PROYECTO FINAL	
Título:	**Silla de ruedas**
Tipo de Documento:	Reporte Técnico
Fecha:	10-06-2022
Preparado por:	Gómez Cruz Erick Iván
Aprobado por:	

Edición	Rev	Fecha	Razones del cambio	Sección afectada

I. Planteamiento del proyecto

- **Objetivo:**
  - Desarrollar un proyecto que resuelva una necesidad presente en nuestra sociedad.
  - Aplicar los conocimientos sobre microcontroladores y microprocesadores en el desarrollo del mismo.

- **Descripción del proyecto:**

El proyecto consiste en un prototipo de silla de ruidas motorizada que le permite al usuario desplazarse de forma ágil y segura sin mayor esfuerzo. Ofrece dos formas de conducción: mediante el desplazamiento de un joystick integrado o mediante una aplicación de dispositivos móviles que se conecta mediante Bluetooth a la sillas de ruenas.

La aplicación ofrece una interfaz amigable y minimalista, en la cual puedes controlar la silla mediante 5 botones diferentes (asociados a las direcciones) o mediante el servicio de voz propio del dispositivo dando ordenes predefinidas.

Para brindar más seguridad, se le agregó un sensor ultrasónico el cual detecta la distancia entre el equipo y un obstáculo. Para este caso se definió una distancia límite de 30 cm. Una vez que se detecta el obstáculo, se detiene la marcha del equipo y le impide al usuario dar instrucciones de desplazamiento. La silla de ruedas avanza hacia atrás hasta que la se superen los 30 cm límites y, una vez superados, se le vuelve a permitir al usuario controlar el sentido de desplazamiento.

Además, ofrece difentes extras que le brindan mayor valor al equipo: un display que indica las direcciones en la cual se desplaza y un zumbador comienza a pitar una vez que se alcanzan los 30 cm límites y se detiene una vez rebasados.

- **Metodología**

Para lograr cumplir con los requerimientos y lograr un proyecto funcional, aplicaremos la metodología Ulrich, que comprende varias etapas:

1. Planeación: declarar los objetivos, metas y limitaciones;
2. Desarrollo del concepto: Indentifcar necesidades y requerimientos, analizar productos existentes y justificar económicamente el proyecto;
3. Diseño nivel sistema: Determinar diseño a bloques, determinar las especificaciones funcionales y determinar un diagrama de flujo del proceso;
4. Diseño de detalle: Especificar completamente elementos como materiales, circuitos, etc.
5. Pruebas y refinamiento: Contruir y evaluar versiones.

II. Marco teórico

El INEGI identifica a las personas con discapacidad como aquellas que tienen dificultad para llevar a cabo actividades consideradas básicas, como: ver, escuchar, caminar, recordar o concentrarse, realizar su cuidado personal y comunicarse.

Dichas dificultades impactan en la persona puesto que su interacción se ve limitada por la forma en que se organiza la sociedad y el mundo, lo cual le impide desarrollarse plenamente.

Según la Organización Mundial de la Salud al 2020, más de 1,000 millones de personas en todo el mundo con algún tipo de discapacidad, lo que equivale a un 15% de la población mundial. Además, casi 190 millones

de personas requieren con frecuencia de servicios de asistencia. Este número está a la alza debido al envejecimiento de la población y al incremento de enfermedades crónicas.

De acuerdo con el Censo de Población y Vivienda 2020, en México hay poco más de seis millones de personas con algún tipo de discapacidad, lo que representa un 4.9% de la población total del país. Este sector poblacional se encuentra compuesto en un 53% de mujeres y un 47% de hombres.

Ilustración 1 INEGI. Centro de Población y Vivienda 2020.

Que a su vez se encuentras distribuido de la siguiente forma

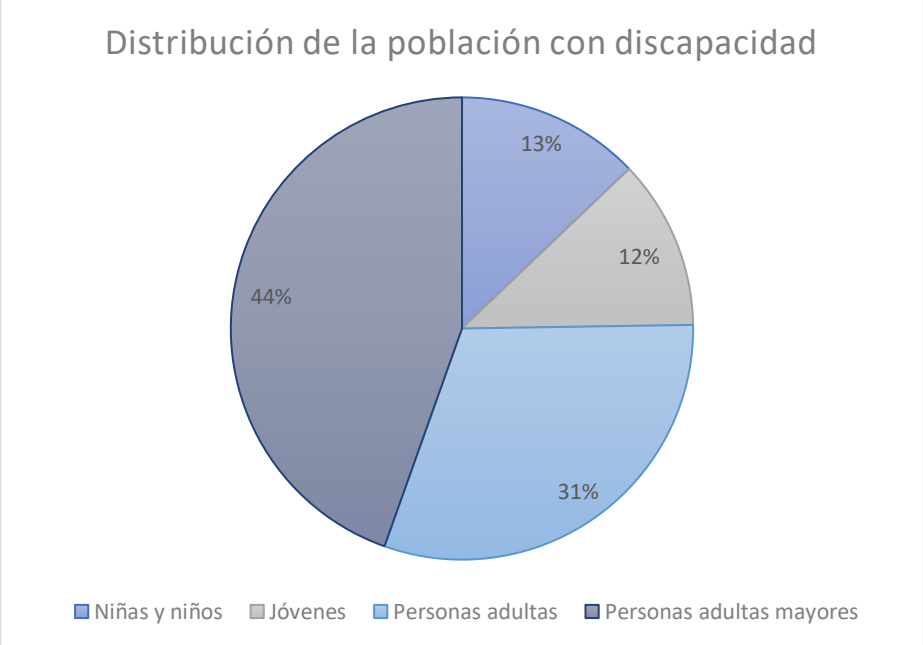


Ilustración 2 INEGI. Centro de Población y Vivienda 2020

Según el INEGI, las discapacidades pueden ser asociadas a un factor físico o a un factor mental, a su vez, las clasifica como sigue:

- Caminar, subir o bajar: Hace referencia a la dificultad de una persona para moverse, caminar, desplazarse o subir escaleras debido a la falta de toda o una parte de sus piernas; incluye también a quienes teniendo sus piernas no tienen movimiento o presentan restricciones para moverse, de tal forma que necesitan ayuda de otras personas, silla de ruedas u otro aparato, como andadera o pierna artificial.
- Ver: Abarca la pérdida total de la vista en uno o ambos ojos, así como a los débiles visuales y a los que aun usando lentes no pueden ver bien por lo avanzado de sus problemas visuales.
- Oír: Incluye a las personas que no pueden oír, así como aquellas que presentan dificultad para escuchar (debilidad auditiva), en uno o ambos oídos, a las que aun usando aparato auditivo tiene dificultad para escuchar debido a lo avanzado de su problema.
- Hablar o comunicarse: Hace referencia a los problemas para comunicarse con los demás, debido a limitaciones para hablar o porque no pueden platicar o conversar de forma comprensible.
- Recordar o concentrarse: Incluye las limitaciones o dificultades para aprender una nueva tarea o para poner atención por determinado tiempo, así como limitaciones para recordar información o actividades que se deben realizar en la vida cotidiana.
- Dificultad para bañarse, vestirse o comer: Son los problemas que tiene una persona para desarrollar tareas del cuidado personal o cuidar su salud.

Y estas discapacidades se distribuyen de la siguiente forma

Ilustración 3 INEGI. Censo de Población y Vivienda 2020

De acuerdo con los datos del Censo 2020, para el 15 de marzo de 2020 en México la prevalencia de discapacidad es del 4.42%. Las entidades con la menor prevalencia son Quintana Roo, Nuevo León y Chiapas; mientras que Oaxaca, Guerrero y Tabasco reportan las prevalencias más altas. Además, las mujeres tienen una prevalencia ligeramente mayor que los hombres en casi todas las entidades, con excepción de Chiapas, Hidalgo, San Luis Potosí y Tabasco.

Debido a la gran presencia de la discapacidad a nivel nacional e internacional, la integración en la sociedad de las personas con discapacidad debe ser prioridad. Se debe garantizar su participación en la elaboración, aplicación, supervisión y evaluación de las políticas y los programas en las esferas política, económica y social con el fin de abatir la desigualdad y fomentar una cultura inclusiva.

III. Diseño

- Concepto

Debido a los datos existentes en nuestro país sobre discapacidad, decidí desarrollar un prototipo de ayuda motriz para este sector de la población. El concepto a grandez rasgos consiste en un equipo que le permita al usuario desplazarse en las 4 direcciones básicas: adelante, atrás, izquierda y derecha. Esta discapacidad afecta a diferentes poblaciones de diferentes edades, por lo que entre más alternativas de conducción se ofrezcan se podrá alcanzar a satisfacer las necesidades de más personas.

Además, como resultado de la integración del sector con discapacidad motriz a la vida cotidiana, se deberá desarrollar una nueva cultura vial que involucre el compromiso de peatones, automovilistas, cilicstas, entre otros; esto con el fin de permitir el respzoeto y sana convivencia entre todos. Si bien es un factor que se puede desarrollar en un plazo relativamente cercano por medio de diferentes estrategias, es pertinente brindarle al usuario la posibilidad de hacerse visible para con los otros.

En resumen, necesitamos un dispositivo que:

- Se desplace hacia la dirección indicada por el usuario;
- La forma de indicar la dirección debe ser más de una;
- Agregar elementos que visibilicen al usuario en las calles de nuestro país.

Hoy en día, existen algunos productos que le brindan la posibilidad a los usuarios de desplazarse mediante una silla de ruedas eléctrica, un ejemplo de ellos es la silla Hercules Lite EX de la empresa Bangeran, que ofrece un joystick para controlarla. El equipo tiene un precio al público mexicano de 21 mil pesos en una reconocida plataforma de ventas en línea

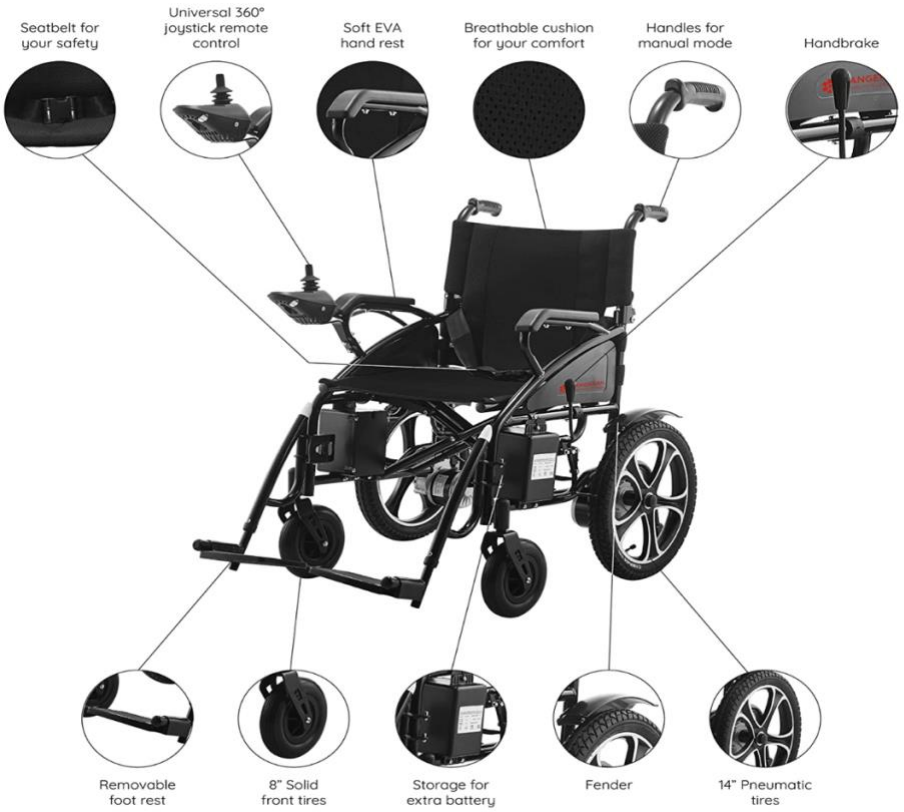


Ilustración 4 Silla de ruedas marca Bangeran

Otro ejemplo viene de la marca Wisging, que ofrece una silla de ruedas eléctricas que ofrece una velocidad máxima de 3.72 mph, siendo sus motores alimentados con 24 Vdc. Este equipo tiene un precio de 40 mil pesos en una página reconocida de ventas en línea.

Ilustración 5 Silla de ruedas eléctrica, Wisging.

Como se aprecia, existen muchas marcas y modelos que comercializan sillas de ruedas eléctricas para un desplazamiento fácil para personas con una discapacidad motriz. Sin embargo, su principal problema es su precio. Muchos de estos precios, incluso los precios de las sillas de ruedas convencionales superan las posibilidades de muchas familias mexicanas que perciben ingresos medios o bajos, por lo que brindarles una alternativa económica es importante. Además, el hecho de producir una silla de ruedas nacional permite la construcción a la medida, ya que conocemos los problemas y retos que la ciudad ofrece y sabemos que soluciones implementar; por otro lado, nos permite producir equipos más económicos para lograr un alcance mayor en las familias mexicanas.

- **Diseño funcional**

Este, al ser un prototipo, es construido en una escala mucho menor a la requerida en un producto final, por lo que los elementos usados al final son de características y especificaciones pequeñas. Primero, retomaremos los requerimientos especificados para la formación de bloques que, juntos, formen el producto final.

- El primer bloque es el encargado de mover el equipo. Por lo que necesitamos motores acoplados a ruedas. Por las condiciones dadas, se usarán motores de DC que puedan desplazarse siendo alimentados con pilas.
- Además, así como en otros ejemplos comerciales, se necesita un Joystick para controlar el equipo. Nuestro producto busca ofrecer una alternativa más para conducir el equipo: hacerlo mediante una aplicación móvil. Por lo que necesitamos agregar una conexión Bluetooth entre

la silla y un celular. Para esto, existen módulos que ofrecen el joystick y el bluethooth de forma fácil. Ambos compatibles con microcontroladores y que operan a bajos niveles de voltaje.

- Para lograr medir constantemente la distancia necesitamos un sensor, el cual será acoplado al chasis del equipo. Una opción para este son los sensores ultrasónicos, que su principio de operación básicamente consiste en emitir una onda sonora que, al chocar con un obstaculo, rebota y determina el instante en que regresa al módulo. Por software, podemos determinar cuando tarda en volver la onda y así la distancia;
- Para visivilizar a los usuarios, de plantean dos formas de hacerlo:
  - Usar un buzzer que sirva como alarma ante ciertas ciertas cirscunstancias. De este existen de dos tipos: pasivos y activos; ambos operan bajo niveles de voltaje conmunes en los microcontroladores;
  - Agregar un display que muestre las direcciones en las que se está desplazando el equipo.Existen de diferentes tamaños, tecnologías y protocolos de comunicación.

- **Diseño a detalle**
  - Para los motores, se usarán 2 motorreductor de 2 ejes. Tiene los siguientes datos:

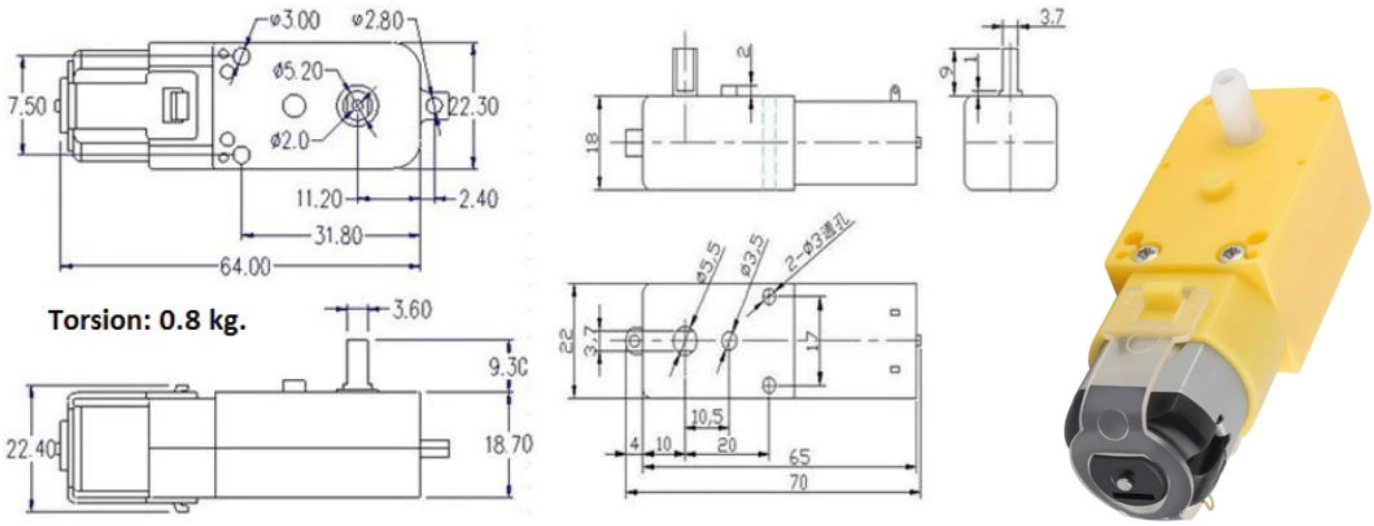


Ilustración 6 Motorreductor [7].

Tabla 1 Datos del motorreductor [7].

Reducción	48:1
Velocidad	170 rpm sin carga
Vpltaje de operación	3-12 V
Corriente	250 mA
Torque máximo	800g/cm @ 3 V

- Para mover el motor, se usará el driver L298N. Este permite controlar 2 motores DC o un motor a pasos bipolar/Unipolar. El módulo es util para controlar el sentido de giro y la velocidad mediante señales PWM provenientes de un microcontrolador.

Tabla 2 Datos del puente H [14]

Voltaje de operación	5 a 35 V
Lógico	5 V
Capacidad de corriente	2 A
Potencia máxima	25 W
Consumo de corriente	0 a 36 mA

Debido a que el módulo necesita voltajes lógicos 5 V y el microcontrolador no ofrece esa cantidad de voltaje. Por eso se usará un circuito usando transistores TBJ, específicamente se trata de 4 BC-547C. El circuito de potencia se muestra en la ilustración 7, cuando nuestros puertos están con un estado alto, a la salida de la etapa de potencia tendremos casi 0 V, y cuando nuestro puerto tiene un valor de 0 nuestra etapa dará 5 V.

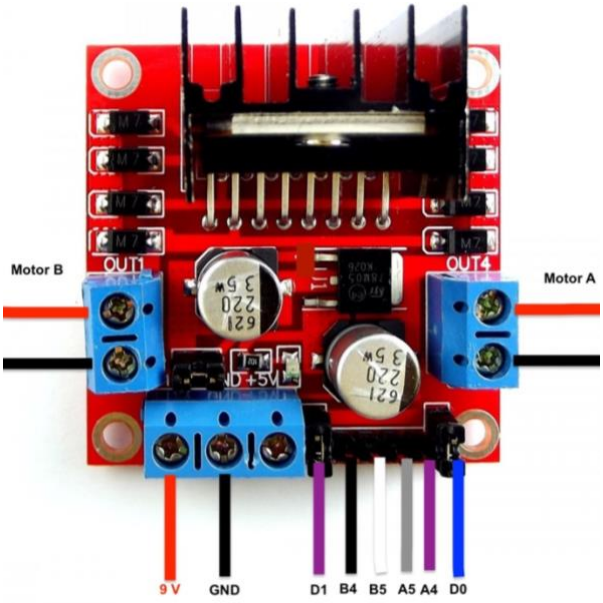


Ilustración 8 Conexiones en el Puente H

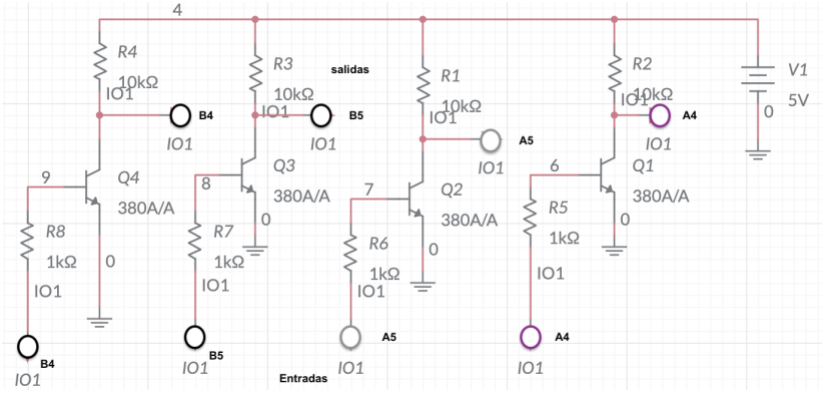


Ilustración 7 Etapa de potencia

- Para el joystick, usamos el módulo KY-023 es un dispositivo electromecánico que consta de dos potenciómetros de 10k en un ángulo de 90 grados, por lo que requiere de 2 pines analógicos para realizar la interfaz con cualquier tarjeta de desarrollo.

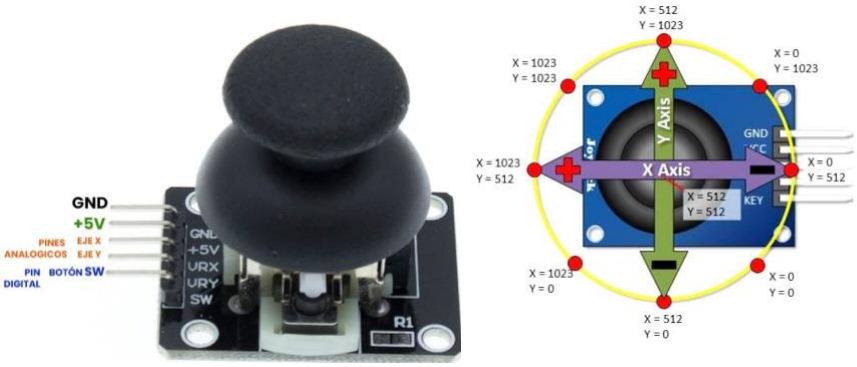


Ilustración 9 Módulo KY-023 [8]

Tabla 3 Datos del módulo KY-023 [8]

Voltaje de alimentación	3.3 – 5 V dc
Salida	Analógica (X,Y) y digital (Z)
Tipo de pulsador central	Normalmente abierto

- El módulo Bluethooth utilizado es el HC-06 se comporta como esclavo y sirve para escuchar peticiones de conexión. Al conectarse el módulo transmite todos los datos recibidos del maestro. El módulo cumple con las especificaciones del estándar Bluethooth 2.0 a 2.4 HGz que es compatible con celulares Android, más no con los Iphone.

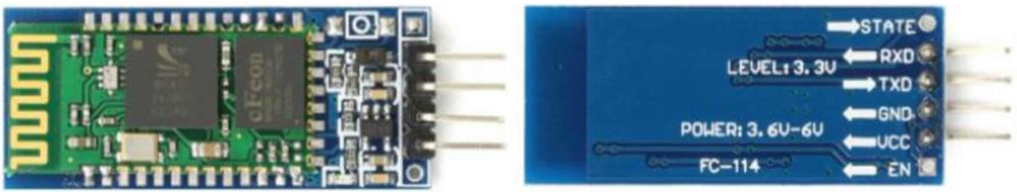


Ilustración 10 Módulo HC-06 [9]

Tabla 4 Datos del módulo HC-06 [9]

Voltaje de operación:	3.3 a 5 V dc
Consumo de corriente	30 a 40 mA
Potencia de emosión	4 dBm, clase 2
Alcance	5 a 10 m
Velocidad asíncrona	2Mbps /160 kbps
Velocidad síncrona	1 MBps/1 Mbps

- El sensor ultrasónico será el HC-SR04 que sirve para medir distancias por medio de sus dos transductores: un micrófono y altavoz. Genera pulsos de alta frecuencia(no perceptible por el ser humano) que rebota en los objetos cercanos y es reflejado hacia el sensor, que es captado por un micrófono. Son sensores económicos y fácil de usar.

Ilustración 11 Sensor Ultrasónico [10].

Tabla 5 Datos del sensor ultrasónico [10]

Voltaje de funcionamiento	5 V dc
Corriente de alimentación	**15 mA**
Rango de medición	**2 – 400 cm**
Frecuencia de trabajo	**40 kHz**
Precisión	**+- 3 mm**

- Para mostrar las direcciones en las que se desplaza el equipo se usará un display Oled de 128x32 pixeles, en un tamaño de 0.91 pulgadas. Su interfaz de comunicación es I2C y puede ser por diferentes microcontroladores. Sirve para crear aplicaciones portátiles donde se requiera el mínimo de tamaño y bajo consumo de energía.

Ilustración 12 Display Oled [11]

Tabla 6 Datos sobre el display Oled [11].

Color de pixeles	Blancos
Driver	SSD1306
Voltaje de operación	3.3 – 5.5 V dc
Consumo de energía	0.08 W cuando todos los pixeles están encendidos

- Para el buzzer usaremos uno de tipo de activo que es un transductor electroacústico. Su construcción consta de una bobina y un electroimán. Es utilizado principalmente para generar alarmas en diferentes aplicaciones.

Ilustración 13 Buzzer activo [12]

Tabla 7 Datos buzzer activo [12]

Voltaje de operación	5 V dc
Corriente máxima	**30 mA**
Frecuencia de resonancia	**23 kHz**
Temperatura de trabajo	**-20 a 70 °C**

- Para alimentar el buzzer, se usará un potenciometro digital, el cual se podrá controlar con el microcontrolador. El integrado empleado es el MCP41010.

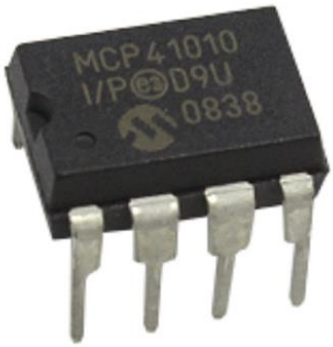


Ilustración 14 Potenciometró digital MCP41010 [13]

Tabla 8 Datos potenciometro MCP41010 [13]

Resistencia	10 k
Resolución	**8 bits, 256 pasos**
Interfaz	**SPI**
Voltaje de operación	**2.7 – 5.5 V dc**

- **Pruebas y refinamiento**

Para el desarrollo del proyecto, hubo muchos cambios para pasar de un estado inicial al final de este. Los más notorios fueron los cambios físicos, que incluyen la distribución de los elementos por el chasis; inicialmente se hicieron pruebas sosteniendo los circuitos en la mano, y solo se usaban un par del total final de elementos.

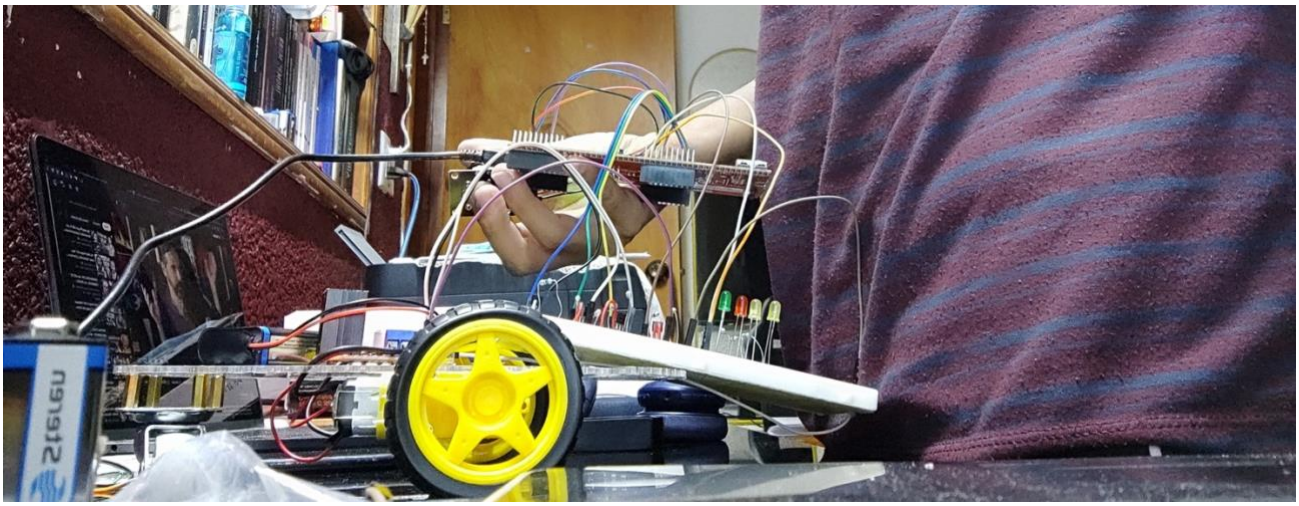


Ilustración 15 Estado del proyecto al 4 de mayo de 2022

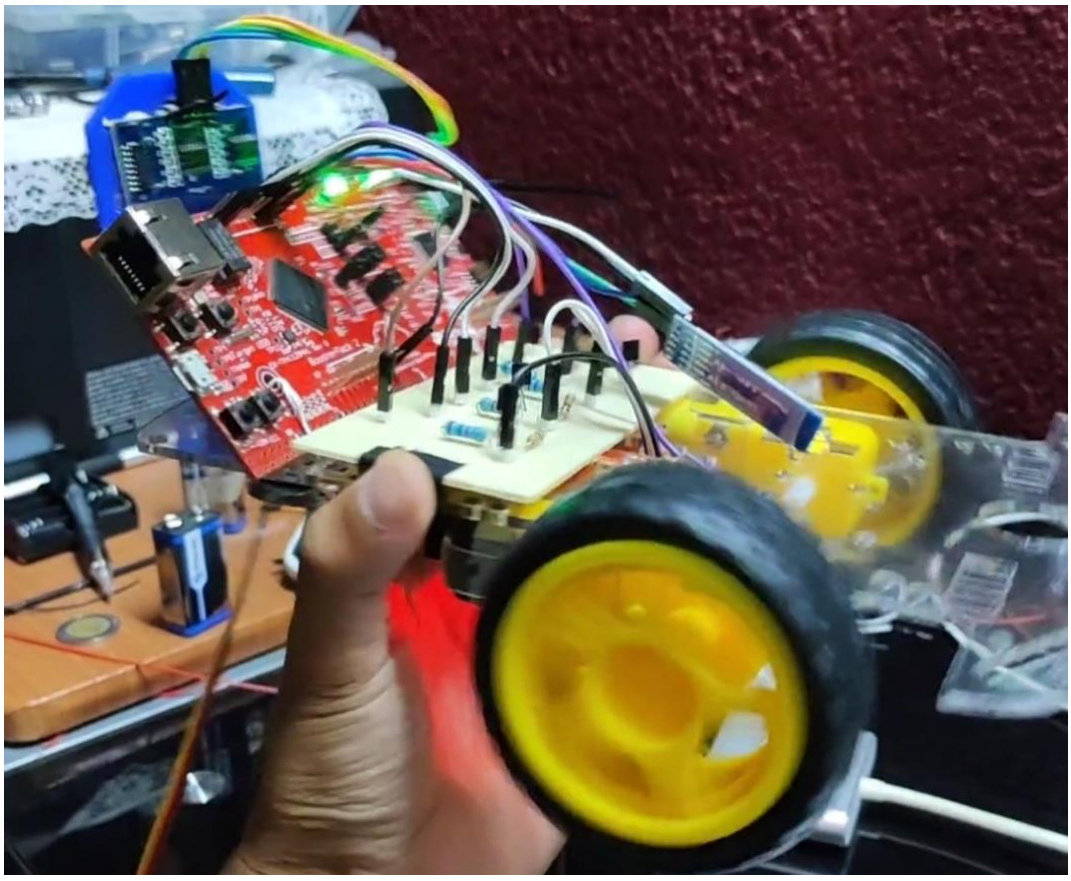


Ilustración 16 Avance del proyecto al 22 de mayo de 2022

Todos estos cambios se hicieron con el fin de distribuir de mejor forma los elementos, permitiendo un orden y una cierta estética.

Otro cambio hecho fue en la aplicación desarrollada en App Inventor. Inicialmente nuestra aplicación solo tenía algunas acciones. Con esta se fueron haciendo pruebas.

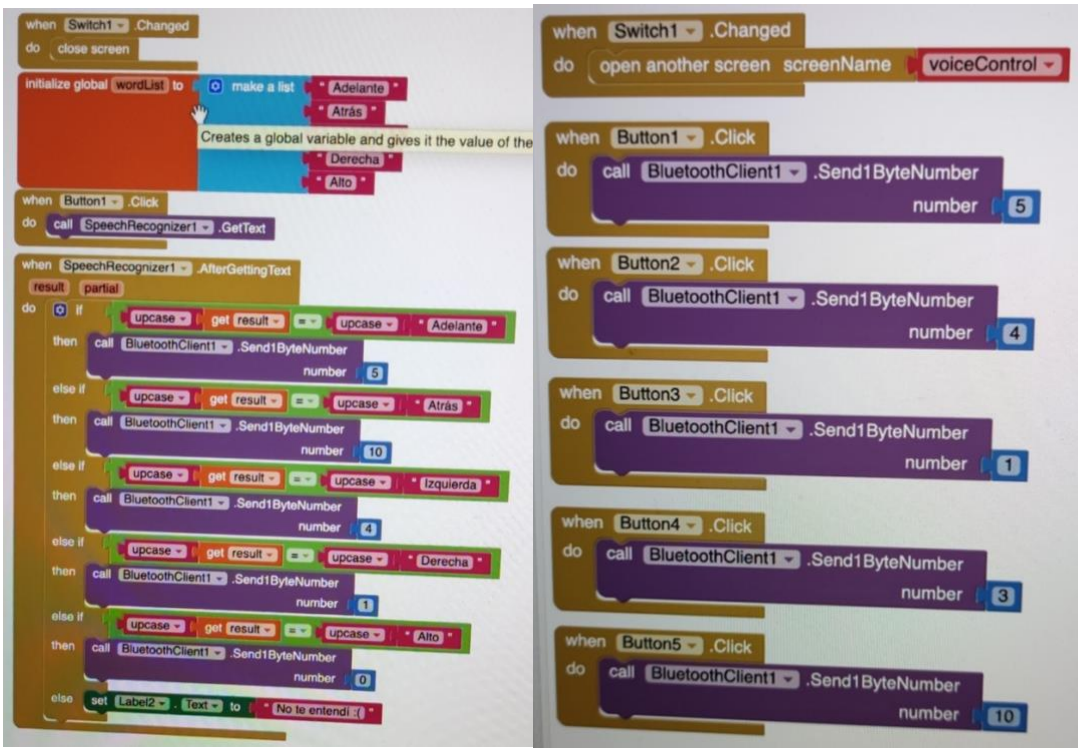


Ilustración 17 Acciones de la aplicación a 17 de mayo de 2022

Como se observa, inicialmente la aplicación tenía diferentes ventanas donde cada uno contenía una forma de mandar instrucciones: por botón o por voz. La forma para cambiar entre pantallas era mediante un switch, dependiendo de su posición era la pantalla que iba a mostrar. Esta opción tenía el inconveniente que cuando se pasaba entre pantallas se perdía la conexión al Bluetooth. Posteriormente se quitaron las múltiples pantallas y se optó por tener todo en una sola, e ir apareciendo o desapareciendo secciones de la pantalla para cada forma de mandar instrucciones, es decir, si se estaba en conducción por voz los botones desaparecían y solo estaba el ícono del micrófono, y mismo caso con la conducción por botón.

Una cosa que cambió de forma notable dentro del código fueron las flechas que se muestran en el display.

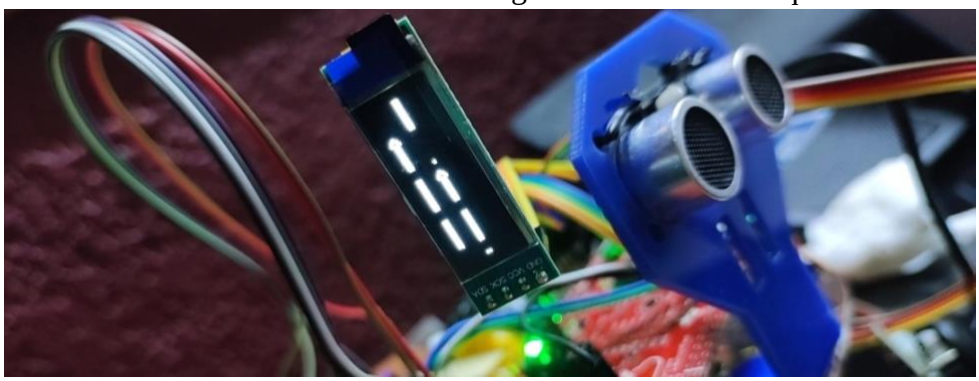
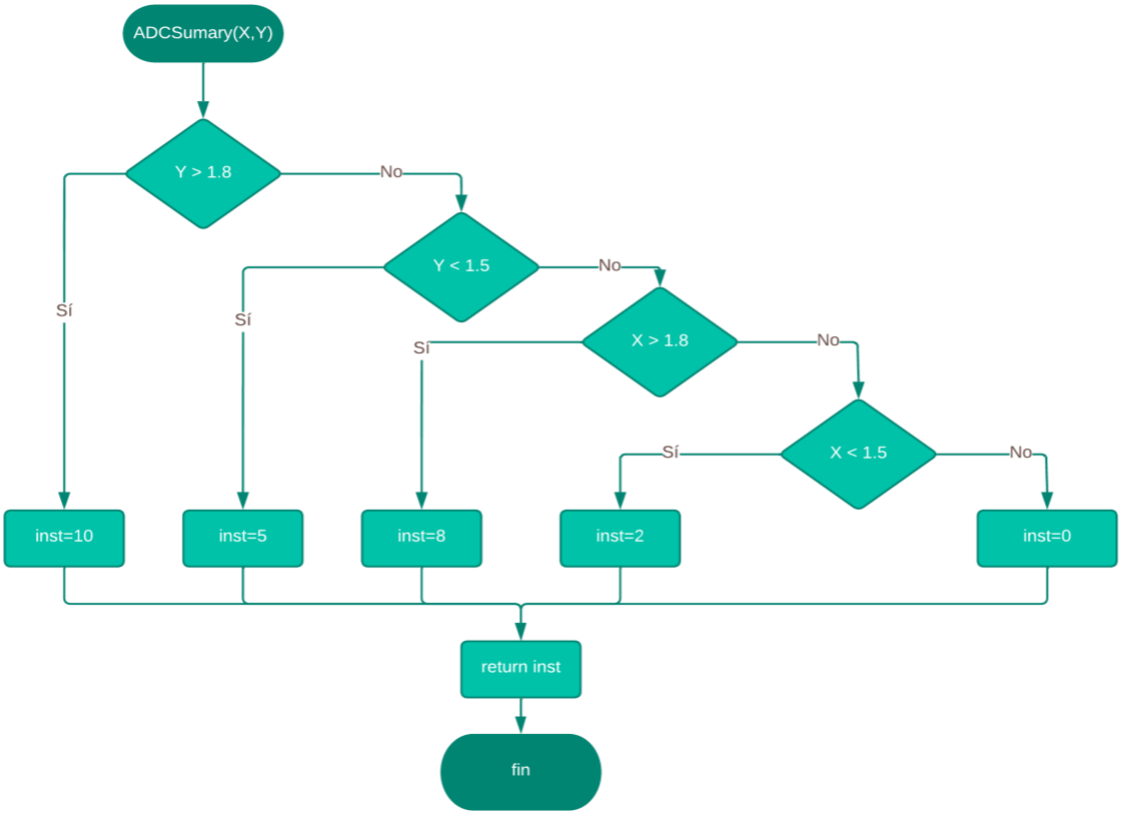


Ilustración 18 Flechas del display a 24 de mayo de 2022

IV. Diagrama de flujo y código comentado.

Comenzaremos por la función **ADCSumary**. Esta función es de tipo entero, recibe dos flotantes y retorna un entero. Su función está en asignar un valor a la variable **inst** dependiendo de los valores que tengan X y Y, cada uno asociada a la posición. Por lo cual, teniendo un valor alto en Y significa que el joystick está apuntando hacia adelante, si Y es bajo está apuntando hacia atrás, si X está en alto está apuntando a la derecha y si es bajo apunta a la izquierda. Entonces, asignamos:

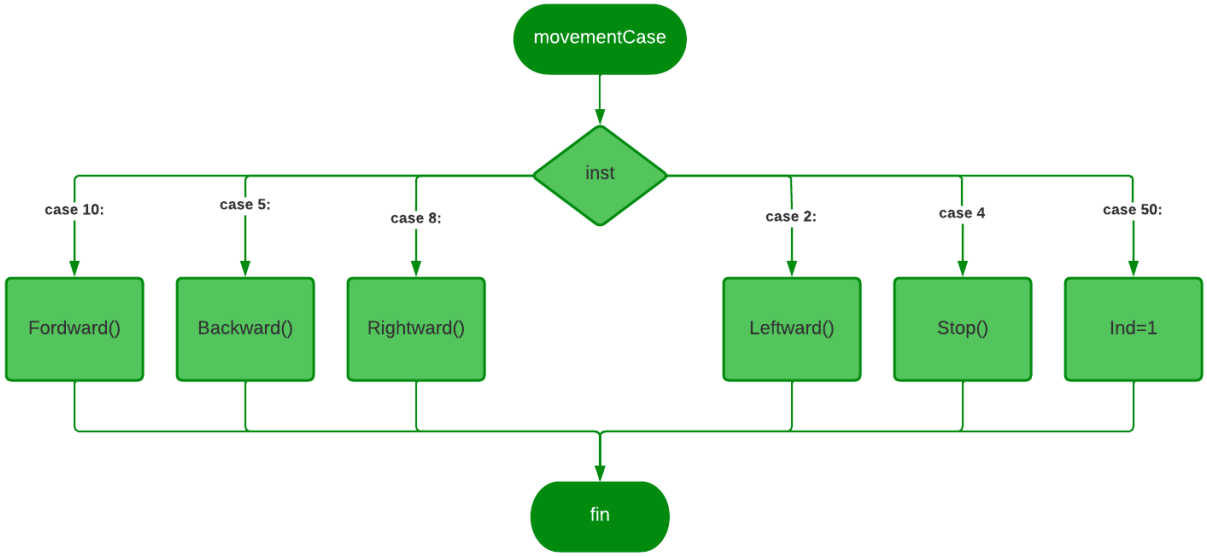
- Adelante=10 y atrás=5
- Derecha= 8 e izquierda=2
- Alto=4.




```
//-------------------------------------------------------------------------------------Función que define la posición del Joystick-------------------------------------
int ADCSumary(float X, float Y){                                                        //Recibe los valores del ADC asociados a X, Y
    int inst=0;
    if(Y>1.8){                                                                          //Si debe ir hacia adelante
        inst=10;
    }else{
        if(Y<1.5){                                                                      //Si debe ir hacia atrás
            inst=5;
        }else{
            if(X>1.8){                                                                  //Si debe ir hacia la izquierda
                inst=8;
            }else{
                if(X<1.5){                                                              //Si debe ir hacia la derecha
                    inst=2;
                }else{
                    inst=4;                                                             //Si debe estar detenido
                }
            }
        }

    }
    return inst;                                                                        //Regresa el valor asociada a la direcciόm de desplazamiento
}
```

Ilustración 19 Diagrama de flujo y código de la función ADCSumary

Con base en el valor que retorna la función **ADCSumary**, usamos la función **movementCase** para, dependiendo del valor, indicar la hacia donde se desplaza y los cambios que cada dirección implican. Esta función consiste en un switch-case que evalúa el valor de **inst** asociado a la dirección pertinente. En casa caso se llama la respectiva función de dirección.




```
//-------------------------------------------------------------------------------------Función de determina la dirección de desplazamiento dependiendo del valor obtenido en el ADCSumary--
void movementCase(int n, float Y){
    switch(n){
    case 10:
        Forward(Y);
        break;
    case 5:
        Backward(Y);
        break;
    case 8:
        Rightward();
        break;
    case 2:
        Leftward();
        break;
    case 4:
        Stop();
        break;
    default:
        break;
    }
}
```

```
//evalua si inst=10
//llama a la función fordward

//evalua si inst=5
//llama a la función backward

//evalua si inst=8
//llama a la función rightward

//evalua si inst=2
//llama a la función leftward

//evalua si inst=4
//llama a la función stop

//por dejecto no ejecuta ninguna función
```

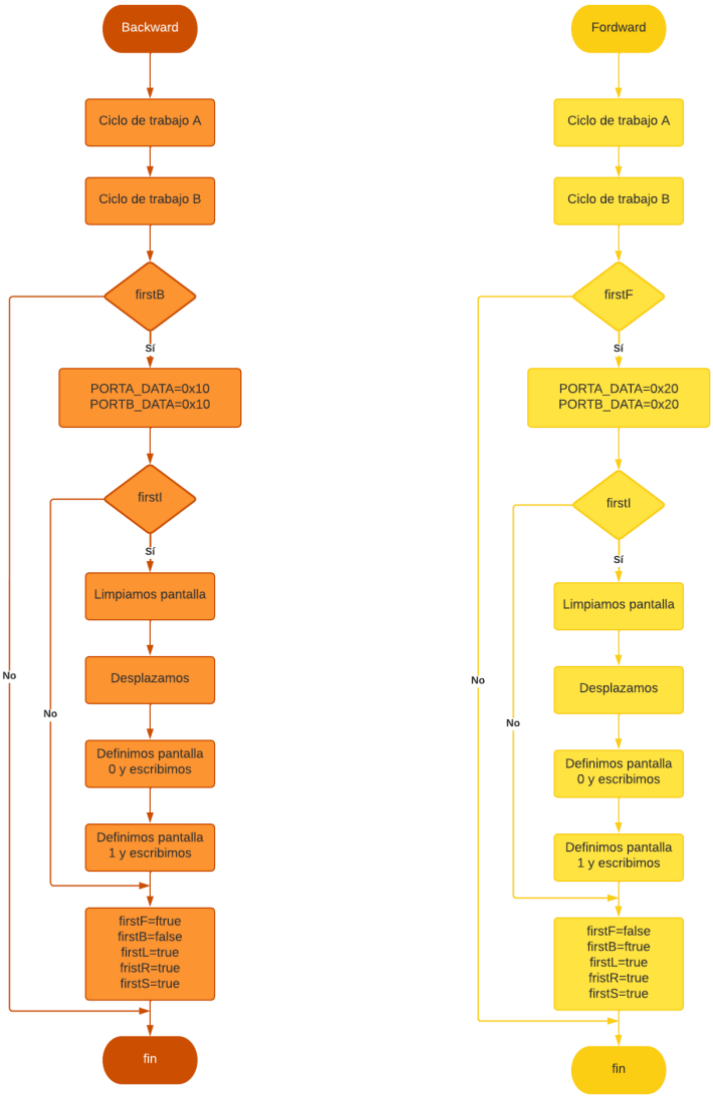
Ilustración 20 Diagrama de flujo y código de la función movementCase

A grandes rasgos, cada función se encarga de cambiar los datos de los puertos que dan la lógica al driver L298N. Como se vio, es necesario aplicar una etapa de transistores entre los puertos y los pines del puente H. Por ello, para detener el coche se encienden los pines (ponemos 0x30), para avanzar hacia adelante se coloca un 0x20 en ambos, para avanzar hacia atrás ponemos un 0x10. Para las vueltas se mantiene un motor activo hacia adelante y el otro se mantiene detenido. Para definir el ciclo de trabajo de las PWM al avanzar hacia adelante o hacia atrás, planteamos las ecuaciones:

$$A + 1.7B = 8000$$

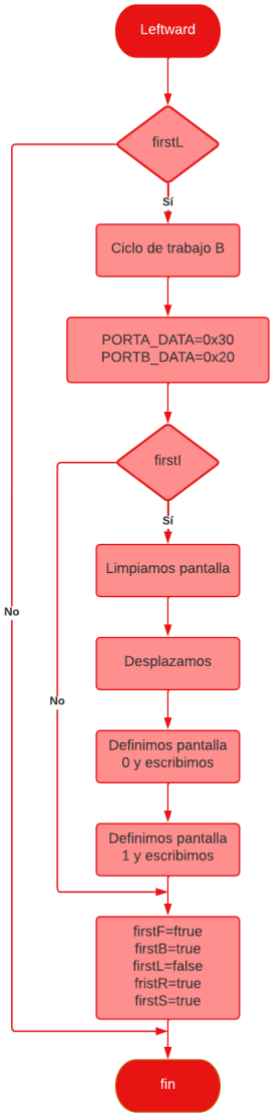
$$A + 3.3\,B = 3200$$

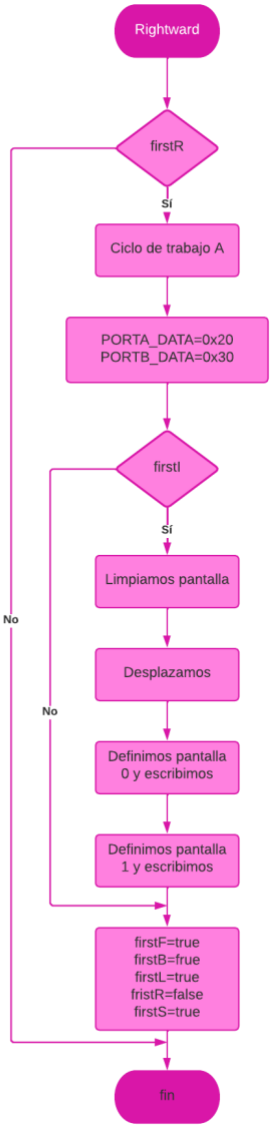
Con esto definimos un ciclo de trabajo mínimo del 50% y máximo del 80%. Por ello, las funciones de backward y fordward reciben el valor de Y. Entonces, las 5 funciones se describen como sigue:




```
void Forward(float Y){
        TIMER0_TBMATCHR_R = A+B*Y;                          //Ciclo de trabajo dependiendo de la inclinación del Joystick
        TIMER0_TAMATCHR_R = A+B*Y;                          //Ciclo de trabajo dependiendo de la inclinación del Joystick
    if(firstF){                                             //Condición de estado del movimiento hacia adelante
        GPIO_PORTA_AHB_DATA_R = 0x20;                       //Asigna 1 en A5 y un 0 en A4
        GPIO_PORTB_AHB_DATA_R = 0x20;                       //Asigna 1 en B5 y un 0 en B4
        if(firstI){                                         //Condición de interrupción
            SSD1306_Clear();                                //Limpia pantalla
            SSD1306_SCROLLINGL();                           //Desplaza a la izquierda
            SSD1306_Command(SSD1306_SETPAGE0);              //Pagina 0
            SSD1306_WriteString("[_____________");          //Mensaje para pagina 0
            SSD1306_Command(SSD1306_SETPAGE1);              //pagina 1
            SSD1306_WriteString("   ]--------------");      //Mensaje para pagina 1
        }
        firstF=false;                                       //Cambia estado de fordward
        firstB=true;                                        //resetea los otros estados
        firstL=true;
        firstR=true;
        firstS=true;
    }
}
void Backward(float Y){
    TIMER0_TBMATCHR_R = A+B*(3.3-Y);                        //Ciclo de trabajo dependiendo de la inclinación del Joystick
    TIMER0_TAMATCHR_R = A+B*(3.3-Y);                        //Ciclo de trabajo dependiendo de la inclinación del Joystick
    if(firstB){                                             //Conducuón de backward
        GPIO_PORTA_AHB_DATA_R = 0x10;                       //Asigna 1 a A4 y 0 a B4
        GPIO_PORTB_AHB_DATA_R = 0x10;                       //Asigna 1 a B5 y 0 a B4
        if(firstI){                                         //Condición de interrupción
            SSD1306_Clear();                                //Limpia pantalla
            SSD1306_SCROLLINGR();                           //Desplaza a la derecha
            SSD1306_Command(SSD1306_SETPAGE0);              //Define página 0
            SSD1306_WriteString("_____________(");          //Cadena para pagina 0
            SSD1306_Command(SSD1306_SETPAGE1);              //Define pagina 1
            SSD1306_WriteString("   --------------)");      //Cadena pagina 1
        }
        firstB=false;                                       //Cambia estado backward
        firstF=true;                                        //Resetea los otros edos
        firstL=true;
        firstR=true;
        firstS=true;
    }
}
```

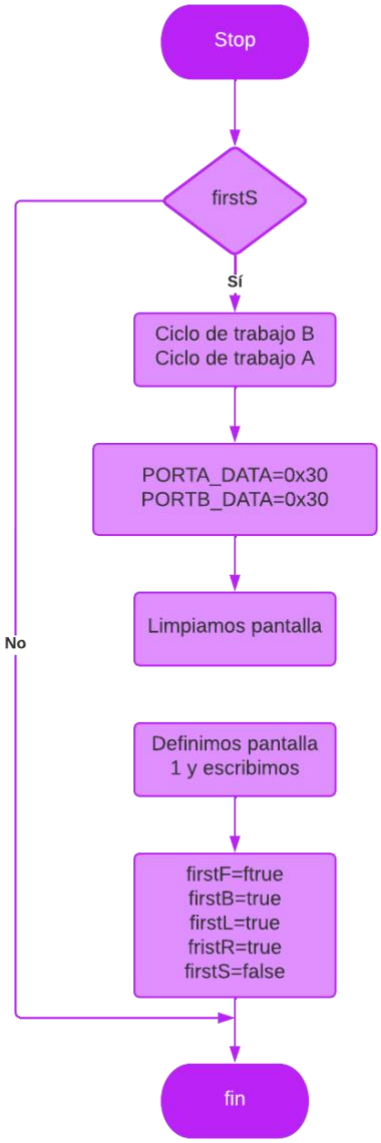
Ilustración 21 Funciones fordward y backward






```
void Leftward(void){
    if(firstL){
        GPIO_PORTB_AHB_DATA_R = 0x20;                                   //Indica desplazamiento hacie adelante de la rueda asociada a B
        GPIO_PORTA_AHB_DATA_R = 0x30;                                   //Impide el movimiento de motor A
        TIMER0_TBMATCHR_R = 6400;                                       //Ciclo de trabajo del 60% asociado al motor
        if(firstI){
            SSD1306_Clear();                                            //Limpia pantalla
            SSD1306_SCROLLINGL();                                       //Desplaza hacia la izquierda
            SSD1306_Command(SSD1306_SETPAGE0);                          //Define pagina 0
            SSD1306_WriteString("*");                                   //Envia cadena a pagina 0
            SSD1306_Command(SSD1306_SETPAGE1);                          //Define pagina 1
            SSD1306_WriteString("                    |__________");     //Cadena pagina 1
        }
        firstF=true;                                                    //Reset a los demas edos
        firstR=true;
        firstB=true;
        firstS=true;
        firstL=false;                                                   //Cambia edo leftward
    }

}
void Rightward(void){
    if(firstR){                                                         //Valida edo right
        GPIO_PORTA_AHB_DATA_R = 0x20;                                   //Movimiendo de A hacia adelante
        GPIO_PORTB_AHB_DATA_R = 0x30;                                   //Impide movimiento de B
        TIMER0_TAMATCHR_R = 6400;                                       //Ciclo de trabajo del 60% asociado al motor
        if(firstI){                                                     //Valida eso de interrupción
            SSD1306_Clear();                                            //Limpia pantalla
            SSD1306_SCROLLINGL();                                       //Desplazamiento a la izquierda
            SSD1306_Command(SSD1306_SETPAGE0);                          //Establece pagina 0
            SSD1306_WriteString("|__________");                         //Cadena a pagina 0
            SSD1306_Command(SSD1306_SETPAGE1);                          //Define pagina 1
            SSD1306_WriteString("                    ^");               //Cadena a pagina 1
        }
        firstF=true;                                                    //reset a todos los demás edos
        firstS=true;
        firstB=true;
        firstL=true;
        firstR=false;                                                   //edo de right cambiado
    }
}
```

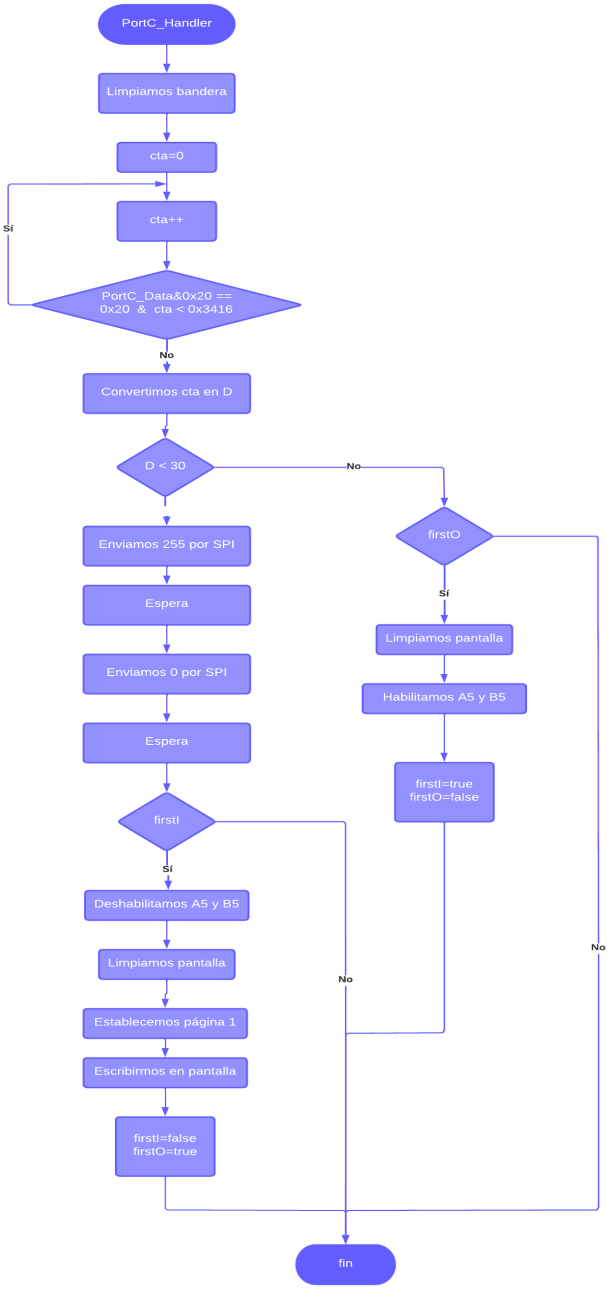
Ilustración 22 Funciones rightward y leftward




```
void Stop(void){
    if(firstS){
        GPIO_PORTA_AHB_DATA_R = 0x30;          //evalua edo de stop
        GPIO_PORTB_AHB_DATA_R = 0x30;          //Pone 1 en A4 y A5
        TIMER0_TBMATCHR_R = 16000;             //Pone 1 en B4 y B5
        TIMER0_TAMATCHR_R = 16000;             //Ciclo de trabajo A
        SSD1306_Clear();                       //Ciclo de trabajo B
        SSD1306_Command(SSD1306_SETPAGE1);     //Limpia pantalla
        SSD1306_WriteString("ALTO");           //Define pagina 1
        firstF=true;                           //Envia dato
        firstB=true;                           //reset a los demás edos
        firstL=true;
        firstR=true;
        firstS=false;                          //cambia edo del stop
    }
    ind1=1;                                    //Cambia edo del indicador 2
    ind2=1;
}
```

Ilustración 23 Función stop

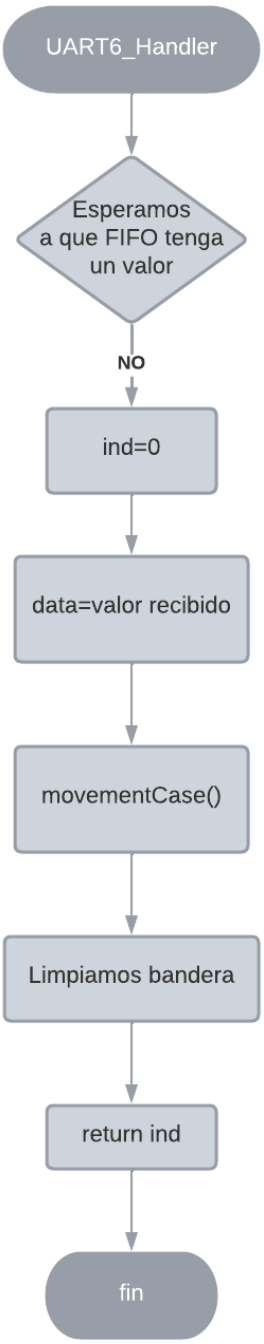
El motivo por el que usan variables booleanas es porque, de no hacerlo, la pantalla OLED constantemente estaría actualizando su pixelado a pesar de seguir en la misma dirección, ya que el proceso que sigue el limpiar y después escribir. Las variables fungen en papel de evaluar si es la primera vez que se entra en la función: si es el caso realiza todo el proceso de forma normal, de no serlo, se salta. Así se evita repetir constantemente pasos como dar valor al puerto, definir los ciclos de trabajo o actualizar las pantallas.



Para la interrupción en el puerto C, tenemos que entra a esta cuando hay un flanco de subida en el puerto C pin 5. Dentro la rutina primero limpiamos la bandera (para que al terminar salga de la rutina), después entra en un ciclo que aumenta una cuenta con valor inicial de 0 (esta cuenta se mantendrá aumentando mientras no haya un cambio en el puerto C5 o mientras la cuenta sea menor a 0x3416), posteriormente por medio de una formula empírica obtenemos el equivalente de la cuenta a distancia. Con el valor de distancia, evaluamos si es menor o no a 30; en caso de serlo, deshabilita pines 5 de puertos A y B para impedir que el equipo se desplace hacia adelante, envía datos al potenciometro por protocolo SPI (para que tenga en sus terminales un voltaje de 5 V y 0 V por cierto tiempo) y limpia la pantalla, así como a mandar una alerta de cuidado al display. En caso de que la distancia no sea menor, habilita los puertos, limpia la pantalla y envía un 0 al potenciómetro digital.

```
void PortC_Handler(void){
    GPIO_PORTC_AHB_ICR_R = 0x20;                    //Limpia la bandera
    cta=0;                                          //Inicializa la cuenta a 0
        do{
            cta++;                                  //Incrementa la cuenta mientras cumpla la condición
        }while(((GPIO_PORTC_AHB_DATA_R & 0x20)==0x20) & (cta  < 0x00003416));   //mientras el eco no haya regresado o mientras la cuenta sea menor a una distancia equivalente a 400 cm
        D=0.03*cta;                                 //Convierte la cuenta en su equivalente en distancia

        if(D<30){                                   //Condición para deter el movimiento
            pot_setVal(255);                        //enviamos el dato al potenciometro digital pasar voltaje
            for (i = 0; i < 20000; i++);            //Tiempo en alto
            pot_setVal(0);                          //Enviamos dato para que en potenciometro no deje pasar nada
            for (i = 0; i < 200000; i++);           //tiempo en bajo
            if(firstI){                             //Evalua si es la primera vez que entramos al ciclo de la interrupción
                GPIO_PORTA_AHB_DEN_R &= ~0x2F;      //Deshabilita A5
                GPIO_PORTB_AHB_DEN_R &= ~0x23;      //Deshabilita B5
                SSD1306_Clear();                    //Limpiamos pantalla
                SSD1306_Command(SSD1306_SETPAGE1);  //Establecemos pagina
                SSD1306_WriteString("CUIDADO");     //Enviamos texto
                firstI=false;                       //Decimos que no es la primera vez que está en la interrupción
                firstO=true;                        //Reiniciamos el valor del indicar de estar fuera de la interrupción
                //ind1=0;
            }
        }
        else{
            if(firstO){                             //Evalua en el indicador de estar fuera
                pot_setVal(0);                      //Envia dato 0 al potenciometro
                SSD1306_Clear();                    //Limpia la pantalla
                GPIO_PORTA_AHB_DEN_R |=0x30;        //Habilita A5
                GPIO_PORTB_AHB_DEN_R |=0x30;        //Habilita B5
                firstI=true;                        //Resetea indicador inside
                firstO=false;                       //Cambia indicador out
                //ind1=1;
            }
        }
}
```

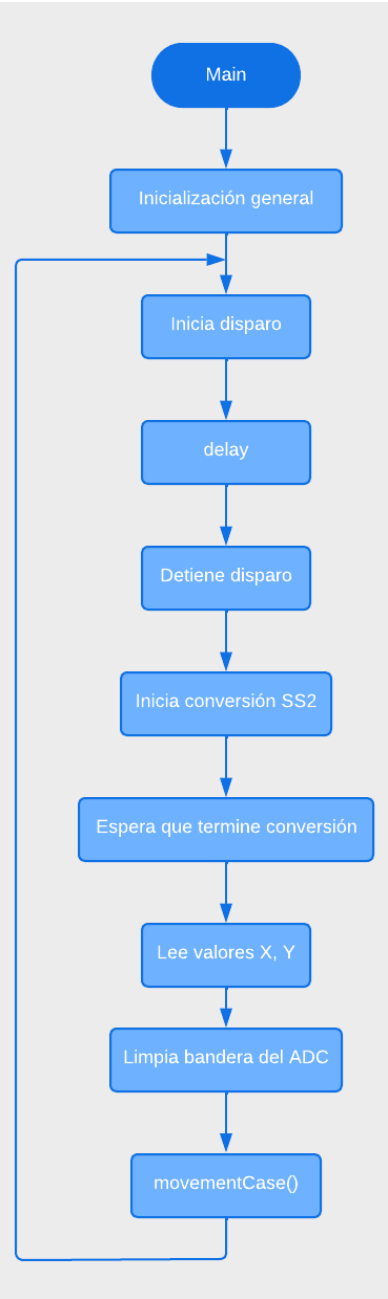
Para el UART, definimos la función **UART6_Handler**; la interrupción empieza cuando la FIFO está en un 1/8 llena de su capacidad total. La rutina empieza esperando a que llegue un valor en la FIFO. Posteriormente, cambia el valor del indicador a 0 (esto hace que se bloqueen las instrucciones por joystick), luego guarda en la variable data el valor que tiene la FIFO, y este lo manda a la función **movementCase** junto a un dato definido para un ciclo de trabajo adecuado que desplace al equipo hacia adelante o hacia atrás sea cual sea el caso. Luego limpia la bandera y finalmente retorna en dato de ind2.




```
int UART6_Handler(void){
        while((UART6_FR_R&UART_FR_RXFE) != 0);          //Se espera a que el FIFO RX tenga un valor
        ind2=0;                                         //Cambia el estado del indicador
        data = UART6_DR_R&0xFF;                         //guarda en valor de la fifo en una variable
        movementCase(data, 1.9);                        //usa el valor de esa variable para enivarlo a la función de movimiento
        UART6_ICR_R = 0x00000030;                       // Limpia bandera de interrupción
        return ind2;
}
```

Ilustración 24 Función de interrupción UART6_Handler

Finalmente, el main inicializa los puertos y periféricos que serán usados a lo largo de todo el programa. Posteriormente en un bucle infinito se encarga de disparar el trigger del sensor ultrasónico, espera 10 μS y después apaga el disparo, posteriormente se encarga de obtener los valores del joystick mediante el convertidos analógico digital (en poleo) y envía estos valores a la función **movementeCase** y **ADCSumary.**




```
int main(void) {
    General_Init();
    while(1){
    GPIO_PORTC_AHB_DATA_R |= 0X10;                  //Enciende el puerto para disparar la señal de trigger
    for(i = 0; i< 11; i++);                         //  Retardo > 10us
    GPIO_PORTC_AHB_DATA_R &= ~0X10;                 //Apaga el puerto para deterner el disparo,
    ADC0_PSSI_R = 0x0004;                           //Inicia conversión del SS2
    while((ADC0_RIS_R&0x04)==0);                    // Espera a que SS2 termine conversión (polling)
    Y = (3.3/4096)*(ADC0_SSFIFO2_R&0xFFF);          // se lee el primer resultado asociado a PE5
    X = (3.3/4096)*(ADC0_SSFIFO2_R&0xFFF);          // se lee el segundo resultado asociado a PE4
    ADC0_ISC_R = 0x0004;                            //Limpia la bandera RIS del ADC0
    movementCase(ADCSumary(X,Y)*ind,Y);             //Función de determinará hacia donde desplazar
    }
}
```

Ilustración 25 main

Para la aplicación, la desarrollamos usando la plataforma App Inventor del MIT. Esta básicamente consiste en dos botones para conectarse o desconectarse al módulo Bluetooth, otros 5 para desplazar el coche (con 4 direcciones y un alto) y otros dos más, uno para salir de la aplicación y otro para dar las indicaciones por voz. Usamos el reconocimiento de voz nativo del celular (que es provisto por Google) y la aplicación se encarga de comparar si el texto reconocido coincide con una serie de palabras ya definidas. Dependiendo de con cual coincide manda dos veces un mismo valor (esto porque la interrupción por FIFO entra después del segundo dato).

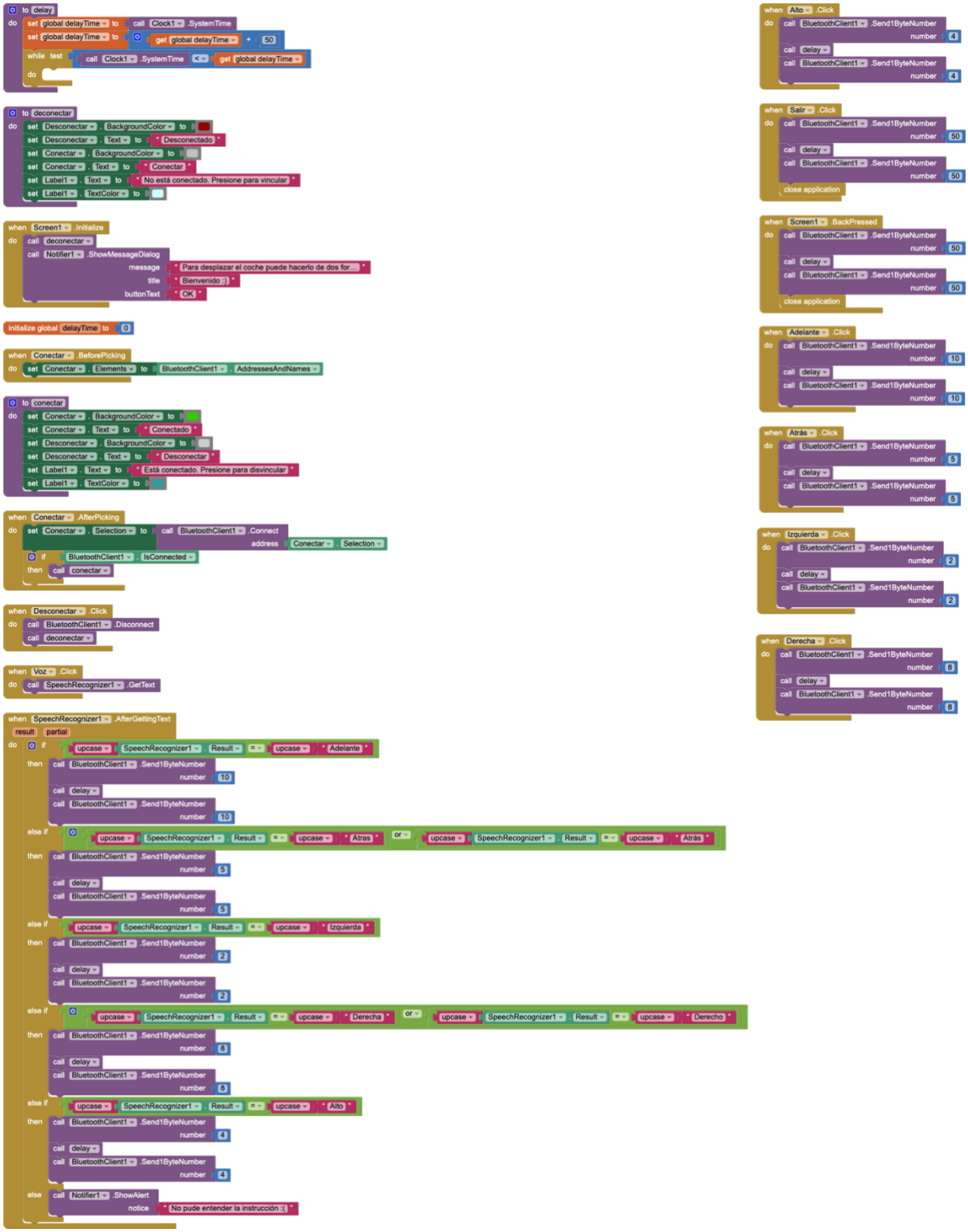


Ilustración 26 Bloques de la app

Y la aplicación misma se muestra a continuación:

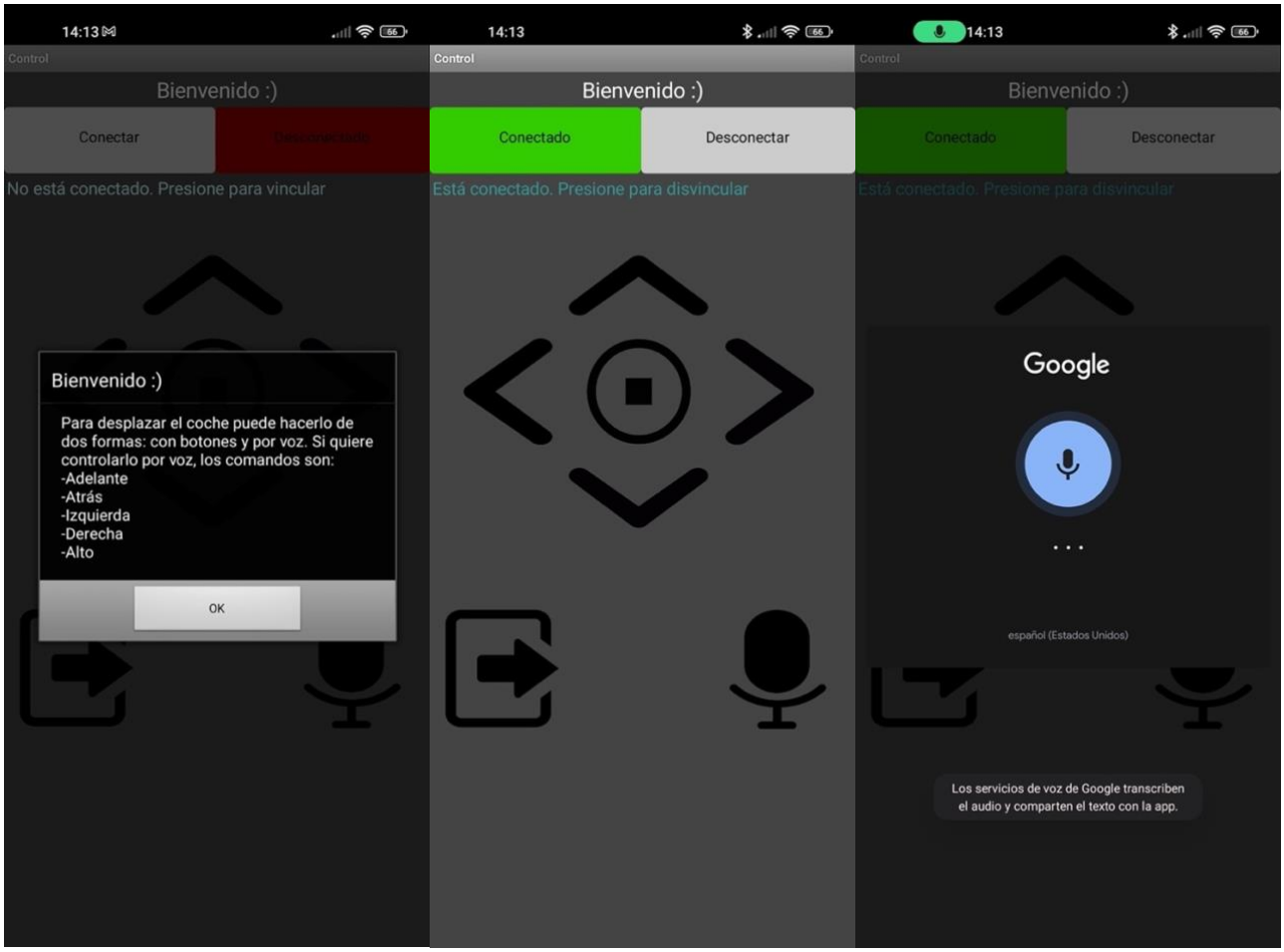


Ilustración 27 Capturas de pantalla de la aplicación

Para el cálculo de los retardos, usamos la herramienta disassembly para ver cuántas instrucciones de ensamblador corresponde al ciclo for en C. Al hacerlo notamos que se trata de 12 ciclos de reloj para las instrucciones no cíclicas y 14 ciclos más que corresponden al ciclo perse. Cada ciclo consume un tiempo de 62.5 nS. Entonces, formamos la ecuación

$$12T_{clk} + 14T_{clk}X = t$$

donde $T_{clk} = 62.5\ nS$, X es el número que irá en el ciclo for y t el tiempo que buscamos obtener.

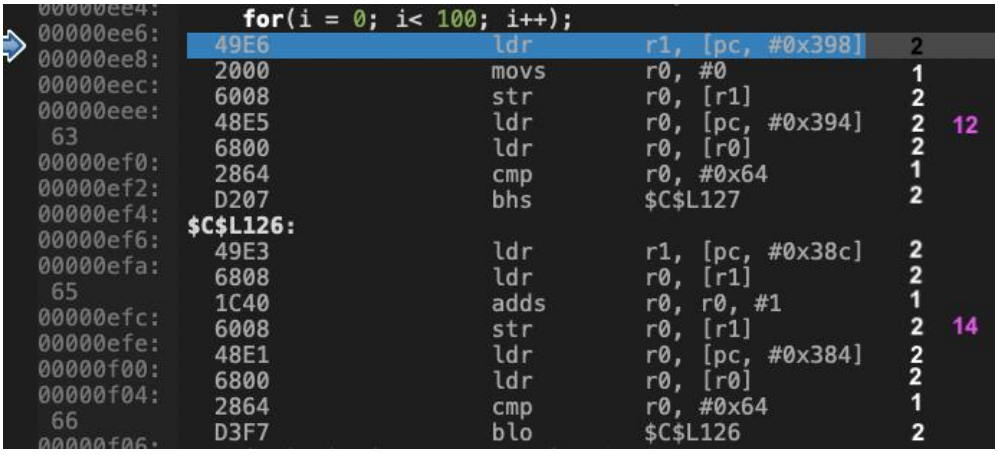


Ilustración 28 Ciclos consumidos en un ciclo for

V. Resultados y – Conclusiones

Finalmente, la versión final del coche se muestra a continuación. Observamos cómo están distribuidos los diferentes elementos, a lo largo del cuerpo, tanto por arriba como por abajo del chasis.

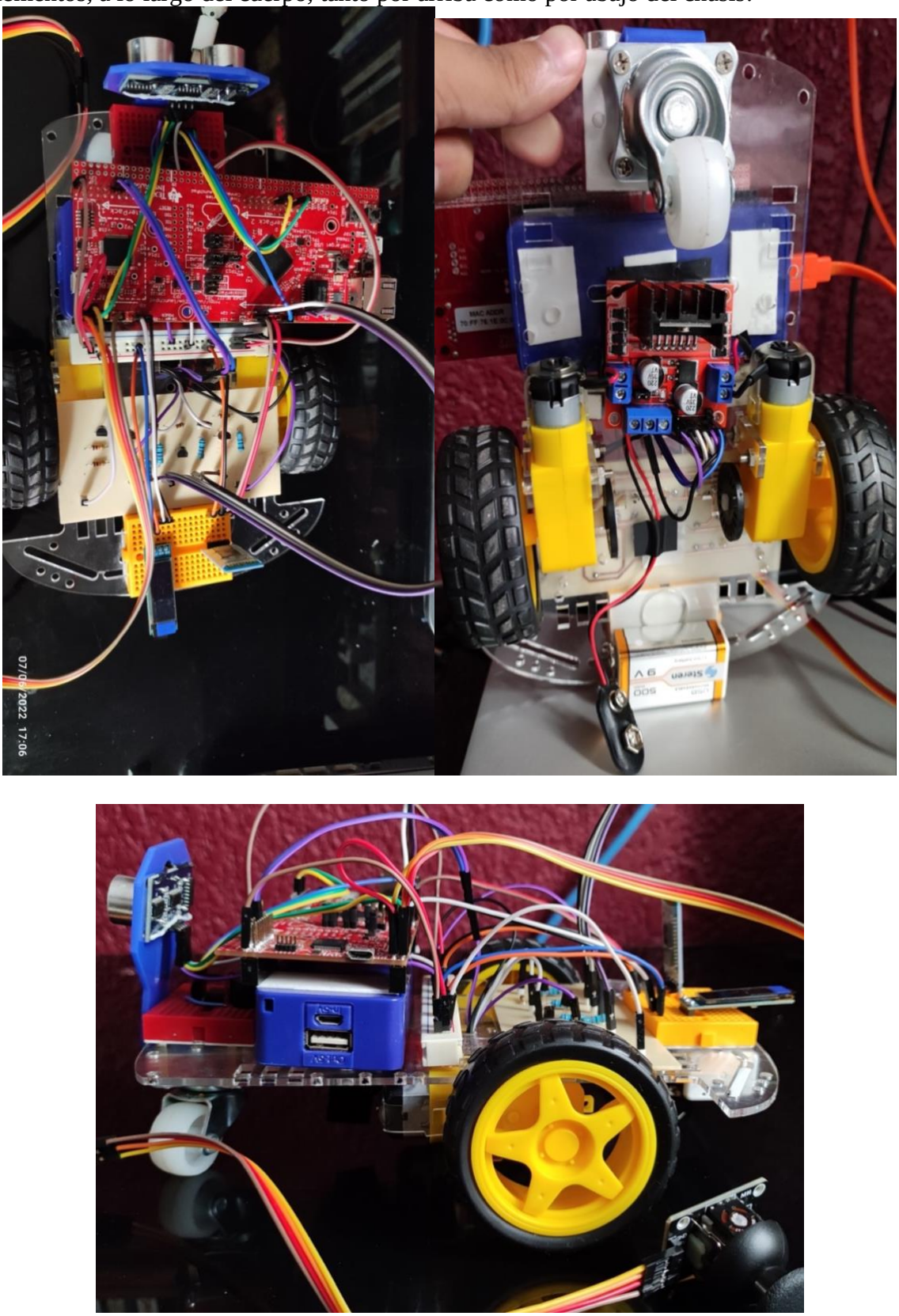


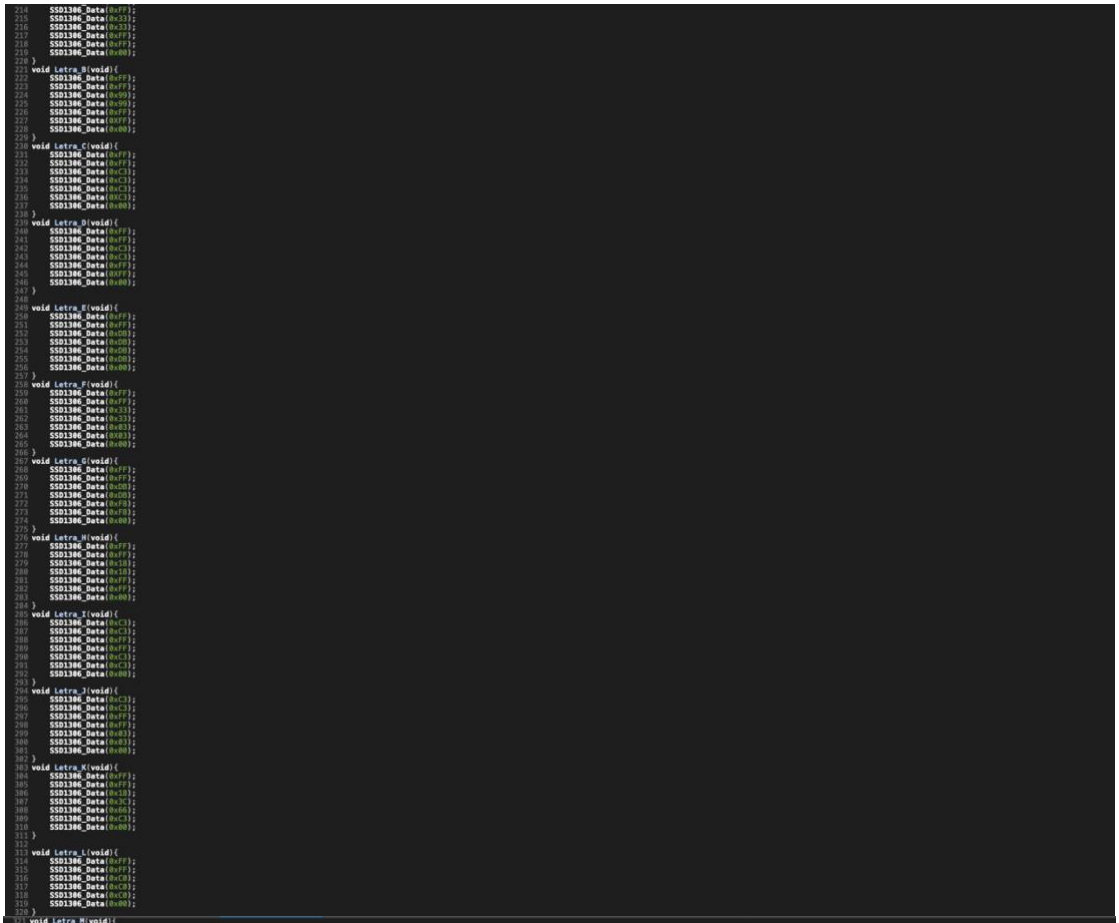
Ilustración 29 Versión final del proyecto

El código en su totalidad se muestra a continuación

```
#include <stdint.h>
#include <stdbool.h>
#include <math.h>
#include "tm4c1294ncpdt.h"
#include "SSD1306.h"
float X, Y;
uint32_t cta, D=0;
int data, A=11500, B=-3800, ind=1;
bool firstF=true, firstB=true, firstL=true, firstR=true, firstS=true, firstI=true, firstD=true;

//-------------------------------------------------------------Inicialización-------------------------------------------------------------
void ADC_Init(void);
void UART_Init(void);
void PortC_Init(void);
void PWM_Init(void);
void General_Init(void);
//-------------------------------------------------------------Interrupciones-------------------------------------------------------------
int UART6_Handler(void);
void PortC_Handler(void);

//-------------------------------------------------------------Re movimiento-------------------------------------------------------------
void Forward(float Y);
void Backward(float Y);
void Rightward(void);
void Leftward(void);
void Stop(void);
//-------------------------------------------------------------Varios-------------------------------------------------------------
int ADCSumary(float X, float Y);
void movementCase(int n, float Y);
//-------------------------------------------------------------SPI-------------------------------------------------------------
void SSI3_init (void);
void SSI3_sendData (uint16_t dat);
void pot_setVal (uint8_t slider);


int main(void) {
    General_Init();
    while(1){
    GPIO_PORTC_AHB_DATA_R |= 0X10;                                  //Enciende el puerto para disparar la señal de trigger
    for(i = 0; i< 50; i++);                                         //  Retardo > 10us
    GPIO_PORTC_AHB_DATA_R &= ~0X10;                                 //Apaga el puerto para detener el disparo.
    ADC0_PSSI_R = 0x0004;                                           //Inicia conversión del SS2
    while((ADC0_RIS_R&0x04)==0);                                    // Espera a que SS2 termine conversión (polling)
    Y = (3.3/4096)*(ADC0_SSFIFO2_R&0xFFF);                          // Se lee el primer resultado asociado a PE5
    X = (3.3/4096)*(ADC0_SSFIFO2_R&0xFFF);                          // Se lee el segundo resultado asociado a PE4
    ADC0_ISC_R = 0x0004;                                            //Limpia la bandera RIS del ADC0
    movementCase(ADCSumary(X,Y)*ind,Y);                             //Función de determinará hacia donde desplazar
    }
}


//-------------------------------------------------------------Función para inicializar el puerto UART6 asociado a P0 como receptor-------------------------------------------------------------
void UART_Init(void){
    //Usaremos USRx en P0 (1)
    SYSCTL_RCGCUART_R |= 0x00000040;                                //Reloj al UART6 (p. 388)
    while((SYSCTL_PRUART_R & 0x00000040)== 0);                      //Reloj listo ?
    UART6_CTL_R &= ~0x00000001;                                     //Deshabilita UART pre-configuración (p.1188)
    UART6_IBRD_R = 8;                                               //Parte entera del divisor de baudios. IBRD = int(16,000,000 / (16 * 115,200)) = int(8.681) (p.1184)
    UART6_FBRD_R = 44;                                              //Parte fraccionaria FBRD = round(0.6806 * 64) = 44 (p. 1185)
    UART6_LCRH_R = (UART_LCRH_WLEN_8|UART_LCRH_FEN);                //Registro line control: palabra de 8 bits habilitando la fifo (0x60|0x10).
    UART6_CC_R = (UART6_CC_R&~UART_CC_CS_M)+UART_CC_CS_PIOSC;       //Configuración del reloj. Fuente alterna del reloj (0x5) definido en ALTCLKCFG (p.1213)
    SYSCTL_ALTCLKCFG_R = (SYSCTL_ALTCLKCFG_R&~SYSCTL_ALTCLKCFG_ALTCLK_M)+SYSCTL_ALTCLKCFG_ALTCLK_PIOSC;    //Fuente PIOSC, oscilador interno 16 MHz

    UART6_IFLS_R &= ~0x0000003F;                                    //Selector de nivel FIFO: Limpia interrupcion de FIFO Rx y Tx poniendo 0 a lo largo del registro
    UART6_IFLS_R += (0x00000000|0x00000000);                        // Habilita interrupción para RX, TX FIFO >= 1/8
    UART6_ICR_R = 0x10;                                             //Limpia bandera de interrupción de recepción (p.1206)
    UART6_IM_R |= 0x00000010;                                       //Desenmascara interrupción de recepción (p.1194)

    UART6_CTL_R &= ~0x00000020;                                     //Define reloj del UART como reloj del sistema dividido en 16 (p.1188)
    UART6_CTL_R |= 0x00000001;                                      //Habilita UART (p. 1180)
    while((SYSCTL_PRGPIO_R & SYSCTL_PRGPIO_R0) == 0);               //Espera que el reloj se estabilice
    GPIO_PORTP_DIR_R &= ~0x01;                                      //Definimos P0 como entrada
    GPIO_PORTP_DEN_R |= 0x01;                                       //Función digital habilitada (p 781)
    GPIO_PORTP_AFSEL_R |= 0x01;                                     //Habilita función alterna de P0
    GPIO_PORTP_PCTL_R = 0x00000001;                                 //Controla P0 con su función de UART6
    GPIO_PORTP_AMSEL_R &= ~0x01;                                    //Deshabilita función analógica de P0 (p. 786)

    NVIC_PRI14_R = (NVIC_PRI14_R&0x00FFFFFF)|0x00000000;            //Define mayor prioridad, numero de interrupcion 59 (4(14)+3)
    NVIC_EN1_R = 1<<(59-32);                                        //Habilita interrupción, en la casilla 27 del registro
}
//-------------------------------------------------------------Función para el ADC asociado a E4-5-------------------------------------------------------------
void ADC_Init(void){
//-------------------------------------------Para el puerto asociado al ADC-------------------------------------------
    GPIO_PORTE_AHB_DIR_R &= ~0x30;                                  // E5-4 entradas (analógica)
    GPIO_PORTE_AHB_AFSEL_R |= 0x30;                                 // Habilita función Alterna en E5, E4 (p. 770)
    GPIO_PORTE_AHB_DEN_R &= ~0x30;                                  // Deshabilita función Digital en E5, E4 (p 781)
    GPIO_PORTE_AHB_AMSEL_R |= 0x30;                                 // Habilita función Analógica de E5, E4 (p. 786)


// -------------------------------------------Para los controladores del motor-------------------------------------------
    GPIO_PORTA_AHB_DIR_R |=0x30;                                    //Define A4, A5 como salida
    GPIO_PORTA_AHB_DEN_R |=0x30;                                    //Define A4, A5 como digital

    GPIO_PORTB_AHB_DIR_R |=0x30;                                    //Define B4, B5 como salida
    GPIO_PORTB_AHB_DEN_R |=0x30;                                    //Define B4, B5 como digital
    SYSCTL_RCGCADC_R |= 0x01;                                       // Habilite reloj para ADC0(p. 396)
    while((SYSCTL_PRADC_R & 0x01) == 0);                            // Se espera a que el reloj se estabilice
    ADC0_PC_R = 0x01;                                               // Configuración del periférico (tasa de muestreo). Configura para 125Ksamp/s (p.1159) 1/(Nhs+12)Tclk
    ADC0_SSPRI_R = 0x0123;                                          // SS3 con la más alta prioridad
    ADC0_ACTSS_R &= ~0x0004;                                        // Control que activa los secuenciadores: Deshabilita SS2 antes de cambiar configuración de registros
    ADC0_EMUX_R = 0x0000;                                           // Evento que dispara el trigger: Se configura SS2 para disparar muestreo por software, PSSI sin efecto
    ADC0_SAC_R = 0x0;                                               // Se configura para no tener sobremuestreo por hardware(default)(p. 1185)
    ADC0_CTL_R = 0x0;                                               // Se configura con referencias internas (default VDDA and GNDA) (p. 1187)
    ADC0_SSOP2_R = 0x0000;                                          // Registro de operación: Se configura para salvar los resultados del ADC en FIFO (default)(p. 1134)
    ADC0_SSTSH2_R = 0x000;                                          // Se configure el ADC para un periodo de 4. TSH=0x0 -> Nhs=4 (default) (p. 1139)
    ADC0_SSMUX2_R = 0x0089;                                         // Se configura entradas 1 muestra=AIN9 2 muestra=AIN8 (p.1128)
    ADC0_SSEMUX2_R &= ~0x0011;                                      // La cuarta muestra es relacionada con AIN(15:0) (p.1137)
    ADC0_SSCTL2_R = 0x0060;                                         // Define la segunda muestra como la última de la secuencia e interrupción al final de segundo muestreo
    ADC0_IM_R &= ~0x0004;                                           // Deshabilita interrupción SS2 porque el estado del 2do secuenciador no afecta el SS2 interrupt status
    ADC0_ACTSS_R |= 0x0004;                                         // Habilita SS2 (p. 1076)

    SYSCTL_PLLFREQ0_R |= 0x00000000;                                // encender PLL. PLL oscilará basado en los valores de los registros PLLFREQ0-1
    while((SYSCTL_PLLSTAT_R&0x01)==0);                              // espera a que el PLL fije su frecuencia
    SYSCTL_PLLFREQ0_R &= ~0x00000000;                               // apagar PLL
    ADC0_ISC_R = 0x0004;                                            // Limpia INR2 y ADCRIS
}
//-------------------------------------------sensor ultrasónico y su inicialización asociado a PC4-5-------------------------------------------
void PortC_Init(void){
    GPIO_PORTC_AHB_DIR_R = 0X10;                                    //Definimos C4 como salida y C5 como entrada
    GPIO_PORTC_AHB_DEN_R |= 0X30;                                   //función digital en C4, C5
    GPIO_PORTC_AHB_DATA_R &= ~0X10;                                 //

    GPIO_PORTC_AHB_IS_R &= ~0x20;                                   //C5 sensible por flanco
    GPIO_PORTC_AHB_IBE_R &= ~0x20;                                  //C5 sensible a un solo flanco
    GPIO_PORTC_AHB_IEV_R |= 0x20;                                   //C5 sensible por flanco de subida
    GPIO_PORTC_AHB_ICR_R = 0x20;                                    //Limpiamos bandera C5
    GPIO_PORTC_AHB_IM_R |= 0x20;                                    //Desenmascaramos C5

    NVIC_EN0_R = 0x00000004;                                        //Numero de interrupción 2, en EN0 casilla 2
    NVIC_PRI0_R = (NVIC_PRI0_R&0xFF00FFFF)|0x00200000;              //Esta en 4n+2 [23:21]. Prioridad 1
}
//-------------------------------------------Señales PWM en TIMER0-------------------------------------------
void PWM_Init(void){
            SYSCTL_RCGCTIMER_R |= 0x01;                             //Habilita timer 0
            GPIO_PORTD_AHB_DIR_R |= 0x03;                           //Define D0, D1 como salidas
            GPIO_PORTD_AHB_DEN_R |= 0x03;                           //Habilita función digital en D0, D1
            GPIO_PORTD_AHB_DATA_R = 0x00;
            GPIO_PORTD_AHB_AFSEL_R = 0x03;                          //Función alterna en D0, D1 (p. 770)
            GPIO_PORTD_AHB_PCTL_R = 0x00000033;                     //MUX0, MUX1 controla D0, D1 respectivamente (p 787)

            TIMER0_CTL_R =0x00000000;                               //Reset en TIMER0 para configurar (p. 986)
            TIMER0_CFG_R =0x00000004;                               //Selecciona configuración de 16-bit (p. 976)
//-------------------------------------------Para subtimer A-------------------------------------------
            TIMER0_TAMR_R = 0x0000000A;                             //Habilita modo PWM/ Modo periódico de TIMER0-A (p. 977)
            TIMER0_TAILR_R = 16000;                                 //Valor de carga para TIMER0-A (p. 1004)
            TIMER0_TAMATCHR_R = 8000;                               //Valor de match para el ciclo de trabajo TIMER0-A (p.1006)
            TIMER0_TAPR_R=0X00;                                     //Sin prescala (p. 1008)
//-------------------------------------------Para subtimer B-------------------------------------------
            TIMER0_TBMR_R = 0x0000000A;                             //Habilita modo PWM/ Modo periódico de TIMER0-B (p. 982)
            TIMER0_TBILR_R = 16000;                                 //Valor de carga para TIMER0-B (p. 1005)
            TIMER0_TBMATCHR_R = 8000;                               //Valor de match para el ciclo de trabajo TIMER0-B (p.1007)
            TIMER0_TBPR_R=0X00;                                     //Sin prescala (p. 1008)

            TIMER0_CTL_R |=0X00000101;                              //Habilita Timer0 A y Timer0 B
}
//-------------------------------------------Para SPI Q[0:2]-------------------------------------------
void SSI3_init (void) {
    SYSCTL_RCGCSSI_R = SYSCTL_RCGCSSI_R3;                           // Activa reloj al SSI3
    while ((SYSCTL_PRSSI_R & SYSCTL_PRSSI_R3) == 0);                // Espera a que este listo

    GPIO_PORTQ_DIR_R = 0x07;                                        // Configura Q0 [reloj], Q1[frame signal], Q2[Bi-directional data pin 0] como salidas (p.760)
    GPIO_PORTQ_DEN_R |= 0x07;                                       // Habilita la función digital (p. 781)
    GPIO_PORTQ_AFSEL_R |= 0x07;                                     // Selecciona la función alterna de Q[0:2] (p. 770)
    GPIO_PORTQ_PCTL_R = 0x00000EEE;                                 // Configura los terminales de Q a su función de SSI3 (p 787)
    GPIO_PORTQ_AMSEL_R &= ~0x07;                                    // Deshabilita la función analógica

    SSI3_CR1_R = 0x00;                                              // QSSI Control 1: Deshabilita SSI3 y establece como maestro (p. 1247)
    SSI3_CPSR_R = 0x02;                                             // QSSI Clock prescale: prescalador (CPSDVSR=2) del reloj SSI (p. 1252)

    SSI3_CR0_R = (0x0100 | SSI_CR0_FRF_MOTO | SSI_CR0_DSS_16) & ~(SSI_CR0_SPO | SSI_CR0_SPH);  // configura para Prescsale SPI; 16bit; 4 Mbps (16MHz/(CPSDVSR=2)(1+(SCR=1))); SPH = 0 (captura de dato

    SSI3_CR1_R |= SSI_CR1_SSE;                                      // Habilita SSI3 (p. 1247)
}

void SSI3_sendData (uint16_t dat) {
    // Envia dato de 8-bit
    while ((SSI3_SR_R & SSI_SR_BSY) != 0);                          // Verifico si el bus está ocupado en transmición/recepción. Espero que está vacio. (1=busy;0=idle)
    SSI3_DR_R = dat;                                                // envio data.
}

void pot_setVal(uint8_t slider) {
    //Combine el valor del control deslizante con el código de comando de escritura.
    // Estructura del mensaje SPI: [comando (8-bits)][deslizador (8-bits)]
    SSI3_sendData(0x1100 | slider);
}


//-------------------------------------------Función de inicialización general-------------------------------------------
void General_Init(void){
    SYSCTL_RCGCGPIO_R |= (0x201F|SYSCTL_RCGCGPIO_R14);              //Habilita reloj para Puerto E, A, B, C,  D, P y Q (p.382)
```

Ilustración 30 Código del programa principal

Y el código de la librería SSD1306.h se muestra a continuación



```
    SSD1306_Data(0xFF);
    SSD1306_Data(0x33);
    SSD1306_Data(0x33);
    SSD1306_Data(0xFF);
    SSD1306_Data(0xFF);
    SSD1306_Data(0x00);
}
void Letra_B(void){
    SSD1306_Data(0xFF);
    SSD1306_Data(0xFF);
    SSD1306_Data(0x99);
    SSD1306_Data(0x99);
    SSD1306_Data(0xFF);
    SSD1306_Data(0xFF);
    SSD1306_Data(0x00);
}
void Letra_C(void){
    SSD1306_Data(0xFF);
    SSD1306_Data(0xFF);
    SSD1306_Data(0xC3);
    SSD1306_Data(0xC3);
    SSD1306_Data(0xC3);
    SSD1306_Data(0xC3);
    SSD1306_Data(0x00);
}
void Letra_D(void){
    SSD1306_Data(0xFF);
    SSD1306_Data(0xFF);
    SSD1306_Data(0xC3);
    SSD1306_Data(0xC3);
    SSD1306_Data(0xFF);
    SSD1306_Data(0xFF);
    SSD1306_Data(0x00);
}

void Letra_E(void){
    SSD1306_Data(0xFF);
    SSD1306_Data(0xFF);
    SSD1306_Data(0xDB);
    SSD1306_Data(0xDB);
    SSD1306_Data(0xDB);
    SSD1306_Data(0xDB);
    SSD1306_Data(0x00);
}
void Letra_F(void){
    SSD1306_Data(0xFF);
    SSD1306_Data(0xFF);
    SSD1306_Data(0x33);
    SSD1306_Data(0x33);
    SSD1306_Data(0x03);
    SSD1306_Data(0x03);
    SSD1306_Data(0x00);
}
void Letra_G(void){
    SSD1306_Data(0xFF);
    SSD1306_Data(0xFF);
    SSD1306_Data(0xDB);
    SSD1306_Data(0xDB);
    SSD1306_Data(0xF8);
    SSD1306_Data(0xF8);
    SSD1306_Data(0x00);
}
void Letra_H(void){
    SSD1306_Data(0xFF);
    SSD1306_Data(0xFF);
    SSD1306_Data(0x18);
    SSD1306_Data(0x18);
    SSD1306_Data(0xFF);
    SSD1306_Data(0xFF);
    SSD1306_Data(0x00);
}
void Letra_I(void){
    SSD1306_Data(0xC3);
    SSD1306_Data(0xC3);
    SSD1306_Data(0xFF);
    SSD1306_Data(0xFF);
    SSD1306_Data(0xC3);
    SSD1306_Data(0xC3);
    SSD1306_Data(0x00);
}
void Letra_J(void){
    SSD1306_Data(0xC3);
    SSD1306_Data(0xC3);
    SSD1306_Data(0xFF);
    SSD1306_Data(0xFF);
    SSD1306_Data(0x03);
    SSD1306_Data(0x03);
    SSD1306_Data(0x00);
}
void Letra_K(void){
    SSD1306_Data(0xFF);
    SSD1306_Data(0xFF);
    SSD1306_Data(0x18);
    SSD1306_Data(0x3C);
    SSD1306_Data(0x66);
    SSD1306_Data(0xC3);
    SSD1306_Data(0x00);
}

void Letra_L(void){
    SSD1306_Data(0xFF);
    SSD1306_Data(0xFF);
    SSD1306_Data(0xC0);
    SSD1306_Data(0xC0);
    SSD1306_Data(0xC0);
    SSD1306_Data(0x00);
}
void Letra_M(void){
    SSD1306_Data(0xFF);
    SSD1306_Data(0xFF);
    SSD1306_Data(0x0C);
    SSD1306_Data(0x1C);
    SSD1306_Data(0x0C);
    SSD1306_Data(0xFF);
    SSD1306_Data(0xFF);
    SSD1306_Data(0x00);
    //ñ jalla
}
void Letra_N(void){
    SSD1306_Data(0xFF);
    SSD1306_Data(0xFF);
    SSD1306_Data(0x38);
    SSD1306_Data(0x1C);
    SSD1306_Data(0xFF);
    SSD1306_Data(0xFF);
    SSD1306_Data(0x00);

}
void Letra_O(void){
    SSD1306_Data(0xFF);
    SSD1306_Data(0xFF);
    SSD1306_Data(0xC3);
    SSD1306_Data(0xC3);
    SSD1306_Data(0xFF);
    SSD1306_Data(0xFF);
    SSD1306_Data(0x00);
}
void Letra_P(void){
    SSD1306_Data(0xFF);
    SSD1306_Data(0xFF);
    SSD1306_Data(0x1B);
    SSD1306_Data(0x1B);
    SSD1306_Data(0x1F);
    SSD1306_Data(0x1F);
    SSD1306_Data(0x00);
}
void Letra_Q(void){
    SSD1306_Data(0x7F);
    SSD1306_Data(0x7F);
    SSD1306_Data(0x63);
    SSD1306_Data(0x63);
    SSD1306_Data(0xFF);
    SSD1306_Data(0xFF);
    SSD1306_Data(0x00);
}
void Letra_R(void){
    SSD1306_Data(0xFF);
    SSD1306_Data(0xFF);
    SSD1306_Data(0x1B);
    SSD1306_Data(0x7B);
    SSD1306_Data(0xF);
    SSD1306_Data(0xEF);
    SSD1306_Data(0x00);
}
void Letra_S(void){
    SSD1306_Data(0xCF);
    SSD1306_Data(0xCF);
    SSD1306_Data(0xDB);
    SSD1306_Data(0xDB);
    SSD1306_Data(0xF3);
    SSD1306_Data(0xF3);
    SSD1306_Data(0x00);
}
void Letra_T(void){
    SSD1306_Data(0x03);
    SSD1306_Data(0x03);
    SSD1306_Data(0xFF);
    SSD1306_Data(0xFF);
    SSD1306_Data(0x03);
    SSD1306_Data(0x03);
    SSD1306_Data(0x00);
}
void Letra_U(void){
    SSD1306_Data(0xFF);
    SSD1306_Data(0xFF);
    SSD1306_Data(0xC0);
    SSD1306_Data(0xC0);
    SSD1306_Data(0xFF);
    SSD1306_Data(0xFF);
    SSD1306_Data(0x00);
}
void Letra_V(void){

    SSD1306_Data(0x1F);
    SSD1306_Data(0x7F);
    SSD1306_Data(0xF0);
    SSD1306_Data(0xF0);
    SSD1306_Data(0x7F);
    SSD1306_Data(0x1F);
    SSD1306_Data(0x00);

}
void Letra_W(void){
    SSD1306_Data(0xFF);
    SSD1306_Data(0xFF);
    SSD1306_Data(0x70);
    SSD1306_Data(0x38);
    SSD1306_Data(0x70);
    SSD1306_Data(0xFF);
    SSD1306_Data(0xFF);
    SSD1306_Data(0x00);
}
```

```
void Letra_X(void){
SSD1306_Data(0xC3);
SSD1306_Data(0xE7);
SSD1306_Data(0x3C);
SSD1306_Data(0x3C);
SSD1306_Data(0xE7);
SSD1306_Data(0xC3);
SSD1306_Data(0x00);
}
void Letra_Y(void){
    SSD1306_Data(0x03);
    SSD1306_Data(0x0F);
    SSD1306_Data(0xFC);
    SSD1306_Data(0xFC);
    SSD1306_Data(0x0F);
    SSD1306_Data(0x03);
    SSD1306_Data(0x00);
}

void Letra_Z(void){
    SSD1306_Data(0xC3);
    SSD1306_Data(0xE3);
    SSD1306_Data(0xF3);
    SSD1306_Data(0xDF);
    SSD1306_Data(0xCF);
    SSD1306_Data(0xC7);
    SSD1306_Data(0xC3);
    SSD1306_Data(0x00);
}

void Espacio (void){
    SSD1306_Data(0x00);
    SSD1306_Data(0x00);
    SSD1306_Data(0x00);
    SSD1306_Data(0x00);
    SSD1306_Data(0x00);
}

void lineaAlta(void){
    SSD1306_Data(0x03);
    SSD1306_Data(0x03);
    SSD1306_Data(0x03);
    SSD1306_Data(0x03);
    SSD1306_Data(0x03);
    SSD1306_Data(0x03);
    SSD1306_Data(0x03);
},
void lineaBaja(void){
    SSD1306_Data(0xC0);
    SSD1306_Data(0xC0);
    SSD1306_Data(0xC0);
    SSD1306_Data(0xC0);
    SSD1306_Data(0xC0);
    SSD1306_Data(0xC0);
    SSD1306_Data(0xC0);
}
void FlechaF1(void){
    SSD1306_Data(0xFE);
    SSD1306_Data(0xFC);
    SSD1306_Data(0xF8);
    SSD1306_Data(0xF0);
    SSD1306_Data(0xE0);
    SSD1306_Data(0xC0);
    SSD1306_Data(0x80);
}
void FlechaF2(void){
    SSD1306_Data(0x7F);
    SSD1306_Data(0x3F);
    SSD1306_Data(0x1F);
    SSD1306_Data(0x0F);
    SSD1306_Data(0x07);
    SSD1306_Data(0x03);
    SSD1306_Data(0x01);
}
void FlechaB1(void){
    SSD1306_Data(0x00);
    SSD1306_Data(0xC0);
    SSD1306_Data(0xE0);
    SSD1306_Data(0xF0);
    SSD1306_Data(0xF8);
    SSD1306_Data(0xFC);
    SSD1306_Data(0xFE);
}
void FlechaB2(void){
    SSD1306_Data(0x01);
    SSD1306_Data(0x03);
    SSD1306_Data(0x07);
    SSD1306_Data(0x0F);
    SSD1306_Data(0x1F);
    SSD1306_Data(0x3F);
    SSD1306_Data(0x7F);
}
void barra(void){
    SSD1306_Data(0x00);
    SSD1306_Data(0x00);
    SSD1306_Data(0x00);
    SSD1306_Data(0x00);
    SSD1306_Data(0x00);
    SSD1306_Data(0xFF);
    SSD1306_Data(0xFF);
}
void FlechaV1(void){
    SSD1306_Data(0x01);
    SSD1306_Data(0x03);
    SSD1306_Data(0x07);
    SSD1306_Data(0x0F);
    SSD1306_Data(0x07);
    SSD1306_Data(0x03);
    SSD1306_Data(0x01);
}
void SSD1306_Clear(void){
  int c;
  SSD1306_Command(SSD1306_SETPAGE0);
for(c=0;c<128;c++){
    SSD1306_Data(0x00);
}
SSD1306_Command(SSD1306_SETPAGE1);
for(c=0;c<128;c++){
    SSD1306_Data(0x00);
}
SSD1306_Command(SSD1306_SETPAGE2);
for(c=0;c<128;c++){
  SSD1306_Data(0x00);
}
SSD1306_Command(SSD1306_SETPAGE3);
for(c=0;c<128;c++){
  SSD1306_Data(0x00);}
SSD1306_Command(SSD1306_SETPAGE0);
}


void SSD1306_SCROLLINGL(void){

SSD1306_Command(SSD1306_SCROLLINGL_);
SSD1306_Command(0x00);
SSD1306_Command(0x00);
SSD1306_Command(0x00);
SSD1306_Command(0x03);
SSD1306_Command(0x00);
SSD1306_Command(0xFF);
SSD1306_Command(0x2F);


}

void SSD1306_SCROLLINGR(void){
SSD1306_Command(SSD1306_SCROLLINGR_);
SSD1306_Command(0x00);
SSD1306_Command(0x00);
SSD1306_Command(0x00);
SSD1306_Command(0x03);
SSD1306_Command(0x00);
SSD1306_Command(0xFF);
SSD1306_Command(0x2F);
}

void SSD1306_WriteChar(char C){

    switch (C){
        case 'A':
            Letra_A();
            break;
        case 'B':
            Letra_B();
            break;
        case 'C':
            Letra_C();
            break;
        case 'D':
            Letra_D();
            break;
        case 'E':
            Letra_E();
            break;
        case 'F':
            Letra_F();
            break;
        case 'G':
            Letra_G();
            break;
        case 'H':
            Letra_H();
            break;
        case 'I':
            Letra_I();
            break;
        case 'J':
            Letra_J();
            break;
        case 'K':
            Letra_K();
            break;
        case 'L':
            Letra_L();
            break;
        case 'M':
            Letra_M();
            break;
        case 'N':
            Letra_N();
            break;
        case 'O':
            Letra_O();
            break;
        case 'P':
            Letra_P();
            break;
        case 'Q':
            Letra_Q();
            break;
        case 'R':
            Letra_R();
            break;
        case 'S':
```

```
            Letra_S();
            break;
        case 'T':
            Letra_T();
            break;
        case 'U':
            Letra_U();
            break;
        case 'V':
            Letra_V();
            break;
        case 'X':
            Letra_X();
            break;
        case 'Y':
            Letra_Y();
            break;
        case 'Z':
            Letra_Z();
            break;
        case '-':
            lineaAlta();
            break;
        case ')':
            flechaF2();
            break;
        case '(':
            flechaF1();
            break;
        case '_':
            lineaBaja();
            break;
        case '[':
            flechaB1();
            break;
        case ']':
            flechaB2();
            break;
        case '|':
            barra();
            break;
        case '^':
            flechaV1();
            break;
        case '*':
            flechaV2();
            break;
        case '+':
            SS();
            break;
        case ' ':
            Espacio();
            break;
    }
}


void SSD1306_WriteString(unsigned char *cadena)
{
    unsigned int i = 0;
    unsigned int l = 0;

    l = cadena[i];
    while ((l) != 0)
    {
SSD1306_WriteChar(l);
        i++;
        l = cadena[i];
    }
}
/************************************************************
LIESE          LIESE          LIESE          LIESE          LIESE
************************************************************/

#endif /* SSD1306_H_ */
```

Ilustración 31 Código SSD1306.h

Y la simbología que se agregó, vistos en la pantalla, serían los siguientes.

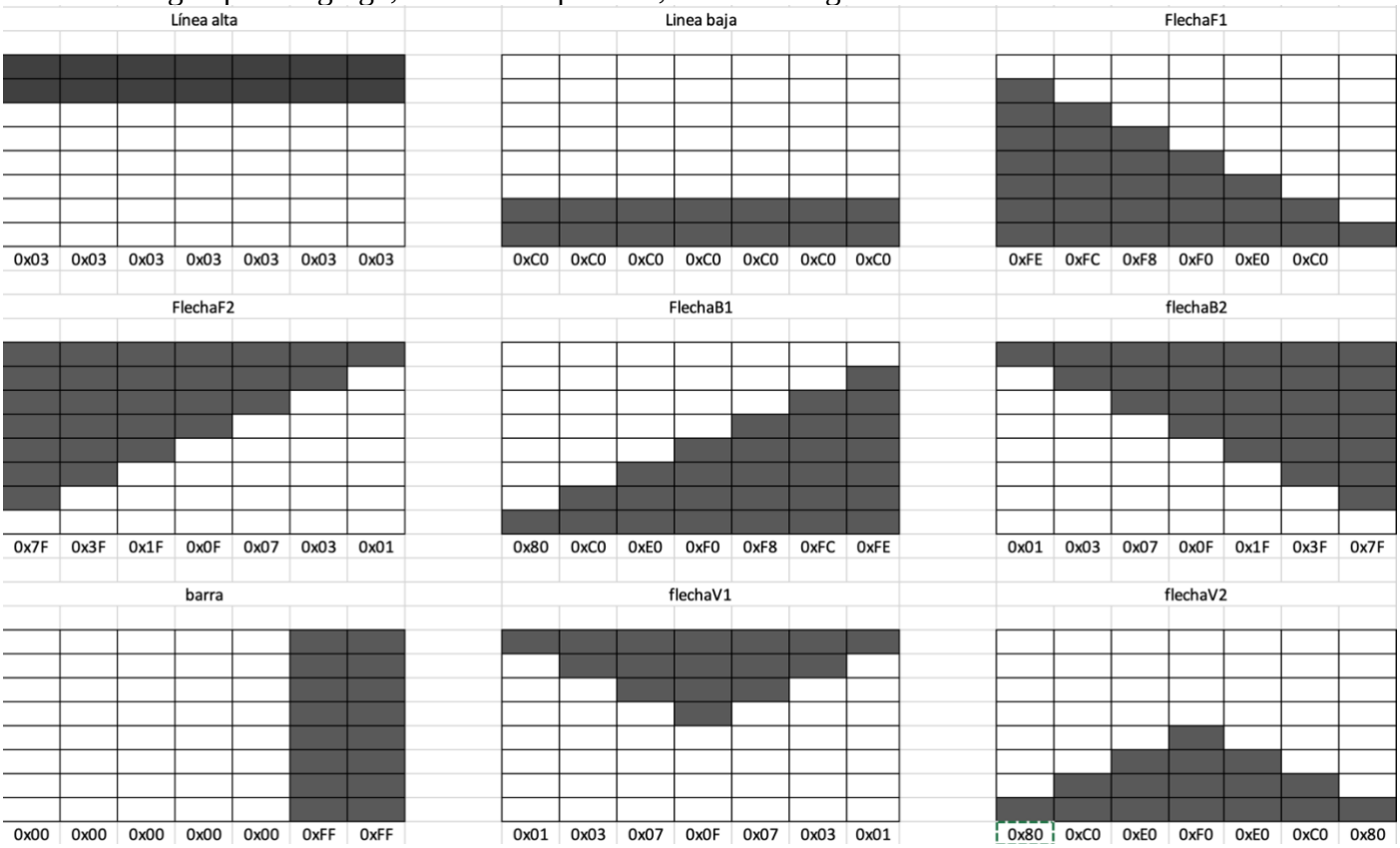


Ilustración 32 Simbología nueva para I2C

Conclusión

Por medio del desarrollo comprendí el poder de instrumentación del microcontrolador empleado y las posibilidades que da para dar solución a problemas en diversas áreas. La particularidad de modificar cada registro de forma individual (sin recurrir a bibliotecas) me brindó la capacidad de entender cómo funciona la máquina en su total y dejar de ver a estos dispositivos como cajas negras. Esto me brinda grandes herramientas de diseño, ya que el entender cómo funciona desde cero me da la posibilidad de manipularlo todo. Por otro lado, el desarrollo del proyecto mediante una metodología específica (como el enfoque Ulrich) me brindó una herramienta vital al plantear proyectos para seguir una serie ordenada de pasos que me amplían el panorama del proyecto mismo.

VI. Referencias

1 INEGI. Estadística a propósito del día internacional de las personas con discapacidad (datos nacionales). (2021, diciembre 3). Recuperado 4 de junio de 2022, de https://www.inegi.org.mx/contenidos/saladeprensa/aproposito/2021/EAP_PersDiscap21.pdf

2 La discapacidad en México. Una situación que nos compete a todos. (2021, septiembre 27). Gaceta UNAM. https://www.gaceta.unam.mx/la-discapacidad-en-mexico-una-situacion-que-nos-compete-a-todos/

3 Clasificación de Tipo de Discapacidad. (s. f.). 54.

4 Población. Discapacidad. (s. f.). Recuperado 4 de junio de 2022, de https://cuentame.inegi.org.mx/poblacion/discapacidad.aspx

5 Hercules Lite EX - Silla de ruedas eléctrica actualizada para adultos, aprobada por aerolíneas, ligera, plegable, para adultos, personas mayores: Amazon.com.mx: Salud y Cuidado Personal. (s. f.). Recuperado 5 de junio de 2022, https://www.amazon.com.mx/Hercules-Lite-EX-eléctrica-aerolíneas/dp/B092KRLP9M/ref=asc_df_B092KRLP9M/?tag=gledskshopmx-20&linkCode=df0&hvadid=514848285060&hvpos=&hvnetw=g&hvrand=10607578214585673784&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=1010043&hvtargid=pla-1324651824937&th=1

6 Silla de ruedas plegable eléctrica portátil de movilidad personal 2019 – pesa sólo 26,3 kg con batería – soporta 400 libras (plata). : Amazon.com.mx: Salud y Cuidado Personal. (s. f.). Recuperado 5 de junio de 2022, https://www.amazon.com.mx/eléctrica-plegable-portátil-compacto-Plateado/dp/B07NKXX8LF/ref=asc_df_B07NKXX8LF/?tag=gledskshopmx-20&linkCode=df0&hvadid=439957921412&hvpos=&hvnetw=g&hvrand=15584600922067333176&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=1010043&hvtargid=pla-669889742066&th=1

7 Motorreductor 2 ejes 1:48 amarillo 3-12v—ElectroCrea. (s. f.). Recuperado 5 de junio de 2022, de https://electrocrea.com/products/motorreductor-amarillo-para-llanta-65x28-mm

8 Módulo KY-023 Sensor JoyStick. (s. f.). UNIT Electronics. Recuperado 5 de junio de 2022, de https://uelectronics.com/producto/modulo-ky-023-sensor-joystick/

9 Modulo Bluetooth HC-06. (s. f.). UNIT Electronics. Recuperado 5 de junio de 2022, de https://uelectronics.com/producto/modulo-bluetooth-hc-06/

10 Sensor Ultrasónico HC-SR04—UNIT Electronics -. (s. f.). UNIT Electronics. Recuperado 5 de junio de 2022, de https://uelectronics.com/producto/sensor-ultrasonico-hc-sr04/

11 Display Pantalla Oled Blanco 128x32 0.91. (s. f.). UNIT Electronics. Recuperado 5 de junio de 2022, de https://uelectronics.com/producto/display-oled-blanco-128x32-0-91-i2c-ssd1306/

12 Buzzer Zumbador 5V Activo—30 mA. (s. f.). UNIT Electronics. Recuperado 5 de junio de 2022, de https://uelectronics.com/producto/buzzer-5v-activo/

13 AG │ Detalles MCP 41010-I/P. (s. f.). Recuperado 5 de junio de 2022, de https://www.agelectronica.com/detalle.php?p=MCP 41010-I/P

14 L298N Modulo Driver Motor A Pasos. (s. f.). UNIT Electronics. Recuperado 7 de junio de 2022, de https://uelectronics.com/producto/l298n-modulo-driver-motor-a-pasos/