CY20202
稲葉周
2022/06/07

プロジェクト演習最終課題

レポート

1. はじめに

このレポートの目的は制作したプログラムを整理、内容を他人に伝えるためにどうすればよいかを学ぶことである。またレポートにまとめることで自分のプログラムの工夫点、コード記述における至らない点を洗い出すことも可能である。そのため自分自身のプログラムを見つめなおすといったことも目的に含まれる。

2. 組み込み開発環境の概要

開発のための機器

・Raspberry Pi
・LAN ケーブル
・USB ケーブル

開発のためのソフト

・Linux（Debian）
・Minicom

役割

・Raspberry Pi：プログラムがここで動く
・LAN ケーブル：Raspberry Pi と Linux を同一ネット環境に置くため
・USB ケーブル：Raspberry Pi と PC を接続するため
・Linux：Raspberry pi で動かすプログラム・実行ファイルの作成およびファイルの転送
・MiniCom：Raspberry Pi の制御

手順

1．PC を立ち上げる
2．mc コマンドより minicom を立ち上げる
3．Raspberry pi と PC を LAN ケーブル、USB ケーブルで接続
4．Raspberry pi にログインしたら PC より SCP コマンドで実行ファイル受け取り
5．Raspberry pi にて実行ファイルを実行

3. 動作の概要

タクトスイッチ全種類、タイマー、に location 関数を用いた LCD 制御を行う組み込み系プログラム。このプログラムは黒、赤、黄、緑、青のどれかに正解が割り振られており、ユーザーはどのタクトスイッチが正解かをタイマー、LCD を用いた 3 秒カウントダウンの間に選択し、正解を当てるゲームである。

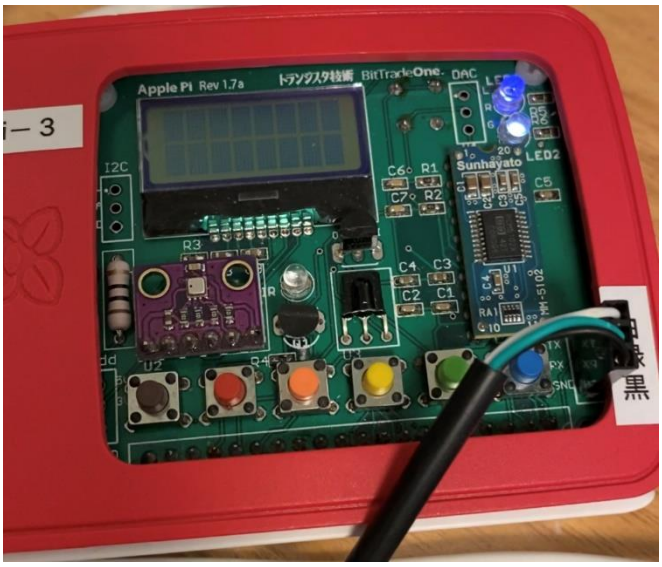

図 3.1　使用する Raspberry pi

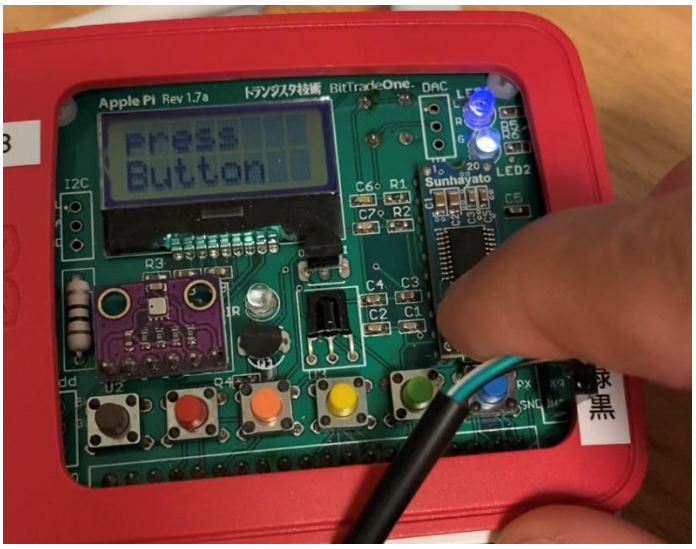

図 3.2　実際に動かしているところの写真

4.　システムの詳細

・乱数を使用した正解のつくり方

```
int random_gen()
{
    int random;
    srand((unsigned int)time(NULL));
    // 0から5までの乱数を発生
   random = rand() % 6;
   return random;
```

```
}
```

今回は 6 つのタクトスイッチを使用するため、0～5 の範囲で関数が呼ばれるたびに代わる乱数を用意する。0～5 の範囲であれば乱数を 6 で割った余りで実現可能である。

```
srand((unsigned int)time(NULL));
```

では呼ばれるたびに種を変更する srand 関数を使用している。

・ボタンが押されたかを判別するビット演算

```
/*
    引数は
    戻り値はどの色のタクトスイッチを押したかを判別する整数型の値
    タクトスイッチが押されていないか、どの色が押されているかを判別する関数
*/
int judgeButton(unsigned int x){
    if((x&0x00400000)==0)
    {
        printf("pushed blacked\n");
        return 0;
    }
    else if((x&0x00800000)==0)
    {
        printf("pushed red\n");
        return 1;
    }
    else if((x&0x01000000)==0)
    {
        printf("pushed orange\n");
        return 2;
    }
    else if((x&0x02000000)==0)
    {
        printf("pushed yellow\n");
        return 3;
    }
    else if((x&0x04000000)==0)
    {
        printf("pushed green\n");
```

```
        return 4;
    }
    else if((x&0x08000000)==0)
    {
        printf("pushed blue\n");
        return 5;
    }
    else{
        printf("un pushed\n");
        return 6;
    }
}
```

```
    if((x&0x00400000)==0)
    {
        printf("pushed blacked\n");
        return 0;
    }
```

何も押されてない状態と何かを押された状態を比べる。黒が押された時の 16 進数は 2f80c1df、何も押されていないときの 16 進数は 2fc0c1df 比べると左から 3 桁目の数値が違うことがわかる。8 と c を 2 進数に直すと 1000、1100 である。論理積を用いて黒だけが変わるような 2 進数を探すと 0100 である。それを 16 進数にすると 4 であるため、条件分岐でビット論理積を行う。このことにより何色のボタンが押されているか、押されていないかを判断することが可能である。

・タイマーを用いたカウントダウンの処理

```
/*
戻り値はどのタクトスイッチが押されたかを judgeButton 関数で判断し、押されたタクトスイッチに対応する整数
三秒のカウントダウンの間にどのタクトスイッチが最後に押されたかを判別する関数
*/
int CountDown(){
    int l=3,answer=6;
    unsigned int x;
        while(1){
            if(off==1){//カウントダウンはタイマーで行う
```

```
                if(l==3){
                    e_clear();
                    val=lcd_datawrite(i2c,"3");
                    if(val==-1)printf("datawrite error");
                    l-=1;
                }
                else if(l==2){
                    e_clear();
                    val=lcd_datawrite(i2c,"2");
                    if(val==-1)printf("datawrite error");
                    l-=1;
                }
                else if(l==1){
                    e_clear();
                    val=lcd_datawrite(i2c,"1");
                    if(val==-1)printf("datawrite error");
                    l-=1;
                }
                else if(l==0)break;

            }
            off=0;
            x = *((unsigned int *)(gpio_baseaddr + GPIO_GPLEV0));
            if(judgeButton(x)!=6)answer=judgeButton(x);
        }
/*answer には何か押された場合、judgeButton 関数でどのボタンが押されたかを判断し、そのタクトスイッチに対応した整数を格納*/
        return answer;
}
```

この関数はタイマーを用いたカウントダウンをするものである。

```
        signal(SIGALRM,sig_handler);/*タイマーの設定*/
        struct itimerval timval;
        timval.it_interval.tv_sec=1;
        timval.it_interval.tv_usec=0;
        timval.it_value.tv_sec=1;//間隔は一秒
        timval.it_value.tv_usec=0;
```

```
setitimer(ITIMER_REAL,&timval,NULL);//timer start
```

上はタイマーの設定とタイマースタートの処理である。
タイマーで発生させるシグナルでグローバル変数を変え、その変数を if 文でチェックし、処理をする。これを用いて LCD 上に一秒ごとに 3->2->1-とカウントダウンさせる。その際 l=3 としてあるのはシグナルが呼ばれるたびに表示させる数値を変化させるためである。

・テスト

sudo で実行する。まず上側”Press”下側”Button”が同時に一秒ごとに点滅する。その際何もタクトスイッチを押さなかった場合、ずっと点滅し続ける。なにかスイッチを押すと、3.2.1 と 3 秒のカウントダウンが始まり、その間に赤のボタンを押し、答えも赤である正解の場合は LCD 上に”You”下側に”Won!”が表示され、次にＬＣＤの上に”Again”下には 3 秒かけての”3””2””1”のカウントダウンが始まる。このカウントダウン中に何も押さなければそのままゲーム終了し、LCD 上に”EndGame”下に”GoodBye!”が表示されプログラムは終了。カウントダウン中にスイッチを押せば LCD 上に”Play”下に”Again!”が表示され PressButton の点滅に戻る。解答フェーズで正解しなかった場合、LCD 上に”Miss!”下には答えのタクトスイッチの色の名前 ex)”Yellow”...が表示される。そしてまた再度プレイするかの LCD 表示“Again？”と何も押さなければそのままゲーム終了、押せばもう一度プレイのカウントダウンが始まる

・プログラムの全体

```
//cy20202 Amane Inaba 最終提出課題 2022/06/07
#include <errno.h>
#include <fcntl.h>
#include <linux/i2c-dev.h>
#include <linux/input.h>
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/mman.h>
#include <sys/types.h>
#include <termios.h>
#include <unistd.h>
#include <sys/time.h>
#include <sys/ioctl.h>
#include <time.h>
```

```
// peripheral register physical address
#define GPIO_PHY_BASEADDR  0x3F200000
#define GPIO_AREA_SIZE 4096 // PAGE_SIZE
#define GPIO_GPFSEL0 0x0000 // for gpio 0..9, MSB 2bits are reserved
// omit GPFSEL1..GPFSEL5
#define GPIO_GPSET0 0x001C // gpio 0..31
#define GPIO_GPSET1 0x0020 // gpio 32..53
#define GPIO_GPCLR0 0x0028 // gpio 0..31
#define GPIO_GPCLR1 0x002C // gpio 32..53
#define GPIO_GPLEV0 0x0034 // gpio 0..31
#define GPIO_GPLEV1 0x0038 // gpio 32..53

int i2c,val,off;
/*シグナルハンドラ*/
void sig_handler(int signum){
 off=1;//グローバル変数への代入を行い，メインのループでその変数をチェックして LCD 制
御
};
unsigned int memread(void *baseaddr, int offset);
void memwrite(void *baseaddr, int offset, unsigned int x);

void *gpio_baseaddr;

unsigned int memread(void *baseaddr, int offset)
{
    unsigned int *p;
    p = baseaddr+offset;
    return *p; // read memory-mapped register
}

void memwrite(void *baseaddr, int offset, unsigned int x)
{
    unsigned int *p;
    p = baseaddr+offset;
    *p = x; // write memory-mapped register
```

```
}

int lcd_cmdwrite(int fd, unsigned char dat)
{
 unsigned char buff[2];
 buff[0] = 0;
 buff[1] = dat;
 return write(fd,buff,2);
}

int lcd_datawrite(int fd, char dat[])
{
 int len;
 char buff[100];

 len = strlen(dat);  // don't count EOS (Null char)
 if (len>99) {printf("too long string\n"); exit(1); }
 memcpy(buff+1, dat, len); // shift 1 byte, ignore EOS
 buff[0] = 0x40; // DATA Write command
 return write(fd, buff, len+1);
}

void initLCD(int fd)
{
 int i;
 unsigned char init1[]={ 0x38, 0x39, 0x14, 0x70, 0x56, 0x6c };
 unsigned char init2[]={ 0x38, 0x0c, 0x01 };

 usleep(100000); // wait 100ms
 for (i=0;i<sizeof(init1)/sizeof(unsigned char);i++) {
  if(lcd_cmdwrite(fd, init1[i])!=2){
   printf("internal error1\n");
   exit(1);
  }
  usleep(50); // wait 50us
 }
```

```
 usleep(300000); // wait 300ms

 for (i=0;i<sizeof(init2)/sizeof(unsigned char);i++) {
  if(lcd_cmdwrite(fd, init2[i])!=2){
   printf("internal error2\n");
   exit(1);
  }
  usleep(50);
 }
 usleep(2000); // wait 2ms
}

int location(int fd, int y)
{
 int x = 0;
 int cmd=0x80 + y * 0x40 + x;
 return lcd_cmdwrite(fd, cmd);
}

int clear(int fd)
{
 int val = lcd_cmdwrite(fd, 1);
 usleep(1000); // wait 1ms
 return val;
}
/*この関数は六つのタクトスイッチのどれかに正解を割り振るために、
関数が呼ばれるたびに異なる乱数を生成し、それを 6 で割る
得られる余りは0～5のため、6 つのスイッチのどれを正解とするかランダムにする
といった関数
戻り値として0～5の整数のどれかを返す
*/
int random_gen()
{
    int random;
    srand((unsigned int)time(NULL));
```

```
    // 0から5までの乱数を発生
   random = rand() % 6;
   return random;
}
/*
    clearとclear関数のエラー処理も同時に行う関数
*/
void e_clear(){
    val=clear(i2c);//LCD上の表示を削除
    if(val==-1)printf("clear error");
}
/*
    引数は
    戻り値はどの色のタクトスイッチを押したかを判別する整数型の値
    タクトスイッチが押されていないか、どの色が押されているかを判別する関数
*/
int judgeButton(unsigned int x){
    if((x&0x00400000)==0)
    {
        printf("pushed blacked\n");
        return 0;
    }
    else if((x&0x00800000)==0)
    {
        printf("pushed red\n");
        return 1;
    }
    else if((x&0x01000000)==0)
    {
        printf("pushed orange\n");
        return 2;
    }
    else if((x&0x02000000)==0)
    {
        printf("pushed yellow\n");
        return 3;
```

```
    }
    else if((x&0x04000000)==0)
    {
        printf("pushed green\n");
        return 4;
    }
    else if((x&0x08000000)==0)
    {
        printf("pushed blue\n");
        return 5;
    }
    else{
        printf("un pushed\n");
        return 6;
    }
}
/*
戻り値なし、返り値なし
i2c の基本設定をする関数
*/
void i2csetup(){

    i2c=open("/dev/i2c-1", O_RDWR);
    if(i2c==-1)printf("open error");
    val=ioctl(i2c, I2C_SLAVE, 0x3e); // I2C 独自の設定
    if(val==-1)printf("ioctl error");
    initLCD(i2c); // lcd.c の中に定義されている(戻り値なし)
}
/*
戻り値は整数型で、三秒カウントの間なにも押さない場合は6を、タクトスイッチが
押された場合はスイッチに応じた整数を返す関数
*/
int playAgain(){
        int x,z=6,l=3;
        e_clear();
        location(i2c,0);
```

```
        val=lcd_datawrite(i2c, "Again?");
        if(val==-1)printf("datawrite error");
        location(i2c,1);
        while(1){
            if(off==1){//タイマーで 3 秒カウントを行う
                if(l==3){
                    val=lcd_datawrite(i2c,"3");
                    if(val==-1)printf("datawrite error");
                    l-=1;
                }
                else if(l==2){
                    e_clear();//一瞬 Again を消す
                    location(i2c,0);
/*再度 Again を表示。コンピュータレベルのため、消えるようには見えない*/
                    val=lcd_datawrite(i2c, "Again?");
                    if(val==-1)printf("datawrite error");
                    location(i2c,1);
                    val=lcd_datawrite(i2c,"2");
                    if(val==-1)printf("datawrite error");
                    l-=1;
                }
                else if(l==1){
                    location(i2c,0);
                    val=lcd_datawrite(i2c, "Again?");
                    if(val==-1)printf("datawrite error");
                    location(i2c,1);
                    val=lcd_datawrite(i2c,"1");
                    if(val==-1)printf("datawrite error");
                    l-=1;
                }
                else if(l==0)break;//3 秒たったらループを抜ける

            }
            off=0;
            x = *((unsigned int *)(gpio_baseaddr + GPIO_GPLEV0));
            location(i2c,0);
```

```
/*z には何か押された場合、judgeButton 関数でどのボタンが押されたかを判断し、そのタクトスイッチに対応した整数を格納*/
            if(judgeButton(x)!=6)z=judgeButton(x);
        }
        return z;

}

/*
戻り値はどのタクトスイッチが押されたかを judgeButton 関数で判断し、押されたタクトスイッチに対応する整数
三秒のカウントダウンの間にどのタクトスイッチが最後に押されたかを判別する関数
*/
int CountDown(){
    int l=3,answer=6;
    unsigned int x;
        while(1){
            if(off==1){//カウントダウンはタイマーで行う
                if(l==3){
                    e_clear();
                    val=lcd_datawrite(i2c,"3");
                    if(val==-1)printf("datawrite error");
                    l-=1;
                }
                else if(l==2){
                    e_clear();
                    val=lcd_datawrite(i2c,"2");
                    if(val==-1)printf("datawrite error");
                    l-=1;
                }
                else if(l==1){
                    e_clear();
                    val=lcd_datawrite(i2c,"1");
                    if(val==-1)printf("datawrite error");
                    l-=1;
                }
```

```
                else if(l==0)break;

            }
            off=0;
            x = *((unsigned int *)(gpio_baseaddr + GPIO_GPLEV0));
            if(judgeButton(x)!=6)answer=judgeButton(x);
        }
/*answer には何か押された場合、judgeButton 関数でどのボタンが押されたかを判断し、そのタクトスイッチに対応した整数を格納*/
        return answer;
}
/*
戻り値、引数はなし
LCDの上側に"Press"LCDの下側に"Button"を表示させる関数
*/
void PressDisplay(){
        location(i2c,0);
        val=lcd_datawrite(i2c, "Press");//hello を LCD 上に表示
        if(val==-1)printf("datawrite error");
        location(i2c,1);
        val=lcd_datawrite(i2c,"Button");
        if(val==-1)printf("datawrite error");
}
/*
戻り値はなし、引数は int 型で二つ、random は正解色と対応する整数、
answer はユーザーが押したタクトスイッチの色と対応する整数
ユーザーがおしたタクトスイッチが正解かどうかを判断する関数
*/
void CheckAnswer(int random,int answer){
        if(random==answer){
            location(i2c,0);
            val=lcd_datawrite(i2c, "You");
            if(val==-1)printf("datawrite error");
            location(i2c,1);
            val=lcd_datawrite(i2c, "Won!");
            if(val==-1)printf("datawrite error");
```

```
            sleep(3);
        }
/*
答えとユーザーの押した色が異なる場合、LCD の上側に Miss!を出力し、location 関数で下側
に正解の色を 3 秒間表示する
*/
        else if(random!=answer){
            val=lcd_datawrite(i2c, "Miss!");
            if(val==-1)printf("datawrite error");
            location(i2c,1);
            if(random==0){
                val=lcd_datawrite(i2c, "Black");
                if(val==-1)printf("datawrite error");
            }
            else if(random==1){
                val=lcd_datawrite(i2c, "Red");
                if(val==-1)printf("datawrite error");
            }
            else if(random==2){
                val=lcd_datawrite(i2c, "Orange");
                if(val==-1)printf("datawrite error");
            }
            else if(random==3){
                val=lcd_datawrite(i2c, "Yellow");
                if(val==-1)printf("datawrite error");
            }
            else if(random==4){
                val=lcd_datawrite(i2c, "Green");
                if(val==-1)printf("datawrite error");
            }
            else if(random==5){
                val=lcd_datawrite(i2c, "Blue");
                if(val==-1)printf("datawrite error");
            }
            sleep(3);
```

```
        }

}
/*
戻り値引数はなし
LCDの上側に"Play"LCDの下側に"Again!!"を表示させる関数
*/
void AgainDisplay(){
    e_clear();
    location(i2c,0);
    val=lcd_datawrite(i2c, "Play");
    if(val==-1)printf("datawrite error");
    location(i2c,1);
    val=lcd_datawrite(i2c, "Again!!");
    if(val==-1)printf("datawrite error");
    sleep(3);
    e_clear();

}
/*
戻り値、引数なし
LCDの上側に"GameEnd"LCDの下側に"goodBye!"を三秒間
表示させ i2c を閉じる関数
*/
void GameEnd(){
    e_clear();
    location(i2c,0);
    val=lcd_datawrite(i2c, "GameEnd");//hello を LCD 上に表示
    if(val==-1)printf("datawrite error");
    location(i2c,1);
    val=lcd_datawrite(i2c, "GoodBye!");//hello を LCD 上に表示
    if(val==-1)printf("datawrite error");
    sleep(3);
    e_clear();
    val=close(i2c);
    if(val==-1)printf("close error");
```

```
}
/*
テスト結果
まず上側”Press”下側”Button”が同時に一秒ごとに点滅する。その際何もタクトスイッチを押さなかった場合、ずっと点滅し続ける。なにかスイッチを押すと、3.2.1 と 3 秒のカウントダウンが始まり、その間に赤のボタンを押し、答えも赤である正解の場合は LCD 上に”You”下側に”Won!”が表示され、次に上に”Again”下には”3””2””1”のカウントダウンが始まる。
*/
int main(){
    while(1){
        unsigned int x,y;
        int i,l=3;
        int fd,random,r,answer=6;
        random=random_gen();
        signal(SIGALRM,sig_handler);/*タイマーの設定*/
        struct itimerval timval;
        timval.it_interval.tv_sec=1;
        timval.it_interval.tv_usec=0;
        timval.it_value.tv_sec=1;//間隔は一秒
        timval.it_value.tv_usec=0;
        fd = open("/dev/mem", O_RDWR);
        if(fd==-1)printf("open error");
        gpio_baseaddr = mmap(NULL, 4096, PROT_READ | PROT_WRITE,
MAP_SHARED, fd, GPIO_PHY_BASEADDR);
        i2csetup();
        setitimer(ITIMER_REAL,&timval,NULL);//timer start
        while(1){
            x = *((unsigned int *)(gpio_baseaddr + GPIO_GPLEV0));
            r=judgeButton(x);
            if(r!=6)break;
            PressDisplay();
            if(off==1){
                e_clear();
                usleep(500000);
                off=0;
            }
```

```
        }
        e_clear();
        answer=CountDown();
        e_clear();
        CheckAnswer(random,answer);
        val=playAgain();
        if(val==6)break;
        AgainDisplay();
    }
    GameEnd();

    return 0;
}
```

6.　おわりに

・シグナルハンドラを用いたタイマー制御

・Raspberry Pi を用いた LCD 制御

・タクトスイッチの押されていない、もしくは押されているかをビット演算での検出

・スレッドを用いた並行プログラミングの構築

・インターネットの仕組み

感想

Web サーバー等がどうやって動いているのかを、自分で手を動かすことで理解することができた。意外と単純な仕組みでインターネットが動いていることを知ることができて、興味深かった。

RaspberryPi を用いた組み込み開発は個人的に一番興味を持った章である。これから組み込み系の勉強をしてみようと思えたプロジェクト演習だった。