Checkpoint #4 Report

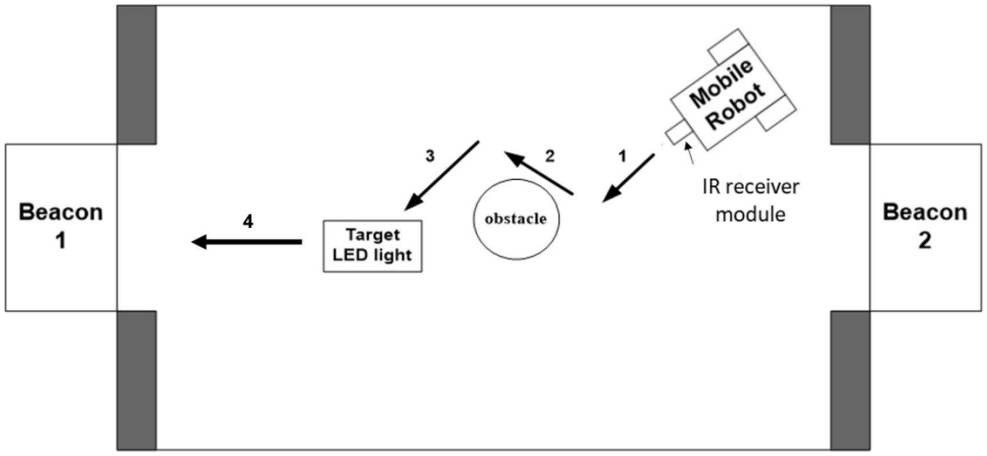
[EECN30169] Mobile Robot 2022

Student ID:310611008 **Name:** 張祐誠 **Date: Dec 6th, 2022**

1. Purpose:

The purpose of this checkpoint required us to:

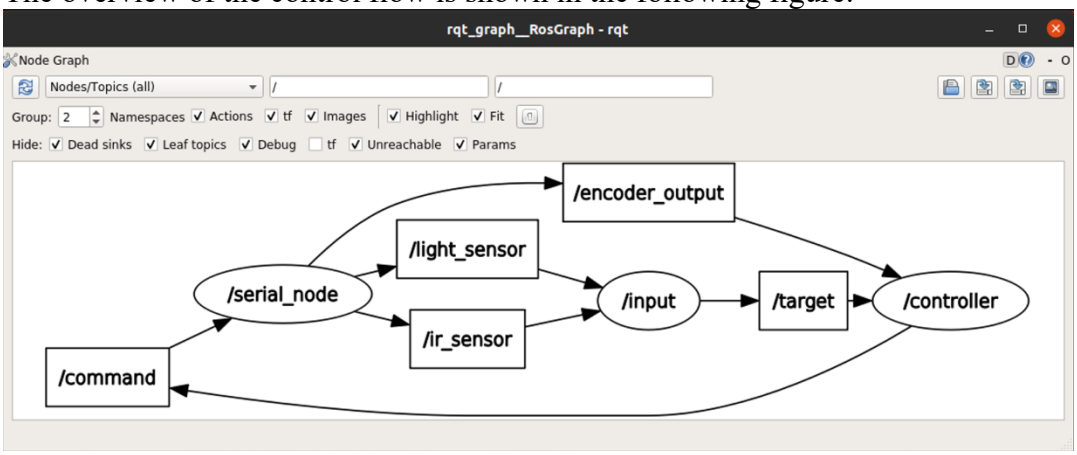
1. Configure the IR sensor for mobile robot to find the door after catching the light ball.
2. The mobile robot should find the LED light object automatically and take the light ball to the door while avoiding the object in the arena.
3. The schematic diagram was shown in the following figure. The figure credits to TA's slide.



2. Description of Design:

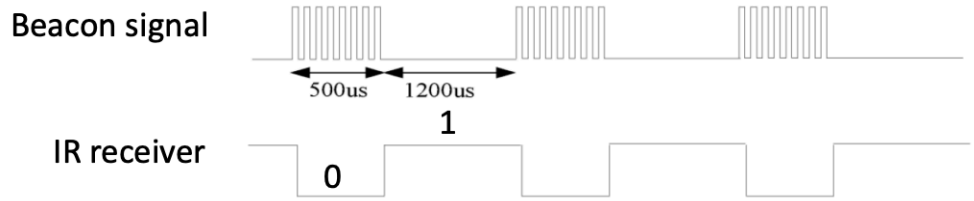
A. Control strategy and searching process

The overview of the control flow is shown in the following figure:



The serial node represents the Arduino board and the input node as well as controller node were executed in the Raspberry Pi processor. Our main configuration is to let the Arduino board handle all of the sensor and PWM signal output. On the other hand, the control command and the searching strategy were implemented in the Rpi processor.

The light sensor and the encoder were implemented in the previous checkpoint and the ir sensor was implemented in this checkpoint. The beacon emits the ir signal and the sensor response is shown as follow figure:
(Figure credits to TA's slide)



For the door 1 and door 2, there were different ratio for each. The way we read the door ratio was done by read the ir sensor output every millisecond in 0.1 second. Then we divided the amount of low output with total amount of we read the ir sensor. By the ratio we can make the robot identify which of the two door is the target. The reading signal is realized by the following scripts:

```
int count_0 = 0;
int count_1 = 0;
int cnt = 0;
float read_ir(){
  int val;
  float rate;
  cnt++;
  val = digitalRead(IR);
  if(val) count_1++;
  if(!val) count_0++;
  rate = count_0*100/(count_0+count_1);
  return rate;
}
```

When it comes to the finding the target door, the searching process is triggered after the light ball be caught by the robot. The finding door procedure is simply completed by the following scripts:

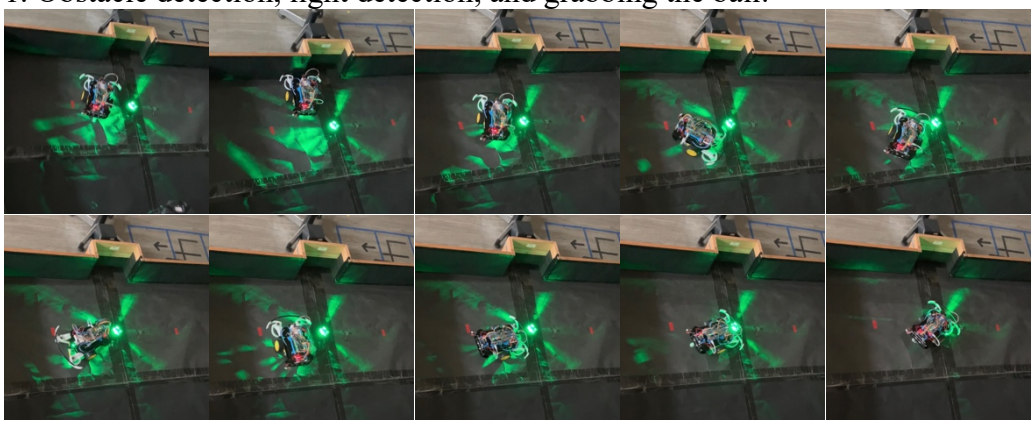
```
def _findDoor(self):
    step = 0
    while(step<70 and self.ir_data < 10):
        self._spinCCL()
        step+=1
        rospy.sleep(0.05)
    self._spinCL()
    rospy.sleep(0.2)
    self._stop()
```

As long as the target ir signal, the target door, the robot will head for the target direction. And, to make sure the light ball is pushing into the door firmly, we make the robot to adjust its facing direction after reaching the location of the door. That is, when the left-touch-sensor is triggered, the robot will tend to spin counter-clockwise while the right one does the opposite. This adjustment will make sure the robot facing direct to the door and consequence the push the ball into the deep place of the door. After the both side of touch sensor was triggered simultaneously, the overall task was then claimed to be complete. The realization was shown in the following code:

```
while(wiringpi.digitalRead(2)==0 or wiringpi.digitalRead(3) ==0):
    if (wiringpi.digitalRead(2) ==0 and wiringpi.digitalRead(3)==0):
        self.target_speed = 80
        self._forward()
    elif(wiringpi.digitalRead(2)==0):
        self._spinCCL()
    else:
        self._spinCL()
self.catch_flag =0
self._backward()
rospy.sleep(1)
self._stop()
rospy.sleep(5)
```

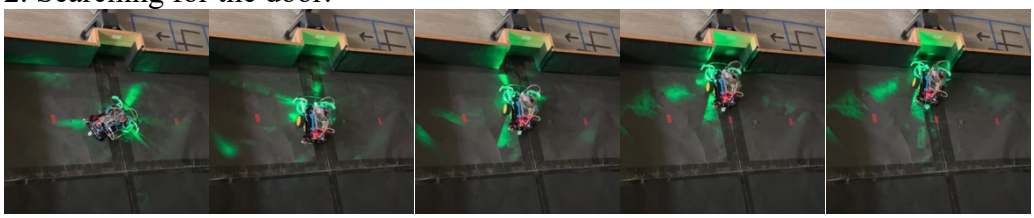
3. Result

1. Obstacle detection, light detection, and grabbing the ball:



These tasks were completed in the previous checkpoint. The frames above showed that out robot was well functional in these procedures. For the details, please refer to the previous reports.

2. Searching for the door:



After catching the light ball, the robot will turn around until the target door is detected by the ir sensor. Then the robot will take the light ball straight forward to the door.

3. Adjust the robot posture to push the ball inside the door:

After either one of the touch sensors is triggered, The robot will then slightly spin to adjust its facing direction to ensure the ball is pushed in the door correctly

4. In the demo, our team finished the task in 18.36 second.

4. Discussion

The biggest challenge for us is to deal with the ir sensor data publishing. The reason that it is difficult is the time duration of every cycle of the beacon. To sample the full duty cycle of the beacon signal, it requires the processor to read the signal for more than 0.1 second. However, the setting of the robot to process the sensor signal at 20Hz, which means that the ir signal could jam the overall sensor's system. Another disadvantage for decreasing the process rate is that the robot will react to the sensor much slower, and this would make the robot seems very dumb. Therefore, we try several solutions:

1. For the very first attempt, we constructed the full function by using for loop. And this is result of causing the jam of the signal processing. The result of this method is that every signal was read under 10 Hz.

2. Then we thought for using Arduino port as input and read for the PWM signal. When the beacon was detected, the Arduino can sense the signal rapidly enough and identify the two different beacon signal. However, when no beacon was detected, the reading speed will sharply drop to about 1 Hz, and, the jam of the signal become even more severe.

3. Next we consider of using the GPIO on the Raspberry pi. That is, taking the advantage of ROS. We wrote a new node to sense the ir signal. The realization is shown in the following scripts:

```
class ir_sensor():
    def __init__(self):
        self.pub = rospy.Publisher('door', Float64, queue_size = 1)
        self.cnt = 0
        self.cnt_1 = 0
        self.cnt_0 = 0
    def read_door(self):
        val = wiringpi.digitalRead(4)
        self.cnt+=1
        if val == 1:
            self.cnt_1+=1
        else:
            self.cnt_0+=1
        if self.cnt ==120:
            self.pub.publish(self.cnt_0*100/(self.cnt_0+self.cnt_1))
            self.cnt = 0
            self.cnt_1 = 0
            self.cnt_0 = 0
def pin_setup():
    wiringpi.wiringPiSetup()
    wiringpi.pinMode(4, 0)

def publisher():
    dctor = ir_sensor()
    rate = rospy.Rate(1000)
    while not rospy.is_shutdown():
        dctor.read_door()
        rate.sleep()
```

After launching the program, everything seems to work perfectly. However, out of nowhere, we found out that the sensing distance was so short that the robot can only sense the signal in front of the emitter for no longer than 10 cm. Still now, we don't know what actually lead to this error. We have confirmed that the signal voltage became much weaker when it connect to the GPIO port on the Raspberry pi.

4. The final solution we choose is to return to the Arduino port. We control the output signal by every millisecond rather than pause for every 50 millisecond, the way we did in previous checkpoint.

```
void loop(){
  //forward(0, 0);
  cnt++;
  ir_msg.data = read_ir();
  if(cnt%50==0){
  publish_encoder(duration_L, duration_R);
  publish_light();
  duration_R = 0;
  duration_L = 0;
  nh.spinOnce();

  }
  if(cnt%100==0){
     publish_ir();
     cnt = 0;
     count_0 = 0;
     count_1 = 0;
  }
  delay(1);
}
```

By this solution, we enable the ir data be published at 10Hz, while the publishing of the other data can remain at around 20Hz.