목차

1. 수행한 과제

Dot Matrix

- Dot Matrix, 74HC595, Switch를 사용하여 아두이노에 연결한다.
- 다음 조건을 만족하도록 프로그램하여 dot matrix를 동작시킨다.
  - Switch를 누르면 다음 내용을 차례로 화면에 표시한다.
    - 학번을 차례로 하나씩 표시한다.
    - 이모티콘 3개를 차례로 표시한다.
  - 표시간격은 1초이내로 한다.
  - 표시방법은 자유로 한다.
    - 예를 들면, 스크롤 업/다운, shift left/right, fade in/out, etc..

Character LCD

- Character LCD, Swtich를 아두이노와 연결한다.
- 다음 내용을 형식에 맞춰 LCD에 표시한다.

① Switch를 첫번째로 누르면 "학번과 영문이름"을 LCD의 좌상단에서부터 2줄로 연속적으로 표시한다. 2라인을 초과한 경우, 표시 가능한 부분까지만 표시한다.

② Switch를 2번째로 누르면 학번과 영문이름을 shift left 한다. 좌 상단에서부터 한 글자씩 사라지고, 우 하단에는 표시되지 않은 내용이나, '학번과 영문이름'을 다시 표시하기 시작한다.

③ Switch를 3번째로 누르면, shift left 동작이 정지한다.

④ Switch를 4번째로 누르면, 정지된 상태가 해제되어 화면 표시를 계속한다.

2. 부품 리스트

Dot Matrix

핵심 사용 부품

아두이노 우노 R3(Arduino Uno R3) x1

시프트 레지스터(Shift Register IC, 74HC595) x1

택트 스위치(Tact Switch) x1

기타 사용 부품

830 포인트 브레드보드(830 Point Breadboard) x1

점퍼 케이블(Jumper Cable)

Character LCD

핵심 사용 부품

아두이노 우노 R3(Arduino Uno R3) x1

16x2 문자형 LCD(16x2 Text LCD With Soldered Header Pin)

택트 스위치(Tact Switch) x1

기타 사용 부품

830 포인트 브레드보드(830 Point Breadboard) x1

10K 가변 저항(10K Variable Resistance) x1

220Ω 저항(220Ω Resistance) x1

점퍼 케이블(Jumper Cable)

3. 구현하고자 하는 기능 및 설명

Dot Matrix

Core – 1. Dot Matrix와 74HC595를 이용한 학번, 이모티콘 출력

아두이노에 Dot Matrix, 74HC595를 연결하고 학번과 이모티콘을 출력하는 기능을 구현하였다.

Idea – 1. Green, Red Dot Matrix Renderer 구현

부호없는 64비트 정수 2개로 Dot Matrix에 각 색깔별 출력 형태를 표시해주는 소프트웨어 폰트 렌더러를 구현하였다.

Idea – 2. Shift 연산을 이용한 Scroll Animation 구현

Shift 연산을 이용하여 각 Dot Matrix Image 전환간 스크롤 업/다운 애니메이션을 구현하였다.

```
DMImage previous = IMAGES[lastIndex];
DMImage current = IMAGES[index];
for (int i = 1; i < 8; i++) {
    long time = millis();
    while (millis() - time < ANIMATION_DELAY) {
        displayDMImage((previous >> (i * 8)) | (current << ((8 - i) *
8)), 0);
        // 8칸씩 Shifting 하여 상하로 이동된 이전, 현재 이미지를 만들고 OR(|)
연산하여 합성된 전환 애니메이션 이미지 상을 얻는다.
    }
}
```

Idea – 3. 자동 넘김 모드, 수동 넘김 모드 구현

매크로 플래그 AUTO_PLAY_MODE를 통해 버튼을 눌렀을 때 학번과 이모티콘을 자동으로 넘기는 모드 (자동 모드, true)와 누를 때마다 넘기는 모드 (수동 모드, false)를 구현하였다.

```
#define AUTO_PLAY_MODE true
```

Character LCD

Core – 1. Character LCD를 이용한 학번과 영문이름 출력

아두이노에 Character LCD와 스위치를 연결하여 버튼을 눌렀을 때 학번과 영문 이름이 Shift left하면서 나타날 수 있도록 구현하였다.

Idea – 1. 인터럽트를 이용한 스위치 제어

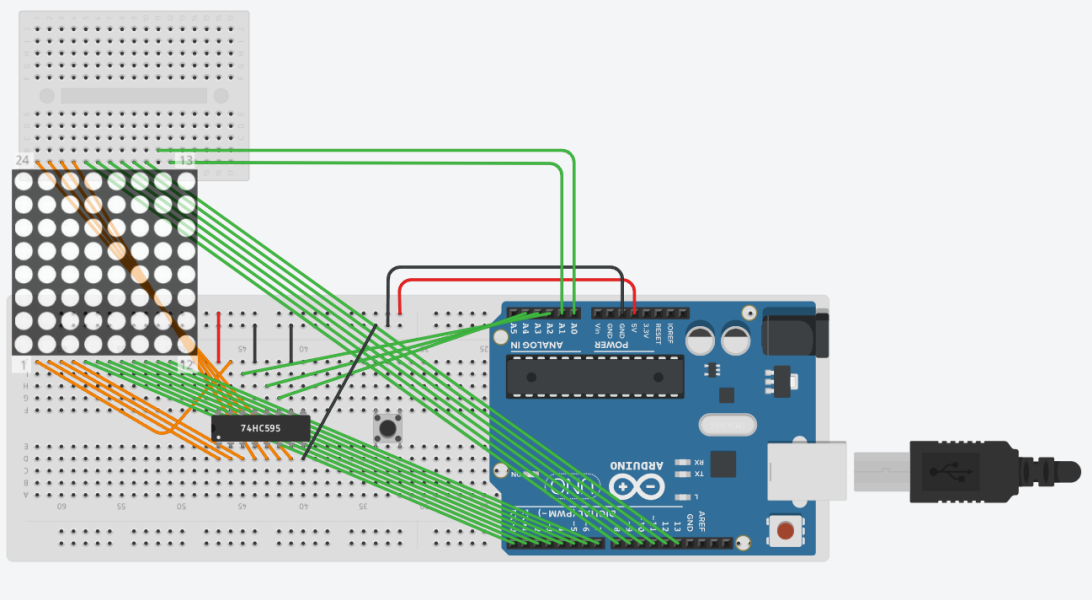
스위치 핀에 인터럽트를 설정하고, switchHandler 함수를 ISR(Interrupt Service Routine)로 사용하여 버튼 로직을 인터럽트를 사용하여 처리하였다.

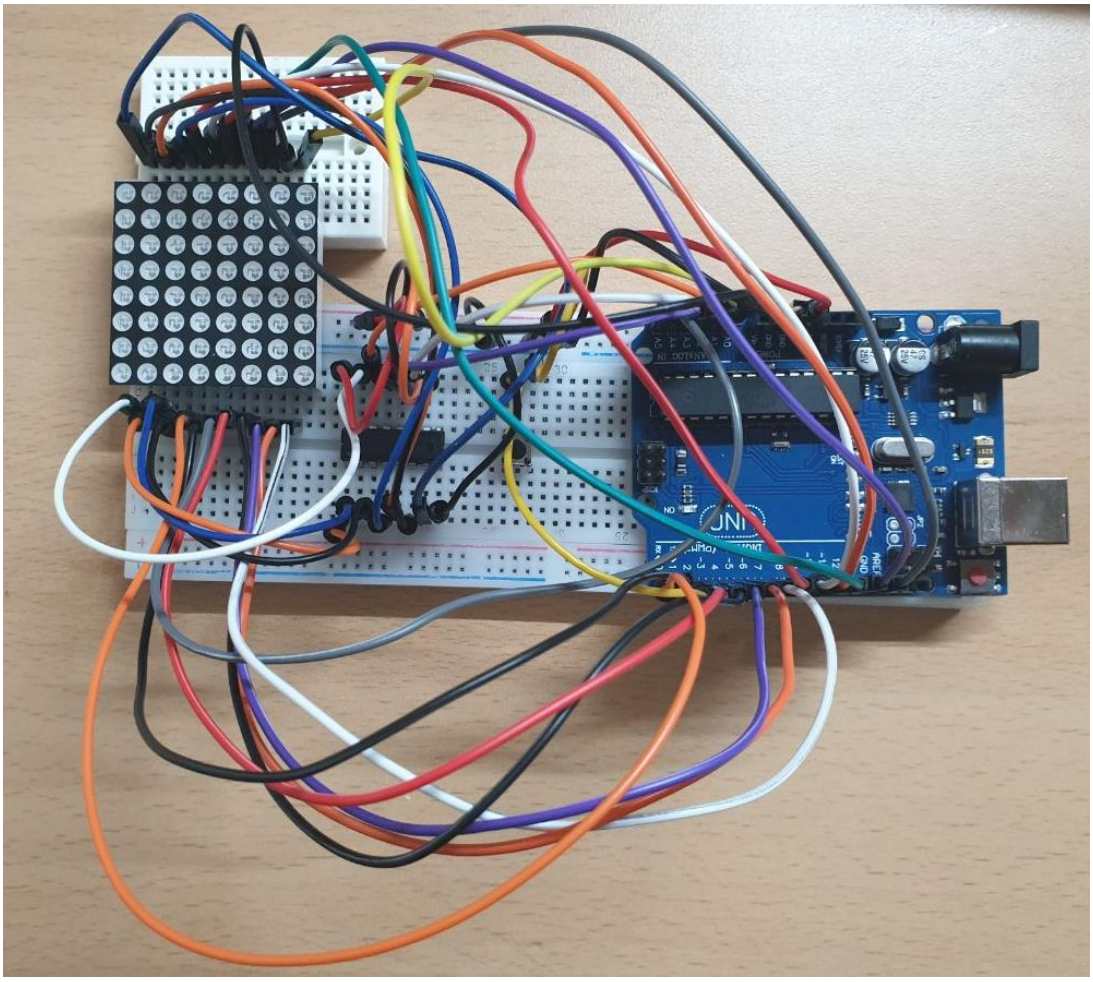
```
void setup() {
    pinMode(SWITCH_PIN, INPUT_PULLUP);
    attachInterrupt(digitalPinToInterrupt(SWITCH_PIN), switchHandler,
CHANGE);
    // 스위치 핀 모드, 인터럽트 설정
        // ……
}

// 스위치 상태를 확인하고, 스위치 카운트를 증가시킨다.
void switchHandler() {
    int switchStatus = !digitalRead(SWITCH_PIN);
    if (switchStatus && !isPressing) {
        isPressing = true;
        switchCount++;
```

```
    } else if (!switchStatus && isPressing) {
        isPressing = false;
    }
}
```

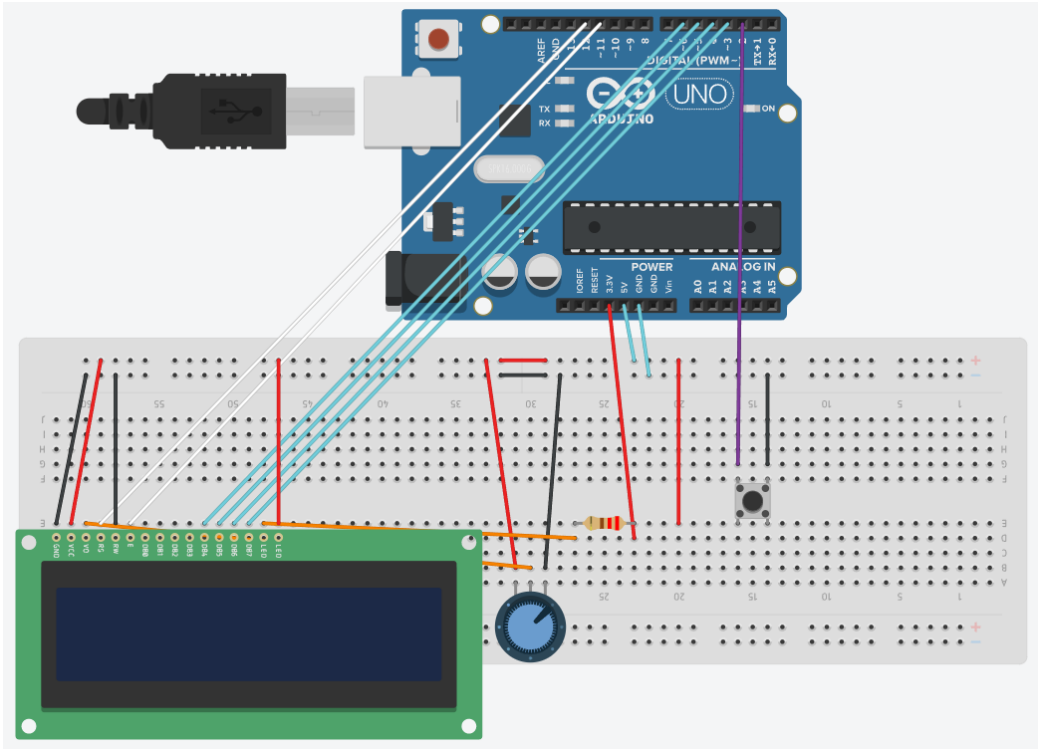
4. 부품과 아두이노 우노 보드와의 연결도

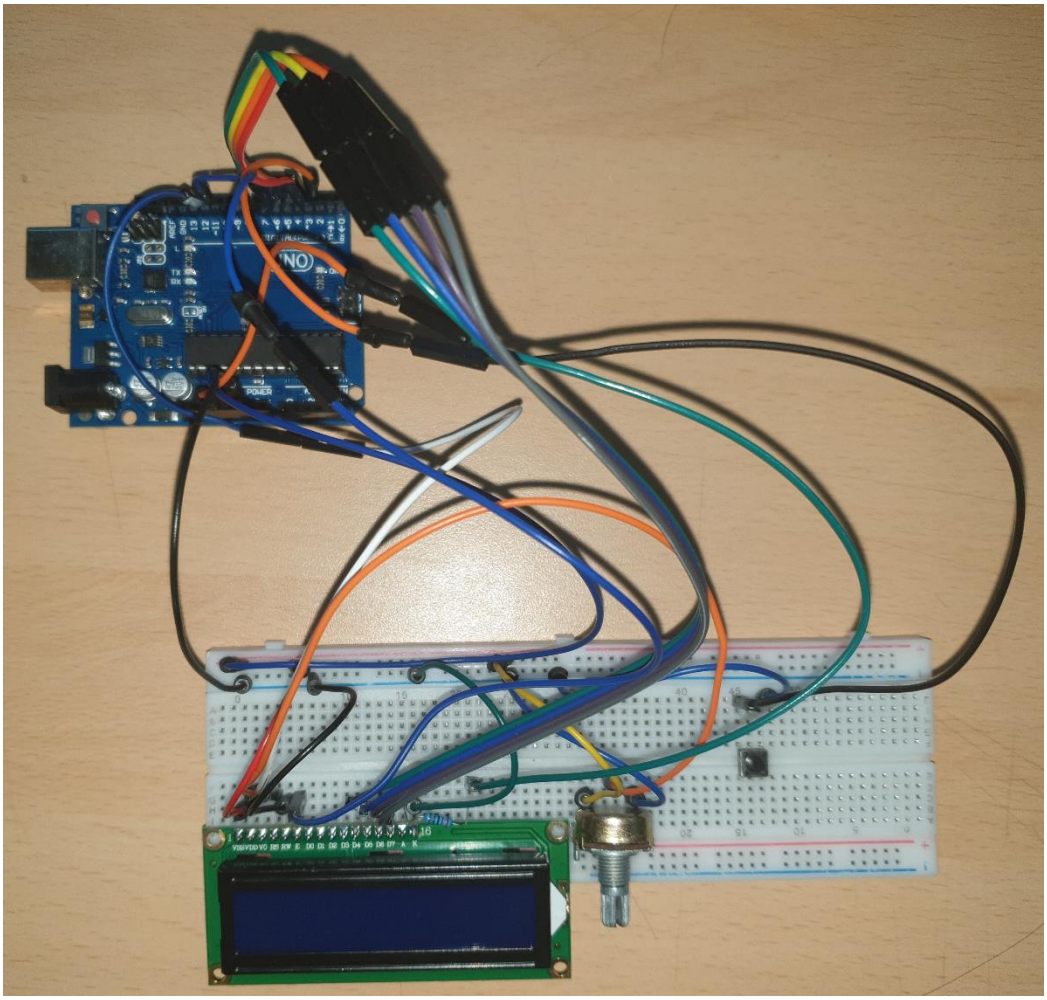
Dot Matrix





Character LCD





5. 프로그램 소스

GitHub: https://github.com/refracta/koreatech-assignment/tree/master/MicroprocessorNPractice/Assignment4

DotMatrix.ino

```
#define DS_PIN A2
#define ST_CP_PIN A3
#define SH_CP_PIN A4
#define SWITCH_PIN 2

#define NUM_OF_PIN 17
#define AUTO_PLAY_MODE true

// 빨강, 초록 도트 매트릭스 이미지 구조체
typedef struct RGDotMatrixImage {
    unsigned long long redImage;
    unsigned long long greenImage;
```

```
13
14      RGDotMatrixImage operator<<(const int shift) {
15          return {redImage << shift, greenImage << shift};
16      }
17
18      RGDotMatrixImage operator>>(const int shift) {
19          return {redImage >> shift, greenImage >> shift};
20      }
21
22      RGDotMatrixImage operator|(const RGDotMatrixImage &image) {
23          return {redImage | image.redImage, greenImage |
24 image.greenImage};
25      }
26 } DMImage;
27
28
29 #define NUM_OF_PIN 16
30 int PINS[NUM_OF_PIN] = {0, 1, A5, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,
31 A0, A1};
32 int GREEN_COLS[] = {PINS[0], PINS[1], PINS[2], PINS[3], PINS[4],
33 PINS[5], PINS[6], PINS[7]};
34 int RED_COLS[] = {PINS[8], PINS[9], PINS[10], PINS[11], PINS[12],
35 PINS[13], PINS[14], PINS[15]};
36
37 // 숫자 폰트
38 #define FONT_0 0x003e676f7b73633e
39 #define FONT_1 0x003f0c0c0c0c0e0c
40 #define FONT_2 0x003f33061c30331e
41 #define FONT_3 0x001E33301C30331E
42 #define FONT_4 0x0078307f33363c38
43 #define FONT_5 0x001e3330301f033f
44 #define FONT_6 0x001e33331f03061c
45 #define FONT_7 0x000c0c0c1830333f
46 #define FONT_8 0x001e33331e33331e
47 #define FONT_9 0x000e18303e33331e
48
49 #define NUM_OF_IMAGE 14
50 DMImage IMAGES[NUM_OF_IMAGE] =
51         {{0,                    0},
52          {FONT_2,               0},
53          {FONT_0,               0},
54          {FONT_2,  FONT_2},
55          {FONT_0,  FONT_0},
```

```
56          {0,       FONT_1},
57          {0,       FONT_3},
58          {~FONT_6,           0},
59          {~FONT_1,           ~FONT_1},
60          {~FONT_1,           0},
61          {~FONT_0, FONT_0},
62          {0x81423c0000000000, 0x0000008142244281},
63          {0x3c243c7edbdb7e3c, 0x3c243c7effff7e3c},
64          {0x3c42818181815a24, 0x0000182400240000}};
65
66 #define ANIMATION_DELAY 50
67 #define STOP_DELAY 600
68
69 // Shift register를 이용한 출력 조작
70 void shiftRegister(unsigned char data) {
71     digitalWrite(ST_CP_PIN, LOW);
72     for (int i = 0; i < 8; i++) {
73         digitalWrite(SH_CP_PIN, LOW);
74         digitalWrite(DS_PIN, (data & (0x80 >> i)) ? HIGH : LOW);
75         digitalWrite(SH_CP_PIN, HIGH);
76     }
77     digitalWrite(ST_CP_PIN, HIGH);
78 }
79
80 // Shift register를 이용하여 열을 컨트롤한다.
81 inline void controlRowPulse(unsigned char data) {
82     shiftRegister(0x80 >> data);
83 }
84
85 // DMImage를 디스플레이에 출력한다.
86 void displayDMImage(DMImage image, int illuminance) {
87     for (int r = 0; r < 8; r++) {
88         unsigned char redRow = (image.redImage >> r * 8) & 0xFF;
89         unsigned char greenRow = (image.greenImage >> r * 8) & 0xFF;
90
91         for (int c = 0; c < 8; c++) {
92             digitalWrite(RED_COLS[c], ((redRow >> c) & 1) ? LOW :
93 HIGH);
94             digitalWrite(GREEN_COLS[c], ((greenRow >> c) & 1) ? LOW :
95 HIGH);
96             controlRowPulse(r);
97             delay(illuminance);
98             controlRowPulse(-1);
```

```
        }
        for (int c = 0; c < 8; c++) {
            digitalWrite(RED_COLS[c], HIGH); // LED OFF
            digitalWrite(GREEN_COLS[c], HIGH); // LED OFF
        }
    }
}

// LED를 모두 지운다.
void clearAllLED() {
    for (int i = 0; i < 8; i++) {
        for (int j = 0; j < 8; j++) {
            digitalWrite(RED_COLS[j], HIGH);
            digitalWrite(GREEN_COLS[j], HIGH);
            controlRowPulse(i);
        }
    }
}


boolean isPressing = false;
int switchCount = 0;

// 스위치 상태를 확인하고, 스위치 카운트를 증가시킨다.
void switchHandler() {
    int switchStatus = !digitalRead(SWITCH_PIN);
    if (switchStatus && !isPressing) {
        isPressing = true;
        switchCount++;
    } else if (!switchStatus && isPressing) {
        isPressing = false;
    }
}

void setup() {
    for (int i = 0; i < NUM_OF_PIN; i++) {
        pinMode(PINS[i], OUTPUT);
    }
    // Red, Green 핀 설정
    pinMode(SH_CP_PIN, OUTPUT);
    pinMode(ST_CP_PIN, OUTPUT);
    pinMode(DS_PIN, OUTPUT);
    // Shift Register 핀 설정
```

```
    pinMode(SWITCH_PIN, INPUT_PULLUP);
    // 스위치 핀 설정
    attachInterrupt(digitalPinToInterrupt(SWITCH_PIN), switchHandler,
CHANGE);
    // 스위치 인터럽트 설정
    clearAllLED();
}

int lastIndex = 0;

void loop() {
    int index = (switchCount) % NUM_OF_IMAGE;
    if (switchCount != 0 && index != lastIndex) {
        DMImage previous = IMAGES[lastIndex];
        DMImage current = IMAGES[index];
        for (int i = 1; i < 8; i++) {
            long time = millis();
            while (millis() - time < ANIMATION_DELAY) {
                displayDMImage((previous >> (i * 8)) | (current << ((8
- i) * 8)), 0);
                // 8칸씩 Shifting 하여 상하로 이동된 이전, 현재
이미지를 만들고 OR(|) 연산하여 합성된 전환 애니메이션 이미지 상을
얻는다.
            }
        }
        lastIndex = index;
    } else {
        // 자동 재생 모드 처리
        if (AUTO_PLAY_MODE) {
            if (switchCount > 0 && switchCount % NUM_OF_IMAGE == 0) {
                switchCount = 0;
            } else if (switchCount != 0) {
                switchCount++;
            }
        }
        long time = millis();
        while (millis() - time < STOP_DELAY) {
            displayDMImage(IMAGES[lastIndex], 0);
        }
    }
}
```

CharacterLCD.ino

```
1  #include <Arduino.h>
2 #include <LiquidCrystal.h>
3 #include <string.h>
4
5 #define SWITCH_PIN 2
6 #define LC_D0_PIN 6
7 #define LC_D1_PIN 5
8 #define LC_D2_PIN 4
9 #define LC_D3_PIN 3
10 #define LC_ENABLE_PIN 11
11 #define LC_RS_PIN 12
12 #define SHIFT_DELAY 500
13
14 int switchCount = 0;
15 boolean isPressing = false;
16
17 // 스위치 상태를 확인하고, 스위치 카운트를 증가시킨다.
18 void switchHandler() {
19     int switchStatus = !digitalRead(SWITCH_PIN);
20     if (switchStatus && !isPressing) {
21         isPressing = true;
22         switchCount++;
23     } else if (!switchStatus && isPressing) {
24         isPressing = false;
25     }
26 }
27
28 // 스크린 글자 배열
29 char screen[16 * 2];
30 const char *TARGET_STRING = "refracta";
31 LiquidCrystal lcd(LC_RS_PIN, LC_ENABLE_PIN, LC_D0_PIN, LC_D1_PIN,
32 LC_D2_PIN, LC_D3_PIN);
33
34 // 스크린에 글자 배열을 출력
35 void display() {
36     lcd.setCursor(0, 0);
37     for (int i = 0; i < 16; i++) {
38         lcd.print(screen[i]);
39     }
40     lcd.setCursor(0, 1);
41     for (int i = 16; i < 32; i++) {
```

```
42          lcd.print(screen[i]);
43      }
44  }
45
46  // 글자를 왼쪽으로 한글자씩 밀기
47  void shiftLeft() {
48      char s0 = screen[0];
49      for (int i = 1; i < 16 * 2; ++i) {
50          screen[i - 1] = screen[i];
51      }
52      screen[16 * 2 - 1] = s0;
53  }
54
55  void setup() {
56      pinMode(SWITCH_PIN, INPUT_PULLUP);
57      attachInterrupt(digitalPinToInterrupt(SWITCH_PIN), switchHandler,
58  CHANGE);
59      // 스위치 핀 모드, 인터럽트 설정
60      memset(screen, ' ', 16 * 2);
61      // 글자 배열을 ' '로 초기화
62      strncpy(screen, TARGET_STRING, strlen(TARGET_STRING));
63      // 글자 배열에 TARGET_STRING을 복사
64      lcd.begin(16, 2);
65      lcd.noCursor();
66      lcd.noBlink();
67      // LCD 초기화
68      Serial.begin(9600);
69  }
70
71  void loop() {
72      if (switchCount == 1) {
73          display();
74      } else if (switchCount != 0 && switchCount % 2 == 0) {
75          display();
76          shiftLeft();
77          delay(SHIFT_DELAY);
        }
    }
```