Motor Control

PWM + PID

Ong Yong Qi [2102646]
Neo Jing Yi, Mandy [2102448]
Loh Jing Yi [2002605]

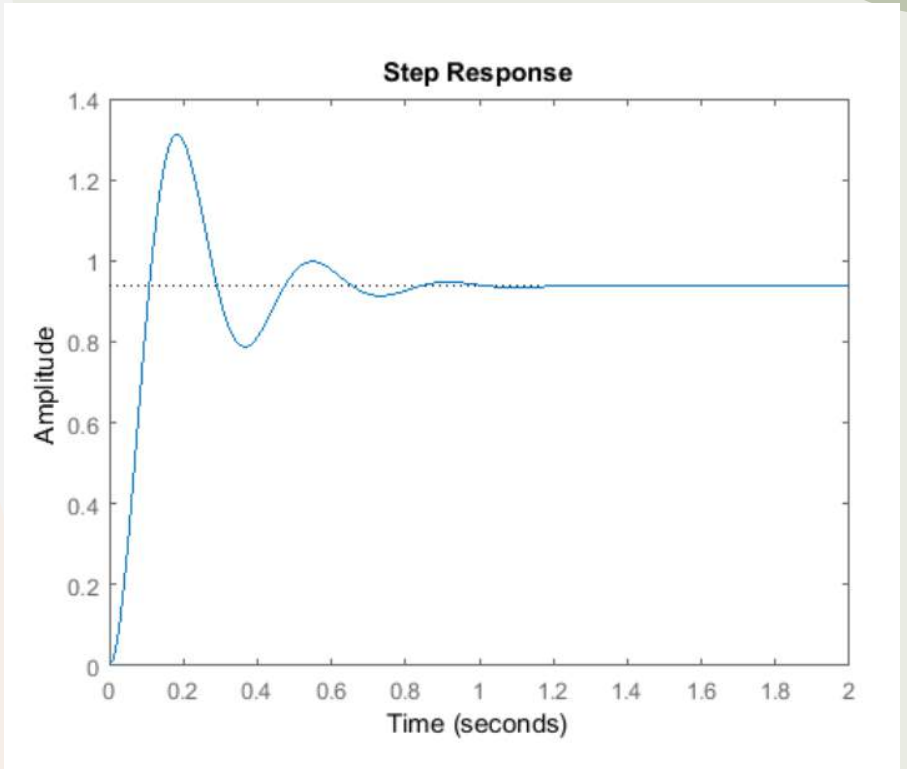
What we found out during our research.

Effects of P, I, and D

CL RESPONSE	RISE TIME	OVERSHOOT	SETTLING TIME	STEADY-STATE ERROR
Kp	Decrease	Increase	Small Change	Decrease
Ki	Decrease	Increase	Increase	Decrease
Kd	Small Change	Decrease	Decrease	No Change

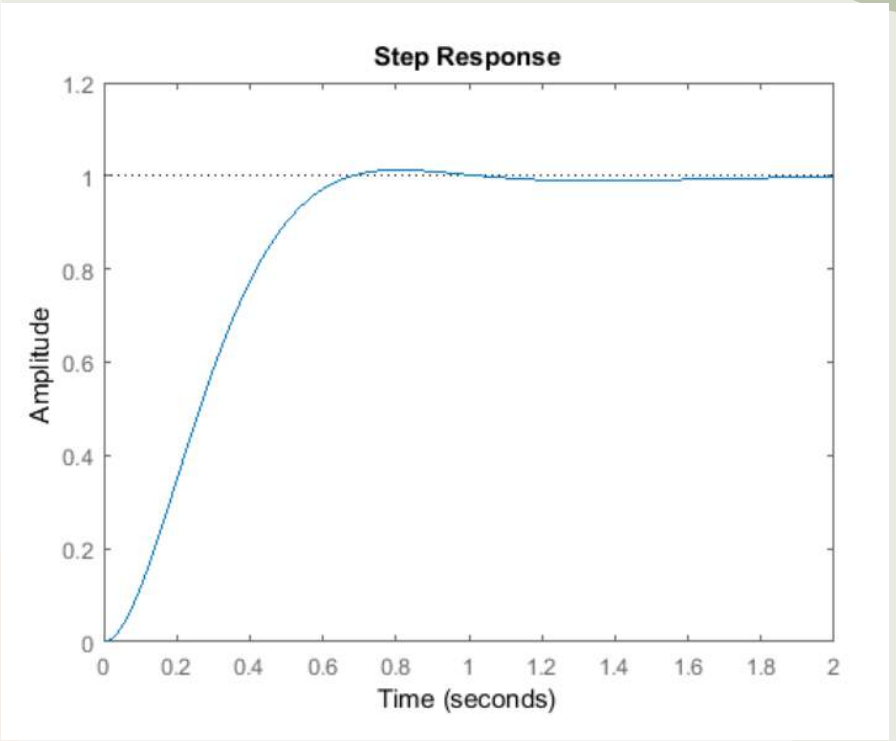
Proportional Control

- The plot on the left shows that the Kp Control:
  - decreases the rise time
  - increases the overshoot
  - increases the settling time
  - decreases the steady-state error



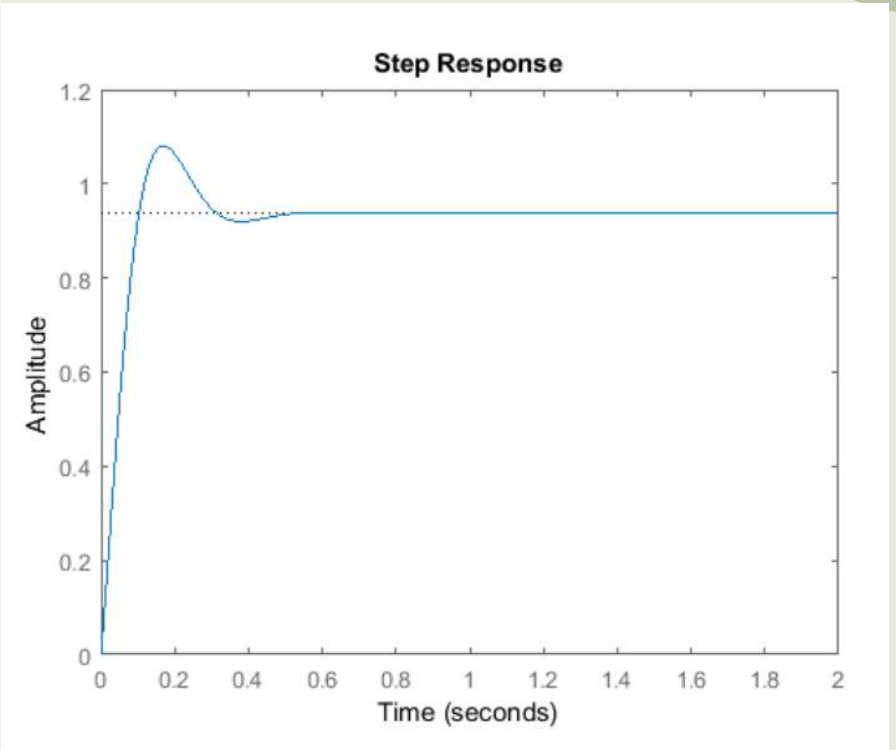
Proportional-Integral Control

- The plot on the left shows that the Ki Control:
  - decreases the rise time
  - increases the overshoot
  - has a minor decrease in the settling time
  - decreases the steady-state error



Proportional-Derivative Control

- The plot on the left shows that the Kd Control:
  - No effect on the rise time
  - decreases the overshoot
  - decreases the settling time
  - No effect on the steady-state error



How we came up with our Kp, Ki and Kd Control values?

Kp Control

- As we increased the value of the Kp Control, we realized that it has the effect of increasing the control signal for the same level of error.
- This will cause the controller to push harder, resulting in the closed-loop system to react more rapidly while increasing the overshoot at the same time.
- We continued playing around with the value of the Kp Control and realized that increasing the value can reduce the steady-state error.
- After going through numerous trial and error, together with the other Ki and Kd control, we fixed our Kp Control value to 49.

Kp Control

- After tuning the value of the Kp Control, we encountered a problem where the proportional gain affects the duty cycle in a way that, if the error value is really close to the desired notches value, the duty cycle will constantly increase until hitting the max (100%) duty cycle.
- We witness this effect as the Kp Control does reduce the error at first, but the wheel moves on to spin fast after hitting max (100%) duty cycle.
- We consulted with our professors, and they highlighted that we must adjust out Ki and Kd Control to prevent this from happening.

Ki Control

- We start to understand the various effects the different controllers have on correcting the error and begun playing with the Ki and Kd Control.
- We studied in detail and found out that tuning the Ki Control can help reduce the steady-state error.
- This prevents the integrator from building up and causing an increase in the control signal as it drives the error down.

Ki Control

- However, we did learn from the consultation with our professors that tuning the Ki Control can make the system slower as it takes a while for the integrator to build up.
- We eventually finalized our Ki Control value to 9, while adjusting the other control values.

Kd Control

- Moving on, we started including the Kd Control. We were unsure how it affects the correction of the error at first.
- But with more research, we found out that the Kd Control allows the system to anticipate the error.
- Although the Kd Control does not affect the steady-state error, it reduces the overshoot of the error.
- We decided to go with 0.003 for our Kd Control value.

Black Box Testing

Effects of P, I, and D

CL RESPONSE	RISE TIME	OVERSHOOT	SETTLING TIME	STEADY-STATE ERROR
Kp	Decrease	Increase	Small Change	Decrease
Ki	Decrease	Increase	Increase	Decrease
Kd	Small Change	Decrease	Decrease	No Change

Decision Table Testing

CAUSES		VALUES	1	2	3	4	5
C1	Kp	Y/N	Y	Y	Y	Y	N
C2	Ki	Y/N	Y	Y	N	N	-
C3	Kd	Y/N	Y	N	Y	N	-
E1	RISE TIME		Decrease	Decrease	Decrease	Decrease	-
E2	OVERSHOOT		Increase	Increase	Decrease	Increase	-
E3	SETTLING TIME		-	Increase	Decrease	-	-
E4	STEADY-STATE ERROR		Decrease	Decrease	Decrease	Decrease	-

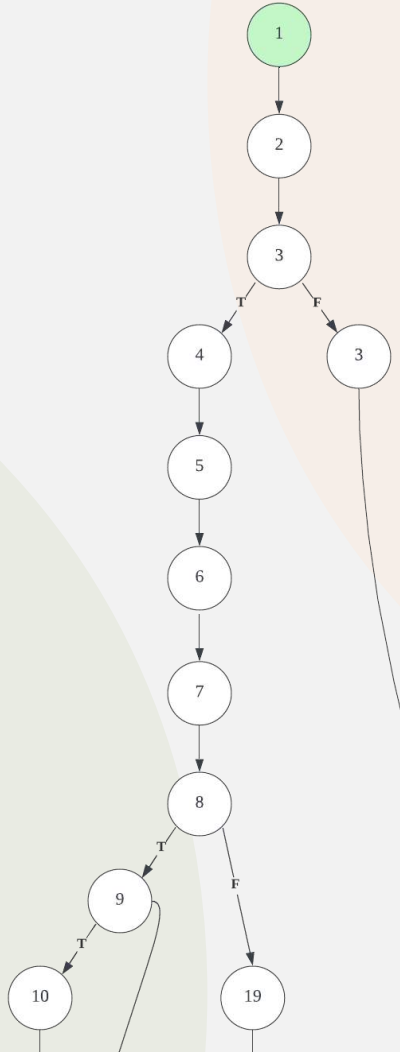
White Box Testing

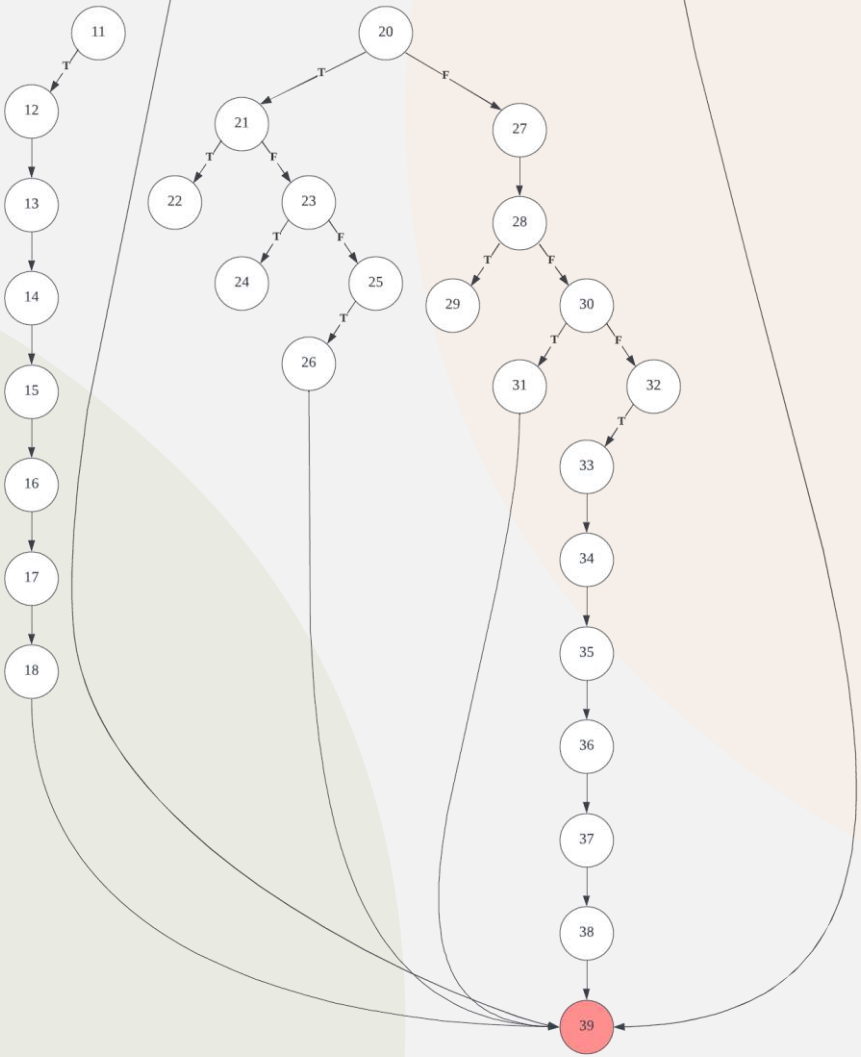
```
1. void TA1_0_IRQHandler(void)
   {
2.     SR04IntTimes++;

3.     if (SR04IntTimes >= 20000)
       {
4.             float leftControl = 0;
5.             float rightControl = 0;

6.             leftControl = getPIDOutputLeft();
7.             rightControl = getPIDOutputRight();

8.             if(turning){

9.                 if(startNotches == 0){
10.                    startNotches = rightNotchesDetected;
                   }

11.                 if(rightNotchesDetected >= startNotches+40){
12.                    turning = 0;
13.                    startNotches= 0;

14.                    GPIO_setOutputLowOnPin(GPIO_PORT_P4, GPIO_PIN4);
15.                    GPIO_setOutputHighOnPin(GPIO_PORT_P4, GPIO_PIN5);
16.                    GPIO_setOutputHighOnPin(GPIO_PORT_P4, GPIO_PIN0);
17.                    GPIO_setOutputLowOnPin(GPIO_PORT_P4, GPIO_PIN2);
18.                    stop();
                    }
               }
```

```
19. else if(forward){
20.                 if (leftControl != 0){

21.                     if(pwmConfig2.dutyCycle <= 10000 && pwmConfig2.dutyCycle >= 1000){
22.                         pwmConfig2.dutyCycle += leftControl;
                        }

23.                     else if(pwmConfig2.dutyCycle > 10000){
24.                         pwmConfig2.dutyCycle = 10000;
                        }

25.                     else if(pwmConfig2.dutyCycle < 1000)
26.                         pwmConfig2.dutyCycle = 1000;
                    }

27.                 if (rightControl != 0){

28.                    if(pwmConfig.dutyCycle <= 10000 && pwmConfig.dutyCycle >= 1000){
29.                        pwmConfig.dutyCycle += rightControl;
                       }

30.                    else if(pwmConfig.dutyCycle > 10000){
31.                        pwmConfig.dutyCycle = 10000;
                       }

32.                    else if(pwmConfig.dutyCycle < 1000)
33.                        pwmConfig.dutyCycle = 1000;
                    }
```

```
34.                 rightNotchesDetected = 0;
35.                 leftNotchesDetected = 0;
36.                 SR04IntTimes = 0;


37.                 Timer_A_generatePWM(TIMER_A0_BASE, &pwmConfig);
38.                 Timer_A_generatePWM(TIMER_A0_BASE, &pwmConfig2);
                }
        }

39.     Timer_A_clearCaptureCompareInterrupt(TIMER_A0_BASE, TIMER_A_CAPTURECOMPARE_REGISTER_0);
    }
```

Control-Flow Graph


1
2
3
T
F
4
3
5
6
7
8
T
F
9
T
10
19


11
T
12
13
14
15
16
17
18
20
T
F
21
27
T
F
22
23
28
T
F
T
F
24
25
29
30
T
26
T
F
31
32
T
33
34
35
36
37
38
39

Branch Coverage:
38 out of 44 total branches
= ~86.3 %

Robot Car Demo

Robot Car Turning at Various Angles

vimeo

The End