PRACTICAS DE INSTRUMENTACION EN EL MICROCONTROLADOR PIC16f1619

ALUMNO Residente:

GABRIEL LEE ALVAREZ ROSADO

PROFESOR:

M.I.E. RICARDO JIMENEZ GARCIA

Oct/05/20

INDICE

PRACTICA 0 – SELECCIÓN DE PRÁCTICAS CON INTERRUPCIONES

Para poder implementar un gran número de prácticas en un mismo microcontrolador, es necesario que la instrucción de cambiar de practica sea correcta, para esto se utilizó la interrupción del puerto A, que nos asegura la lectura de cambio de práctica.

para esto se habilitaron las interrupciones internas con el registro INTCON, el cual se habilitaron las interrupciones globales (GIE = 1), las interrupciones periféricas (PEIE = 1) y las interrupciones en cambio (IOCIE = 1).

Al activar las interrupciones en el cambio (IOCIE), se eligió que ocurriera en el flanco de bajada, o sea, cuando es presionado el botón utilizado, ya que está funcionando en configuración pull-up, este botón se colocó en el pin RA3, el cual fue configurado como entrada, con el pull-up interno encendido, la configuración del Pull-Up es mostrada en la **Figura 1**.

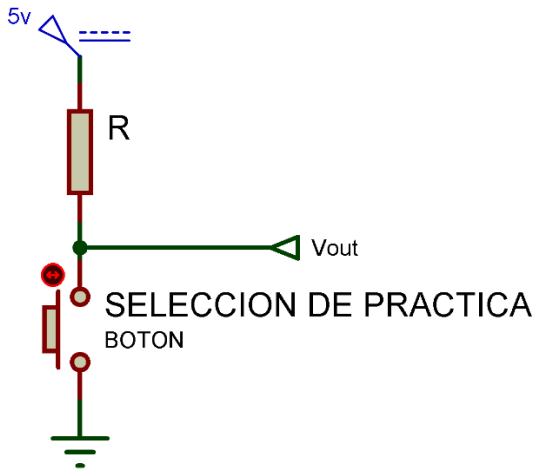


Figura 1. *Circuito configuración Pull-UP*

Para activar la interrupción cuando es presionado el botón, es necesario que sea activada la interrupción cuando el voltaje cambia de 5v a 0v o de un estado lógico 1 a 0, se puede ver el cambio en la **Figura 2**, para esto se utilizó el registro **IOCAN**, el cual nos permite seleccionar que pin del puerto A que queremos activar una interrupción en el flanco de bajada, en nuestro caso fue el bit 3 o RA3 (IOCAN3 = 1).

Al tener configuradas las interrupciones es necesario indicar que se ejecutara al realizar una interrupción en el lenguaje PBP, para esto se utiliza la instrucción **ON INTERRUPT GOTO**, esta función nos permite seleccionar la subrutina a utilizar cuando ocurra una interrupción.

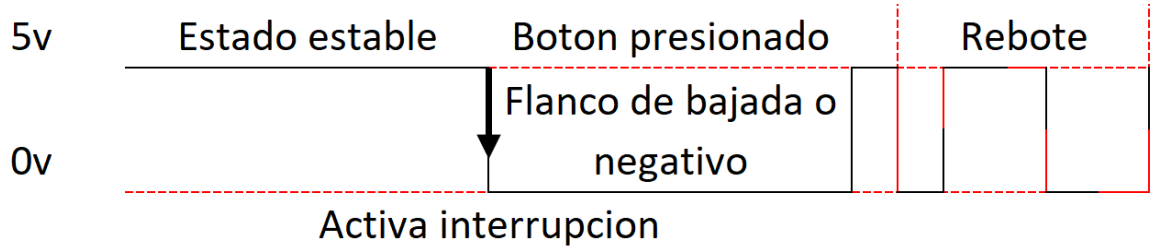


Figura 2. *Cambio de estado de la señal de un botón en pull-up*

Para este caso se seleccionó la etiqueta SEL que tiene la subrutina a ejecutar cuando se realice una interrupción.

Antes de entrar a una subrutina, es importante colocar la función **DISABLE** antes de la subrutina; Esta función deshabilita las interrupciones debajo de la línea donde este colocada, debido a que es importante no salir de la subrutina por rebotes ocasionados por el botón; Después de la subrutina es necesario utilizar la función **ENABLE**, que habilita las interrupciones debajo de la línea de código que se encuentre.

En este caso la subrutina consta en sumar 1 a una variable que es la que determina la practica en la que se encuentra, al igual de una pausa de 200 mSeg para evitar el rebote del botón. Es necesario borrar las banderas después de ejecutar la subrutina. Se utiliza la instrucción **RESUME** que nos devuelve a la sección del código que estábamos colocados anteriormente, el código de la subrutina de ejemplo se puede ver a continuación:

```
DISABLE; DISABLE INTERRUPT
SEL:  ;PRACTICE SELECTION
   c=c+1;
   IF C >6 THEN C = 1;  THE MAX PRACTICES IS 6 IN THIS CASE
   PAUSE 200
   IOCAF.3 = 0; CLEAR FLAG
resume
ENABLE; ENABLE INTERRUPT
```

Al regresar a la practicas, se encuentran unas líneas de código con la siguiente estructura:

```
IF C<>1 THEN START; IF DIFFERENT PRACTICE
```

En donde se pregunta si la práctica es distinta a la seleccionada, si el número de practica es distinto a la práctica que esta, se va a la etiqueta **START**, que se encarga de dirigirnos a la práctica seleccionada con un switch case.

Algoritmo Practica 0

```
on interrupt goto SEL; IF THERE INTERRUPT GO TO LABEL SEL
INTCON = %11001000;ENABLE INTERRUPT
IOCAN = %001000; RA3 INTERRUPT

START:      ; SELECT THE PRACTICE TO USE

SELECT CASE C
   CASE 1
      GOTO PRACTICE_1
   CASE 2
      GOTO PRACTICE_2
   CASE 3
      GOTO PRACTICE_3
   CASE 4
      GOTO PRACTICE_4
   CASE 5
      GOTO PRACTICE_5
   CASE 6
      GOTO PRACTICE_6
   END SELECT
GOTO START


DISABLE; DISABLE INTERRUPT

SEL:  ;PRACTICE SELECTION
   c=c+1;
   IF C >6 THEN C = 1;  THE MAX # of PRACTICES IS 6
   PAUSE 200
   IOCAF.3 = 0; CLEAR FLAG

resume

ENABLE; ENABLE INTERRUPT
```

PRACTICA 1 – VOLTIMETRO CON DISPLAY

La práctica uno, consiste en la medición de voltaje aplicado al microcontrolador.

Para realizar la práctica se utiliza el módulo ADC de 10 bits incorporado en el microcontrolador PIC16f1619; Para iniciar tenemos que determinar la resolución de cada bit para saber que voltaje estamos obteniendo, para eso utilizamos la Ec.1

$$ADC_{bit} = \frac{V_{MAX}}{2^n - 1} = \frac{5v}{2^{10} - 1} = \frac{5v}{1023} = 4.887mV/bit$$

Ya teniendo la resolución por bit, realizamos la conversión de los bits obtenidos por el ADC. Esto se hace multiplicando la constante por los bits obtenidos, pero para esto es necesario quitar el punto decimal, ya que el lenguaje pbp no permite operaciones con punto flotante, como se ve en la **Ec.2**.

$$REMAINDER = ADC * 4887 \quad [2]$$

Al realizar esta operación, se puede obtener el valor máximo posible con el ADC (1023), podemos ver el resultado en la **Ec.3.**

$$REMAINDER = 1023 * 4887 = 4{,}999{,}401 \quad [3]$$

Al hacer esto se supera el valor máximo de las variables de 16 bits (65,535), por lo que se utiliza la instrucción DIV32 para poder realizar multiplicaciones con hasta 31 bits (2,147,483,647) y ser divididas por una variable de 15 bits (32,767), esto nos permite dividir el valor obtenido por 1000 y solo obtener los valores necesarios. Se puede ver en la **Ec.4.1 y Ec.4.2**. La función se coloca seguida de una multiplicación y esta función toma el resultado de la multiplicación.

$$VIN = DIV32\ 1000 \quad [4.1]$$

$VIN = 4999$; ósea 4.999v [4.2]

Al obtener el voltaje, guardamos los dígitos obtenidos y las guardamos en un vector de 4 espacios con la función **DIG** y un ciclo for para obtener los 4 dígitos. En donde el digito **0** es el primer digito de derecha a izquierda y el digito 3 el primer digito de izquierda a derecha, se puede apreciar en la siguiente tabla:

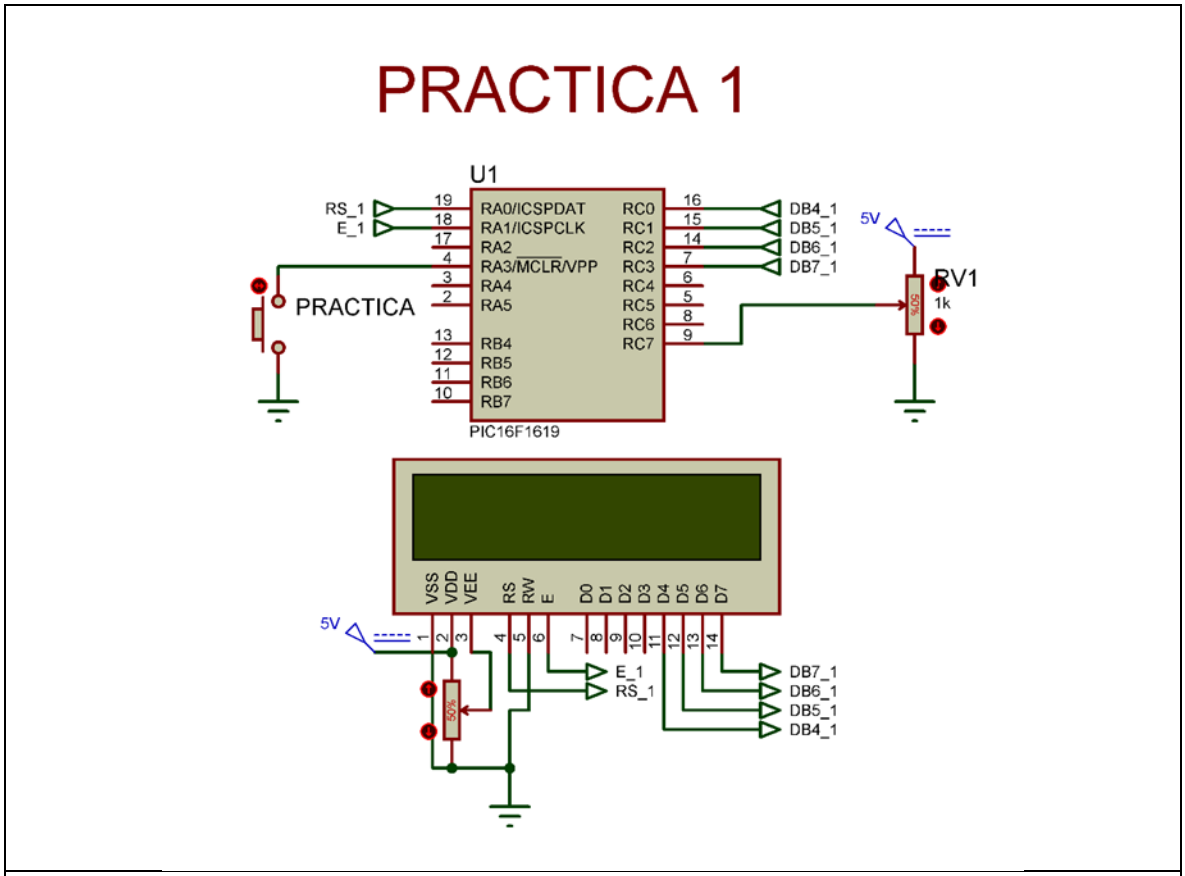
Digito 3	Digito 2	Digito 1	Digito 0
Millares de mv	Centenas de mV	Decenas de mV	Unidades de mV
4	9	9	9

Tabla 1. *Significado de los dígitos.*

Todo esto ocurre al llamar la etiqueta **ADC_READ**, en el cual solo se selecciona el canal con la variable **CH,** en este caso se utilizó el canal AN9, en este caso sería **CH=9**.

Para mostrar en la lcd los dígitos se utilizó la función DEC, que traduce los dígitos a código ASCII para que la lcd pueda entender la información transmitida, al mostrar

la información en la lcd se coloca el punto decimal después del digito 3 para representar el voltaje obtenido.



Algoritmo Practica 1

```
PRACTICE_1:
   GOSUB SECUENCE;
   lcdout $fe,$80,"PRACTICE"    ; MESSAGES
   lcdout $fe,$c0,"-- 1 ---"
   PAUSE 1000
   lcdout $fe,$80,"VOLT    "
   lcdout $fe,$c0,"--------"
   PAUSE 1000;


HERE1:
      CH = 9;CHANNEL 9
      GOSUB ADC_READ; GO TO SEQUENCE
      LCDOUT $FE,$C0,"V=",DEC VD[3],".",DEC VD[2],DEC VD[1],DEC VD[0]," "
      IF C<>1 THEN START; IF DIFFERENT RPACTICE
GOTO HERE1;

ADC_READ:
   CH = CH<<2;
   ADCON0 = CH^%10000001; BITWISE XNOR
```

```
   PAUSE 1
   ADCON0.1 = 1;
   HERE_ADC: IF ADCON0.1 = 1 THEN HERE_ADC; STAY HERE WHILE CONVERSION IS ready
   ADCON0.0 = 0;
   ADC.BYTE0 = ADRESL;     SAVE LOWER REGISTER OF THE ADC IN VARIABLE VIN
   ADC.BYTE1 = ADRESH;     SAVE HIGHER REGISTER OF THE ADC IN VARIABLE VIN
   DISABLE
   REMAINDER = ADC*4887;            MULTIPLYING BY RESLSB = 4.8887
   VIN = div32 1000;            PERFORM 16-BITS DIVISION
   ENABLE
   FOR X = 0 TO 3
      VD[X] = VIN DIG X
   NEXT X
return
```

PRACTICA 2 – MEDICIÓN DE TEMPERATURA CON DISPLAY

La practica 2 consiste en la medición de temperatura con un sensor lm34, que suministra 10mV por cada grado Fahrenheit aplicado.

Para realizar la medición se utiliza el ADC de 10 bits en el microcontrolador PIC16f1619. El cual este se realizará seleccionando el canal del ADC en donde se encuentre colocado el sensor LM34, en este caso es el canal 8 (AN8), y dirigirse a la etiqueta **ADC_READ**, en esta sección se realizará la lectura del voltaje obtenido del sensor y la obtención de los dígitos de manera independiente del voltaje.

Al tener los dígitos del voltaje, es importante mostrarlos de la manera adecuada, al saber que cada 10mV representa un grado, se puede decir que el digito 1 es la unidad de los grados aplicados y el digito 0 es el punto decimal de los grados, como se puede hacer referencia a la **Tabla 1** de la practica 1, se puede observar la **tabla 2**.

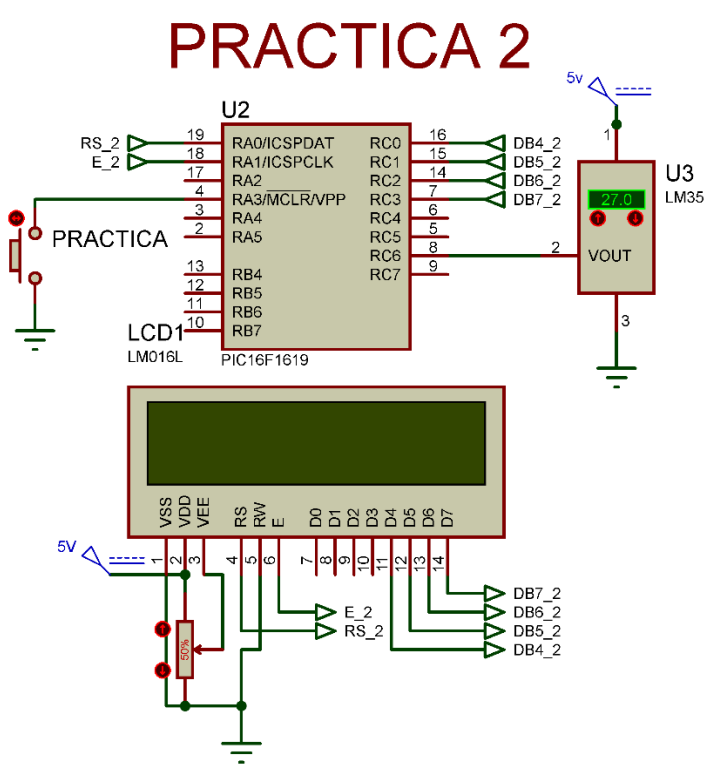
Digito 3	Digito 2	Digito 1	Digito 0
Centenas de ºF	Decenas de ºF	Unidades de ºF	Decima de ºF
4	9	9	9

***Tabla 2**. Orden de los dígitos para mostrar temperatura*

Para mostrarlos con la función **LCDOUT** se colocará el punto después del digito 1, y se utilizará la función **DEC** que traduce los dígitos decimales obtenidos a código ASCII.

Se puede ver en el código siguiente la manera de colocar los datos con la función **LCDOUT** en el inicio de la segunda línea de la pantalla LCD utilizada para mostrar la información.

```
LCDOUT$FE,$C0,"F=, DEC VD[3], DEC VD[2], DEC VD[1],"." ,DEC VD[0], "  "
```



Algorimto practica 2

```
PRACTICE_2:
   GOSUB SECUENCE;
   lcdout $fe,$80,"PRACTICE"
   lcdout $fe,$c0,"-- 2 ---"
   PAUSE 1000
   lcdout $fe,$80,"TEMP    "
   lcdout $fe,$c0,"LM34    "
   PAUSE 1000

   HERE2:
      CH = 8;CHANNEL 8
      GOSUB ADC_READ;GO TO SEQUENCE
      lcdout $FE,$C0,"F=",dec vD[2],dec vD[1],".",DEC VD[0],"  "
      IF C <> 2 THEN START; IF DIFFERENT RPACTICE
   GOTO HERE2

ADC_READ:
   CH = CH<<2;
   ADCON0 = CH^%10000001; BITWISE XNOR
   PAUSE 1
   ADCON0.1 = 1;
   HERE_ADC: IF ADCON0.1 = 1 THEN HERE_ADC; STAY HERE WHILE CONVERSION IS ready
   ADCON0.0 = 0;
   ADC.BYTE0 = ADRESL;     SAVE LOWER REGISTER OF THE ADC IN VARIABLE VIN
```

```
   ADC.BYTE1 = ADRESH;     SAVE HIGHER REGISTER OF THE ADC IN VARIABLE VIN
   DISABLE
   REMAINDER = ADC*4887;            MULTIPLYING BY RESLSB = 4.8887
   VIN = div32 1000;            PERFORM 16-BITS DIVISION
   ENABLE
   FOR X = 0 TO 3
      VD[X] = VIN DIG X
   NEXT X
return
```

PRACTICA 3 – CONTADOR DE PULSOS Y CONSTANTE PARA SENSOR DE FLUJO

La practica 3 consiste en la medición de pulsos obtenidos por el sensor de flujo al suministrar un litro de líquido.

Para realizar la medición de los pulsos, se utilizó el timer 1 para realizar el conteo. Se borran los registros del timer y se inicializa, como se puede ver en el algoritmo a continuación:

```
TMR1L = 0; borrado de los registros
TMR1H = 0;
T1CON.0 = 1; encendido del timer 1
```

Al pasar un litro de líquido por el sensor de flujo se presiona el botón PB1, para indicar que se ha suministrado un litro de líquido, se registra la cantidad de pulsos obtenidos en el timer 1, y se divide entre la cantidad de mililitros. Por ejemplo, si tenemos un total de 1500 pulsos por 1 litro de líquido se obtiene la constante utilizando la **Ec.5**

$$K = \frac{Pulsos}{mL} = \frac{1500\ pulsos}{1000mL} = 1.5\,{}^{pulsos}/_{mL} \qquad [5]$$

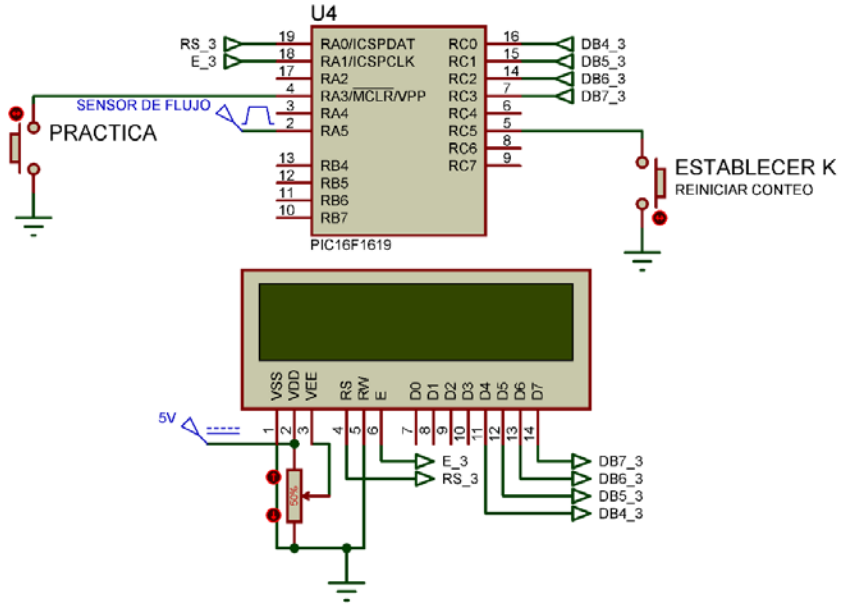
Con lo que obtendremos la constante de 1.5 p/ml, que nos indica que cada 1.5 pulsos es suministrado 1 mililitro de líquido. Pero para poder utilizarlo en el lenguaje PBP se multiplica por 100 para eliminar el punto decimal, ya que no se permiten operaciones con punto flotante, en donde se puede simplificar la división a solo dividir entre 10, podemos suponer que se han obtenido 1500 pulso al suministrado un litro de líquido, como se muestra a continuación:

$$K = \frac{Pulsos * 100}{1000mL} = \frac{Pulsos}{10mL} = \frac{1500}{10mL} = 150\,{}^{p * 100}/_{ml}$$

Al obtener la constante, se guarda en la memoria HEF (High-Endurance Flash Data Memory) utilizando un GOSUB a la etiqueta SEQUENCE, que sigue los pasos predeterminados por el fabricante para guardar datos en la memoria, esto se utiliza para poder ser usar la constante K en la práctica siguiente y no tener que calibrar el sensor o el líquido suministrado nuevamente.

Al terminar el proceso de la obtención de la constante, se reinicia el contador, y espera los pulsos del sensor nuevamente.

PRACTICA 3



Algorimto practica 3

```
PRACTICE_3:
   GOSUB SEQUENCE;
   lcdout $fe,$80,"PRACTICE"
   lcdout $fe,$c0,"-- 3 ---"
   PAUSE 1000
   lcdout $fe,$80,"COUNTER "
   lcdout $fe,$c0,"FLOW    "
   PAUSE 1000
   lcdout $fe,$80,"PULSES X"
   lcdout $fe,$c0,"1 LITRE "
   PAUSE 1000;

   HERE_31:
      LCDOUT $FE,$80,"PULSES  ";
      TMR1L = 0;
      TMR1H = 0;

   HERE3:

      T1CON.0 = 1
      TMR.BYTE0 = TMR1L; LOW REGISTER
      TMR.BYTE1 = TMR1H; HIGH REGISTER
      LCDOUT $FE,$C0,DEC TMR,"       "
      IF PB1 = 0 THEN
            DISABLE
            T1CON.0 = 0
```

```
          K = TMR/10; OBTAIN THE CONSTANT, EXAMPLE 800 PULSES IN
THOUSAND MILLILITER, 800/1000 =0.8
                    ; BE MULTIPLYING BY 100 FOR THE DECIMALS ;   100/1000 = 1/10
          FOR X = 0 TO 2
             KD[X]= K DIG X;
          NEXT X
      LCDOUT $FE,$80,"CONSTANT"
      LCDOUT $FE,$C0,"K=",DEC KD[2],".",DEC KD[1],DEC KD[0],"      ";P/ML
      PAUSE 3000
      TMR1L = 0;
      TMR1H = 0;
      gosub SEQUENCE
      ENABLE
      GOTO HERE_31;
   ENDIF

IF C <> 3 THEN START;IF DIFFERENT RPACTICE
GOTO HERE3

SEQUENCE:        ;data save sequence
   GOSUB ERASE
   DIR = $1FF5
   DAT = 0;
   GOSUB SAVE
   DIR =$1FF6;
   DAT = K;
   GOSUB SAVE;
   DIR =$1FF7
   DAT = C
   GOSUB SAVE
RETURN

SAVE:              ;instruccion set
   PMADRL = DIR.BYTE0;
   PMADRH = DIR.BYTE1;
   PMDATL= DAT.BYTE0;
   PMDATH= DAT.BYTE1;
   PMCON1 =%00000110
   GOSUB UNLOCK;
   STAY:IF (PMCON1.1 = 1) AND (PMCON1.3 = 1)THEN STAY;
   IF PMCON1.3 = 1 THEN save;
   PMCON1=%00000000
RETURN;

ERASE:             ;required sequence
   PMCON1.4 = 1;
   PMCON1.2 = 1;
   GOSUB UNLOCK;
   PMCON1.2=0;
   PMCON1.4=0;
RETURN
```

PRACTICA 4 – CONTROL DE LIQUIDO

La practica 4 consiste en la medición del líquido dispensado y el flujo del agua que es suministrada, utilizando la constante K obtenida en la práctica anterior.

Para esto utilizamos el timer 1 para medir la cantidad de pulsos obtenidos por el sensor.

Para obtener el flujo de líquido a través de los pulsos, es necesario obtener la diferencia de pulsos al transcurrir un segundo. Para esto guardamos los pulsos antes de realizar la próxima lectura en un segundo, esta lectura del timer 1 se guardará en la variable ANT. Al pasar la el tiempo determinado, se realiza nuevamente la lectura del timer 1 y se realizara la diferencia entre una y otra.

El algoritmo para obtener la diferencia se puede ver a continuación:

```
TMR.BYTE0 = TMR1L; leemos el registro bajo
 TMR.BYTE1 = TMR1H; leemos el registro alto
 ANT= TMR; guardamos los pulsos
 PAUSE 1000; pausa de 1 segundo
 TMR.BYTE0 = TMR1L; leemos el registro bajo
 TMR.BYTE1 = TMR1H; leemos el registro alto
 DIF= TMR-ANT;          obtenemos la diferencia de los pulsos
```

La cantidad de líquido suministrado, es la cantidad de conteos guardados en el timer 1.

Para traducir el flujo obtenido y la cantidad de líquido suministrado, es necesario dividir la cantidad de pulsos o la diferencia de pulsos obtenidos entre la constante obtenida en la práctica anterior y multiplicar por 100 para trabajar con las decimales, por ejemplo, si hemos obtenido una cantidad de 1500 pulsos, nuestra constante K es $150^{p*100}/_{mL}$, al igual se obtuvo la lectura anterior de los pulsos fue de 1450.

Se puede ver las **Ec.6** y **Ec. 7** el procedimiento a seguir para obtener el flujo y la cantidad de líquido suministrado, estas operaciones se realizan usando la función **DV32**, que nos permite trabajar con números de mayor tamaño**.**

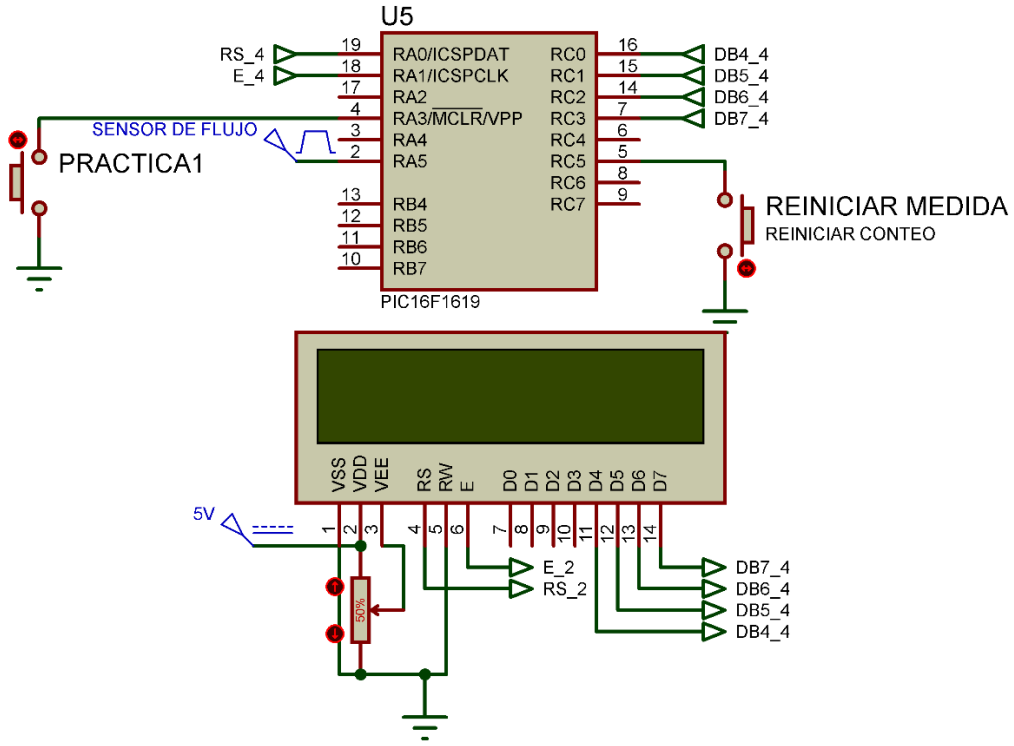
$$LITER = \frac{Pulsos*100}{K} = \frac{1500*100\,p}{150^{p*100}/_{mL}} = 1000\,mL = 1L \quad [6]$$

$$FLU = \frac{(Pulsos - Pulsos_{ant})*100}{K} = \frac{(1500-1450)^{p}/_{s}*100}{150^{p*100}/_{mL}} \quad [7]$$

$$FLU = \frac{50^{p}/_{s}*100}{150^{p*100}/_{mL}} = \frac{5000^{p*100}/_{s}}{150^{p*100}/_{mL}} = 33\,^{mL}/_{s} \quad [7.1]$$

Al presionar el botón PB1 se reinicia el timer 1, eliminando el líquido suministrado.

PRACTICA 4



Algorimto practica 4

```
PRACTICE_4:
   GOSUB SEQUENCE;
   lcdout $fe,$80,"PRACTICE"
   lcdout $fe,$c0,"-- 4 ---"
   PAUSE 1000
   lcdout $fe,$80,"LIQUID  "
   lcdout $fe,$c0,"CONTROL "
   PAUSE 1000

   TMR1L = 0;
   TMR1H = 0;
   HERE4:
      IF PB1 = 0 THEN
         flu = 0;
         liter = 0
         TMR1L = 0
         TMR1H = 0;
         ANT=0
         DIF = 0
      ENDIF
```

```
      T1CON.0 = 1
      TMR.BYTE0 = TMR1L; LOW REGISTER
      TMR.BYTE1 = TMR1H; HIGH REGISTER
      ANT= TMR;
      PAUSE 1000
      TMR.BYTE0 = TMR1L; LOW REGISTER
      TMR.BYTE1 = TMR1H; HIGH REGISTER
      DIF= TMR-ANT;          OBTAIN THE DIFFERENCE OF PULSES
      REMAINDER= TMR*100;
      LITER = DIV32 K;       DIVIDE BY THE CONSTANT
      REMAINDER = DIF*100
      FLU = DIV32 K;         DIVIDE BY THE CONSTANT

      FOR X = 0 TO 4
         FD[X] = FLU DIG X
         LD[X] = LITER DIG X
      NEXT X


      LCDOUT $FE,$80,DEC LD[4],DEC LD[3],".",DEC LD[2],DEC LD[1],DEC
LD[0],"L "
      LCDOUT $FE,$C0,DEC FD[2],DEC FD[1],DEC FD[0],"mL/S   "
      IF PB1 = 0 THEN
         flu = 0;
         liter = 0
         TMR1L = 0
         TMR1H = 0;
         ANT=0
         DIF = 0
      ENDIF
IF C <> 4 THEN START; IF DIFFERENT RPACTICE

GOTO HERE4
```

PRÁCTICA 5 – TRANSMISOR DE VOLTAJE Y TEMPERATURA SERIAL

La practica 5 consiste en tomar lectura del potenciómetro y de un sensor de temperatura LM34.

Para esto se utiliza el módulo EUSART (Enhanced Universal Synchronous Asynchronous Receiver Transmitter) que nos ayuda a transmitir información por un solo cable. Este módulo se configuro para trabajar a una velocidad de 2400 bps (Bits Por Segundo) utilizando los registros SP1BRGH:SP1BRGL utilizando las ecuaciones de la **Tabla 3**, donde el divisor puede ser 4, 16 o 64, dependiendo de los bits seleccionados en los registros TX1STA y BAUD1CON.

n = SP1BRGH:SP1BRGL.

Bits de configuración			Modo BTG/EUSART	Ecuación Baud Rate
SYNC	BRG16	BRGH		
0	0	0	8 bits/asíncrono	$\frac{FOSC}{64(n+1)}$
0	0	1	8 bits/asíncrono	$\frac{FOSC}{16(n+1)}$
0	1	0	16-bits/asíncrono	
0	1	1	16-bits/asíncrono	$\frac{FOSC}{4(n+1)}$
1	0	No importa	8-bits/síncrono	
1	1	No importa	16 bits/asíncrono	

Tabla 3. Ecuaciones para obtener el Baud Rate

Donde es importante conoces el valor de n para establecer la velocidad de transmisión, se puede utilizar la **Ec.8** para encontrar el valor de n.

$$n = \frac{{}^{FOSC}/_{Baud\ Rate\ deseado}}{Divisor} - 1 \qquad [8]$$

En esta práctica trabajamos en modo asíncrono (SYNC = 0) a baja velocidad (BRGH = 0) y de 8 bits (BRG16 = 0), por lo que podemos determinar que utilizamos un divisor de 64, el FOSC que utilizamos es de 4Mhz, y queremos una velocidad de transmisión de 2400bps, reemplazando el valor en la **Ec.8** y obtenemos los resultados de n:

$$n = \frac{{}^{4Mhz}/_{2400}}{64} - 1 = 25$$

Después de configurar la velocidad de transmisión se utiliza la instrucción **HSEROUT**, que nos permite utilizar el módulo EUSART.

Al presionar el botón PB1, la variable A cambia a 0, y manda con la instrucción HSEROUT la señal a el circuito receptor que se a iniciado la comunicación. Y

después de mandar la señal de inicialización al receptor, el transmisor mide los canales AN8 (Temperatura) y AN9 (Voltaje), respectivamente.

Antes de mandar cadenas de datos es necesario se les coloca un código para que el receptor entienda que dato es, para el caso del voltaje se coloca “VA” y en la temperatura “TB” y para finalizar se envía una tercera cadena que indica si se continua con la transmisión o no, que inicia con los caracteres “ZZ”. El ejemplo del algoritmo de envió de datos se ve a continuación:

```
HSEROUT ["VA", VD[3], VD[2], VD[1],VD[0],10]; SEND DATA BY WAY OF SERIAL
HSEROUT ["TB", T[3],T[2],T[1],T[0],10]
HSEROUT ["ZZ",DEC A,10]
```

Es importante recalcar que el orden de transmisión es de izquierda a derecha, siendo primero enviado “V” y seguido “A” por dar un ejemplo.

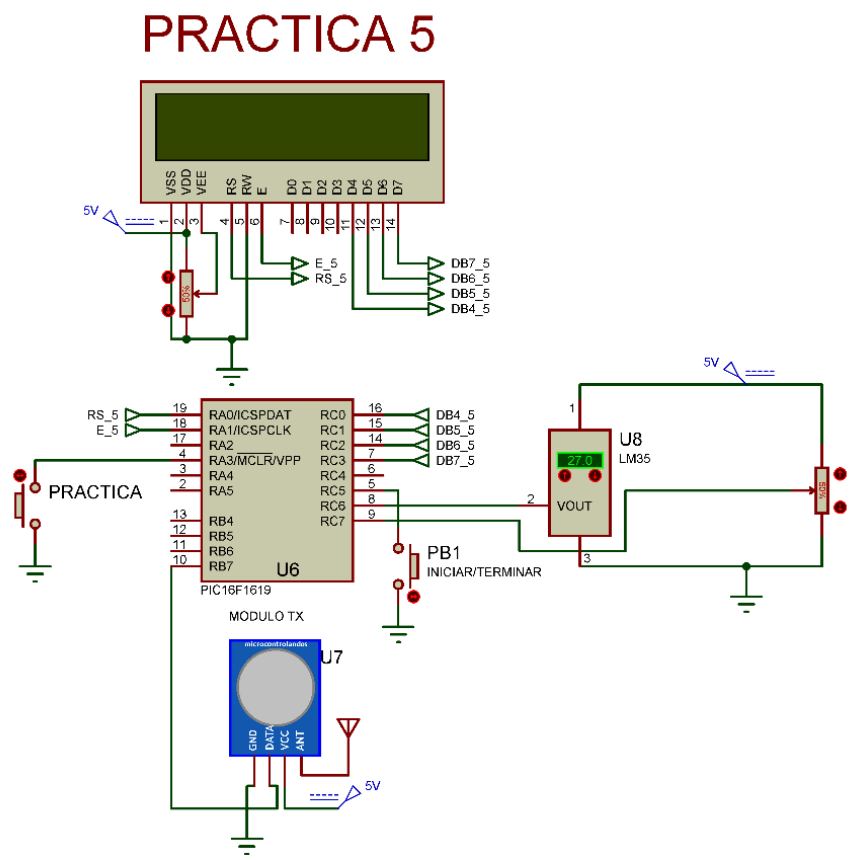
Antes de enviar los datos es importante tener en cuenta en que formato se están enviando los dígitos, en este caso se utilizó la función **LOOKUP** con los dígitos del 0 al 9 en código ASCII antes de hacer la transmisión de los datos; Este paso pudo ser suprimido y solo utilizar la función **DEC** en el dato a querer enviar, por ejemplo:

```
lookup V[0],["0123456789"],VD[0]
```

es igual a

```
VD[0] = DEC V[0]
```

Al presionar el botón PB1, la variable de control de transmisión A, cambia a 1 dando señal al receptor que se a terminado la transmisión de datos.



Algorimto practica 5

```
;CONFIGURATION OF THE SERIAL PORT
TX1STA = %00100000;$20, ENABLE TRANSMISSION, 8 BITS, LOW VELOCITY
BAUD1CON = %00000000; NO INVERTER, DETECTION OF THE FALLING EDGE
SPBRGL = 25; CALCULATED OF 2400 BPS
RB7PPS = %10010; ACTIVATING PPS MODULE
RC1STA = %10010000; ENABLED SERIAL PORT
RXPPS = %01101
DEFINE HSER_CLROERR 1; CLEAR OVERRUN ERROR
PRACTICE_5:
   GOSUB SEQUENCE;
   lcdout $fe,$80,"PRACTICE"
   lcdout $fe,$c0,"-- 5 ---"
   PAUSE 1000
HERE5:
   lcdout $fe,$80,"TX VOLT "
   lcdout $fe,$c0,"& TEMP  "
   PAUSE 1000

MAIN5:
```

```
IF PB1 = 0 THEN;START THE QUESTION FOR THE BUTTON
   PAUSE 150
   A = 0;CONTROL VARIABLE IS 0
   GOSUB SEND
ENDIF; END OF THE QUESTION
IF C <> 5 THEN START; IF DIFFERENT RPACTICE
IF A = 0 THEN; START THE QUESTION FOR THE VARIABLE OF CONTROL
  LCDOUT $FE,$80,"START---";
  LCDOUT $FE,$C0,"TX------";
  PAUSE 1000;

  STAY5:; LABEL STAY

      CH = 8;CHANNEL 8
      GOSUB ADC_READ;GO TO SEQUENCE
      FOR X = 0 TO 3
           T[X]=VD[X]
      NEXT X

      CH = 9
      GOSUB ADC_READ; GO TO THE LABEL ADC AND RETURN;
      lcdout $FE,$80,"V=",DEC VD[3],".",DEC VD[2],DEC VD[1],DEC VD[0],"     "

      lcdout $FE,$C0,"F=",dec T[3],dec T[2],dec T[1],".",DEC T[0],"     "


      GOSUB SEND;GO TO THE LABEL SEND AND RETURN

      if pb1 = 0 then;START THE QUESTION FOR THE BUTTON
            A = 1; VARIABLE OF CONTROL IS 1, END THE TRANSMISSION
            PAUSE 200;
      ENDIF; END OF THE QUESTION
      IF C <> 5 THEN START; IF DIFFERENT RPACTICE
   IF A = 0 THEN STAY5; IS CONTROL IS "0" GO TO THE LABEL STAY

   A = 1;; VARIABLE OF CONTROL IS "1"
   IF C <> 5 THEN START; IF DIFFERENT RPACTICE
   GOSUB SEND
   ;NEXT X
   LCDOUT $FE,$80,"END-----";
   LCDOUT $FE,$C0,"TX------";
   PAUSE 1000
   GOTO HERE5

ENDIF

GOTO MAIN5

SEND:; LABEL SEND;
   for x = 0 to 3
        in = vd[x]
```

```
        IN2 =T[X]
        lookup IN,["0123456789"],out
        LOOKUP IN2,["0123456789"],OUT2
        VD[X]=OUT
        T[X] = OUT2
    next x

    FOR X = 0 TO 1
       HSEROUT ["VA", VD[3], VD[2], VD[1],VD[0],10]; SEND DATA BY WAY OF
SERIAL
       HSEROUT ["TB", T[3],T[2],T[1],T[0],10]
       HSEROUT ["ZZ",DEC A,10]
    NEXT X

RETURN; GO TO THE IS CALLING
```

PRACTICA 6 – RECEPTOR DE VOLTAJE Y TEMPERATURA SERIAL

La practica 6 consiste en la recepción de información con el puerto serie y mostrando los datos en la LCD utilizada.

Para esto se utiliza el módulo EUSART como receptor, se realiza la misma configuración de la **practica 5** de los registros, SP1BRGH:SP1BRGL con valor de 25, transmisión encendida y recepción encendida.

Para esta práctica es necesario utilizar la instrucción **HSERIN**, la cual utiliza el puerto serie del microcontrolador para leer datos en puerto serie.

El orden de la instrucción HSERIN es el siguiente:

HSERIN (Tiempo), (Etiqueta), [(Dato a recibir)]

En donde el tiempo se ingresa en mSeg, la etiqueta se refiere a donde ir al superar el tiempo especificado y en la sección de recibir datos, se puede utilizar instrucciones como **WAIT,** que espera a que ingrese un dato en específico y si este no ha llegado en el tiempo especificado, se va a la etiqueta marcada; Al igual se puede utilizar la instrucción **STR**, con esta instrucción podemos especificar cuantos datos se quieren recibir en la instrucción, y estos se almacenaran en una variable tipo vector con las cantidad de columnas especificadas en la instrucción, estas funciones se pueden apreciar en el siguiente código:

```
HERE_6A:
IF C <> 6 THEN START; IF DIFFERENT RPACTICE
HSERIN 100,HERE_6A,[WAIT("VA"),STR VD\4];WAIT THE INSTRUCTION "VA" AND SAVE THE DATA
```

En se asignó un tiempo de espera de 100 mSeg, ir a la etiqueta HERE_6A al superar el tiempo especificado, se espera a recibir los datos en ASCII “VA” y después se recibe una cadena de 4 dato y es guardada en la variable tipo vector de 4 espacios (0 a 3). En el espacio 0 se guarda el primer dato recibido y en el espacio 3 se guarda el ultimo.

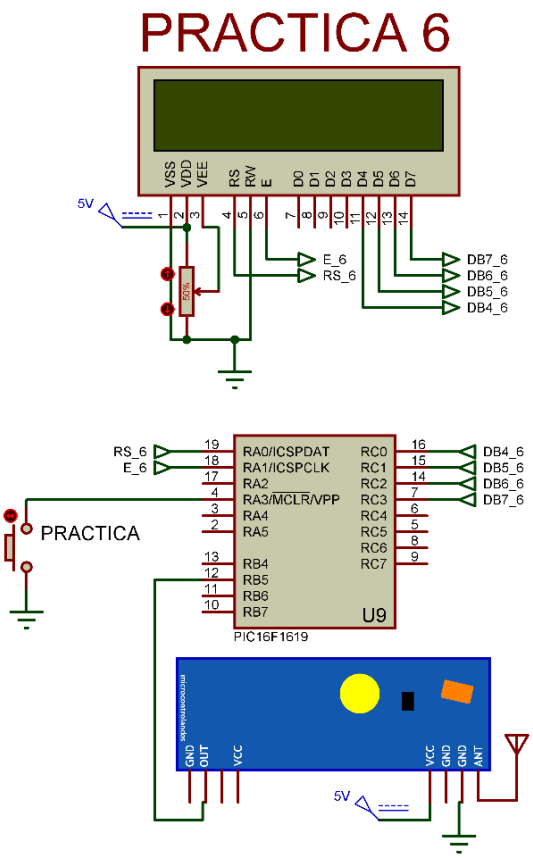
Por ejemplo, si el transmisor manda un mensaje como el siguiente, “VA1234”, las letras “VA” se discriminan y el carácter “1” se guarda en VD[0], “2” en VD[1], “3” en VD[2] y “4” en VD[3]

Al principio del algoritmo se pregunta ingresado el valor de inicialización “ZZ” y si el dato es diferente o no ingresa ningún dato en el tiempo especificado se mantiene en un ciclo permanente, como se ve a continuación:

```
MAIN6:;LABEL MAIN1
IF C <> 6 THEN START; IF DIFFERENT RPACTICE
HSERIN 1000,MAIN6,[WAIT("ZZ"), STR A\1];WAIT 1 SECOND TO RECEIVE THE INSTRUCTION
```

Cuando se recibe en la variable de control A un “0” inicia la recepción de datos, lo que nos permite solicitar el voltaje y temperatura respectivamente, El voltaje se espera que llegue con los caracteres “VA”, la temperatura con los caracteres “TB” y la variable de control con “ZZ”.

Cuando la variable de control (A) es recibida con un valor de “1”, el algoritmo termina la recepción de datos, mandando un mensaje de final de transmisión, después se mantiene en un ciclo preguntado si se inicializa o no la recepción de datos.

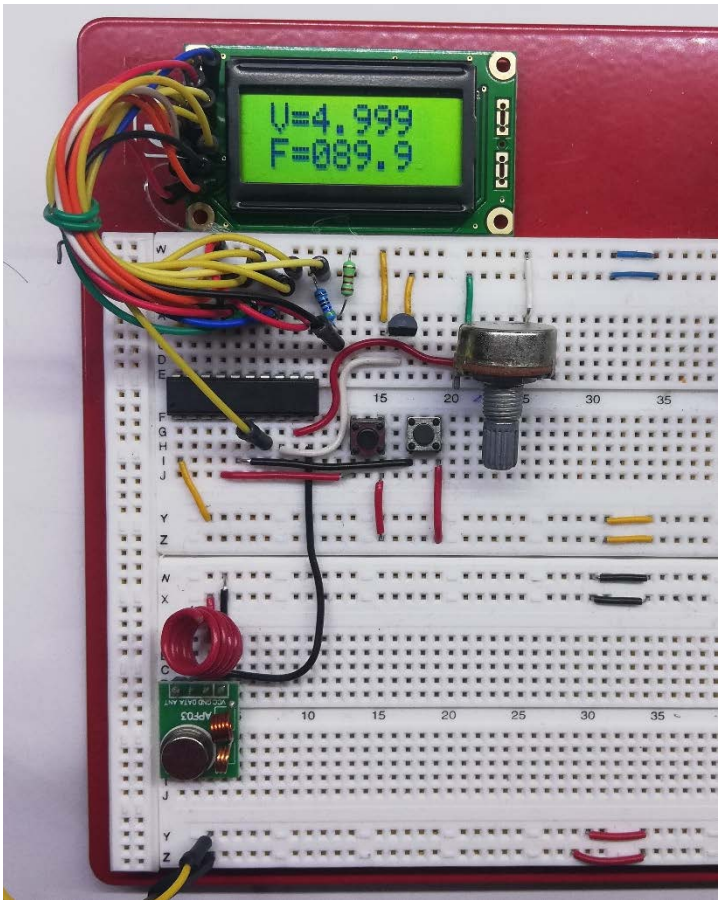



```
PRACTICE_6:
   GOSUB SEQUENCE;
   lcdout $fe,$80,"PRACTICE"
   lcdout $fe,$c0,"-- 6 ---"
   PAUSE 1000

   lcdout $fe,$80,"RX VOLT "
   lcdout $fe,$c0,"& TEMP  "
   PAUSE 1000
   ;CH = 8;
   A = 1;
```

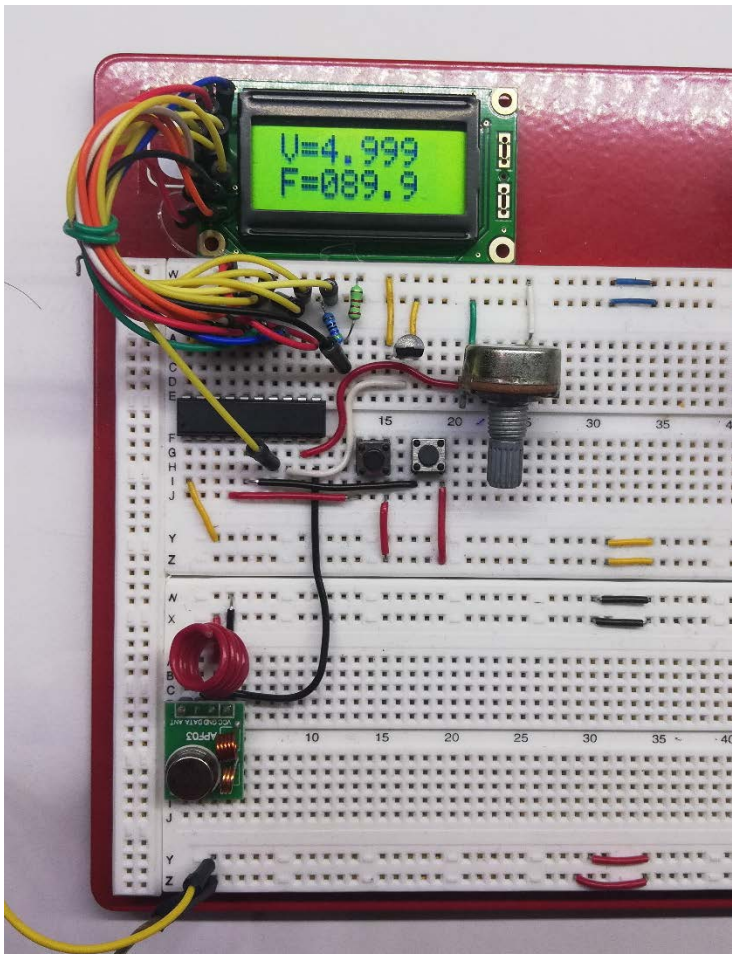
```
HERE6:
LCDOUT $FE,$80,"WAITING-";
LCDOUT $FE,$C0,"SIGNAL--";

MAIN6:;LABEL MAIN1
IF C <> 6 THEN START; IF DIFFERENT RPACTICE
HSERIN 1000,MAIN6,[WAIT("ZZ"), STR A\1];WAIT 1 SECOND TO RECEIVE THE INSTRUCTION

IF  A = "0" THEN;IF A IS EQUAL TO DIGIT "0" CONTINUE
   LCDOUT $FE,$80,"START---";
   LCDOUT $FE,$C0,"RX------";
   PAUSE 1000
   HERE_6:; LABEL HERE
   HERE_6A:
   IF C <> 6 THEN START; IF DIFFERENT RPACTICE
   HSERIN 100,HERE_6A,[WAIT("VA"),STR VD\4];WAIT THE INSTRUCTION "VA" AND SAVE THE DATA
   HERE_6B:
   IF C <> 6 THEN START; IF DIFFERENT RPACTICE
   HSERIN 10,HERE_6B,[WAIT("TB"),STR T\4];WAIT THE INSTRUCTION "TB" AND SAVE THE DATA

   LCDOUT $FE,$80,"V=",VD[0],".",VD[1],VD[2],VD[3],"          ";
   LCDOUT $FE,$C0,"F=",T[0],T[1],T[2],".",T[3],"          ";
   HSERIN 10,HERE_6, [WAIT("ZZ"),STR A\1];WAIT THE INSTRUCTION "ZZ" AND SAVE THE DATA
   IF C <> 6 THEN START; IF DIFFERENT RPACTICE
   if A <> "1" then HERE_6; IF A IS DIFERENT TO "1" THEN HERE
ENDIF
   LCDOUT $FE,$80,"END-----";
   LCDOUT $FE,$C0,"RX------";
   PAUSE 1000
GOTO HERE6

ADC_READ:
   CH = CH<<2;
   ADCON0 = CH^%10000001; BITWISE XNOR
   PAUSE 1
   ADCON0.1 = 1;
   HERE_ADC: IF ADCON0.1 = 1 THEN HERE_ADC; STAY HERE WHILE CONVERSION IS ready
   ADCON0.0 = 0;
   ADC.BYTE0 = ADRESL;       SAVE LOWER REGISTER OF THE ADC IN VARIABLE VIN
   ADC.BYTE1 = ADRESH;       SAVE HIGHER REGISTER OF THE ADC IN VARIABLE VIN
```

```
   DISABLE
   REMAINDER = ADC*4887;          MULTIPLYING BY RESLSB = 4.8887
   VIN = div32 1000;          PERFORM 16-BITS DIVISION
   ENABLE
   FOR X = 0 TO 3
      VD[X] = VIN DIG X
   NEXT X
return
```

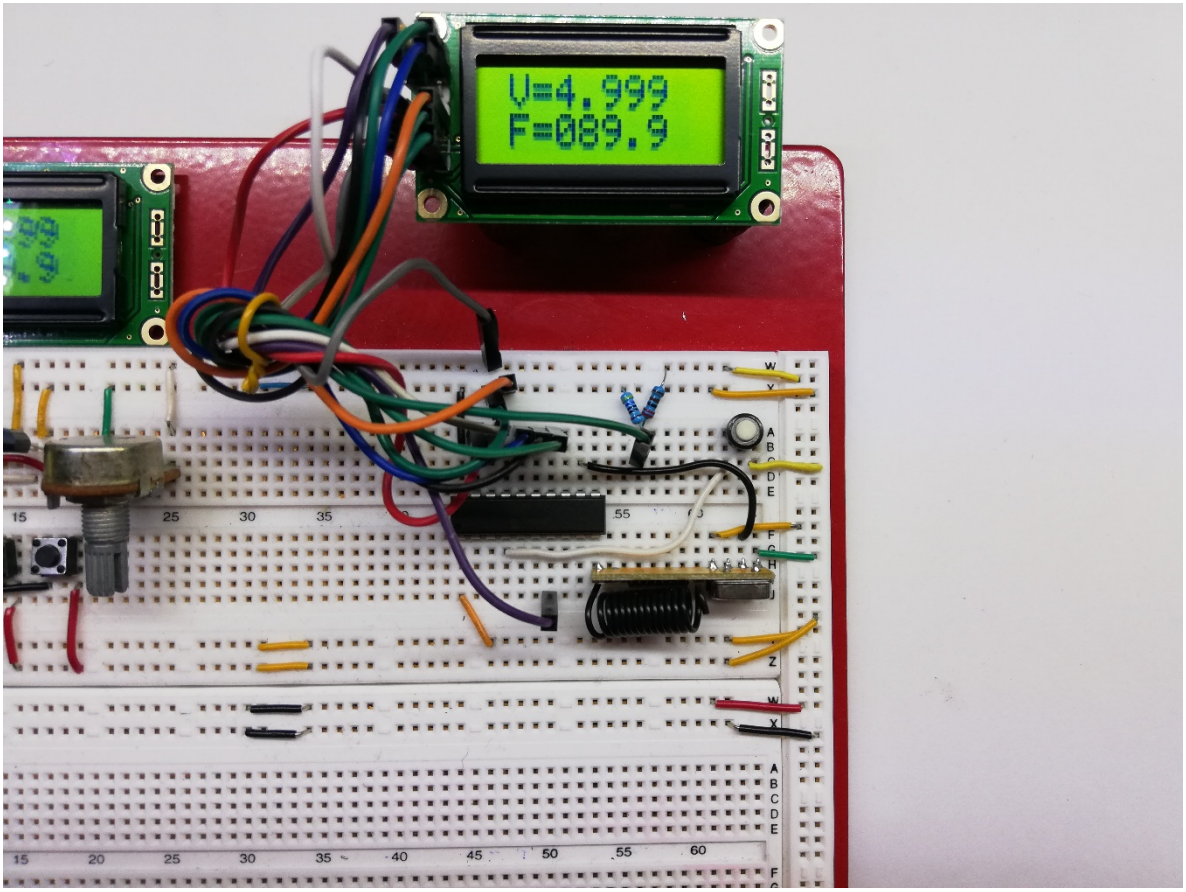
Fotos



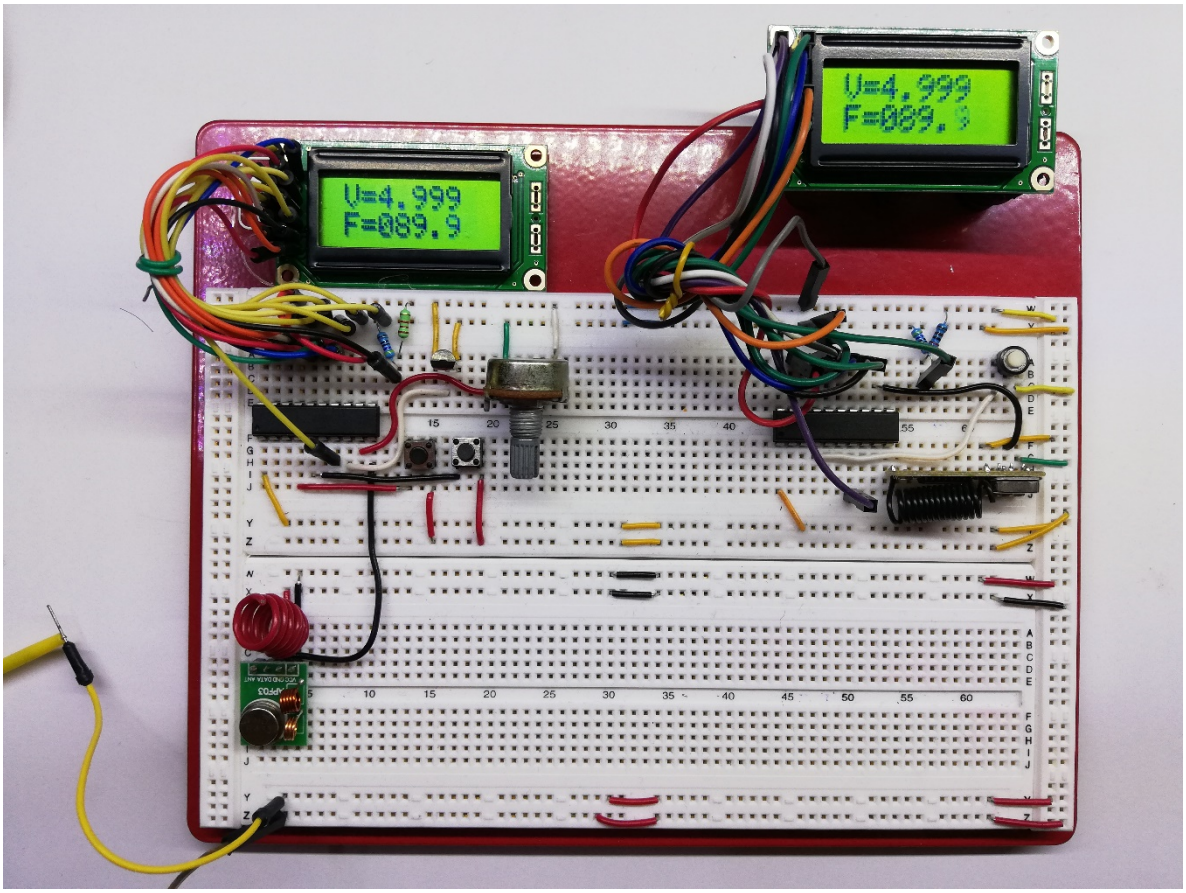
Practica 1 a 5



Practica 1 a 5



Practica 6



Practica 5 y 6, transmisor y receptor