**TFT LCD Project (Lab3)**

학과 : 전자공학과

학번 : 201501149

성명 : 이성범

**목차**

**1. Project(과제) 설명**

1.1 서론

1.2 본론

1.3 결론

**2. Project Flow**

**3. Algorithm**

3.1 start

3.2 cursor control

3.3 print

3.4 delete

**4. Exception Handling**

4.1 Out of range Handle

4.2 Push several buttons at same time

4.3 Print character

**1. Project(과제) 설명**

Odroid-C1 board를 기반으로 Swith를 이용해서 TFT LCD를 제어하는 프로그램

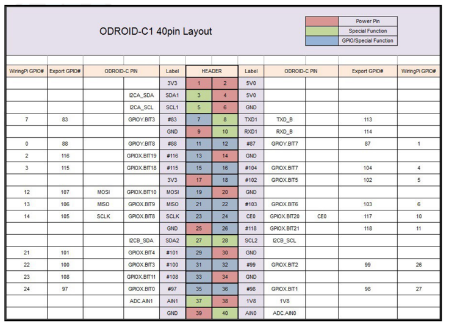
- 개발환경은 리눅스로 minicom을 통해서 개발한다.

- WiringPi의 기본 라이브러리로 작성한다.

- 외부 스위치로 알파벳 문자와 ‘ . ‘ 을 제어한다.

- TFT LCD는 오작동이 없도록 구현되어야 하고 일반 편집 프로그램에서 insert 모드로 구현이 되어있어야 한다. ( 즉, 커서가 있는 위치에 글씨가 써지고 커서가 움직이기 위해서는 방향키를 이용해야 한다. )

1.1 서론

- Odroid Pin map과 wiringPi 에서 제공해주는 Pin map을 보고 매칭시켜서 Hardware 설계를 한다.

- Hardware 설계 후 기본적인 헤더파일 선언과 핀매핑을 define을 통해서 다루기 쉽게 변수를 선언한다.

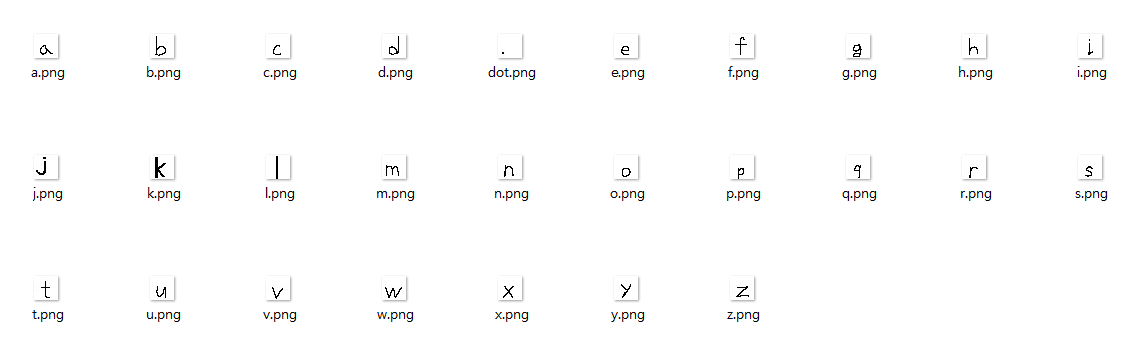
- 다음 코드를 통해 SPI 통신을 시작한다.

$ sudo modprobe spicc

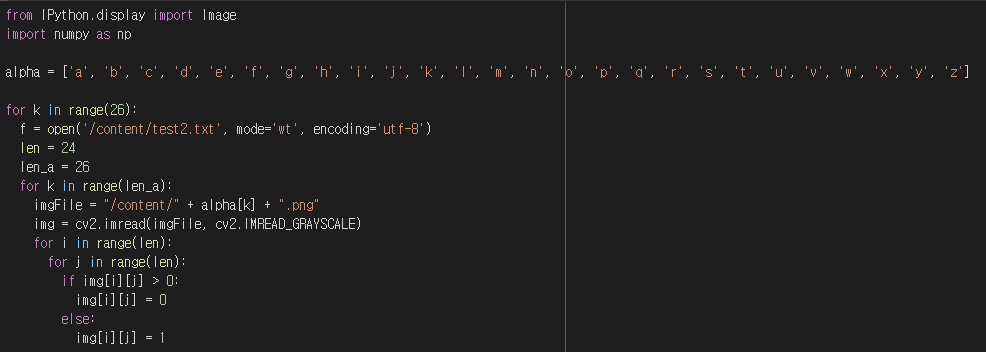
$ sudo modprobe fbtft\_device name=odroidc\_tft32 rotate=270 gpios=reset:116,dc:115 speed=32000000 cs=0

- TFT LCD를 사용하기 위해서 핀 설정을 하고 안정성을 위해서 Ground 핀을 두개 연결한다.

- 사용할 알파벳 이미지를 그린다.



- opencv 를 이용해서 간단하게 24 \* 24 행렬 값을 받아온다.



1.2 본론.

- Print, Delete, Cursor의 움직임 모두 Cursor의 위치를 기준으로 작동시킨다.

- 각 문자의 좌표를 24 \* 24 글자의 왼쪽 상단 즉, 글자를 기준으로 봤을때 (0, 0)으로 잡는다. 이 좌표를 언제나 현재 위치의 좌표로 설계하였습니다.

- 모든 알파벳과 ‘ . ‘ 을 24 \* 24행렬로 선언하고 구해놓은 행렬 값을 대입한다.

- 커서는 해당 위치에서 blinking을 하고 있고, swich가 눌리면 인터럽트가 발생하여서 빠져나온 후 그에 맞는 코드를 실행한다.

- 문자를 출력할 때는 **mmap** 방식으로 출력을 한다.

- 외부 스위치는 문자를 출력하는데 각 스위치는 눌릴 때마다 3개의 값이 있고 그 값들이 순환할 수 있게 idx를 더해주고 mod 3를 통해서 해결한다.

- Delete를 할 경우에 커서 위치의 문자는 지워지면서 그 뒤로 있는 모든 픽셀 값들은 한 칸 ( 28 \* 24 ) 만큼 당겨지고, 행 가장 앞에 있는 문자는 행을 올려주면서 가장 뒤의 열로 보낸다.

- 모든 값을 쉽게 이동시키거나 제어하기 위해서는 total이라는 240 \* 320 행렬이 TFT LCD에 적혀있는 모든 픽셀 값들을 계속해서 저장한다.

- Cursor의 이동은 left, right 함수를 통해서 제어한다. 마지막 위치에 커서가 위치했을 경우에right 버튼을 누를 경우에는 커서가 처음으로 이동하고, 처음 위치에서 커서가 위치했을 경우에 left 버튼을 누를 경우에는 커서가 마지막으로 이동한다.

1.3 결론

- TFT LCD가 동작을 하게되면 커서가 깜빡거린다.

- 외부 스위치 Button을 누를 때는 문자가 출력된다. 스위치의 동작은 a,b,c 버튼을 누를 경우에 a -> b -> c -> a -> b -> c 가 되도록 설계되었다.

- 8 line 까지 문자를 입력하게 되면 글씨가 짤리기 때문에 그것을 방지하기 위해서, 7 line의 마지막에서 다음으로 넘어가면 처음으로 커서가 이동하기 때문에 overflow가 발생할 수 없다.

- 이 TFT LCD는 insert mode로 되어있는 편집기로 커서의 움직임이나 글자의 출력으로 순환을통해서 Overflow가 발생할 수 없고 원하는 글자(문자열)를 출력할 수 있는 제품입니다.

**2. Project Flow**

|  |
| --- |
| 1. 헤더파일 선언, Switch Pin mapping |
| 2. 문자 이미지 행렬 가져오기 |
| 3. 스위치 눌릴 때 print |
| 4. Cursor Left , Right , 문자 Delete |
| 5. main 함수 |

1. 헤더파일 선언, (LCD, Switch) Pin mapping

#include <stdio.h>

#include <stdlib.h> /\* for exit \*/

#include <string.h>

#include <unistd.h> /\* for open/close .. \*/

#include <fcntl.h> /\* for O\_RDWR \*/

#include <sys/ioctl.h> /\* for ioctl \*/

#include <sys/mman.h> /\* for mmap \*/

#include <linux/fb.h> /\* for fb\_var\_screeninfo, FBIOGET\_VSCREENINFO \*/

#include <wiringPi.h> /\* for switch \*/

#define BT1 24 #define BT2 23 #define BT3 27 #define BT4 22 #define BT5 21

#define BT6 26 #define BT7 7 #define BT8 6 #define BT9 11 #define LEFT 1

#define RIGHT 5 #define DEL 4 -> Pin mapping 세팅

1. 문자 이미지 행렬 가져오기

예) a, b

int a[24][24] = {

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,1,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,1,1,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,1,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,1,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,1,0,0,0,0,0,1,0,1,1,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,1,1,0,0,0,1,1,0,0,1,1,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,1,1,1,0,0,0,0,0},

{0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0} };

int b[24][24] =

{ {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,1,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,1,0,1,1,0,0,0,1,1,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,1,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,1,1,1,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0} };

1. 스위치 눌릴 때 print

if (digitalRead(BT1) == 1)

{

if (idx1 % 3 == 0) print(dot, cur.x, cur.y, 0);

if (idx1 % 3 == 1) print(q, cur.x, cur.y, 0);

if (idx1 % 3 == 2) print(z, cur.x, cur.y, 0);

delay(300);

idx1++;

idx2 = idx3 = idx4 = idx5 = idx6 = idx7 = idx8 = idx9 = 0;

}

-> switch를 누를 경우에 print 함수를 실행하고 idx mod 3를 이용해서 사용자가 원하는 문자를 출력하게 한다.

- Print 함수의 일부 -

for (i = x; i < x + 24; i++)

{

for (j = y; j < y + 24; j++)

{

offset = i \* fbvar.xres + j;

pixel = makepixel(0, 0, 0);

total[i][j] = 0;

\*(pfbdata + offset) = pixel;

}

}

for (i = x; i < x + 24; i++)

{

for (j = y; j < y + 24; j++)

{

offset = i \* fbvar.xres + j;

if (alpha[i - cur.x][j - cur.y] > 0)

{

pixel = makepixel(255, 255, 255);

total[i][j] = 1;

}

else {

pixel = makepixel(0, 0, 0);

total[i][j] = 0;

}

\*(pfbdata + offset) = pixel;

}

}

-> 첫번째 for문은 중복하여 글자가 써지는 것을 방지하기 위해서 지우고 점을 찍는 것입니다.

-> 두번째 for문은 만들어 놓은 알파벳 행렬에 맞게 1일 경우 흰색 점, 0일 경우 검정색 점을 찍습니다.

1. Cursor Left , Right , 문자 Delete

void left()

{

if (cur.x == 0 && cur.y == 0) // 맨 처음 상황

{

cur.x = 192;

cur.y = 252;

}

else if (cur.y == 0) // 위에 줄이 있을 때 맨 앞에 있는 경우

{

cur.y = 252;

cur.x -= 32;

}

else { // 왼쪽으로 갈 수 있는 상황

cur.y -= 28;

}

idx1 = idx2 = idx3 = idx4 = idx5 = idx6 = idx7 = idx8 = idx9 = 0;

}

-> left 함수는 y값을 28씩 왼쪽으로 이동시킵니다. 만약, 왼쪽 끝일 경우 1행을 위로 올려서 왼쪽 오른쪽 끝에 위치 시킵니다.

void right()

{

if (cur.y == 252 && cur.x == 192) // 맨 마지막 상황

{

cur.x = 0;

cur.y = 0;

}

else if (cur.y == 252) // 아랫 줄로 좌표를 내려준다

{

cur.y = 0;

cur.x += 32;

}

else {

cur.y += 28;

}

idx1 = idx2 = idx3 = idx4 = idx5 = idx6 = idx7 = idx8 = idx9 = 0;

}

-> right 함수는 y값을 28씩 오른쪽으로 이동시킵니다. 만약, 오른쪽 끝일 경우 1행을 아래로 내려서 왼쪽 끝에 위치 시킵니다.

1. main 함수

int main()

{

int i, j;

idx1 = idx2 = idx3 = idx4 = idx5 = idx6 = idx7 = idx8 = idx9 = 0;

memset(&cur, 0, sizeof(struct cursor)); // p1을 구조체 크기만큼 0으로 설정

memset(total, 0, sizeof(total)); //

wiringPiSetup(); // wiringPi로 핀을 사용하겠다는 시작의미

initialize\_button();

fbfd = open(FBDEVFILE, O\_RDWR); // 적을 파일을 연다.

if (fbfd < 0)

{

perror("fbdev open");

exit(1);

}

ret = ioctl(fbfd, FBIOGET\_VSCREENINFO, &fbvar); //

if (ret < 0)

{

perror("fbdev ioctl");

exit(1);

}

if (fbvar.bits\_per\_pixel != 16)

{

fprintf(stderr, "bpp is not 16\n");

exit(1);

}

pfbdata = (unsigned short\*)mmap(0, fbvar.xres \* fbvar.yres \* 16 / 8, PROT\_READ | PROT\_WRITE, MAP\_SHARED, fbfd, 0);

if ((unsigned)pfbdata == (unsigned)-1)

{

perror("fbdev mmap");

exit(1);

}

// while 문 내부에서 스위치를 누르는 interrupt가 발생하면 일을 처리한다.

for (i = 0; i < 240; i++)

{

for (j = 0; j < 320; j++)

{

offset = i \* fbvar.xres + j; /\* 위치 \*/

pixel = makepixel(0, 0, 0);

total[i][j] = 0;

\*(pfbdata + offset) = pixel;

}

}

start();

munmap(pfbdata, fbvar.xres \* fbvar.yres \* 16 / 8);

close(fbfd);

return 0;

}

**-> main 함수에서는 파일을 읽어오고 오류처리**

**-> cursor 구조체, total 배열 초기화**

**-> 메모리 매핑, 매핑된 메모리 반환 함수로 구성되어 있습니다.**

**3. Algorithm**

3.1 start 함수

while(1){

blinking\_cursor(cur.x, cur.y);

if (digitalRead(BT1) == 1)

{

if (idx1 % 3 == 0) print(dot, cur.x, cur.y, 0);

if (idx1 % 3 == 1) print(q, cur.x, cur.y, 0);

if (idx1 % 3 == 2) print(z, cur.x, cur.y, 0);

delay(300);

idx1++;

idx2 = idx3 = idx4 = idx5 = idx6 = idx7 = idx8 = idx9 = 0;

}

/\* BT2 ~ BT9 반복 \*/

if (digitalRead(LEFT) == 0)

{

left();

delay(300);

}

if (digitalRead(DEL) == 0)

{

erase(cur.x, cur.y);

delay(300);

}

if (digitalRead(RIGHT) == 0)

{

right();

delay(300);

}

}

- Start 함수 설명 -

1. 무한루프를 돌면서 cursor를 blinking 하다가 스위치가 눌릴 경우 각 스위치의 동작에 맞게 인터럽트가 발생합니다.

2. 각 스위치들이 의도치하지 않게 여러 번 동작할 수 있으므로 delay를 통해서 해결했습니다.

3. 스위치마다 idx를 할당해주어서 문자가 순환되게 출력될 수 있게 idx % 3을 하여서 해결하였고 idx1이 눌릴 때는 나머지 idx(2 ~ 9)를 0으로 초기화 해주면서 다시 다른 버튼을 눌렀을 때 예) (a , b, c), (d , e, f) 스위치가 있을 때 d -> a -> e 가 아닌 d -> a -> d 가 출력되게 제어 하였습니다.

3.2 cursor control

struct cursor {

int x;

int y;

};

void left()

{

if (cur.x == 0 && cur.y == 0) // 맨 처음 상황

{

cur.x = 192;

cur.y = 252;

}

else if (cur.y == 0) // 위에 줄이 있을 때 맨 앞에 있는 경우

{

cur.y = 252;

cur.x -= 32;

}

else { // 왼쪽으로 갈 수 있는 상황

cur.y -= 28;

}

idx1 = idx2 = idx3 = idx4 = idx5 = idx6 = idx7 = idx8 = idx9 = 0;

}

void right()

{

if (cur.y == 252 && cur.x == 192) // 맨 마지막 상황

{

cur.x = 0;

cur.y = 0;

}

else if (cur.y == 252) // 아랫 줄로 현재 위치를 내려준다

{

cur.y = 0;

cur.x += 32;

}

else {

cur.y += 28;

}

idx1 = idx2 = idx3 = idx4 = idx5 = idx6 = idx7 = idx8 = idx9 = 0;

}

- cursor control –

1. cursor를 편하게 제어하기 위해서 cursor 구조체를 만들어서 사용하였습니다.

2. left 함수로 cursor 를 좌측( 24 pix + 4 pix )으로 이동하게 하였고 왼쪽 끝에서는

cur.y = 252;

cur.x -= 32;

코드를 통해서 1행 위로 올리고 cursor를 오른쪽 끝으로 위치시켰습니다.

3. right 함수로 cursor 를 오측( 24 pix + 4 pix )으로 이동하게 하였고 오른쪽 끝에서는

cur.y = 0;

cur.x += 32;

코드를 통해서 1행 아래로 올리고 cursor를 왼쪽 끝으로 위치시켰습니다.

3.3 print

void print(int alpha[][24], int x, int y, int key)

{

int i, j;

for (i = x; i < x + 24; i++)

{

for (j = y; j < y + 24; j++)

{

offset = i \* fbvar.xres + j; /\* 위치 \*/

pixel = makepixel(0, 0, 0);

total[i][j] = 0;

\*(pfbdata + offset) = pixel;

}

}

for (i = x; i < x + 24; i++)

{

for (j = y; j < y + 24; j++)

{

offset = i \* fbvar.xres + j; /\* 위치 \*/

if (alpha[i - cur.x][j - cur.y] > 0)

{

pixel = makepixel(255, 255, 255);

total[i][j] = 1;

}

else {

pixel = makepixel(0, 0, 0);

total[i][j] = 0;

}

\*(pfbdata + offset) = pixel;

}

}

return;

}

- Print 함수 설명 –

1. 현재 커서 위치에서부터 24 \* 24 사각형을 만든다는 생각으로 검정 화면으로 지웁니다.

2. 다음, 비슷한 방식으로 현재 커서 위치에서부터 24 \* 24 사각형을 print 함수에서 받은 alpha[][24] 배열에서 받은 문자 배열로 1일 때 흰색 점, 0일 때 검정색 점을 찍어서 print를 했습니다.

3. TFT LCD의 화면에 있는 pixel 값들을 유동적으로 제어할 수 있도록 total 이라는 배열에 계속적으로 TFT LCD의 픽셀들과 같은 값으로 저장합니다.

3.4 delete ( source 상 erase 함수 )

int i, j;

int temp[24][24];

// total 에 있는 값 (뒤 칸에 있는 값)을 temp에 받아온다.

for (i = x; i < x + 24; i++)

{

for (j = y; j < y + 24; j++)

{

// 뒤에 값을 가져오는 것인데

// 맨 오른쪽에 있던 값은 다음 줄 맨 앞의 값을 가져와야 한다.

if (y == 252)

{

if (x == 192) // 맨 마지막 문자 처리

{

total[i][j] = 0;

temp[i - x][j - y] = 0;

}

else {

total[i][j] = total[i + 32][j - y];

temp[i - x][j - y] = total[i + 32][j - y];

}

}

else {

total[i][j] = total[i][j + 28];

temp[i - x][j - y] = total[i][j + 28];

}

}

}

- delete(erase) 함수 설명 1 –

1. 위 for문은 temp라는 행렬에 뒤에 있는 값 즉, 뒤에 있는 문자를 그대로 넣어 놓습니다.

2. TFT LCD의 오른쪽 끝 문자에 cursor가 왔을 때는 다음 행 왼쪽 첫 번째 열의 문자를 temp에 저장합니다.

3. cur(cursor 구조체) 이 전역변수이지만 x, y 로 따로 받아온 이유는 x, y 로 모든 문자를 방문해야하는데 cur.x, cur.y가 직접 움직이면 delete를 할 때마다 cursor의 위치가 바뀌기 때문입니다.

// temp 에 있는 값 즉 뒤에있던 값을

for (i = x; i < x + 24; i++)

{

for (j = y; j < y + 24; j++)

{

offset = i \* fbvar.xres + j; /\* 위치 \*/

if (temp[i - x][j - y] > 0)

{

pixel = makepixel(255, 255, 255);

total[i][j] = 1;

}

else {

pixel = makepixel(0, 0, 0);

total[i][j] = 0;

}

\*(pfbdata + offset) = pixel;

}

}

if (x == 192 && y == 252)

{

return;

}

y += 28;

if (y == 280)

{

x += 32;

y = 0;

}

erase(x, y);

}

- delete(erase) 함수 설명 2 –

1. 다음 두번째 for문은 temp에서 받았던 값을 현재 좌표 위치의 문자 칸( 24 \* 24 )에 그대로 적습니다.

2. 다음 문자칸으로 x, y 좌표를 이동시키면서 erase 함수를 **재귀 함수**로 실행시킵니다. 두 개의 for문을 보시면 뒤의 문자를 당겨오면서 현재 위치에 적는 방법을 반복하게 됩니다. ( 즉, delete한 문자 위치 이후의 TFT LCD의 전체 픽셀을 앞으로 당기는 효과를 내었습니다. )

3. 아래의 조건문을 만나면서 재귀 함수 호출이 끝나게 되고 가장 마지막 문자까지 앞으로 당겨지게 됩니다.

if (x == 192 && y == 252)

{

return;

}

**4. Exception Handling**

4.1 Out of range Handle

/\* left 함수 내의 코드\*/

if (cur.x == 0 && cur.y == 0) // 맨 처음 상황

{

cur.x = 192;

cur.y = 252;

}

/\* right 함수 내의 코드\*/

if (cur.y == 252 && cur.x == 192) // 맨 마지막 상황

{

cur.x = 0;

cur.y = 0;

}

- 설명 –

1. 사용자가 문자를 입력하기 위해서는 cursor의 위치에서 스위치를 눌러서 동작시켜야 합니다. 그 의미는 즉, 커서의 위치가 TFT LCD의 범위 밖으로 벗어나지 않으면 out of range로 인해서 보드가 멈추거나 비정상적인 동작을 하지 않습니다.

2. 이 방법을 생각한 후에 cursor의 위치를 끝에서 처음으로 처음에서 끝으로 이동시켜주면서 에러를 핸들링 했습니다.

3. 252, 192, 0, 0 의 숫자가 나오는 이유는 본론에서 말했다시피 cur.x, cur.y의 좌표 ( 현재 좌표) 가 한 문자의 왼쪽 상단이기 때문에 다음 숫자가 나오게 되었습니다.

4.2 Push several buttons at same time

if (digitalRead(BT1) == 1)

{

if (idx1 % 3 == 0) print(dot, cur.x, cur.y, 0);

if (idx1 % 3 == 1) print(q, cur.x, cur.y, 0);

if (idx1 % 3 == 2) print(z, cur.x, cur.y, 0);

delay(300);

idx1++;

idx2 = idx3 = idx4 = idx5 = idx6 = idx7 = idx8 = idx9 = 0;

}

if (digitalRead(RIGHT) == 0)

{

right();

delay(300);

}

- 설명 –

1. 여러 스위치을 동시에 누를 시에는 미세하게 먼저 인식된 버튼의 문자가 출력되고 delay로 인해서 다른 버튼은 인식이 되지 않으므로 delay로 핸들링 했습니다.

4.3 Print character

if (digitalRead(BT1) == 1)

{

if (idx1 % 3 == 0) print(dot, cur.x, cur.y, 0);

if (idx1 % 3 == 1) print(q, cur.x, cur.y, 0);

if (idx1 % 3 == 2) print(z, cur.x, cur.y, 0);

delay(300);

idx1++;

idx2 = idx3 = idx4 = idx5 = idx6 = idx7 = idx8 = idx9 = 0;

}

- 설명 –

1. BT1 내 코드 idx2 = idx3 = idx4 = idx5 = idx6 = idx7 = idx8 = idx9 = 0; 를 통해서

idx1이 눌릴 때는 나머지 idx(2 ~ 9)를 0으로 초기화 해주면서 다시 다른 버튼을 눌렀을 때 예) (a , b, c), (d , e, f) 스위치가 있을 때 d -> a -> e 가 아닌 d -> a -> d 가 출력되게 제어 하였습니다.

2. left, right 스위치를 통해 left(), right() 함수를 이용해서 커서를 이동할 때는 (idx1 ~ idx9) 까지 0으로 초기화 시켜주면서 다른 작업을하다가 a 버튼을 눌렀을 때 a가 아닌 b or c 가 먼저 출력되는 것을 방지합니다.