

電子サイコロ

概要

いろいろなゲームに使われる、サイコロ(ダイス)を、出来るだけ小さくを目標に、8ピンのPICで実現してみました。

動作原理

8ピンのPICではI/O数の制約(電源を除くと6個)が大きいので、当初サイコロには向かないのではと考えました。

- LEDが6個
- スタートスイッチが1個
- ブザーが1個

このように8個のI/Oが必要となり、2個足りません。 18ピンのPICなら、何も問題はないのですが。。

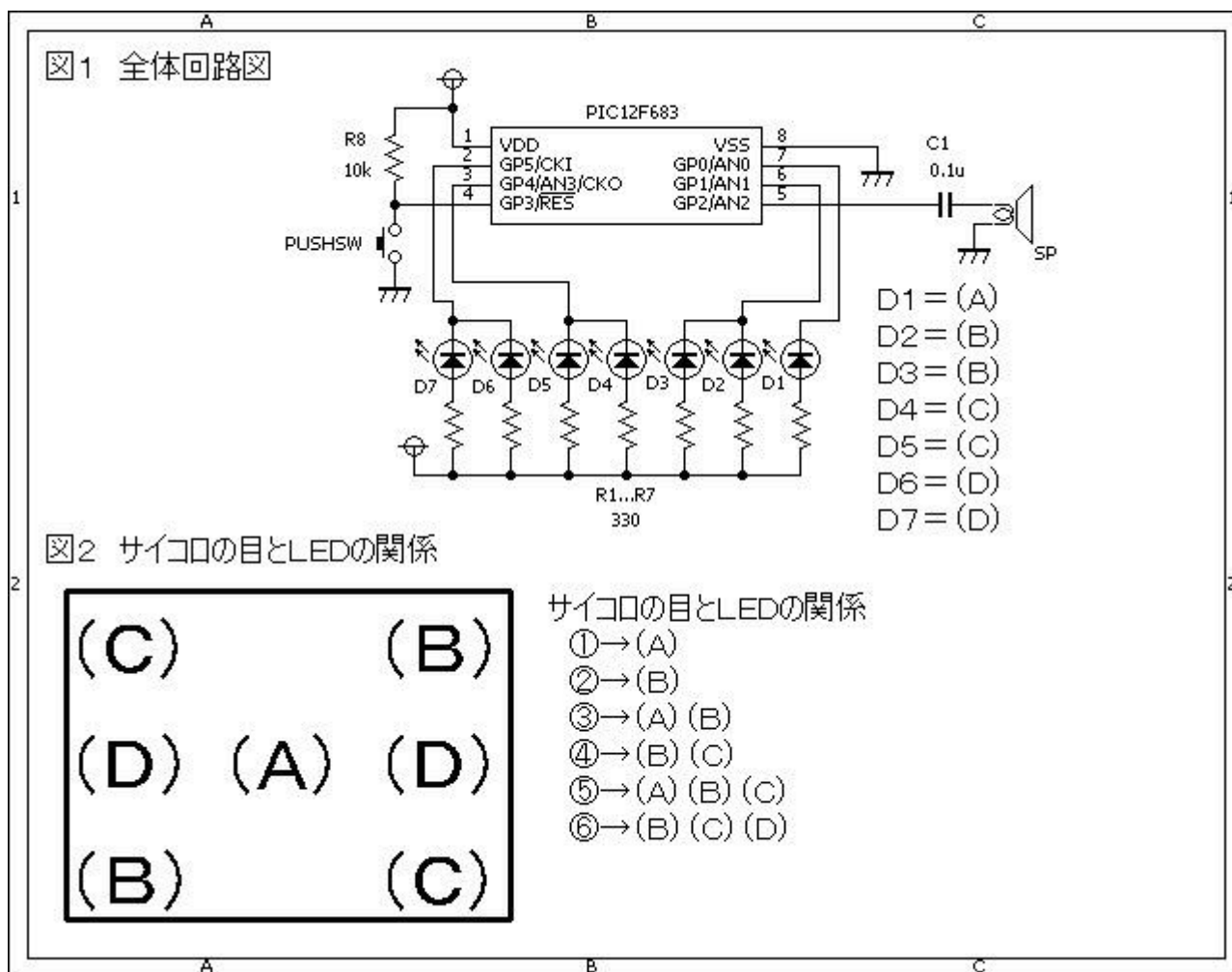
そこで、次のように考えてみました。

- サイコロの1-6迄の目を重ね合わせてみると、図2のように7個のLEDで表現することができます。
- そして7個のLEDを4つのグループに分類します。
- * (A) 1個
- * (B) 2個
- * (C) 2個
- * (D) 2個
- サイコロの目とLEDの関係を次のようにします。
- 1 (A)
- 2 (B)
- 3 (A)(B)
- 4 (B)(C)
- 5 (A)(B)(C)
- 6 (B)(C)(D)

このようにすることでLED部分は、4個のI/Oで足りることになります。

この考え方は、電子マスカットさんの「[青い電子サイコロ](#)」を参考にしました。

回路図



ソースコード

<プログラム上の工夫>

- mikroCには、乱数に関する関数が用意されています。
rand関数 擬似乱数を発生させます。
srand関数 rand関数で発生させる擬似乱数の発生系列を変更します。
- rand関数は、同じ繰返しで擬似乱数を発生させるので、起動し直しても、毎回同じパターンの乱数値になってしまい、利用者に覚えてしまいます。
- そこで、起動時に、srand関数で発生系列を変更するようにしました。その時の系列値は、タイマー割り込みで常にインクリメントしている変数を用います。系列を変更するタイミングは、利用者がプッシュスイッチを押した時としましたので、どの系列になるかは不定です。こうすることによって、起動し直しても系列が異なるので利用者に覚えられることがありません。
- プッシュスイッチを押して、直ぐに結果が出るのでは面白くありませんので、徐々に速度を落としながら結果が出るようにしました。
- またLEDの点灯だけでは、寂しいのでPWMによるブザー音も発生させました。

[dice.c](#)

```
//*****
*

#define      LED_A      GPIO.F0
#define      LED_B      GPIO.F1
#define      LED_C      GPIO.F4
#define      LED_D      GPIO.F5

#define      SW          GPIO.F3

#define      ON          0
#define      OFF         1

//*****
*

static unsigned   int      seed;

void   interrupt(){
    if (INTCON.T0IF == 1) {
        INTCON.T0IF = 0;
        //
        seed++;
    }
}

//*****
*

void   Pwm_Change_DutyEx(unsigned int duty_ratio)
{
    CCPR1L = duty_ratio >> 2;
    CCP1CON.F6 = duty_ratio & 0b00000001;
    CCP1CON.F7 = (duty_ratio & 0b00000010) >> 1;
}

//*****
*

void   diceChange()
{
    int      tmp;
    //
    tmp = rand();
    tmp = (((double)tmp) / 32768.0) * 6.0;
    //
    switch (tmp + 1) {
    case 1:
        LED_A = ON;
        LED_B = OFF;
        LED_C = OFF;
```

```
        LED_D = OFF;
        break;
    case 2:
        LED_A = OFF;
        LED_B = ON;
        LED_C = OFF;
        LED_D = OFF;
        break;
    case 3:
        LED_A = ON;
        LED_B = ON;
        LED_C = OFF;
        LED_D = OFF;
        break;
    case 4:
        LED_A = OFF;
        LED_B = ON;
        LED_C = ON;
        LED_D = OFF;
        break;
    case 5:
        LED_A = ON;
        LED_B = ON;
        LED_C = ON;
        LED_D = OFF;
        break;
    case 6:
        LED_A = OFF;
        LED_B = ON;
        LED_C = ON;
        LED_D = ON;
        break;
    }
}

//*****
*

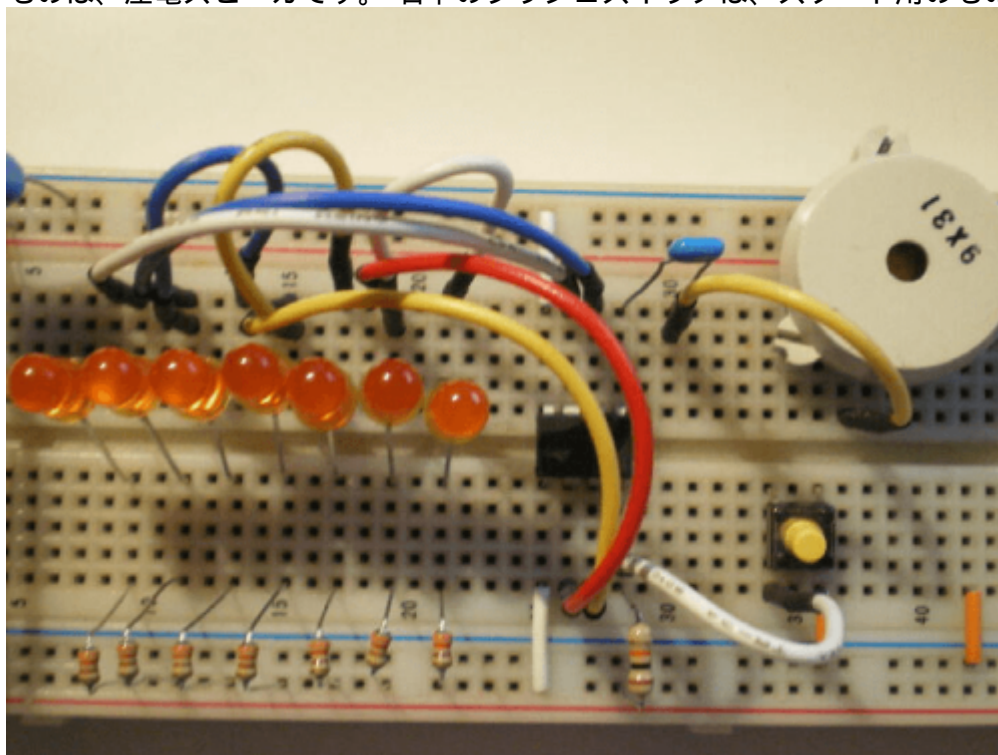
void main()
{
    char    cnt;
    //
    OSCCON = 0b01111000;    // クロックは8Mhz
    CMCON0 = 0b00000111;    // コンパレータは使用しない。
    ANSEL  = 0b00000000;    // □□□変換を使用しない。
    TRISIO = 0b00001000;
    //TIMER0の設定
    INTCON.T0IE = 1;
    INTCON.T0IF = 0;
    OPTION_REG.T0CS = 0;
```

```
OPTION_REG.PSA = 0;
OPTION_REG.PS0 = 0;
OPTION_REG.PS1 = 0;
OPTION_REG.PS2 = 0;
//
Pwm_Init(1200);
Pwm_Change_DutyEx((PR2 * 4) / 2);
//
LED_A = OFF;
LED_B = OFF;
LED_C = OFF;
LED_D = OFF;
//
INTCON.PEIE = 1;    // これ以降の処理で割り込みを許可する。
INTCON.GIE = 1;    // これ以降の処理で割り込みを許可する。
//スイッチが押されるまでLEDの点滅を繰り返す。
while (SW == OFF) {
    LED_A = ON;
    LED_B = ON;
    LED_C = ON;
    LED_D = ON;
    Delay_ms(100);
    LED_A = OFF;
    LED_B = OFF;
    LED_C = OFF;
    LED_D = OFF;
    Delay_ms(100);
}
//スイッチが離されるのを待つ。
while (SW == ON) {
    Delay_ms(100);
}
//ブザー音を鳴らす。
Pwm_Start();
Delay_ms(200);
Pwm_Stop();
//乱数の種をセットする。
srand(seed);
//
while (1) {
    //スイッチが押されるまで待つ。
    if (SW == OFF) {
        Delay_ms(10);
        continue;
    }
    //サイコロの目を切り替える。
    for (cnt = 0; cnt < 30; cnt++) {
        diceChange();
        //ブザー音を鳴らす。
        Pwm_Start();
        Delay_ms(100);
    }
}
```

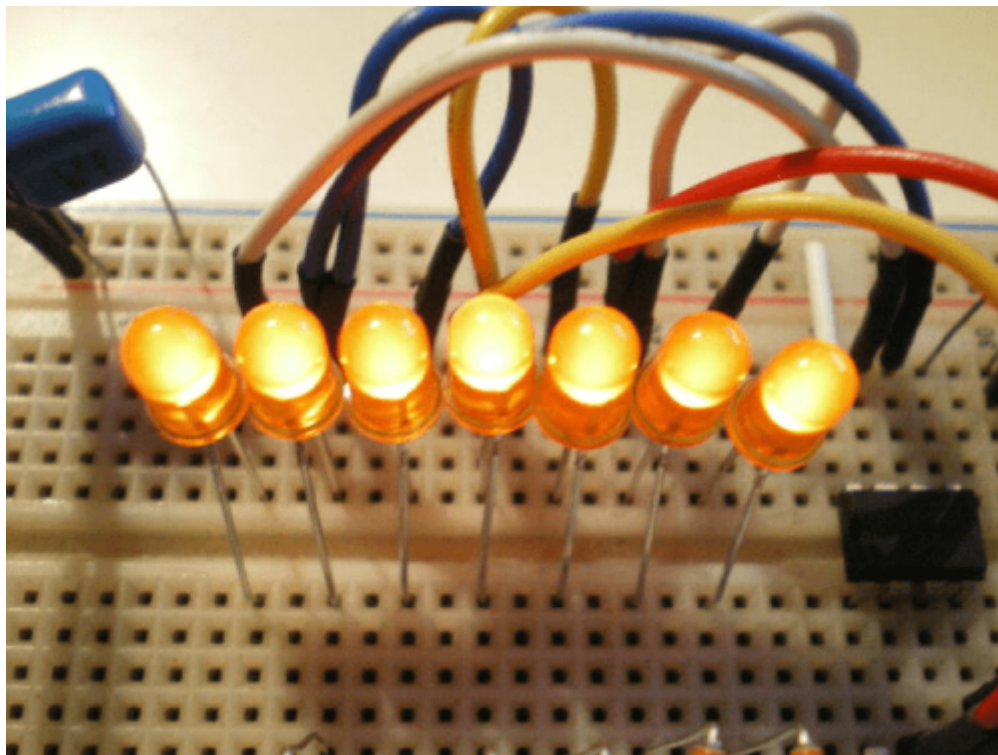
```
Pwm_Stop();  
//可変型スリープ  
Vdelay_ms(50 + (cnt * 10));  
}  
}  
}  
  
//*****  
*
```

動作確認

ブレッドボードで確認したのでLEDをサイコロの目のように並べていません。基板等に組み込むときには、並べてください。向かって左から(A)(B)(B)(C)(C)(D)(D)の順になります。右上の大きい丸状のものは、圧電スピーカです。右下のプッシュスイッチは、スタート用のものです。

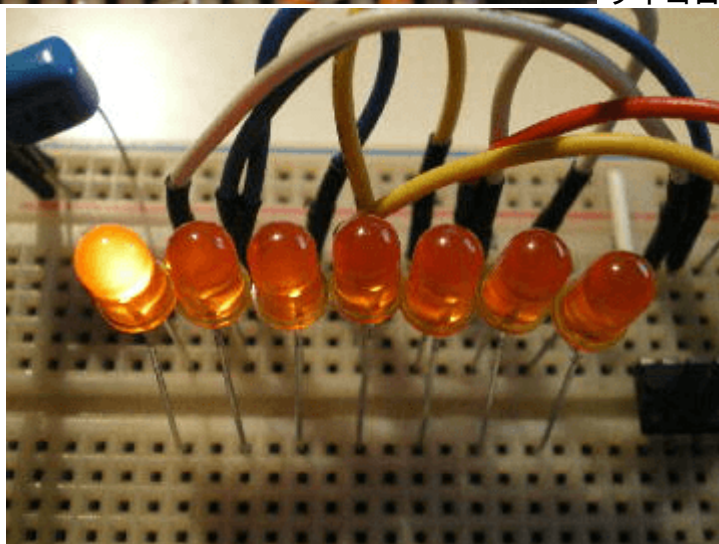


起動時には、全LEDが点滅しますので、この間にプッシュスイッチを押してください。そうすることにより擬似乱数の発生系列が

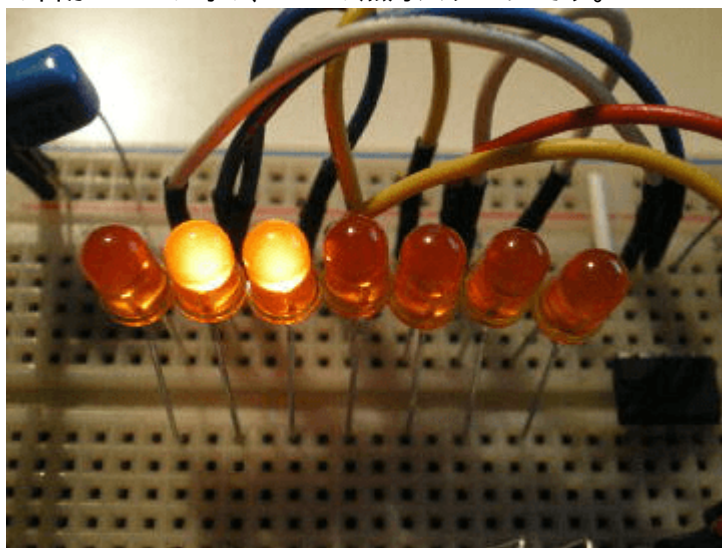


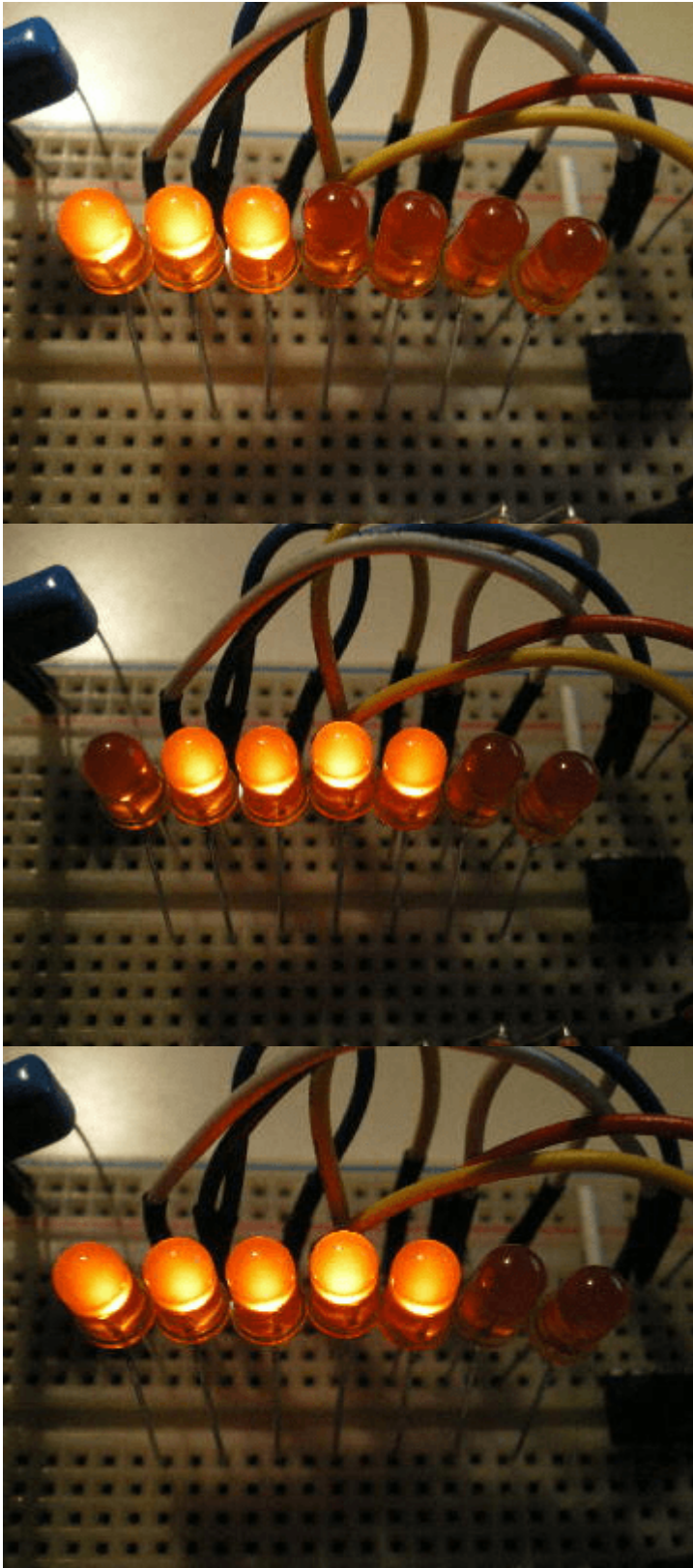
変更されます。

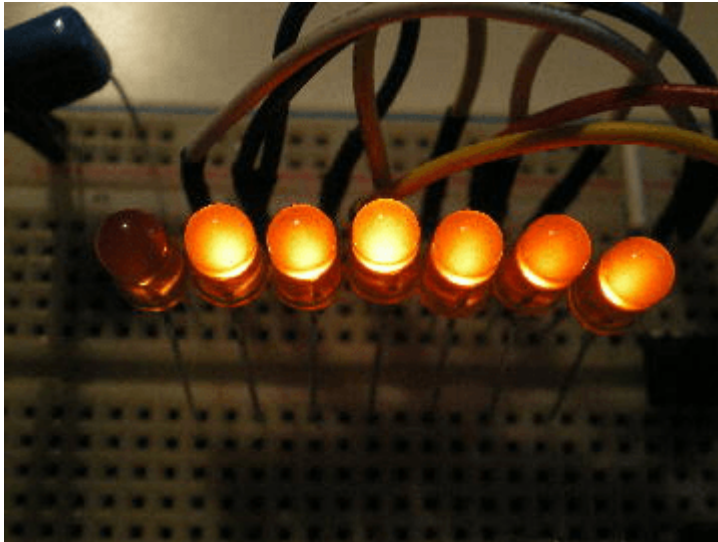
サイコロ



の目が1～6の時の、LEDの点灯パターンです。







幅が広がるかもしれませんね。{😊}!

如何ですか? これを2セット用意するとゲームの

From:

<http://deepsky.jp/wiki/> - うごくといいな

Permanent link:

<http://deepsky.jp/wiki/doku.php?id=elechobby:picdic:pic12f683:16&rev=1588066343>

Last update: 2025/10/17 14:27

