Universidad Nacional Autónoma de México

Facultad de Ingeniería

Semestre 2023-2

Laboratorio de Microcomputadoras

Proyecto Final:

Simulación de automóvil inteligente.

Profesor:

M.I. Ruben Anaya García

Alumnos:	Número de cuenta:
Barreiro Valdez Alejandro	317520888
Gil Márquez Arath Emiliano	317083875
Herrera Carrillo Cristhian	317094662
Zepeda Baeza Jessica	317520747

Grupo Teoría: 1

Fecha de realización: 20 de junio de 2023

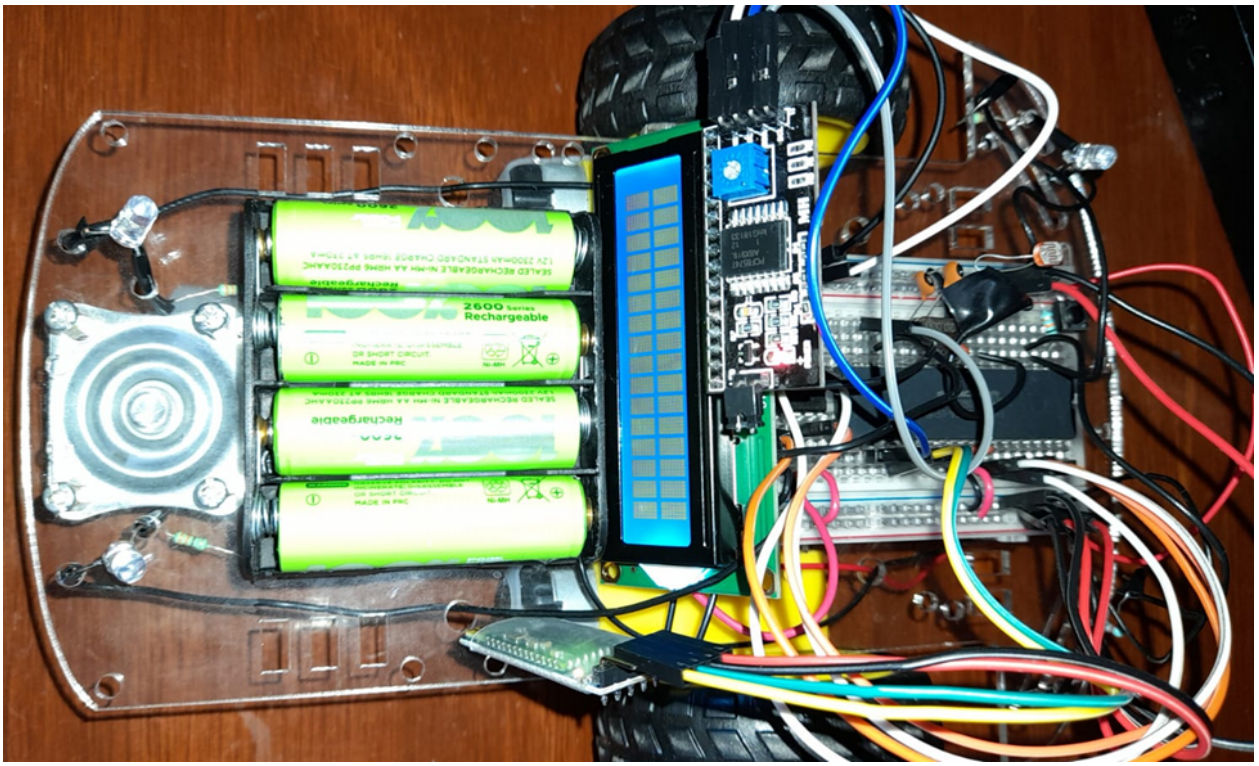
Fecha de entrega: 28 de junio de 2023

Introducción.

En la actualidad las microcomputadoras desempeñan un papel fundamental en diversos ámbitos de la vida cotidiana. Desde dispositivos móviles hasta sistemas de control industrial, estas pequeñas pero poderosas máquinas se han vuelto súper presentes en nuestra sociedad.

El presente proyecto tiene como objetivo diseñar y desarrollar una simulación de un automóvil inteligente, con base a un conjunto de conocimientos previamente adquiridos a lo largo del semestre, el prototipo será capaz de ir hacia enfrente, mostrando la distancia recorrida. podrá girar hacia los lados, mostrando el lado destino con direccionales delanteros y traseros. y podrá ir hacia atrás denotando con intermitentes.

Desarrollo.



El proyecto final utiliza el microcontrolador PIC16F877A como componente principal y combina varias funciones como la implementación de sus puertos, el control de un LCD a través de I2C, la generación de señales PWM para controlar LEDs, el desarrollo de interrupciones y la comunicación a través de Bluetooth para la realización de movimientos por medio de una aplicación.

El microcontrolador es el cerebro del proyecto que simula las características y el funcionamiento de un automóvil, se utiliza para controlar y coordinar todas las funciones codificadas.

El LCD se implementa para mostrar información al usuario de los movimientos que realiza el robot, así como la distancia que este recorre, pero en lugar de conectar el LCD directamente al microcontrolador se usa el protocolo de comunicación I2C, lo cual permite reducir la cantidad de pines requeridos para la conexión y simplificar el cableado. El I2C utiliza dos líneas de comunicación, SDA y SCL, para transmitir datos de forma bidireccional entre el microcontrolador y el LCD.

Mientras que para la funcionalidad PWM se optó por usar LEDs que se conectan a los pines del microcontrolador y que representan las intermitentes y direccionales del automóvil, para así controlar la intensidad del brillo y simular el comportamiento del robot ante los movimientos que realiza.
Para la comunicación inalámbrica, el circuito utiliza un módulo Bluetooth que se conecta al microcontrolador, esto permite establecer una conexión Bluetooth con dispositivos externos, en este caso un teléfono móvil para indicar comandos que representan los movimientos del automóvil: adelante, atrás, giro a la derecha, giro a la izquierda y parar.

Finalmente, todo este circuito es colocado en el robot que cuenta con motores los cuales son responsables de los movimientos con ayuda de unas ruedas. Es necesario utilizar baterías adecuadas para alimentar el sistema de manera eficiente (microcontrolador, motores y los demás componentes).

Algoritmos.

Para el programa de este proyecto se realizó la carta ASM mostrada en la siguiente imagen.

Configuraciones

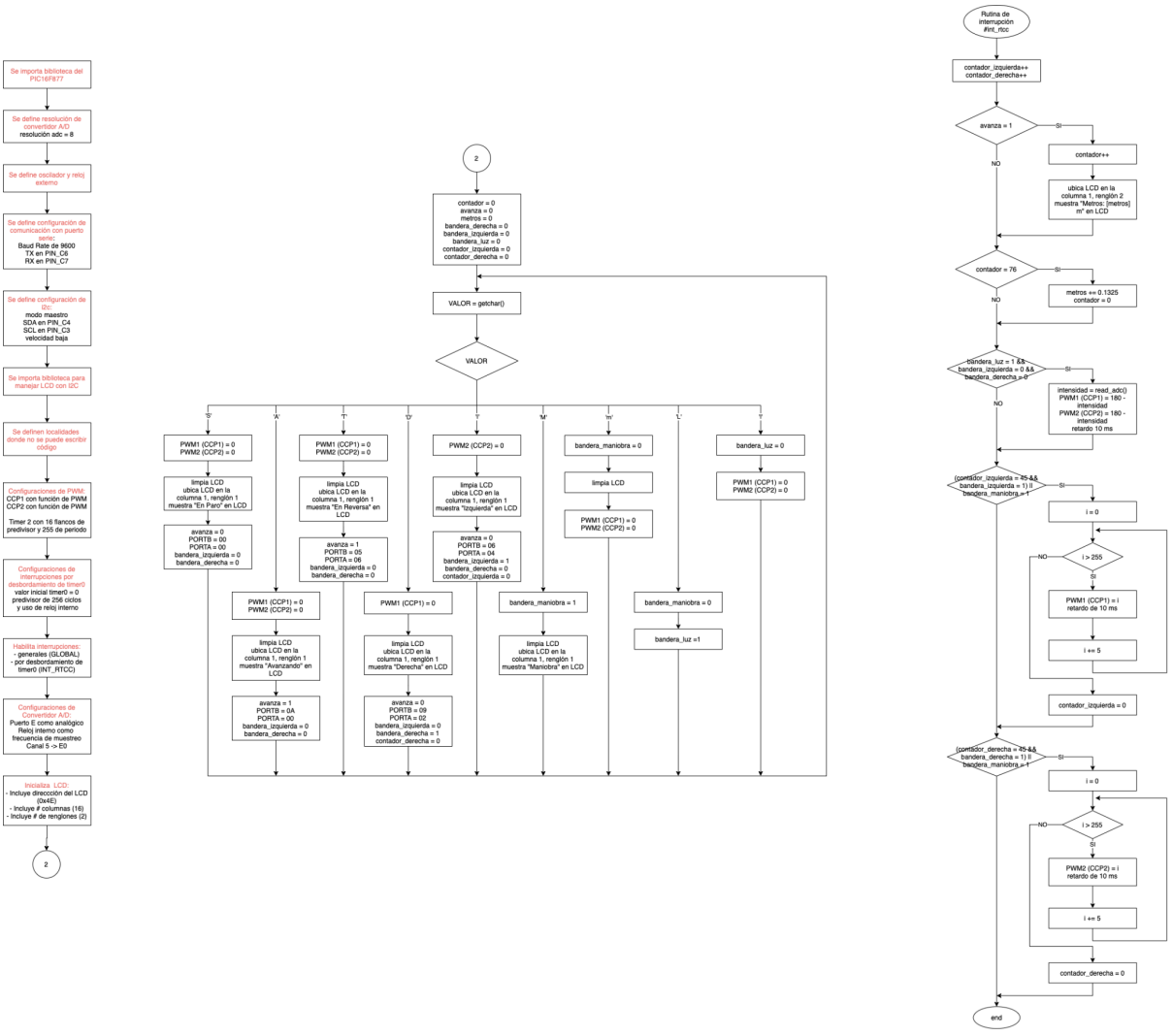
En la primera columna se incluyen las bibliotecas, directivas y configuraciones iniciales que utiliza el programa como: las bibliotecas del PIC16F877 y la del control del LCD por medio de I2C, también se define una resolución del convertidor A/D de 8 bits junto con el reloj externo y el oscilador del microcontrolador.
Se configura la comunicación asíncrona por el puerto serie usando un Baud Rate de 9600 con TX en el pin C6 y RX en C7. Las configuraciones del I2C incluyen la definición del

microcontrolador como maestro, SLC en el pin C3, SDA en el pin C4 y velocidad baja. Para el LCD se inicializa incluyendo su dirección, sus renglones y sus columnas.

La configuración del PWM involucró utilizar la función de PWM tanto del módulo CCP1 como del CCP2. Debido a que estos módulos utilizan el Timer 2 también se configura con un predivsor de 16 flancos y con 255 como valor del periodo. De manera que se definirá el tiempo de encendido con un valor de 1 a 255.

También se configuran interrupciones por desbordamiento de Timer 0 para provocar que cierto LED se prenda con PWM cada ciertos segundos. Por lo tanto se establece un valor inicial de 0 en Timer 0 y se configura con un predivisor de 256 ciclos utilizando el reloj interno (la interrupción sucederá cada 13.1 ms). Se habilitan las interrupciones globales y la particular de desbordamiento de Timer 0.

La última configuración es la del convertidor A/D donde se define al Puerto E0 como analógico, se utiliza el reloj interno como frecuencia de muestreo y se usa el canal 5 (E0).



Algoritmo

Como entrada se tiene una fotorresistencia, conectada al puerto E0. Como salidas se tienen 4 LEDS: dos faros (izquierda y derecha conectados a C1 y C2 debido a que utilizan los módulos CCP1 y CCP2) y dos luces de reversa conectados al puerto A1 y A2; dos motores con llantas conectados al puerto B y un LCD conectado mediante I2C.
Se definieron 5 banderas y 3 contadores:

- bandera_izquierda: indica cuándo se debe prender el faro izquierdo como direccional.
- bandera_derecha: indica cuándo se debe prender el faro derecho como direccional.
- bandera_luz: indica si se debe obtener un valor analógico de la fotorresistencia.
- bandera_maniobra: indica cuándo se deben prender las intermitentes.
- Avanza: bandera que indica cuándo el coche avanza.
- contador: lleva la cuenta de las interrupciones para llegar a 1 segundo.
- contador_izquierda: lleva la cuenta de las interrupciones para llegar a 0.6 segundos para el faro izquierdo.
- contador_derecha: lleva la cuenta de las interrupciones para llegar a 0.6 segundos para el faro derecho.

El programa inicializa todas las banderas y contadores es cero. Después tiene un loop infinito donde se recibe un caracter desde terminal, se guarda en una variable y se analiza en un switch para realizar una cierta acción:

- ‘S’ (stop): se establece un ciclo de trabajo de 0 en el PWM de ambos faros. Se muestra en el LCD: “En Paro”. La bandera Avanza, los motores, las luces de reversa y las banderas de cada faro se apagan.
- ‘A’ (adelante): se establece un ciclo de trabajo de 0 en el PWM de ambos faros. Se muestra en el LCD: “Avanzando”. La bandera Avanza se prende, al Puerto B (motores) se le pasa el valor de 0A. Las luces de reversa y las banderas de cada faro se apagan.
- ‘T’ (atrás): se establece un ciclo de trabajo de 0 en el PWM de ambos faros. Se muestra en el LCD: “En Reversa”. La bandera Avanza se prende, al Puerto B (motores) se le pasa el valor de 05. Se prenden ambas luces de reversa (pasando el valor 06 al Puerto A) y las banderas de cada faro se apagan.
- ‘D’ (derecha): se establece un ciclo de trabajo de 0 en el PWM para el faro izquierdo. Se muestra en el LCD: “Derecha”. La bandera Avanza se apaga. Al Puerto B (motores) se le pasa el valor de 09. Se prende solamente la luz de reversa derecha (02 en Puerto A). Se prende la bandera_derecha y se apaga la bandera_izquierda. Y se inicializa el contador_derecha con 0 para empezar a contar los segundos.

- ‘I’ (izquierda): se establece un ciclo de trabajo de 0 en el PWM para el faro derecho. Se muestra en el LCD: “Izquierda”. La bandera Avanza se apaga. Al Puerto B (motores) se le pasa el valor de 06. Se prende solamente la luz de reversa derecha (04 en Puerto A). Se prende la bandera_izquierda y se apaga la bandera_derecha. Y se inicializa el contador_izquierda con 0 para empezar a contar los segundos.
- ‘M’ (maniobra): se prende la bandera_maniobra y en el LCD se muestra “Maniobra”.
- ‘m’ (no maniobra): se apaga la bandera_maniobra, se limpia el LCD y se define el ciclo de trabajo para el PWM de ambos faros como 0.
- ‘L’ (luz): se apaga la bandera_maniobra y se prende la bandera_luz.
- ‘l’ (no luz): se apaga la bandera luz y se establece cero como ciclo de trabajo para el PWM de ambos faros.

Después de realizar las asignaciones, se vuelve a recibir un caracter de terminal.

Por último, se tiene la rutina de interrupción donde se hacen uso de todas las banderas antes mencionadas.

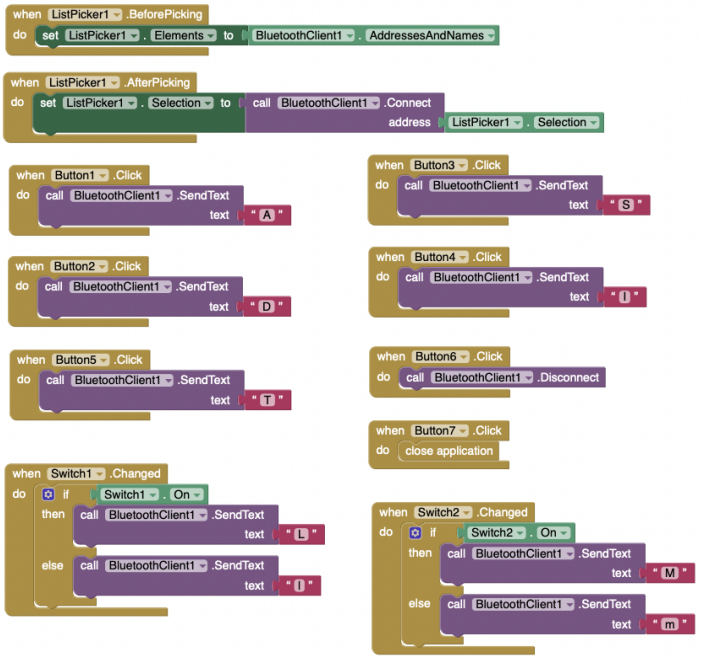
Cada vez que se genera la interrupción por desbordamiento de Timer 0, se incrementa el contador_izquierda y contador_derecha. Luego se verifican 5 condiciones y para cada una se realiza una acción diferente en caso de cumplirse:

- Si avanza es 1: significa que el coche está avanzando para atrás o adelante por lo que contador aumenta y se muestra en el LCD “Metros: “ y los metros recorridos.
- Si contador es 76: significa que el coche a avanzado un segundo por lo que aumentan se le aumenta 0.1325 a los metros recorridos ya que el coche avanza aproximadamente 0.1325 m/s. Además se reinicia contador para poder llegar de nuevo a 76 y contar otro segundo.
- Si bandera_luz = 1 y bandera_derecha = 0 y bandera_izquierda = 0: significa que se activan los faros y de acuerdo a la intensidad de luz se prenden los LEDs. Por lo tanto se obtiene un valor analógico de la fotorresistencia. Se le resta dicho valor a 180 para que el cambio de luces sea más notorio y ese valor se le asigna al PWM de ambos faros junto con un retardo de 10 ms. Esto se hace cada vez que se genera la interrupción mientras la bandera_luz siga prendida.
- Si contador_derecha = 45 y bandera_derecha = 1 o si bandera_maniobra = 1: significa que o se va a hacer un giro a la derecha (direccionales) o se prendieron las intermitentes. Entonces se hace un ciclo *for* donde se va aumentando de 5 en 5 el valor de tiempo de encendido del PWM para el faro derecho para ir de 0 a 255 o de 0 a 100%. Esto se realiza cada 0.6 segundos, en caso de ser direccionales, o cada que se entra a la interrupción, en caso de ser intermitentes.

- Si contador_izquierda = 45 y bandera_izquierda = 1 o si bandera_maniobra = 1: se realiza lo mismo que el caso anterior pero para el faro izquierdo.

Debido a que el coche se controló mediante Bluetooth, se desarrolló una aplicación utilizando el entorno de desarrollo MIT App Inventor. La interfaz está compuesta por 5 botones que controlan la dirección en la que se moverá el robot, cada botón con la letra que se manda y en forma de flecha. También se tienen 3 botones más para conectar el dispositivo Bluetooth, para desconectarlo y para salir de la aplicación. Y se tienen 2 switches para activar y desactivar los faros y las intermitentes. El diseño de la interfaz es el siguiente:

Además se programó la aplicación mediante la unión de bloques de código. En ellos se define como mostrar los dispositivos Bluetooth y cómo enlazar el que se escoja. También se programa qué letra mandar al presionar cierto botón, cómo desconectar el dispositivo Bluetooth y cómo cerrar la aplicación. Los bloques se muestran a continuación.



Programa comentado.

```
#include <16f877.h>                          //Biblioteca para PIC16F877
#fuses HS,NOWDT,NOPROTECT,
#device ADC = 8                              //8 bits de lectura de entrada analógica
#use delay(clock=20000000)                   //Configurando el reloj
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)          //Configurando comunicación serie asíncrona
#use i2c(MASTER, SDA=PIN_C4, SCL=PIN_C3, SLOW, NOFORCE_SW)
/*
Directiva para protocolo I2C, se configura:
-Modo maestro
-SDA en Pin C4
-CSL en Pin C3
-Velocidad baja
*/
#include <i2c_LCD.c>                         //Biblioteca para LCD con I2c
#org 0x1F00,0x1FFF void loader16F877(void){}

int contador, avanza, i;
/*
contador:   contador para los metros del vehículo
avanza:     bandera para contar el avance en metros
*/
unsigned int intensidad;                     //Intensidad del LED a partir del convertidor
int bandera_derecha, bandera_izquierda, bandera_luz, bandera_maniobra;
/*
BANDERAS
derecha:    bandera para activar direccional derecha
izquierda:  bandera para activar direccional izquierda
luz:        bandera para activar faros de luz
```

```
maniobra:  bandera para activar intermitentes
*/
int contador_izquierda,contador_derecha;
/*
CONTADORES
izquierda:  contador para direccional izquierda
derecha:    contador para direccional derecha
*/
float metros;                                   //número de metros recorridos
char valor;                                     //valor de control del coche

#int_rtcc                                       //Interrupción desbordamiento TIMER0
clock_isr(){
  contador_izquierda++;                         //Incrementa contador izquierda
  contador_derecha++;                           //Incrementa contador derecha

  //En caso de que se active bandera de avanzar
  if(avanza == 1){
    contador++;                                 //Incrementa contador de metros
    lcd_gotoxy(1,2);                            //Posicionar cursor en segundo renglón
    printf(lcd_putc,"Metros: %f m", metros);    //Muestra metros en LCD
  }

  //En caso de que se cumpla el segundo
  if(contador == 76){
    metros += 0.1325;                           //Aumentar metros por 0.1325
    contador = 0;                               //Reinicia contador
  }

  //En caso de que se prendan las luces y se apaguen direccionales
  if(bandera_luz == 1 && bandera_izquierda == 0 && bandera_derecha == 0){
    intensidad = read_adc();                    //La intensidad es el valor del convertidor
    set_pwm1_duty(180-intensidad);              //PWM1 es el inverso del convertidor
    set_pwm2_duty(180-intensidad);              //PWM2 es el inverso del convertidor
    delay_ms(10);                               //Retardo de 10ms
  }

  //En caso de activar la direccional izquierda
  if((contador_izquierda == 45 && bandera_izquierda == 1) || bandera_maniobra == 1 ){
    for(i  = 0;i < 255; i += 5){                //De 0 a 255
      set_pwm1_duty(i);                         //Incrementar PWM1 de 5 en 5
      delay_ms(10);                             //Retardo de 10ms
     }
    contador_izquierda = 0;                     //Reiniciar contador direccional izquierda
  }

  //En caso de activar la direccional derecha
  if((contador_derecha == 45 && bandera_derecha == 1) || bandera_maniobra == 1){
    for(i  = 0;i < 255; i += 5){                //De 0 a 255
      set_pwm2_duty(i);                         //Incrementar PWM2 de 5 en 5
      delay_ms(10);                             //Retardo de 10ms
    }
    contador_derecha = 0;                       //Reiniciar contador direccional derecha
  }
}

void main(){
```

```
//configuracion de pwm
setup_ccp1(CCP_PWM);                    //Habilita CCP1 como PWM
setup_ccp2(CCP_PWM);                    //Habilita CCP2 como PWM
setup_timer_2(T2_DIV_BY_16, 255, 1);        //Configura el TIMER2 con predivisor, periodo y postescalador

//configuracion de interrupciones
set_timer0(0);                  //Inicia Timer en 0
setup_counters(RTCC_INTERNAL,RTCC_DIV_256);     //Fuente de reloj y pre-divisor
enable_interrupts(INT_RTCC);            //Habilita interrupción de Timer0
enable_interrupts(GLOBAL);              //Habilita interrupciones globales

//configuracion de convertidor a/d
setup_adc(ADC_CLOCK_INTERNAL);              //Configura el reloj del convertidor
setup_adc_ports(32);                //Configura el puerto E0 como analógico
set_adc_channel(5);                 //Se configura el canal como el puerto E0
delay_ms(100);                  //Retardo de 100ms

lcd_init(0x4E, 16, 2);                  //Inicializa LCD en 0x4E

//Se inicializan todas las variables en 0
contador = 0;
avanza = 0;
metros = 0;
bandera_derecha = 0;
bandera_izquierda = 0;
bandera_luz = 0;
contador_izquierda = 0;
contador_derecha = 0;
i = 0;

while(TRUE){
  valor = getc();               //Leer valor de comunicación en serie asíncrona
  switch(valor){                //Switch con el valor introducido
    case 'S':                   //S -> Valor para parar el coche
      set_pwm1_duty(0);             //Cambiar valor PWM1 a 0
      set_pwm2_duty(0);             //Cambiar valor PWM2 a 0
      lcd_clear();              //Limpiar el LCD
      lcd_gotoxy(1,1);              //Posicionar cursor en primer renglón
      printf(lcd_putc,"EN PARO");       //Indicar en LCD que está en paro el coche
      avanza = 0;               //Desactivar bandera de avance
      output_b(0x00);               //Apagar motores
      output_a(0x00);               //Apagar luces reversa
      bandera_derecha = 0;          //Apagar direccional derecha
      bandera_izquierda = 0;            //Apagar direccional izquierda
      break;
    case 'A':                   //A -> Valor para que el coche avance
      set_pwm1_duty(0);             //Cambiar valor PWM1 a 0
      set_pwm2_duty(0);             //Cambiar valor PWM2 a 0
      lcd_clear();              //Limpiar el LCD
      lcd_gotoxy(1,1);              //Posicionar cursor en primer renglón
      printf(lcd_putc,"AVANZANDO");       //Indicar en LCD que el coche está avanzando
      avanza = 1;               //Activar bandera de avance
      output_b(0x0A);               //Motores en avance
      output_a(0x00);               //Apagar luces reversa
      bandera_derecha = 0;          //Apagar direccional derecha
      bandera_izquierda = 0;            //Apagar direccional izquierda
      break;
```

```
case 'T':                       //T -> Valor para que el coche retroceda
  set_pwm1_duty(0);                 //Cambiar valor PWM1 a 0
  set_pwm2_duty(0);                 //Cambiar valor PWM2 a 0
  lcd_clear();                  //Limpiar el LCD
  lcd_gotoxy(1,1);                  //Posicionar cursor en primer renglón
  printf(lcd_putc,"EN REVERSA");       //Indicar en LCD que el coche está en reversa
  avanza = 1;                   //Activar bandera de avance
  output_b(0x05);                   //Motores en reversa
  output_a(0x06);                   //Prender luces reversa
  bandera_derecha = 0;              //Apagar direccional derecha
  bandera_izquierda = 0;            //Apagar direccional izquierda
  break;
case 'D':                       //D -> Valor para girar a la derecha
  set_pwm1_duty(0);                 //Cambiar valor PWM1 a 0
  lcd_clear();                  //Limpiar el LCD
  lcd_gotoxy(1,1);                  //Posicionar cursor en primer renglón
  printf(lcd_putc,"DERECHA");          //Indicar en LCD que el coche gira a la derecha
  avanza = 0;                   //Desactivar bandera de avance
  output_b(0x09);                   //Motores giran a la derecha
  output_a(0x02);                   //Prender luz de reversa derecha
  bandera_derecha = 1;              //Prender direccional derecha
  bandera_izquierda = 0;            //Apagar direccional izquierda
  contador_derecha = 0;             //Reinicia contador derecha
  contador_izquierda = 0;           //Reinicia contador izquierda
  break;
case 'I':                       //I -> Valor para girar a la izquierda
  set_pwm2_duty(0);                 //Cambiar valor PWM2 a 0
  bandera_maniobra = 0;             //Apagar bandera maniobra
  lcd_clear();                  //Limpiar el LCD
  lcd_gotoxy(1,1);                  //Posicionar cursor en primer renglón
  printf(lcd_putc,"IZQUIERDA");        //Indicar en LCD que el coche gira a la izquierda
  avanza = 0;                   //Desactivar bandera de avance
  output_b(0x06);                   //Motores giran a la izquierda
  output_a(0x04);                   //Prender luz de reversa izquierda
  bandera_derecha = 0;              //Apagar direccional derecha
  bandera_izquierda = 1;            //Prender direccional izquierda
  contador_derecha = 0;             //Reinicia contador derecha
  contador_izquierda = 0;           //Reinicia contador izquierda
  break;
case 'M':                       //M -> Valor para activar maniobra
  bandera_maniobra = 1;             //Prender bandera maniobra
  lcd_clear();                  //Limpiar el LCD
  lcd_gotoxy(1,1);                  //Posicionar cursor en primer renglón
  printf(lcd_putc,"MANIOBRA");         //Indicar en LCD que el coche está en maniobra
  break;
case 'm':                       //m -> Valor para apagar Maniobra
  bandera_maniobra = 0;             //Apagar la bandera de maniobra
  lcd_clear();                  //Limpiar el LCD
  set_pwm1_duty(0);                 //Cambiar valor PWM1 a 0
  set_pwm2_duty(0);                 //Cambiar valor PWM2 a 0
  break;
case 'L':                       //L -> Valor para prender las luces
  bandera_maniobra = 0;             //Apagar la bandera de maniobra
  bandera_luz = 1;                  //Prender la bandera de faros
  break;
case 'l':                       //l -> Valor para apagar luces
  bandera_luz = 0;                  //Apagar la bandera de faros
```

```
        set_pwm1_duty(0);               //Cambiar valor PWM1 a 0
        set_pwm2_duty(0);               //Cambiar valor PWM2 a 0
        break;
      default:

    };
  }
}
```

Conclusiones y/o comentarios.

Barreiro Valdez Alejandro:

Este proyecto engloba todos los conocimientos y habilidades adquiridas a lo largo del curso de Microcontroladores, al implementar diversas funcionalidades que han sido estudiadas en conjunto y que se interrelacionan entre sí. Al simular un automóvil, se pone de manifiesto la utilidad y versatilidad de estos circuitos en un entorno profesional y educativo, dejando demostrado las habilidades generadas con los conocimientos absorbidos, dando como resultado este proyecto.

Gil Márquez Arath Emiliano:

Este proyecto final representa todo lo visto y aprendido durante el curso de Microcontroladores, ya que implementa varias funcionalidades antes vistas en conjunto y que trabajan entre sí, como la implementación del sistema mínimo del microcontrolador, el uso de sus puertos, el uso del LCD en conjunto con el I2C, la función de PWM, la comunicación por Bluetooth por medio de una aplicación móvil. La simulación de un automóvil es un buen ejemplo para demostrar cómo podríamos usar estos circuitos en un ámbito laboral y nos da la oportunidad de mostrar nuestras habilidades desarrolladas y adquiridas relevantes en el campo de la electrónica y la programación.

Herrera Carrillo Cristhian:

En este proyecto final logramos hacer la conjunción de todo lo visto en clase. haciendo un sistema integrado funcional. Desde el sistema mínimo de nuestro microcontrolador, hasta la comunicación síncrona por Bluetooth. escogimos hacer un prototipo de automóvil inteligente porque decidimos que era una buena forma de integrar todos los temas del semestre, demostrando la habilidad obtenida durante el semestre.

Zepeda Baeza Jessica:

Este proyecto final de microcontroladores ha sido un testimonio de nuestro crecimiento y competencia en el campo de nuestra carrera. Al implementar diversas funcionalidades en conjunto y aplicarlas en una simulación de automóvil, hemos demostrado nuestra capacidad para aprovechar los microcontroladores de manera efectiva y destacar en un ámbito laboral.

La simulación de un automóvil ha sido un excelente ejemplo para ilustrar cómo estos circuitos pueden ser empleados en un contexto laboral. A través de esta representación, hemos podido demostrar la utilidad y la versatilidad de los microcontroladores en la industria, así como nuestra capacidad para utilizarlos de manera efectiva.