Checkpoint # 2 Report

[ICN5406] Mobile Robot 2021

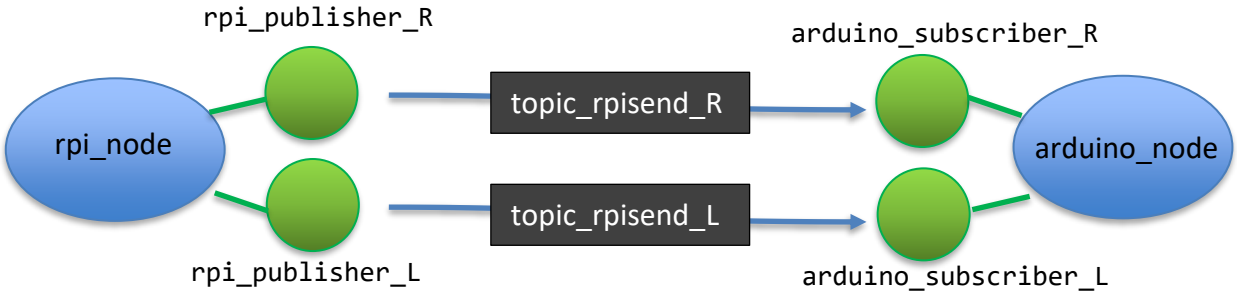
Student ID: 310515010 **Name:** 洪子茵 **Date:** 10/28

1. Purpose:

make sure you can control the motion of a basic DC motors by using PWM with Raspberry Pi and Arduino.

To control two motors with encoder signals.
• Move forward.
• Move backward.
• Turn right.
• Turn left.
• How straight the robot can move when moving forward.

2. Description of Design:



I designed two `publisher and subscriber for raspberry pi node and` Arduino node respectively.

```
user's right is 120
user's left is 120
user's right is -100
user's left is 50
user's right is 100
user's left is 200
user's right is 0
user's left is 0
user's right is 100
user's left is 100
user's right is -50
user's left is 50
user's right is 0
user's left is 0
```

So like the above picture, when the user type right wheel and left wheel PWM signal on the RPI command, the value of PWM will than send to Arduino, and Arduino will control the dc motors.

(The +/- of the PWM value we give means that we want the motor rotates clockwise or counterclockwise, and the absolute value of it, mean the speed of the motor we want it to rotate.

The code Arduino receive user input data from rpi:

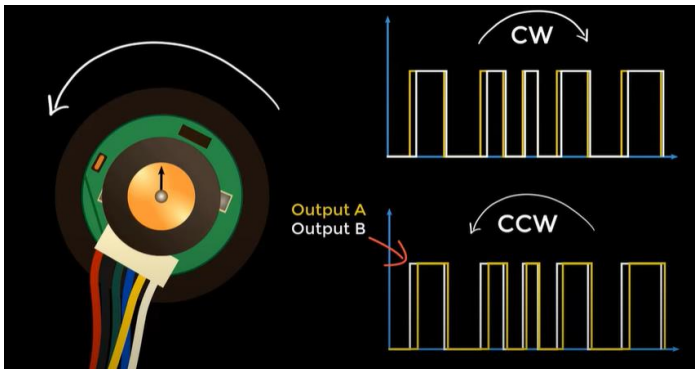
```
52  void messageCb_L(const std_msgs::Int32 &msg){
53    int L_pwm;
54    L_pwm = msg.data;
55
56    if (L_pwm < 0){
57      Direction_L = false;
58    }
59    else{
60      Direction_L = true;
61    }
62
63    L_pwm_val = abs(static_cast<double>(L_pwm));
64    //duration_L = 0;
65    flagL = true;
66  }
67
68  ros::Subscriber<std_msgs::Int32> R_arduino_sub("topic_rpisend_R",&messageCb_R);
69  ros::Subscriber<std_msgs::Int32> L_arduino_sub("topic_rpisend_L",&messageCb_L);
```

Encoder & Interrupt

Interrupts are useful for making things happen automatically in microcontroller programs and can help solve timing problems.

In this checkpoint, I use encoder and Interrupt to detect weather the motor rotates clockwise or counterclockwise, below is the principle:
An encoder works by observing the magnetic field created by a magnet attached to the motor shaft, as the motor rotates the encoder outputs will trigger periodically.

When the magnet spins clockwise, output A will(interrupt pin) trigger first, when it spins counterclockwise in the other hand, output B will(digital pin) trigger first.



So due to which output trigger first, we can get the direction of the motor rotation, then we can calculate how many pulse it occurs, to get the duration of each direction.
(duration is +,if motor's direction is clockwise(forward), vice versa. And the faster the motor speed, the greater the absolute value of duration.)

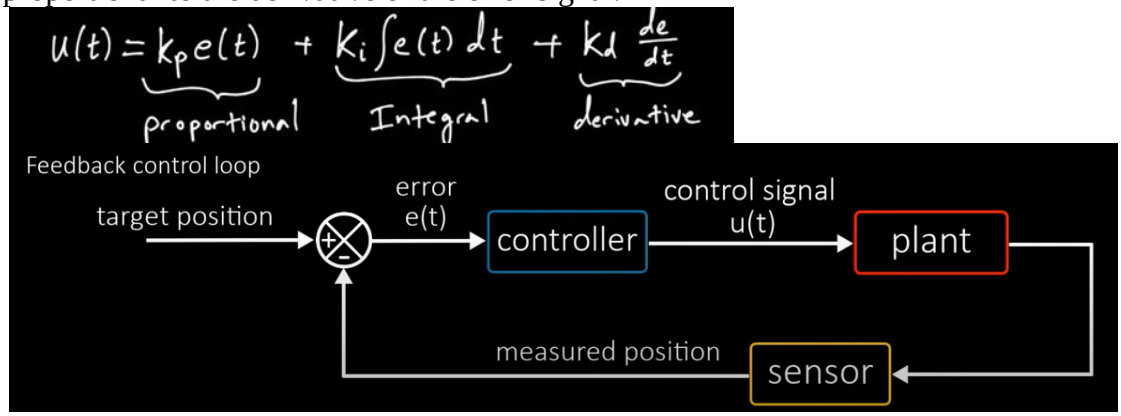
```
//Initialize the Encoder Interupt
attachInterrupt(0,MotorStateChanged_R, CHANGE);
attachInterrupt(1,MotorStateChanged_L, CHANGE);
```

```
181   void MotorStateChanged_L(){
182     int Lstate = digitalRead(ENCL_A);
183     if((encoder0PinALast_L == LOW) && Lstate==HIGH)
184     {
185       int val = digitalRead(ENCL_B);
186       if(val == LOW && Direction_L)
187       {
188         Direction_L = false; //Reverse
189       }
190       else if(val == HIGH && !Direction_L)
191       {
192         Direction_L = true;  //Forward
193       }
194     }
195     encoder0PinALast_L = Lstate;
196
197     if(!Direction_L)  duration_L++;
198     else  duration_L--;
199   }
```

PID controller

Beside encoder and interrupt, to make motor rotation more robust, we can add PID controller on it.

PID controller has three parameter: Kp, KI, Kd.
Kp proportional to error signal, KI proportional to integral of error signal and Kd proportional to the derivative of the error signal.



(I use package from Brett Beauregard for implement)
https://github.com/br3ttb/Arduino-PID-Library/

PID controller code

```
PID myPID_R(&abs_duration_R, &val_output_R, &R_pwm_val, Kp_R, Ki_R, Kd_R, DIRECT);
PID myPID_L(&abs_duration_L, &val_output_L, &L_pwm_val, Kp_L, Ki_L, Kd_L, DIRECT);
```

In void setup()

```
myPID_L.SetMode(AUTOMATIC);//PID is set to automatic mode
myPID_L.SetSampleTime(100);//Set PID sampling frequency is 100ms
myPID_L.SetOutputLimits(0, 255);
```

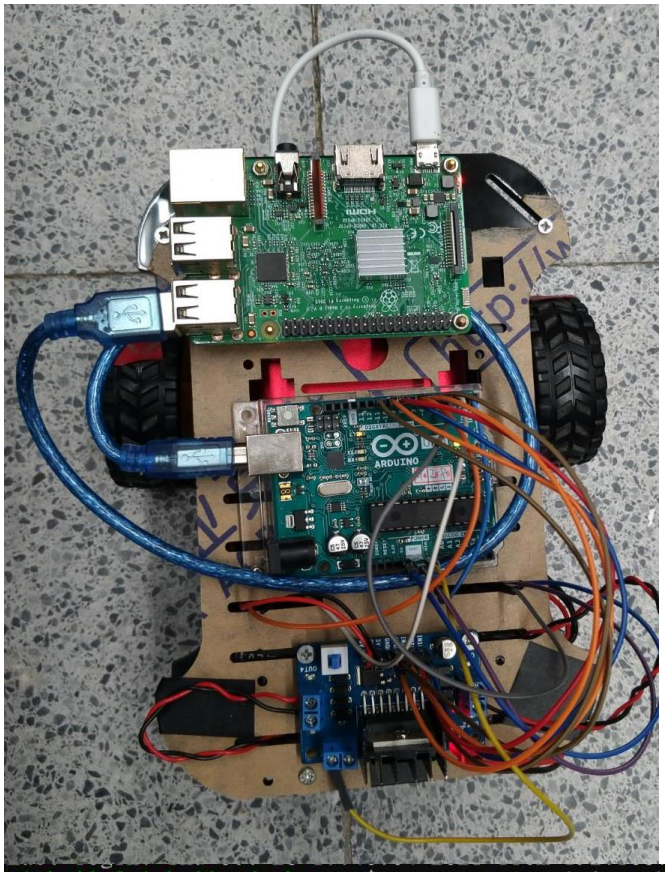
```
void loop()
{
    if(flagR && flagL){
      abs_duration_R=abs(duration_R);
      myPID_R.SetTunings(Kp_R, Ki_R, Kd_R);
      result_R = myPID_R.Compute();//PID conversion is complete and returns 1
      if (result_R){
        duration_R = 0;
      }
      abs_duration_L=abs(duration_L);
      myPID_L.SetTunings(Kp_L, Ki_L, Kd_L);
      result_L = myPID_L.Compute();//PID conversion is complete and returns 1
      if (result_L){
        duration_L = 0;
      }
      setMotor(Direction_R, val_output_R, PWM_R, IN1, IN2);
      setMotor(Direction_L, val_output_L, PWM_L, IN3, IN4);
    }
    nh.spinOnce();
}
```

DC motor

Code to control DC motor

```
// For one motor
void setMotor(boolean dir, double pwmVal, int pwm, int in1, int in2){
  analogWrite(pwm,pwmVal);
  if (pwmVal != 0){
    if(dir == true){  //CW
      digitalWrite(in1,HIGH);
      digitalWrite(in2,LOW);
    }
    else if(dir == false){  //CCW
      digitalWrite(in1,LOW);
      digitalWrite(in2,HIGH);
    }
  }
  else{
    digitalWrite(in1,LOW);
    digitalWrite(in2,LOW);
  }
}
```

3. Result



```
michelle@michelle-desktop:~$ rosrun rosserial_python serial_node.py /dev/ttyACM0
[INFO] [1635414469.349484]: ROS Serial Python Node
[INFO] [1635414469.385517]: Connecting to /dev/ttyACM0 at 57600 baud
[INFO] [1635414471.510869]: Requesting topics...
[INFO] [1635414471.578855]: Note: subscribe buffer size is 280 bytes
[INFO] [1635414471.589025]: Setup subscriber on topic_rpisend_R [std_msgs/Int32]
[INFO] [1635414471.612900]: Setup subscriber on topic_rpisend_L [std_msgs/Int32]
```

The result was quite fine, but when doing task 5, either I should make the motor rotate slower to make it walk straight , or because two motor have different horsepower, I should try to give it different rpm value to make the result better.

4. Discussion

***Some issue I occur during this checkpoint**

1. It you want to automatically connect Wi-Fi once raspberry boot, you should add a "**wpa_supplicant.conf"** file on "boot" drive.
2. Remember to common ground Arduino and L298n.
3. PID controller (Arduino library) kp ki kd meaning ,and how to adjust.
4. Topic, node, subscriber, publisher design
5. Use flag (flagR, flagL) to make sure Arduino already got, before doing rest of the work.

5. Reference

https://www.youtube.com/watch?v=dTGITLnYAY0

Micro DC Motor with Encoder

https://wiki.dfrobot.com/Micro_DC_Motor_with_EncoderSJ01_SKU__FIT0450

Digital Pins With Interrupts
https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/