



PROYECTO FINAL



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 ruedas 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 silla de ruedas.

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: Identificar 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 encuentran distribuido de la siguiente forma

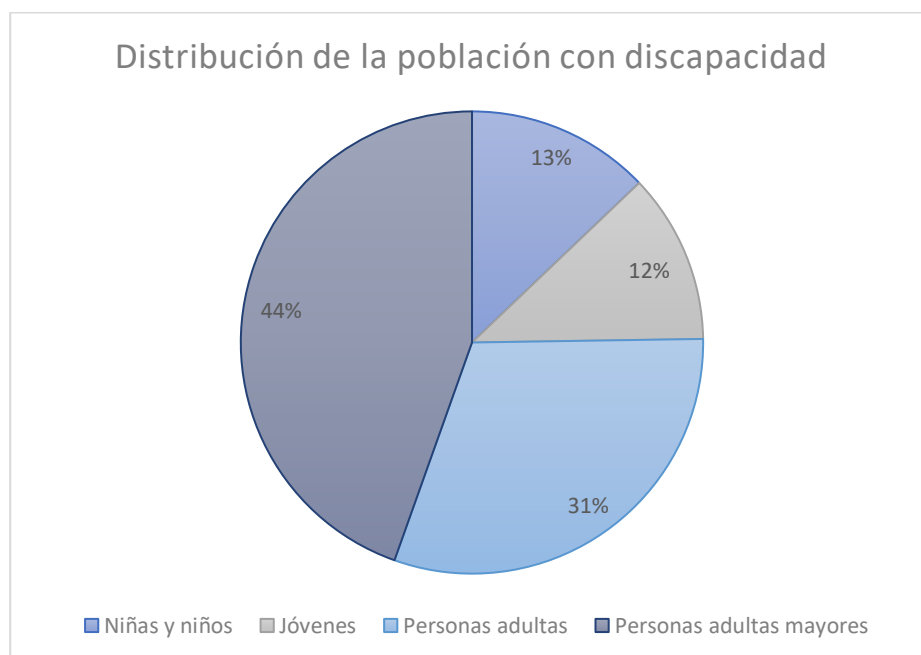


Ilustración 2 INEGI. Centro de Población y Vivienda 2020

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 grandes 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, ciclistas, entre otros; esto con el fin de permitir el respeto 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

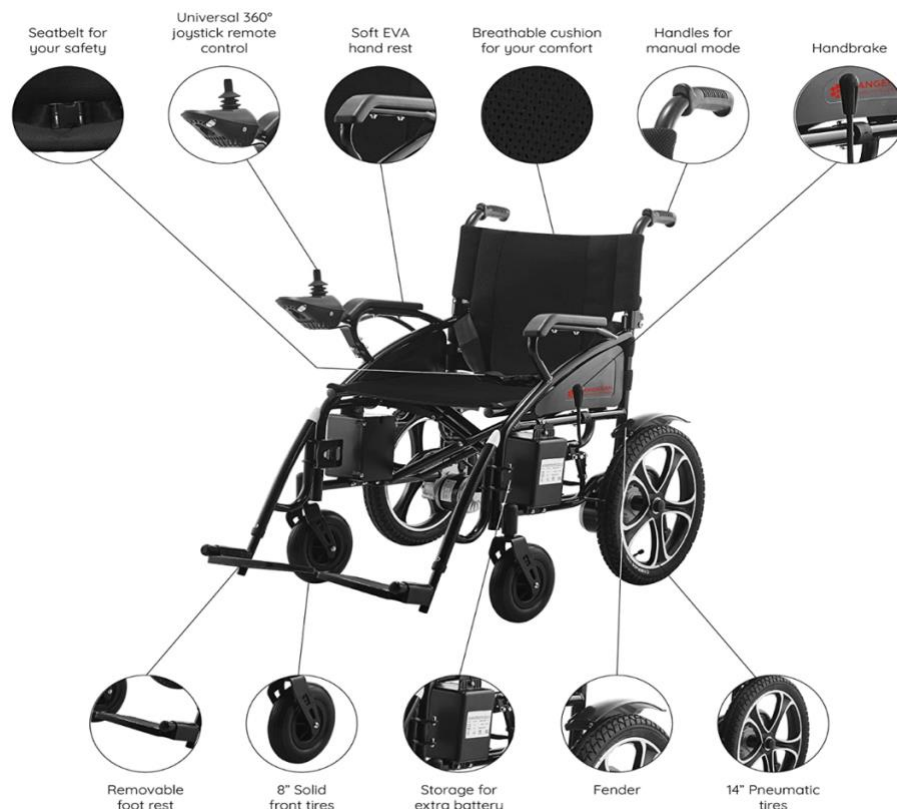


Ilustración 4 Silla de ruedas marca Bangeran

Otro ejemplo viene de la marca Wisging, que ofrece una silla de ruedas eléctrica 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 bluetooth 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 obstáculo, 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 visibilizar a los usuarios, se plantean dos formas de hacerlo:
 - Usar un buzzer que sirva como alarma ante ciertas circunstancias. De este existen de dos tipos: pasivos y activos; ambos operan bajo niveles de voltaje comunes 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:

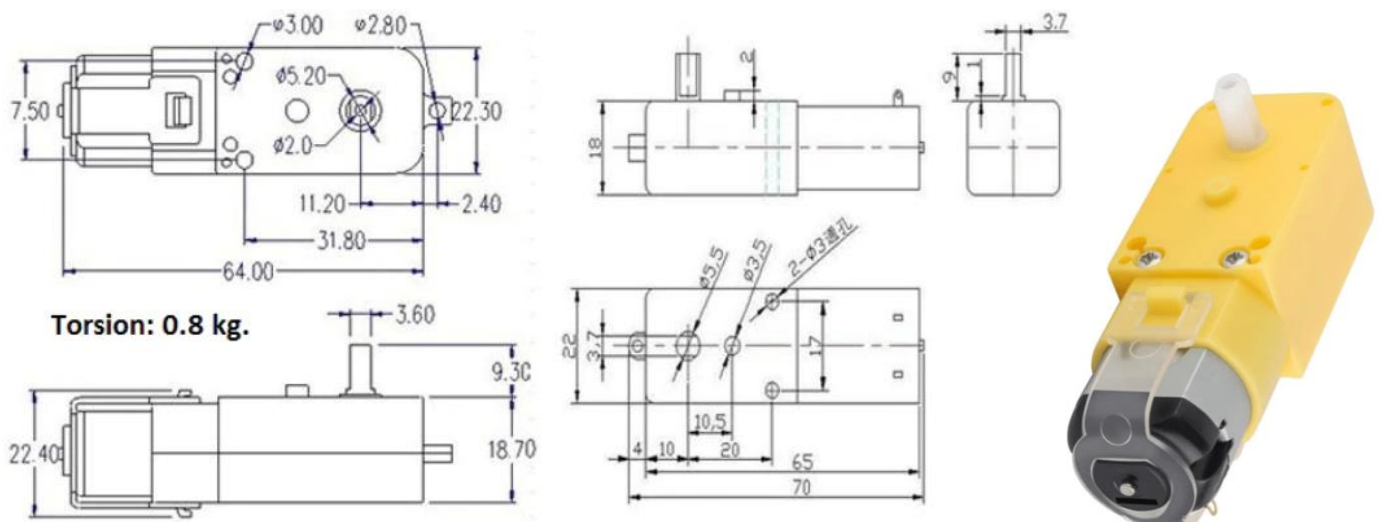


Ilustración 6 Motorreductor [7].

Tabla 1 Datos del motorreductor [7].

Reducción	48:1
Velocidad	170 rpm sin carga
Vplaje 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 útil 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.

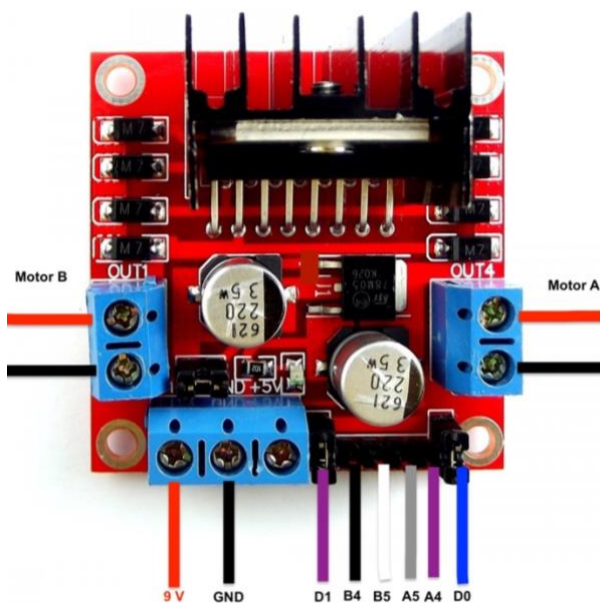


Ilustración 8 Conexiones en el Puente H

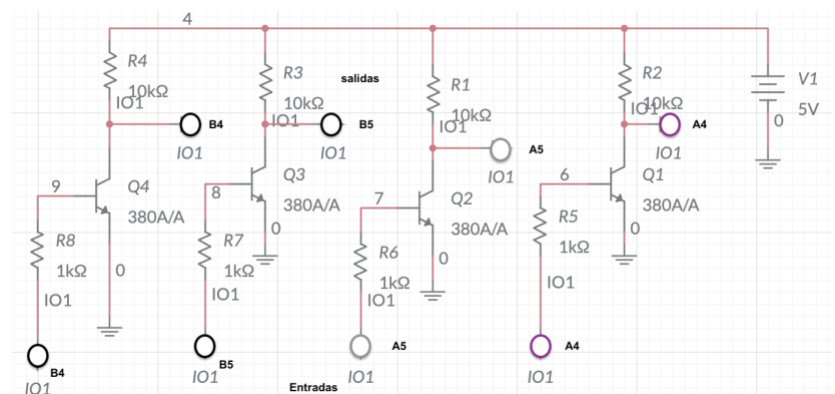


Ilustración 7 Etapa de potencia

- 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.

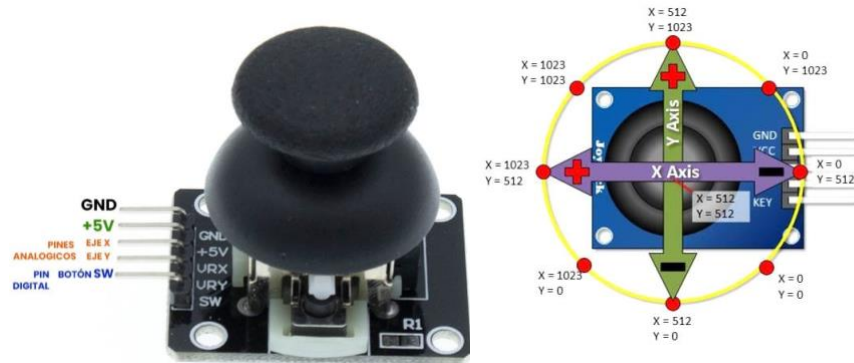


Ilustración 9 Módulo KY-023 [8]

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 Bluetooth 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 Bluetooth 2.0 a 2.4 HGz que es compatible con celulares Android, más no con los Iphone.

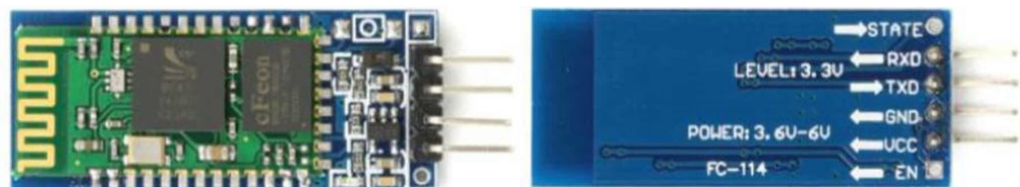


Ilustración 10 Módulo HC-06 [9]

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 emisión	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 potenciómetro digital, el cual se podrá controlar con el microcontrolador. El integrado empleado es el MCP41010.

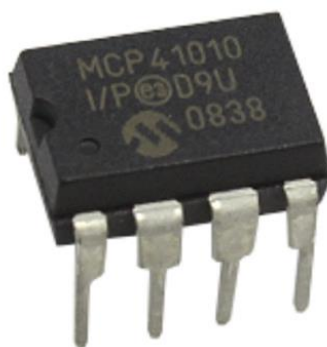


Ilustración 14 Potenciometró digital MCP41010 [13]

Tabla 8 Datos potenciómetro 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.

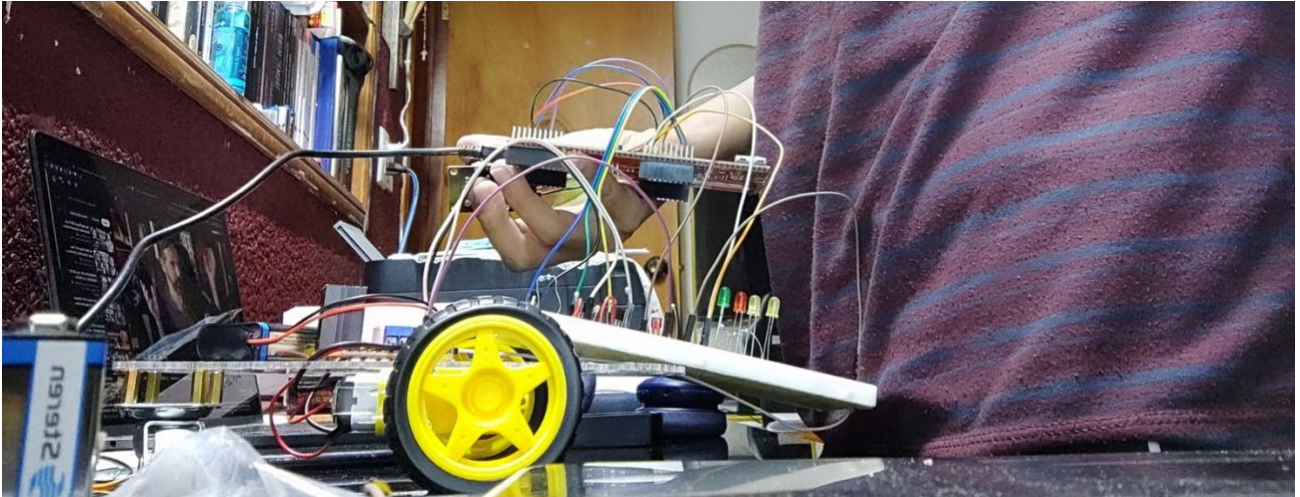


Ilustración 15 Estado del proyecto al 4 de mayo de 2022

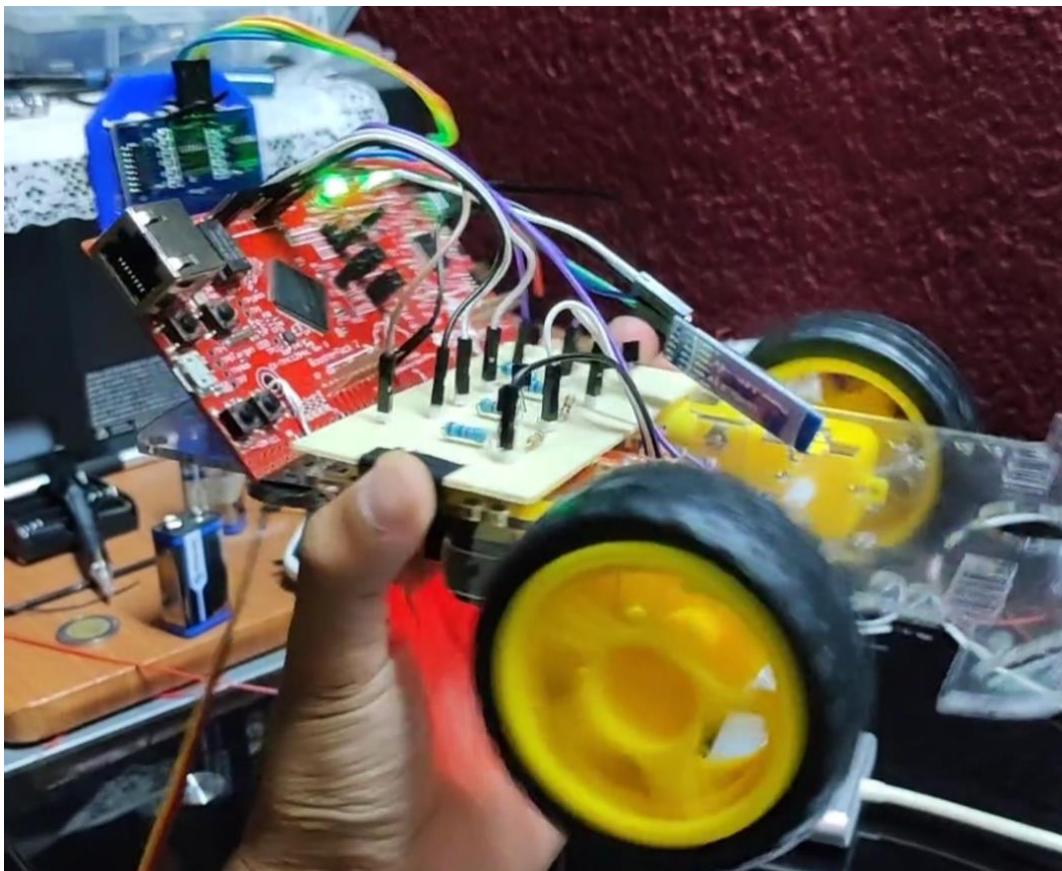


Ilustración 16 Avance del proyecto al 22 de mayo de 2022

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.

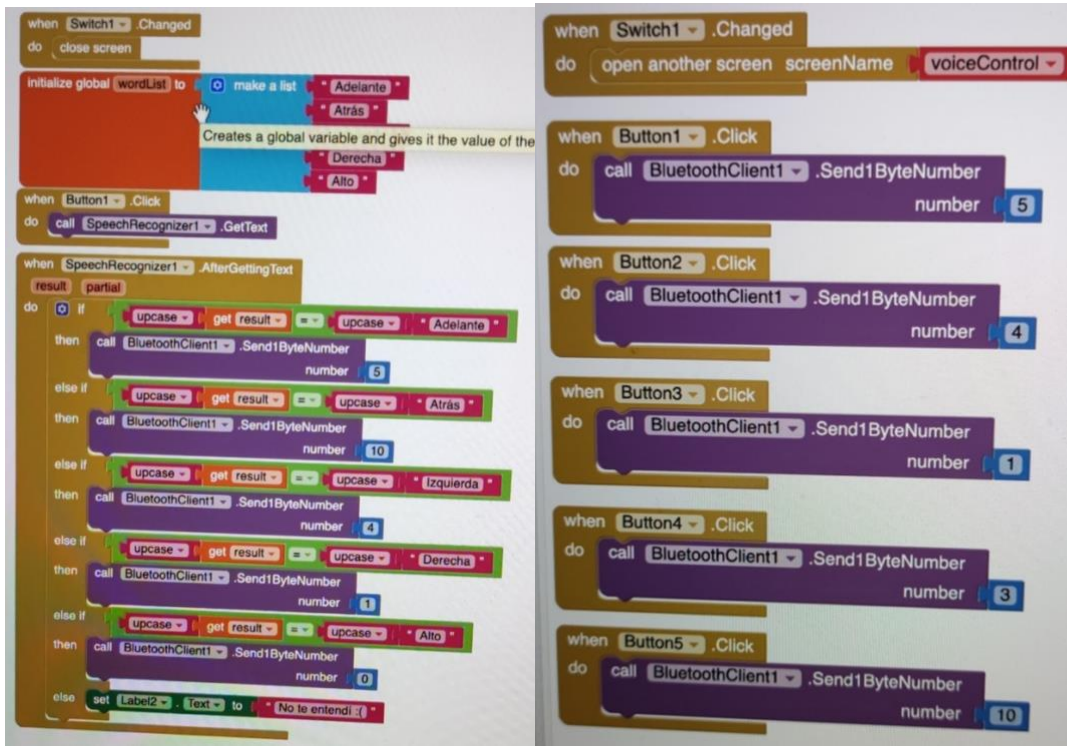


Ilustración 17 Acciones de la aplicación a 17 de mayo de 2022

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.

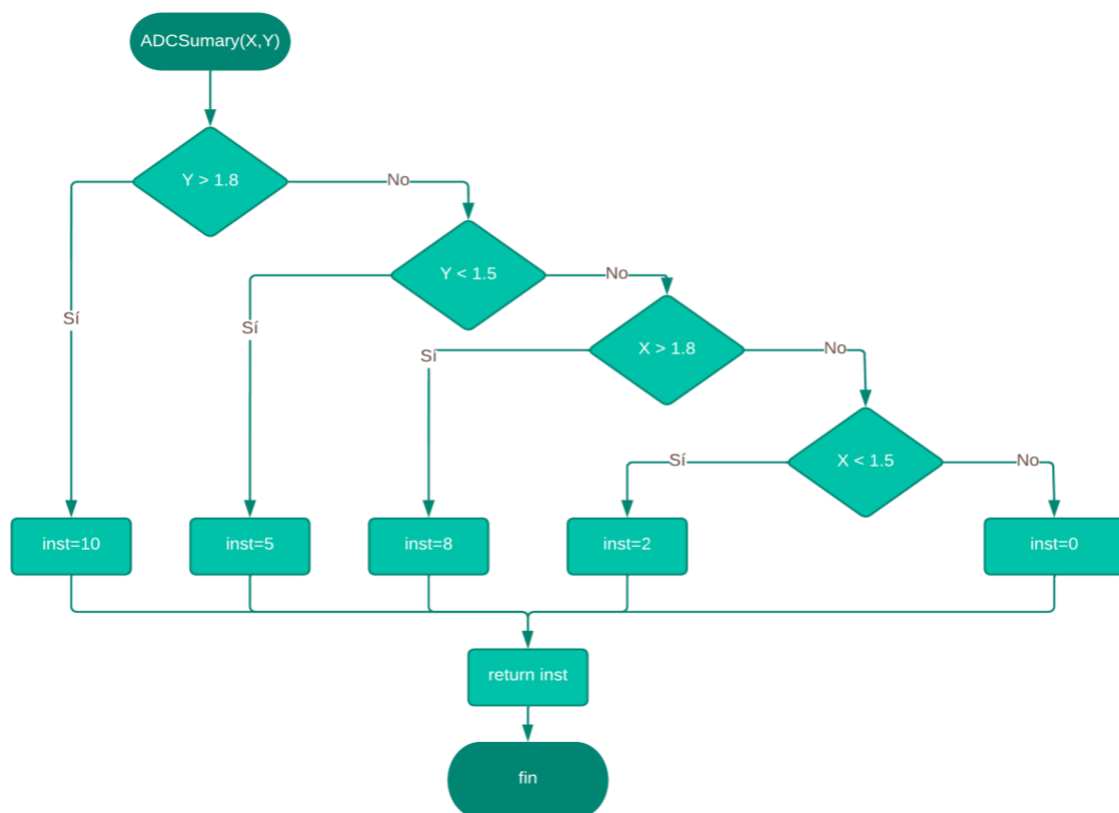


Ilustración 18 Flechas del display a 24 de mayo de 2022

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.



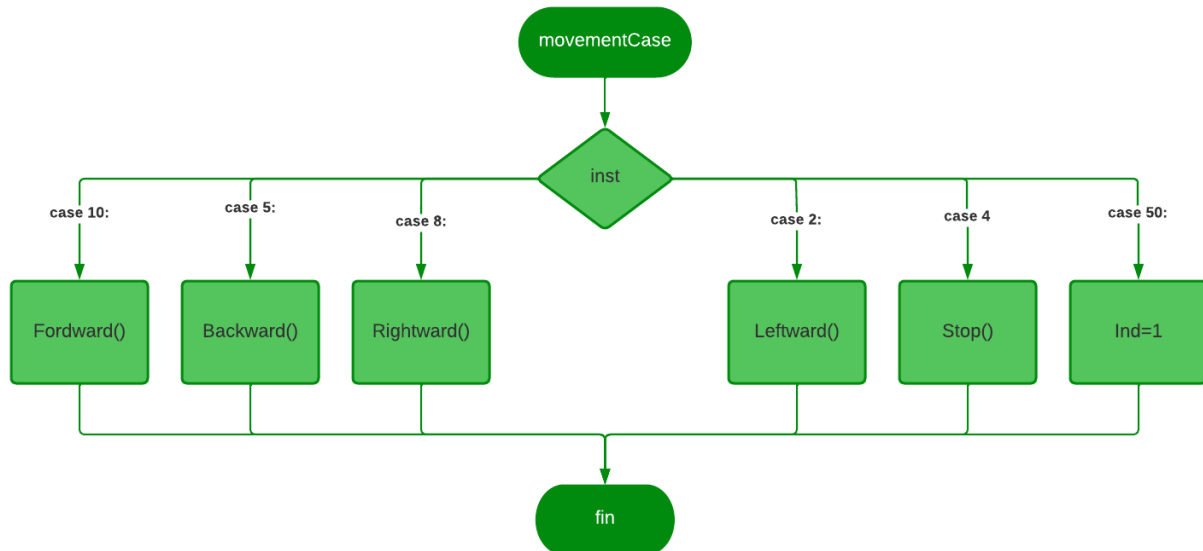
```

// Función que define la posición del joystick
// Recibe los valores del ADC asociados a X, Y
// Si debe ir hacia adelante
// Si debe ir hacia atrás
// Si debe ir hacia la izquierda
// Si debe ir hacia la derecha
// Si debe estar detenido
// Regresa el valor asociado a la dirección de desplazamiento

int ADCSummary(float X, float Y){
    int inst=0;
    if(Y>1.8){
        inst=10;
    }else{
        if(Y<1.5){
            inst=5;
        }else{
            if(X>1.8){
                inst=8;
            }else{
                if(X<1.5){
                    inst=2;
                }else{
                    inst=4;
                }
            }
        }
    }
    return inst;
}
  
```

Ilustración 19 Diagrama de flujo y código de la función ADCSummary

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 cada caso se llama la respectiva función de dirección.



```
// Función de determina la dirección de desplazamiento dependiendo del valor obtenido en el ADCSummary
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;
    }
}
```

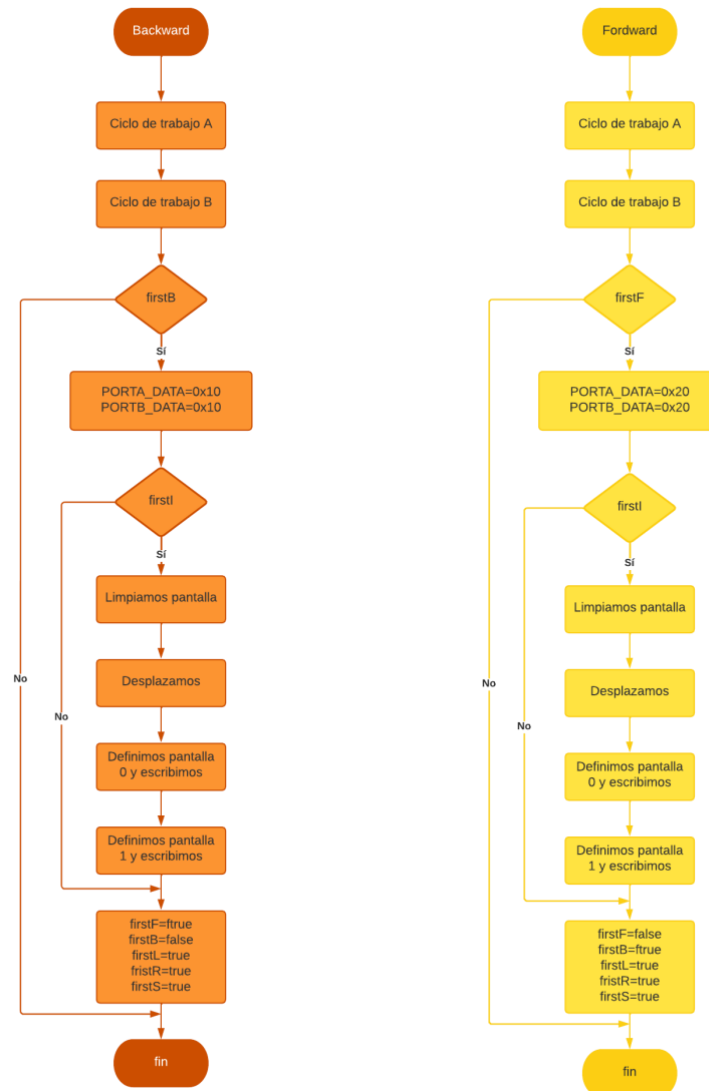
Ilustración 20 Diagrama de flujo y código de la función movementCase

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:

$$A + 1.7B = 8000$$

$$A + 3.3B = 3200$$

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



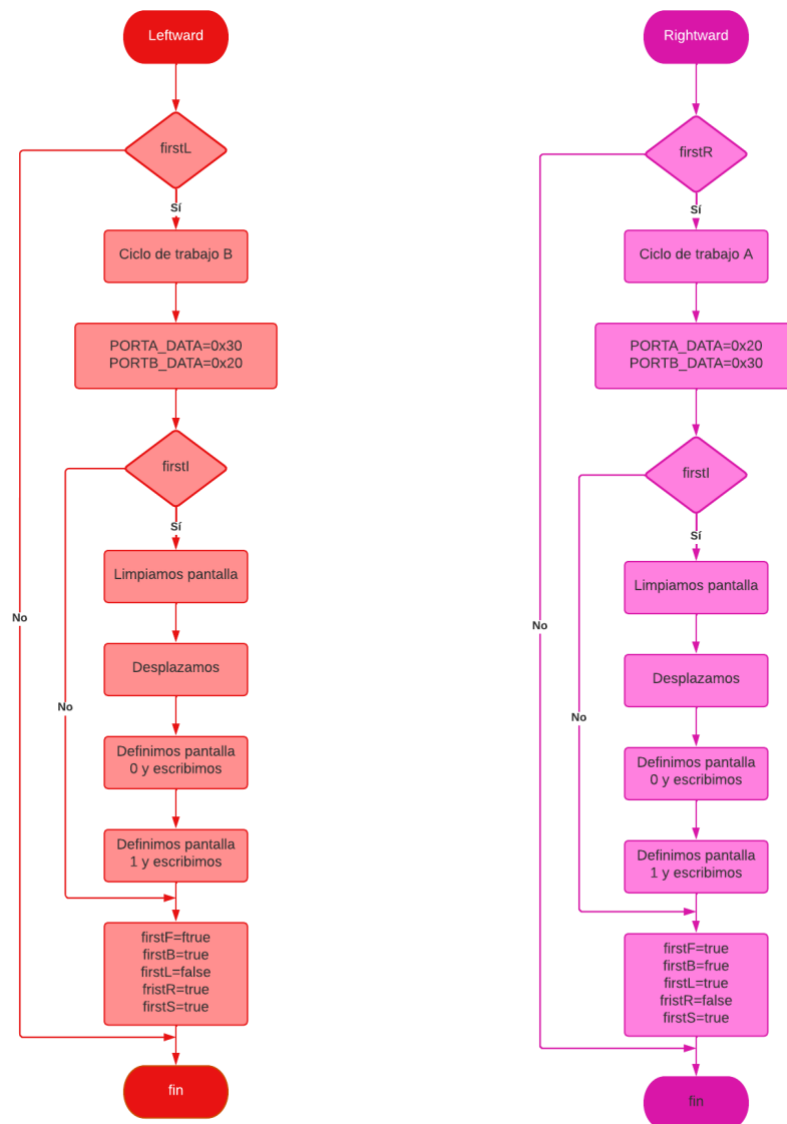
```
void Forward(float Y){
    TIMER0_TBMATCHR_R = A+B*Y;
    TIMER0_TBATCHR_R = A+B*Y;
    if(firstF){
        GPIO_PORTA_AHB_DATA_R = 0x20;
        GPIO_PORTB_AHB_DATA_R = 0x20;
        if(firstI){
            SSD1306_Clear();
            SSD1306_SCROLLINGL();
            SSD1306_Command(SSD1306_SETPAGE0);
            SSD1306_WriteString(" ");
            SSD1306_Command(SSD1306_SETPAGE1);
            SSD1306_WriteString(" ");
        }
        firstF=false;
        firstB=true;
        firstL=true;
        firstR=true;
        firstS=true;
    }
}

void Backward(float Y){
    TIMER0_TBMATCHR_R = A+B*(3.3-Y);
    TIMER0_TBATCHR_R = A+B*(3.3-Y);
    if(firstB){
        GPIO_PORTA_AHB_DATA_R = 0x10;
        GPIO_PORTB_AHB_DATA_R = 0x10;
        if(firstI){
            SSD1306_Clear();
            SSD1306_SCROLLINGR();
            SSD1306_Command(SSD1306_SETPAGE0);
            SSD1306_WriteString(" ");
            SSD1306_Command(SSD1306_SETPAGE1);
            SSD1306_WriteString(" ");
        }
        firstB=false;
        firstF=true;
        firstL=true;
        firstR=true;
        firstS=true;
    }
}
```

//Ciclo de trabajo dependiendo de la inclinación del Joystick
//Ciclo de trabajo dependiendo de la inclinación del Joystick
//Condición de estado del movimiento hacia adelante
//Asigna 1 en A5 y un 0 en A4
//Asigna 1 en B5 y un 0 en B4
//Condición de interrupción
//Limpia pantalla
//Desplaza a la izquierda
//Página 0
//Mensaje para página 0
//Página 1
//Mensaje para página 1
//Cambia estado de forward
//Resetea los otros estados

//Ciclo de trabajo dependiendo de la inclinación del Joystick
//Ciclo de trabajo dependiendo de la inclinación del Joystick
//Condición de estado del movimiento hacia adelante
//Asigna 1 en A4 y 0 a B4
//Asigna 1 a B5 y 0 a B4
//Condición de interrupción
//Limpia pantalla
//Desplaza a la derecha
//Define página 0
//Cadena para página 0
//Define página 1
//Cadena página 1
//Cambia estado backward
//Resetea los otros estados

Ilustración 21 Funciones forward y backward

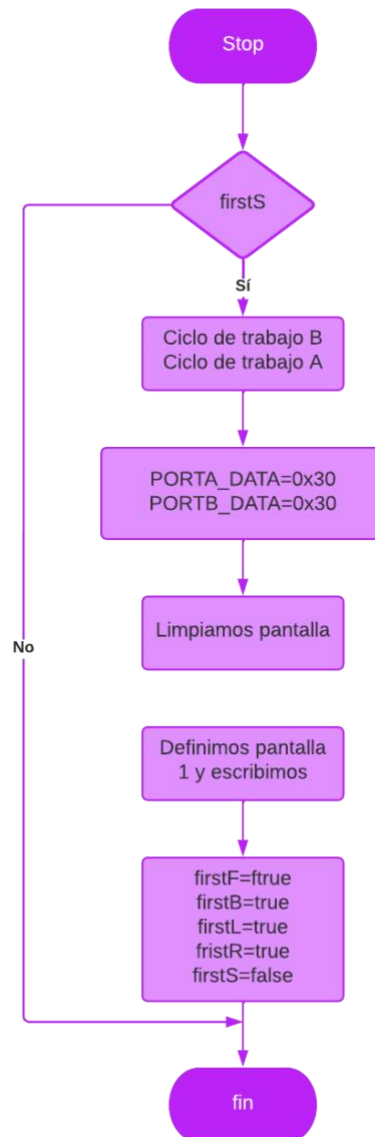


```
void Leftward(void){
    if(firstL){
        GPIO_PORTB_AHB_DATA_R = 0x20;
        GPIO_PORTA_AHB_DATA_R = 0x30;
        TIMER0_TBMATCHR_R = 6400;
        if(firstI){
            SSD1306_Clear();
            SSD1306_SCROLLINGL();
            SSD1306_Command(SSD1306_SETPAGE0);
            SSD1306_WriteString("0");
            SSD1306_Command(SSD1306_SETPAGE1);
            SSD1306_WriteString("1");
        }
        firstF=true;
        firstR=true;
        firstB=true;
        firstS=true;
        firstL=false;
    }
}

void Rightward(void){
    if(firstR){
        GPIO_PORTA_AHB_DATA_R = 0x20;
        GPIO_PORTB_AHB_DATA_R = 0x30;
        TIMER0_TAMATCHR_R = 6400;
        if(firstI){
            SSD1306_Clear();
            SSD1306_SCROLLINGL();
            SSD1306_Command(SSD1306_SETPAGE0);
            SSD1306_WriteString("0");
            SSD1306_Command(SSD1306_SETPAGE1);
            SSD1306_WriteString("1");
        }
        firstF=true;
        firstS=true;
        firstB=true;
        firstL=true;
        firstR=false;
    }
}
```

//Indica desplazamiento hacia adelante de la rueda asociada a B
//Impide el movimiento de motor A
//Ciclo de trabajo del 60% asociado al motor
//Limpia pantalla
//Desplaza hacia la izquierda
//Define pagina 0
//Envia cadena a pagina 0
//Define pagina 1
//Cadena pagina 1
//Reset a los demas edos
//Cambia edo leftward
//Valida edo right
//Movimiento de A hacia adelante
//Impide movimiento de B
//Ciclo de trabajo del 60% asociado al motor
//Valida eso de interrupción
//Limpia pantalla
//Desplazamiento a la izquierda
//Establece pagina 0
//Cadena a pagina 0
//Define pagina 1
//Cadena a pagina 1
//reset a todos los demás edos
//edo de right cambiado

Ilustración 22 Funciones rightward y leftward

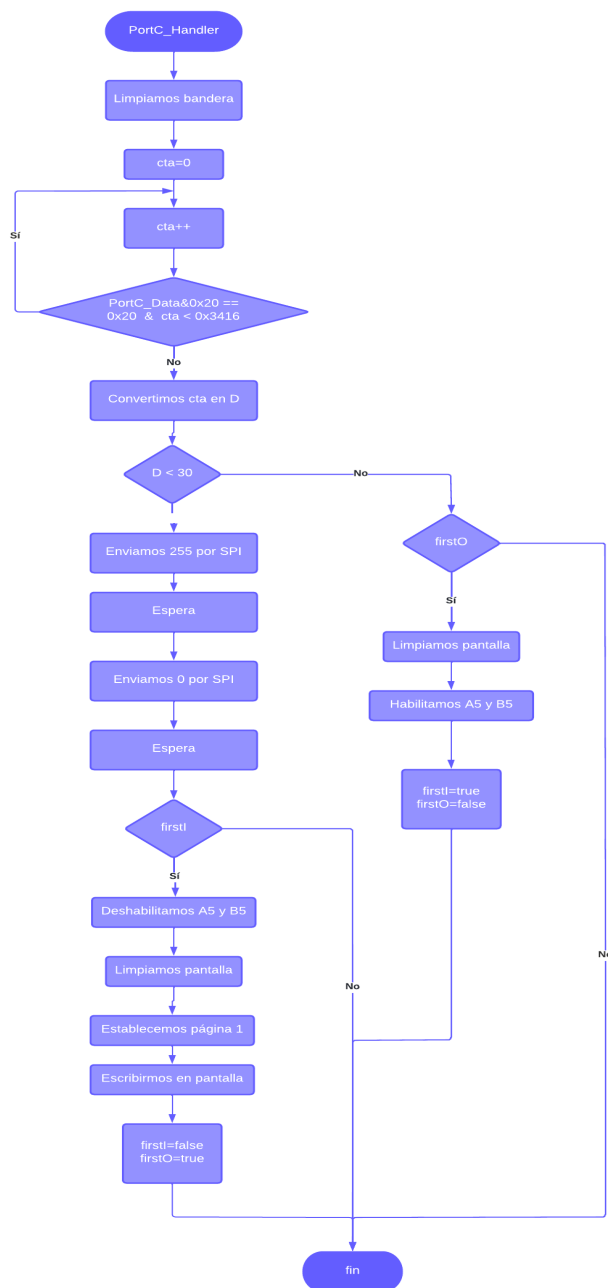


```

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);      //Limpiar pantalla
    SSD1306_WriteString("ALTO");             //Define pagina 1
    firstF=true;                             //Envia dato
    firstB=true;                             //reset a los demás edos
    firstL=true;
    firstR=true;
    firstS=false;
  }
  ind1=1;                                   //cambia edo del stop
  ind2=1;                                   //Cambia edo del indicador 2
}
  
```

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 funcionan 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.



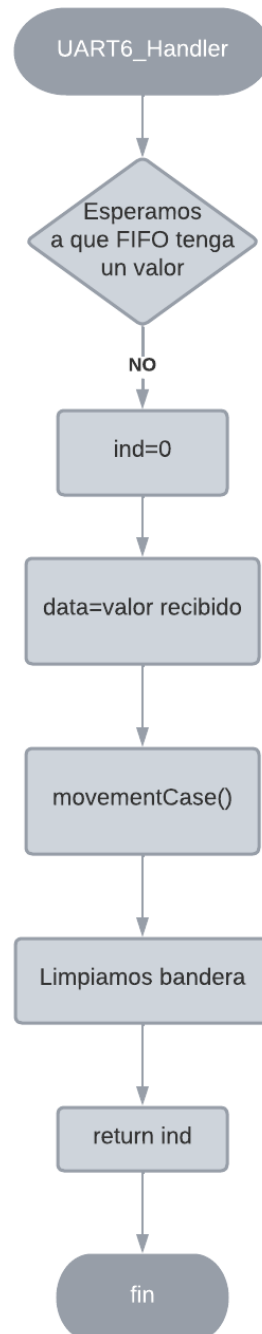
Para la interrupción en el puerto C, tenemos que entra a esta cuando hay un flanco de subida en el puerto C pin 5. Dentro la rutina primero limpiamos la bandera (para que al terminar salga de la rutina), después entra en un ciclo que aumenta una cuenta con valor inicial de 0 (esta cuenta se mantendrá 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 potenciómetro 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;
  cta=0;
  do{
    cta++;
  }while(((GPIO_PORTC_AHB_DATA_R & 0x20)==0x20) & (cta < 0x0003416));
  D=0.03*cta;

  if(D<30){
    pot_setVal(255);
    for (i = 0; i < 20000; i++);
    pot_setVal(0);
    for (i = 0; i < 20000; i++);
    if(firstI){
      GPIO_PORTA_AHB_DEN_R &= ~0x2F;
      GPIO_PORTB_AHB_DEN_R &= ~0x23;
      SSD1306_Clear();
      SSD1306_Command(SSD1306_SETPAGE1);
      SSD1306_WriteString("CUIDADO");
      firstI=false;
      firstO=true;
      //IndI=0;
    }
  }
  else{
    if(firstO){
      pot_setVal(0);
      SSD1306_Clear();
      GPIO_PORTA_AHB_DEN_R |=0x30;
      GPIO_PORTB_AHB_DEN_R |=0x30;
      firstI=true;
      firstO=false;
      //IndI=1;
    }
  }
}
```

//limpia la bandera
 //Inicilaiza la cuenta a 0
 //Incrementa la cuenta mientras cumple la condición
 //Mientras el acc no haya respondido o mientras la cuenta sea menor a una distancia equivalente a 400 cm
 //Convierte la cuenta en su equivalente en distancia
 //Condición para detar el movimiento
 //Enviamos el dato al potenciómetro digital pasar voltaje
 //Tiempo en alto
 //Enviamos dato para que en potenciómetro no pase nada
 //Tiempo en bajo
 //Reseta si es la primera vez que entramos al ciclo de la interrupción
 //Deshabilita A5
 //Deshabilita B5
 //Limpiamos pantalla
 //Establecemos pagina
 //Enviamos texto
 //Decimos que no es la primera vez que está en la interrupción
 //Reiniciamos el valor del indicador de estar fuera de la interrupción
 //Evalua en el indicador de estar fuera
 //Envia dato 0 al potenciómetro
 //Limpia la pantalla
 //Habilita A5
 //Habilita B5
 //Reseta Indicador inside
 //Cambia indicador out

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.

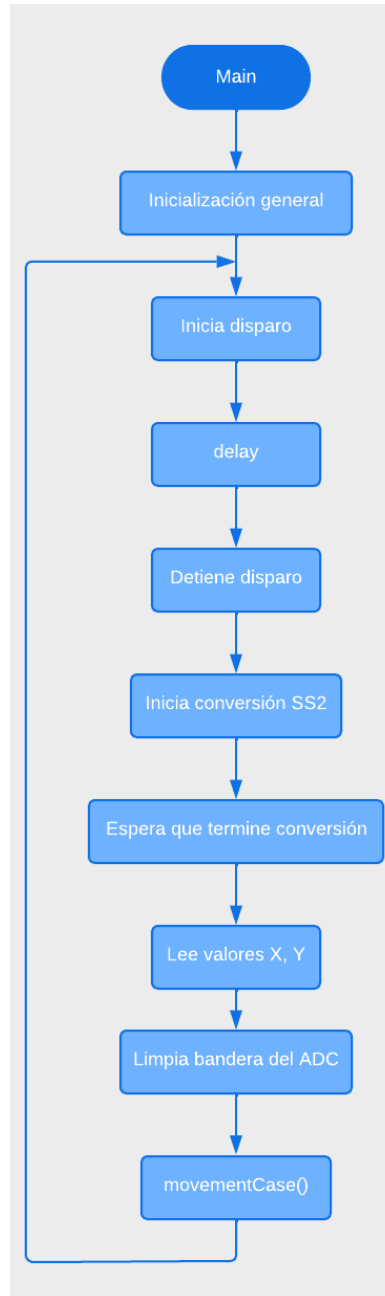


```
int UART6_Handler(void){  
    while((UART6_FR_R&UART_FR_RXFE) != 0);  
    ind2=0;  
    data = UART6_DR_R&0xFF;  
    movementCase(data, 1.9);  
    UART6_ICR_R = 0x00000030;  
    return ind2;  
}
```

//Se espera a que el FIFO RX tenga un valor
//Cambia el estado del indicador
//guarda en valor de la fifo en una variable
//usa el valor de esa variable para enviarlo a la función de movimiento
// Limpia bandera de interrupción

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 $10\ \mu\text{s}$ 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;
        for(i = 0; i < 11; i++){
            GPIO_PORTC_AHB_DATA_R &= ~0X10;
            ADC0_PSSI_R = 0x0004;
            while((ADC0_RIS_R & 0x04) == 0);
            Y = (3.3/4096)*(ADC0_SSIF02_R & 0xFFF);
            X = (3.3/4096)*(ADC0_SSIF02_R & 0xFFF);
            ADC0_ISC_R = 0x0004;
            movementCase(ADCSumary(X,Y)*ind,Y);
        }
    }
}
  
```

//Enciende el puerto para disparar la señal de trigger
 // Retardo > 10us
 //Apaga el puerto para detener el disparo.
 //Inicia conversión del SS2
 // Espera a que SS2 termine conversión (polling)
 // se lee el primer resultado asociado a PE5
 // se lee el segundo resultado asociado a PE4
 //Limpia la bandera RIS del ADC0
 //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).

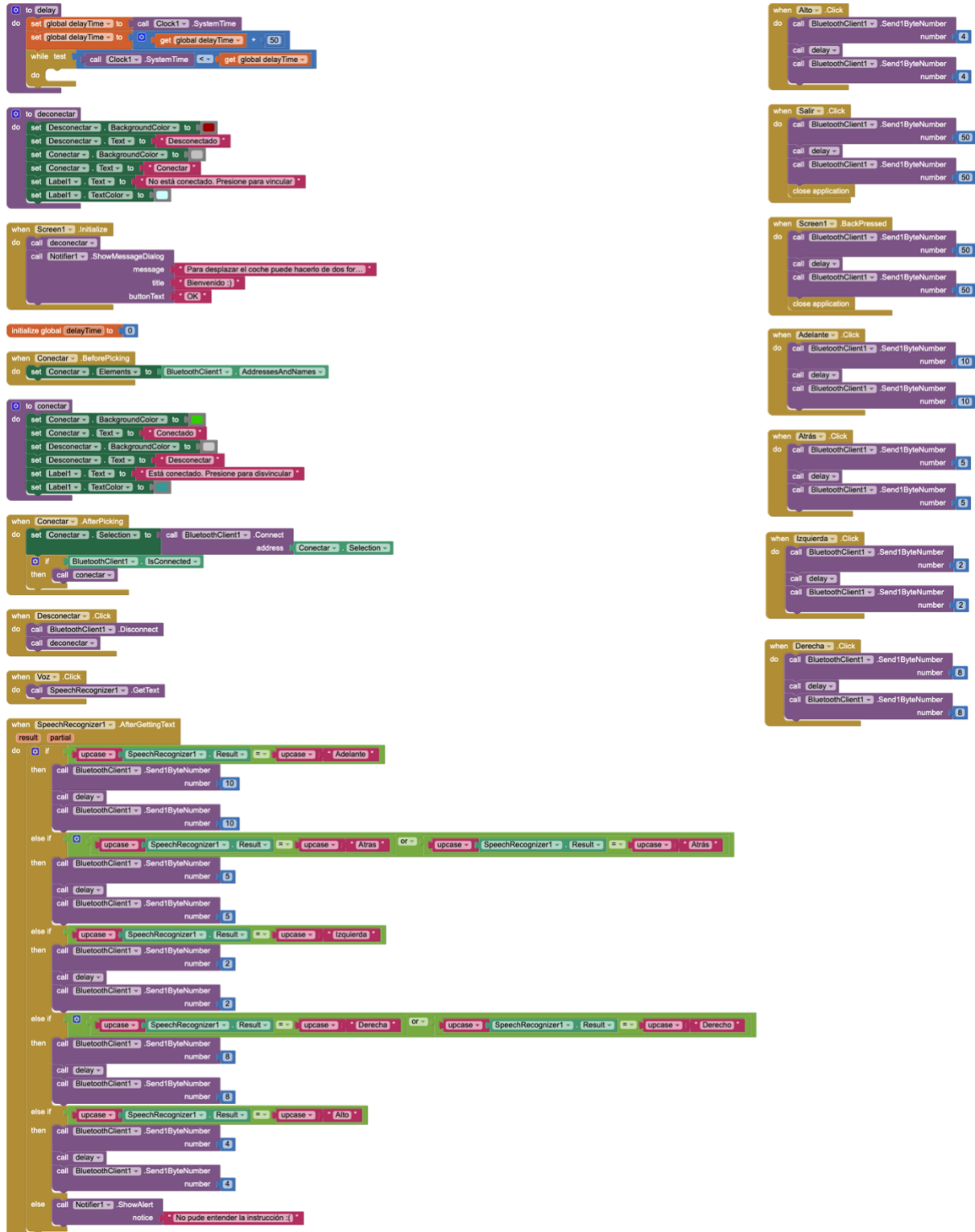


Ilustración 26 Bloques de la app

Y la aplicación misma se muestra a continuación:

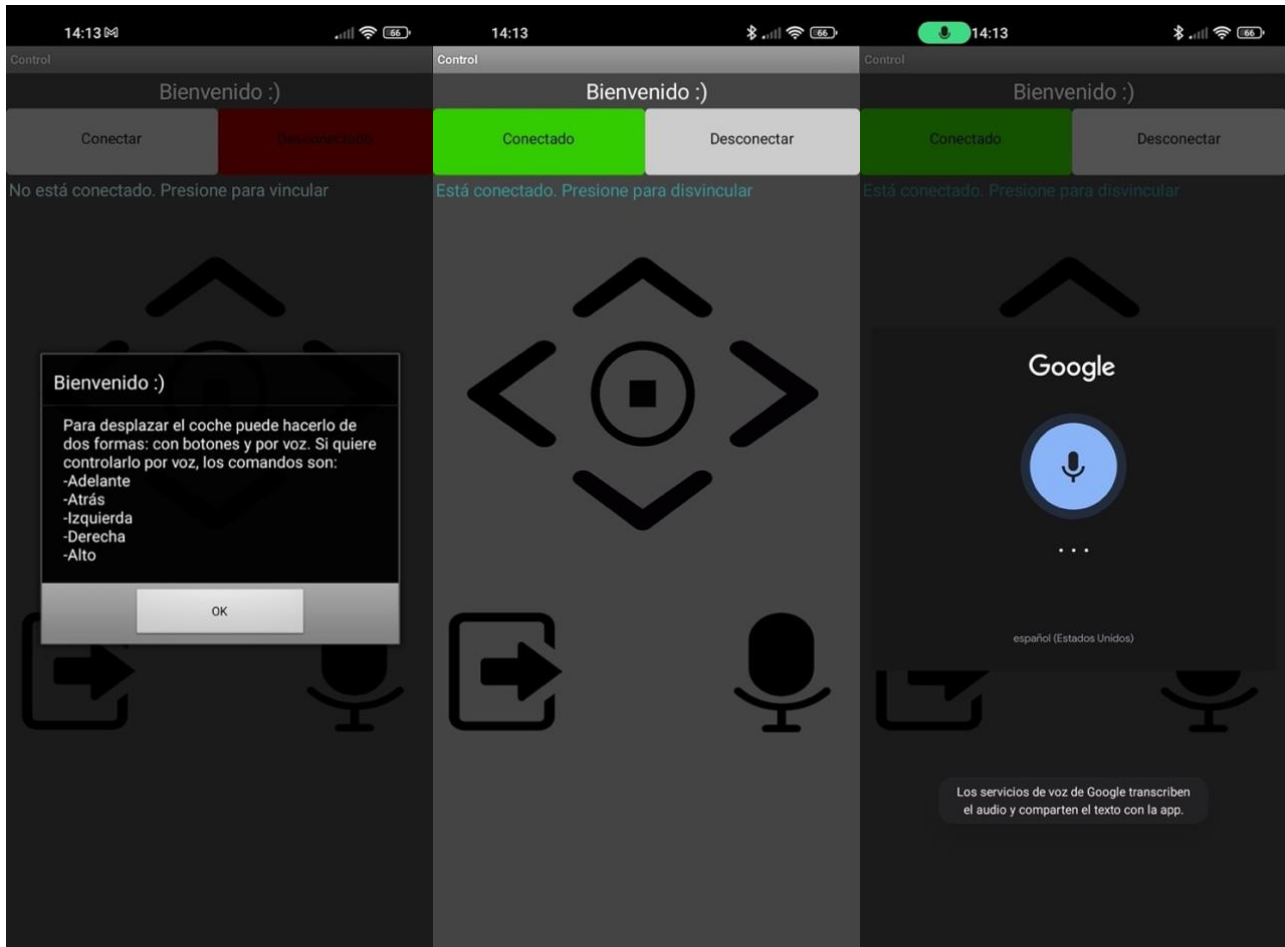


Ilustración 27 Capturas de pantalla de la aplicación

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

00000ee4:	for(i = 0; i< 100; i++);		
00000ee6:	49E6	ldr	r1, [pc, #0x398] 2
00000ee8:	2000	movs	r0, #0 1
00000eec:	6008	str	r0, [r1] 2
00000eee:	48E5	ldr	r0, [pc, #0x394] 2 12
63	6800	ldr	r0, [r0] 2
00000ef0:	2864	cmp	r0, #0x64 1
00000ef2:	D207	bhs	\$C\$L127 2
00000ef4:	\$C\$L126:		
00000ef6:	49E3	ldr	r1, [pc, #0x38c] 2
00000efa:	6808	ldr	r0, [r1] 2
65	1C40	adds	r0, r0, #1 1
00000efc:	6008	str	r0, [r1] 2 14
00000efe:	48E1	ldr	r0, [pc, #0x384] 2
00000f00:	6800	ldr	r0, [r0] 2
00000f04:	2864	cmp	r0, #0x64 1
66	D3F7	blo	\$C\$L126 2
00000f06:			

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.

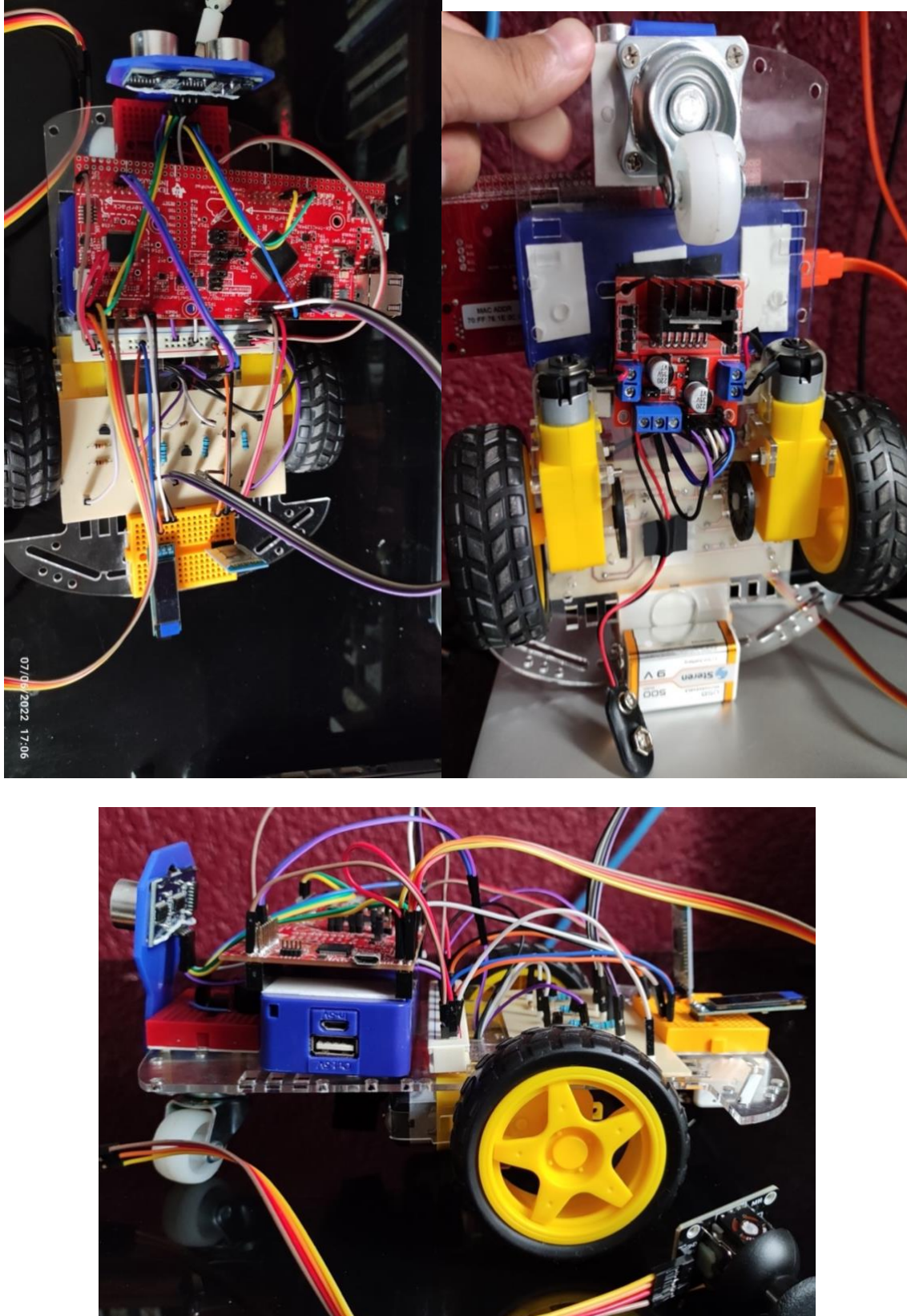
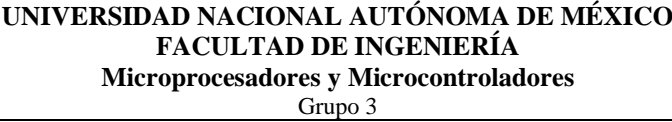


Ilustración 29 Versión final del proyecto



```

195 while((SYSCTL_PWDIO_R & (0x20)) || (SYSCTL_ACCDET_R14) == 0) {
196     UMT_Init();
197     ADC_Init();
198     PortC_Init();
199     PWM_Init();
200     SPI_Init();
201     I2C_Init();
202     while(!((PNS_R & 0x00000001) != 0));
203     SSD1306_Init();
204 }
205 //----- Función que define la posición del joystick -----
206 int ADCSummary(float X, float Y){
207     int inst=0;
208     if((x>3.0){
209         inst=0;
210     }else{
211         if((x<1.0){
212             inst=1;
213         }else{
214             if((x>1.0){
215                 inst=0;
216             }else{
217                 if((x<1.5){
218                     inst=2;
219                 }else{
220                     inst=4;
221                 }
222             }
223         }
224     }
225     return inst;
226 }
227 //----- Funciones para el movimiento -----
228 void Forward(float Y){
229     TIMERS_TMRATCHR_R = A+B*(3.3-Y);
230     TIMERS_TMRATCHR_R = A+B*(3.3-Y);
231     if(first){
232         GPIO_PORTA_AHB_DATA_R = 0x20;
233         GPIO_PORTB_AHB_DATA_R = 0x20;
234         if(first){
235             SSD1306_Clear();
236             SSD1306_Command(SSD1306_SETPAGE);
237             SSD1306_WriteString("-----");
238             SSD1306_Command(SSD1306_SETPAGE1);
239             SSD1306_WriteString("-----");
240         }
241         first=False;
242         firstBtrue;
243         firstLtrue;
244         firstRtrue;
245         firstStrue;
246     }
247 }
248 void Backward(float Y){
249     TIMERS_TMRATCHR_R = A+B*(3.3-Y);
250     TIMERS_TMRATCHR_R = A+B*(3.3-Y);
251     if(first){
252         GPIO_PORTA_AHB_DATA_R = 0x10;
253         GPIO_PORTB_AHB_DATA_R = 0x30;
254         if(first){
255             SSD1306_Clear();
256             SSD1306_Command(SSD1306_SETPAGE);
257             SSD1306_WriteString("-----");
258             SSD1306_Command(SSD1306_SETPAGE1);
259             SSD1306_WriteString("-----");
260         }
261         first=False;
262         firstBtrue;
263         firstLtrue;
264         firstRtrue;
265         firstStrue;
266     }
267 }
268 void Leftward(void){
269     if(first){
270         GPIO_PORTA_AHB_DATA_R = 0x20;
271         GPIO_PORTB_AHB_DATA_R = 0x30;
272         TIMERS_TMRATCHR_R = 0x00;
273         if(first){
274             SSD1306_Clear();
275             SSD1306_Command(SSD1306_SETPAGE);
276             SSD1306_WriteString("-----");
277             SSD1306_Command(SSD1306_SETPAGE1);
278             SSD1306_WriteString("-----");
279         }
280         first=True;
281         firstBtrue;
282         firstLtrue;
283         firstRtrue;
284         firstStrue;
285     }
286 }
287 void Rightward(void){
288     if(first){
289         GPIO_PORTA_AHB_DATA_R = 0x20;
290         GPIO_PORTB_AHB_DATA_R = 0x30;
291         TIMERS_TMRATCHR_R = 0x00;
292         if(first){
293             SSD1306_Clear();
294             SSD1306_Command(SSD1306_SETPAGE);
295             SSD1306_WriteString("-----");
296             SSD1306_Command(SSD1306_SETPAGE1);
297             SSD1306_WriteString("-----");
298         }
299         first=True;
300         firstBtrue;
301         firstLtrue;
302         firstRtrue;
303         firstStrue;
304     }
305 }
306 void Stop(void){
307     if(first){
308         GPIO_PORTA_AHB_DATA_R = 0x20;
309         GPIO_PORTB_AHB_DATA_R = 0x30;
310         TIMERS_TMRATCHR_R = 0x00;
311         if(first){
312             SSD1306_Clear();
313             SSD1306_Command(SSD1306_SETPAGE);
314             SSD1306_WriteString("ALTO");
315             SSD1306_Command(SSD1306_SETPAGE1);
316             SSD1306_WriteString("ALTO");
317         }
318         first=True;
319         firstBtrue;
320         firstLtrue;
321         firstRtrue;
322         firstStrue;
323     }
324 }
325 //----- Función de detección de la dirección de desplazamiento -----
326 void movementCase(int n, float Y){
327     switch(n){
328         case 0:
329             Forward(Y);
330             break;
331         case 1:
332             Backward(Y);
333             break;
334         case 2:
335             Leftward(Y);
336             break;
337         case 3:
338             Rightward(Y);
339             break;
340         case 4:
341             Stop();
342             break;
343         case 5:
344             Leftward(Y);
345             break;
346         case 6:
347             Rightward(Y);
348             break;
349         case 7:
350             Stop();
351             break;
352     }
353 }
354 //----- Interacción del eje C (asociado al ultrasonido) -----
355 void PortC_Handler(void){
356     GPIO_PORTC_AHB_ICR_R = 0x20;
357     ctas=0;
358     do{
359         ctas++;
360     }while(((GPIO_PORTC_AHB_DATA_R & 0x20) == 0x20) & (ctas < 0x00003410));
361     Do_Delay(1);
362     if(ctas==255){
363         for (i = 0; i < 20000; i++){
364             port_setval(0);
365         }
366         for (i = 0; i < 20000; i++){
367             port_setval(0);
368         }
369         if(first){
370             GPIO_PORTA_AHB_DEN_R &= ~0x20;
371             GPIO_PORTB_AHB_DEN_R &= ~0x20;
372             SSD1306_Clear();
373             SSD1306_Command(SSD1306_SETPAGE1);
374             SSD1306_WriteString("CUIDADO");
375             first=False;
376             firstBtrue;
377         }
378     }
379     else{
380         if(first){
381             port_setval(0);
382             SSD1306_Clear();
383             GPIO_PORTA_AHB_DEN_R |= 0x20;
384             GPIO_PORTB_AHB_DEN_R |= 0x20;
385             first=True;
386             firstBfalse;
387             firstStrue;
388         }
389     }
390 }
391 //----- Función de detección de la dirección de desplazamiento -----
392 void movementCase(int n, float Y){
393     switch(n){
394         case 0:
395             Forward(Y);
396             break;
397         case 1:
398             Backward(Y);
399             break;
400         case 2:
401             Leftward(Y);
402             break;
403         case 3:
404             Rightward(Y);
405             break;
406         case 4:
407             Stop();
408             break;
409         case 5:
410             Leftward(Y);
411             break;
412         case 6:
413             Rightward(Y);
414             break;
415         case 7:
416             Stop();
417             break;
418     }
419 }
420 //----- Interacción del eje C (asociado al ultrasonido) -----
421 void PortC_Handler(void){
422     GPIO_PORTC_AHB_ICR_R = 0x20;
423     ctas=0;
424     do{
425         ctas++;
426     }while(((GPIO_PORTC_AHB_DATA_R & 0x20) == 0x20) & (ctas < 0x00003410));
427     Do_Delay(1);
428     if(ctas==255){
429         for (i = 0; i < 20000; i++){
430             port_setval(0);
431         }
432         for (i = 0; i < 20000; i++){
433             port_setval(0);
434         }
435         if(first){
436             GPIO_PORTA_AHB_DEN_R &= ~0x20;
437             GPIO_PORTB_AHB_DEN_R &= ~0x20;
438             SSD1306_Clear();
439             SSD1306_Command(SSD1306_SETPAGE1);
440             SSD1306_WriteString("CUIDADO");
441             first=False;
442             firstBtrue;
443         }
444     }
445     else{
446         if(first){
447             port_setval(0);
448             SSD1306_Clear();
449             GPIO_PORTA_AHB_DEN_R |= 0x20;
450             GPIO_PORTB_AHB_DEN_R |= 0x20;
451             first=True;
452             firstBfalse;
453             firstStrue;
454         }
455     }
456 }

```




Ilustración 30 Código del programa principal

[illegible]



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
FACULTAD DE INGENIERÍA
Microprocesadores y Microcontroladores
Grupo 3

Semestre: 2022-2
Página 28 de 32



```
214 S501386_Data[0xFF];
215 S501386_Data[0x33];
216 S501386_Data[0x33];
217 S501386_Data[0xFF];
218 S501386_Data[0xFF];
219 S501386_Data[0x00];
220 }
221 void Letra_B(void){
222 S501386_Data[0xFF];
223 S501386_Data[0xFF];
224 S501386_Data[0x99];
225 S501386_Data[0x00];
226 S501386_Data[0xFF];
227 S501386_Data[0xFF];
228 S501386_Data[0x00];
229 }
230 void Letra_C(void){
231 S501386_Data[0xFF];
232 S501386_Data[0xFF];
233 S501386_Data[0xC3];
234 S501386_Data[0xC3];
235 S501386_Data[0xC3];
236 S501386_Data[0xC3];
237 S501386_Data[0x00];
238 }
239 void Letra_D(void){
240 S501386_Data[0xFF];
241 S501386_Data[0xFF];
242 S501386_Data[0xC3];
243 S501386_Data[0xC3];
244 S501386_Data[0xFF];
245 S501386_Data[0xFF];
246 S501386_Data[0x00];
247 }
248 void Letra_E(void){
249 S501386_Data[0xFF];
250 S501386_Data[0xFF];
251 S501386_Data[0x00];
252 S501386_Data[0x00];
253 S501386_Data[0x00];
254 S501386_Data[0x00];
255 S501386_Data[0x00];
256 S501386_Data[0x00];
257 }
258 void Letra_F(void){
259 S501386_Data[0xFF];
260 S501386_Data[0xFF];
261 S501386_Data[0xC3];
262 S501386_Data[0x33];
263 S501386_Data[0x00];
264 S501386_Data[0x00];
265 S501386_Data[0x00];
266 }
267 void Letra_G(void){
268 S501386_Data[0xFF];
269 S501386_Data[0xFF];
270 S501386_Data[0x00];
271 S501386_Data[0x00];
272 S501386_Data[0x00];
273 S501386_Data[0x00];
274 S501386_Data[0x00];
275 }
276 void Letra_H(void){
277 S501386_Data[0xFF];
278 S501386_Data[0xFF];
279 S501386_Data[0x10];
280 S501386_Data[0x10];
281 S501386_Data[0xFF];
282 S501386_Data[0xFF];
283 S501386_Data[0x00];
284 }
285 void Letra_I(void){
286 S501386_Data[0xC3];
287 S501386_Data[0xC3];
288 S501386_Data[0xFF];
289 S501386_Data[0xFF];
290 S501386_Data[0xC3];
291 S501386_Data[0xC3];
292 S501386_Data[0x00];
293 }
294 void Letra_J(void){
295 S501386_Data[0xC3];
296 S501386_Data[0xC3];
297 S501386_Data[0xFF];
298 S501386_Data[0xFF];
299 S501386_Data[0x00];
300 S501386_Data[0x00];
301 S501386_Data[0x00];
302 }
303 void Letra_K(void){
304 S501386_Data[0xFF];
305 S501386_Data[0xFF];
306 S501386_Data[0x10];
307 S501386_Data[0xC3];
308 S501386_Data[0x00];
309 S501386_Data[0xC3];
310 S501386_Data[0x00];
311 }
312 void Letra_L(void){
313 S501386_Data[0xFF];
314 S501386_Data[0xFF];
315 S501386_Data[0xC0];
316 S501386_Data[0xC0];
317 S501386_Data[0xC0];
318 S501386_Data[0xC0];
319 S501386_Data[0x00];
320 }
321 void Letra_M(void){
322 S501386_Data[0xFF];
323 S501386_Data[0xFF];
324 S501386_Data[0xC3];
325 S501386_Data[0xC3];
326 S501386_Data[0xC3];
327 S501386_Data[0xFF];
328 S501386_Data[0xFF];
329 S501386_Data[0x00];
330 }
331 }
332 void Letra_N(void){
333 S501386_Data[0xFF];
334 S501386_Data[0xFF];
335 S501386_Data[0x10];
336 S501386_Data[0xC3];
337 S501386_Data[0xFF];
338 S501386_Data[0xFF];
339 S501386_Data[0xFF];
340 S501386_Data[0x00];
341 }
342 void Letra_O(void){
343 S501386_Data[0xFF];
344 S501386_Data[0xFF];
345 S501386_Data[0xC3];
346 S501386_Data[0xC3];
347 S501386_Data[0xFF];
348 S501386_Data[0xFF];
349 S501386_Data[0xFF];
350 S501386_Data[0x00];
351 }
352 void Letra_P(void){
353 S501386_Data[0xFF];
354 S501386_Data[0xFF];
355 S501386_Data[0xC0];
356 S501386_Data[0x10];
357 S501386_Data[0xFF];
358 S501386_Data[0xFF];
359 S501386_Data[0xFF];
360 }
361 void Letra_Q(void){
362 S501386_Data[0xFF];
363 S501386_Data[0xFF];
364 S501386_Data[0x33];
365 S501386_Data[0x63];
366 S501386_Data[0xFF];
367 S501386_Data[0xFF];
368 S501386_Data[0x00];
369 }
370 void Letra_R(void){
371 S501386_Data[0xFF];
372 S501386_Data[0xFF];
373 S501386_Data[0x10];
374 S501386_Data[0x00];
375 S501386_Data[0xFF];
376 S501386_Data[0xFF];
377 S501386_Data[0x00];
378 }
379 void Letra_S(void){
380 S501386_Data[0xFF];
381 S501386_Data[0xFF];
382 S501386_Data[0x00];
383 S501386_Data[0x00];
384 S501386_Data[0xFF];
385 S501386_Data[0xFF];
386 S501386_Data[0x00];
387 }
388 void Letra_T(void){
389 S501386_Data[0x00];
390 S501386_Data[0x00];
391 S501386_Data[0x00];
392 S501386_Data[0x00];
393 S501386_Data[0x00];
394 S501386_Data[0x00];
395 S501386_Data[0x00];
396 }
397 void Letra_U(void){
398 S501386_Data[0xFF];
399 S501386_Data[0xFF];
400 S501386_Data[0xC0];
401 S501386_Data[0xC0];
402 S501386_Data[0xFF];
403 S501386_Data[0xFF];
404 S501386_Data[0x00];
405 }
406 void Letra_V(void){
407 S501386_Data[0xFF];
408 S501386_Data[0xFF];
409 S501386_Data[0x00];
410 S501386_Data[0x00];
411 S501386_Data[0x00];
412 S501386_Data[0xFF];
413 S501386_Data[0xFF];
414 S501386_Data[0x00];
415 }
416 void Letra_W(void){
417 S501386_Data[0xFF];
418 S501386_Data[0xFF];
419 S501386_Data[0x00];
420 S501386_Data[0x00];
421 S501386_Data[0x00];
422 S501386_Data[0x00];
423 S501386_Data[0xFF];
424 S501386_Data[0xFF];
425 S501386_Data[0x00];
426 }
427 }
```



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
FACULTAD DE INGENIERÍA
Microprocesadores y Microcontroladores
Grupo 3

Semestre: 2022-2
Página 29 de 32



```
427 void Letra_X(void){
428   SSD1306_Data(0x3);
429   SSD1306_Data(0x7F);
430   SSD1306_Data(0x7F);
431   SSD1306_Data(0x7F);
432   SSD1306_Data(0x7F);
433   SSD1306_Data(0x3);
434   SSD1306_Data(0x00);
435 }
436 void Letra_Y(void){
437   SSD1306_Data(0x03);
438   SSD1306_Data(0x07);
439   SSD1306_Data(0x07);
440   SSD1306_Data(0x07);
441   SSD1306_Data(0x07);
442   SSD1306_Data(0x03);
443   SSD1306_Data(0x00);
444 }
445 void Letra_Z(void){
446   SSD1306_Data(0xC3);
447   SSD1306_Data(0xC3);
448   SSD1306_Data(0xC3);
449   SSD1306_Data(0xC3);
450   SSD1306_Data(0xC3);
451   SSD1306_Data(0xC3);
452   SSD1306_Data(0x00);
453 }
454 void Espacio(void){
455   SSD1306_Data(0x00);
456   SSD1306_Data(0x00);
457   SSD1306_Data(0x00);
458   SSD1306_Data(0x00);
459   SSD1306_Data(0x00);
460   SSD1306_Data(0x00);
461 }
462 void linea1to(void){
463   SSD1306_Data(0x03);
464   SSD1306_Data(0x03);
465   SSD1306_Data(0x03);
466   SSD1306_Data(0x03);
467   SSD1306_Data(0x03);
468   SSD1306_Data(0x03);
469   SSD1306_Data(0x03);
470   SSD1306_Data(0x03);
471   SSD1306_Data(0x03);
472 }
473 void linea2to(void){
474   SSD1306_Data(0x03);
475   SSD1306_Data(0x03);
476   SSD1306_Data(0x03);
477   SSD1306_Data(0x03);
478   SSD1306_Data(0x03);
479   SSD1306_Data(0x03);
480   SSD1306_Data(0x03);
481   SSD1306_Data(0x03);
482 }
483 void f1to(void){
484   SSD1306_Data(0xC3);
485   SSD1306_Data(0xC3);
486   SSD1306_Data(0xC3);
487   SSD1306_Data(0xC3);
488   SSD1306_Data(0xC3);
489   SSD1306_Data(0xC3);
490   SSD1306_Data(0xC3);
491   SSD1306_Data(0xC3);
492   SSD1306_Data(0xC3);
493 }
494 void f2to(void){
495   SSD1306_Data(0x7F);
496   SSD1306_Data(0x7F);
497   SSD1306_Data(0x7F);
498   SSD1306_Data(0x7F);
499   SSD1306_Data(0x7F);
500   SSD1306_Data(0x7F);
501   SSD1306_Data(0x7F);
502   SSD1306_Data(0x7F);
503   SSD1306_Data(0x7F);
504 }
505 void f3to(void){
506   SSD1306_Data(0x03);
507   SSD1306_Data(0x03);
508   SSD1306_Data(0x03);
509   SSD1306_Data(0x03);
510   SSD1306_Data(0x03);
511   SSD1306_Data(0x03);
512   SSD1306_Data(0x03);
513   SSD1306_Data(0x03);
514   SSD1306_Data(0x03);
515 }
516 void barra(void){
517   SSD1306_Data(0x00);
518   SSD1306_Data(0x00);
519   SSD1306_Data(0x00);
520   SSD1306_Data(0x00);
521   SSD1306_Data(0x00);
522   SSD1306_Data(0x00);
523   SSD1306_Data(0x00);
524   SSD1306_Data(0x00);
525   SSD1306_Data(0x00);
526 }
527 void f4to(void){
528   SSD1306_Data(0x03);
529   SSD1306_Data(0x03);
530   SSD1306_Data(0x07);
531   SSD1306_Data(0x07);
532   SSD1306_Data(0x07);
533   SSD1306_Data(0x03);
534 }
535 void SSD1306_Clear(void){
536   int c;
537   SSD1306_Command(SSD1306_SETPAGER);
538   for(c=0;c<512;c++){
539     SSD1306_Data(0x00);
540   }
541   SSD1306_Command(SSD1306_SETPAGE1);
542   for(c=512;c<1024;c++){
543     SSD1306_Data(0x00);
544   }
545   SSD1306_Command(SSD1306_SETPAGE2);
546   for(c=1024;c<1536;c++){
547     SSD1306_Data(0x00);
548   }
549   SSD1306_Command(SSD1306_SETPAGE3);
550   for(c=1536;c<2048;c++){
551     SSD1306_Data(0x00);
552   }
553   SSD1306_Command(SSD1306_SETPAGE4);
554   for(c=2048;c<2560;c++){
555     SSD1306_Data(0x00);
556   }
557 }
558 void SSD1306_SCROLLINGL(void){
559   SSD1306_Command(SSD1306_SCROLLINGL_);
560   SSD1306_Command(0x00);
561   SSD1306_Command(0x00);
562   SSD1306_Command(0x00);
563   SSD1306_Command(0x00);
564   SSD1306_Command(0x00);
565   SSD1306_Command(0x00);
566   SSD1306_Command(0x00);
567   SSD1306_Command(0x00);
568   SSD1306_Command(0x00);
569 }
570 void SSD1306_SCROLLINGR(void){
571   SSD1306_Command(SSD1306_SCROLLINGR_);
572   SSD1306_Command(0x00);
573   SSD1306_Command(0x00);
574   SSD1306_Command(0x00);
575   SSD1306_Command(0x00);
576   SSD1306_Command(0x00);
577   SSD1306_Command(0x00);
578   SSD1306_Command(0x00);
579   SSD1306_Command(0x00);
580   SSD1306_Command(0x00);
581 }
582 void SSD1306_WriteChar(char c){
583   switch (c){
584     case 'A':
585       Letra_A();
586       break;
587     case 'B':
588       Letra_B();
589       break;
590     case 'C':
591       Letra_C();
592       break;
593     case 'D':
594       Letra_D();
595       break;
596     case 'E':
597       Letra_E();
598       break;
599     case 'F':
600       Letra_F();
601       break;
602     case 'G':
603       Letra_G();
604       break;
605     case 'H':
606       Letra_H();
607       break;
608     case 'I':
609       Letra_I();
610       break;
611     case 'J':
612       Letra_J();
613       break;
614     case 'K':
615       Letra_K();
616       break;
617     case 'L':
618       Letra_L();
619       break;
620     case 'M':
621       Letra_M();
622       break;
623     case 'N':
624       Letra_N();
625       break;
626     case 'O':
627       Letra_O();
628       break;
629     case 'P':
630       Letra_P();
631       break;
632     case 'Q':
633       Letra_Q();
634       break;
635     case 'R':
636       Letra_R();
637       break;
638     case 'S':
639       Letra_S();
640       break;
641     case 'T':
642       Letra_T();
643       break;
644     case 'U':
645       Letra_U();
646       break;
647     case 'V':
648       Letra_V();
649       break;
650     case 'W':
651       Letra_W();
652       break;
653     case 'X':
654       Letra_X();
655       break;
656     case 'Y':
657       Letra_Y();
658       break;
659     case 'Z':
660       Letra_Z();
661       break;
662     case ' ':
663       Espacio();
664       break;
665     case '-':
666       barra();
667       break;
668     case '_':
669       f1to();
670       break;
671     case '=':
672       f2to();
673       break;
674     case '+':
675       f3to();
676       break;
677     case '*':
678       f4to();
679       break;
680     case '<':
681       SSD1306_SCROLLINGL();
682       break;
683     case '>':
684       SSD1306_SCROLLINGR();
685       break;
686     case 'c':
687       SSD1306_Clear();
688       break;
689     case 'l':
690       SSD1306_SCROLLINGL();
691       break;
692     case 'r':
693       SSD1306_SCROLLINGR();
694       break;
695     case 'A':
696       Letra_A();
697       break;
698     case 'B':
699       Letra_B();
700       break;
701     case 'C':
702       Letra_C();
703       break;
704     case 'D':
705       Letra_D();
706       break;
707     case 'E':
708       Letra_E();
709       break;
710     case 'F':
711       Letra_F();
712       break;
713     case 'G':
714       Letra_G();
715       break;
716     case 'H':
717       Letra_H();
718       break;
719     case 'I':
720       Letra_I();
721       break;
722     case 'J':
723       Letra_J();
724       break;
725     case 'K':
726       Letra_K();
727       break;
728     case 'L':
729       Letra_L();
730       break;
731     case 'M':
732       Letra_M();
733       break;
734     case 'N':
735       Letra_N();
736       break;
737     case 'O':
738       Letra_O();
739       break;
740     case 'P':
741       Letra_P();
742       break;
743     case 'Q':
744       Letra_Q();
745       break;
746     case 'R':
747       Letra_R();
748       break;
749     case 'S':
750       Letra_S();
751       break;
752     case 'T':
753       Letra_T();
754       break;
755     case 'U':
756       Letra_U();
757       break;
758     case 'V':
759       Letra_V();
760       break;
761     case 'W':
762       Letra_W();
763       break;
764     case 'X':
765       Letra_X();
766       break;
767     case 'Y':
768       Letra_Y();
769       break;
770     case 'Z':
771       Letra_Z();
772       break;
773     case ' ':
774       Espacio();
775       break;
776     case '-':
777       barra();
778       break;
779     case '_':
780       f1to();
781       break;
782     case '=':
783       f2to();
784       break;
785     case '+':
786       f3to();
787       break;
788     case '*':
789       f4to();
790       break;
791     case '<':
792       SSD1306_SCROLLINGL();
793       break;
794     case '>':
795       SSD1306_SCROLLINGR();
796       break;
797     case 'c':
798       SSD1306_Clear();
799       break;
800     case 'l':
801       SSD1306_SCROLLINGL();
802       break;
803     case 'r':
804       SSD1306_SCROLLINGR();
805       break;
806     case 'A':
807       Letra_A();
808       break;
809     case 'B':
810       Letra_B();
811       break;
812     case 'C':
813       Letra_C();
814       break;
815     case 'D':
816       Letra_D();
817       break;
818     case 'E':
819       Letra_E();
820       break;
819     case 'F':
821       Letra_F();
822       break;
823     case 'G':
824       Letra_G();
825       break;
826     case 'H':
827       Letra_H();
828       break;
829     case 'I':
830       Letra_I();
831       break;
832     case 'J':
833       Letra_J();
834       break;
835     case 'K':
836       Letra_K();
837       break;
838     case 'L':
839       Letra_L();
840       break;
839     case 'M':
841       Letra_M();
842       break;
843     case 'N':
844       Letra_N();
845       break;
846     case 'O':
847       Letra_O();
848       break;
847     case 'P':
849       Letra_P();
850       break;
851     case 'Q':
852       Letra_Q();
853       break;
852     case 'R':
854       Letra_R();
855       break;
853     case 'S':
856       Letra_S();
857       break;
854     case 'T':
858       Letra_T();
859       break;
855     case 'U':
860       Letra_U();
861       break;
856     case 'V':
862       Letra_V();
863       break;
857     case 'W':
864       Letra_W();
865       break;
858     case 'X':
866       Letra_X();
867       break;
859     case 'Y':
868       Letra_Y();
869       break;
860     case 'Z':
869       Letra_Z();
870       break;
861     case ' ':
870       Espacio();
871       break;
862     case '-':
871       barra();
872       break;
863     case '_':
872       f1to();
873       break;
864     case '=':
873       f2to();
874       break;
865     case '+':
874       f3to();
875       break;
866     case '*':
875       f4to();
876       break;
867     case '<':
876       SSD1306_SCROLLINGL();
877       break;
868     case '>':
877       SSD1306_SCROLLINGR();
878       break;
869     case 'c':
878       SSD1306_Clear();
879       break;
870     case 'l':
879       SSD1306_SCROLLINGL();
880       break;
871     case 'r':
880       SSD1306_SCROLLINGR();
881       break;
872     case 'A':
881       Letra_A();
882       break;
873     case 'B':
882       Letra_B();
883       break;
874     case 'C':
883       Letra_C();
884       break;
875     case 'D':
884       Letra_D();
885       break;
876     case 'E':
885       Letra_E();
886       break;
877     case 'F':
886       Letra_F();
887       break;
878     case 'G':
887       Letra_G();
888       break;
879     case 'H':
888       Letra_H();
889       break;
880     case 'I':
889       Letra_I();
890       break;
881     case 'J':
890       Letra_J();
891       break;
882     case 'K':
891       Letra_K();
892       break;
883     case 'L':
892       Letra_L();
893       break;
884     case 'M':
893       Letra_M();
894       break;
885     case 'N':
894       Letra_N();
895       break;
886     case 'O':
895       Letra_O();
896       break;
887     case 'P':
896       Letra_P();
897       break;
888     case 'Q':
897       Letra_Q();
898       break;
889     case 'R':
898       Letra_R();
899       break;
890     case 'S':
899       Letra_S();
900       break;
891     case 'T':
900       Letra_T();
901       break;
892     case 'U':
901       Letra_U();
902       break;
893     case 'V':
902       Letra_V();
903       break;
894     case 'W':
903       Letra_W();
904       break;
895     case 'X':
904       Letra_X();
905       break;
896     case 'Y':
905       Letra_Y();
906       break;
897     case 'Z':
906       Letra_Z();
907       break;
898     case ' ':
907       Espacio();
908       break;
899     case '-':
908       barra();
909       break;
900     case '_':
909       f1to();
910       break;
901     case '=':
910       f2to();
911       break;
902     case '+':
911       f3to();
912       break;
903     case '*':
912       f4to();
913       break;
904     case '<':
913       SSD1306_SCROLLINGL();
914       break;
905     case '>':
914       SSD1306_SCROLLINGR();
915       break;
906     case 'c':
915       SSD1306_Clear();
916       break;
907     case 'l':
916       SSD1306_SCROLLINGL();
917       break;
908     case 'r':
917       SSD1306_SCROLLINGR();
918       break;
909     case 'A':
918       Letra_A();
919       break;
910     case 'B':
919       Letra_B();
920       break;
911     case 'C':
920       Letra_C();
921       break;
912     case 'D':
921       Letra_D();
922       break;
913     case 'E':
922       Letra_E();
923       break;
914     case 'F':
923       Letra_F();
924       break;
915     case 'G':
924       Letra_G();
925       break;
916     case 'H':
925       Letra_H();
926       break;
917     case 'I':
926       Letra_I();
927       break;
918     case 'J':
927       Letra_J();
928       break;
919     case 'K':
928       Letra_K();
929       break;
920     case 'L':
929       Letra_L();
930       break;
921     case 'M':
930       Letra_M();
931       break;
922     case 'N':
931       Letra_N();
932       break;
923     case 'O':
932       Letra_O();
933       break;
924     case 'P':
933       Letra_P();
934       break;
925     case 'Q':
934       Letra_Q();
935       break;
926     case 'R':
935       Letra_R();
936       break;
927     case 'S':
936       Letra_S();
937       break;
928     case 'T':
937       Letra_T();
938       break;
929     case 'U':
938       Letra_U();
939       break;
930     case 'V':
939       Letra_V();
940       break;
931     case 'W':
940       Letra_W();
941       break;
932     case 'X':
941       Letra_X();
942       break;
933     case 'Y':
942       Letra_Y();
943       break;
934     case 'Z':
943       Letra_Z();
944       break;
935     case ' ':
944       Espacio();
945       break;
936     case '-':
945       barra();
946       break;
937     case '_':
946       f1to();
947       break;
938     case '=':
947       f2to();
948       break;
939     case '+':
948       f3to();
949       break;
940     case '*':
949       f4to();
950       break;
941     case '<':
950       SSD1306_SCROLLINGL();
951       break;
942     case '>':
951       SSD1306_SCROLLINGR();
952       break;
943     case 'c':
952       SSD1306_Clear();
953       break;
944     case 'l':
953       SSD1306_SCROLLINGL();
954       break;
945     case 'r':
954       SSD1306_SCROLLINGR();
955       break;
946     case 'A':
955       Letra_A();
956       break;
947     case 'B':
956       Letra_B();
957       break;
948     case 'C':
957       Letra_C();
958       break;
949     case 'D':
958       Letra_D();
959       break;
950     case 'E':
959       Letra_E();
960       break;
951     case 'F':
960       Letra_F();
961       break;
952     case 'G':
961       Letra_G();
962       break;
953     case 'H':
962       Letra_H();
963       break;
954     case 'I':
963       Letra_I();
964       break;
955     case 'J':
964       Letra_J();
965       break;
956     case 'K':
965       Letra_K();
966       break;
957     case 'L':
966       Letra_L();
967       break;
958     case 'M':
967       Letra_M();
968       break;
959     case 'N':
968       Letra_N();
969       break;
960     case 'O':
969       Letra_O();
970       break;
961     case 'P':
970       Letra_P();
971       break;
962     case 'Q':
971       Letra_Q();
972       break;
963     case 'R':
972       Letra_R();
973       break;
964     case 'S':
973       Letra_S();
974       break;
965     case 'T':
974       Letra_T();
975       break;
966     case 'U':
975       Letra_U();
976       break;
967     case 'V':
976       Letra_V();
977       break;
968     case 'W':
977       Letra_W();
978       break;
969     case 'X':
978       Letra_X();
979       break;
970     case 'Y':
979       Letra_Y();
980       break;
971     case 'Z':
980       Letra_Z();
981       break;
972     case ' ':
981       Espacio();
982       break;
973     case '-':
982       barra();
983       break;
974     case '_':
983       f1to();
984       break;
975     case '=':
984       f2to();
985       break;
976     case '+':
985       f3to();
986       break;
977     case '*':
986       f4to();
987       break;
978     case '<':
987       SSD1306_SCROLLINGL();
988       break;
979     case '>':
988       SSD1306_SCROLLINGR();
989       break;
980     case 'c':
989       SSD1306_Clear();
990       break;
981     case 'l':
990       SSD1306_SCROLLINGL();
991       break;
982     case 'r':
991       SSD1306_SCROLLINGR();
992       break;
983     case 'A':
992       Letra_A();
993       break;
984     case 'B':
993       Letra_B();
994       break;
985     case 'C':
994       Letra_C();
995       break;
986     case 'D':
995       Letra_D();
996       break;
987     case 'E':
996       Letra_E();
997       break;
988     case 'F':
997       Letra_F();
998       break;
989     case 'G':
998       Letra_G();
999       break;
990     case 'H':
999       Letra_H();
1000       break;
991     case 'I':
1000       Letra_I();
1001       break;
992     case 'J':
1001       Letra_J();
1002       break;
993     case 'K':
1002       Letra_K();
1003       break;
994     case 'L':
1003       Letra_L();
1004       break;
995     case 'M':
1004       Letra_M();
1005       break;
996     case 'N':
1005       Letra_N();
1006       break;
997     case 'O':
1006       Letra_O();
1007       break;
998     case 'P':
1007       Letra_P();
1008       break;
999     case 'Q':
1008       Letra_Q();
1009       break;
1000     case 'R':
1009       Letra_R();
1010       break;
1001     case 'S':
1010       Letra_S();
1011       break;
1002     case 'T':
1011       Letra_T();
1012       break;
1003     case 'U':
1012       Letra_U();
1013       break;
1004     case 'V':
1013       Letra_V();
1014       break;
1005     case 'W':
1014       Letra_W();
1015       break;
1006     case 'X':
1015       Letra_X();
1016       break;
1007     case 'Y':
1016       Letra_Y();
1017       break;
1008     case 'Z':
1017       Letra_Z();
1018       break;
1009     case ' ':
1018       Espacio();
1019       break;
1010     case '-':
1019       barra();
1020       break;
1011     case '_':
1020       f1to();
1021       break;
1012     case '=':
1021       f2to();
1022       break;
1013     case '+':
1022       f3to();
1023       break;
1014     case '*':
1023       f4to();
1024       break;
1015     case '<':
1024       SSD1306_SCROLLINGL();
1025       break;
1016     case '>':
1025       SSD1306_SCROLLINGR();
1026       break;
1017     case 'c':
1026       SSD1306_Clear();
1027       break;
1018     case 'l':
1027       SSD1306_SCROLLINGL();
1028       break;
1019     case 'r':
1028       SSD1306_SCROLLINGR();
1029       break;
1020     case 'A':
1029       Letra_A();
1030       break;
1021     case 'B':
1030       Letra_B();
1031       break;
1022     case 'C':
1031       Letra_C();
1032       break;
1023     case 'D':
1032       Letra_D();
1033       break;
1024     case 'E':
1033       Letra_E();
1034       break;
1025     case 'F':
1034       Letra_F();
1035       break;
1026     case 'G':
1035       Letra_G();
1036       break;
1027     case 'H':
1036       Letra_H();
1037       break;
1028     case 'I':
1037       Letra_I();
1038       break;
1029     case 'J':
1038       Letra_J();
1039       break;
1030     case 'K':
1039       Letra_K();
1040       break;
1031     case 'L':
1040       Letra_L();
1041       break;
1032     case 'M':
1041       Letra_M();
1042       break;
1033     case 'N':
1042       Letra_N();
1043       break;
1034     case 'O':
1043       Letra_O();
1044       break;
1035     case 'P':
1044       Letra_P();
1045       break;
1036     case 'Q':
1045       Letra_Q();
1046       break;
1037     case 'R':
1046       Letra_R();
1047       break;
1038     case 'S':
1047       Letra_S();
1048       break;
1039     case 'T':
1048       Letra_T();
1049       break;
1040     case 'U':
1049       Letra_U();
1050       break;
1041     case 'V':
1050       Letra_V();
1051       break;
1042     case 'W':
1051       Letra_W();
1052       break;
1043     case 'X':
1052       Letra_X();
1053       break;
1044     case 'Y':
1053       Letra_Y();
1054       break;
1045     case 'Z':
1054       Letra_Z();
1055       break;
1046     case ' ':
1055       Espacio();
1056       break;
1047     case '-':
1056       barra();
1057       break;
1048     case '_':
1057       f1to();
1058       break;
1049     case '=':
1058       f2to();
1059       break;
1050     case '+':
1059       f3to();
1060       break;
1051     case '*':
1060       f4to();
1061       break;
1052     case '<':
1061       SSD1306_SCROLLINGL();
1062       break;
1053     case '>':
1062       SSD1306_SCROLLINGR();
1063       break;
1054     case 'c':
1063       SSD1306_Clear();
1064       break;
1055     case 'l':
1064       SSD1306_SCROLLINGL();
1065       break;
1056     case 'r':
1065       SSD1306_SCROLLINGR();
1066       break;
1057     case 'A':
1066       Letra_A();
1067       break;
1058     case 'B':
1067       Letra_B();
1068       break;
1059     case 'C':
1068       Letra_C();
1069       break;
1060     case 'D':
1069       Letra_D();
1070       break;
1061     case 'E':
1070       Letra_E();
1071       break;
1062     case 'F':
1071       Letra_F();
1072       break;
1063     case 'G':
1072       Letra_G();
1073       break;
1064     case 'H':
1073       Letra_H();
1074       break;
1065     case 'I':
1074       Letra_I();
1075       break;
1066     case 'J':
1075       Letra_J();
1076       break;
1067     case 'K':
1076       Letra_K();
1077       break;
1068     case 'L':
1077       Letra_L();
1078       break;
1069     case 'M':
1078       Letra_M();
1079       break;
1070     case 'N':
1079       Letra_N();
1080       break;
1071     case 'O':
1080       Letra_O();
1081       break;
1072     case 'P':
1081       Letra_P();
1082       break;
1073     case 'Q':
1082       Letra_Q();
1083       break;
1074     case 'R':
1083       Letra_R();
1084       break;
1075     case 'S':
1084       Letra_S();
1085       break;
1076     case 'T':
1085       Letra_T();
1086       break;
1077     case 'U':
1086       Letra_U();
1087       break;
1078     case 'V':
1087       Letra_V();
1088       break;
1079     case 'W':
1088       Letra_W();
1089       break;
1080     case 'X':
1089       Letra_X();
1090       break;
1081     case 'Y':
1090       Letra_Y();
1091       break;
1082     case 'Z':
1091       Letra_Z();
1092       break;
1083     case ' ':
1092       Espacio();
1093       break;
1084     case '-':
1093       barra();
1094       break;
1085     case '_':
1094       f1to();
1095       break;
1086     case '=':
1095       f2to();
1096       break;
1087     case '+':
1096       f3to();
1097       break;
1088     case '*':
1097       f4to();
1098       break;
1089     case '<':
1098       SSD1306_SCROLLINGL();
1099       break;
1090     case '>':
1099       SSD1306_SCROLLINGR();
1100       break;
1091     case 'c':
1099       SSD1306_Clear();
1100       break;
1092     case 'l':
1100       SSD1306_SCROLLINGL();
1101       break;
1093     case 'r':
1101       SSD1306_SCROLLINGR();
1102       break;
1094     case 'A':
1102       Letra_A();
1103       break;
1095     case 'B':
1103       Letra_B();
1104       break;
1096     case 'C':
1104       Letra_C();
1105       break;
1097     case 'D':
1105       Letra_D();
1106       break;
1098     case 'E':
1106       Letra_E();
1107       break;
1099     case 'F':
1107       Letra_F();
1108       break;
1100     case 'G':
1108       Letra_G();
1109       break;
1101     case 'H':
1109       Letra_H();
1110       break;
1102     case 'I':
1110       Letra_I();
1111       break;
1103     case 'J':
1111       Letra_J();
1112       break;
1104     case 'K':
1112       Letra_K();
1113       break;
1105     case 'L':
1113       Letra_L();
1114       break;
1106     case 'M':
1114       Letra_M();
1115       break;
1107     case 'N':
1115       Letra_N();
1116       break;
1108     case 'O':
1116       Letra_O();
1117       break;
1109     case 'P':
1117       Letra_P();
1118       break;
1110     case 'Q':
1118       Letra_Q();
1119       break;
1111     case 'R':
1119       Letra_R();
1120       break;
1112     case 'S':
1120       Letra_S();
1121       break;
1113     case 'T':
1121       Letra_T();
1122       break;
1114     case 'U':
1122       Letra_U();
1123       break;
1115     case 'V':
1123       Letra_V();
1124       break;
1116     case 'W':
1124       Letra_W();
1125       break;
1117     case 'X':
1125       Letra_X();
1126       break;
1118     case 'Y':
1126       Letra_Y();
1127       break;
1119     case 'Z':
1127       Letra_Z();
1128       break;
1120     case ' ':
1128       Espacio();
1129       break;
1121     case '-':
1129       barra();
1130       break;
1122     case '_':
1130       f1to();
1131       break;
1123     case '=':
1131       f2to();
1132       break;
1124     case '+':
1132       f3to();
1133       break;
1125     case '*':
1133       f4to();
1134       break;
1126     case '<':
1134       SSD1306_SCROLLINGL();
1135       break;
1127     case '>':
1135       SSD1306_SCROLLINGR();
1136       break;
1128     case 'c':
1136       SSD1306_Clear();
1137       break;
1129     case 'l':
1137       SSD1306_SCROLLINGL();
1138       break;
1
```



```
661     Letra_S();
662     break;
663     case 'T':
664         Letra_T();
665         break;
666     case 'U':
667         Letra_U();
668         break;
669     case 'V':
670         Letra_V();
671         break;
672     case 'X':
673         Letra_X();
674         break;
675     case 'Y':
676         Letra_Y();
677         break;
678     case 'Z':
679         Letra_Z();
680         break;
681     case '-':
682         LineaAlta();
683         break;
684     case ')':
685         flechaF2();
686         break;
687     case '(':
688         flechaF1();
689         break;
690     case '_':
691         LineaBaja();
692         break;
693     case '[':
694         flechaB1();
695         break;
696     case ']':
697         flechaB2();
698         break;
699     case '|':
700         barra();
701         break;
702     case '^':
703         flechaV1();
704         break;
705     case '*':
706         flechaV2();
707         break;
708     case '+':
709         SS();
710         break;
711     case ' ':
712         Espacio();
713         break;
714 }
715 }
716 }
717 }
718 }
719
720 void SSD1306_WriteString(unsigned char *cadena)
721 {
722     unsigned int i = 0;
723     unsigned int l = 0;
724     while (cadena[i] != 0)
725     {
726         SSD1306_WriteChar(i);
727         i++;
728         l = cadena[i];
729     }
730 }
731
732
733 /*****
734 LIESE LIESE LIESE
735 *****/
736
737 #endif /* SSD1306_H_ */
```

Ilustración 31 Código SSD1306.h

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

Línea alta	Línea baja	FlechaF1
0x03 0x03 0x03 0x03 0x03 0x03 0x03	0xC0 0xC0 0xC0 0xC0 0xC0 0xC0 0xC0	0xFE 0xFC 0xF8 0xF0 0xE0 0xC0
FlechaF2	FlechaB1	flechaB2
0x7F 0x3F 0x1F 0x0F 0x07 0x03 0x01	0x80 0xC0 0xE0 0xF0 0xF8 0xFC 0xFE	0x01 0x03 0x07 0x0F 0x1F 0x3F 0x7F
barra	flechaV1	flechaV2
0x00 0x00 0x00 0x00 0x00 0xFF 0xFF	0x01 0x03 0x07 0x0F 0x07 0x03 0x01	0x80 0xC0 0xE0 0xF0 0xE0 0xC0 0x80

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&hvrnd=10607578214585673784&hvpone=&hvptwo=&hvmqmt=&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&hvrnd=15584600922067333176&hvpone=&hvptwo=&hvmqmt=&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](https://www.agelectronica.com/detalle.php?p=MCP%2041010-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/>