

輝度可変型LED(10灯)ユニット(I2C対応)

概要

前回製作した□NightRiderV1(残像方式)を、I2C対応に改良してみました□ I2C対応なので、他のPICからLEDの輝度データを自由に設定することができます。このようにすることで□NightRider以外にも使い道が考えられるので、汎用的な名称のユニットとしました。

動作原理

I2Cのスレーブ機能としての、基本的な構造は、前回製作した□LCDモニタ□(I2C対応)と同じです。

<メモリ構造> 10個のLEDの輝度データを設定するために、10バイトのメモリを使用します□
□0x00□LED1の輝度データ(0x00~0xFF) □0x01□LED2の輝度データ(0x00~0xFF) □0x02□LED3の輝度データ(0x00~0xFF) □0x03□LED4の輝度データ(0x00~0xFF) □0x04□LED5の輝度データ(0x00~0xFF)
□0x05□LED6の輝度データ(0x00~0xFF) □0x06□LED7の輝度データ(0x00~0xFF) □0x07□LED8の輝度データ(0x00~0xFF) □0x08□LED9の輝度データ(0x00~0xFF) □0x09□LED10の輝度データ(0x00~0xFF)

輝度の制御方式は、前回製作した□NightRiderV1(残像方式)と同様です。

<マスター側から、スレーブ側への□LED輝度データ出力> マスター側から、アドレス(0x00~0x09)のメモリに、輝度データを書き込みます。

```
Soft_I2C_Start();  
Soft_I2C_Write(0xE0);  
Soft_I2C_Write(0x00);  
Soft_I2C_Write(255);  
Soft_I2C_Write(128);  
Soft_I2C_Write(64);  
Soft_I2C_Write(32);  
Soft_I2C_Write(16);  
Soft_I2C_Write(8);  
Soft_I2C_Write(4);  
Soft_I2C_Write(2);  
Soft_I2C_Write(1);  
Soft_I2C_Write(0);  
Soft_I2C_Stop();
```

回路図



```

// *****
*
/*

```

```

<輝度可変型□□□□□灯) ユニット□□□□対応) >
*/
//*****
*

#define          LED          PORTA.F2

#define          SW_ADDR1     PORTA.F3
#define          SW_ADDR2     PORTA.F4
#define          SW_ADDR3     PORTA.F5

#define          ADDR_BASE    0xE0

#define          ON           0
#define          OFF          1

#define          DATA_SIZE   10

#define          LED1         PORTB.F0
#define          LED2         PORTB.F2
#define          LED3         PORTB.F3
#define          LED4         PORTB.F5
#define          LED5         PORTB.F6
#define          LED6         PORTB.F7
#define          LED7         PORTA.F6
#define          LED8         PORTA.F7
#define          LED9         PORTA.F0
#define          LED10        PORTA.F1

//*****
*

void  i2c_Write(unsigned short dat)
{
    while (SSPSTAT.BF == 1)
        ;
    while (1) {
        SSPCON.WCOL = 0;
        SSPBUF = dat;
        if (SSPCON.WCOL == 1)
            continue;
        //
        SSPCON.CKP = 1;
        return;
    }
}

//*****
*

static unsigned    short    i2c_memory[DATA_SIZE], i2c_pnt, i2c_flg,

```

```

i2c_tmp;

//*****
*

void i2c_Handler()
{
    if (SSPSTAT.P == 1)           //ストップビットを検出するとTIMER0を再開する。
        INTCON.T0IE = 1;
    if (SSPSTAT.S == 1)           //スタートビットを検出するとTIMER0を一時停止す
る。
        INTCON.T0IE = 0;
    //
    i2c_tmp = SSPSTAT & 0b00101101;
    //
    if (i2c_tmp == 0b00001001) {   //書き込みモード、デバイスアドレス
        i2c_tmp = SSPBUF;
        i2c_flg = 0;
        i2c_pnt = 0;
        return;
    }
    if (i2c_tmp == 0b00101001) {   //書き込みモード、データ
        if (i2c_flg == 0) {
            i2c_pnt = SSPBUF;
            i2c_flg = 1;
            return;
        }
        if (i2c_flg == 1) {
            i2c_memory[i2c_pnt] = SSPBUF;
            i2c_pnt++;
            return;
        }
    }
    if (i2c_tmp == 0b00001100) {   //読み込みモード、デバイスアドレス
        i2c_Write(i2c_memory[i2c_pnt]);
        i2c_pnt++;
        return;
    }
    if (i2c_tmp == 0b00101100) {   //読み込みモード、データ□□□□□
        i2c_Write(i2c_memory[i2c_pnt]);
        i2c_pnt++;
        return;
    }
    if (i2c_tmp == 0b00101000) {   //読み込みモード、データ□□□□□□□□
        i2c_tmp = SSPBUF;
        SSPCON = 0b00111110;
        return;
    }
}

//*****

```

```
*

static unsigned short pwm_cnt;

void interrupt()
{
    if (PIR1.SSPIF == 1) {
        PIR1.SSPIF = 0;
        //
        LED = ON;
        i2c_Handler();
        LED = OFF;
    }
    //
    if (INTCON.T0IF == 1) { // 約0.128msec周期
        INTCON.T0IF = 0;
        //
        LED1 = (pwm_cnt <= i2c_memory[0]) ? ON : OFF;
        LED2 = (pwm_cnt <= i2c_memory[1]) ? ON : OFF;
        LED3 = (pwm_cnt <= i2c_memory[2]) ? ON : OFF;
        LED4 = (pwm_cnt <= i2c_memory[3]) ? ON : OFF;
        LED5 = (pwm_cnt <= i2c_memory[4]) ? ON : OFF;
        LED6 = (pwm_cnt <= i2c_memory[5]) ? ON : OFF;
        LED7 = (pwm_cnt <= i2c_memory[6]) ? ON : OFF;
        LED8 = (pwm_cnt <= i2c_memory[7]) ? ON : OFF;
        LED9 = (pwm_cnt <= i2c_memory[8]) ? ON : OFF;
        LED10 = (pwm_cnt <= i2c_memory[9]) ? ON : OFF;
        //
        if (pwm_cnt < 255)
            pwm_cnt++;
        else
            pwm_cnt = 1;
    }
}

//*****
*

static unsigned short Luminance[10] = {255, 128, 64, 32, 16, 8,
4, 2, 1, 0};

void main()
{
    unsigned short cnt;
    //
    CMCON = 0b00000111; // コンパレータは使用しない。
    ANSEL = 0b00000000; // A/Dコンバータは使用しない。
    OSCCON = 0b01110000; // クロックは内臓8MHzを使用する。
    TRISA = 0b00111000; // PORTAを設定する。
    TRISB = 0b00010010; // PORTBを設定する。
    OPTION_REG.NOT_RBPU = 0; // PORTBをプルアップする。
}
```

```

// TIMER0を設定する。
INTCON.T0IE = 1;
INTCON.T0IF = 0;
OPTION_REG.T0CS = 0;
OPTION_REG.PSA = 1;
OPTION_REG.PS0 = 0;
OPTION_REG.PS1 = 0;
OPTION_REG.PS2 = 0;
TMR0 = 0;
//□□□を設定する。
SSPSTAT.SMP = 1;
SSPSTAT.CKE = 1;
SSPCON.WCOL = 0;
SSPCON.SSP0V = 0;
SSPCON.SSPEN = 1;
SSPCON.CKP = 1;
SSPCON.SSPM0 = 0;
SSPCON.SSPM1 = 1;
SSPCON.SSPM2 = 1;
SSPCON.SSPM3 = 1;
SSPADD = ADDR_BASE + ((SW_ADDR3 == 1) ? 8 : 0) + ((SW_ADDR2 == 1) ?
4 : 0) + ((SW_ADDR1 == 1) ? 2 : 0);
PIE1.SSPIE = 1;
PIR1.SSPIF = 0;
//
LED = OFF;
i2c_pnt = 0;
i2c_flg = 0;
pwm_cnt = 1;
// 割り込み(全体)の設定
INTCON.PEIE = 1;
INTCON.GIE = 1;
//
for (cnt = 0; cnt < DATA_SIZE; cnt++) {
    i2c_memory[cnt] = Luminance[cnt];
}
Delay_ms(500);
for (cnt = 0; cnt < DATA_SIZE; cnt++) {
    i2c_memory[cnt] = Luminance[9 - cnt];
}
Delay_ms(500);
for (cnt = 0; cnt < DATA_SIZE; cnt++) {
    i2c_memory[cnt] = 0x00;
}
//
while (1) {
}
}

//*****
*
```

<参考> テスト用(PIC12F683)のプログラムです。

LedDisplay_i2c_master.c

```
//*****
*
/*
『輝度可変型□□□□□灯) ユニットのテストデータ送信用 ( マスター ) 』
*/
//*****
*

#define      SW      GPIO.F3
#define      LED      GPIO.F2

#define      ON      1
#define      OFF      0

#define      ACK      1
#define      NO_ACK    0

//*****
*

void  SwitchONcheck()
{
    while (Button(&GPIO, 3, 1, 0) == 0)
        ;
    while (Button(&GPIO, 3, 1, 1) == 0)
        ;
}

//*****
*

void main()
{
    unsigned    short    cnt, dat;
    //
    CMCON0 = 0b00000111;
    ANSEL.ANS0 = 0;
    ANSEL.ANS1 = 0;
    ANSEL.ANS2 = 0;
    ANSEL.ANS3 = 0;
    ADCON0.VCFG = 0;
    TRISIO = 0b00001011;
    OSCCON = 0b01110000;
    //
    for (cnt = 0; cnt < 10; cnt++) {
        LED = ON;
        Delay_ms(50);
    }
}
```

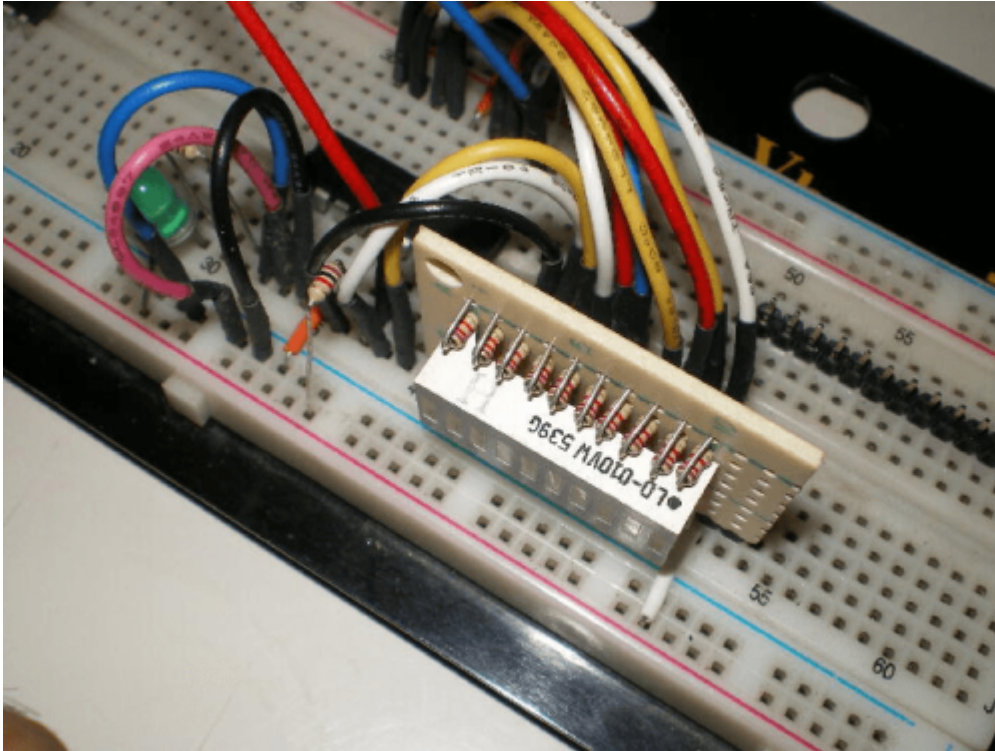
```
    LED = OFF;
    Delay_ms(50);
}
//
Soft_I2C_Config(&GPIO, 4, 5);    // SDA, SCL
//
while (1) {
    SwitchONcheck();
    //
    Soft_I2C_Start();
    Soft_I2C_Write(0xE0);
    Soft_I2C_Write(0x00);
    Soft_I2C_Write(255);
    Soft_I2C_Write(128);
    Soft_I2C_Write(64);
    Soft_I2C_Write(32);
    Soft_I2C_Write(16);
    Soft_I2C_Write(8);
    Soft_I2C_Write(4);
    Soft_I2C_Write(2);
    Soft_I2C_Write(1);
    Soft_I2C_Write(0);
    Soft_I2C_Stop();
    //
    SwitchONcheck();
    //
    Soft_I2C_Start();
    Soft_I2C_Write(0xE0);
    Soft_I2C_Write(0x00);
    Soft_I2C_Write(0);
    Soft_I2C_Write(1);
    Soft_I2C_Write(2);
    Soft_I2C_Write(4);
    Soft_I2C_Write(8);
    Soft_I2C_Write(16);
    Soft_I2C_Write(32);
    Soft_I2C_Write(64);
    Soft_I2C_Write(128);
    Soft_I2C_Write(255);
    Soft_I2C_Stop();
    //
    SwitchONcheck();
    //
    Soft_I2C_Start();
    Soft_I2C_Write(0xE0);
    Soft_I2C_Write(0x00);
    Soft_I2C_Write(1);
    Soft_I2C_Write(4);
    Soft_I2C_Write(16);
    Soft_I2C_Write(64);
    Soft_I2C_Write(255);
```

```
Soft_I2C_Write(255);
Soft_I2C_Write(64);
Soft_I2C_Write(16);
Soft_I2C_Write(4);
Soft_I2C_Write(1);
Soft_I2C_Stop();
//
SwitchONcheck();
//
Soft_I2C_Start();
Soft_I2C_Write(0xE0);
Soft_I2C_Write(0x00);
Soft_I2C_Write(255);
Soft_I2C_Write(64);
Soft_I2C_Write(16);
Soft_I2C_Write(4);
Soft_I2C_Write(1);
Soft_I2C_Write(1);
Soft_I2C_Write(4);
Soft_I2C_Write(16);
Soft_I2C_Write(64);
Soft_I2C_Write(255);
Soft_I2C_Stop();
//
SwitchONcheck();
//
Soft_I2C_Start();
Soft_I2C_Write(0xE0);
Soft_I2C_Write(0x00);
Soft_I2C_Write(rand() / 256);
Soft_I2C_Write(rand() / 256);
Soft_I2C_Write(rand() / 256);
Soft_I2C_Write(rand() / 256);
Soft_I2C_Write(rand() / 256);
Soft_I2C_Write(rand() / 256);
Soft_I2C_Write(rand() / 256);
Soft_I2C_Write(rand() / 256);
Soft_I2C_Write(rand() / 256);
Soft_I2C_Write(rand() / 256);
Soft_I2C_Stop();
//
for (cnt = 0; cnt < 10; cnt++) {
    LED = ON;
    Delay_ms(50);
    LED = OFF;
    Delay_ms(50);
}
}
}
```

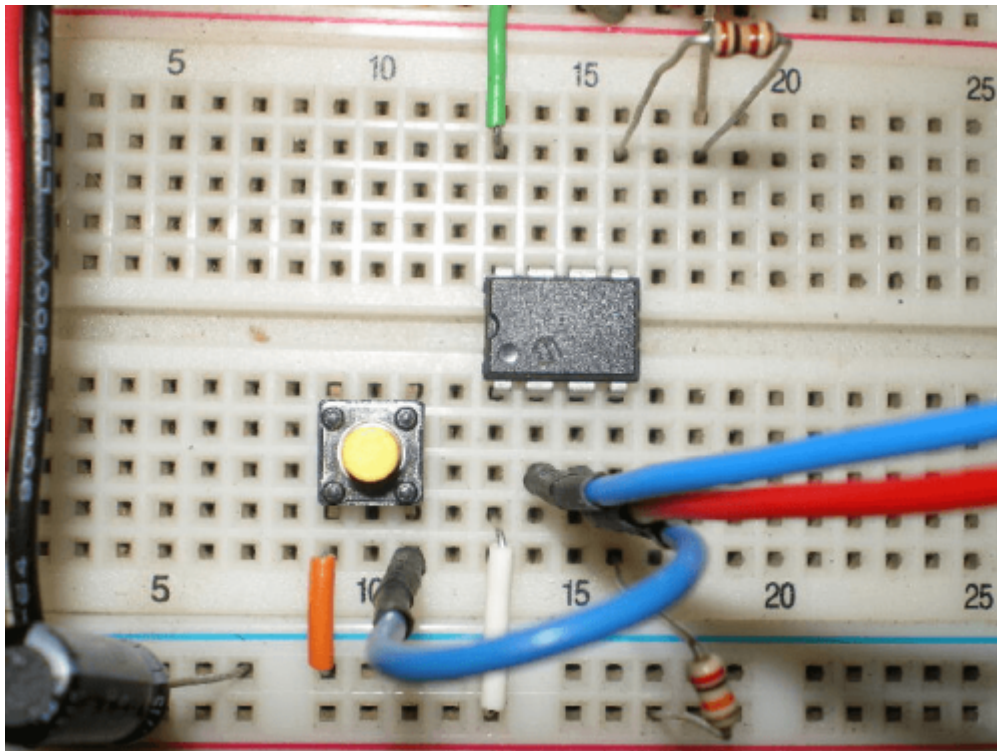
//*****

*

動作確認

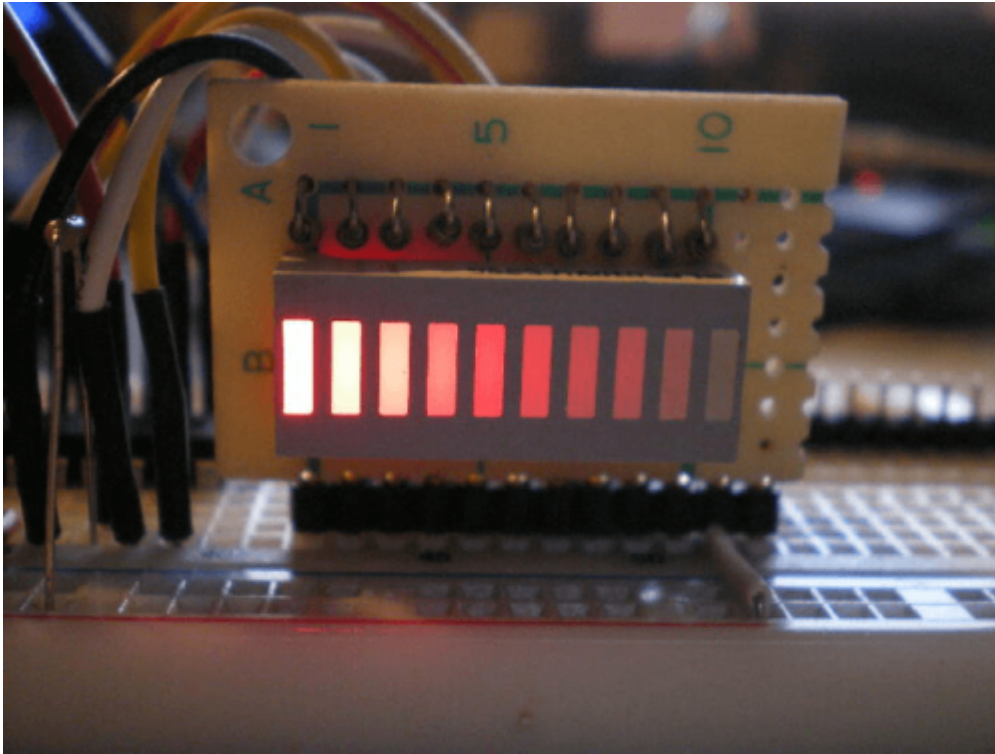


本ユニットへのテストデータ送信用のPIC12F683です。<右側に伸びている、赤色線=SDA□青色

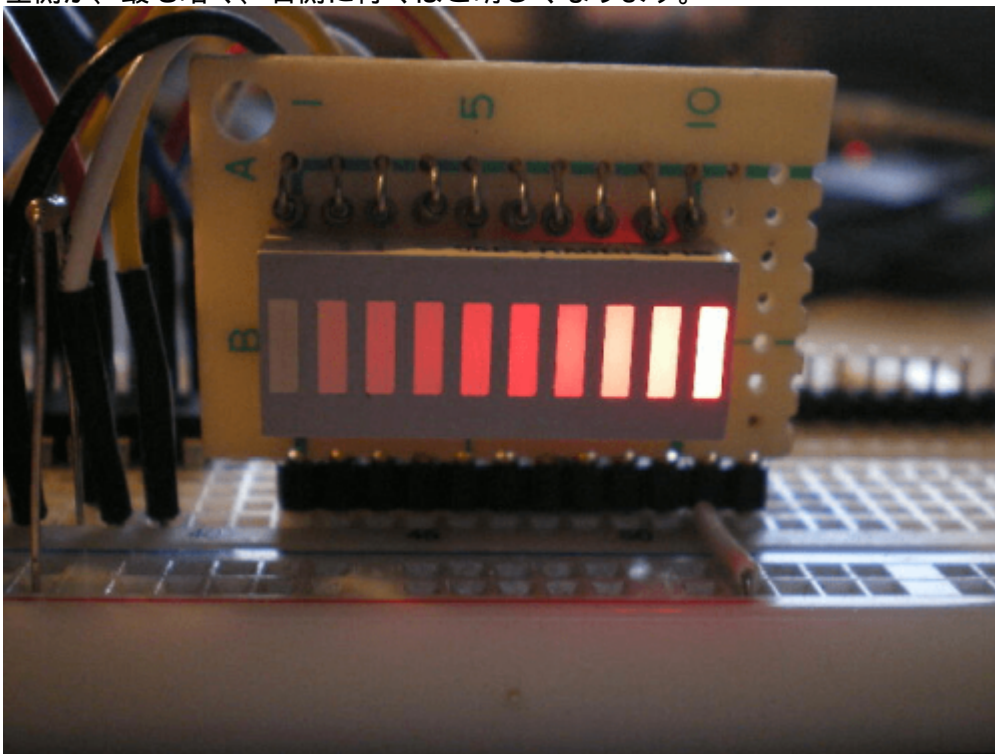


線=SCL>

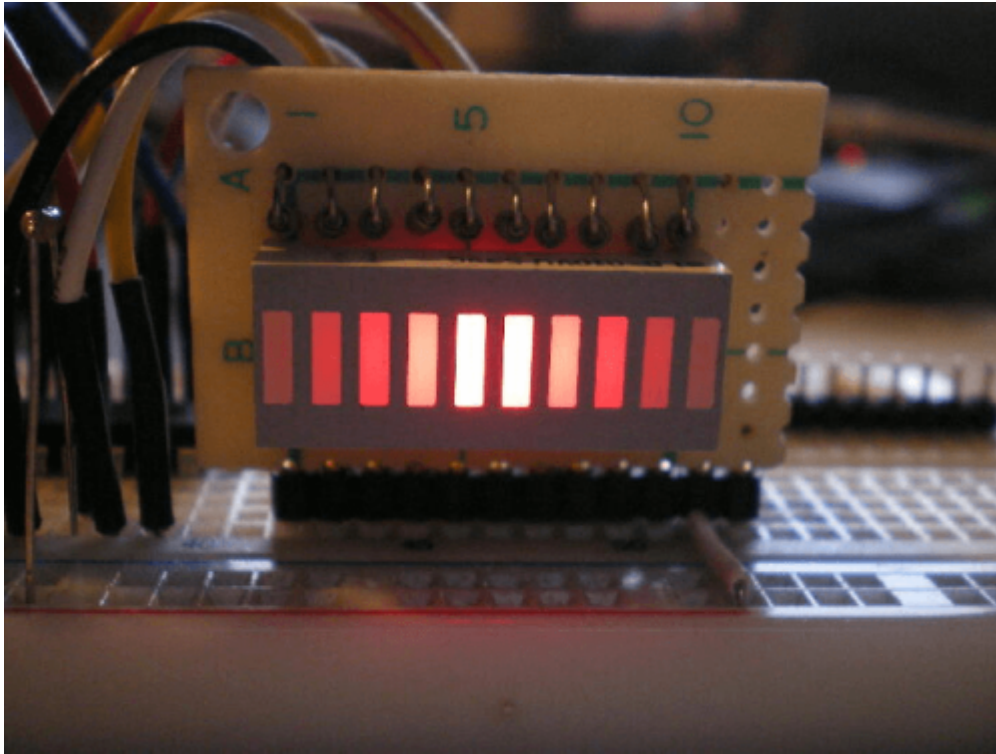
左側が、最も明るく、右側に行くほど暗くなります。



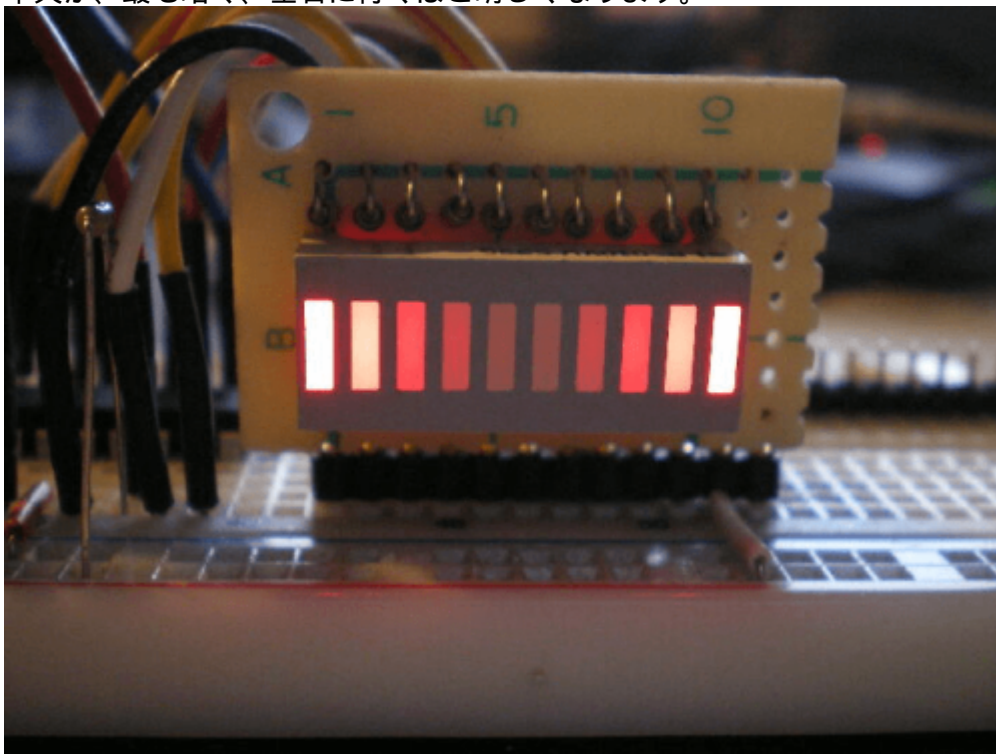
左側が、最も暗く、右側に行くほど明るくなります。



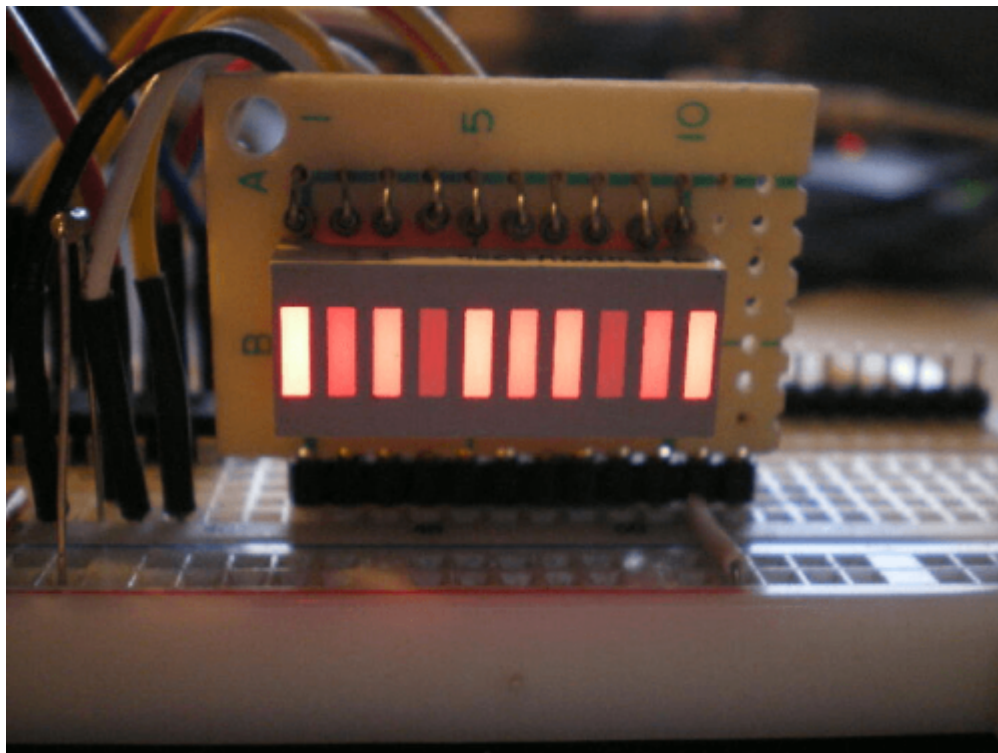
中央が、最も明るく、左右に行くほど暗くなります。



中央が、最も暗く、左右に行くほど明るくなります。



ランダムに輝度を設定しました。



如何ですか? 本ユニットを8台接続すると、80個のLEDの輝度を制御することができるので、クリスマスのイルミネーション等には打って付けですね。また□LEDの色を変えたり、フルカラ□LEDを接続することにより、とても幻想的な雰囲気を出し出す事ができるのではないのでしょうか? 😊

フルカラ□LEDを接続すると、1000色の色を表現できます。赤色(R)10段階×緑色(G)10段階×青色(B)10段階

著作権表示 copyright notice

このページは稲崎様の閉鎖したHPのコピーで、著作権は稲崎様にあります。詳細 This page is a copy of Mr. Inasaki's closed website, and the copyright is held by him. Details

From:

<http://deepsky.jp/wiki/> - うごくといいな

Permanent link:

<http://deepsky.jp/wiki/doku.php?id=elechobby:picdic:pic16f88:99>

Last update: 2020/05/01 19:05

