Autonomous Vehicle with VR Control

ECE 445 Design Document for MDR -- FALL 2018

Team 19: Kefan Tu, Kewei Sui

Team 32: Chenliang Li, Honglu He, Siping Meng

TA: Amr Martini

1 Introduction

1.1 Objective

Everyday, people spend more and more time outside of their homes. We may go vacations overseas; we may hangout after work. We can certainly ask some specific robot to finish their pre-decided job, but what if you need to feed your cat while having a coming meeting, organize your stuff when you want to invite someone you meet at the pub to your home, or move your lovely plants away from the burning sunlight when you're at school? There are numerous detailed situations that are hard to accomplish by simply using programmed command and smart appliances nowadays in the market. Though smart house is already able to support remote control to turn air condition on or off through host's mobile phone, there are still something that a smart house system can not achieve. We still need a robot to help us arrange the indoor objects and shift them for various needs. In order to minimize the scope of our project, we will focus on the problem of picking up a cup of water or a can of food and then finding a path to place it on the given position to feed the pet.

Thus, we decide to create a small automated robot which could grab and shift objects indoor offline after receiving the command given by the user. Our robot has a robot arm to pick objects and the lower part is a chassis vehicle for moving. A camera and a LiDAR sensor are placed in the front of the robot to help it detect objects and obstacles. Various algorithms are implemented in the microcontroller to give this robot abilities to find the path automatically and to place the object in the right place under acceptable tolerance. What's more, we will try to use a new interacting method, VR, to help user visualize and produce the result they want the robot to accomplish first and then the robot could accomplish this task remotely and offline in real world.

1.2 Background

The smart appliances are very common for modern families. Robot cleaner, kitchen robot and even robot for entertaining cat gradually appear in our house. But they all use traditional UI to interact with user which means that user can only send pre-programmed commands to these robots but the nonvisualized result is actually unclear to user. For instance, when we click the "clean" button on the vacuuming robot, we in fact don't know which part of the room will be cleaned and how clean it will be. We want to try some new interaction ways that can enlighten the smart appliance industry. With the VR experience, we designed a unique interaction way with future smart appliance that we can manually produce the result we want in virtual world and then a programmed robot could accomplish this task automatically in real world and offline. We hope this will inspire more innovations in the smart appliance manufacturers and designers.

In this case, our goal is to build a robot that can do tasks which is relatively flexible for normal appliances such as grab things and send things indoor, with an immersive interaction

way. Also, our robot could accomplish the task offline and automatically so that user don't need to care about the process but just the result of the task.

1.3 High-level Requirements

- The robot can go to the location specified by user (either directed given from PC or VR)
- The robot can recognize the object user specified.
- The robot can grip, hold, and place the object.

1.4 Experiment Settings

The experiment area is a 4m*4m flat tile surface. We divided it into 100 grids. In our coordinate system, every unit length is 0.4m.

- "Robot" represents the robot vehicle and it always starts at the initial position.
- "Object" represents the target object (like cat food can in real life) with color band on it. Object can be placed on any known place in the experiment area for testing object recognition, path planning, self-navigate and robot-arm function.
- "Furniture" represents obstacle with known size and position (like sofa, table in real life). They are not movable and are placed at the beginning of the experiment. We are going to use big artificial rectangular-shaped objects for the demo.
- "Obstacle" represents obstacle with unknown size and position (like trash on the ground in real life) which is not shown in the initial map. It can be placed anywhere in the experiment area during the experiment. It's used for testing the path planning and obstacle detection function. We are going to use small artificial rectangular-shaped objects for the demo.

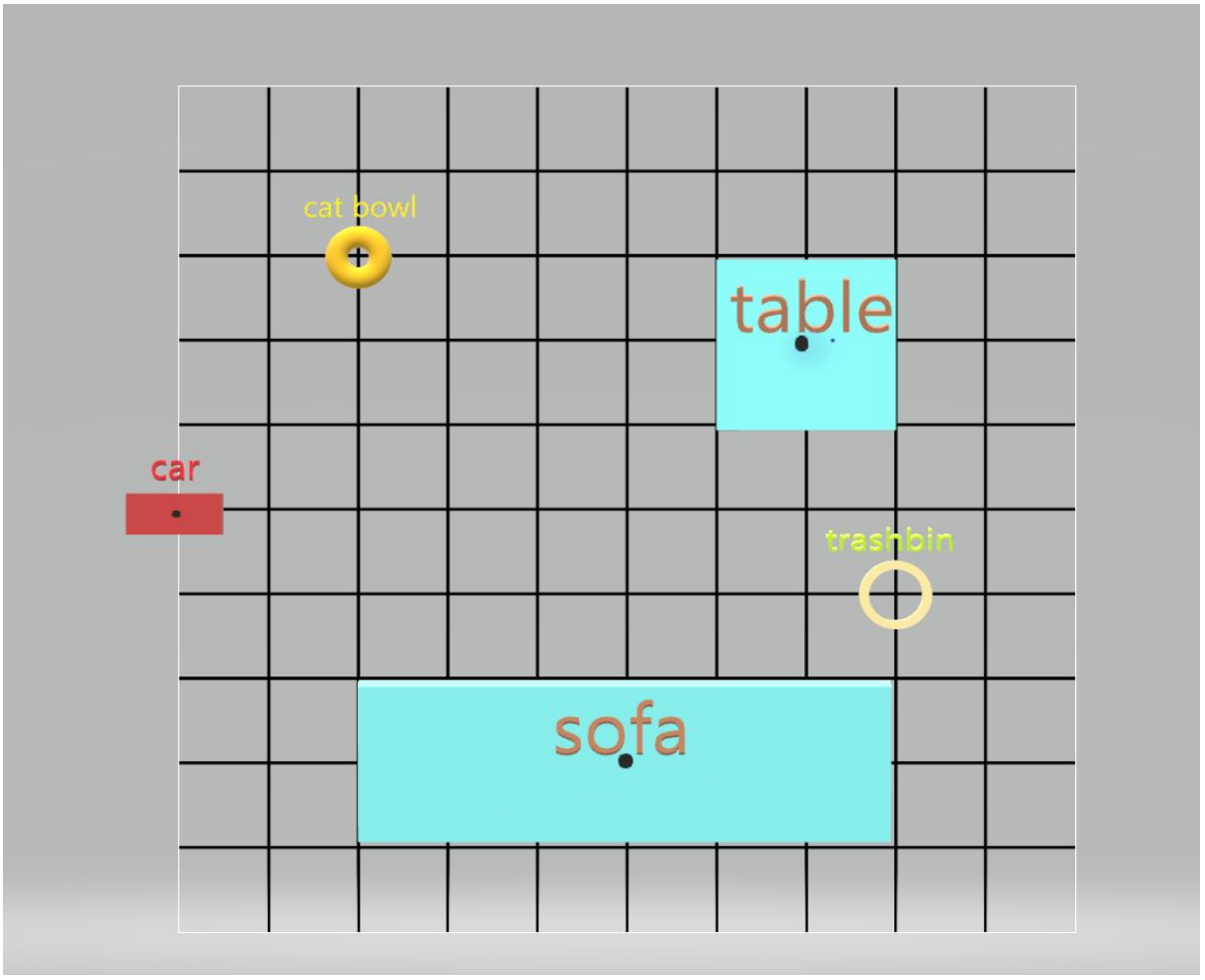


Figure 1: Experiment setting map (above)

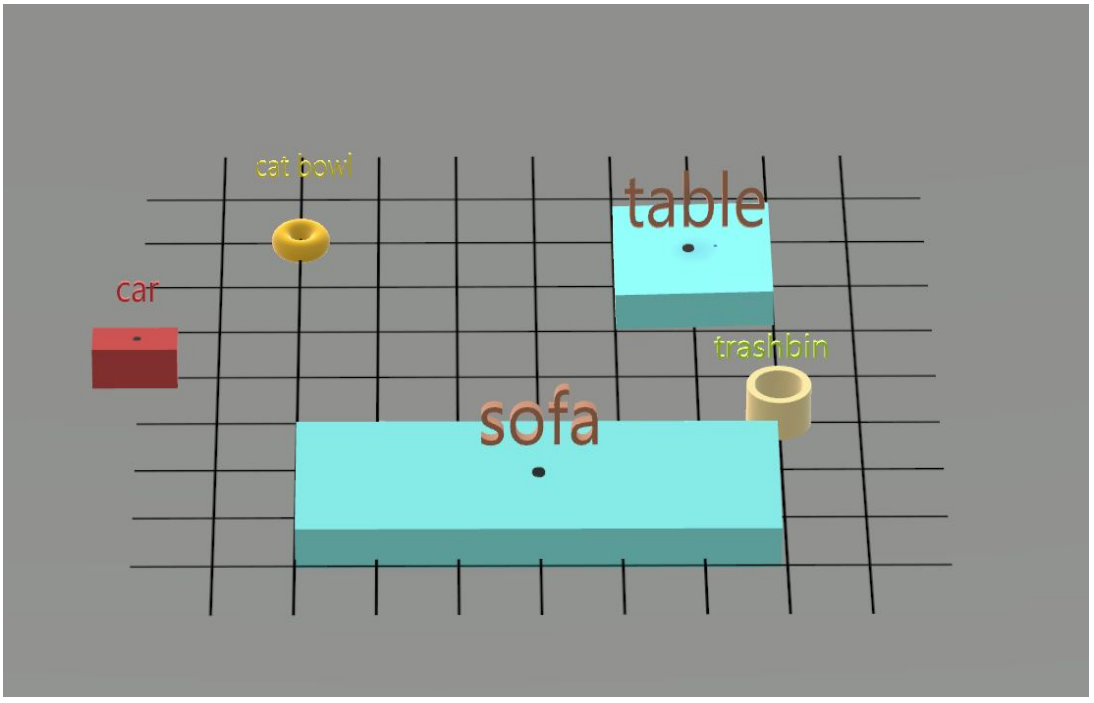


Figure 2: Experiment setting map (aside)

1.5 Solution Description

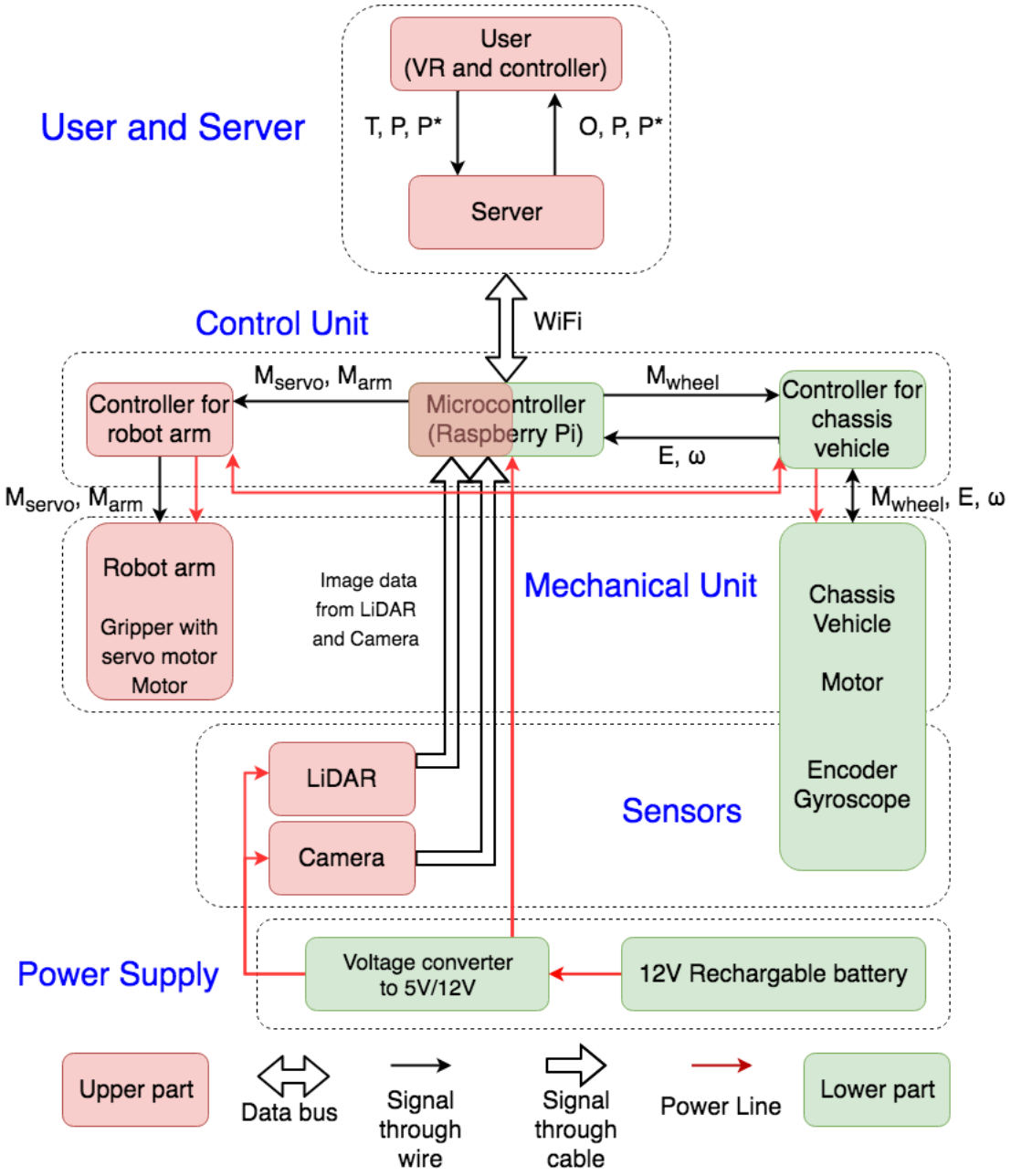
In this system, user will first produce an expected result in a virtual environment which is a mimic of the real situation, like a living room or a street. By using VR helmet and controller, user could freely change not only their position but their view perspective in the VR world in order to have a better understanding of the correlation between the world and the object that will be modified next. Then user could move, rotate and even create objects (if possible) in VR. All of these data will be stored and transmit to an autonomous robot through Wi-Fi. This robot can automatically find the object and search a path to move it to the target position with correct orientation. In this case, user will never be worried about the procedure of completing the task and they can even save this task into the database for the next time.

This project can majorly be separated into two big parts, the VR control frontend and the autonomous robot. For the VR control part, we need to create a complete virtual world that is exactly the same with an experimental real world. Because of the difficulty of mapping a complex real environment, we should find an empty space (research lab) and place several objects in it for experiment. We also need to build a server for storing and transmitting data between the VR frontend and the robot.

For the autonomous robot, we need to create our own robot based on a pre-build robot vehicle. Our robot should have the ability to move objects and find path automatically. Several sensors and electronic parts need to be used to obtain this goal, such as LiDAR, gyroscope and servos. A camera and LiDAR will be placed on the robot in order to recognize the object and calculate the position of that in world coordinates. We won't transmit the video of the camera to the VR helmet because it will severely affect the convenience. Furthermore, a microcontroller (Raspberry Pi) should be placed on this robot and we plan to utilize ROS system to control this robot. By using these sensors and several algorithms, this robot is able to do path planning and obstacle avoidance. We hope our robot could also memorize all the positions of obstacles encountered and store all of them into its own database. In this way, the path planning algorithm can be optimized time to time by using ML.

2 Design

2.1 Block Diagram



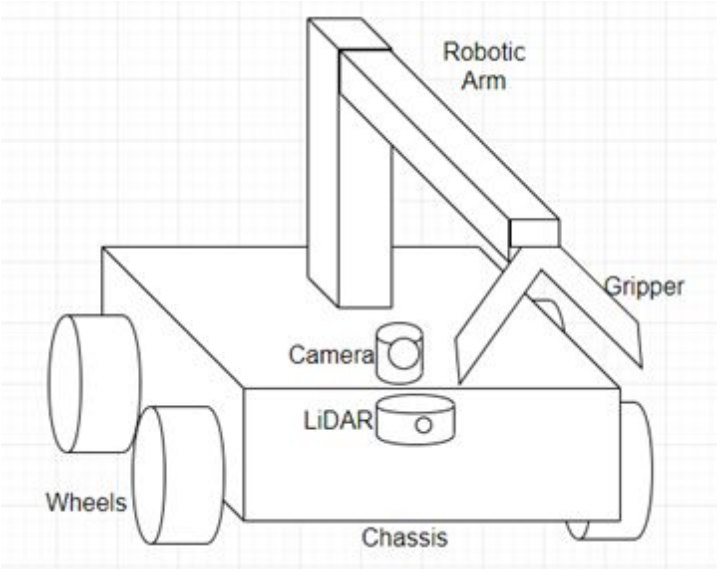
T	Target object	**P**	Old position	**E**	Data from encoder
O	Obstacles	**P***	New position	**ω**	Data from gyroscope
Mwheel	Signal to control wheel motor	**Marm**	Signal to control robot arm motor	**Mservo**	Signal to control gripper servo motor

The block diagram is consist of five main units: user and server, control unit, mechanical unit, sensors and power supply. They are balanced separated into two parts and each group of our big team is in charge of one part. User and server unit is the block which allows the user to give commands to the robot and obtain the new-generated 2D map from the microcontroller and then show it to the user. Control unit is the brain of the robot which contains one microcontroller (Raspberry Pi) and two control circuits (two PCBs for both upper and lower parts). Then, Mechanical unit is the physical structure unit for the entire robot. The robot arm is placed on the upper part and the lower part is mainly a chassis vehicle. What's more, various sensors are used in this project in order to implement multiple algorithms and accomplish the task. Finally, the power supply unit provides the power with different voltages to the entire robot.

The group in charge of upper part of the robot majorly contributes the user and server unit and the blocks arranged on the upper part of the robot in control, mechanical and sensors unit. Likewise, except dealing with the blocks placed at the bottom, the lower part group also handles the power supply unit. Both groups will work on the microcontroller together because it is the most crucial component to the entire system.

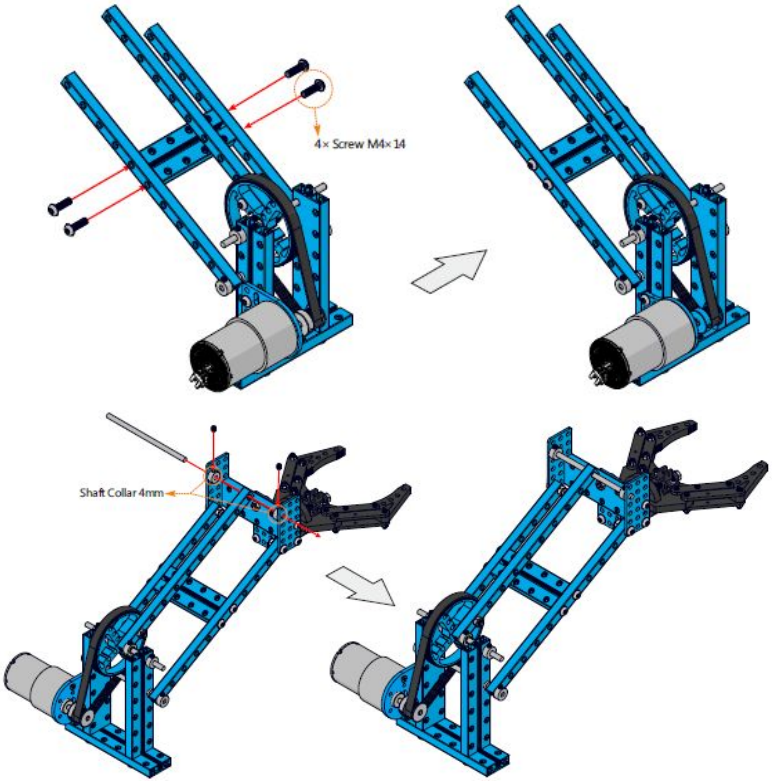
2.2 Physical Design

The physical design of our robot basically consists of a chassis vehicle with four wheels on two sides for moving, a robotic arm on the top of the chassis to grip items, and a camera and LiDAR on the front of the vehicle as sensors to detect the surroundings. The dimensions of the chassis vehicle is 200mm x 170mm x 105mm. The chassis basically has two levels: we put the microcontroller, PCBs and LiDAR on the first level and fix the robotic arm as well as camera on the second level.



2.3 Proposal 1

2.3.1 Robotic Arm



Robotic Arm Overview [1]

The robotic arm is divided into two parts, the upper part has a gripper at the end of beam while the other end is connected by a rotatable joint with the lower part, which is fixed to the chassis vehicle. The robotic arm needs to adjust the height of gripper. This is achieved by using a timing belt pulley that is driven by a motor; the pulley rotates with the upper beam so that we can adjust height of the gripper.

Robotic Arm	
Requirements	Verification
Able to grip a cylinder that weighs at least 500g and to hold it while moving for at least 20 seconds without slippery	(a) Place a cylinder that weighs approximately 500g on the floor and allows it to be detected by camera (b) Run arm control program to pick the cylinder up (c) Ensure the robot can grip and hold the cylinder for at least 20 seconds

2.3.1.1 Robot Arm with Servo Motor

We need a gripper at the very front of our robot arm to grip items. The gripper is controlled using a N20 servo motor. When the servo spins, the gripper will be pushed to open or pulled to close thanks to the mechanical design. The operating voltage range of the gripper is 5-12V DC.

Gripper	
Requirements	Verification
Open-width of gripper is at least 60mm	(a) Run arm control program (b) Use a standard ruler to measure the distance between two ends of the gipper (c) Ensure the measured value is greater than 60mm

2.3.1.2 DC Motor

We need a DC motor to control the timing belt pulley in order to adjust the height of gripper. The operating voltage range of the motor is 5-12V DC. It will interface with the motor controller circuit to be able to run in opposite directions.

DC Motor	
Requirements	Verification
Able to provide enough torque to change the angle between parts of arm as well as the height of gripper	(a) Give 12V input to the motor (b) Ensure the angle between the upper part and lower part of the arm is adjustable

2.3.1.3 Arm Control Algorithm

To grip the item the robot just confirms, the pseudocode is the following:

If object is recognized:
Activate 12V DC motor to lower the height of gripper and stop it when a certain height is reached
Activate servo motor to open the gripper to its maximum width
Move the robot forward to allow the item is in the gripper
Reverse the direction of servo motor; it keeps running until the gripper width equals a pre-determined value
Reverse the direction of DC motor, activate it and stop it at a pre-determined height

To place an item on the floor, the pseudocode is very similar to the pickup one:

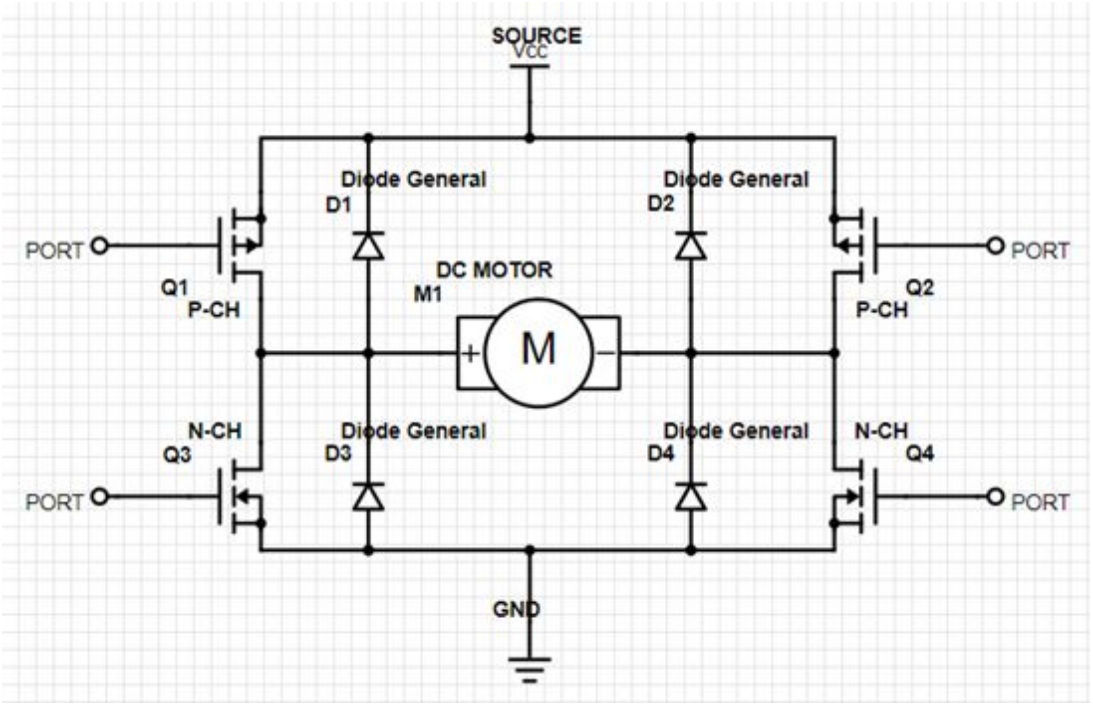
If robot is at destination:
Activate 12V DC motor to lower the height of gripper and stop it when a certain height is reached
Activate servo motor to open the gripper to its maximum width
Move the robot backward so that the item is out of gripper
Reverse the direction of DC motor, activate it and stop it at a pre-determined height

2.3.2 PCB design (for motor and servo)

2.3.2.1 Motor Controller

A motor controller will be implemented in our PCB to handle large current and high voltage (12V) to drive motors. Basically we use an H-bridge design to drive the motors, using two pairs of transistors to control the direction of current and thus the direction of motor rotation. The controller receives signals from the microcontroller in order to control motor motion.

Motor Controller	
Requirements	Verification
Able to change the direction of motor running	(a) Give a PWM input with a 12V amplitude and a duty cycle of 50% (b) Ensure a direction control of motor



Schematics of Motor Driver [2]

2.3.2.2 Circuit Schematic

2.3.3 User (VR)

We will first ensure that by directly inputting P* and T into the microcontroller of the robot, it could work functionally. VR part is a kind of a stretch goal for our project. We decide to use Oculus Go with controller as our front-end in this project, because of the portability of the helmet and good maneuverability of the controller. In VR, user could emerge in a virtual environment that is exactly the same with the real experimental environment (the design of map is shown in introduction section). User is able to see the target objects and the obstacles stored. Because we only save the 2D position for every objects and obstacles, models for them will be placed in the same x,y coordinates in VR but the height for those models may be differ from that in real world. By using the controller, we can grab the object freely and move it to somewhere else. Then, the new position P* of the target object T will be transmitted to the server and then sent to the robot in order to finish the task in real world.

2.3.4 Server

We will use a computer as a server in our project and for data transmission. We are going to use TCP as a way to enable communication between devices, more specifically the <socket> package in python 3 to realize TCP.

2.3.5 Sensors

2.3.5.1 Camera

The Camera we used is the camera module for raspberry Pi, and the use for the camera is to detect the direction of the object so that our robot vehicle will correct itself facing the object directly. And then the grabbing mechanism could work. Another use is for the robot to identify which way it's facing during calibration. The horizontal FoV is 62.2 degrees and the vertical FoV is 48.8 degrees which is sufficient for our project.

Camera	
Requirements	Verification
Camera must shoot at least two 640x480 picture per second	A. Write a program to output the png file of the captured pics B. Manually check the created time of those files to ensure at least two pics are created in a second.

2.3.5.2 LiDAR

The model we use is the SLAMTEC RPLIDAR A2m8. The frequency of the LiDAR is set to 10 Hz, and 400 data points for 360-degree scan. We'll use as many data points as possible, depending on the dimension of the robot arm.

LiDAR	
Requirements	Verification
On the microprocessor will process the LiDAR data and output the obstacle location to the map. The map will show at least 90% of the obstacle that the current robot can detect.	Put the robot at home position, and turn the robot on but without running. Wait for 5 seconds and then check the map generated by the robot. Compare the obstacles location with the real ones, and people stay at home position and count if those obstacles can be seen by human eye.

2.3.6 Control Unit

2.3.6.1 Microcontroller (Introduce Raspberry Pi)

Raspberry Pi 3B+ will receive data from all sensors, include LiDAR (USB), Camera (Pi module), Encoder (GPIO), Gyroscope (ADC to GPIO). It will send data to robot arm (Servo) (GPIO/PWM), motors (wheel) (GPIO). It will also send location data to user's laptop through TCP/IP communication.

2.3.6.1.1 Total Flow Chart

2.3.6.2 Algorithms

2.3.6.2.1 Object Detection

Each object will have a color band at the top, and after the robot reaches the approximate location, the camera will play a role as a feedback loop to correctly making sure the robot is facing to directly toward the object

Object Detection	
Requirements	Verification
The robot can adjust it self facing object, with an angle of +/- 10°	We put the robot near the object and turn on camera detection, and then check the angle between the robot and the object

2.3.6.2.2 Localization Calibration

We'll calibrate the robot position by setting waypoint at each corner of the environment. Every 2 min, the robot will generate a signal to calibrate itself, so it will go to the nearest corner, facing the green side, and then push itself against the wall. At this time both the position and the angle will be calibrated.

Calibration	
Requirements	Verification

The robot should get to the nearest corner, and update its position and angle. After update, its location and its actual location should be within 5% difference.	At any position in the map, we give the robot a calibration signal, the the robot will go to the nearest corner immediately. Then we check if it's calibrated or not.

2.3.6.2.3 Obstacle Avoidance

The LiDAR will pass in raw value (distance in each data point) to the microprocessor, and we'll process them into world frame position and mark the grid near them as obstacle. Once an obstacle position is determined, it can't be cleared unless the robot re-start with a fresh new map. A* algorithm will avoid any obstacle.

Obstacle Avoidance	
Requirements	Verification
While robot is moving, it should avoid all obstacle or just slide against or touch the obstacle. It wouldn't crash into an obstacle and get stuck.	Starting at home position, giving a object location and let the robot run itself. The environment should have 4 boxes. The robot should've successfully bypass the obstacle in the way.

2.4 Proposal 2

2.4.1 Mechanical Unit

2.4.1.1 Chassis

We choose to use acrylic board robot car chassis and bought from the Internet.

2.4.1.2 Motor

The DC brushed motor we'll use to drive our robot is CHR-GM25-370, DC 12V, 220RPM. It comes with A 6-PIN Magnetic Holzer Encoder.

We choose to use brushed motor rather than the brushless motor because it is cheap, steady, and easy to control: we can simply use voltage to control the speed of the motor and we can also change the voltage direction to change the rotation direction of the motor. The maximum speed is 160-300rpm and the working voltage is 3-12V DC. The max torque is 1.8kg.cm.

2.4.2 Sensors

2.4.2.1 Encoder

The motor comes with 6 -PIN holzer encoder, which counts to 224.2 pulse per revolution. With rotation rate information, we could read it to microprocessor and use the setting diameter of our wheels to calculate the linear distance that the robot have travelled.

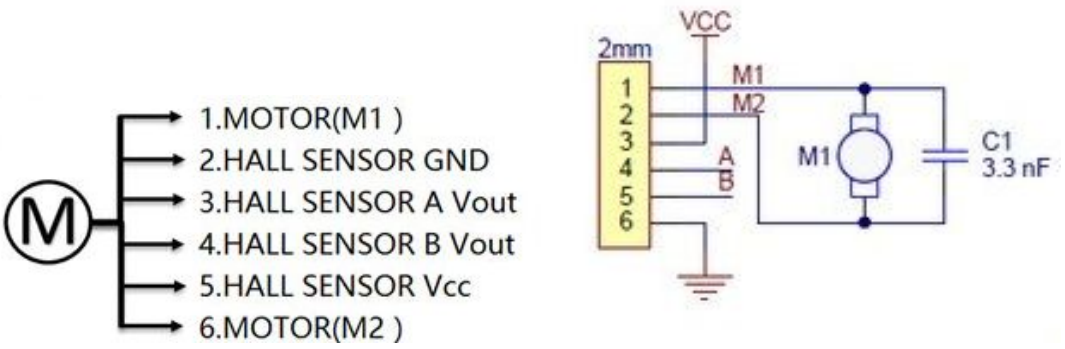


Figure 2.encoder & motor schematic

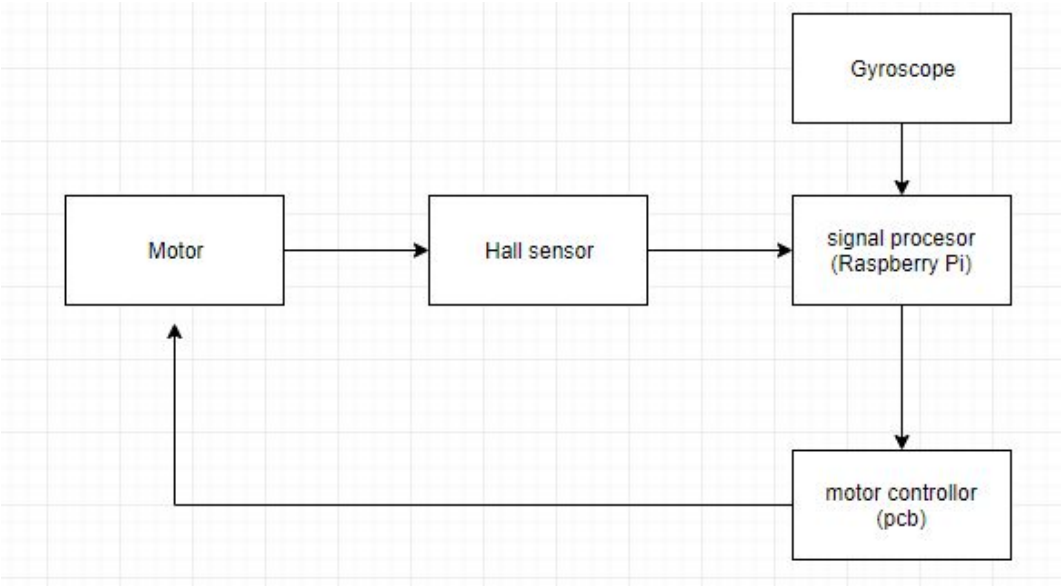


Figure 2.4 Motor control loop

We have two method to implement the wheel speed, one is based on frequency and another is based on period cycle. Since our wheel speed is relatively low, so we decide to use the measure method based on period cycle.

$$n = \frac{60f}{p \times m}$$

In the formula above, n is the rad/s, f is the base frequency we produce by controller, m is the pulse number the encoder produce.

As long as we get the rotation rate we can calculate distance we travelled and we can determine our moving direction from the information we get from the gyroscope.

Encoder	
Requirements	Verification
Receiving signal from raspberry Pi and control the voltage and current direction across the motors.	calculate the distance by the information of the encoder and then compare to the real distance the wheels have travelled. The difference should less than 5%.

2.4.2.2 Gyroscope

The single Axis gyroscope will output voltage at 0.67mV/deg./sec, where still voltage is 1.35v. We'll use the ADC chip connecting the gyro and the pi to read in information and convert to actual angular velocity.

Gyroscope	
Requirements	Verification
sending voltage signal to the Pi to calculate the information of angel rotation and we can get the moving direction	We can calculate the direction by the information we get from the gyroscope and compare to the real direction, the difference should less than 5%

2.4.3 PCB design (for motor, gyro and encoder)

2.4.3.1 Motor Controller

We are using 3-12V DC brushed motor to drive our robot. To control the DC brushed motor we need to control the voltage across the motor. So we need to build a voltage control circuit.

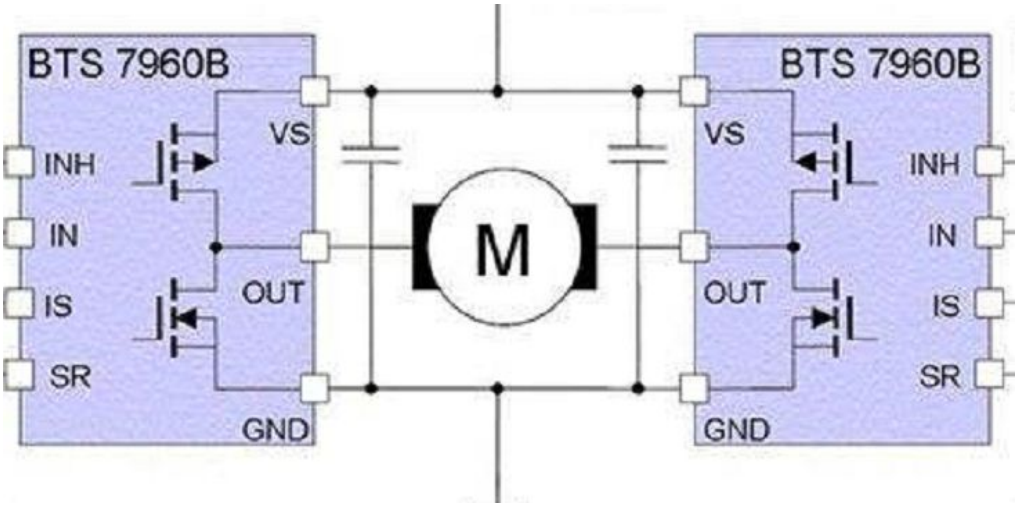


Figure 2.4.31 circuit schematic

For brushed 130-motor, the speed control mostly depend on the voltage provided to the motor, and the direction of rotation is depend on the current direction.

We use two BTS7960B chips to build a H bridge to control since we need to control the current direction to make the motors run in two directions. We choose to use BTS7960B is because that chip can provide precisely voltage control to the motor at the high current condition(43A).

Motor Controller	
Requirements	Verification
Output the pulse in proportion of the rotation rate.	Check the rotation rate of the wheel under different input we give to the controller circuit.

2.4.3.2 Protection Circuit

A protection circuit in the PCB to prevent our circuit from any misconnection of the power or overload.

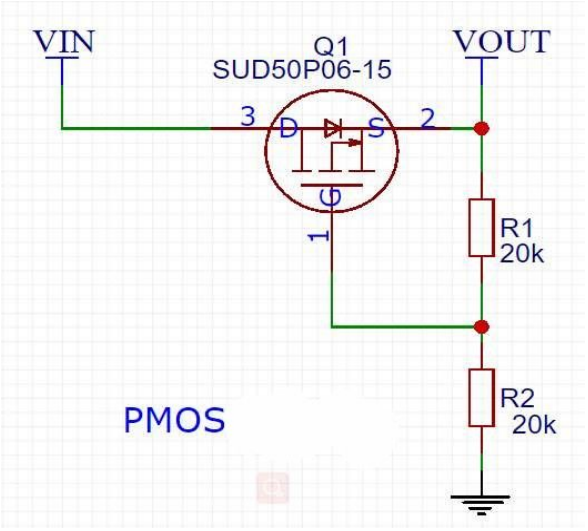


Figure 2.4

The PMOS we choose may change eventually depend on the circumstance. It will restrict the direction of the current.

We will also use resettable fuse in the circuit to constrain the current and make sure our circuit will not burn due to excessive current.

2.4.3.3 Circuit Schematic

2.4.4 Algorithms

2.4.4.1 Localization

Since the microprocessor will process the raw information from the gyro and both encoders, we can calculate the robot position relative to world frame as follow:

$$\begin{pmatrix} x \\ y \\ \theta \end{pmatrix} = \begin{pmatrix} \cos\theta\_old & 0 \\ \sin\theta\_old & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} v \\ w \end{pmatrix} t + \begin{pmatrix} x\_old \\ y\_old \\ \theta\_old \end{pmatrix}$$

Figure 2.4.4.1

2.4.4.2 2D Mapping

Our map will be a 10*10 gridded map, all obstacles will be a 1*1 square box. With LiDAR passing length information in different angles, we can calculate its relative position and transform into world position and then check which obstacle it's close to, and update that obstacle.

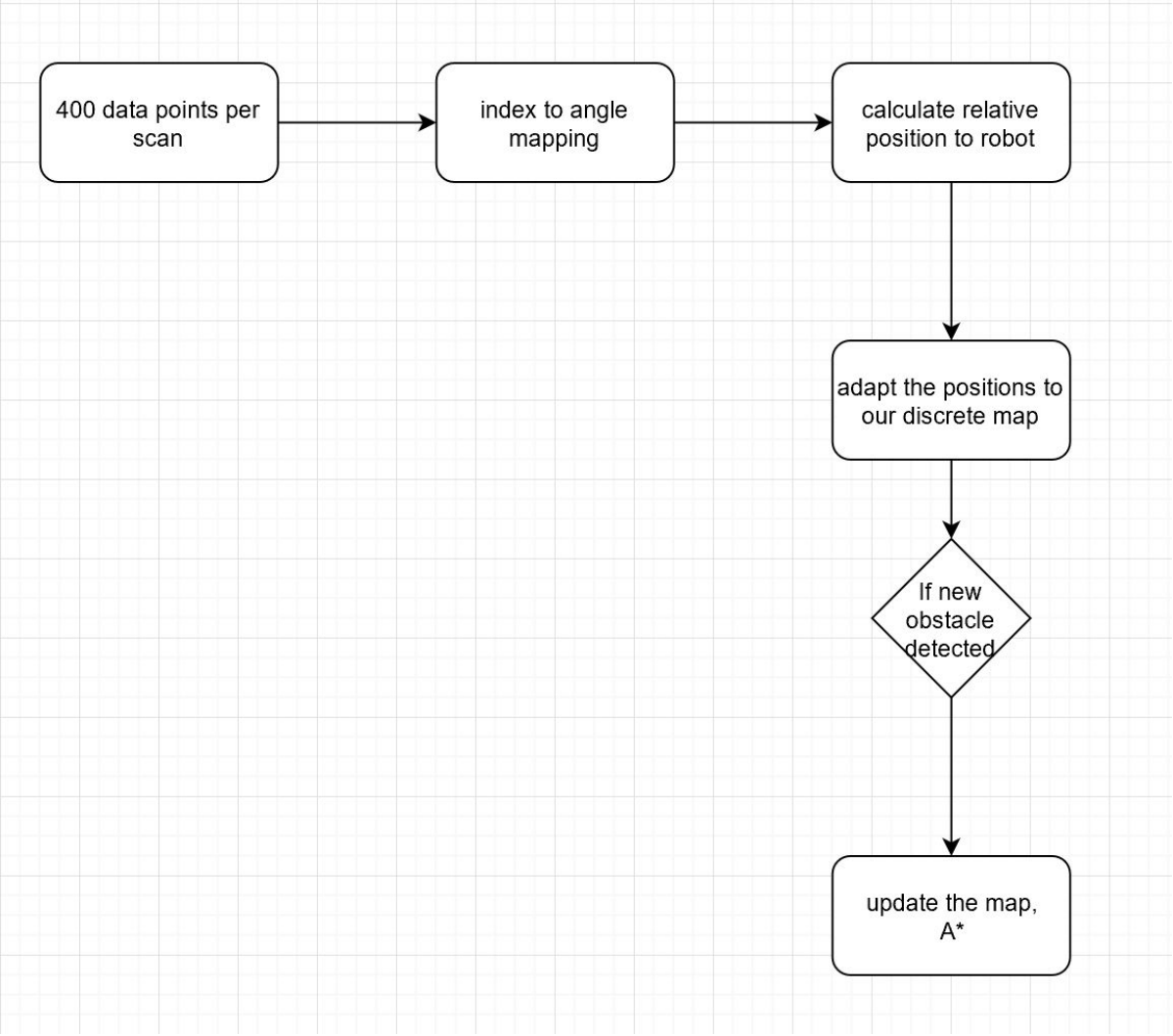


Figure 2.4.4.2

2.4.4.3 Path Planning

Based on the up-to-date map and the predefined location of the object, the robot will just generate a path using A*, with each step at the center of the gridded block on the map. While moving, the LiDAR will also collecting surrounding information, and if a new obstacle is detected, the map will be updated and the path will be recalculated by A* based on the up-to-date map. Moreover, if there is no available path for the robot and target, the robot will go back to its starting position and standby.

Algorithm	
Requirements	Verification
1. The robot will not hit obstacles during the whole process, including	1. By observation, the robot should not hit any objects.

walls. 2. The robot will walk down the optimal path.	 2. Robot will send back its current path to show this on 2d map.

2.4.5 Power Supply

2.4.5.1 Battery

2.5 Tolerance Analysis

3 Cost and Schedule

3.1 Cost Analysis

PARTS					
Index	**Part Name**	**Manufacturer**	**Qt.**	**Price Per Item**	**Total Price**
1	RPLIDAR A2		1	319.95	319.95
2	CAMERA		1	25	25
3	ROBOT ARM		1	89.99	89.99
4	VEHICLE		1		
5	RASPBERRY PI		5	39.95	199.75
6	GYRO		1	15.61	15.61
7	MICROSD		5	9.8	49
TOTAL COST					

LABOR			
Name	**Salary ($/hour)**	**Hours**	**Total**
Kewei Sui			
Kefan Tu			
Honglu He			
Chengliang Li			
Siping Meng			

3.2 Schedule

4 Stretch goals

5 Ethics and Safety

In our project we will use PCB, motors, Li-po battery and VR headset. We will buy our components and devices from qualified sellers, and follow the product instructions during the whole experiment.

As for the Li-po battery, which needs to be particularly taken care about due to the explosibility, we will make sure the temperature of the battery during the processing, charging and storing is in the safe range of the industrial standard. Our charger is an industry made IC device and it shuts charge controller off when charging input is higher than required voltage range, which will reduce the likelihood of hazards while charging. We also design the protect circuit. To protect the PCB, we are using flyback diode and Transient Voltage Suppressor to protect the circuit since we will connect the motor to the PCB which will also have other functioning circuits.

For the VR headset, we will operate it follow the safety manual of the product. We will also make sure only to test VR device with other teammates around and objectively predicting whether a user will experience discomfort from our content without obtaining feedback from inexperienced users can be difficult. [a]

We will follow the electricity using manual during the experiment, and will also check our power and circuit before connecting it to battery in order to prevent short circuit which may lead to electric shock.

For the ethical issues, we will follow IEEE and ACM code of ethics. We may encounter many problems in the project. When the problems occur, we will not try to disguise the problems and move on recklessly. Based on #5 of IEEE Code of Ethics, “to seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors” [b]. Therefore, when the problems show up, we will try our best to find a way with our teammates to solve the problem. If we can’t solve the problem by ourselves, we will turn to our TA for help.

6 Citation

[1] “Robotic Arm Add-on Pack for Starter Robot Kit,” github.com, July 23, 2014. [Online]. Available: https://github.com/Makeblock-official/Robotic-Arm-Add-On-Pack-For-Starter-Robot-Kit/blob/master/Assembly%20Instructions.pdf. [Accessed Sep. 20, 2018]

[2] Sam, “Motor Drivers vs. Motor Controllers,” core-electronics.com, June 27, 2017. [Online]. Available: https://core-electronics.com.au/tutorials/motor-drivers-vs-motor-controllers.html. [Accessed Sep. 20, 2018]

[3] Infineon Technologies, “High Current PN Half Bridge NovalithIC” ,BTS7960, Dec 2004[Accessed 17 Sep.2018]

[4] Glaser, Chris. “Five Steps to a Great PCB Layout for a Step-down Converter.” Analog Applications Journal, Texas Instruments, www.ti.com/lit/an/slyt614/slyt614.pdf.

[5] Oculus VR stuff, Oculus Best Practice, Oculus VR, 2017.

[6] Ieee.org. (2018). IEEE IEEE Code of Ethics. [online] Available at: https://www.ieee.org/about/corporate/governance/p7-8.html [Accessed 19 Sep. 2018].