Proyecto Final

Equipo 5

Jean Carlo Álvarez Guerra A01635182
Rodrigo Serrano García A01630978
Andrés Antonio Bravo Orozco A01630783

Laboratorio de Microcontroladores

Grupo 500

Profesor Jesús Esteban Cienfuegos Zurita

Introducción

Maquina de remado:

Las máquinas de práctica de remo funcionan a base de un asiento que se mueve en un riel. La máquina genera el esfuerzo usando el mismo peso del usuario o usando un motor que genera la resistencia deseada. El usuario idealmente solo se tiene que impulsar hacia atrás para comenzar con el ejercicio.

Timer:

Un timer en un microcontrolador funciona a base de contar cuántos ciclos ha dado su fuente de tiempo (esta puede ser interna o externa). En sí, un timer es un registro en el cual se divide entre instrucciones y el conteo. Los timers pueden ser controlados para parar y comenzar y también su escala de conteo. La escala de conteo convierte un ciclo en una cantidad de valor que se agrega al registro de conteo, lo cual cambia el tiempo que se tarda en llenar el registro. El timer también se puede manejar con interrupciones. Las interrupciones nos permiten hacer alguna instrucción en cualquier momento sin tener la necesidad de esperar que un ciclo del timer acabe. Esto nos da la versatilidad de manejar instrucciones de manera instantánea y sin necesidad de esperar.

Pantalla LCD:

Una pantalla LCD 16X2 es un periférico que puede recibir texto e instrucciones para mostrar texto. Este tipo de pantalla tiene como entrada 8 pines de datos y 3 de control. 2 de los 3 pines de control nos dejan decidir si queremos ingresar una instrucción o texto para mostrar en la pantalla, el otro pin es solo para habilitar la pantalla. los 8 pines de datos reciben un valor binario que equivale a un símbolo ASCII que se mostrará en pantalla. En caso de recibir instrucciones, los 8 pines reciben un valor binario que equivale a una instrucción exacta.

PWM:

El modo de PWM utiliza el Timer2 para modular el ancho de pulso de una señal a generar. Se le llama duty cycle al porcentaje del periodo de la señal en que esta se mantiene en “alto” o 1 lógico. Por ejemplo, el duty cycle de una señal de reloj cuadrada es de 50% ya que permanece en alto y bajo la misma cantidad de tiempo.

El módulo ADC:

El módulo ADC en un microcontrolador nos funciona para convertir datos análogos a digitales. Un ejemplo de esto es si recibieron un dato de algún sensor de temperatura. Este tipo de sensores regresan un voltaje en vez de algún dato digital. Lo que hace el ADC es recibir este cambio de voltaje y guarda una muestra. Después cuantifica esta muestra para determinar la resolución. Al final cambia esto a un número binario el cual es enviado al sistema.

Desarrollo

LCD:

Descripción: Para desplegar los datos internos, se utiliza un LCD. Este cuenta con dos vistas distintas: La primera donde se despliega la cuenta de calorías al igual que el tiempo transcurrido y la segunda donde se muestra el conteo de remadas, la meta y el promedio de remadas por minuto. Cada pantalla permanece en el LCD por 5 segundos. La implementación de esta lógica se puede observar en la Figura 1, donde la rutina se encuentra dentro del modulo del timer que se abordará más adelante.

Pines/Recursos: RA1, RA2, RA3 RB0-7.

```
if(on_flag){          //rutina para desplegar la información necesaria en el LCD
    if (reloj == 0){ //si el reloj está en 0, desplegar calorias y tiempo
      page =0;
      print();
      page =1;
      print();
    }else if (reloj > 0 && reloj < 5){ //si el reloj está entre 0 y 5, desplegar unicamente si hay cambios
        if(calorias != prev_calorias){
            prev_calorias = calorias;
            page =0;
            print();
        }
      page =1;
      print_time();
    }else if(reloj == 5){//si el reloj está en 5, desplegar remadas, meta y promdio
        page =2;
        print();
        page =3;
        print();
    }else if(reloj > 5){//si el reloj es mayor a 5 desplegar unicamente si hay cambios
       if(remadas != prev_remadas){
            prev_remadas = remadas;
            page =2;
            print_remadas();
        }
       if(goal != prev_goal){
            prev_goal = goal;
            page =2;
            print_metas();
        }
    }
}
```

Figura 1: Rutina del LCD para desplegar los datos

Registro del tiempo transcurrido:

Descripción: Se tiene un pin designado al prendido del sistema y a poner el sistema en pausa, al igual que otro pin para apagar el sistema. Cuando el sistema se prende, comienza el timer0 que cuenta hasta llegar a 1 segundo, al igual que se prende el led de prendido. La interrupción del overflow se utiliza para llevar la cuenta de los segundos transcurridos y así poder marcar el tiempo total. Cuando se pone en pausa el sistema se apaga el timer0 y el led, para reanudarlo simplemente se prende el timer0 y el led. Para apagar el programa se limpia la pantalla LCD al igual que se reinician las variables y se apaga el timer0 y el led de

encendido. La manera en la que se obtiene el tiempo transcurrido se puede observar en la Figura 2

Pines/Recursos: RC0 (input de prendido/apagado/pausa), RC1 (input de apagado) Timer0 (interrupt enabled), contador de overflows TMR0

```
if(reloj == 9){              //rutina para desplegar obtener el tiempo actual cada segundo
    if(reloj2==5){
        if(reloj3==9){
            if(reloj4==5){
                reloj4=0;
            }else{
                reloj4= reloj4+1;
            }
            reloj3=0;
            minute_count++;//si los minutos cambian actualizar el promedio
            promedio = remadas/minute_count; //formula para obtener el promedio
            print_promedio();

        }else{
            reloj3 = reloj3+1;
            minute_count++;//si los minutos cambian actualizar el promedio
            promedio = remadas/minute_count; //formula para obtener el promedio
            print_promedio();
        }
        reloj2 = 0;
    }else{
       reloj2 = reloj2+1;
    }
    reloj = 0;
}else{
    reloj = reloj+1;
}
reloj_total = 600*reloj4+60*reloj3+10*reloj2+reloj;//obtener cantidad total de tiempo


//show_currentTime();

T0CONbits.TMR0ON = 0;
TMR0H = TimerCount_high;
TMR0L = TimerCount_low;
INTCONbits.TMR0IF =    0;
T0CONbits.TMR0ON = 1;
```

Figura 2: Rutina para obtener el tiempo total

Conteo de remadas:

Descripción: Se tiene un pin designado a obtener la entrada de las remadas a través de un switch. Cada que el pin se encuentra en 1 lógico, una remada se agrega al contador para indicar las remadas totales que lleva el sistema. Para obtener la entrada del usuario se utilizó la rutina descrita en la figura 3

Pines/Recursos: RD6 (input), contador de Remadas

```
if (PORTDbits.RD6){//Para remadas
    if(!done_remada){//si no se ha realizado la remada y se detecta un 1 en la entrada, aumentar remadas y marcar com hecho
        remadas ++;
        done_remada = 1;
        if(remadas == goal_count){//Usar buzzer, se compara con una suma de metas en vez de la meta per se, ya que si se supera la meta debe seguir indicando la siguiente
            goal_count += goal;
            PORTCbits.RC4 = 1;
            longDelay();
            PORTCbits.RC4 = 0;
        }
    }
}else if(!PORTDbits.RD6){//si se detecta un 0 en la entrada y se realizó una remada, habilitar que se pueda marcar una nueva
    if(done_remada){
        done_remada = 0;
    }
}
```

Figura 3: Obtención de remadas

Promedio de remadas por minuto:

Descripción: Se utilizó el timer0 que ya cuenta el tiempo para obtener una variable de los minutos transcurridos, por lo tanto se utilizó un contador de minutos transcurridos que actualiza la variable cada minuto utilizando la cantidad de remadas que lleva el usuario. Al cambiar de minuto se debe hacer el promedio. Para realizar el promedio se agregó unas cuantas lineas de código que se pueden observar en la figura 2 donde cada vez que la variable reloj3 cambia, se actualiza el promedio

Pines/Recursos: Timer0 , contador de minutos, contador de remadas que se usó anteriormente

Alarma cuando se haya alcanzado una cantidad de remadas

Descripción: Se habilitó un buzzer que se prende cuando las remadas totales alcanzan una meta especificada por el usuario. La meta se puede cambiar utilizando un switch de incremento y otro de decremento, que añaden o disminuyen en intervalos de 10 remadas la meta total. Una vez llegada a la meta se puede volver a prender el buzzer si se hacen de nuevo la cantidad de remadas especificadas. Para implementar la alarma se agregó una pequeña rutina al código que obtiene las remadas como se mostró en la figura 3. con el cual se compara si las remadas llegaron a la meta.

Pines/Recursos: RC4 (output buzzer), delay simple sin timer, RD0 y RD1 (input para variar la meta)

Indicar quema de calorías:

Descripción: Se generó un ritmo cardíaco con un pwm para poder representar distintos cálculos de quema de calorías. Se cuenta con 4 ritmos cardíacos distintos, compuestos del pwm con frecuencias de 1kHz, 500 Hz, 333Hz y 250Hz. Para escoger la frecuencia del PWM se utiliza el módulo ADC, donde cada 25% de la resistencia de entrada representa un modo distinto. El módulo ECCP1 se utilizó para poder medir la frecuencia que se genera a partir del PWM. Estas dos rutinas se muestran en la figura 4 y figura 5

```
void pwm_interrupt(void){
   T2CONbits.TMR2ON = 0;
   //PR2 = valor_PR2_4;
   //CCPR1L = valor_CCPRIL_4;

   if (heart_rate_mode == 4){ //asignar la frecuencia deseada al pwm dependiendo del modo
       PR2 = valor_PR2_4;
       CCPR1L = valor_CCPRIL_4;
   }else if (heart_rate_mode == 3){
       PR2 = valor_PR2_3;
       CCPR1L = valor_CCPRIL_3;
   }else if (heart_rate_mode == 2){
       PR2 = valor_PR2_2;
       CCPR1L = valor_CCPRIL_2;
   }else {
      PR2 = valor_PR2_1;
      CCPR1L = valor_CCPRIL_1;
   }

   PIR1bits.TMR2IF=0;
   T2CONbits.TMR2ON = 1;
}
```

Figura 4: Interrupción del módulo PWM

```
void eccp1_interrupt(void){

   if(compared_flag == 0){ //se realiza una comparación para determinar la duración de la señal pwm
       data1 = ECCPR1;
       compared_flag = 1;
   }else {
       data2 = ECCPR1;
       total = data2-data1;
       compared_flag = 0;
   }
   if(total != prev_total){
       prev_total = total;
       if(total == count_1){
           calorias = -55.0969 + 0.6309*180 + 15.904 + 5.0425/4.184*reloj_total/60;   //se calcula la quema de calorias
       }else if(total == count_2){
           calorias = -55.0969 + 0.6309*160 + 15.904 + 5.0425/4.184*reloj_total/60;
       }else if(total == count_3){
           calorias = -55.0969 + 0.6309*140 + 15.904 + 5.0425/4.184*reloj_total/60;
       }else if(total == count_4){
           calorias = -55.0969 + 0.6309*100 + 15.904 + 5.0425/4.184*reloj_total/60;
       }
   }

   PIR2bits.ECCP1IF = 0;
```

Figura 5: interrupción del módulo eccp1

Siendo la frecuencia máxima en el circuito de 1000 Hz que es igual a 60,000 BPM, se hizo una relación donde 1000 Hz equivale a la frecuencia máxima normal que una persona tendría al hacer ejercicio siendo esta 180 BPM. La frecuencia de 500 Hz se usó como una frecuencia cardiaca alta de 160 BPM, la frecuencia de 333 Hz como ejercicio moderado de 140 BPM y la frecuencia de 250 Hz se usó como la referencia más baja para una persona que hace ejercicio que es de alrededor 100 BPM.

Para calcular la quema de calorías se utilizó la siguiente fórmula:
Calorías = ((-55.0969 + (0.6309 x HR) + (0.1988 x W) + (0.2017 x A))/4.184) x 60 x T

HR = Heart rate (en beats/minute)
W = Weight (en kilogramos)
A = Age (en años)
T = Exercise duration time (en horas)

Utilizando la fórmula únicamente con valores predeterminados de hombre con W = 80kg, A = 25 y el tiempo en segundos, la fórmula resultante es:

Calorías: ((-55.0969 + (0.6309 x HR) + (15.904) + (5.0425))/4.184) x (T)/60

Pines/Recursos: RC2/CCP1 (output pwm), RD4/ECCP1 (input ritmo cardiaco), Timer2 (para pwm), RA0 (input adc), Timer1 (para ECCP1).

Utilización del proyecto

Para utilizar el proyecto existen varios inputs que el usuario puede usar para controlar el sistema. Empezando por el potenciómetro que se muestra en la figura 6 descrito como "Ritmo cardiaco". Este potenciómetro cambia la frecuencia cardiaca simulada por el pwm cada 25% de la resistencia. Para prender el sistema se hace click en el botón de "Prendido/Pausa" que como dice el nombre, puede poner en pausa el sistema si se vuelve a presionar. Si se presiona este botón cuando está en pausa el sistema se reanuda. Cuando el sistema está encendido el LED "D1" prende para indicar su correcto funcionamiento. Para apagar el sistema se utiliza el botón de "Apagado". Para incrementar la meta se utiliza el switch "Incrementar meta" que incrementa en intervalos de 10 la meta que se quiere llegar, lo mismo sucede pero de manera inversa con el switch "Decrementar meta". Cuando se llega a la meta el buzzer se enciende y emite una breve alarma. Para obtener la entrada de las remadas realizadas por el usuario se utiliza el switch "Remadas", donde cada vez que el switch está en un 1 lógico se cuenta como una remada por el usuario.
NOTA: el uso del PWM y el ECCP1 crean bastante trabajo para proteus y/o el procesamiento del micro, por lo que los inputs por parte de los botones y los switches pueden tardar en ser captados por el sistema. En la aplicación real del proyecto el usuario proporcionaría su ritmo cardíaco por lo que el sistema tendría un mejor desempeño.

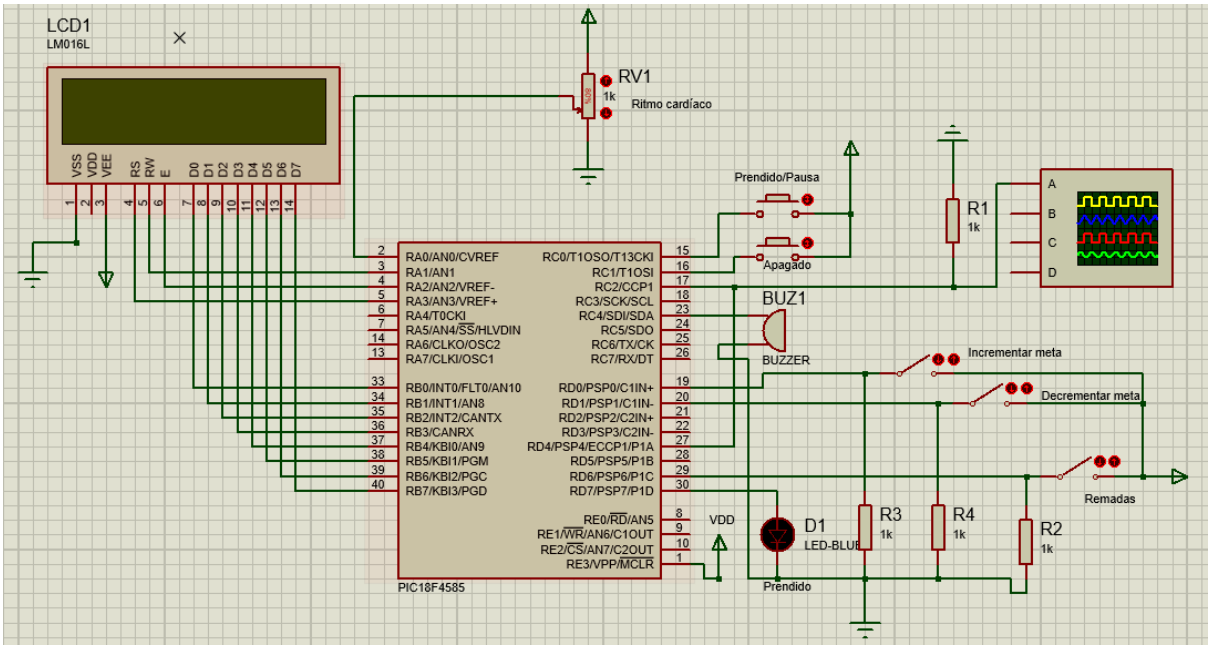

Figura 6: Esquemático del proyecto

Conclusión

Andrés

Este proyecto fue un gran reto de integración para aplicar todos los conocimientos adquiridos a lo largo del semestre. Lo más retador fue sin duda lograr que todas las partes funcionaran armónicamente para que se realicen todas las funcionalidades requeridas. Este proyecto me ayudó a poner en práctica todo lo visto en las clases para crear un programa más complejo de lo que habíamos realizado durante el semestre. Creo que este proyecto fue importante para conectar los conceptos de microcontroladores y así ver el posible resultado de integrar varias de estos juntos.

Rodrigo

El proyecto final fue una buena manera de implementar la mayoría de lo que aprendimos a través del semestre. Lo más complicado de este proyecto fue hacer funcionar todos los módulos al mismo tiempo. Esto es por el hecho que se tiene que tomar en consideración más variables y por el hecho de que todas fueron hechas por diferentes personas. Este proyecto también me ayudó a visualizar las múltiples aplicaciones que puede tener un microcontrolador.

Jean Carlo

Este proyecto integrador nos ayudó a poner a prueba no solo nuestros conocimientos en la materia sino también a coordinar como equipo cómo juntar nuestros respectivos fragmentos y que todo siguiera funcionando en conjunto. Fue un reto ya que trabajar de manera remota y asincrónica puede dar lugar a malentendidos. Por suerte, esto no ocurrió, creo que lo logramos manejar eficientemente, especialmente gracias a asignar a una persona como lider y supervisor.

Bibliografía

- http://www.shapesense.com/fitness-exercise/calculators/heart-rate-based-calorie-burn-calculator.shtml