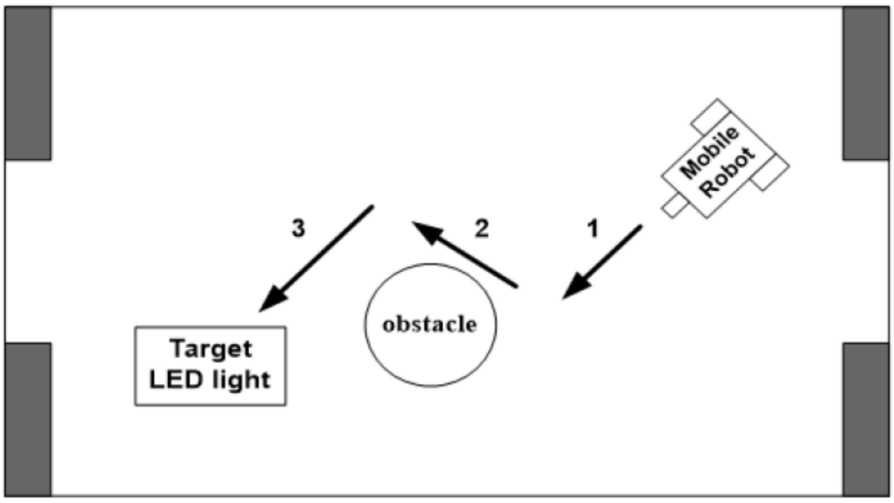
Checkpoint #3 Report

[EECN30169] Mobile Robot 2022

Student ID:310611008 **Name:** 張祐誠 **Date: Nov 11, 2022**

1. Purpose:

The purpose of this checkpoint required us to:
1. Configure the touch sensor for mobile robot to react when hitting the obstacle or the wall.
2. Configure the light sensor for mobile to detect the LED light object.
3. The mobile robot should find the LED light object automatically while avoiding the object in the arena.
4. The schematic diagram was shown in the following figure. The figure credits to TA's slide.



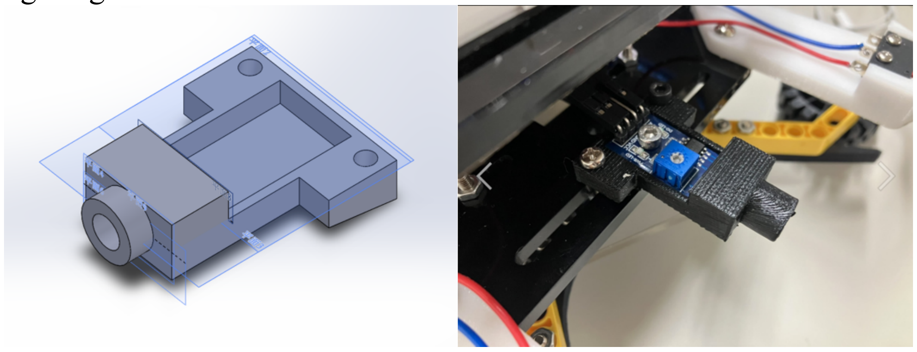
2. Description of Design:

A. Design and implement of sensors:

Inspired by the advantage of 3D-printer, we designed the components and printed all by ourselves.

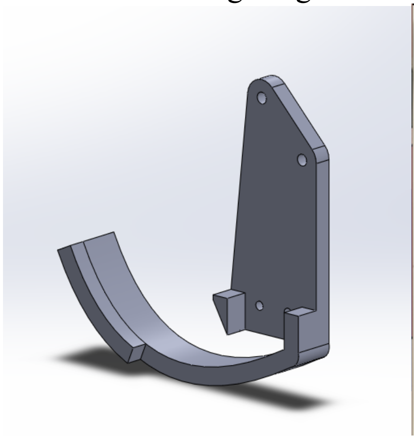
1. Light sensor:

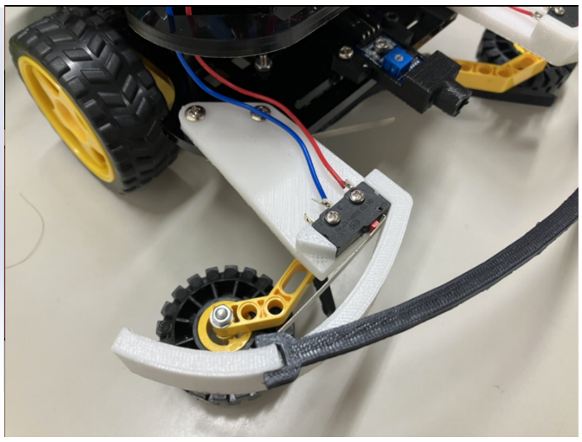
The design figure of light sensor is illustrated in the left figure. Then it was printed by 3D-printer with PLA material. The installation was shown in the right figure.



2. Touch sensor:

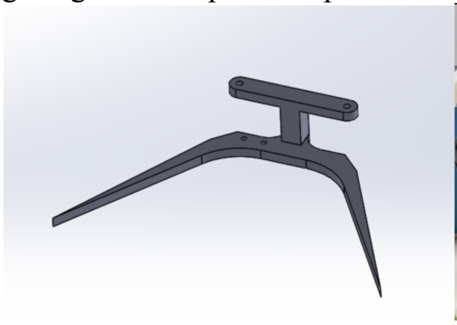
The very special point is that we choose TPU material to build this element. The characteristic of this material makes the component bendable and flexible. The design is shown in the left figure and the installation is shown in the right figure.

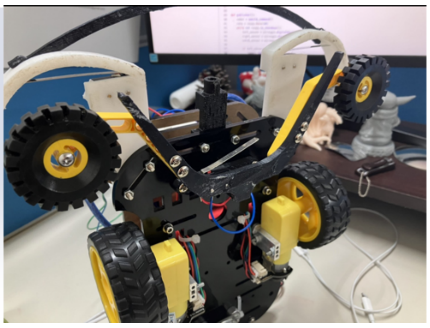




3. Gripper:

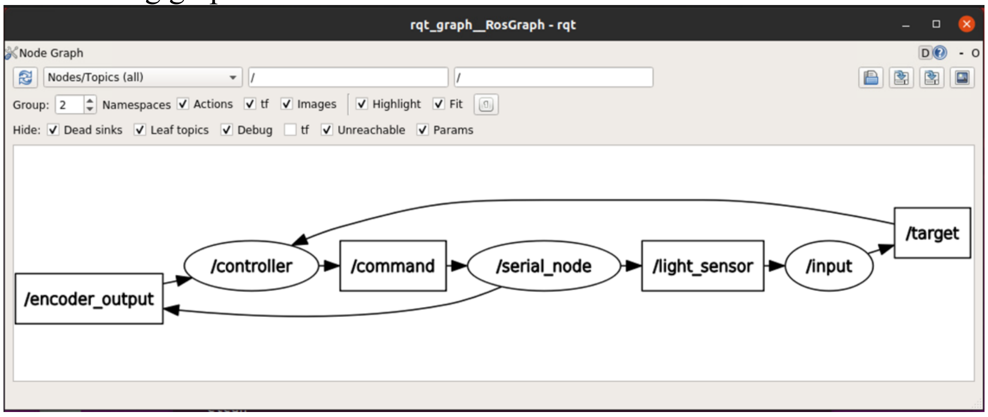
The design is shown in the left figure and the installation is shown in the right figure. This part was printed with PLA material.





B. Control strategy and searching process

1. The very first different we changed from previous checkpoint is that we wrote a new node to control the motor speed. Taking out the controller from Arduino to the Raspberry pi reduce the compute loading of Arduino and make the signal publishing much faster. The control process is shown in the following graph.



rqt_graph__RosGraph - rqt

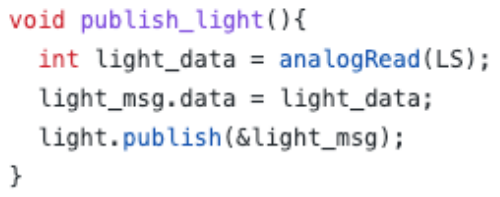
Node Graph

Nodes/Topics (all) | / | /

Group: 2 | ✓ Namespaces | ✓ Actions | ✓ tf | ✓ Images | ✓ Highlight | ✓ Fit

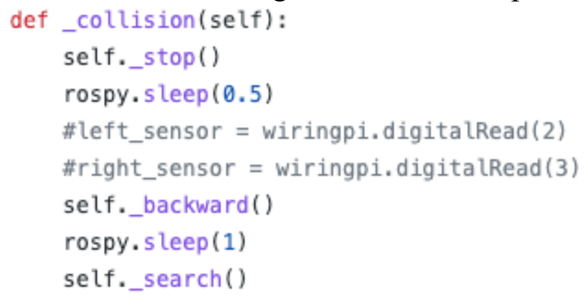
Hide: ✓ Dead sinks | ✓ Leaf topics | ✓ Debug | ☐ tf | ✓ Unreachable | ✓ Params

2. We found out that Raspberry Pi GPIO can only read digital signal. To find the precise position of light object, we need the brightness value, the analog signal. Thanks to the previous adaption, we are able to connect the light sensor on the Arduino board and publish the analog value to the topic.



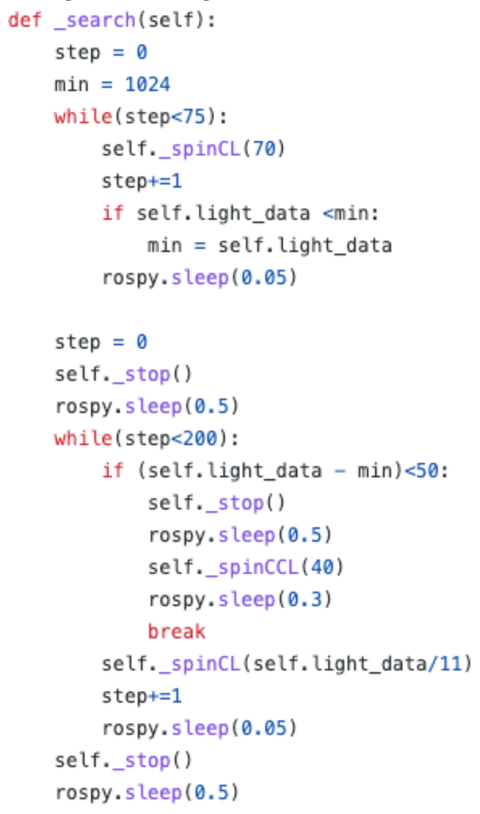
```
void publish_light(){
  int light_data = analogRead(LS);
  light_msg.data = light_data;
  light.publish(&light_msg);
}
```

3. The touch sensor was connected to the pin of Raspberry Pi and read by wiring pi library. After touch sensor signal is triggered, the mobile robot will move backward then turn for other direction. In this checkpoint, we let the robot search for the light sensor after impacting.



```
def _collision(self):
    self._stop()
    rospy.sleep(0.5)
    #left_sensor = wiringpi.digitalRead(2)
    #right_sensor = wiringpi.digitalRead(3)
    self._backward()
    rospy.sleep(1)
    self._search()
```

4. In the searching process, the robot will first spin for a full 360 degree and record the minimum light value. Then the minimum value is recognized as the light source. The robots will then turn again until the light sensor data close enough to the light source.



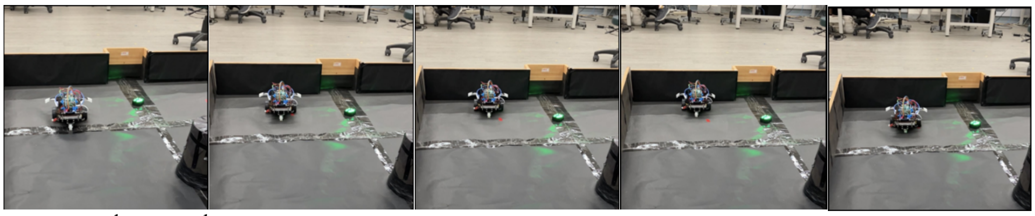
```
def _search(self):
    step = 0
    min = 1024
    while(step<75):
        self._spinCL(70)
        step+=1
        if self.light_data <min:
            min = self.light_data
        rospy.sleep(0.05)

    step = 0
    self._stop()
    rospy.sleep(0.5)
    while(step<200):
        if (self.light_data - min)<50:
            self._stop()
            rospy.sleep(0.5)
            self._spinCCL(40)
            rospy.sleep(0.3)
            break
        self._spinCL(self.light_data/11)
        step+=1
        rospy.sleep(0.05)
    self._stop()
    rospy.sleep(0.5)
```

5. Lastly, when the catch sensor detects the target be gripped, the robot will stop immediately. Although writing an extra function to stop the robot seems to be unnecessary, this function is preserved for the next checkpoint to execute the next action.

```
def _catch(self):
    self._stop()
```

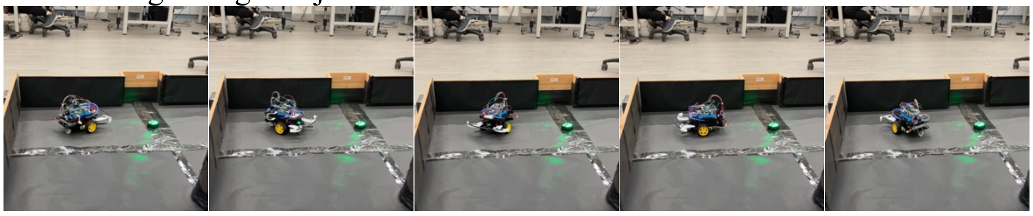
3. Result

1. Obstacle detection:



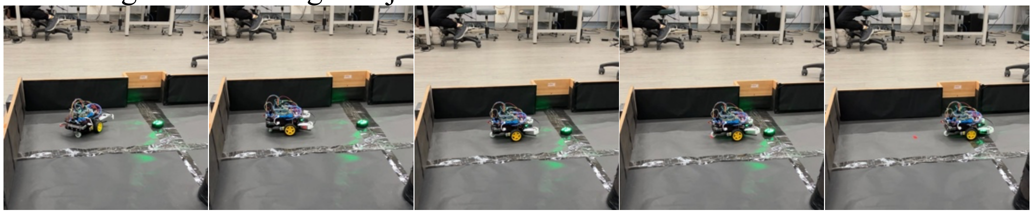
In the 3rd and 4th frame cans easily observe the bending of impact sensor component. After detection the robot moved backward, which can be told from the direction of caster wheel.

2. Searching for light object:



After the collision, the robot turned around to search for the minimum light sensor value. The motion can be told from the caster wheel.

3. Moving toward the light object:



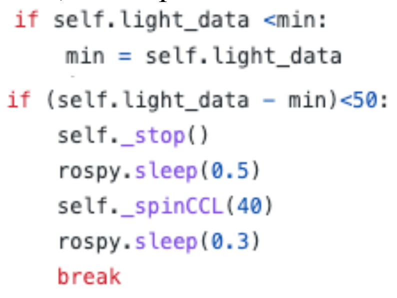
Then the robot will search for the value obtain in the previous step, then move forward until the light object is grabbed. The robot stop in the last frame and the whole task is finished.

4. In the demo, our team finished the task in 17.36 second.

4. Discussion

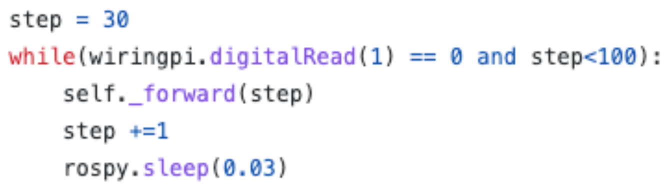
We make several efforts on this checkpoint:

1. We applied soft threshold on light detection. As describe in the Design paragraph, every time when the robot is searching for the light source, it will record the minimum value. We found out that the value can have huge different when the robot is closer or further form the light. Another problem we discovered during testing is that the robot will turn much more than we expected because of the robot inertia. To avoid this problem, we make a soft limit to the stop condition, that is, if the sensor value is small enough, 50 to the minimum in this case, the stop command will execute.



```
if self.light_data <min:
    min = self.light_data

if (self.light_data - min)<50:
    self._stop()
    rospy.sleep(0.5)
    self._spinCCL(40)
    rospy.sleep(0.3)
    break
```

2. We have noticed that the wheel often slips when activate from static. After several debugging, it is the controller contributed the over shoot that could cause one of the wheels spined slightly faster than the other. Despite that the steady state error is reduced, the slide of the wheel could ultimately cause the robot head for wrong direction and fail the task. Therefore, we modified the step input to ramp input while start from static.



```
step = 30
while(wiringpi.digitalRead(1) == 0 and step<100):
    self._forward(step)
    step +=1
    rospy.sleep(0.03)
```

3. After the demo, we discover that the strategy is important to the time performance. The strategy we choose is relative conservative, that is, we search the full 360 degree to ensure the position of the light source then spin again to reach the position. Some other team came out with the idea that the robot will move forward as long as the light source is detected. This strategy reduced the time to finished the task. In the next checkpoint, we might modified the strategy to make our robot execute the task more efficient while remaining the high successful rate of completing the task.