ROBOTIC WAITER FOR RESTAURANTS

By

Cheng Jin

Jun Pun Wong

Kausik Venkat

Final report for ECE 445, Senior Design, Spring 2018

TA: Xinrui Zhu

02 May 2018

Project No. 26

Abstract

This report describes in detail the design and functionality of our project; the robotic waiter for restaurants. It is a robotic assistant used to help restaurant staffs to deliver food to tables in the restaurant. Our robot is able to get instructions from a central microcontroller (in the kitchen), calculate its path to the destination and navigate its way there. The central microcontroller will also be connected to a display that will show the table that is currently being served. The final implementation which was not completely functional - we could not integrate the arduino software with the physical robot; however the individual parts work.

Contents

1. Introduction **3**
1.1 Purpose 3
1.2 High level Requirements 4
1.3 Block diagram 4

2. Design **5**
2.1 Physical Design 5
2.3.1 Motor and Motor driver 7
2.3.2 Encoder 9
2.3.3 Arduino Software 10
2.3.4 Arduino-to-Raspberry Pi communication 13
2.3.5 Path-finding algorithm 14

3. Design verifications **16**

4. Cost **20**
4.1 Parts 20
4.2 Labor 20

5. Conclusion **21**
5.1 Accomplishments 21
5.2 Uncertainties 21
5.4 Future Work & Improvements 22

References **23**

Appendix A Requirement and Verification Table **25**

Appendix B Schematics **29**

1. Introduction

1.1 Purpose

The purpose of this project was to create a robotic waiter that would assist restaurant staffs in delivering food. Manpower is still an issue in the industry with workers being inefficient and having no-shows. They would still need to be trained and paid. We also estimate that in the long term, robots would be cheaper (fixed cost at ~$1500) versus paying workers minimum wage (~$1000 monthly). With our robots, restaurants could hire fewer staff and have lesser problems.

In our project, the microcontroller on the robot would be able to receive instructions from the Central Unit - the microcontroller in the kitchen on where to deliver the food. It would digest that information and communicate with the Raspberry Pi which would calculate a path for the robot to take. The microcontroller in the kitchen also displays the tables being served. Due to time constraints, we mapped out the senior design lab and the benches (supposed to simulate a restaurant) into our software.

1.2 High level Requirements

- Send information from the kitchen's MCU to the robot's MCU and vice versa.
- Given a destination, the robot is able to calculate and navigate a path to its final destination.
- Kitchen's MCU is able to display the list of tables our robot is serving to on the LED display.

1.3 Block diagram

The project can be divided into two main components; the Kitchen Unit (Central Unit) and the robot (Navigation Unit). As shown in Figure 1, both units have its very own power distribution unit. In the power distribution units, 9V batteries are used to power the units. Specifically, 36V is used for the Robot and 9V is used for the kitchen. The power distribution unit also has voltage regulators and LED indicators. The voltage regulators are to provide a stable input voltage to the components of each unit. The LED indicator functions as a safety feature as to indicate that the circuit is powered.

Next, in the Central Unit, there is a microcontroller unit (MCU), Bluetooth module and a display and user interface. The display is used to display the current order number and to display if an instruction has been executed successfully. Then, the user interface is used by the user to provide instructions on which table the robot should go. The Bluetooth module is used to facilitate communication between the robot and the Central Unit. The MCU is used to decide when the next order should be sent depending on the robot's execution status.

Lastly, in the Navigation Unit, there is a MCU, Bluetooth module, motor, encoder and a Raspberry Pi. The Raspberry Pi is used to perform the path-traversing algorithm. As the ATMEGA328P chip used as the MCU has insufficient memory for the aforementioned algorithm execution, the Raspberry Pi is solely used for this function. The encoder is used to calculate the distance travelled by the robot and the motor is used to move the robot. The MCU on the robot is used to handle the main controls of this unit. Lastly, similar to that of the Central Unit, the Bluetooth module is used to transmit and receive data from the Central Unit.

One addition to the Design Block Diagram from the design document is that we decided to implement the path-traversing algorithm using A-star (A*) algorithm. Hence, this requires that we use Raspberry Pi for the aforementioned memory issues.

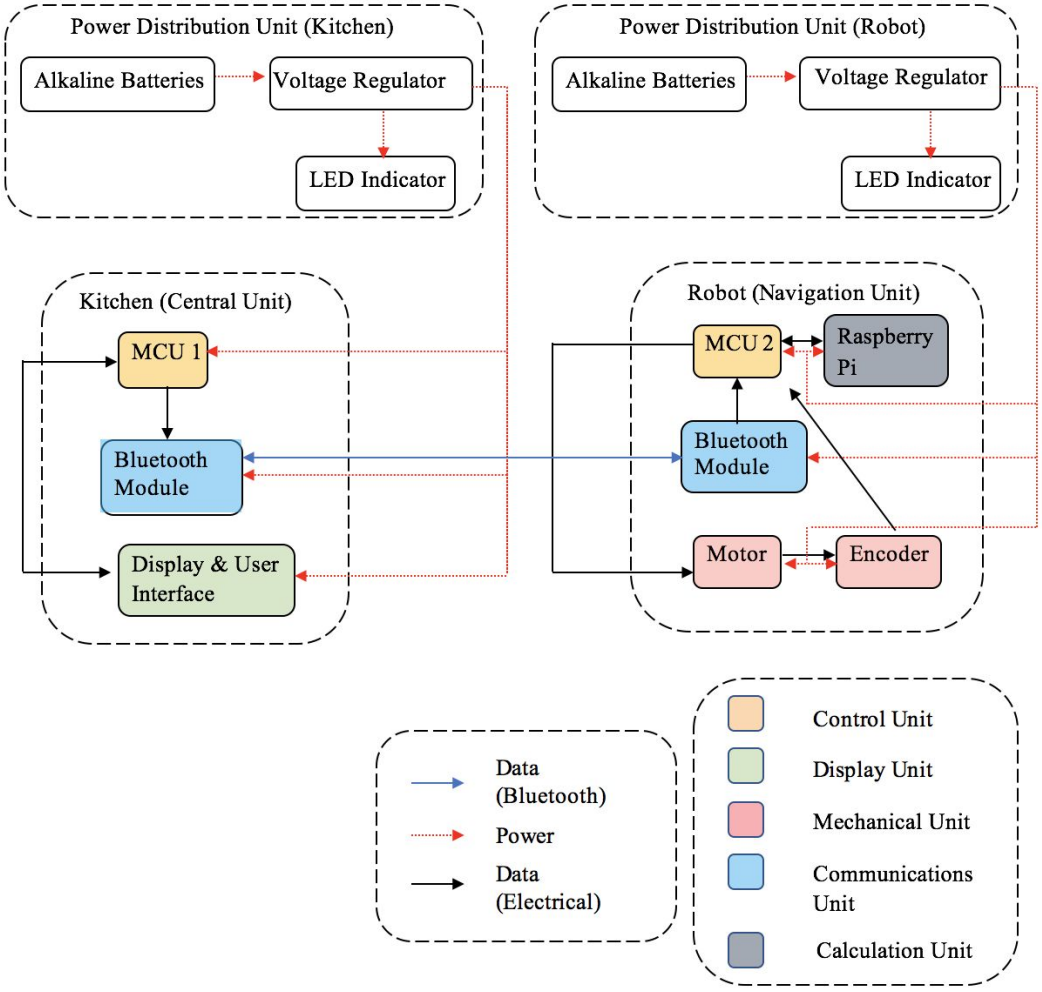

Figure 1. Block Diagram

2. Design

2.1 Physical Design

The structure of our robot is similar to that of a cart. As shown in Figure 2, it has two 6 inches drive wheels at the front sides of the robot and two caster wheels in the back. Having three levels of compartments, the bottommost compartment is designed for all the electrical and mechanical components. The power unit and adaptor are placed on the middle compartment. Lastly, food and other loads are placed at the topmost compartment.

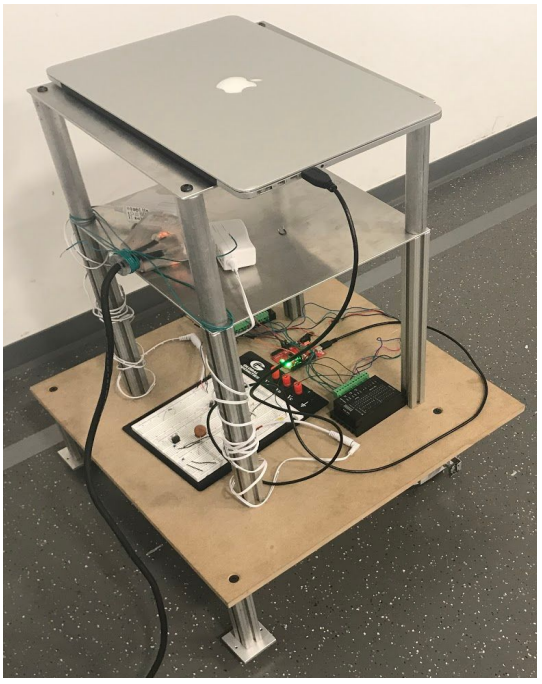

Figure 2. Physical Design of Robot

As shown in Figure 3, both motors and wheels are at the front sides of the robot. Through this configuration, the encoders can be fixed outside of the wheels.

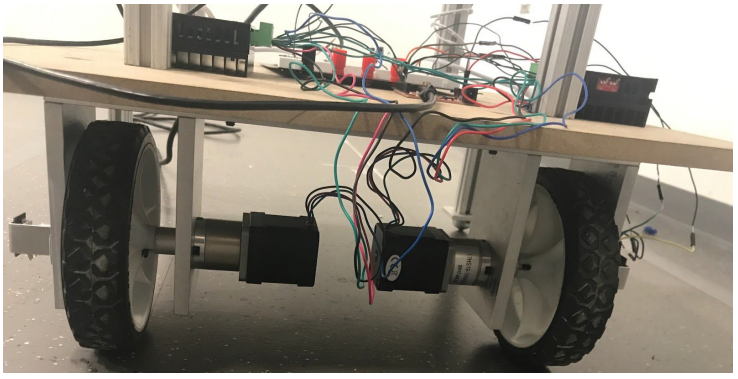

Figure 3. Motor and Wheels of the Robot

2.2 Power Distribution Unit

There are some design constraints that had to be adhered to, while designing the Power Distribution Unit. First, the motor driver requires an input voltage of at least 20V and an input current of at least 1.7A. Second, both MCUs in the Central Unit and the Navigation Unit operates in voltage between 1.8V and 5.5V. Third, the Bluetooth modules used in the Central Unit and the Navigation Unit requires and input voltage of 3.3V and a current of approximately 50mA. Lastly, the Raspberry Pi requires 5V of input voltage and 3A of input current.

Therefore, we have decided to use 9V 560mAh batteries are the batteries can be easily obtained. An input voltage of 36V is used on the robot and a 9V input is used on the kitchen unit. The aforementioned input voltage on the robot is obtained by connecting 4 9V 560mAh batteries in series.

There are 3 voltage regulators used to provide steady DC voltage to the components of each unit. First, we used the MCP1700 3.3V voltage regulator to power the Bluetooth module. Second, we used the L7824C 24V voltage regulator to power the motor driver. Lastly, we used the MC7805CT 5V voltage regulator to power the ATMEGA328P MCU and the Raspberry Pi.

$$Battery\ Life\ (hour) = \frac{Battery\ Capacity\ (mAh)}{Load\ Current\ (mA)} \times 0.7 \quad (1)$$

As shown in the battery life calculation formula in equation (1), 0.7 is used to account for external factors. For example, manufacturing tolerance of the battery and the internal resistance of the battery. Therefore, using equation (1), by drawing 1.7A using the 24V Voltage Regulator onboard the robot, we estimate that the battery life on the robot to be 0.23 hours (equivalent to 13.8 minutes). Consequently, we had to resort to powering the robot through an adapter that converts the 120V AC voltage from the wall plug to a DC voltage of 24V. In contrast, for the Central Unit, the display requires 2.5mA, the bluetooth module requires 50mA and the MCU requires 0.2mA. Therefore, using equation (1), the battery life is estimated to be 10.6 hours (equivalent to 636 minutes).

2.3 Robot (Navigation Unit)

2.3.1 Motor and Motor driver

We chose two bipolar stepper motors for robot, because stepper motor has very small error margin. Stepper motors also operate in a large range of voltage. However, since the speed of the motor is related to the torque, we also chose to use stepper motors that has gearbox. In this project, we used the stepper motors from the 17HS19-1684S series from StepperOnline. This is because that motor came with a gearbox and had various gear ratios. Using a gear

ratio of 19:1, the motor could provide speed range from 4 to 39 RPMs which was enough for us to accelerate the robot from a relative low speed. This is shown in Figure 4.

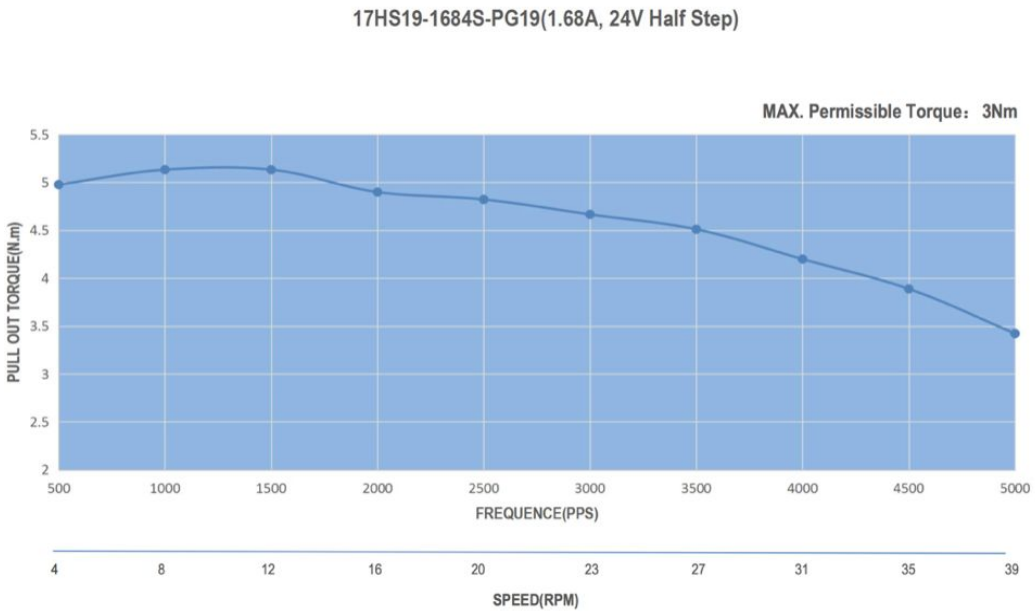

Figure 4. Torque Curve of 17HS19-1684S-PG19 Stepper Motor[1]

To interface between the circuit and the motor, we chose the MYSWEETY TB6600 4A 9-42V Stepper Motor Driver. This is so that the motor is able to receive the correct input voltage. Since the stepper motor requires 1.68A rated current, we configure the driver to provide the motor with 1.7 A.

$$Step\ Angle = Motor\ Step\ Angle \div Micro\ Step \tag{2}$$

Using Equation (2), the final step angle is calculated. Due to the fact that our motor uses a gear box, the step angle is too small to output such relatively high torque. Therefore, we used 1 micro step for our driver.

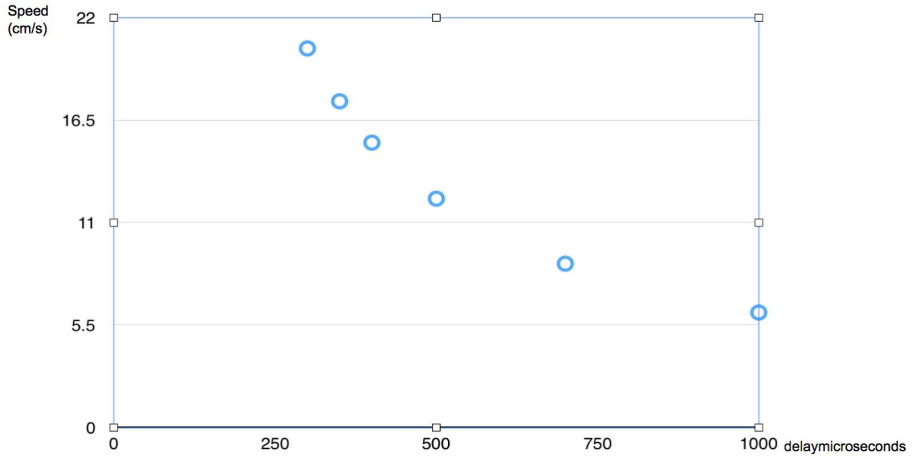

Figure 5. Graph of Speed vs Delaymicroseconds

The relationship between the speed of the motor and the delay time can be represented using a hyperbolic curve as shown in Figure 5. Since the slope is not a constant, we used step functions to simplify the acceleration code. In Figure 6, we divided the whole movement into three parts: acceleration, stable part, deceleration. The robot can accelerate to highest speed after several steps. We increased the number of steps to make acceleration more smooth.

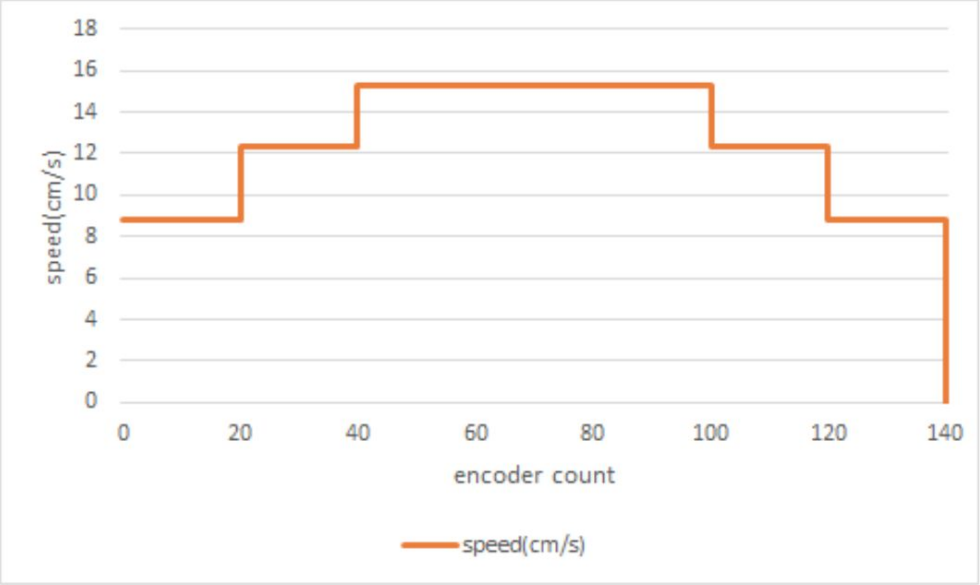

Figure 6. Speed vs Encoder Count Curve of Acceleration

2.3.2 Encoder

The encoder is connected to the motor. It is used to calibrate the rotation speed of the wheels as well as calculate the distance the robot has moved (using number of rotations of the wheel). Since the encoder can provide information about the robot's position and this can be accessed in the Arduino code, there is no need for sophisticated equations to determine speed and time. We also chose the KY-040 rotary encoder because it was cheap and easy to use. One rotation of the knob on the encoder is equivalent to 30 counts. Inside the encoder there are two switches: A to C and B to C, as shown in the left figure in Figure 7.

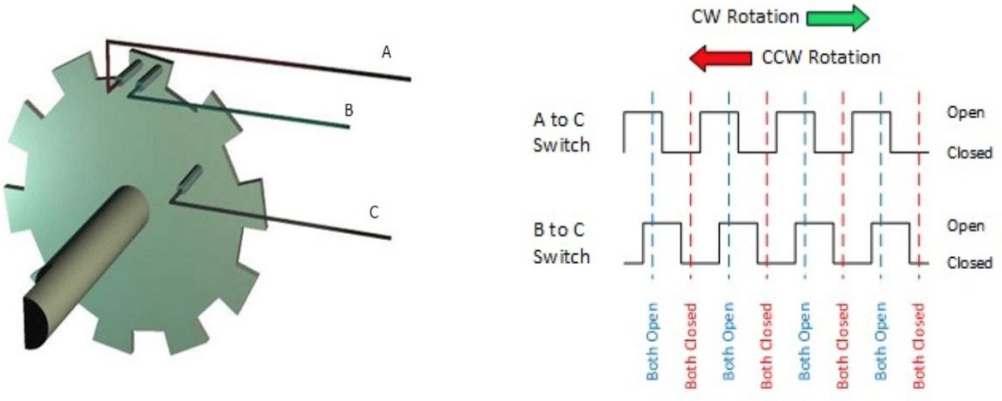

Figure 7. Encoder Switches and output pin signals[2]

When the encoder rotates, this two switches will change from open to closed or from closed to open, creating square waves on pin A and pin B, as shown in the right graph of Figure 7. When both the switches is open or closed, the rotation is recorded and we renew the count. If the knob is rotated clockwise, A to C Switch will open first and the encoder will increase the count. Otherwise, B to C Switch will open first and encoder will decrease the count.[7]

2.3.3 Arduino Software

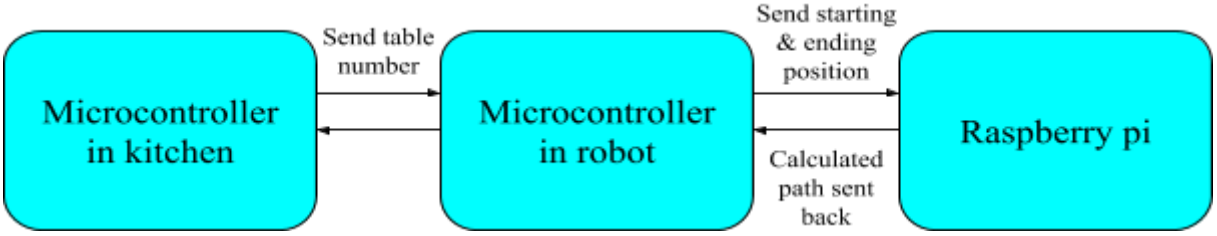

Figure 8. Overview of information relay in entire robotic waiter system

Figure 8 shows the overview of how information is distributed in our robotic system. The microcontroller in the kitchen would send the destination (a table number) to the microcontroller on the robot. Then, it would send the starting position and the destination to the Raspberry Pi which would calculate the optimized path to take and send it back to the Arduino.

Figure 9 shows the flowchart consisting of the various states that the microcontroller in the robot can be in. If it is in "WAITING", the robot is either waiting for a new delivery at its homebase (kitchen) or it has reached its destination and is waiting for the food to be served. If it is in "SEEKING PATH", it has received a new destination to travel to and has already sent the coordinates to the Raspberry Pi which is still calculating the path. The only way to exit this state is for the Raspberry Pi to send the encoded path back to the Arduino. The 'DIRECTION HANDLER' is the intermediate state before every action, whether turning or moving straight, takes place. In Figure 11, this state is also the exit condition once the robot has reached its destination. "TURN", "ACCELERATE", "MAINTAIN" and "DECELERATE" are action states where the robot is moving.

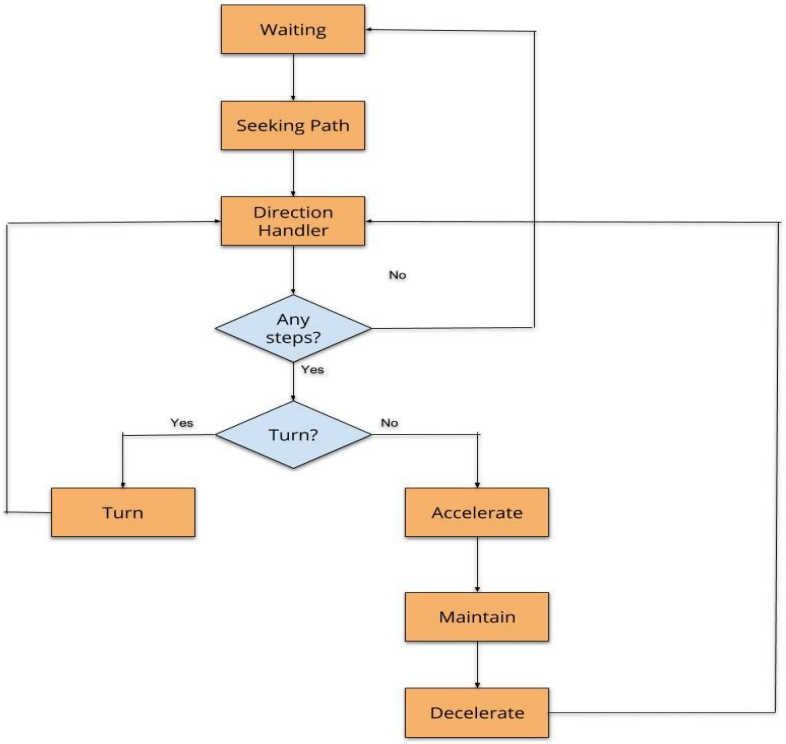

Figure 9. Flowchart for microcontroller in robot

In the "WAITING" state in Figure 10, the first course of action would be to check if the robot is at homebase or not. This would result in different chunks of code run. If the robot is at its homebase, it's waiting for the next instruction from the kitchen microcontroller. It will check that from the `getDestination()` function call. If a valid table number returned from that, the robot would update `message_to_pi`, which would be sent to the Raspberry Pi through an interrupt call (not shown in Figure 12) and it will move to state "SEEKING PATH". If the robot is at a table making a delivery, we would run `getButtonValue()` to check if the button has been pressed to indicate that the delivery has been made. If it has, it would update `message_to_pi` and change state to "SEEKING PATH" like before.

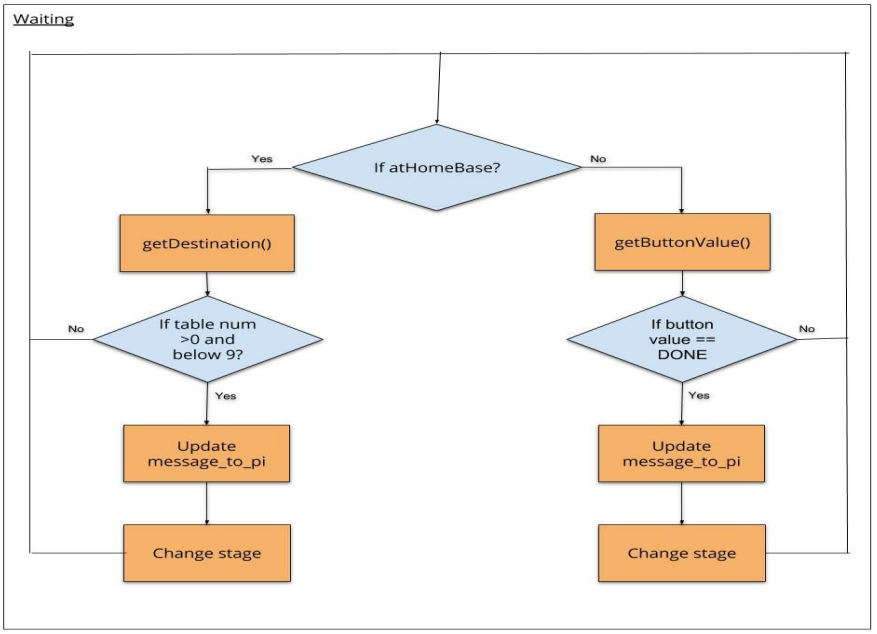

Figure 10. Flowchart for the "WAITING" state

In "SEEKING PATH", the code will run in a loop that will only break when it receives the entire encoded message from the Raspberry Pi. More information on this would be covered in section 2.3.4.

Figure 13 below shows the flowchart for the the "DIRECTION HANDLER" state. Each run through the state only happens for one action sequence. First, the letter is extracted from the `message_from_pi` to identify what that action is. If it is 'D', then we have come to the end of the instruction and the robot has reached its destination. So we would flip the boolean `atHomebase` to recognize that and move to WAITING state from earlier. If not, we would extract the number that followed the letter. If the letter was 'S', then we would calculate and set pre-calculated distances for the robot to travel in the "ACCELERATE", "MAINTAIN" and "DECELERATE" states and set state to "ACCELERATE". If the letter was 'T', then we would set `turn_count`, the amount the robot has to turn and set the state to "TURN".

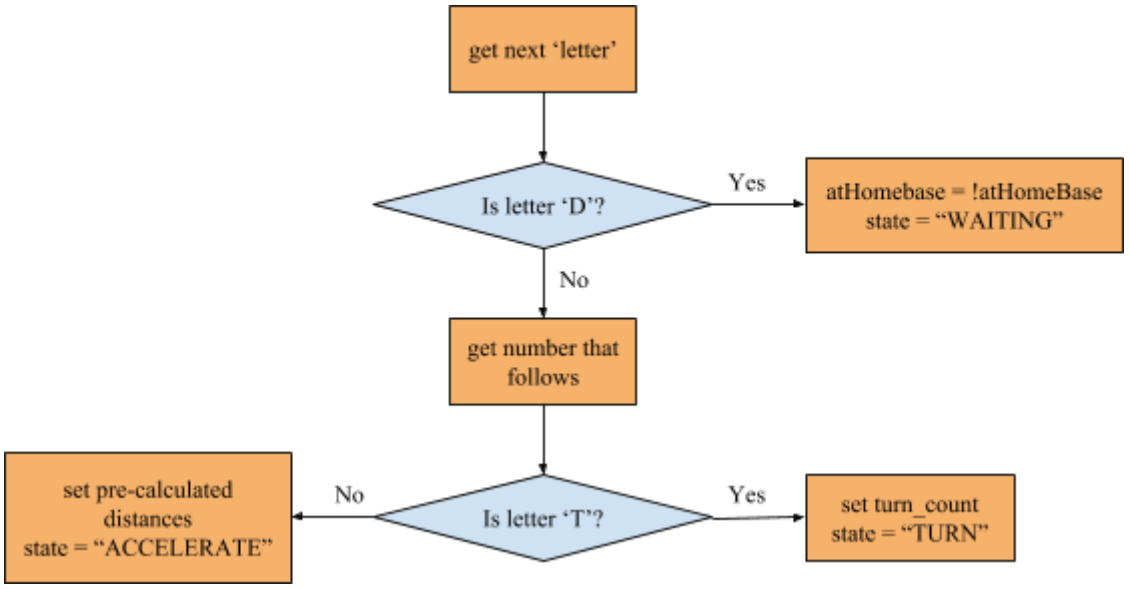

Figure 11. Flowchart for 'DIRECTION HANDLER' state

If the robot has reached the "TURN" state, we want it to keep turning until the turn_count runs out. Since we don't want the arduino loop to keep waiting until the entire turn happens, in each `loop()` we would execute the turn action for 100 counts or until turn_count is goes to 0.

If the robot has reached the "ACCELERATE" state, that it means it is ready to start moving forward. We wanted to ensure that the robot moves at a fast pace but at the same time, we could not immediately shoot the speed of the robot because it might damage the motors so we have an "ACCELERATE" and "DECELERATE" phases. The part of the motion when the robot moves at its max velocity is called "MAINTAIN".

The logic in each of these three states are actually pretty similar. First, we would move the robot and then update the encoder counter. If the encoder counter has exceeded the `max_encoder_value` which was pre-calculated in "DIRECTION HANDLER", then we would move to the next state. For "DECELATE" this would mean moving to "DIRECTION HANDLER" state.

2.3.4 Arduino-to-Raspberry Pi communication

Communication between Arduino and Raspberry Pi is done through i2c communication protocol. I2c communication protocol allows one "master" chip to connect with one or "slave" chip. But in our scenario, we only need "slave" chip. The configuration for this can be seen in figure 14 below. Since Raspberry Pi is the master, it would keep send requests to the Arduino when it wants data but because in our setup, Arduino sends the coordinates, the Arduino would send a blank message whenever it does not need any calculations. When the arduino needs computations to be done, it would send a message with the starting and ending coordinates that the Raspberry Pi would be trained to identify and perform the `calculate_path` and return the solution.

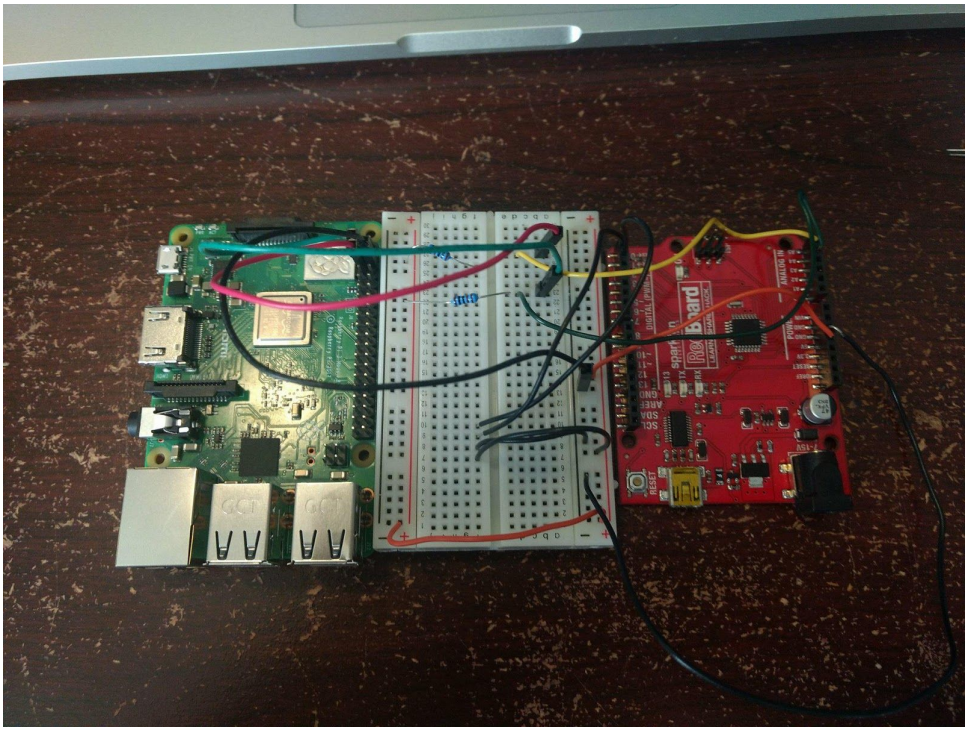

Figure 12. Arduino Uno and Raspberry Pi i2c connection

2.3.5 Path-finding algorithm

We plotted the room in a 2D grid where each square is a 1cmx1cm square. We also included all the obstacles in the room and the tables and the delivery locations. This allows for us to quickly navigate there. Astar graph search is used to calculate the path taken. We went with this over Breadth-First-Search due to its speed and also the ability to include our own heuristic function. This heuristic function allows us to minimize the amount of turning done.

2.4 Kitchen Unit (Central Unit)

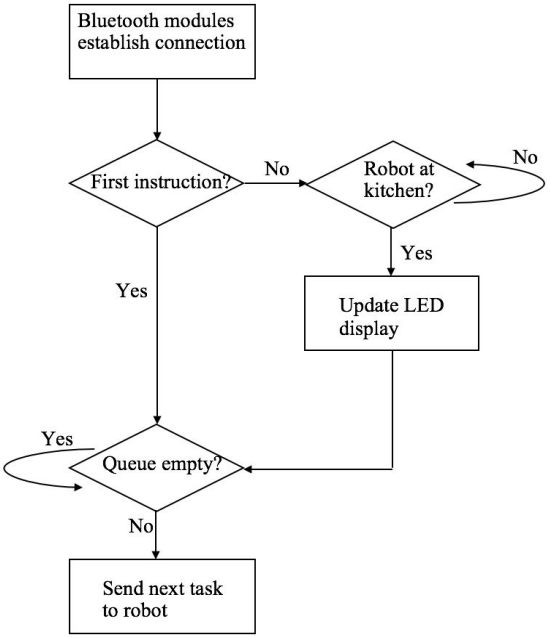

Figure 13. Algorithm of the Kitchen Unit Software

As mentioned in Section 1.3, the main function of the Kitchen Unit's MCU is to decide on when the Kitchen Unit should send the next instruction to the robot. Therefore, the first step is to establish a Bluetooth connection via the Bluetooth modules between the robot and the Kitchen Unit. Next, the software checks if the robot has executed the first instruction. The reason for this is because if the robot is not executing the first instruction, the MCU has to change the LCD display when the robot returns to the kitchen. The kitchen knows if the robot is at the kitchen when the Bluetooth module in the Kitchen Unit receives a flag from the Bluetooth module in the robot. The kitchen would then check if the queue is empty. If the queue is not empty, the kitchen would send the next task to the robot when the robot is at the kitchen.

3. Design verifications

3.1 Power Distribution Unit

To verify that the power supply from the voltage regulators meet our requirements, we connected the input of the voltage regulators to the output of the batteries. First, the input of the 3.3V voltage regulator is connected to 36V. This is achieved by connecting 4 9V batteries in series. Second, we connect the output of the 3.3V voltage regulator to a multimeter. Then through the multimeter, the output of the 3.3V voltage regulator is measured. Then, these steps are repeated with the 5V voltage regulator and the 24V voltage regulator. Lastly, the steps are repeated with the 3.3V voltage regulator and the 5V voltage regulator using an input voltage of 9V. This is because the input voltage of the Power Distribution Unit in the robot is 36V and the input voltage of the Power Distribution Unit in the Kitchen Unit is 9V. Shown in Figure 14, Figure 15 are the output voltage from the 3.3V voltage regulator, 5V voltage regulator and the 24V voltage regulator.

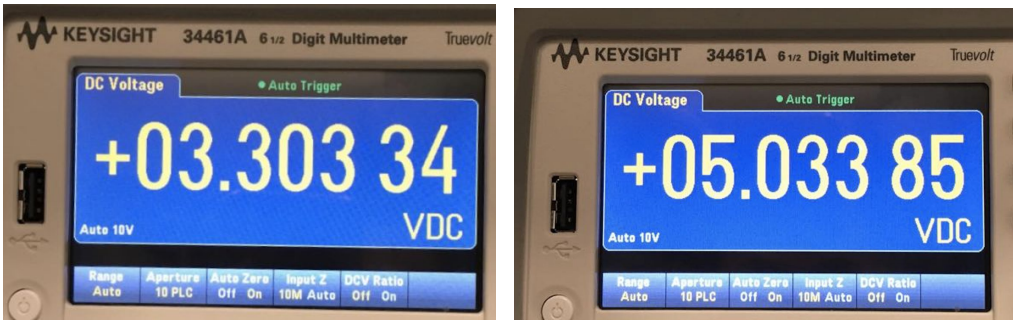

Figure 14. Output voltage of the 3.3V and 5V voltage regulator

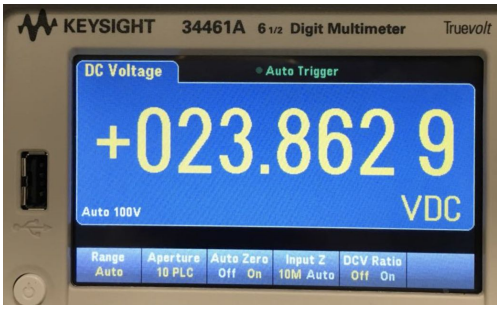

Figure 15. Output voltage of the 24V voltage regulator

3.2 Microcontroller in Robot

This unit controls the movement of the robot. It calculates the route and generate instructions to the motor driver. So the requirements of this part are focused on accuracy of movement. Hence, this part focuses on the calibration tests, achieved by decreasing the error in software. For convenience, we assume the direction in front of the robot is positive y direction. The horizontal direction is x direction. Robot's left side is positive direction. We use the right back corner of the robot as the starting point. We measure movement of that point to get the

error of the robot. There are three requirements for movement. First, error in displacement in x-axis should be less than 5 cm for every 1 meter moved in y-axis. This means that the robot should go straight in y direction. Second, error in movement in y-axis should be less than ±0.8cm for every 1 meter moved. Third, error in angle when turning should be within ±5°.

$$Distance\ (cm) = \frac{Diameter\ of\ wheel \times \pi}{Encoder\ count} \quad (3)$$

First, we allow the robot to go 200 encoder counts in different places of the test room. Using Equation (3), one encoder count equals to 1.6 cm as the diameter of the wheel is 15.24 cm and the encoder count for one rotation is 30. Using the same formula, 200 encoder counts equal to 319.2 cm. We marked the end points of right caster wheels and measured the movements in y direction and x direction. The two pictures in Figure 15 shows the result of this calibration.

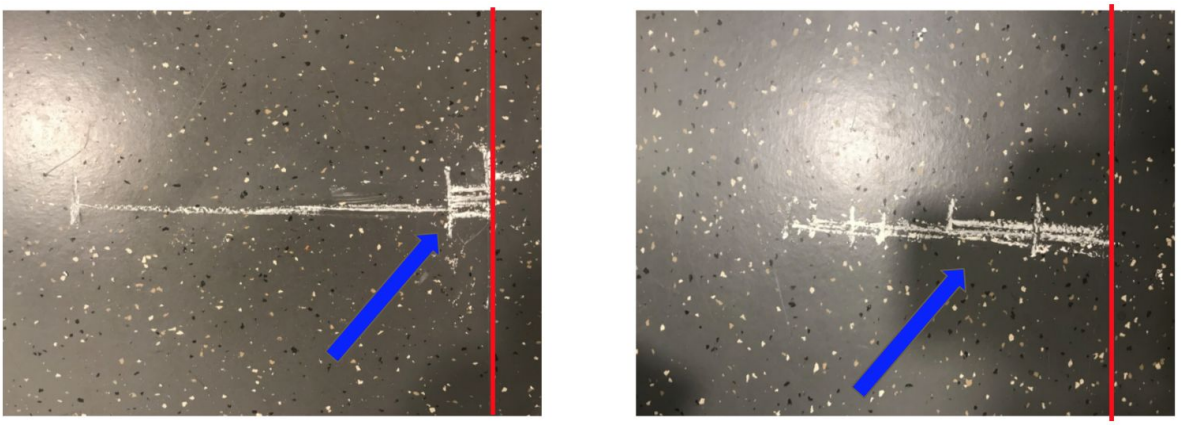

Figure 16. Two results of calibration test

The red lines on the figures is the y axis. The blue arrows point to the most frequent end point of the robot. In Figure 16, the left figure was taken at southern part of the test room. Right figure was taken at eastern part of the test room. As shown, the two figures gave us very different result. This may be due to the slight gradient of the floor. Small difference of inclination angles of floor could change the results of this calibration. Hence, we have to find a average value for a certain location. The errors in y direction are relatively small which goes from -0.4 cm to 0.8 cm. The reason for this is the encoder and motor stop backlash. One thing that we found is that the robot tend to slant towards the left side. This can be due to manufacturing errors such as the slight difference of diameter of wheels and the abrasion of the wheels. The result of the average error of the x direction was 6.67 cm, which is equivalent to 4 encoder counts. It is smaller than 5 cm per meter in movement. We adjusted the robot position for every 100 encoder count movement to decrease the error. After 100 encoder counts in movement, the robot will go right for 2 encoder counts.

To calibrate the turning of the robot, we also used the encoder count. When the encoder count equals to 16, the degree of turning was closed to 90°. However, this method gives us a very

large error margin; the error is bigger than 5°. Then we changed the code and used Arduino execution cycles to control when to stop, because the Arduino execution cycles have much smaller resolution as compared to the encoder. We used a counter and when the counter equals to 1934, the robot stops at 90°. We marked the midpoint between two front wheels and midpoint between two caster wheels in the back before robot turned. We marked the two points again after the robot turned. Then we connected the points in two straight lines as shown in Figure 17. The turning angle error is smaller than 3°.

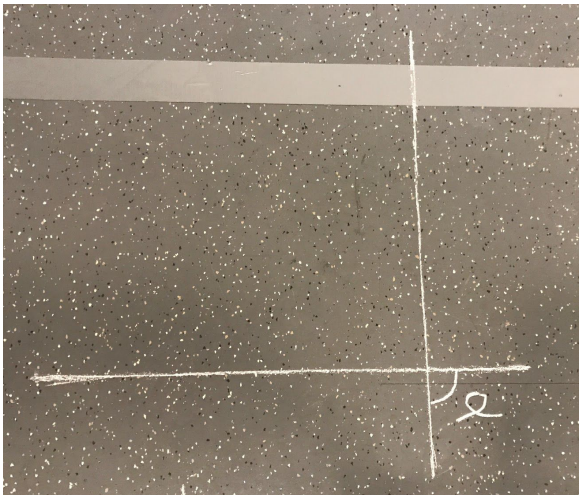

Figure 17. Result of turning code

3.5 Communications Unit

To verify that the Bluetooth module is able to transmit and receive data from 0.05m to 8m, we have created an Arduino program. In Figure 17, the first step is to establish a Bluetooth connection with the Bluetooth module. Using a phone application, I am able to establish a connection between my phone and the Bluetooth module. Then, the Arduino checks if 0 (in text) or 1 (in text) is received by the Bluetooth module. When the Bluetooth module receives 1, the Arduino will turn on the LED. On the other hand, when the Bluetooth module receives 0, the Arduino will turn off the LED. From this, I am able to change the state of the LED from a varying distance. The process of switching the state of the LED is repeated for distances between 0.05m and 8m.

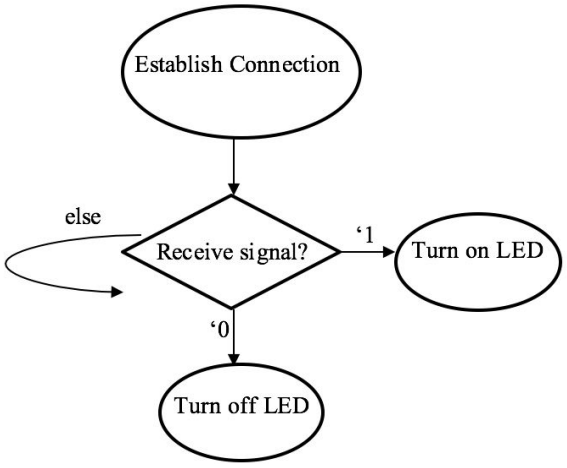

Figure 17. Algorithm of Communications Unit Verification

3.6 Motor

As what we talked about in design part, our motor should keep its velocity while carrying a relative big load. In our requirement, the motors should provide enough power to carry about 2kg weight in 10(±20%) RPM (about 10 meter per minute). Then we did the test with our robot. We put a dumbbell which was marked 5kg on the top of the robot. We let our robot run 200 encoder counts. One encoder count was $15.24\pi/30 \approx 1.6\ cm$. 200 encoder counts was 319.2 cm. We set the delay time between pulse as 600 microsecond in order to get a relatively large speed. Then we used stopwatch to measure the time for movement. Then we repeated the steps four times and calculated the average time. The result was 19.09s. So that the speed was 16.7142 cm/s, which was equal to 10.0285 m/s. And that just reached our requirement. Because delay time could not be smaller than 600 microseconds when we did the unloaded motor test, we didn't use that speed when we actually built our robot. Our motor can reach that speed for short-time use.

4. Cost

4.1 Parts

Part	Retail Cost ($)	Quantity	Total Cost ($)
Stepper motor	47.00	2	94.00
Stepper motor driver	14.99	2	29.98
Bluetooth receiver/transmitter	10.99	2	21.98
Rotary encoder	9.29	1	9.29
LCD 20x4 character display	19.95	1	19.95
Arduino Uno	19.99	2	39.98
Raspberry Pi 3 B+	35.00	1	35.00
24V AC to DC Adaptors	8.96	2	17.92
Heavy duty 50 Ft extension cord	21.99	1	21.99
DC Power Adaptor	8.99	1	8.99
		Grand Total	299.08

Table 1. Parts Costs

4.2 Labor

Name	Hourly Rate	Total Hours Invested	Total = Hourly Rate × 2.5 × Total Hours Invested
Cheng Jin	$30.00	150	$11,250
Jun Pun Wong	$30.00	150	$11,250
Kausik Venkat	$30.00	150	$11,250
Total	N/A	450	$33,750

Table 2. Labor Costs

Machine shop cost: $55/hour * 5 hours = $275

Total cost = $299.08 + $33,750 + $275 = $34324.08

5. Conclusion

5.1 Accomplishments

We did not manage to completely build the robotic waiter system. The kitchen microcontroller was able to take in input for a table and send that information to the robot's microcontroller through bluetooth. The software on the robot's microcontroller was able to gather that and pass the starting and ending coordinates to the Raspberry Pi which was able to calculate and encode a path back to the robot's microcontroller. Unfortunately, while the path-finding algorithm did work on smaller mazes (6x6, 10x10), it did not work for a the replica of the Senior Design Lab. While we were able to simulate that the robot's software was able to cycle through the various states correctly and we were able to control the motors and encoders on the robot to enable it to move straight or turn, we were not able to integrate both of them together. Also, the power distribution units of both the Central Unit and the Navigation Unit has provided the correct voltage and current to each components in the aforementioned units.

5.2 Uncertainties

Some hindrance that we are unable to overcome in this project is that the ATMEGA328P chips could not be bootloaded. This means that the process of loading the program data into the chip's memory has failed. This can be related to our PCB design. As we have failed to consider the miniscule sizes of the capacitors used on the Arduino, some of the pads tore off as a result of multiple desoldering and resoldering. Hence, some capacitors may not be attached correctly on the PCB. A solution that we propose is to have the chip bootloaded using a circuit implemented on a breadboard. In this way, we are able to make sure that the chip is fully functioning. Also, to counter this problem, use bigger components on the PCB to ease soldering.

Next, we have failed to consider that the required power far exceed the ability of our batteries. In other words, our batteries do not have enough capacity to run the robot for a significant amount of time. One solution is to scale down the robot and use motor drivers that requires smaller amount of voltage. By doing this, power sources may be cheaper and can be obtained more easily.

We also faced trouble with our software. While developing, the A-star path-finding algorithm was able calculate the path successfully for smaller mazes (grids of 6x6, 10x10 size). Unfortunately, it did not successfully calculate the path for the maze that reflected the Senior Design Lab. We did spend time trying to resolve that issue but we could not get to bottom of it. If we had more time, we would have used python stepper to step through the recursive function to see why the search function did not return a path.

5.3 Ethical Considerations

There are several safety concern in our projects. First, we have a power system on the robot which we need to regulate. We use power sources and AC to DC adaptor on our robot. The power adaptor has rise for aging and leaking. The adaptor and extend wire should be checked every year and exchanged if they are aging. This acts according to IEEE Code of Ethics Conduct #1 [].

Also looking at IEEE Code of Ethics Conduct #9, we need to ensure the robot's navigation is not harmful to humans around it. To ensure that, we plan on controlling the speed of the robot to a fairly slow pace - slower than humans' walking speed. If somehow, the speed crosses a certain range that we deem risky, we will have some circuit breakers or safety measures that will completely stop power to the robot so that it does not harm anyone by moving extremely rapidly.

5.4 Future Work & Improvements

There are several ways to improve our project. First, we suggest to integrate the path-finding algorithm with the robot movement code. In this way, we can have a fully functioning robot. Next, we can also use rechargeable batteries with higher capacity to power our robot. As rechargeable batteries with voltage above 20V are not common, we suggest to scale down the robot and use a motor driver that is able to operate in the range of 12V. Also, to save space, we also suggest that an electrical casing can be implemented to encapsulate all of the electrical components for safety purposes. If this step is to be implemented, space may be an issue and hence, we suggest to use a smaller bluetooth communication module that is able to change its function to be the slave (receiver) or the master (transmitter) readily.

For future development, we suggest building a charging port. Using IR beacons, the robot is projected to be able to return to its charging port once it has completed all the task. By implementing this, the users will not have to manually remove its battery and charge it before putting it back when the robot has to be used again. Next, we recommend implementing collision avoidance and collision detection. This is so that the robot can be used in a real-life setting safely. Lastly, we suggest to improve the path-finding methodology such that the tables and the obstacles would not have to be pre-programmed. By doing this, the robot is able to be used whenever the user wants it to be used.

References

[1] "Arduino – Sending A String Over Bluetooth Using The HM-10", Joshua Hrisko, Engineers' Maker Portal, 2017, [Online]. Available at: https://engineersportal.com/blog/2017/9/20/hm-10-bluetooth-module

[2] "ATMEGA328/P", Microchip, n.d., [Online]. Available at: http://ww1.microchip.com/downloads/en/DeviceDoc/ATmega328_P%20AVR%20MCU%20with%20picoPower%20Technology%20Data%20Sheet%2040001984A.pdf

[3] Bei Xuying, Ping Xueliag, Gao Wenyan, "Calibration of systematic odometry errors for wheeled mobile robots," Application Research of Computers, Vol. 35, No.9, Sept 2017. IEEE Code of Ethics, IEEE, web page. Available at: https://www.ieee.org/about/corporate/governance/p7-8.html

[4] "Bluetooth Interface with HM-10", Thrasyvoulos Karydis, 2015, [Online]. Available at: http://fab.cba.mit.edu/classes/863.15/doc/tutorials/programming/bluetooth.html

[5] "CSTCE16M0V53-R0", Murata Electronics North America, n.d., [Online]. Available at: https://www.murata.com/~/media/webrenewal/support/library/catalog/products/timingdevice/ceralock/p16e.ashx

[6] "HOW TO USE BLUETOOTH 4.0 HM10", Dan Chen, Instructables, 2015, [Online]. Available at: http://www.instructables.com/id/How-to-Use-Bluetooth-40-HM10/

[7] Keyes KY-040 Arduino Rotary Encoder User Manual, Henry's Bench, 2015, Available at: http://henrysbench.capnfatz.com/henrys-bench/arduino-sensors-and-input/keyes-ky-040-arduino-rotary-encoder-user-manual/

[8] "LMV358MMX", Texas Instrument, n.d., [Online]. Available at: http://www.ti.com/lit/ds/symlink/lmv358-n.pdf

[9] "L78L24C", STMicroelectronics, n.d., [Online]. Available at: http://www.alldatasheet.com/datasheet-pdf/pdf/22640/STMICROELECTRONICS/L78L24C.html

[10] "MC7805CT", Micro Commercial Components, n.d., [Online]. Available at: http://www.mccsemi.com/up_pdf/MC7805CT(TO-220).pdf

[11] “MCP1700”, Microchip, n.d., [Online]. Available at: http://ww1.microchip.com/downloads/en/DeviceDoc/20001826D.pdf

[12] TB6600 Stepper Motor Driver User Guide, datasheet, DFRobot, 2017. Available at: https://forum.arduino.cc/index.php?action=dlattach;topic=531835.0;attach=247209

Appendix A Requirement and Verification Table

Module	Requirements	Verification	Verification status (Y or N)
Power Distribution Unit	1. For the kitchen unit, the 5V voltage regulator must provide 5V(±5%) given a 9V(±10%) input. The 3.3V voltage regulator has to provide 3.3V(±5%) from an input voltage of 9V(±5%).	1a. For the kitchen unit, connect the output of the 5V voltage regulator to a multimeter. 1b. Connect an input voltage of 9V to the 5V voltage regulator. 1c. Measure the voltage. 1d. Repeat for the 3.3V voltage regulator. It should provide stable voltage of 5V (±5%) for and 3.3V (±5%) respectively.	Y
	2. For the robot unit, the 24V voltage regulator must provide 24V (±5%) given an input voltage of 36V (±5%). The 5V voltage regulator has to provide 5V (±5%) given an input voltage of 36V (±5%). The 3.3V voltage regulator has to provide 3.3V (±5%) from an input voltage of 36V (±5%).	2a. For the robot unit, connect the output of the 24V voltage regulator to a multimeter. 2b. Connect an input voltage of 36V to the 24V voltage regulator. 2c. Measure the voltage. 2d. Repeat the same for 5V and 3.3V voltage regulator. It should provide stable voltage of 24V(±5%), 5V (±5%) and 3.3V (±5%) respectively.	Y
	3. The 24V voltage regulator in the robot must provide stable current of 1.7A(±5%).	3a. Connect the regulator with its battery power source. 3b. Use a multimeter to measure the output current. The current must be remain 1.7A.	Y
	4. The voltage regulators must maintain temperature of chip below 150°C.	4a. While doing the measurement of current, use temperature gun sensor to measure the IC temperature. Ensure that it will not be higher than 150°C.	Y

Control Unit (Kitchen Unit)	5. Delay of data transmission between bluetooth and microcontroller should be less than 2s.	5a. Connect both a bluetooth receiver and transmitter to two different arduinos. 5b. Measure the timestamp as the transmitter sends the signal. 5c. Measure the timestamp as the receiver receives the signal. 5d. Subtract the two values to get the time taken to transmit the message (data).	Y
Control Unit (Robot)	6. Delay of data transmission between bluetooth and microcontroller should be less than 2s.	6a. Connect both a bluetooth receiver and transmitter to two different arduinos. 6b. Measure the timestamp as the transmitter sends the signal. 6c. Measure the timestamp as the receiver receives the signal. 6d. Subtract the two values to get the time taken to transmit the message (data)	Y
	7. Error in displacement in x-axis should be less than 5 cm for every 1 meter moved in y-axis. x-axis is the axis perpendicular to the starting position of the robot. y-axis is the axis parallel with the starting position of the robot. The y-axis and x-axis 90° apart.	7a. Align the robot along the edge of the tiles in the discussion room (beside the senior design lab). 7b. Using arduino, get the robot to move a certain distance. 7c. Once the robot has reached his destination, measure the perpendicular distance from the side of the tile to the robot. This is the error in displacement.	Y
	8. Error in movement in y-axis should be less than ±0.8cm for every 1 meter moved x-axis is the axis perpendicular to the starting position of the robot. y-axis is the axis parallel with the	8a. Align the robot along the edge of the tiles in the discussion room (beside the senior design lab). 8b. Mark the starting position using chalk.	Y

	starting position of the robot. The y-axis and x-axis 90° apart.	8c. Using arduino, we get the robot to move a certain distance. 8d. Once the robot has stopped moving, we measure the distance travelled from the starting position. This is parallel with the side of the tiles.	
	9. Error in angle when turning should be within ±5°	9a. Before moving, mark a straight line on the robot, cutting across the midpoint and perpendicular with the front & back. 9b. Using chalk, mark points on the floor exactly below the end of the line on the robot. 9c. Using arduino, make the robot turn 90°. 9d. Using chalk, remeasure the points as in step 2. 9e. Remove the robot from the area with the chalk markings. 9f. Draw straight lines across between each pair of points. The acute angle between the line is the angle of the turn	Y
Calculation Unit (Robot)	10. Time to calculate path is smaller than 5s.	10a. Run the path-finding software on the raspberry pi and time it electronically. The time taken should be lesser than 5s 10b. Repeat this with multiple table set-ups to ensure that the path-finding algorithm does not just pass the test because of an easy setup	Y
Communication Unit	11. The bluetooth transmitter and receiver should be able to transmit data for a distance of 0.05m to 8m without the presence of physical barriers between them.	11a. Configure the Bluetooth module on a breadboard, connected to an Arduino. 11b. Connect the Bluetooth module with phone application. 11c. Place the bluetooth	Y

		transmitter 0.05m away from receiver. 11d. Send a flag from the phone application to toggle the LED on an arduino. Repeat for a distance of 0.08m.	
Mechanical Unit	12. The motors should provide enough power to carry about 2kg weight and move with a speed of 10 (±20%) RPM (equivalent to about 10 meter per minute).	12a. Add 2kg load onto the robot and get it to move 10m. 12b. Measure the time taken by a stopwatch. Ensure the speed is 2.5m (±20%) per minute.	Y
	13. The encoder can count the rotations of wheel accurately. The acceptable error rate of result is within ±5%.	13a. Get the robot to move a distance of 2-3m. Measure this distance accurately. 13b. While the robot is moving, data from the encoder connected to the MCU must be recorded to find the total rotations the wheels have done. 13c. Using the number of rotations, calculate the distance and compare it with the actual distance (measured earlier). The difference should be within 5%.	Y

Table 3. Requirement and Verifications

Appendix B Schematics

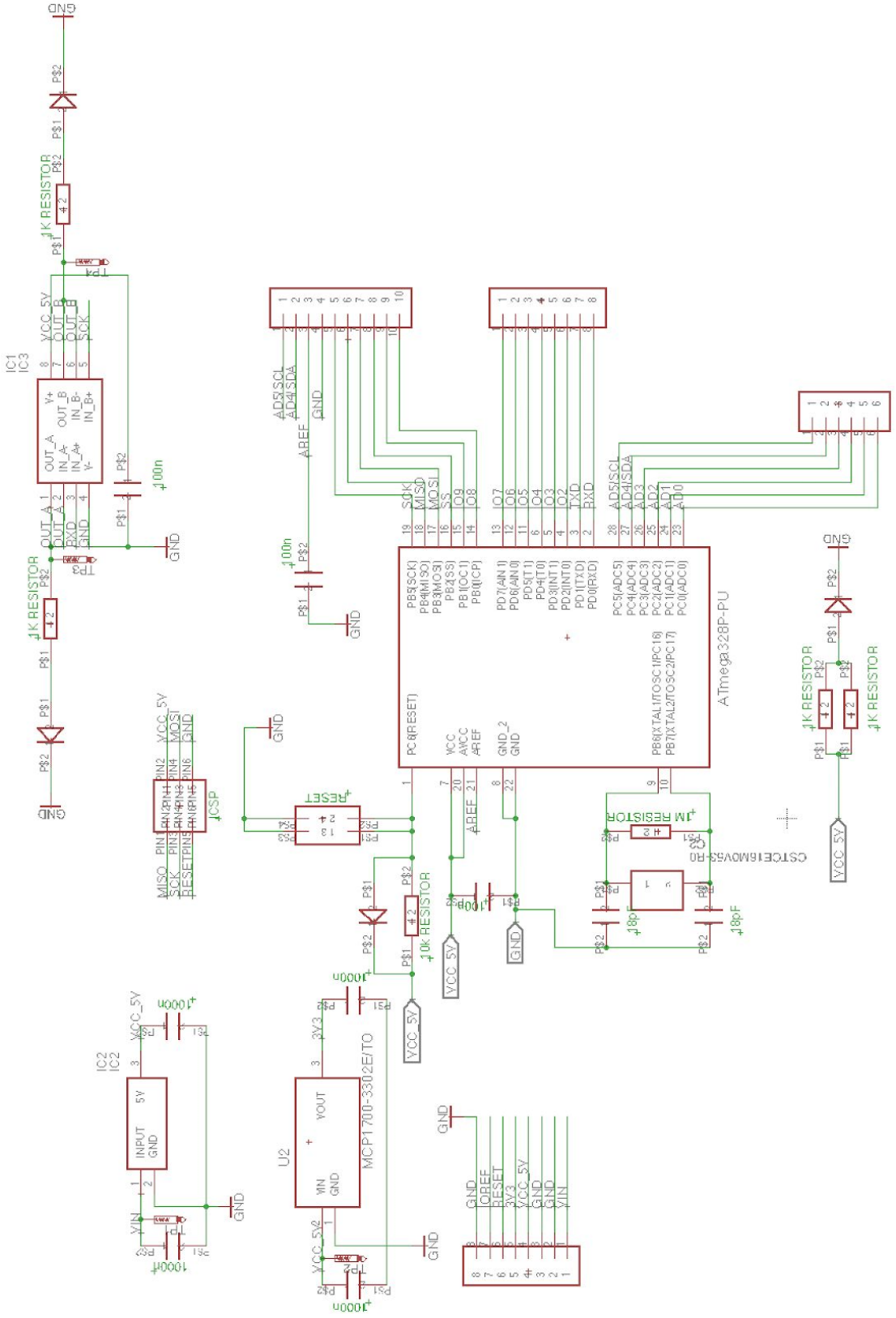

Figure 18. PCB Schematics of the Kitchen Unit

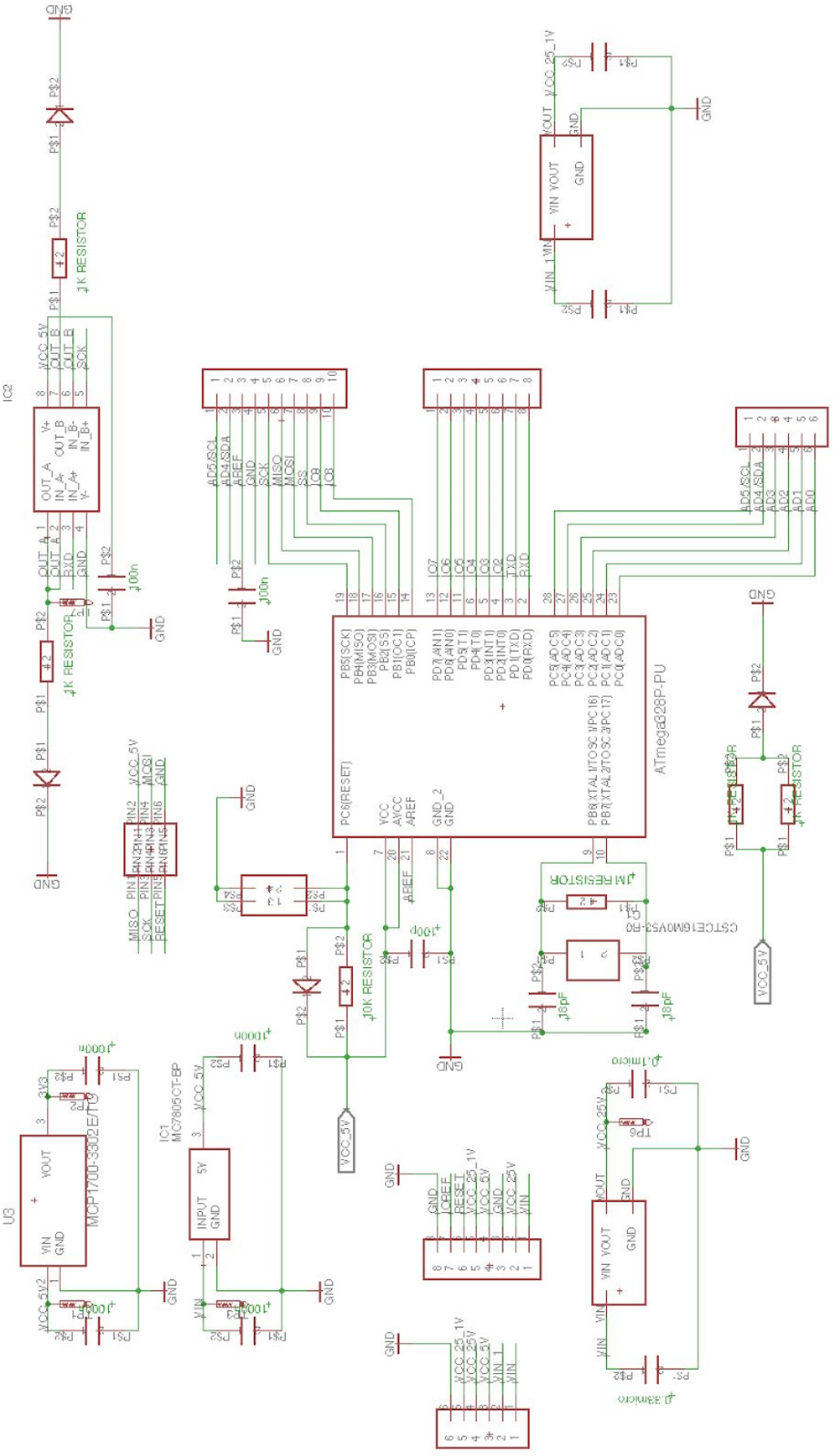

Figure 19. PCB Schematics of the Navigating Unit