UNIVERSIDAD NACIONAL "PEDRO RUIZ GALLO"

FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS

ESCUELA PROFESIONAL DE INGENIERÍA ELECTRÓNICA

DISEÑO DE UN SISTEMA DE CONTROL, MONITOREO Y SEGURIDAD PARA MEJORAR LAS ETAPAS DE UN MODELO DE SILLA DE RUEDAS MOTORIZADA PARA DISCAPACITADOS - HOSPITAL REGIONAL LAMBAYEQUE

TESIS

PARA OPTAR EL TÍTULO PROFESIONAL DE

INGENIERO ELECTRÓNICO

PRESENTADO POR:

MANRIQUE ARANCIBIA, ROGER DENNIS
ESCOBAR MURO, MIGUEL ANGEL

ASESOR
ING. VICTOR JARA SANDOVAL

LAMBAYEQUE – PERÚ
2016

UNIVERSIDAD NACIONAL PEDRO RUIZ GALLO

FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS

ESCUELA PROFESIONAL DE INGENIERÍA ELECTRÓNICA

DISEÑO DE UN SISTEMA DE CONTROL, MONITOREO Y SEGURIDAD PARA MEJORAR LAS ETAPAS DE UN MODELO DE SILLA DE RUEDAS MOTORIZADA PARA DISCAPACITADOS - HOSPITAL REGIONAL LAMBAYEQUE

TESIS

PARA OPTAR EL TÍTULO PROFESIONAL DE:

INGENIERO ELECTRÓNICO

PRESENTADO POR:

MANRIQUE ARANCIBIA, ROGER DENNIS
ESCOBAR MURO, MIGUEL ANGEL

ASESOR
ING. VICTOR JARA SANDOVAL

LAMBAYEQUE – PERÚ
2016

UNIVERSIDAD NACIONAL PEDRO RUIZ GALLO

FACULTAD DE CIENCIAS FISICAS Y MATEMATICAS

ESCUELA PROFESIONAL DE INGENIERÍA ELECTRÓNICA

DISEÑO DE UN SISTEMA DE CONTROL, MONITOREO Y SEGURIDAD PARA MEJORAR LAS ETAPAS DE UN MODELO DE SILLA DE RUEDAS MOTORIZADA PARA DISCAPACITADOS - HOSPITAL REGIONAL LAMBAYEQUE

TESIS

PARA OPTAR EL TÍTULO PROFESIONAL DE:
INGENIERO ELECTRÓNICO

SUSTENTADA POR:

MANRIQUE ARANCIBIA, ROGER DENNIS
AUTOR

ESCOBAR MURO, MIGUEL ANGEL
AUTOR

APROBADA POR:

ING.MANUEL JAVIER RAMIREZ CASTRO
PRESIDENTE

ING.CARLOS LEONARDO OBLITAS VERA
SECRETARIO

ING.LUCIA ISABEL CHAMAN CABRERA
VOCAL

ING. VICTOR OLEARIO JARA SANDOVAL
ASESOR

Ing. Hugo Javier Chiclayo Padilla
DIRECTOR ESCUELA PROF. DE INGENIERIA ELECTRÓNICA
FACFYM - U.N.P.R.G.

DEDICATORIA

Con mucho amor a las personas más importantes en mi vida, que siempre me brindaron su apoyo y estuvieron conmigo en todo momento, para así yo hiciera posible que lograrà todos mis sueños, con todo mi cariño para ustedes: papá y mamá.

Roger Dennis

Con todo mi cariño y mi amor para las personas que hicieron todo en la vida para que yo pudiera lograr mis sueños, por motivarme y darme la mano cuando sentía que el camino se terminaba, a ustedes por siempre mi corazón y mi agradecimiento: papá y mamá.

Miguel Ángel

AGRADECIMIENTOS

En primer lugar a Dios, por darnos ese tesoro más hermoso que puede existir, como es la vida. Y por darnos esa fortaleza para seguir adelante en busca de nuestras aspiraciones y/o metas que nos hemos propuesto en los diferentes aspectos de nuestras vidas.

A nuestros padres, por ese apoyo incondicional para con nosotros, por sus consejos y aliento que nos brindan día a día, sobre todo en esos momentos más difíciles.

A nuestros amigos, por darnos ánimos y su ayuda para realizar este proyecto.

Al ingeniero Víctor Oleario Jara Sandoval quien nos brindó su asesoría en la preparación de este informe y que con paciencia y buena voluntad atendió amablemente a nuestras inquietudes y/o consultas.

Los autores

HOMENAJE PÓSTUMO AL INGENIERO VÍCTOR JARA SANDOVAL

Unos de los sentidos más profundos del ser humano es el de trascender más allá de la mera existencia terrenal. Todos buscamos de distintas maneras alcanzar este sentido. Entre quienes en verdad lo logran existen siempre dos atributos que les son comunes: el amor que imponen en todo lo que emprenden y la honestidad con que rigen su conducta en todo momento.

Hoy rendimos homenaje a uno de esos hombres, a un ilustre ingeniero lambayecano y amigo, cuya obra ha trascendido su tiempo y su espacio, un hombre de bien, un visionario, un maestro de saber y tesón, un dirigente, un hombre inagotable en cualidades.

La vida solo tiene utilidad si cada día la trocamos por algo de valor. Siendo el recurso más valioso que poseemos, un recurso irrecuperable, no renovable, la clave para aprovecharlo esté en decidir si vamos a invertirlo o a derrocharlo... Y el ingeniero Víctor Jara sí que supo invertirlo día a día.

A lo largo de su muy fructífera vida, el amigo y profesor Víctor Jara Sandoval, el ingeniero, el científico, el humanista, el servidor público, el dirigente de la escuela profesional de ingeniería electrónica de la Universidad Nacional Pedro Ruiz Gallo, y sobre todo su capacidad de ser y hacer amigos, supo encontrar los caminos para construir con amor y honestidad un espléndido legado que mantendrá viva su memoria y permanente gratitud por su generosidad. Este legado incluye desde luego haber llegado a la Universidad Nacional Pedro Ruiz Gallo como ingeniero electrónico, y ser uno de los impulsadores de esa moderna carrera profesional para su tiempo, asimismo colaboró en solucionar las innumerables dificultades de sus alumnos en su especialidad y posteriormente liderar la Jefatura de la escuela profesional de ingeniería electrónica con eficiencia, responsabilidad y cariño a sus alumnos y colegas. Amante de la puntualidad como pocos y del desarrollo de su escuela profesional.

En nuestra calidad de egresados de la escuela profesional de ingeniería electrónica de la Universidad Nacional Pedro Ruiz Gallo, y como alumnos y futuros colegas, damos fe, que nuestras vidas se vio' enriquecida por el contacto directo con él, rendimos homenaje a este hombre excepcional, cuyo legado vivirá por siempre en sus contribuciones a la ingeniería y docencia de nuestra universidad y de la Región del Norte del país. Homenaje póstumo que expresamos, en nombre de todos los alumnos y amigos de trabajo. Gracias una vez más Maestro. Descansa en Paz.

Gracias.

Los Autores

RESUMEN

Se investigará, estudiará y mejorará el funcionamiento de las etapas de un modelo de silla de ruedas motorizada para discapacitados, asegurándonos de lograr una óptima interacción con las personas a las cuales están destinadas, utilizando materiales de bajo costo, de fácil accesibilidad para cualquier persona y disponibles en la mayoría de tiendas electrónicas locales.

El núcleo principal del proyecto será el microcontrolador STM32F103RBT6 perteneciente a la familia Arm cortex M3 el cual se encargará de hacer la comunicación con los diferentes periféricos que la componen y de la comunicación entre la etapa de potencia y la etapa de control.

Básicamente el sistema esta alimentado por dos baterías de 12v. Cada una, las cuales hacen girar a dos motores del tipo A9Y1X00272, de 24v. , 15A y con una revolución de 172 RPM cada uno, a continuación se encuentra la etapa de potencia, que básicamente su función es controlar los movimientos de los motores, donde se utilizaron principalmente transistores (IRFP150N), integrados (IR2112), reguladores de voltaje y componentes menores.

En la etapa de control, que es la parte que interactúa con el usuario, se utiliza una pantalla táctil (en la que se encuentran los diferentes menús), un control de videojuegos (joystick) y un bluetooth; la comunicación puede ser de dos modos: comunicación manual y comunicación de manera remota (bluetooth), con un rango de 15 metros. Cabe señalar que la comunicación vía bluetooth es controlada por un Smartphone, el cual utiliza un software, en nuestro caso es el programa App inventor.

Por último, podemos dar cuenta de las ventajas de las nuevas aplicaciones que se le ha añadido a la silla eléctrica, como la comunicación vía bluetooth, un mejor control de velocidad, tratando de lograr una calidad de vida más plena para aquellas personas que cuenten con cierta discapacidad.

Contribuyendo de esta manera así a la mejora de la sociedad.

ABSTRACT

We investigate, study and improve the functioning of the stages of a model of scooter for disabled, ensuring optimal interaction with the people they are intended, using inexpensive materials, easily accessible for anyone and available in most local electronic stores.

The core of the project will be the microcontroller STM32F103RBT6 belonging to the family ARM Cortex M3 which was to be undertaken to make communication with peripherals that compose and communication between the power unit and the control stage.

Basically the system is powered by two 12V batteries. Each, which rotate two motors A9Y1X00272 type 24v. 15A and a revolution of 172 rpm each, then is the power stage, which basically their function is to control the movements of the engines, which mainly transistors (IRFP150N), integrated (IR2112), voltage regulators used and minor components.

In the control stage, which it is the part that interacts with the user, a touch screen (which are different menus), video game control (joystick) and a Bluetooth used; communication can be of two modes: manual and remote communication (Bluetooth), with a range of 15 meters. It notes that Bluetooth communication is controlled by a smartphone, which uses software in our case is the App Inventor program.

Finally, we can realize the benefits of new applications has been added to the electric chair, as communication via Bluetooth, better speed control, trying to achieve a quality of life more fulfilling for those who have some disability.

And thus contributing to the improvement of society.

INDICE

INDICE DE FIGURAS

I. ASPECTOS GENERALES

1.1. Generalidades de la empresa

1.1.1. Situación Problemática

El Hospital Regional Lambayeque, es una de las instituciones de mayor importancia y complejidad de la Región Lambayeque, tanto que constituye un Hospital con nivel III-1, es decir nosocomio de ALTA COMPLEJIDAD, ubicado en la ciudad de Chiclayo - VÍA DE EVITAMIENTO NORTE CON AVENIDA EL PROGRESO (Fig.1.1). Este hospital en el servicio de medicina física y rehabilitación atienden a personas con discapacidad para tratamiento especializado, la gran mayoría para personas con discapacidad motora. Estas sillas de ruedas motorizadas controlan la velocidad del motor utilizando para su control tecnología desfasada, ocasionando un inadecuado control de variables involucradas en dicho proceso.
La constante necesidad de mejorar las distintas etapas que intervienen en un modelo de silla de ruedas motorizada para discapacitados, demandó que se ejecuten modernizaciones respecto al diseño inicial de dichas instalaciones, así como una nueva programación teniendo en cuenta aplicaciones de seguridad como redundancia aplicada a las variables controladas con el fin de ayudar en todo momento al discapacitado.

Tomando en cuenta la necesidad de El Hospital Regional Lambayeque, de mejorar las etapas de un modelo de modelo de silla de ruedas motorizada para discapacitados del servicio de medicina física y rehabilitación, se deben modificar las instalaciones y configurar el proceso para que trabaje de manera óptima.

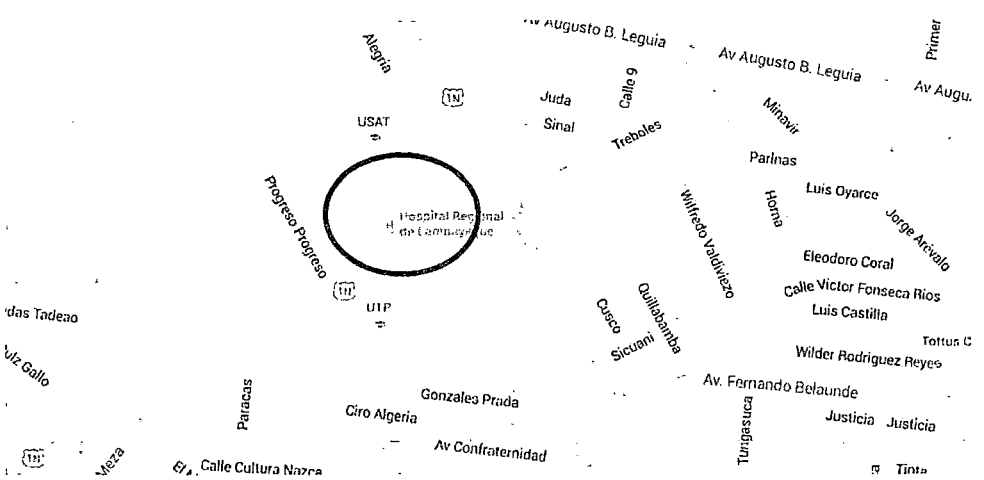


Fig.1.1. ubicación geográfica de El Hospital Regional Lambayeque

1.2. Diagnóstico de la situación actual

El Hospital Regional Lambayeque, actualmente cuenta con el servicio de MEDICINA FISICA Y REHABILITACION, en dónde se realizan tratamientos y rehabilitación a pacientes de toda la región y departamentos anexos.

En la actualidad existe un porcentaje de personas con discapacidad motora que como consecuencia de ello se ven en la necesidad de utilizar sillas de ruedas del tipo Manuales y Motorizadas, así mismo se evidencia que las personas que cuentan con sillas eléctricas motorizadas gran parte tiene problemas con el funcionamiento de su medio de transporte:

- Componentes deteriorados y costosos en el caso de que se desee comprar (importación).
- Costos de mantenimientos excesivos preventivos y/o correctivos por personal cualificado o de fábrica para dichas sillas motorizadas; esto da como consecuencia que dichas sillas queden inoperativas al no poder acceder a dichos mantenimientos.

Por lo antes expuesto hay una carencia de un sistema de control (software y hardware) monitoreo y seguridad necesario para poder operar y mejorar dichas sillas eléctricas motorizadas toda vez que se requiera tener una solución accesible y de bajo costo para los usuarios finales (personas con discapacidad).

1.3. Delimitación del proyecto

1.3.1. Formulación del problema

¿El diseño de un sistema de control y monitoreo mejorará las etapas de un modelo de silla de ruedas motorizada para discapacitados - hospital regional Lambayeque?

1.3.2. Hipótesis

El diseño de un sistema de control y monitoreo sí mejorará las etapas de un modelo de silla de ruedas motorizada para discapacitados - hospital regional Lambayeque

1.3.3. Objetivos

1.3.3.1. Objetivo General

Diseñar un sistema de control y monitoreo para mejorar las etapas de un modelo de silla de ruedas motorizada para discapacitados - hospital regional Lambayeque.

1.3.3.2. Objetivos Específicos

- Plantear el diseño para el sistema de control.
- Realizar la selección de equipos y el diseño final
- Poner en marcha la ejecución del diseño final

1.3.4. Justificación e importancia

Porque se quiere dar una solución a bajo coste y accesible para las personas con discapacidad, así como mejorar las etapas de un modelo de silla de ruedas motorizada para discapacitados para así lograr beneficiar en forma directa al hospital Regional Lambayeque y servicio de rehabilitación.

Este proyecto permitirá aumentar el número de pacientes en el hospital Regional Lambayeque por realizar un valor agregado al servicio de rehabilitación y tratamiento así como mejorar y/o optimizar en el funcionamiento de un modelo de silla de ruedas motorizada.

I. MARCO TEORICO

2.1. Sistemas de control

Un sistema dinámico puede definirse conceptualmente como un ente que recibe unas acciones externas o variables de entrada, y cuya respuesta a estas acciones externas son las denominadas variables de salida.

Las acciones externas al sistema se dividen en dos grupos, variables de control, que se pueden manipular, y perturbaciones sobre las que no es posible ningún tipo de control. La Figura 2.1 ilustra de un modo conceptual el funcionamiento de un sistema.

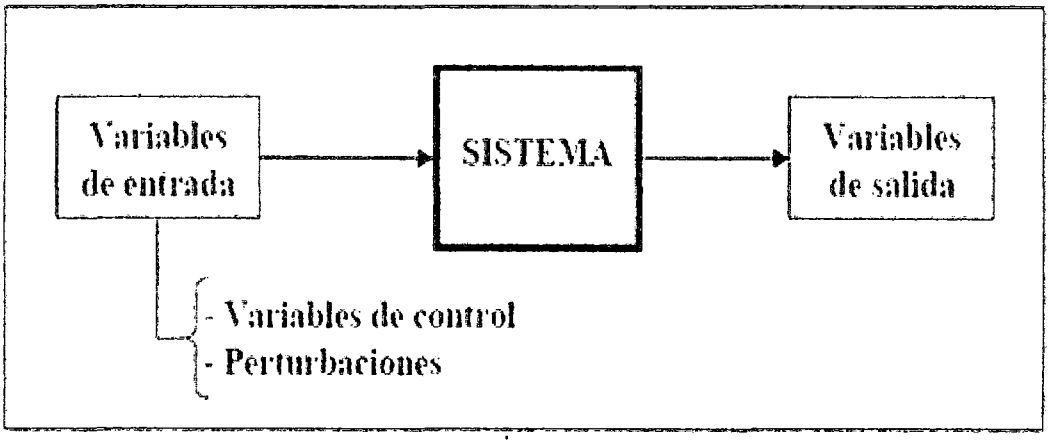


Fig.2.1. modo conceptual del funcionamiento de un sistema de control

Control

Es un tipo de sistema que se caracteriza por la presencia de una serie de elementos que permiten influir en el funcionamiento del sistema. La finalidad de un sistema de control es conseguir, mediante la manipulación de las variables de control, un dominio sobre las variables de salida, de modo que estas alcancen unos valores prefijados (consigna). Un sistema de control ideal debe ser capaz de conseguir su objetivo cumpliendo los siguientes requisitos:

Garantizar la estabilidad y, particularmente, ser robusto frente a perturbaciones y errores en los modelos.

Ser tan eficiente como sea posible, según un criterio preestablecido. Normalmente este criterio consiste en que la acción de control sobre las variables de entrada sea realizable, evitando comportamientos bruscos e irreales.

Ser fácilmente implementado y cómodo de operar en tiempo real con ayuda de un ordenador. Los elementos básicos que forman parte de un sistema de control y permiten su manipulación son los siguientes:

Sensores. Permiten conocer los valores de las variables medidas del sistema.

Controlador. Utilizando los valores determinados por los sensores y la consigna impuesta, calcula la acción que debe aplicarse para modificar las variables de control en base a cierta estrategia.

Actuador. Es el mecanismo que ejecuta la acción calculada por el controlador y que modifica las variables de control.

La Figura 2.2. ilustra el esquema de funcionamiento de un sistema de control genérico.

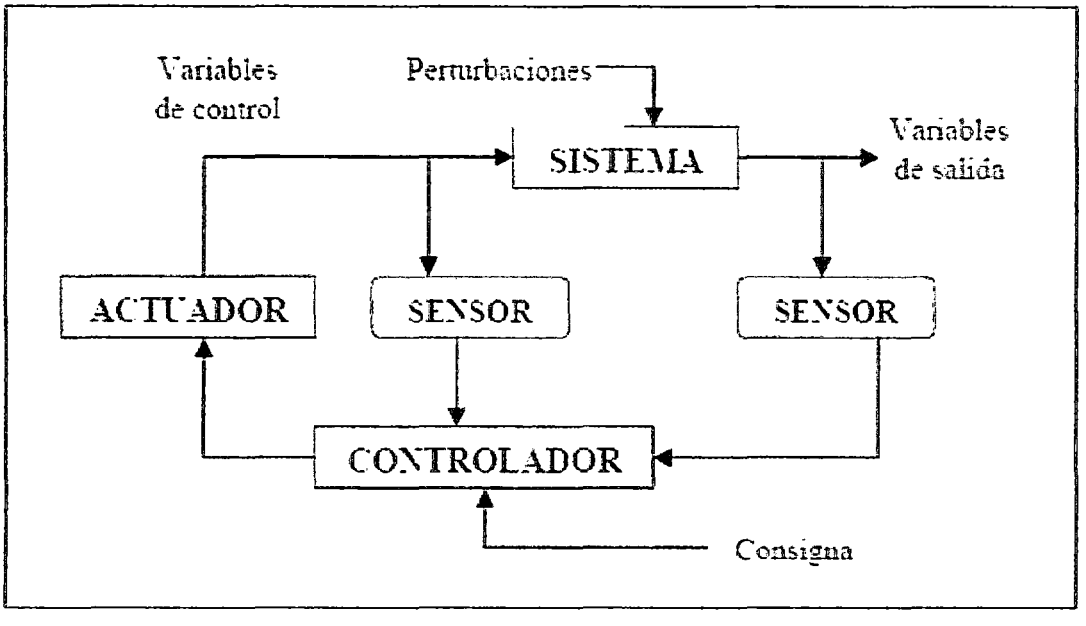


Fig.2.2. Esquema general de un Sistema de Control

2.2. Control Proporcional

Un controlador proporcional calcula la diferencia entre la señal de variable de proceso y la señal de setpoint, lo que vamos a llamar como error. Este valor representa cuanto el proceso se está desviando del valor del setpoint, y puede ser calculado como SP-PV o como PV-SP, dependiendo si es que o no el controlador tiene que producir un incremento en su señal de salida para causar un incremento en la variable de proceso, o tener un decremento en su señal de salida para hacer de igual manera un incremento de PV (variable de proceso).

Los controladores proporcionales nos dan la opción de decirle que tan "sensible" deseamos que el controlador se comporte entre cambios en la variable de proceso (PV) y setpoint (SP).

Entonces aquí, nosotros programamos al controlador para cualquier nivel de agresividad del controlador. La ganancia (Kp) de un controlador es algo que podemos alterar, en controladores analógicos tomará la forma de un potenciómetro, en sistemas de control digitales será un parámetro programable.

Una manera de expresar la "sensibilidad" de la acción proporcional, y que es la inversa de la ganancia (Kp) llamada Banda Proporcional (BP):

$$Kp = 1/PB \quad ; \quad PB = 1/Kp$$

2.3. Sensores electrónicos

El sensor está siempre en contacto con la variable de instrumentación con lo que puede decirse también que es un dispositivo que aprovecha una de sus propiedades con el fin de adaptar la señal que mide para que la pueda interpretar otro dispositivo. Como por ejemplo el termómetro de mercurio que aprovecha la propiedad que posee el mercurio de dilatarse o contraerse por la acción de la temperatura.

Un sensor también puede decirse que es un dispositivo que convierte una forma de energía en otra.

Áreas de aplicación de los sensores: Industria automotriz, Industria aeroespacial, Medicina, Industria de manufactura, Robótica, etc., Los sensores pueden estar conectados a un computador para obtener ventajas como son el acceso a una base de datos, la toma de valores desde el sensor, etc.

2.3.1. Características de un sensor

Rango de medida: dominio en la magnitud de medida en el que puede aplicarse el sensor.

Offset o desviación de cero: valor de la variable de salida cuando la variable de entrada es nula. Si el rango de medida no llega a valores nulos de la variable de entrada, habitualmente se establece otro punto de referencia para definir el offset.

Sensibilidad de un sensor: suponiendo que es de entrada y de salida, sería la variación de la magnitud de entrada.

Rapidez de respuesta: puede ser un tiempo fijo o depender de cuánto varíe la magnitud a medir. Depende de la capacidad del sistema para seguir las variaciones de la magnitud de entrada.

Derivas: son otras magnitudes, aparte de la medida como magnitud de entrada, que influyen en la variable de salida. Por ejemplo, pueden ser condiciones ambientales, como la humedad, la temperatura u otras como el envejecimiento (oxidación, desgaste, etc.) del sensor.
Repetitividad: error esperado al repetir varias veces la misma medida.

Un sensor es un tipo de transductor que transforma la magnitud que se quiere medir o controlar, en otra, que facilita su medida. Pueden ser de indicación directa (ejemplo un termómetro de mercurio) o pueden estar conectados a un indicador (posiblemente a través de un convertidor analógico a digital, un computador y un display) de modo que los valores detectados puedan ser leídos por un humano.
Por lo general, la señal de salida de estos sensores no es apta para su lectura directa y a veces tampoco para su procesado, por lo que se usa un circuito de acondicionamiento, como por ejemplo un puente de Wheatstone, amplificadores y filtros electrónicos que adaptan la señal a los niveles apropiados para el resto de los circuitos.

Resolución y precisión: La resolución de un sensor es el menor cambio en la magnitud de entrada que se aprecia en la magnitud de salida. Sin embargo, la precisión es el máximo error esperado en la medida.
La resolución puede ser de menor valor que la precisión. Por ejemplo, si al medir una distancia la resolución es de 0,01 mm, pero la precisión es de 1 mm, entonces pueden apreciarse variaciones en la distancia medida de 0,01 mm, pero no puede asegurarse que haya un error de medición menor a 1 mm. En la mayoría de los casos este exceso de resolución conlleva a un exceso innecesario en el coste del sistema.

No obstante, en estos sistemas, si el error en la medida sigue una distribución normal o similar, lo cual es frecuente en errores accidentales, es decir, no sistemáticos, la repetitividad podría ser de un valor inferior a la precisión.

Sin embargo, la precisión no puede ser de un valor inferior a la resolución, pues no puede asegurarse que el error en la medida sea menor a la mínima variación en la magnitud de entrada que puede observarse en la magnitud de salida.

2.3.2. Tipos de sensores

En la siguiente tabla se indican algunos tipos y ejemplos de sensores electrónicos.

Magnitud	Transductor	Característica
Posición lineal o angular	Potenciómetro	Analógica
	Encoder	Digital
	Sensor Hall	Digital
Desplazamiento y deformación	Transformador diferencial de variación lineal	Analógica
	Galga extensiométrica	Analógica
	Magnetoestrictivos	A/D
	Magnetorresistivos	Analógica
	LVDT	Analógica
Velocidad lineal y angular	Dinamo tacométrica	Analógica
	Encoder	Digital
	Detector inductivo	Digital
	Servo-inclinómetros	A/D
	RVDT	Analógica
	Giróscopo	
Aceleración	Acelerómetro	Analógico
	Servo-acelerómetros	
Fuerza y par (deformación)	Galga extensiométrica	Analógico
	Triaxiales	A/D
Presión	Membranas	Analógica
	Piezoeléctricos	Analógica
	Manómetros Digitales	Digital
Caudal	Turbina	Analógica
	Magnético	Analógica
Temperatura	Termopar	Analógica

	RTD	Analógica
	Termistor NTC	Analógica
	Termistor PTC	Analógica
	[Bimetal - Termostato]]	I/0
Sensores de presencia	Inductivos	I/0
	Capacitivos	I/0
	Ópticos	I/0 y Analógica
Sensores táctiles	Matriz de contactos	I/0
	Piel artificial	Analógica
Visión artificial	Cámaras de video	Procesa. Digital
	Cámaras CCD o CMOS	Procesamiento digital
Sensor de proximidad	Sensor final de carrera	
	Sensor capacitivo	
	Sensor inductivo	
	Sensor fotoeléctrico	
Sensor acústico (presión sonora)	micrófono	
Sensores de acidez	IsFET	
Sensor de luz	fotodiodo	
	Fotorresistencia	
	Fototransistor	
	Célula fotoeléctrica	
Sensores captura de movimiento	Sensores inerciales	

2.4. Actuadores

Un actuador es un dispositivo capaz de transformar energía hidráulica, neumática o eléctrica en la activación de un proceso con la finalidad de generar un efecto sobre un proceso automatizado. Este recibe la orden de un regulador o controlador y en función a ella genera la orden para activar un elemento final de control.

Existen varios tipos de actuadores como son:

- Electrónicos
- Hidráulicos
- Neumáticos
- Eléctricos

2.4.1 Actuadores electrónicos

Los actuadores son dispositivos electrónicos por medio de los cuales se modifican los estados de los sistemas. Para cada tipo de carga existe un determinado tipo de actuador. Según se trate de un circuito de iluminación, de un motor o de una válvula, habrá que seleccionar el actuador correspondiente para el correcto funcionamiento del sistema.

Los actuadores electrónicos también son muy utilizados en los aparatos mecatrónicos, como por ejemplo, en los robots. Los servomotores CA sin escobillas se utilizarán en el futuro como actuadores de posicionamiento preciso debido a la demanda de funcionamiento sin tantas horas de mantenimiento.

2.5. Técnicas de control de velocidad de motores

A la técnica de controlar motores eléctricos se denomina electrónica de potencia, a la rama de la ingeniería eléctrica que consigue adaptar y transformar la electricidad, con la finalidad de alimentar otros equipos, transportar energía, controlar el funcionamiento de máquinas eléctricas, etc.

Se refiere a la aplicación de dispositivos electrónicos, principalmente semiconductores, al control y transformación de potencia eléctrica.

Esto incluye tanto aplicaciones en sistemas de control como de suministro eléctrico a consumos industriales o incluso la interconexión sistemas eléctricos de potencia.

2.5.1. Dispositivos Semiconductores de Potencia

Para estas aplicaciones se han desarrollado una serie de dispositivos semiconductores de potencia, todos los cuales derivan del diodo o el transistor. Entre estos se encuentran los siguientes:

- Rectificador controlado de silicio (SCR en inglés)
- Triac
- Transistor IGBT
- Tiristor IGCT
- MCT

2.5.2. Convertidores de la Energía Eléctrica

Conversión de potencia es el proceso de convertir una forma de energía en otra, esto puede incluir procesos electromecánicos.

Dichos dispositivos son empleados en equipos que se denominan convertidores estáticos de potencia, clasificados en:

Rectificadores: convierten corriente alterna en corriente continua

Ciclo-conversores: convierten corriente alterna en corriente alterna

En la actualidad esta disciplina está cobrando cada vez más importancia debido principalmente a la elevada eficiencia de los convertidores electrónicos en comparación a los métodos tradicionales, y su mayor versatilidad.

Un paso imprescindible para que se produjera esta revolución fue el desarrollo de dispositivos capaces de manejar las elevadas potencias necesarias en tareas de distribución eléctrica o manejo de potentes motores.

2.5.2.1. Aplicaciones

Las principales aplicaciones de los convertidores electrónicos de potencia son las siguientes:

Fuentes de alimentación: En la actualidad han cobrado gran importancia un subtipo de fuentes de alimentación electrónicas, denominadas fuentes de alimentación conmutadas. Estas fuentes se caracterizan por su elevado rendimiento y reducción de volumen necesario. El ejemplo más claro de aplicación se encuentra en la fuente de alimentación de los ordenadores.

Control de motores eléctricos: La utilización de convertidores electrónicos permite controlar parámetros tales como la posición, velocidad o par suministrado por un motor. Este tipo de control se utiliza en la actualidad en los sistemas de aire acondicionado. Esta técnica, denominada comercialmente como "inverter" sustituye el antiguo control encendido/apagado por una regulación de velocidad que permite ahorrar energía. Asimismo, se ha utilizado ampliamente en tracción ferroviaria, principalmente en vehículos aptos para corriente continua (C.C.) durante las décadas de los años 70 y 80, ya que permite ajustar el consumo de energía a las necesidades

reales del motor de tracción, en contraposición con el consumo que tenían los vehículos controlados por resistencias de arranque y frenado.

Actualmente el sistema chopper sigue siendo válido, pero ya no se emplea en la fabricación de nuevos vehículos, puesto que actualmente se utilizan equipos basados en el motor trifásico, mucho más potente y fiable que el motor de colector.

Las líneas de investigación actuales buscan la integración de dispositivos de potencia y control en un único chip, reduciendo costes y multiplicando sus potenciales aplicaciones. No obstante existen dificultades a salvar como el aislamiento entre zonas trabajando a altas tensiones y circuitería de control, así como la disipación de la potencia perdida.

III. HADWARE

3.1. DESCRIPCION DE UN MODELO DE SILLA DE RUEDAS CONVENCIONAL (CHASIS)

3.1.1 Silla de ruedas eléctrica

La silla de ruedas es una estructura que contribuye a solventar las limitaciones de desplazamiento que tienen algunas personas con algún tipo de discapacidad. Para los usuarios de silla de ruedas, ésta toma la función de extremidades inferiores lo cual permite que estas personas desarrollen una mejor calidad de vida.

Para nuestra implementación, se utilizó un chasis de la marca fortress scientific, modelo 760fs, esta marca es americana. (Fig.3.1.).

Fig.3.1. chasis de silla de ruedas motorizada

3.1.2 Tipos de sillas de ruedas

SILLA ELÉCTRICA ESTÁNDAR INTERIOR – EXTERIOR

Es la más común teniendo los siguientes tipos de tracciones:

1. **TRACCIÓN CENTRAL:** Giran sobre sí mismas por lo que necesitan menos espacio para maniobrar, habiendo modelos casi exclusivamente para interiores que usan este tipo de tracción en combinación con unas ruedas estabilizadoras de menor tamaño.

2. **TRACCIÓN DELANTERA:** Al llevar las ruedas de mayor tamaño en la parte delantera son buenas salvando obstáculos, sin embargo la dirección es algo más compleja.

3. TRACCIÓN TRASERA: Es más cómoda de conducir siendo la elegida por la mayoría de usuarios. Como opción, algunas sillas pueden llevar un accesorio llamado sube bordillos, que mediante un resorte que se apoya en el borde, nos ayuda a subir. Figura 3.2.

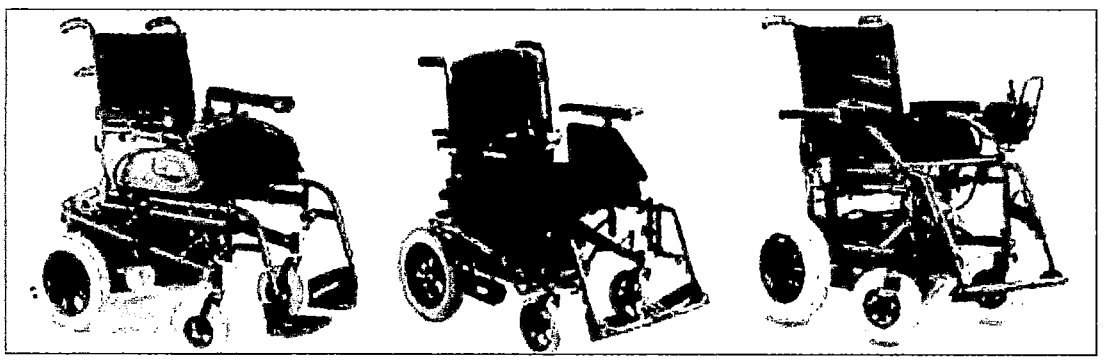
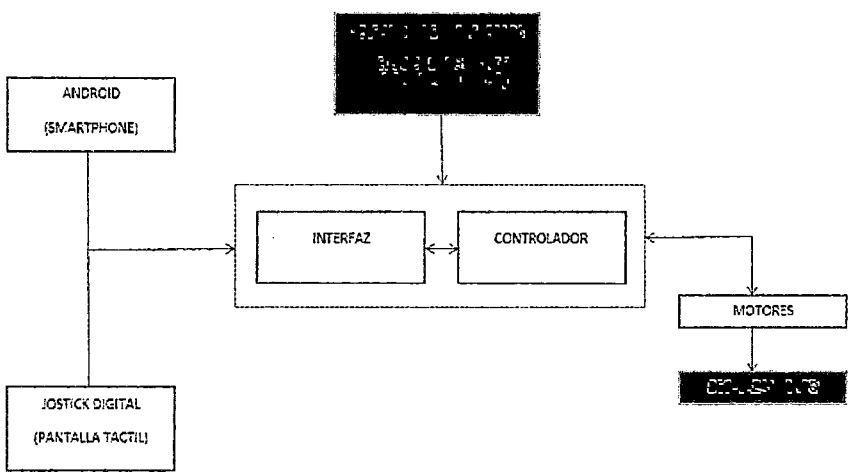


Fig.3.2. chasis de silla de ruedas motorizada de tracción trasera

3.1.3 Diagrama de bloques de modelo de silla motorizada



3.2. Placa de Evaluación STM32-H103

El microcontrolador seleccionado cumple con las siguientes características:

- bajo Consumo de energía
- Bus I2C
- Bus SPI para comunicación
- Comunicación por Protocolo RS232.
- Comunicación por Protocolo USB.
- Manejo de datos tipo float.

En el mercado de los microcontroladores existen infinidad de placas usando microcontroladores de diferentes marcas, pero para este proyecto de tesis se decidió usar la placa STM32-H103 pues no solo ofrece todas las características descritas anteriormente si no que es una plataforma compacta y versátil que usa un microcontrolador ARM Cortex M3 muy potente que también es de bajo costo y alto rendimiento accesible a cualquier aficionado a la electrónica.

3.2.1. Historia

El diseño del ARM comenzó en 1983 como un proyecto de desarrollo en la empresa Acorn Computers. Sophie Wilson y Steve Furber lideraban el equipo, cuya meta era, originalmente, el desarrollo de un procesador avanzado, pero con una arquitectura similar a la del MOS 6502.

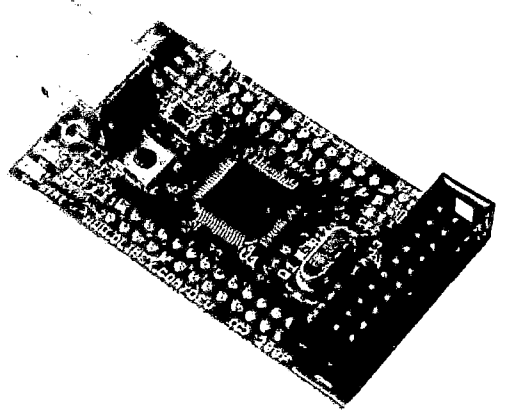


Figura. 3.3. Placa de Evaluación STM32-H103

El equipo terminó el diseño preliminar y los primeros prototipos del procesador en el año 1985, al que llamaron ARM1. La primera versión utilizada comercialmente se bautizó como ARM2 y se lanzó en el año 1986.

La arquitectura del ARM2 posee un bus de datos de 32 bits y ofrece un espacio de direcciones de 26 bits, junto con 16 registros de 32 bits. Uno de estos registros se utiliza como contador de programa, aprovechándose sus 4 bits superiores y los 2 inferiores para contener los flags de estado del procesador.

El ARM2 es probablemente el procesador de 32 bits útil más simple del mundo, ya que posee solo 30.000 transistores. Su simplicidad se debe a que no está basado en microcódigo (sistema que suele ocupar en torno a la cuarta parte de la cantidad total de transistores usados en un procesador) ya que, como era común en aquella época, no incluye caché. Gracias a esto, su consumo en energía es bastante bajo, a la vez que ofrece un mejor rendimiento que un 286. Su sucesor, el ARM3, incluye una pequeña memoria caché de 4 KB, lo que mejora los accesos a memoria repetitivos.

A finales de los años 80, Apple computer comenzó a trabajar con acorn en nuevas versiones del núcleo ARM. En Acorn se dieron cuenta de que el hecho de que el fabricante de un procesador fuese también un fabricante de ordenadores podría echar para atrás a los clientes, por lo que se decidió crear una nueva compañía llamada Advanced RISC Machines, que sería la encargada del diseño y gestión de las nuevas generaciones de procesadores ARM. Ocurría esto en el año 1990.

Este trabajo derivó en el ARM6, presentado en 1991. Apple utilizó el ARM 610 (basado en el ARM6), como procesador básico para su innovador PDA, el Apple Newton. Por su parte, Acorn lo utilizó en 1994 como procesador principal en su RiscPC.

El núcleo mantuvo su simplicidad a pesar de los cambios: en efecto, el ARM2 tiene 30.000 transistores, mientras que el ARM6 solo cuenta con 35.000. La idea era que el usuario final combinara el núcleo del ARM con un número opcional de periféricos integrados y otros elementos, pudiendo crear un procesador completo a la medida de sus necesidades.

La mayor utilización de la tecnología ARM se alcanzó con el procesador ARM7TDMI, con millones de unidades en teléfonos en móviles y sistemas de videojuegos portátiles.
DEC licenció el diseño, lo cual generó algo de confusión debido a que ya se producía el DEC alpha, y creó el strongARM. Con una velocidad de reloj de 233MHZ, este procesador consumía solo 1W de potencia (este consumo de energía se ha reducido en versiones más recientes).esta tecnología posteriormente a manos de Intel, como fruto de un acuerdo jurídico, que la integró en su línea de procesadores Intel i960 e hizo más ardua la competencia.
Freescale (una empresa que derivó de Motorola en el año 2004), IBM, infineon tecnologies, OKI, Texas instruments, nintendo, Philips, VLSI, Atmel, Sharp, Samsung y ST-Microelectronics también licenciaron el diseño básico del ARM. El diseño del ARM se ha convertido en uno de los más usados del mundo, desde discos duros hasta juguetes. Hoy en día, cerca del 75 % de los procesadores de 32 bits poseen este chip en su núcleo.

3.2.2. El Core Cortex-M3

El núcleo Cortex-M3 presenta un proceso de interrupción rápido y de alta determinación, por lo que es muy apropiado para aplicaciones de microcontroladores. Además Cortex-M3 pone en práctica la nueva arquitectura Thumb-2 de conjunto de instrucciones mixtas.

El núcleo central del Cortex-M3 está basado en la arquitectura Harvard caracterizada por buses separados para instrucciones y datos. El core Cortex- M3 puede desarrollar varias operaciones en paralelo. Este núcleo esta segmentado en tres etapas Fetch (búsqueda de instrucción), decodificación de instrucción y Ejecución. El core Cortex-M3 contiene una avanzada ALU la cual realiza operaciones de división y multiplicación de 32 bits por hardware, lógica de control e interfaces con otros componentes del procesador.

Como se mencionó, el Cortex-M3 es un procesador de 32-bit, con un ancho de banda para el flujo de datos de 32 bits, así como banco de registros e interface de memoria. Tiene

13 registros de propósito general, dos punteros de pila, un registro de búsqueda, un contador de programa y registros especiales, incluido un registro de estado.

Dicho procesador cuenta con un sistema de mapa de memoria sencillo, es un mapa de memoria fijo de 4 gigabytes, dividiendo el espacio de memoria con espacios predefinidos y dedicados para código (espacio de código), SRAM (espacio de memoria), memoria/dispositivos externos, periféricos externos y periféricos externos/internos. (Figura 3.4).

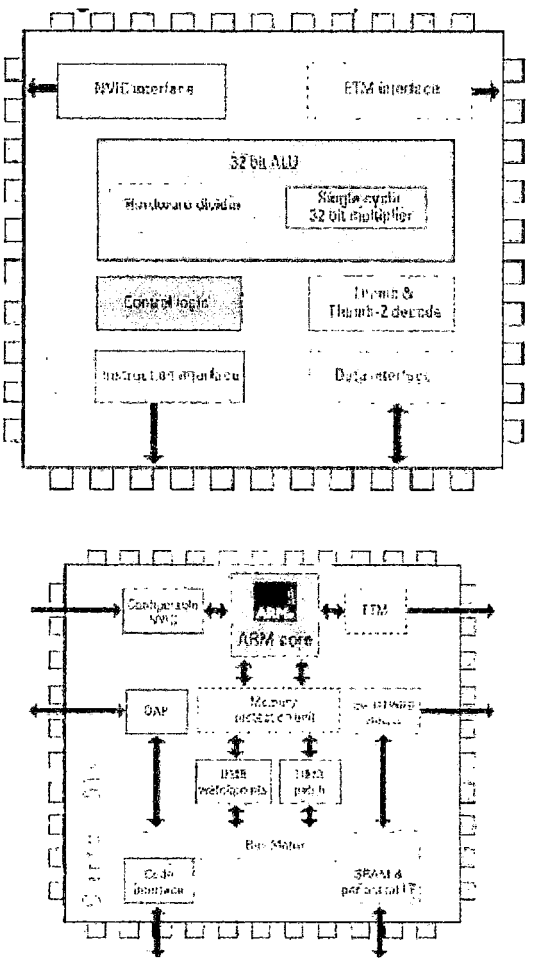


Figura 3.4: Arquitectura Arm

3.2.3 Características Técnicas

La placa de evaluación STM32-H103 ofrece múltiples posibilidades para aplicaciones relacionadas con la electrónica y la automática en general. En la figura 3.5 se puede observar una imagen real de la placa de evaluación empleada, con todos los periféricos que la componen. Las características técnicas más importantes de la placa se citan a continuación:

- MCU: STM32F103RBT6 ARM CORTEX M3 de 32 bits con 128 K Bytes programa Flash, 20K Bytes de RAM, USB, CAN, I2C x2, x2 de ADC de 12 bits, USART x3, x2 SPI, temporizadores x3, hasta la operación 72Mhz.
- Conector JTAG estándar con distribución de pines ARM 2x10 para la programación/depuración con ARM-JTAG.
- Conector USB.
- Botón de usuario.
- Botón RESET.
- LED de estado.
- Fuente de alimentación del LED.
- El regulador de voltaje de 3.3V a bordo con un máximo de 800 mA de corriente.
- Sola fuente de alimentación: toma energía del puerto USB o la extensión de pines del conector.
- 8 MHz oscilador de cristal.
- 32768 Hz cristal y conector de la batería de respaldo de RTC.
- Cabeceras de extensión para todos los puertos uC.
- Distancia entre los conectores extensión: 25,4 mm (1").

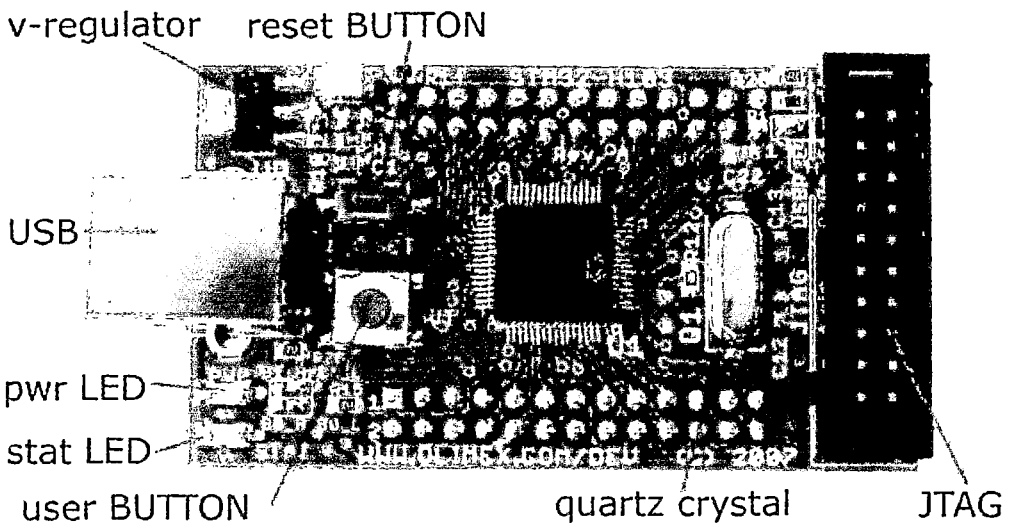
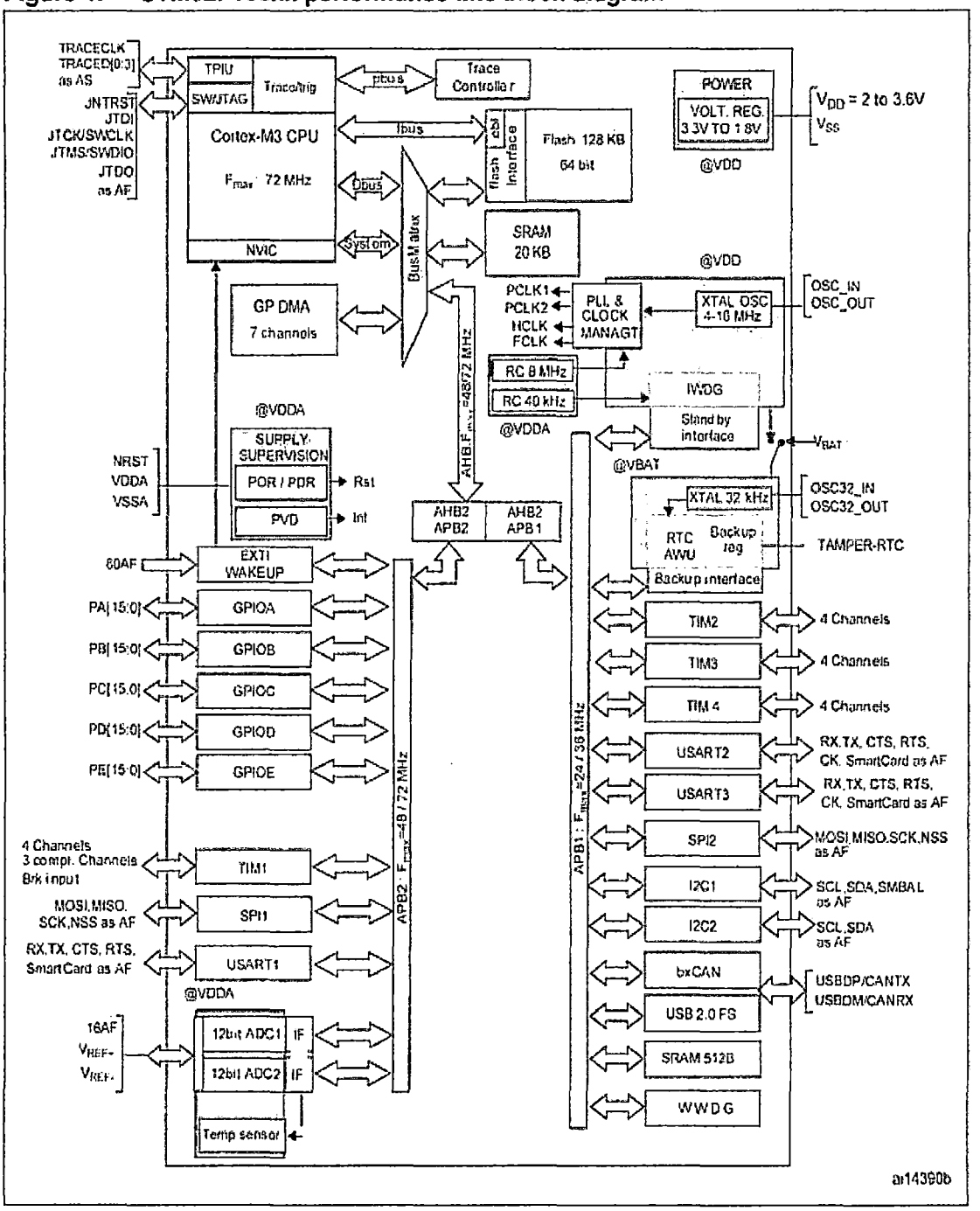


Figura 3.5: placa de evaluación empleada, con todos los periféricos que la componen.

En la figura 3.4. Se puede apreciar una imagen a modo de diagrama de bloques extraída del Manual de Usuario de la placa de evaluación con el microcontrolador y los periféricos que componen.

Figure 1. STM32F103xx performance line block diagram



1. T_A = –40 °C to +105 °C (junction temperature up to 125 °C).
2. AF = alternate function on I/O port pin.

Figura 3.6: Diagrama de Bloques del microcontrolador STM32-H103

3.2.4. Microcontrolador

La placa de evaluación STM32-H103 utiliza un microcontrolador STM32F103RBT6 ARM Cortex M3 de la familia ST Microelectronics Inc. basado en un procesador (Véase figura 3.7) con las siguientes características:

- Reloj de la CPU hasta 72Mhz.
- 128KB FLASH.
- 20KB RAM.
- Canales x7 DMA.
- RTC.
- WDT.
- Timers x3 +1. X2 SPI.
- X2 I2C. USART x3. X1 USB.
- X1 CAN (multiplexado con USB, por lo tanto no se puede utilizar en el mismo tiempo).
- GPIO hasta 51 (multiplexado con periféricos). X2 ADC de 12 bits.
- Tensión de funcionamiento 2.0-3.6V. Temperatura de-40C +85 C.

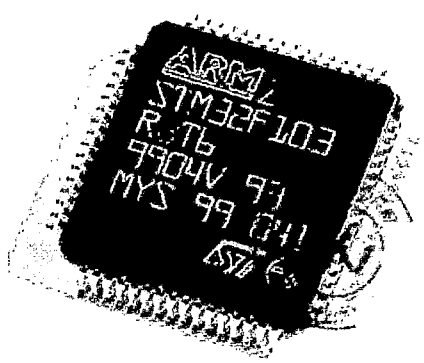


Figura 3.7: Microcontrolador STM32-H103

3.2.5. Mapa de memoria

El mapa se muestra en la figura 3.8 con los principales registros a configurar.

Memory mapping

The memory map is shown in *Figure 7.*

Figure 7. Memory map

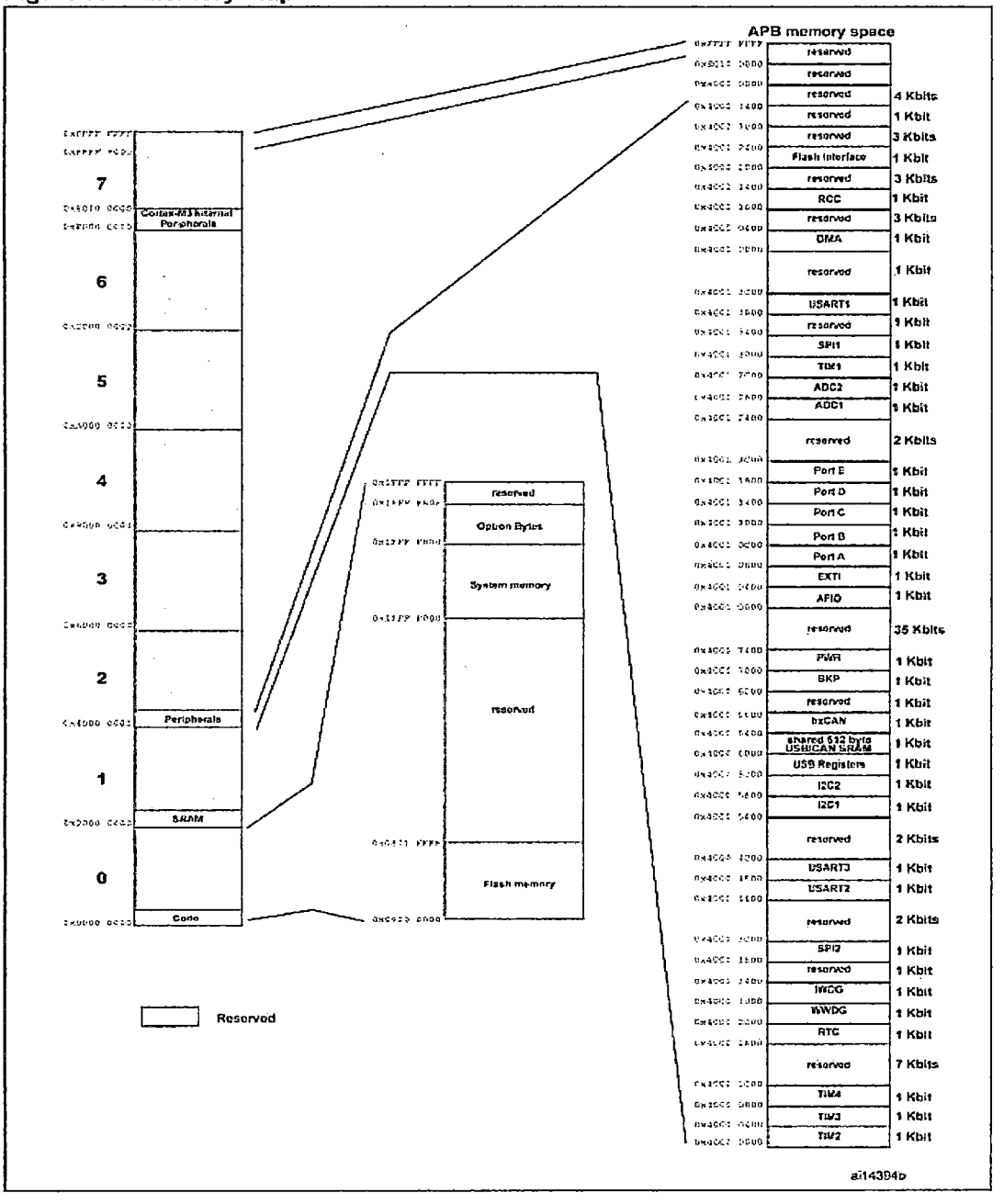


Figura 3.8: Mapa de Memoria

3.2.6. Esquema eléctrico STM32H103

La figura 3.9 muestra la distribución y conexión de los diferentes periféricos que posee la placa de evaluación STM32- H103.

3.2.7. CMSIS

CMSIS, siglas en inglés de cortex microcontroller software interface estándar, al cual le otorgamos gran parte del éxito que está teniendo la arquitectura en los últimos tiempos. Este interfaz estándar de software suele estar integrado en las librerías que proporcionan los fabricantes y permite el acceso a serie de funciones del sistema y periféricos transversales a todos los modelos. Nació en 2008 con el propósito de mejorar la portabilidad y reusabilidad de código creado, evitar problemas de compatibilidad de drivers, facilitar el trabajo a los creadores de herramientas de desarrollo y compilación, y en definitiva, crear una plataforma que permita un desarrollo más estándar, rápido y sencillo.

En cualquier caso, gran parte de los aspectos relacionados con la arquitectura quedan enmascarados cuando este tipo de microcontroladores se programan en lenguaje c. La mayoría de los fabricantes de microcontroladores proporcionan librerías escritas en c que permiten controlar todos los aspectos del dispositivo. Dado que los compiladores modernos generan código muy eficiente, cada vez tiene menos sentido utilizar lenguaje ensamblador (Figura 3.8).

CMSIS define:

- Una manera común de acceso a registros de periféricos del core y de definir vectores de excepción.
- Los nombres de los registros de los periféricos del core y de los vectores de excepción del mismo.

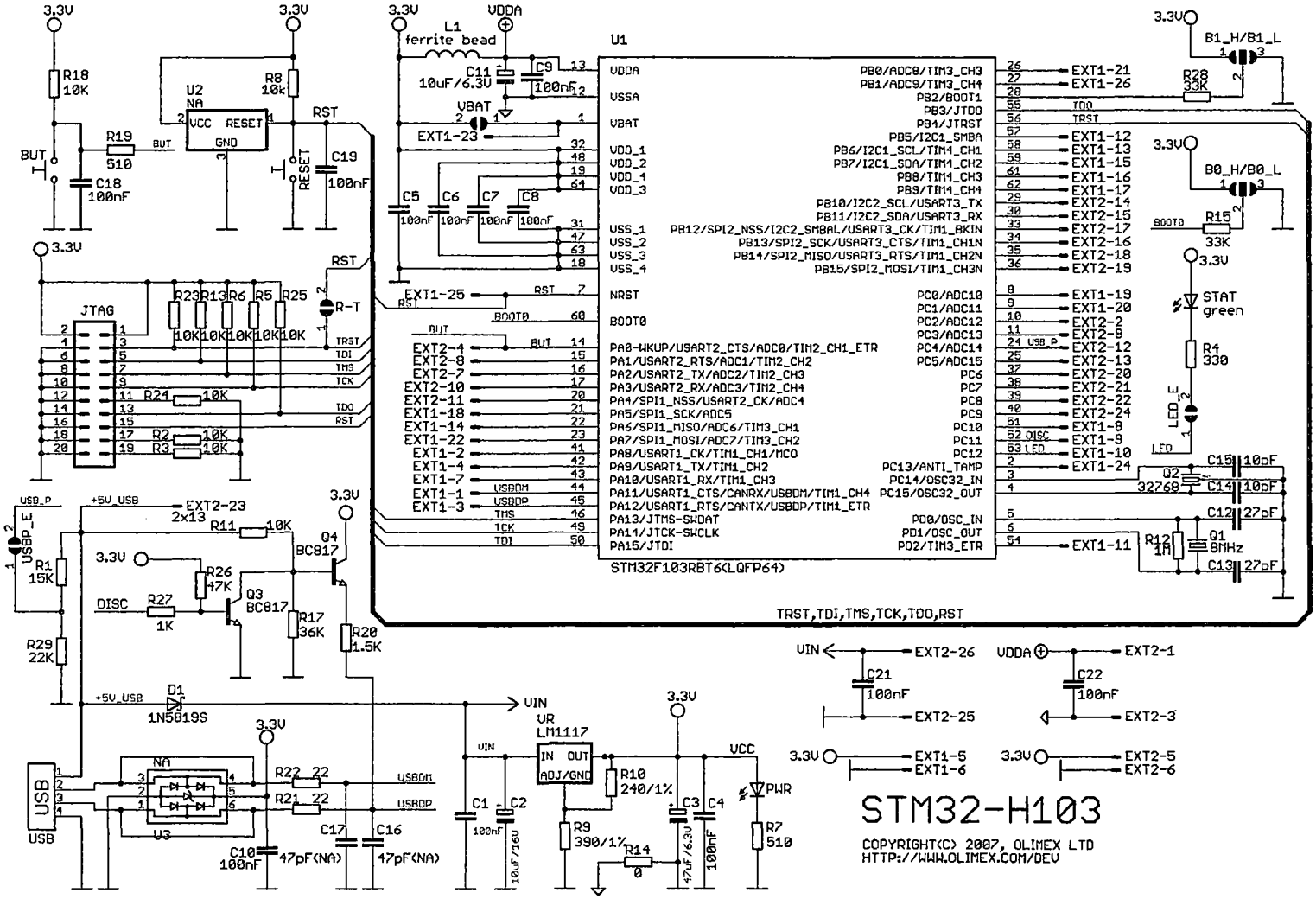


Figura 3.9: Esquema Eléctrico STM32H103

- Una interfaz independiente del dispositivo para RTOS Kernels incluyendo un canal de depuración.

- Mediante el uso de software CMSIS compatible, el usuario puede fácilmente reutilizar el código. CMSIS tiene por objeto permitir la combinación de componentes de software de múltiples proveedores de componentes.

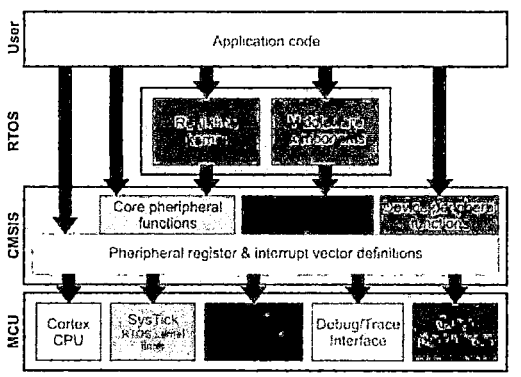


Figura 3.10: interface estándar del software cortex microcontroller (CMSIS).

3.2.8. Librerías

STMicroelectronics facilita una librería que hemos descargado desde su página web llamada STM32L1xx
Standard Peripherals library y STM32F10xxx USB Library. Esta cubre dos niveles de abstracción:

- ✓ Un completo mapeo de registros con todos los bits, campos de bits y registros declarados en lenguaje C. Esto evita conocer a fondo cada periférico ya que son de elevada complejidad y con esto se facilita el aprendizaje en fases iniciales.

- ✓ Colección de funciones y estructuras de datos que cubren todas las funcionalidades de los periféricos.

Cada driver consiste en un conjunto de funciones cubriendo todas las funcionalidades de periféricos. El desarrollo de cada driver está basado en un API común (Application Programming Interface) que estandariza la estructura del driver.

El código de los drivers ha sido escrito en C. Está documentado y es conforme a los requisitos MISRA-C 2004 y CMSIS. La librería es independiente del toolchain empleado, solo los ficheros de inicio startup dependen del toolchain.

La ventaja de utilizar esta librería reside en el tiempo que se necesita para comenzar a desarrollar aplicaciones para STM32L1xx. Gracias a ella, no necesitamos conocer en profundidad todas las facetas y posibilidades del periférico que deseamos programar. Sin embargo, tiene la desventaja que no optimiza el tamaño de la aplicación ni la velocidad de ejecución. En casos en los que tengamos restricciones de tamaño o tiempo, es conveniente utilizar la librería como una referencia para aprender como programar el periférico.

A) **STM32L1XX standard periperals library:** descargamos de la página web de STMicroelectronics la librería en un fichero comprimido Zip. La extracción de este fichero generará la carpeta: STM32F10x stdperip lib. V.3.5.0. el contenido de esta carpeta es el que se muestra en la figura 3.11.

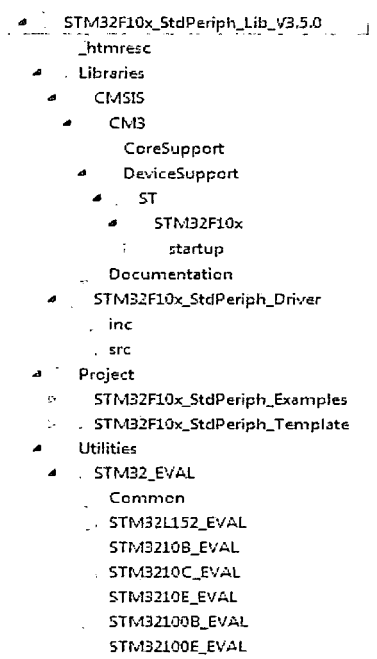


Figura 3.11: STM32L1xx Standard Peripherals library

Htmresc: Esta carpeta contiene todas las páginas de recursos de HTML de la librería.

Libraries: Contiene todos los ficheros CMSIS y STM32L1xx Standard Peripheral's Drivers.

- CMSIS: Contiene todos los ficheros CMSIS de STM32L1xxxx: device peripheral access layer y core peripheral access layer.
- STM32L1xx StdPeriph Driver: Contiene todos los subdirectorios y ficheros que constituyen el núcleo de la librería:
  - ❖ Inc.: Contiene los ficheros de cabecera de los drivers de periféricos, contiene un fichero cabecera por cada driver de periférico.
  - ❖ Src: Contiene los ficheros fuente de los drivers de periféricos. Contiene un fichero fuente por cada driver de periférico.

Project: Esta carpeta contiene los template para diferentes toolchain y ejemplos de cada periférico.

- STM32L1xx StdPeriph Examples: Esta carpeta contiene una subcarpeta por cada periférico, y dentro de ella, ejemplos de código para aprender a usar dicho periférico.
- STM32L1xx StdPeriph Templates: Esta carpeta contiene templates para proyectos para los toolchain: EWARM, MDK-ARM, RIDE, HiTOP, y TrueSTUDIO.
- STM32 EVAL: Implementa una capa de abstracción para que el usuario interactúe con la placa: botones, leds, LCD, puertos COM, etc. Contiene varias subcarpetas en función de la placa de evaluación que estemos utilizando.

B) STM32F10xxx USB Library: descargamos de la página web de ST Microelectronics la librería en un fichero comprimido Zip. La extracción de este fichero generará la carpeta: um0424. El contenido de esta carpeta es el que se muestra en la (figura 3.12).

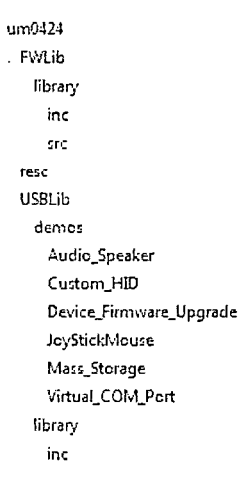


Figura 3.12: STM32L1xx USB library

FW Lib: Contiene todos los subdirectorios y ficheros que constituyen Standard Peripheral's Drivers.

- Inc.: Contiene los ficheros de cabecera de los drivers de periféricos. Contiene un fichero cabecera por cada driver de periférico.
- Src: Contiene los ficheros fuente de los drivers de periféricos. Contiene un fichero fuente por cada driver de periférico.

Resc: Contiene algunas imágenes de la página HTML.

USB Lib: Contiene todos los subdirectorios y ficheros que constituyen a la comunicación USB.

- Demos: Esta carpeta contiene ejemplos de código para aprender a usar dicha librería.
- Library: Contiene todos los subdirectorios y ficheros que constituyen el núcleo de la librería
  - ✓ Inc: Contiene los ficheros de cabecera de los drivers de USB. Contiene un fichero cabecera por cada driver de USB.
  - ✓ Src: Contiene los ficheros fuente de los drivers de USB. Contiene un fichero fuente por cada driver de USB.

3.2.9. ARM-JTAG Wiggler

JTAG, un acrónimo para Joint Test Action Group, es el nombre común utilizado para la norma IEEE 1149.1 titulada Standard Test Access Port and Boundary-Scan Architecture para test access ports utilizada para testear PCBs utilizando escaneo de límites.

JTAG se estandarizó en 1990 como la norma IEEE 1149.1-1990. En 1994 se agregó un suplemento que contiene una descripción del boundary scan description language (BSDL). Desde entonces, esta norma fue adoptada por las compañías electrónicas de todo el mundo. Actualmente, Boundary-scan y JTAG son sinónimos.

Diseñado originalmente para circuitos impresos, actualmente es utilizado para la prueba de sub módulos de circuitos integrados, y es muy útil también como mecanismo para depuración de aplicaciones empotradas, puesto que provee una puerta trasera hacia dentro del sistema. Cuando se utiliza como herramienta de depuración, un emulador en circuito que usa JTAG como mecanismo de transporte permite al programador acceder al módulo de depuración que se encuentra integrado dentro de la CPU. El módulo de depuración permite al programador corregir sus errores de código y lógica de sus sistemas.

Una interfaz JTAG es una interfaz especial de cuatro o cinco pines agregadas a un chip, diseñada de tal manera que varios chips en una tarjeta puedan tener sus líneas JTAG conectadas en daisy chain, de manera tal que una sonda de testeo JTAG necesita conectarse a un solo "puerto JTAG" para acceder a todos los chips en un circuito impreso (Figura 3.13). Los pines del conector son:

- TDI (Entrada de Datos de Testeo).
- TDO (Salida de Datos de Testeo).
- TCK (Reloj de Testeo).
- TMS (Selector de Modo de Testeo).
- TRST (Reset de Testeo) es opcional.

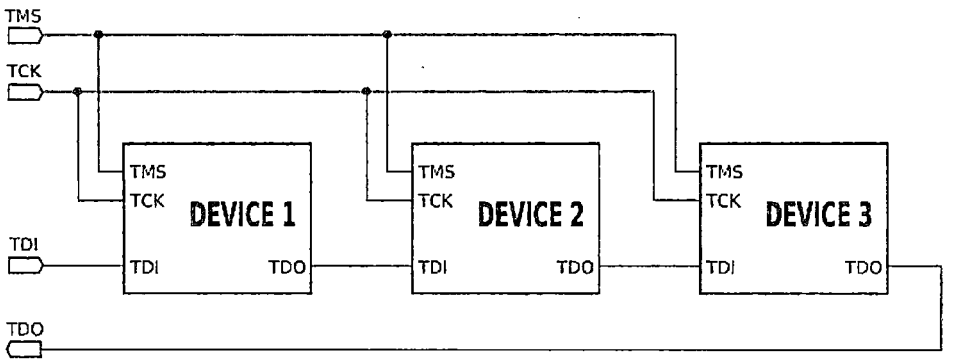


Figura 3.13: Pines jtag (pruebas de puertos de acceso)

Ya que posee una sola línea de datos, el protocolo es necesariamente serial, como el serial pheriperal interface. La entrada de la señal de reloj es por el pin TCK. La configuración del dispositivo la realizamos manipulando una máquina de estados de un bit empleando el pin TMS. Un bit de datos es cargado en TDI y otro sacado en TDO por cada pulso de reloj de la señal TCK. Se pueden cargar diferentes modos de instrucción como leer el ID del chip, muestrear el valor de pines de entrada/salida, manejar pines de salida, manipular funciones del chip o funciones de bypass que unen el pin TDI con TDO para lógicamente unir cadenas de varios chips (chips en cascada).la frecuencia de trabajo de la señal de reloj del pin TCK varía en función de cada chip, pero típicamente está en el rango de 10-100MHZ (10-100ns/bit).

El pin TRST es una señal opción bajo-activa para reseteo o reinicio de la prueba lógica (por lo general asíncrona, pero que a veces esta sincronizada con el reloj, dependiendo del chip).si no se dispone de dicho pin, la prueba lógica puede reiniciarse mediante una instrucción reset.

La figura 3.14 muestra el esquema eléctrico para el ARM-JTAG wiggler.

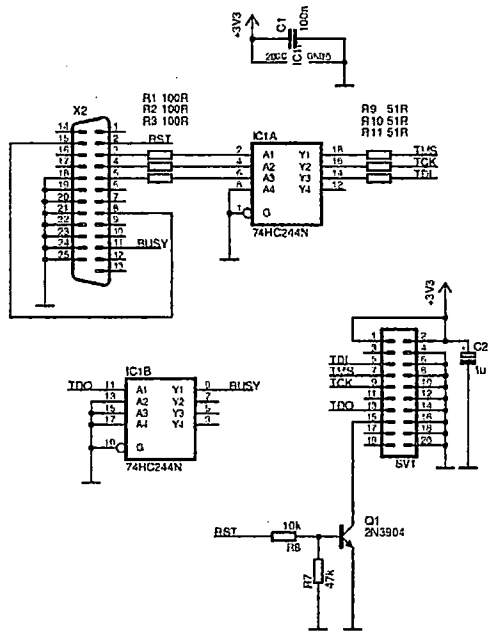


Figura 3.14: Esquema Eléctrico ARM-JTAG Wiggler

El ARM-JTAG Wiggler es una interfaz que se usa en el diseño, depuración y programación de microprocesadores y microcontroladores basados en sistemas embebidos. Un extremo de la interfaz se conecta al puerto paralelo de un host PC y el otro extremo se conecta a un puerto JTAG del sistema de destino tal como se muestra en la figura 3.15.

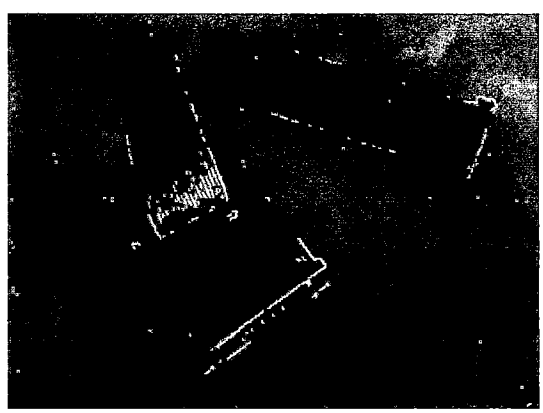


Figura. 3.15: ARM-JTAG Wiggler

IV. SOFTWARE

4.1. Introducción

Una vez presentado el hardware necesario para el desarrollo del prototipo de una silla eléctrica motorizada a construir el siguiente paso será dar a conocer los diferentes tipos de programas que se utilizaron en este proyecto.

Dentro de los programas a mencionar se encuentran Keil uVision4 que es utilizado para editar y compilar el programa principal y distintas librerías en lenguaje C para el microprocesador STM32F103 ARM, H-JTAG encargado de la programación del microcontrolador ARM, y la aplicación para el Smartphone, App inventor.

4.2. Keil uVision4

El entorno de desarrollo uVision de Keil es una herramienta de carácter profesional de enorme calidad que junto con algunas otras más se ha convertido en un estándar para el desarrollo de aplicaciones basadas en las arquitecturas ARM-Cortex (Figura 4.1).

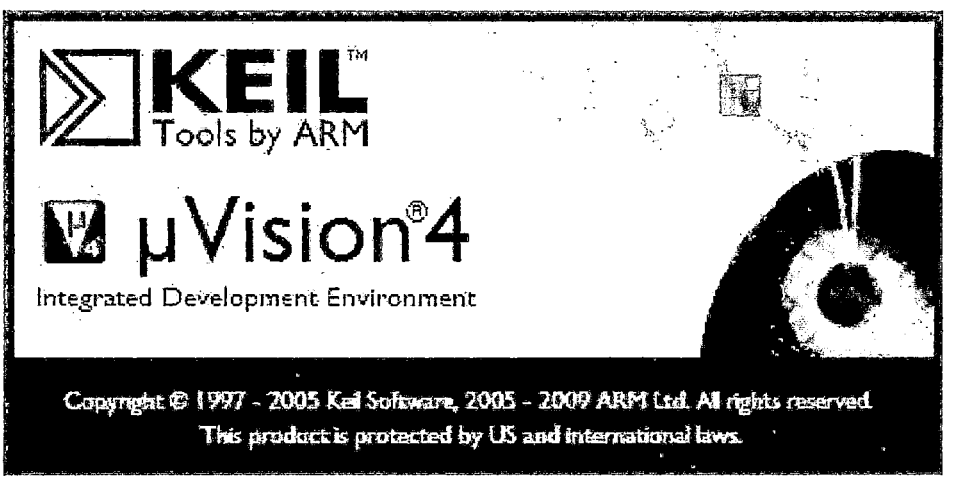


Figura 4.1: Keil uVision4

Las herramientas desarrolladas por Keil apoyan los microcontroladores más populares y son distribuidas en varios paquetes y configuraciones, dependiendo de la arquitectura.

MDK-ARM: Kit de desarrollo de microcontroladores, para varios ARM7, ARM9 y dispositivos basados en Cortex-Mx.

- PK166: Kit de desarrollo profesional de keil, para dispositivos C166, XE166 y XC2000.
- DK251: Herramientas de desarrollo Keil 251, para dispositivos 251.
- PK51: Herramientas de desarrollo Keil 8051, para dispositivos Classic & Extended 8051.

Esta parte proporciona una introducción sobre software MDK-ARM (versión 4.1.0 y anteriores).

4.2.1. MDK-ARM

El MDK-ARM es una plataforma de desarrollo de software basado en ventanas que combina un robusto y moderno editor con un administrador de proyectos y lo hace una herramienta fácil. Integra todas las herramientas necesarias para desarrollar aplicaciones incluyendo el compilador C/C++, ensamblador de macros, enlazador/buscador, y un generador de archivos AXF. En la figura 4.2 muestra una vista general del software.

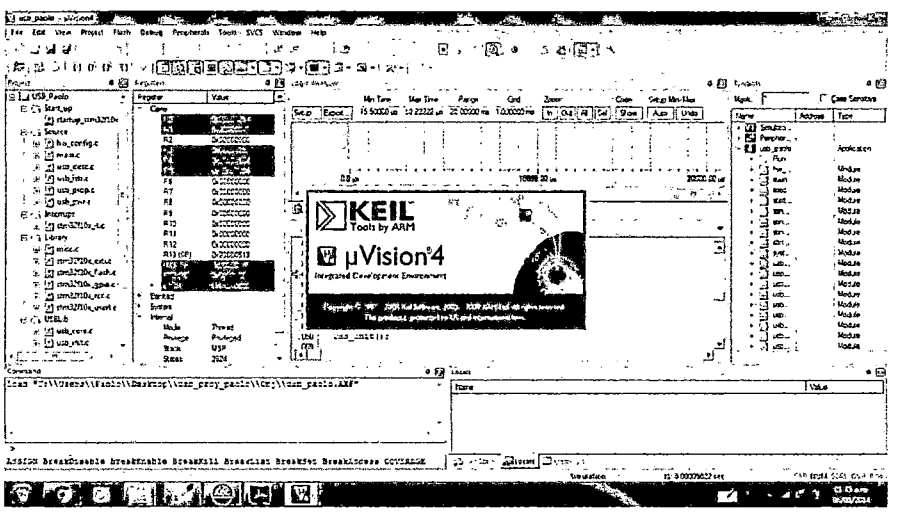


Figura 4.2: Ventana principal MDK-ARM

MDK-ARM nos ayuda acelerando el desarrollo de procesos de aplicaciones integradas proporcionando lo siguiente:

- Editor de código fuente con todas las funciones.
- Base de datos de los dispositivos para configurar la herramienta del desarrollo.
- Administrador de proyectos para la creación y mantenimientos de tus proyectos.
- Utilidad make integrado para ensamblar, compilar y vincular sus aplicaciones embebidas.
- Diálogo para todas las opciones del entorno de desarrollo.
- Verdadero depurador a nivel de fuente y nivel ensamblador integrado con CPU de alta velocidad y un simulador de periféricos.
- Utilidad de programación flash para la descarga del programa de aplicación en flash ROM.
- Enlace a manuales, ayuda en línea, hoja de datos de dispositivos y guías de usuarios.

4.2.2. Ventanas de diseño

Nosotros podemos configurar el entorno de trabajo de uVision según nos sea conveniente. Sin embargo definiremos tres áreas más importantes. Estas definiciones nos ayudarán para el entendimiento futuro de comentarios, ilustraciones e instrucciones (Figura 4.3).

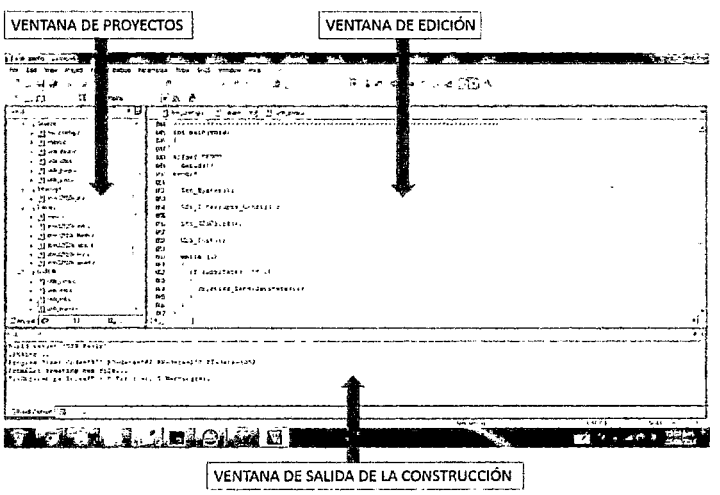


Figura 4.3: Áreas de trabajo

El área de la ventana de proyectos es la parte de la pantalla en la cual, por defecto, el proyecto, las funciones, los libros y registros son mostrados.

Con el área de la ventana de edición, nosotros somos capaz de cambiar el código fuente, ver la performance y análisis de información, y verificar el código de desmontaje.

El área de la ventana de salida nos proporciona información relacionada con la depuración, memoria, símbolos, llamada de pila, variables locales, comandos, buscador de información, y encuentra los resultados en archivos.

Si, por alguna razón, nosotros no pudiéramos ver una ventana en particular y hemos tratado de mostrar/ocultar esto varias veces, tendríamos que hacer uso del diseño por defecto de uVisión a través del menú Windows - Reset Current Layout.

4.2.3. Redistribución de la zona de trabajo

A la zona de trabajo podemos cambiarla de aspecto y de organización. Para ello es necesario que actuemos sobre alguna o algunas de las restantes ventanas, lo que nos origina un cambio conjunto de las distribuciones de todas las ventanas y zona de trabajo. Los procesos que comentamos se pueden aplicar sucesivamente con diferentes ventanas o pestañas una tras otra. La manera de proceder es la siguiente:

- Primero, damos click en la barra superior de la ventana que se deseamos reorganizar (o en la pestaña oportuna, en el caso de una organización en pestañas y que solo se desee reubicar esa pestaña en concreto y no todas las de la ventana). En nuestro caso, la pestaña Templates.
- Manteniendo pulsado el botón izquierdo del ratón, arrastramos. En ese preciso momento nos aparecerán unos símbolos u operadores de organización, tal y como se aprecia en la figura 4.4.

Figura 4.4: Reorganización de ventanas

4.2.4. Modos de uVision

Uvisión opera en dos modos: modo de construcción y modo de depuración. Ajustes de pantalla, ajustes de barra de herramientas y opciones de proyecto son almacenados en el contexto del modo. La barra de herramientas de archivo está activa en todos los modos,

mientras que la barra de herramientas de depuración y construcción está mostrados solamente en sus respectivos modos. Botones, iconos y menús son activados si son relevantes para un modo específico.

- **Modo de Construcción:** Es el modo de trabajo estándar, el cual es el que se ha utilizado para nuestro proyecto.En este modo escribimos nuestra aplicación, configuramos el proyecto, establecemos preferencias, seleccionamos el hardware de destino y el dispositivo; compilamos, enlazamos, ensamblamos el programa, corregimos errores, y establecemos ajustes generales válidos para toda la aplicación.

- **Modo de Depuración:** En este modo nosotros también podemos cambiar algunas opciones generales y editar archivos de código fuente, pero esos cambios solo se harán efectivos después de que haya cambiado de nuevo a modo de construcción, y reconstruir la aplicación. Los cambios de ajustes de depuración son inmediatamente efectivos.

4.2.5. Barra de menú de herramientas

La figura 4.5 muestra en la parte superior la barra de menú, mediante la cual nosotros podemos acceder a todas las opciones de uVision, y debajo de ella se encuentra la barra de herramientas, con los iconos de las funciones más usuales. Estos iconos suponen un atajo alternativo para realizar tales tareas.

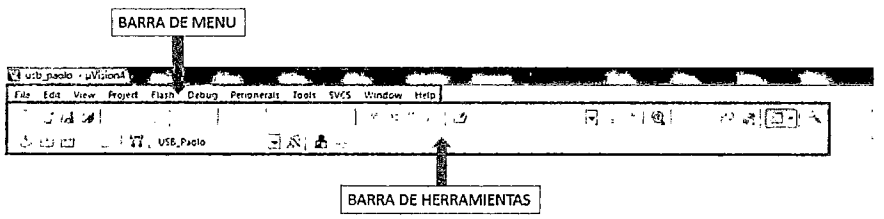


Figura 4.5: Barra de menú y de herramienta

Es importante advertir que si ya se hubiese trabajado con algún proyecto y se hubiese modificado la apariencia y forma de las ventanas de la interfaz, entonces al entrar en uVision4 se mostrará la interfaz exactamente igual a como quedó al cerrarse la última vez.

4.2.6. Ventana de proyecto

Una vez que ya hemos empezado a configurar, y quisiéramos ver nuestro avance, en las diferentes librerías, la ventana de proyecto (figura 4.6), nos muestra información acerca del proyecto actual. Las fichas en la parte inferior de esta área proporcionan acceso a:

Proyect: Estructura y gestiona el proyecto. Agrupa los archivos para mejorar la visión general del proyecto.

Funtions: muestra las funciones del proyecto. Encuentra y navega rápidamente entre funciones del código fuente.

Registers: registros del microcontrolador. Solamente está disponible durante la depuración.

Templates: plantillas de bloques de texto de uso frecuente. Doble click en la definición para insertar el texto predeterminado en la posición del cursor.

Books: libros específicos para el IDE u visión del proyecto y a veces del microcontrolador usado. Configura y añade sus propios libros a cualquier sección.

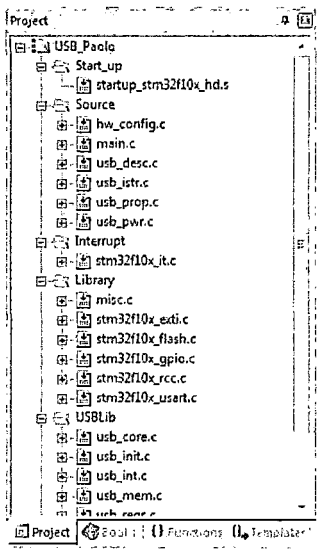


Figura 4.6: Ventana de proyecto

4.2.7. Ventana de Edición

La ventana de edición es usada para:

- Escribir, editar, y depurar archivos fuente.
- Establece puntos de ruptura (breakpoint) y marcas de libro (bookmarks).
- Establece opciones de proyecto e inicializa el sistema de destino mediante potentes asistentes de configuración.
- Ver el código de desmontaje y traza durante la depuración.

Típicamente, esta área contiene el editor de texto con los archivos de código fuente, la ventana de desmontaje, el analizador de rendimiento, y el analizador lógico (figura 4.7).

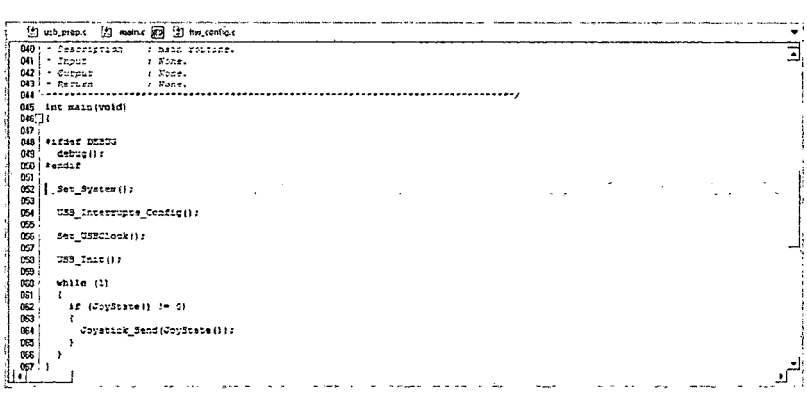


Figura 4.7: Ventana de edición

4.2.8. Editor de configuración

En esta parte, procedemos a configurar las opciones del editor, colores y fuentes, palabras clave definidas por el usuario, y plantillas a través del diálogo de configuración (figura 4.8).

Podemos invocar este diálogo vía el Menú Edit - Configuration.

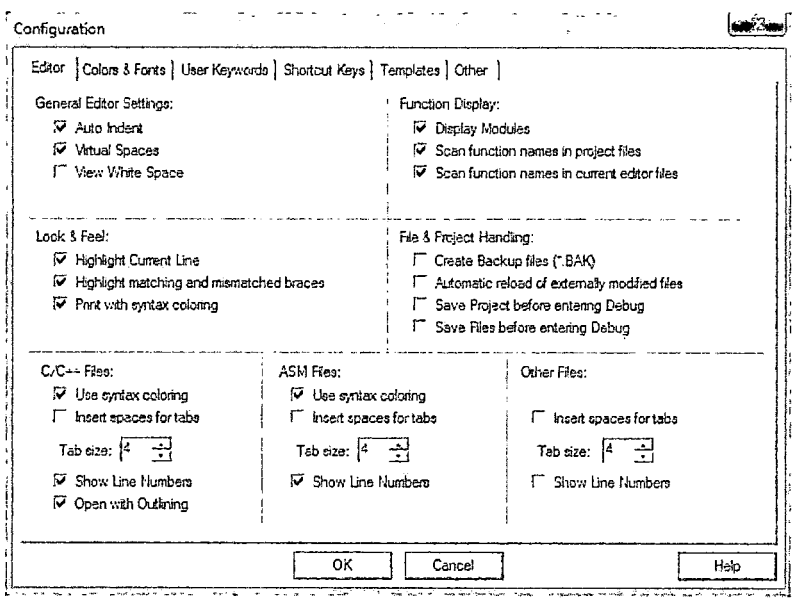


Figura 4.8: Dialogo de configuración

4.2.9. Ventanas de Salida

Por defecto, las ventanas de salida se muestran en la parte inferior de la pantalla de uVisión e incluyen:

- La Ventana de Salida de Construcción incluye errores y avisos del compilador, ensamblador, y enlazador.
- La Ventana de Comandos permite ingresar comandos y evaluar.
- La ventana de Búsqueda de Archivos nos permite hacer doble clic en un resultado para localizar el código fuente que desencadenó el mensaje.
- La Ventana Serial y Usart muestra I/O de información de los periféricos.

- La Ventana de Llamada de Pila permite seguir el árbol de llamadas de programa.
- La Ventana Local muestra información acerca de variables locales de la función actual.
- La Ventana de Reloj nos suministra una conveniente manera de personalizar un conjunto de variables que les gustaría investigar. Objetos, estructuras, uniones, y arreglos pueden ser monitoreados en detalle.
- La Ventana de Símbolos es una opción útil para localizar definiciones de objetos. La Ventana de Memoria permite revisar valores en el área de memoria. Define las direcciones preferidas para ver los datos.
- La ventana de buscador de Fuente ofrece un modo rápido para encontrar ocurrencias y definiciones de objetos. Use sus criterios de búsqueda para reducir la salida.

4.2.10. Ayuda en Línea

UVisión incluye muchas páginas de manuales en línea y ayuda sensible al contexto. La principal ayuda del sistema es disponible desde el Menú Help (figura 4.9).

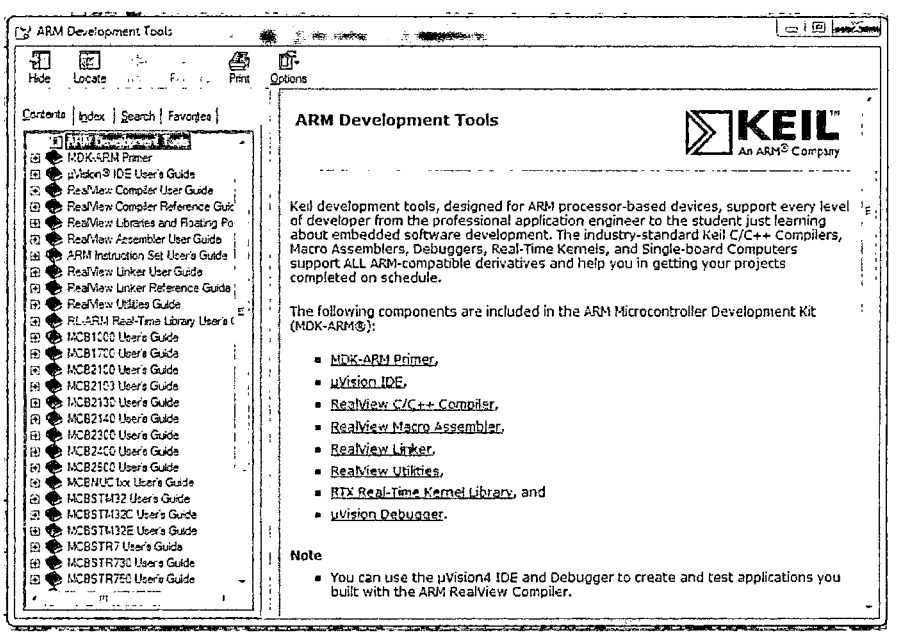


Figura 4.9: Ayuda en línea

4.3. H-JTAG

La siguiente sección describe la programación en la memoria flash en controladores ARM7TDMI con software libre. Se mostrará como la combinación uVisión/H-JTAG/Wiggler, en nuestro proyecto, ha sido usado como una solución de programación. El software usado es H-JTAG V1.0 (Build 20100120) de Twentyone y uVisión V4.10 de Keil.

4.3.1. Configuración

En primer lugar conectamos el Wiggler ARM-JTAG con el puerto paralelo de la PC y la tarjeta de destino. Se verifica que la interfaz JTAG del microcontrolador esté habilitada y aplicamos la alimentación a la placa de evaluación. Luego se Lanza el software H-JTAG como se muestra en la Figura 4.10 para conectar con el microcontrolador ARM.

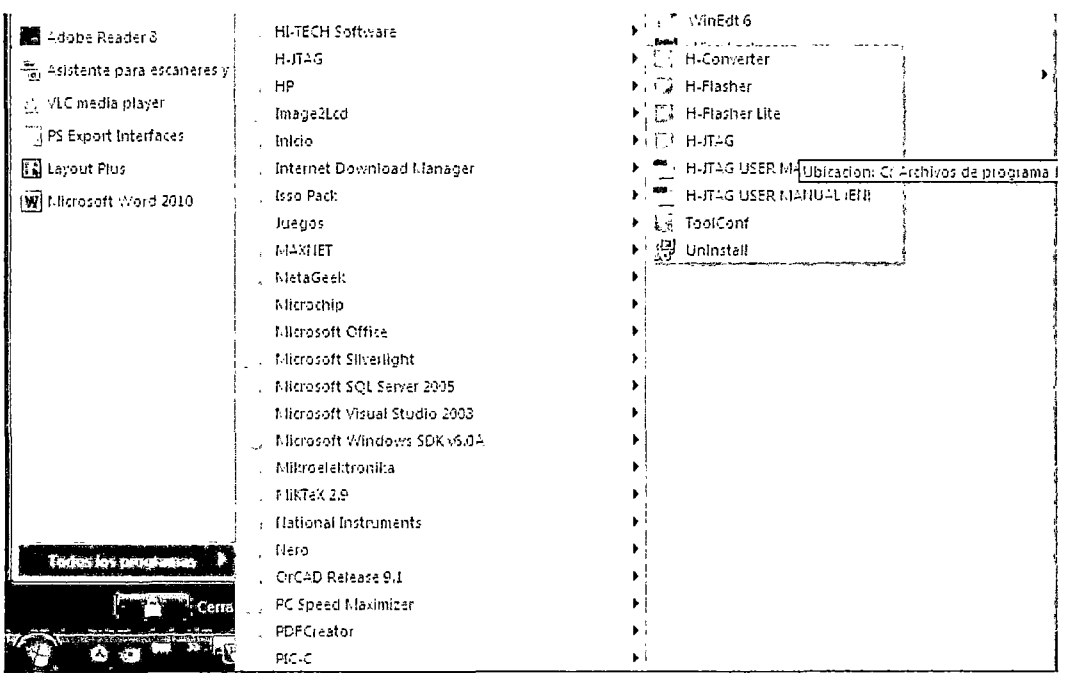


Figura 4.10: se lanza el software H-JTAG

H-JTAG Server automáticamente mostrara el IDCODE detectado.

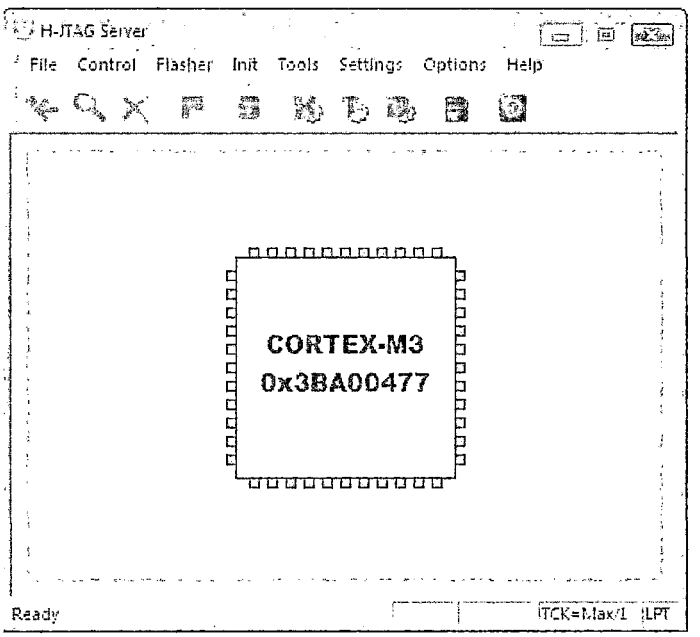


Figura 4.11: H-JTAG Server

Normalmente H-JTAG Server detecta automáticamente la interfaz y establece una conexión. Solo en el caso de que haya un error, nos vamos a LPT JTAG Setting del menú Settings para que aparezca el menú de configuración. Asegurándonos que la opción Wiggler es seleccionada y nTRST establecida con Pin2 D0. Intentamos diferentes valores para la velocidad de TCK para probar la estabilidad. Finalmente se minimiza H-JTAG Server. Sin cerrar la ventana.

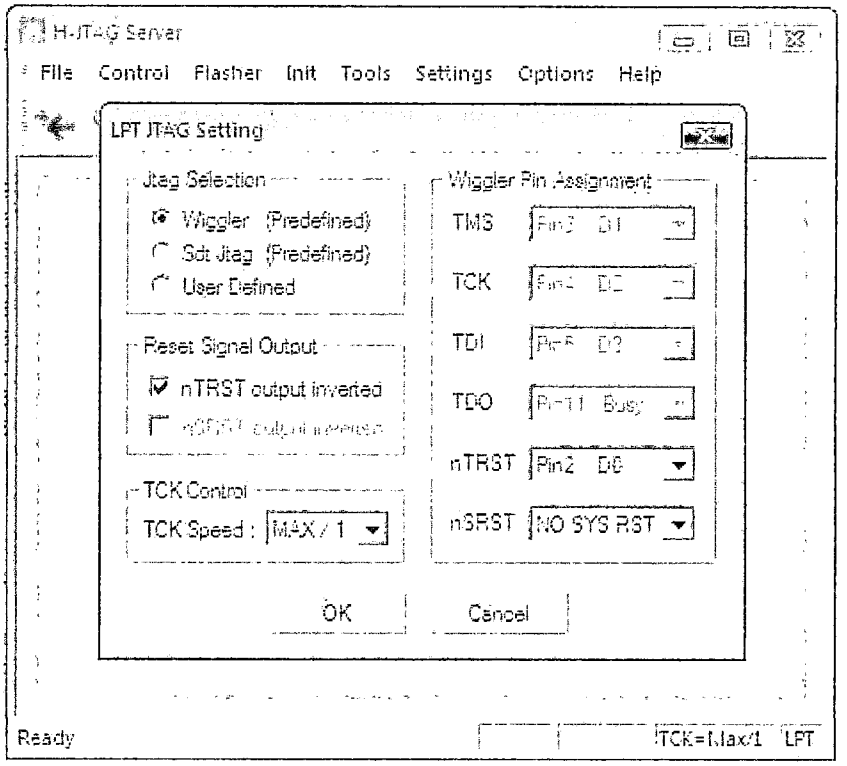


Figura 4.12: Opciones LPT JTAG

4.3.2. Depurando el Programa

Para poder depurar nuestro programa con uVision y H-JTAG usando Wiggler se debe de agregar las características del software dentro de uVision lanzando la herramienta ToolConf del mismo H-JTAG como se muestra en la figura 4.13.

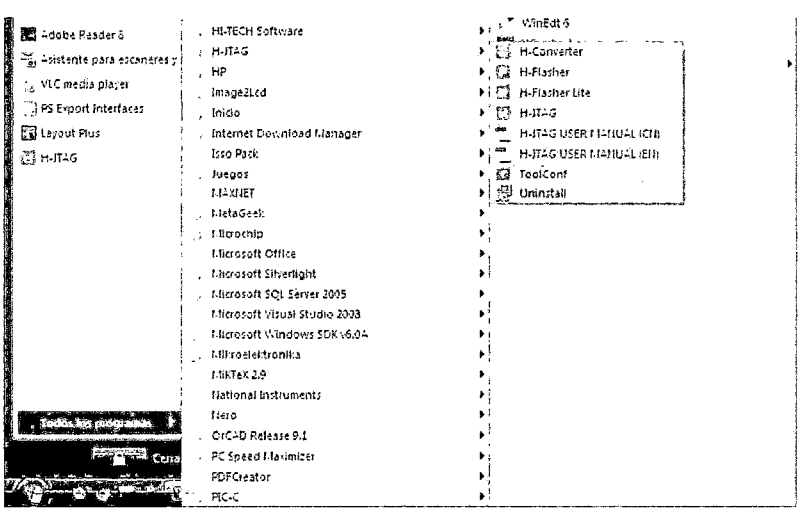


Figura 4.13: Configuración H-JTAG con uVision

ToolConf automáticamente nos muestra la siguiente ventana como se muestra en la figura 4.14 donde se tiene que seleccionar la dirección del archivo TOOL.INI de uVision.

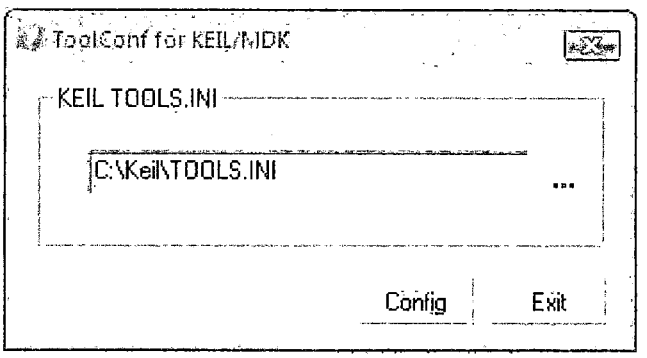


Figura 4.14: Ventana ToolConf

Una vez configurado se mostrará un mensaje como se muestra en la figura 4.15.

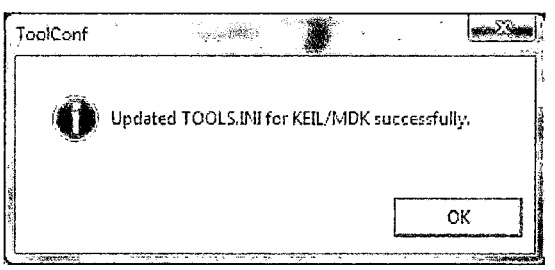


Figura 4.15: Mensaje de configuración de uVision4

Luego abrimos la ventana de Options en uVision y se da clic en la ficha Debug. Verificamos H-JTAG CORTEX-M3 esté seleccionado y que las opciones Load Application at Star- tup () y Run to main () estén habilitadas como se muestra en la figura 4.16.

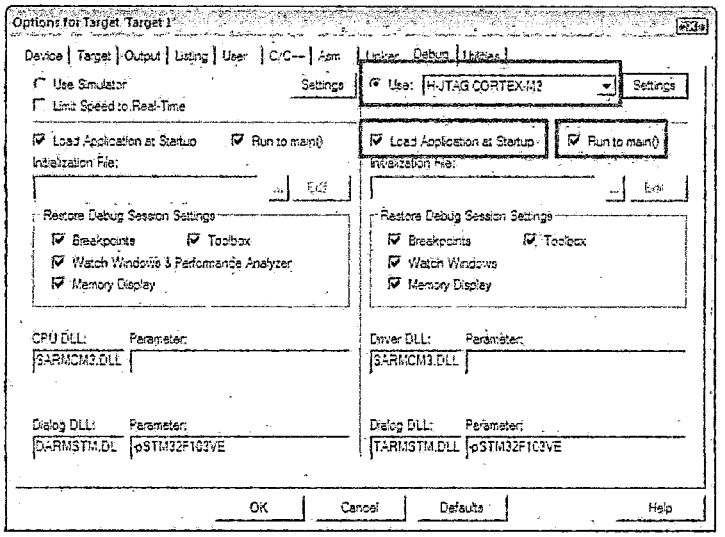


Figura 4.16: Opción Debug

Se da click en la ficha Utilities para abrir las opciones de programación en flash. Necesitamos descargar el programa antes de la depuración.se verifica la opción Use Esternal Tool for Flash Programing donde usaremos H-Flasher para descargar. Buscamos H-Flash.exe y marcamos la opción Run Independent.

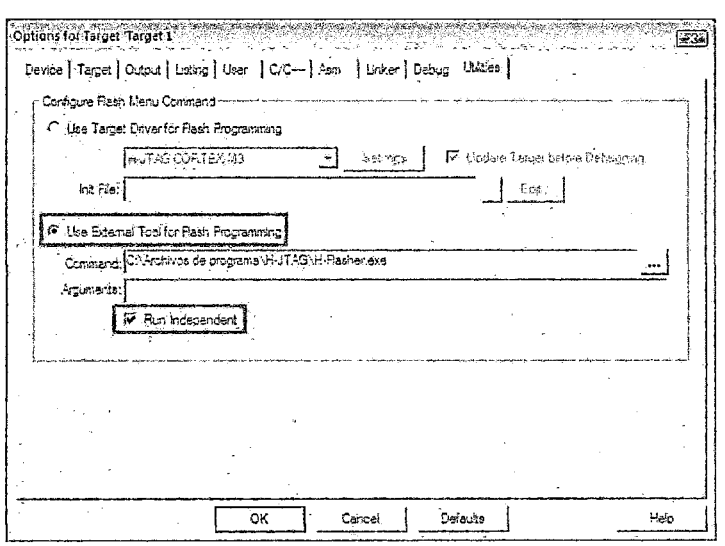


Figura 4.17: Opción Utilities

Clic en OK para salir de la ventana de configuración y regresar al menú principal. Luego vamos al menú Flash y clic en Download. Se debe recordar que antes de depurar el programa se tiene que descargar el programa.

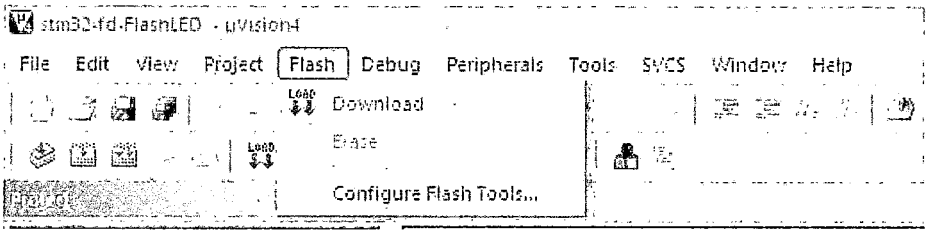


Figura 4.18: Menú flash

Esta acción debería abrir automáticamente la aplicación H-Flasher. Luego se resalta la opción Flash Selection de la izquierda, y elegimos el chip de destino, STM32F y luego STM32F103XE.

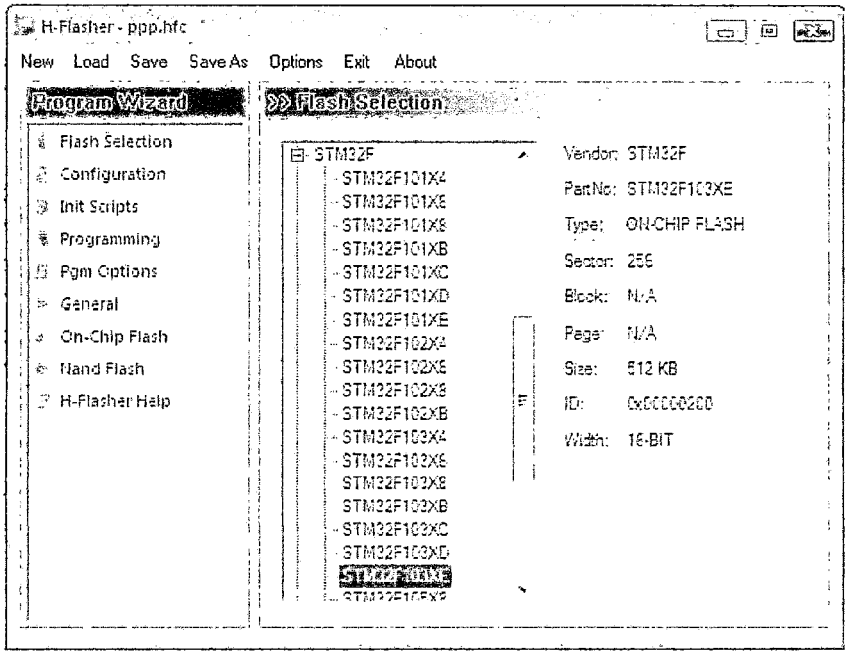


Figura 4.19: H-Flasher

Resaltamos la opción de Programación a la izquierda, y se da clic en el botón Check en el lado derecho cerca de la parte superior el cual detectará al microcontrolador. En la derecha de la caja de texto Src File, dar clic en el botón (...) para buscar el código .HEX de destino.

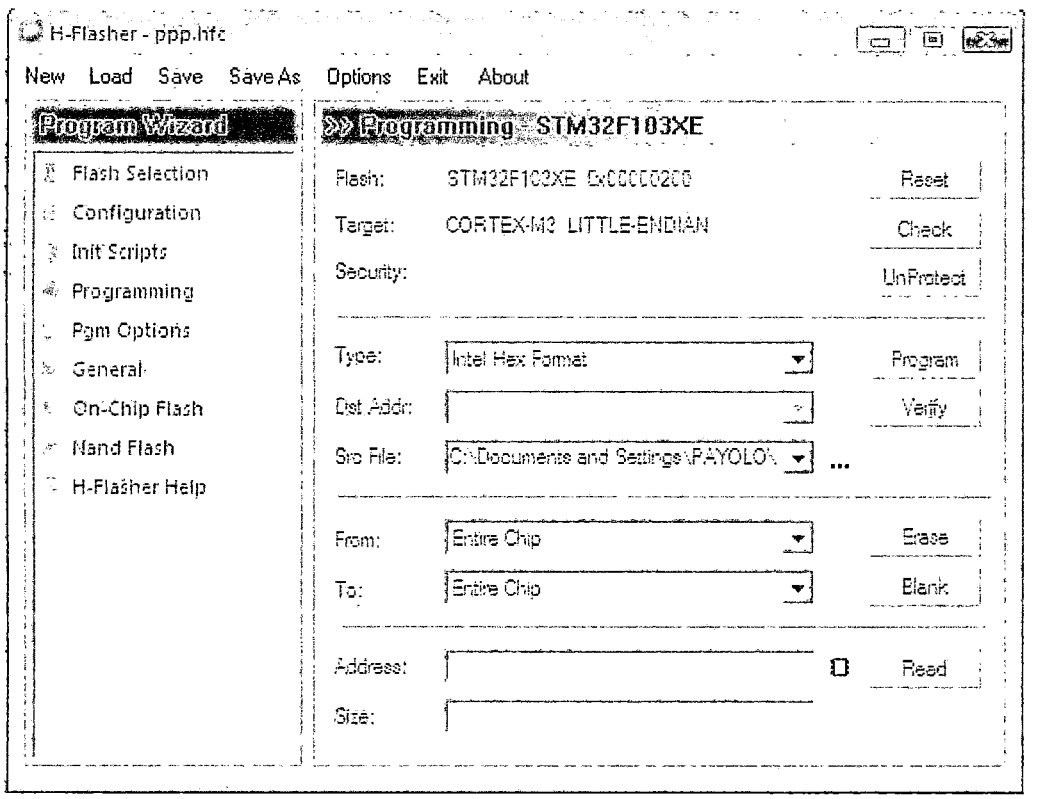


Figura 4.20: Opción de programación

Una vez que todo ha sido configurado, clic en el botón Program para descargar el código hex al microcontrolador. También podemos realizar otras tareas como borrado, verificación de banco, y limitar la región de programación.

Después de finalizar, minimizamos la aplicación de H-Flasher. Se regresa a uVision4, vamos al menú Debug y seleccionamos Start/Stop Debug Session o Ctrl+F5 en forma abreviada para iniciar la depuración. Podremos ver la ventana de depuración con los registros R0- R15, CPSR, y SPSR los cuales son el núcleo de los microcontroladores ARM en el panel izquierdo. Un cursor se para en la primera sentencia en este caso porque la opción Run to main () que fue marcado en la ficha de depuración. Se presiona la tecla F10 para pasar por cada sentencia.

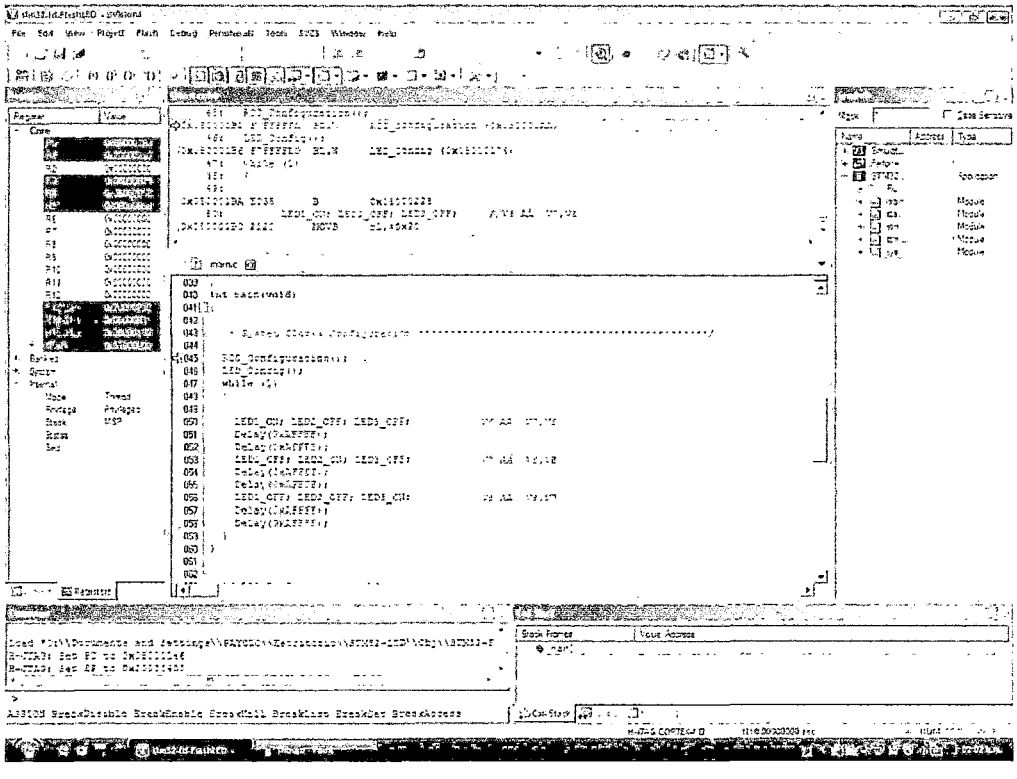


Figura 4.21: Ventana de Depuración

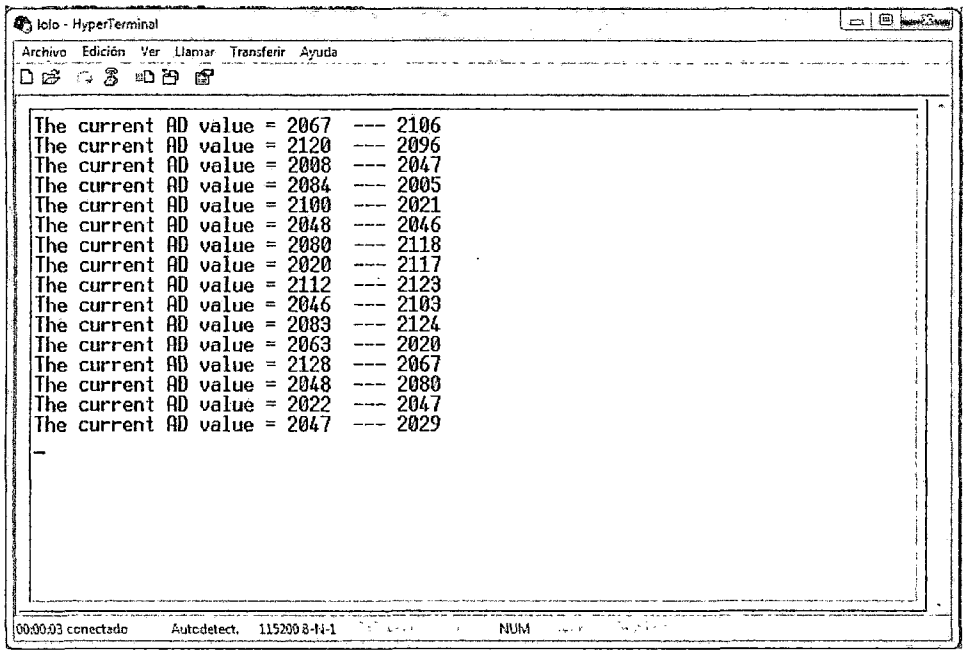


Figura 4.22: Ventana de Hyperterminal

Cuando cerramos la ventana del hyperterminal nos mostrará primero una ventana de confirmación de desconexión y luego otra por si queremos guardar la sesión del hyperterminal con la configuración que hayamos determinado al iniciarla.

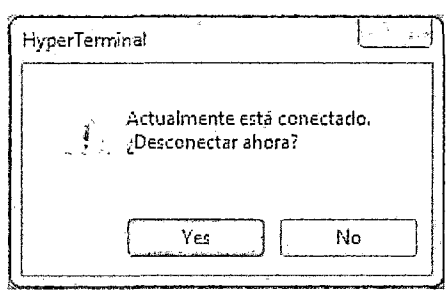


Figura 4.23: Ventana de confirmación de desconexión

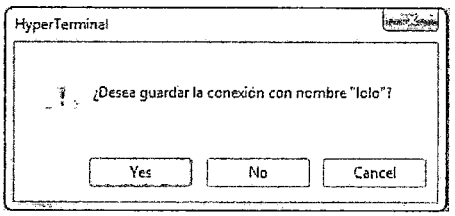


Figura 4.24: Ventana para guardar la sesión de comunicación

4.4. APP INVENTOR

Google App Inventor es una plataforma de Google Labs para crear aplicaciones de software para el sistema operativo Android. De forma visual y a partir de un conjunto de herramientas básicas, el usuario puede ir enlazando una serie de bloques para crear la aplicación. El sistema es gratuito y se puede descargar fácilmente de la web. Las aplicaciones fruto de App Inventor están limitadas por su simplicidad, aunque permiten cubrir un gran número de necesidades básicas en un dispositivo móvil.

Con Google App Inventor, se espera un incremento importante en el número de aplicaciones para Android debido a dos grandes factores: la simplicidad de uso, que facilitará la aparición de un gran número de nuevas aplicaciones; y Google Play, el centro de distribución de aplicaciones para Android donde cualquier usuario puede distribuir sus creaciones libremente.

Diseñando en el App inventor

Una vez que se ha descargado (http://www.appinventor.mit.edu/) e instalado nuestra aplicación procedemos a realizar nuestro diseño. Nos aparecerá la siguiente pantalla (Figura 4.25):

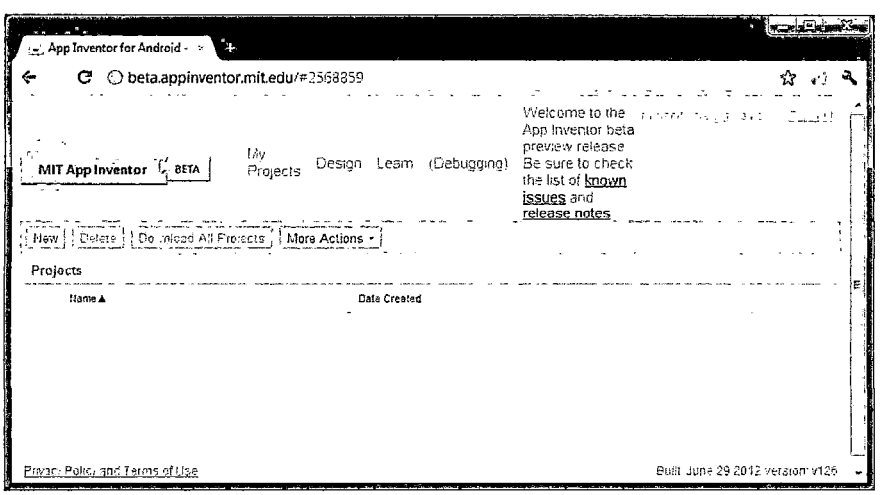


Figura 4.25: primera pantalla de diseño en App inventor

Luego Presionamos el botón "New" para generar un nuevo proyecto, al cual le hemos denominado "silla de ruedas", para empezar a diseñar en sí (Figura 4.26).

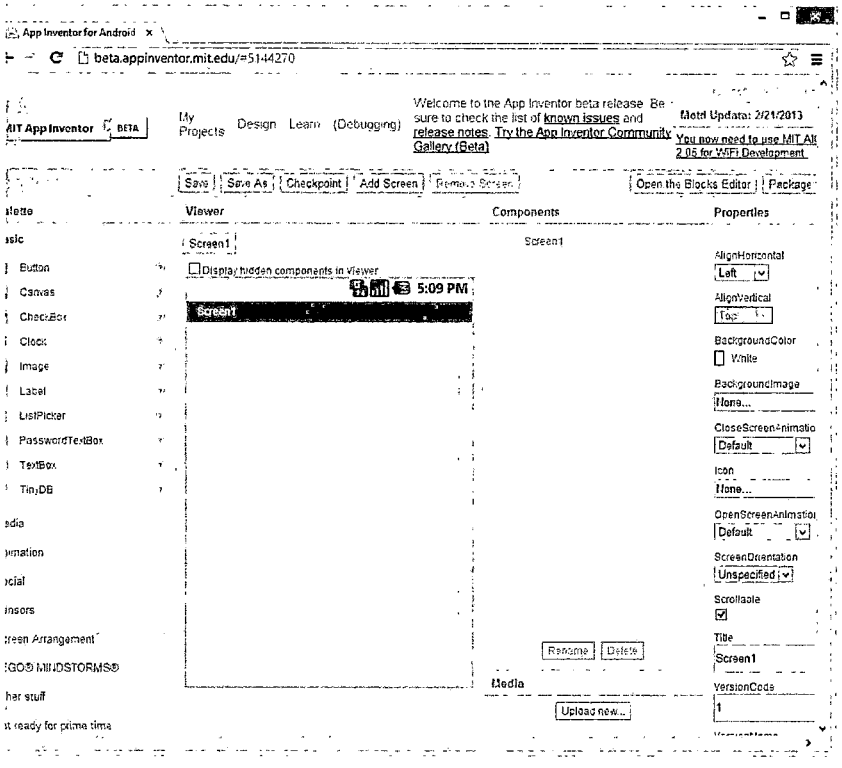


Figura 4.26: visor de App inventor

La App inventor nos brinda múltiples herramientas, tanto de diseño, lógica, operaciones matemáticas e interfaces. En nuestro caso lo utilizaremos netamente como una interface, para eso nos ubicamos en la izquierda (paleta de componentes), seleccionamos la opción "Button", y la arrastramos a la pantalla "Screen1" (visor).

Haremos eso 10 veces para obtener 10 botones o teclas, y las nombraremos de la siguiente manera (Figura 4.27):

boton1.....arriba

boton2.....izquierda

boton3.....derecha

boton4.....abajo

boton5.....v1

boton6.....v2

boton7.....v3

boton8.....v4

boton9.....v5

boton10.....timbre

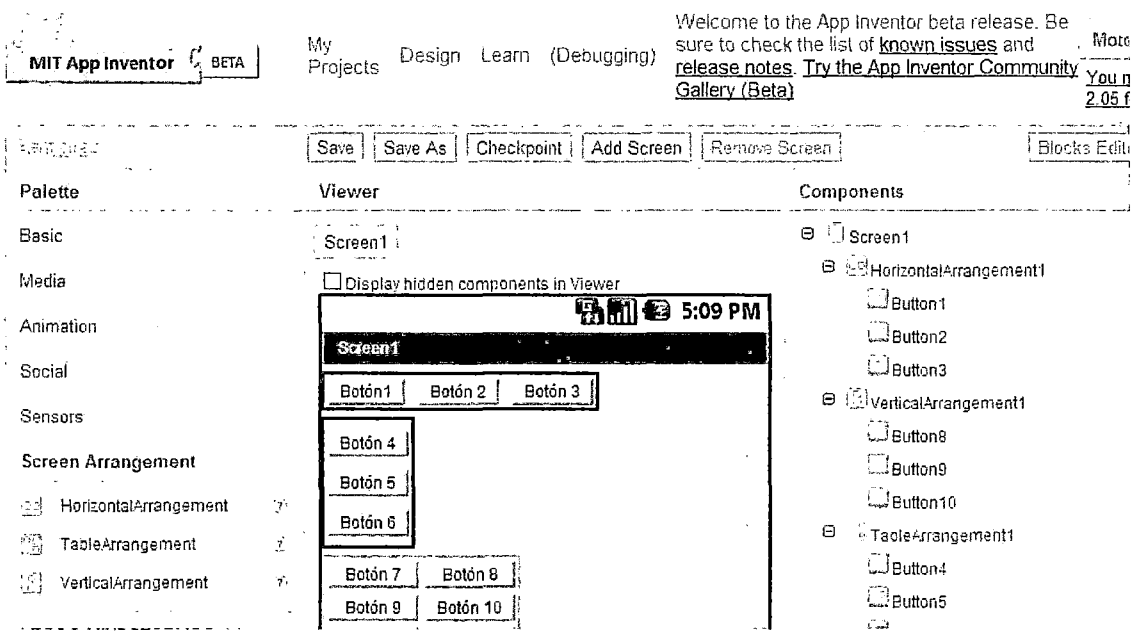


Figura 4.27: teclas de nuestra aplicación

Luego ordenamos los botones o teclas, regulamos sus tamaños, de tal forma, que se asemejen al control de la silla, ya habiéndoles cambiado de nombres, procedemos a guardar nuestro programa, para cargarlo en nuestro móvil (Figura 4.28).

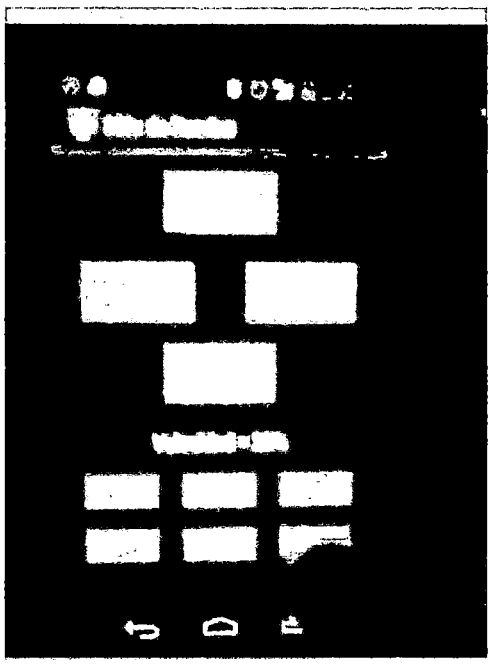


Figura 4.28: control en el móvil

La comunicación es directa hacia el panel del control de la silla (control digital), la configuración del reconocimiento de cada tecla con su similar en el control digital, está incluido en la programación del microcontrolador stm32f103, que es la siguiente:

BOTON4 ABAJO

```
if (cadena2[0] == 'a')
                    {
                        bt_flag3 = 0;
                    }
                    if (cadena2[0] == 'b')
                    {
                        bt_flag3 = 1;
                    }
```

BOTON1 ARRIBA

```
if (cadena2[0] == 'c')
                    {
                        bt_flag1 = 0;
```

```
}
if (cadena2[0] == 'd')
{
        bt_flag1 = 1;
}
```

BOTON3 DERECHA

```
if (cadena2[0] == 'e')
{
        bt_flag4 = 0;
}
if (cadena2[0] == 'f')
{
        bt_flag4 = 1;
}
```

BOTON2 IZQUIERDA

```
cadena2[0] == 'g')
{
        bt_flag2 = 0;
}
if (cadena2[0] == 'h')
{
        bt_flag2 = 1;
}
```

BOTON10 TIMBRE

```
if (cadena2[0] == 'i')
{
        GPIO_ResetBits(GPIOD, GPIO_Pin_6);
}
```

```
if (cadena2[0] == 'j')
{
        GPIO_SetBits(GPIOD, GPIO_Pin_6);
}
```

BOTON VELOCIDAD

V1:

```
if (cadena2[0] == 'k')
{
        porcentaje = 20;
        pwm_const = 6;
}
```

V2:

```
if (cadena2[0] == 'l')
{
        porcentaje = 40;
        pwm_const = 12;
}
```

V3:

```
if (cadena2[0] == 'm')
{
        porcentaje = 60;
        pwm_const = 18;
}
```

V4:

```
if (cadena2[0] == 'n')
{
        porcentaje = 80;
        pwm_const = 24;
}
```

V5:

```
if (cadena2[0] == 'o')
{
```

```
                porcentaje = 100;

                pwm_const = 30;

            }

        }

}
```

Lo único que tenemos que tener en cuenta, en esta parte, es que exista una óptima comunicación (bluetooth).

La App inventor es muy sencilla y fácil de usar, pero no cualquiera podría maniobrar la silla eléctrica por bluetooth, debido a que la silla te exige una contraseña de acceso, la cual solo la sabría el propietario.

Programación

Nuestra programación se divide en los siguientes bloques:

Hw_config.c, app.c, logos de la pantalla táctil.

Hw_config.c

En esta sección vamos a declarar las diferentes librerías y variables (Figura 4.29).

En esta parte configuramos las Gpios que vienen a ser las Entrada/Salida de Propósito General

Los pines GPIO no tienen ningún propósito especial definido, y no se utilizan de forma predeterminada. La idea es que a veces, para el diseño de un sistema completo que utiliza el chip podría ser útil contar con un puñado de líneas digitales de control adicionales, y tenerlas a disposición ahorra el tiempo de tener que organizar circuitos adicionales para proporcionarlos.

Los usart, que vienen a ser nuestros transmisores/receptores universales tanto asíncronos/síncronos.

El DMA que es nuestro acceso directo a memoria.

El Bus SPI (del inglés Serial Peripheral Interface) que es un estándar de comunicaciones, usado principalmente para la transferencia de información entre circuitos integrados en equipos electrónicos.

1. Declaración de librerías y variables

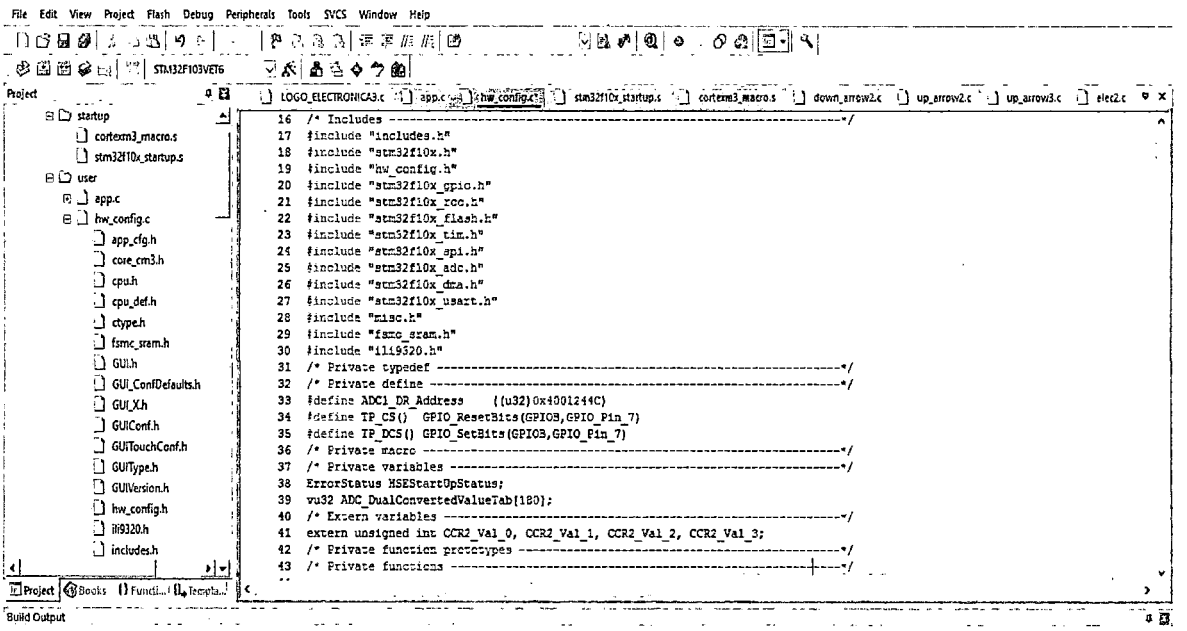


Figura 4.29: Declaración de librerías y variables

```
#include "includes.h"

#include "stm32f10x.h"

#include "hw_config.h"

#include "stm32f10x_gpio.h"

#include "stm32f10x_rcc.h"

#include "stm32f10x_flash.h"

#include "stm32f10x_tim.h"

#include "stm32f10x_spi.h"

#include "stm32f10x_adc.h"

#include "stm32f10x_dma.h"

#include "stm32f10x_usart.h"

#include "misc.h"

#include "fsmc_sram.h"

#include "ili9320.h"
```

2. Desarrollo de la función: **RCC_Configuration**
   En esta sección Configuramos los diferentes relojes del sistema (Figura 4.30).

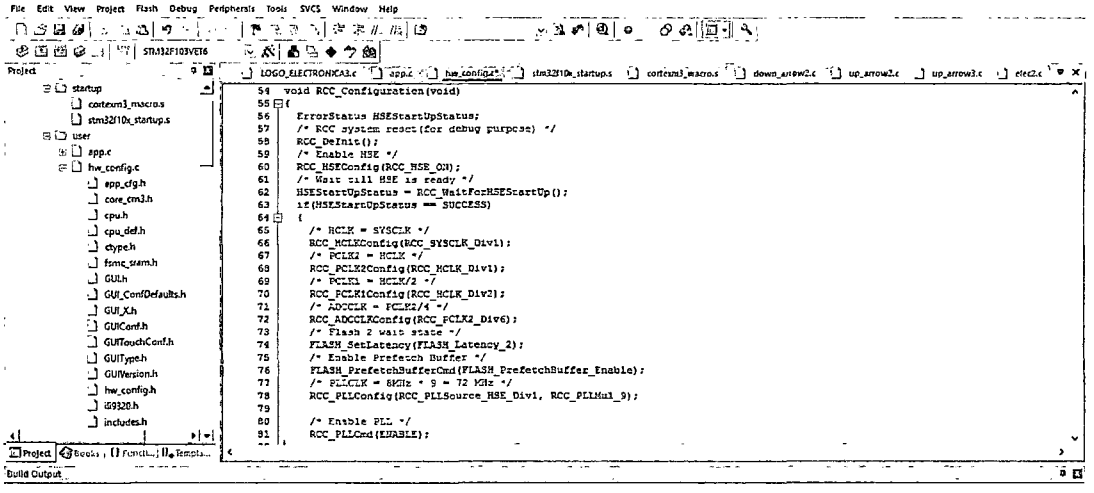


Figura 4.30: RCC_Configuration

```
void RCC_Configuration(void)
{
        ErrorStatus HSEStartUpStatus;
 /* RCC system reset(for debug purpose) */
 RCC_DeInit();

 /* Enable HSE */
 RCC_HSEConfig(RCC_HSE_ON);

 /* Wait till HSE is ready */
 HSEStartUpStatus = RCC_WaitForHSEStartUp();

 if(HSEStartUpStatus == SUCCESS)
 {
  /* HCLK = SYSCLK */
  RCC_HCLKConfig(RCC_SYSCLK_Div1);

  /* PCLK2 = HCLK */

  RCC_PCLK2Config(RCC_HCLK_Div1);

  /* PCLK1 = HCLK/2 */
  RCC_PCLK1Config(RCC_HCLK_Div2);

                /* ADCCLK = PCLK2/4 */
  RCC_ADCCLKConfig(RCC_PCLK2_Div6);

  /* Flash 2 wait state */
```

```
FLASH_SetLatency(FLASH_Latency_2);
/* Enable Prefetch Buffer */
FLASH_PrefetchBufferCmd(FLASH_PrefetchBuffer_Enable);

/* PLLCLK = 8MHz * 9 = 72 MHz */
RCC_PLLConfig(RCC_PLLSource_HSE_Div1, RCC_PLLMul_9);

/* Enable PLL */
RCC_PLLCmd(ENABLE);

/* Wait till PLL is ready */
while(RCC_GetFlagStatus(RCC_FLAG_PLLRDY) == RESET)
{
}

/* Select PLL as system clock source */
RCC_SYSCLKConfig(RCC_SYSCLKSource_PLLCLK);

/* Wait till PLL is used as system clock source */
while(RCC_GetSYSCLKSource() != 0x08)
{
}
}
```


3. Desarrollo de la función: **GPIO_Configuration**
   En esta sección Configuramos los diferentes puertos GPIO (Figura 4.31).

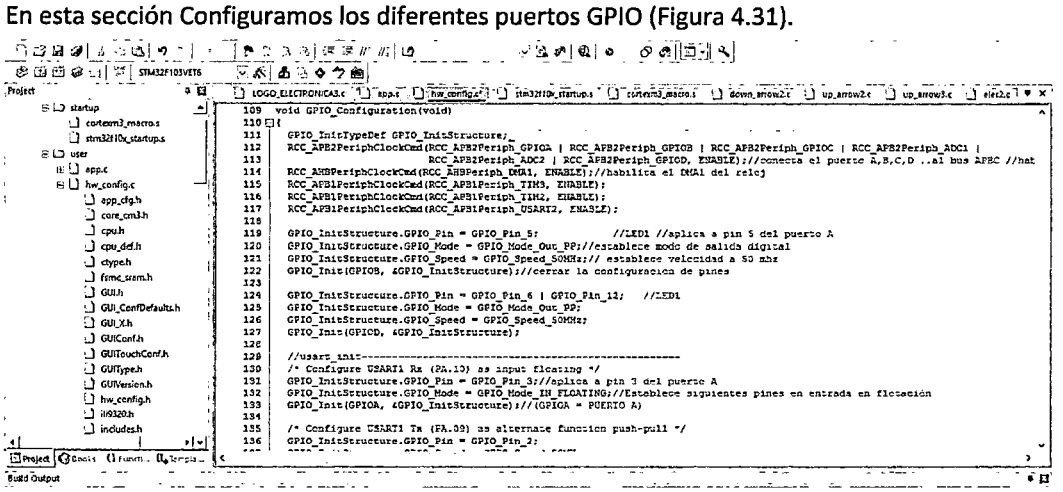


Figura 4.31: GPIO_Configuration

Las tres siguientes órdenes establecen la velocidad a 50 Mhz, que el pin actuará como una salida en Push-Pull, y la tercera a que pines afectarán estos parámetros, en concreto vemos que afecta a GPIO_Pin = GPIO_Pin_5, es decir, al Pin_5 y así para el resto de los pines a utilizar.

void GPIO_Configuration(void)

```
{
 GPIO_InitTypeDef GPIO_InitStructure;

        RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_GPIOB |
RCC_APB2Periph_GPIOC | RCC_APB2Periph_ADC1 |

         RCC_APB2Periph_ADC2 | RCC_APB2Periph_GPIOD, ENABLE);
        RCC_AHBPeriphClockCmd(RCC_AHBPeriph_DMA1, ENABLE);
        RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);
        RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);
        RCC_APB1PeriphClockCmd(RCC_APB1Periph_USART2, ENABLE);

 GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5;                    //LED1
 GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
 GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
 GPIO_Init(GPIOB, &GPIO_InitStructure);

 GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6 | GPIO_Pin_12;
//LED1
 GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
 GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
 GPIO_Init(GPIOD, &GPIO_InitStructure);
```

4. Desarrollo de la función: **DMA_Configuration**
   Configuramos las interrupciones USB en el caso del DMA (Figura 4.32).

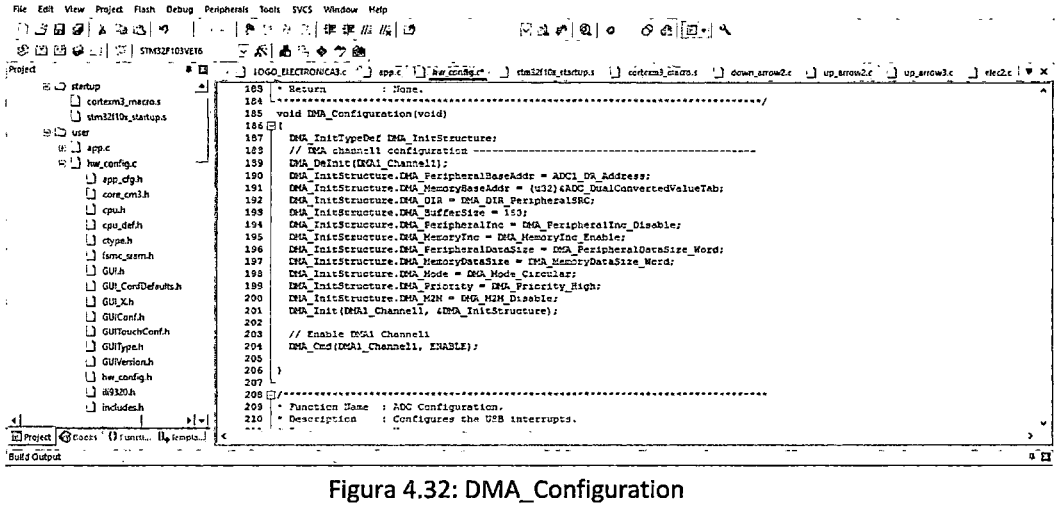


Figura 4.32: DMA_Configuration

Se da las respectivas configuraciones teniendo en cuenta que es para el Canal 1 y luego se habilita el Canal.

```
void DMA_Configuration(void)
```

```
{
 DMA_InitTypeDef DMA_InitStructure;
 // DMA channel1 configuration ----------------------------------------------
 DMA_DeInit(DMA1_Channel1);
 DMA_InitStructure.DMA_PeripheralBaseAddr = ADC1_DR_Address;
 DMA_InitStructure.DMA_MemoryBaseAddr = (u32)&ADC_DualConvertedValueTab;
 DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralSRC;
 DMA_InitStructure.DMA_BufferSize = 180;
 DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
 DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Enable;
 DMA_InitStructure.DMA_PeripheralDataSize = DMA_PeripheralDataSize_Word;
 DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_Word;
 DMA_InitStructure.DMA_Mode = DMA_Mode_Circular;
 DMA_InitStructure.DMA_Priority = DMA_Priority_High;
 DMA_InitStructure.DMA_M2M = DMA_M2M_Disable;
 DMA_Init(DMA1_Channel1, &DMA_InitStructure);

 // Enable DMA1 Channel1
 DMA_Cmd(DMA1_Channel1, ENABLE);

}
```

5. Desarrollo de la función: **ADC_Configuration**
   Configuramos las interrupciones USB en el caso del ADC (Figura 4.33).

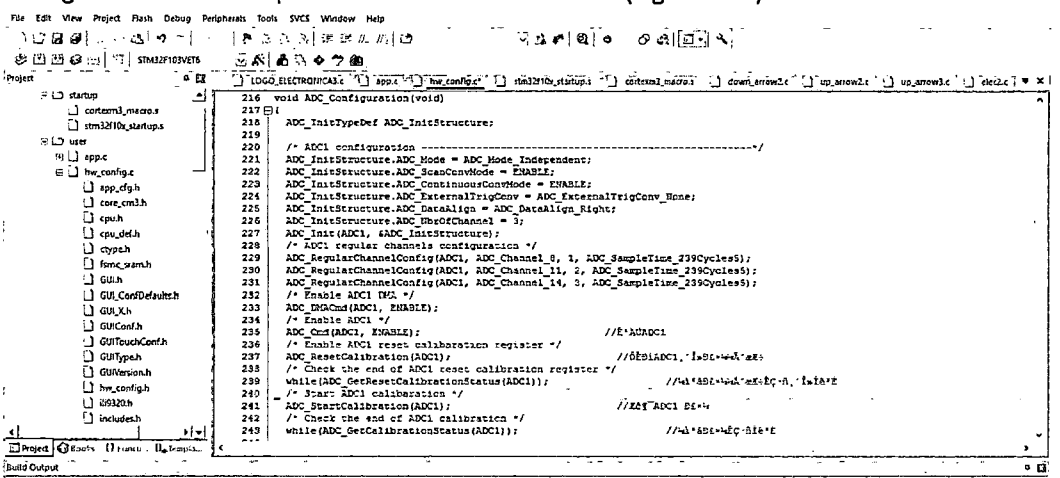


Figura 4.33: ADC_Configuration

Primero se configura la estructura para el ADC1, luego se pasa a la configuración de los canales regulares entre ellos Canal 8, Canal 11, Canal 14.
Habilitamos el ADC1, restablecemos el calibrado para ADC1y luego se le da comienzo a su calibración para ya habilitar el software de conversión para ADC1.

```
void ADC_Configuration(void)
{
```

```
ADC_InitTypeDef ADC_InitStructure;

/* ADC1 configuration ------------------------------------------------------*/
ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;
ADC_InitStructure.ADC_ScanConvMode = ENABLE;
ADC_InitStructure.ADC_ContinuousConvMode = ENABLE;
ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;
ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
ADC_InitStructure.ADC_NbrOfChannel = 3;
ADC_Init(ADC1, &ADC_InitStructure);
/* ADC1 regular channels configuration */
ADC_RegularChannelConfig(ADC1, ADC_Channel_8, 1,
ADC_SampleTime_239Cycles5);
      ADC_RegularChannelConfig(ADC1, ADC_Channel_11, 2,
ADC_SampleTime_239Cycles5);
      ADC_RegularChannelConfig(ADC1, ADC_Channel_14, 3,
ADC_SampleTime_239Cycles5);
```

6. Desarrollo de la función: **NVIC_Configuration**
   Configuramos la tabla de vectores ubicación de la base (Figura 4.34).

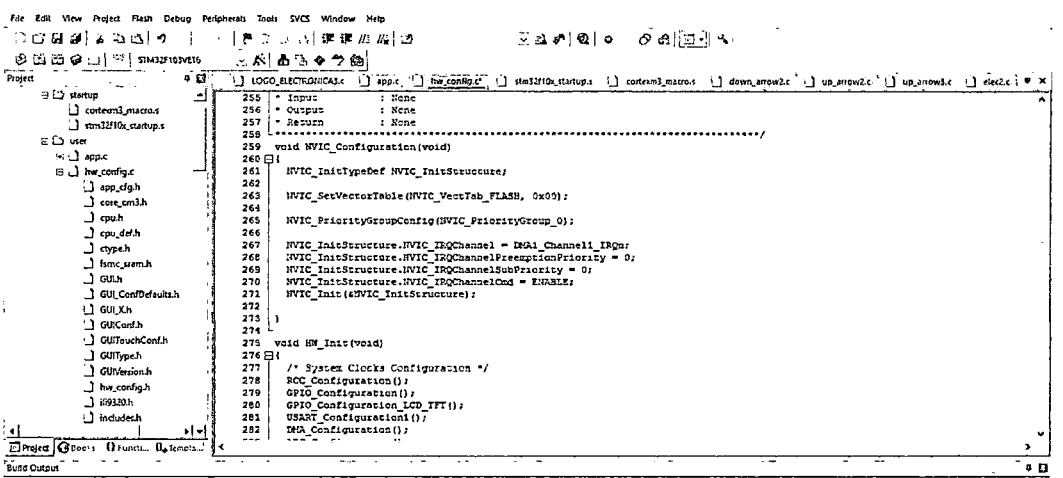


Figura 4.34: NVIC_Configuration

Se da la configuración par el NVIC que es un controlador de interrupciones teniendo en cuenta el DMA1 y el Canal 1 y luego se habilita.

```
void NVIC_Configuration(void)
{
      NVIC_InitTypeDef NVIC_InitStructure;

NVIC_SetVectorTable(NVIC_VectTab_FLASH, 0x00);
```

```
NVIC_PriorityGroupConfig(NVIC_PriorityGroup_0);

NVIC_InitStructure.NVIC_IRQChannel = DMA1_Channel1_IRQn;
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
        NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
        NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
NVIC_Init(&NVIC_InitStructure);

}
```

7. Desarrollo de la función: **void HW_Init(void)**
   Llamado de todas las funciones (Figura 4.35).

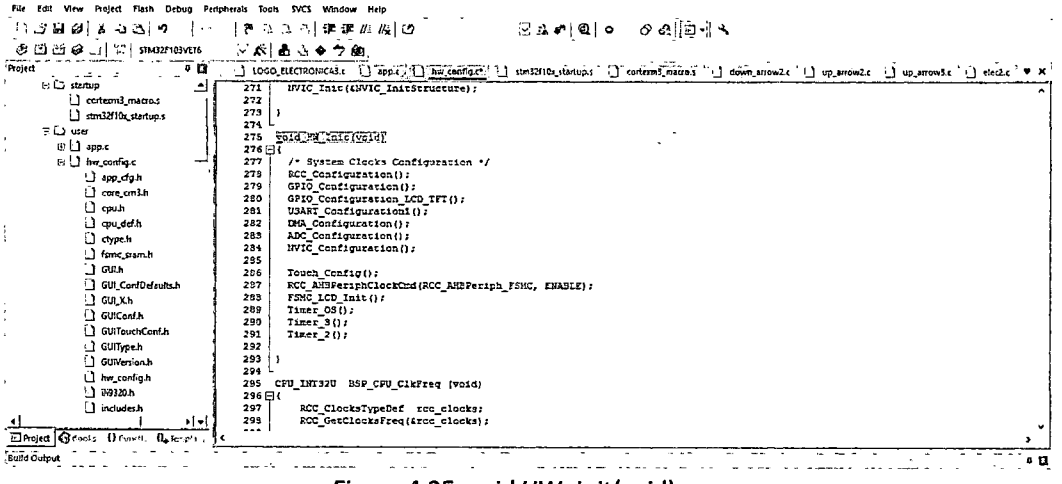


Figura 4.35: void HW_Init(void)

8. Desarrollo de la función: **void Timer_OS(void)**
   Configura el TIM4 *(*Figura 4.36).

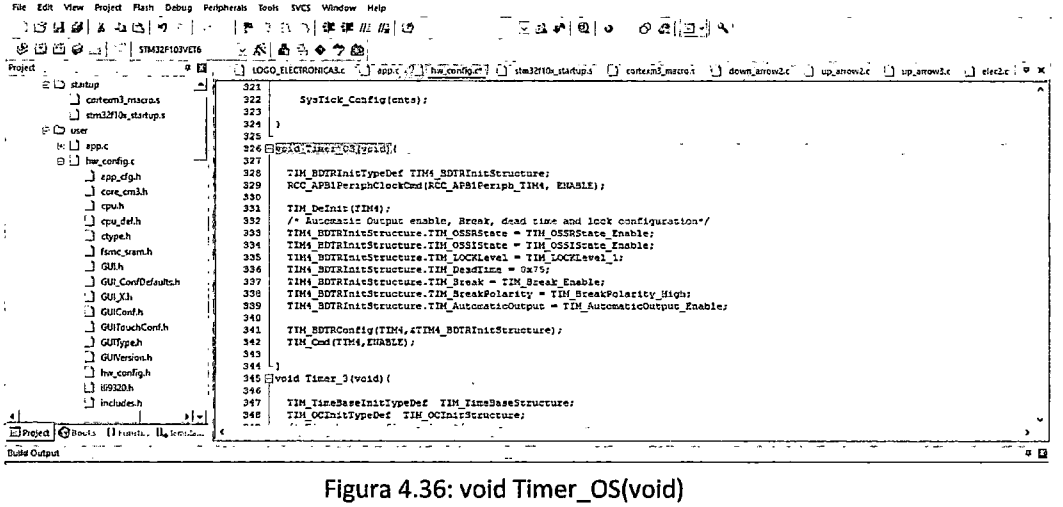


Figura 4.36: void Timer_OS(void)

En este caso configuramos la Salida automática que va a permitir la rotura, tiempo muerto y la configuración de bloqueo, luego de configurar la estructura del TIM4 lo habilitamos.

9. Desarrollo de la función: **void Timer_3(void)**
   Configura el TIM3 (Figura 4.37).

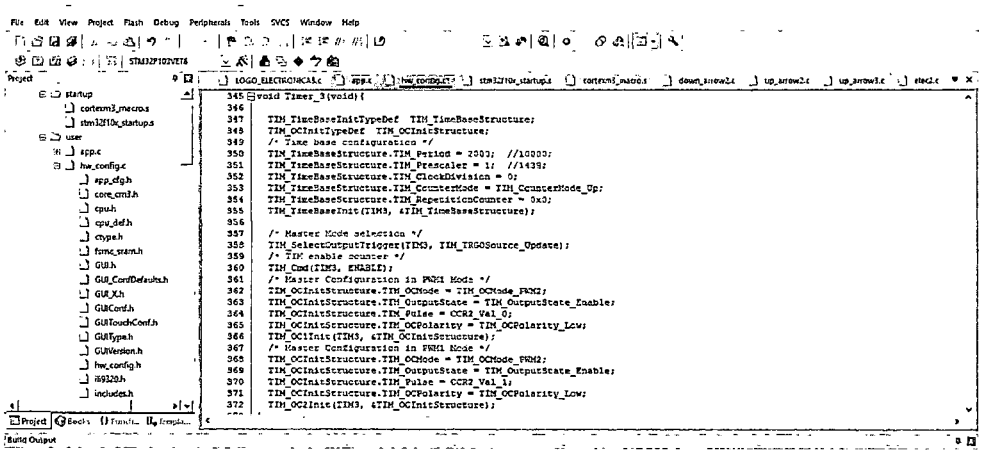


Figura 4.37: void Timer_3 (void)

Aquí se configura en el PWM2

10. Desarrollo de la función: **void Timer_2**
    Configura el TIM2 (Figura 4.38).

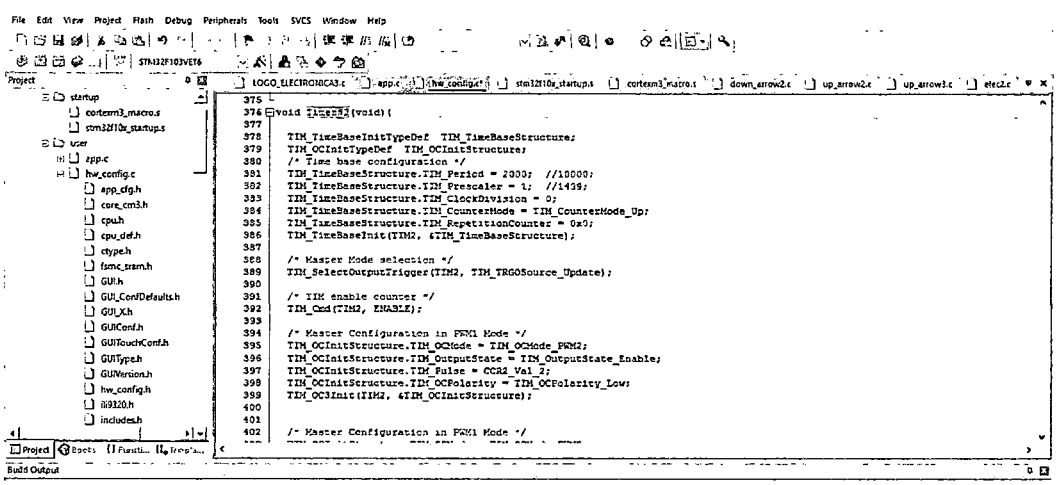


Figura 4.38: void Timer_2

11. Desarrollo de la función: **void USART_Configuration1(void)**
    Configuramos el USART1 (Figura 4.39).

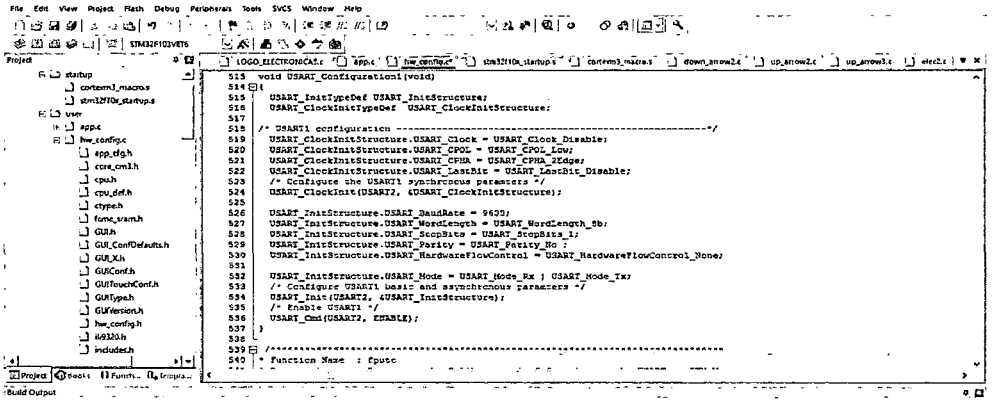


Figura 4.39: void USART_Configuration1 (void)

```
Void USART_Configuration1 (void)
{
 USART_InitTypeDef USART_InitStructure;
 USART_ClockInitTypeDef  USART_ClockInitStructure;

/* USART1 configuration ------------------------------------------------------*/
 USART_ClockInitStructure.USART_Clock = USART_Clock_Disable;
        USART_ClockInitStructure.USART_CPOL = USART_CPOL_Low;
        USART_ClockInitStructure.USART_CPHA = USART_CPHA_2Edge;
        USART_ClockInitStructure.USART_LastBit = USART_LastBit_Disable;
        /* Configure the USART1 synchronous paramters */
        USART_ClockInit(USART2, &USART_ClockInitStructure);

        USART_InitStructure.USART_BaudRate = 9600;
        USART_InitStructure.USART_WordLength = USART_WordLength_8b;
        USART_InitStructure.USART_StopBits = USART_StopBits_1;
        USART_InitStructure.USART_Parity = USART_Parity_No ;
        USART_InitStructure.USART_HardwareFlowControl =
USART_HardwareFlowControl_None;

        USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
        /* Configure USART1 basic and asynchronous paramters */
        USART_Init(USART2, &USART_InitStructure);
 /* Enable USART1 */
 USART_Cmd(USART2, ENABLE);
}
```

App.c

1. **Declaración de librerías y variables** (Figura 4.40).
   Son las mismas de Hw_config.c

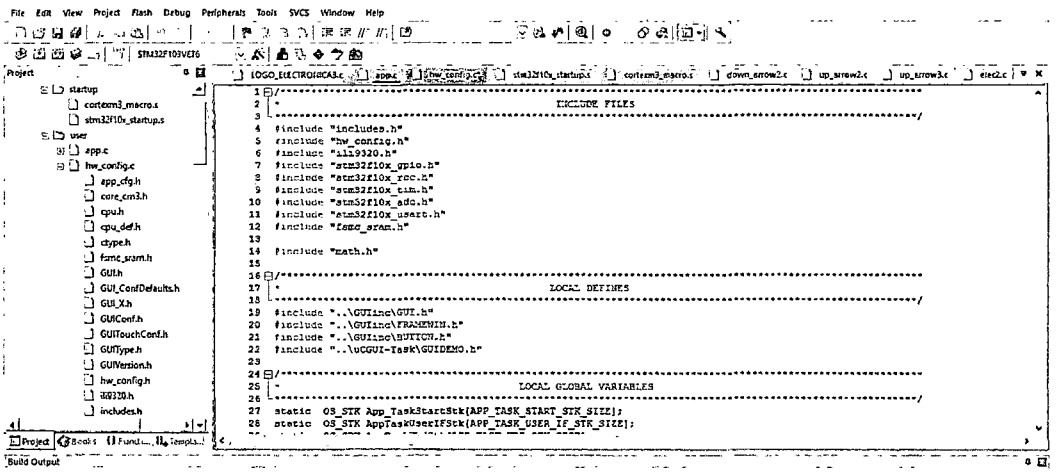


Figura 4.40: Declaración de librerías y variables

2. **Función main():**
   Punto de partida de la programación después de haber inicializado las variables (Figura 4.41).

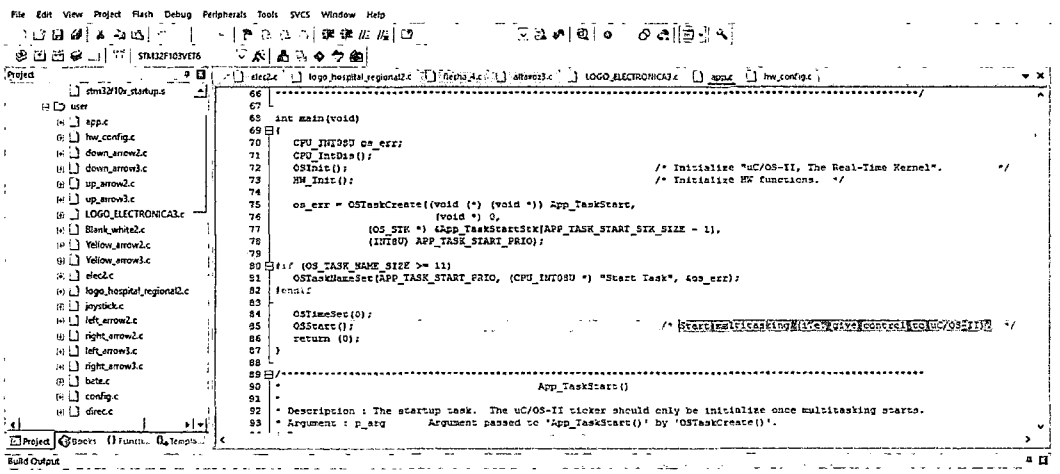
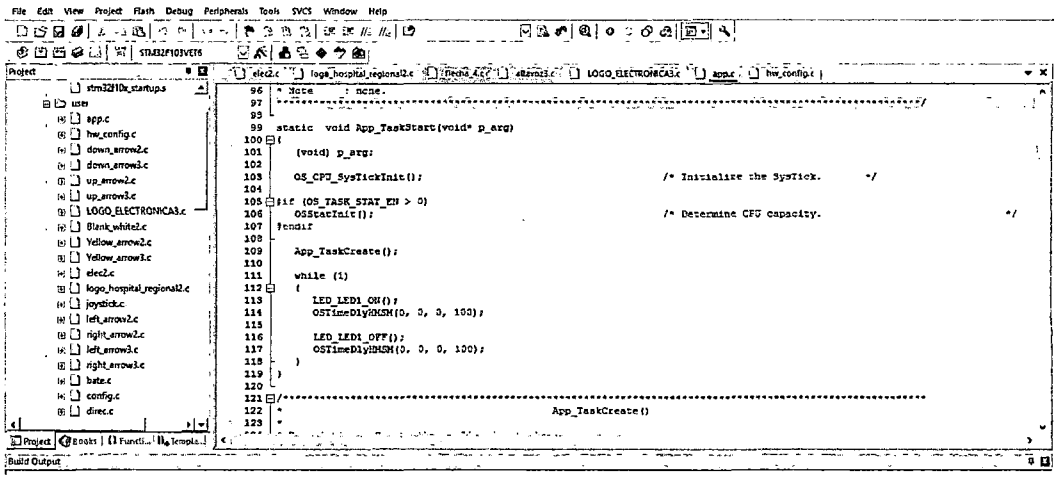


Figura 4.41: Función main

Se inicializa el Micro uC/OS-II, The Real-Time Kernel, para empezar sus multitareas, es decir darle el control al Micro uC/OS-II.

3. **Función App_TaskStart()**

   Es una tarea de inicio. Después de inicializar la multitarea se inicializa el ticker UC del OS-II. (Figura 4.42).

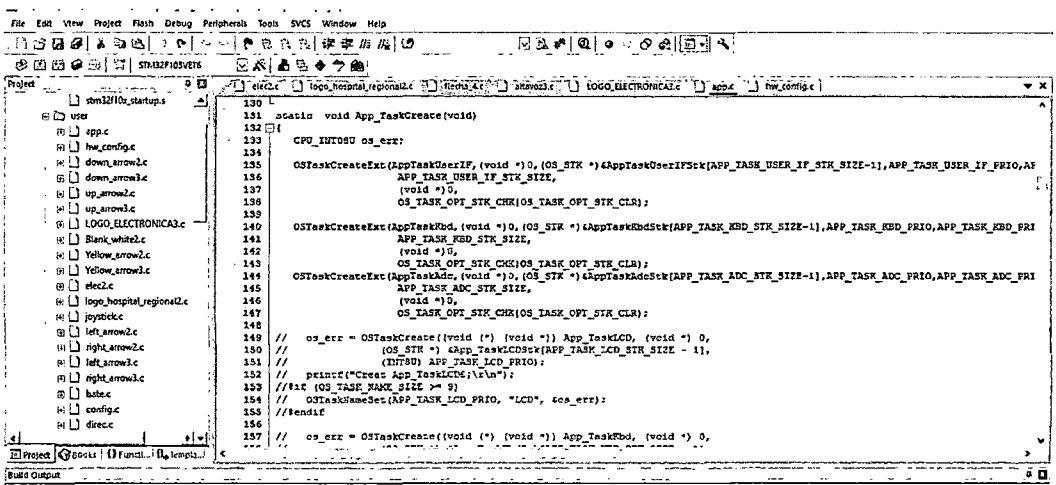


```
 96  * Note     : none.
 97  *********************************************************************************************************/
 98
 99  static  void App_TaskStart(void* p_arg)
100  {
101      (void) p_arg;
102
103      OS_CPU_SysTickInit();                                   /* Initialize the SysTick.      */
104
105  #if (OS_TASK_STAT_EN > 0)
106      OSStatInit();                                           /* Determine CPU capacity.      */
107  #endif
108
109      App_TaskCreate();
110
111      while (1)
112      {
113          LED_LED1_ON();
114          OSTimeDlyHMSM(0, 0, 0, 100);
115
116          LED_LED1_OFF();
117          OSTimeDlyHMSM(0, 0, 0, 100);
118      }
119  }
120
121  /*********************************************************************************************************
122  *                                          App_TaskCreate()
123  *
```

Figura 4.42: App_TaskStart ()

4. **Función App_TaskCreate()**

   Aquí se crean las tareas de la aplicación. (Figura 4.43).



```
130
131  static  void App_TaskCreate(void)
132  {
133      CPU_INT08U os_err;
134
135      OSTaskCreateExt(AppTaskUserIF,(void *)0,(OS_STK *)&AppTaskUserIFStk[APP_TASK_USER_IF_STK_SIZE-1],APP_TASK_USER_IF_PRIO,AP
136                      APP_TASK_USER_IF_STK_SIZE,
137                      (void *)5,
138                      OS_TASK_OPT_STK_CHK|OS_TASK_OPT_STK_CLR);
139
140      OSTaskCreateExt(AppTaskKbd,(void *)0,(OS_STK *)&AppTaskKbdStk[APP_TASK_KBD_STK_SIZE-1],APP_TASK_KBD_PRIO,APP_TASK_KBD_PRI
141                      APP_TASK_KBD_STK_SIZE,
142                      (void *)0,
143                      OS_TASK_OPT_STK_CHK|OS_TASK_OPT_STK_CLR);
144      OSTaskCreateExt(AppTaskAdc,(void *)0,(OS_STK *)&AppTaskAdcStk[APP_TASK_ADC_STK_SIZE-1],APP_TASK_ADC_PRIO,APP_TASK_ADC_PRI
145                      APP_TASK_ADC_STK_SIZE,
146                      (void *)0,
147                      OS_TASK_OPT_STK_CHK|OS_TASK_OPT_STK_CLR);
148
149  //  os_err = OSTaskCreate((void (*) (void *)) App_TaskLCD, (void *) 0,
150  //                        (OS_STK *) &App_TaskLCDStk[APP_TASK_LCD_STK_SIZE - 1],
151  //                        (INT8U) APP_TASK_LCD_PRIO);
152  //  printf("Creat App_TaskLCD;\r\n");
153  //#if (OS_TASK_NAME_SIZE >= 9)
154  //  OSTaskNameSet(APP_TASK_LCD_PRIO, "LCD", &os_err);
155  //#endif
156
157  //  os_err = OSTaskCreate((void (*) (void *)) App_TaskKbd, (void *) 0,
```

Figura 4.43: App_TaskCreate

5. **Función AppTaskUserIF()**

Esta es la tarea para la interfaz del usuario (Figura 4.44).

Esta tarea actualiza la pantalla LCD basado en mensajes que se le pasan por AppTaskKbd ( )

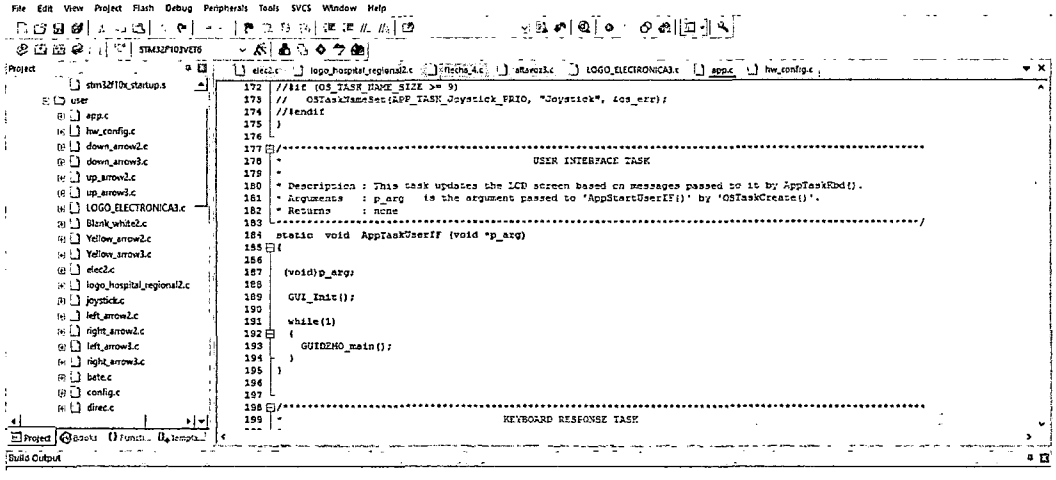


Figura 4.44: AppTaskUserIF

6. **Función AppTaskKbd()**

Área teclado de respuesta (Figura 4.45).

Esta tarea supervisa el estado de los pulsadores y pasa mensajes a la tarea de interfaz de usuario AppTaskUserIF ( )

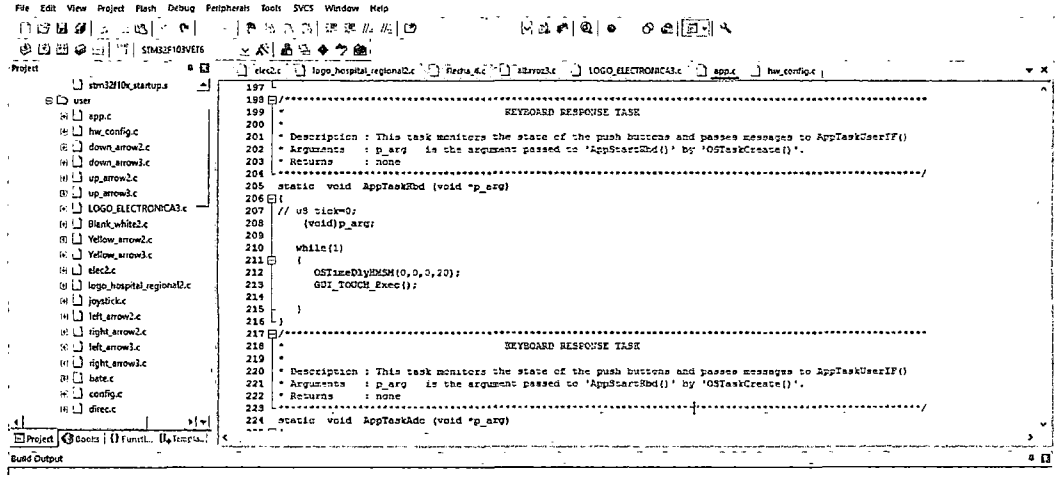


Figura 4.45: AppTaskKbd

7. **Función AppTaskAdc()**

En esta tarea primero hacemos uso del Conversor Analógico Digital ADC para el voltaje analógico a un número (Figura 4.46).

Luego calculamos el valor medio de corriente AVG.

Si al leer los bits de entrada de datos pertenecen al puerto D y al Pin 3, entonces colocamos la palanca en 1.

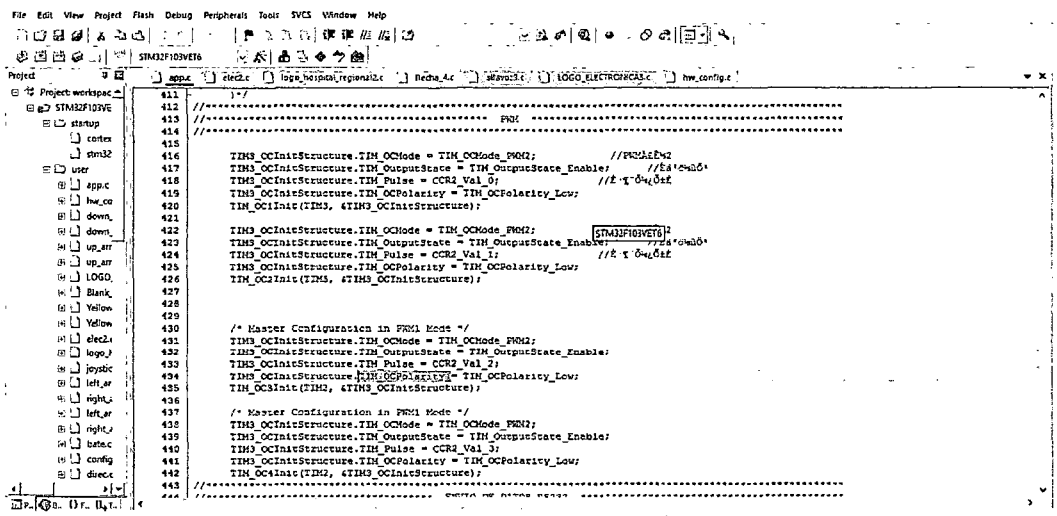
```
static  void  AppTaskAdc (void *p_arg)
{
// u8 tick=0;
   TIM_OCInitTypeDef  TIM3_OCInitStructure;
   int i = 0, j = 0, k = 0;
   uint16_t cadena2[2] = {0,0};
   uint16_t valor0[20],valor1[20],valor2[20];
   uint16_t valor3[20],valor4[20],valor5[20];
   uint16_t valor6[20],valor7[20],valor8[20];
   (void)p_arg;

  while(1)
  {
     OSTimeDlyHMSM(0,0,0,25);
//*************************************************************************************************
//*************************************************  ADC  *****************************************
//*************************************************************************************************
     for (i = 0; i < 20; i++){
       valor0[i]= ADC_DualConvertedValueTab[3*i];
       valor1[i]= ADC_DualConvertedValueTab[3*i+1];
       valor2[i]= ADC_DualConvertedValueTab[3*i+2];
     }
     for (j = 0; j < 20; j++){
       valor3[j]= ADC_DualConvertedValueTab[3*j+60];
       valor4[j]= ADC_DualConvertedValueTab[3*j+61];
       valor5[j]= ADC_DualConvertedValueTab[3*j+62];
     }
```

Figura 4.46: AppTaskAdc

Luego programamos el PWM,

esta funcionalidad se utiliza para controlar una forma de onda de salida o indicar cuando ha transcurrido un período de tiempo.

Primero se especifica modo de temporizador **TIM_OCMode**, se especifica la salida del temporizador comparado con el estado **TIM_OutputState**, se Especifica el valor del pulso que se va a cargar en el CCR **TIM_Pulse**, luego se especifica la polaridad de salida del temporizador **TIM_OCPolarity.** (Figura 4.47).



```
//*************************************************  PWM  *****************************************
//*************************************************************************************************
      TIM3_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM2;
      TIM3_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
      TIM3_OCInitStructure.TIM_Pulse = CCR2_Val_0;
      TIM3_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_Low;
      TIM_OC1Init(TIM3, &TIM3_OCInitStructure);

      TIM3_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM2;
      TIM3_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
      TIM3_OCInitStructure.TIM_Pulse = CCR2_Val_1;
      TIM3_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_Low;
      TIM_OC2Init(TIM3, &TIM3_OCInitStructure);


      /* Master Configuration in PWM1 Mode */
      TIM3_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM2;
      TIM3_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
      TIM3_OCInitStructure.TIM_Pulse = CCR2_Val_2;
      TIM3_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_Low;
      TIM_OC3Init(TIM2, &TIM3_OCInitStructure);

      /* Master Configuration in PWM1 Mode */
      TIM3_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM2;
      TIM3_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
      TIM3_OCInitStructure.TIM_Pulse = CCR2_Val_3;
      TIM3_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_Low;
      TIM_OC4Init(TIM2, &TIM3_OCInitStructure);
//*************************************************************************************************
```

Figura 4.47: modo de temporizador

LOGO ELECTRONICA.C

1. Definición de variables

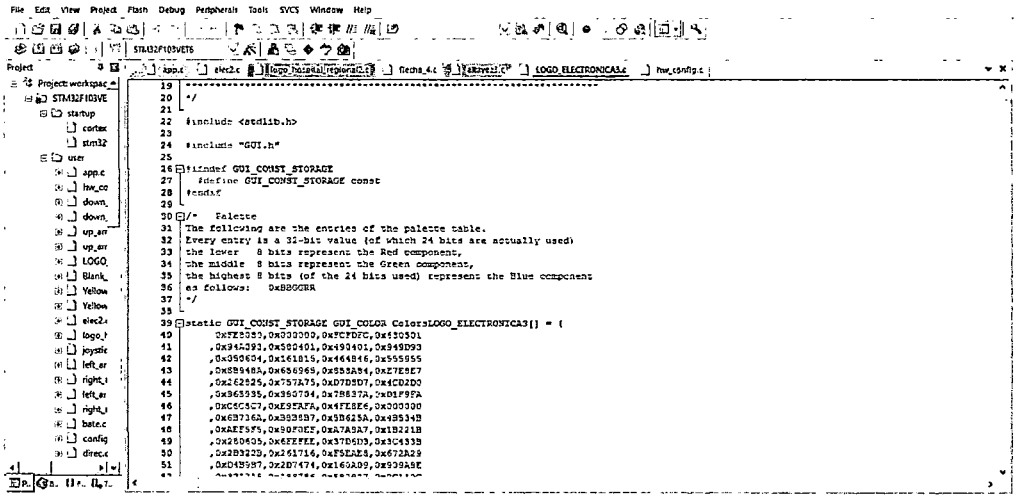


Figura 4.48: Definición de variables

2. Luego se da la Gama de colores **GUI_CONST_STORAGE GUI_COLOR ColorsLOGO_ELECTRONICA3[] = {** (Figura 4.49).

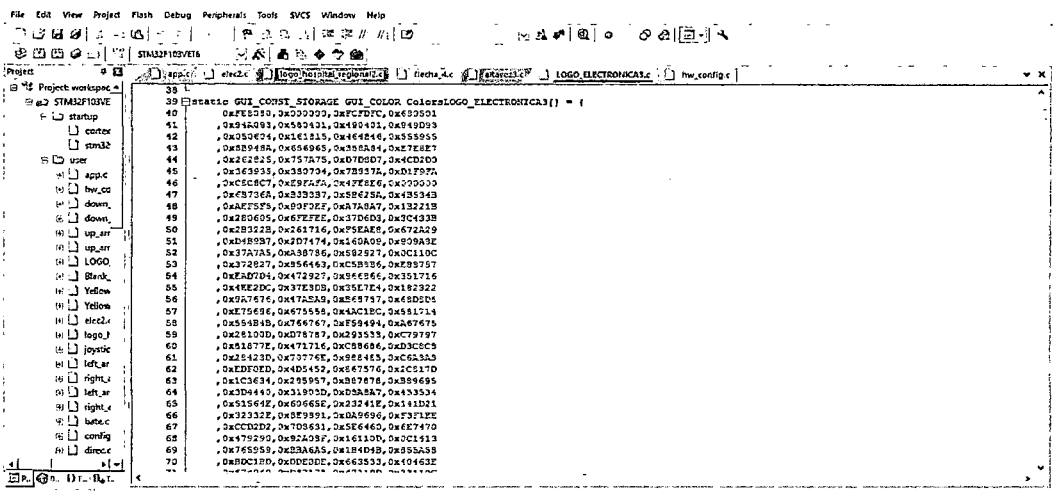


Figura 4.49: Gama de colores

3. Se da los datos de los pixeles (Figura 4.50).

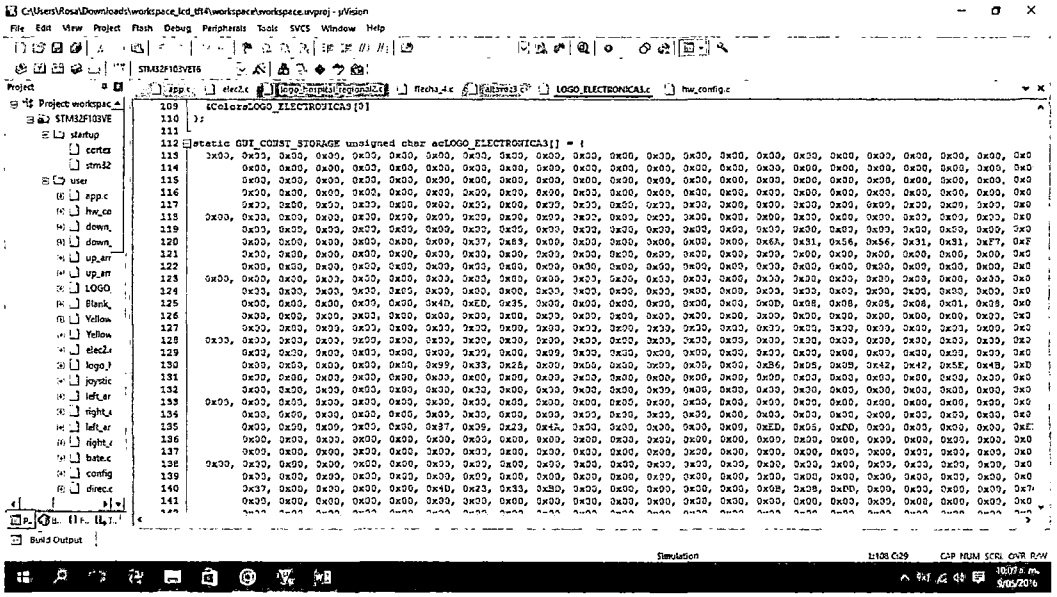


Figura 4.50: datos de los pixeles

4. Por ultimo en la función **GUI_CONST_STORAGE GUI_BITMAP bmLOGO_ELECTRONICA3 = {**
Se da el formato de mapa de bits colocando sus tamaños y ubicando los punteros, se dan los bytes y pixeles que se utilizaran por líneas (Figura 4.51).

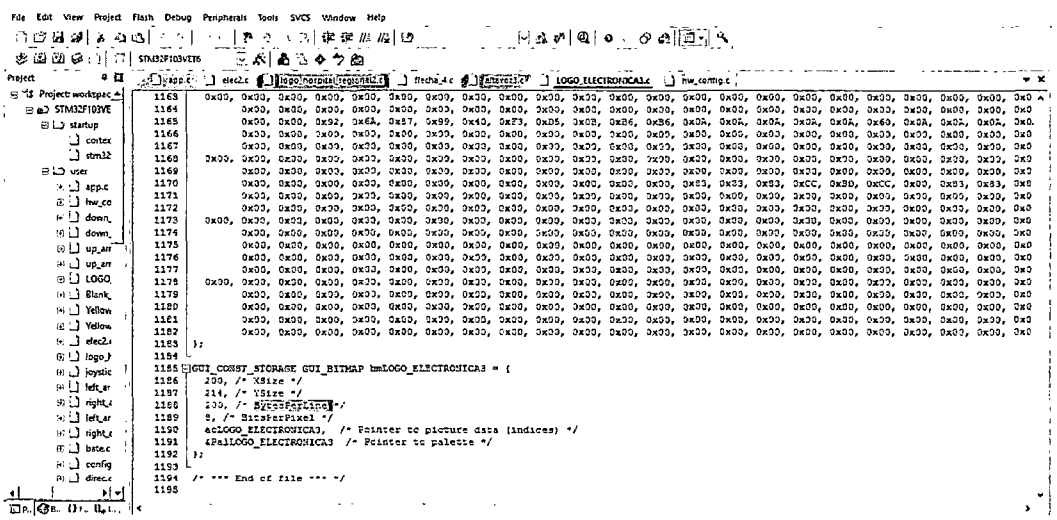


Figura 4.51: mapa de bits

V. DESARROLLO DEL PROYECTO

5.1. INTRODUCCION

En este capítulo se describe los requerimientos previos, diseños y construcción de las etapas de una silla eléctrica motorizada para discapacitados.
A continuación se muestra un diagrama general, dándonos ideas de los diferentes sistemas que componen a nuestro proyecto, así como también señalando la relación que existe entre ellos. (Figura 5.1).

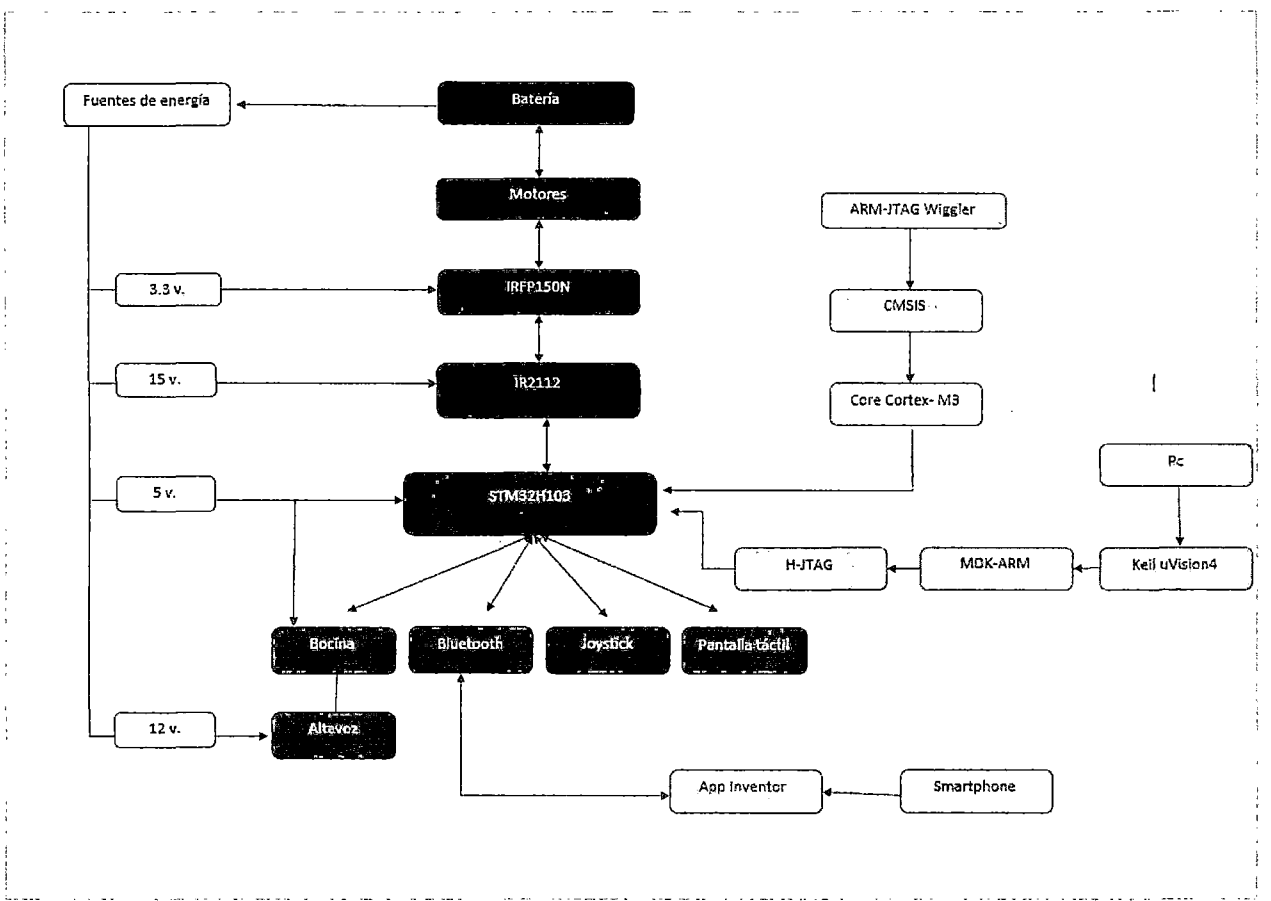


Figura 5.1: diagrama general

Donde:
Color negro: etapa de potencia
Color gris: etapa de control
Color verde: fuentes de alimentación/activación
Color naranja: software
Color amarillo: hardware del microcontrolador

5.2. DISEÑO Y CONSTRUCCION DEL HARDWARE

Dentro de esta sección se describirá detalladamente todas las mejoras que hemos creído conveniente.

5.2.1. BATERIAS

Este tipo de sillas motorizadas, utilizan 2 baterías de 12 voltios cada una, las cuales se conectan en serie; lo que significa que todo el sistema es alimentado por 24 voltios (fig. 5.2).

Fig. 5.2. Baterías de la silla eléctrica

5.2.2. MOTORES

Utiliza 2 motores del tipo A9Y1X00272 de la marca fortress scientific, de 24 voltios DC, con una corriente de 15 A, y una revolución de 172 RPM (fig.5.3).

Fig.5.3. Motores tipo A9Y1X00272

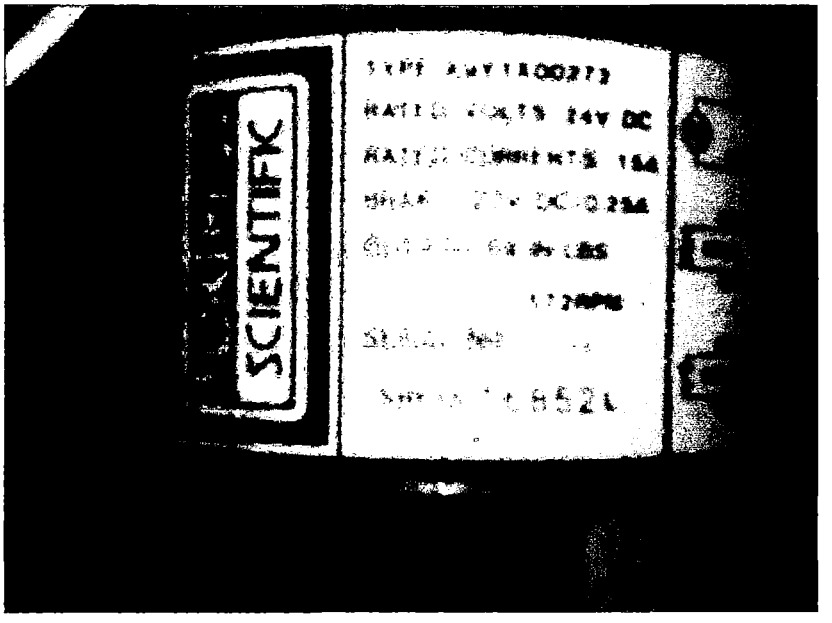


Fig.5.4. especificaciones técnicas del motor

5.2.3. ETAPA DE POTENCIA

Es la etapa encargada de controlar todos los movimientos de los motores, en nuestro caso hemos utilizado adaptación de voltajes, también el uso de puentes (puente H), asimismo se utilizaron fuentes de diferentes voltajes para alimentar los diferentes circuitos que explicaremos luego.

Para el presente proyecto se optó por utilizar el software proteus para diseñar y simular las siguientes etapas:

PUENTE H

Esta parte es la encargada de realizar y controlar los giros de los motores, sea en sentido horario como anti horario.

Un puente H es un circuito electrónico que permite a un motor eléctrico DC girar en ambos sentidos, avance y retroceso. Son ampliamente usados en robótica y como convertidores de potencia. Están disponibles como circuitos integrados, pero también pueden construirse a partir de componentes discretos. En nuestro caso, los encargados son los MOSFET IRFP150N.

ETAPA DE POTENCIA DE MOTOR DERECHO

Básicamente su función, es la de arrancar y detener a los motores (en este caso el motor derecho), en nuestro proyecto hemos utilizado el transistor MOSFET IRFP150N (puente H), el cual trabaja como 24 V., con una activación de entrada de 3.3V., el cual recibe las órdenes del microcontrolador hacia los motores a través del MOSFET IR2112 (adaptación de voltajes).

Donde IR1_H e IR1_L son las salidas altas y bajas, respectivamente de los IR2112, con un voltaje aproximado de 15 v., con una frecuencia de 18khz. (Fig.5.5).

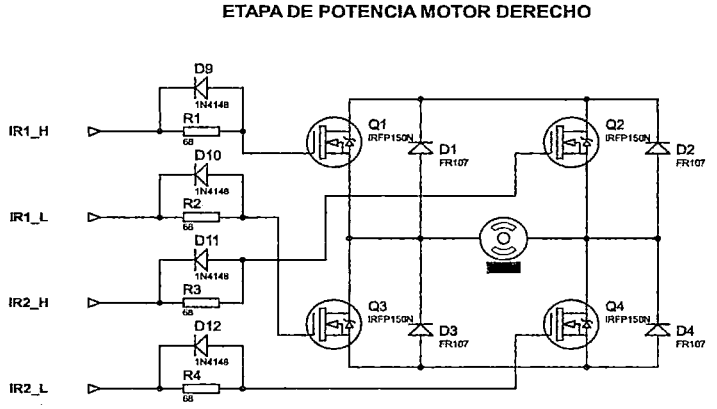


Fig.5.5. etapa de potencia de motor derecho, donde IR_ H e IR_L son las salidas de los IR2112

ETAPA DE POTENCIA DE MOTOR IZQUIERDO

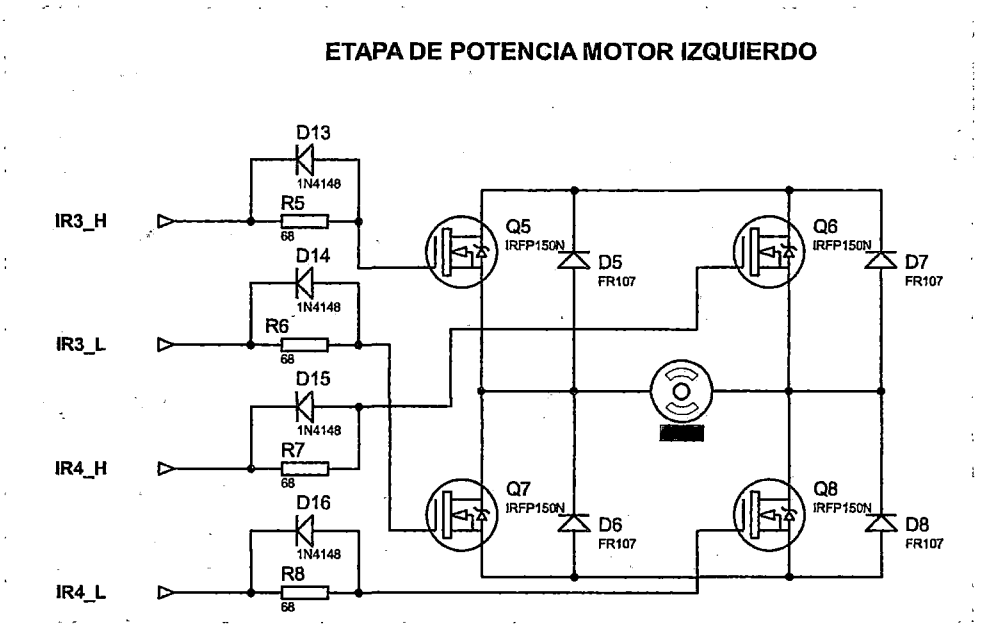


Fig.5.6. etapa de potencia de motor izquierdo, donde IR_ H e IR_L son las salidas de los IR2112

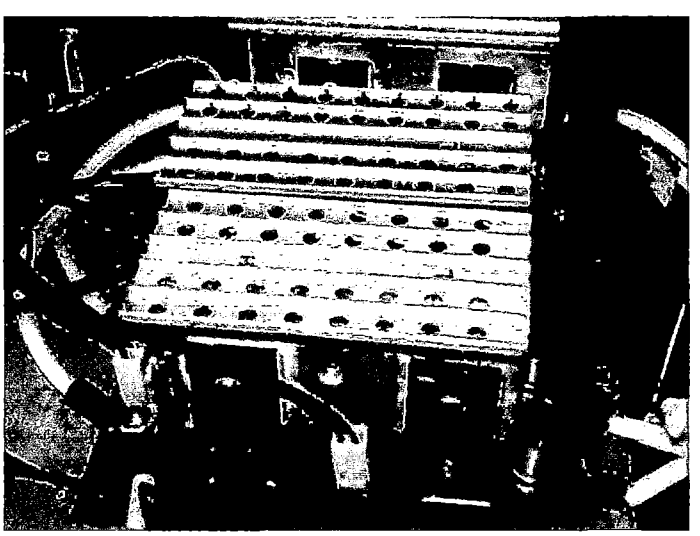


Fig.5.7. vista lateral de los MOSFET IRFP150N

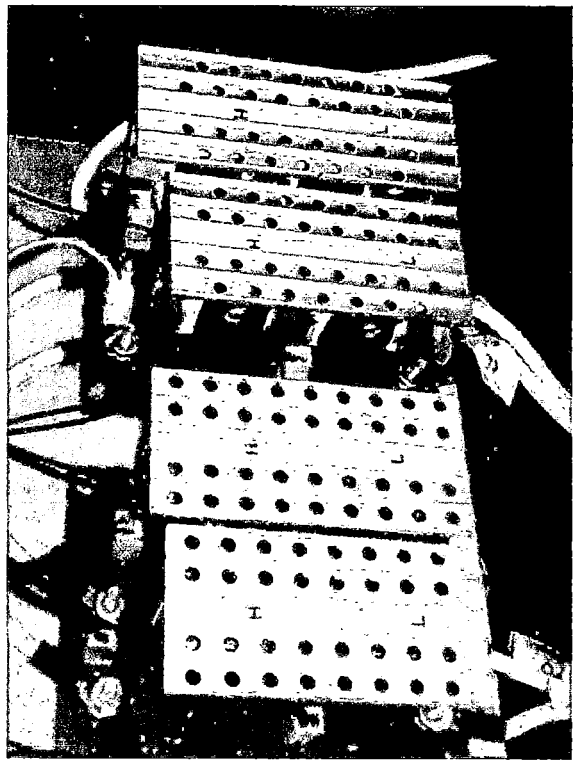


Fig.5.8. vista superior de los MOSFET IRFP150N

ADAPTACION DE VOLTAJES

Esta parte es la encargada de adecuar los voltajes tanto para los IRFP150N (24V.), como para el microcontrolador STM32H103 (3.3v- 5v).

En esta etapa se utilizaron los transistores MOSFET IR2112, debido a su gran robustez y facilidad de trabajo, así como facilidad para encontrarlo en cualquier electrónica local. (Fig.5.9).

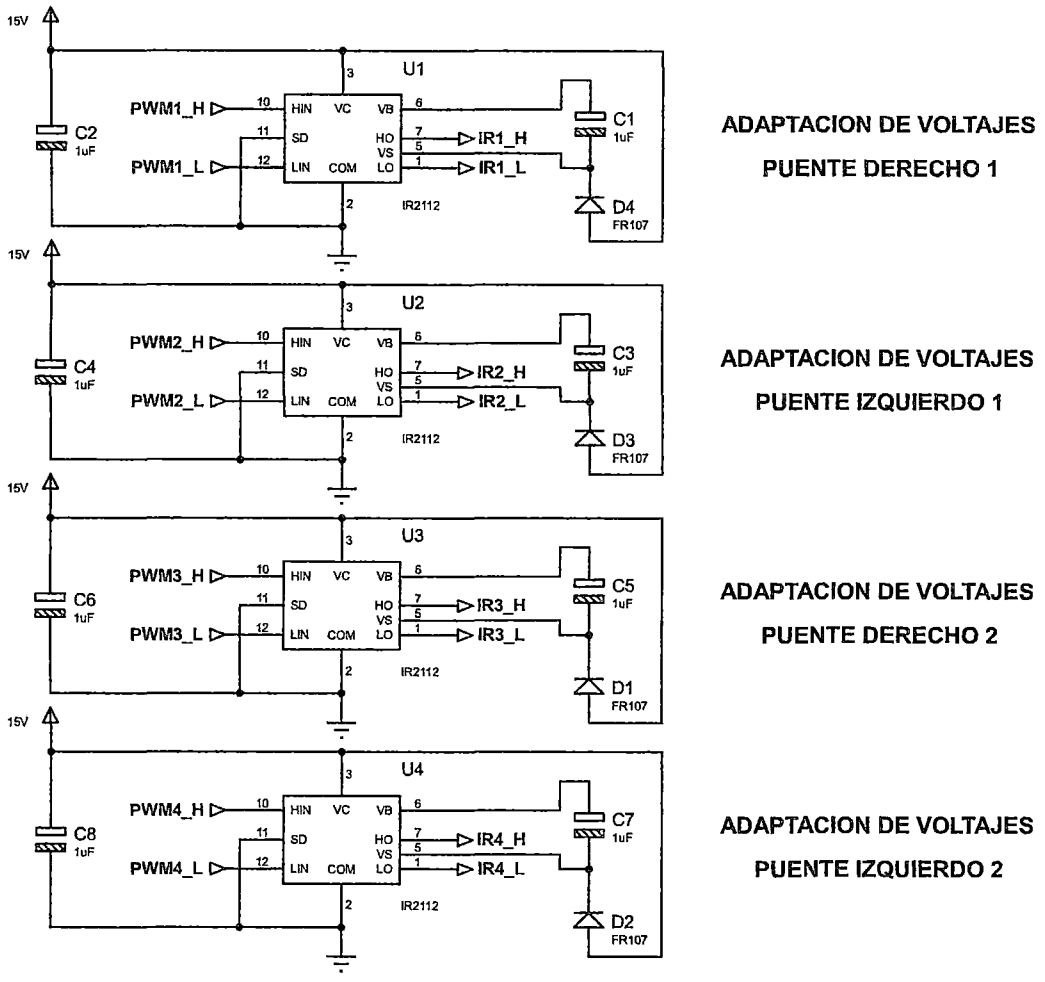


Fig.5.9. adaptación de voltajes

FUENTES DE VOLTAJE DE 15 V.; 5 V. Y 3.3 V

Fuente de voltaje de 15V. : La finalidad de esta fuente de voltaje, es la de alimentar a los transistores MOSFET IR2112 (adaptación de voltajes).

Fuente de voltaje de 5V. : La función de esta fuente de voltaje, es la de alimentar al microcontrolador STM32H103.

Fuente de voltaje de 3.3V. : La función de esta fuente de voltaje, es para activar la entrada de los transistores MOSFET IRFP150N. (Que son los encargados de controlar los movimientos de los motores). (Fig.5.10).

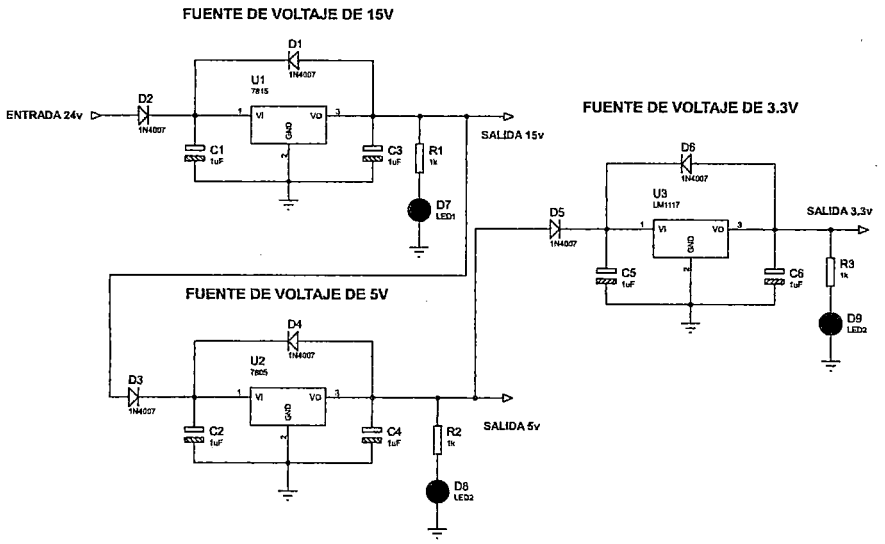


Fig.5.10. diseño de las fuentes de voltajes

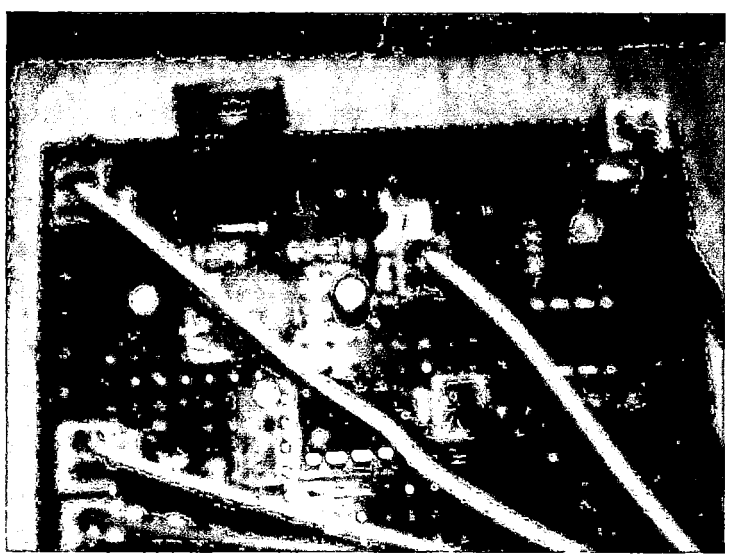


Fig.5.11. fuente de voltaje de 5V. y 3.3V.

FUENTES DE VOLTAJES DE 12V. Y DE 5V.

Estas fuentes de voltaje son exclusivas del audio de nuestra silla.

Fuente de voltaje de 12V. : La función de esta fuente de voltaje, es la de alimentar el altavoz, donde utilizamos el regulador de voltaje LM317.

Fuente de voltaje de 5V. : La función de esta fuente de voltaje, es la de alimentar el circuito de la bocina (LM 555). (Fig.5.12).

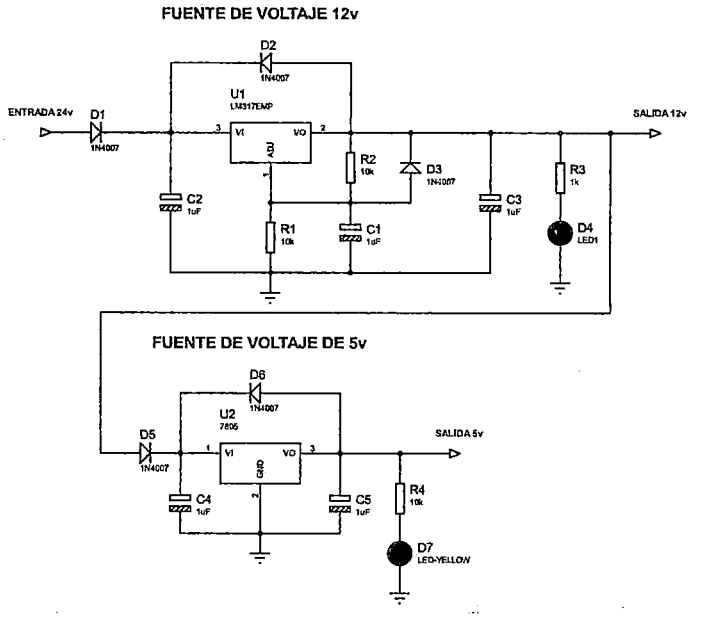


Fig.5.12.diseño de fuentes de voltaje de 12V. y 5V.

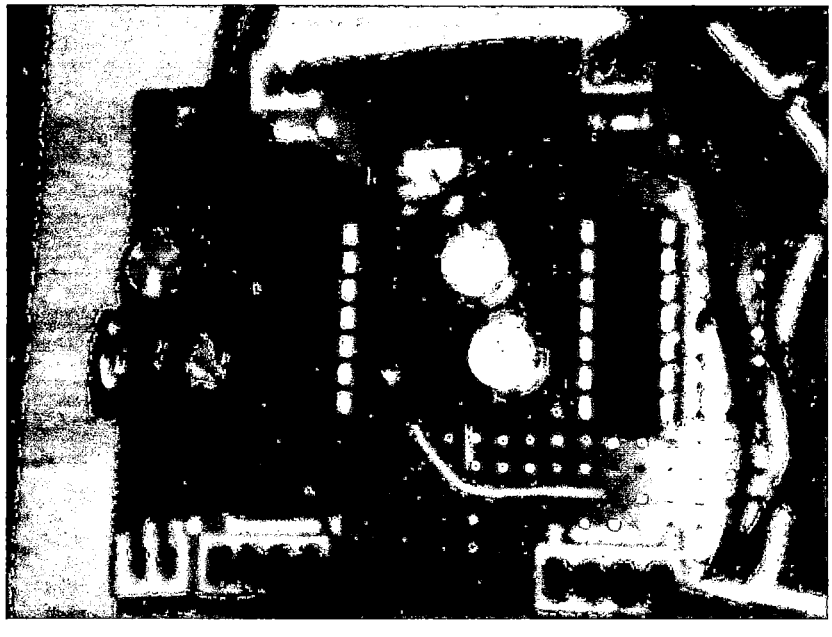


fig.5.13.fuentes de voltaje

Circuito de la bocina

Circuito de bocina o claxon, utilizado básicamente para uso voluntario del usuario, formado principalmente por el integrado 555, un transistor BD135 (fig.5.14 y fig.5.15).

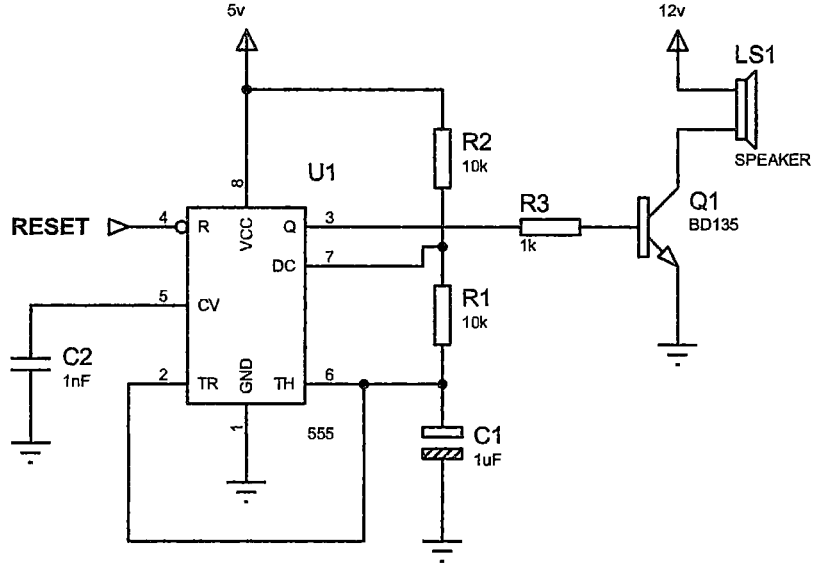


Fig.5.14.diseño del circuito de bocina

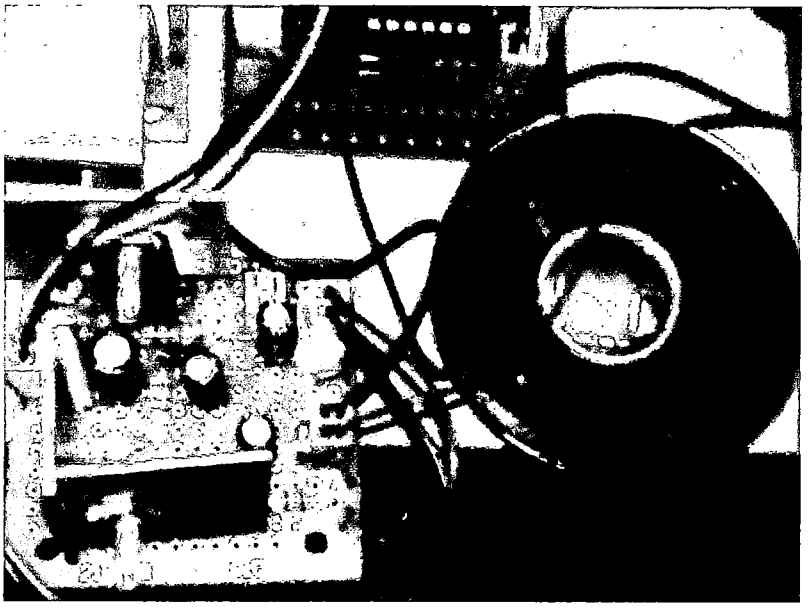


Fig.5.15.circuito de bocina

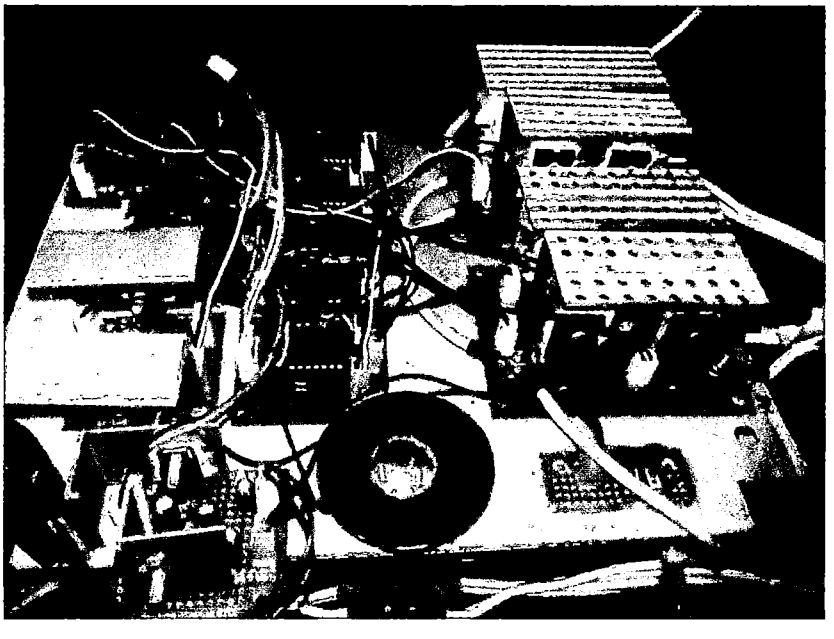


Fig.5.16.vista superior de toda la etapa de potencia

5.2.4. ETAPA DE CONTROL

Es la etapa que interactúa con el usuario, aunque también se encuentra en esta parte el microcontrolador STM32H103.

Microcontrolador (STM32H103)

El microcontrolador STM32H103, es el encargado de realizar y dirigir, la comunicación con las diferentes etapas de la silla eléctrica motorizada para discapacitados.

.) Microcontrolador STM32H103-adaptacion de voltajes: la comunicación es digital, por pulsos PWM.

En este caso utilizamos PWM H y PWM L (altos y bajos respectivamente).

PWM L: el PWM bajo, son básicamente señales de activación para los IR2112.

PWM H: el PWM alto, es en si el encargado de llevar la información a través de pulsos, en nuestro caso a una frecuencia de 18khz.

Los IR2112, cumplen la función principal de elevar el voltaje proveniente del microcontrolador (3.3v.) a 15v., en nuestro caso.

Su rango es de 3-25v., a su vez, las salidas de los IR2112 (IR-H e IR-L), sirven para el funcionamiento de los IRFP150N, con un rango de 0-24v.

.)Microcontrolador STM32H103-joystick: en este caso nuestro joystick, es del modelo thumbs, y es analógico, por lo tanto, su comunicación en nuestro caso es netamente analógica, a 3.3v. DC., unidos por 2 conexiones analógicas.

.) Microcontrolador STM32H103-pantalla LCD: la comunicación es analógica (3.3v.), a través de bus de datos, en nuestro caso la pantalla táctil, cuenta con 16 pines, 4 pines para controles, 4 pines para pantalla táctil, 2 pines para alimentación y 2 pines para tierra.

Pantalla táctil (ILI 9325)

Es un módulo practico con pantalla LCD TFT de 2.4 pulgadas con membrana sensible al tacto y controladores, para ambos incluidos en el módulo.

Este módulo permite conectar fácilmente una pantalla LCD a color a cualquier desarrollo con microcontroladores. El panel tiene integrado un controlador ILI9325 y un control táctil ADS7843.

El modulo dispone también de una ranura de conexión para una tarjeta SD. El LCD se controla mediante un bus paralelo de 16 bits, por lo que se recomienda su uso con microcontroladores con buen número de pines de I/O. (Fig.5.17).

Características:

- Controlador de pantalla táctil: ILI9325 y ADS7843.
- Controlador de pantalla con buffer de video incluido.
- Resolución de pantalla 240 *320.
- Regulador de 3.3V. Incluido.
- Nivel lógico de los pines de IO: 3.3V.
- Dimensiones del módulo con conectores incluidos: 75(longitud)*55(ancho)*14mm (alto).

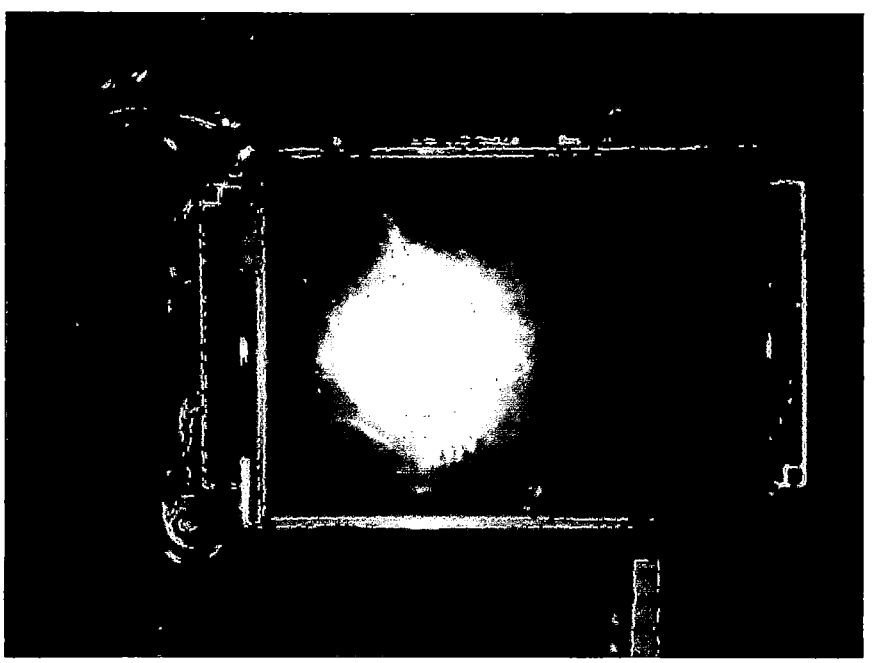


Fig.5.17.pantalla táctil ILI9325

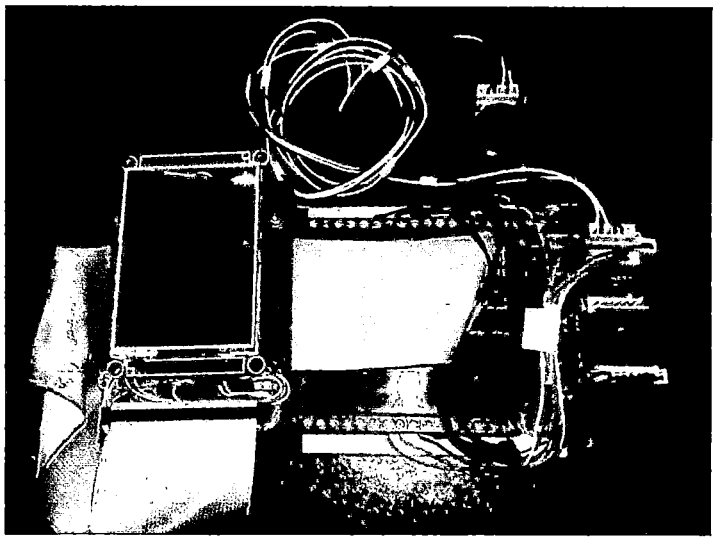


Fig.5.18.pantalla táctil y joystick

Joystick

Un joystick viene a ser una palanca de control que permite desplazarse manualmente, y con gran rapidez, el cursor en una pantalla de computadora o videojuego; se usa especialmente en programas informáticos de juego.

En nuestro proyecto hemos utilizado un joystick modelo thumbs, el cual es analógico, funciona con 3.3v., en su estructura interna cuenta con 2 potenciómetros unidos entre sí, de tal forma que pueda hacer giros rápidos y precisos. Se conecta al microcontrolador por medio de 2 salidas analógicas.

Decidimos utilizar este modelo de joystick por su alta sensibilidad y fácil conexión, en comparación con otros modelos. (Fig.5.19).

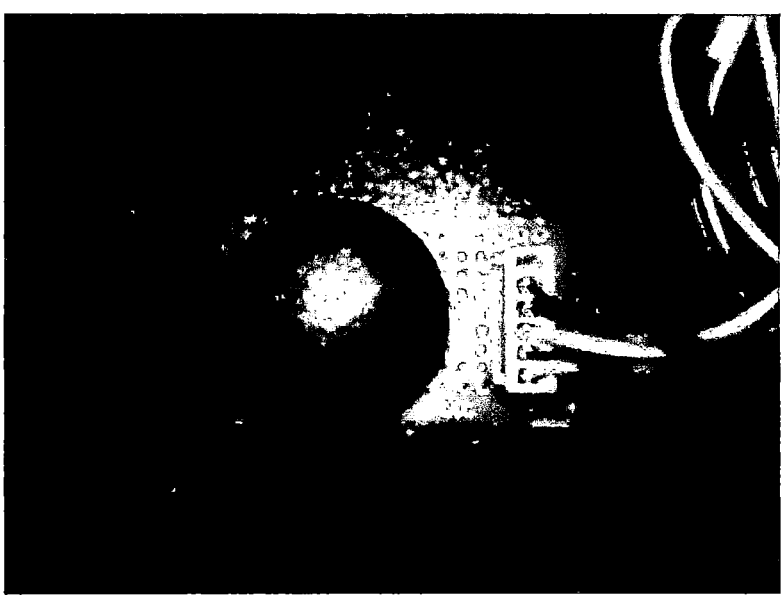


Fig.5.19.vista superior del joystick

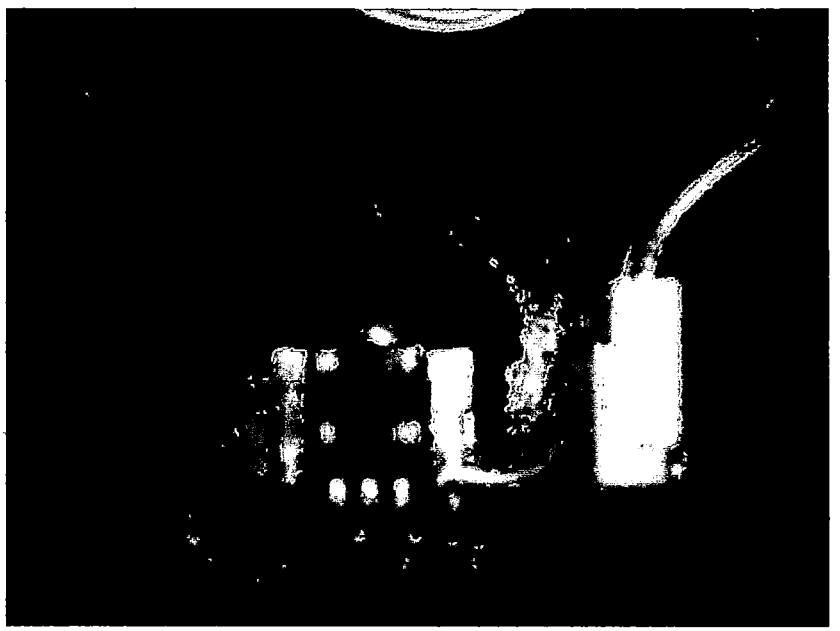


Fig.5.20.vista lateral del joystick

Bluetooth (HC06)

El modulo Bluetooth (HC06) es un dispositivo muy fácil de obtener, económico y sencillo de utilizar. (Fig.5.21).

Ventajas principales

Además de su pequeño tamaño y sus buenas características de transmisión y recepción que le brindan un alcance muy amplio, tiene bajo consumo de corriente, tanto en funcionamiento como en modo espera.

Cuando realiza un enlace con otro dispositivo es capaz de recordarlo en su memoria y no solicita validación alguna.

Su alimentación es de 3.3V. Y es de bajo consumo (8mA).tiene un rango aproximadamente de 15 metros.

La comunicación es por medio del protocolo RS232 (puertos USART), su transmisión es de 8 bits, a una velocidad de 115200 kb/s.

Trabaja a una frecuencia de 2.4 GHZ, banda ISM Y modulación GFSK.

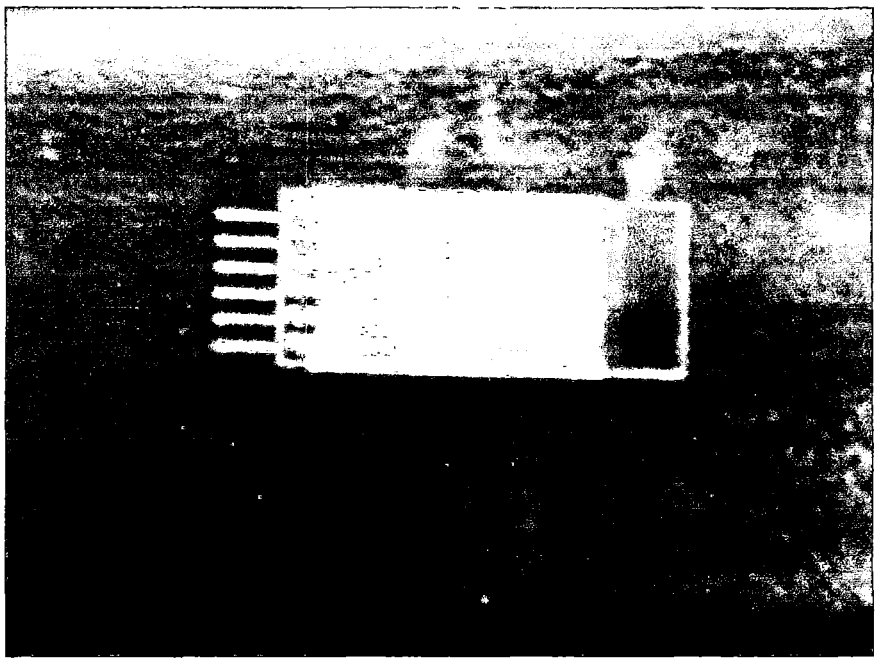


Fig.5.21.bluetooth HC06

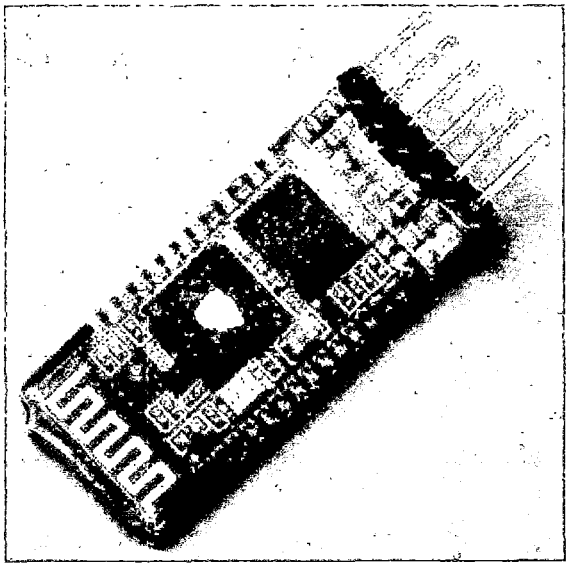


Fig.5.22.vista posterior del bluetooth HC06

Configuración del módulo bluetooth HC-06 usando comandos AT

En esta parte vamos a explicar cómo configuramos nuestro módulo HC-06, cambiar la velocidad de transmisión, el nombre y código de vinculación de nuestro hc-06 entre otras cosas. (Figura 5.23).

EL modulo Bluetooth HC-06 viene configurado de fábrica como Esclavo y no se lo puede cambiar, pero otras características si las podemos configurar usando comandos AT, estas caracterizas vienen por defecto con valores predeterminados que se muestran a continuación:

- Nombre por defeco: "linvor" o "HC-06"

- Código de emparejamiento por defecto: 1234

- La velocidad por defecto (baud rate): 9600

EL Modulo HC-06 tiene dos estados los cuales es importante conocer:

- Modo AT (Desconectado)
- Modo Conectado

Hacemos la comunicación entre la PC y el módulo de forma Directa usando un conversor USB-Serial. (Figura 5.24).

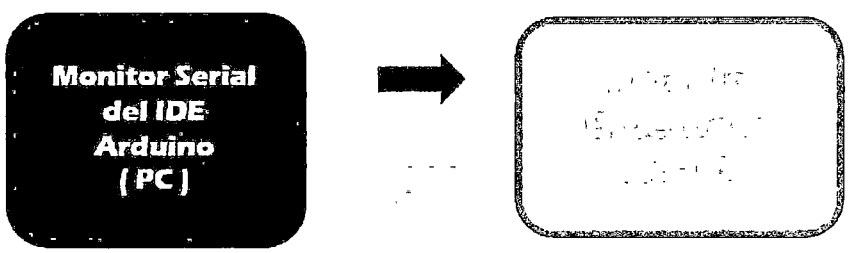


Figura 5.23: Configuración del módulo bluetooth HC-06 usando comandos AT

Las conexiones serían las siguientes:

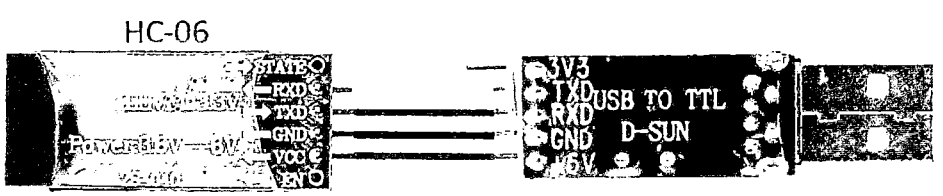


Figura 5.24: conversor USB-Serial

En nuestro caso usamos un conversor USB serial PL2303 que se ha instalado como puerto serial COM5. (Figura 5.25).

Una vez hecho las conexiones correspondientes, abrimos el Monitor serial:

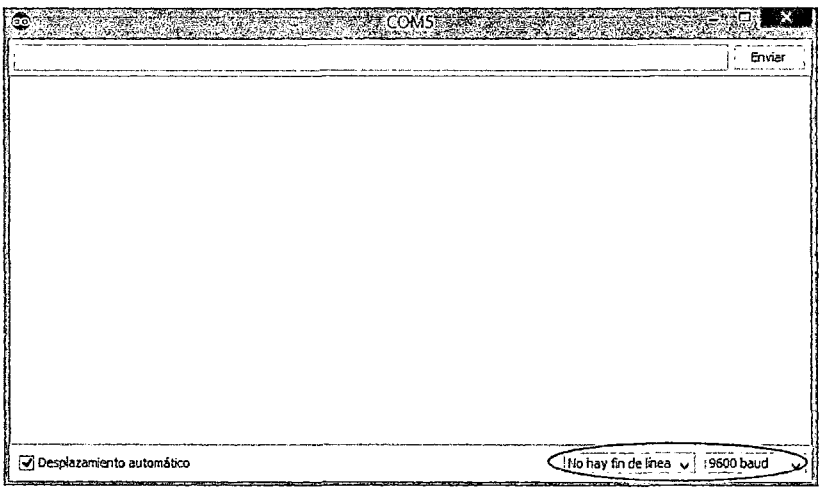


Figura 5.25: puerto serial COM5

En la parte inferior debemos escoger "No hay fin de línea" y la velocidad.

Hecho esto Podemos empezar a enviar los comandos AT a nuestro Bluetooth

Test de comunicación

Lo primero es comprobar si nuestro bluetooth responde a los comandos AT

Enviar: AT

Recibe: OK

Si recibimos como respuesta un OK entonces podemos continuar, sino verificar las conexiones.

Cambiamos nombre de nuestro módulo HC-06

En nuestro caso, nuestro módulo bluetooth se llama *silla*.

Enviar: AT+NAME<silla>

Respuesta: OKsetname

Configuramos la velocidad de comunicación

La velocidad por defecto es de 9600 baudios, en nuestro caso será de 115200. (Figura 5.26).

Enviar: AT+BAUD<8>

Respuesta: OK<115200>

Numero---baudrate

1 ---------1200

2 ---------2400

3 ---------4800

4 ---------9600

5 ---------19200

6 ---------38400

7 ---------57600

8 ---------115200

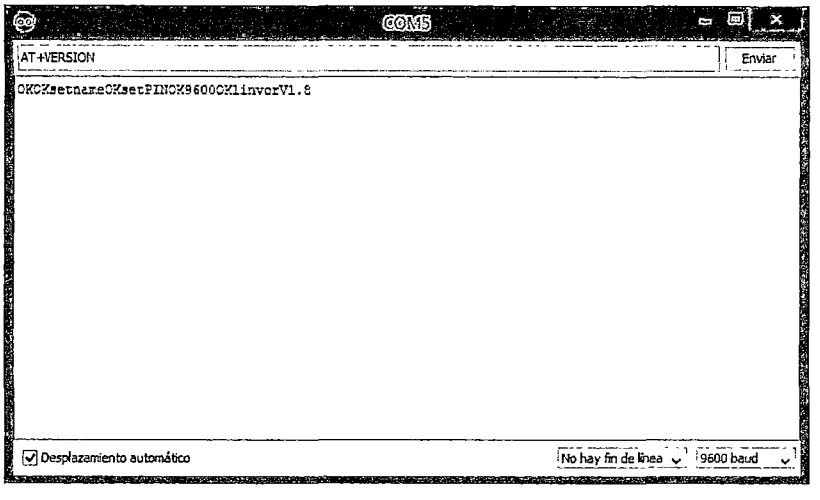


Figura 5.26: Configuramos la velocidad de comunicación

Existen más comandos, pero en nuestro caso, solo hemos modificado los mencionados. De esta manera quedaría habilitado nuestro bluetooth, listo para usarse.

Menús de la pantalla

El menú, es muy sencillo y fácil de operar, por lo que el usuario se va a familiarizar con mucha rapidez a él, a la hora de diseñar, se tuvo en cuenta ese aspecto, por el tipo de usuarios que mayormente utiliza este tipo de sillas, asegurándonos de que cuenten con la máxima comodidad posible.

Menú Principal:

En el menú principal se encuentran 4 opciones, como son: modo manual (joystick), modo digital (pantalla táctil), porcentaje de carga de batería y la opción de herramientas (configuraciones). (fig.5.27).

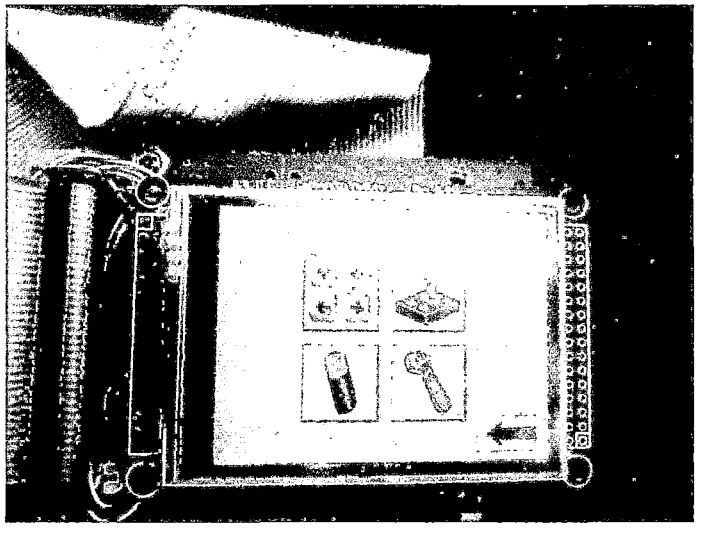


Fig.5.27.menu principal

Seleccionando el modo de control: existen 2 modos, tanto manual (joystick), como digital (pantalla táctil)

- **Modo manual:** los movimientos se controlan por medio del joystick. (fig.5.28).

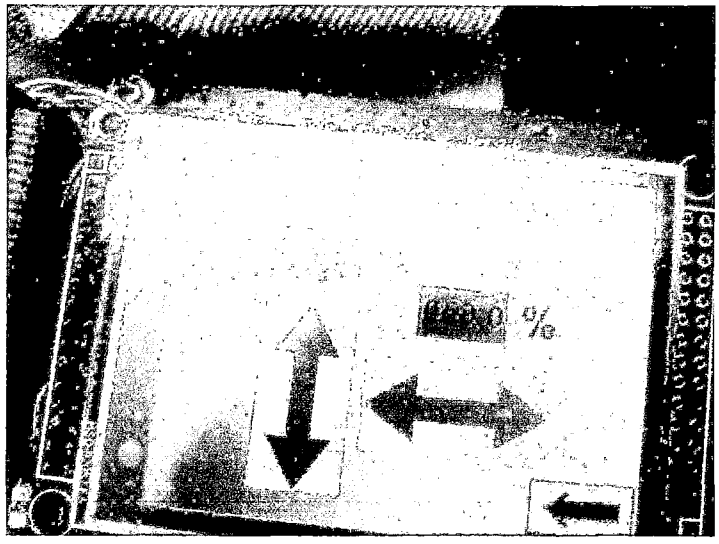


Fig.5.28.modo manual

- **Modo Digital:** los movimientos son controlados por medio de los controles que encuentran en la pantalla táctil. (Fig.5.29).

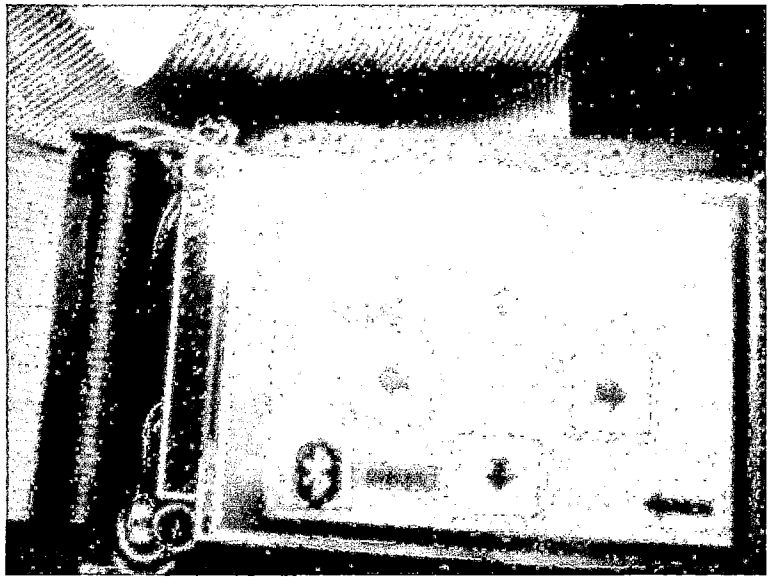


Fig.5.29.modo digital

Configurando la velocidad: en este caso, hemos creído conveniente controlar la velocidad por rangos (porcentajes), debido a que no se sabe exactamente el estado de salud del usuario, por lo que la velocidad no sería la misma. Y quedaría a elección del mismo usuario, teniendo en cuenta que la velocidad de nuestra silla es aproximadamente de 7 km/h. (Fig.5.30).

Ejm.

V1 (desde 0% hasta 20%)

V2 (desde 20% hasta 40%) y así sucesivamente.

En nuestro menú ingresamos a configuraciones, y nos aparecerá la siguiente pantalla.

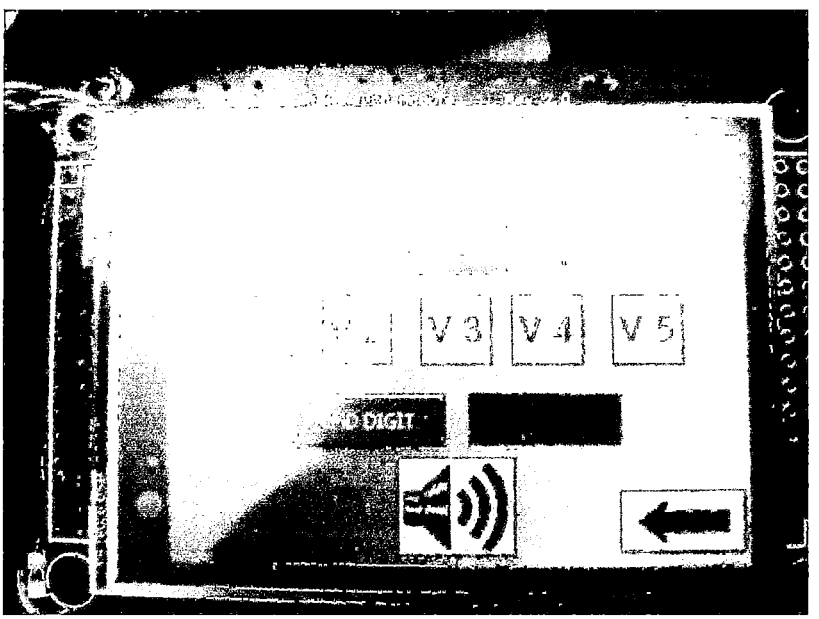


Fig.5.30.menu de velocidades

Porcentaje de la carga de la batería

Con esta opción podemos visualizar el nivel de carga de la batería, ya que es de mucha importancia para el usuario, para que de esta manera pueda tomar las previsiones pertinentes (fig.5.31).

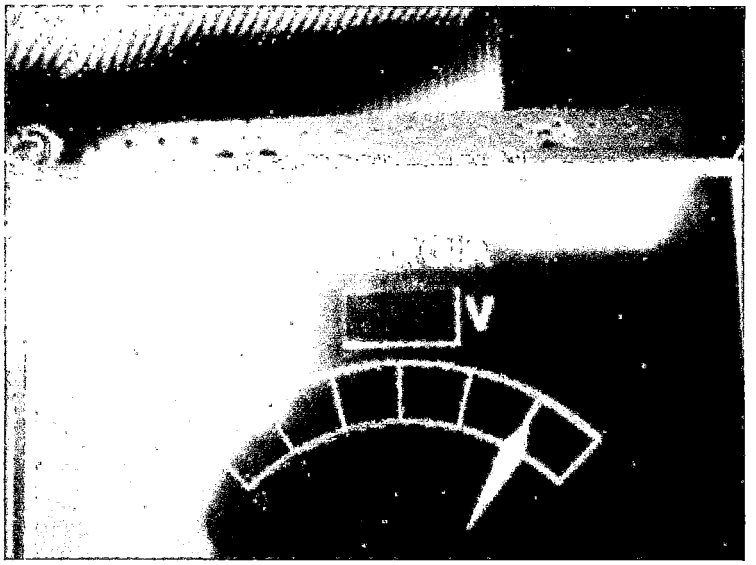


Fig.5.31.indicador de carga de la batería

Activando la comunicación inalámbrica: ingresamos en el botón de configuraciones, y activamos el bluetooth. De este modo quedaría lista la silla para la comunicación inalámbrica vía bluetooth por medio del Smartphone. (Figura 5.32).

Bluetoot activado (enable)

Bluetoot desactivado (disable)

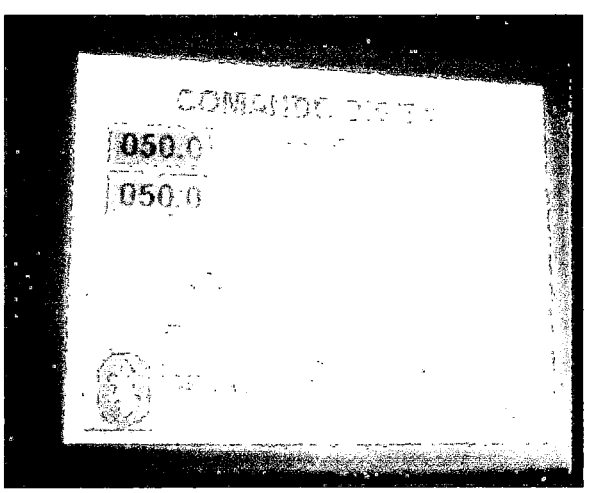


Figura 5.32: Bluetoot activado

Esquema general de los diferentes menús

Es un resumen de todas las opciones que tiene el usuario a su disposición, para el uso que crea pertinente (fig.5. 33).

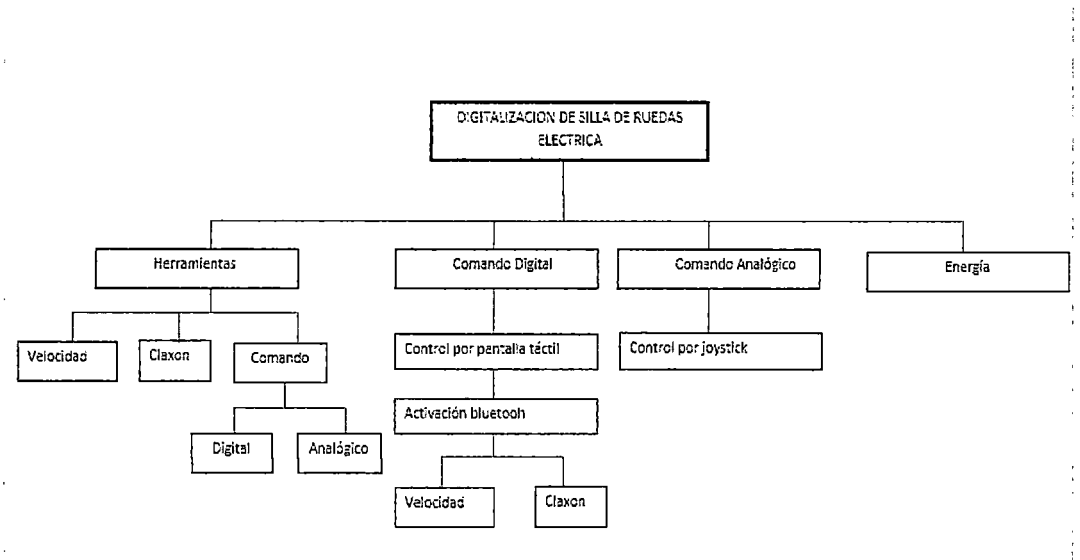


Fig.5.33. diagrama general de los diferentes menús

VI. RECUPERACIÓN DE LA INVERSIÓN

Para determinar y demostrar la factibilidad del proyecto de optimización se podría recuperar en un corto tiempo la inversión ya que lo que se tendría es un diseño de control para cualquier modelo de silla de ruedas eléctrica.

Tomando en cuenta el costo de reparación por cuenta de la marca de la silla en cuestión estamos hablando de casi el 50 % del costo total de la silla. Por lo consiguiente solamente tomando como ejemplo:

Costo de silla eléctrica estándar: en promedio s/12000.00

Reparación del sistema de control: en promedio s/6000.00

Por lo antes expuesto considerando solamente una silla a optimizar se recuperaría la inversión realizada en este proyecto.

6.1. Gastos totales del proyecto

COSTO TOTAL DEL PROYECTO	
DESCRIPCIÓN	**CANTIDAD**
Microcontrolador arm cortés m3	S/ 800.00
Maqueta de presentación	S/ 700.00
Silla eléctrica	S/ 750.00
Libros, revistas, folletos, etc.	S/ 250.00
Impresiones , Internet u otros	S/ 250.00
Entrevistas a especialistas (viajes, llamadas, etc.)	S/ 250.00
COSTO TOTAL	S/ 3000.00

VII. CONCLUSIONES

- Se logró diseñar un sistema de control y monitoreo, de tal forma, que se mejoró las etapas de un modelo de silla de ruedas motorizada para discapacitados.
- Se logró plantear un diseño para el sistema de control de una silla eléctrica, de forma general sin importar la marca y/o procedencia de ésta.
- Se puso en marcha la ejecución de un modelo final, obteniéndose muy buenos resultados.
- Se realizó la selección idónea de componentes, tales como motores, microcontrolador, etc. los cuales hicieron posible el buen desempeño de nuestra silla eléctrica.
- Se logró construir un sistema de control usando elementos de bajo costo.
- Se recopiló información sobre conceptos y criterios teóricos necesarios para el diseño y construcción del sistema de control.
- Despierta el interés docente en utilizar nuevas estrategias para controlar una silla eléctrica.
- Las funcionalidades que ofrece el prototipo de sistema de control de silla de ruedas eléctrica es tan competitivo como otras marcas en el mercado comercial.

VIII. RECOMENDACIONES

- Una mejora que se podría realizar en la parte de control, sería la comunicación por reconocimiento de voz, el cual le daría una interacción más completa a nuestra silla de ruedas con los usuarios.
- Se podría, incorporar una cámara a nuestra silla; para que de esta forma, pudiéramos hacer una monitorización más completa a distancia.
- El radio de cobertura para el control vía bluetooth es de 15 metros, es una distancia bastante amplia, sin embargo, si se deseara mayor cobertura, podría utilizarse bluetooths más potentes.
- Cambiar cada cierto tiempo la batería del equipo.
- Evitar derramar líquidos sobre el chasis que aloja al sistema de control.
- El control por joystick es súper sensible, por lo que se recomienda su manipulación con cuidado.

IX. REFERENCIAS BIBLIOGRAFICAS

BIBLIOGRAFÍA

- ALCOR MARTIN, Enrique. Estudio y desarrollo de interfaces hombre- máquina basados en sensores inalámbricos. España: Escuela Técnica Superior de Ingeniería, 2008.
- CHUNG LEE, Johnny. Projector-Based Location Discovery and Tracking. Estados Unidos: Carnegie Mellon University, 2008.
- CONDE RAMOS, David. Control accesible de videojuegos para personas con discapacidad. España: Universitat Politécnica de Catalunya, 2013.
- AXELSON, Jan. USB Mass Storage: Designing and Programming Devices and Embedded Hosts. Estados Unidos: Lakeview Research LLC, 2006.
- Molina, Rafael. Movimiento rígido de planos en el espacio. Proyección en el plano de la imagen. Depto. Ciencias de la Computación e I.A.
- Device Class Definition for Human Interface Devices (HID). 2001.

GLOSARIO

- **ACTUADOR:** Es el mecanismo que ejecuta la acción calculada por el controlador y que modifica las variables de control.
- **CONTROLADOR:** Utiliza los valores determinados por los sensores, calcula la acción que debe aplicarse para modificar las variables de control en base a cierta estrategia.
- **CONTROL PROPORCIONAL:** Acción de un dispositivo de control que modifica la actuación del elemento regulador proporcionalmente a la desviación entre la magnitud medida y el punto de consigna.
- **OFFSET:** También llamada desviación de cero, es el valor de la variable de salida cuando la variable de entrada es nula.
- **SETPOINT:** Set Point es cualquier punto de ajuste de alguna variable de un sistema de control automático. Puede ser: Nivel; presión, temperatura; desplazamiento; rotación; etc.
- **SENSOR:** Permite conocer los valores de las variables medidas del sistema.