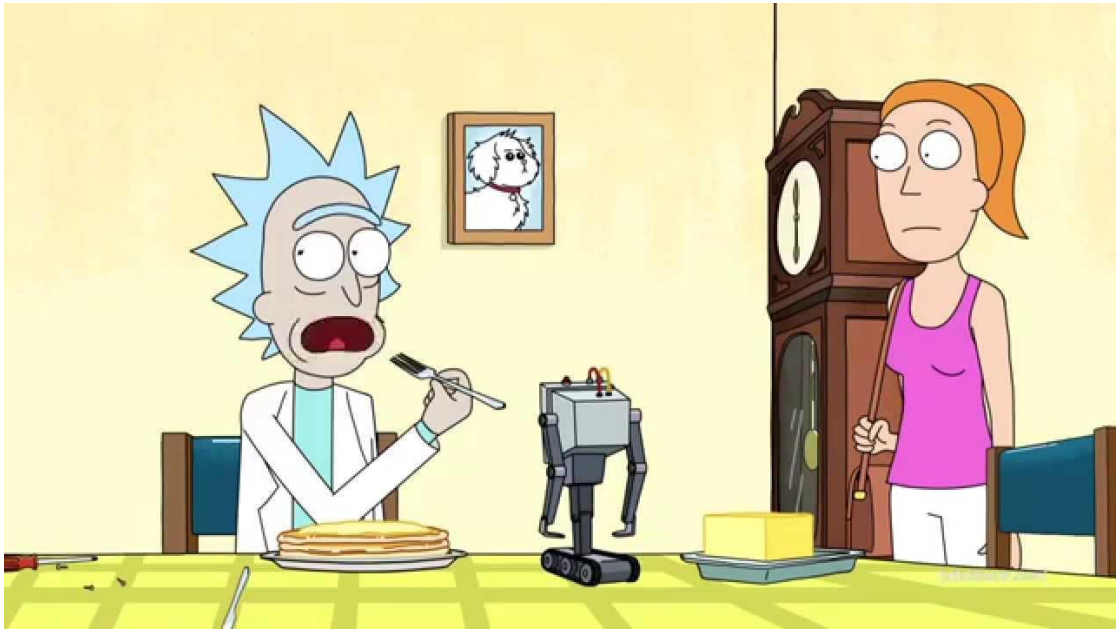
ECE445 Presentation

Team 18: Butter Passing Robot
Yujie Hsiao, Yuchen He and Yuxiang Sun

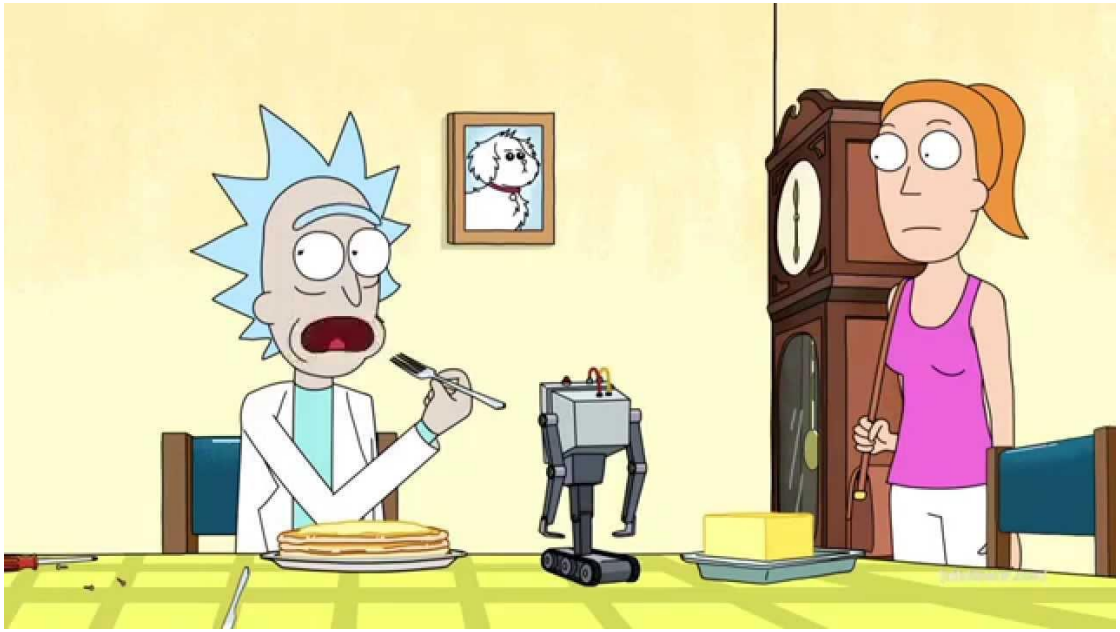
Introduction



- Idea from the cartoon Rick and Morty.

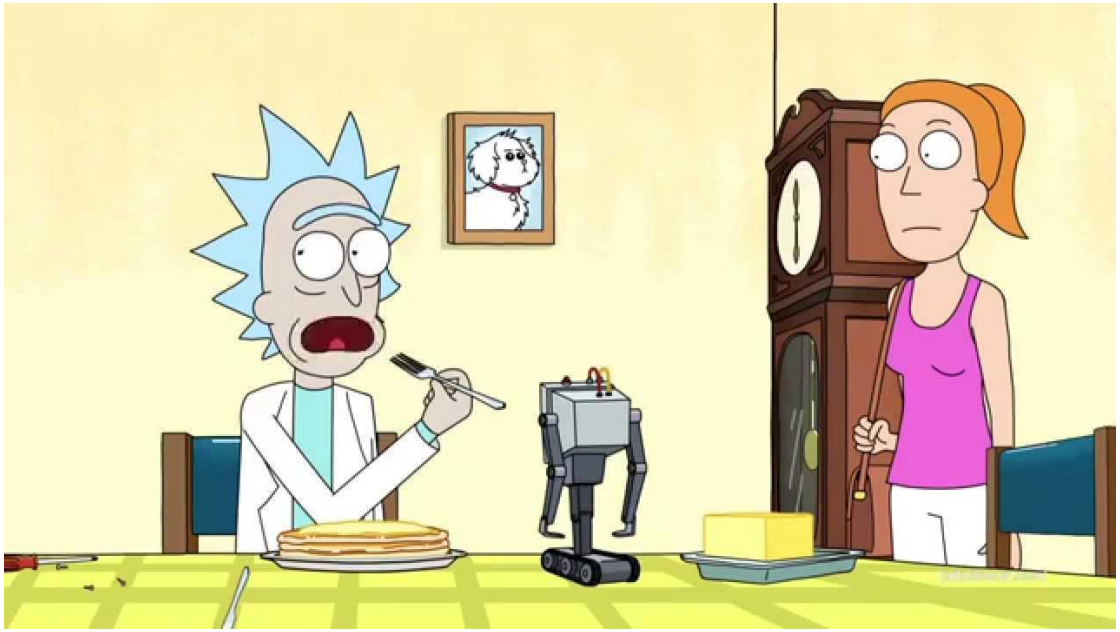
Pass the butter!!

Problem Statement & High Level Requirement



- Self moving on a regular-sized (~ 2m*1m) table.
- Edge detection.
- The object detection program can distinguish yellow, cubed butter from other common breakfast objects and direct the vehicle toward butter.

Objectives



- It’s fun to have a robot to work for you in daily breakfast time.

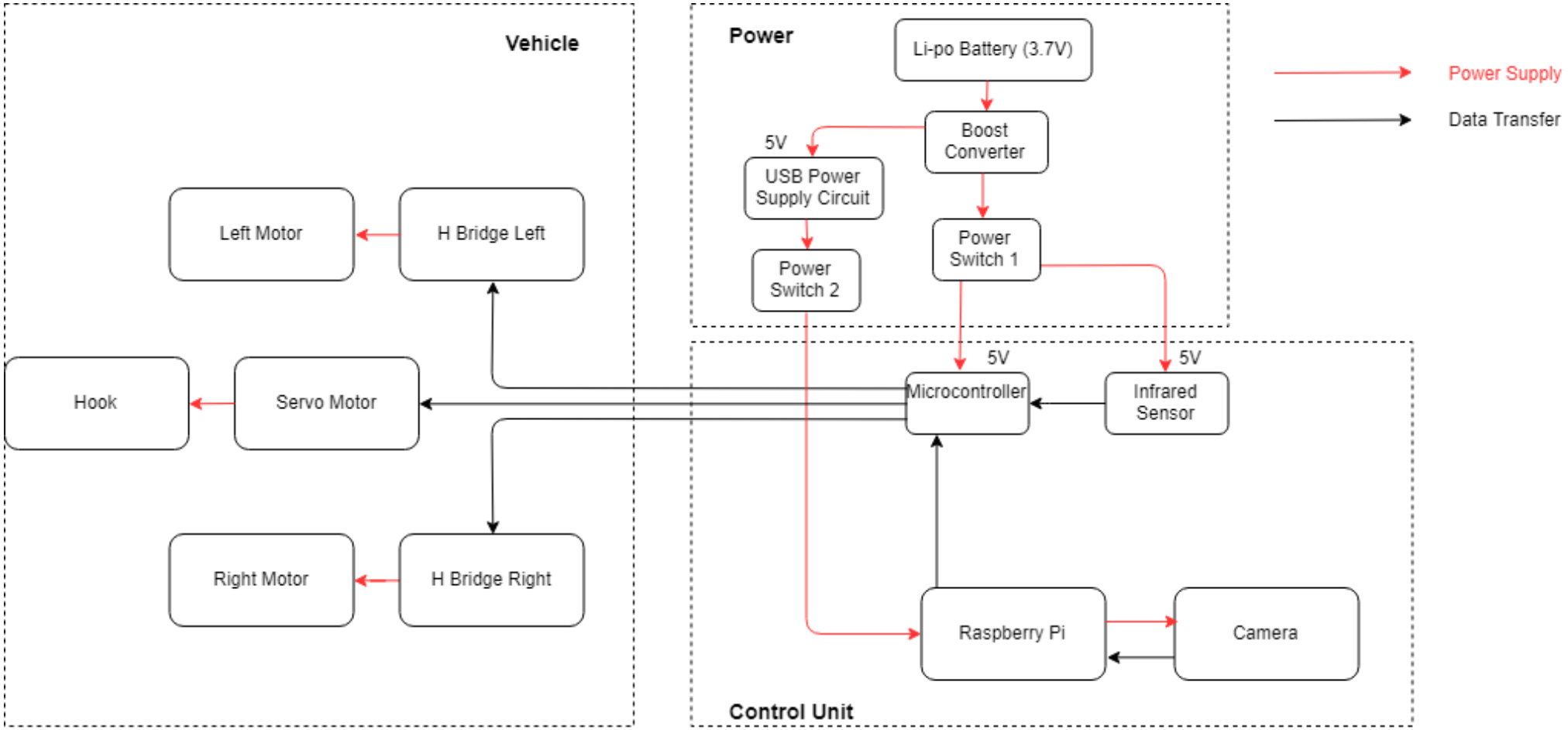
- As long as this works, we may apply it to get other stuffs which is beneficial to those who have physical disabilities.

Problem Description

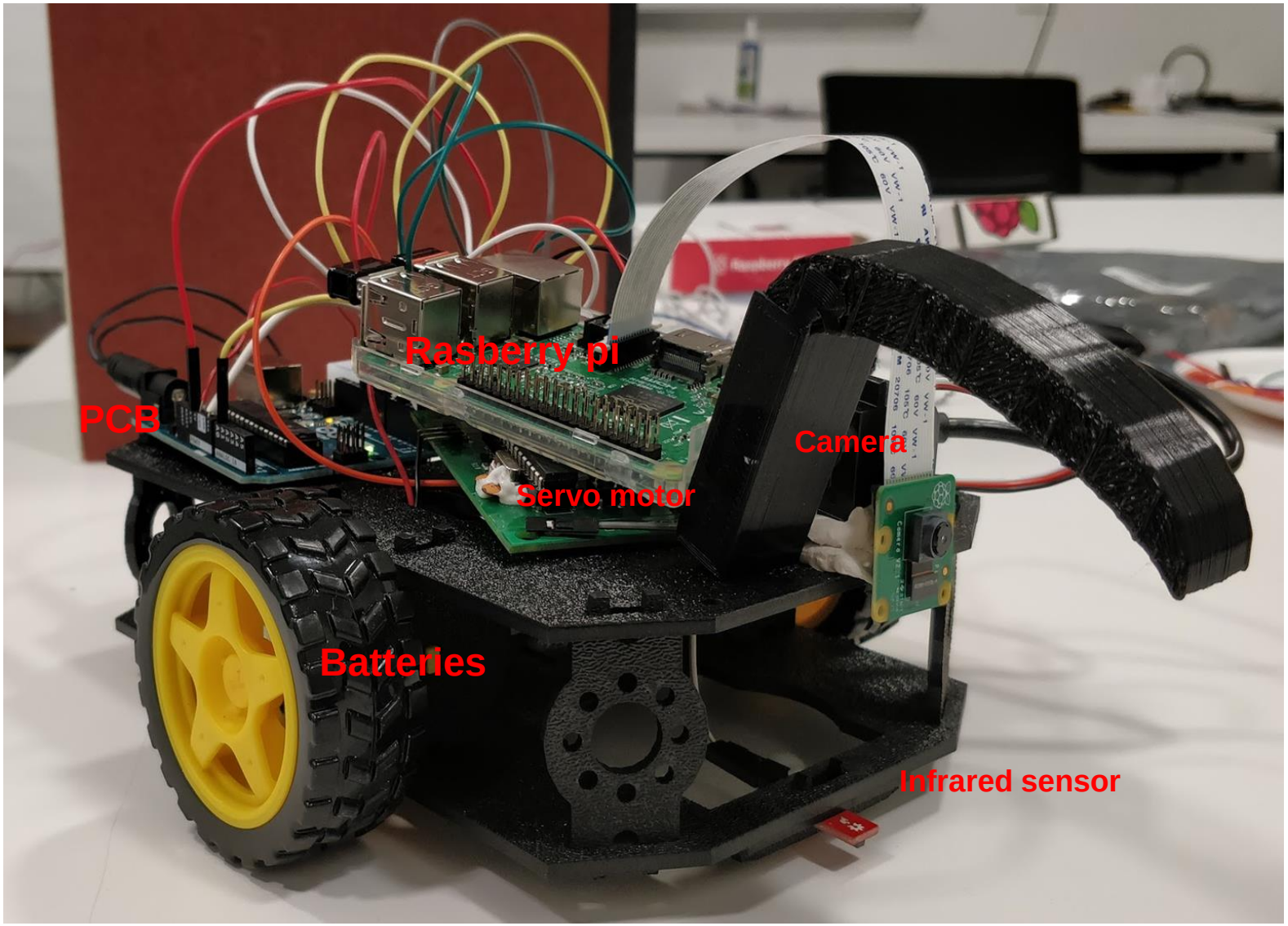
Define **butter**:
Butter with or without the thin package all count.

Plate:
Can be any plates but their edges should be black.

Block Diagram



Physical Design



Control

The microcontroller is mainly operating 2 parts of the motions, one for moving the car and one for controlling the hook.

1. **Moving the car**

   We use H-bridge to driver our motors and make the turns that we need to the butter and back to the user.

Control

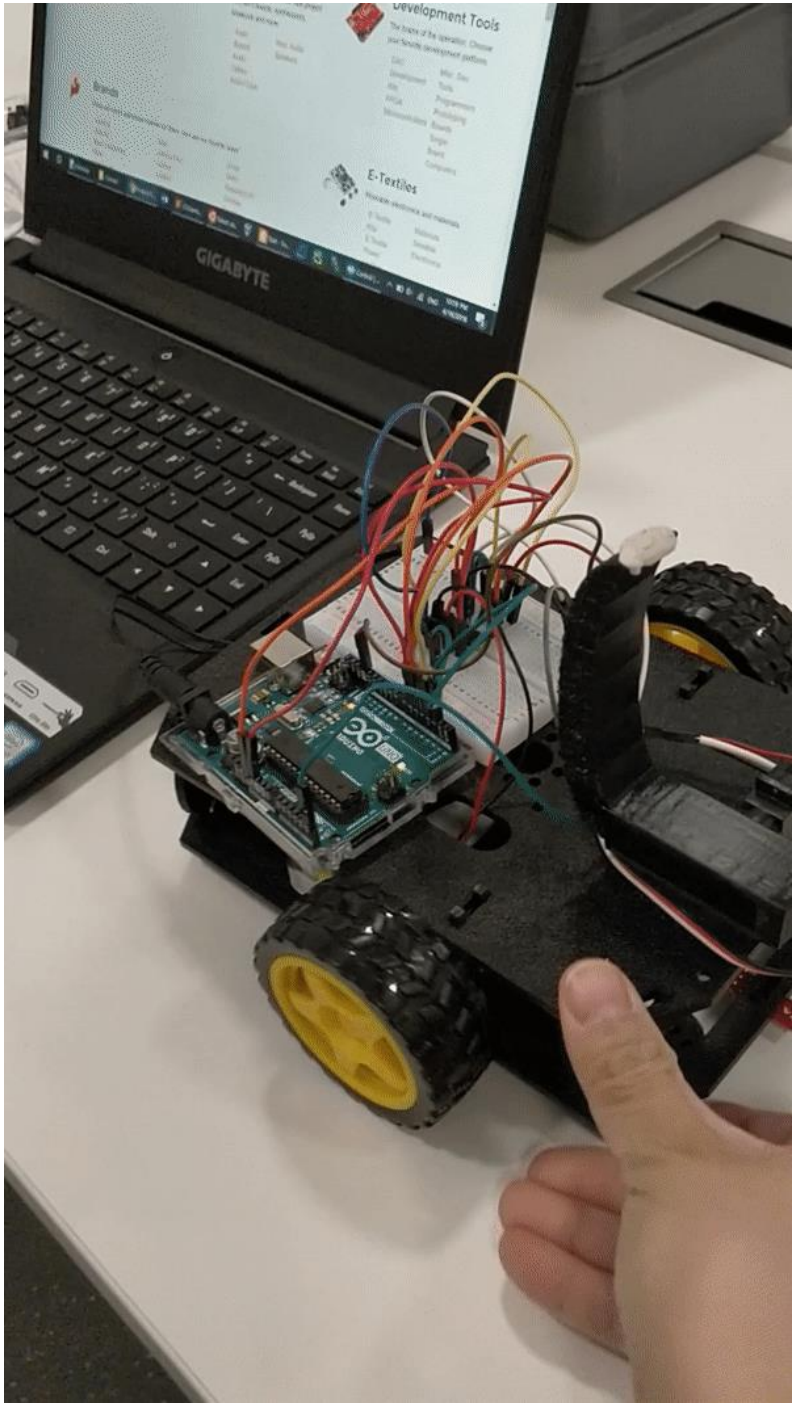
Going Forward and Making Turns

MCU code:

1. Turn 60°clockwise
2. Move Forward
3. Turn 30° clockwise
4. Move Forward

Control

Stop at Edge



Control

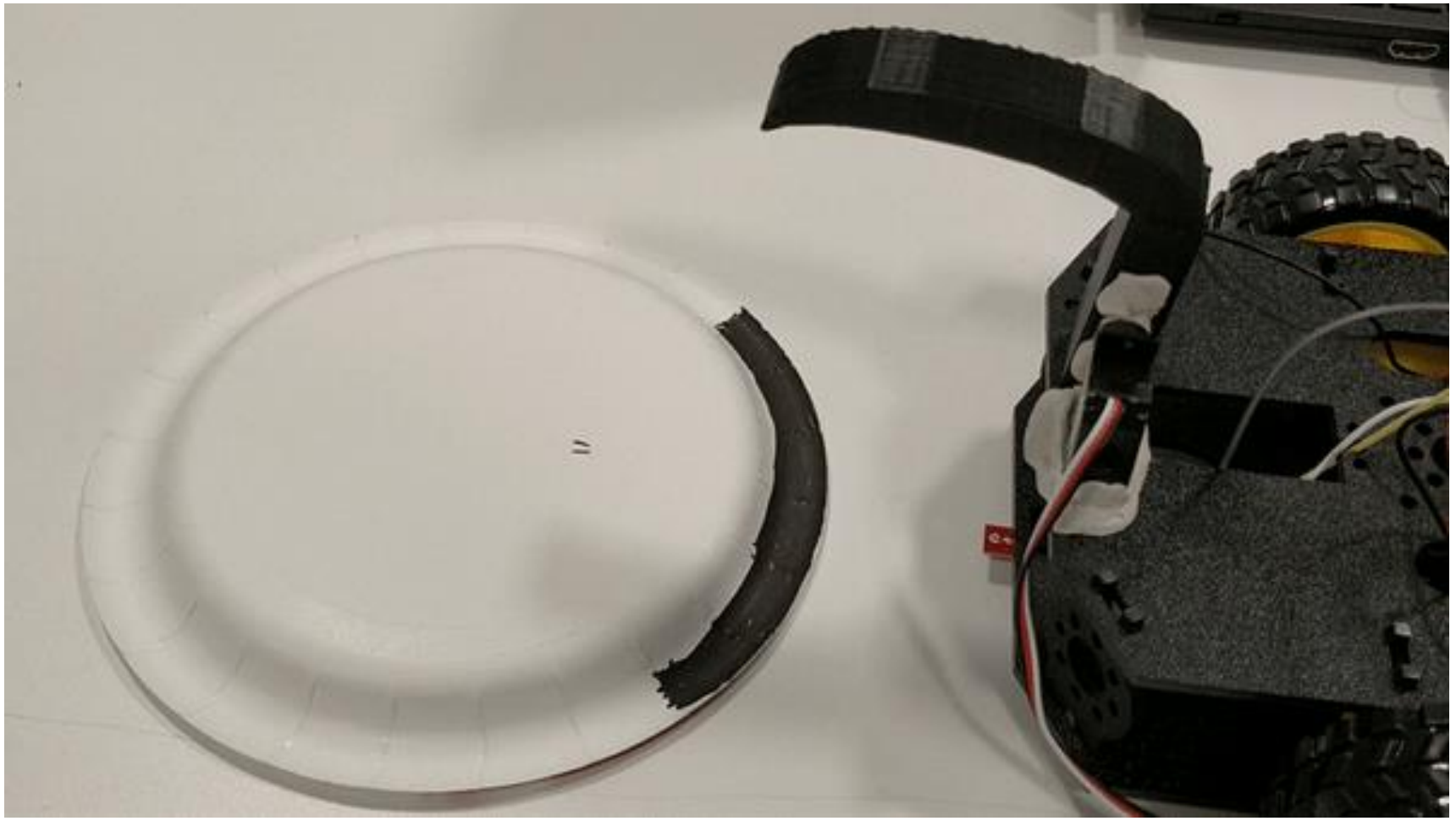
Control

The microcontroller is mainly operating 2 parts of the motions, one for moving the car and one for controlling the hook.

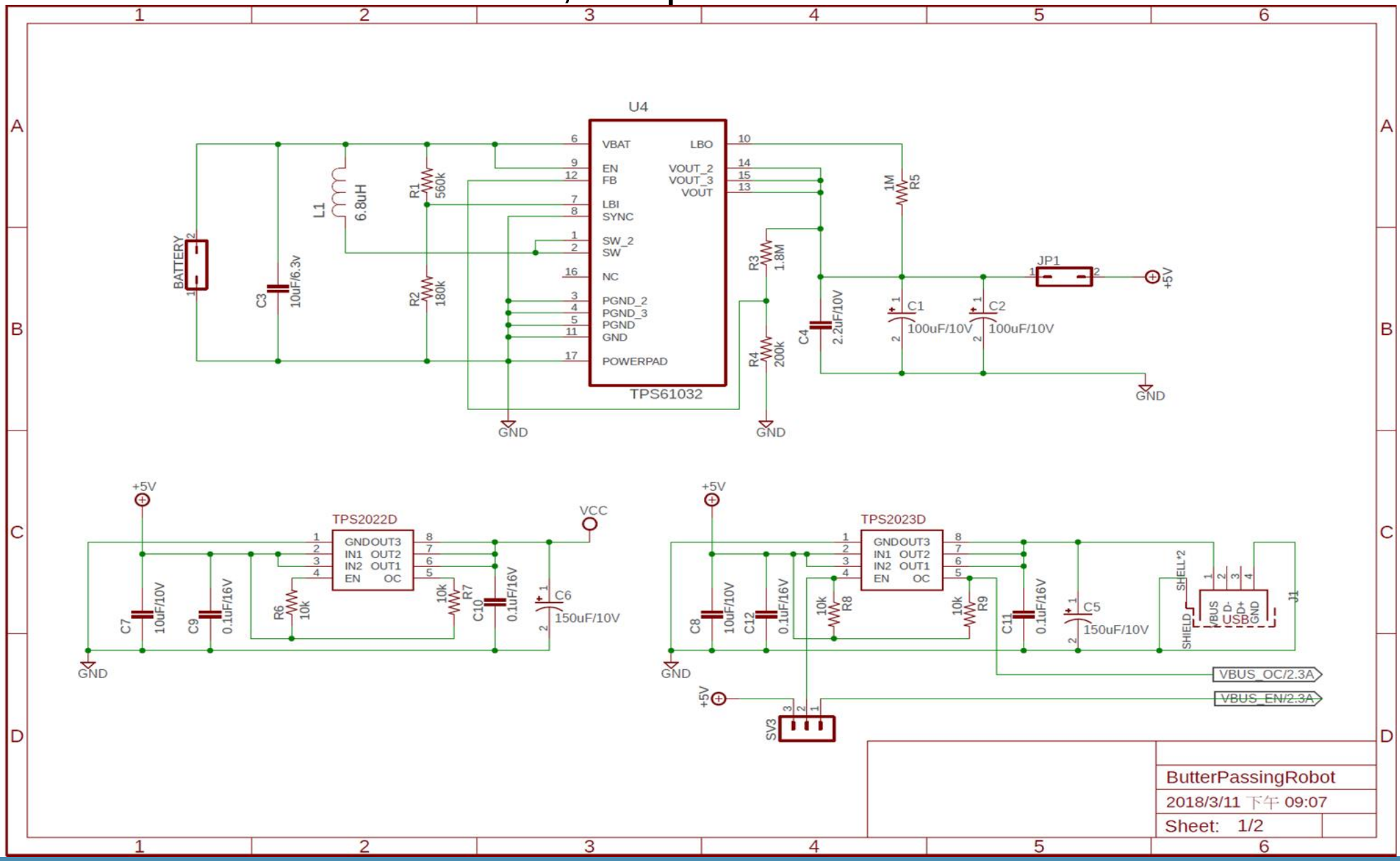
2. **Controlling the hook**

   After the infrared sensor senses the plate, the hook will be driven down by a servo motor to get the plate of the butter, and then raises up as the butter is back to the user.

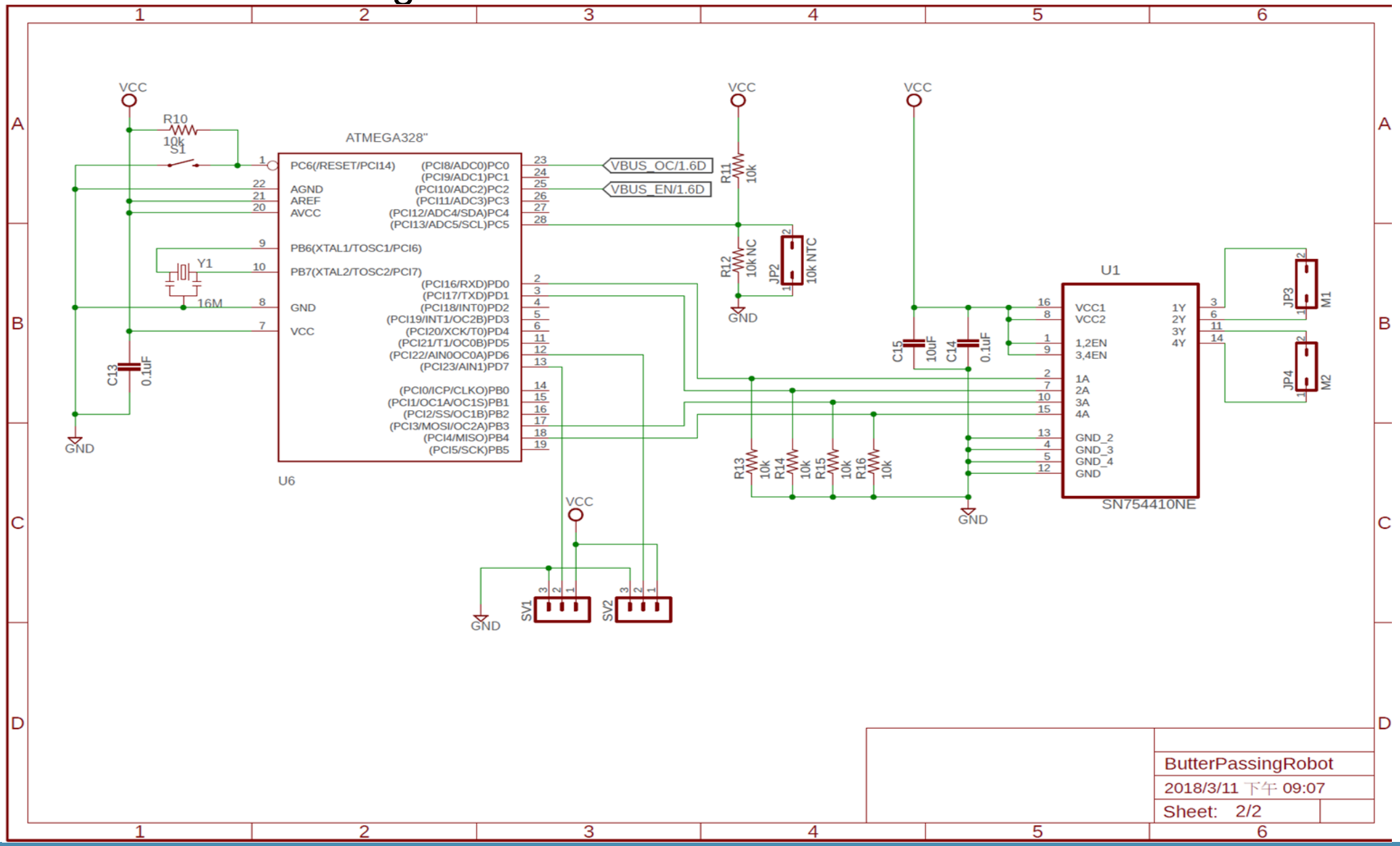
Control



Schematic: Power Booster, two power Switches



Schematic: H-bridge circuit and MCU

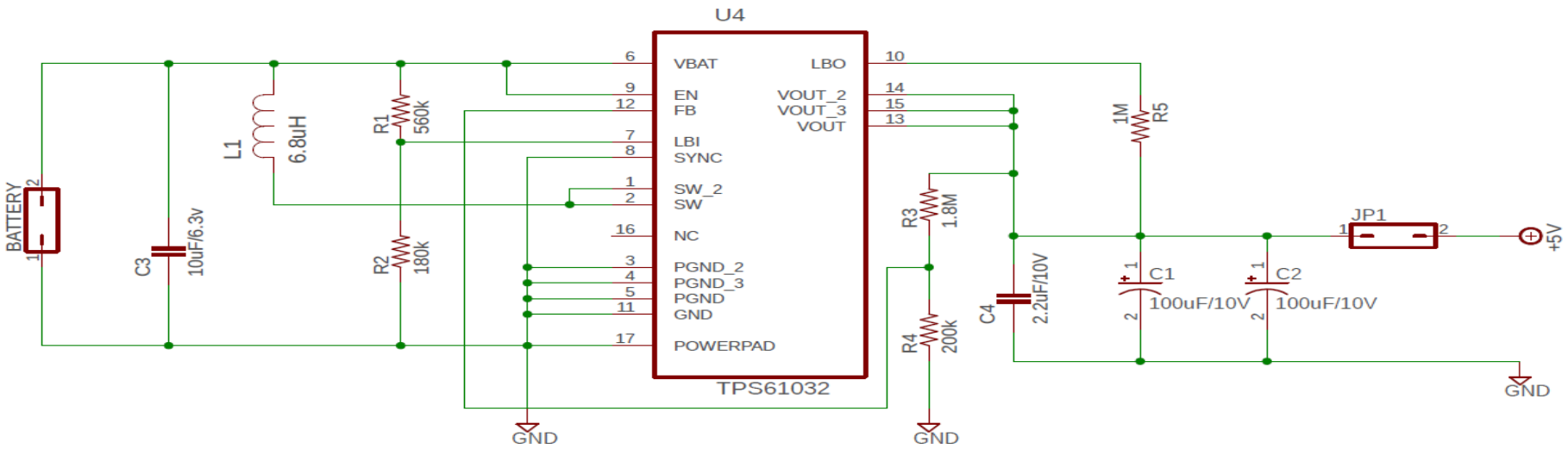


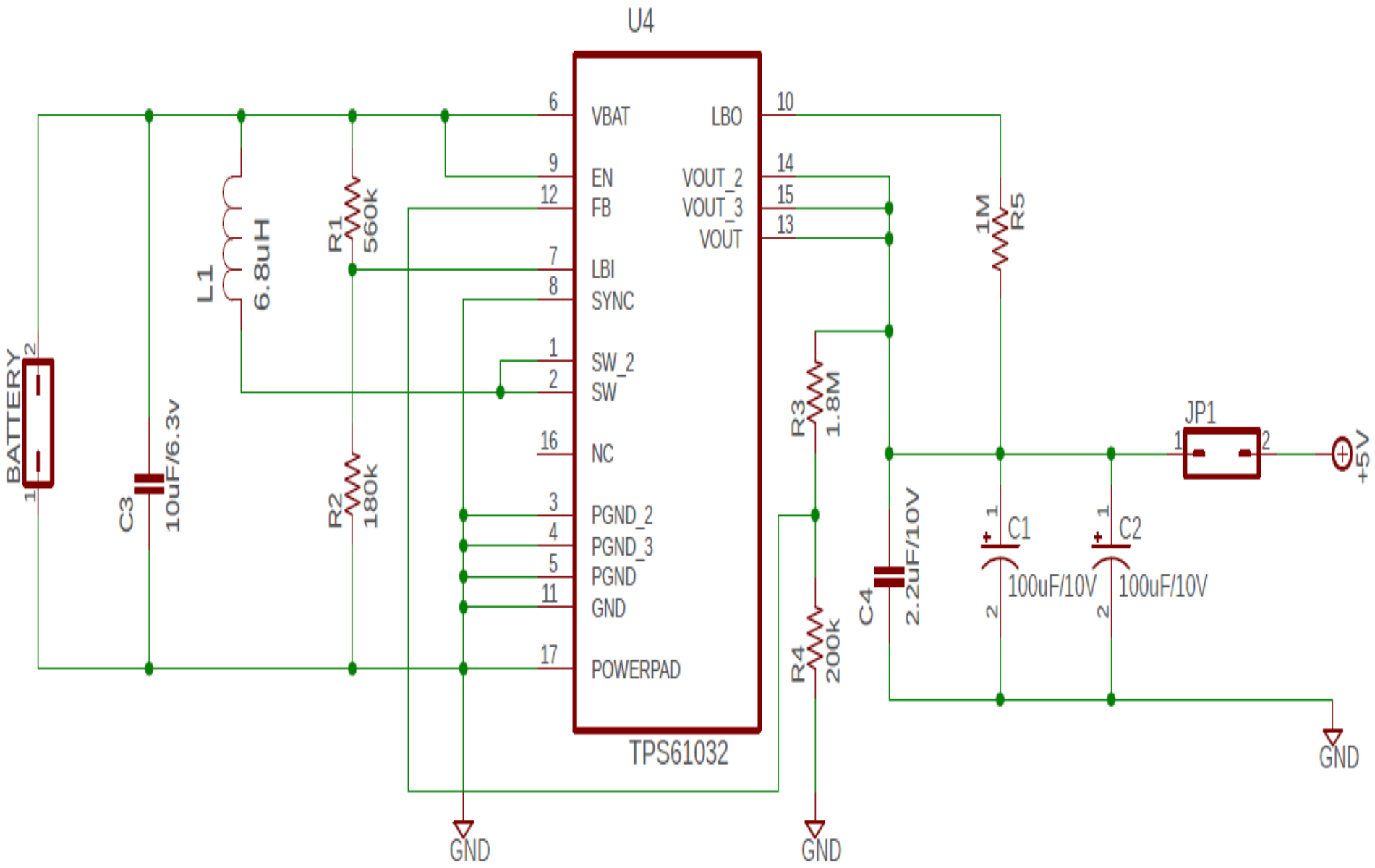
Booster (TPS61032)

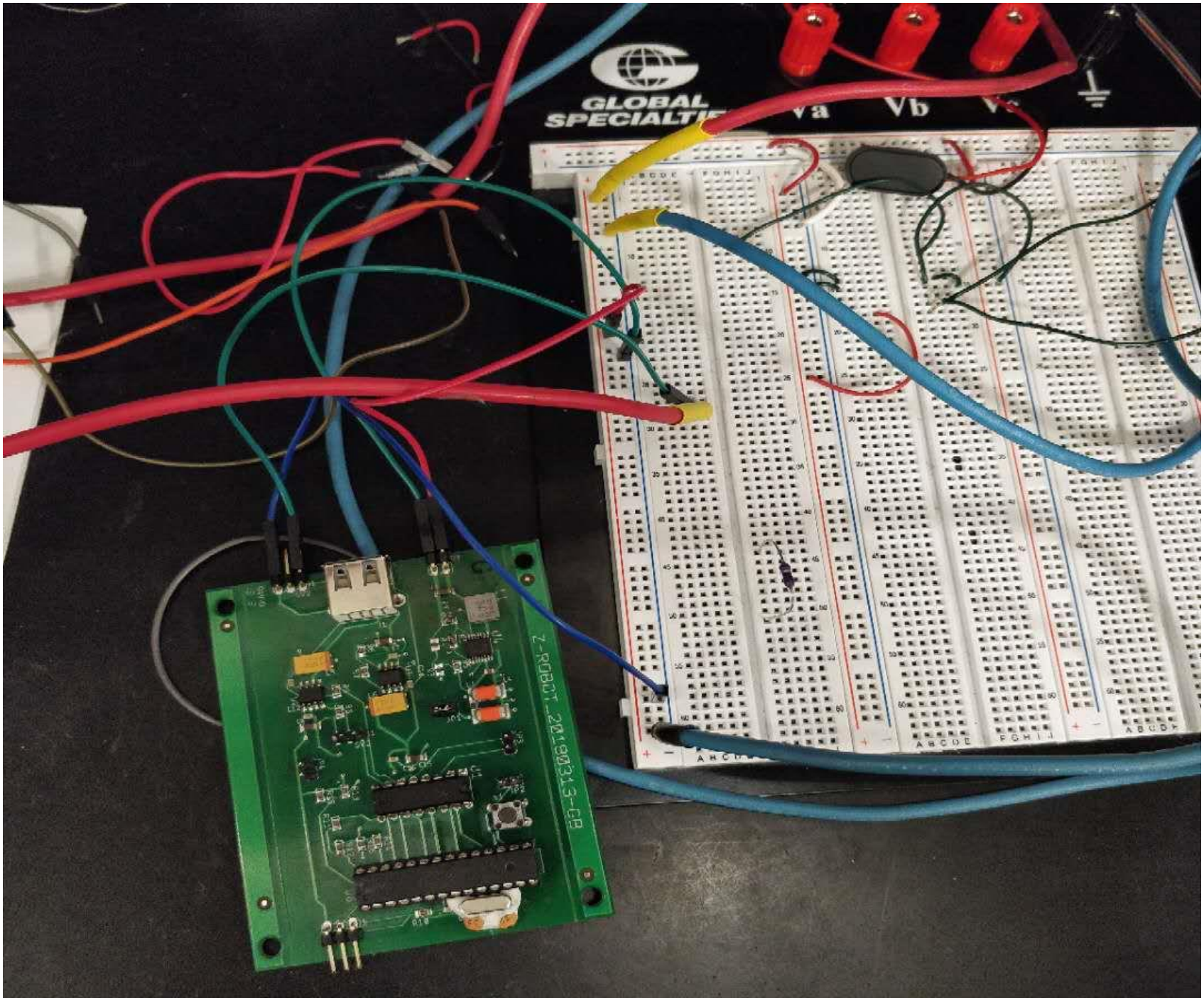
In order to meet requirement of supplying robot at least 30 minutes running, I designed two Lithium batteries (3400mAh * 2) in parallel instead of in series as power source for whole system. Because when the batteries connected in series, we need to check whether the batteries are in saturation. So I use boot converter TPS61032 as first part of circuit to provide system 5V since the main chip and some components are used a stable 5V power

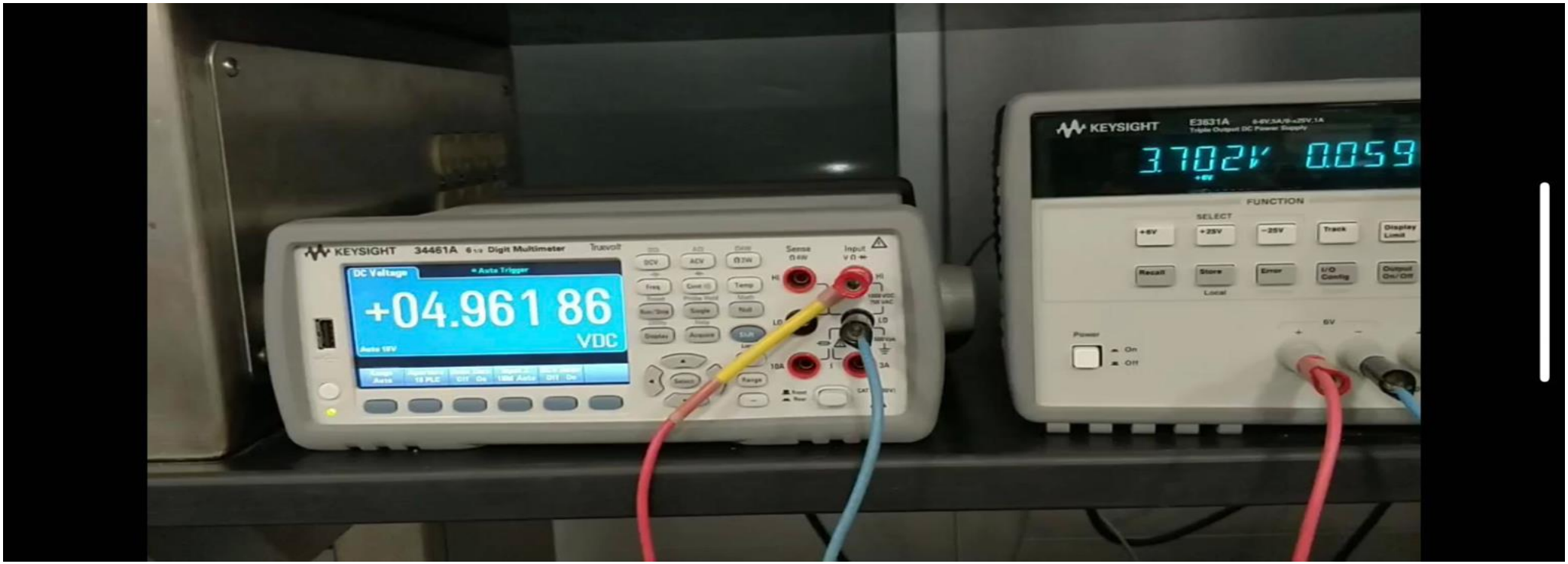
The input range of booster (TPS61032) could be **1.8V to 5.5V**, which cover the both Lithium battery and power bank output. The output current at 5V is up to 4000mA that also meet our design requirement.
I also reserved the R3 and R4 for tuning 5V output accuracy and a jumper JP1 for isolation.

For verification, when I received my PCB board, I could connect the batteries as input and see if the output is fixed 5V. I plan to apply different voltages (2.5V – 3.7V) to the boost converter using function generator in lab. Meanwhile, we will use the multimeter to measure the output voltage of the regulator, which should always be 5V.




U4
VBAT
LBO
EN
VOUT_2
FB
VOUT_3
VOUT
LBI
SYNC
SW_2
SW
NC
PGND_2
PGND_3
PGND
GND
POWERPAD
TPS61032
6
10
9
14
12
15
13
7
8
1
2
16
3
4
5
11
17
BATTERY
L1
6.8uH
R1
560k
R2
180k
C3
10uF/6.3v
R3
1.8M
R4
200k
R5
1M
C4
2.2uF/10V
C1
100uF/10V
C2
100uF/10V
JP1
+5V
GND
GND
GND


GLOBAL
Va
Vb
Z-ROBOT_20190313-00



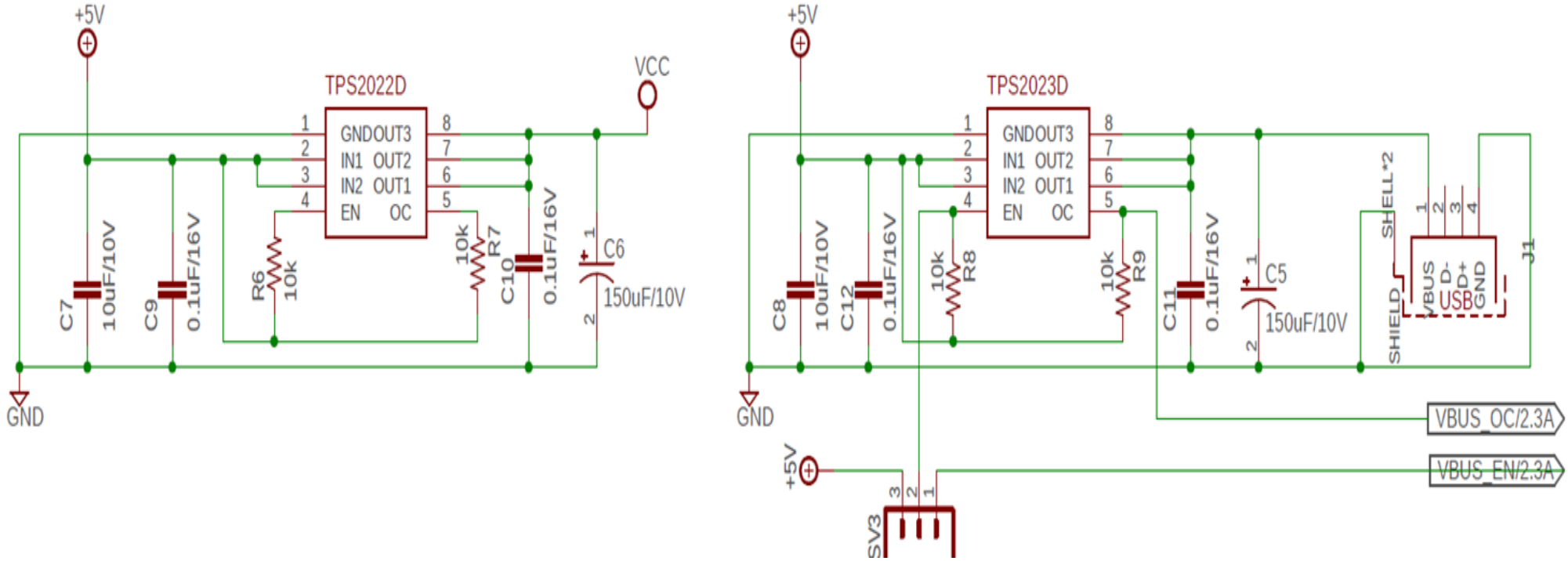
Input Voltage (V)	Output Voltage (V)
1.8	4.935
2.2	4.962
2.6	4.977
3.0	4.965
3.4	4.963
3.7	4.961
4.2	4.965
5.8	5.119

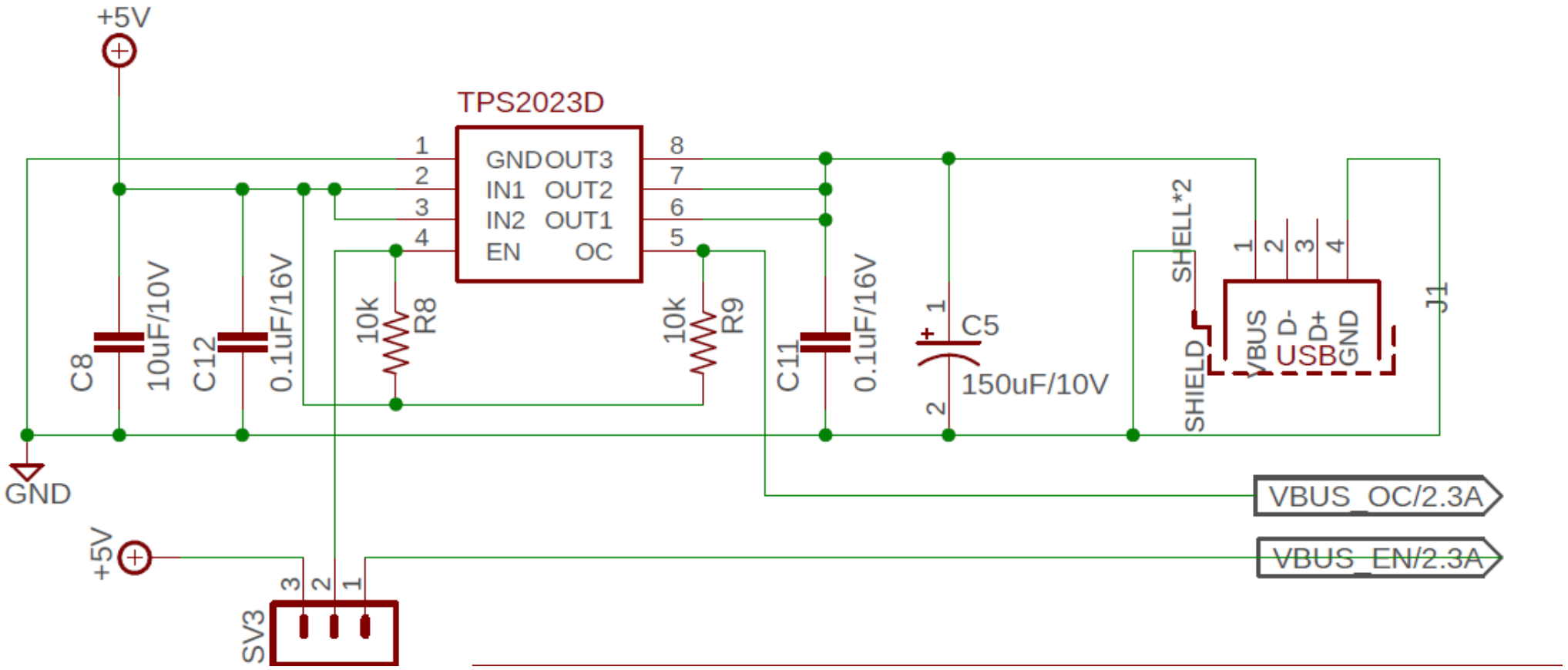
Power Switch (TPS2022D) (TPS2023D)

I added two power-distribution switch with overcurrent protection and undervoltage 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.

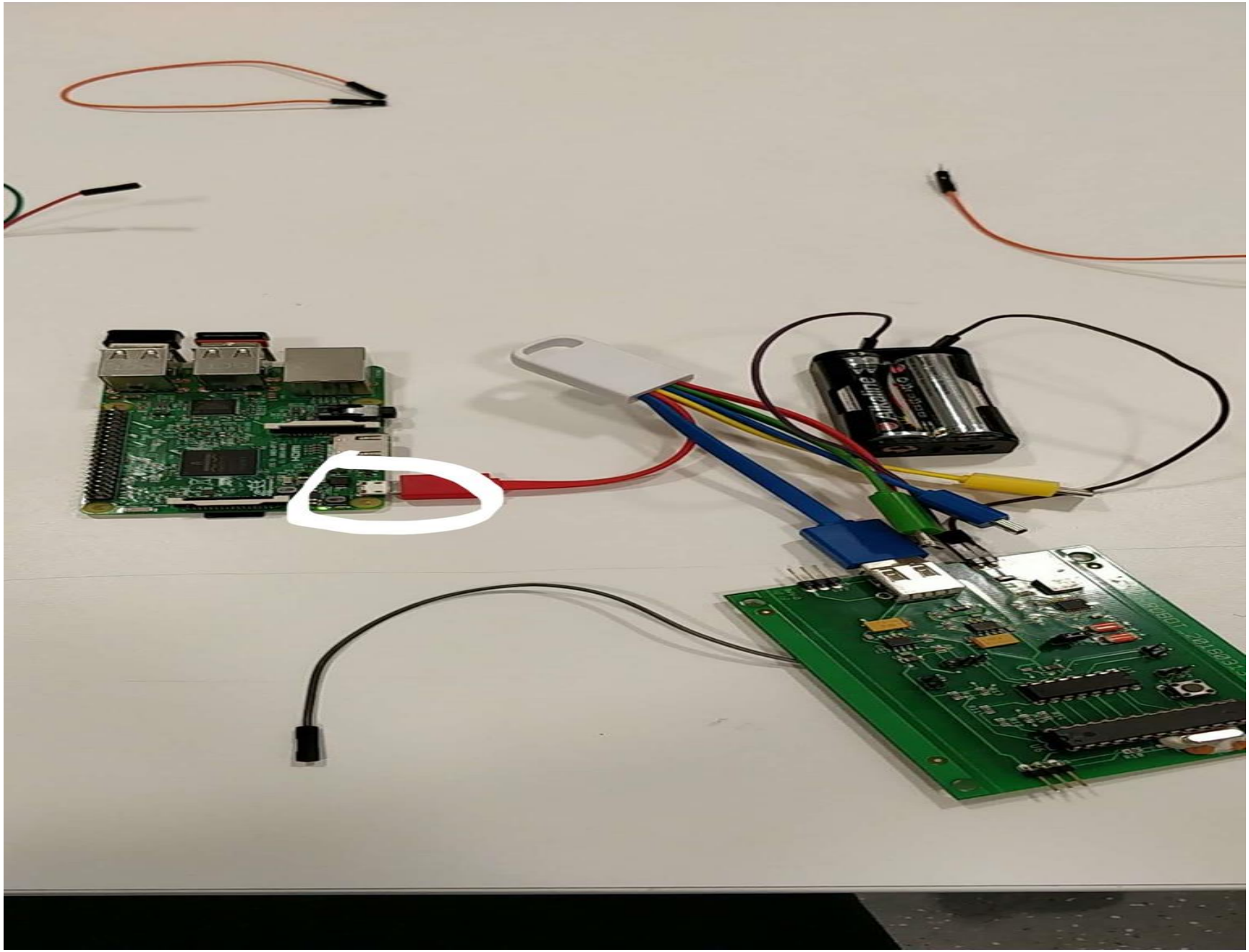
When the output load exceeds the current-limit threshold or a short is present, the TPS202x limits the output current to a safe level by switching into a constant-current mode, pulling the overcurrent (OC) logic output low. When continuous heavy overloads and short circuits increase the power dissipation in the switch, causing the junction temperature to rise, a thermal protection circuit shuts off the switch to prevent damage.

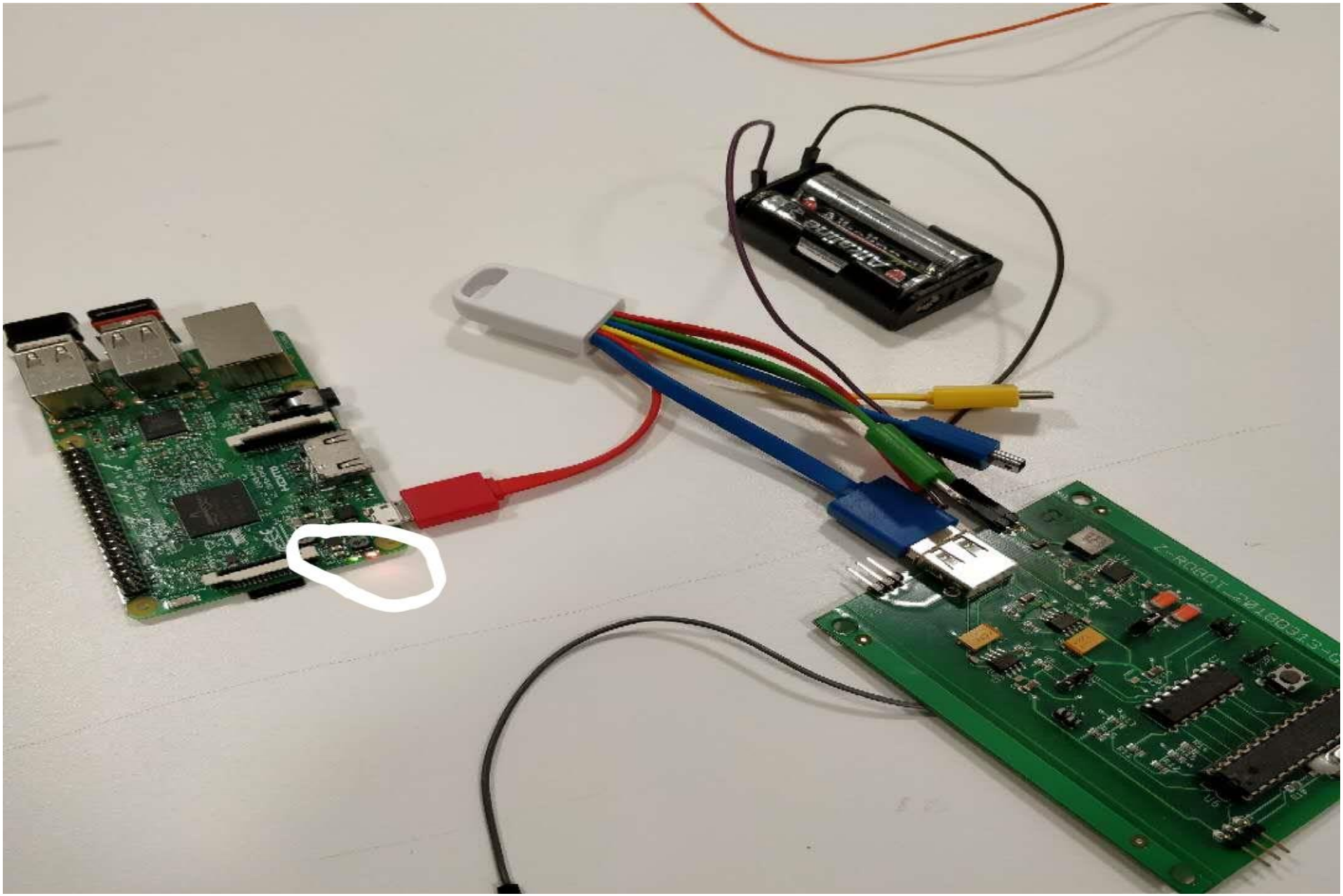
The TPS2022 at 1.5-A load, the TPS2023 at 2.2-A load

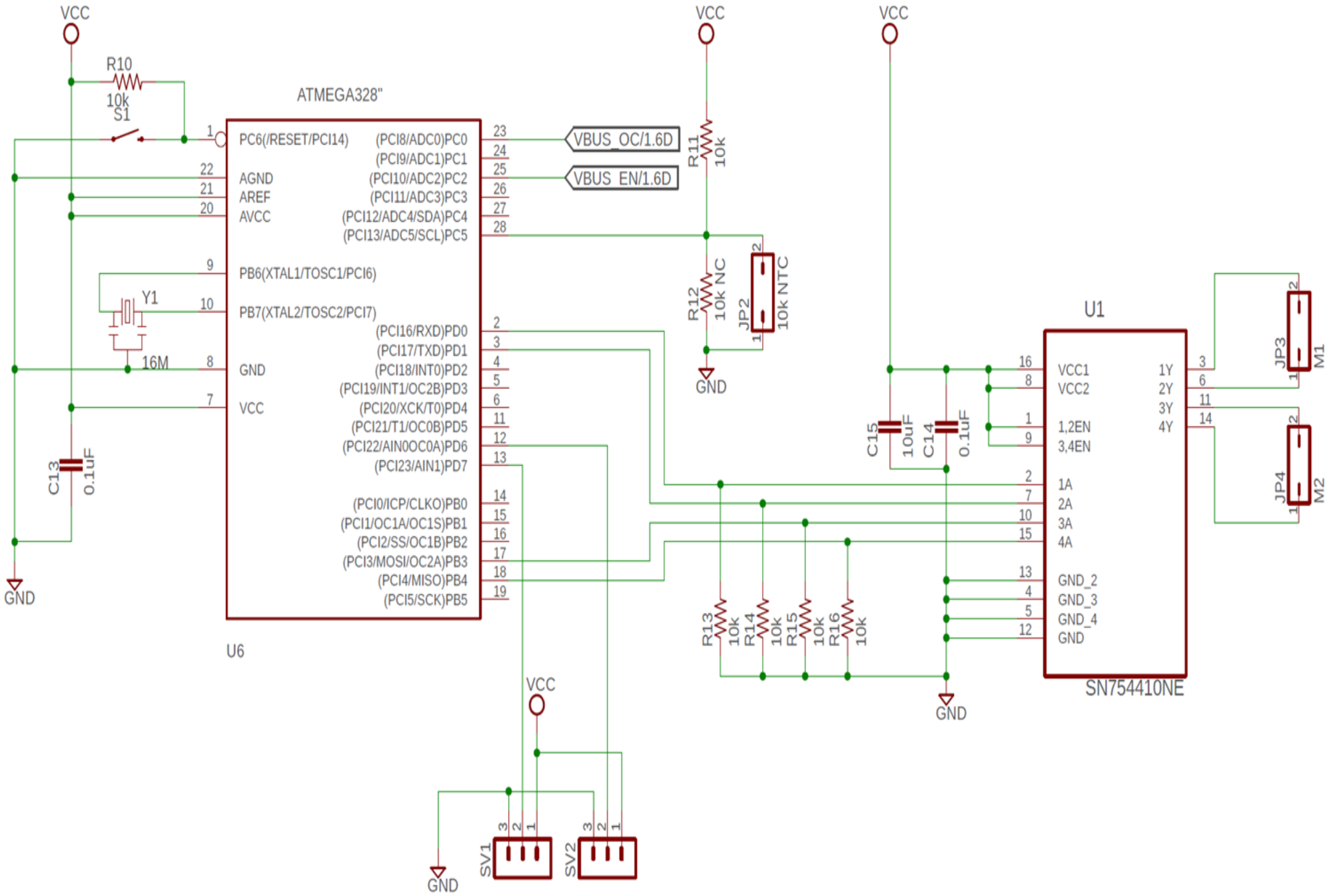




The TPS2023 at 2.2-A load



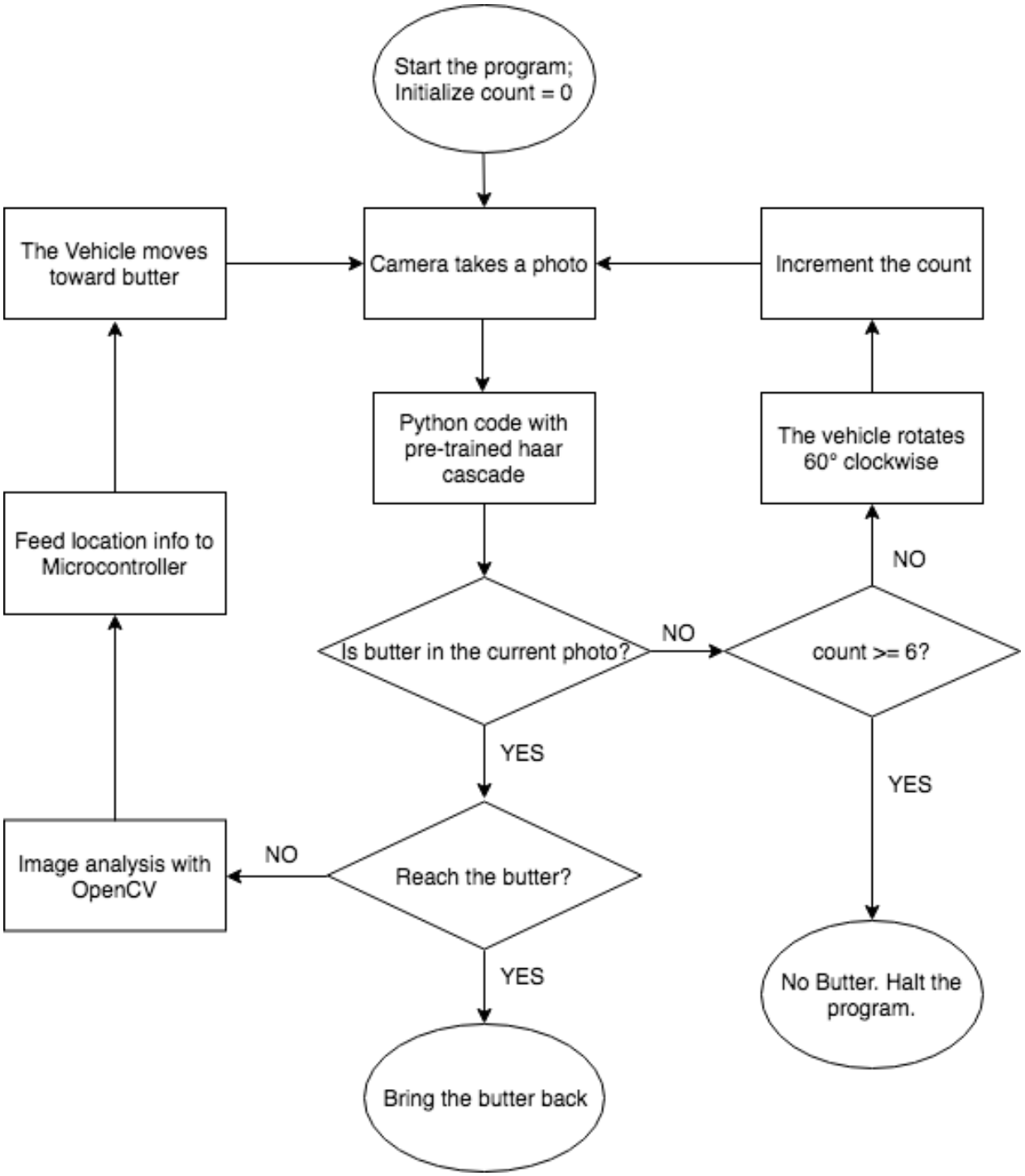



VCC
R10
10k
S1
ATMEGA328"
PC6(/RESET/PCI14)
AGND
AREF
AVCC
PB6(XTAL1/TOSC1/PCI6)
PB7(XTAL2/TOSC2/PCI7)
GND
VCC
(PCI8/ADC0)PC0
(PCI9/ADC1)PC1
(PCI10/ADC2)PC2
(PCI11/ADC3)PC3
(PCI12/ADC4/SDA)PC4
(PCI13/ADC5/SCL)PC5
(PCI16/RXD)PD0
(PCI17/TXD)PD1
(PCI18/INT0)PD2
(PCI19/INT1/OC2B)PD3
(PCI20/XCK/T0)PD4
(PCI21/T1/OC0B)PD5
(PCI22/AIN0OC0A)PD6
(PCI23/AIN1)PD7
(PCI0/ICP/CLKO)PB0
(PCI1/OC1A/OC1S)PB1
(PCI2/SS/OC1B)PB2
(PCI3/MOSI/OC2A)PB3
(PCI4/MISO)PB4
(PCI5/SCK)PB5
U6
Y1
16M
C13
0.1uF
VBUS_OC/1.6D
VBUS_EN/1.6D
R11
10k
R12
10k NC
JP2
10k NTC
C15
10uF
C14
0.1uF
U1
VCC1
VCC2
1,2EN
3,4EN
1A
2A
3A
4A
GND_2
GND_3
GND_4
GND
1Y
2Y
3Y
4Y
SN754410NE
JP3
JP4
M1
M2
R13
10k
R14
10k
R15
10k
R16
10k
SV1
SV2

H-Bridge Logic

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

Flow Chart


Start the program; Initialize count = 0
Camera takes a photo
The Vehicle moves toward butter
Increment the count
Python code with pre-trained haar cascade
The vehicle rotates 60° clockwise
Feed location info to Microcontroller
NO
Is butter in the current photo?
NO
count >= 6?
YES
YES
Image analysis with OpenCV
NO
Reach the butter?
YES
No Butter. Halt the program.
Bring the butter back

Object Detection

Requirement	Verification
The program can detect yellow, cubed butter in the size smaller than 12cm * 3cm * 3cm[10]. The accuracy rate should be 80% at least.	We will write a test program asking the detector to run with at least 500 positive images of cubed butter (either 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.
The program can finish processing an image in 0.5 s, so that the motor can always act in time	We will import the datetime library in python and time the our detection program on Raspberry Pi.
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.

Table. 7. Raspberry Pi and Camera R&V

Object Detection

Trial 1: OpenCV Object Detection API -- Haar Classifier

First Training Session: 1500 images in total

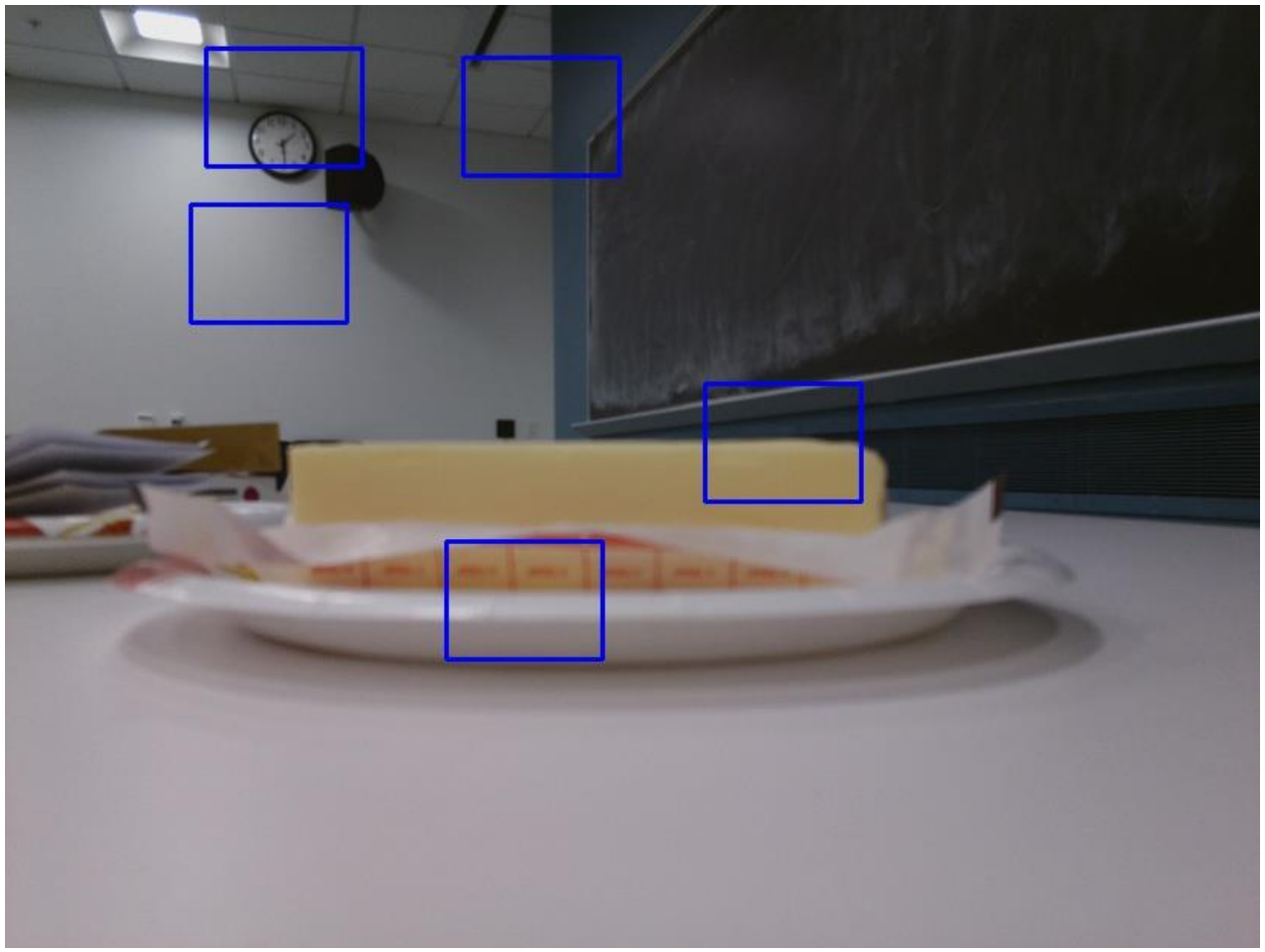
Second Training Session: 10500 images in total

Two Types of Positive Images:

- photos of butter (different angles) taken in the similar environment
- positive photos inserted into backgrounds by OpenCV

Object Detection

Trial 1: OpenCV Object Detection API -- Haar Classifier



Too many false positive

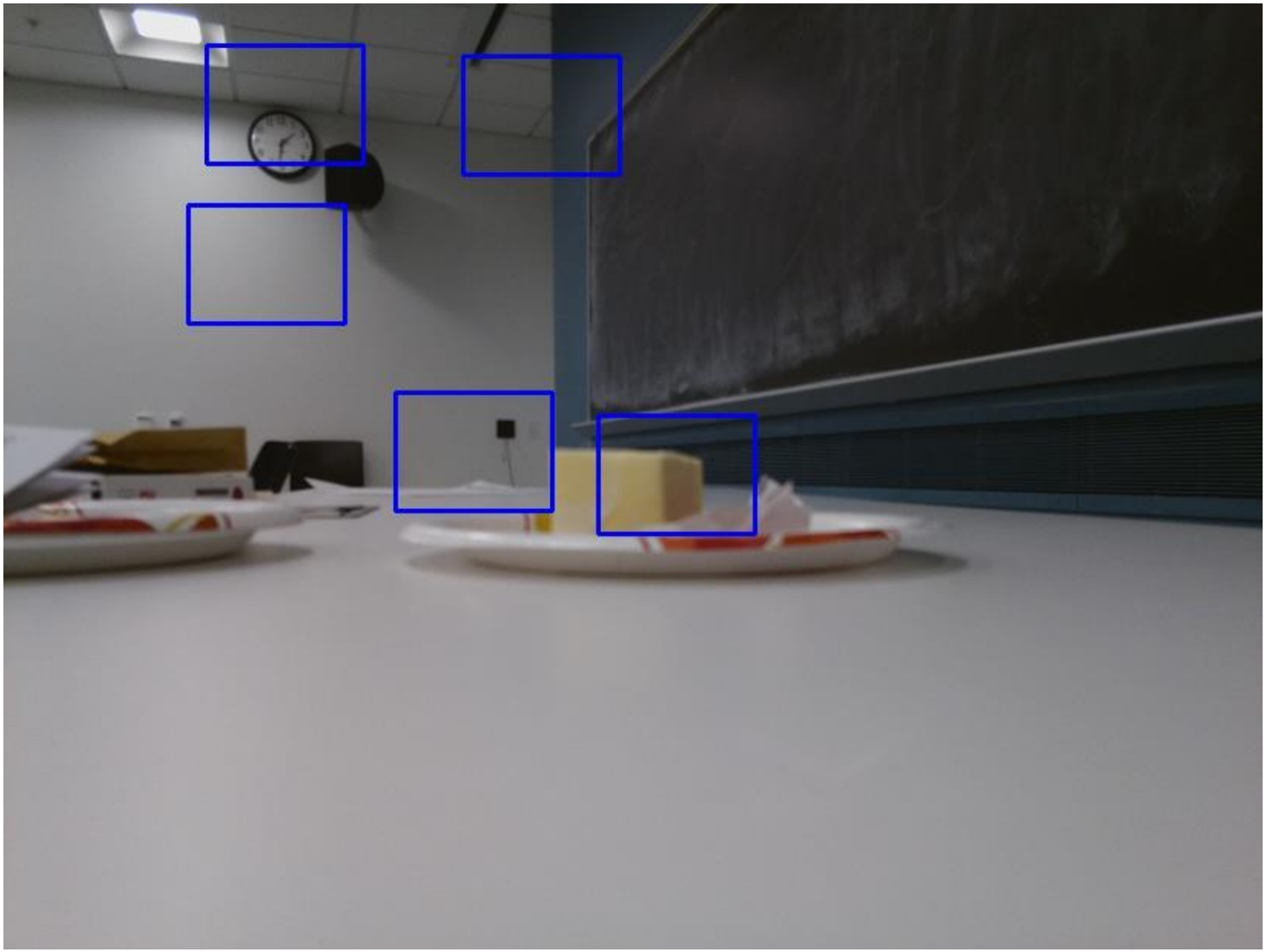
Object Detection

Trial 1: OpenCV Object Detection API -- Haar Classifier

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

Object Detection

Trial 1: OpenCV Object Detection API -- Haar Classifier



Object Detection

Trial 2: Tensorflow Object Detection API with ssd mobilenet

Model name	Speed (ms)	COCO mAP[^1]	Outputs
ssd_mobilenet_v1_coco	30	21	Boxes
ssd_mobilenet_v2_coco	31	22	Boxes
ssd_inception_v2_coco	42	24	Boxes
faster_rcnn_inception_v2_coco	58	28	Boxes
faster_rcnn_resnet50_coco	89	30	Boxes
faster_rcnn_resnet50_lowproposals_coco	64		Boxes
rfcn_resnet101_coco	92	30	Boxes
faster_rcnn_resnet101_coco	106	32	Boxes
faster_rcnn_resnet101_lowproposals_coco	82		Boxes

Object Detection

Trial 2: Tensorflow Object Detection API with ssd mobilenet

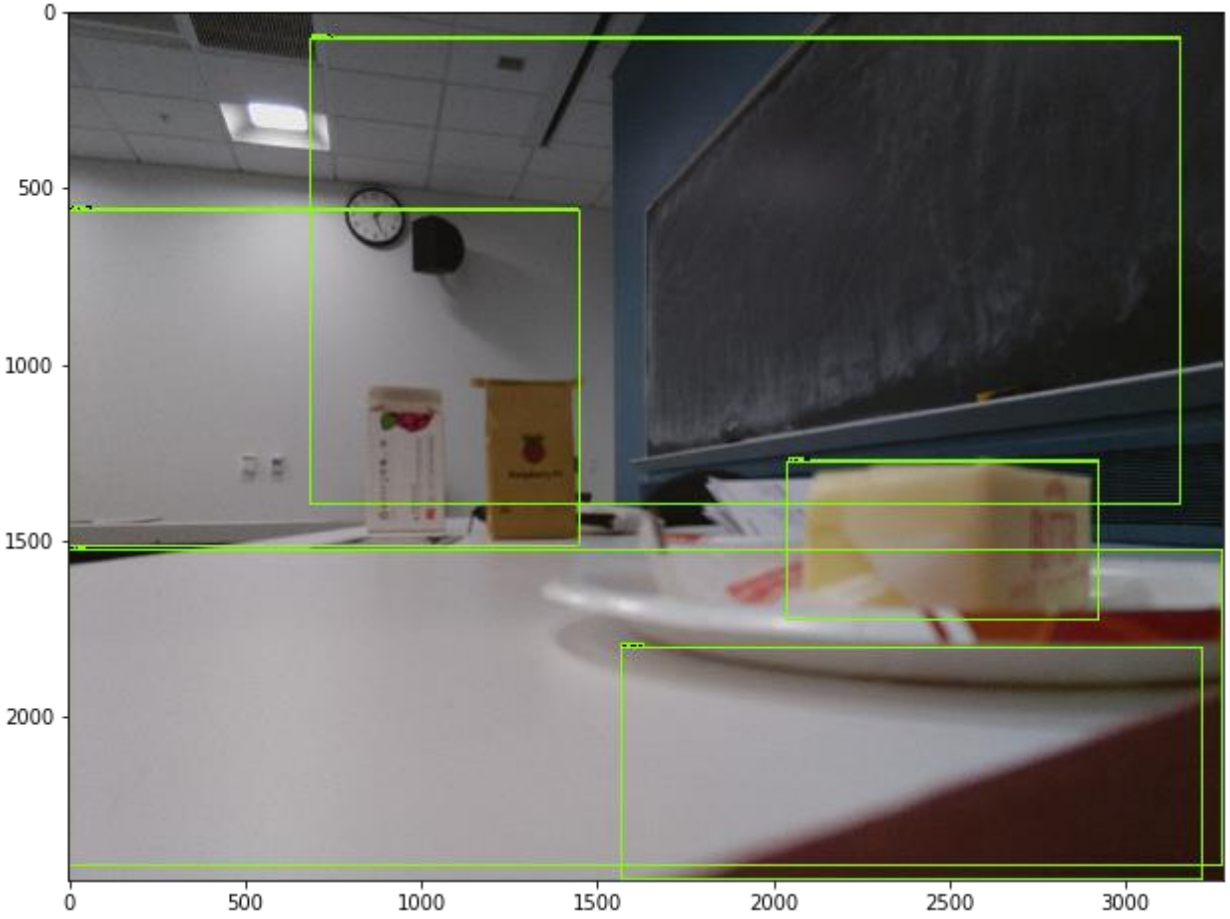
First

Secc

Object Detection

Trial 2: Tensorflow Object Detection API with ssd mobilenet

Works Okay with images

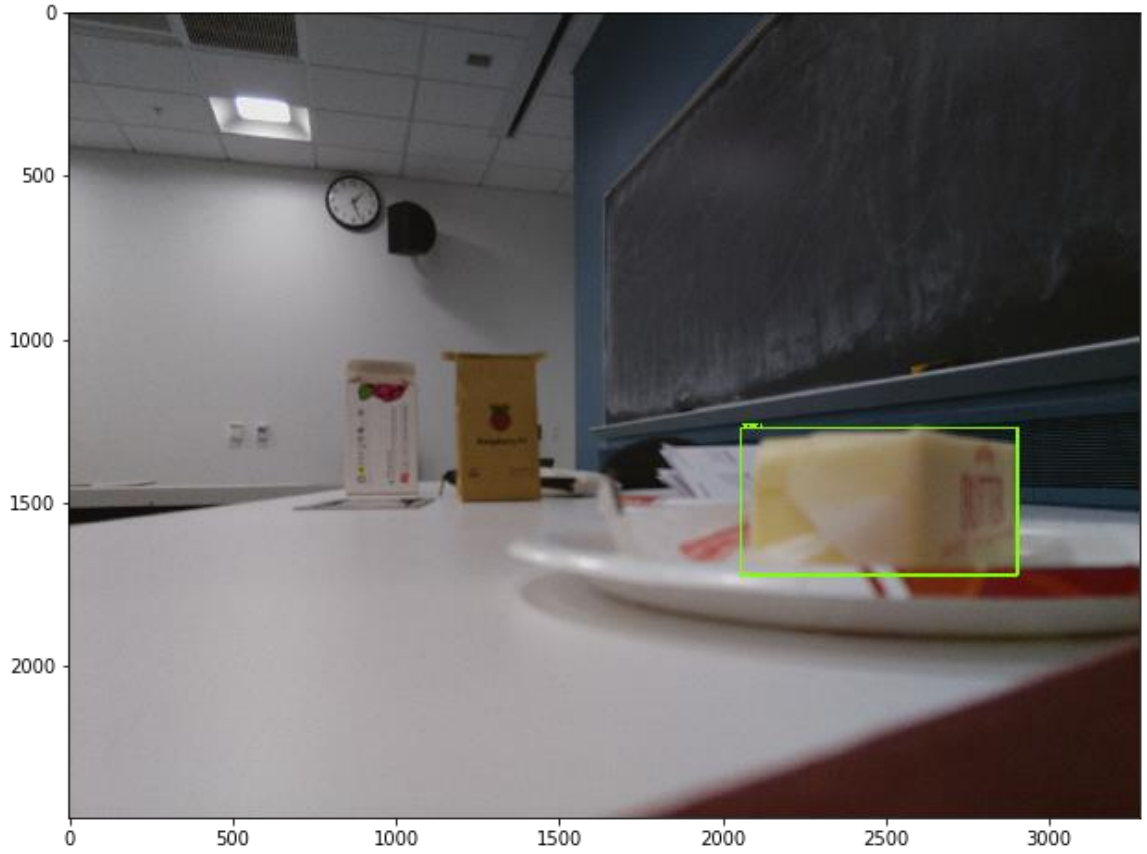


After 1587 steps

Object Detection

Trial 2: Tensorflow Object Detection API with ssd mobilenet

Works Okay with images

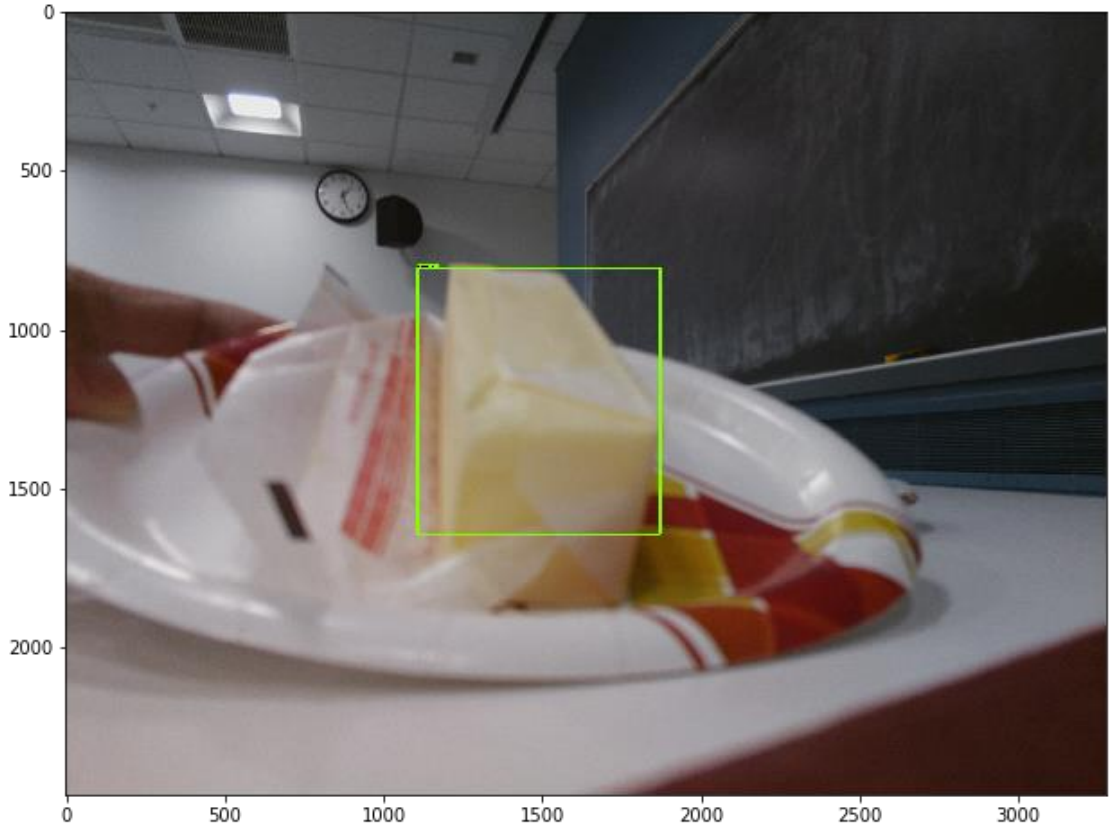


After 17349 steps

Object Detection

Trial 2: Tensorflow Object Detection API with ssd mobilenet

Accuracy decreases later



After 17349 steps

Object Detection

Trial 2: Tensorflow Object Detection API with ssd mobilenet

Accuracy decreases later

After 59640 steps

Object Detection

Trial 2: Tensorflow Object Detection API with ssd mobilenet

	Image Index	17349_graph	59640_graph
2	1	yes	no
3	2	yes	yes
4	3	yes	yes
5	4	yes	no
6	5	yes	yes
7	6	yes	yes
8	7	no	no
9	8	yes	no
10	9	yes	no
11	10	yes	yes
12	11	no	no
13	12	no	no
14	13	no	no

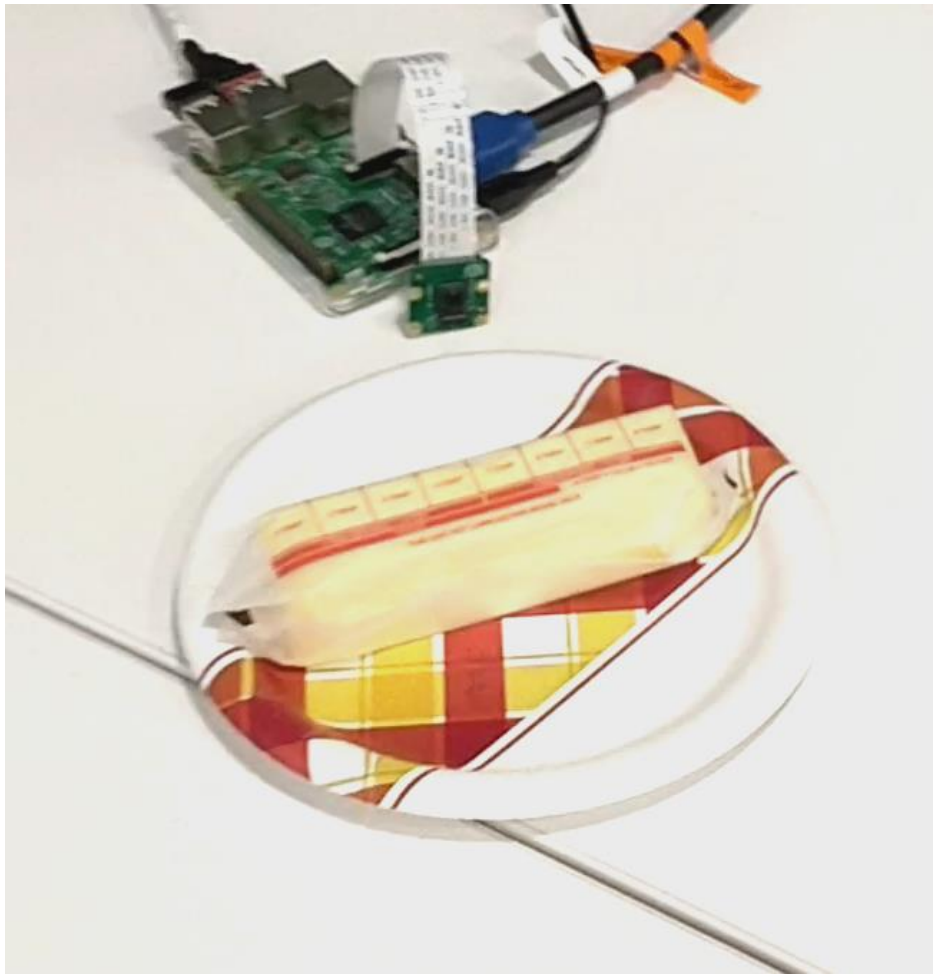
67%

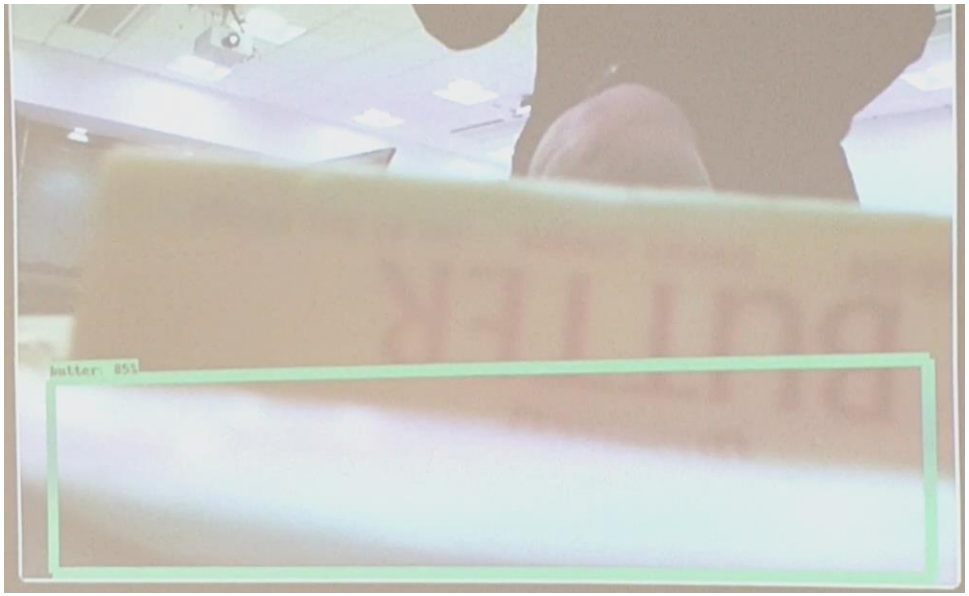
35%

Object Detection

Trial 2: Tensorflow Object Detection API with ssd mobilenet

Won’t work with video stream





The butter has to be very close to the camera

Object Detection

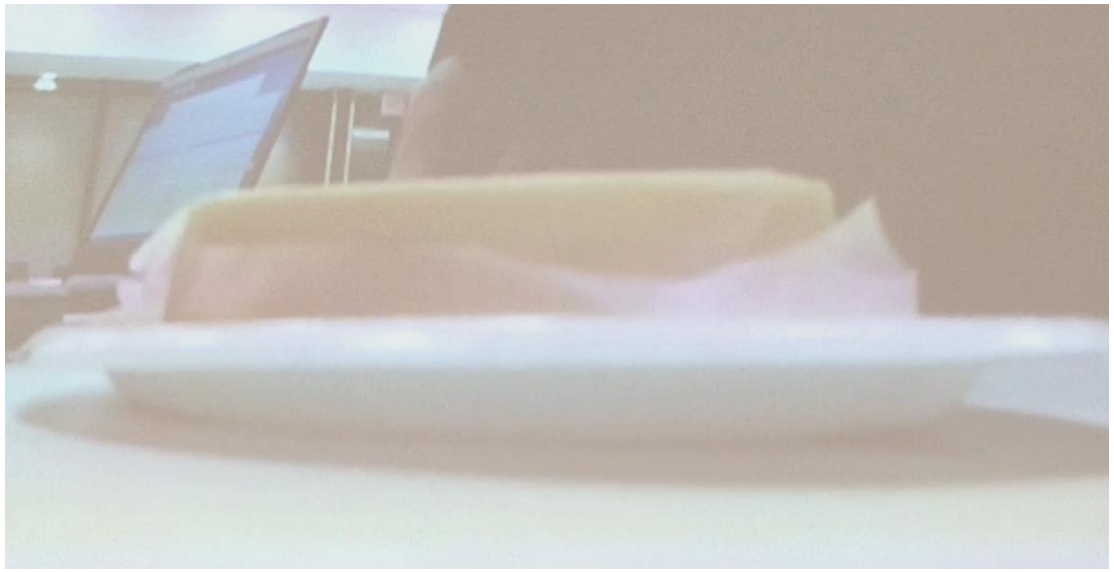
Trial 2: Tensorflow Object Detection API with ssd mobilenet

Second Training Session: 2300 images

- Took more photos in the target environment
- Placed the butter further away from camera
- when manually selecting target region in images, included the plate

Object Detection

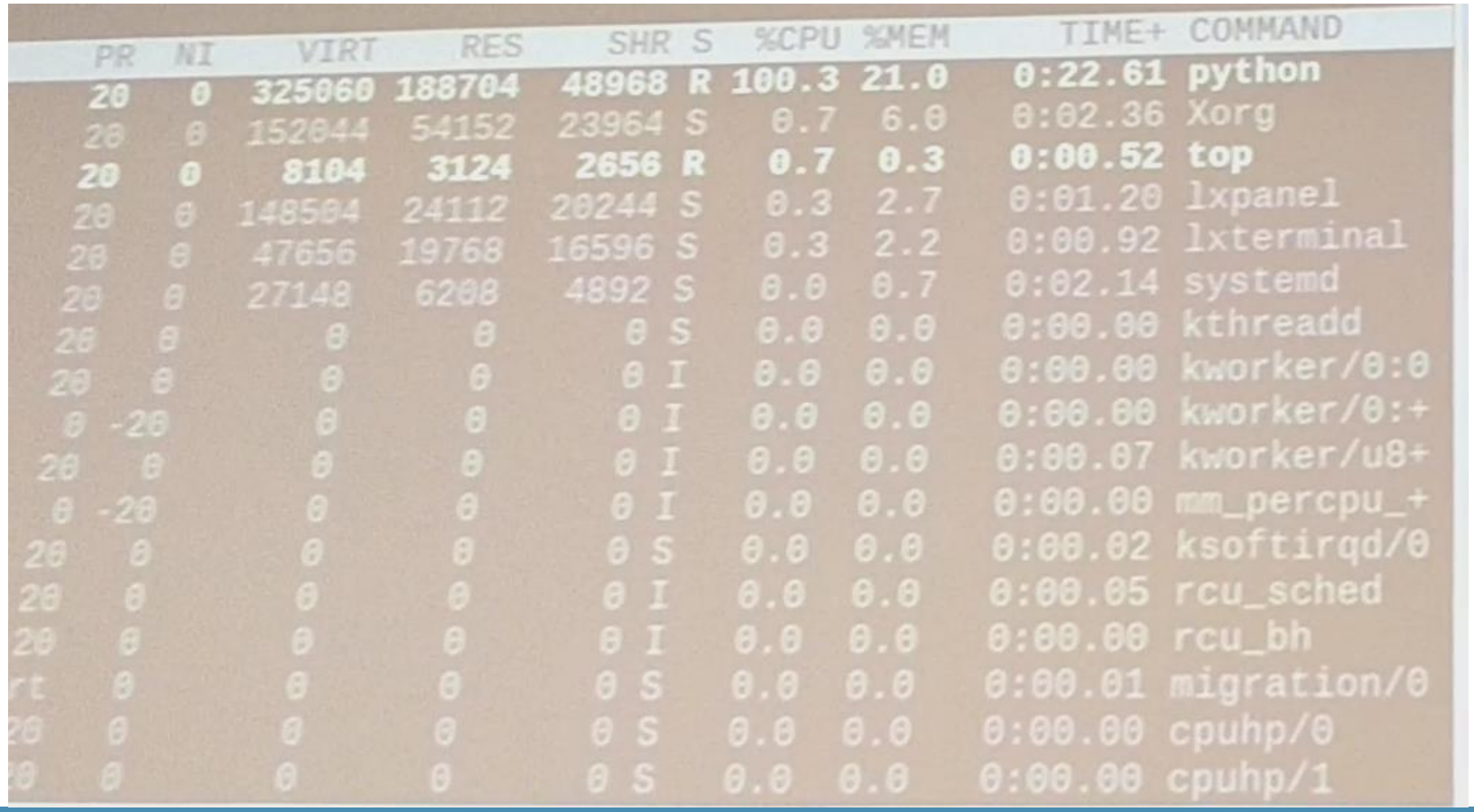
Trial 2: Tensorflow Object Detection API with ssd mobilenet



Object Detection

Trial 2: Tensorflow Object Detection API with ssd mobilenet

High CPU load (which can go up to 300%) causes Pi to freeze



PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
20	0	325060	188704	48968	R	100.3	21.0	0:22.61	python
20	0	152044	54152	23964	S	0.7	6.0	0:02.36	Xorg
20	0	8104	3124	2656	R	0.7	0.3	0:00.52	top
20	0	148504	24112	20244	S	0.3	2.7	0:01.20	lxpanel
20	0	47656	19768	16596	S	0.3	2.2	0:00.92	lxterminal
20	0	27148	6208	4892	S	0.0	0.7	0:02.14	systemd
20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
20	0	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0
0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:+
20	0	0	0	0	I	0.0	0.0	0:00.07	kworker/u8+
0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_+
20	0	0	0	0	S	0.0	0.0	0:00.02	ksoftirqd/0
20	0	0	0	0	I	0.0	0.0	0:00.05	rcu_sched
20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_bh
rt	0	0	0	0	S	0.0	0.0	0:00.01	migration/0
20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/1

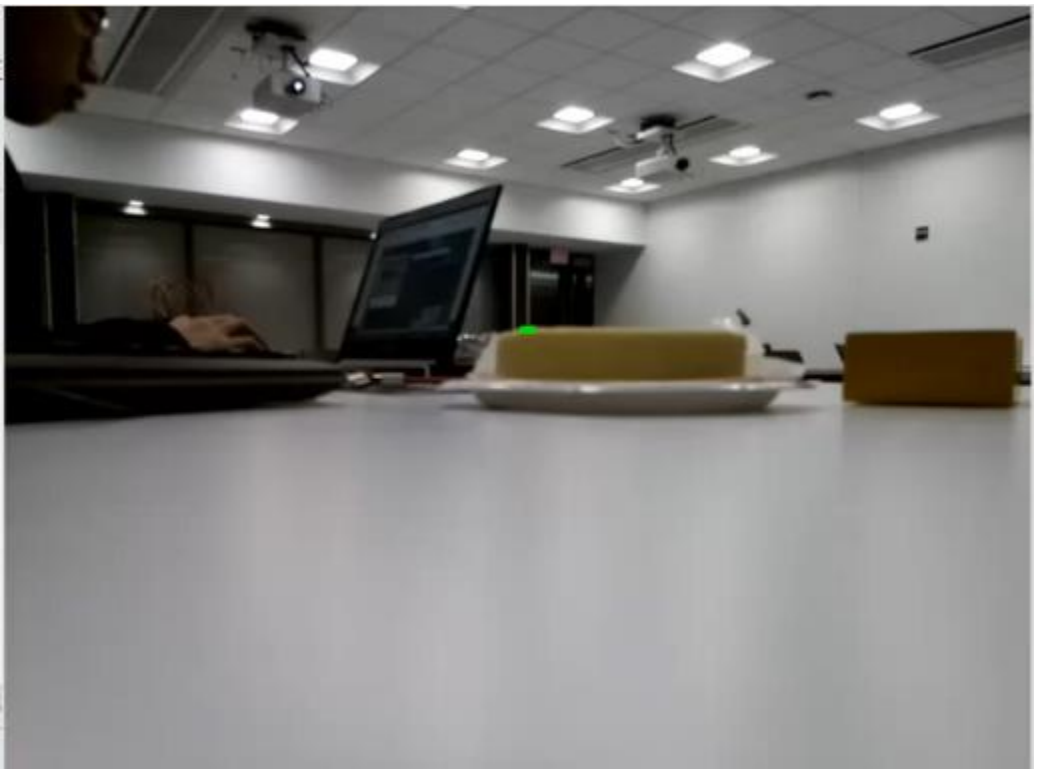
Object Detection Trial 3: HSV Color Detection

- Define a HSV color range
  - tested with different values
  - also used opencv to get the color value in the actual image. Surprisingly, this range doesn't work well

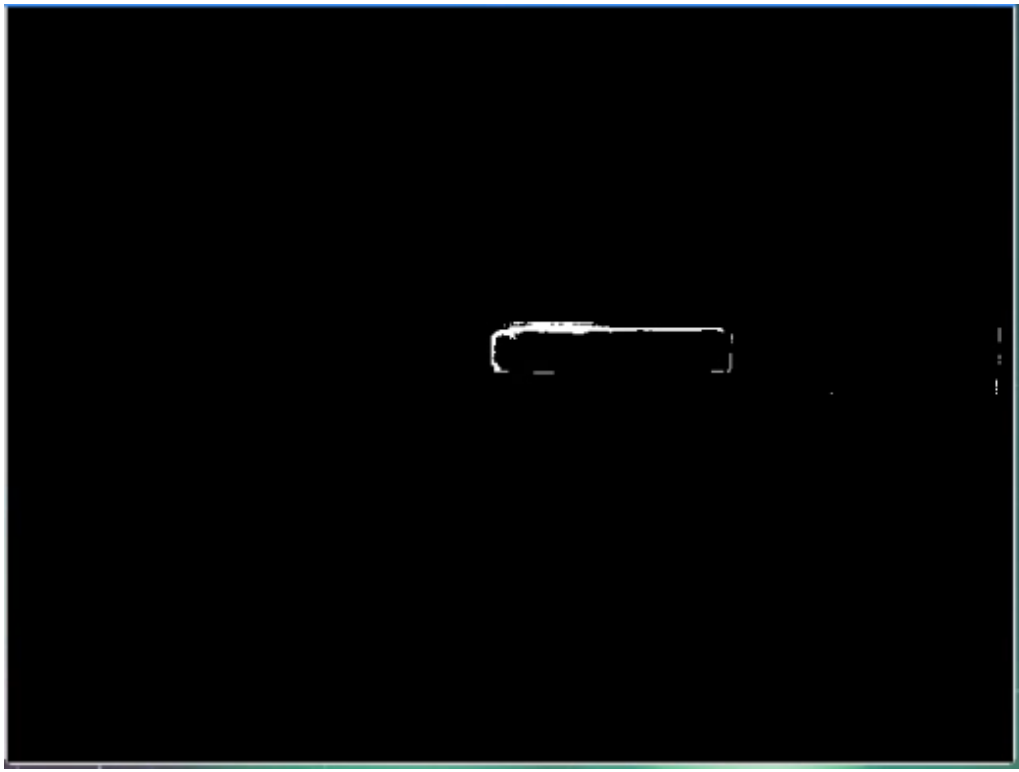
- Apply the color range on the image to get a "mask" image

- Find contours in the "mask" image
  - Among all the contours, we only used the largest one to derive location info about the butter

Object Detection Trial 3: HSV Color Detection



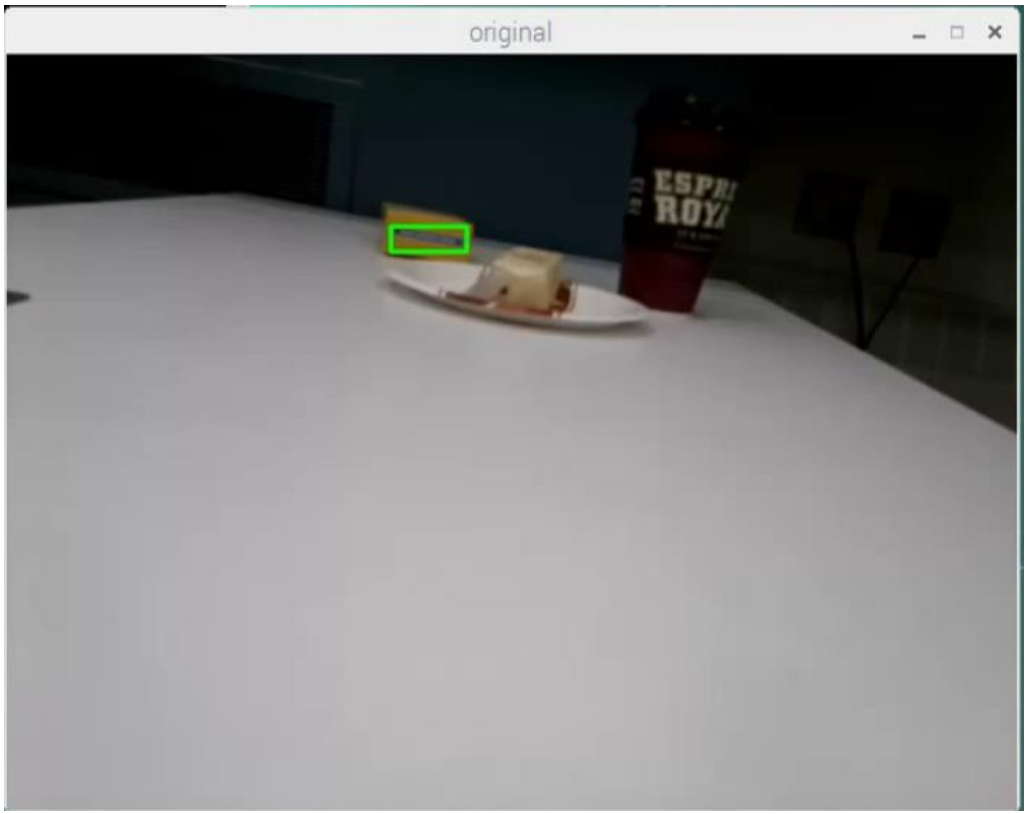
actual image

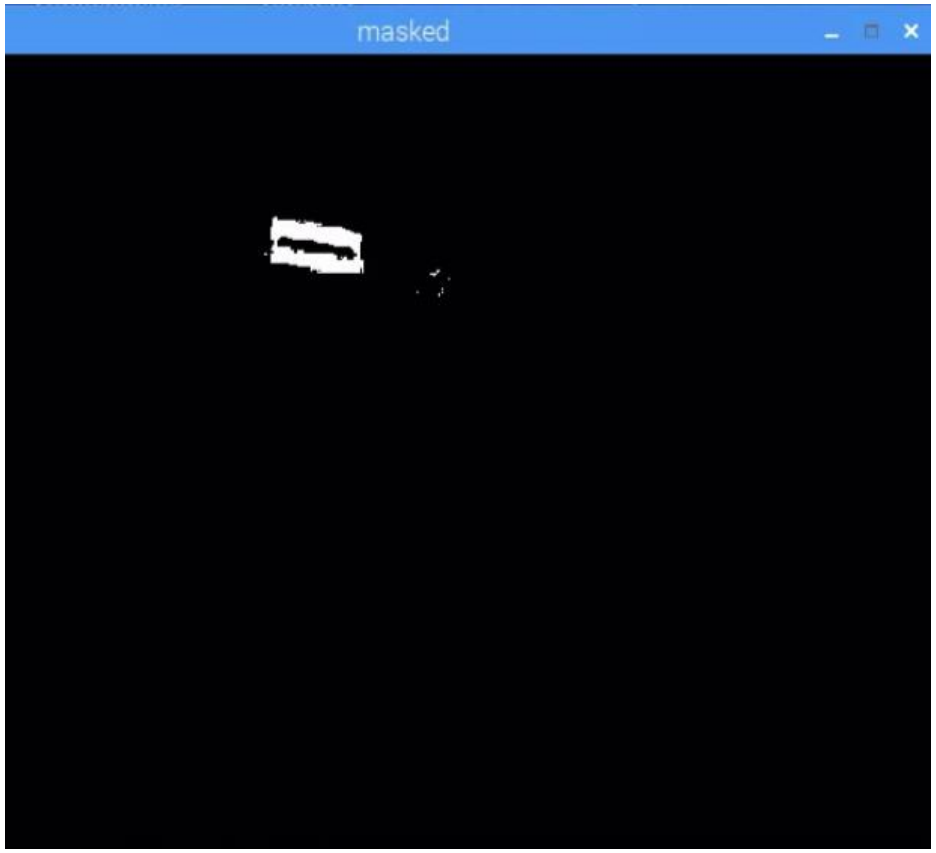


mask image

Object Detection Trial 3: HSV Color Detection

- works well with just butter
- accuracy drops drastically with similar-color object present





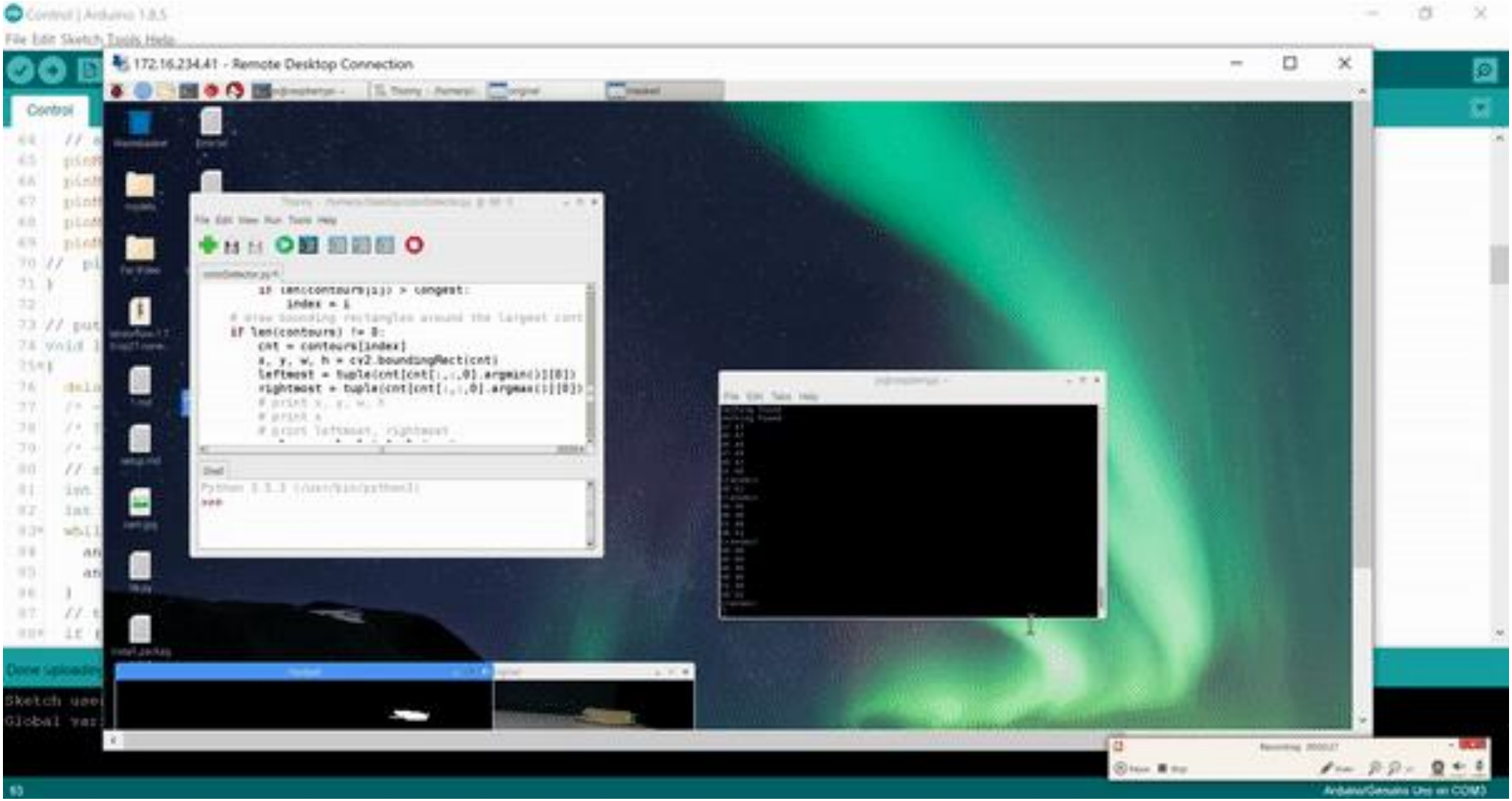
Object Detection Trial 3: HSV Color Detection

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

Object Detection Trial 3: HSV Color Detection

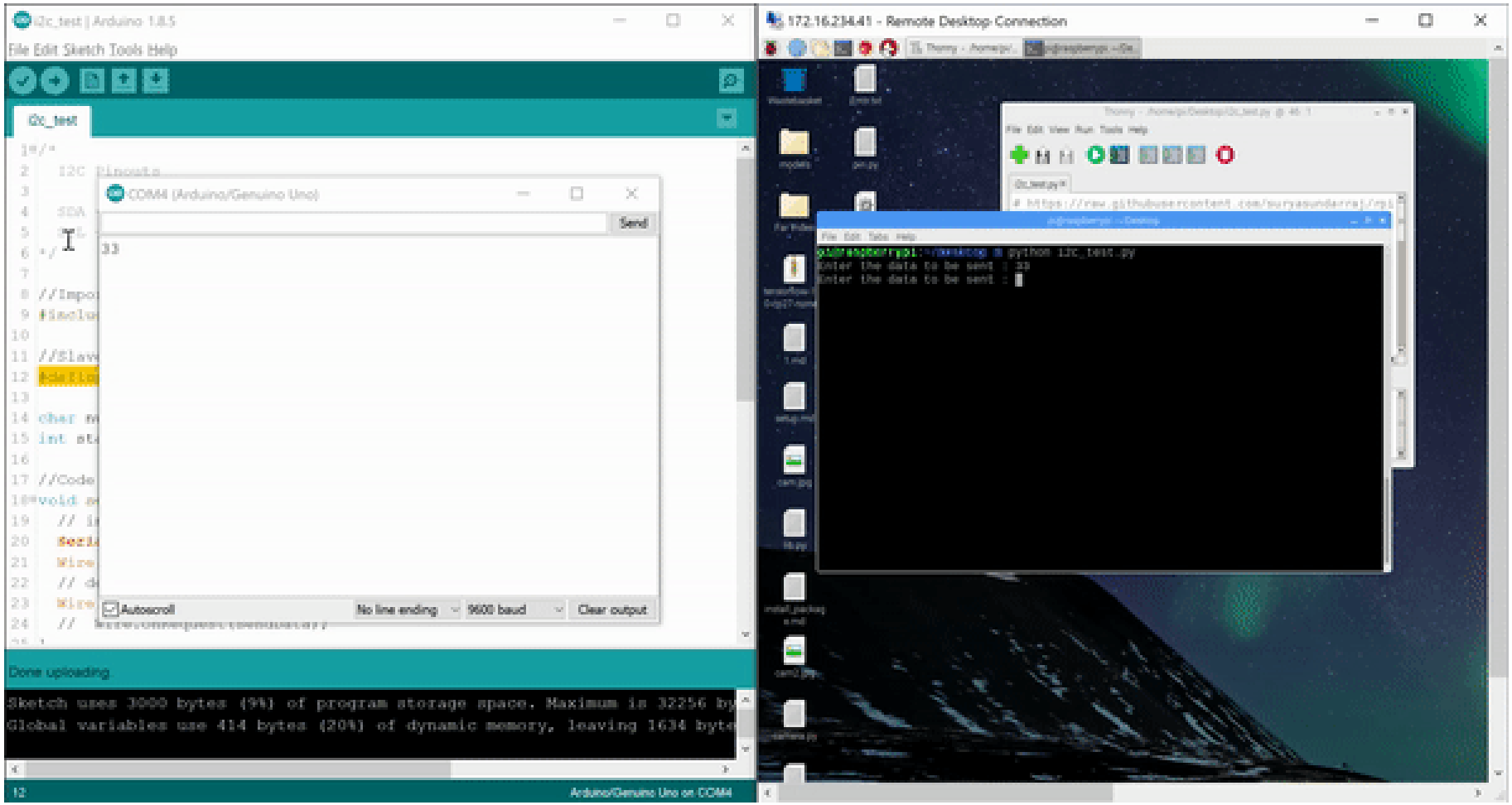
	Detected	Not Detected
With Similar Color Object	49.13%	50.87%
Without Similar Color Object	99.70%	0.30%
Overall	75.98%	24.02%

Object Detection Trial 3: HSV Color Detection



Object Detection Pi-MCU Communication

- i2c communication protocol



Future Work

- integration between software parts and hardware platform
- object detection
  - Optimization for HSV method:
    - could merge close contours
  - Use different photos to train another model
    - keep the package, which may contain more unique features

Acknowledgement

We would like to express our appreciation to:

- our TA, Xinrui Zhu
- Professor Kumar
- all the course staff of ECE 445