ESCUELA POLITÉCNICA

UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

Grado en Ingeniería de Sonido e Imagen en Telecomunicaciones

Trabajo Fin de Grado

Adaptación de control de silla de ruedas para mejora de su ergonomía y funcionalidad.

David Flores Román

Junio 2021

ESCUELA POLITÉCNICA

UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

Grado en Ingeniería de Sonido e Imagen en Telecomunicaciones

Trabajo Fin de Grado

Adaptación de control de silla de ruedas para mejora de su ergonomía y funcionalidad

Autor: David Flores Román.

Tutor: Antonio Gordillo Guerrero.

Tribunal calificador

Presidente: Horacio González Velasco

Secretario: Ramón Gallardo Caballero

Vocal: Marino Linaje Trigueros

A todos aquellos que nunca dejaron de creer y me guiaron hasta aquí.

ÍNDICE GENERAL DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

En el presente trabajo se ha desarrollado el control de una silla de ruedas, para tratar de mejorar la ergonomía de ésta y el funcionamiento con respecto a una convencional, de la forma más económica posible y utilizando hardware y software libre. Se ha desarrollado tanto para tener un control físico por parte del usuario, como a través de comunicación inalámbrica vía Bluetooth, por parte de un posible cuidador.

Se ha construido un primer prototipo a pequeña escala, usando un pequeño coche de pruebas, que dio paso al diseño y montaje real sobre una silla de ruedas comercial estándar.

Se ha obtenido un sistema básico de movilidad para el uso cotidiano de personas con discapacidad con una silla de ruedas real.

1. INTRODUCCIÓN

Según el Instituto Nacional de Estadística (INE) [1] hay un gran número de personas con discapacidad de las cuales, un porcentaje significativo poseen una movilidad reducida y que necesitan el uso de una silla de ruedas. Solo en 2008 se registraron 3,85 millones de personas con discapacidad de las cuales el 67,2% presentaban problemas de movilidad.

Algunas de estas personas no son capaces de usar correctamente los mandos de una silla de ruedas convencional.

1.1. HISTORIA DE LA SILLA DE RUEDAS

¿Cómo surgió la silla de ruedas y cómo ha ido evolucionando hasta llegar a la silla que se ha desarrollado en este trabajo y posibles sillas que se están desarrollando para un futuro mejor [2]?

Aunque no se sabe a ciencia cierta cuando fue inventada la silla de ruedas, la primera que se conoce fue inventada para el rey Felipe II de España en 1595.

Algunos años después, el relojero Stephen Farffler desarrolló una silla de ruedas constituida por un chasis y que permitía impulsarse con los brazos, lo cual hacía a la persona que lo usara algo más autosuficiente.

En 1783, el inglés John Dawson, fabricante, desarrolló una silla de ruedas con dos ruedas grandes atrás y una pequeña en la parte delantera, pero no era nada cómoda para el usuario y tuvo que ser mejorada.

Más tarde se inventaría la silla de ruedas plegable por Harry Jennings, la cual es un referente, ya que todos los demás diseños de sillas de ruedas parten de esta base. La silla es muy ligera, medianamente cómoda y fácil de transportar. Su forma puede verse en la Ilustración 1.

Ilustración 1: Silla de ruedas de Jennings.

A pesar del avance de las sillas de ruedas, estas tenían que ser impulsadas por el propio usuario de alguna manera o a través de otra persona que ayudara al discapacitado, lo cual hacía depender de otros o de la propia fuerza física para poder moverse.

El inventor George Klein desarrolló una silla de ruedas motorizada para ayudar a los veteranos parapléjicos que volvían de la Segunda Guerra Mundial, que contaba con un joystick y un sistema preciso para el manejo de la silla de ruedas. Con su evolución tenemos sillas aún más ligeras, ya que las baterías se han ido mejorando desde entonces, y mucho más precisas [3], ya que ha mejorado su control eléctronico.

Este tipo de silla es el que se presenta en este trabajo, aunque hoy día se están desarrollando tipos de sillas de ruedas motorizadas capaces incluso de subir escaleras, como la que se encuentra en desarrollo por un equipo de estudiantes de ingeniería mecánica de la universidad de Zurich (Scewo) [4].

1.2. DISCAPACIDADES

¿Qué es una discapacidad?

Según la RAE [5] una discapacidad es la situación de la persona que por sus condiciones físicas o mentales duraderas se enfrenta a determinados obstáculos que le impiden de alguna manera su participación en la sociedad.

Este trabajo se centra en las discapacidades de condición física y que impiden a las personas poder moverse por su propio pie, dependiendo de una herramienta que lo haga por ellos, por ejemplo, una silla de ruedas.

Tipos de discapacidad física:

- *Monoplejía:*

Parálisis total o parcial de una de las extremidades del cuerpo, ya sea en un músculo o grupo de músculos del brazo o la pierna [6].

- *Paraplejía:*

Parálisis total o parcial de la zona inferior del cuerpo, debido a una lesión medular y que afecta básicamente a las piernas y los pies, por lo que las personas que padecen de paraplejía no pueden volver a caminar [7].

- *Tetraplejía:*

Parálisis total o parcial de brazos y piernas de la persona que la padece, debido a un daño en la medula espinal o bien por una enfermedad que afectan a las neuronas encargadas de realizar ese tipo de movimientos [8].

- *Distrofia muscular:*

Enfermedad o grupo de enfermedades que provocan una debilidad y una pérdida de masa muscular, [9] lo que hace que el paciente pierda fluidez a la hora de realizar un movimiento.

1.3. PROBLEMA PROPUESTO

Este trabajo nace de un problema real propuesto por la asociación placentina PLACEAT (ONG que lucha a favor de las personas con discapacidad intelectual), a través de su trabajador José Manuel Abolafio [10].

Su problema consiste en una silla de ruedas en la que el usuario, perteneciente a esta asociación, padece una movilidad muy reducida en los brazos, por lo tanto, a la hora de presionar los botones del mando de control, también accionaba sin querer el joystick de la silla, provocando movimientos no deseados en ésta.

La asociación proporcionó una silla de ruedas relativamente antigua, que contaba con el mando de control y los drivers de los motores, los motores y las baterías.

De todo esto, solo se ha utilizado la silla y los motores, puesto que las baterías no funcionaban y el mando y drivers originales se perdieron durante la pandemia de COVID-19.

A raíz de esto, el trabajo pasa de querer simplemente cambiar la estructura del joystick del mando de control, al desarrollo total de la electrónica de una silla que pueda servir para distintos tipos de discapacidad y que no suponga un gran impacto económico.

1.4. MOTIVACIÓN

La motivación principal de este trabajo consiste en la contribución con nuestro granito de arena ayudar a las personas con discapacidad.

Además, se desean obtener conocimientos sobre las distintas materias que se van a abordar. A saber:

- Electrónica de potencia.

- Electrónica de control.

- Electromecánica.

- Diseño 3D.

- Programación en C.

2. OBJETIVOS

Como objetivo principal se pretende conseguir desarrollar el sistema de control de una silla de ruedas que resulte económica y que se pueda adaptar a las distintas discapacidades del usuario. En concreto se desea:

- Utilizar un joystick para controlar la velocidad y la dirección de la silla de forma segura.

- Controlar la velocidad de forma eficiente y hacer que los movimientos de la silla sean suaves y fluidos.

- Seleccionar distintas velocidades máximas mediante botones.

- Introducir un indicador de batería mediante LEDs controlados por un botón.

- Introducir un claxon mediante un zumbador controlado por un botón.

Sin embargo, por limitaciones temporales y económicas:

- La silla no tendrá ningún tipo de homologación o certificación.

- Como la silla no se probará con pacientes reales no se podrá determinar el grado de efectividad de ésta.

3. ANTECEDENTES / ESTADO DEL ARTE

3.1. MANDO DE CONTROL DE LA SILLA DE RUEDAS

El mando de control de una silla de ruedas estándar consta de un joystick para el control de la movilidad y de las direcciones que puede tomar el usuario; botones de velocidad, para que la silla no pueda superar ciertos límites dependiendo de la situación en la que se encuentre el usuario; indicador de batería; varios LEDs que se van apagando a medida que las baterías se agotan; y bocina, para evitar posibles accidentes [11]. Los botones del mando van detrás del joystick, lo cual es un problema porque si queremos pulsar un botón, es muy fácil accionar sin querer el joystick. Problema del que se ha hablado anteriormente.

Ilustración 2: Mando de silla estándar

En este trabajo se ha tratado de realizar un mando parecido, de tal manera que los componentes estén distribuidos de una forma más eficiente y seguros para el usuario que vaya a utilizarlo.

Proporcionándole también los mismos botones de velocidad, bocina, indicador de batería y joystick, lo cual podría ser perfectamente ampliable en un futuro.

En cuanto al manejo de los motores de la silla, aunque en un principio se había pensado desarrollar solamente el mando de control y utilizar los drivers de los motores originales de la propia silla, finalmente se ha optado por la implementación del sistema de control electrónico de la marcha de los motores (y sus frenos).

4. MATERIAL Y MÉTODO

En primer lugar, se ha planteado un primer prototipo, con un modelo básico de coche pequeño, como se puede ver en la Ilustración 3, en el cual se desarrollan todas las funciones que se quieren implementar en la silla de ruedas. Utilizando como microcontrolador ESP32 [12], que también va a ser usado para la silla de ruedas.

Se trata de hacer funcionar los pequeños motores del coche para que sean dirigidos con un joystick y pueda desempeñar las mismas funciones de velocidad, indicador de batería, bocina, etc.

Esto nos ayudó a entender mejor qué es lo que necesitábamos a la hora de ponernos a trabajar con la silla de ruedas real.

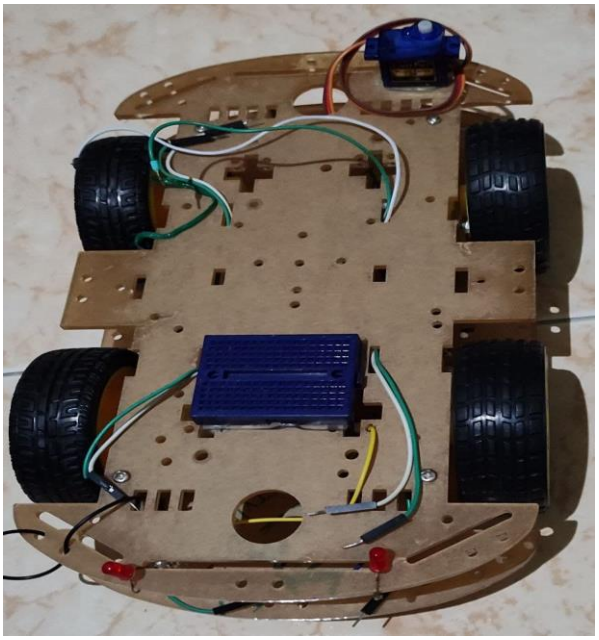

Ilustración 3: Coche de pruebas

Una vez conseguido esto, se pasó al trabajo con silla de ruedas, donde se han cambiado algunos materiales para llevarlo a cabo, ya que los motores son mayores y las baterías de mayor voltaje y amperaje.

4.1. MATERIALES DEL PROTOTIPO

En este apartado se describen todos los materiales que se han utilizado para la implementación del primer prototipo; qué son, cómo funcionan, para qué se han utilizado, etc.

- Placa ESP32 Devkit-32E [12]:

Se ha decidido usar esta placa ya que ocupa menos espacio y es un microcontrolador más potente que Arduino UNO y con la capacidad de proporcionar WiFi y Bluetooth. Además, puede ser programada desde el propio IDE de Arduino con pocas diferencias.

Es el que se va a usar para el mando de control de la silla de ruedas.

Para hacer funcionar ESP32 en el PC se debe instalar anteriormente Python 2.7 y el entorno gráfico de GIT, con el cual se descarga el código necesario de ESP32 en el IDE de Arduino [13].

Características:

Arquitectura de 32bits que cuenta con 36 pines GPIO (General Purpose Input Output), una memoria flash de 16MB, 512KB de memoria RAM y una frecuencia base de procesamiento de 160MHz [14]. Puede verse en la Ilustración 4 su distribución de pines.

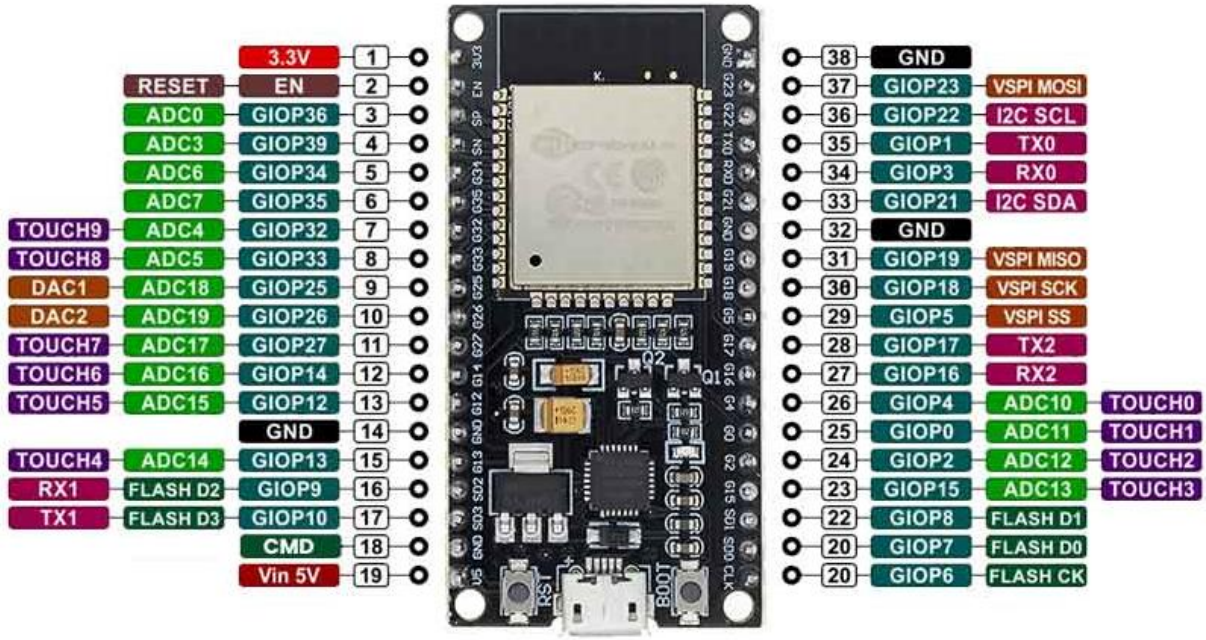

Ilustración 4: Esquema ESP32.

Ilustración 5: Placa ESP32 utilizada.

- **Batería de polímero de litio (LiPo)** [15]:

Batería de 11.1V utilizada para alimentar los motores del coche, usados para las pruebas del prototipo. Su valor real máximo medido con multímetro en cargar máxima es de aproximadamente 12 V. Puede verse en la Ilustración 6.

Ilustración 6: Batería de Litio usada para el prototipo.

Necesita de un cargador específico para baterías LiPo, el cual realiza balances de carga para que todas las celdas estén al mismo voltaje (no sobrepasando nunca los valores de voltaje máximos de cada celda).

Es importante recordar que las baterías de litio pueden ser muy inestables en la carga pudiendo llegar incluso a explotar si no se hace correctamente.

- **LEDs** [16]:

Para el indicador de batería se han utilizado 5 LEDs (Diodo Emisor de Luz): tres de color verde, uno de color amarillo y uno de color rojo, todos de 5mm.

Ilustración 7: Tipo de LEDs utilizados.

Su función consiste en que, una vez que se ha apretado el botón para el indicador de batería se encenderán ciertos LEDs, en función del voltaje que tengan las baterías en ese momento.

- **Resistencias:**

Se han utilizado varias resistencias, ya sean limitadoras para los LEDs, como de Pull Down para los botones, todas de 10kΩ para asegurar los componentes.

- Botones [17]:

Botones utilizados para las velocidades, indicador de batería y bocina.

Con dos posiciones, pulsado y no pulsado, ideales para ser pinchados en una "protoboard".

Ilustración 8: Botones utilizados para el prototipo.

- Chip LM7805 [18]:

Regulador de tensión a 5V utilizado para alimentar el joystick.

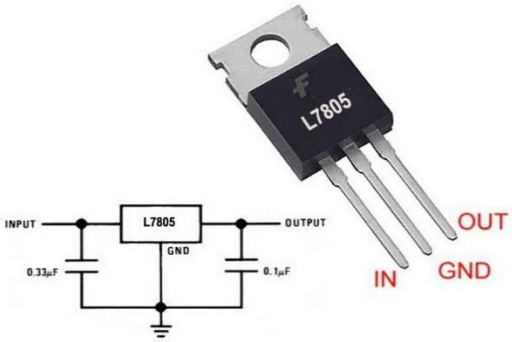

Ilustración 9: Chip LM7805.

- Regulador LM2596s: [19]

Regulador a 5V utilizado para alimentar el ESP32 con la batería de 11.1V

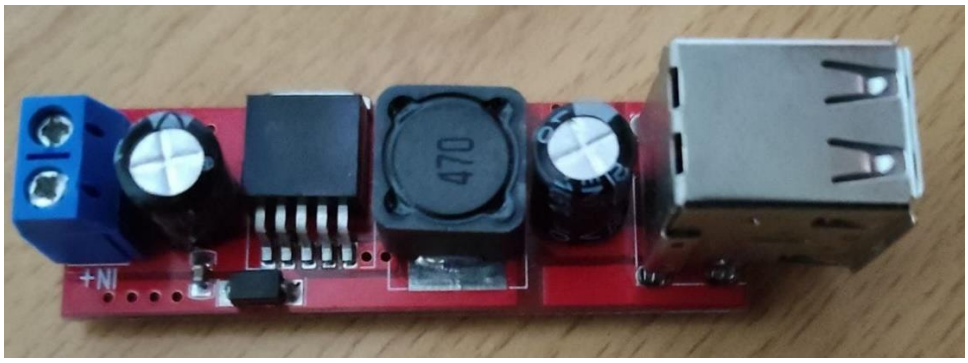

Ilustración 10: Regulador LM2596S.

- Controlador para motores L298N [20]:

Aunque no es el que se va a utilizar para los motores de la silla de ruedas, vale perfectamente para usar en el prototipo y comprobar todo lo necesario en cuanto al comportamiento de la silla.

Cuenta con dos puentes de transistores en H para poder controlar hasta dos motores al mismo tiempo. Solo soporta hasta 2 A, por este motivo no se podrá utilizar para el manejo de los motores de la silla de ruedas, ya que necesitan hasta 9,4 A.

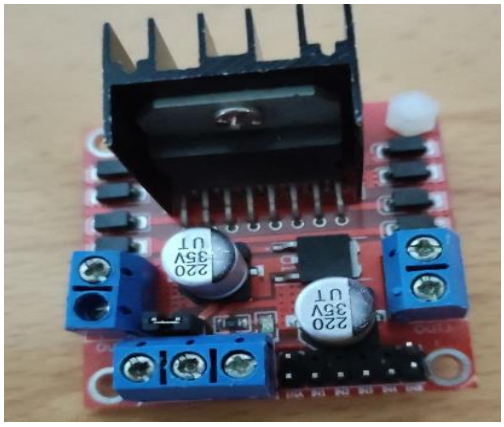

Ilustración 11: Controlador L298N para los motores del prototipo.

- Zumbador [21]:

Buzzer piezoeléctrico estándar de corriente continua, usado como bocina.

- Condensadores:

Para la estabilidad del regulador LM7805, de 0,33 y 0,1µF. Se han usado condensadores de tipo electrolítico.

- Cables:

Cables para la placa de pruebas utilizados para el conexionado con conectores tipo *Dupont.*

- Soldador:

Soldador de hasta 400º utilizado para soldar la placa de prototipos. Modelo JBC CD-2BQF [22].

Ilustración 12: Soldador.

- Joystick [23]:

Modulo joystick, que funciona a 5V, con dos potenciómetros, uno para cada eje, que vuelve a la posición de reposo una vez se deja de mover. Válido para Arduino y para ESP32.

Bastante manejable, aunque no lo suficientemente robusto para ser usado en la silla de ruedas. No obstante, para las pruebas del primer prototipo, es más que suficiente. Incluso se ha usado más adelante para detectar posibles fallos en el código utilizado para el control de la silla de ruedas.

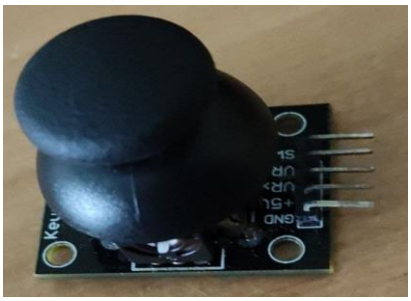

Ilustración 13: Joystick.

- Motores de corriente continua.

Dos motores pequeños DC utilizados para mover las ruedas del coche, convierten la energía eléctrica en mecánica, controlados por el driver L298N. Para cambiar el sentido de giro, hay que cambiar el sentido de la corriente [24]. Solo se han utilizado para las pruebas dos de los cuatro motores que tiene el coche, ya que solo se van a controlar dos motores en la silla de ruedas.

Tabla 1.Presupuesto aproximado del prototipo.

Unidades	**Nombre**	**Precio, Unidad (€)**	**Precio total (€)**
1	ESP32	10	10
1	Batería de polímero de litio	30	30
5	LED	0,77	3,85
14	Resistencia	0,1	1,4
4	Pulsador	0,3	1,2
1	LM7805	0,9	0,9
1	LM2596s	0,4	0,4
1	L298N	1,5	1,5
1	Zumbador	1,5	1,5
2	Condensador	0,7	1,4
-	cable	-	3
1	Coche de pruebas	16	16
1	joystick	1	1
		Presupuesto total:	72,15€

En la tabla 1 puede verse el presupuesto desglosado del prototipo, que asciende a 72,15€.

4.1.1. ESQUEMA DEL PROTOTIPO

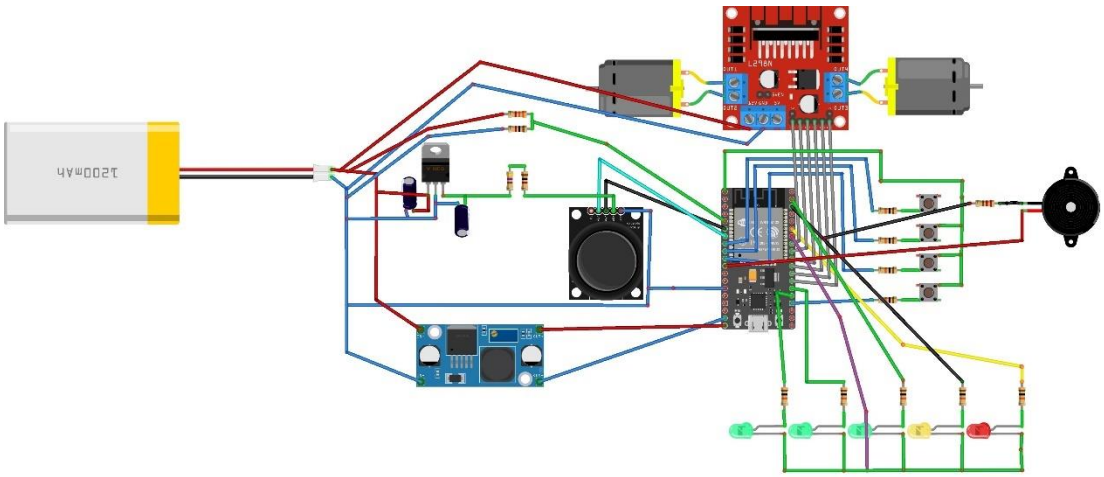

Ilustración 14: Prototipo de coche pequeño.

En la Ilustración 14 puede verse el esquema del circuito utilizado para el prototipo, donde podemos ver: cómo con la batería, a través del regulador LM2596s (en este caso en azul ya que no se encontró para el esquema el mismo utilizado), hace funcionar el ESP32. El regulador LM7805 alimenta el joystick. Como se ha dicho, los condensadores son para darle estabilidad y las dos resistencias, de 470Ω y 1kΩ, se han puesto debido a que los ejes X e Y (cada uno conectado a dos GPIO de la placa ESP32) del joystick no marcaban el valor que deberían, ya que estando en reposo el joystick debe marcar un zona intermedia entre 0 y 4095 y esto no es así debido a que se alimentaba con un poco más de voltaje del necesario, de modo que con las resistencias se conseguía disminuir el voltaje hasta llegar a esa zona intermedia.

- Dos resistencias para hacer un divisor de tensión, una de 3,3kΩ y la otra de 1kΩ, se han elegido estos valores de resistencias para que la tensión que entre en el ESP32 no sea superior a 3,3V ya que es lo máximo que puede soportar un GPIO de la placa. Estos valores están pensados para la batería de 11,1V y por tanto son cambiados para las dos baterías de 12V en serie (24V) que llevará la silla de ruedas.

$$Vout = \frac{R2}{R1 + R2} * Vin$$

$Vout$ es el voltaje de las baterías en voltios, $R1$ es la resistencia de 3,3kΩ, $R2$ es la resistencia de 1kΩ y Vin es el voltaje en voltios que llega al GPIO34 del ESP32.

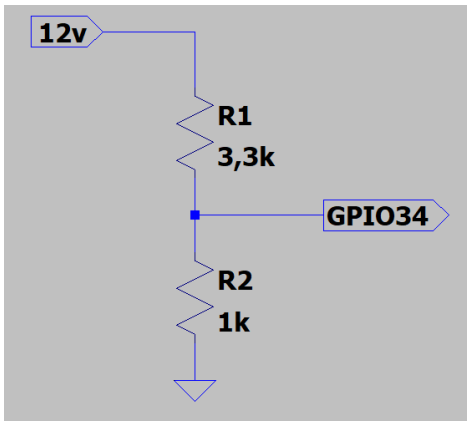

Ilustración 15: Divisor de tensión para el coche de pruebas.

- Cinco LEDs para monitorear el voltaje restante de la batería a través de uno de los botones con el valor que detecta el ESP32 a través del divisor de tensión.

- Dos botones conectados a 3,3V y a los GPIOs de la placa para cambiar la velocidad de la silla de ruedas, con el movimiento del joystick a medida que se va accionando poco a poco, la silla irá más rápido hasta llegar a un límite, con los botones de velocidad tendrá dos límites. Esto se explicará más adelante en el apartado de programación.

- Un botón conectado a 3,3V y a un GPIO para hacer funcionar la bocina, que ayudará al usuario a prevenir posibles accidentes.

- La placa L298N está alimentada por la batería y conectada a la placa ESP32 y a los motores, los cuales están programados a través de los GPIOs de la placa ESP32 para su velocidad y su dirección.

Se ha usado una placa experimental de matriz de anillos de cobre (Perfboard) [25] para soldar los componentes, como puede verse en la Ilustración 16.

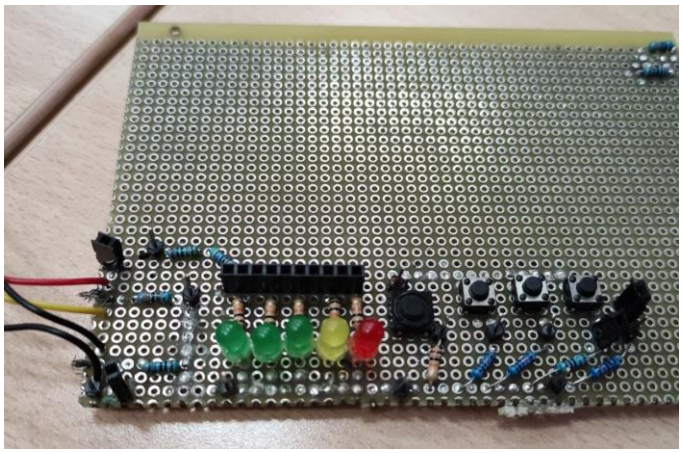

Ilustración 16: Placa experimental tipo Perfboard.

.

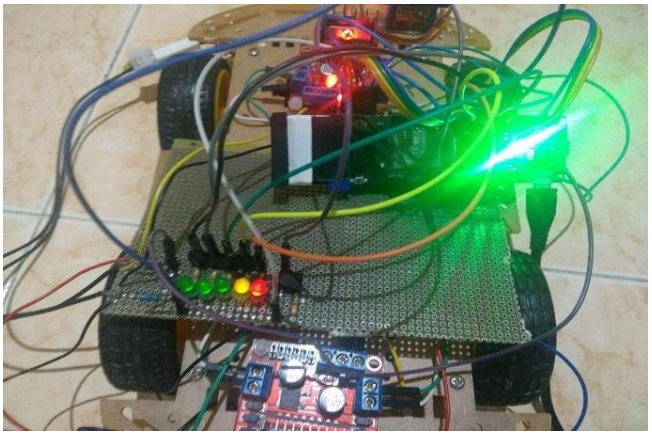

Ilustración 17: Prueba Bluetooth e indicador de batería.

4.2. MATERIALES Y EXPLICACIÓN DE LA SILLA DE RUEDAS REAL

En este apartado se describen los materiales utilizados en la silla de ruedas real y por qué se han sustituido en relación con el prototipo descrito anteriormente.

- **Baterías** [26]:

Adquirimos 2 baterías de 12V y 33Ah de plomo, selladas y de descarga profunda, y que son específicas para el tipo de motores utilizados en sillas de ruedas. Conectadas en serie alimentan los motores a través de los drivers, y alimentan también a todo el circuito de control.

Ilustración 18: Batería con las conexiones.

Podemos analizar brevemente las ventajas y desventajas de las baterías de plomo:

Ventajas:

- Muy baratas con respecto a otras.

- Sencillez a la hora de cargarlas. Solo necesitan un voltaje constante.

- Pueden soportar descargar profunda.

Desventajas:

- Muy pesadas.

- Menor número de cargas con respecto a otras.

- La carga es lenta, entre 6 y 8 horas.

- **Motores** [27]:

Se trata de 2 motores de la marca *Sunrise Medical* de corriente continua que funcionan a unos valores de potencia máximos de 180w y corriente de 9,4A a 24 V, que pueden ver en la Ilustración 19.

Estos motores presentan cuatro conexiones: dos para la bobina que mueve al motor y dos para los frenos eléctricos que llevan incorporados, los cuales siempre están frenando hasta que se alimentan con un voltaje superior a 10V y los cuales se accionarán a través de un módulo de MOSFETs del que se hablará más adelante.

No se han utilizado los drivers originales de la silla para moverlos, sino que se ha decidido utilizar drivers programables también con ESP32, que pueden soportar el voltaje de las baterías y corrientes altas, y que se describen a continuación.

Ilustración 19: Motores utilizados en la silla.

- Drivers BTS7960 [28]:

Controladores para motores de hasta 43A. Puede verse en la Ilustración 20. Se ha utilizado una librería que proporciona Luis Llamas en su página web [29]. Aunque la librería solo sirve para el funcionamiento de un driver, para este trabajo se necesitan dos (uno para cada motor). Se ha hecho una modificación en ésta para que puedan utilizarse los dos drivers. La modificación consistió simplemente en duplicar las funciones de la librería existentes y cambiarles el nombre para que no hubiera ningún conflicto entre variables. Puede verse en el "GitHub" del proyecto [30].

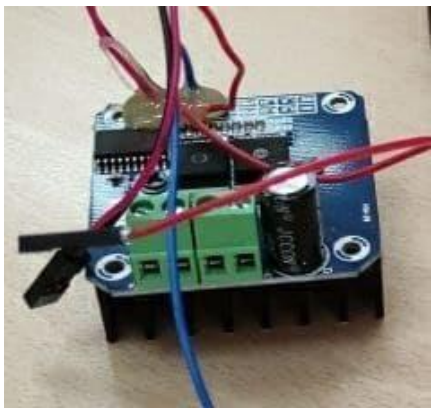

Ilustración 20: Driver del motor.

- Joystick [31]:

Se ha utilizado un mando robusto y estable para mover las ruedas a diferentes velocidades. Se trata de un joystick de 4 ejes y un botón, modelo R400B-M4.

La programación usada es parecida a la que se ha hecho en las pruebas, aunque se han ido añadiendo mejoras, se han cambiado las resistencias para reajustar el valor de salida en el ESP32, una resistencia de 4,7kΩ para un eje y una resistencia de 3,3kΩ para el otro.

Y se ha decidido alimentarlo a través del regulador de 5V que va al ESP32.

Ilustración 21: Joystick robusto.

- Modulo MOSFET de 4 canales [32]:

Se trata de un Módulo de 4 canales (transistores MOSFET IRF540N) con características de conmutación para controlar cargas altas y que permite hasta 33A de corriente y 100V de bloqueo entre drenador y fuente.

Además, cuenta con un chip optoacoplador PS2801 [33] para controlar las señales que le llegan del ESP32 sin dañar éste.

Con éste podremos accionar y bloquear los frenos de los motores de una forma muy sencilla cada vez que los motores se pongan en funcionamiento, es decir, cada vez que se accione el joystick. Para alimentar los frenos se ha usado el regulador ajustable a 10V (LM2596s del que se ha hablado anteriormente), ya que los frenos precisan de menor voltaje en comparación con los motores. Otra de las entradas del módulo también se ha utilizado para accionar la bocina que incorporamos a la silla de ruedas.

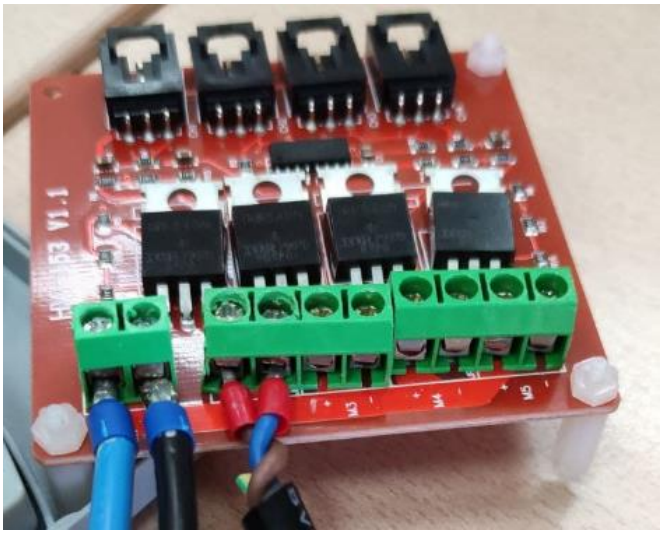

Ilustración 22: Escudo con 4 MOSFETss.

Al principio, nos encontramos con el problema de que las conexiones con las que cuentan estos motores son cuatro, dos para los motores y dos para los frenos. Y se pensaba que los frenos no harían falta, pero al ver que no funcionaban y después de una búsqueda de información se descubrió que era imprescindible usarlos, ya que son frenos eléctricos y necesitan que se les aplique un voltaje para poder funcionar.

Usar los 24V de las baterías solo para los frenos supone un aumento de consumo, por lo tanto, se hizo una prueba con una fuente de alimentación regulable para comprobar qué voltaje mínimo era necesario para accionar los frenos. Se vio que se necesitaban aproximadamente 10V.

- Divisor de tensión.

Para el indicador de batería se ha diseñado un divisor de tensión con una resistencia de 100KΩ y otra de 10KΩ, de esta manera nos aseguramos de que el voltaje que llega al ESP32 no será perjudicial para éste, poniendo además un diodo Zener de 3,4 voltios en paralelo con la resistencia de 10KΩ para garantizar de que no sobrepase dicho voltaje.

$$Vout = \frac{R2}{R1 + R2} * Vin$$

$Vout$ es el voltaje de las baterías en voltios, $R1$ es la resistencia de 100kΩ, $R2$ es la resistencia de 10kΩ y Vin es el voltaje en voltios que llega al GPIO34 del ESP32.

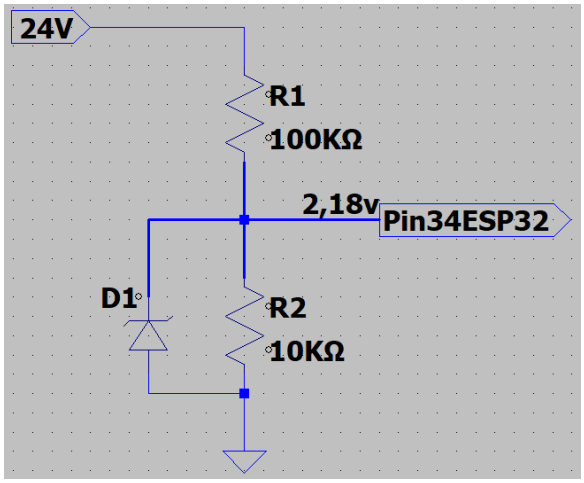

Ilustración 23: Divisor de tensión para la silla.

- Diodo rectificador 1N4001[34]:

Se ha colocado un diodo rectificador a la entrada de 24V que llegan al ESP32 con el fin de evitar que se quemen los componentes en el caso de una equivocación a la hora de conectar las baterías.

- Pulsadores [35]:

Se han utilizado otro tipo de botones más grandes en comparación a los usados en el prototipo, ya que estos se adaptan mejor a la caja del mando de control y son más robustos.

- Fusible:

Se ha utilizado un fusible de 0,5A en la entrada de voltaje del mando de control para proteger todos los componentes de la placa en caso de malas funcionalidades que pudieran aparecer en el circuito.

- Zumbador [21]:

Utilizado como bocina, es el mismo usado en el coche de pruebas y conectado al módulo MOSFET para ser controlado de forma más eficiente. Se le ha colocado una resistencia de 2KΩ en serie para evitar excesos de corriente y volumen.

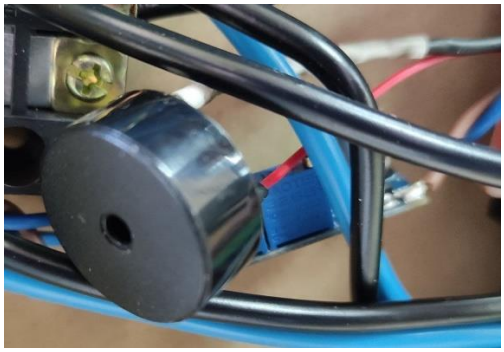

Ilustración 24: Zumbador en la caja de drivers.

- ESP32 [12]:

El mismo utilizado para el coche de pruebas, solo se han cambiado los pines utilizados. Es el modelo oficial de Espressif, que puede adquirirse en los distribuidores oficiales habituales por menos de 10€, como Mouser o RS online.

- Regulador LM2596s [19]:

Utilizado para alimentar el ESP32 a través de las baterías.

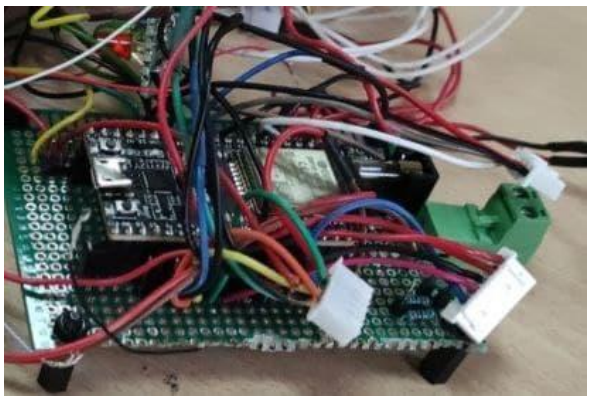

Ilustración 25: Placa de circuito.

En la Ilustración 25 vemos cómo estarían colocados los componentes en la placa, donde el ESP32 va colocado en el centro y los demás cables van hacia el joystick, los drivers BTS de los motores, los LEDs, los pulsadores y el regulador de tensión a 5V.

En cuanto a los cables de las baterías y los motores, al trabajar con corrientes altas necesitan ser de un grosor mayor, o podrían terminar quemados, en este caso se ha utilizado cable de un grosor de 1,5mm^2 y que no ha presentado ningún problema en cuanto a temperaturas altas provocadas por la alta corriente de las baterías.

Para los conectores, se han usado *Fast-On* para las conexiones de las baterías al interruptor de encendido y apagado y para los frenos de los motores, puntas huecas

de crimpado para los cables que van a los drivers y conectores (JST) de 5 terminales para los cables que van del mando de control a los drivers.

Tabla 2. Presupuesto aproximado de la silla de ruedas.

Unidades	Nombre	Precio, Unidad (€)	Precio total (€)
1	ESP32	8	8
2	Batería plomo	70	140
1	Joystick	13	13
5	LED	0,1	0,5
4	Botón	1,75	7
10	Resistencia	0,04	0,4
1	Diodo rectificador	0,1	0,1
1	Diodo zener	0,2	0,2
2	BTS7960	10	20
1	Zumbador	1,5	1,5
1	Módulo MOSFET	2	2
1	Botón de encendido	1	1
2	Clema	2,5	5
1	Fusible 0,5A	0,7	0,7
2	LM2596s	0,4	0,8
2	Motor 24v	90	180
-	Cable	-	4
		Presupuesto total:	384,2€

En la Tabla 2 puede verse que, sin contar el precio de la silla, tenemos un total de 384,2€ un precio aún muy por debajo de lo que suele estar una silla de ruedas motorizada estándar, unos 800€ aproximadamente. [36]

4.2.1. ESQUEMA DE COMPONENTES DE LA SILLA

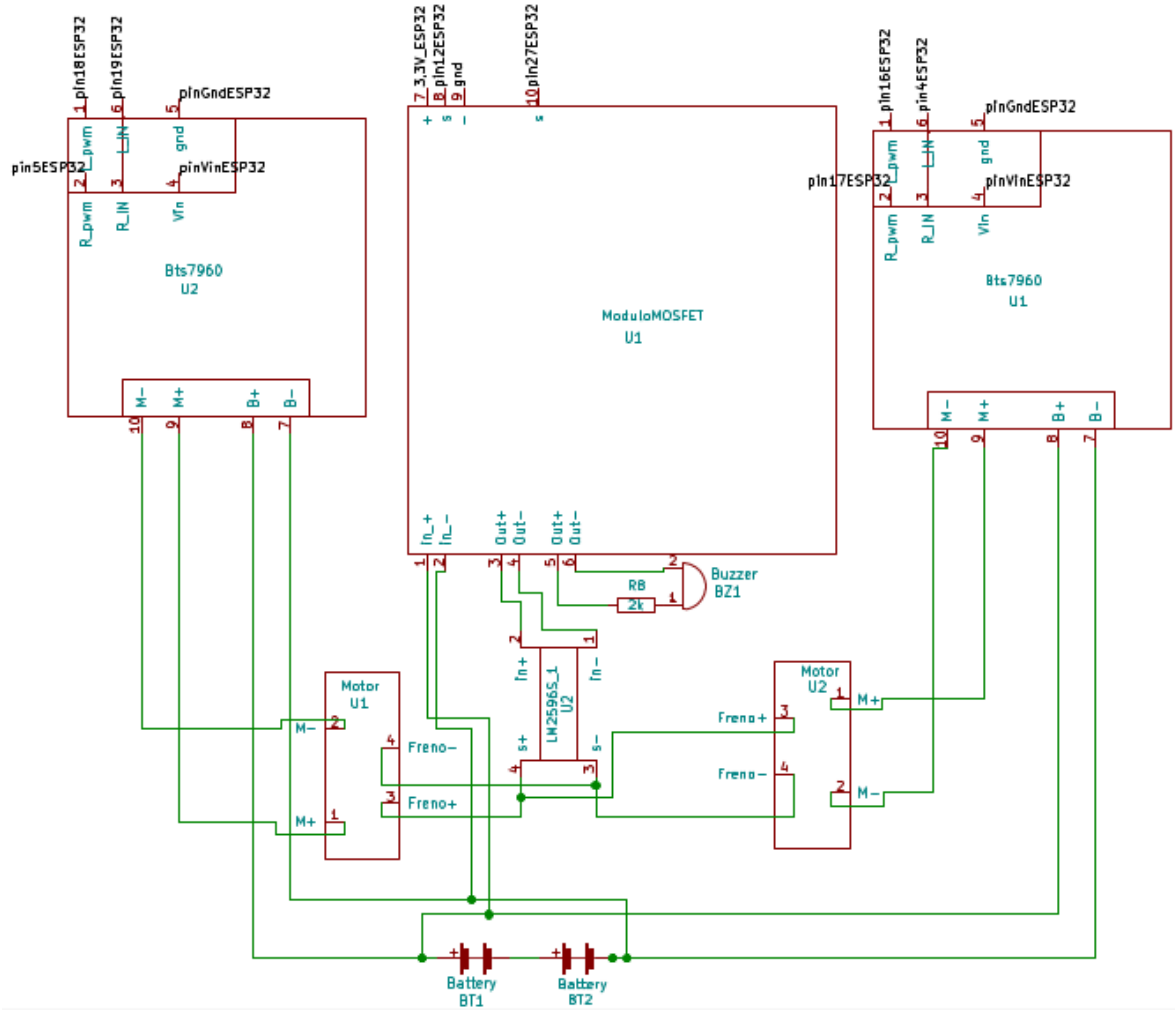

Ilustración 26: Esquema de drivers de los motores.

Utilizando el programa de diseño de circuitos, KiCad [37], en la Ilustración 26, podemos observar cómo irían conectados los componentes, los BTS7960, el módulo de MOSFET (todos ellos conectados a los pines del ESP32 mediante cables de unos 45cm de largo), luego, las baterías que van conectadas a los BTS7960 y estos a su vez a los motores para alimentarlos.

En el módulo MOSFET van conectadas las baterías en la entrada y en la salida a través del regulador LM2596s van conectadas a los frenos de los motores, en la otra salida se conecta la bocina con una resistencia.

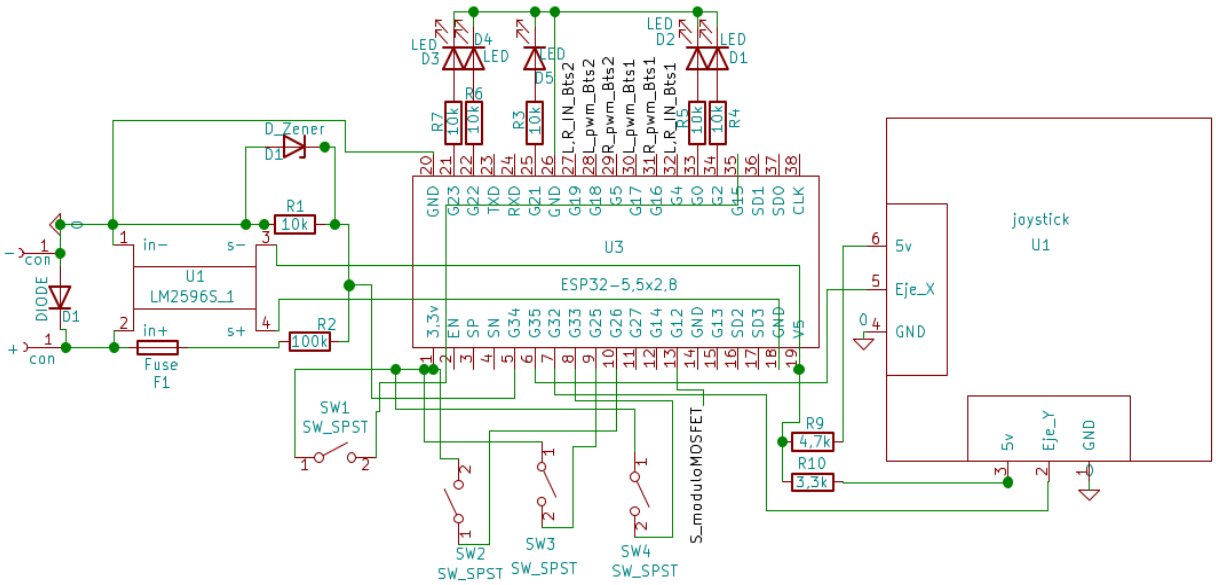

Ilustración 27: Esquema del mando de control de la silla.

En la Ilustración 27 podemos observar cómo irían distribuidos los componentes del mando de control, en los terminales de la izquierda es donde van los cables de las baterías para alimentar el ESP32 y éste a los demás componentes.

4.3. CONTROL VÍA BLUETOOTH

Para utilizar el coche del prototipo vía Bluetooth, así como también con la silla, se ha utilizado la app gratuita "Dabble" [38] que se puede encontrar en el *Play Store* de Android. La app se puede conectar al ESP32 mediante Bluetooth [39] y así podremos programarlo para controlar los motores.

Ilustración 28: Herramienta Dabble.

Estas son algunas de las funciones que nos proporciona la herramienta, aunque solo se ha usado la sección "*Gamepad"*, esta herramienta consta de más secciones muy interesantes y que podrían ayudar en otro tipo de proyectos.

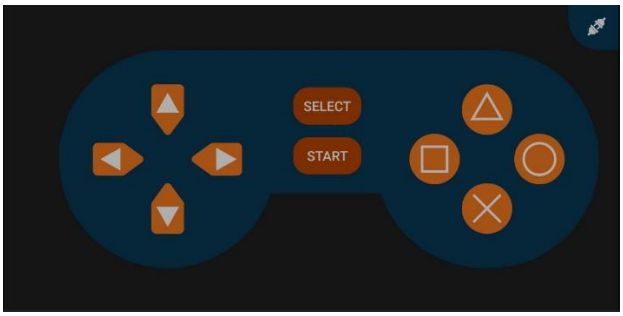

Ilustración 29: Gamepad utilizado de la herramienta Dabble.

Para este trabajo solo se ha utilizado la parte de las flechas para dirigir la silla, aunque si se añadieran más funciones se podrían usar los demás botones para ello.

Para la comunicación con ESP32 se ha utilizado la librería "*DabbleESP32.h*" que permite la comunicación vía Bluetooth con esta herramienta.

En la parte Bluetooth no se ha puesto ningún cambio en la velocidad ya que al no ser el usuario quién maneja el control de la silla es preferible usar una velocidad constante y muy baja.

Esta herramienta, aunque muy sencilla e intuitiva, nos limita al uso de cada una de sus funciones, por lo que si queremos poner un indicador de batería en la pantalla del móvil por ejemplo mientras se están accionando las flechas para mover la silla, no podremos. Para hacerlo tendríamos que modificar la propia aplicación.

4.4. DISEÑO DE LA CAJA DE DRIVERS Y EL MANDO DE CONTROL

Los diseños se han realizado utilizando el programa de diseño "*Inkscape*" [40] y posteriormente cortados con la cortadora laser de CO2 de 120W proporcionada por Smart Open Lab, el Fablab de la Escuela Politécnica de Cáceres [41].

- Primer prototipo:

En un principio se pensó en una misma carcasa tanto para las tarjetas de drivers como del mando de control. Puede verse el resultado en la Ilustración 30.

Ilustración 30: Primera carcasa.

Se utilizaron medidas aproximadas ya que era la primera prueba, utilizando como punto de partida un diseño de "*festi.info/boxes.py*" (web muy interesante que ayuda a generar cajas en formato vectorial) [42].

Después de ver como había que añadir más componentes electrónicos a la silla, se decidió cambiar y utilizar una carcasa solo para los drivers y la bocina y otra para el mando de control.

- Caja de drivers:

Se ha diseñado como una caja cuadrada de 18x18x7cm, varios agujeros para pasar los cables, tanto los que van las baterías como los que van a la caja de control.

Utilizando el programa "*Inkscape*" para el diseño el resultado se puede ver en la siguiente ilustración.

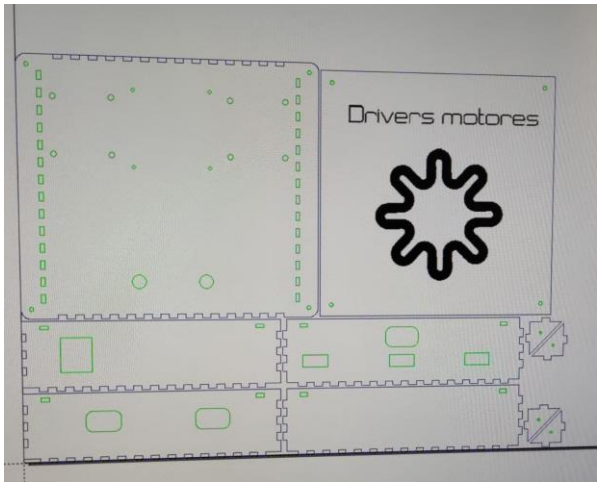

Ilustración 31: Diseño de la carcasa de drivers con Inkscape.

En la Ilustración 31 puede verse uno de los diseños, la utilización de diferentes colores es debido a que así es como se le asigna a la cortadora laser qué partes debe cortar primero y a qué velocidad y potencia debe hacerlo.

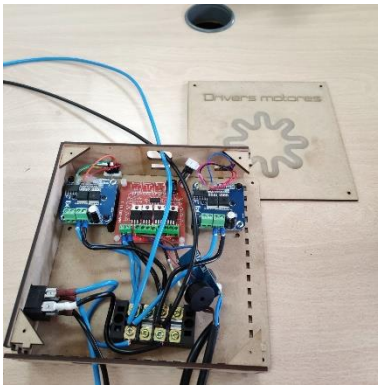

Ilustración 32: Montaje de la carcasa de drivers.

En la Ilustración 32 puede verse el montaje de los componentes de los drivers en su respectiva caja.

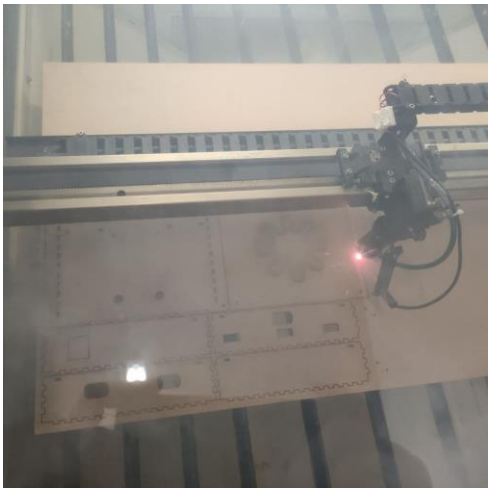

Ilustración 33: Cortadora láser.

- Caja de control:

La caja de control se ha diseñado con forma rectangular, aunque un poco más estrecha al final, con agujeros para los cables, los indicadores de batería LEDs, los botones y el joystick.

Con unas medidas de 18 cm de largo, 8 cm en la parte de atrás y 6 cm en la parte de delante, y 8 cm de altura, puede verse en la ilustración 34.

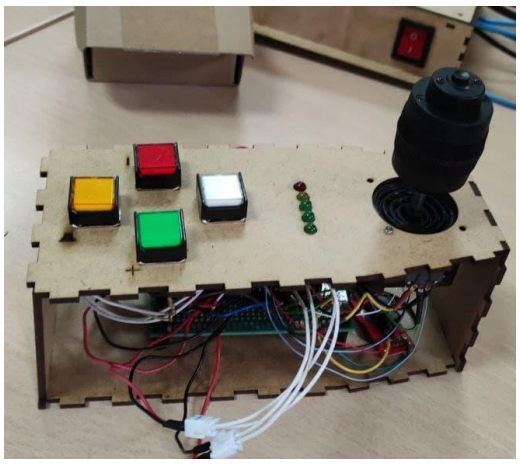

Ilustración 34: Prueba de la caja de control.

Después, se procede al cableado entre los distintos módulos, el resultado puede verse en la Ilustración 35.

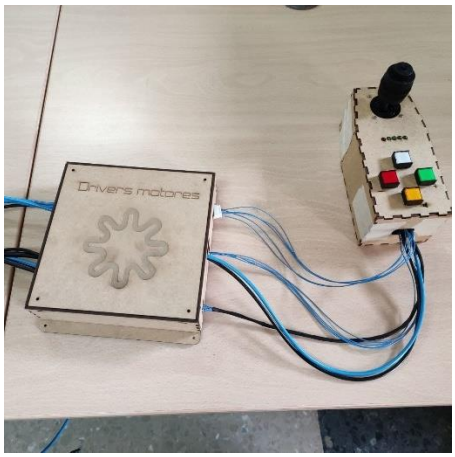

Ilustración 35: Carcasa de drivers y de control.

Estos son los prototipos finales que se van a poner a la silla de ruedas para poder probar su funcionamiento completo, para mejorarlos habría que hacer unas carcasas selladas con cables y conectores impermeables, para evitar que entre agua, polvo o cualquier otro elemento no deseado.

4.5. PROGRAMACIÓN

4.5.1. SECCIÓN DE LIBRERÍAS

En este apartado se explicarán algunas de las partes más importantes del código utilizado para la silla de ruedas. El código completo se encuentra al final de este documento, en la parte de anexos.

```
//bluetooth libraries
#define CUSTOM_SETTINGS
#define INCLUDE_GAMEPAD_MODULE
#include <DabbleESP32.h>

//Motors controlls libraries
#include "BTS79602M.h"

#include <analogWrite.h>
```

Ilustración 36: Librerías.

Se han utilizado tres librerías para la programación.

- La librería *DabbleESP32.h*, encargada de la utilización de los comandos necesarios para comunicar el ESP32 con la app del teléfono móvil *Dabble*, y que se puede encontrar en la página de listas de bibliotecas de Arduino. [43]

- La librería *BTS79602M.h*, encargada de la utilización de los comandos para el control de los motores (dirección, velocidad y stop), está librería es una modificación que se ha hecho para este trabajo de la librería *BTS7960.h*, la cual nos proporciona su autor Luis Llamas desde su página web [44].

- La librería *analogWrite.h*, proporcionada por el IDE de Arduino, y utilizada para controlar señales analógicas, el inconveniente estaba en que este comando no funcionaba con ESP32 a menos que se utilizara una librería propia para su arquitectura. También fácil de encontrar en la página de listas de bibliotecas de Arduino [45].

4.5.2. SECCIÓNES PRINCIPALES DE LA FUNCIÓN LOOP

- Rampa de aceleración/deceleración y velocidad de los motores:

Con esto podemos incrementar o reducir la velocidad de los motores de forma progresiva, y así evitar los arranques o frenadas bruscas.

Se ha utilizado una rampa lineal con el mismo tiempo de aceleración que de deceleración. [46]

```
  //if flag_b is 0, speed is LOW and if flag_b is 1, speed is HIGH
if (flag_b == 0) {

       if(millis() > V_last_update + V_update_period){
            V_last_update = millis();

            v_d=(distancia(X, Y)/13);
                                 //if we have to increase speed
                  if(v_d - v > TOL){  //comparison between desired speed (V_d)  and actual speed (v)
                    v=v+INC_v;
                              }
        if ((v_d-v < TOL) && v>0){ //if we have to decrease speed
            v=v-INC_v;
                               }

                   if(v<0){ //v can never be negative
                      v=0;

                           }

         }
}
```

Ilustración 37: Implementación de la rampa de aceleración.

Lo primero que vemos en esta parte del código es un retraso por parte de la función "millis" la cual repetirá este código cada 80ms.

La velocidad deseada *v_d*, la cual va en función de la distancia del joystick al punto de reposo central, dividida entre 13 para mapear la distancia entre los valores PWM de los motores (de 0 a 141, para no llegar al máximo de los motores, 255), se compara con la velocidad real que se le va a dar a los motores "v" y si está por encima de la tolerancia de 5 se aumenta la velocidad real en *INC_v* cada 80ms, por el contrario, si la diferencia está por debajo de esa tolerancia disminuye *v* en *INC_v* cada 80ms, si "v" toma valores negativos se pone a 0.

A continuación, para explicar los movimientos de la silla de ruedas hay que tener en cuenta que la posición del joystick se puede dividir en cuatro formas de acción, hacia adelante, hacia atrás, izquierda y derecha. Veremos solo la parte de código utilizada para girar a la izquierda y la parte para moverse hacia adelante ya que tanto la parte para girar a la derecha como para moverse hacia atrás están programadas de la misma forma. Lo único que cambia es el cuadrante en el que se encuentra el joystick y las direcciones que deben seguir cada uno de los motores.

- Giro a la izquierda:

```
    //left
if (ang >= 15 && ang <= 165 && v>0 ) { //range of values depending on the quadrant in which the joystick is located
     // Serial.println("L");
      if (flag_forward == 1 && flag_back == 0){ //if it has gone forward before, turn left with an engine turning faster

          dif1=abs(165-angulos(X,Y,v));

            if((v-dif1)<=1){
               dif1=v-1;
                      }

           digitalWrite(controlPIN, HIGH);
           //delay(100);

           motorController.TurnLeftM2(v);
           motorController.TurnLeftM1((v)-dif1);

      }

      if(flag_back == 1 && flag_forward == 0){ //if it has gone backwards before, turn left with an engine turning faster

      dif2=abs(15-angulos(X,Y,v));

            if((v-dif2)<=1){
               dif2=v-1;
                   }

      digitalWrite(controlPIN, HIGH);
      //delay(100);

      motorController.TurnRightM2(v);
      motorController.TurnRightM1((v)-dif2);

      }

       if(flag_forward == 0 && flag_back == 0){ // if the joystick was initially at rest, turn left with one wheel forward and one wheel back
           digitalWrite(controlPIN, HIGH);
            //delay(100);

          motorController.TurnLeftM2(v/2);
           motorController.TurnRightM1(v/2);
                        }
  }
```

Ilustración 38: Parte del código para girar a la izquierda.

En la Ilustración 38 podemos ver al principio el rango de valores en los que se tiene que hallar el joystick para que coincida con el movimiento que le corresponde (giro a la izquierda). Para girar a la derecha será igual, pero con otro rango de valores.

Después, nos encontramos con tres condiciones que dependen de dos variables que actúan de banderas y que nos van a decir en qué estado se encontraba el joystick anteriormente.

Si el joystick se encontraba con la dirección hacia adelante, se calcula el número, *dif1* en función del ángulo de giro y se resta ese número a la velocidad de uno de los motores, entonces a medida que se gire el joystick también lo hará la silla.

Si el joystick se encontraba con la dirección hacia atrás se calcula otro número *dif2* también en función del ángulo de giro y se resta ese número a la velocidad de uno de los motores, entonces a medida que se gire el joystick también lo hará la silla.

La última condición es solo para cuando la silla se encuentra parada en reposo anteriormente, de ese modo el giro, tanto a la izquierda como a la derecha, se efectúa haciendo girar cada una de las ruedas en una dirección distinta, por lo que es un giro más cerrado. Y se pone la velocidad a la mitad ya que no se necesita una velocidad muy alta para este tipo de giros.

- Hacia adelante:

```
    //forward
if (ang >= 170 && ang <= 190 && v>0 ) {//range of values depending on the quadrant in which the joystick is located
      //Serial.println("F");

         if(flag_back==0 && flag_forward==0){
         flag_forward=1;
                 }

         if(flag_forward == 1 && flag_back==0){ //the chair goes forward unless the joystick is moved from state to another bypassing 0

             digitalWrite(controlPIN, HIGH);

              motorController.TurnLeftM2(v);
                motorController.TurnLeftM1(v);

                           }
         if(flag_back==1 &&flag_forward==0){

              motorController.TurnRightM2(v);
                motorController.TurnRightM1(v);

                           }

}
```

Ilustración 39: Parte del código para moverse hacia adelante.

En la Ilustración 39 podemos ver la primera condición que indica el cuadrante para moverse hacía adelante, como se comentó en la parte de giro a la izquierda, también es lo mismo para moverse hacía a atrás.

Después nos encontramos una condición que nos dice que si estábamos anteriormente parados en reposo, la bandera de moverse hacia adelante (en este caso) se pone a 1 y esto efectúa el movimiento hacia adelante.

En el caso de que estuviésemos moviéndonos hacia atrás y moviéramos el joystick de forma imprudente hacia adelante, deprisa, sin haber parado la silla anteriormente, se activa la última condición y la silla seguirá girando hacia atrás. Esto se ha hecho para evitar que los motores de la silla hagan un cambio de sentido de forma brusca, entonces, con esto le dará al usuario tiempo para poder parar la silla y volver a efectuar el movimiento como es debido. Esto será igual en la parte de código para moverse hacia atrás.

- Cálculo del voltaje de las baterías:

```
//voltage divider
  Div = analogRead(34); //reading pin 34
  Div_v = ((Div * 25)/2300); //range conversion operation using ESP32 (0-2300) to volts.
```

Ilustración 40: Parte de código para medir el voltaje de las baterías.

Se puede ver en la Ilustración 40 la lectura del GPIO34, que es donde va conectado el divisor de tensión explicado anteriormente. El valor se guarda en la variable de tipo entero "*Div*".

En cuanto a los cálculos, con el divisor de tensión utilizado, el ESP32 devuelve un valor entre 0 y 2300 con el valor máximo que puede dar para las baterías (25V), por lo tanto, de esta forma cambiamos el valor dado de ESP32 por el voltaje real que tienen las baterías.

- Movimientos de la silla vía Bluetooth:

Para esta parte del código solo se explicará el movimiento hacia adelante y el de parar, usando comunicación Bluetooth, ya que los demás movimientos se hacen de la misma forma.

- Giro hacia adelante vía bluetooth:

```
  //forward
  ////////////////////////////////////////////////////////////////////////////////////////////////
if (GamePad.isUpPressed() == 1 && GamePad.isLeftPressed() == 0 && GamePad.isRightPressed() == 0 && GamePad.isDownPressed() == 0 && v == 0) {
    digitalWrite(controlPIN, HIGH);
    delay(100);
    motorController.TurnLeftM2(50);
    motorController.TurnLeftM1(50);

  }
```

Ilustración 41: Parte de código para moverse hacia adelante vía bluetooth.

Se aplica una condición sobre los botones que aparecen en la pantalla del móvil y que la velocidad sea 0, con esto nos aseguramos que mientras se está usando el control de la silla por Bluetooth el joystick debe de estar en reposo o de lo contrario no funcionará el control desde el móvil.

Una vez detectado qué botón se ha pulsado se mueven los motores en la dirección indicada.

- Parar la silla:

```
//Stop motors
////////////////////////////////////////////////////////////////////////////////////////////////
if (GamePad.isDownPressed() == 0 && GamePad.isUpPressed() == 0 && GamePad.isLeftPressed() == 0 && GamePad.isRightPressed() == 0 && v==0) {
   motorController.Stop();
   motorController.Disable();
   digitalWrite(controlPIN, LOW);

   flag_forward=0;
   flag_back=0;

  }
```

Ilustración 42: Parte del código para parar la silla.

Se usa la condición de que todos los botones vía Bluetooth no deben estar pulsados y la velocidad debe de ser 0, entonces se paran y se desconectan los motores y se activan los frenos de los motores, gracias a la librería usada para ellos. Después se colocan todas las banderas a 0.

4.5.3. FUNCIONES ADICIONALES

En este apartado se hablará de algunas las funciones adicionales que se han utilizado en el código.

- Función *leds( ):*

No se muestra la función entera, ya que es extensa. Se puede ver completa en el anexo.

Esta es una función muy simple, en la que a medida que el valor del voltaje de las baterías va disminuyendo se irán apagando los LEDs utilizados para el indicador de batería.

```
  //leds function
  ////////////////////////////////////////////////////////////////////
void leds(int aux, int Div_v) { //button and value of voltage

    if (aux == 1) {

      if (Div_v >= 24 ) {

        digitalWrite(ledPIN1 , HIGH);
        digitalWrite(ledPIN2 , HIGH);
        digitalWrite(ledPIN3 , HIGH);
        digitalWrite(ledPIN4 , HIGH);
        digitalWrite(ledPIN5 , HIGH);

                                    }

      if (Div_v >= 20 && Div_v < 24 ) {
        digitalWrite(ledPIN1 , HIGH);
        digitalWrite(ledPIN2 , HIGH);
        digitalWrite(ledPIN3 , HIGH);
        digitalWrite(ledPIN4 , HIGH);
        digitalWrite(ledPIN5 , LOW);

                                    }

      if (Div_v <= 18 && Div_v < 20 ) {

        digitalWrite(ledPIN1 , HIGH);
        digitalWrite(ledPIN2 , HIGH);
        digitalWrite(ledPIN3 , HIGH);
        digitalWrite(ledPIN4 , LOW);
        digitalWrite(ledPIN5 , LOW);
                    .
                    .
                    .
```

Ilustración 43: Parte del código, encendido de LEDs.

- Función *`angulos( )`*:

```
  //angulos function
int angulos (int x, int y, int V) { //returns the angle from 0 to 359 at which the joystick is, calculating its arc tangent.

    auxi = atan2(x, y);
    Angle = (auxi * 180) / PI;

   if (Angle < 0) {
      Angle = Angle + 360;
    }

       if(v>0 && distancia(X,Y)>0 ){
         local=Angle;
                }

          if( v==0 && distancia(X,Y)==0){
              return 14;
                   }

            else
                if( v>0 && distancia(X,Y)==0){
                     return local;
                          }
                            else
                               return Angle;
}
```

Ilustración 44: Parte del código para saber el ángulo del joystick.

En ella se calcula el ángulo en el que se encuentra el joystick en función de los ejes "*X*" e "*Y*" leídos por los potenciómetros, usando el arco tangente y pasándolo a grados, después se le suman 360º cuando los ángulos son negativos.

A continuación, se guarda el ángulo en una variable siempre que la velocidad de los motores sea mayor a 0.

Entonces, si la silla está parada devuelve un valor que no se encuentra entre los utilizados anteriormente para mover la silla y así evitar posibles conflictos, se ha utilizado el valor 14 ya que el valor por defecto cuando se paraba era 0 y coincidía con el cuadrante del joystick para moverse hacia atrás.

Si la distancia es 0, pero la velocidad es mayor a 0, devuelve el valor del ángulo cuando la distancia aún era mayor a 0, que se almacena en la variable local.

Y si no está en ninguna de las situaciones anteriores devuelve el ángulo en el que se encuentra de forma normal con la variable *Angle*.

- Función *`distancia()`*:

```
  //Distance function
int distancia(int x, int y) { //returns the distance from the center of the joystick doing the modulus.

    distance = (int)sqrt(pow(x, 2) + pow(y, 2));

    return distance;
  }
```

Ilustración 45: Parte del código para saber la distancia del joystick.

Se calcula la distancia desde el centro hasta donde se encuentre el joystick haciendo el módulo de ambos ejes.

5. RESULTADOS Y DISCUSIÓN

5.1. RESULTADOS

Se explicarán todas las funcionalidades finales conseguidas para la silla de ruedas.

Las más importantes se encuentran en el joystick, ya que es donde se inician todos los movimientos de la silla. Para los movimientos se ha usado el módulo de la distancia del eje "*X*" con el eje "*Y*" y el ángulo según el cuadrante en el que se encuentran, con una tolerancia de 5º entre movimientos.

- **Rampa de aceleración:**

El resultado es que al iniciar un movimiento con el joystick, la velocidad de los motores empieza a subir un tiempo después (80ms) de manera progresiva. Con este fin evitamos aceleraciones bruscas que puedan provocar lesiones al usuario o una pérdida del control de la silla.

-Movimientos del joystick con respecto a la silla:

- **Movimiento adelante:**

Las ruedas de la silla se mueven hacia adelante con velocidad progresiva hasta alcanzar la velocidad deseada marcada por el joystick.

En el caso de que se quisiera girar a la izquierda o la derecha mientras se está moviendo hacia adelante se deberá girar el joystick hacia esa posición ligeramente, ya que a medida que se mueva el joystick el giro es más cerrado.

En el caso imprudente de que se mueva el joystick de adelante a atrás sin haber parado por completo la silla antes, las ruedas siguen girando hacia adelante (para evitar el cambio repentino de adelante a atrás) pero se debe volver al estado de reposo y parar por completo.

- **Movimiento atrás**:

Las ruedas de la silla se mueven hacia atrás con velocidad progresiva hasta alcanzar la velocidad deseada marcada por el joystick.

En el caso de que se quisiera girar a la izquierda o la derecha mientras se está moviendo hacia atrás se debe girar el joystick hacia esa posición ligeramente, ya que a medida que se mueve el joystick el giro es más cerrado.

En el caso imprudente de que se mueva el joystick de atrás a adelante sin haber parado por completo la silla antes, las ruedas siguen girando hacia atrás (para evitar el cambio repentino de atrás a adelante) pero debe volver al estado de reposo y parar por completo.

- **Movimiento izquierda y derecha:**

Si se empieza a girar el joystick hacia la izquierda o la derecha desde el reposo, éstas giran, una hacia adelante y la otra hacia atrás, dependiendo de si es a la izquierda o a la derecha.

Como este tipo de movimientos no requiere mucha velocidad, se ha disminuido ésta a la mitad.

En el caso imprudente de cambiar el joystick de izquierda a derecha sin haber parado por completo la silla antes, ésta hace un cambio de sentido de las ruedas, como la velocidad no debiera ser muy grande, el cambio no debe afectar mucho al usuario, aunque es un movimiento que no debería hacerse.

- **Botones:**

La caja de control consta de cuatro botones, el botón blanco es para encender los LEDs del indicador de batería, el rojo es para activar la velocidad inferior de la silla y el verde para activar la velocidad superior, que según la norma UNE-EN 12184 [47] no debe superar los 15km/h. El botón naranja sirve para activar la bocina.

- Consumo de potencias aproximadas:

Para calcular las potencias que consumen los dos motores se ha utilizado el multímetro de pinza *Limit 21* [48], que mide amperios con una precisión del 2%. Con él se han medido las intensidades de las baterías y la que llegan a los motores. Se han medido en función de los botones de velocidad, puesto que con cada botón la potencia máxima alcanzada no es la misma, y nunca se usa la potencia máxima que pueden alcanzar los motores, ya que después de realizar algunas pruebas en movimiento con la silla se vio que no era necesario utilizar tanta potencia.

Ilustración 46: Multímetro Limit 21 utilizado.

$$P(W) = I(A) * V(V)$$

P es la potencia en watios, I es la intensidad en amperios y V es el voltaje en voltios.

Tabla 3: Consumos aproximados.

	Botón velocidad baja			*Botón velocidad alta*		
	I(A)	*Potencia (w)*	*Autonomía*	*I(A)*	*Pontencia(w)*	*Autonomía*
Baterías	*6*	*150w*	*Imax=33Ah Vmax= 25V*	*8*	*200w*	*Imax=33Ah Vmax= 25V*
Motor1	*3*	*75w*		*4*	*100w*	
Motor2	*3*	*75w*		*4*	*100w*	
Freno1	*0,5*	*5w*		*0,5*	*5w*	
Freno2	*0,5*	*5w*		*0,5*	*5w*	
caja de control	*0,1*	*2,5w*		*0,1*	*2,5w*	
Suma consumos	*7,1*		*825wh/150w= 5,5h*	*9,1*		*825wh/200w= 4,125h*

En la Tabla 3 se puede observar que aproximadamente, la potencia generada por las baterías coincide con la consumida por los motores. Los motores como máximo consumen 100w cada uno ya que no se están utilizando a plena potencia. En cuanto a la autonomía, se ha calculado en base a la potencia máxima con la que se puede usar la silla dando como resultado, a velocidad baja una autonomía de 5,5h y a velocidad alta una autonomía de 4,125h.

Se podría hacer un cálculo del consumo de la parte de control más exacto, aunque su consumo es muy bajo comparado con los motores.

En la Ilustración 47 puede verse el resultado final de la silla de ruedas diseñada.

Ilustración 47: Resultado de la silla final.

5.2. DISCUSIÓN

Se han cumplido los objetivos, aunque quedan por realizar las mejoras que se describen a continuación.

- Carga de las baterías:

Cargar las baterías de forma eficiente sin tener que sacarlas de su compartimento y conectarlas a la red eléctrica mediante un enchufe.

No se ha diseñado el circuito de carga de las baterías, en su lugar se usa un cargador comercial económico de la marca "*RIKIN*"[49].

- Velocímetro:

Se podría añadir un indicador de velocidad para que el usuario pueda saber en qué momento está yendo demasiado rápido.

- Incluir sensores de distancia:

Sensores de proximidad para detectar la distancia a la que se encuentra la silla de un obstáculo y avisar de ello al usuario. Se podrían usar los sensores LIDAR[50] que utiliza pulsos de láser, y tienen alta precisión y frecuencia de medidas.

- Incluir luces de posición e indicadores de giro:

Para que otros vehículos puedan detectar mejor la presencia de la silla y saber cuál es la dirección que va a tomar.

- Modificar el control Bluetooth:

Disponer de un botón para activar o desactivar la función de bluetooth de la silla e incorporar una contraseña para evitar que personas no deseadas puedan conectarse a la silla.

- Mejorar la ergonomía:

Ajustar aún más la ergonomía de la silla para que pueda usarse para cualquier tipo de persona, según su tipo de discapacidad.

- Aumentar la funcionalidad de la bocina:

Cuando las baterías estén a punto de agotarse, hacer sonar la bocina de forma intermitente para avisar al usuario. Cuando se va marcha atrás para avisar a las demás personas, etc.

- Definir una base de datos:

Proporcionar en una base de datos un ID de la silla y el nombre del propietario de esta para saber quién, qué tipo silla y qué discapacidad tiene. De este modo se podrán controlar mejor este tipo de vehículos, o incluso añadir un nódulo GPS para determinar su posición y su velocidad.

Para este trabajo, como se ha visto anteriormente, se han utilizado baterías de plomo ya que son las más utilizadas en este tipo de vehículos, pero si nos preguntamos cuales serían mejores con respecto a las baterías de litio tenemos que ver qué diferencias hay entre ellas.

Las baterías de lito son mucho más ligeras que las de plomo y poseen un mayor número de ciclos de vida con un mayor rendimiento de hasta un 30% en comparación. También puedes descargarlas hasta el 100% mientras que las de plomo no es recomendable (aunque en el caso de estas baterías son de descarga profunda). Las de plomo como se ha dicho anteriormente, son mucho más baratas que las de litio.

El mayor inconveniente que pueden tener las baterías de litio es su inestabilidad, lo que obliga a usar sensores de temperatura e indicadores de voltaje para evitar que surja cualquier problema. [51]

Por lo tanto, ¿por qué no usar baterías de litio? La respuesta a la pregunta va ligada al precio, ya que las baterías de litio son muy caras y aún más caras si necesitan diferentes tipos de sensores. Por lo tanto, las de plomo son más baratas y no presentan tantas dificultades.

No se han implementado otras funcionalidades por falta de tiempo, este trabajo se ha llevado a cabo en aproximadamente un año.

6. CONCLUSIONES

A lo largo de este proyecto se han ido cumpliendo los objetivos que se habían marcado, centrándonos en el diseño de la electrónica de la silla de ruedas motorizada de la forma más económica posible y utilizando hardware y software libre.

Hemos conseguido controlar la velocidad de forma segura y hacer giros de manera suave.

Hemos mejorado la seguridad a la hora de utilizar los botones, ya que, con el mando de control estándar original de la silla de ruedas, era difícil pulsar un botón sin accionar el joystick para personas con movilidad muy reducida. Con este nuevo diseño del mando de control solo se podrán accionar los botones si la silla está parada o utilizando la otra mano.

En mi opinión, este desarrollo le proporciona aún más seguridad y mejor estabilidad con la ayuda de la rampa de aceleración, de la que se habla anteriormente, que puede ayudar a que usuarios de la silla con una movilidad poco fluida puedan acelerar poco a poco sin provocar cambios bruscos en el movimiento de los motores.

Aunque se ha intentado que esta silla de ruedas pudiera ser utilizada para cualquier tipo de discapacidad, es evidente de que aún estamos muy lejos para llegar a lograr ese objetivo, teniendo en cuenta algunas de las discapacidades ya mencionadas con anterioridad.

El resultado, aunque bueno, no se puede decir que sea plenamente satisfactorio, ya que la forma en la que está estructurado el mando de control hace que no valga la pena cambiar la placa desarrollada por una PCB, debido a que la reducción del número de cables que supondría es solo de dos cables. Sí que podría aumentarse la estabilidad con conectores homologados más robustos.

En cuanto a cuestiones de aprendizaje, se puede decir que se han obtenido nuevos conocimientos sobre distintas materias como:

- El uso adecuado de un soldador de estaño, con el que se ha soldado la placa del mando de control y los cables.

- El funcionamiento de los motores DC con freno eléctrico, controlados por los drivers y el módulo MOSFET.

- El uso del IDE de Arduino para programar el microcontrolador ESP32, modificar librerías, etc.

- Nociones básicas de KiCad para el desarrollo del esquema del circuito utilizado, de aquí a crear una PCB solo hay un paso.

- Nociones básicas del programa de diseño Inkscape, para el desarrollo de la caja de drivers y el mando de control, usando diseño gráfico virtual.

- Manejo de cables de distintas medidas, corte y pelado de cables para ser ajustados a los distintos componentes utilizados.

REFERENCIAS BIBLIOGRÁFICAS

[1] **Instituto Nacional de Estadistica.** Discapacidades imprenta-INE. [En línea] Octubre de 2009. [Consultado el 14 de Mayo de 2021] https://www.ine.es/revistas/cifraine/1009.pdf

[2] **Sunrise medical.** Evolución e historia de la silla de ruedas. [En línea] 18 de septiembre de 2018. [Consultado el 17 de enero de 2021] https://www.sunrisemedical.es/blog/historia-silla-de-ruedas

[3] **Karma mobility España.** Historia de la silla de ruedas. [En línea] 26 de Abril de 2020. [Consultado el 16 de Mayo de 2021] https://www.karmamobility.es/2020/04/historia-de-la-silla-de-ruedas/

[4] **Scewo.** Scewo. [En línea] 2017. [Consultado el 24 de Abril de 2021] http://www.scewo.ch/

[5] **RAE.** Asociación de Academias de la Lengua Española. [En línea] 2020. [Consultado el 14 de Mayo de 2021] https://dle.rae.es/discapacidad

[6] **Fisioterapia, Centro de rehabilitación y.** FREIMO. [En línea] 2017. [Consultado el 15 de Mayo de 2021] http://www.freimo.com/patologias/Patologias-de-Sistema-Nervioso/Monoplejias/

[7] **España, Karma mobility.** Paraplejía: Causas, tratamientos y cuidados. [En línea] 4 de Junio de 2020. [Consultado el 16 de Junio de 2021] http://www.freimo.com/patologias/Patologias-de-Sistema-Nervioso/Monoplejias/

[8] **Mimenza, Oscar Castillero.** Tipos de discapacidad física. [En línea] 2020. [Consultado el 16 de Mayo de 2021] https://psicologiaymente.com/salud/tipos-de-discapacidad-fisica

[9] **clinic., Mayo.** Mayo clinic. [En línea] 31 de Enero de 2020. [Consultado el 17 de Mayo de 2021] https://www.mayoclinic.org/es-es/diseases-conditions/muscular-dystrophy/symptoms-causes/syc-20375388#:~:text=La%20distrofia%20muscular%20es%20un,muchos%20tipos%20de%20distrofia%20muscular

[10] **Abolafio, J.M.** Asociación PLACEAT. [En línea] 2020. [Consultado el 17 de Mayo de 2021] https://placeat.org/

[11] **Industrial, Curtiss-Wright.** PG DRIVES TECHNOLOGY VR2. [En línea] 2014. [Consultado el 13 de Marzo de 2020] http://sunrise.pgdrivestechnology.com/manuals/pgdt_vr2_manual_SK77898-05.pdf

[12] **arduino, Arduino y solo.** Arduino y solo arduino. [En línea] 1 de Marzo de 2017. [Consultado el 17 de Mayo de 2021] https://soloarduino.blogspot.com/2017/03/que-es-un-esp32.html

[13] **Prometec.** Prometec. *Instalando el ESP32.* [En línea] 20 de Febrero de 2018. [Consultado el 16 de Mayo de 2021] https://www.prometec.net/instalando-esp32/

[14] **Espressif.** esp32_datasheet_en. [En línea] 2021. [Consultado el 17 de Mayo de 2021] https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

[15] **Shoptronica.** Shoptronica. [En línea] 2020. [Consultado el 17 de Mayo de 2021] https://www.shoptronica.com/curiosidades-tutoriales-y-gadgets/1128-que-es-las-baterias-lipo-litio-0689593937322.html

[16] **Kingbright.** shop.adafruit. [En línea] 11 de Mayo de 2007. [Consultado el 17 de Mayo de 2021] https://cdn-shop.adafruit.com/datasheets/WP7113SRD-D.pdf

[17] **HDK.** components101. *datasheet/Push-Button.* [En línea] 29 de Mayo de 2007. [Consultado el 18 de Mayo de 2021] https://components101.com/asset/sites/default/files/component_datasheet/Push-Button.pdf

[18] **datasheet.es.** DataSheet. [En línea] 2020. [Consultado el 18 de Mayo de 2021] http://www.datasheet.es/PDF/919052/LM7805-pdf.html

[19] **Instruments, Texas.** Alldatasheet. [En línea] 2013. [Consultado el 18 de Mayo de 2021] https://pdf1.alldatasheet.com/datasheet-pdf/view/524001/TI/LM2596.html

[20] **technology., Handson.** components101. [En línea] [Consultado el 18 de Mayo de 2021] http://www.handsontec.com/dataspecs/L298N%20Motor%20Driver.pdf

[21] **TDK.** Piezoelectronic buzzers. [En línea] Julio de 2019. [Consultado el 17 de Mayo de 2021]

https://product.tdk.com/system/files/dam/doc/product/sw_piezo/sw_piezo/piezo-buzzer/catalog/piezoelectronic_buzzer_ps_en.pdf

[22] **JBC.** Soldering Station. [En línea] 2019. [Consultado el 9 de Junio de 2021] https://www.tme.eu/Document/daa3caaa94d0d0011c9b6c12237c7de6/manual-cd-b-compact-station.pdf

[23] **components101.** Módulo de joystick. [En línea] 2 de Abril de 2018. [Consultado el 18 de Mayo de 2021] https://components101.com/modules/joystick-module

[24] **Aprendiendo arduino.** aprendiendoarduino.wordpress. [En línea] 4 de Julio de 2016. [Consultado el 18 de Mayo de 2021] https://aprendiendoarduino.wordpress.com/2016/07/04/motores/

[25] **RDuinostar.** Protoboard y placas perforadas PCB. [En línea] 20 de Octubre de 2014. [Consultado el 23 de Mayo de 2021] https://rduinostar.com/documentacion/general/protoboard-tipos-placas-perforadas-pcb/

[26] **ENERGIVM.** Reguero Baterías. [En línea] 2004. [Consultado el 23 de Mayo de 2021] https://cdn.reguerobaterias.es/archivos/media/mvd12330.pdf

[27] **Sunrise Medical.** Motores Silla de Ruedas. Modelo 3718B. [Descatalogado].

[28] **Alldatasheet.** Hoja de datos de BTS7960 (PDF) - Infineon Technologies AG. [En línea] 7 de Diciembre de 2004. [Consultado el 18 de Mayo de 2021] https://pdf1.alldatasheet.com/datasheet-pdf/view/152657/INFINEON/BTS7960.html

[29] **Llamas, Luis.** Controla motores de gran potencia con arduino y BTS7960. [En línea] 28 de Octubre de 2019. [Consultado el 16 de Marzo de 2021] https://www.luisllamas.es/controla-motores-de-gran-potencia-con-arduino-y-bts7960/

[30] **Flores Román, David.** MyWheelchair0.2. [En línea] 29 de Mayo de 2021. [Consultado el 30 de Mayo de 2021] https://github.com/D-Flores01/MyWheelchair0.2

[31] **SejooMotion.** sjmdt. [En línea] 2013. [Consultado el 18 de Mayo de 2021] http://www.sjmdt.com/sopage/product/product_view.php?ct_code=sb0101&idx=963

[32] **International rectifier.** Alldatasheet. [En línea] 13 de Marzo de 2001. [Consultado el 22 de Mayo de 2021] https://www.alldatasheet.com/datasheet-pdf/pdf/68172/IRF/IRF540N.html

[33] **NEC.** Alldatasheet. [En línea] 1998. [Consultado el 26 de Mayo de 2021] https://www.alldatasheet.com/datasheet-pdf/pdf/6481/NEC/PS2801.html

[34] **Alldatasheet.** Datasheet1N4001. [En línea] [Consultado el 18 de Mayo de 2021] https://pdf1.alldatasheet.com/datasheet-pdf/view/207672/PANJIT/1N4001.html

[35] **Omron.** A16 Datasheet. [En línea] [Consultado el 18 de Mayo de 2021] https://docs.rs-online.com/5ace/0900766b8157da04.pdf

[36] **Online medical.** onlinemedica. [En línea] [Consultado el 2019 de Mayo de 2021] https://www.onlinemedical.es/sillas-de-ruedas-electricas-baratas/49-libercar-power-chair.html?gclid=CjwKCAjwy42FBhB2EiwAJY0yQkgnpNXWB2Ws4M3vzC5Qdfq0TexrQyhglmRGHSaBXn1YTkr3crk66xoCN8oQAvD_BwE

[37] **Charras, J.** KiCad. [En línea] 1992. [Consultado el 23 de Mayo de 2021] https://www.kicad.org/

[38] **STEMpedia.** STEMpedia. [En línea] 2020. [Consultado el 19 de Mayo de 2021] https://thestempedia.com/docs/dabble/getting-started-with-dabble/

[39] **Digital guide Ionos.** Qué es Bluetooth. [En línea] 2020. [Consultado el 23 de Mayo de 2021] https://www.ionos.es/digitalguide/servidores/know-how/que-es-bluetooth/

[40] **Levien., R.** InKscape. [En línea] 2003. [Consultado el 23 de Mayo de 2021] https://inkscape.org/es/

[41] **Smart Open Lab.** Smart Open Lab. [En línea] 2015. [Consultado el 22 de Mayo de 2021] https://www.smartopenlab.com/

[42] **Boxes.py.** [En línea] [Consultado el 22 de Mayo de 2021] https://www.festi.info/boxes.py/

[43] **STEMpedia.** arduinolibraries. [En línea] [Consultado el 19 de Mayo de 2021] https://www.arduinolibraries.info/libraries/dabble

[44] **Llamas., Luis.** GitHub. [En línea] 1 de Octubre de 2019. [Consultado el 19 de Mayo de 2021] https://github.com/luisllamasbinaburo/Arduino-BTS7960

[45] **ERROPiX.** Aruduinolibraries. [En línea] 28 de Octubre de 2018. [Consultado el 19 de Mayo de 2021] https://www.arduinolibraries.info/libraries/esp32-analog-write

[46] **Sancho, José Ramón Vaello.** automatismoindustrial. *Arranque y control de velocidad en variadores de frecuencia.* [En línea] 1 de Julio de 2019. [Consultado el 19 de Mayo de 2021] https://automatismoindustrial.com/curso-variadores-de-frecuencia/arranque-y-control-de-velocidad-en-variadores-de-frecuencia/

[47] **AENOR.** UNE-EN 12184 . [En línea] 7 de Abril de 2010. [Consultado el 19 de Mayo de 2021] https://sid.usal.es/version-imprimir/leyes/discapacidad/17992/3-4-4/norma-une-en-12184-2010-sillas-de-ruedas-con-motor-electrico-scooters-y-sus-cargadores-requisitos-y-metodos-de-ensayo.aspx

[48] **ASLAK.** PINZA AMPERIMéTRICA LIMIT 21. [En línea] 2020. [Consultado el 23 de Mayo de 2021] https://www.aslak.es/export/pdf/product/sku/144870102/

[49] **RIKIN.** Cargador de batería. [En línea] 2021. [Consultado el 6 de Junio de 2021] https://www.lt10digital.com.ar/cargadores-para-baterias/

[50] **Pulsed Light.** Sparkfun. [En línea] 2015. [Consultado el 24 de Mayo de 2021] https://cdn.sparkfun.com/datasheets/Sensors/Proximity/LIDAR-Lite-Data-Sheet.pdf

[51] **Monsolar.** Monsolar. [En línea] 2015. [Consultado el 19 de Mayo de 2021] https://www.monsolar.com/blog/baterias-de-ion-litio-ventajas-e-inconvenientes/#:~:text=Inconvenientes%20de%20las%20bater%C3%ADas%20de,la%20temperatura%20de%20las%20celdas

ANEXO

Código de programación en entorno de Arduino.

```
  /*

  Wheelchair control adaptation.

  Motor movements through joystick and using mobile using bluetooh.

  ************************include*******************************
  -Control via bluetooth.

  -Speed buttons.

  -Acceleration ramp.

  -Chair movements with the joystick.

  -Movements of the chair with the mobile phone.

  -buzzer button

  -LED battery indicator

  -Battery indicator button


  Thanks to:
   -www.luisllamas.com for the library "BTS79602M.h" (modified by me
to this proyect)


  Created in 2020.

  By David Flores Román.
  dfloresr@alumnos.unex.es

  Modified on May 31, 2021.

  */

  ////////////////////////////////////////////////////////////////
//////////////
  ////////////////////// Libraries and includes
/////////////////////////////////////
  ////////////////////////////////////////////////////////////////
//////////////

  //bluetooth libraries
  #define CUSTOM_SETTINGS
```

```
  #define INCLUDE_GAMEPAD_MODULE
  #include <DabbleESP32.h>

  //Motors controlls libraries
  #include "BTS79602M.h"

  #include <analogWrite.h>

  ////////////////////////////////////////////////////////////////////
//////////////
  ////////////////////// Variable definitions
//////////////////////////////////////
  ////////////////////////////////////////////////////////////////////
//////////////

  //motors electric brake
  int controlPIN= 12;

  // delay
  int V_update_period=80; //time in milliseconds
  unsigned long V_last_update = 0; //time since ESP32 is activated

  //Motor variable
  ////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////
  const uint8_t EN1 = 19; //power for BTS1
  const uint8_t L_PWM1 = 18;//pin to turn the motor1 to the left
  const uint8_t R_PWM1 = 5;//pin to turn the motor1 to the Right

  const uint8_t EN2 = 4;//power for BTS2
  const uint8_t L_PWM2 = 16;//pin to turn the motor2 to the left
  const uint8_t R_PWM2 = 17;//pin to turn the motor2 to the Right

  BTS79602M motorController(EN1, L_PWM1, R_PWM1, EN2, L_PWM2,
R_PWM2); // initialize motorController, function of BTS79602M.h

  //battery indicator with LEDs
  ////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////

  float Div_v = 0; //battery power in volts
  int Div = 0; //value battery power

  //definition of LEDs
  ////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////
  const int ledPIN1 = 23;//green LED
  const int ledPIN2 = 22;//yellow LED
  const int ledPIN3 = 21; //red LED
  const int ledPIN4 = 0; //green LED2
  const int ledPIN5 = 2; //green LED3

  //button to turn on the LEDs

  const int buttonLED_PIN = 15; // pin LED boton

  bool aux = 0; // auxiliary button variable

  //joystick variables
```

```
  ////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////
  int Xvalue = 0; // X axis joystick
  int Yvalue = 0; // Y axis joystick
  int JX_PIN = 35; // pin 35 X axis
  int JY_PIN = 32; // pin 32 Y axis

  int threshold=200; //midpoint threshold
  int Pmean=2048;//midpoint

  int X = 0; //saves X axis between -2048 to 2048
  int Y = 0; //saves Y axis between -2048 to 2048
  float auxi = 0; //auxiliary variable

  int Angle; //variable for angulos function
  int ang=0; // variable for saving the value of the angulos
function
  int distance; //variable for range function

  int v=0;//pwm for motors 0 to 255

  int dif1=0; //speed difference as a function of angle to move
forward and left
  int dif2=0; //speed difference as a function of angle to move back
and left
  int dif3=0; //speed difference as a function of angle to move
forward and right
  int dif4=0;//speed difference as a function of angle to move back
and right

  //speed buttons
  ////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////
  const int buttonHighPIN = 33; //pin 33 for high speed button
  const int buttonLowPIN = 25; //pin 25 for low speed button
  bool flag_b=0; //flag to switch from one button to another

  int buttonHigh = 0;//saves information of pin 33
  int buttonLow = 0; //saves information of pin 25

  //button and buzzer variable
  ////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////
  const int BuzzButtonPIN = 26; // buzzer button
  const int BuzzPIN = 27; // voltage buzzer
  int lecb; //auxiliary variable

  //flag forward and back
  ////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////
  int flag_forward=0; //forward
  int flag_back=0; // back

  int v_d=0; //real speed of the motors


  int local; // auxiliary variable used in function angulos
  int TOL=5; //tolerance of velocity comparal
  int INC_v=3; //increase of real velocity
```

```
  ////////////////////////////////////////////////////////////////
/////////////////
  ////////////////////////////////Setup///////////////////////////
/////////////////
  ////////////////////////////////////////////////////////////////
/////////////////

  void setup() {

    //Serial.begin(115200);

    Dabble.begin("MyEsp32");       // bluetooth name

    ////////////////////////////////////////////////////////////////
/////////////
    pinMode(ledPIN1 , OUTPUT); //initialize digital pin ledPIN1 as
an OUTPUT
    pinMode(ledPIN2 , OUTPUT);//initialize digital pin ledPIN2 as an
OUTPUT
    pinMode(ledPIN3 , OUTPUT);//initialize digital pin ledPIN3 as an
OUTPUT
    pinMode(ledPIN4 , OUTPUT);//initialize digital pin ledPIN4 as an
OUTPUT
    pinMode(ledPIN5 , OUTPUT);//initialize digital pin ledPIN5 as an
OUTPUT

    pinMode(BuzzPIN, OUTPUT);//initialize digital pin PINz as an
OUTPUT
    pinMode(controlPIN, OUTPUT);//initialize digital pin control  as
an OUTPUT

    pinMode(JX_PIN,INPUT);//initialize analog pin pinJX as an INPUT
    pinMode(JY_PIN,INPUT);//initialize analog pin pinJY as an INPUT

    pinMode(buttonLED_PIN  , INPUT_PULLDOWN);//initialize digital
pin boton as an INPUT_PULLDOWN
    pinMode(buttonHighPIN, INPUT_PULLDOWN);//initialize digital pin
PINjoy1 as an INPUT_PULLDOWN
    pinMode(buttonLowPIN, INPUT_PULLDOWN);//initialize digital pin
PINjoy2 as an INPUT_PULLDOWN
    pinMode(BuzzButtonPIN, INPUT_PULLDOWN);//initialize digital pin
PINbotz as an INPUT_PULLDOWN

  }

  ////////////////////////////////////////////////////////////////
/////////////////////////////////////////////////////
  ////////////////////////////////////////////////////Loop//////////
/////////////////////////////////////////////////////
  ////////////////////////////////////////////////////////////////
/////////////////////////////////////////////////////

  void loop() {

    Dabble.processInput(); // activate bluetooth to use the app

    motorController.Enable(); //active motors

    //joystick readings
```

```
    ////////////////////////////////////////////////////////////////
////////////////////////////////
    Xvalue = analogRead ( JX_PIN); // reading X axis
    Yvalue = analogRead ( JY_PIN); // reading Y axis

  //change X and Y values from -2048 to 2048
  ////////////////////////////////////////////////////////////////
////////////////////////////////
    X=Xvalue-Pmean;

if(abs(X) < threshold){
    X=0;
}
                              //X and Y are 0 when the joystick is in
the threshold
Y=Yvalue-Pmean;

if(abs(Y) < threshold){
   Y=0;
}

  //test code
  ////////////////////////////////////////////////////////////////
////////////////////////////////

   // Serial.print("ejeX;  ");
   // Serial.print(X);
    //Serial.print("  ejeY;  ");
    //Serial.print(Y);
    //Serial.print("   valor de v: ");
    //Serial.print(v);

    //Serial.print ("angulo: ");
    //Serial.print(angulos (X,Y));

    //Serial.print ("   velocidad: ");
    //Serial.println(v);

  ////////////////////////////////////////////////////////////////
////////////////////////////////

    //readings speed buttons
    buttonHigh = digitalRead(buttonHighPIN);
    buttonLow = digitalRead(buttonLowPIN);



  //change from one speed button to another
if(buttonHigh==HIGH && buttonLow==LOW && v==0){

  flag_b=1;
}

if(buttonLow==HIGH && buttonHigh==LOW && v==0){

  flag_b=0;
}

  //if flag_b is 0, speed is LOW and if flag_b is 1, speed is HIGH
if (flag_b == 0) {
```

```
       if(millis() > V_last_update + V_update_period){
            V_last_update = millis();

            v_d=(distancia(X, Y)/13);
                                  //if we have to increase speed
                  if(v_d - v > TOL){  //comparison between desired
speed (V_d)  and actual speed (v)
                    v=v+INC_v;
                          }
        if ((v_d-v < TOL) && v>0){ //if we have to decrease speed
            v=v-INC_v;
                           }

                  if(v<0){ //v can never be negative
                     v=0;

                          }

          }
 }

else {

        if(millis() > V_last_update + V_update_period){
              V_last_update = millis();

               v_d=(distancia(X, Y)/20)*2;
                                     //if we have to increase speed
                  if(v_d - v > TOL){ //comparison between desired
speed (V_d)  and actual speed (v)
                        v=v+INC_v;
                                }
                 if ((v_d-v < TOL) && v>0){  //if we have to
decrease speed
                         v=v-INC_v;
                               }

                     if(v<0){ //v can never be negative
                       v=0;

                          }

             }

 }

    ang=angulos(X,Y,v);

  ////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////
///////////
    //movement of the motors according to the joystick

    //left
if (ang >= 15 && ang <= 165 && v>0 ) { //range of values depending
on the quadrant in which the joystick is located
     // Serial.println("L");
      if (flag_forward == 1 && flag_back == 0){ //if it has gone
forward before, turn left with an engine turning faster
```

```
          dif1=abs(165-angulos(X,Y,v));

            if((v-dif1)<=1){
               dif1=v-1;
                        }

           digitalWrite(controlPIN, HIGH);
           //delay(100);

           motorController.TurnLeftM2(v);
           motorController.TurnLeftM1((v)-dif1);

      }

      if(flag_back == 1 && flag_forward == 0){ //if it has gone
backwards before, turn left with an engine turning faster

      dif2=abs(15-angulos(X,Y,v));

            if((v-dif2)<=1){
               dif2=v-1;
                   }

      digitalWrite(controlPIN, HIGH);
      //delay(100);

      motorController.TurnRightM2(v);
      motorController.TurnRightM1((v)-dif2);

      }

       if(flag_forward == 0 && flag_back == 0){ // if the joystick
was initially at rest, turn left with one wheel forward and one
wheel back
           digitalWrite(controlPIN, HIGH);
            //delay(100);

          motorController.TurnLeftM2(v/2);
           motorController.TurnRightM1(v/2);
                           }
  }

    //right
if (ang >= 195 && ang <= 345 && v>0) {//range of values depending on
the quadrant in which the joystick is located
  //Serial.println("R");
      if (flag_forward == 1 && flag_back == 0){//if it has gone
forward before, turn right with an engine turning faster

      dif3=abs(195-angulos(X,Y,v));

          if((v-dif3)<=1){
              dif3=v-1;
                  }

          digitalWrite(controlPIN, HIGH);
          //delay(100);

          motorController.TurnLeftM2((v)-dif3);
```

```
          motorController.TurnLeftM1(v);

      }
      if(flag_back == 1 && flag_forward == 0){//if it has gone
backwards before, turn right with an engine turning faster

          dif4=abs(345-angulos(X,Y,v));

              if((v-dif4)<=1){
                  dif4=v-1;
                           }

         digitalWrite(controlPIN, HIGH);
        // delay(100);

         motorController.TurnRightM2((v)-dif4);
         motorController.TurnRightM1(v);

      }

      if(flag_forward == 0 && flag_back == 0){// if the joystick was
initially at rest, turn right with one wheel forward and one wheel
back

         digitalWrite(controlPIN, HIGH);
         // delay(100);

         motorController.TurnRightM2(v/2);
         motorController.TurnLeftM1(v/2);
         }

  }

    //forward
if (ang >= 170 && ang <= 190 && v>0 ) {//range of values depending
on the quadrant in which the joystick is located
      //Serial.println("F");

         if(flag_back==0 && flag_forward==0){
         flag_forward=1;
                   }

         if(flag_forward == 1 && flag_back==0){ //the chair goes
forward unless the joystick is moved from state to another bypassing
0

             digitalWrite(controlPIN, HIGH);

              motorController.TurnLeftM2(v);
                motorController.TurnLeftM1(v);

                           }
         if(flag_back==1 &&flag_forward==0){

              motorController.TurnRightM2(v);
                motorController.TurnRightM1(v);

                          }

}
```

```
    //backward
if (((ang >= 0 && ang <= 10) || (ang >= 350 && ang <= 355)) && v>0)
{//range of values depending on the quadrant in which the joystick
is located
  //Serial.println("B");

      if(flag_forward==0 && flag_back==0){
            flag_back = 1;
                        }

      if(flag_back == 1 && flag_forward==0){//the chair goes
backwards unless the joystick is moved from state to another
bypassing 0

            digitalWrite(controlPIN, HIGH);

            motorController.TurnRightM2(v);
            motorController.TurnRightM1(v);

                        }
            if(flag_forward==1 && flag_back==0){

            motorController.TurnLeftM2(v);
            motorController.TurnLeftM1(v);

                    }

 }

  ////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////

  //Serial.println(ang);

  aux = digitalRead(buttonLED_PIN); //reading pin 15

  //voltage divider

    Div = analogRead(34); //reading pin 34

    Div_v = ((Div * 25)/2300); //range conversion operation using
ESP32 (0-2300) to volts.


    lecb = digitalRead(BuzzButtonPIN); //reading pin buzzer button

    //motors movement with bluetooth
    ////////////////////////////////////////////////////////////////////
////////////////////////////////////

  //code test
  //Serial.print("velocidad deseada: ");
  //Serial.print(v_d);
  //Serial.print("         velocidad real: ");
  //Serial.println(v);

    //forward
    ////////////////////////////////////////////////////////////////////
////////////////////////////////////
```

```
  if (GamePad.isUpPressed() == 1 && GamePad.isLeftPressed() == 0 &&
GamePad.isRightPressed() == 0 && GamePad.isDownPressed() == 0 && v
== 0) {
      digitalWrite(controlPIN, HIGH);
      delay(100);
      motorController.TurnLeftM2(50);
      motorController.TurnLeftM1(50);

    }

  //back
  ////////////////////////////////////////////////////////////////////
////////////////////////////////

  if (GamePad.isDownPressed() == 1  && GamePad.isUpPressed() == 0 &&
GamePad.isLeftPressed() == 0 && GamePad.isRightPressed() == 0 && v
== 0) {
      digitalWrite(controlPIN, HIGH);
      delay(100);
      motorController.TurnRightM2(50);
      motorController.TurnRightM1(50);

    }

  //Stop motors
  ////////////////////////////////////////////////////////////////////
////////////////////////////////
  if (GamePad.isDownPressed() == 0 && GamePad.isUpPressed() == 0 &&
GamePad.isLeftPressed() == 0 && GamePad.isRightPressed() == 0 &&
v==0) {
     motorController.Stop();
     motorController.Disable();
     digitalWrite(controlPIN, LOW);

     flag_forward=0;
     flag_back=0;

    }

    //Left
    ////////////////////////////////////////////////////////////////////
////////////////////////////////////
  if (GamePad.isLeftPressed () == 1 && GamePad.isDownPressed() == 0
&& GamePad.isUpPressed() == 0  && GamePad.isRightPressed() == 0  &&
v == 0) {
      digitalWrite(controlPIN, HIGH);
      delay(100);
      motorController.TurnLeftM2(50);
      motorController.TurnRightM1(50);

    }

    //right
    ////////////////////////////////////////////////////////////////////
////////////////////////////////////
  if (GamePad.isLeftPressed () == 0 && GamePad.isDownPressed() == 0
&& GamePad.isUpPressed() == 0  && GamePad.isRightPressed() == 1 && v
== 0) {
      digitalWrite(controlPIN, HIGH);
      delay(100);
```

```
      motorController.TurnRightM2(50);
      motorController.TurnLeftM1(50);

    }

    //LEDs
    ////////////////////////////////////////////////////////////////////
////////////////////////////////////

    leds(aux, Div_v); //leds function call

  //buzzer
  ////////////////////////////////////////////////////////////////////
////////////////////////////////////////////

  if (lecb == HIGH) { //if button is Hight buzzer sounds

   digitalWrite(BuzzPIN, HIGH); //buzzer in high

         }

  else {

     digitalWrite(BuzzPIN, LOW); //buzzer in low

        }

}

  ////////////////////////////////////////////////////////////////////
////////////////////////////////////////////
  ////////////////////////////////////////////////Functions////////////
////////////////////////////////////////////
  ////////////////////////////////////////////////////////////////////
////////////////////////////////////////////

  //leds function
  ////////////////////////////////////////////////////////////////////
/
void leds(int aux, int Div_v) { //button and value of voltage

    if (aux == 1) {

      if (Div_v >= 24 ) {

        digitalWrite(ledPIN1 , HIGH);
        digitalWrite(ledPIN2 , HIGH);
        digitalWrite(ledPIN3 , HIGH);
        digitalWrite(ledPIN4 , HIGH);
        digitalWrite(ledPIN5 , HIGH);

                                    }

      if (Div_v >= 20 && Div_v < 24 ) {
        digitalWrite(ledPIN1 , HIGH);
        digitalWrite(ledPIN2 , HIGH);
        digitalWrite(ledPIN3 , HIGH);
        digitalWrite(ledPIN4 , HIGH);
        digitalWrite(ledPIN5 , LOW);
```

```
                                    }

      if (Div_v <= 18 && Div_v < 20 ) {

        digitalWrite(ledPIN1 , HIGH);
        digitalWrite(ledPIN2 , HIGH);
        digitalWrite(ledPIN3 , HIGH);
        digitalWrite(ledPIN4 , LOW);
        digitalWrite(ledPIN5 , LOW);

                                    }

      if (Div_v <= 15 && Div_v < 18 ) {

        digitalWrite(ledPIN1 , LOW);
        digitalWrite(ledPIN2 , HIGH);
        digitalWrite(ledPIN3 , HIGH);
        digitalWrite(ledPIN4 , LOW);
        digitalWrite(ledPIN5 , LOW);

                                     }

      if (Div_v < 15) {

        digitalWrite(ledPIN1 , LOW);
        digitalWrite(ledPIN2 , LOW);
        digitalWrite(ledPIN3 , HIGH);
        digitalWrite(ledPIN4 , LOW);
        digitalWrite(ledPIN5 , LOW);

                                     }

    }
  else if (aux == 0) {
      digitalWrite(ledPIN1 , LOW);
      digitalWrite(ledPIN2 , LOW);
      digitalWrite(ledPIN3 , LOW);
      digitalWrite(ledPIN4 , LOW);
      digitalWrite(ledPIN5 , LOW);
                                      }

}

  //angulos function
int angulos (int x, int y, int V) { //returns the angle from 0 to
359 at which the joystick is, calculating its arc tangent.

    auxi = atan2(x, y);
    Angle = (auxi * 180) / PI;

   if (Angle < 0) {
      Angle = Angle + 360;
    }

       if(v>0 && distancia(X,Y)>0 ){
         local=Angle;
                 }

           if( v==0 && distancia(X,Y)==0){
               return 14;
```

```
                    }

            else
                if( v>0 && distancia(X,Y)==0){
                    return local;
                        }
                          else
                             return Angle;
 }

  //Distance function
int distancia(int x, int y) { //returns the distance from the center
of the joystick doing the modulus.

    distance = (int)sqrt(pow(x, 2) + pow(y, 2));

    return distance;
  }
```