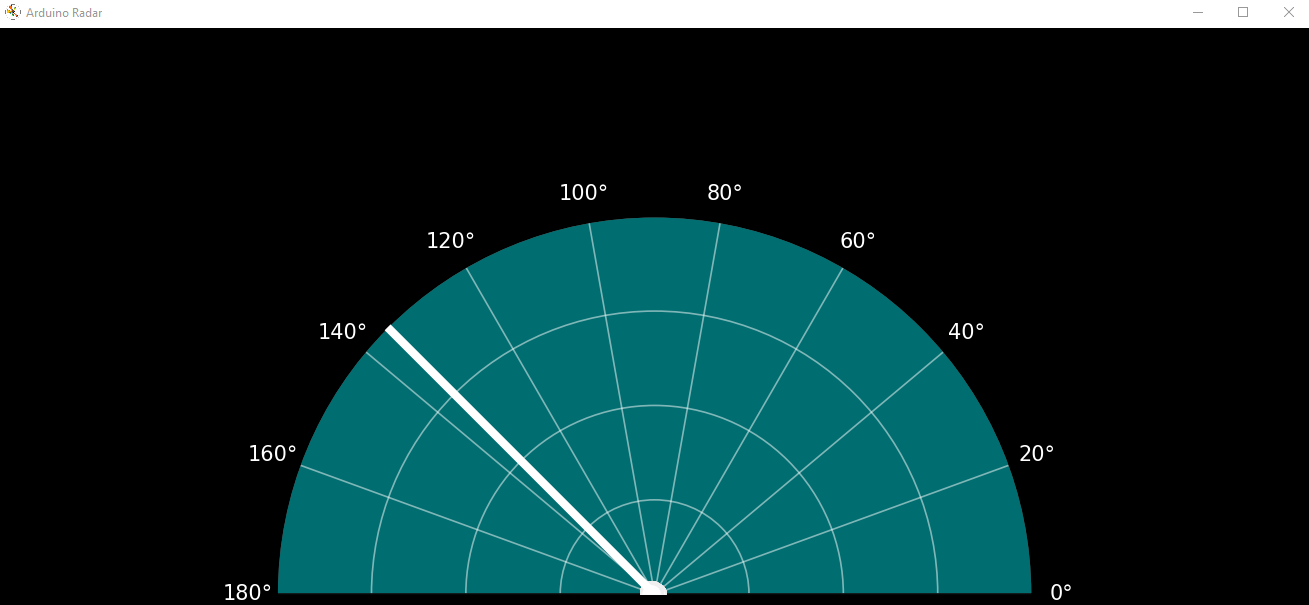


Figure 3.16: Graphical User Interface (GUI) of the RADAR for 0 to 180 degrees

Secondly, remember the stream of data continuously incoming in the TCP/IP. Now, we can program to say- find the Arduino ports, select the active port, and start communication with it. This is done using the Python library called ‘pyserial’. The ‘pyserial’ library encapsulates the access for the serial port and provides backend for Python running on operating systems such as Windows, OSX, Linux etc. So, the PyCharm IDE can now access the TCP/IP, fetch the data, as it is, and put those data coordinates on the GUI that has already been created.

Finally, the system requires a method to clean the already populated GUI after a few iterations. An infinite loop removes some of the previous data points and constantly updates the GUI radar after every 180 degrees with new incoming Arduino data.

Note: We are not updating the hardware data in any way nor processing it. The data picked up by the Lidar set-up on the motor, stays as it is. Python is only used to plot that stream of absolute real data from the TCP/IP to build an almost accurate map of the real environment.

3 Methodology

**3.3.4 Rover Control Block with Arduino Nano and Esp8266:**

Controlling rover under the same network is very important, in our design we have used Arduino Nano along with ESP8266 to BTS796o motor driver.

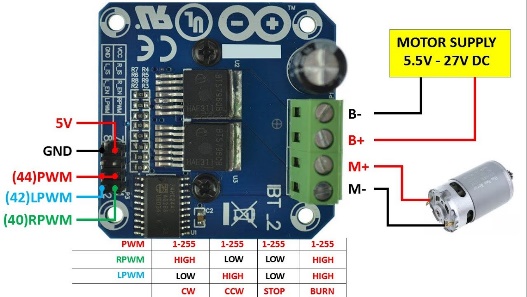


Figure 3.17: BTS motor driver

Connection of BTS with Arduino nano is specified in code, as we have design out rover to control through wi-fi network now we will be creating a hotspot through ESP8266 as well as we will create a source code format from RemoteXY platform and put our main code in it with specified format to generate hotspot and as well as mobile app to control our rover.

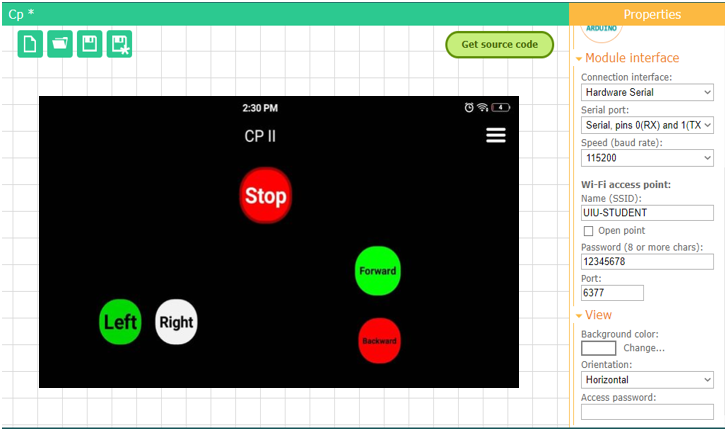


Figure3.18: RemoteXY coding Platform

Arduino nano, esp8266 and type of network is selected now we need to design GUI through this platform for user interface. Installing mobile app (RemoteXY) now if we get connected to our generated network hotspot, we will get to see the same GUI as we have designed and uploaded to the Arduino Nano. And here it is the end of our control system design. So now if we powerup the whole system and we will be all set to go.

3 Methodology

**3.3.5 Radio Frequency communication**

AM transmitter module was used here to communicate through our rover. It’s a half duplex model which means that data can be transmitted in both directions on a single carrier, but not at the same time. But in our rover , It’s a one way commiunication which means we can only send data to the receiver side but the receiver side can’t send any data to the sender side.

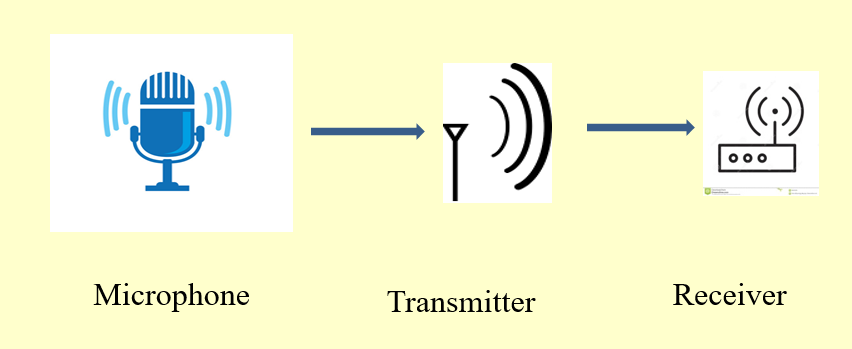
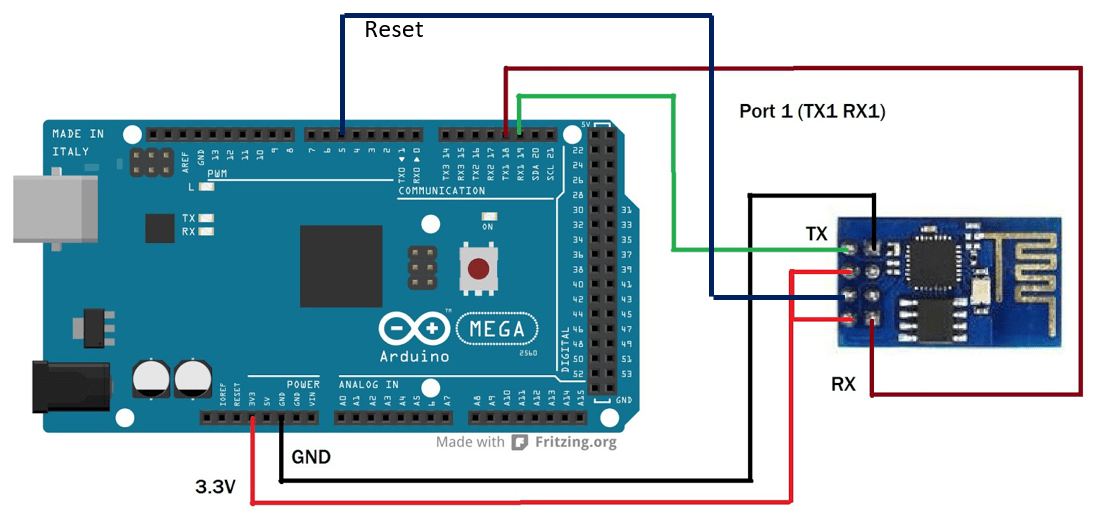


Figure 3.19: Block diagram of the Radio frequency communication block

It has a low frequency system which means it has higher pitch. It also has a higher Amplitude and wavelength, which means it has louder sound which can travel through all directions. It has a range of 150 to 200 meters. But it’s range can improve if we use a larger antenna.

**3.3.6 Processing & Control Unit**

Arduino has been used here along with BTS 7960 Motor driver and has been controlled over the command from developed app through ESP 8266 Wi-Fi module.



Chapter 4

# Results

# 4.1 Mapping Output of the Rover

The figures below show the picture of a real environment and then a GUI created by Python. Without any doubt it can be said that the coordinates plotted on the GUI maps the environment accurately. And, the Rover can navigate based on this map which is continuously updating. This is how the most essential part ‘Mapping & Navigation’ is achieved in the project.

# 

Figure 4.1: Real life environment

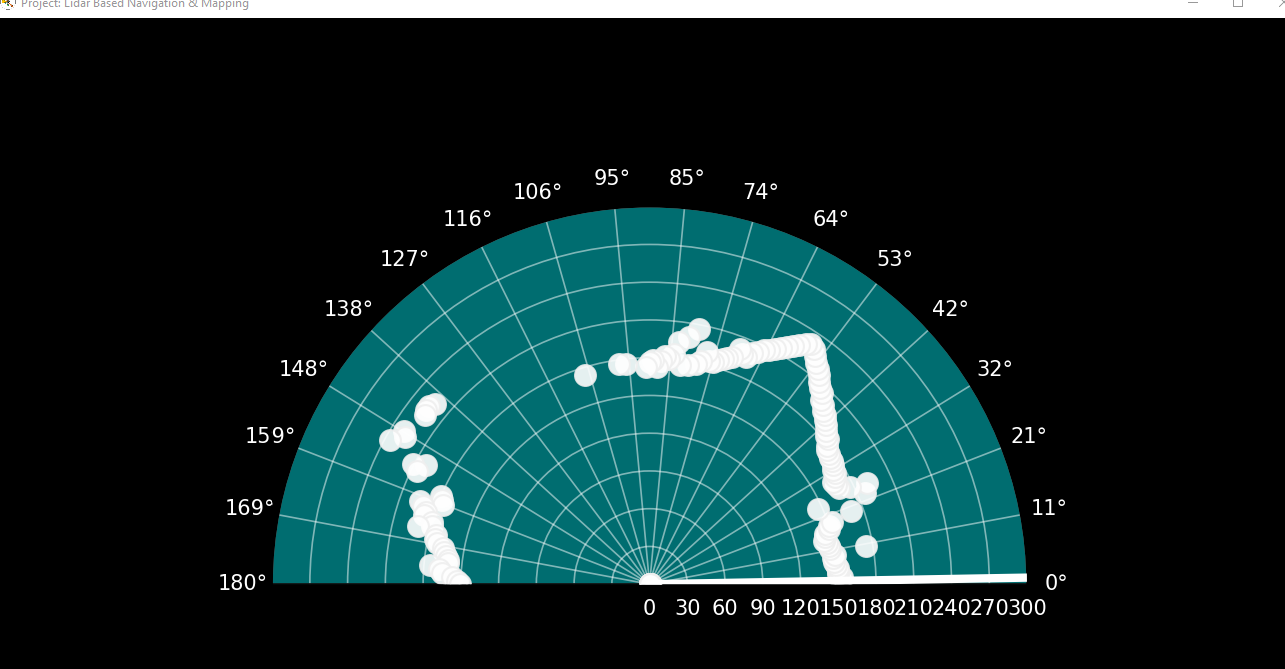


Figure 4.2: Mapped environment

4 Results

# 4.2 Live Video Output of the Rover

After opening any browser connected to the same network if we browse the IP address we have gotten from the serial pot, the video will be started stimming.

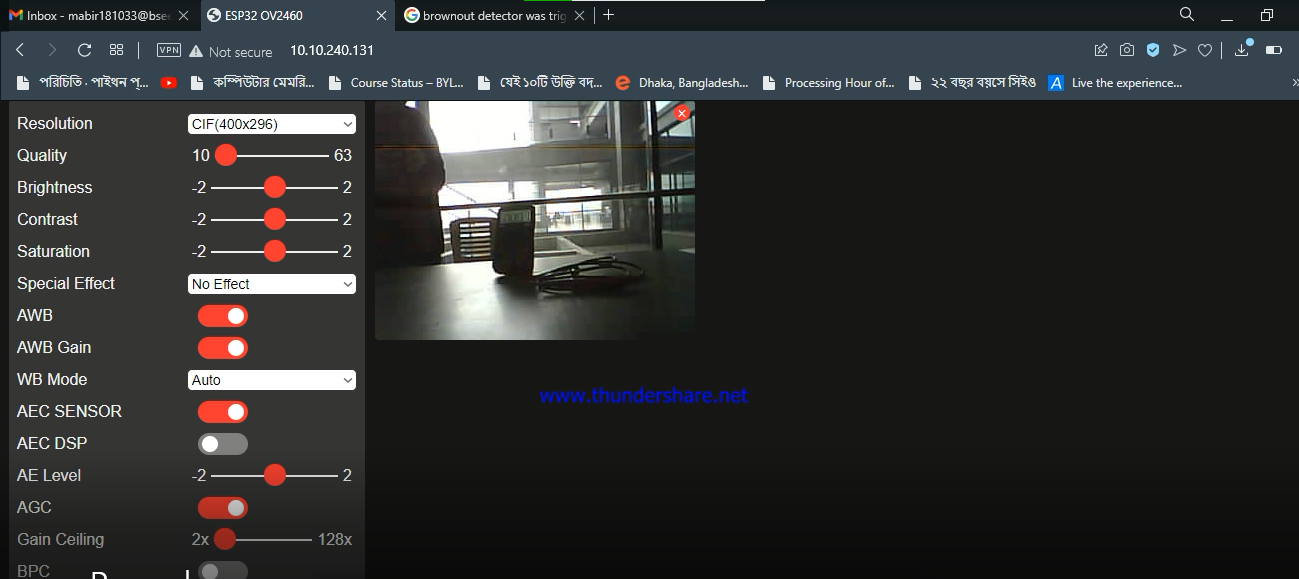


Figure 4.3: Live video of surroundings

# 4.3 Cost Analysis

Table 4.1: Cost analysis of the project

|  |  |
| --- | --- |
| Component | Price (Tk) |
| Li-DAR | 5000 |
| Arduino Uno & Nano | 2000 |
| ESP 32 Wrover & ESP 8266 | 1500 |
| Motor Drivers & Motor | 4500 |
| Communication System (RF) | 1000 |
| Power Supply | 1000 |
| Others | 2000 |
| Total | 17,000 |

4 Results

# 4.4 Future Scope

As for the future, there were many areas where we had limited resources and also have rooms for much improvement. As for improvement, we could have used gyroscope which would give us the data for maintaining orientation and angular velocity. It also gives the precise location of the rover where GPS can’t help us. Again, by calculating angular velocity, it will give us the x,y and z axis of the location and with that we would be able to know whether our rover is facing the correct direction.

Secondly, we used 2D lidar for mapping. If we had used 3D Lidar, it would have given us much precise accuracy. Although, a 3D lidar is costly, but it would give the most precision mapping.

Since, we have shown the result of 2D TF Mini Lidar, here is what 3D Lidar’s result looks like.

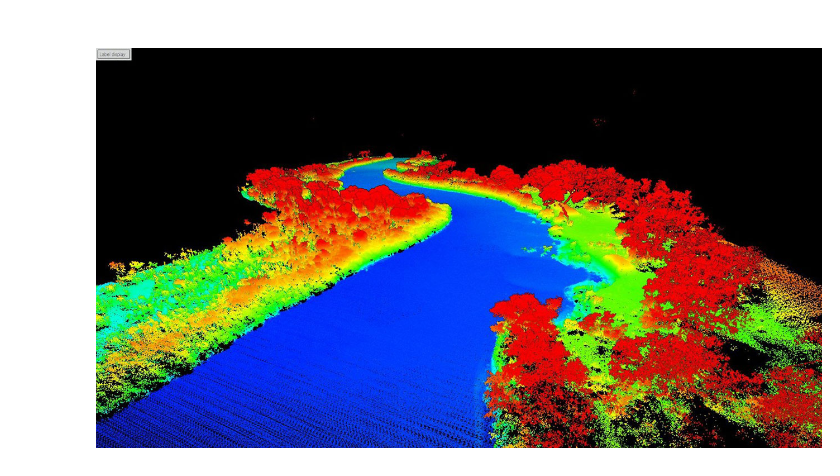


Figure 4.4: 3D mapping on sea using 3D Li-DAR

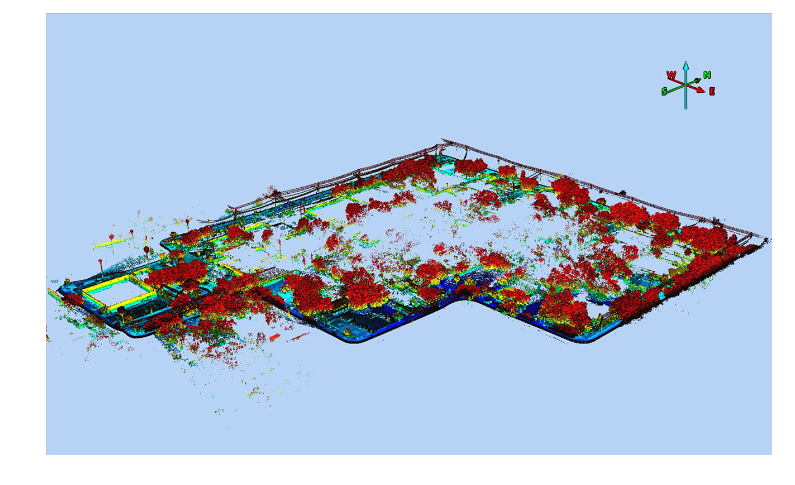


Figure 4.5: 3D mapping on land using 3D Li-DAR

This is how 3D mapping looks like from drone point of view. It certainly shows far better result than our 2D lidar.

As for limitations, we used PyCharm IDE for compiling Python code to create an Graphical User Interface (GUI) for mapping, Arduinos for incoming lidar data and rover control and lastly used TCP.com for sending Lidar data to tcp/ip. If there were a software which could do all the integration in a single environment, the system would have been less complex and would have saved a lot of time and money.

Chapter 5

# Project Attributes

## 5.1 Complex of Engineering Problem

Throughout the project we faced many challenges, we overcame some of the challenges, but we had to look for new solutions for the project in order to make the project successful.

The complex engineering challenges we faced are as follows,

**I.** In the beginning we tried to work directly with esp8266 but it didn't work with lidar due to the lower baud rate of ESP8266 then we had to go for Lidar-Arduino-ESP8266 to make it fruitful.

**II.**The next challenge we faced while working with lidar data format, as lidar has a Higher Baud rate of 115200 we had to suit a lot to put the data in a specified format which is most commonly described as "Angle, Degree" format to get the perfect picture view of the environment.

**III.**Thirdly we were thinking of working with a web server from the very beginning and in the initial stage of project design but as the webserver is synchronous we had to face a lot of trouble loading data to an HTML webpage through ESP8266 with a refresh rate as well as it took a lot of time from our total work but after all switching to asynchronous TCP/IP-Virtual Comport was a wise decision from our team.

**IV.** At the end we tried adding some extra features like RF communication and Navigation with MPU 9250 9-axis Gyroscope. Although We could only work with RF communication due to time limitations still further development of the project can be done with MPU 9250.

5 Project Attributes

**5.2 Gantt Chart of the Project**

Our approximate work planning map is shown in this section by using a Gantt chart for three trimesters.

Gantt chart for capstone project I

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Week | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | | 9 | 10 | 11 | | 12 | 13 | 14 |
| Group forming |  | |  |  |  |  | MID |  | |  |  |  | |  |  | FINAL |
| Mentor selection |  |  |  |  |  |  |  | |  |  |  | |  |  |
| Planning |  |  |  |  | |  |  | |  |  |  | |  |  |
| Specification |  |  |  |  |  |  |  | |  |  |  | |  |  |
| Mid |  |  |  |  |  |  |  | |  |  |  | |  |  |
| Research |  |  |  |  |  |  |  | | |  |  | |  |  |
| Design |  |  |  |  |  |  |  |  | |  |  | |  |  |
| Report writing |  |  |  |  |  |  |  |  | |  |  | | |  |
| Presentation |  |  |  |  |  |  |  |  | |  |  |  | |  |

Gantt chart for capstone project II

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Week | 1 | 2 | 3 | 4 |  | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| Analysis |  |  |  |  |  |  |  | MIDMID |  |  |  |  |  |  | FINAL |
| System implementation |  |  | |  |  |  |  |  |  |  |  |  |  |
| Implementation development |  |  |  |  |  | | |  |  |  |  |  |  |
| System construction |  |  |  |  |  |  |  |  | | |  |  |  |
| Report writing |  |  |  |  |  |  |  |  |  |  |  | |  |
| Presentation |  |  |  |  |  |  |  |  |  |  |  |  |  |

Gantt chart for capstone project III

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Week | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| Assessment |  | | |  |  |  | MID |  |  |  |  |  |  | FINAL |
| Completion |  |  |  |  | | |  |  |  |  |  |  |
| System checking |  |  |  |  |  |  |  | |  |  |  |  |
| Evolution |  |  |  |  |  |  |  |  |  | | |  |
| Quality checking |  |  |  |  |  |  |  |  |  |  |  |  |

5 Project Attributes

## 5.3 PO Mapping

Table 5.1: Program Outcome

|  |  |  |
| --- | --- | --- |
| PO1 | **Engineering Knowldge** | We use knowladge of Mathematic, Natural Science, Programming (Python), Microcontroller, Communication and science of engineering. |
| PO2 | **Problem analysis** | By completing this project we are capable of describing the works, identifying issues, limitations and ideas for solutions |
| PO3 | **Design / development of solutions** | We are able to identify and use suitable parameters, assumptions and design criteria taking safety, economics and sustainability into consideration. |
| PO4 | **Investigation** | We evaluate every step of the process to determine which parts we can obtain. |
| PO5 | **Modern tool usage** | This project is mainly hardware-based. We use the ESP8266 Wi-Fi module. And we integrate Li-DAR with Arduino. A servo motor is used to rotate with the Li-DAR sensor. We have used Python software to map the environment from Li-DAR and servo motor data. ESP 32 camera module is used for live video stemming. |
| PO6 | **The engineer and society** | This mapping and navigation rover is beneficial for our society and our engineering committee. In many situations, this rover can easily consume our time to find victims or our desirable objects as well as provide us maps of any unknown environment. |
| PO7 | **Environment and sustainability** | This rover is totally environmentally friendly and saves time, this can go anyplace easily, and work in places where humans can't reach. |
| PO8 | **Ethics** | We follow our social and moral principle. And we have finished our project by considering honesty, morality, privacy, fidelity, and confidentiality. |
| PO9 | **Individual work and teamwork** | We divided work among us when it was needed so we could do more work. But when the time came we sat down and explain to each other what we did and decided on how to put the whole thing together. Working with groups teaches us valuable lessons that help us become a better person. |
| PO10 | **Communication** | We communicate clearly with one another, Will cooperate well and work together to accomplish a goal more quickly. Improved idea exchange, project execution, and teamwork are all made possible through effective communication within a team. |
| PO11 | **Project management and finance** | In this project cost estimation is the process of estimating the amount of money and other resources required to finish the project within a specific time frame. |
| PO12 | **Life-long learning** | There are two main benefits of lifelong learning. One is personal development, and another is professional development. In this project, we can learn lots of new technologies, and new programming languages, improve our communication skills and acquire our knowledge in so many ways. Also, we can improve time management, and grow capable to do tasks in under pressure and improving. |

Chapter 6

# Conclusion

“Onnesha” is designed effectively in every regard to assist any rescue operation. It can map the surrounding and navigate its way in and out of the location where people cannot. It is equipped with real- time communication features such as- live video streaming and audio system. From the view of available technology and affordability it’s one of its kind. Within a few years unmanned mobile robots like “Onnesha” will be significant to human welfare and safety, especially in a country like Bangladesh where the recent chain of unfortunate tragic events like Rana Plaza tragedy, Chittagong container depot explosion has caused much turmoil.

# 

# References

[1] X. Zhang, J. Lai, D. Xu, H. Li and M. Fu, “2D Lidar-Based SLAM and Path Planning for Indoor Rescue Using Mobile Robots,” *Journal of Advanced Transportation*, vol. 2020, Article ID 8867937, 14 pages, 2020.

[2] J. Shou, Z. Zhang, Y. Su, and Z. Zhong, “Design and implementation of indoor positioning and navigation system of mobile robot based on ROS and lidar,” *Machinery and Electronics*, vol. 36, no. 11, pp. 76–80, 2018.

[3] W. Hess, D. Kohler, H. Rapp and D. Andor, "Real-time loop closure in 2D LIDAR SLAM," 2016 IEEE International Conference on Robotics and Automation (ICRA), 2016, pp. 1271-1278, doi: 10.1109/ICRA.2016.7487258.

[4] J. J. Leonard and H. F. Durrant-Whyte, “Simultaneous map building and localization for an autonomous mobile robot,” in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, Osaka, Japan, Nov. 1991, pp. 1442–1447.

[5] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, “FastSLAM: A factored solution to the simultaneous localization and mapping problem,” in *Proc. of Theaaai National Conference on Artificial Intelligence. American Association for Artificial Intelligence*, Edmonton, Canada, Jul. 2002, pp. 593–598.

[6] B. Zhang and J. Zhang, "Robot Mapping and Navigation System Based on Multi-sensor Fusion," 2021 4th International Conference on Artificial Intelligence and Big Data (ICAIBD), 2021, pp. 632-636, doi: 10.1109/ICAIBD51990.2021.9459053.

[7] Y. K. Tee and Y. C. Han, "Lidar-Based 2D SLAM for Mobile Robot in an Indoor Environment: A Review," 2021 International Conference on Green Energy, Computing and Sustainable Technology (GECOST), 2021, pp. 1-7, doi: 10.1109/GECOST52368.2021.9538731.

**Appendix**

**1. Li-DAR test code**

#include <SoftwareSerial.h> //header file of software serial port

SoftwareSerial Serial1(5, 6); //define software serial port name as Serial1 and define pin2 as RX & pin3 as TX

int dist; //actual distance measurements of LiDAR

int strength; //signal strength of LiDAR

int check; //save check value

int i;

int uart[9]; //save data measured by LiDAR

const int HEADER = 0x59; //frame header of data package

void setup()

{

Serial.begin(9600); //set bit rate of serial port connecting Arduino with computer

Serial1.begin(115200); //set bit rate of serial port connecting LiDAR with Arduino

}

void loop() {

if (Serial1.available()) //check if serial port has data input

{

if (Serial1.read() == HEADER) //assess data package frame header 0x59

{

uart[0] = HEADER;

if (Serial1.read() == HEADER) //assess data package frame header 0x59

{

uart[1] = HEADER;

for (i = 2; i < 9; i++) //save data in array

{

uart[i] = Serial1.read();

}

check = uart[0] + uart[1] + uart[2] + uart[3] + uart[4] + uart[5] + uart[6] + uart[7];

if (uart[8] == (check & 0xff)) //verify the received data as per protocol

{

dist = uart[2] + uart[3] \* 256; //calculate distance value

strength = uart[4] + uart[5] \* 256; //calculate signal strength value

Serial.print("dist = ");

Serial.print(dist); //output measure distance value of LiDAR

Serial.print('\t');

Serial.print("strength = ");

Serial.print(strength); //output signal strength value

Serial.print('\n');

}

}

}

}

}

**2. Li-DAR to Arduino**

// Includes the Servo library

#include <Servo.h>.

#include <SoftwareSerial.h> //header file of software serial port

SoftwareSerial Serialx(2, 3); //define software serial port name as Serial1 and define pin2 as RX & pin3 as TX

Servo myServo;

int dist; //actual distance measurements of LiDAR

int strength; //signal strength of LiDAR

int check; //save check value

int i;

String dataf;

int uart[9]; //save data measured by LiDAR

const int HEADER = 0x59; //frame header of data package

void setup() {

// put your setup code here, to run once:

Serial.begin(115200); //set bit rate of serial port connecting Arduino with computer

Serialx.begin(115200); //set bit rate of serial port connecting LiDAR with Arduino

Serial.println("Radar Start");

myServo.attach(12); // Defines on which pin is the servo motor attached

}

void loop() {

if (Serialx.available()) //check if serial port has data input

{ for(int deg=1;deg<=180;deg++){

myServo.write(deg);

if (Serialx.read() == HEADER) //assess data package frame header 0x59

{

uart[0] = HEADER;

if (Serialx.read() == HEADER) //assess data package frame header 0x59

{

uart[1] = HEADER;

for (i = 2; i < 9; i++) //save data in array

{

uart[i] = Serialx.read();

}

check = uart[0] + uart[1] + uart[2] + uart[3] + uart[4] + uart[5] + uart[6] + uart[7];

if (uart[8] == (check & 0xff)) //verify the received data as per protocol

{

dist = uart[2] + uart[3] \* 256; //calculate distance value

int distance =dist;

int \_degrees =deg;

//Serial.print(distance);

Serial.print(\_degrees);

Serial.print(","); //output measure distance value of LiDAR

Serial.println(distance); //output measure distance value of LiDAR

//dataf= String(deg)+String(",")+String(dist);

//Serial.println(dataf);

delay(60);

}

}

}

}

}

}

**3. Arduino to esp8266 to TCP/IP**

#include <ESP8266WiFi.h>

#define SendKey 0 //Button to send data Flash BTN on NodeMCU

int port = 8888; //Port number

WiFiServer server(port);

//Server connect to WiFi Network

const char \*ssid = "Ethereum"; //Enter your wifi SSID

const char \*password = "12345679"; //Enter your wifi Password

int count=0;

//=======================================================================

// Power on setup

//=======================================================================

void setup()

{

Serial.begin(115200);

pinMode(SendKey,INPUT\_PULLUP); //Btn to send data

Serial.println();

WiFi.mode(WIFI\_STA);

WiFi.begin(ssid, password); //Connect to wifi

// Wait for connection

Serial.println("Connecting to Wifi");

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

delay(500);

}

Serial.println("");

Serial.print("Connected to ");

Serial.println(ssid);

Serial.print("IP address: ");

Serial.println(WiFi.localIP());

server.begin();

Serial.print("Open Telnet and connect to IP:");

Serial.print(WiFi.localIP());

Serial.print(" on port ");

Serial.println(port);

while (!Serial) {

; // wait for serial port to connect. Needed for native USB port only

}

}

// Loop

void loop()

{

WiFiClient client = server.available();

if (client) {

if(client.connected())

{

Serial.println("Client Connected");

}

while(client.connected()){

if (Serial.available()) {

size\_t len = Serial.available();

uint8\_t sbuf[len];

Serial.readBytes(sbuf, len);

client.write(sbuf, len);

Serial.write(sbuf,len);

delay(0);

}

}

}

client.stop();

}

**4. Li-DAR data plotting - Python Pycharm**

# Python + Arduino-based Radar Plotter

#

# \*\* Works with any motor that outputs angular rotation

# \*\* and with any distance sensor (HC-SR04, VL53L0x,LIDAR)

#

import numpy as np

import matplotlib

matplotlib.use('TkAgg')

import matplotlib.pyplot as plt

from matplotlib.widgets import Button

import serial,sys,glob

import serial.tools.list\_ports as COMs

#

#

############################################

# Find Arudino ports, select one, then start communication with it

############################################

#

def port\_search():

if sys.platform.startswith('win'): # Windows

ports = ['COM{0:1.0f}'.format(ii) for ii in range(1, 256)]

elif sys.platform.startswith('linux') or sys.platform.startswith('cygwin'):

ports = glob.glob('/dev/tty[A-Za-z]\*')

elif sys.platform.startswith('darwin'): # MAC

ports = glob.glob('/dev/tty.\*')

else:

raise EnvironmentError('Machine Not pyserial Compatible')

arduinos = []

for port in ports: # loop through to determine if accessible

if len(port.split('Bluetooth')) > 1:

continue

try:

ser = serial.Serial(port)

ser.close()

arduinos.append(port) # if we can open it, consider it an arduino

except (OSError, serial.SerialException):

pass

return arduinos

arduino\_ports = port\_search()

ser = serial.Serial(arduino\_ports[0], baudrate=115200) # match baud on Arduino

ser.flush() # clear the port

#

############################################

# Start the interactive plotting tool and

# plot 180 degrees with dummy data to start

############################################

#

fig = plt.figure(facecolor='k')

win = fig.canvas.manager.window # figure window

screen\_res = win.wm\_maxsize() # used for window formatting later

dpi = 150.0 # figure resolution

fig.set\_dpi(dpi) # set figure resolution

# polar plot attributes and initial conditions

ax = fig.add\_subplot(111, polar=True, facecolor='#006d70')

ax.set\_position([-0.05, -0.05, 1.1, 1.05])

r\_max = 180.0 # can change this based on range of sensor

ax.set\_ylim([0.0, r\_max]) # range of distances to show

ax.set\_xlim([0.0, np.pi]) # limited by the servo span (0-180 deg)

ax.tick\_params(axis='both', colors='w')

ax.grid(color='w', alpha=0.5) # grid color

ax.set\_rticks(np.linspace(0.0, r\_max, 11)) # show 5 different distances

ax.set\_thetagrids(np.linspace(0.0, 180.0, 18)) # show 10 angles

angles = np.arange(0, 181, 1) # 0 - 180 degrees

theta = angles \* (np.pi / 180.0) # to radians

dists = np.ones((len(angles),)) # dummy distances until real data comes in

pols, = ax.plot([], linestyle='', marker='o', markerfacecolor='w',

markeredgecolor='#EFEFEF', markeredgewidth=1.0,

markersize=7.0, alpha=0.9) # dots for radar points

line1, = ax.plot([], color='w',

linewidth=4.0) # sweeping arm plot

# figure presentation adjustments

fig.set\_size\_inches(0.96 \* (screen\_res[0] / dpi), 0.96 \* (screen\_res[1] / dpi))

plot\_res = fig.get\_window\_extent().bounds # window extent for centering

win.wm\_geometry('+{0:1.0f}+{1:1.0f}'. \

format((screen\_res[0] / 2.0) - (plot\_res[2] / 2.0),

(screen\_res[1] / 2.0) - (plot\_res[3] / 2.0))) # centering plot

fig.canvas.toolbar.pack\_forget() # remove toolbar for clean presentation

fig.canvas.manager.set\_window\_title('Project: Lidar Based Navigation & Mapping')

fig.canvas.draw() # draw before loop

axbackground = fig.canvas.copy\_from\_bbox(ax.bbox) # background to keep during loop

############################################

# button event to stop program

############################################

def stop\_event(event):

global stop\_bool

stop\_bool = 1

prog\_stop\_ax = fig.add\_axes([0.85, 0.025, 0.125, 0.05])

pstop = Button(prog\_stop\_ax, 'Stop Program', color='#FCFCFC', hovercolor='w')

pstop.on\_clicked(stop\_event)

# button to close window

def close\_event(event):

global stop\_bool, close\_bool

if stop\_bool:

plt.close('all')

stop\_bool = 1

close\_bool = 1

close\_ax = fig.add\_axes([0.025, 0.025, 0.125, 0.05])

close\_but = Button(close\_ax, 'Close Plot', color='#FCFCFC', hovercolor='w')

close\_but.on\_clicked(close\_event)

fig.show()

############################################

# inifinite loop, constantly updating the

# 180deg radar with incoming Arduino data

############################################

#

start\_word, stop\_bool, close\_bool = False, False, False

while True:

try:

if stop\_bool: # stops program

fig.canvas.toolbar.pack\_configure() # show toolbar

if close\_bool: # closes radar window

plt.close('all')

break

ser\_bytes = ser.readline() # read Arduino serial data

decoded\_bytes = ser\_bytes.decode('utf-8') # decode data to utf-8

data = (decoded\_bytes.replace('\r', '')).replace('\n', '')

if start\_word:

vals = [float(ii) for ii in data.split(',')]

if len(vals) < 2:

continue

angle, dist = vals # separate into angle and distance

if dist > r\_max:

dist = 0.0 # measuring more than r\_max, it's likely inaccurate

dists[int(angle)] = dist

if angle % 1 == 0: # update every 5 degrees

pols.set\_data(theta, dists)

fig.canvas.restore\_region(axbackground)

ax.draw\_artist(pols)

line1.set\_data(np.repeat((angle \* (np.pi / 180.0)), 2),

np.linspace(0.0, r\_max, 2))

ax.draw\_artist(line1)

fig.canvas.blit(ax.bbox) # replot only data

fig.canvas.flush\_events() # flush for next plot

else:

if data == 'Radar Start': # stard word on Arduno

start\_word = True # wait for Arduino to output start word

print('Radar Starting...')

else:

continue

except KeyboardInterrupt:

plt.close('all')

print('Keyboard Interrupt')

break

**5. Rover control**

#define REMOTEXY\_MODE\_\_ESP8266\_HARDSERIAL\_POINT

#include <RemoteXY.h>

// connection settings

#define REMOTEXY\_SERIAL Serial

#define REMOTEXY\_SERIAL\_SPEED 115200

#define REMOTEXY\_WIFI\_SSID "CP II"

#define REMOTEXY\_WIFI\_PASSWORD "12345678"

#define REMOTEXY\_SERVER\_PORT 6377

// RemoteXY configurate

#pragma pack(push, 1)

uint8\_t RemoteXY\_CONF[] =

{ 255,5,0,0,0,76,0,15,24,0,

1,6,77,25,13,13,135,24,70,111,

114,119,97,114,100,0,1,6,78,44,

12,12,36,24,66,97,99,107,119,97,

114,100,0,1,6,4,39,12,12,134,

24,76,101,102,116,0,1,6,20,39,

12,12,16,24,82,105,103,104,116,0,

1,2,44,4,15,15,36,31,83,116,

111,112,0 };

// this structure defines all the variables and events of your control interface

struct {

// input variables

uint8\_t Forward; // =1 if button pressed, else =0

uint8\_t Backward; // =1 if button pressed, else =0

uint8\_t Left; // =1 if button pressed, else =0

uint8\_t Right; // =1 if button pressed, else =0

uint8\_t Stop; // =1 if button pressed, else =0

// other variable

uint8\_t connect\_flag; // =1 if wire connected, else =0

} RemoteXY;

#pragma pack(pop)

/////////////////////////////////////////////

// END RemoteXY include //

/////////////////////////////////////////////

#define REN 3

#define LEN 4

#define ren 7

#define len 8

#define RPWM 5

#define LPWM 6

#define rpwm 9

#define lpwm 10

int vSpeed=135;

void setup()

{

RemoteXY\_Init ();

pinMode(REN, OUTPUT);

pinMode(LEN, OUTPUT);

pinMode(ren, OUTPUT);

pinMode(len, OUTPUT);

pinMode(RPWM, OUTPUT);

pinMode(LPWM, OUTPUT);

pinMode(rpwm, OUTPUT);

pinMode(lpwm, OUTPUT);

}

void loop()

{

digitalWrite(REN,HIGH);

digitalWrite(LEN,HIGH);

digitalWrite(ren,HIGH);

digitalWrite(len,HIGH);

RemoteXY\_Handler ();

if (RemoteXY.Forward == HIGH){

analogWrite(RPWM, vSpeed);

analogWrite(LPWM, 0);

analogWrite(rpwm, 175);

analogWrite(lpwm, 0);

}

else if (RemoteXY.Backward == HIGH){

analogWrite(RPWM, 0);

analogWrite(LPWM, vSpeed);

analogWrite(rpwm, 0);

analogWrite(lpwm, 175);

}

else if (RemoteXY.Left== HIGH) {

analogWrite(RPWM, 0);

analogWrite(LPWM, vSpeed);

analogWrite(rpwm, 155);

analogWrite(lpwm, 0);

}

else if (RemoteXY.Right== HIGH) {

analogWrite(RPWM, vSpeed);

analogWrite(LPWM, 0);

analogWrite(rpwm, 0);

analogWrite(lpwm,155);

}

else if (RemoteXY.Stop==HIGH) {

analogWrite(RPWM, 0);

analogWrite(LPWM, 0);

analogWrite(rpwm, 0);

analogWrite(lpwm,0);

}

}