拍手スイッチ(パターン信号検知)

概要

夜中に部屋の明かりを点灯したいときに、暗闇ではスイッチを探すのに苦労します。 そこでスイッチではなく、拍手(ある種のパターン性のある)によって、明かりのON/OFFができればとても 便利ですね。

今回は、事前に拍手のパターン(基準値)を登録し、以降、その基準値と同じパターンの拍手があれば□LEDを 点滅させるようにしてみました。

これを応用すればいろいろな制御が出来そうです。

- 灯りのON/OFF
- 扇風機のON/OFF
- TVのON/OFF

動作原理

◎無音時(基準電圧)を認識する仕組み

- 1msec周期の割り込みを発生させる。
- 割り込み処理の中で、音声データ(A/D変換)を取り込む。□V1
- 1秒(1000回の割り込み)毎に累積したV1の平均値(基準電圧)を求める□□V2

◎拍手(音声信号)パターンを検出する仕組み

- 1msec周期の割り込みを発生させる。
- 割り込み処理の中では、音声データを10回測定する。
- 測定した電圧が、基準電圧(V2)よりも設定した回数以上(今回は3回)高ければ、音声を検出したとみなし、カウントアップする□□CNT1
- 100msec毎に、CNT1の値が設定した回数以上(今回は5回)あれば、拍手を検出したとみなす。
- これを更に1.6秒(16回)繰り返し、拍手パターンとして認識する。

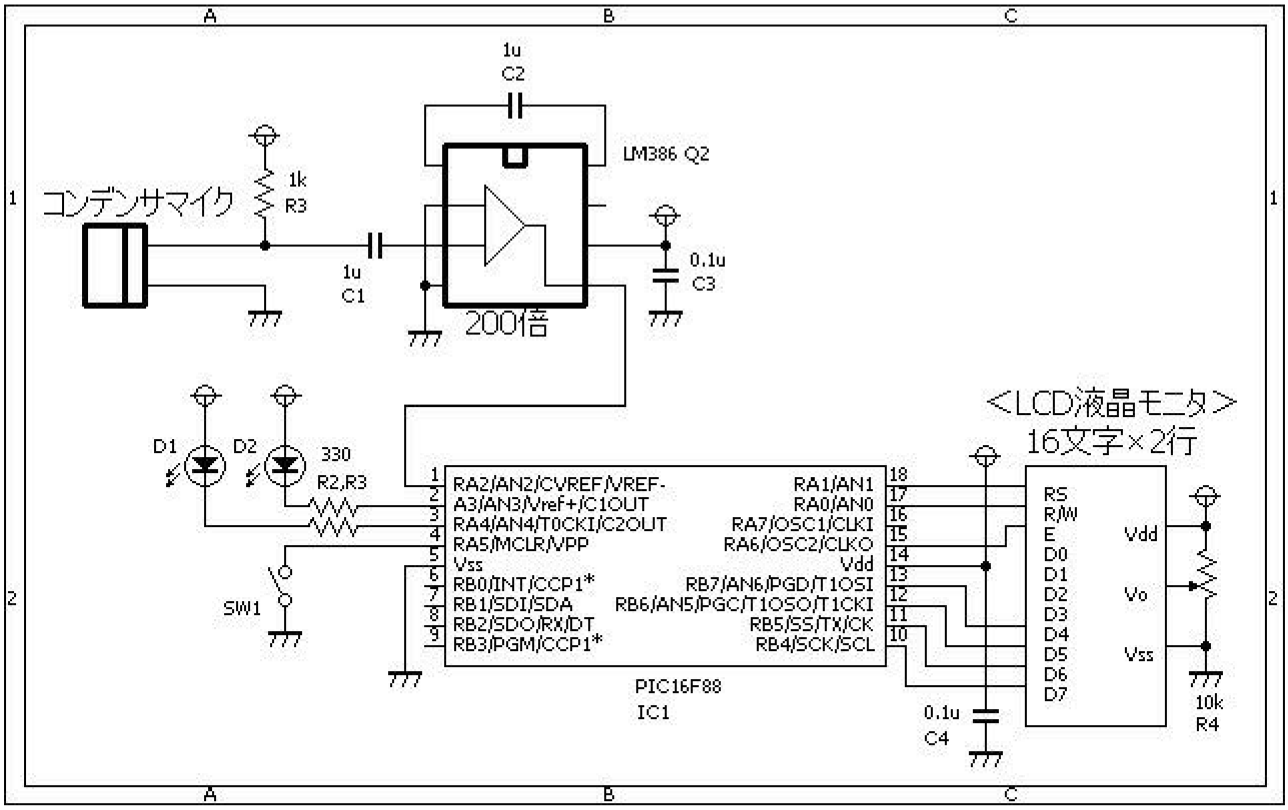
◎基準となる拍手パターンを登録する仕組み

- SWがOFF(1)の時には、検出した拍手パターンを基準拍手パターンとして登録する。

◎比較する拍手パターンと基準拍手パターンを比較する仕組み

- SWがON(0)の時には、検出した拍手パターンと基準拍手パターンを比較し、同一であればLEDを点滅させる。

回路図



ソースコード

ClapSwitch.c

```
//*****************************************************************************
*
/*
  <拍手スイッチ>
  ■機能概要
     ・拍手パターンを記録（基準パターン）する。(SW=1)
     ・拍手パターンを取り込み、基準パターンと比較する。(SW=0)
     ・もしも、等しければLEDを5回点滅させる。
  ■表示内容
     ・基準用の拍手パターンデータ(SW=1)
     ・比較用の拍手パターンデータ(SW=0)
□■CONFIGの設定
     ・LVP_OFF
□□・MCLR_OFF
□□・WDT_OFF
□□・INTRC_IO
*/
//*****************************************************************************
*

#define       LED1        PORTA.F3
#define       LED2        PORTA.F4
#define       SW            PORTA.F5
```

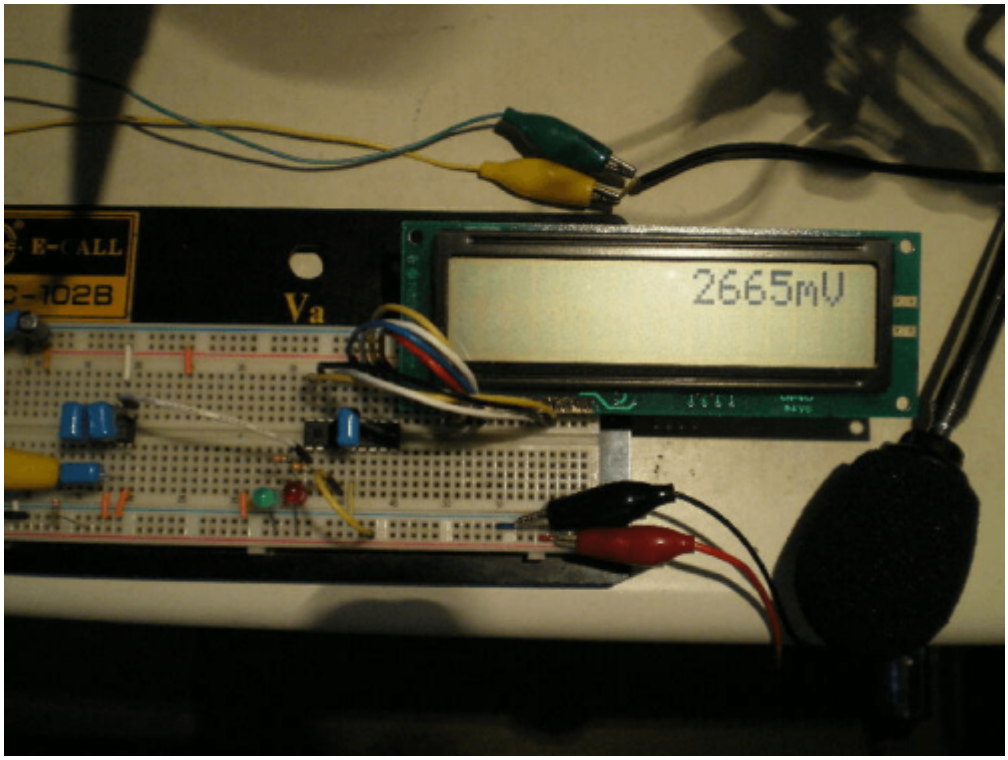
```
//*****************************************************************************
*

static unsigned    int        aveAd;
static unsigned    long    tmpAd;
static unsigned    int        cntAd, aveCnt;

static unsigned    char    highCnt, startFlag, cycleCnt;

static unsigned    int        clapData;

//*****************************************************************************
*

char   VoiceCheck()
{
    unsigned    char    cnt, cnt2;
    //
    cnt2 = 0;
    for (cnt = 0; cnt < 10; cnt++) {
        if (Adc_Read(2) >= (aveAd + 20))
            cnt2++;
    }
    if (cnt2 > 3)
        return(1);
    else
        return(0);
}

//*****************************************************************************
*

void   interrupt()
{
    static    unsigned    int        ad;
    //□□□□□□の割り込み処理
    if (PIR1.CCP1IF == 1) {
        PIR1.CCP1IF = 0;
        //アナログデータを□□□変換で取り込み累積する。
        ad = Adc_Read(2);
        tmpAd += ad;
        aveCnt++;
        //平均電圧を求める。
        if (aveCnt == 1000) {
            aveAd = tmpAd / 1000;
            tmpAd = 0;
            aveCnt = 0;
        }
        //開始するかどうかを判断する。
        if (startFlag == 0) {
```

```
            return;
        }
        //最初の拍手かを判断する。
        if ((startFlag == 1) && (VoiceCheck() == 1)) {
            startFlag = 2;
            clapData = 0x0001;
            cycleCnt++;
            LED2 = 0;
            return;
        }
        if (startFlag == 2) {
            if (VoiceCheck() == 1) {
                highCnt++;
            }
            //0. 1秒毎にhighかlowかをチェックする。
            cntAd++;
            if (cntAd == 100) {
                //
                if (highCnt > 5) {
                    clapData = clapData | 0x0001;
                } else {
                    clapData = clapData | 0x0000;
                }
                clapData = clapData << 1;
                cntAd = 0;
                highCnt = 0;
                //
                cycleCnt++;
                if (cycleCnt == 16) {
                    startFlag = 0;
                    cycleCnt = 0;
                    LED2 = 1;
                }
            }
        }
    }
}

//*********************************************************************
*

void main()
{
    static    unsigned    char    buf[20], cnt;
    static    unsigned    int        clapDataTemp, clapDataSave;
    //
    OSCCON = 0b01110000;    //クロックは8Mhz
    CMCON = 0b00000111;        //コンパレータは使用しない。
    //アナログの設定
    ANSEL  = 0b00000100;    //使用する。
```

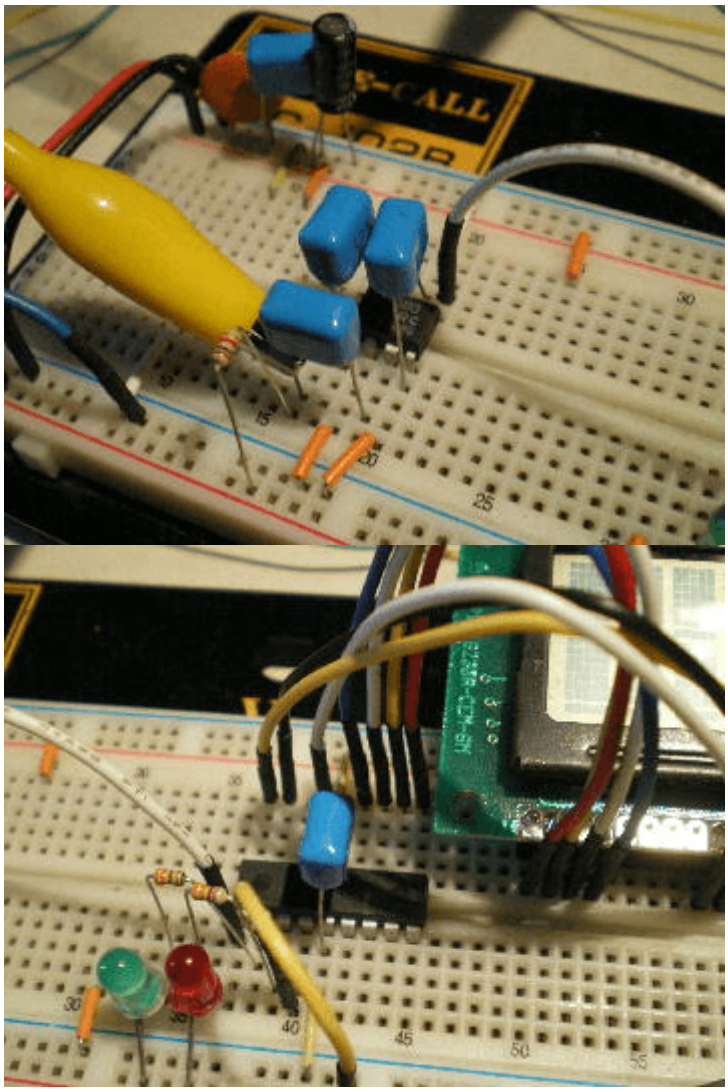
```
    //ポートの設定
    TRISA  = 0b10100100;
    TRISB  = 0b00001111;
    OPTION_REG.F7 = 0;          // PORTBをプルアップする。
    // CCPの設定
    PIE1.CCP1IE = 1;
    PIR1.CCP1IF = 0;
    CCP1CON = 0b00001011;
    CCPR1L = 0xD0;    // 0.001sec...(1÷8000000)*4*2000
    CCPR1H = 0x07;
    // TIMER1の設定
    PIE1.TMR1IE = 0;
    PIR1.TMR1IF = 0;
    TMR1L = 0;
    TMR1H = 0;
    T1CON.T1CKPS0 = 0;
    T1CON.T1CKPS1 = 0;
    T1CON.TMR1ON = 1;
    //変数の初期化
    TMR1L = 0;
    TMR1H = 0;
    // □□□□液晶モニタ）の初期化
    Lcd_Custom_Config(&PORTB,4,5,6,7,&PORTA,1,0,6);
    Lcd_Custom_Cmd(LCD_CURSOR_OFF);
    Lcd_Custom_Out(1, 1, "ClapSwitch V1");
    Delay_ms(1000);
    Lcd_Custom_Cmd(LCD_CLEAR);
    //
    LED1 = 1;          // LED off!
    LED2 = 1;          // LED off!
    aveAd = 0;
    aveCnt = 0;
    tmpAd = 0;
    cntAd = 0;
    highCnt = 0;
    cycleCnt = 0;
    clapData = 0;
    startFlag = 0;
    //割り込み(全体)の設定
    INTCON.PEIE = 1;
    INTCON.GIE = 1;
    //
    while (aveAd == 0)
        ;
    //
    WordToStr(aveAd * 5, buf);
    Lcd_Custom_Out(1, 9, buf);
    Lcd_Custom_Out(1, 14, "mV");
    //
    while(1) {
        //
```

```
        cntAd = 0;
        highCnt = 0;
        cycleCnt = 0;
        clapData = 0;
        startFlag = 1;
        while(startFlag != 0) {
            Delay_ms(1);
        }
        if (SW == 1) {     //拍手のパターンを記録する。
            clapDataTemp = clapData;
            for (cnt = 0; cnt < 16; cnt++) {
                if ((clapDataTemp & 0x8000) != 0)
                    Lcd_Custom_Chr(1, cnt + 1, 0xFF);
                else
                    Lcd_Custom_Chr(1, cnt + 1, ' ');
                clapDataTemp = clapDataTemp << 1;
            }
            //
            clapDataSave = clapData;
        } else {     //拍手のパターンを比較する。
            clapDataTemp = clapData;
            for (cnt = 0; cnt < 16; cnt++) {
                if ((clapDataTemp & 0x8000) != 0)
                    Lcd_Custom_Chr(2, cnt + 1, 0xFF);
                else
                    Lcd_Custom_Chr(2, cnt + 1, ' ');
                clapDataTemp = clapDataTemp << 1;
            }
            //
            if (clapData == clapDataSave) {
                for (cnt = 0; cnt < 5; cnt++) {
                    LED1 = 0;
                    Delay_ms(100);
                    LED1 = 1;
                    Delay_ms(100);
                }
            }
        }
    }
}

//*****************************************************************************
*
```

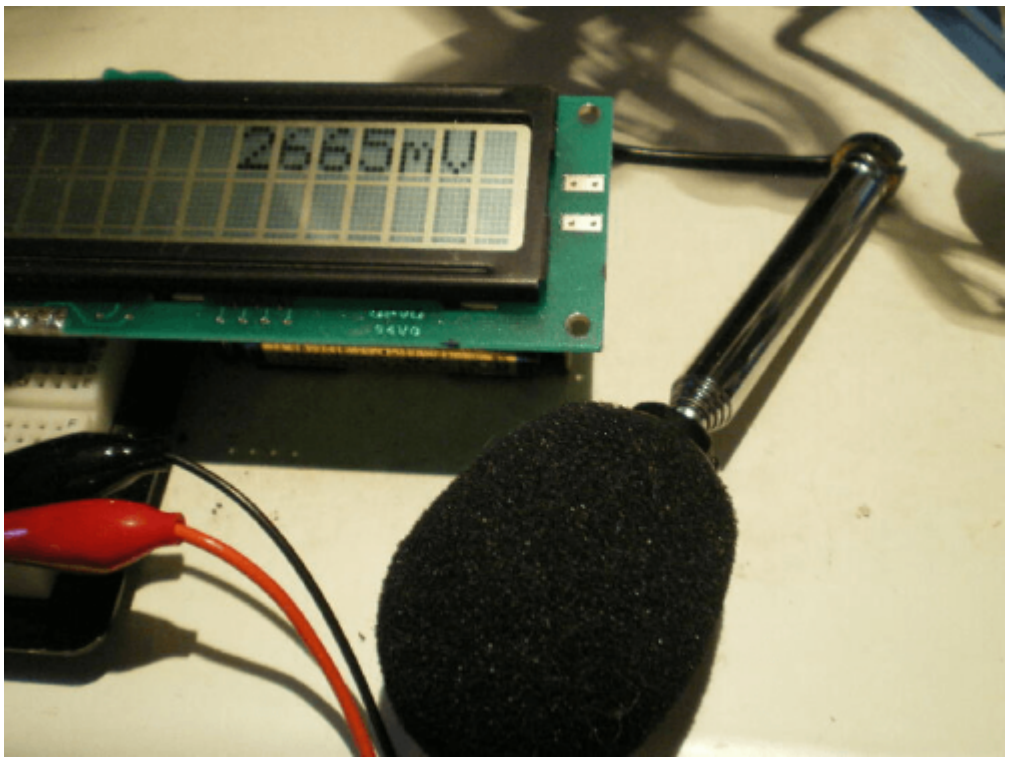
動作確認



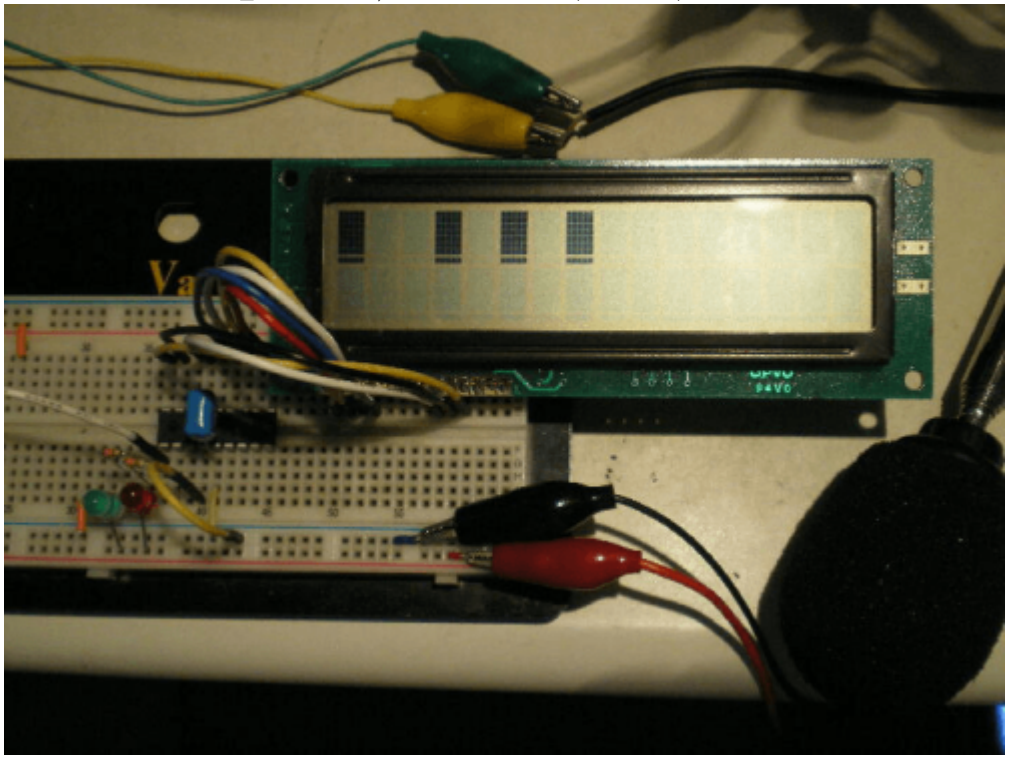
左側:LM386を利用したアンプ部分です。約200倍の増幅率があります。 右側:PIC16F88周りです。



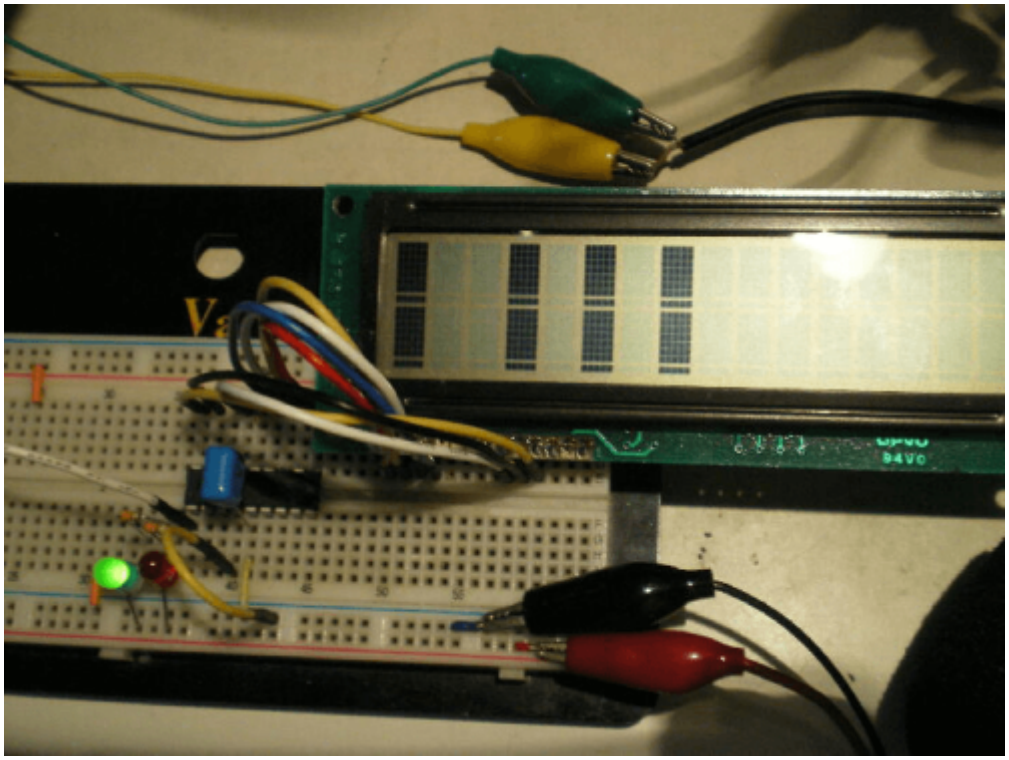
LCDおよびコンデンサマイクです LCDに表示されている電圧は、無音時のLM386の出力電圧です。→基準電圧 この電圧よりも高いときに何らかの音声信号が入力されたものと判断します。



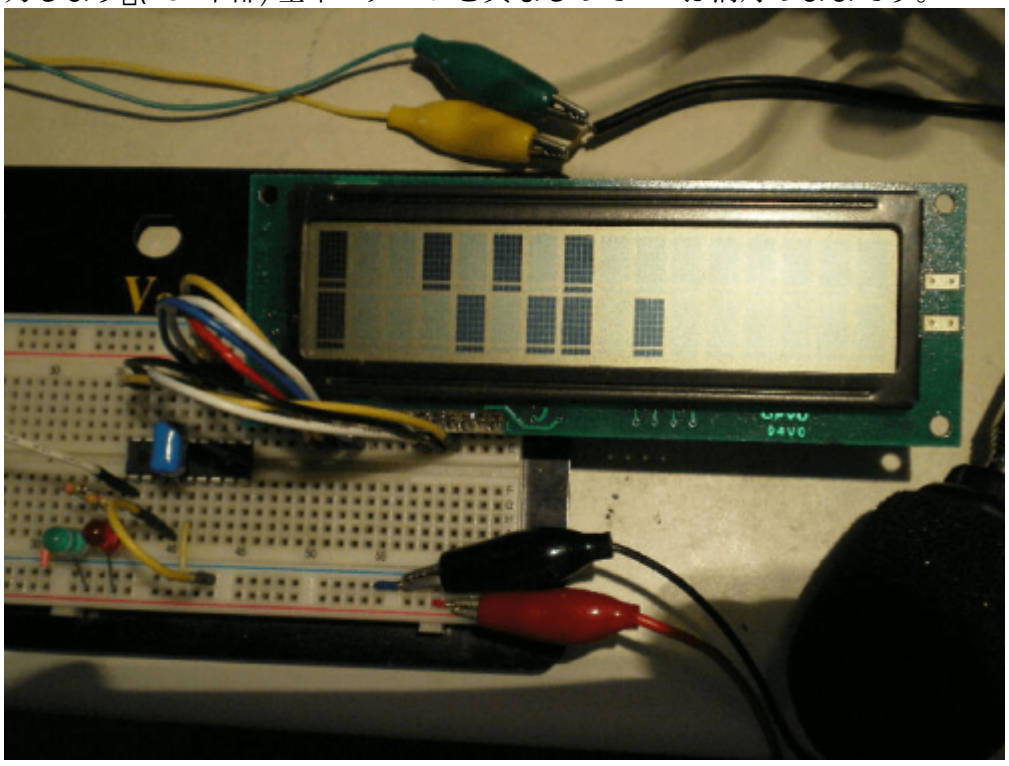
基準となる拍手パターンを入力し記憶します(LCD上部) パン、パパパ(-・・・)というような感じです。



比較用の拍手パターンを入力します(LCD下部) 基準パターンと同じなのでLEDが点滅します。



比較用の拍手パターンを入力します(LCD下部) 基準パターンと異なるのでLEDは消灯のままです。



如何ですか? いろいろなパターンで試してみてください。 図のようなモールス符号風にしてみるのも宜しいのでは。。。{ }!

A	・－	J	・－－－	S	・・・
B	－・・・	K	－・－	T	－
C	－・－・	L	・－・・	U	・・－
D	－・・	M	－－	V	・・・－
E	・	N	－・	W	・－－
F	・・－・	O	－－－	X	－・・－
G	－－・	P	・－－・	Y	－・－－
H	・・・・	Q	－－・－	Z	－－・・
I	・・	R	・－・		

※“ - “ はパン、“ ・ ” はパ