**/\* 소스코드 \*/**

#include *<stdio.h>*

#include *<string.h>*

#include *<wiringPi.h>*

#define LED0 14

#define LED1 26

#define LED2 27

#define LED3 24

#define LED4 23

#define LED5 21

#define LED6 22

*/\**

*\* @@(0)*

*\* @(5) @(1)*

*\* @ @*

*\* @@(6)*

*\* @(4) @(2)*

*\* @ @*

*\* @@(3)*

*\*/*

char char\_to\_dec(char);

int main(int argc, char\* argv[])

{

**if** (argc != 2 && argc != 3)

{

printf("report 1**\n**");

printf("report 2 [hex]**\n**");

**return** 0;

}

printf("Blinktest!**\n**");

wiringPiSetup();

pinMode(LED0, OUTPUT);

pinMode(LED1, OUTPUT);

pinMode(LED2, OUTPUT);

pinMode(LED3, OUTPUT);

pinMode(LED4, OUTPUT);

pinMode(LED5, OUTPUT);

pinMode(LED6, OUTPUT);

**if**(strcmp(argv[1], "1") == 0)

{

char code[16] = {63, 6, 91, 79 ,102, 109, 125, 39, 127, 111,

119, 124, 57, 94, 121, 71};

int i;

**for** (i = 0; i < 16; i++)

{

digitalWrite(LED0, (code[i] & 0x1) == 0x1);

digitalWrite(LED1, (code[i] & 0x2) == 0x2);

digitalWrite(LED2, (code[i] & 0x4) == 0x4);

digitalWrite(LED3, (code[i] & 0x8) == 0x8);

digitalWrite(LED4, (code[i] & 0x10) == 0x10);

digitalWrite(LED5, (code[i] & 0x20) == 0x20);

digitalWrite(LED6, (code[i] & 0x40) == 0x40);

delay(500);

char bit = 0x40;

int j;

**for** (j = 0; j < 7; j++)

{

printf("%d", (code[i] & bit) == bit);

bit = bit >> 1;

}

printf(" : %d**\n**", code[i]);

}

}

**else** **if**(strcmp(argv[1], "2") == 0)

{

**if** (argc != 3 ||

strlen(argv[2])>4 ||

argv[2][0] != '0' ||

argv[2][1] != 'x' ||

char\_to\_dec(argv[2][2]) == 255 ||

char\_to\_dec(argv[2][3]) == 255 && argv[2][3] != '\0')

printf("argument2 must be hex [0x0 ~ 0xff] : %s**\n**", argv[2]);

**else**

{

int j;

char code;

char bit = 0x40;

code = char\_to\_dec(argv[2][2]);

**if**(argv[2][3] != '\0')

{

code = code << 4;

code = code + char\_to\_dec(argv[2][3]);

}

**for** (j = 0; j < 7; j++)

{

printf("%d", (code & bit) == bit);

bit = bit >> 1;

}

printf(" : %d**\n**", code);

digitalWrite(LED0, (code & 0x1) == 0x1);

digitalWrite(LED1, (code & 0x2) == 0x2);

digitalWrite(LED2, (code & 0x4) == 0x4);

digitalWrite(LED3, (code & 0x8) == 0x8);

digitalWrite(LED4, (code & 0x10) == 0x10);

digitalWrite(LED5, (code & 0x20) == 0x20);

digitalWrite(LED6, (code & 0x40) == 0x40);

delay(60000);

}

}

**else**

printf("unknown option : %s %s**\n**", argv[1], argv[2]);

digitalWrite(LED0, LOW);

digitalWrite(LED1, LOW);

digitalWrite(LED2, LOW);

digitalWrite(LED3, LOW);

digitalWrite(LED4, LOW);

digitalWrite(LED5, LOW);

digitalWrite(LED6, LOW);

printf("off**\n**");

**return** 0;

}

char char\_to\_dec(char c)

{

**if** (c >= 48 && c <= 57)

c = c - 48;

**else** **if** (c >= 65 && c <= 70)

c = c - 55;

**else** **if** (c >= 97 && c <= 102)

c = c - 87;

**else**

c = 255;

**return** c;

}

**/\* 소스코드 구조 \*/**

int main(int argc, char\* argv[])

{

**if** (**option을 잘못 입력했을 때**)

{

**명령어 안내 출력**

}

**if**(**option이 1일 때**)

{

**0 ~ f의 7-segment 코드 출력**

**}**

**else** **if**(**option이 2일 때**)

{

**입력받은 16진수 출력**

**}**

**else**

**option이 잘못입력 됨을 출력**

**}**

char char\_to\_dec(char c)

{

**0 ~ F의 문자를 10진수로 변환하여 반환**

**}**

**/\* 문자를 10진수로 변환하는 함수 \*/**

char char\_to\_dec(char c)

{

**if** (c >= 48 && c <= 57) **// 0 ~ 9**

c = c - 48;

**else** **if** (c >= 65 && c <= 70) **// A ~ F**

c = c - 55;

**else** **if** (c >= 97 && c <= 102) **// a ~ f**

c = c - 87;

**else**

c = 255;

**return** c;

}

**/\***

**문자 하나를 전달받아 10진수로 변환하여 반환하는 함수입니다.**

**문자의 아스키코드를 검사하고 이에 적절한 정수를 빼주어 10진수로 변환합니다.**

**알파벳의 경우 대소문자 모두 변환할 수 있습니다.**

**단, 16진수 이외의 문자를 전달받으면 오류코드로 255를 반환합니다.**

**\*/**

**/\***

**첫번째 옵션으로 1을 입력받았을 때**

**example) ./report 1**

**\*/**

**if**(strcmp(argv[1], "1") == 0) **// option의 문자열이 1인지 검사**

{

char code[16] = {63, 6, 91, 79 ,102, 109, 125, 39, 127, 111,

119, 124, 57, 94, 121, 71};

int i;

**for** (i = 0; i < 16; i++)

{

digitalWrite(LED0, (code[i] & 0x1) == 0x1); **// LED0 점멸**

digitalWrite(LED1, (code[i] & 0x2) == 0x2); **// LED1 점멸**

digitalWrite(LED2, (code[i] & 0x4) == 0x4); **// LED2 점멸**

digitalWrite(LED3, (code[i] & 0x8) == 0x8); **// LED3 점멸**

digitalWrite(LED4, (code[i] & 0x10) == 0x10); **// LED4 점멸**

digitalWrite(LED5, (code[i] & 0x20) == 0x20); **// LED5 점멸**

digitalWrite(LED6, (code[i] & 0x40) == 0x40); **// LED6 점멸**

delay(500); **// 숫자마다 0.5초의 딜레이를 가짐**

char bit = 0x40;

int j;

**for** (j = 0; j < 7; j++) **// 현재 LED출력을 화면으로 보여줌.**

{

printf("%d", (code[i] & bit) == bit);

bit = bit >> 1;

}

printf(" : %d**\n**", code[i]); **// example) 01111111 : 63**

}

}

**/\***

**0 ~ F각각에 해당하는 7-segment코드를 배열에 정수형태로 저장해놓고 반복문으로 순차적으로 출력할 수 있게 하였습니다.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| i | 7-segment | decimal | i | 7-segment | decimal |
| 0000(0) | 0111111 | 63 | 1000(8) | 1111111 | 127 |
| 0001(1) | 0000110 | 6 | 1001(9) | 1101111 | 111 |
| 0010(2) | 1011011 | 91 | 1010(A) | 1110111 | 119 |
| 0011(3) | 1001111 | 79 | 1011(B) | 1111100 | 124 |
| 0100(4) | 1100110 | 102 | 1100(C) | 0111001 | 57 |
| 0101(5) | 1101101 | 109 | 1101(D) | 1011110 | 94 |
| 0110(6) | 1111101 | 125 | 1110(E) | 1111001 | 121 |
| 0111(7) | 0100111 | 39 | 1111(F) | 1000111 | 71 |

**비트가 0이면 꺼지고 1이면 켜지게 해야하므로 7-segment의 각 비트를**

**0x1(0000 0001), 0x8(0000 1000) 같이 비트가 하나만 1인 16진수로 각각 검사합니다.**

**example) LED0 : (0111111 && 0000001) == 0000001 => 1**

**LED1 : (0111111 && 0000010) == 0000010 => 1**

**. . .**

**LED6 : (0111111 && 1000000) == 1000000 => 0**

**HIGH가 1이고 LOW가 0이므로 1이면 LED가 켜지고 0이면 LED가 꺼질것입니다.**

**\*/**

**/\***

**첫번째 옵션으로 2를 입력받았을 때**

**example) ./report 2 0x4f**

**\*/**

**else** **if**(strcmp(argv[1], "2") == 0) **// option의 문자열이 2인지 검사**

{

**if** (argc != 3 || **// 16진수를 입력받지 않았거나**

strlen(argv[2])>4 || **// 3자리 이상의 16진수를 입력받았거나**

argv[2][0] != '0' ||

argv[2][1] != 'x' || **// 제대로 된 16진수를 입력받지 않았으면**

char\_to\_dec(argv[2][2]) == 255 || **// 해당 기능이 실행되지 않음**

char\_to\_dec(argv[2][3]) == 255 && argv[2][3] != '\0')

printf("argument2 must be hex [0x0 ~ 0xff] : %s**\n**", argv[2]);

**else**

{

int j;

char code;

char bit = 0x40; **// 1000 0000**

code = char\_to\_dec(argv[2][2]); **// 16진수 뒷자리를 10진수로 저장**

**if**(argv[2][3] != '\0') **// 입력받은 16진수가 두자리일 경우**

{

code = code << 4; **// 기존 정수를 4비트 왼쪽으로 쉬프트**

code = code + char\_to\_dec(argv[2][3]); **// 앞자리 16진수 저장**

}

**for** (j = 0; j < 7; j++) **// 현재 LED출력을 화면으로 보여줌**

{

printf("%d", (code & bit) == bit);

bit = bit >> 1;

}

printf(" : %d**\n**", code); **// example) 0101101 : 45**

digitalWrite(LED0, (code & 0x1) == 0x1);

digitalWrite(LED1, (code & 0x2) == 0x2);

digitalWrite(LED2, (code & 0x4) == 0x4);

digitalWrite(LED3, (code & 0x8) == 0x8);

digitalWrite(LED4, (code & 0x10) == 0x10);

digitalWrite(LED5, (code & 0x20) == 0x20);

digitalWrite(LED6, (code & 0x40) == 0x40);

delay(60000); **// 출력을 1분간 유지**

}

}

**/\***

**16진수를 출력하기 전에 제대로 된 16진수를 입력했는지 먼저 검사합니다.**

**1. 아무것도 입력받지 못했을 경우 example) ./report 2**

**2. 3자리 이상의 16진수를 입력받았을 경우 example) ./report 2 0x77f**

**3. 제대로 된 16진수를 입력받지 못했을 경우 example) ./report 2 38**

**example) ./report 2 0xrt**

**위의 사항에 해당되면 기능이 실행되지 않습니다.**

**\*/**

**/\***

**두자리가 아닌 한자리 숫자의 16진수를 입력받아도 동작할 수 있습니다.**

**example) 0x0f => 0xf**

**유효한 16진수를 입력받았으면 다음과 같은 과정을 거칩니다.**

**1. 뒷자리를 10진수로 변환하여 저장한다. example) 0x4f, num = 4**

**2. 만약 앞자리가 존재하면 이전에 저장한 정수의 비트를 왼쪽으로 4칸 시프트하고 앞자리를 10진수로 변환하여 기존 정수에 더한다. example) 0x4f, num = 64 + 15**

**3. 이후 option 1에서의 출력과정처럼 결과의 비트를 각각 검사하여 LED에서 출력한다.**

**\* LED 출력은 1분간 지속됩니다.**

**\*/**