simo85

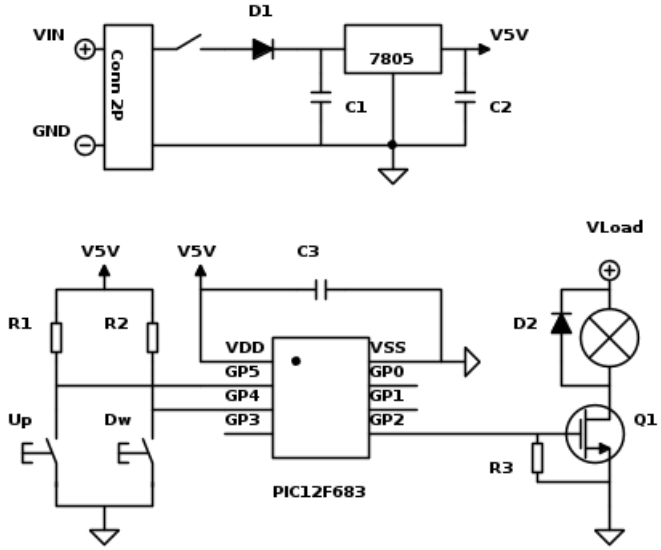
PIC DIMMER

9 November 2011

Introduzione

Quello a presentare è un piccolo dimmer fatto con un **PIC12F683**. Il circuito è molto semplice, ed è stato concepito per una lampadina 12VDC, e consiste di un regolatore di tensione **7805** package TO-220, due pulsanti per l' incremento o decremento del duty cycle ed un MOSFET a canale N usato come driver per la lampadina.

Circuito



- **R1 = R2: 470k**
- **R3 = 1M**
- **D1 = D2: 1N4007**
- **C2 = C2 = C3: 100nF**

La regolazione di luminosità viene effettuata tramite PWM ad una frequenza di 312.5Hz. Grazie a due push button esterni, rispettivamente **Up** & **Dw**, è possibile incrementare o decrementare il duty cycle.

D1 serve come protezione per il circuito e la alimentazione del microcontrollore. Il funzionamento di quest'ultimo non porta ad un consumo notevole, difatti è il carico visto dal transistore che apporterà un maggior consumo di corrente.

Per Q1 si è scelto un N-Channel MOSFET Logic Level **IRLZ44N** che si rivela una buona soluzione per quando si lavora con segnali digitali a 5V.

Grazie all'uso dei push button rispetto al controllo con potenziometro, a mio parere si ottiene una selezione più accurata della luminosità.

Il firmware scritto in Assembly è stato programmato in maniera tale che, per ogni volta che si preme un push button il duty cycle viene incrementato/decrementato di una unità, mentre se si preme il push button per un tempo superiore a 2 secondi, si ottiene un incremento/decremento ciclico.

Il circuito può essere altrettanto usato come variatore di velocità di un motore DC che rispetti i valori massimi del transistore. **D2** serve come protezione per quest'ultimo.

Firmware

```
;*************************************************
;         DIMER PIC12F683
;         Freq          = 312,5 Hz : 1 / 312,5 = 3,2ms
;
;         3.2ms         = [(x + 1) ] · 16u
;         PR2          = 0xc7
;         T2CON          = 0x07
;
;*************************************************

INCLUDE P12F683.INC
LIST p = 12F683

__CONFIG _CP_OFF&_CPD_OFF&_BOD_ON&_PWRTE_OFF&_WDT_OFF&_INTOSCIO&_MCLRE_OFF

CBLOCK        0x20
        W_TMP
        S_TMP
        P_TMP
        DCYCLE
        CNTR
ENDC
```

```
        org             0x000

        bsf STATUS,     RP0
        bcf STATUS,     RP1

        clrf        WPU
        clrf        INTCON
        movlw        0x86
        movwf        OPTION_REG
        clrf        ANSEL
        movlw        0x38
        movwf        TRISIO
        movlw        0xc8
        movwf        PR2
        bcf             STATUS,     RP0
        movlw        0x0c
        movwf        CCP1CON
        movlw        0x07
        movwf        T2CON
        clrf        CCPR1L
        clrf        DCYCLE

MAIN:
        btfss   GPIO,       5
        goto        INC
        btfss   GPIO,       4
        goto        DEC
        goto        MAIN


INC:
        btfsc   GPIO,       5
        goto        MAIN
        call        DELAY20MS
        clrf        CNTR
        movf        CCPR1L,     W
        movwf        DCYCLE
        movlw        0xfe
        subwf        DCYCLE
        btfsc        STATUS,     Z
        goto        MAIN
        incf        CCPR1L,     1
```

```
        btfsc   GPIO,        5
        goto        MAIN

T2I_LOOP:
        movlw        0x60
        subwf        CNTR

        btfss        STATUS,        Z
        goto        I_LOOP2S
        goto        I_10MS

I_LOOP2S:
        btfsc        GPIO,        5
        goto        MAIN
        incf        CNTR,        1
        call        DELAY20MS
        goto        T2I_LOOP

I_10MS:
        btfsc        GPIO,        5
        goto        MAIN
        movf        CCPR1L,        W
        movwf        DCYCLE
        movlw        0xfe
        subwf        DCYCLE
        btfsc        STATUS,        Z
        goto        MAIN
        movlw        0xfa
        movwf        CNTR
        incfsz        CNTR
        call        DELAY20MS
        incf        CCPR1L,        1
        goto        I_10MS

DEC:
        btfsc   GPIO,        4
        goto        MAIN
        call        DELAY20MS
        btfsc        GPIO,        4
        goto        MAIN
        movf        CCPR1L,        1
        btfsc        STATUS,        Z
        goto        MAIN
```

```
        decf        CCPR1L,        1
        btfsc   GPIO,        4
        goto        MAIN

        clrf        CNTR
T2D_LOOP:
        movlw        0x60
        subwf        CNTR,

        btfss        STATUS,        Z
        goto        D_LOOP2S
        goto        D_10MS

D_LOOP2S:
        btfsc        GPIO,        4
        goto        MAIN
        incf        CNTR,        1
        call        DELAY20MS
        goto        T2D_LOOP

D_10MS:
        btfsc        GPIO,        4
        goto        MAIN
        movf        CCPR1L,        1
        btfsc        STATUS,        Z
        goto        MAIN

        movlw        0xfa
        movwf        CNTR
        incfsz        CNTR
        call        DELAY20MS
        decf        CCPR1L,        1
        goto        D_10MS

DELAY20MS
        movlw        0x64
        movwf        TMR0

        btfss        INTCON,        T0IF
        goto        $-1

        bcf                INTCON,        T0IF
        return
```

```
end
```

Conclusioni

Il programma può essere compilato usando **MPLAB** versione **8.6**. Nello schema sono state omesse le connessioni ISCP in quanto ogni circuito programmatore può adoperare l'ordine di connessioni che più preferisce o ritiene opportuno. In ogni caso si consiglia di fare riferimento al datasheet con link riportato sopra o eventualmente consultare la **ISCP Guide** della Microchip.

Spero che il piccolo progetto possa essere di aiuto ed utile per gli utenti che leggeranno l'articolo.

Simone