BUTTER PASSING ROBOT

By

Yu Jie Hsiao

Yuchen He

Yuxiang Sun

Final Report for ECE 445, Senior Design, Spring 2018

TA: Xinrui Zhu

26 April 2018

Project No. 18

Abstract

This document demonstrates the design, development, test work we devoted into our project, the Butter Passing Robot. It also demonstrated the result of our project. The purpose of our project is to build a robot that can find and bring back butter by itself on a dining table. The project idea came from the sitcom Rick and Morty. With further work, our project can potentially be helpful to people with physical disability.

Contents

1. Introduction

1.1 Objective

For the senior design project, we are trying to build a small robot that can find butter on the table and bring it back. This idea was inspired by the famous sitcom “Rick and Morty”. In one of the episodes, Rick built a robot which could fetch the butter once it received a verbal command from its owner. [1] Such robot has the ability to move around on its own, detect target objects, recognize human speech and recognize faces. To make our project useful but also manageable, we will mainly try to implement a small autonomous vehicle with an object detection module. We assume that the butter is cube-shaped, yellow in color, with or without boxing and placed in a plate. After this project is done and works well, we can then make it stronger and apply it to detect some other objects, which can potentially be helpful for those with physical disabilities.

1.2 High-Level Requirement

- The vehicle can move by itself on a regular-sized (~ 2m*1m) table.
- The vehicle can detect the edge of the table and it will stop in order to prevent itself from falling
- The object detection program can distinguish yellow, cubed butter from other common breakfast objects (juice, bread… etc) and direct the vehicle toward butter. It should have an accuracy of at least 80%.

1.3 Block Diagram

Our project consists of three modules: the power supply, the control module and the robotic platform. Figure 1 contains a detailed diagram for different modules in the project. Compared to the original block diagram, we replaced the voltage regulator with the booster, replaced the current-limiting resistor with power switches and removed the temperature sensor. These decisions were made after we consulted our PCB design with our TA and other experienced people.

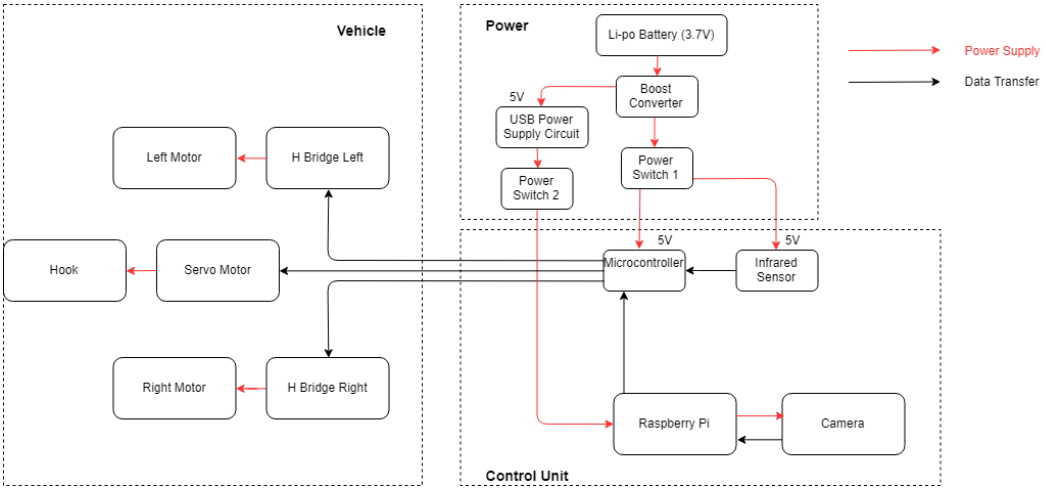

Figure 1 Block Diagram

2 Design

2.1 Power Module

2.1.1 Booster

To meet requirement of supplying robot at least 30 minutes running, we decided to use two Lithium batteries (3400mAh * 2) in parallel instead of in series as the power source for the whole system. The reason is that we need to check whether the batteries are in saturation when the batteries are connected in series. Since the microcontroller and other components require a stable 5V power, we added a boost converter TPS61032 in the circuit.

According to the data sheet from TI, the input range of our booster could be 1.8V to 5.5V, which cover both the Lithium battery and power bank output. The output current at 5V is up to 4000mA that also meet our design requirement. We also reserved the R3 and R4 for tuning 5V output accuracy and a jumper JP1 for isolation.

The current through the resistive divider should be about 100 times greater than the current into the LBI pin. The typical current into the LBI pin is 0.01uA, and the voltage across R2 is equal to the LBI voltage threshold that is generated on-chip, which has a value of 500 mV. The recommended value for R2 is therefore in the range of 500K ohms. From that the value of resistor R1, depending on the desired minimum battery voltage V_{bat}, can be calculated using the following equation

$$R1 = R2 \times \left(\frac{V_{BAT}}{V_{LBI-threshold}} - 1 \right) = 390\ k\Omega \times \left(\frac{V_{BAT}}{500\ mV} - 1 \right)$$

The output of the low battery supervisor is a simple open-drain output that goes active low if the dedicated battery voltage drops below the programmed threshold voltage on LBI. The output requires a pullup resistor with a recommended value of 1M ohms. The maximum voltage which is used to pull up the LBO outputs must not exceed the output voltage of the dc/dc converter. If not used. The LBO pin can be left floating or tied to GND.

Reference EVM:

TPS61032 LBI (R1, R2) Robot R1, R2 ==> EVM R1, R2

TPS61032 Robot --> L1, C1, C2, C3, C4 mapping to EVM ----> L1, C3, C4, C1, C2

Table X TPS6103x EVM Bill of Materials

Reference	Description	Manufacturer	Comments
C3, C4	100 µF 10 V, Low ESR tantalum size D	Vishay	594D–107X0016C2T or 594D–107X0010C2T
C1	10 µF X5R 6.3 V, capacitor SMD1206	TDK	C3216X5R0J106M
C2	2.2 µF X5R 10 V, capacitor SMD0805	TDK	C2012X5R1A225M
L1	6.8 µH, CDRH124-6R8	Sumida	SUMIDA CDRH104R–7R0, CDRH104R–100, or EPCOS B82464–G4682-M
R4	200 kΩ,1%, resistor SMD0805		TPS61030EVM, not used on the fixed output voltage versions
R3	1.8 MΩ,1%, resistor SMD0805		TPS61030EVM, not used on the fixed output voltage versions
R6	1 MΩ, 1%, resistor SMD0805		
R1	560 kΩ, 1%, resistor SMD0805		
R2	180 kΩ, 1%, resistor SMD0805		
J1, J2	Header 1×4, 0.1" pitch		
J6, J7	Header 1×2, 0.1" pitch		
J5	Header 1×3, 0.1" pitch		With jumper set to V_{BAT}
J4	Header 1×3, 0.1" pitch		With jumper set to GND
U1	TPS61030PW, TSSOP16 PowerPAD	TI	TPS61030EVM–208
	TPS61031PW, TSSOP16 PowerPAD		TPS61031EVM–208
	TPS61032PW, TSSOP16 PowerPAD		TPS61032EVM–208

The output voltage of the TPS61030 dc/dc converter section can be adjusted with an external resistor divider. The typical value of the voltage on the FB pin is 500 mV. The maximum allowed value for the output voltage is 5.5V. The typical current into the FB pin is 0.01uA, and the voltage across R6 is typically 500 mV. Recommended value for R4 from datasheet is 500K ohms, in order to set the divider current at 1uA or higher. From datasheet,

$$R3 = R4 \times \left(\frac{V_O}{V_{FB}} - 1\right) = 180\ k\Omega \times \left(\frac{V_O}{500\ mV} - 1\right)$$

Capacitors can be divided into two groups, input capacitors and output capacitors. From datasheet, at least a 10-uF input capacitor is recommended to improve transient behavior of the regulator and EMI behavior of the total power supply circuit. On the other hand, the major parameter necessary to define the output capacitor is the maximum allowed output voltage ripple of the converter. This ripple is determined by two parameters of the capacitor, the capacitance and the ESR. It is possible to calculate the minimum capacitance needed for the defined ripple using equation:

$$C_{min} = \frac{I_{OUT} \times \left(V_{OUT} - V_{BAT}\right)}{f \times \Delta V \times V_{OUT}}$$

Schematic R5 is pull-up resistor, usually use 100k ~ 1M OHM, it is for keeping a logic level.

The circuit for boost converter, I've followed the TI's TPS61032 evaluation modules (EVM) design and also bill of material that include schematic, PCB layout guide, component manufacture and selection. And power-distribution switch too. So that we can make sure there is lower risk in the power circuit design.

2.1.2 Power Switch

We used two power-distribution switches with overcurrent protection and under-voltage lockout. One is for the main system and one is for Raspberry subsystem (USB). The main controller can enable/disable

subsystem power and monitor subsystem overcurrent. A reset switch is to make warm reset if needed. The TPS2022 is at 1.5-A load, and the TPS2023 is at 2.2-A load.

According to the data sheet from TI, when the output load exceeds the current-limit threshold or a short is present, the TPS202x limits the output current to a safe level by switching into constant-current mode. Pulling the overcurrent (OC) logic output low to inform main controller. In our circuit, we connect the OC pin to the EN pin, so that when the current is beyond the current threshold and OC logic output goes low, the power switch will be cut off.

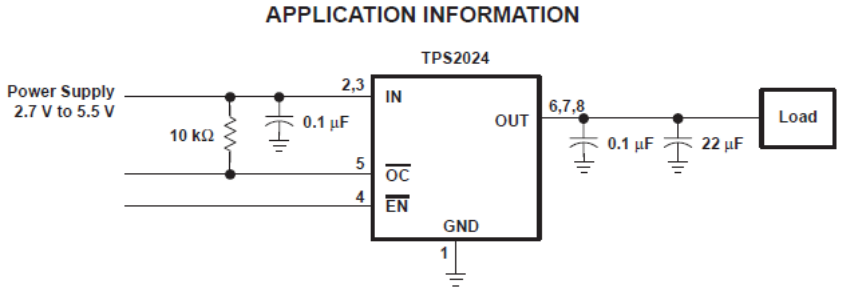

Figure X Power Switch Application

TPS2022/2023 --- power distribution switch

Robot schematic R6, R7, R8, R9 are pull-up resistor, 10k OHM

TPS2202D schematic C7, C9, C10, C6 follow EVN design and mapping to C1, C2, C3 and C4 (Table 4)

TPS2203D schematic C8, C12, C11, C5 follow EVN design and mapping to C1, C2, C3 and C4 (Table 4)

QTY												
-001	-002	-003	-004	-005	-006	-007	RefDes	Value	Description	Size	Part Number	MFR
1	1	1	1	1	1	1	C1	10uF	Capacitor, ceramic, 10-µF, X7R, 10V, 10%	1206	STD	STD
2	2	2	2	2	2	2	C2, C3	0.1uF	Capacitor, Ceramic, 16V, X7R, 10%	0805	STD	STD
1	1	1	1	1	1	1	C4	150uF	Capacitor, Tantalum, 150µF, 10V, 100mΩ, 10%	7343 (D)	B45197A2157K409	Kemet
1	0	0	0	0	0	0	U1	TPS2041BDGN	IC, Current-Limited Power Distribution Switch, 5.5V, 500mA	MSOP-8	TPS2041BDGN	TI
0	1	0	0	0	0	0	U1	TPS2051BDGN	IC, Current-Limited Power Distribution Switch, 5.5V, 500mA	MSOP-8	TPS2051BDGN	TI
0	0	1	0	0	0	0	U1	TPS2061DGN	IC, Current-Limited Power Distribution Switch, 5.5V, 1000mA	MSOP-8	TPS2061DGN	TI
0	0	0	1	0	0	0	U1	TPS2065DGN	IC, Current-Limited Power Distribution Switch, 5.5V, 1000mA	MSOP-8	TPS2065DGN	TI
0	0	0	0	1	0	0	U1	TPS2065DGN-1	IC, Current-Limited Power Distribution Switch, 5.5V, 1000mA	MSOP-8	TPS2065DGN-1	TI
0	0	0	0	0	1	0	U1	TPS2068DGN	IC, Current-Limited Power Distribution Switch, 5.5V, 1500mA	MSOP-8	TPS2068DGN	TI
0	0	0	0	0	0	1	U1	TPS2069DGN	IC, Current-Limited Power Distribution Switch, 5.5V, 1500mA	MSOP-8	TPS2069DGN	TI
1	1	1	1	1	1	1	–	HPA292	PCB, 3 in x 3 in x 0.062 in	2.25" x 2.225"	HPA292	Any
2	2	2	2	2	2	2	R1, R2	10.0K	Resistor, Chip, 1/10W, 1%	0805	CRCW0805-1002F	Vishay

Figure X TPS20xxEVM-292 Bill of Materials

The circuit for power switch, We've followed the evaluation modules (EVM) design and also bill of material that include schematic, PCB layout guide, component manufacture and selection.

2.2 Vehicle Module

2.2.1 H-bridge Circuit

We used H-Bridge circuits to drive our motors. The advantage of a H-Bridge circuit is that it can drive the motor in both directions. For example, we could let the motor move forward, backward, or stop by controlling the logic pin. This will give us the ability to rotate the vehicle about its center and to drive the vehicle backwards. Logic table is shown below.

Table X H-Bridge Control Table

Enable	Logic Pin 1	Logic Pin 2	Result
High	Low	High	Forward
High	High	Low	Reverse
High	Low	Low	Stop
High	High	High	Stop
Low	/	/	Off

2.2.2 Servo Motor

We decided to use a hook and a servo motor as the "mechanical arm" of our robot. The hook was 3D-printed so that it could match the dimension of the servo motor. Both the hook and the servo motor will be attached on the vehicle with self-adhesive rubber, which enables us to change the position of the mechanical arm easily.

2.2.3 Infrared Sensor

The infrared sensor serves two purposes in our project. It will tell the Microcontroller to stop the vehicle when approaching edge of the table. It will also tell the Microcontroller to stop the vehicle and drop the hook when approaching the plate. We gathered readings from the infrared sensor above different surfaces: white table surface, ground, and black plate edge. We then added two boundary values in the control code to differentiate between these three different types of scenarios.

2.2.4 Physical Design

The physical design of our project is based on the Sparkfun RedBot Basic Kit[c]. We added the servo motor and the Raspberry Pi camera in the front. We also placed the infrared sensors at the bottom of the vehicle. Figure 2 demonstrates the physical layout of our project.

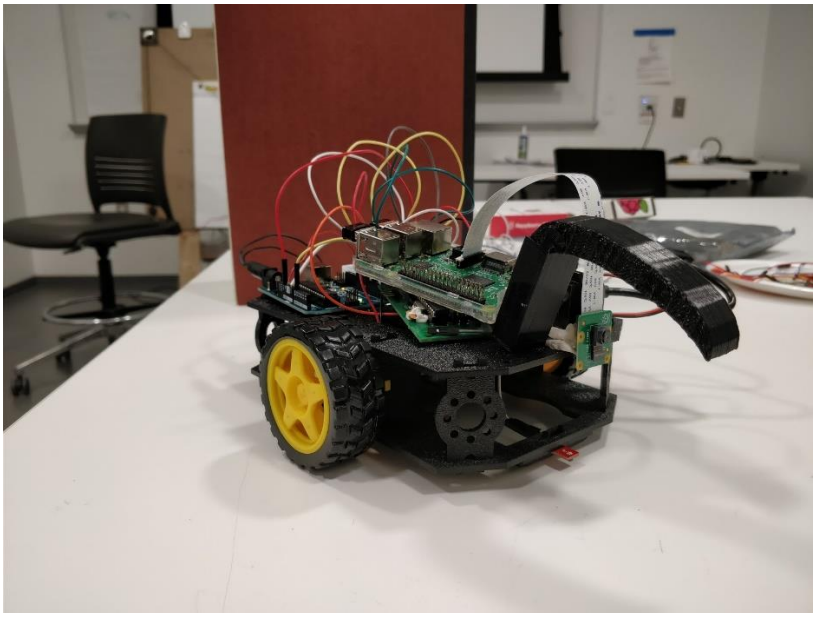

Figure 2 Physical Design

2.3 Control Module

2.3.1 Microcontroller (ATmega328p)

We used an ATmega328p microprocessor manufactured by Microchip Technology for this project. It will receive signals from Raspberry Pi and the infrared sensors. Analog pins are necessary to read accurate inputs from infrared sensor, while digital pins can suffice for the control of the motors. Initially we planned to use two digital pins as communication lines between the microcontroller and Raspberry Pi. However, as we moved forward with the project, we decided to implement i2c communication between Pi and microcontroller since it provided more convenient interface. Unfortunately, i2c communication required specific pins from both sides, which gave us some trouble assembling different hardware parts.

Table X Pin Layout for MCU[a]

Type	Pin Number	Pin Name	Description
Digital Output	2	PD0	connected to H-Bridge Pin 2 to control left motor
Digital Output	3	PD1	connected to H-Bridge Pin 7 to control left motor
Ground	8	GND	connected to ground
Analog Input	6	PD6	connected to Infrared sensor output
Analog Input	7	PD7	connected to Infrared sensor output
Digital Output	11	PB3	connected to H-Bridge Pin 10 to control right motor
Digital Output	12	PB4	connected to H-Bridge Pin 15 to control right motor
Power	20	AVCC	connected to voltage booster
Analog Input	18	A4	Connected to Raspberry Pi
Analog Input	19	A5	Connected to Raspberry Pi

2.3.2 Raspberry Pi and Camera

Raspberry Pi and its camera served as “eyes” to our robot. Initially we plan to write a shell script that ran on Raspberry Pi at start up to constantly taking photos. Then our object detection script will run on the photos taken and derive location information of the butter. However, later on we realized that it took around 5 seconds for Pi’s camera to take a photo and store it to disk. Such latency will greatly reduce the effectiveness of our object detection code. As a result, we modified the design to make the camera constantly taking videos of the environment. And the object detection program will run on each frame of the video stream. We used PiCamera library for the video stream implementation[b].

2.4 Object Detection Module

Our object detection module will detect the butter in an image (video frame) and output a bounding rectangle of the butter. For the detection methods we tried and used, please see section 3.3. According to the coordinates of the bounding rectangle, we could derive the location information of the butter using the equation described below.

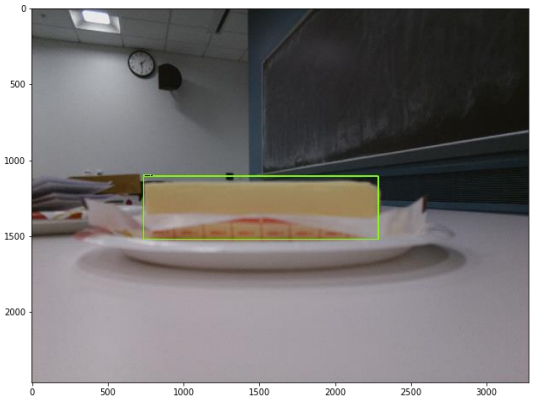

Figure X Example of Bounding Rectangle

(x, y, w, h) is the coordinates of the rectangle, where x, y denotes the top-left corner of the rectangle and w, h denotes the size of the rectangle.

(w_{image}, h_{image}) denotes the size of the image

$view$ denotes the field of view of our camera, which is 62°

The angle our vehicle should turn will be calculated as:

$$angle = \left(\frac{x + \frac{w}{2}}{w_{image}} - 0.5\right) \times view$$

3. Design Verification

3.1 Circuit Verification

3.1.1 Booster Verification

We did the bench test in order to check whether the power booster works. Since our battery supply is providing 3.7 V, and the input range of the booster we used is 1.8V to 5.5V, we apply the voltage from 1.8V to 5.8V to the booster and use multimeter to check whether the output pin gives stable 5V. And according to the data below, our booster satisfied the requirement.

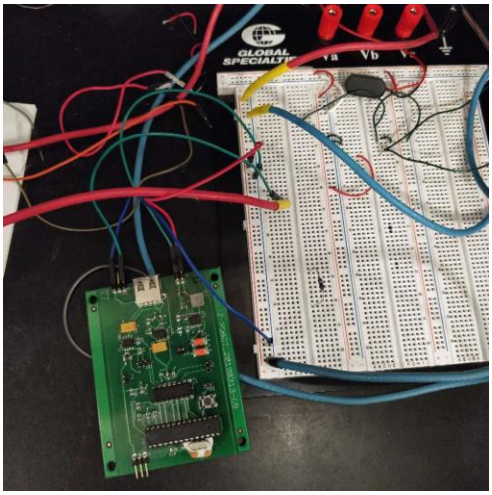

Figure X Test Setup for Booster

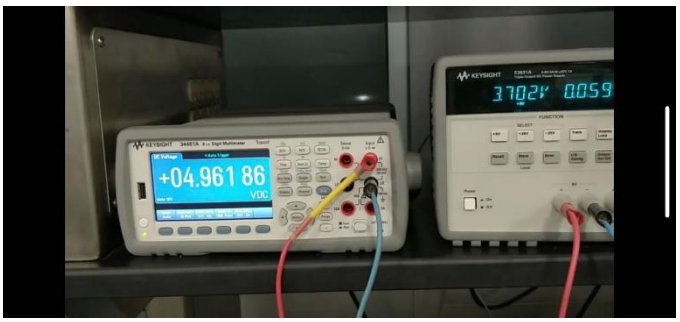

Figure X Test Results for Booster

Table X Test Results for the Booster

Input Voltage (V)	Output Voltage (V)
1.8	4.935
2.2	4.962
2.6	4.977
3.0	4.965
3.4	4.963
3.7	4.961
4.2	4.965
5.8	5.119

3.1.2 Power Switch Verification

Based on our design, we plan to use 2 batteries in parallel to provide 3.7V input. To test whether the power switches could prevent overcurrent, we used 4 batteries which will provide 6.1V instead. The consequence is that the power switches become hot and the USB port is no longer powering the Raspberry Pi. Also, the MCU chip is disabled, preventing the motor from running. After we disconnect the batteries for a while, we apply 2 batteries again which makes all components work normally. The test shows that the power switches work as we expected.

3.1.3 USB port verification

The USB port in our PCB is simply designed to power the Raspberry Pi. To test if we could successfully convert the batteries power to the USB port, we apply 3.7V as input and connect the Raspberry Pi through USB to check whether it has been activated. As the figures shown below, the Raspberry Pi has been activated.

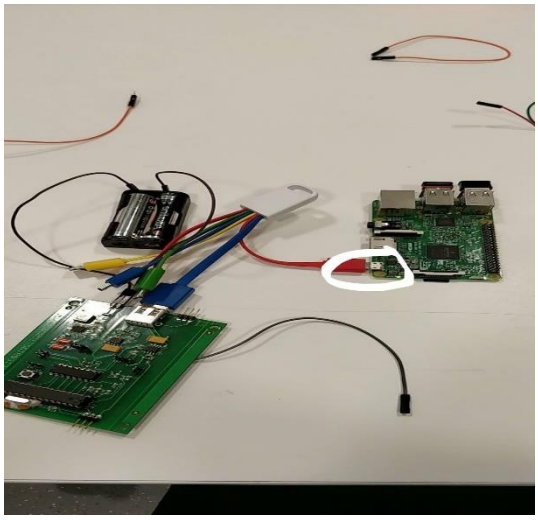

Figure X PCB Powering Raspberry Pi (1)

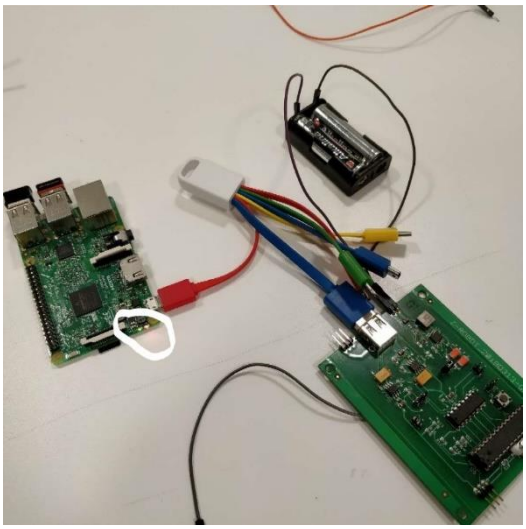

Figure X PCB Powering Raspberry Pi (2)

3.2 Vehicle Verification

3.2.1 H-Bridge Chip and Motor Verification

The ability for our vehicle to execute the command from Raspberry Pi accurately is critical to our project. We loaded different test programs onto an Arduino Board, telling the vehicle to turn different angles, and compared the result with the expected result. The figure below showed the car's position after we issued the command for it to first turn 60°, move forward and then turn 30°. The initial position of the vehicle was vertical to the edge of the table. As can be seen, the final position was almost parallel to the edge of the table, satisfying our requirement of an error smaller than 5°.

Figure X After Turning 90°

3.2.2 Infrared Sensor and Hook Verification

To test the command chain between our infrared sensor and our hook, we issued the command for the car to move toward a plate and observed the vehicle's behavior.

Figure 3 Approaching the Plate

Figure 4 Reaching the Plate and Dropping the Hook

3.3 Object Detection Verification

It is crucial for our object detection program to find butter as well as to have a low false positive detection rate. Thus, we must take both parameters into account when testing our object detection module. We have tried three object detection modules during the development of our project and we listed the verification for each of them below.

3.3.1 Verification of Haar Classifier model

We started with Haar Classifier using OpenCV for object detection. After training the Haar Classifier, we ran them against 40 test images we took manually and logged the results of each test image. Since we were not satisfied with the result of our first classifier, we increased sample size and trained a second

classifier. We did comparison between those two classifiers as well. The full results can be found in Appendix B.

In short, both classifiers hold the same accuracy of around 62%. The only progress is that the second model has a smaller number of false positive detections. The false positive detections decreased by about 20% from first round to second round. Below is the result we got from both classifiers on the same test image.

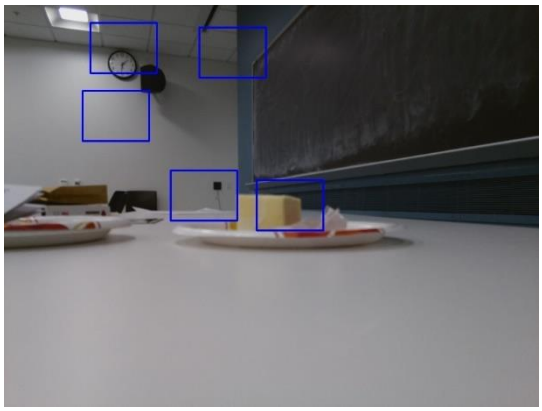

Figure X Result of 1st Classifier

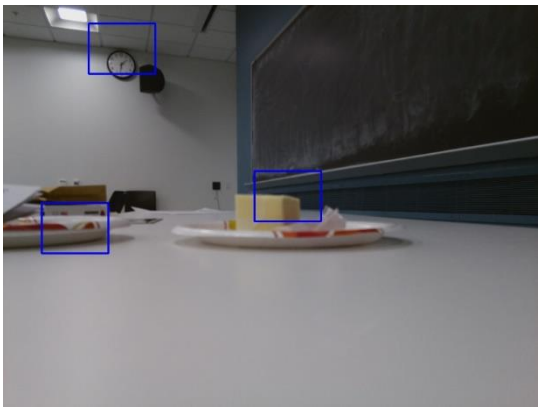

Figure X Result of 2nd Classifier

3.3.2 Verification of model trained with Tensorflow Object Detection API

For the Tensorflow object detection model, we also ran the training result against 40 test images we took. To get better results, we exported the trained graph after different number of steps to compare the results. Among the models we exported, the one after 17349 steps gives the best results. It has a detection rate of around 68% and it has very few false positive detections. The model after 59640 steps, surprisingly, has only a detection rate of 35.29%. We also exported two other models after that, and they gave even worse results. Appendix C contains a complete comparison between the model after 17349 steps and the model after 59640 results.

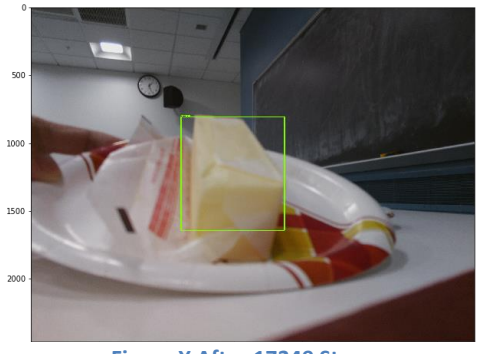

Figure X After 17349 Steps

Figure X After 59640 Steps

After testing with images, we also incorporated the model with video stream. However, the model performed terribly with video stream. It could only detect butter that is very close to the camera. To get

better results, we added more photos of butter being far away from the camera into the training process. Unfortunately, the exported model performed even more poorly with video stream.

3.3.3 Verification of HSV Color Detection Model

After the failure of our first two methods, we turned to HSV Color Detection as a last resort. And learning from the experience with Tensorflow models, we decided to test the new detector directly with video stream. We took a video of about 90 seconds on the Raspberry Pi with color detection running and wrote a python script to break the video into frames. While the video is rolling, we were manually moving the butter and similar-color objects (in this case, a yellow box) around the camera to simulate different scenarios. Then we selected around 600 frames randomly from all the frames and counted the number of correct detections. Below is the result we got:

Table X Color Detection Tests

	Detected	Not Detected
With Similar Color Object	141	146
Without Similar Color Object	324	1
Overall	465	147

In the case with similar color object, our detection program gave a detection rate of 49.13%. In the case without similar color objects, our detection program gave a detection rate of 99.70%. The overall accuracy is 75.98%.

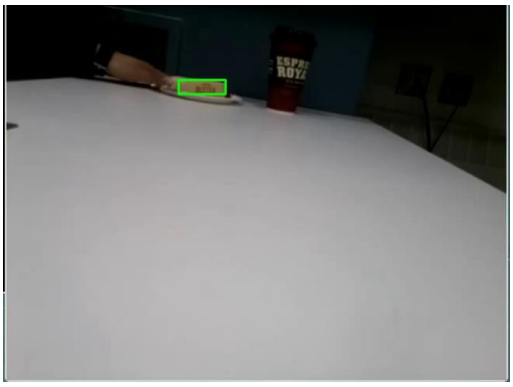

Figure X Detecting Butter

Figure X Detecting Box

4. Costs

4.1 Parts

For the parts used in this project, we mainly purchased them from Digi-Key and Amazon. Below is the table of individual part cost. Notice that not all the parts were used in the final project. Some of the parts (e.g. the Arduino board) were mainly used for the convenience of development and testing.

Table X Parts Cost

Part	Manufacturer	Retail Cost ($)	Count	Total Cost ($)
Raspberry Pi 3B Motherboard	Raspberry Pi Foundation	35.91	1	35.91
Raspberry Pi Camera Module V2	Raspberry Pi Foundation	26.45	1	26.45
Rechargeable Li Battery	Amazon Basics	13.88 /8-pack	1	13.88
Li Battery Charger	Amazon Basics	14.99	1	14.99
Hobby Gear Motor	Sparkfun	3.95	1	3.95
Servo Motor	Sparkfun	8.95	1	8.95
Sparkfun QRE1113 Infrared Sensor	Sparkfun	2.95	2	5.90
SN754410 H-Bridge Chip	Texas Instruments	2.35	2	4.7
ATMEGA328P	Microchip Technology	2.20	2	4.40
Arduino Uno	Arduino	22.00	2	44.00
Total				**163.13**

4.2 Labor

We will use the following formula to calculate the labor cost for each team member:

$$labor\ cost = ideal\ rate \times actual\ hours\ spent \times 2.5$$

Table X Labor Cost

Team Member	Hourly Rate ($/Hour)	Total Hours (hour)	Total Cost ($)
Yu Jie Hsiao	30	60	4500
Yuxiang Sun	30	60	4500
Yuchen He	30	60	4500
Total			13500

The total cost of our project is $\$163.13 + \$13500 = \$13663.13$

5. Conclusion

5.1 Accomplishments

In the end, we were able to get all the individual modules of our project working. Our vehicle was able to move by itself on the table, without falling off. Our vehicle was also able to direct itself toward the butter and drop the hook after detecting the plate.

5.2 Uncertainties

The major problem with our project is that we had difficulty integrating all different parts together. We believed the main reason was the inconsistency between software design and hardware design. Our design change in the software module were not fully demonstrated in our hardware design. For the communication between our Raspberry Pi and microcontroller, we initially planned to use two GPIO pins from Raspberry Pi and two digital pins from the microcontroller. Later on in the project, we realized such communication protocol is naïve and did not provide real-time communication. So we did some research and decided to implement i2c communication instead. However, i2c communication required specific pins from the microcontroller, which we already connected with other parts on the PCB.

5.3 Ethical considerations

The main ethical consideration of our project is the use of open-source framework. Following IEEE code of ethics, we made sure that we cited properly in the comment section of our code. We also made sure we included the license paragraph, if any, in our code.

5.4 Future work

For future work, we believed there were two major aspects we could work on to improve our project.

The first aspect would be the optimization of our object detection program. Currently our python script is only reporting the largest contour it finds in the video frame as “butter”. However, due to change in environment lighting, it’s very likely that there will be discrete portions on the butter that did not satisfy the color range. If we could implement some program to merge contours that are close to each other using concepts like Manhattan distance, we could probably getting more accurate location information about the butter.

The second aspect we could improve is the integration between hardware parts and software parts. As mentioned in section 5.2, we should come up with a different microcontroller pin layout to satisfy the communication needs of all parts.

References

[a] *Arduino Pin Layout.* Available at: https://www.pinterest.com/pin/728457308447211461/

[b] *Picamera Documents.* Available at: https://picamera.readthedocs.io/en/release-1.13/

[c] *SparkFun RedBot Basic Kit.* Available at: https://www.sparkfun.com/products/retired/13166

Appendix A Requirement and Verification Table

Table X System Requirements and Verifications

Requirement	Verification	Verification status (Y or N)
Boost Converter		
Given input in the range of 2.5V to 3.7V, the boost converter must provide constant 5V output to satisfy the needs of all components.	We plan to apply different voltages (2.5V ~ 3.7V) to the boost converter using function generator in lab. And we will use the multimeter to measure the output voltage of the regulator, which should always be 5V.	Y
Power Switch		
The power switch is able to limit the output current even if the input current is exceeding the threshold.	We plan to connect the power switch with resistors on the breadboard. And we will use function generator to apply different voltages across the resistor. We will use a multimeter to measure the output current from the switch, which should be below the threshold.	Y
Infrared Sensor and Microcontroller		
The microcontroller should stop the vehicle when the infrared sensor's reading suggests that the vehicle is approaching the edge.	We loaded test programs that drive the vehicle toward the edge onto the vehicle and observed the vehicle's movement	Y
The microcontroller should stop the vehicle when the infrared sensor's reading suggests that the vehicle already reached the plate.	We loaded test programs that drive the vehicle toward the palte onto the vehicle and observed the vehicle's movement.	Y
The vehicle can move straight forward and backward. The angle offset should be within ±10°.	We plan to write a test program. it will apply a constant voltage V to both motors for 10 seconds. We will draw the path on the table and measure the angle offset using a protractor.	Y
The vehicle can rotate about its center. The center should only shift within 2 cm every time the vehicle rotates 60°.	We will write a test program and load that to the microcontroller. It will perform a rotation of the vehicle every 10 seconds. And we measure the center shifting using a ruler.	Y
Object Detection Module		
The program can detect yellow, cubed butter in the size smaller than 12cm * 3cm * 3cm[10].	We will write a test program asking the detector to run with at least 500 positive images of cubed butter (either	Y

The accuracy rate should be 80% at least.	automatically created by classifier or manually taken by us). We will re-train our detector until the 80% accuracy is met.	
The program can distinguish butter from other kitchen objects of similar color: orange juice, honey mustard. The accuracy rate should be 80% at least.	We will write a test program asking the detector to run with at least 500 negative images of orange juice or honey mustard. We will re-train our detector until the 80% accuracy is met.	N
The programs should output an angle between the vehicle and the target. And this angle should be within ±10° from the actual angle.	We will place the butter and Raspberry Pi camera both on the table. Then we will compare the actual angle measured by a protractor with the angle calculated by our detector.	Y

Appendix B Comparison between Haar Classifiers

Table X Comparison between Two Classifiers

Image Name	Previous Classifier (500 +1000)		New Classifier (3500 + 7000)	
	Find Butter?	Total # of detections	Find Butter?	Total # of detections
TestPos_0	Yes	5	Yes	3
TestPos_1	Yes	5	Yes	5
TestPos_2	No	4	Yes	3
TestPos_4	Yes	8	Yes	4
TestPos_5	Yes	4	Yes	2
TestPos_6	Yes	7	Yes	4
TestPos_7	Yes	6	Yes	2
TestPos_8	Yes	4	Yes	5
TestPos_9	Yes	4	Yes	2
TestPos_10	Yes	9	Yes	9
TestPos_11	Yes	2	No	4
TestPos_12	No	3	No	2
TestPos_13	No	3	No	2
TestPos_14	Yes	3	No	1
TestPos_15	Yes	4	Yes	3
TestPos_16	No	4	Yes	2
TestPos_17	No	4	No	4
TestPos_18	No	2	No	3
TestPos_19	Yes	5	Yes	3
TestPos_20	No	7	No	9
TestPos_21	Yes	8	Yes	3
TestPos_22	Yes	5	Yes	10
TestPos_23	Yes	3	No	1
TestPos_24	Yes	5	Yes	2
TestPos_25	Yes	6	Yes	8
TestPos_26	No	3	Yes	3
TestPos_27	No	4	No	1
TestPos_28	Yes	3	No	3
TestPos_29	Yes	6	Yes	2
TestPos_30	No	2	No	1
TestPos_31	N/A	4	N/A	3
TestPos_32	Yes	7	Yes	7
TestPos_33	No	7	Yes	8
TestPos_34	N/A	7	N/A	5
TestPos_35	N/A	7	N/A	5
TestPos_36	N/A	5	N/A	5
TestPos_37	N/A	5	N/A	5
TestPos_38	N/A	5	N/A	5
TestPos_39	N/A	6	N/A	5

Appendix C Test Results of Tensorflow Model

Table X Comparison of two Tensorflow Models

Image No.	17349_graph	59640_graph
1	yes	no
2	yes	yes
3	yes	yes
4	yes	no
5	yes	yes
6	yes	yes
7	no	no
8	yes	no
9	yes	no
10	yes	yes
11	no	no
12	no	no
13	no	no
14	yes	no
15	no	yes
16	no	yes
17	no	no
18	yes	yes
19	yes	no
20	yes	yes
21	yes	no
22	yes	yes
23	no	no
24	yes	no
25	yes	yes
26	yes	no
27	no	no
28	yes	no
29	yes	no
30	yes	no
31	no	no
32	yes	yes
33	yes	no
34	no	no
Accuracy	67.60%	35.29%