Autonomous Vehicle with VR Control (lower part)

Siping Meng, Chenliang Li, Honglu He
ECE 445 Project Proposal – Fall 2018
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 control some appliance when we are out, but what if you need to feed your cat while having a coming meeting, or organize your stuff when you want to invite someone you meet at the pub to your home? 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. In order to find a better way of interaction way we decide to use virtual reality as the communication method.

1.2 Background

The smart appliances are very common for modern families. 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. We hope this will inspire more innovations in the smart appliance manufacturers and designers.

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 in our house, with an immersive interaction way.

1.3 High Level Requirements

- The robot can go to the location we specify from VR.
- The robot can recognize the object we specify.
- 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.

- The “furnitures”(including sofa, table, trash bin and cat bowl)will be placed as the graph shows.
- The “car” will always start at the initial position.
- The “obstacle” which is not shown in the initial map can be placed anywhere in the experiment area for testing the path planning, self-navigate,obstacle detection function.
- The “object” with color label which is also not shown in the map can be placed anywhere in the experiment area for testing object recognition, path planning, self-navigate and robot-arm function.

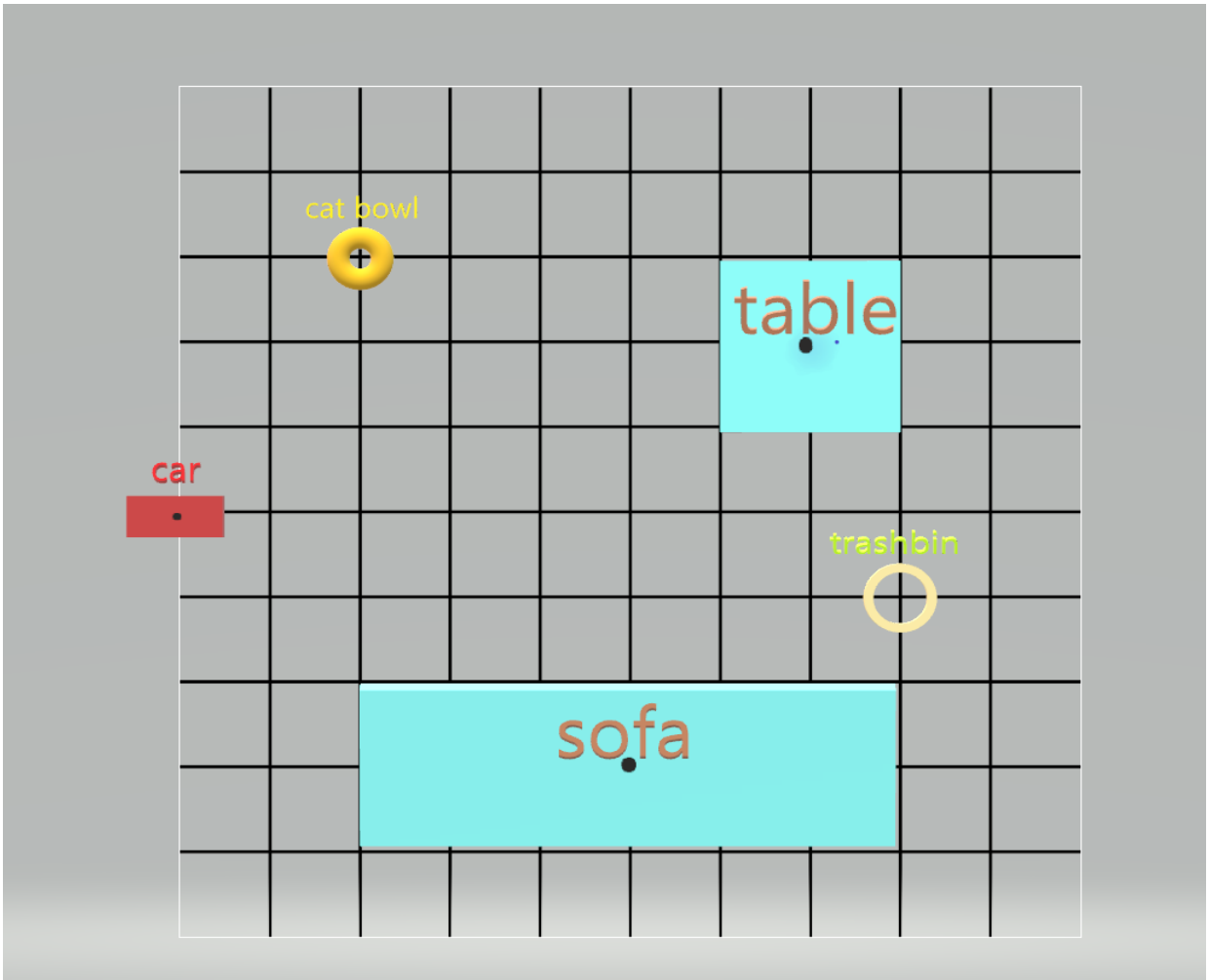

Figure 1: Experiment setting map (above)

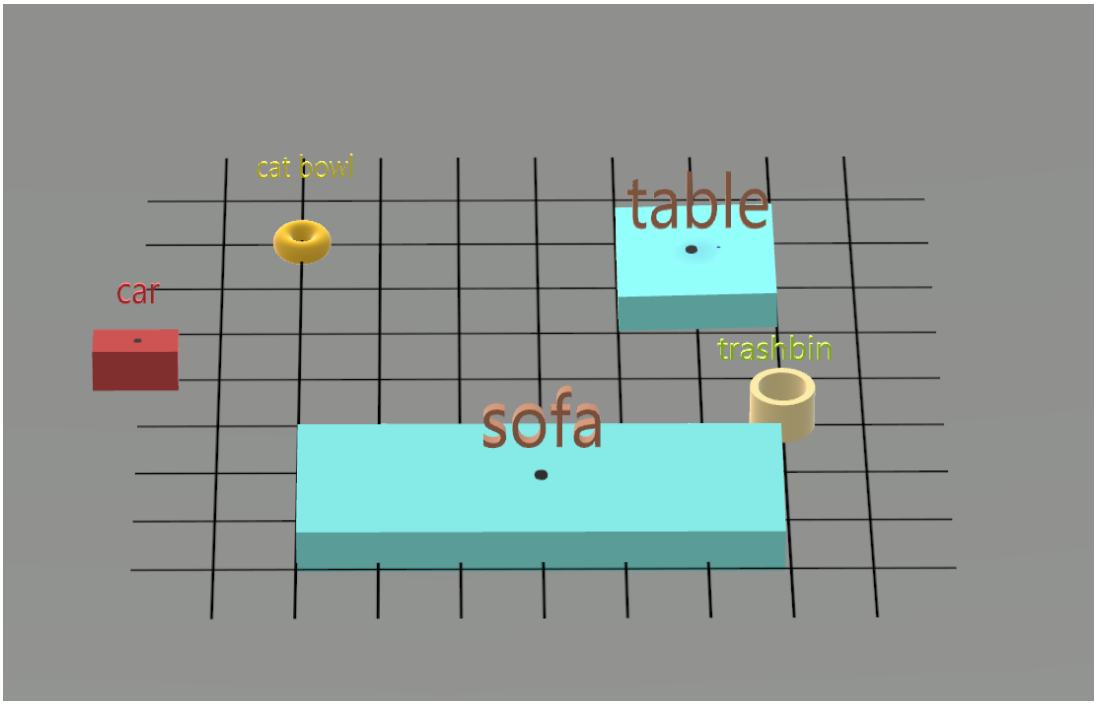

Figure 2: Experiment setting map (aside)

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

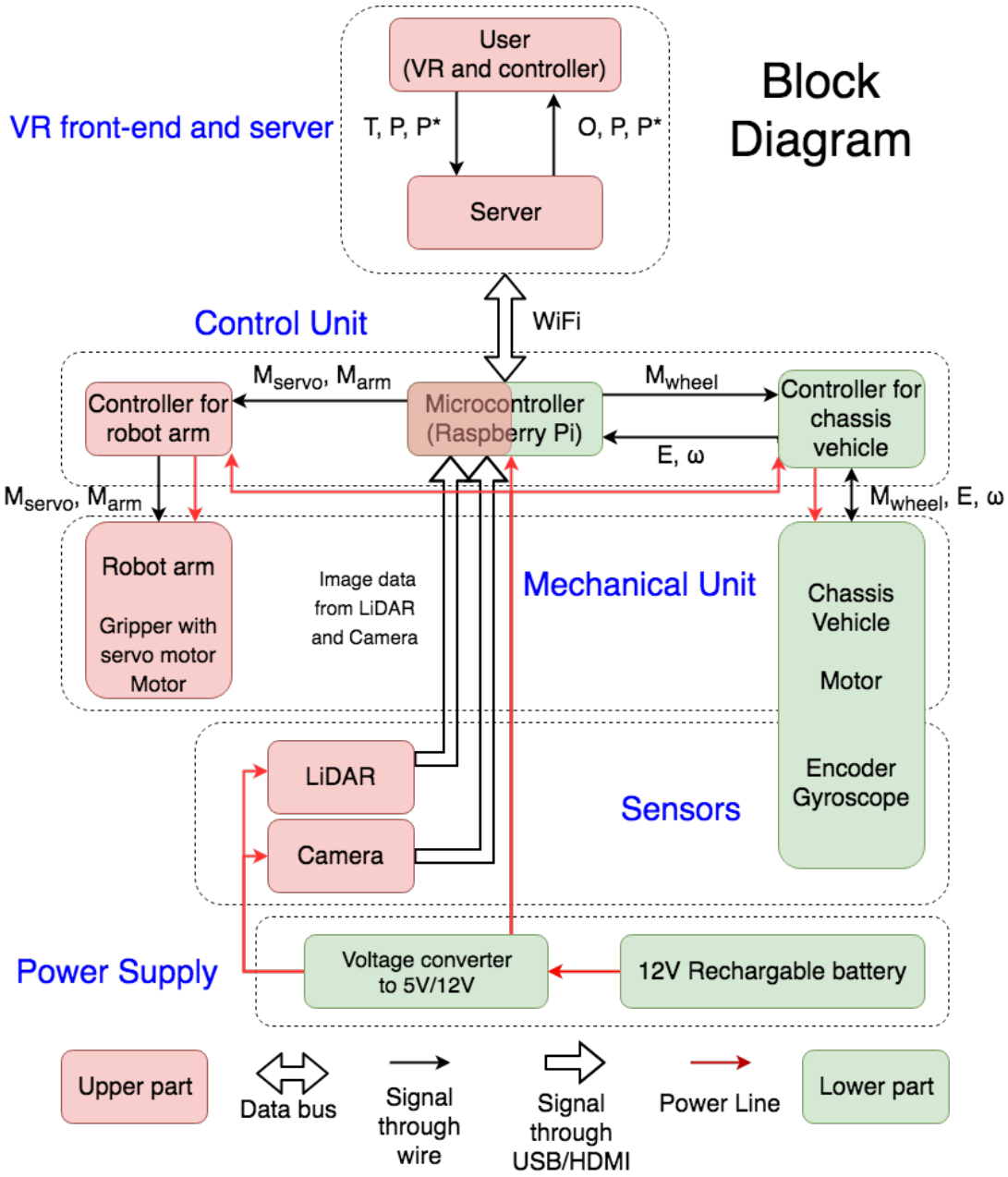

Figure 3: Block diagram

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

Figure 4: Variables Reference Table

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.

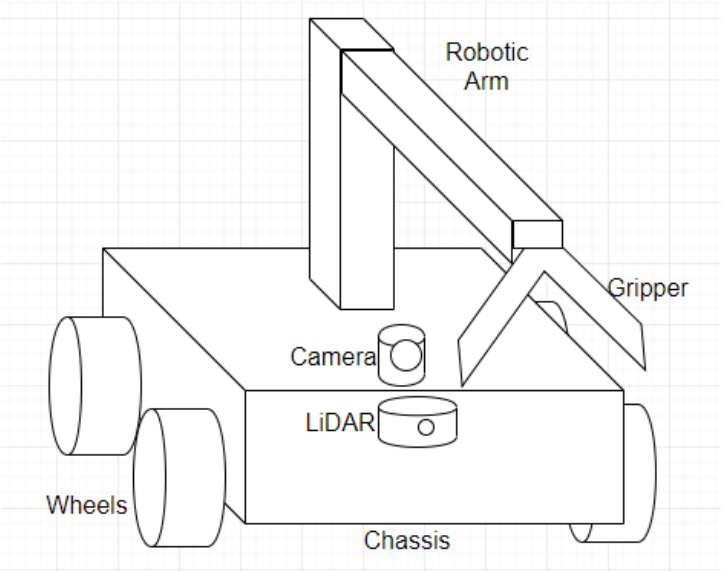

Figure 5: Physical Design Overview

2.3 Proposal 1 (upper part)

2.3.1 Robotic Arm

The robotic arm is divided into two beams, the upper beam has a gripper at its end while the other end is connected by a rotatable joint with the lower beam, 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.

Requirement: Able to grip a cylinder that weighs at least 500g and to hold it while moving for at least 10 seconds without slippery

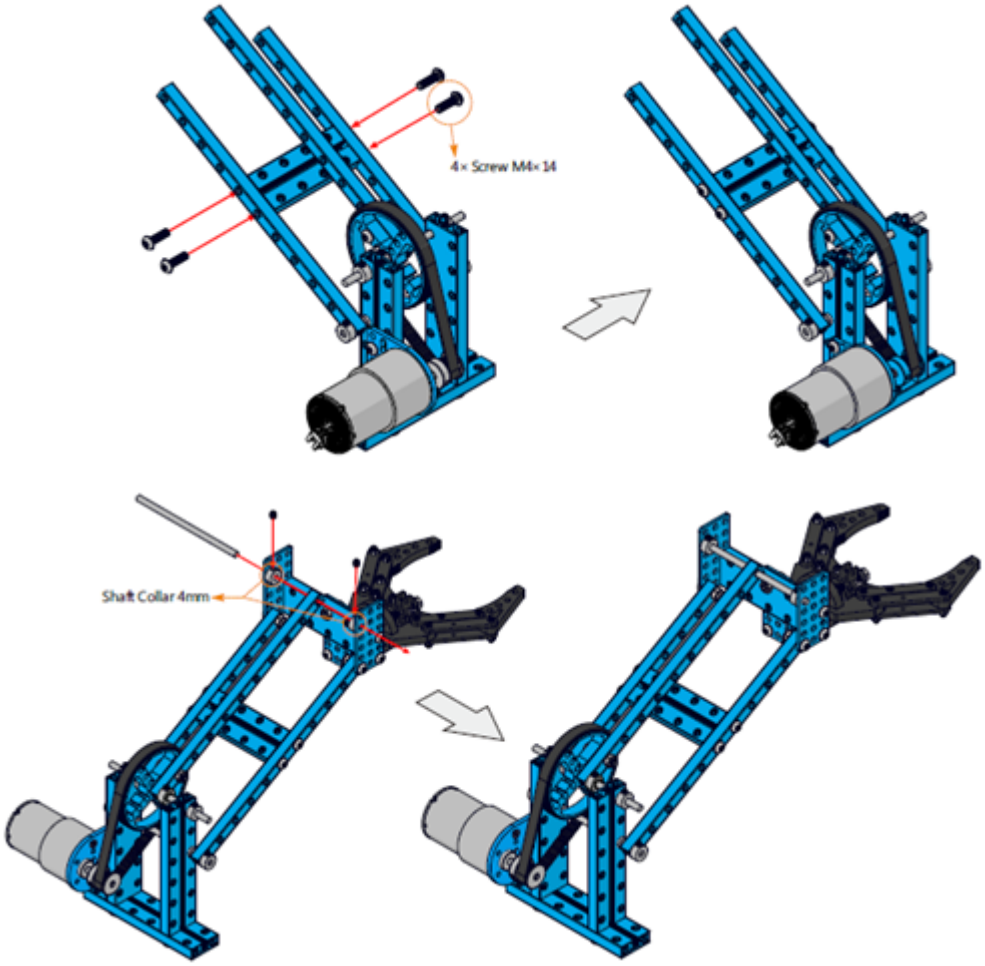

Figure 6: Robotic Arm Overview [1]

2.3.1.1 Robot Gripper 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.

Requirement 1: Open-width is adjustable given different and proper inputs to screw motor

Requirement 2: Maximum open-width should be at least 60mm

2.3.1.2 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.

Requirement: Be able to provide enough torque to change the angle of the beam as well as the height of gripper

2.3.1.3 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.
Requirement: The direction of motor rotation is adjustable

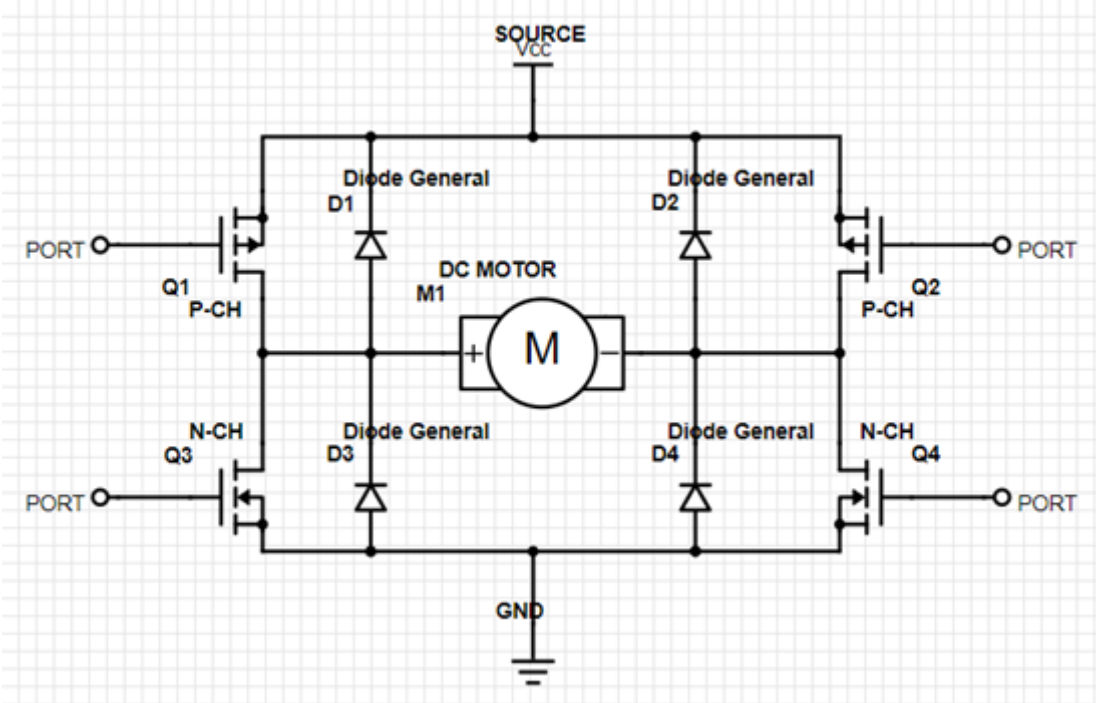

Figure 7: Schematics of Motor Driver [2]

2.3.2 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.
Requirement 1: Must implement a virtual world identical to the real experimental environment and enable user to move the avatar and objects in VR freely and comfortably

Requirement 2: Latest obstacles will to be shown in VR if the robot detects a new one

2.3.3 Server

The raspberry pi will open up a TCP/IP communication port, and on the PC side we'll use labview to receive the map info including the object location, obstacle location, and the robot location and plot them out. In this project, we will use computer as the server. First we will connect our computer and raspberry pi to a router and then find the IP address of the raspberry pi. Then we can let them communicate easily through WiFi. If we want to remote access the Pi we can either use the third-party service such as remote.it or configuring port forwarding on our router. If we want to remotely access the Pi's command line, we could enable SSH locally. If we want to copy the files, we could use secure file transfer protocol using FileZilla or Nautilus while using Ubuntu.
Requirement: On the PC side we should see the 2D representation of the environment and the robot.

2.3.4 Control Unit

2.3.4.1 Microcontroller

The microcontroller will take in data from Camera (2D matrix of pixels), LiDAR (1D array), wheel encoder (2 values from both wheels), and gyroscope (1 single value). The microcontroller will store current frame of the camera as 2D matrix of pixels containing RGB values through USB. Readings from Lidar through USB are distances will be converted to relative position in robot's frame, and then convert to world frame, with origin at start point. The value passed by wheel encoder through analog to digital converter will need to converted to an average linear velocity. The gyro value will also be read through analog to digital converter and converted to actual angular velocity.

Requirement: The microcontroller should be able to receive data from all sensors, including LiDAR, camera, gyroscope, wheel encoder, and be able to send outputs to motors of robotic arm and motors of the wheels.

2.3.4.2 Algorithm for Localization Calibration

Due to the continuously accumulated error from dead-reckoning system, a way of calibration has to be performed. Everytime the robot has to starts from home position, which is the origin of world frame. Everytime the robot finds the object, the location of the robot will be calibrated, because the position of the objects are specified beforehand.

Depending on how previous calibration works, we might add waypoints at each corner, so that each time the robot receives a command, it will go to the nearest waypoint first for calibration, and then go to the object. At the corner, we'll have a red paper on one side, and a blue paper on the other side. The color in the image

will determine which side the robot is facing to correct gyro, and the fact that the robot is at the corner will correct the position.

Requirement: The robot will successfully correct its position and orientation with an error range within +/- 5cm and +/- 5 degrees at waypoints, the origin and the location of the target object.

2.3.5 Sensors

2.3.5.1 LiDAR

The LiDAR will pass in an array of distance in different angles around every 0.2 seconds. The data will be sent to the microcontroller and be continuously used for obstacle avoidance.

Requirement: The robot will receive the array of LiDAR and successfully produce the map with obstacle, and continuously update the map based on its location. Because of the grid environment, the obstacle should be shown up exactly the same as real environment.

2.3.5.2 Camera

The camera will output a 2D matrix of pixels, where each pixel contains its RGB value. However the camera will only be used when the robot gets close enough to the object. The position of the object is given, and the robot will find its only path to that object, but when it gets around 0.5m away from the object, the camera will be used to calculate the center of the object (with Green cap) in the frame. And then the robot will adjust its position to facing the object directly, following by go straight and pick it up. If the memory is not enough for the microcontroller, we'll change the camera to grayscale to save memory. The camera will also be used when calibrating the location of the robot, but only to check the color the robot is facing to.

Requirement: The robot will correctly identify the object, and report the object identified signal to user end.

2.4 Proposal 2 (lower part)

2.4.1 Mechanical Unit

2.4.1.1 Motor

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 0.9 m/s and the working voltage is 3-12V DC.

Requirement: The motor should provide enough torque to start the car with all the components on it, and output the speed and direction precisely controlled by the controller's voltage signal.

2.4.1.2 Chassis

We need a chassis vehicle as the main body of our robot that can be assembled with four wheels and is able to carry all other components, including microcontroller, PCB, battery and all sensors. The material of our chassis is aluminum and the size is 200mm x 170mm x 105mm.

Requirement: The chassis should have great weight-bearing performance, and provide enough space and interfaces for all the components.

2.4.2 Control Unit

2.4.2.1 Motor Controller

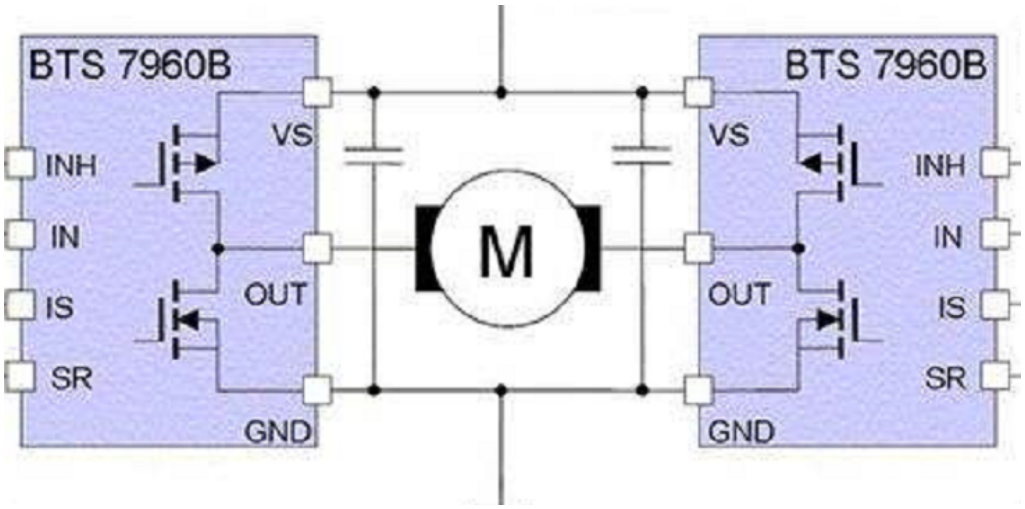

Figure 8: Schematics of motor controller [3]

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 control since we need 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).

Requirement: The motor controller part must send precise voltage to the motor according to the signal it received from the higher level controller.

2.4.2.2 Protection Circuit

We will design a protection circuit in the PCB to prevent our circuit from any misconnection of the power or overload.

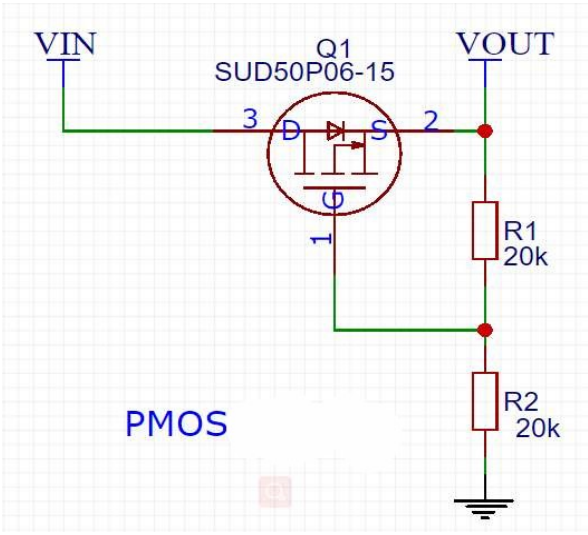

Figure 9: Schematics of protection circuit

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.

Requirement: Protect the whole circuit from damaging by mis-operation.

2.4.3 Power Supply

2.4.3.1 Battery

We will use 4S Li-po rechargeable battery(14.8V, 7000mA) as the power source for the whole vehicle including the motors, controller, the signal receiver, and the sensor system. To power the different systems we need to connect the battery to the pcb board which will convert the voltage of the battery 14.8V for different components.

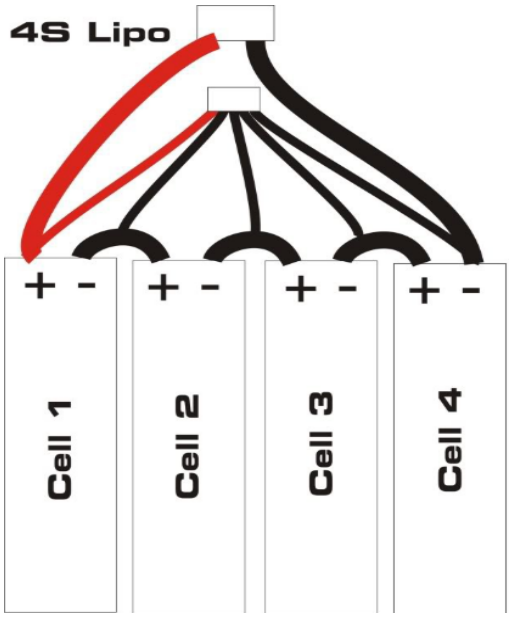

Figure 10: Battery overview

Requirement: The battery should provide steady power for our components and work safely.

2.4.3.2 DC-DC Converter

We are going to use the 4S Li-Po battery whose voltage is about 14.8V. The controller we use is raspberry Pi, which works at 5V. The motors work at 3-12V. Therefore we need converters to output different voltage for the different components in our robot.

Due to the components working voltage range is lower than the voltage that the battery provides, we choose to use Buck DC/DC converter, which can step down the dc voltage. The Buck converter needs the TPS62130A chip as the center component. The circuit layout shows below is an example for 12V input to 3.3V output.

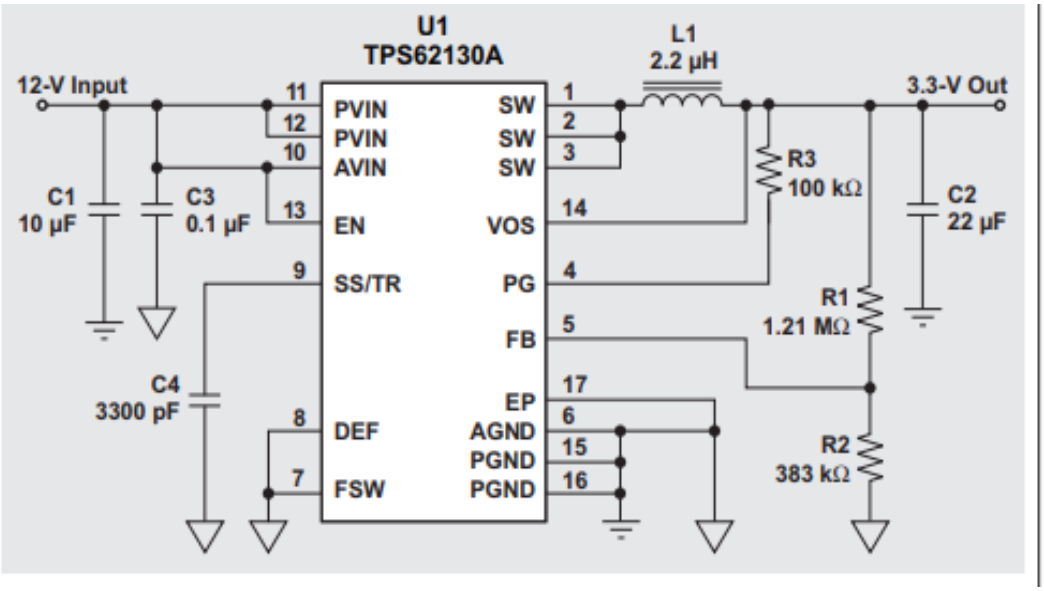

Figure 11: Circuit layout of TPS62130A [4]

Requirement: The converters on the PCB need to work individually without any disturbance to each other. The whole circuit should generate heat in a normal amount range.

2.4.4 Algorithms

2.4.4.1 Localization

With the linear velocity and current rotation we could get from wheel encoder and gyro, it's plausible to calculate the current position in the previous robot's frame. Once we get this vector, we could convert it to world frame and add to previous location of the robot.

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

x, y, theta are current location regarding to world frame, and theta_old is the old total rotation. V and w are current linear velocity and angular velocity of the robot respectively; while t is the time difference between the data published. x_old, y_old and theta_old are previous location.

Requirement: The robot should report its current location with less than 0.5m error within 10 min after the robot starts from home.

2.4.4.2 Speed Control

User will set the desired speed for the motor, with the proportional control and feedback from wheel encoder, the linear velocity should stay around the desired speed.

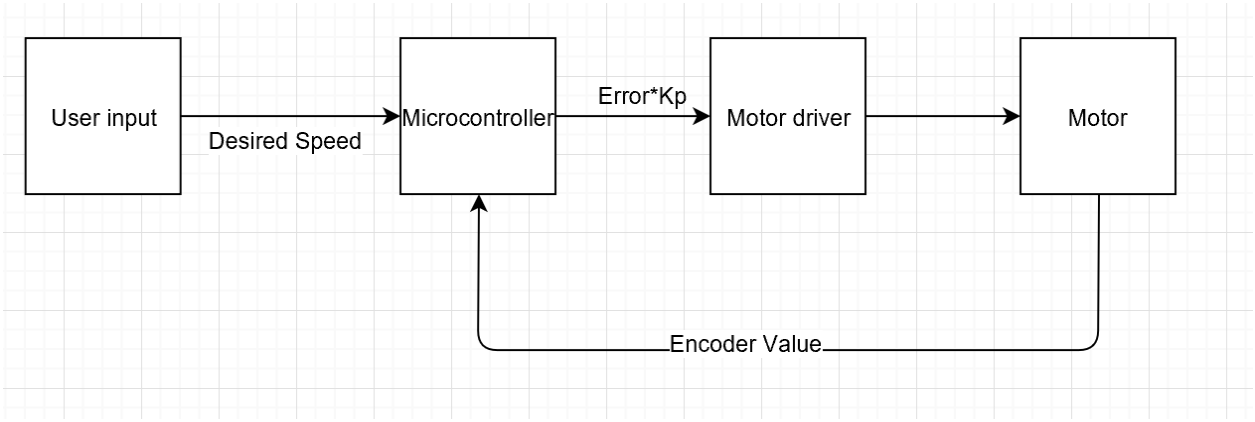

Figure 12: Flow chart of speed control

Requirement: The robot should be able to go along a straight path with the user specified velocity +/- 0.1m/s.

As for turning, with a given angle, the robot should turn that angle, while the position stays the same. This will also be accomplished by the proportional control, and the rotation direction for wheels on two sides will be different.

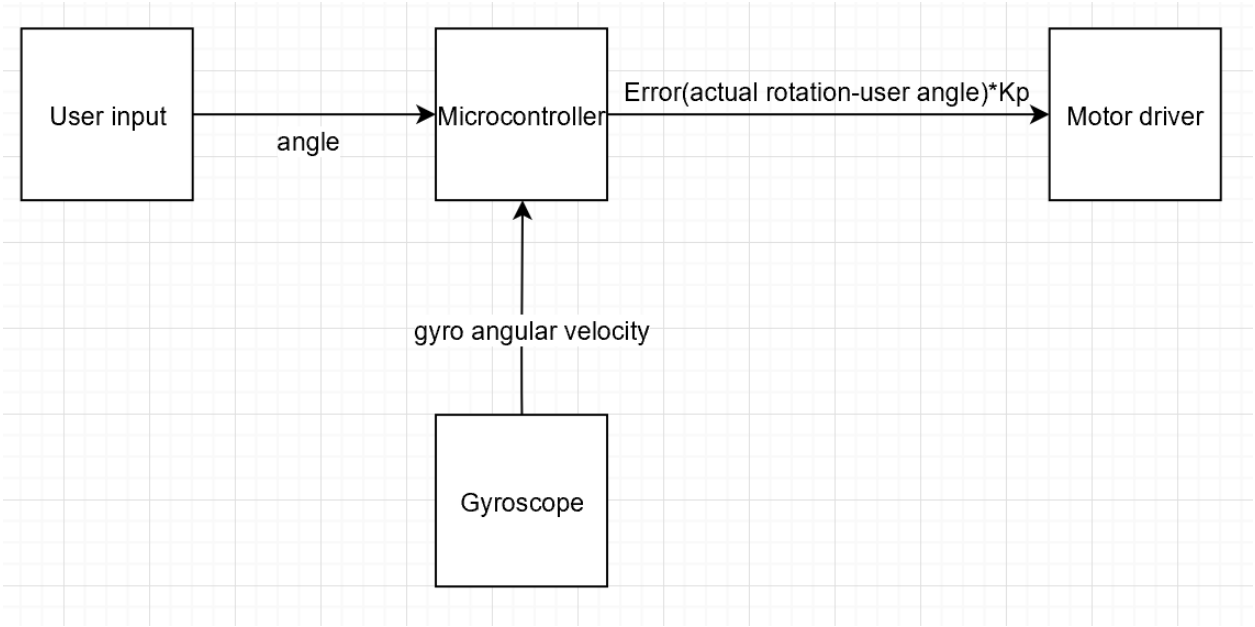

Figure 13: Flow chart of angle control

Requirement: The robot should turn the user desired angle with +/- 5 degrees error. And the position after turning should be within 0.03m from the position before turning.

2.4.4.3 Object Detection

The camera will continuously stream photos back to the microcontroller and we will setup a time interval to regularly fetch image data but not consume all memories of the controller. Then we will change the picture to grayscale image or directly get grayscale image in order to further cut down the usage of the memory. Because a bundle with high-saturation color will be stick to the target object, our implemented convolutional neural networks (CNNs) could clarify the region with prominent color quickly. The microcontroller will calculate the vector from the center of the image to the mass center of the claried region and then adjust the vehicle to move closer to the target object in order to pick it up.

Requirement: Utilize at most 5 masks to recognize the object and calculate the relative position to the robot.

2.4.4.4 Obstacle Avoidance

The basic path planning has already circumvent all the obstacles. Nonetheless there will be situations that the robot will crush into obstacles. As an emergency method, basic obstacle avoidance will be implemented. If the LiDAR has detect something closer than 20cm to the robot at front, the robot will turn left or right based on the final goal location until there are no obstacle at front.

Requirement: The robot will bypass all obstacles if they are at front, and not even a touch.

2.4.4.5 Path Planning

This is a known environment, 2 dimension and known target problem. So one solution is to go through A* search every time the robot encounters an obstacle. This dynamically computation makes sure our path planning is up to date. If there is no possible path between robot and object, the robot will go back to its starting position and send back a signal. Moreover, consider such case, if there are three obstacles that lies in front of our robot. Two of them locates closely that the robot cannot pass through and another two of them has a relative bigger space, which is big enough to pass through. Our robot will choose a safest path, which means the path passing through the bigger space using free segments and turning points algorithm.

Requirement: The robot should go to the user input object location without crashing into obstacles. The final goal the robot reaches should be within 0.1m from the actual designated point.

2.5 Risk Analysis

In our project, there will be several risks that we need to come through.

The major problem we will face is the localization of the robot. Gyroscope sensor could drift dramatically and affect the precision of the localization severely. What's more, two motors for the wheels could rotate in different speeds even under the same power supply. In this case, we decide to first take several careful tests for the gyroscope and encoder to make sure that their data won't drift away too much in short time. Then we would speed control to minimize the speed difference between two motors. What's more, by using local motion calibration algorithm and our LiDAR and camera, the robot could recalculate its position in 2D map. We design to ask the robot to implement local motion calibration after finishing one task given by the user.

Another problem we need to pay attention is the memory cost. We only use one Raspberry Pi as our microcontroller but we have at least four motors, two sensors, one camera and one LiDAR that should be connected to this controller. This controller is also required to run more algorithms at the same time. Thus, memory of the Raspberry Pi will soon be consumed if don't be cautious about using memory.

Definitely, we should try to mitigate the memory usage for every softwares as much as possible. Our robot could also storage the data that isn't in high priority in the server to further cut down the usage of memory.

Improper physical design could be a huge trouble for us still. The robot should balance itself while shipping the object to the destination. During this process, it would probably encounter new obstacles and implement the obstacle avoidance algorithm after. It is possible for it to turn around so fast or stop immediately for avoiding the obstructs and lose its balance as well as drop off the object finally. Thus, we need to cautiously control the speed of the wheels and shorten the processing time between the moment of finding a obstacle and the time sending the signal to stop the wheel. We will put additional weight on the vehicle in order to stabilize the whole robot and control the speed at the same time.

2.6 Hardware components

LiDAR: YDLiDAR or RPLiDAR A2
Camera: Diatone 600TVL 120° Mini Camera Black
Gyroscope: Sparkfun MPU6050
Motor with encoder: 2WD Beginner Robot Chassis
Microcontroller: Raspberry Pi
Robot Arm: DC Motor-37 12V/50RPM, Strong Robot Gripper
VR: Oculus Go with controller

3 Safety and Ethics

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.

We will use 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.

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.

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. [5]

We will follow the electricity using manual during the experiment, and will also check our 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” [6]. 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.

4 Stretch Goal

- VR
- Automatic object detection: Automatically find objects and put them in base station without user specify object location
- Moving obstacles detection

Reference

[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].