I2C LCD와 8x8 도트 매트릭스로 즐기는 스네이크 게임

19010671 김택천

10cheon00@sju.ac.kr

1. 목표

게임에 필요한 정보를 출력할 I2C LCD, 8x8 도트 매트릭스와, 사용자의 입력을 받을 버튼을 조합한 회로를 구현하고 스네이크 게임 로직을 작성하여 게임기를 개발한다.

2. 구현 과정

1) I2C LCD 및 도트 매트릭스 회로 구현

스네이크 게임에 등장하는 스네이크와 맵은 8x8 도트 매트릭스로 표현한다. 게임 중 발생한 메시지는 LCD로 표현한다.

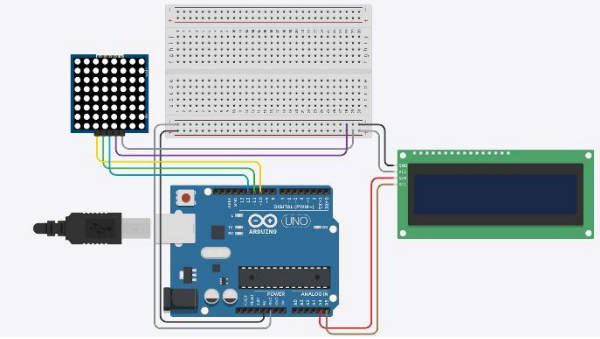

그림 1.1. LCD와 도트 매트릭스 아두이노 회로 구성

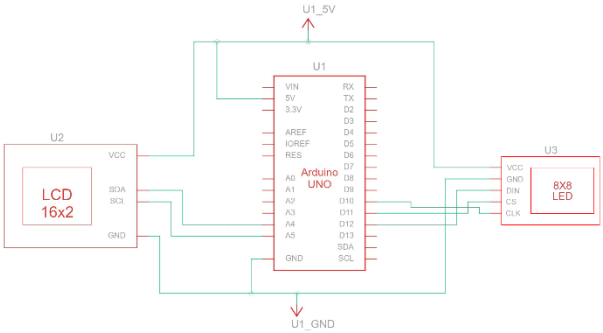

그림 1.2. 그림 1.1의 도식화

그림 1.1은 LCD와 도트 매트릭스를 아두이노에 연결하여 회로를 구성한 모습이고, 그림 1.2는 그림 1.1을 도식화한 것이다.

LCD의 전원과 접지를 연결한 후 SDA핀과 SCL핀을 각각 A4와 A5 아날로그 핀에 연결하기만 하면 프로그래밍을 통해 LCD의 액정에 원하는 텍스트를 보여줄 수 있게 된다.

도트 매트릭스의 경우 전원과 접지를 연결한 후 DIN핀을 데이터 핀인 12핀과 연결하고 CLK핀과 CS핀을 각각 PWM 핀인 10번 핀과 11번 핀에 연결하기만 하면 프로그래밍을 통해 도트 매트릭스의 LED를 켜고 끌 수 있다.

2) 버튼 회로 구현

스네이크 게임에서 사용자는 스네이크의 진행 방향을 시계 방향 또는 반시계 방향으로

꺾을 수 있다. 진행 방향을 바꾸는 요청은 버튼을 통해 입력 받게 한다.

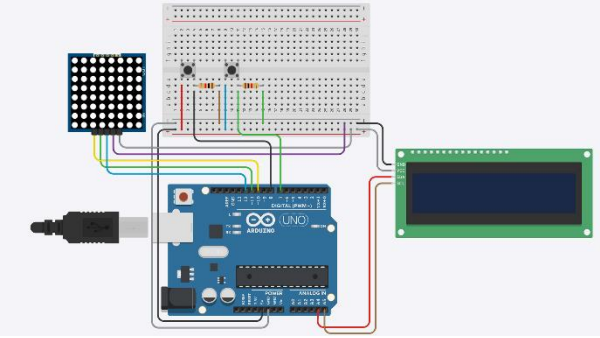

그림 2.1 그림 1.1의 구성에 버튼을 추가한 아두이노 회로 구성

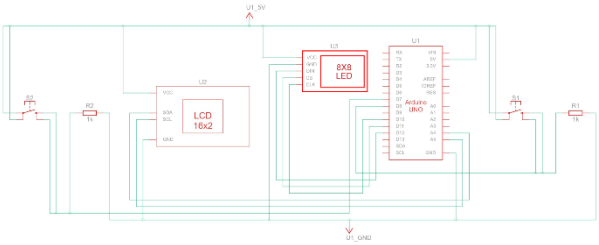

그림 2.2 그림 2.1의 회로의 도식화

그림 2.1은 그림 1.1에서 구현한 아두이노 회로에 버튼 회로를 이어서 구현한 모습이고, 그림 2.2는 그림 2.1을 도식화한 것이다. 버튼 두 개를 각각 7, 8번 핀에 연결하여 프로그래밍을 통해 매 루프마다 사용자의 입력을 받아 처리하도록 한다.

3) 게임 로직 프로그래밍

스네이크 게임의 규칙은 참고 문헌의 항목 [3]을 참고한다.

(1) 게임 초기화

아두이노의 전원이 들어오면 `Game` 클래스의 생성자를 통해 LCD와 도트 매트릭스를 초기화 하고 버튼과 연결된 핀을 입력모드로 설정한다. LCD와 도트 매트릭스를 다루기 위해 추가 라이브러리인 `LedControl`과 `Liquid Crystal I2C`를 설치한 후 `#include “LedControl.h”`, `#include “LiquidCrystal_I2C.h”` 전처리문을 수행하여 실행 코드에서 라이브러리를 사용한다. 이후 `Game::initialize()`함수를 통해 게임 로직을 초기화한다.

(2) 게임 준비

초기화가 종료되면 매 루프마다 실행되는 `Game::render()`함수를 통해 LCD와 도트 매트릭스를 항상 출력한다. LCD에는 `Snake Game`과 `Press any Btn`이 출력되고 도트 매트릭스에는 스네이크와 획득할 수 있는 포인트에 해당하는 LED가 점등된다. 사용자가 아무 버튼을 누르게 된다면 게임이 진행된다.

(3) 게임 진행

매 루프마다 `Game::update()`함수가 호출되어 스네이크가 이동하게 된다. 사용자가 버튼을 누르게 된다면 누른 버튼에 따라 스네이크의 이동 방향을 시계 방향 또는 반시계 방향으로 회전한다. 만약 스네이크가 포인트와 같은 좌표에서 만나게 된다면 점수가

올라가고 스네이크의 몸이 늘어난다. 만약 스네이크가 도트 매트릭스의 바깥으로 향하거나 자기 자신과 부딪치게 된다면 게임을 종료한다.

게임이 진행되는 동안 LCD에는 획득한 점수가 표시되고 획득한 점수에 비례하여 스네이크의 길이를 나타내는 문자열이 출력된다. 도트 매트릭스에는 움직이고 있는 스네이크를 나타내는 LED와 획득할 수 있는 포인트에 해당하는 LED가 점등된다. 이에 대한 코드는 `Game::update()`함수의 switch-case 블록에서 `case RUN:`의 하위에 작성되어 있다.

(4) 게임 종료

게임이 종료되면 LCD에는 `YOU DIED!`와 `Press any Btn`이 출력된다. 사용자가 아무 버튼을 누르면 다시 (2) 게임 준비 상태로 돌아가 `Game::initialize()`함수를 호출하여 게임을 초기화하고 사용자의 입력을 기다리게 된다.

3. 결과

사용자는 LCD에 출력된 텍스트를 통해 스네이크 게임을 시작하기 위해 버튼을 누르게 되고, 게임이 시작된 후에는 도트 매트릭스에 점등된 LED를 통해 스네이크의 진행 방향을 확인하게 된다. 사용자는 게임이 진행되는 동안 스네이크가 포인트로 향하도록 버튼을 눌러 방향을 조절하게 된다. 사용자가 획득한 포인트는 LCD에 표시된다. 종료 조건을 만족하는 상황이 될 경우 게임이 종료되고 LCD에 사용자가 패배했다는 텍스트가 출력된다. 사용자는 버튼을 눌러 게임을 다시 시작하게 된다.

4. 결론

회로 구현이 간단했으나 로직 구현이 매우 어려웠다. LCD 와 LED 를 다루기 위해 추가 라이브러리를 설치하는 과정을 통해 다른 모듈을 다룰 때 굉장히 수월함을 깨달았다.

게임 클래스를 별도로 작성함으로 아두이노 확장자 파일 속에 모든 로직을 넣지 않고 파일 분할을 통해 모듈화하여 개발할 수 있는 것을 확인했다.

첨부 #1 : 실행 코드

1) main.ino

```
#include "Game.h"

/*
1. 플레이어는 가운데에서 생성된다.
2. 생성된 플레이어의 길이는 2 다.
3. 플레이어는 항상 앞으로 전진한다.
4. 버튼을 누르면 플레이어의 진행 방향을 전환한다.
5. 시계방향 버튼을 누르면 시계방향으로 90 도 회전한다.
6. 반시계방향 버튼을 누르면 반시계방향으로 90 도 회전한다.
7. 맵에 생성된 포인트를 획득하면 플레이어의 길이가 1 늘어난다.
8. 플레이어가 맵 밖으로 나가거나 자기 자신과 부딪칠 경우 게임이 종료된다.
*/

Game* game;

void setup() {
  game = new Game();
}

unsigned long elapsedTime = millis();
int frame = 4;

void loop() {
  game->input();

  if (millis() - elapsedTime > (unsigned long)(1000.0 / (double)frame)){
    game->update();
```

```
        game->refreshInput();
        elapsedTime = millis();
    }
    game->render();

    delay(50);
}
```

2) Game.h

```
#ifndef __GAME__
#define __GAME__

#include "Coord.h"

#include "LiquidCrystal_I2C.h"
#include "LedControl.h"

class Game {
public:
    Game();
    void initialize();
    void refreshInput();
    void input();
    void update();
    void render();

private:
    bool isAnyBtnPressed();
    bool isLeftBtnPressed();
    bool isRightBtnPressed();
    bool isGameOver();
```

```
    void addBody(Coord coord);
    void createPoint();
    void moveBody();
    void rotateLeft();
    void rotateRight();
    void writePointToLcdStr();

private:
    static const int MAP_WIDTH = 8;
    static const int MAP_HEIGHT = 8;

    enum GameState { INIT, READY, RUN, END };
    GameState gameState;

    Coord bodyCoords[64];
    int bodyCount = 0;

    Coord directions[4] = {
        {0, -1}, // UP
        {1, 0},    // RIGHT
        {0, 1},    // DOWN
        {-1, 0}    // LEFT
    };
    int angle;

    Coord pointCoord;
    int point;
private:
    LiquidCrystal_I2C* lcd;
    LedControl* led;
    const static int PIN_SIZE = 14;
    const int PIN_LED_DIN = 12;
```

```
    const int PIN_LED_CS = 11;
    const int PIN_LED_CLK = 10;
    const uint8_t ADDRESS_LCD = 0x27u;
    const int PIN_LEFT_BTN = 8;
    const int PIN_RIGHT_BTN = 7;
    bool btnState[PIN_SIZE];
    char lcdStr[2][17];
};

#endif
```

3) Game.cpp

```
#include "Game.h"
#include <Arduino.h>
#include <stdlib.h>
#include <time.h>

Game::Game() {
    // initialize 8x8 dot matrix.
    led = new LedControl(PIN_LED_DIN, PIN_LED_CLK, PIN_LED_CS);
    led->shutdown(0, false);
    led->setIntensity(0, 8);
    led->clearDisplay(0);
    // initialize i2c lcd.
    lcd = new LiquidCrystal_I2C(ADDRESS_LCD, 16, 2);
    lcd->init();
    lcd->backlight();
    // initialize buttons.
    pinMode(PIN_LEFT_BTN, INPUT);
    pinMode(PIN_RIGHT_BTN, INPUT);
    // initialize other game logic.
    refreshInput();
```

```
    srand(time(NULL));
    gameState = INIT;
}

void Game::initialize() {
    angle = 0;
    bodyCount = 0;

    addBody(Coord(MAP_WIDTH / 2, MAP_HEIGHT / 2));
    addBody(Coord(MAP_WIDTH / 2, MAP_HEIGHT / 2 + 1));
    createPoint();
    point = 0;

    // set lcd texts.
    strcpy(lcdStr[0], "        Snake Game    ");
    strcpy(lcdStr[1], "    Press any Btn ");
}

void Game::refreshInput() {
    for(int i=0; i<PIN_SIZE; i++){
        btnState[i] = false;
    }
}

void Game::input() {
    btnState[PIN_LEFT_BTN] |= digitalRead(PIN_LEFT_BTN) == 1;
    btnState[PIN_RIGHT_BTN] |= digitalRead(PIN_RIGHT_BTN) == 1;
}

void Game::update() {
    switch (gameState) {
        case INIT:
```

```
    initialize();
    gameState = READY;
case READY:
    if (isAnyBtnPressed()) {
        gameState = RUN;
    }
    break;
case RUN:
    // run game logics.
    if (isGameOver()) {
        gameState = GameState::END;
        break;
    }
    if (bodyCoords[0] + directions[angle] == pointCoord) {
        addBody(pointCoord);
        createPoint();
        point++;
    }
    moveBody();
    if (isLeftBtnPressed()) {
        rotateLeft();
    }
    if (isRightBtnPressed()) {
        rotateRight();
    }

    // set lcd texts.
    writePointToLcdStr();
    break;
case END:
    strcpy(lcdStr[0], "       YOU DIED!      ");
    strcpy(lcdStr[1], "    Press any Btn ");
```

```
            if (isAnyBtnPressed()) {
                gameState = INIT;
            }
            break;
    }

}

void Game::moveBody() {
    for (int i = bodyCount - 1; i > 0; i--) {
        bodyCoords[i] = bodyCoords[i - 1];
    }
    bodyCoords[0] += directions[angle];
}

bool Game::isAnyBtnPressed() {
    return isLeftBtnPressed() || isRightBtnPressed();
}

bool Game::isLeftBtnPressed() {
    return btnState[PIN_LEFT_BTN];
}

bool Game::isRightBtnPressed() {
    return btnState[PIN_RIGHT_BTN];
}

void Game::rotateLeft() {
    angle = (angle + 1) % 4;
}

void Game::rotateRight() {
```

```
    angle = (angle + 3) % 4;
}

void Game::writePointToLcdStr() {
    strcpy(lcdStr[0], "Point :                ");
    strcpy(lcdStr[1], "                       ");

    int pointCopy = point, i = 15;
    while (pointCopy) {
        lcdStr[0][i--] = pointCopy % 10 + '0';
        pointCopy /= 10;
    }
    for (i = 0; i < point / 4; i++) {
        lcdStr[1][i] = '-';
    }
    lcdStr[1][i] = '*';
}

bool Game::isGameOver() {
    for (int i = 0; i < bodyCount; i++) {
        for (int j = i + 1; j < bodyCount; j++) {
            if (bodyCoords[i] == bodyCoords[j]) {
                return true;
            }
        }
    }
    int x = bodyCoords[0].x, y = bodyCoords[0].y;
    if (x < 0 || x >= Game::MAP_WIDTH || y < 0 || y >= Game::MAP_HEIGHT) {
        return true;
    }
    return false;
}
```

```
void Game::addBody(Coord coord) {
    for (int i = bodyCount++; i > 0; i--) {
        bodyCoords[i] = bodyCoords[i - 1];
    }
    bodyCoords[0].x = coord.x;
    bodyCoords[0].y = coord.y;
}

void Game::createPoint() {
    int coord[64], size = 0;
    for (int i = 0; i < 64; i++) {
        bool isExist = false;
        for(int j = 0; j < bodyCount; j++) {
            if (i == bodyCoords[j].x + bodyCoords[j].y * MAP_HEIGHT) {
                isExist = true;
            }
        }
        if (isExist) {
            continue;
        }
        coord[size++] = i;
    }
    int randomCoord = rand() % size;
    pointCoord.x = coord[randomCoord] % MAP_HEIGHT;
    pointCoord.y = coord[randomCoord] / MAP_HEIGHT;
}

void Game::render() {
    lcd->setCursor(0, 0);
    lcd->print(lcdStr[0]);
    lcd->setCursor(0, 1);
```

```
    lcd->print(lcdStr[1]);

    led->clearDisplay(0);
    for (int i = 0; i < bodyCount; i++) {
        led->setLed(0, bodyCoords[i].y, bodyCoords[i].x, true);
    }
    led->setLed(0, pointCoord.y, pointCoord.x, true);
}

void strcpy(char* dest, char* src) {
    while (*src) {
        *dest++ = *src;
    }
}
```

4) Coord.h

```
#ifndef __COORD__
#define __COORD__

struct Coord {
    Coord();
    Coord(int, int);

    int x;
    int y;
    bool operator==(const Coord &other) {
        return x == other.x && y == other.y;
    }
    Coord &operator+=(const Coord &other) {
        x += other.x;
        y += other.y;
        return *this;
```

```
    }
    Coord operator+(const Coord &other) {
        return Coord(x + other.x, y + other.y);
    }
};

#endif
```

5) Coord.cpp

```
#include "Coord.h"

Coord::Coord() {
    x = 0;
    y = 0;
}

Coord::Coord(int __x, int __y) {
    x = __x;
    y = __y;
}
```

참고 문헌

[1] [8X8 도트 매트릭스 제어하기], https://kocoafab.cc/tutorial/view/734, 2018-01-10 15:53:40

[2] [스케치-I2C LCD], https://kocoafab.cc/tutorial/view/733, 2018-01-10 14:48:19

[3] [14-4-바.snake], http://soen.kr/lecture/ccpp/cpp2/14-4-6.htm, 2023. 10. 12 검색