목차

1. 수행한 과제 3

Ultrasonic Sensor 3

Joystick Module 3

2. 부품 리스트 4

Ultrasonic Sensor 4

핵심 사용 부품 4

기타 사용 부품 4

Joystick Module 4

핵심 사용 부품 4

기타 사용 부품 5

3. 구현하고자 하는 기능 및 설명 5

Ultrasonic Sensor 5

Core – 1. 초음파 센서를 사용하여 장애물 스캐너를 구현하기 5

Idea – 1. 이동 평균 필터(Moving Average Filter)를 이용한 초음파 센서 값 보정 5

Idea – 2. 펄스 폭 변조(PWM)를 이용한 LED 세부 밝기 조절 5

Idea – 3. DC 모터를 이용한 근접 작동 선풍기 기능 5

Joystick Module 6

Core – 1. 참참참 게임을 구현하기 6

Idea – 1. 4열 7세그먼트 LED를 이용한 시각적 피드백 추가 6

Idea – 2. 수동 부저를 이용한 게임 사운드 기능 추가 6

4. 부품과 아두이노 우노 보드와의 연결도 7

Ultrasonic Sensor 7

Joystick Module 8

5. 프로그램 소스 9

UltrasonicSensor.ino 9

JoystickModule.ino 13

1. 수행한 과제

Ultrasonic Sensor

- 초음파 센서, LED 5개를 아두이노에 연결한다.
- 초음파 센서를 사용하여 장애물 스캐너를 구현한다.

- 초음파 센서 전면에 있는 장애물의 거리에 따라 LED를 표시한다.
  - 장애물의 거리 40cm이하인 경우, 모든 LED off, 40cm~45cm 인 경우 LED 1개를 on, 46~50cm 경우, 2개, 51~55cm 인 경우 3개, 55~60cm 인 경우 4개, 61cm 이상은 모든 LED on 으로 표시한다.
- 측정 주기는 100ms 단위로 하여, 초음파 센서를 수평 또는 수직 이동시켜서 40cm 거리에 있는 장애물의 형태를 5cm 해상도로 스캔할 수 있어야 한다.

Joystick Module

- 조이스틱과 LED 3개, 스위치 1개를 아두이노에 연결한다.
- '참참참' 게임을 구현한다.

- LED 3개를 수평으로 배치한다.
- 스위치를 누르면 전체 LED 를 1초 간격으로 3번 blinking 하여 게임 시작을 표시한다.
- 게임이 시작되면, 1초 대기 후, 중앙 LED가 0.3초 간격으로 ' 참참참 ' 을 하고 좌, 우 LED를 사용하여 랜덤으로 방향을 지시한다.
- 사용자는 조이스틱을 사용하여 중앙 LED에 맞춰 동시에 좌, 우 방향을 결정한다.
- LED 표시와 조이스틱 방향 설정의 허용 지연시간은 10ms이다. 즉 LED 표시가 된
- 직후 10ms 이내에 방향을 결정해야 한다. 허용 지연시간을 초과한 경우 '패 ' 로 판정한다.
- 사용자가 방향을 맞추면(사용자 ' 승) 전체 LED를 0.5초 간격으로 blinking 한다.
- 사용자가 '패 ' 한 경우, 중앙 LED를 2초 간 ON 후, OFF 한다.
- 다음 게임 시작을 대기하는 동안 전체 LED 를 OFF 한다.

2. 부품 리스트

Ultrasonic Sensor

핵심 사용 부품

- 아두이노 우노 R3(Arduino Uno R3) x1
- 초음파 센서 HC-SR04 (Ultrasonic Sensor) x1
- DC 모터(DC Motor) x1
- DC모터 드라이버 IC(DC Motor Driver IC, L293D)
- 흰색 LED(White LED) x2
- 초록색 LED(Green LED) x1
- 파란색 LED(Blue LED) x1
- 빨간색 LED(Red LED) x1

기타 사용 부품

- 830 포인트 브레드보드(830 Point Breadboard) x1
- 9V 배터리(9V Battery) x1
- 배터리 케이블(Battery Cable)
- 220Ω 저항(220 Resistance) x5
- 점퍼 케이블(Jumper Cable)

Joystick Module

핵심 사용 부품

- 아두이노 우노 R3(Arduino Uno R3) x1
- 조이스틱 모듈(Joystick) x1
- 빨간색 LED(Red LED) x1
- 노란색 LED(Yellow LED) x1
- 초록색 LED(Green LED) x1

택트 스위치(Tact Switch) x1

4열 7세그먼트 LED(4 Digit 7 segment LED) x1

수동 부저(Passive Buzzer) x1

10K 가변 저항(10K Variable Resistance) x1

기타 사용 부품

220Ω 저항(220 Resistance) x3

330Ω 저항(330 Resistance) x4

점퍼 케이블(Jumper Cable)

3. 구현하고자 하는 기능 및 설명

Ultrasonic Sensor

Core – 1. 초음파 센서를 사용하여 장애물 스캐너를 구현하기

장애물과의 거리를 100MS 측정 주기로 센싱하여 장애물의 거리 40cm이하인 경우 모든 LED off, 40cm~45cm 인 경우 LED 1개를 on, 46~50cm 경우, 2개, 51~55cm 인 경우 3개, 55~60cm 인 경우 4개, 61cm 이상은 모든 LED on 으로 표시한다.

Idea – 1. 이동 평균 필터(Moving Average Filter)를 이용한 초음파 센서 값 보정

$$\bar{x}_k = \frac{x_{k-n+1}+x_{k-n+2}+\cdots+x_k}{n}$$

초음파 센서의 Noise 보정을 위해 최근 10개의 거리 측정 값을 평균하여 로직 처리에 사용하였다.

Idea – 2. 펄스 폭 변조(PWM)를 이용한 LED 세부 밝기 조절

각 LED를 단순히 구간 별로 On/Off 하는 것이 아니라, 각 구간에 대해서 대응하는 LED의 밝기를 세밀한 밝기 조절로 표현하였다. 예를 들어 40~45cm 구간에서 44cm 거리에서의 첫번째 LED 밝기는 41cm 거리에서의 첫번째 LED 밝기보다 60%가량 밝게 표시된다.

Idea – 3. DC 모터를 이용한 근접 작동 선풍기 기능

DC 모터와 L293D 모터 드라이버를 이용하여 초음파 센서에 장애물 있으면 작동하는 선풍기 기능을 구현하였다. 60cm 이내에 장애물이 있을 때 작동하며, 가까이에

있을수록 선풍기의 강도가 더 세게 조절된다.

Joystick Module

Core – 1. 참참참 게임을 구현하기

조이스틱 모듈과, 스위치 1개, LED 3개를 연결하고, 조이스틱 입력과 LED 조명을 통해 진행하는 '참참참' 게임을 구현하였다.

Idea – 1. 4열 7세그먼트 LED를 이용한 시각적 피드백 추가

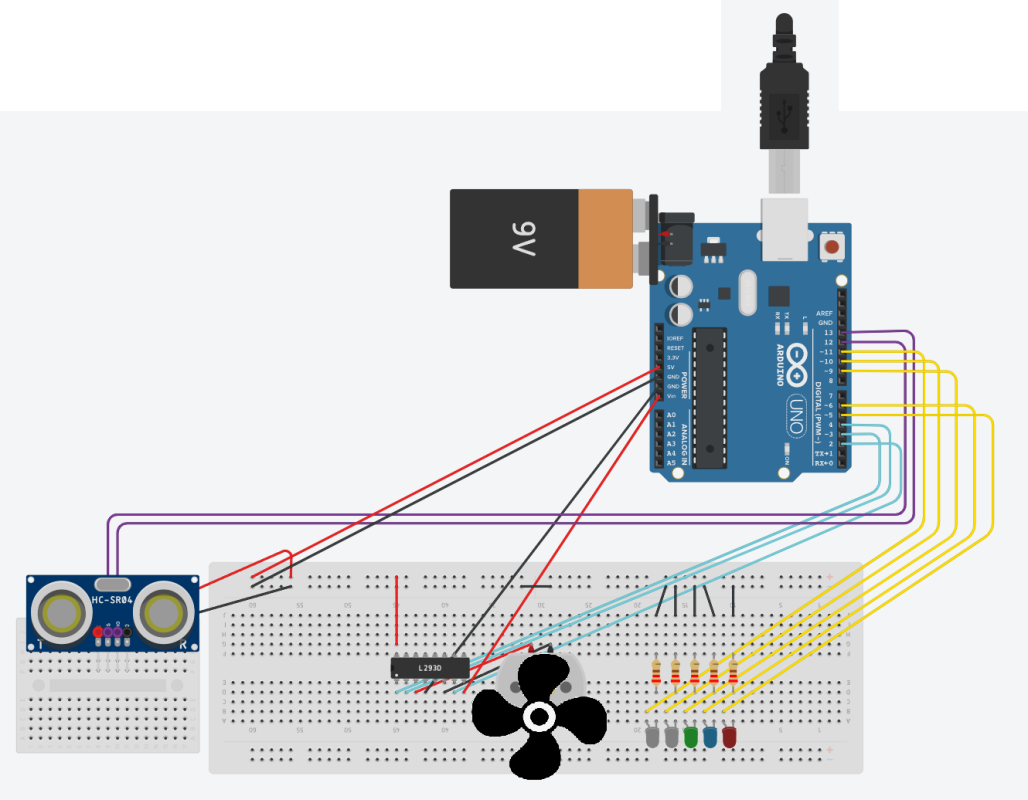
4열 7세그먼트 LED를 연결하고, 기존 '참참참' 게임이 시작할 때 "REDY"(Ready)와 "STAT"(Start) 문구를, 게임이 진행될 때에는 "CHA1" (참1), "CHA2"(참2), "LEFT", "RIGT"(Right)의 표시 문구를, 게임의 승패를 표시할 때에는 "LOSE"와 "COOL"(승리)의 문구를 표시하여 게임의 시각적 피드백 기능을 업그레이드 하였다.

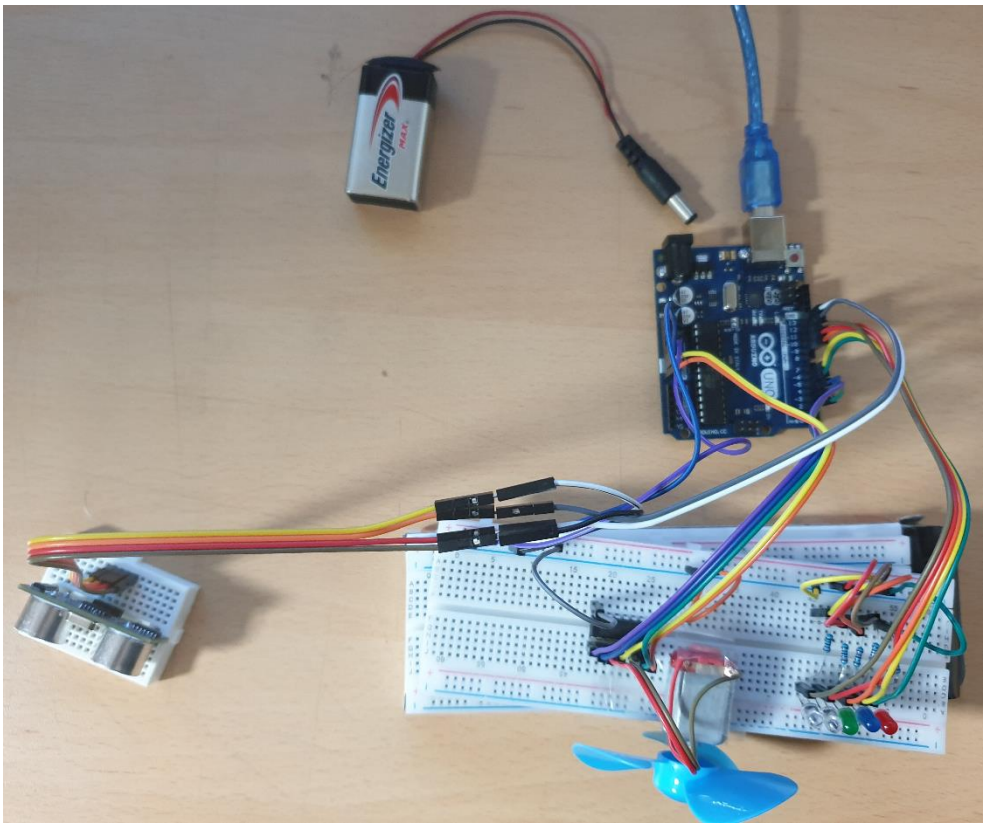
Idea – 2. 수동 부저를 이용한 게임 사운드 기능 추가

수동 부저를 연결하여 '참참참' 게임 시작 대기시의 사운드 이펙트와, 참참참 진행 타이밍 사운드 피드백, 패배 비프음, 승리시의 빵빠레 재생 기능을 추가하였다. 또한 연결된 가변 저항을 조절하여 부저의 음량을 조절할 수 있도록 했다.

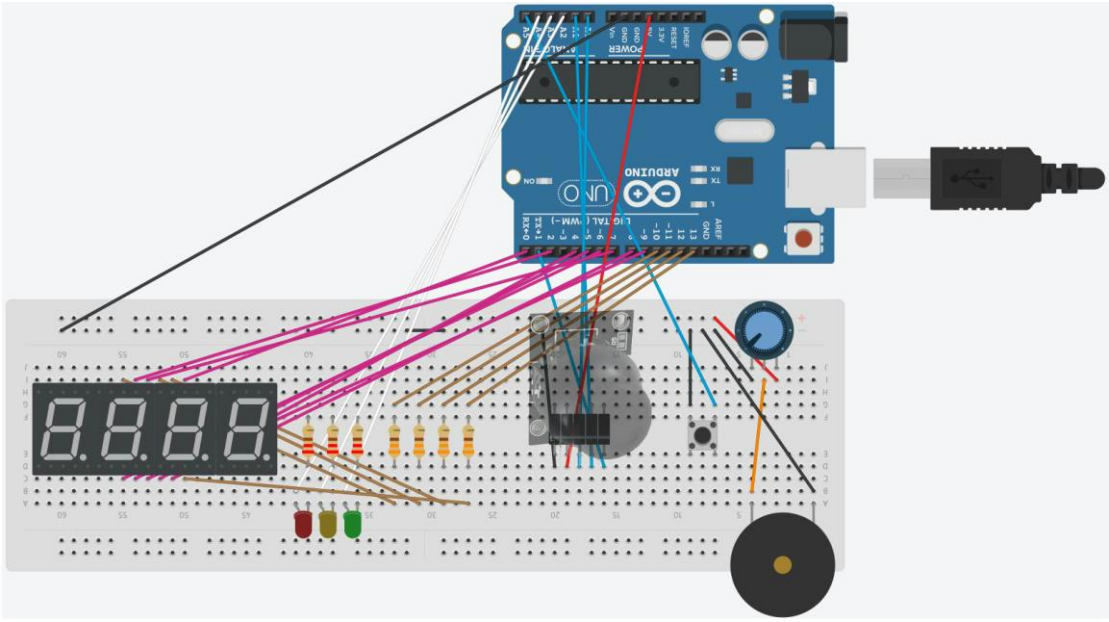
4. 부품과 아두이노 우노 보드와의 연결도

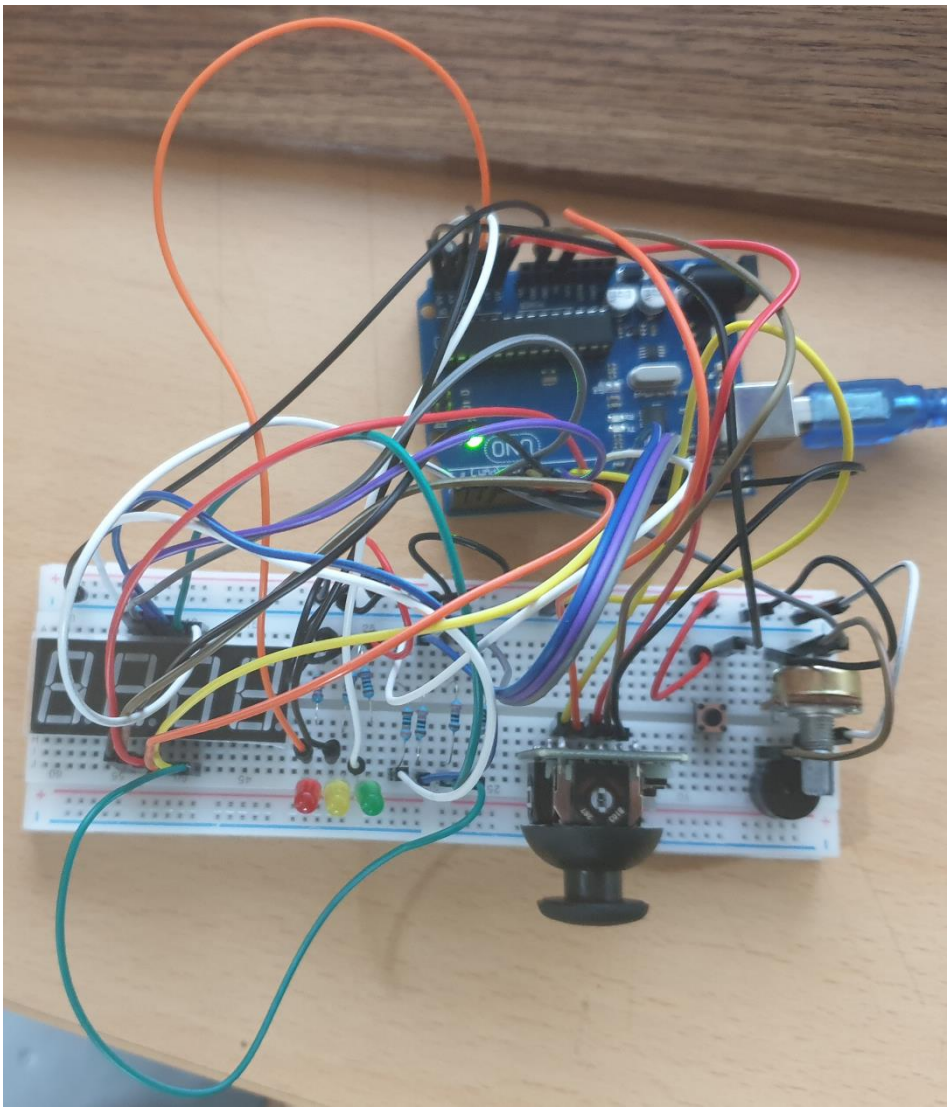
Ultrasonic Sensor





Joystick Module





5. 프로그램 소스

GitHub: https://github.com/refracta/koreatech-assignment/tree/master/MicroprocessorNPractice/Assignment2

SevSeg: https://github.com/DeanIsMe/SevSeg

UltrasonicSensor.ino

```
/**
 * refracta - 마이크로프로세서및실습 (CSE124-02)
 * 과제 2 Ultrasonic Sensor 소스 코드
 */

#define DEBUG_MODE true
// 디버그 플래그

#if DEBUG_MODE
  #define debug(log) Serial.println(log);
#else
  #define debug(log)
#endif

#define LED1_PIN 5
#define LED2_PIN 6
#define LED3_PIN 9
#define LED4_PIN 10
#define LED5_PIN 11
#define NUMBER_OF_LED 5
// LED 핀 배열 선언
const int LED_PINS[] = {LED1_PIN, LED2_PIN, LED3_PIN, LED4_PIN,
LED5_PIN};

#define USS_TRIGGER_PIN 12
#define USS_ECHO_PIN 13

#define L293D_ENABLE_PIN 3
#define L293D_IN1_PIN 4
#define L293D_IN2_PIN 2

#define USE_MOTOR
#define MOTOR_SPEED_MIN 130
#define SOUND_SPEED 0.0343

#define GET_INTENSITY(DELTA) ((int) ((distance - DELTA) / 5 * 255))
```

```
37
38 // LED, 조이스틱, 초음파 센서, 모터 드라이버의 핀을 초기화한다.
39 void setup() {
40   Serial.begin (9600);
41
42   pinMode(LED1_PIN, OUTPUT);
43   pinMode(LED2_PIN, OUTPUT);
44   pinMode(LED3_PIN, OUTPUT);
45   pinMode(LED4_PIN, OUTPUT);
46   pinMode(LED5_PIN, OUTPUT);
47
48   pinMode(USS_TRIGGER_PIN, OUTPUT);
49   pinMode(USS_ECHO_PIN, INPUT);
50
51   pinMode(L293D_ENABLE_PIN, OUTPUT);
52   pinMode(L293D_IN1_PIN, OUTPUT);
53   pinMode(L293D_IN2_PIN, OUTPUT);
54 }
55
56 // 초음파 센서로부터 현재 거리 값을 읽는다. (cm)
57 double getDistance() {
58   digitalWrite(USS_TRIGGER_PIN, LOW);
59   delayMicroseconds(5);
60   digitalWrite(USS_TRIGGER_PIN, HIGH);
61   delayMicroseconds(10);
62   digitalWrite(USS_TRIGGER_PIN, LOW);
63   return (pulseIn(USS_ECHO_PIN, HIGH) * SOUND_SPEED) / 2;
64 }
65
66 // LED 세기 배열
67 int intensities[NUMBER_OF_LED];
68
69 // LED의 세기를 설정한다.
70 void setIntensities(int i1, int i2, int i3, int i4, int i5) {
71   intensities[0] = i1;
72   intensities[1] = i2;
73   intensities[2] = i3;
74   intensities[3] = i4;
75   intensities[4] = i5;
76 }
77
78 // 모든 LED의 세기를 LED 세기 배열에 있는 값으로 설정한다.
79 void setLED() {
```

```
  for(int i = 0; i < NUMBER_OF_LED; i++) {
    analogWrite(LED_PINS[i], intensities[i]);
  }
}

// 속도와 회전 방향을 매개변수로 받아 모터 동작을 설정한다.
void setMotor(int speed, boolean reverse) {
  analogWrite(L293D_ENABLE_PIN, speed);
  digitalWrite(L293D_IN1_PIN, reverse);
  digitalWrite(L293D_IN2_PIN, !reverse);
}

// 모터를 작동시킨다.
bool isMotorDown = true;
void runMotor(int speed) {
  #ifdef USE_MOTOR
  if(isMotorDown && speed > 0) {
        // 저출력으로 모터 사용 시, 작동하지 않는 경우 잠시 역방향
출력을 걸어주었다. 정방향 출력을 걸어주면 원활한 작동에 도움이 되는
것을 관찰하고 정지 후 모터 구동시 모터에게 역방향 출력을 가하는 부분을
추가하였다.
    debug("Motor init");
    setMotor(255, true);
    delay(250);
    setMotor(255, false);
    delay(250);
    isMotorDown = false;
  }
  setMotor(speed, false);
  if(speed == 0){
    isMotorDown = true;
  }
  #endif
}

#define SAMPLE_LENGTH 10
double dataArray[SAMPLE_LENGTH] = {0, };
// 최근 10개 초음파 센서 거리 센싱 결과의 평균을 구한다. (이동 평균
필터)
double getAverageDistance() {
  for(int i = 0; i < SAMPLE_LENGTH - 1; i ++){
    dataArray[i] = dataArray[i + 1];
  }
```

```
123   dataArray[SAMPLE_LENGTH - 1] = getDistance();
124   double distance = 0;
125   for(int i = 0; i < SAMPLE_LENGTH; i ++){
126     distance += dataArray[i];
127   }
128   distance /= SAMPLE_LENGTH;
129   return distance;
130 }
131
132 #define BASE_DISTANCE 40
133 void loop() {
134   double distance = getAverageDistance();
135
136   // 거리에 따라 LED 를 제어한다.
137   if(distance <= BASE_DISTANCE) {
138     setIntensities(0, 0, 0, 0, 0);
139   } else if (distance <= BASE_DISTANCE + 5) {
140     setIntensities(GET_INTENSITY(BASE_DISTANCE), 0, 0, 0, 0);
141   } else if(distance <= BASE_DISTANCE + 10) {
142     setIntensities(255, GET_INTENSITY(BASE_DISTANCE + 5), 0, 0, 0);
143   } else if(distance <= BASE_DISTANCE + 15) {
144     setIntensities(255, 255, GET_INTENSITY(BASE_DISTANCE + 10), 0, 0);
145   } else if(distance <= BASE_DISTANCE + 20) {
146     setIntensities(255, 255, 255, GET_INTENSITY(BASE_DISTANCE + 15),
147 0);
148   } else {
149     int intensity = GET_INTENSITY(BASE_DISTANCE + 20);
150     intensity = intensity > 255 ? 255 : intensity;
151     setIntensities(255, 255, 255, 255, intensity);
152   }
153   setLED();
154
155 // 거리에 따라 모터를 제어한다.
156   if(distance < 10) {
157     runMotor(255);
158   } else if(60 <= distance) {
159     runMotor(0);
160   } else {
161     double motorSpeed = (distance - 10) / (60 - 10);
162     motorSpeed = MOTOR_SPEED_MIN + (1 - motorSpeed) * (255 -
163 MOTOR_SPEED_MIN);
      runMotor((int) motorSpeed);
      // 10cm ~ 60cm 을 255 ~ 120 모터 강도로 변환
```

```
    }

    delay(100);
    debug(distance);
  }
```

JoystickModule.ino

```
/**
 * refracta - 마이크로프로세서및실습 (CSE124-02)
 * 과제 2 Joystick Module 소스 코드
 */

#include "SevSeg.h"

#define BUZZER_PIN 3

#define SEV_SEG_A_PIN 2
#define SEV_SEG_B_PIN 0
#define SEV_SEG_C_PIN 4
#define SEV_SEG_D_PIN 5
#define SEV_SEG_E_PIN 6
#define SEV_SEG_F_PIN 7
#define SEV_SEG_G_PIN 8
#define SEV_SEG_DP_PIN 9

#define SEV_SEG_D1_PIN 10
#define SEV_SEG_D2_PIN 11
#define SEV_SEG_D3_PIN 12
#define SEV_SEG_D4_PIN 13

#define JOY_X_PIN A0
#define JOY_Y_PIN A1

#define LED1_PIN A2
#define LED2_PIN A3
#define LED3_PIN A4
#define NUM_OF_LED 3
// LED 핀 배열 선언
const int LED_PINS[] = {A2, A3, A4};

#define SWITCH_PIN A5
```

```
35
36 // #define DEBUG_MODE
37
38 #ifdef DEBUG_MODE
39         #define debug(log) Serial.println(log);
40 #else
41         #define debug(log)
42 #endif
43
44 #define JOY_TRIGGER_VALUE 256
45 #define GET_JOY_X() map(analogRead(JOY_X_PIN), 0, 1023, -512, 512)
46 #define GET_JOY_Y() map(analogRead(JOY_Y_PIN), 0, 1023, -512, 512)
47
48 #define delay(ms) delayWithRefresh(ms)
49
50 SevSeg Display;
51
52 // 7 세그먼트 모듈의 디스플레이 출력을 유지하기 위해서 전용 delay
53 함수를 정의하고 매크로를 이용하여 바꿔치기한다.
54 void delayWithRefresh(long ms) {
55         long start = millis();
56         while(true) {
57                 Display.refreshDisplay();
58                 if(millis() - start >= ms){
59                         break;
60                 }
61         }
62 }
63
64 // 7 세그먼트, 조이스틱, LED, 스위치의 핀과 설정들을 초기화한다.
65 void setup() {
66         byte digitPins[] = {SEV_SEG_D1_PIN, SEV_SEG_D2_PIN,
67 SEV_SEG_D3_PIN, SEV_SEG_D4_PIN};
68         byte segmentPins[] = {SEV_SEG_A_PIN, SEV_SEG_B_PIN,
69 SEV_SEG_C_PIN, SEV_SEG_D_PIN, SEV_SEG_E_PIN, SEV_SEG_F_PIN,
70 SEV_SEG_G_PIN, SEV_SEG_DP_PIN};
71         Display.begin(COMMON_ANODE, 4, digitPins, segmentPins, true);
72         Display.setBrightness(100);
73
74         pinMode(JOY_X_PIN, INPUT);
75         pinMode(JOY_Y_PIN, INPUT);
76
77         pinMode(LED1_PIN, OUTPUT);
```

```
78          pinMode(LED2_PIN, OUTPUT);
79          pinMode(LED3_PIN, OUTPUT);
80
81          pinMode(SWITCH_PIN, INPUT_PULLUP);
82
83          #ifdef DEBUG_MODE
84          Serial.begin(9600);
85          #endif
86          // 7 세그먼트 모듈이 0 번핀을 사용하기 때문에 Serial 기능을
87 사용하면 해당 모듈의 출력에 영향이 있다.
88 }
89
90 /*
91  * data from
92 https://www.arduino.cc/en/Tutorial/BuiltInExamples/toneMelody
93  * 아두이노 튜토리얼 문서의 헤더 파일을 배열 형태로 정리한 것, 각
94 음계의 진동수의 상수 정의이다.
95  */
96 const int frequencies[] =
97 {
98          31,
99
100         // octave 0
101         33, 35, 37, 39, 41, 44, 46, 49, 52, 55, 58, 62,
102                                         // octave 1
103         65, 69, 73, 78, 82, 87, 93, 98, 104, 110, 117, 123,
104                                 // octave 2
105         131, 139, 147, 156, 165, 175, 185, 196, 208, 220, 233, 247,
106                         // octave 3
107         262, 277, 294, 311, 330, 349, 370, 392, 415, 440, 466, 494,
108                         // octave 4
109         523, 554, 587, 622, 659, 698, 740, 784, 831, 880, 932, 988,
110                         // octave 5
111         1047, 1109, 1175, 1245, 1319, 1397, 1480, 1568, 1661, 1760,
112 1865, 1976, // octave 6
113         2093, 2217, 2349, 2489, 2637, 2794, 2960, 3136, 3322, 3520,
114 3729, 3951, // octave 7
115         4186, 4435, 4699, 4978
116                                                         // octave 8
117 };
118
119
120
```

```
// "도 도# 레 레# 미 파 파# 솔 솔# 라 라# 시" 순의 출력 처리용
character 배열
const char notes[] = {'c', 'C', 'd', 'D', 'e', 'f', 'F', 'g', 'G',
'a', 'A', 'b'}; // length = 12

// note에 해당하는 notes의 index를 반환한다.
int getNoteIndex(char note) {
        for(int i = 0; i < 12; i++){
                if(notes[i] == note) {
                        return i;
                }
        }
        return -1;
}

// note, octave의 출력으로 duration 동안 부저에 소리를 출력하는 함수
void playTone(char note, int octave, int duration){
        int frequency = frequencies[getNoteIndex(note) - 11 + octave *
12];
        tone(BUZZER_PIN, frequency, duration);
        delay(duration);
        noTone(BUZZER_PIN);
}

// 현재 스위치의 눌림 여부를 반환한다.
bool isSwitchActive() {
        return digitalRead(SWITCH_PIN) == LOW;
}

void loop() {
        if(isSwitchActive()){
                startGame();
        }
}

// 모든 LED의 상태를 설정한다.
void setAllLED(bool status) {
        for(int j = 0; j < NUM_OF_LED; j++) {
                digitalWrite(LED_PINS[j], status ? HIGH : LOW);
        }
}

// 빵빠레 음을 재생한다. (게임 승리시 재생)
```

```
164 void playFanfare() {
165         playTone('a', 5, 200);
166         playTone('a', 5, 100);
167         playTone('a', 5, 300);
168         playTone('b', 5, 200);
169         playTone('a', 5, 200);
170         playTone('b', 5, 200);
171         playTone('c', 6, 200);
172         playTone('c', 6, 100);
173         playTone('c', 6, 700);
174 }
175
176 void startGame() {
177         randomSeed(millis());
178         // 사용자가 버튼을 누른 시점의 millis() 함수 값을 이용하여
179 의사난수 시드를 초기화한다.
180
181         // 대기 및 시작
182         Display.setChars("REDY");
183         for(int i = 0; i < 3; i++) {
184                 playTone('g', 5, 200);
185                 setAllLED(true);
186                 delay(1000);
187                 setAllLED(false);
188                 delay(1000);
189         }
190         Display.setChars("STAT");
191         playTone('g', 7, 600);
192
193         // 본 게임 시작 1초 대기
194         setAllLED(true);
195         delay(1000);
196         setAllLED(false);
197
198         // 참참참 시작
199         Display.setChars("CHA1");
200         digitalWrite(LED2_PIN, HIGH);
201         playTone('f', 4, 250);
202         Display.blank();
203         digitalWrite(LED2_PIN, LOW);
204         delay(50);
205
206         Display.setChars("CHA2");
```

```
207         digitalWrite(LED2_PIN, HIGH);
208         playTone('f', 4, 250);
209         Display.blank();
210         digitalWrite(LED2_PIN, LOW);
211         delay(50);
212
213         // 조이스틱 방향 결정과 그에 따른 처리
214         int randomLR = random(2);
215         if(randomLR == 0) {
216                 Display.setChars("LEFT");
217                 digitalWrite(LED1_PIN, HIGH);
218         } else {
219                 Display.setChars("RIGT");
220                 digitalWrite(LED3_PIN, HIGH);
221         }
222         playTone('f', 6, 300);
223
224         // 사용자 입력 처리
225         long waitInput = millis();
226         bool selected = false;
227         int joy;
228         while(millis() - waitInput <= 10) {
229                 joy = GET_JOY_Y();
230                 if(abs(joy) > JOY_TRIGGER_VALUE) {
231                         selected = true;
232                 }
233         }
234         joy = joy < 0 ? 0 : 1;
235
          // 최종 결과 출력
          if(selected && joy == randomLR) {
                  Display.setChars("COOL");
                  playFanfare();
                  for(int k = 0; k < 3; k++){
                          setAllLED(true);
                          delay(500);
                          setAllLED(false);
                          delay(500);
                  }
          } else {
                  Display.setChars("LOSE");
                  digitalWrite(LED2_PIN, HIGH);
                  playTone('F', 6, 100);
```

```
                playTone('F', 6, 100);
                delay(2000);
        }

        // 시각 피드백 초기화
        Display.blank();
        setAllLED(false);
}
```