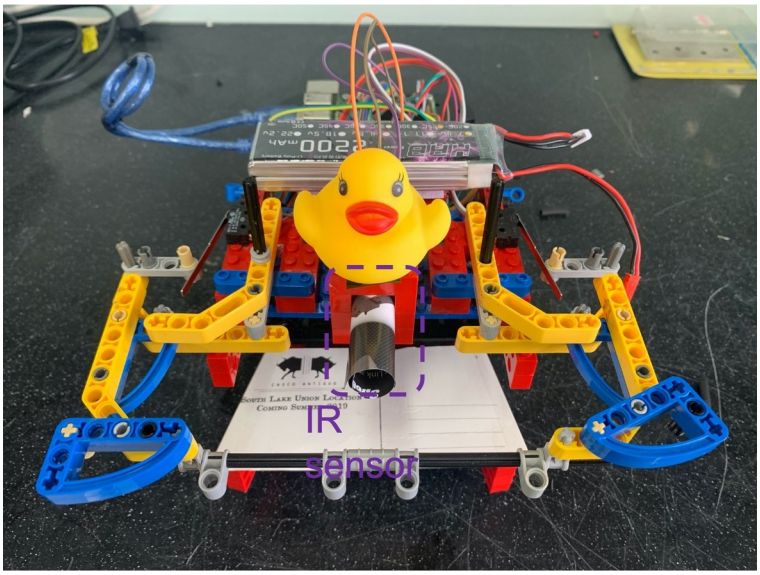
Checkpoint4 Report

一、大綱

延續 checkpoint3 基礎，新增 IR 接收器，以及控制程式中，有限狀態機中的相關 state。

二、IR 接收器配線與通訊設定



IR 硬體配置

IR 接收器的訊號線接到 Arduino 中的 A1 腳位，Arduino 端程式在 loop 中不斷讀值並 publish 到 ROS topic 上。在 RPI 端程式透過 ROS 接收，每 subscribe 50 次計算一次平均，已平均值來判斷 IR 訊號的 pulse proportion。

```
96      pinMode(A0, INPUT);//analog input
97      nh.advertise(pub_led);
98      pinMode(A1, INPUT);//analog input
99      nh.advertise(pub_IR);
100
101
102
103     Serial.print("INIT DONE");
104 }
105
106
107
108 void loop()
109 {
110
111         led.data = analogRead(A0);
112         pub_led.publish( &led );
113         IR.data = analogRead(A1);
114         pub_IR.publish( &IR );
```

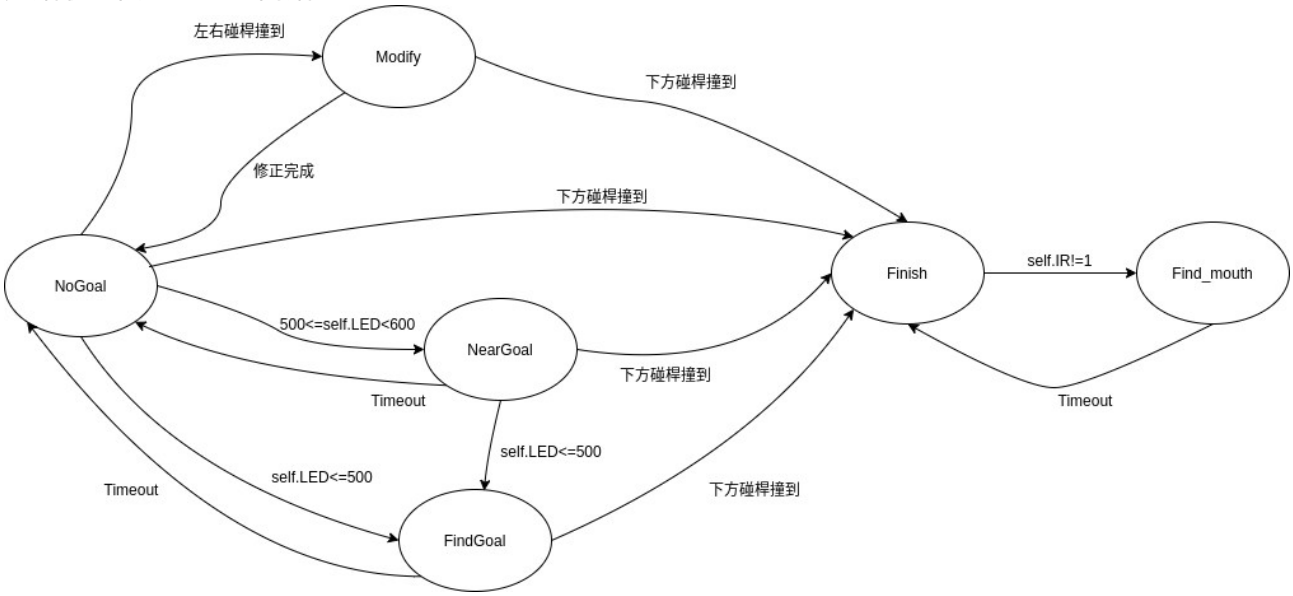
Arduino 端程式 IR 相關配置

```python
def cb_led(self,msg):
    self.led=msg.data

def cb_IR(self,msg):
    if msg.data>512:
        self.IR_1 = self.IR_1+1

    self.get_IR=self.get_IR+1
    if self.get_IR>=self.sample_times:
        self.IR = float(float(self.IR_1)/float(self.sample_times))
        self.get_IR=0
        self.IR_0=0
        self.IR_1=0
```

RPI 端程式 IR 相關配置

三、有限狀態機

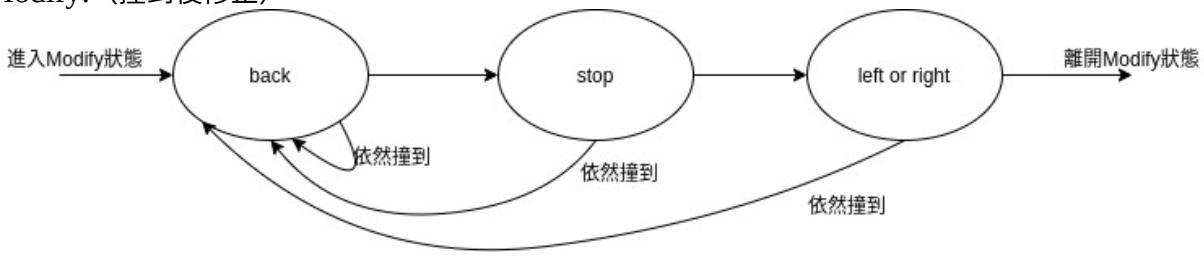
起始狀態由 NoGoal 開始



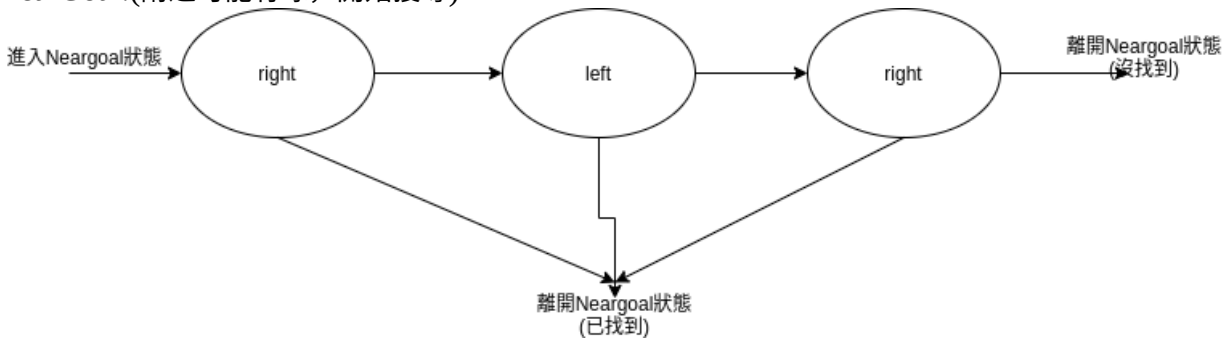
四、各 state 對應 active

1.NoGoal：直走

2.Modify:（撞到後修正）

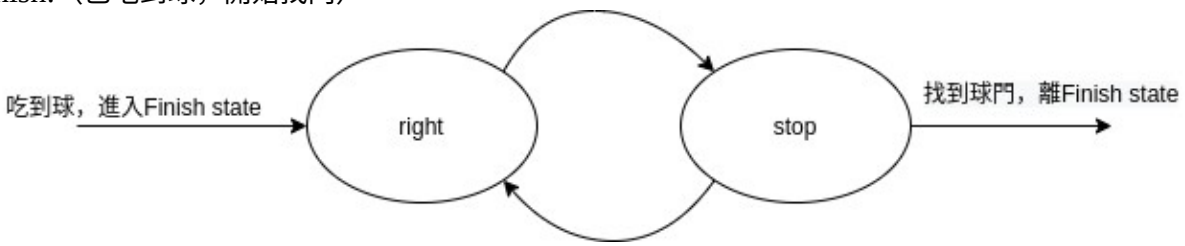


3.NearGoal:(附近可能有球，開始搜尋)



4.FindGoal:直走(確定前方有球)

5.Finish:（已吃到球，開始找門）



6.Find_mouth:直走(確定前方有門)

五、程式碼

A.RPI 端程式

```
#!/usr/bin/env python
import wiringpi
import time
import random
import rospy
import numpy as np
from std_msgs.msg import Int32

class Control(object):
    def __init__(self):
        wiringpi.wiringPiSetup()
        wiringpi.pinMode(7, 0)
        wiringpi.pinMode(8, 0)
        wiringpi.pinMode(9, 0)
        self.excute_time=0
        self.led=-1000
        self.last_led=-1000
        self.last_action=None
        self.now_action=None
        self.next_next_action=None
        self.state="NoGoal"
        self.time=0
        self.sub_led = rospy.Subscriber("pub_led", Int32, self.cb_led, queue_size=1)
        self.sub_IR = rospy.Subscriber("pub_IR", Int32, self.cb_IR, queue_size=1)
        self.pub_int = rospy.Publisher("array", Int32, queue_size=1)
        self.timer = rospy.Timer(rospy.Duration(0.1), self.control_loop)
        self.get_IR=0
        self.IR_1=0
        self.IR=0
        self.sample_times=50
        self.search_goal_mouth_points=11
        self.search_goal_mouth_array=np.zeros(self.search_goal_mouth_points)
        self.have_eat_ball=False
        print("init done")

    def cb_led(self,msg):
        self.led=msg.data

    def cb_IR(self,msg):
        if msg.data>512:
            self.IR_1 = self.IR_1+1
        self.get_IR=self.get_IR+1
        if self.get_IR>=self.sample_times:
            self.IR = float(float(self.IR_1)/float(self.sample_times))
            self.get_IR=0
            self.IR_0=0
            self.IR_1=0

    def motor_control(self):
        if self.now_action=="advance":
            self.pub_int.publish(1)
            # print('~~~pub advance~~~')
        if self.now_action=="right":
            self.pub_int.publish(2)
            # print('~~~pub right~~~')
        if self.now_action=="left":
```

```
            self.pub_int.publish(3)
            # print('~~~pub left~~~')
        if self.now_action=="back":
            self.pub_int.publish(4)
            # print('~~~pub back~~~')
        if self.now_action=="stop":
            self.pub_int.publish(5)
            # print('~~~pub stop~~~')

    def control_loop(self,event):
        #for bump sensor
        my_input7 = wiringpi.digitalRead(7)
        my_input8 = wiringpi.digitalRead(8)
        my_input9 = wiringpi.digitalRead(9)

        # if self.state!="Modify":
        if my_input7==0 and my_input8==0 and my_input9==1: #hit
            self.now_action="back"
            self.next_action="stop"
            self.next_next_action="left"
            self.state="Modify"
            self.time=13+4+3

        elif my_input7==1 and my_input8==0 and my_input9==1: #hit
            self.now_action="back"
            self.next_action="stop"
            self.next_next_action="left"
            self.state="Modify"
            self.time=13+4+3

        elif my_input7==0 and my_input8==1 and my_input9==1: #hit
            self.now_action="back"
            self.next_action="stop"
            self.next_next_action="right"
            self.state="Modify"
            self.time=13+4+3

        if my_input7==1 and my_input8==1 and my_input9==1: #free

            if not self.have_eat_ball:
            #for LED sensor
                if self.led<=500:
                    # print("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!FindGoal!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1!!!!")
                    self.state="FindGoal"
                    self.time=10

                elif self.led<=600 and self.led>500 and self.state!="Modify" and self.excute_time>200:
                    #
print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~NearGoal~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
")
                    if self.state!="NearGoal":
                        self.time=40
                        self.now_action="left"
                        self.next_action="right"
                    self.state="NearGoal"
```

```
        if my_input9==0 : #have catched the ball
            self.have_eat_ball=True
            if self.state!="Finish":
                self.time=30*10
                self.state="Finish"
                self.now_action="right"
                self.next_action="stop"



        #update
        if self.state=="Modify":
            if self.time==16:
                self.now_action=self.next_action
                self.next_action=self.next_next_action
                self.next_next_action=None
            elif self.time==13:
                self.now_action=self.next_action
                self.next_action=None
                self.next_next_action=None
            elif self.time<0:
                if self.have_eat_ball:
                    self.state="Finish"
                    self.time=0
                else:
                    self.state="NoGoal"
                    self.time=0


        if self.state=="NearGoal":
            if self.time==30:
                self.now_action=self.next_action
                self.next_action="left"
            if self.time==10:
                self.now_action=self.next_action
                self.next_action=None
            if self.time<0:
                self.state="NoGoal"
                self.time=0


        if self.state=="FindGoal":
            self.now_action="advance"
            if self.time<0:
                self.state="NoGoal"
                self.time=0

        if self.state=="NoGoal":
            self.now_action="advance"


        if self.state=="Finish":
            if self.time>0:
                if (self.time%2)>0:
                    self.now_action="right"
                else:
                    self.now_action="stop"
```

```
            if self.IR<0.95:
                self.state=="Find_mouth"
                self.now_action="advance"
                self.time=100

        if self.time==0:
            self.time=30*10
            self.state="Finish"
            self.now_action="right"

    if self.state=="Find_mouth":
        self.now_action="advance"
        if self.time<0:
            self.time=30*10
            self.state="Finish"
            self.now_action="right"

    self.excute_time=self.excute_time+1
    self.time=self.time-1
    self.last_led=self.led
    self.last_action=self.now_action
    self.motor_control()
    # print('sample_time=',self.get_IR,'IR=',self.IR)
    # print(self.led,my_input7,my_input8,my_input9,self.state,self.now_action,self.time)
    print(self.IR,self.led,self.state,self.now_action,self.time)



if __name__ == "__main__":
    rospy.init_node("Control")
    control=Control()
    rospy.spin()
```

B.Arduino 端程式

```
#include <PID_v1.h>
#include <ros.h>
#include <math.h>
#include <std_msgs/Int32.h>
const byte encoder0pinA = 2;//A pin -> the interrupt pin 0
const byte encoder0pinB = 12;//B pin -> the digital pin 3
int in1 =8; //The enabling of L298PDC motor driver board connection to the digital interface port 5
int in2 =9; //The enabling of L298PDC motor driver board connection to the digital interface port 4
int ena =5;
const byte encoder1pinA = 3;//A pin -> the interrupt pin 0
const byte encoder1pinB = 13;//B pin -> the digital pin 3
int in3 =10; //The enabling of L298PDC motor driver board connection to the digital interface port 5
int in4 =11; //The enabling of L298PDC motor driver board connection to the digital interface port 4
int enb =6;
byte encoder0PinALast;
byte encoder1PinALast;
double durationright,abs_durationright;//the number of the pulses
double durationleft,abs_durationleft;
boolean Directionright;//the rotation direction
boolean Directionleft;
```

```
boolean resultright;
boolean resultleft;
ros::NodeHandle nh;
double val_outputright;//Power supplied to the motor PWM value.
double val_outputleft;
double Setpointright=0;
double Setpointleft=0;
double Kp=0.6, Ki=5, Kd=0;

int MODE= -1;
int last_MODE;
int count=0;



PID rightPID(&abs_durationright, &val_outputright, &Setpointright, Kp, Ki, Kd, DIRECT);
PID leftPID(&abs_durationleft, &val_outputleft, &Setpointleft, Kp, Ki, Kd, DIRECT);


void num(const std_msgs::Int32& msg){

  if (msg.data==1){
    MODE=1;
    Setpointleft=255;
    Setpointright=255;
  }
  if (msg.data==2){
    MODE=2;
    Setpointleft=100;
    Setpointright=100;
  }
  if (msg.data==3){
    MODE=3;
    Setpointleft=100;
    Setpointright=100;
  }
  if (msg.data==4){
    MODE=4;
    Setpointleft=255;
    Setpointright=255;
  }
  if (msg.data==5){
    MODE=5;
    Setpointleft=0;
    Setpointright=0;
  }
}
ros::Subscriber<std_msgs::Int32> sub("array",&num);
std_msgs::Int32 led;
ros::Publisher pub_led("pub_led", &led);
std_msgs::Int32 IR;
ros::Publisher pub_IR("pub_IR", &IR);


void setup()
{
  Serial.begin(57600);//Initialize the serial port
  pinMode(in1, OUTPUT);   //L298P Control port settings DC motor driver board for the output mode
  pinMode(in2, OUTPUT);
  pinMode(ena, OUTPUT);
```

```
  pinMode(in3, OUTPUT);   //L298P Control port settings DC motor driver board for the output mode
  pinMode(in4, OUTPUT);
  pinMode(enb, OUTPUT);
  pinMode(encoder0pinA, INPUT);
  pinMode(encoder0pinB, INPUT);
  pinMode(encoder1pinA, INPUT);
  pinMode(encoder1pinB, INPUT);
  nh.initNode();
  nh.subscribe(sub);
  rightPID.SetMode(AUTOMATIC);//PID is set to automatic mode
  rightPID.SetSampleTime(100);//Set PID sampling frequency is 100ms
  leftPID.SetMode(AUTOMATIC);//PID is set to automatic mode
  leftPID.SetSampleTime(100);//S
  EncoderInit();//Initialize the module


  pinMode(A0, INPUT);//analog input
  nh.advertise(pub_led);
  pinMode(A1, INPUT);//analog input
  nh.advertise(pub_IR);



  Serial.print("INIT DONE");
}



void loop()
{
    led.data = analogRead(A0);
    pub_led.publish( &led );
    IR.data = analogRead(A1);
    pub_IR.publish( &IR );

    if(last_MODE!=MODE){
      if(MODE==1){advance();}//Motor Forward
      if(MODE==2){right();}
      if(MODE==3){left();}
      if(MODE==4){back();}
      if(MODE==5){Stop();}
    }
    abs_durationright=abs(durationright);
    resultright=rightPID.Compute();//PID conversion is complete and returns 1
    if(resultright)
    {
      durationright = 0; //Count clear, wait for the next count
    }

    abs_durationleft=abs(durationleft);
    resultleft=leftPID.Compute();//PID conversion is complete and returns 1
    if(resultleft)
    {
      durationleft = 0; //Count clear, wait for the next count
    }

    last_MODE=MODE;
    nh.spinOnce();
}
```

```
void EncoderInit()
{
  Directionright = true;//default -> Forward
  pinMode(encoder0pinB,INPUT);
  attachInterrupt(0, wheelSpeedright, CHANGE);
  Directionleft = true;//default -> Forward
  pinMode(encoder1pinB,INPUT);
  attachInterrupt(1, wheelSpeedleft, CHANGE);
}
void wheelSpeedright()
{
  int Lstate = digitalRead(encoder0pinA);
  if((encoder0PinALast == LOW) && Lstate==HIGH)
  {
    int val = digitalRead(encoder0pinB);
    if(val == LOW && Directionright)
    {
      Directionright = false; //Reverse
    }
    else if(val == HIGH && !Directionright)
    {
      Directionright = true;  //Forward
    }
  }
  encoder0PinALast = Lstate;
  if(!Directionright)  durationright++;
  else  durationright--;
}
void wheelSpeedleft()
{
  int Rstate = digitalRead(encoder1pinA);
  if((encoder1PinALast == LOW) && Rstate==HIGH)
  {
    int valR = digitalRead(encoder1pinB);
    if(valR == LOW && Directionleft)
    {
      Directionleft = false; //Reverse
    }
    else if(valR == HIGH && !Directionleft)
    {
      Directionleft = true;  //Forward
    }
  }
  encoder1PinALast = Rstate;
  if(!Directionleft)  durationleft++;
  else  durationleft--;
}

void advance()//Motor Forward
{
     digitalWrite(in1,HIGH);
     digitalWrite(in2,LOW);
     analogWrite(ena,val_outputright);
     digitalWrite(in3,HIGH);
     digitalWrite(in4,LOW);
     analogWrite(enb,val_outputleft);
}
```

```
void right()//Motor Forward
{
    digitalWrite(in1,HIGH);
    digitalWrite(in2,LOW);
    analogWrite(ena,val_outputright);
    digitalWrite(in3,LOW);
    digitalWrite(in4,HIGH);
    analogWrite(enb,val_outputleft);
}
void left()//Motor Forward
{
    digitalWrite(in1,LOW);
    digitalWrite(in2,HIGH);
    analogWrite(ena,val_outputright);
    digitalWrite(in3,HIGH);
    digitalWrite(in4,LOW);
    analogWrite(enb,val_outputleft);
}

void back()//Motor reverse
{
    digitalWrite(in1,LOW);
    digitalWrite(in2,HIGH);
    analogWrite(ena,val_outputright);
    digitalWrite(in3,LOW);
    digitalWrite(in4,HIGH);
    analogWrite(enb,val_outputleft);
}

void Stop()//Motor stops
{
    digitalWrite(ena, LOW);
    digitalWrite(enb, LOW);
}
```