EE405A Electronic Design Lab – Robocam

Lab 3

20170275 박정웅 20170616 정희진

1. Purpose

- 3개의 바퀴와 2개의 LED가 있는 로봇의 회로를 구현한다.

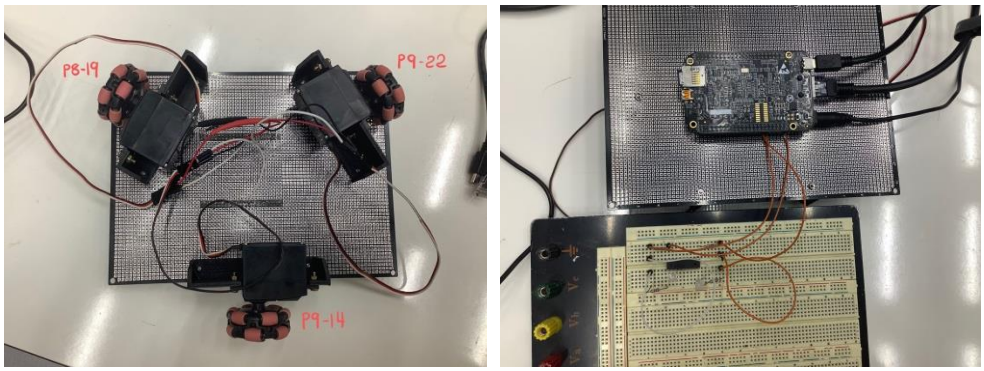
- shell script와 c파일을 컴파일한 실행 파일로 각각의 바퀴를 제어한다.

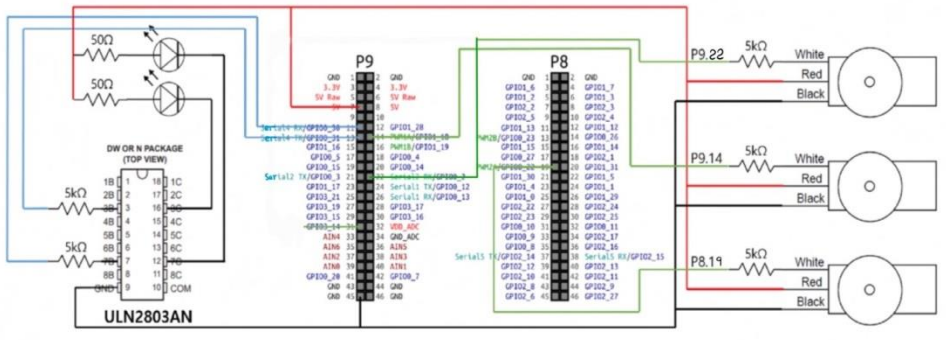
- 키보드 인풋으로 로봇을 제어한다.

2. Result

1) Problem 3A.

a) TMR circuit diagram.




위의 회로도처럼 세 바퀴는 각각 Beaglebone의 P9-14, P9-22, P8-19에 연결해 사용했고, 2개의 LED는 각각 P9-11, P9-13, LED의 GND는 P9-1에 연결해 사용했다.

2) Problem 3B.

```
root@beaglebone:/sys/devices/bone_capemgr.9# echo am33xx_pwm > ./slots
root@beaglebone:/sys/devices/bone_capemgr.9# ls
baseboard  driver  modalias  power  slot-4  slot-5  slot-7  slots  subsystem  uevent
root@beaglebone:/sys/devices/bone_capemgr.9# echo bone_pwm_P9_22 > ./slots
root@beaglebone:/sys/devices/bone_capemgr.9# echo bone_pwm_P9_14 > ./slots
root@beaglebone:/sys/devices/bone_capemgr.9# echo bone_pwm_P8_19 > ./slots
root@beaglebone:/sys/devices/bone_capemgr.9# ls ../ocp.3/pwm_test_P9_22.15/
driver  duty  modalias  period  polarity  power  run  subsystem  uevent
root@beaglebone:/sys/devices/bone_capemgr.9# ls ../ocp.3/pwm_test_P9_14.16/
driver  duty  modalias  period  polarity  power  run  subsystem  uevent
root@beaglebone:/sys/devices/bone_capemgr.9# ls ../ocp.3/pwm_test_P8_19.17/
driver  duty  modalias  period  polarity  power  run  subsystem  uevent
root@beaglebone:/sys/devices/bone_capemgr.9# cd ../ocp.3/pwm_test_P9_22.15/
root@beaglebone:/sys/devices/ocp.3/pwm_test_P9_22.15# ls
driver  duty  modalias  period  polarity  power  run  subsystem  uevent
```

실험을 하기 전 먼저 am33xx_pwm 모듈과 각각의 PWM 모듈을 slots에 넣어주고, 폴더를 확인하였다.

a) Captured Ubuntu terminal after testing.

```
root@beaglebone:/sys/devices/ocp.3/pwm_test_P9_22.15# echo 3000000 > period
root@beaglebone:/sys/devices/ocp.3/pwm_test_P9_22.15# echo 2000000 > duty
root@beaglebone:/sys/devices/ocp.3/pwm_test_P9_22.15# echo 1 > run
root@beaglebone:/sys/devices/ocp.3/pwm_test_P9_22.15# echo 0 > run
root@beaglebone:/sys/devices/ocp.3/pwm_test_P9_22.15# cd ..
root@beaglebone:/sys/devices/ocp.3# cd pwm_test_P9_14.16/
root@beaglebone:/sys/devices/ocp.3/pwm_test_P9_14.16# ls
driver  duty  modalias  period  polarity  power  run  subsystem  uevent
root@beaglebone:/sys/devices/ocp.3/pwm_test_P9_14.16# echo 3000000 > period
root@beaglebone:/sys/devices/ocp.3/pwm_test_P9_14.16# echo 2000000 > duty
root@beaglebone:/sys/devices/ocp.3/pwm_test_P9_14.16# echo 1 > run
root@beaglebone:/sys/devices/ocp.3/pwm_test_P9_14.16# echo 0 > run
root@beaglebone:/sys/devices/ocp.3/pwm_test_P9_14.16# cd ..
root@beaglebone:/sys/devices/ocp.3# cd pwm_test_P8_19.17/
root@beaglebone:/sys/devices/ocp.3/pwm_test_P8_19.17# ls
driver  duty  modalias  period  polarity  power  run  subsystem  uevent
root@beaglebone:/sys/devices/ocp.3/pwm_test_P8_19.17# echo 3000000 > period
root@beaglebone:/sys/devices/ocp.3/pwm_test_P8_19.17# echo 2000000 > duty
root@beaglebone:/sys/devices/ocp.3/pwm_test_P8_19.17# echo 1 < run
1
root@beaglebone:/sys/devices/ocp.3/pwm_test_P8_19.17# echo 1 > run
root@beaglebone:/sys/devices/ocp.3/pwm_test_P8_19.17# echo 0 > run
```

각각의 바퀴의 제어 파일(pwm_test_P9_22.15, pwm_test_P9_14.16, pwm_test_P8_19.17)로 가 period와 duty값을 넣어 주었다. 세 바퀴가 모두 잘 작동하는 것을 확인했다.

b) Value of the dead-band.

```
ol_PWM0A.sh one:/home/jungwungpark/lab3/3_MobileRobot/c_PWM_Servo_Shell# ./contro
Type in Duty:1475000
Type in Duty:1585000
Type in Duty:-1
root@beaglebone:/home/jungwungpark/lab3/3_MobileRobot/c_PWM_Servo_Shell#
```

Period에 3000000값을 사용하고, duty에 1400000에서 1600000 사이 값을 넣으면서 dead-band를 찾았다. 1400000에서 값을 조금씩 올렸더니 1475000에서 바퀴가 멈추었다. 또한, 1600000에서 조금씩 값을 내렸더니 1585000에서 바퀴가 멈추었다. 따라서 dead-band는 1475000에서 1585000 사이라고 할 수 있고 절댓값은 110000이다. Zero_duty는 1530000으로 계산되었다.

3) Problem 3C.

a) Shell scripts

A. Acquire_Triple_PWMs.sh

KLMS에 올라온 template 코드와 변함이 없다.

```
echo am33xx_pwm > /sys/devices/bone_capemgr.9/slots
ls /sys/devices/bone_capemgr.9

echo bone_pwm_P9_22 > /sys/devices/bone_capemgr.9/slots
echo bone_pwm_P9_14 > /sys/devices/bone_capemgr.9/slots
echo bone_pwm_P8_19 > /sys/devices/bone_capemgr.9/slots

echo "Checking for Individual PWM"
ls /sys/devices/ocp.3/pwm_test*
ls /sys/devices/ocp.3/pwm_test_P9_22.*/
ls   /sys/devices/ocp.3/pwm_test_P9_14.*/
ls /sys/devices/ocp.3/pwm_test_P8_19.*/
echo "PWM Acquired"
```

B. control_PWM0A.sh

KLMS에 올라온 template 코드와 변함이 없다.

```
echo 0 > /sys/devices/ocp.3/pwm_test_P9_22.*/run
echo 0 > /sys/devices/ocp.3/pwm_test_P9_22.*/polarity
echo 3000000 > /sys/devices/ocp.3/pwm_test_P9_22.*/period
echo 2000000 > /sys/devices/ocp.3/pwm_test_P9_22.*/duty
while true; do
    read -p "Type in Duty:" duty;
    if [ $duty -lt 0 ] ; then
        break
    else
        echo 1 > /sys/devices/ocp.3/pwm_test_P9_22.*/run
        echo $duty > /sys/devices/ocp.3/pwm_test_P9_22.*/duty

    fi
done
echo 0 > /sys/devices/ocp.3/pwm_test_P9_22.*/run
```

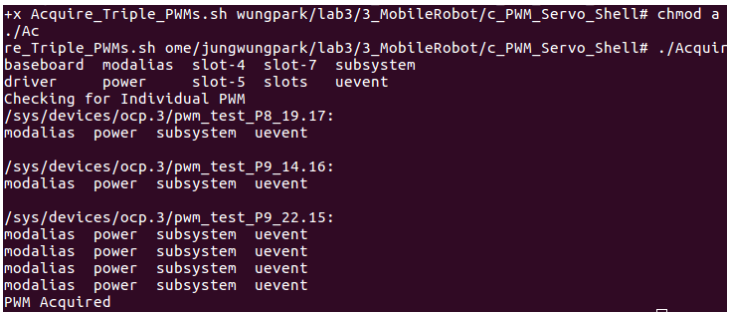
C. Control_Triple_PWMs.sh

KLMS에 올라온 template 코드와 변함이 없다.

```
while true; do
    read -p "Type in PWM and Duty:" pwm duty;
    if [ $duty -lt 0 ] ; then
        break
    fi
    echo $pwm
    echo $duty

    if [ $pwm = 0 ] ; then
        echo 0 > /sys/devices/ocp.3/pwm_test_P9_22.*/polarity
        echo 1 > /sys/devices/ocp.3/pwm_test_P9_22.*/run
        echo 3000000 > /sys/devices/ocp.3/pwm_test_P9_22.*/period
        echo $duty > /sys/devices/ocp.3/pwm_test_P9_22.*/duty
    elif [ $pwm = 1 ] ; then
        echo 0 > /sys/devices/ocp.3/pwm_test_P9_14.*/polarity
        echo 1 > /sys/devices/ocp.3/pwm_test_P9_14.*/run
        echo 3000000 > /sys/devices/ocp.3/pwm_test_P9_14.*/period
        echo $duty > /sys/devices/ocp.3/pwm_test_P9_14.*/duty
    elif [ $pwm = 2 ] ; then
        echo 0 > /sys/devices/ocp.3/pwm_test_P8_19.*/polarity
        echo 1 > /sys/devices/ocp.3/pwm_test_P8_19.*/run
        echo 3000000 > /sys/devices/ocp.3/pwm_test_P8_19.*/period
        echo $duty > /sys/devices/ocp.3/pwm_test_P8_19.*/duty
    else
        echo "Invalid PWM Index"
        break
    fi
done
echo 0 > /sys/devices/ocp.3/pwm_test_P9_22.*/run
echo 0 > /sys/devices/ocp.3/pwm_test_P9_14.*/run
echo 0 > /sys/devices/ocp.3/pwm_test_P8_19.*/run
```

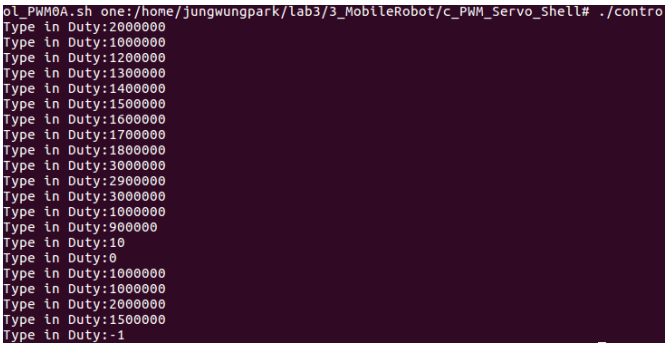
b) Captured Ubuntu terminal after testing.

A. Acquire_Triple_PWMs.sh



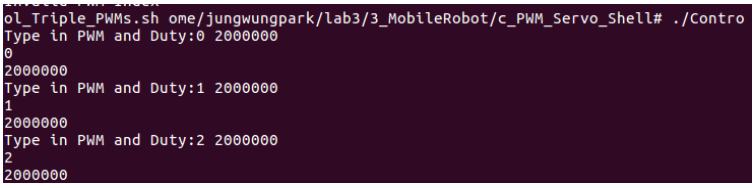

Problem B에서 한대로 리눅스 명령어가 있는 셸 스크립트를 실행했더니 오류없이 잘 작동하였다.

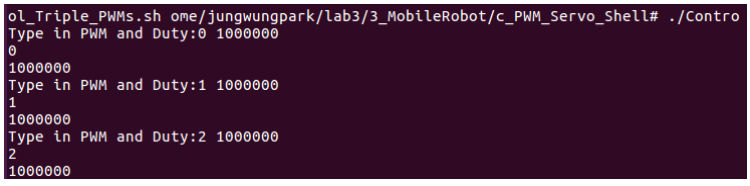
B. control_PWM0A.sh




셸 스크립트를 이용하여 P9-22 pin과 연결된 바퀴의 duty에 값을 여러가지 넣었더니 속도와 방향이 달라짐을 확인하였다. duty에 1500000을 넣어주었더니 바퀴가 멈추었고, 이보다 클 때는 시계방향으로, 작을 때는 시계 반대 방향으로 돌았다.

C. Control_Triple_PWMs.sh




각각의 바퀴에 duty 값을 넣을 때도 바퀴가 잘 작동함을 보여주었다.

4) Problem 3D.

a) C codes

A. Control_Triple_PWM_Servos.c

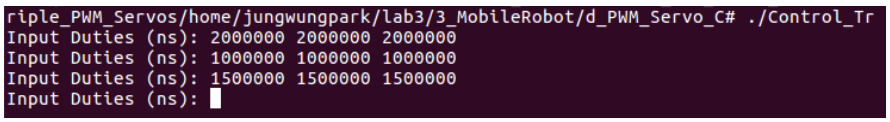
KLMS에 올라온 template 코드와 변함이 없다.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <unistd.h>
#include <fcntl.h>
#include <poll.h>
int main()
{
	FILE *duty, *run;
	int duty_ms[3];
	int duty_ns[3];
	int d0,d1,d2;
	int i;
	while (1){

		printf("Input Duties (ns): ");
		scanf("%d%d%d",&d0,&d1,&d2);
		duty_ns[0] = d0;
		duty_ns[1] = d1;
		duty_ns[2] = d2;
			/*
		for ( i = 0; i < 3; i++){
			duty_ns[i] = duty_ms[i] * 1000000;
			if (duty_ms[i] < 0){
			goto finish;
			}
		}
			*/
		if ((duty = fopen("/sys/devices/ocp.3/pwm_test_P9_22.15/duty", "w")) == NULL){
			printf("Error: PWM0 may not have been acquired₩n");
			exit(0);
```

```
		}

	if ((run = fopen("/sys/devices/ocp.3/pwm_test_P9_22.15/run", "w")) == NULL){
		printf("Error: PWM0 may not have been acquired₩n");
		exit(0);
		}

	fprintf(duty, "%d", duty_ns[0]);
	fprintf(run, "%d", 1);
	fflush(duty);
	fflush(run);
	if ((duty = fopen("/sys/devices/ocp.3/pwm_test_P9_14.16/duty", "w")) == NULL){
		printf("Error: PWM1 may not have been acquired₩n");
		exit(0);
		}

	if ((run = fopen("/sys/devices/ocp.3/pwm_test_P9_14.16/run", "w")) == NULL){
		printf("Error: PWM1 may not have been acquired₩n");
		exit(0);
		}

	fprintf(duty, "%d", duty_ns[1]);
	fprintf(run, "%d", 1);
	fflush(duty);
	fflush(run);
	if ((duty = fopen("/sys/devices/ocp.3/pwm_test_P8_19.17/duty", "w")) == NULL){
		printf("Error: PWM2 may not have been acquired₩n");
		exit(0);
		}

	if ((run = fopen("/sys/devices/ocp.3/pwm_test_P8_19.17/run", "w")) == NULL){
		printf("Error: PWM2 may not have been acquired₩n");
		exit(0);
		}

	fprintf(duty, "%d", duty_ns[2]);
	fprintf(run, "%d", 1);
	fflush(duty);
	fflush(run);
```

```
        }

finish:

        if ((run = fopen("/sys/devices/ocp.3/pwm_test_P9_22.15/run", "w")) == NULL){
                printf("Error: PWM0 may not have been acquired₩n");
                exit(0);
        }
        if ((duty = fopen("/sys/devices/ocp.3/pwm_test_P9_22.15/duty", "w")) == NULL){
                printf("Error: PWM0 may not have been acquired₩n");
                exit(0);
        }
        fprintf(run, "%d", 0);
        if ((duty = fopen("/sys/devices/ocp.3/pwm_test_P9_14.16/duty", "w")) == NULL){
                printf("Error: PWM1 may not have been acquired₩n");                    ₩
                exit(0);
                }

        if ((run = fopen("/sys/devices/ocp.3/pwm_test_P9_14.16/run", "w")) == NULL){
                printf("Error: PWM1 may not have been acquired₩n");
                exit(0);
                }
        fprintf(run, "%d", 0);
        if ((duty = fopen("/sys/devices/ocp.3/pwm_test_P8_19.17/duty", "w")) == NULL){
                printf("Error: PWM2 may not have been acquired₩n");
                exit(0);
                }

        if ((run = fopen("/sys/devices/ocp.3/pwm_test_P8_19.17/run", "w")) == NULL){
                printf("Error: PWM2 may not have been acquired₩n");
                exit(0);
                }
        fprintf(run, "%d", 0);

        fclose(duty);
        fclose(run);
}
```

b) Captured Ubuntu terminal after testing




c파일을 컴파일하고 실행하여 각각의 바퀴에 duty값을 넣으니 잘 작동되었다.

5) Problem 3D.

a) C codes

A. Keyboard_Control_TMR.c

KLMS에 올라온 template 코드와 변함이 없다.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <unistd.h>
#include <fcntl.h>
#include <poll.h>
#include <termios.h>
#include <unistd.h>
#include "gpio_control.h"
static struct termios old_tio;
static struct termios new_tio;

int main(void){
        FILE *duty0, *duty1, *duty2, *run0, *run1, *run2;



        int ivx = 0;
        int ivy = 0;
        int iw = 0;                             //those three to record net pressed times
        int led_one_on = 0;             //to control LEDs
        int led_two_on = 0;
        float vx, vy, wr;                       //entries of vector v
        float r, G;                             //r is radius of the wheel, G is the gain
        r = 0.02;
        G = 30000;
```

```
        int g2 = 3000000;

        float p[3];
        float w[3];
        float v[3];
        float mT[3][3] =   { {0, -0.67, 0.33*0.08}, {-0.577, 0.33, 0.33*0.08}, {0.577, 0.33, 0.33*0.08}};
//the transfrom matrix
        int input;

        gpio_export(30);
        gpio_export(31);

        gpio_set_dir(30, 1);
        gpio_set_dir(31, 1);

        if ((duty0 = fopen("/sys/devices/ocp.3/pwm_test_P9_22.15/duty", "w")) == NULL){
                printf("Error: PWM0 may not have been acquired₩n");
                exit(0);
        }
        if ((run0 = fopen("/sys/devices/ocp.3/pwm_test_P9_22.15/run", "w")) == NULL){
                printf("Error: PWM0 may not have been acquired₩n");
                exit(0);
        }
        if ((duty2 = fopen("/sys/devices/ocp.3/pwm_test_P9_14.16/duty", "w")) == NULL){
                printf("Error: PWM1 may not have been acquired₩n");
                exit(0);
                }
        if ((run2 = fopen("/sys/devices/ocp.3/pwm_test_P9_14.16/run", "w")) == NULL){
                printf("Error: PWM1 may not have been acquired₩n");
                exit(0);
        }

        if ((duty1 = fopen("/sys/devices/ocp.3/pwm_test_P8_19.17/duty", "w")) == NULL){
                printf("Error: PWM2 may not have been acquired₩n");
                exit(0);
        }
        if ((run1 = fopen("/sys/devices/ocp.3/pwm_test_P8_19.17/run", "w")) == NULL){
                printf("Error: PWM2 may not have been acquired₩n");
                exit(0);
```

```
}

printf(" Key Input Info₩n");

while (1){
        input = getchar();
        if (input == 't')           //to quit
        break;
        switch (input){
                        case 'a':
                                ivy += 10;
                                break;
                        case 'd':
                                ivy -= 10;
                                break;
                        case 'w':
                                ivx += 10;
                                break;
                        case 'x':
                                ivx -= 10;
                                break;
                        case 's':
                                ivx = 0; ivy = 0; iw = 0;   //reset;
                                break;
                        case 'z':
                                iw += 1000;
                                break;
                        case 'c':
                                iw -= 1000;
                                break;
                        case 'q':
                                if (led_one_on)
                                {
                                gpio_set_value( 31, 0);
                                led_one_on = 0;
                                }
                                else
                                {
                                gpio_set_value( 31, 1);
```

```
                                led_one_on = 1;
                                }

                                break;
                        case 'e':
                                if (led_two_on)
                                {
                                gpio_set_value( 30, 0);
                                led_two_on = 0;
                                }
                                else
                                {
                                gpio_set_value( 30, 1);
                                led_two_on = 1;
                                }

                                break;
        }
        vx = 0.02 * (float)ivx;
        vy = 0.02 * (float)ivy;
        wr = 0.01 * (float)iw;
        printf("v: %f , %f, %f₩n", vx, vy, wr);
        v[0] = vx;
        v[1] = vy;
        v[2] = wr;

        for(int i = 0; i < 3; i++){
                w[i] = 0;
                for(int j = 0; j < 3; j++){
                        w[i] += mT[i][j] * v[j];
                }
        }
        w[0] /= r;
        w[1] /= r;
        w[2] /= r;
        printf("w: %f , %f, %f₩n", w[0], w[1], w[2]);

        p[0] = /*1510095*/ 1500000 - G * w[0];
        p[1] = /*1465450*/ 1500000 - G * w[1];
```

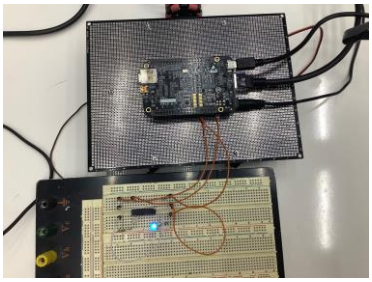
```
		p[2] = /*1462500*/ 1500000 - G * w[2];
		if (p[1] < 1500000){
			p[1] -= 500000;//39795;
		}
		if (p[2] < 150000){
			p[2] -= 500000;//1434950 45050
		}
		if (p[0] < 1500000){
			p[0] -= 500000;
		}
		if (p[1] > 1500000){
			p[1] += 500000;
		}
		if (p[2] > 1500000){
			p[2] += 500000;
		}
		if (p[0] > 1500000){
			p[0] += 500000;
		}

		printf("p: %f , %f, %f₩n", p[0], p[1], p[2]);


		fprintf(duty0, "%d",(int)p[1] );
		fprintf(run0, "%d", 1);
		fflush(duty0);
		fflush(run0);
		fprintf(duty1, "%d", (int)p[2]);
		fprintf(run1, "%d", 1);
		fflush(duty1);
		fflush(run1);
		fprintf(duty2, "%d", (int)p[0]);
		fprintf(run2, "%d", 1);
		fflush(duty2);
		fflush(run2);
	}

	fprintf(run0, "%d", 0);
```

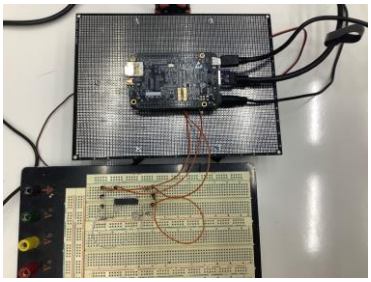
```
        fprintf(run1, "%d", 0);

        fprintf(run2, "%d", 0);

        gpio_unexport(30);
        gpio_unexport(31);

        fclose(run0);
        fclose(run1);
        fclose(run2);

        fclose(duty0);
        fclose(duty1);
        fclose(duty2);
        return 0;
        }
```

b) Captured Ubuntu terminal after testing.

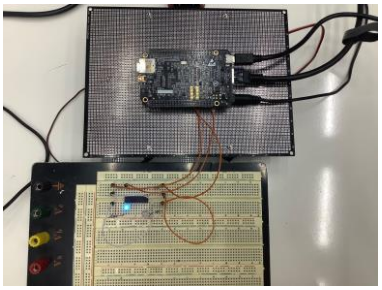
< LED Control >

```
ght_control one:/home/jungwungpark/lab3/3_MobileRobot/e_TMR_Control_C# ./test_lig
 Key Input Info
q
v: 0.000000 , 0.000000, 0.000000
w: 0.000000 , 0.000000, 0.000000
p: 1500000.000000 , 1500000.000000, 1500000.000000
v: 0.000000 , 0.000000, 0.000000
w: 0.000000 , 0.000000, 0.000000
p: 1500000.000000 , 1500000.000000, 1500000.000000
q
v: 0.000000 , 0.000000, 0.000000
w: 0.000000 , 0.000000, 0.000000
p: 1500000.000000 , 1500000.000000, 1500000.000000
v: 0.000000 , 0.000000, 0.000000
w: 0.000000 , 0.000000, 0.000000
p: 1500000.000000 , 1500000.000000, 1500000.000000
e
v: 0.000000 , 0.000000, 0.000000
w: 0.000000 , 0.000000, 0.000000
p: 1500000.000000 , 1500000.000000, 1500000.000000
v: 0.000000 , 0.000000, 0.000000
w: 0.000000 , 0.000000, 0.000000
p: 1500000.000000 , 1500000.000000, 1500000.000000
e
v: 0.000000 , 0.000000, 0.000000
w: 0.000000 , 0.000000, 0.000000
p: 1500000.000000 , 1500000.000000, 1500000.000000
v: 0.000000 , 0.000000, 0.000000
w: 0.000000 , 0.000000, 0.000000
p: 1500000.000000 , 1500000.000000, 1500000.000000
```

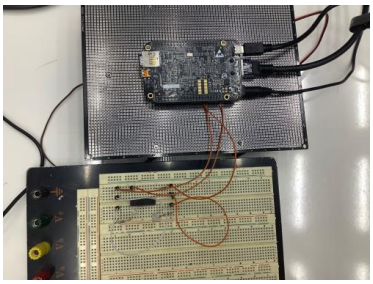




<처음 q를 눌렀을 때 >  <후에 q를 눌렀을 때>



<처음 e를 눌렀을 때 >



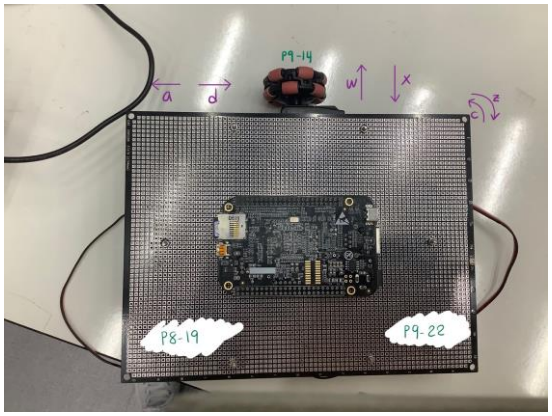
<후에 e를 눌렀을 때>

q와 e를 누르면서 각각의 LED가 켜졌다 꺼짐을 확인하였다. q는 사진 상 오른쪽 LED를, e는 왼쪽 LED를 제어하였다.

< Wheel control >

```
w
v: 0.200000 , 0.000000, 0.000000
w: 0.000000 , -5.770000, 5.770000
p: 1500000.000000 , 2173100.000000, 1326900.000000
v: 0.200000 , 0.000000, 0.000000
w: 0.000000 , -5.770000, 5.770000
p: 1500000.000000 , 2173100.000000, 1326900.000000
s
v: 0.000000 , 0.000000, 0.000000
w: 0.000000 , 0.000000, 0.000000
p: 1500000.000000 , 1500000.000000, 1500000.000000
v: 0.000000 , 0.000000, 0.000000
w: 0.000000 , 0.000000, 0.000000
p: 1500000.000000 , 1500000.000000, 1500000.000000
x
v: -0.200000 , 0.000000, 0.000000
w: 0.000000 , 5.770000, -5.770000
p: 1500000.000000 , 826900.000000, 2173100.000000
v: -0.200000 , 0.000000, 0.000000
w: 0.000000 , 5.770000, -5.770000
p: 1500000.000000 , 826900.000000, 2173100.000000
s
v: 0.000000 , 0.000000, 0.000000
w: 0.000000 , 0.000000, 0.000000
p: 1500000.000000 , 1500000.000000, 1500000.000000
v: 0.000000 , 0.000000, 0.000000
w: 0.000000 , 0.000000, 0.000000
p: 1500000.000000 , 1500000.000000, 1500000.000000
a
v: 0.000000 , 0.200000, 0.000000
w: -6.700000 , 3.300000, 3.300000
p: 2201000.000000 , 901000.000000, 1401000.000000
v: 0.000000 , 0.200000, 0.000000
w: -6.700000 , 3.300000, 3.300000
p: 2201000.000000 , 901000.000000, 1401000.000000
s
```

```
s
v: 0.000000 , 0.000000, 0.000000
w: 0.000000 , 0.000000, 0.000000
p: 1500000.000000 , 1500000.000000, 1500000.000000
v: 0.000000 , 0.000000, 0.000000
w: 0.000000 , 0.000000, 0.000000
p: 1500000.000000 , 1500000.000000, 1500000.000000
d
v: 0.000000 , -0.200000, 0.000000
w: 6.700000 , -3.300000, -3.300000
p: 799000.000000 , 2099000.000000, 2099000.000000
v: 0.000000 , -0.200000, 0.000000
w: 6.700000 , -3.300000, -3.300000
p: 799000.000000 , 2099000.000000, 2099000.000000
s
v: 0.000000 , 0.000000, 0.000000
w: 0.000000 , 0.000000, 0.000000
p: 1500000.000000 , 1500000.000000, 1500000.000000
v: 0.000000 , 0.000000, 0.000000
w: 0.000000 , 0.000000, 0.000000
p: 1500000.000000 , 1500000.000000, 1500000.000000
z
v: 0.000000 , 0.000000, 10.000000
w: 13.200000 , 13.200000, 13.200000
p: 604000.000000 , 604000.000000, 1104000.000000
v: 0.000000 , 0.000000, 10.000000
w: 13.200000 , 13.200000, 13.200000
p: 604000.000000 , 604000.000000, 1104000.000000
s
v: 0.000000 , 0.000000, 0.000000
w: 0.000000 , 0.000000, 0.000000
p: 1500000.000000 , 1500000.000000, 1500000.000000
v: 0.000000 , 0.000000, 0.000000
w: 0.000000 , 0.000000, 0.000000
p: 1500000.000000 , 1500000.000000, 1500000.000000
c
v: 0.000000 , 0.000000, -10.000000
w: -13.200000 , -13.200000, -13.200000
p: 2396000.000000 , 2396000.000000, 2396000.000000
v: 0.000000 , 0.000000, -10.000000
w: -13.200000 , -13.200000, -13.200000
p: 2396000.000000 , 2396000.000000, 2396000.000000
s
v: 0.000000 , 0.000000, 0.000000
w: 0.000000 , 0.000000, 0.000000
p: 1500000.000000 , 1500000.000000, 1500000.000000
v: 0.000000 , 0.000000, 0.000000
w: 0.000000 , 0.000000, 0.000000
p: 1500000.000000 , 1500000.000000, 1500000.000000
t
```



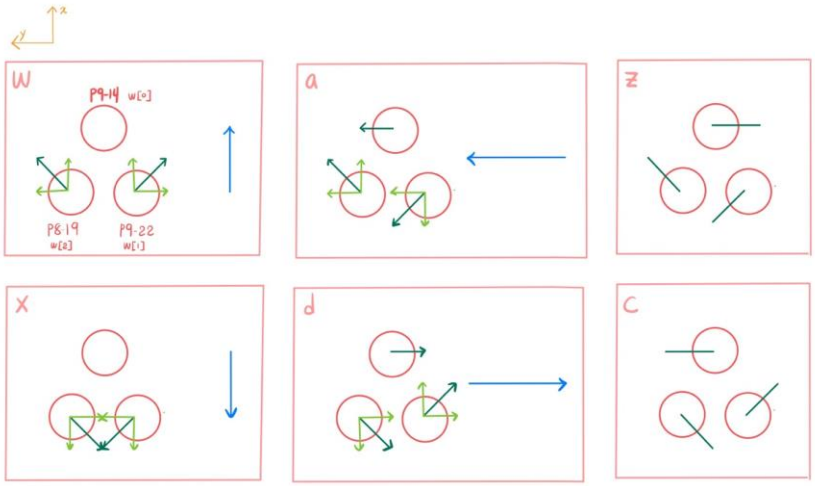

키보드의 w, x, a, d, c, z 를 누르니 위의 사진에 나타난 방향으로 움직임을 확인했다. s를 누르면 멈추었고, t를 누르니 종료되었다. 터미널에 보여진 v는 첫 번째가 w, x 방향, 두 번째가 a, d 방향, 세 번째가 c, z 방향으로 로봇이 움직이고 있음을 나타내고, 크기는 속도를 나타낸다. w는 각 바퀴의 속도를 나타낸다. 첫 번째는 P9-14 바퀴, 두 번째는 P9-22 바퀴, 세 번째는 P8-19 바퀴를 나타낸다. 양수는 반시계방향, 음수는 시계방향으로 돌고 있음을 나타낸다. p는 바퀴가 w의 속도로 움직이기 위해 넣은 duty 값을 나타낸다.

c) Explanation of each servos required speed for x and y-axis shift and rotation.

다음은 각각의 키를 눌렀을 때 바퀴의 방향을 나타낸다.

| | P9-14 | P8-19 | P9-22 |
|---|---|---|---|
| W | 멈춤 | 반시계방향 | 시계방향 |
| X | 멈춤 | 시계방향 | 반시계방향 |
| A | 시계방향 | 반시계방향 | 반시계방향 |
| D | 반시계방향 | 시계방향 | 시계방향 |
| Z | 반시계방향 | 반시계방향 | 반시계방향 |
| C | 시계방향 | 시계방향 | 시계방향 |

위의 표를 그림으로 나타내면 다음과 같다.




위의 그림에서 빨간색 원은 바퀴의 위치를 나타내고, 초록색 화살표는 위에서 beagle bone을 내려다 보았을 때 바퀴가 움직이는 방향을 나타낸다. 이때, 각 칸 안에 있는 초록색 벡터끼리 길이가 같다. 연두색 화살표는 각각의 초록색 벡터를 x축, y축 두 개의 성분으로 나눈 것이다. 파란색 화살표는 합 벡터를 나타낸다.

따라서 로봇이 움직이는 방향은 합벡터의 방향이라고 할 수 있다.

'W'키와 'X'키는 로봇을 x축 방향으로 움직이게 한다. 이를 위해 y축 성분의 합벡터는 0이 되도록 한다. 최종적인 합 벡터는 각각 x축의 양의 방향, x축의 음의 방향이 된다. 이때 각 칸의 초록색 벡터끼리 크기가 같아야 하므로 움직이는 두 바퀴의 스피드는 같아야 한다.

'A'키와 'D'키는 아래쪽에 위치한 두 바퀴의 x축 성분의 벡터가 0이 되기 때문에 합 벡터는 각각 y축의 양의 방향, y축의 음의 방향이 된다. 이때 아래쪽의 두 바퀴의 초록색 벡터는 크기가 같아야 하므로 P9-22, P8-19 바퀴의 스피드는 같아야 한다. 여기서 얻은 합벡터의 크기와 w, x의 합벡터의 크기가 다를 수 있다. 크기를 같게 하려면 (사방으로 움직이는 거리를 같게 하려면) 같은 칸 안에 있는 초록색 벡터끼리 같은 크기의 수를 곱하여 합벡터의 크기를 보정할 수 있다.

'Z'키와 'C키'는 회전에만 영향을 주어야 하기 때문에 (x축, y축 방향의 이동이 없기 때문에) 각 칸 안에 있는 초록색 벡터끼리 크기가 같아야 한다. 따라서 세 바퀴의 스피드가 모두 같아야 한다. 회전에 관해서는 각속도를 이용하기 때문에 로봇의 중심으로부터 바퀴까지의 거리도 파악해야 한다.

모터의 움직임을 구현하기 위해선 먼저 필요한 x,y,rotation 속도의 정보를 파악한 후, matrix formation에 위 속도를 대입하면 바퀴가 필요한 속도를 구할 수 있다. 위 코드에서는 mT[3][3] = { {0, -0.67, 0.33*0.08}, {-0.577, 0.33, 0.33*0.08}, {0.577, 0.33, 0.33*0.08}};로 구현하였다.