전자회로실험 프로젝트

12181465 김재욱, 12161509최장호

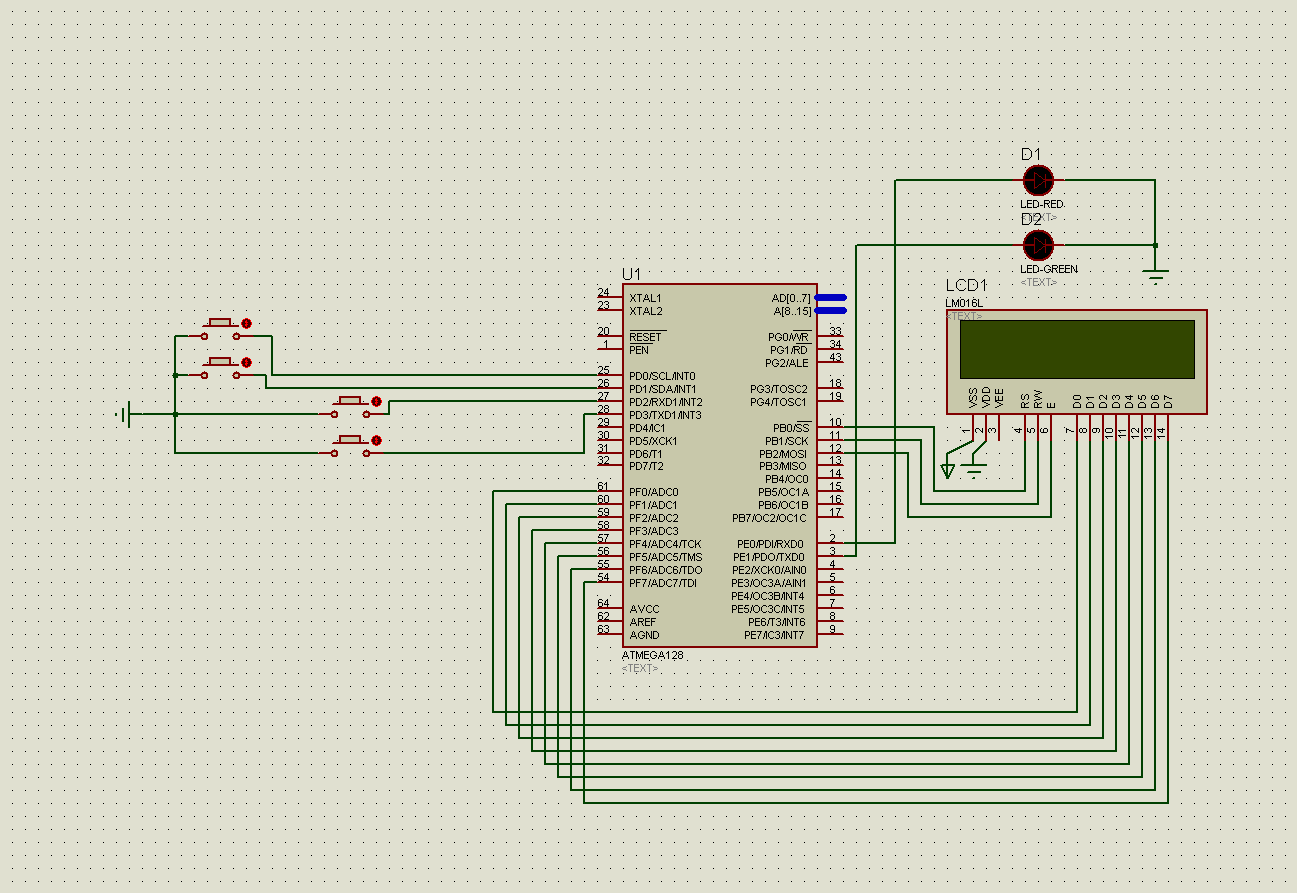
프로젝트 선정이유

대학교 근처 학생들은 대부분 학교 도서관, 스터디 카페나 일반 카페에서 공부를 한다. 그러나 코로나 19로 인한 사회적 거리두기에 의해 학교시설 사용이 통제되어 학생들이 학교근처 스터디 카페나 카페에 몰렸다. 이러한 이유뿐만 아니라 공용으로 사용되던 시설이 닫게 되면서 사설로 운영되는 곳을 이용하지만 공간내에서 사회적 거리두기가 지켜지지 않아 공용시설을 닫은 이유가 무색해졌다. 그래서 스터디 카페나 여느 카페에서 강제적으로 거리를 두게 하는 것이 필요하다고 생각했다.

도서관이나 스터디 카페처럼 자신의 자리를 선택하여 사용하는 시스템이 적용된 시설을 이용하는 사람들끼리 충분한 거리를 유지하여 사회적 거리두기를 실천할 수 있도록 시스템을 구현하고자 한다.

구현 목표는 궁극적으로 빈자리 중에서 내 양옆으로 사람을 두지 않기 위해 떨어져 앉아야 한다. 사회적 거리두기가 적용된 시스템 상에서 앉을 수 없는 자리를 선택할 수 없다.

프로테우스 회로구성



버튼 4개와 Atmega128 LCD 녹색 LED 1개 빨간색 LED 1개로 회로구성이 되어 있다.

LCD를 통해 출력되는 것은 이용가능한 좌석을 0, 이용하고있는 좌석을 1로 표시한다.

좌석은 왼쪽에 있는 버튼 4개를 이용하여 고를 수 있는데, 가장 위에 버튼은 좌측으로 이동하는 버튼이고 2번 버튼은 우측이로 이동, 그리고 3번 버튼은 좌석을 선택하는 버튼이고, 마지막 가장 아래 버튼은 좌석을 반납하는 버튼이다.

프로젝트 선정이유에도 나와 있듯이 누군가 사용중인 바로 옆자리를 사용 할 수 없게 설계하였다.

만약 바로 옆자리를 선택하게 된다면 빨간 LED가 ON이 되며 LCD에는 “BAD”라는 문자가 출력된다.

그리고 옆자리가 만약 비어있다면 녹색 LED가 ON이되며 LCD에는 “GOOD”이라고 출력된다.

네 번째 버튼을 눌러 반납을 할 때도 1이라고 표시되어있는 자리말고 다른자리에서 반납버튼을 누르게 되면 “BAD” 출력과 함께 빨간 LED가 ON이 된다.

정상적으로 반납을 하면 “BE RETURNED” 출력과 함께 녹색 LED가 ON상태가 된다.

소스코드

#define F\_CPU 16000000

#include <avr/io.h>

#include <util/delay.h>

#include <avr/interrupt.h>

#define delay 150

int index =15;

int dd[16]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}; //lcd에 표시될 자리의 수를 설정하기 위함

volatile char str1[1][16] = {{"000000000000000"}}; // 자리마다 디스플레이되는 초기값

volatile char str2[1][16] = {{"good "}}; // 문제가 없을 때 출력되는 안내문구

volatile char str3[1][16] = {{"bad "}}; // 문제가 있을 때 출력되는 안내문구

volatile char str4[1][16] = {{"be returned"}}; // 자리 반납 성공시 출력되는 안내문구

void cmd(int command){

PORTF = command;

PORTB = 0x04; // 데이터 통신 enable, lcd 위치 설정 명령

\_delay\_ms(1);

PORTB = 0x00; // 데이터 통신 disable

}

void data(char str){

PORTF = str;

PORTB = 0x05; // 데이터 통신 enable, 레지스터에 저장해놨던 위치에 값을 쓰는 명령

\_delay\_ms(1);

PORTB = 0x01; // 데이터 통신 disable

}

void display2(char\*str){

cmd(0x80|0x40); // lcd 첫 위치 (1,0)

int i = 0;

while (str[i] != '\0' )

{

data(str[i++]);

}

}

void display1(char\*str){

cmd(0x80); // lcd 첫 위치 (0,0)

int i = 0;

while (str[i] != '\0' )

{

data(str[i++]);

}

display2(str1[0]);

}

ISR(INT0\_vect){ //좌측으로 이동 버튼

cmd(0x10);// 커서를 왼쪽으로 움직임

index--;

\_delay\_ms(delay);

}

ISR(INT1\_vect){//우측으로 이동 버튼

cmd(0x14); //커서를 오른쪽으로 움직임

index++;

\_delay\_ms(delay);

}

ISR(INT2\_vect){//선택

if(str1[0][index]=='0' && dd[index]==0){ //해당 값이 0이고 lcd에서 선택된 출력값도 0이면 실행

str1[0][index] ='1'; // lcd에 디스플레이 되는 값을 1로 변경

dd[index-1] -= 1;

dd[index] -= 1;

dd[index+1] -= 1;

display1(str2[0]); //굿

PORTE=0b00000010; //초록불

}

else{

display1(str3[0]); //이미 선택된 자리 거나 그 주변에 앉지 못할때 bad출력

PORTE=0b00000001; //빨간불

}

\_delay\_ms(delay);

index =15; // 바뀐 자리 번호때문에 다시 lcd에 출력하게 되면 커서가 리셋됨으로 index도 리셋

}

ISR(INT3\_vect){// 반납

if(str1[0][index]=='1'&& dd[index]==-1){//해당 값이 -1이고 lcd에 선택된 출력값도 1이면 실행

str1[0][index] ='0';

dd[index-1] += 1;

dd[index] += 1;

dd[index+1] += 1;

display1(str4[0]);

PORTE=0b00000010; //초록불

\_delay\_ms(delay);

index=15;

}

else{ display1(str3[0]); // 반납이 불가능한 자리 선택 시

PORTE=0b00000001; //빨간불

\_delay\_ms(delay);

index=15; // 바뀐 자리 번호때문에 다시 lcd에 출력하게 되면 커서가 리셋됨으로 index도 리셋

}

}

int main(){

DDRB=0xff;

DDRF=0xff;

DDRD=0b00000000;

DDRE=0xff;

PORTD =0xff;

EICRA = 0b10000000; //인터럽트 트리거 방식 설정

EIMSK = 0b00001111; //인터럽트 허용 설정

SREG |= 0x80; //전체 인트럽트 허가

cmd(0x38); // 8bit 데이터 버스 사용

cmd(0x0D); // Display clear

cmd(0x0e); // Display on Cursor on

cmd(0x06); // Entry Mode

display1(str2[0]);

while(1){

}

}

위 코드에서 거리두기에 의해 사용불가능한 좌석과 이미 사용중인 좌석을 인식하는 알고리즘에 대해 설명한다. 먼저 커서의 위치를 나타내는 index 변수를 이용하여 각 자리의 상태를 나타내는 int형 15개의 배열 dd에 접근 할 수 있도록 한다.

좌 버튼을 누르면 커서의 위치 index는 감소하고 우 버튼을 누르면 커서의 위치 index는 증가한다.

자리선택 버튼을 누르면 해당 커서 위치와 양옆 위치의 배열이 -1이 된다.

자리반납 버튼을 누르면 해당 커서 위치와 양옆 위치의 배열이 +1이 된다.

자리 선택 버튼을 눌렀을때 해당 위치의 dd 값이 0이고 str1값이 0일때만 자리 선택이 가능하다.

자리 반납 버튼을 눌렀을때 해당 위치의 dd 값이 -1이고 str1값이 1일때만 반납 가능하다.

1. 프로그램 실행 직후

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| dd | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| dd 인덱스 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| strr1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

1. 5번,7번13번 자리 선택

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| dd | 0 | 0 | 0 | 0 | -1 | -1 | -2 | -1 | -1 | 0 | 0 | 0 | -1 | -1 | -1 |
| dd 인덱스 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| strr1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

1. 5번, 13번자리 반납

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| dd | 0 | 0 | 0 | 0 | 0 | 0 | -1 | -1 | -1 | 0 | 0 | 0 | 0 | 0 | 0 |
| dd 인덱스 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| strr1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

작동영상

첨부파일 참조

고찰

LCD의 구체적인 동작에 대한 고찰 이전에 기본 설정에 대한 옵션을 정리한다.

|  |  |
| --- | --- |
| 동작 | value |
| LCD 초기화 설정 (8bit, 화면 2행, 5x8 font) | 0x38 |
| 엔트리 모드 설정 (커서 우측 이동, 화면이동 없음) | 0x06 |
| 디스플레이 옵션 설정(화면 ON, 커서 ON) | 0x0E |

LCD 초기 설정이 끝난 이후 좌우 버튼을 이용해 커서를 움직여 자리를 선택 해야 하므로 다음 옵션을 사용한다.

커서는 마지막으로 쓰여진 행에서만 이동이 가능하며 전달받은 데이터를 LCD에 출력할 때마다 가장 끝에 커서가 위치하게 된다. 즉, 커서를 좌우로 이동시키기 위해서는 출력이 완료된 이후 다시 출력되기 전까지 좌,우 이동 값이 보장된다.

LCD 상에서 현재 위치를 표현하기 위해 커서 옵션을 사용하였다.

|  |  |
| --- | --- |
| 동작 | value |
| 커서 선택, 커서 좌측이동 | 0x10 |
| 커서 선택, 커서 우측이동 | 0x14 |
| 화면 선택, 화면 좌측이동 | 0x18 |
| 화면 선택, 화면 우측이동 | 0x1C |

커서를 두개의 행 사이에서 행과 열을 모두 바꾸려고 하였으나 옵션자체가 같은 행에서 좌,우로 움직이는 동작만 존재하여 하나의 행에서만 자리를 선택하도록 하였다. 또한 커서를 움직이는 동작에서 delay를 주지 않으면 사용자가 한번 버튼을 클릭했을지라도 인터럽트 감지 주기가 버튼을 누르는 동안 여러번 동작 되어 커서가 많이 움직이게 된다. 1ms delay를 주어도 충분히 이 문제를 해결 할 수 있다.

또한 LCD의 display를 초기화 하지 않은 상태에서 새로운 데이터를 쓰게 되면 이전의 데이터길이보다 짧은경우 이전 데이터가 남아 있게 되어 새로운 데이터 +이전 데이터 일부가 같이 출력된다. 이를 방지하기위해 가장긴 be returned 의 글자수에 맞춰 good과 bad의 char 배열을 공백으로 채워주었다.

추가로 초기 코드에서는 버튼 1,2,3은 작동 하였으나 4번은 실행되지 않았다.

문제는 풀업저항을 주지 않았기 때문이었다. 풀업저항을 설정하지 않아 floating 현상일 일어나 4 번이 실행되지 않았다고 예상하고 있다.