REPORT

임베디드 소프트웨어 텀프로젝트

21511675 이예인

21611668 이현정

1.Introduction

텀프로젝트의 제출 기한이 크리스마스 근처이기 때문에 크리스마스 컨셉으로 텀프로젝트를 먼저 생각하게 되었고, 평소에 크리스마스 시즌마다 하던 크리스마스 트리 꾸미기를 프로그래밍으로 구현해보고 싶었기 때문에 트리를 스크린에 띄우고 그 트리를 꾸미는 프로그램을 제작하게 되었다.

11월 14일에 처음 팀프로젝트 과제가 올라왔을 때부터 초기 컨셉을 잡기 위해서 아이디어 회의를 했고, 그 이후 중간 제안서 발표를 위해서 역할을 분담하고, 알고리즘과 환경을 구축했다. 중간 제안서를 발표한 후에 코드 작성 및 수정을 했다. 그리고 기말시험이 끝난 뒤 최종적으로 코드를 수정하고 PPT 작성 및 최정 데모 영상을 촬영했다. 코드 구성, PPT 작성, 결과 보고서 작성은 함께 수행했고, 중간 제안서 발표, 중간 발표, 최종발표는 이현정 조원이 맡아서 준비했다.

초기에는 스위치 마다 다른 기능을 넣고, 그 스위치를 누른 뒤 화면을 터치하면 트리 꼭대기의 별, 크리스마스 볼, 눈, 글리터 등으로 트리를 꾸밀 수 있게 제작하려고 계획했었다. 하지만 트리 꼭대기의 별과 크리스마스 볼을 png 파일로 제작하였는데, achro-5250은 bmp로 이미지 파일을 올리는 것이 권장됨으로 배경을 투명화 할 수 없어서 트리꾸미기에는 bmp파일이 적합하지 않다고 생각하여 계획을 변경하게 되었다.

트리 꼭대기의 별과 크리스마스 볼을 넣을 수 없게 되었기 때문에 트리 꼭대기의 별은 꼭 필요하다고 판단되어 초기 이미지에 넣게 되었고, 크리스마스볼을 구현할 수 있는 방법이 없어서 나머지 기능이었던 글리터와 눈을 강조하여 글리터의 색을 다양화하고 눈은 글리터보다 크기를 키워서 더욱 눈에 띄게 했다. 글리터와 눈은 수업시간에 실습했던 'fbrect.c' 코드를 응용하여 색상과 크기를 입력으로 받아서 다양한 크기와 색상으로 사각형을 만들 수 있도록 변경했다.

2. Design Detail

1. Make Algorithm

Switch 1 :gritter	스위치[0] ->touch 함수 실행/ 좌표받기 -> 그 자리에 make rect
Switch 2 :ball	스위치[1] -> touch 함수 실행/좌표받기 -> 그 자리에 image 불러오기
Switch 3 :stars	스위치[2] -> change image with star
Switch 4: reset	스위치[3] -> all clear

2. Select, and Search the function that we can use

이번 텀프로젝트에서는 기본을 충분히 이해하고, 잘 응용할 수 있게 되는 것에 초점을 두었다. 그래서 수업시간에 배운 스위치와 touch 함수와 임베디드 기기에서 rect를 자체 생성 하는 것, 그리고 bmp파일을 출력하는 함수를 적극적으로 사용하기로 했다.

우리가 초점에 둔 것은 어떻게 함수끼리 유기적으로 묶으며, 어떻게 활용을 하고 합치면서 일어나는 오류들을 어떻게 해결할 것인가 였다.

3. Prepare to configure

우리는 크리스마스 트리 만들기라는 프로그램을 만들기로 하였는데, 특성상 이미지파일이 중요할 수 밖에 없었다. 일단 '꾸미기' 이므로 귀엽고 보기 좋아야 하기 때문이다. 그래서 컨셉을 잡은 직후 바로 이미지 파일을 만들었다.

4. Coding and Porting

앞서 말 했던 알고리즘을 바탕으로 코드를 구성하였다. 그 내용은 밑과 같다.

```
// Bmp File display to Frame-buffer example

#ifndef _BMPINFO_
#define _BMPINFO_

#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <string.h>
#include <sys/ioctl.h>
#include <sys/types.h>
#include <linux/fb.h>

// display area size
#define WIDTH 1024
#define HEIGHT 600
#define READ_FILE_NAME1 "tree.bmp"
#define READ_FILE_NAME2 "tree_star.bmp"

// make type definition - windows type
typedef unsigned short U16;
typedef unsigned short WORD;
typedef unsigned int DWORD;
typedef unsigned int LONG;
typedef unsigned char BYTE;

// make type definition - bmp file header
typedef struct tagBITMAPFILEHEADER{
        WORD bfType;
        DWORD bfSize;
        WORD bfReserved1;
        WORD bfReserved2;
        DWORD bfOffBits; } __attribute((packed))BITMAPFILEHEADER;


// make type definition - bmp image infomation header
typedef struct tagBITMAPINFOHEADER{
        DWORD biSize;
        LONG biWidth;
        LONG biHeight;
        WORD    biplanes;
        WORD    biBitCount;
        DWORD   biCompression;
        DWORD   biSizeImage;
        LONG    biXPelsPerMeter;
        LONG    biYPelsPerMeter;
        DWORD   biClrUsed;
        DWORD   biClrImportant; } __attribute((packed)) BITMAPINFOHEADER;
/* other mathod : Group Attribute Packed Mathod
// #pragma pack(1)
// #pragma pack() */
#endif
```

```c
#ifndef TOUCH_H
#define TOUCH_H

#include <fcntl.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>


typedef struct  touch_process_data__t
{
    unsigned    char    finger_cnt;
    unsigned    int     x1;
    unsigned    int     y1;
    unsigned    int     x2;
    unsigned    int     y2;
}touch_process_data_t;


#endif // TOUCH_H
```

Bmp 파일을 불러오기위한 헤더파일과, touch 코드를 사용하기 위한 헤더파일이다.

기존의 bmp_info.h 헤더파일과 거의 유사하나, tree_star.bmp을 뒤에 쓸 함수에 변수명으로서 할당하기 위하여 헤더파일에 다른 이름으로 지정해주었다.

```c
#include "bmp_info_tree.h"
#include <unistd.h>
#include <string.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <sys/types.h>
#include <linux/fb.h>
#include "touch.h"
#include <sys/stat.h>
#include <signal.h>
#define MAX_BUTTON 9
#define DEPTH 3 /* 24 Bit */

typedef unsigned int U32;
unsigned int makepixel(U32  r, U32 g, U32 b)
{
    return (U32)((r<<16)|(g<<8)|b);
}

unsigned int m_touch_pos[14];

int fbrect(int a, int b,int c,int d)
{
    int check;
    int frame_fd;
    U32 pixel;
    int offset;
    int posx1, posy1, posx2, posy2;
    int repx, repy;
    struct fb_var_screeninfo fvs;

    if((frame_fd = open("/dev/fb0",O_RDWR))<0) {
        perror("Frame Buffer Open Error!");
        exit(1);
```

```c
int fbrect(int a, int b,int c,int d)
{
    int check;
    int frame_fd;
    U32 pixel;
    int offset;
    int posx1, posy1, posx2, posy2;
    int repx, repy;
    struct fb_var_screeninfo fvs;

    if((frame_fd = open("/dev/fb0",O_RDWR))<0) {
        perror("Frame Buffer Open Error!");
        exit(1);
    }

    if((check=ioctl(frame_fd,FBIOGET_VSCREENINFO,&fvs))<0) {
        perror("Get Information Error - VSCREENINFO!");
        exit(1);
    }

    if(fvs.bits_per_pixel != 32) {
        perror("Unsupport Mode. 32Bpp Only.");
        exit(1);
    }

    if(lseek(frame_fd, 0, SEEK_SET) < 0) {  // Set Pixel Map
        perror("LSeek Error.");
        exit(1);
    }

    pixel = makepixel(a,b,c); // 색상 만들기
     unsigned int data[1024];
    unsigned char temp;
    /* multi-touch data process struct */
```

```c
    pixel = makepixel(a,b,c); // 색상 만들기
     unsigned int data[1024];
    unsigned char temp;
    /* multi-touch data process struct */
    typedef struct  touch_process_data__t   {
        unsigned    char    finger_cnt;
        unsigned    int     x1;
        unsigned    int     y1;
        unsigned    int     x2;
        unsigned    int     y2;
    }
    touch_process_data_t;
    touch_process_data_t   ts_data;

    int count=0;
    int ret = 0;
    int press=0;
    int isMulti=0;
    int start_flag=1;
    int fingerCnt=1;
    int j;
    int cor_num;
    int fd;
    int i=0;
    memset(data,0,sizeof(data));
     fd=open("/dev/input/event2",O_RDWR);
    if(fd<0) {
        printf("Error!\n");
        return -1;
    }

    for(i=0;i<14;i++)
        m_touch_pos[i]=0;
```

```
for(i=0;i<14;i++)
    m_touch_pos[i]=0;

    if(start_flag)
    {
        for(j=0;j<5;j++) //1 x y 0 0
        {
            ret=read(fd,&ts_data,sizeof(ts_data));
            if(ret!=-1) {
                printf("%d ",ts_data.x2);
                m_touch_pos[j]=ts_data.x2;
            }
        }
        start_flag=0;
    }
    else
    {
        cor_num=1;
        for(j=0;j<4;j++) //x y 0 0
        {
            ret=read(fd,&ts_data,sizeof(ts_data));
            if(ret!=-1) {
                m_touch_pos[j]=ts_data.x2;
            }
        }
    }

    if(m_touch_pos[4-cor_num] > 1)
    {
        fingerCnt=2;
        for(j=0;j<3;j++)
        {
            ret=read(fd,&ts_data,sizeof(ts_data));
            if(ret!=-1)
                m_touch_pos[j+5-cor_num]=ts_data.x2;
```

```
    if(m_touch_pos[4-cor_num] > 1)
    {
        fingerCnt=2;
        for(j=0;j<3;j++)
        {
            ret=read(fd,&ts_data,sizeof(ts_data));
            if(ret!=-1)
                m_touch_pos[j+5-cor_num]=ts_data.x2;
        }
    }
    else if(m_touch_pos[4-cor_num]==1)
        fingerCnt=0;
    else
    {
        fingerCnt=1;
    }

    if(m_touch_pos[7-cor_num] != 0 )
    {
        fingerCnt=3;
        for(j=0;j<3;j++)
        {
            ret=read(fd,&ts_data,sizeof(ts_data));
            if(ret!=-1)
                m_touch_pos[j+8-cor_num]=ts_data.x2;
        }
    }

    if(m_touch_pos[10-cor_num] != 0)
    {
        fingerCnt=4;
        for(j=0;j<3;j++)
        {
            ret=read(fd,&ts_data,sizeof(ts_data));
```

```
        if(m_touch_pos[10-cor_num] != 0)
        {
            fingerCnt=4;
            for(j=0;j<3;j++)
            {
                ret=read(fd,&ts_data,sizeof(ts_data));
                if(ret!=-1)
                    m_touch_pos[j+11-cor_num]=ts_data.x2;
            }
        }

int k;
    for(k=0;k<4;k++)
            printf("%d ",m_touch_pos[k]);
        printf("\n");
        fingerCnt=0;

    posx1 = m_touch_pos[0]-d;  // 사각형의 크기를 결정한다.
    posx2 = m_touch_pos[0]+d;
    posy1 = m_touch_pos[1]-d;
    posy2 = m_touch_pos[1]+d;

    for(repy=posy1; repy < posy2; repy++) {
        offset = repy * fvs.xres * (32/8) + posx1 * (32/8);
        if(lseek(frame_fd, offset, SEEK_SET) < 0) {
            perror("LSeek Error!");
            //exit(1);
        }
        for(repx = posx1; repx <= posx2; repx++)
            write(frame_fd, &pixel,(32/8));
    }


    close(fd);
```

```
                m_touch_pos[j+11-cor_num]=ts_data.x2;
            }
        }

int k;
    for(k=0;k<4;k++)
            printf("%d ",m_touch_pos[k]);
        printf("\n");
        fingerCnt=0;

    posx1 = m_touch_pos[0]-d;  // 사각형의 크기를 결정한다.
    posx2 = m_touch_pos[0]+d;
    posy1 = m_touch_pos[1]-d;
    posy2 = m_touch_pos[1]+d;

    for(repy=posy1; repy < posy2; repy++) {
        offset = repy * fvs.xres * (32/8) + posx1 * (32/8);
        if(lseek(frame_fd, offset, SEEK_SET) < 0) {
            perror("LSeek Error!");
            //exit(1);
        }
        for(repx = posx1; repx <= posx2; repx++)
            write(frame_fd, &pixel,(32/8));
    }


    close(fd);


    close(frame_fd);
    return 0;
}
```

이 함수는 사람이 화면에 터치를 하면, 터치한 곳의 좌표를 입력받는다. 그리고 입력받은 좌표를사용하여 그 위치에 사각형의 오브젝트를 생성한다. 터치 함수는 x,y 두 좌표를 생성한다. 이 두 좌표를 사각형의 첫 좌표, 그리고 마지막 좌표에 d의 차이를 두고 입력되도록 설정한다. 여기서 fbrect 함수는 네개의 변수

를 입력 받는다. 일단 Int a, int b, int c는 각각 픽셀의 색을 결정한다. 사각형을 생성하는 함수에는 RGB를 조합하여 내가 원하는 색을 만들수있다. 그 기능을 이용하여 우리는 이 함수를 사용하여 다양한 색을 조합할 수 있다. 또한 마지막 int d는 픽셀의 크기를 결정짓는다. 입력받은 좌표는 한 점이므로, 사각형을 생성하기 위해서는 첫 점과 마지막 점에 틈이 필요하기 때문이다. 그래서 int d를 받아서 픽셀의 크기를 조절하도록 했다.

```
void treestar_bmp()
{
        BITMAPFILEHEADER BmpFileHd;
        BITMAPINFOHEADER BmpInfoHd;

        unsigned char ImageBuffer[WIDTH*HEIGHT*DEPTH];
        unsigned char Temp,r,g,b;
        int i,h,w;
        size_t retval;
        int check;
        int fb_fd;
        U32 pixel;
        int offset;
        int posx1, posy1, posx2, posy2;
        int repx, repy;
        int pixcnt=0;
        struct fb_var_screeninfo fvs;

        FILE *fp=NULL;

        // File Open
        if ((fp = fopen(READ_FILE_NAME2,"rb")) == NULL) {
                printf("File Open Error2\n");
                exit(1);
        }

        // File Read - read File Header
        fread(&BmpFileHd, sizeof(BITMAPFILEHEADER), 1, fp);
        if(BmpFileHd.bfType!=0x4D42) {
                printf("Not Bitmap file or Unsupport Bitmap file\n");
                exit(1);
        }

        printf("* BMP File Header - \n");
```

```
        printf("* BMP File Header - \n");
        printf("  Type    : 0x%04X\n", BmpFileHd.bfType);
        printf("  Size    : %d\n", BmpFileHd.bfSize);
        printf("  Offbits : %d\n", BmpFileHd.bfOffBits);

        // File Read
        fread(&BmpInfoHd, sizeof(BITMAPINFOHEADER), 1, fp);
        printf("* BMP Info Header - \n");
        printf("  FileName   : %s\n",READ_FILE_NAME2);
        printf("  Infosize   : %d\n", BmpInfoHd.biSize);
        printf("  Width      : %d\n", BmpInfoHd.biWidth);
        printf("  Height     : %d\n", BmpInfoHd.biHeight);
        printf("  Planes     : %d\n", BmpInfoHd.biplanes);
        printf("  Bitcount   : %d\n", BmpInfoHd.biBitCount);
        printf("  Compressin : %d\n", BmpInfoHd.biCompression);
        printf("  Sizeimage  : %d\n", BmpInfoHd.biSizeImage);
        printf("  XPelsPerMeter : %d\n", BmpInfoHd.biXPelsPerMeter);
        printf("  YPelsPerMeter : %d\n", BmpInfoHd.biYPelsPerMeter);
        printf("  ClrUsed       : %d\n", BmpInfoHd.biClrUsed);
        printf("  ClrImportant  : %d\n", BmpInfoHd.biClrImportant);
        retval=fread(ImageBuffer, sizeof(ImageBuffer), 1, fp);
        if(retval<0) {
                perror("Buffer Empty or Error Occured!");
                exit(1);
        }

        retval=sizeof(ImageBuffer);

        for(h=0;h<(HEIGHT/2);h++) {
                for(w=0;w<(WIDTH*3);w++) {
                        Temp = ImageBuffer[(h*(WIDTH*3))+w];
                        ImageBuffer[(h*(WIDTH*3))+w] = ImageBuffer[(retval-((h+1)*(WIDTH*3)))+w];
                        ImageBuffer[(retval-((h+1)*(WIDTH*3)))+w] = Temp;
                }
```

```
                        ImageBuffer[(retval-((h+1)*(WIDTH*3)))+w] = Temp;
                }
        }

        // For RGB Sorting
        for(i=0;i<retval;i+=3) {
                Temp = ImageBuffer[i+2];
                ImageBuffer[i+2] = ImageBuffer[i];
                ImageBuffer[i]=Temp;
        }

        if((fb_fd = open("/dev/fb0",O_RDWR))<0) {
                perror("Frame Buffer Open Error!\n");
                exit(1);
        }

        if((check=ioctl(fb_fd,FBIOGET_VSCREENINFO,&fvs))<0) {
                perror("Get Information Error - VSCREENINFO!");
                exit(1);
        }

        if(fvs.bits_per_pixel != 32) {
                perror("Unsupport Mode. 32Bpp Only!\n");exit(1);
        }

        if(lseek(fb_fd, 0, SEEK_SET) < 0) {
                perror("LSeek Error!!\n");
                exit(1);
        }

        posx1=0;
        posx2=WIDTH;
        posy1=0;
        posy2=HEIGHT;
```

```
        posx1=0;
        posx2=WIDTH;
        posy1=0;
        posy2=HEIGHT;

        for(repy=posy1;repy<posy2;repy++) {
                offset = repy * fvs.xres * (32/8) + posx1 * (32/8);
                if(lseek(fb_fd, offset, SEEK_SET) < 0) {
                        perror("LSeek Error!!!");
                        exit(1);
                }
                for(repx = posx1;repx < posx2; repx++) {
                        pixel = makepixel(ImageBuffer[pixcnt++],ImageBuffer[pixcnt++],\
                        ImageBuffer[pixcnt++]);
                        write(fb_fd, &pixel,(32/8));
                }
        }
        close(fb_fd);

        fclose(fp);

}
```

Tree에 star가 합성된 사진을 출력하기 위한 함수이다. 원래의 fb_from_bmp와 다르게 사진을 함수의 입력값으로 받는 것이 아니라, 위에서 READ_FILE_NAME으로 지정했던 tree_star bmp 파일을 여기에서 기본값으로 넣어주었다. 그래서 이 함수는 실행만 하여도 tree_star.bmp를 화면에 출력시켜준다.

```
int main(void)
{
        int i;
        int dev;
        int buff_size;
        int image;
        int a;
        unsigned char push_sw_buff[MAX_BUTTON];//button open error

        dev = open("/dev/fpga_push_switch", O_RDWR);

        if (dev<0){
                printf("Device Open Error\n");
                close(dev);
                return -1;
        }

        (void)signal(SIGINT, user_signal1);
        treestar_bmp();
        buff_size=sizeof(push_sw_buff);
        printf("Press <ctrl+c> to quit. \n");
        while(!quit){
                usleep(400000);
                read(dev, &push_sw_buff, buff_size);
                if(push_sw_buff[0]) {
                        while(1){
                                fbrect(255,255,255,5);//white
                                read(dev, &push_sw_buff, buff_size);
                                if(push_sw_buff[0]==0) break;
                        }

                }

                else if(push_sw_buff[1]){
```

```
                else if(push_sw_buff[1]){
                        while(1){
                                fbrect(0,255,255,2);//greenblue
                                read(dev, &push_sw_buff, buff_size);
                                if(push_sw_buff[1]==0) break;
                        }

                }
                else if(push_sw_buff[2]){
                        while(1){
                                fbrect(255,0,0,2);//red
                                read(dev, &push_sw_buff, buff_size);
                                if(push_sw_buff[2]==0) break;
                        }
                }
                else if(push_sw_buff[3]){
                        while(1){
                                fbrect(254,204,0,2);//yellow
                                read(dev, &push_sw_buff, buff_size);
                                if(push_sw_buff[3]==0) break;
                        }
                }
                else if(push_sw_buff[4]) {
                        treestar_bmp();//reset
                        }
        }
        close(dev);
}
```

메인함수에서는 임베디드 기기가 여러가지 기능을 하게 하기 위해 스위치를 사용하였다. 스위치를 사용하기 위하여 device driver을 사용하였다. 디바이스 드라이버를 사용하니까 어떤 스위치가 눌리고있는지 기계에서 읽어, 컴퓨터가 바로바로 읽을수 있었다. 어느 스위치가 눌리고있는 것인지 확인하고 값을받는 것은 push_sw_bmp[] 의 배열이다. 만약 1번 버튼을 눌리면 push_sw_bmp[0] 이 1이 되는 식이다. 그래서 if 조건문에 배열의 몇번째에 1이 들어가는지 확인하고 만약 조건에 적합한 스위치가 눌리면 while 무한 루프를 돌게 하였다. 이 상태에서는 계속 반복문을 돌고있기 때문에 손가락을 떼고 있어도 수행 하고자하는 기능이 계속해서 실행되게 하였다. 그리고 무한으로 돌다가, 다른 스위치 버튼이 눌리면 그 값을 확인하고 while문에서 break 해서 다른 기능을 수행하게 코드를 짜보았다.

5. Fix the error

1. png 파일을 지원하는 코드가 존재하지않음.

처음에 우리가 구상했던 christmas-ball은 외부에서 이미지파일을 그려서, 배경 투명화를 한 뒤에 좌표를 불러오는 함수를 사용하여 그 자리에 크기에 맞춰 생성하는 방법을 구상했었다. 여기서, 자연스럽게 christmas-ball을 가져오기 위해서

는 배경투명화가 반드시 필요했다. Png 파일은 배경투명화를 지원하지만 Bmp 파일은 배경투명화를 지원하지않는다. 하지만 우리가 가지고 있던 것은 bmp 파일을 불러오는 코드뿐이였고, png 파일을 불러오는 코드는 archro5250에서 호환이 되지않았다. 왜냐하면 png는 무손실 압축포맷이여서 용량도 크고, 출력하는 방식이 복잡하다. achro5250은 임베디드 기기이며, 뛰어난 이미지 출력 시스템을 갖추지않고있기 때문에 지원이 안된다고 설명이되어있었다. 그래서 png를 포기하고 bmp를 사용하여 할 수 있는것만이라도 수행하기로 하였다.

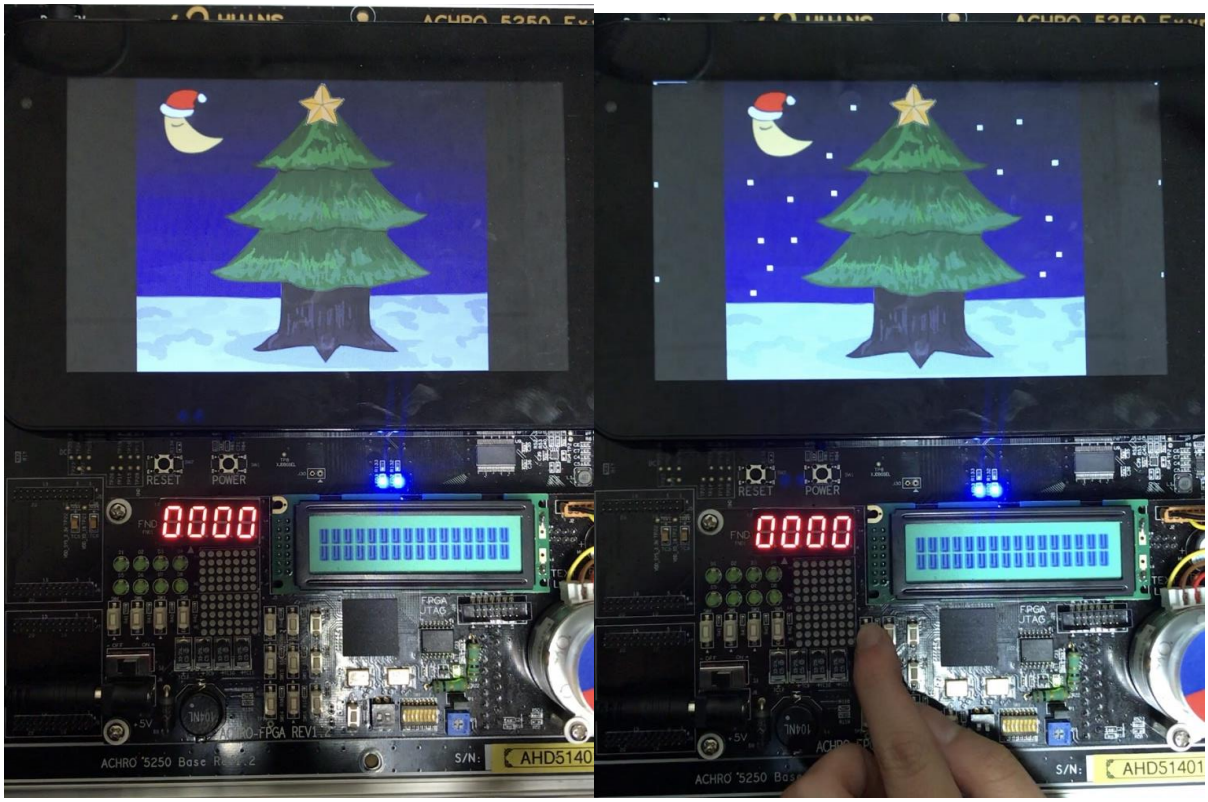
2. 자잘한 오류와 임베디드 리눅스에대한 숙련도 부족

코드는 정상적이고, 컴파일이 되었는데 막상 실행해보니 오류가 나기도 했고 제대로 작동이 잘 되다가도 갑자기 값이 이상한 값으로 튀어 프로그램이 종료되기도 했다. 이 경우에는 문제를 크게 두가지로 생각하였다. 1. 포팅이 제대로 되지않았다. 2. 컴파일에서 오류가 났는데, 겉보기엔 이상이 없는것이다. 이 두가지 전제를 두고 코드를 계속 보았는데 결국 해결하지 못하고 그냥 껏다키니까 정상 값으로 돌아오는 경우가 있었다.

그리고 우리 조는 코딩은 여러 번 해보았지만 코드를 실제로 디바이스에 적용시켜보는 것은 처음이라 수시로 뜨는 오류 대처에도 미숙했고, 코드가 멀쩡해도 디바이스에서 받아들이지 못하는 내용들에 대해선 무력했다. 그것 때문에 많은 어려움을 겪고 원래 계획했던 방향에서 많은 수정을 거치게 되었다.

실질적으로 코드를 짜고, 그것을 임베디드기계에 적용하는 것은 우리가 비주얼스튜디오로 코드를 짜고 컴파일하던것과는 완전 다른 문제라는 것을 뼈깊이 느끼게 되었다.

3. RESULT



기본 화면/ SWITCH 1 : make SNOW

SWITCH 2: make gritter(greenblue) / SWITCH 3: make gritter(red)

SWITCH 4: make gritter(yellow) / SWITCH 5: clear

첫번째 사진은 메인 함수에서 기본으로 띄운 이미지파일이다. 두번째 사진부터 우리가 추가한 기능을 보여준다.

스위치 1에서는 터치 한 곳에 10x10px의 하얀색 픽셀을 만들어 눈을 표현하였다. fbrect라는 함수를 사용하였다.

스위치2에서는 터치한곳에 4x4px의 청록색 픽셀을 만들어 글리터를 표현하였다.

스위치3, 스위치4 에서는 크기는 같지만 색상이 다른 글리터를 표현하였다.

스위치 5에서는 all clear 기능을 넣었다.

3. Conclusion

이 레포트를 마지막으로 대학 생활 4년의 마지막을 장식하게 되었습니다. 학기 초부터 텀프로젝트가 기말시험 대체라는 사실을 알았기 때문에 계속 고민하고, 한 학기 동안 배운 내용을 모두 사용하여 텀프로젝트를 완성했기 때문에 많은 시간을 투자한 만큼 뜻 깊은 과제입니다.

초기에는 png파일을 올릴 수 없다는 사실을 알 수 없었기 때문에 조원이 그린 그림으로 크리스마스트리 꾸미기를 만드려고 했지만 achro-5250의 한계 때문에 bmp파일 밖에는 로드 할 수가 없고 그렇게 되면 배경을 투명화 할 수 없기 때문에 크리스마스트리 꾸미기를 구상했던 저희 조에게는 치명적인 문제점이었습니다. png파일을 올릴 수 있는 방법도 찾아보고, 아예 꾸며진 사진을 바꾸는 등의 생각을 해보았지만 png파일을 올리는 방법은 찾지 못했고, 꾸며진 사진을 한 장 한 장 바꾸는 것은 우리 조의 처음 취지와 맞지 않다고 생각했기 때문에 초기에 생각했던 기능에서 이미지 파일로 꾸미는 기능은 삭제하고 자체적으로 사각형을 만들어서 꾸미는 기능에 더 집중하여 그 기능을 더욱 응용하도록 계획을 수정했습니다.

텀프로젝트를 진행하는 중 가장 힘들었던 점은 achro-5250 키트가 불안정했기 때문인지 계속해서 구동을 하면 같은 코드로도 오류가 나고 부팅이 되지 않거나 갑자기 꺼지는 문제가 있었습니다. 텀프로젝트 발표를 하루 앞두고 마지막 데모 영상을 찍을 때도 같은 문제가 발생하여 다른 학생의 키트를 급하게 빌려 데모영상을 찍어서 문제를 해결했습니다.

아쉬운 점은 역시 이미지파일을 이용하지 못하여서 처음에 구상했던 그림과 많이 다른 결과가 나온 점이 있고, 가능하다면 버저를 사용하여 크리스마스 캐롤을 나오게 하는 등의 기능도 추가하고 싶었지만 achro-5250의 버저로는 크리스마스 캐롤 같은 소리를 나오게 할 수 없었기 때문에 그 기능을 사용하지 못한 것도 많이 아쉽습니다.

아쉬웠지만 유익한 수업이었고, 이런 활동을 하게되어 굉장히 뿌듯하고 뜻깊은 경험이었습니다.