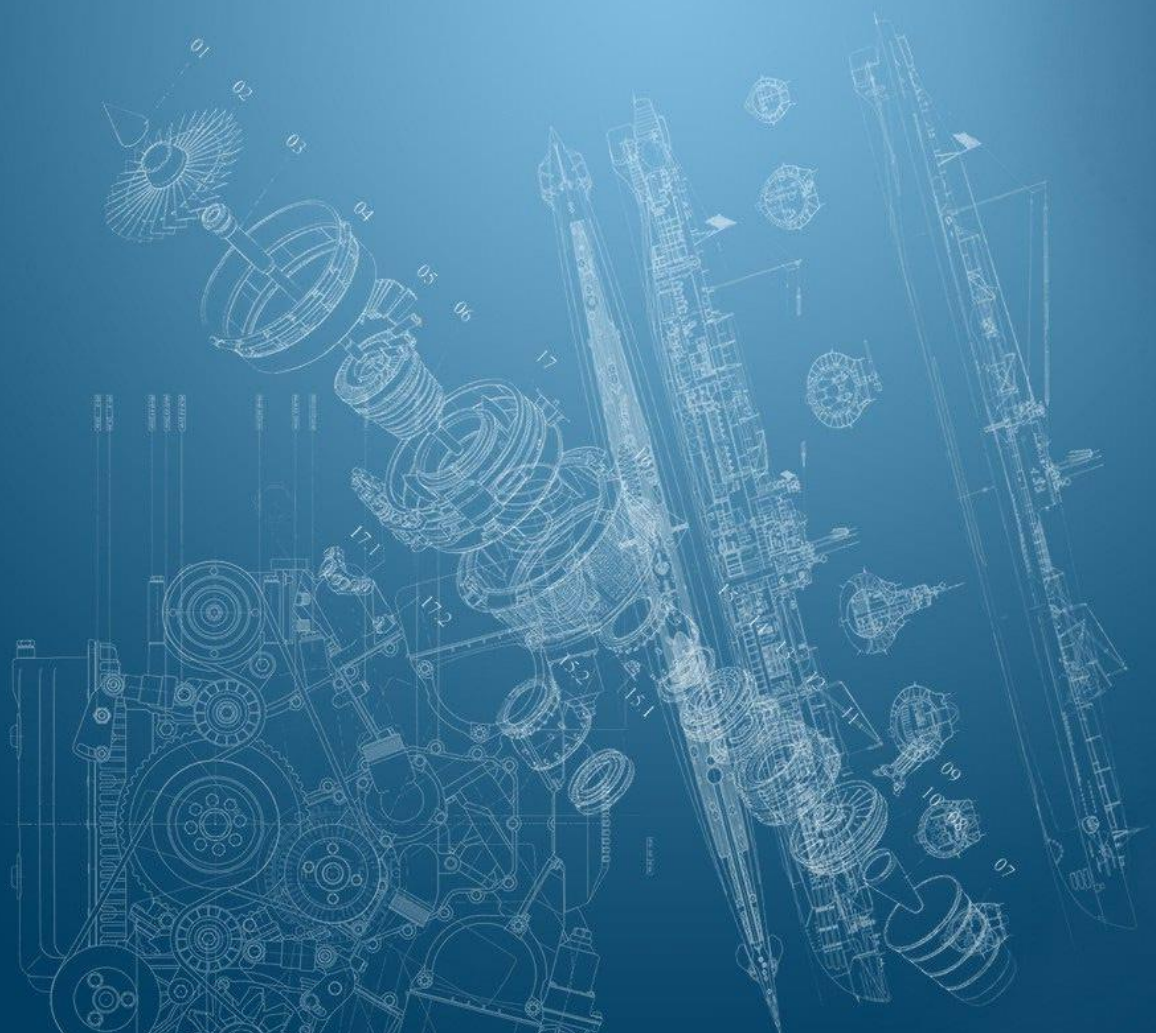
# Final Project Report
# Mechatronics and System Design-Mechatronics IV

*Advisor: Jui-Jen Chou*

*Group 7*
*B07611048 Cheng-Yen Chung*
*B07611001 Li-Wei Yang*
*B07611004 Chuan-Che Lin*



# Table of Contents

# Work Division

## Hardware

Camera design – Cheng Yen
Caster wheel design – Li Wei
Gripper design – Cheng Yen
Circuit design – Cheng Yen
Chassis – Chuan Che

## Software

Arduino software design – Chuan Che
Processing software design – Li Wei
Video Streaming design – Li Wei

## Others

Competition operator – Chuan Che
Video editing – Chuan Che
Word report – Li Wei, Cheng Yen
PowerPoint presentation – Li Wei, Cheng Yen
Parts shopping – Chuan Che, Li Wei, Cheng Yen

# Abstract

The goal of our final project is to

- Control the mobile robot remotely to grab the ball to the finish line.
- The onboard accelerometer would measure the translational acceleration as well as rotational acceleration to serve as the grading criteria.
- The mobile robot should send visual feedback to the operator, and be able to avoid obstacles under remote commands.

To achieve all the goals above, we designed a mobile robot which satisfies all functions we need.

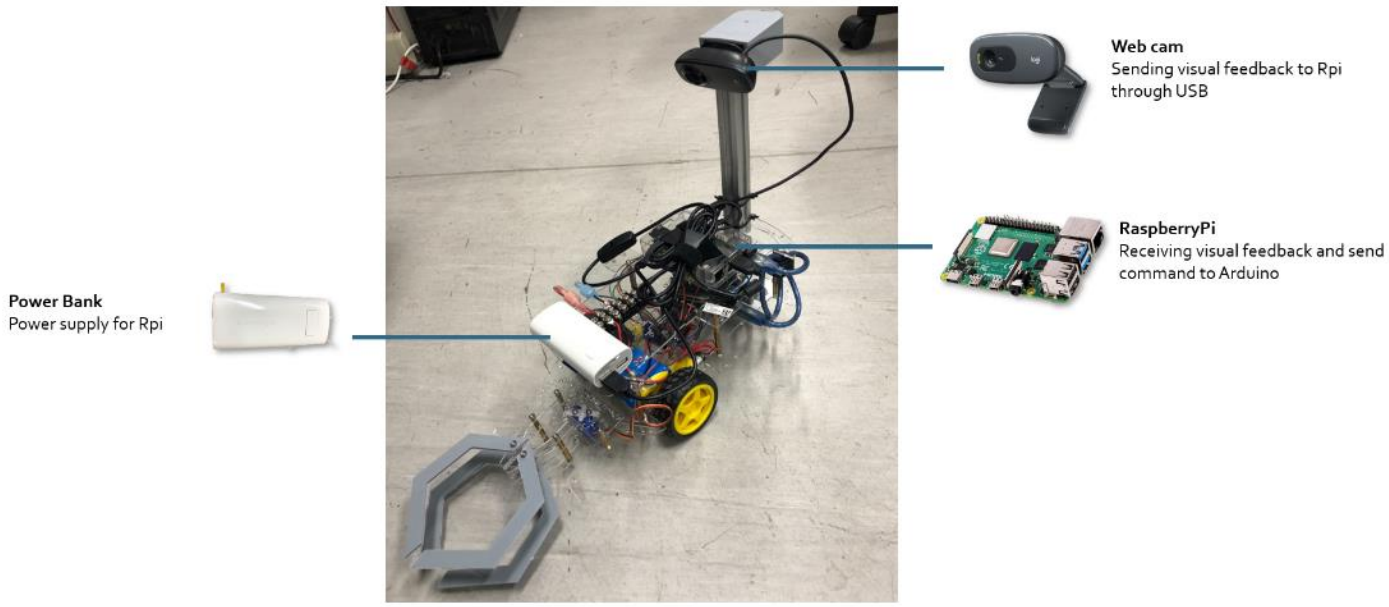
# System introduction

## Hardware Design

### Devices

*Top-layer*

- WebCam (Logitech C310): Sending visual feedback to RaspberryPi through USB
- RaspberryPi model 3b+: Receiving visual feedback and send command to Arduino
- Power Bank (5V): 5V Power supply for Rpi




*Lower-layer*

-Arduino UNO: Sending PWM signals to mortors

-Servo Motor SG-90: Control gripper

-Gripper: Designed to grab ball

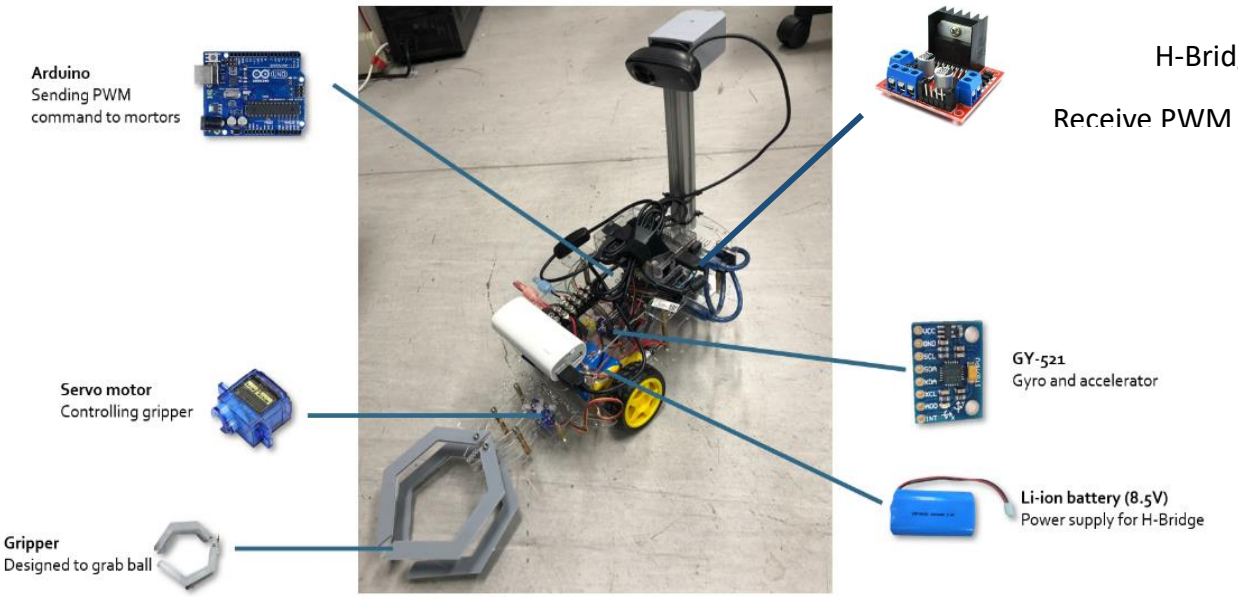
-Gyro (GY-521): Gyro and accelerator

-H-Bridge (L-298N): receiving PWM signal and send it to wheel mortors

-8.5V Li-ion Battery: Power supply for H-Bridge

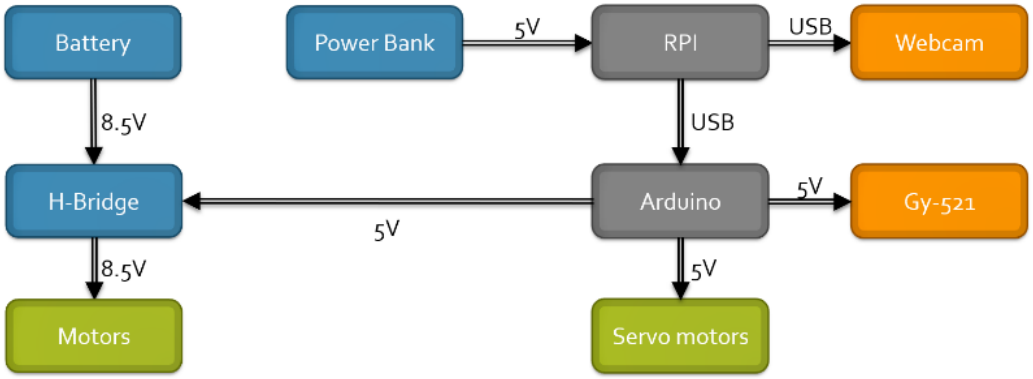
-Caster wheel (LEGO EV3): support the chassis

-Motors (TT motors): move the chassis

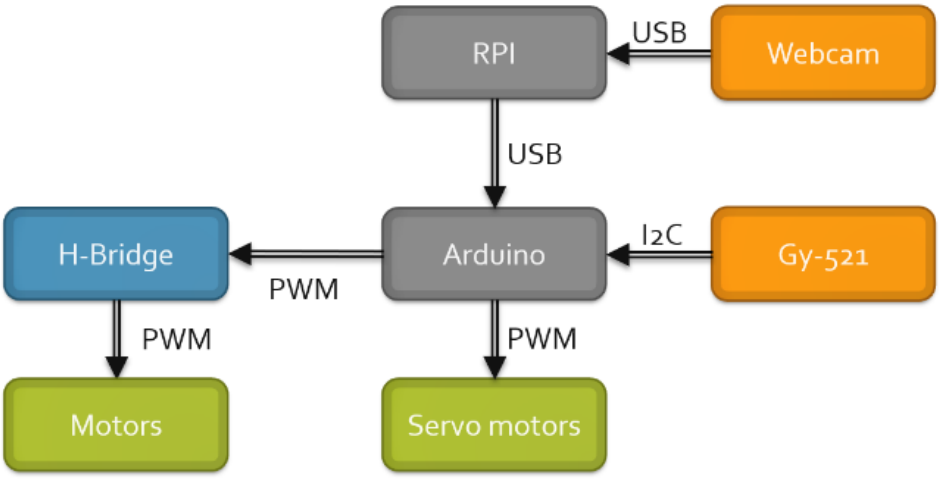



## Circuit design

***For Power,*** we design the circuit by having two powers. Power bank for RPI by 5V and battery for H-Bridge by 8.5V. The Gy-521 and servo motor will be powered up by Arduino at the same time. (5V as well) As ***for signal,*** the operator can receive visual feedback by webCam and send command by RPI. RPI will then send command to Arduino and Arduino will send PWM signal to all drivers/motors.

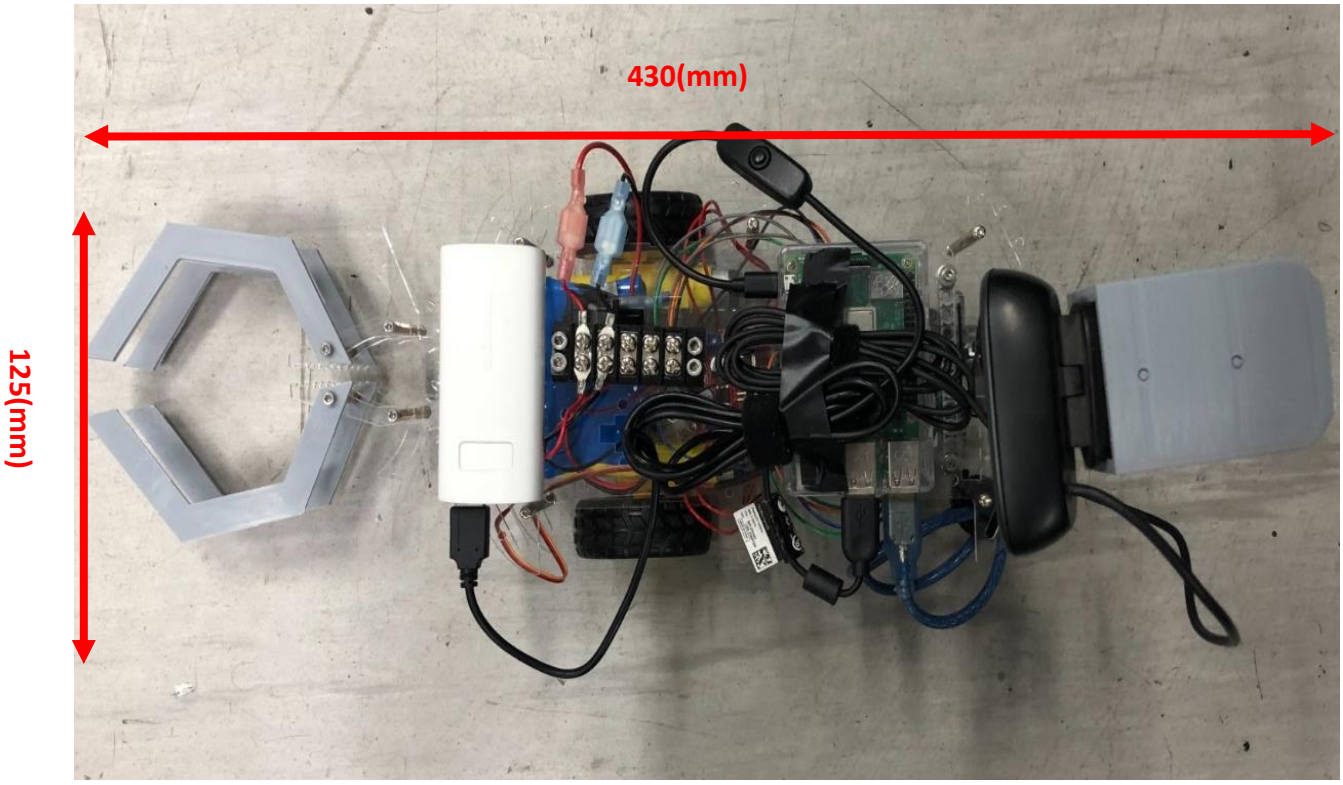



Power circuit

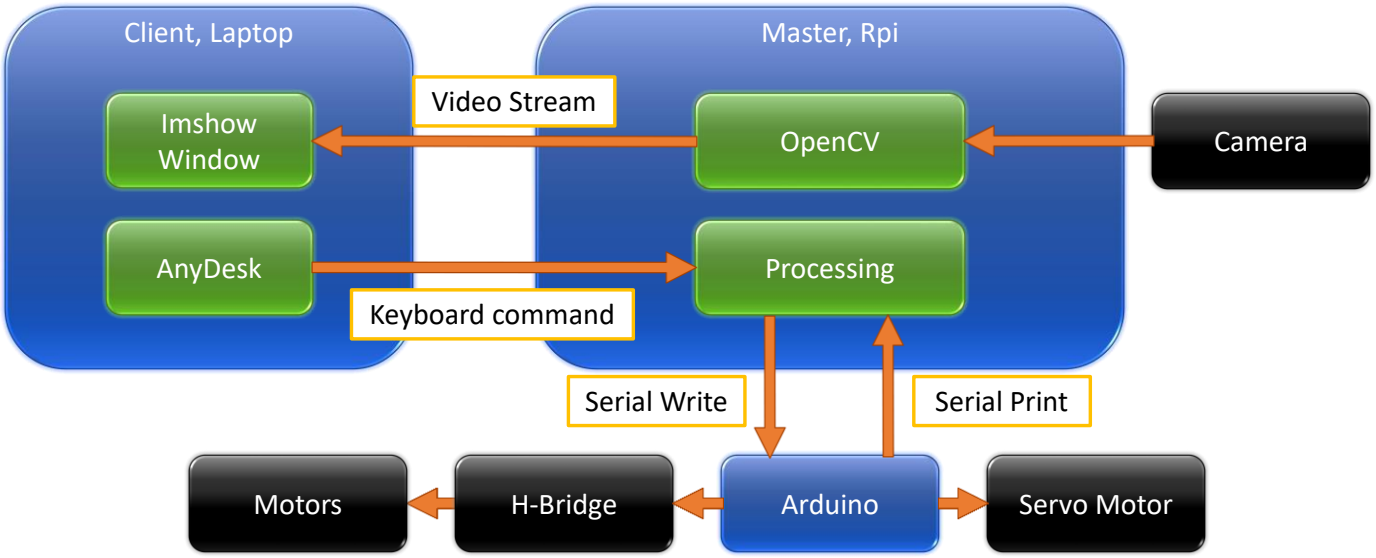



Signal circuit

Car Size




## Software Design




### Remote access

To control RaspberryPi, we access it remotely through ***AnyDesk***. AnyDesk has some useful and common functions including

(1)Remote desktop

(2)file transport

(3)Wake on LAN, noted that AnyDesk is access through WAN (wide area network)

### Video Stream

Using SSH and “Forward X11” argument to pass the imshow window to the client, but the delay still gets larger as the window size gets larger, so we use a small window to reduce the delay.

The delay is about 1 sec.

## Keyboard command

**Processing IDE**

Since the Arduino Arduino IDE cannot read command from serial monitor without pressing “Send” button, we found a Java-based IDE called “Processing” to directly write into the serial.

Forward: w

Forward (High speed): 2

Backward: x

Stop: s

On point left turn: a

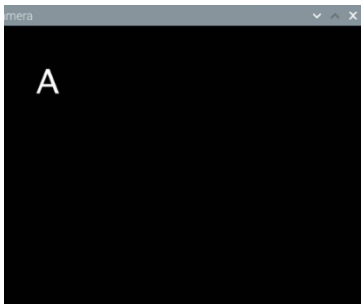
On point right turn: d

High speed curved left turn: q

High speed curved right turn: e

Gripper open: l

Gripper close: o




## Arduino Programming

After receiving keyboard command from Processing IDE, we wrote Arduino program to connect serial port to drivers/motors (H-Bridge and Servo motor). We then design several control strategies to send corresponding PWM signal to the connected drivers/motors.

Arduino reads the cmds from the serial and perform actions

Forward: 170, 140

Forward (High speed): 220, 160

Backward: -170, -140

Stop: 0, 0

On point left turn: -150, 150

On point right turn: 150, -150

High speed curved left turn: 100, 255

High speed curved right turn: 255, 100

Gripper open: 60

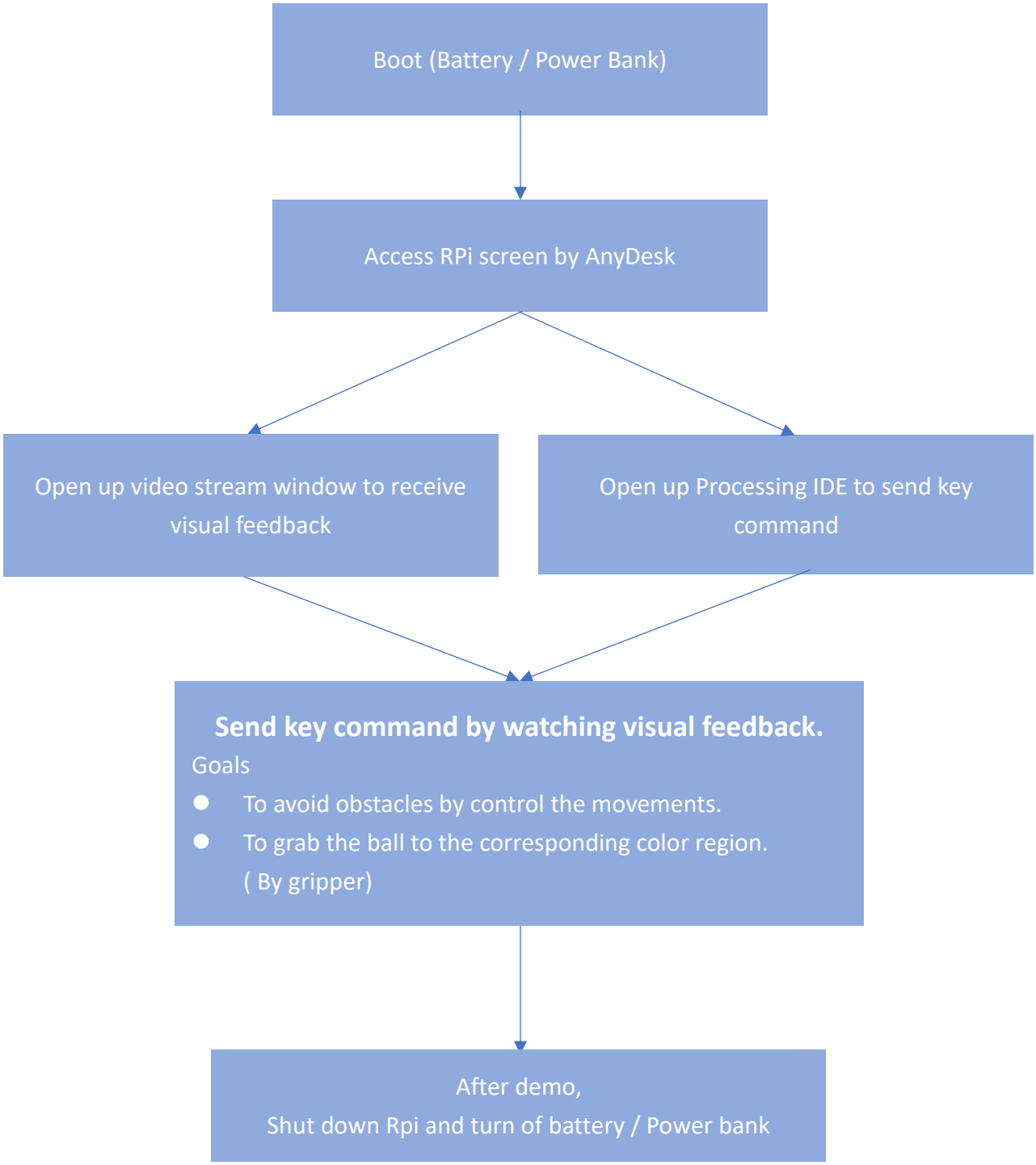
Gripper close: 0


```
void loop()
{
  digitalWrite(en_A, HIGH);
  digitalWrite(en_B, HIGH);

  if (Serial.available() > 0)
  {
    char serialread = Serial.read();
    switch(serialread)
    {
      case 'w':
      analogWrite(Motorforward_A, 170);
      analogWrite(Motorbackward_A, 0);
      analogWrite(Motorforward_B, 140);
      analogWrite(Motorbackward_B, 0);
      break;

      case '2':
      analogWrite(Motorforward_A, 220);
      analogWrite(Motorbackward_A, 0);
      analogWrite(Motorforward_B, 160);
      analogWrite(Motorbackward_B, 0);
      break;
```

# Operation system instructions

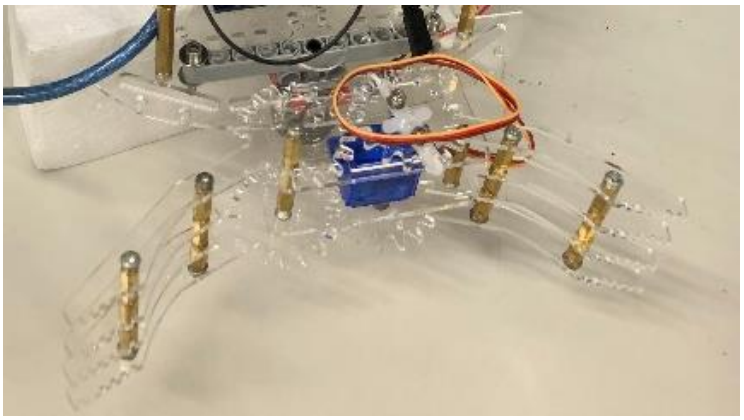


# Methods and Problem solved

## Gripper design

### Problem

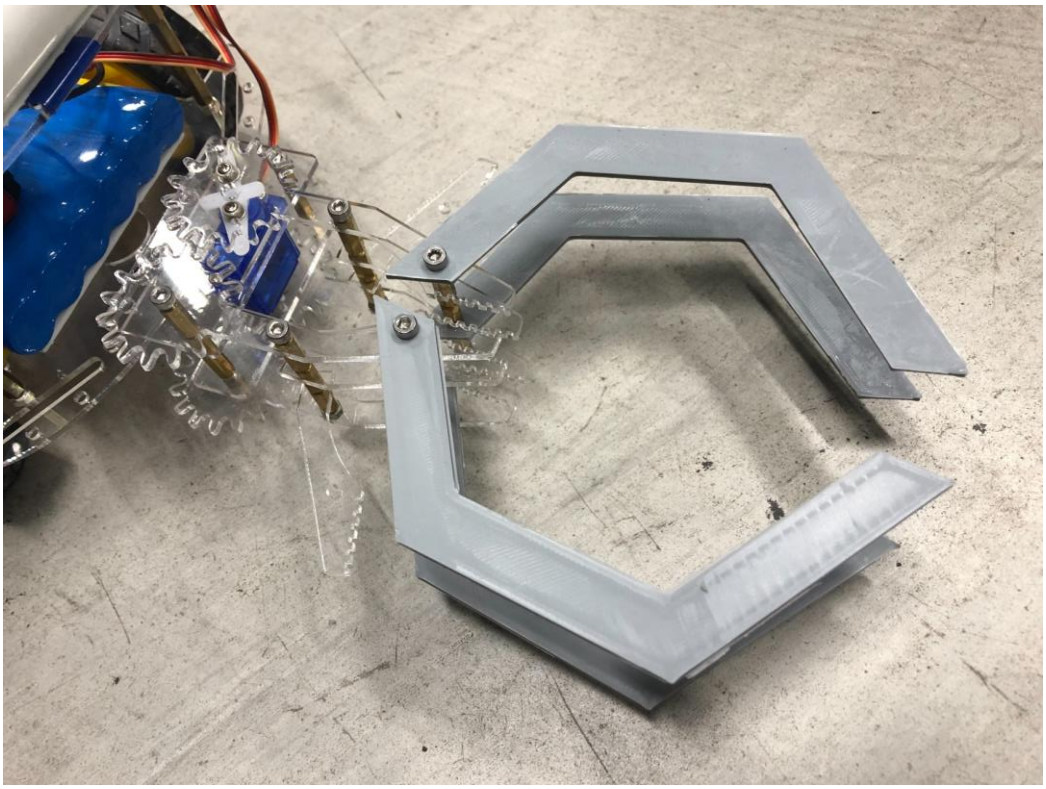
Ideally, when the ball enters the middle of the gripper accurately and the visual feedback works without delay; the strategy will work perfectly. However, **with a short delay**, we sometimes missed it and it went even farther. Therefore, we change our gripper design just in case we fail.



Gripper version1

### Solving Strategy

To improve the previous problem, we design a longer gripper to extend the region. Therefore, the strategy we implement previously remains the same. However, the previous problem will be solved. After the ball enters the middle of the gripper, it is hard for it to slip out again since the region of the gripper is extended.



Gripper version 2

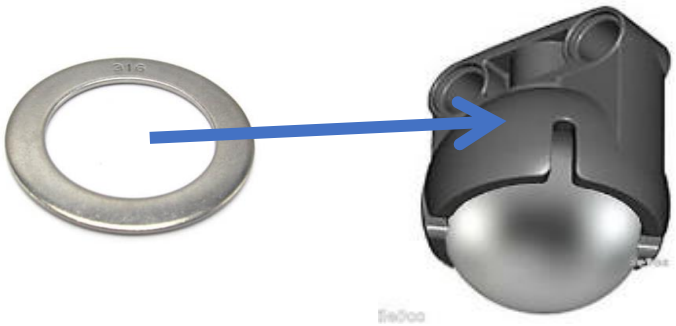
## Caster Wheel

Problem

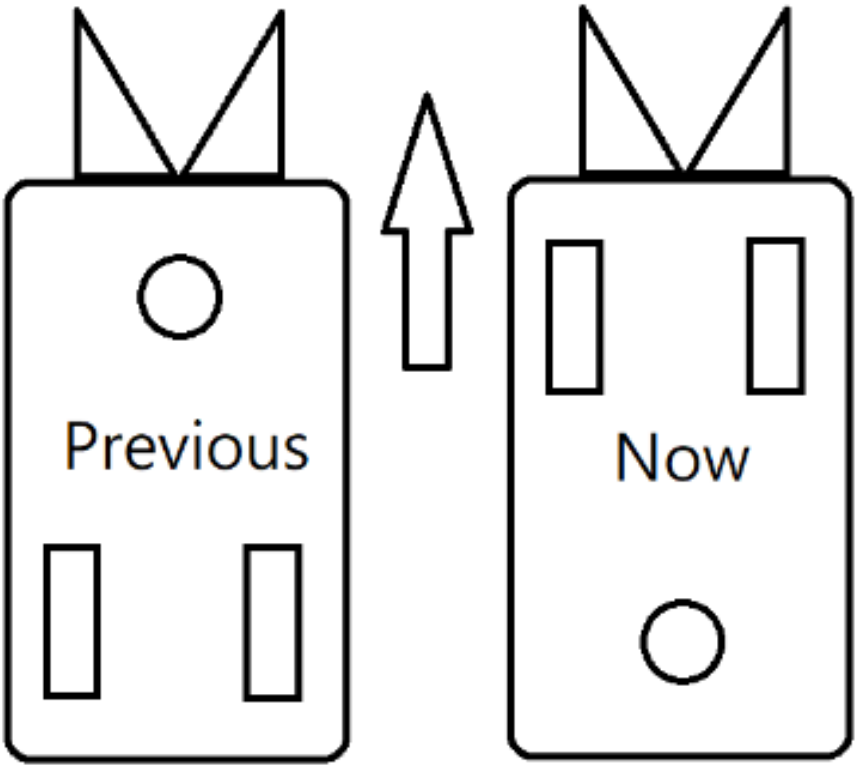
The ball caster would stuck into the crevices of the tiles, and the gap of the ball and the holder is too large—both problems would cause bumping, making the car hard to control

Solving strategy

We use a washer to fill the leeway in the ball caster, which reduce the car to bump seriously.



We rotate the car direction by 180, let the driven wheel at front. It makes the caster wheel easier to get out of the crevices.




## Reading command from serial

Problem

Arduino IDE cannot read command from serial monitor without pressing “Send” button.

Solving strategy

We use another Java-based IDE called “Processing” to directly write into the serial.

## Visual feedback latency

### Problem

We combine the visual feedback and the command window for Arduino together, hoping to give the user a better experience. Yet, the latency of visual feedback viewed from Anydesk is large (we view the window of Processing via Anydesk), and it makes the control of vehicle hard.

### Solving strategy

We use ssh protocol to access Rpi, and running a OpenCV code in python to show the image. Then, we turn the argument "X11Forwarding" in /etc/ssh/sshd_config of Rpi to "yes" to pass the imshow image window to the client. In the terminal, make sure the –X argument is added (eg. ssh -X pi@192.168.225.52).
Ref: https://ostechnix.com/how-to-configure-x11-forwarding-using-ssh-in-linux/

## Power circuit stability

### Problem

At first, we use a buck converter turn 8.5V to 5V to power up Rpi, but the stability of the circuit is poor because the battery also has to power the motors—sometimes Rpi would lag seriously.

### Solving strategy

We added a power bank to power up Rpi separately from the motors, which not only made the Rpi run smoothly, but also gave H-bridge more power to drive the motors.

## Stalled situation

### Problem

Because the chassis of our car is for four wheels, when we modified it to three wheels (one caster two active), the moment arm of the caster to the centroid of the car is large. Thus, the friction force exerted on the caster wheel would cause a large opposite torque against turning direction. Besides, the caster wheel would often get stuck at the crevices of the tiles. As a result, the on-point turn for robot would be difficult.

### Solving strategy

We add a strong forward command to the car, and two curve-turn command in case of the stalled situation. The robot can escape from the crevices after the strong command.

## Camera Position

Problem

At first, we put the WebCam in the front region of the car.
However, we found out that the view is too narrow which we cannot observe the surroundings easily.

Solving strategy

We change the camera position to the tail of the car and adjust the position angle so that we can have a wider view and also see the gripper to ensure we have grabbed the correct ball.

## Servo.h disable certain PWM pins

Problem

When we first connect the servo motor to the Arduino, we connected them through Arduino pin9, pin10 which are supposed to be able to send PWM signal.
However, we found out that the servo motor doesn't work.

Solving strategy

After searching solutions on the internet, we found out that pin9 and pin10 are not able to send PWM signals after connecting servo motor with Arduino.
Therefore, we change the pin position to pin3 and pin11.

## Rpi placement

Problem

After we finished placing all the hardware devices, we found out the position of RaspberryPi is inappropriate that it is hard for us to connect it to screen by HDMI.

Solving strategy

We rearrange the position and move RaspberryPi a little forward so that we can easily connect it to screen by HDMI without being stuck.

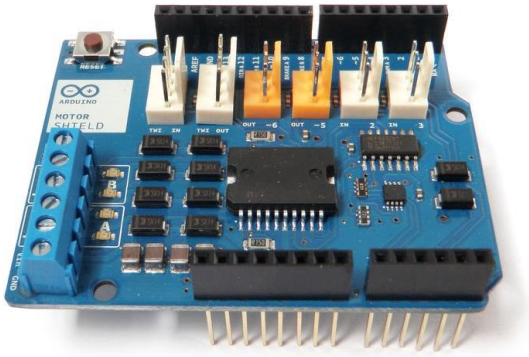
### Wire circuit routing

Problem

- The 5V and GND port in Arduino is not enough for our use.
- The wire circuit is messy, and the wheel will sometimes be interrupted by wire circuits.

Solving strategy

- We imply a Arduino shield so that we may have enough port to use.



Arduino shield

- We entangled and reorganized the wire circuit to avoid interrupting the wheels.

## External link

Code: https://github.com/liver121888/NTUBME-2021-MechatronicsIV-FinalProject
Video: https://youtu.be/2J82phaPbww