UNIVERSIDAD VASCO DE QUIROGA

ESCUELA DE INGENIERÍA EN MECATRÓNICA

"Sistema de control personalizado y seguridad inteligente para automóviles a base de un sistema embebido con múltiples etapas y almacenado de información"

TESIS

Que para obtener el título de:

INGENIERO EN MECATRÓNICA

PRESENTA

Héctor Espinoza Heredia

ASESOR

Ing. Sergio Armando Galván Chávez

CLAVE: 16PSU02140

ACUERDO: LIC120614

MORELIA, MICHOACÁN FEBRERO, 2017

Resumen (Abstrac)

Sistema de control personalizado y seguridad inteligente para automóviles a base de un sistema embebido con múltiples etapas y almacenado de información.

El robo de autos es uno de los problemas que mayor impacto ha generado en México ya que los índices son cada vez mayores y la posibilidad de recuperarlos es casi nula. Una de las ventajas que brinda la tecnología, además de automatizar un proceso para reducir el margen de error en la producción, es brindar seguridad a la información, por lo que las grandes industrias hacen uso de sistemas tecnológicos capaces de controlar múltiples procesos mediante redes seguras donde sólo determinados trabajadores tienen acceso a esa información. Así como los automóviles, diversos aparatos electrónicos son controlados por sistemas embebidos que les ayuda a tener una respuesta mucho más rápida debido a que no contienen tanta información almacenada como los sistemas utilizados en la industria, la desventaja radica en que al no tener tanto espacio de almacenamiento no son capaces de realizar procesos tan complejos. Mediante una unidad de almacenamiento y un control de seguridad por huella digital se pretende adquirir las ventajas que brindan los equipos industriales a los sistemas embebidos ya existentes en los automóviles para tener un sistema de seguridad más amplio en cuestiones de antirrobo y el mantenimiento de nuestro vehículo, reuniendo gracias al sistema de almacenado más funciones controlables y poder generar una experiencia más placentera. Se seguirán teniendo las mismas variables independientes (microcontrolador maestro) y las variables dependientes (subsistemas controlables) de los sistemas embebidos presentes como la ventaja de rapidez en respuesta, pero se adicionará una unidad de almacenado que permitirá personalizar el vehículo con las características de diversos usuarios que estén autorizados para manipular el sistema.

Índice

Introducción

Un sistema embebido es una combinación de hardware y software computacional programado para la realización de una sola tarea específica. Dicho sistema puede estar compuesto de varios subsistemas embebidos, como es en los automóviles de nuevas generaciones, los cuales contienen dispositivos enfocados en funciones específicas tales como ABS, sensores de aceleración o velocidad, con o sin restricciones en el tiempo de procesamiento.

Los sistemas embebidos pueden encontrarse en un gran número de componentes y estar enfocados a muchos procesos de diversa índole, desde un MP3, cámaras fotográficas, videojuegos hasta el control de una industria completa. Esto no significa que sean muy simples, todo lo contrario, el diseño para tareas muy específicas significa que son tareas complejas por lo que es necesario separar los procesos en procesos independientes. Los sistemas embebidos cuenta con 2 categorías: los microcontroladores y los microprocesadores.

Un microprocesador es un circuito integrado, algunas veces denominado CPU, encargado del procesamiento y realización de operaciones matemáticas. Con el paso de los años estos han tenido una tendencia a integrar más núcleos, memorias y controladores de memoria dentro del mismo empaque que anteriormente iban montados sobre la placa base como dispositivos individuales.

Los microcontroladores son pequeños computadores en un circuito integrado a los cuales se les añade componentes extras (RAM, ROM, A/D, D/A, PWM, SPI, I^2C, RS-232) dependiendo de su necesidad, lo que reduce el tamaño y el costo. Los microcontroladores son usados en aplicaciones pequeñas, y productos o dispositivos automatizados, ya que brindan una versatilidad enorme a la hora de realizar funciones, pero a diferencia de los microprocesadores estos requieren la compra de dispositivos extras. (Valencia, 2010)

Antecedentes

El vehículo es una herramienta que se usa todos los días y por ende se debe brindarle la atención adecuada para su mantenimiento y protegerlo lo mejor posible. Esto a veces es difícil de determinar ya que los usuarios no son expertos en la materia, pero por medio de almacenado de fechas y lectura de sensores de estado se llega a conocer cada detalle, además de tener un sistema personalizado que cambie el uso de llaves por el uso de huellas digitales para hacerlo más seguro y simplifique las tareas a la hora de operar los ajustes del asiento, espejos, música, aire acondicionado, etc.

Los productos con una estructura de sistemas embebidos están orientados a minimizar los costos, tener un bajo consumo de energía, maximizar la confiabilidad, e incorporar las cuestiones de seguridad, con funciones y protocolos criptográficos para proteger la información en todas las fases del proceso. A diferencia de las computadoras un sistema embebido sólo tiene una memoria flash interna que guardara información durante un periodo muy breve de tiempo para ser manipulada con un programa muy pequeño. Tradicionalmente estos procesos se hacían a través de una terminal serie con muchas computadoras, pero con el paso del tiempo se tuvieron grandes avances en el monitoreo y control de gestión de procesos a distancia, hasta los sistemas embebidos que se tienen hoy en día en cualquier aparato eléctrico. (PAC, 2011)

Uno de los mayores aspectos de interés de los sistemas embebidos es que tienen una respuesta en tiempo real. Los microprocesadores utilizados para ello, al poseer una memoria flash y programas muy pequeños a diferencia de una computadora con un disco duro y miles de funciones, son ideales para procesos donde sus variables cambian rápidamente ya que las computadoras al cargar mucha información la mayoría de las veces no alcanzan a mostrar un valor cuando este valor ya ha sido reemplazado por una nueva medición. Una metodología de prototipado rápido de control, ayudará a que los sistemas que se han implementado sean totalmente funcionales y no dependan de librerías ni controladores que no sean de software libre. De este modo se tendrá una respuesta más rápida en tiempo real en comparación con sistemas operativos ya existentes en el mercado. (Valencia Puentes, 2010)

Con el desarrollo del internet y las redes, el curso de la innovación se ha inclinado por tener una conexión masiva de diversos elementos y que a estos se pueda acceder a distancia desde cualquier punto. Los sistemas embebidos son elementos que se encuentran en todo aparato electrónico y con el uso de internet la idea de implementar interfaces para acceder al control de los equipos vía TCP/IP resulta muy viable. Por medio de ethernet se es capaz de conectar diversos equipos a una red personal incluyendo los sistemas embebidos, que puedan interactuar entre ellos y sean controlados a distancia. La desventaja y área donde se sigue trabajando constantemente es el tema de la seguridad de información ya que al viajar por internet debe de ser de una manera segura. (Galiana Linares, 2005)

Dentro del área de las aeronaves no tripuladas (drones) se ha tenido un gran avance en cuanto a la manera de su manipulación a distancia, donde antiguamente era un elemento demasiado grande y era controlado por un piloto. Hoy en día su tamaño se ve bastante reducido y pueden ser controladas a distancia normalmente por radio frecuencias. Gracias al uso de tecnología avanzada es posible conocer su posicionamiento, tener el control de estabilidad y poder guiarlas. Los drones en general contienen 5 componentes que son: sensores de posicionamiento, controladores de velocidad, motores, microcontroladores y un elemento radio para su comunicación. Para el uso de drones orientados a una actividad específica se han implementado nuevos mecanismos a base de sistemas embebidos donde se especifica una tarea única a cada microcontrolador, teniendo así una respuesta de información más rápida. (Kharsansky, 2013)

Planteamiento del problema

Las estadísticas de robo de automóviles aumentaron de un 21% a un 35.6% en México entre los años 2017 al 2019, los estados de Puebla, el estado de México, Jalisco y Baja California fueron los que registraron el alza más importante de ellos. Estos crímenes se dan como robo parcial, total, con violencia o sin violencia, dependiendo de la hora y el lugar donde se cometa el ilícito con muy baja probabilidad de poder recuperar el automóvil inclusive para aquellos que se encuentran asegurados, debido a que son objetos muy fáciles de manipular en el mercado.

Los usuarios no necesitan tener otro instrumento más que la llave de encendido o en su defecto una réplica de ella. Estas circunstancias y el simple hecho de que los vehículos necesitan constantemente chequeos y servicios que los propietarios no están completamente calificados para determinar exactamente en qué momento se deberán realizar, genera que el comprar un auto no sea realmente una buena inversión, debido a que se generan más pérdidas que los beneficios que se otorgan. Por esta razón se busca generar un proyecto que pueda brindar mayor protección contra robos, conocimiento del estado vehicular y que sea capaz de agendar fechas para futuros mantenimientos. Partiendo de que un automóvil es una herramienta de uso cotidiano, se busca que este otorgue una experiencia placentera con tecnología innovadora y no un martirio en el recorrido del tráfico hacia su destino.

Objetivo general

Desarrollar un sistema embebido como prototipo montado sobre un vehículo tipo go-kart que manipule las funciones de interacción entre el usuario y el automóvil a través de una aplicación táctil y registro de huellas digitales de acceso.

Objetivos particulares

- Cambiar el antiguo mecanismo de encendido del vehículo sin la necesidad de una llave física por el uso de huellas digitales.
- Controlar y monitorear desde una pantalla touch todos los elementos a los que el usuario tenía acceso de gestión, eliminando la utilización de botones, manivelas y seguros manuales.
- Desarrollar un sistema para múltiples usuarios con una cuenta personalizada con almacenado de información y que anticipe los estados de riesgo para el automóvil.
- Diseñar un servicio personalizado para diferentes usuarios en el que se identifiquen características propias de cada uno sobre cuestiones de música, posición de los espejos, posición del asiento y aire acondicionado.

Justificación

Dado que los sistemas embebidos son mecanismos de procesos de muy alto control utilizados en la industria y pueden ser desarrollados a muy bajo costo, la implementación de ellos en equipos y actividades dentro de más procesos humanos ha estado en pleno auge en los últimos años. Con la cuarta revolución industrial y con una mayor competencia tecnológica en el mercado, para que un producto sea valorado, este debe cumplir más demandas y cubrir más necesidades de los usuarios, porque no es lo mismo tener un producto que sólo resuelva un problema a un producto que sea capaz de satisfacer varias necesidades.

Una gran ventaja de los sistemas embebidos es que son compatibles con múltiples plataformas y componentes electrónicos adversos a ellos, tienen gran rapidez por lo que son perfectos para su implementación cuando se requiere de una respuesta en tiempo real y un control de precisión con un margen casi nulo de error. Sin lugar a duda el uso de un sistema embebido para el conocimiento del estado del automóvil brindará una mayor seguridad al tener presente las condiciones en las que se encuentra el vehículo, además de poder generar una agenda interna de fechas en las que se requerirán los próximos servicios. La seguridad de un automóvil en cuestión de encendido será resuelta mediante un control por huella digital que sólo permitirá su operación por los usuarios autorizados, reduciendo así el riesgo a robos o el uso por usuarios no capacitados para hacerlo.

También ayudará al usuario a guardar información propia que pueda llegar a necesitar en algún momento para funciones establecidas de forma autónomas con acceso sencillo y rápido. Tal es el caso del control de la música en un vehículo, donde se identifican características de cada usuario registrado para una selección de música personalizada, además de una programación de elementos como son la posición de los asientos, nivel de ventilación, posición de los espejos, entre otras cosas. Todo este proceso también podrá ser manipulado desde una pantalla táctil que tiene acceso a las mismas funciones y permitirá modificar algunas cuestiones técnicas.

La implementación de este sistema es un proceso viable para los futuros automóviles ya que por una décima parte del costo total de un inmueble se puede tener una mayor comodidad y seguridad que a la larga traerá grandes beneficios. A su vez el mismo sistema puede ser desarrollado para otras áreas como casa/habitación, espacios públicos, oficinas, escuelas, bancos y muchos más lugares en los cuales las personas interactúen día con día con un enfoque muy similar al de este proyecto donde se desee tener una experiencia tanto más placentera como segura y con un control más preciso.

Hipótesis

Un sistema embebido que interactúe a través de una pantalla táctil de uso fácil en el automóvil permitirá una administración y automatización adecuada de cada aspecto controlable del vehículo para múltiples usuarios.

Metodología

Enfoque de la investigación

- Cualitativa: describe las características propias de cada usuario dentro de un mismo sistema que integra la información de todos sin riesgo de mezclarse. Un sistema de almacenado de información con acceso para varios microcontroladores que se guardará en diversas memorias independientes para la realización de tareas muy específicas e interacción dentro de una misma red.

Es una investigación tecnológica que busca reunir la velocidad de varios sistemas embebidos para controlar un vehículo eléctrico y brindar la capacidad de almacenado de información como los sistemas operativos dentro de una pequeña red.

Alcance

- Descriptivos: describe las propiedades de cada parte dependiente del sistema y como estos interactúan dentro de un sistema global que tiene acceso a un almacenado de información.
- Correlacionales: relaciona los cambios en el trabajo que se tiene del microcontrolador principal (identificación del usuario) con cada uno de los diferentes controladores dependientes de este y como ellos interactúan entre sí.

Se pretende llegar a un primer modelo montado sobre un vehículo tipo go-kart que cuente con mecanismos eléctrico-manuales previamente, agregándole un sistema inteligente que elimine sus antiguos mecanismos manuales para lograr mayor seguridad, comodidad e innovación, con la intención de tener un prototipo similar a lo que es el sistema de un automóvil comercial.

El diseño de la investigación

Experimental: con una documentación previamente investigada sobre como trabaja cada parte del sistema en forma independiente se buscará integrar todos los elementos en un sólo sistema que permita una interacción entre ellos con un control de gestión simple para cualquier usuario.

Universo (N) y muestra (n)

Microcontroladores de la marca ATMEL, PICS y PSOC, motores DC, motor de gasolina, memorias eeprom, detector de huellas digitales, pantalla touch marca nextion, sensores básicos, reproductor de música y unidades microSD.

Técnicas

Se hará uso de lenguaje en C para la programación de los microcontroladores debido a su universalidad y fácil uso de aprendizaje, así como protocolos de comunicación (I^2C, UART, SPI, entre otros.) para la integración de cada proceso en un sólo sistema.

Se utilizará conocimiento de electrónica para el diseño e impresión de las tarjetas electrónicas, la adecuación de sensores y activación de relevadores. También se utilizará conocimiento de programación para los dispositivos microcontroladores y conocimientos de diseño mecánico para algunos ajustes del prototipo vehicular tipo go-kart donde se montará el proyecto terminado.

Capítulo 1 Qué son los sistemas embebidos

En este capítulo se explicará un poco sobre lo que son los sistemas embebidos y sus diversas aplicaciones en el mundo real, que van desde simples tareas hasta el control automatizado de grandes industrias. Además, se diferenciarán los diversos elementos utilizados para la optimización de tareas y sus principales características.

1.1. Sistemas embebidos

Los sistemas embebidos son un conjunto de dispositivos que trabajan de manera asociada con la finalidad de minimizar los costos, tener una respuesta más rápida, mejorar la confiabilidad e incorporar las cuestiones de seguridad a los procesos. Un sistema embebido sólo tendrá una memoria que guardará información por periodos pequeños de tiempo y controlará el proceso con un pequeño programa, A diferencia de las computadoras que cuentan con todo un sistema operativo para realizar una multitud de tareas independientes, los sistemas embebidos son programados para una sola tarea específica, obteniendo así las grandes ventajas mencionadas anteriormente. (PAC, 2011)

1.2. Automatización industrial

La automatización industrial es la actividad donde se aplican diferentes tecnologías para controlar y monitorear un proceso, maquinarias, aparatos o dispositivos electrónicos para cumplir tareas o realizar acciones que se repetirán consecutivamente, haciendo procesos automáticos, sin tener la necesidad de la intervención humana. (Crespo, 2011)

1.3. Domótica

La tecnología aplicada al hogar, conocida como domótica se basa en la automatización, la informática y las nuevas tecnologías de comunicación para generar comodidad, seguridad y bienestar dentro de los hogares. Reúne diversas tareas que se realizan cotidianamente dentro del hogar en un mismo mando de control inteligente y brinda experiencias más placenteras a los habitantes de la casa. Los avances tecnológicos que se han experimentado han hecho posible el control de aspectos como son la iluminación, climatización, seguridad, comunicación, audio, vídeo, etc. Los arquitectos y constructores deben familiarizarse con este tipo de tecnologías rápidamente para así poder incorporarlas en sus proyectos y poder tener competitividad laboral. (FENERCOM, 2007)

1.4. Microprocesadores

El microprocesador es el cerebro central de toda computadora, en él se almacenan las instrucciones a seguir para cada proceso y mantiene un orden estructurado de la toma de decisiones. Los microprocesadores son la pieza más importante de una computadora, su capacidad de almacenamiento de información es demasiada pequeña para sólo soportar el código de instrucciones. Su función es brindar instrucciones al sistema para buscar información en otras unidades de almacenamiento en lugar de almacenarla dentro de sí mismo. (Parra Reynada, 2012)

1.5. Microcontroladores

El microcontrolador es un integrado muy similar en funcionamiento al microprocesador, pero a diferencia del microprocesador, cuenta con más funciones que elimina la dependencia de elementos externos para su utilización. Sin embargo, no cuenta con gran capacidad en su espacio de almacenamiento por lo que para funciones muy complejas seguirá dependiendo de unidades externas del mismo modo que el microprocesador lo hace. Algunos de los elementos que diferencían uno de otro, es que el microcontrolador ya cuenta con elementos internos como son: microprocesador, convertidor A/D, temporizadores, pequeña memoria RAM, pequeña memoria ROM, puertos de comunicación serial y osciladores. (Parra Reynada, 2012)

1.6. Tarjetas programables

Según la página especializada en desarrollo y venta de tecnología óptica para procesos industriales lightpath.com, existe una gran diversidad de tarjetas programables en el mercado con muchas características diferentes por lo que no se pueden establecer normas específicas en general. Una tarjeta programable es una tarjeta electrónica prefabricada que se usa para el control de procesos y que ya incorpora un programador interno. La tarjeta maneja su propio lenguaje y en la mayoría de los casos se requiere una aplicación en la PC para poder hacer uso de su código programable. Esto brinda varias ventajas ya que al tener un lenguaje preestablecido se cuenta con librerías que facilitan la sintaxis del código, a diferencia de los microcontroladores que necesitan generar esas librerías con procesos más complejos para las mismas operaciones. (Quispe, 2017)

1.7. PLC

Según la documentación contenida en la página oficial de SIEMENS, empresa dedicada al desarrollo de dispositivos programables, un PLC es un dispositivo de control utilizado en la industria para la automatización de procesos que ya cuenta con los periféricos y protecciones eléctricas requeridas en su placa interna. Las ventajas que otorga un PLC es que permite una programación en código y también una programación visual denominada en "escalera". Debido a que en la industria se trabaja con motores y sensores, su programación se basa en la activación y desactivación de contactores dependientes de variables. El PLC tiene un costo elevado en comparación con otros dispositivos pero se justifica con la calidad del producto, cuando se trabaja con motores de grandes voltajes, estos generan ruido eléctrico que altera las señales eléctricas dentro de los circuitos. El PLC tiene la capacidad de aislar ese ruido externo de todo el sistema. (Siemens , 2017)

1.8. Raspberry

La página oficial raspberry.org menciona que la raspberry es una tarjeta electrónica que se encuentra clasificada entre las tarjetas programables porque se puede usar para el control de procesos y una PC porque se puede cargar con un sistema operativo como cualquier otra computadora. Lo que la diferencia de las demás tarjetas programables es que tiene la capacidad de funcionar como una computadora normal y se diferencia de las PC porque no cuenta con la misma capacidad ni tamaño. En una descripción del foro su comunidad online la nombró como una minicomputadora de bajo costo y con acceso a pocas funciones computacionales. Las ventajas que otorga es que su costo es muy económico, funciona como una PC, es mucho más pequeña que un CPU y brinda funciones de control programable. (Raspberry pi foundation, 2017)

1.9. PC y CPU

Un computador o computadora (PC o CPU) es una máquina capaz de almacenar información y realizar funciones especializadas de acuerdo con una lista de instrucciones programadas. A la lista de instrucciones se le conoce como programa y el medio de almacenamiento interno como memoria del computador. El computador cuenta con un gran número de elementos internos como son el microcontrolador, memoria RAM, disco duro, tarjeta gráfica, parlantes, monitor, teclado, ratón, puertos USB y lector de CD. Todos ellos trabajan en conjunto para satisfacer las necesidades de una buena interacción entre un usuario y una máquina a través de una pantalla.

Además, las computadoras tienen la capacidad de programar otros controladores por medio de sus puertos USB y permiten al usuario generar instrucciones basadas en protocolos ya establecidos. (Vásquez Gómez, 2012)

Capítulo 2 Unidades de almacenamiento

Este capítulo tratará sobre los elementos secundarios para todo sistema de control. Como se sabe no sólo es necesario que un microcontrolador sea programado para realizar una tarea, pues además de este, en cualquier proceso de gran complejidad o gran procesamiento de información, es necesario tener unidades de almacenamiento y estas a su vez se dividen en varios tipos que funcionan en conjunto para facilitar el tratamiento de datos y ayudar a tener rapidez de procesamiento.

1.1. Memoria RAM

Comúnmente llamada memoria de acceso aleatorio (Random Accesss Memory: RAM). Esta memoria de tipo escritura/lectura tiene como característica guardar información de forma temporal, hasta que un nuevo programa sea cargado o se apague todo el sistema. Optimiza el funcionamiento de las computadoras ya que no tienen que buscar en toda una unidad de almacenamiento cada vez que necesiten acceder a cierta información. Su funcionalidad se da cuando se requiere interactuar con cierta información que puede cambiar y no se guardará de forma permanente en un patrón designado, tal es el caso de los mensajes telefónicos. El patrón viene siendo la estructura de interfaz del mensaje, cuando se accede a él será guardado de forma temporal en la memoria RAM para que el usuario llene el contenido, sea enviado y sea borrado de la memoria RAM. De esta manera, el patrón para un nuevo mensaje podrá ser llamado en futuras ocasiones sin mantener el texto del mensaje enviado anteriormente. Cabe señalar que los mensajes enviados pueden ser almacenados, pero en otro tipo de memoria. (Vásquez Gómez, 2012)

1.2. Memoria ROM

Se llama memoria de sólo lectura (Read-Only Memory: ROM). En esta memoria se guardan las primeras instrucciones que tendrá una computadora al momento de conectarse a la energía eléctrica, en ella se encuentran los sistemas operativos, los códigos, las interfaces gráficas, los archivos, los programas a utilizar, las contraseñas y toda la información de respaldo del usuario.

Este tipo de memoria almacena la mayor parte de la información requerida por un sistema y su búsqueda en ella muchas veces es muy lenta debido a que se necesita buscar entre todo el contenido almacenado. Es por ello que se requiere trabajar también con una memoria RAM que guardará la información encontrada de manera temporal para agilizar el proceso de interacción con el usuario. Cabe señalar que también existen memorias ROM de tipo lectura/escritura que son ideales para los sistemas embebidos, pero inicialmente estas eran de sólo tipo lectura. (Vásquez Gómez, 2012)

1.3. Memoria FLASH

Las unidades FLASH funcionan de manera muy similar a las memorias ROM ya que en ellas se encuentra gran contenido que será utilizado por el usuario como son las aplicaciones, los programas, los sistemas operativos, etc. Este tipo de unidades pueden ser de tipo interno como los discos duros o de tipo externo como las unidades USB. Su ventaja principal es que son unidades de mucha mayor capacidad en comparación con algunas memorias ROM portátiles y también que su contenido (mensajes, documentos, agendas y todo tipo de información interna) puede editarse de manera sencilla y manual por el usuario desde una computadora. Para los sistemas embebidos este tipo de unidades no son necesarias, debido a que el sistema embebido busca realizar tareas muy específicas en las que no se requiere que el usuario genere algún documento extra de interés personal, a menos de que el sistema embebido requiera de una búsqueda externa en alguna unidad para su interacción como es en el caso de los reproductores multimedia con unidades USB. (Vásquez Gómez, 2012)

Capítulo 3 Sistemas de seguridad

Para todo sistema personalizado que guarda información importante de los usuarios, es necesario que se incluya un sistema de seguridad que sólo permita el acceso a usuarios autorizados. Al igual que las cuentas bancarias, cuentas de correo o facebook, desbloqueos de celulares, entre otros, es importante tener un sistema de acceso de seguridad por contraseña o algún otro método de identificación para cualquier unidad que resguarde datos personales y en este capítulo se mencionará sólo algunos de los métodos más utilizados.

1.1. Contraseñas de usuario

La protección de información es algo muy importante y un tema muy delicado ya que se trabaja con contenido privado de diversos usuarios que requieren de absoluta protección y exclusividad de acceso. Las formas más usuales de generar contraseñas son mediante el uso de claves que sólo conoce el usuario propietario de la información. Como recomendación el Instituto Nacional de Tecnologías de la Comunicación menciona que para que una contraseña sea segura debe tener al menos 8 caracteres como mínimo y que estos incluyan letras, números y signos, sin embargo algunas aplicaciones han optado por cambiar este método por uso de huellas digitales ya que se han tenido un índice elevado de fraudes por páginas de internet que piden la revelación de contraseñas. (Instituto Nacional de Tecnologías de la Comunicación, 2013)

1.2. Lector de huella digital

Los lectores de huellas digitales son también instrumentos de protección de datos por contraseña. Son muy utilizados en los trabajos para brindar acceso sólo a ciertas personas en particular donde se digitalizan las imágenes tomadas de sus huellas por un sensor óptico, para convertirlas en valores binarios: “0” (espacios en blanco) y “1” (espacio con una línea). De esta manera se puede convertir la imagen de una huella digital a código entendido por computadoras y dado que todas las huellas digitales son diferentes, es muy difícil que estas sean copiadas. Sin embargo cabe resaltar que la capacidad de generar una alta protección dependerá de la calidad de los componentes que se adquieran ya que algunos sólo toman una pequeña parte de la huella para su identificación. (Guadarrama Fortoul, 2012)

1.3. Escaneo de retina

Este proceso convierte un mapa de rasgos físicos en código binario entendido por la computadora y hace uso de reflexiones de luz emitida sobre la retina para generar una copia de ella con un procesamiento de tratado de imágenes por diversos filtros. Este tipo de lectores se basa en la detección de los vasos sanguíneos oculares para su conversión. Brinda una mayor seguridad en comparación con otros métodos ya que es más complicado crear una copia ocular que una copia de cualquier otra parte del cuerpo. El desarrollo de tecnologías ha permitido que también sea posible detectar a través de la retina miles de enfermedades como es el caso del SIDA. Es un método superior a los sistemas más precisos de huellas digitales. (R. Ganorkar, 2007)

Capítulo 4 Programación de control

Finalmente se hablará de las diversas formas de realizar una automatización de tareas. Este método dependerá del tipo de unidad de control que se elija, algunos pueden ser muy sencillos o pueden brincarse algunos pasos, pero este capítulo se apegará al método que se requerirá para el desarrollo de este proyecto. También se dará una breve explicación de las formas más utilizadas para la programación de sistemas embebidos.

1.1. Programación de microcontroladores

La programación de cualquier equipo de control se da por medio de una computadora que contiene el programa específico y maneja los protocolos de comunicación permitidos entre el hardware y software. Existen controladores que tienen un propio código de programación y simplifican la creación de un procedimiento en función de una tarea a realizar, pero por otro lado existen otros microprocesadores que no cuentan con programas personalizados para su programación pero si son compatibles con algún tipo de lenguaje estándar como son C++ o ensamblador. El usar este tipo de lenguajes requiere de mayores habilidades y conocimiento a la hora de programar, pero brindan la ventaja de eliminar muchos elementos innecesarios en código al momento de importar librerías. (Arenas Mas, 2008)

1.2. Acceso a memorias

Como un microprocesador está diseñado para tener sólo instrucciones de procedimientos, es indispensable que en la automatización de procesos se tengan unidades de almacenamiento (memorias) donde el microprocesador pueda buscar información de acuerdo a los requerimientos. Una ventaja de las memorias es que pueden almacenar diferentes valores para un mismo proceso en múltiples localidades donde el microcontrolador buscará en la localidad indicada los valores de las variables dependiendo de los requerimientos de la situación. Este tipo de comunicación se da con el uso de ciertos protocolos. (Vásquez Gómez, 2012)

1.3. Integración de múltiples unidades

Diversos protocolos de comunicación cuentan con comunicación entre un dispositivo maestro y múltiples esclavos. El protocolo I^2C permite tener varias unidades de almacenamiento donde el maestro accede a ellas por un mismo canal de comunicación, sin mezclar información entre dispositivos o dañar las unidades de almacenamiento. (Salas Arriarán, 2015)

1.4. Buses y puertos

1.4.1. Buses

El término Bus se refiere a la ruta de comunicación entre los componentes de una computadora para el envío y la recepción de datos, ya sea para componentes dentro del sistema o unidades externas que se conectan a los puertos del computador. (Vásquez Gómez, 2012)

1.4.1.1. Bus asíncrono

La forma de transmisión de señales asíncrona se da mediante protocolos de comunicación que no dependen de una señal de reloj. Este tipo de comunicación es entre 2 dispositivos que envían y reciben información, al no depender de una señal de reloj sólo requieren conectar un canal de envío de datos y otro de recepción como es el caso del protocolo de comunicación UART. (Salas Arriarán, 2015)

1.4.1.2. Bus síncrono

La forma de transmisión de señales síncrona se da mediante protocolos de comunicación que dependen del uso de una señal de reloj además de los puertos de envío y recepción de datos entre dispositivos esclavo-maestro. Algunos protocolos de comunicación como el SPI requieren de una salida de reloj, una salida de datos y una entrada de datos, mientras que otros protocolos como el I^2C sólo necesitan una salida de reloj y un único canal compartido para la transmisión y recepción. Estos tipos de protocolos de comunicación son muy útiles cuando se tienen múltiples dispositivos esclavos que son controlados por un dispositivo maestro. (Salas Arriarán, 2015)

1.4.2. Puertos

Entradas o salidas que poseen los componentes electrónicos para mandar y recibir información hacia otros dispositivos. Los puertos pueden estar conformados por un sólo pin de salida o por un conjunto de pines para mandar la información por separado y de manera simultánea. (Mayné, 2009)

1.4.2.1. Puerto serie

El puerto serie se caracteriza por mandar información a través de un único bus, bit por bit, hasta completar el envío total de datos. Este tipo de proceso es más lento que la comunicación paralela y requiere mandar valores de inicio, el mensaje y final de la comunicación, en ese orden por el mismo bus. (Mayné, 2009)

1.4.2.2. Puerto paralelo

El puerto paralelo manda información a través de múltiples buses. Permite separar los datos entre varios pines de salida para enviar un mensaje repartido en todos los buses al mismo tiempo. Es una transferencia más rápida que la comunicación serie. (Mayné, 2009)

1.5. USB

El USB es un bus punto a punto: dado que el lugar de partida es el host (dispositivo maestro) y el destino final un periférico. El protocolo USB se basa en el "paso de testigo", donde los periféricos comparten la banda de paso del USB proporcionada por el testigo (host) al periférico seleccionado y este devuelve al testigo su respuesta sin la necesidad de apagar el equipo. Cada computadora trae varios puertos USB donde se conectan dispositivos periféricos (unidades de almacenamiento). Este bus parte del protocolo UART mejor conocido como RS-232. (Lopéz Pérez, Protocolo USB (UNIVERSAL SERIAL BUS), 2008)

1.6. SPI

SPI es un protocolo de bus de 4 líneas que va de 10 Mbps a 20 Mbps, sobre el cual se transmiten paquetes de información de 8 bits en distancias reducidas. Los dispositivos conectados al bus pueden trabajar como transmisor y receptor al mismo tiempo, por lo que la comunicación serial es de tipo full dúplex. 2 de estas líneas trasfieren los datos (una en cada dirección). La comunicación de maestro a esclavo es por el canal "MOSI" y la comunicación de esclavo a maestro es por el canal "MISO", la tercer línea es la del reloj (SCL) y la cuarta activa la lectura o escritura. Se puede tener varios dispositivos esclavos conectados a un maestro. El maestro es quien inicia la transferencia de información sobre el bus y genera las señales de reloj. (Lopéz Pérez, Protocolo SPI (Serial Peripherical Interface), 2008)

1.7. UART

El protocolo UART es un protocolo de comunicación que va de 230 Kbps a 460 Kbps y 15 metros de alcance. Se utiliza en el puerto RS-232 de las computadoras para la conexión entre 2 equipos. Este protocolo hace uso de 2 buses de comunicación, “TX” para la transmisión y “RX” para la recepción. Se pueden conectar únicamente 2 dispositivos que envíen y reciban datos a través de estos puertos, donde el “TX” del primero envía datos al “RX” del segundo sin requerir una señal de reloj y viceversa. (Valencia González, Berríos Miranda, & Carreño Bonilla, 2012)

1.8. I^2C

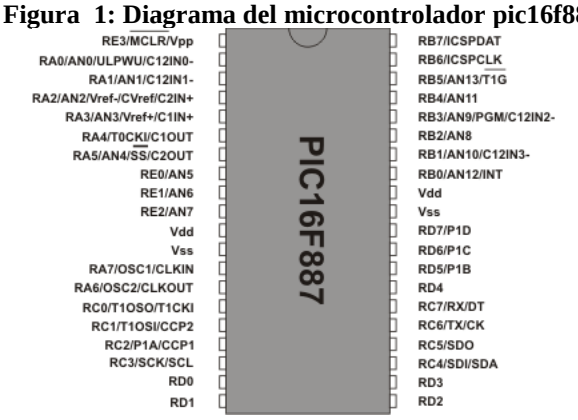
El protocolo I²C es un bus bidireccional que utiliza 2 líneas para transferencias de 3.4 Mbps en distancias reducidas. Ambas líneas requieren resistencias de polarización a positivo (RPA). “SCL” es la línea de reloj y se utiliza para sincronizar todos los datos de transferencia durante la transmisión (SDA es la línea de datos). Las líneas “SCL” y “SDA” están conectadas a todos los dispositivos que pueden ser maestros o esclavos en el I²C bus. El maestro es siempre el dispositivo que maneja la línea de reloj “SCL” y los esclavos son los dispositivos que responden al maestro. Un esclavo no puede iniciar una transferencia a través del I²C bus ya que sólo un maestro puede hacer esa función. Generalmente son varios esclavos en el I²C bus y un sólo maestro. Tanto el maestro, como el esclavo puede transferir datos a través del I²C bus, pero la transferencia siempre es controlada por el maestro. (Garcia, 2012)

Capítulo 5 Revisión técnica

1.1. Microcontroladores

El microcontrolador pic16f887 (figura 5) es ideal para el desarrollo del prototipo porque estos modelos brindan algunas características que los hacen superiores a otros para el proyecto, ya que cuentan con la posibilidad de ser programados mediante software libre por el desarrollador MPLAB IDE (figura 6) y el programador PICkit2. Además cuentan con un número de 35 pines de salida o entrada que son más que suficientes a diferencia de microcontroladores de 18 pines e incorporan un oscilador interno de gran estabilidad.

Figura 1: Diagrama del microcontrolador pic16f887



Fuente: imagen tomada de la hoja de datos del microcontrolador pic16f887.

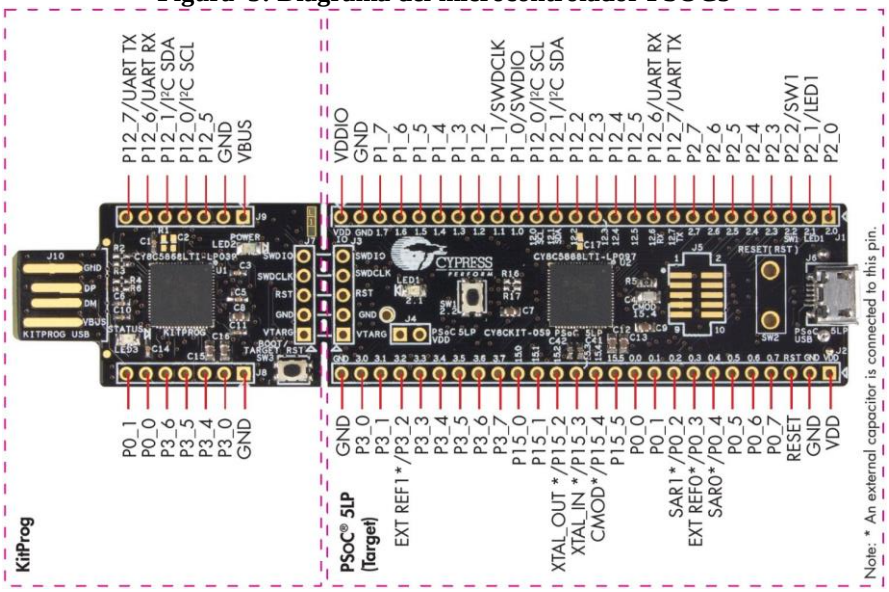
El pic16f887 tiene una arquitectura RISC de 8 bits, con una frecuencia de operación de 0-20 MHz. Su oscilador interno es de alta precisión y brinda lo opción de un oscilador externo de fácil implementación en código. Tiene el modo ahorro de energía en suspensión y la opción de programación serial incorporada en el circuito. Cuenta con 8 KB de memoria ROM, 256 bytes de memoria eeprom, 368 bytes de memoria RAM, convertidor A/D de 14 canales a resolución de 10 bits, 3 temporizadores, módulos PWM, UART, SPI e I^2C, entradas análogas y digitales.

Figura 2: Logo del programa MPLAB IDE

Fuente: imagen tomada de la página oficial de descarga MPLAB.

También es necesario el uso de microcontroladores PSOC5 (figura 7) ya que brindan una velocidad más rápida de procesamiento para la lectura de los sensores, tienen una mayor precisión en las mediciones, mayor capacidad de almacenamiento y tienen una interfaz programable más sencilla de utilizar.

Figura 3: Diagrama del microcontrolador PSOC5



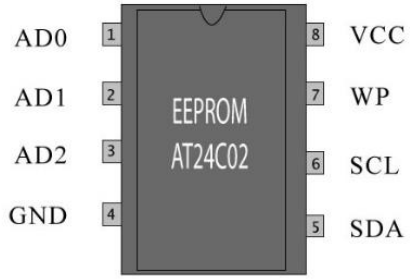
Fuente: imagen tomada de la hoja de datos del microcontrolador POSC5.

Además de tener una programación combinada en código y a bloques que hace el proceso mucho más sencillo, el PSOC5 cuenta con una arquitectura ARM CORTEX-M3 que es compatible con código y librerías de arduino. Un microcontrolador de 32 bits con un oscilador interno de alta precisión (difícilmente requerirá uno externo) que alcanza frecuencias de 80 MHz con una capacidad de 256 KB de memoria ROM, 64 KB de memoria RAM y 2KB de memoria eeprom. Brinda el uso de 48 pines de entrada/salida, convertidor A/D de 62 canales a resolución de 12 bits, un convertidor delta-sigma de 16 bits, 3 temporizadores independientes, multiplexores, amplificadores internos, comparadores, voltajes de referencia, módulo PWM, UART, SPI, I^2C, entradas análogas y digitales.

1.2. Memoria EEPROM

La memoria eeprom (figura 8) permite almacenar las posiciones de los asientos y espejos por intervalos de tiempo, con la intención de tener en cada memoria la información de cada usuario con su próxima fecha de servicio. De esta manera mediante el protocolo I^2C y un mismo método de búsqueda para todas las memorias, se puede obtener la información requerida por cada usuario con la posibilidad de realizar cambios existentes.

Figura 4: Diagrama de memoria eeprom AT24C02



Fuente: imagen tomada de la hoja de datos de memoria eeprom AT24C02.

La memoria eeprom "AT24CO2" cuenta con un mayor número de direcciones de almacenamiento (8 direcciones) a diferencia de las siguientes matrículas "C04", "C08" y "C016". Aunque su capacidad de almacenamiento se reduce a 256 bits, es más que suficiente para almacenar las posiciones de los motores expresadas en valores numéricos.

1.3. Lectores de huellas digitales

La detección de usuarios por huellas digitales se realiza a través del lector "R308" (figura 9) de la marca OPEN-SMART, que tiene compatibilidad con cualquier controlador y se comunica por medio del protocolo UART. Hace uso de una librería cuando se utiliza arduino, pero como el sensor se comunica por medio de los puertos "TX" y "RX", es posible adaptarlo a cualquier otro microcontrolador que admita el protocolo UART. El lector soporta hasta 255 huellas diferentes.

Figura 5: Lector de huellas digitales R308 de la marca OPEN-SMART

Fuente: imagen tomada de la hoja de datos del lector R308.

Para su conexión hace uso de 4 cables.

- Vin: alimentación a 5V para los TTL.
- TX: canal de comunicación (transmisión).
- RX: canal de comunicación (recepción).
- GND: tierra.

Este sensor realiza la conexión por medio de los canales “RX” y “TX” con el pic16f887. Sin embargo este tipo de lector sólo permite el uso de un único dispositivo ya que no se puede modificar la dirección de acceso a él, por lo que se ocasionarían errores de transmisión en caso de realizarlo.

1.4. Pantalla touch TFT

Para la manipulación del sistema por el conductor se busca la manera de hacerlo a través de una pantalla touch a color que permita controlar todos los elementos. Para ello, la pantalla touch TFT de la marca nextion de 7 pulgadas (figura 10), mediante el protocolo de comunicación UART permite tener conexión con un microcontrolador (en este caso un PSOC5) que mandará los datos recolectados por los sensores para ser mostrados en pantalla con una interfaz más amigable con el operador mediante dibujos, gráficas y botones. Esta pantalla manda letra por letra, número por número en cierto orden específico.

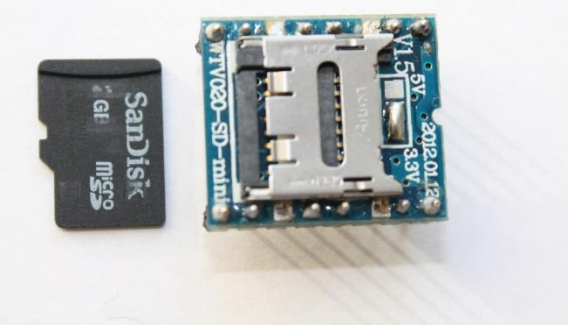
Figura 6: Pantalla nextion tft de 7 pulgadas

Fuente: imagen tomada de vendedores en mercado libre.

1.5. Reproductor de música

Dado que los vehículos ya contienen un estéreo instalado, en lugar de cambiarlo se usa un módulo que reproduzca música de una unidad externa de almacenamiento. El módulo “WTV020SD” (figura 11) permite reproducir la música dentro de una microSD, controlada por otro microcontrolador mediante el protocolo UART. Para ello y de igual manera que en el caso de los asientos, se manda a un microcontrolador PSOC5 el número de usuario que accedió al sistema y con ello se puede decidir cual módulo de memoria reproducir (existe un módulo para cada usuario). El módulo es controlador por el PSOC5 a través de comandos específicos y los buses de comunicación “RX” y “TX”.

Figura 7: Módulo reproductor de microSD WTV020SD



Fuente: imagen tomada de vendedores en mercado libre.

1.6. Protocolo UART

El protocolo USB hace uso del puerto RS-232 en las computadoras y por lo tanto del protocolo UART. Este protocolo contiene 4 cables de comunicación: alimentación, tierra, "RX", "TX" y es posible la comunicación entre 2 dispositivos, con una velocidad máxima de 230 Kbps a 460 Kbps asíncrona y hasta 15 metros de distancia.

Como es en el caso del lector de huellas digitales, la pantalla nextion y el módulo reproductor de música, se hace uso del protocolo UART para su comunicación con un microcontrolador superior, el cual gestiona cadenas de texto para mandar o recoger valores.

1.7. Protocolo de comunicación I^2C

El protocolo I^2C es el más indicado para la comunicación entre microcontroladores y memorias ya que por 2 canales "SDA" y "CLK" se puede acceder a múltiples dispositivos sin la necesidad de utilizar gran número de pines del microcontrolador a una velocidad máxima de 3.4 Mbps síncrona, pero con una distancia limitada al tamaño de la misma placa. El bus "CLK" es el bus de reloj que determina mediante pulsos altos y bajos a diferentes frecuencias, cuando inicia la comunicación, tipo de lectura o escritura, si hay más peticiones o es la última y la detención de comunicación para liberar el bus. El bus "SDA" es el bus de envió de datos y en él se transmite la información necesaria ya sea del microcotrolador a las unidades o de las unidades al microcontrolador. Este bus trabaja a la par con el bus "CLK" para determinar las actividades por pulsos altos y bajos.

Dentro del manual del programa MPLAB y CYPRESS se explican todas las funciones y el código que se requiere para la programación de cada microcontrolador, así como la estructura que debe seguirse para su correcto funcionamiento. Mediante la programación, una vez determinado el número de usuario que desea ingresar al sistema, el microcontrolador maestro accederá a la dirección de la memoria correcta que se haya designado con el mismo número de usuario, de esta manera se obtendrán todos los valores de las posiciones de los espejos, el asiento, fecha de servicio y se tendrá comunicación con el dispositivo esclavo que controla al módulo de música. La programación debe tener un método para realizar consultas de información y otro método para editar valores almacenados.

1.8. Convertidor analógico/digital (AVCC)

Algunos sensores son medidos mediante el conteo del número de pulsos en alto ingresados en una entrada digital durante un periodo de tiempo, mientras que otros son medidos por señales analógicas que varían el voltaje de entrega en un rango de 0V a 5V. Una vez ingresado el voltaje al microcontrolador el convertidor analógico/digital permite cambiar el rango eléctrico a un rango en bits (en el caso del PSOC5 va desde 0 hasta 65535) para poder gestionar su información internamente.

Se debe tener en cuenta que el rango eléctrico en las entradas del microcontrolador no debe sobrepasar su voltaje de alimentación (5V) por lo que lo sensores deben tener el mismo voltaje y de no ser así, debe existir un divisor de voltaje que regule el flujo máximo de entrada en 5V. Para múltiples sensores el convertir analógico/digital del PSOC5 cuenta con 62 canales multiplexados. Las variables a medir consideradas de mayor importancia son las siguientes.

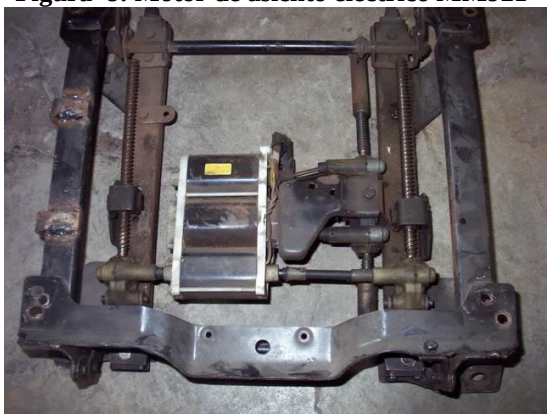
- Nivel de gasolina por división de voltaje
- Temperatura por termistor
- Velocidad por magnetismo
- Distancia por ultrasónico
- Capacidad de batería por divisor de voltaje
- Posición de asientos por sensor inductivo
- Posición de espejos por divisor de voltaje y magnetismo

1.9. Motores de asientos

Los motores a utilizar en los asientos son de alto torque porque requieren la capacidad de mover el asiento con una persona de peso variable montada sobre él. Se optó por trabajar con motores DC en lugar de servomotores, aunque estos últimos son más precisos porque se controlan por grados, siempre deben estar energizados, lo que ocasionaría un consumo innecesario de energía.
Estos motores ya están estandarizados para su uso en automóviles dentro del mercado, si se llegan a energizar, sus engranes de metal son capaces de mantener la posición en la que se encuentran aunque se les aplique una fuerza manual para forzar su movimiento.

Debido a dichas características estos motores se adaptan fácilmente a un diseño mecánico sobre rieles y engranajes, que se mueven desde una posición inicial por pulsos de tiempo a la posición deseada por el usuario. El prototipo cuenta con motores de un golf 2016 que trabajan con transmisión de movimiento por engranes y un tornillo sin fin (figura 12). Se pueden adquirir a muy buen costo en cualquier deshuesadero de autos.

Figura 8: Motor de asiento eléctrico MM921



Fuente: fotografía tomada por el autor.

1.10. Motores de espejos

Al igual que los motores de los asientos, estos son de tipo DC y engranaje metálico (2 por cada espejo), pero con mucho menor torque ya que no requieren mover demasiado peso. Se tiene un motor para el movimiento izquierda-derecha y otro para el movimiento arriba-debajo de cada espejo. Estos motores son controlados por pulsos y detección magnética en un anillo de varios interruptores magnéticos, los cuales mandan una señal a las entradas digitales de un microcontrolador y de esta manera es posible determinar las posiciones de los espejos (figura 13).

Figura 9: Estructura de los retrovisores con motores de 21 kg-cm y 200rpm

Fuente: fotografía tomada por el autor.

1.11. Go-kart con motor de gasolina 6.5Hp y encendido eléctrico

El motor Oakland (figura 14) a gasolina es ideal para el uso en el prototipo ya que contiene el mecanismo de encendido eléctrico prefabricado, una potencia de 6.5 hp, flecha de ¾ in, un tanque de 5 litros y un peso de 15.5 kg. Tiene un encendido manual a bobina y un encendido eléctrico por batería de 7 A/HR (capacidad de batería especificada por el proveedor).

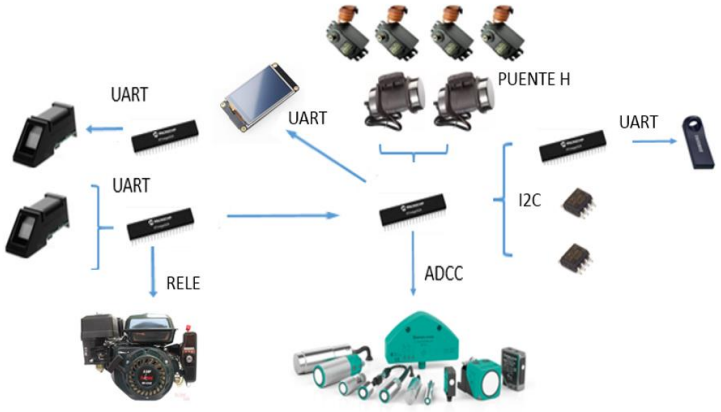
Figura 10: Motor de gasolina 6.5hp con encendido eléctrico marca Oakland

Fuente: imagen tomada de vendedores en mercado libre.

1.12. Estructura del sistema

Finalmente todo el sistema se comunica con dependencia interna de cada elemento hacia los demás y una comunicación global de los sistemas embebidos que utilizan diversos protocolos de comunicación. Se puede observar la estructura del sistema en la siguiente imagen (figura 15) con todos los componentes a utilizar para el proceso de comunicación.

Figura 11: Diagrama de la estructura de comunicación del proyecto

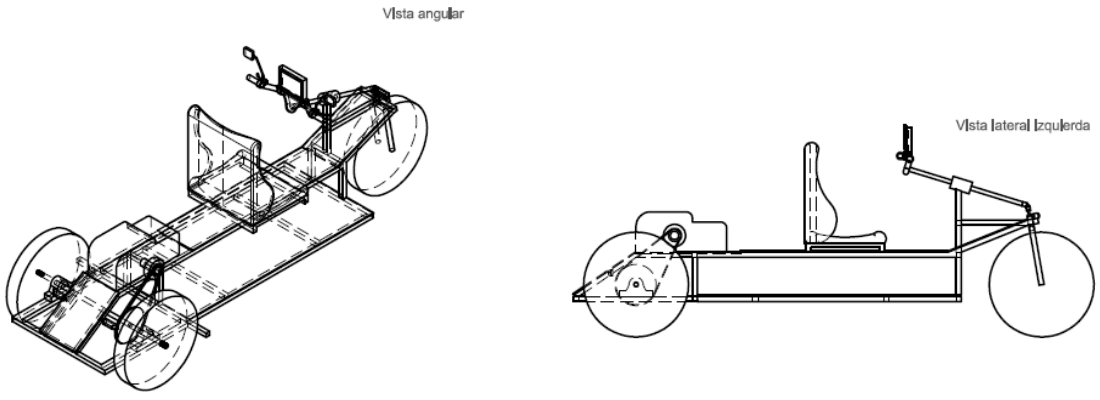


Fuente: diagrama hecho en power point por el autor.

Capítulo 6 Diseño mecánico

Para realizar las pruebas correspondientes del sistema de seguridad y verificar su correcto funcionamiento, este debe ser instalar en un vehículo móvil impulsado por un motor a gasolina que cuente con características similares a un automóvil real. Dado que el costo de compra y el margen de daños posibles a un auto es muy elevado, se ha utilizado como modelo de prueba para el sistema inteligente un pequeño vehículo go-kart de 3 ruedas (figura 16) que pueda ser manipulado en las calles y cuente con los elementos esenciales de un automóvil real.

Figura 12: Planos del vehículo go-kart



Fuente: planos realizados en AUTOCAD.

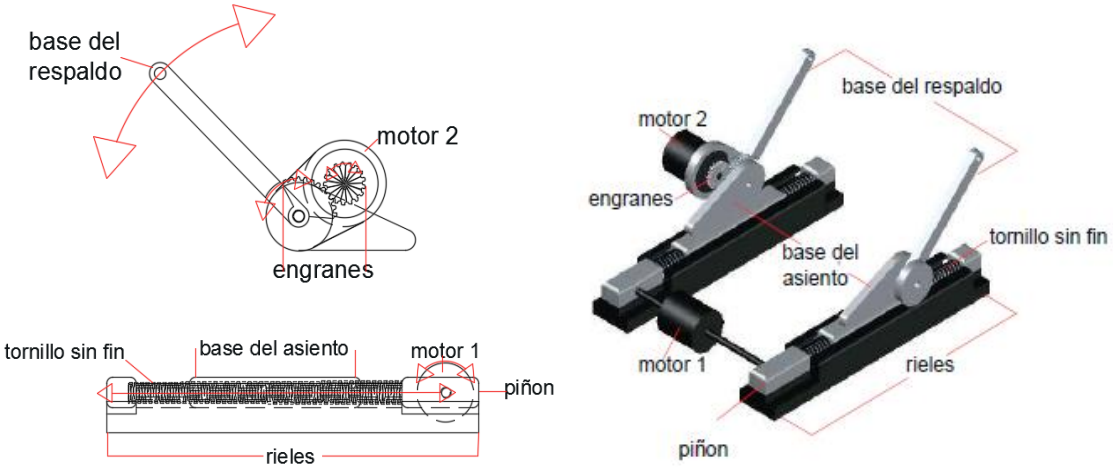
El diseño mecánico es un triciclo motorizado a gasolina, motor de 4 tiempos y 6.5hp marca Oakland (figura 17). Su diseño estructural es a base de PTR de una pulgada calibre 14 con lámina galvanizada de 1.5 mm de grosor para los acabados estéticos. El vehículo go-kart tiene dimensiones de 240 cm por 90 cm y un peso total de 89 kg (15.5 kg del motor, 7.6 kg del asiento y 65.9 kg de la estructura), cuenta con manubrio y llantas de bicicleta rodada número 20, tracción trasera con un eje de 1 pulgada de diámetro y una transmisión directa al motor con una relación de poleas 1:5.

Figura 13: Prototipo go-kart donde se montará el sistema de control

Fuente: fotografía tomada por el autor.

Para manipular el movimiento del asiento ajustable se tiene una estructura controlada por 2 motores eléctricos, uno para la base y otro para el respaldo (figura 18). El primer motor (motor 1) es el de la base del asiento, el cual transmite un movimiento circular a un movimiento rectilíneo a través de un piñón y un tornillo sin fin por cada lado. Dichos tornillos son los encargados de manipular la base donde va el sillón para cambiar su posición hacia atrás y hacia adelante respecto al manubrio (figura 18). Los soportes de la base del sillón a su vez están montados sobre un riel con balines y aceite para facilitar su movimiento y de igual manera impedir que este se salga de su eje de desplazamiento.

Figura 14: Diagramas de la base del asiento



Fuente: diagramas realizados en AUTOCAD.

El segundo motor (motor 2) es el del respaldo, cuenta con un mecanismo más sencillo de 2 engranes en transmisión directa 1:4 y genera un movimiento de ajuste de inclinación del respaldo con el mismo ángulo de abertura que el ángulo que tiene el engrane conducido. Para mantener la comodidad de un asiento inclinado no es viable conservar el mecanismo original de control de una bicicleta (manubrio).

Al tener un asiento inclinado, al operador del vehículo go-kart le resulta incómodo girar un volante sobre un eje horizontal. Con el uso de una junta universal (figura 18) de cualquier automóvil a una inclinación de 75^{o} respecto del eje "Y" se logra convertir un volante que gira sobre un eje horizontal a uno que gire sobre un eje vertical, conservando la posición inclinada del asiento y al mismo tiempo manipular el manubrio cómodamente.

Figura 15: Junta universal para inclinación de 75^{o} del volante

Fuente: fotografía tomada por el autor.

Además de los 89 kilogramos que pesa el vehículo go-kart se debe agregar el peso del conductor (70 kg) para hacer las pruebas necesarias de velocidad y torque del motor. Con un peso total de 159 kg se alcanza una velocidad estable e ideal de 30 km/hra comprobada con un velocímetro magnetico (figura 20). A mayor velocidad, el vehículo go-kart pierde su estabilidad porque no cuenta con un sistema de amortiguación adecuada.

Figura 16: Velocímetro magnético para bicicletas

Fuente: fotografía tomada por el autor.

Tomando como velocidad nominal 30 km/hra y un díametro de rueda de 51.56 cm se conoce el perímetro de la llanta mediante la fórmula del círculo.

$$\text{Perímetro de la llanta} = 2r * \pi$$

$$\text{Perímetro} = 2 * \left(\frac{51.56}{2}\right) * \pi = 162 \text{ cm} = 1.62 \text{ m}$$

Con el perímetro de la llanta y la velocidad que alcanza el vehículo go-kart se determinan las RPM del motor. Primero se realiza la conversión de km/hra a m/min y posteriormente se divide entre el perímetro de las ruedas.

$$30 \ \frac{\text{km}}{\text{hra}} * \left[\frac{1000 \text{ m}}{1 \text{ km}}\right] * \left[\frac{1 \text{ hra}}{60 \text{ min}}\right] = 500 \ \text{m/min}$$

$$\text{RMP} = 500 \ \frac{\text{m}}{\text{min}} \div 1.62 \text{ m} = 308 \text{ RPM}$$

Ahora si es posible conocer el torque real que mantiene el motor de 6.5 hp y las revoluciones a las que gira con las siguientes fórmulas.

$$\text{Torque} = \frac{30\ \text{Potencia (watts)}}{\pi \text{RPM}} \qquad \frac{d1}{d2} = \frac{T1}{T2} \qquad n1 * d1 = n2 * d2$$

1HP = 746 watts

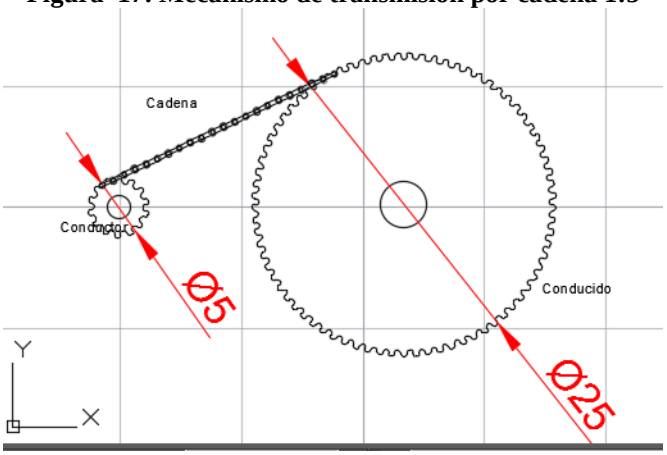
n = RPM

T = torque

d = díametro

Partiendo de que el engrane conducido tiene las mismas propiedades que las ruedas, se pueden determinar las características a las que trabaja el motor y la relación de los engranes (figura 21) mediante las fórmulas anteriores.

Figura 17: Mecanismo de transmisión por cadena 1:5



Fuente: diagramas realizados en AUTOCAD.

RPM del engrane conductor y del motor.

$$n1 = \frac{d2 * n2}{d1} = \frac{25\ \text{cm} * 308\ \text{RPM}}{5\ \text{cm}} = 1540\ \text{RPM}$$

Torque del engrane conductor y del motor.

$$\text{Potencia (watts)} = 6.5\ \text{HP} * \frac{746\ \text{watts}}{1\ \text{HP}} = 4.849\ \text{kW}$$

$$T1 = \frac{30 * \text{potencia}}{\pi * n} = \frac{30 * 4.849\ \text{kW}}{\pi * 1540\ \text{RPM}} = 30\ \text{N} * \text{m}$$

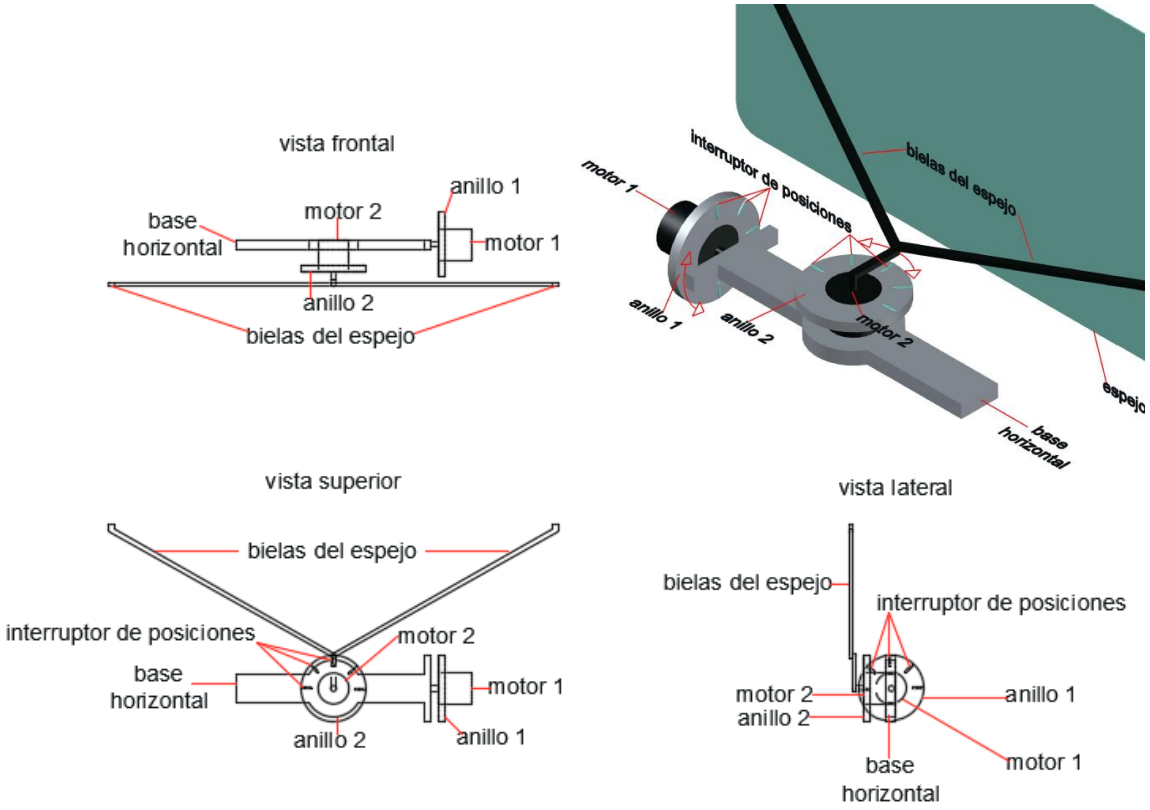
Torque del engrane conducido y del eje del vehículo go-kart.

$$n2 = 308\ \text{RPM}$$

$$T2 = \frac{T1 * d2}{d1} = \frac{30\ \text{N} * \text{m} * 25\ \text{cm}}{5\ \text{cm}} = 150\ \text{N} * \text{m}$$

El prototipo go-kart no cuenta con el sistema de control de espejos indicado para este tipo de control. El modelo requerido de control ya se encuentra en el mercado para diversas aplicaciones y parte de un concepto muy simple de magnetismo, en el cual las bielas giratorias de los motores cuentan con un imán que circula alrededor de un anillo con múltiples interruptores magnéticos (figura 22).

Figura 18: Diseño de mecanismos para el ajuste de la posición de los espejos



Fuente: diagramas realizados en AUTOCAD.

Cuando el imán activa un interruptor magnético, cierra su circuito eléctrico y emite una señal a un microcontrolador central en alguna de sus entradas digitales, indicando así la posición en la que se encuentra el espejo. Un espejo lateral retrovisor tiene 2 motores, uno para el sistema giratorio horizontal y otro para el sistema giratorio vertical. Un dato importante que se debe tomar en cuenta es que para evitar que el imán active 2 entradas de manera simultánea el sistema esta recubierto con aislante magnético (grafito pirolítico) que permite sólo el contacto entre el imán y el elemento que se encuentra justo enfrente de él. Algunos otros prototipos hacen uso de servomotores o motores a pasos para mayor precisión y un sistema mecánico de anclaje para evitar que la posición se desajuste con el movimiento.

Capitulo 7 Diseño electrónico

Se alcanza una mejor optimización en el proceso si se dividen las tareas en varios microcontroladores en lugar de una sola unidad central, de esta manera los procesos se realizan más rápido al tener pequeñas acciones repetitivas para cada módulo (sistemas embebidos) en lugar de todas las tareas dentro de uno mismo, brinda la posibilidad de tener módulos reactivos, los cuales sólo se activan cuando existe una reacción en el medio ambiente que lo amerite y los costos de producción son más reducidos al adaptar este modelo en lugar de dispositivos de mayor potencia como podrían ser las FPGA.

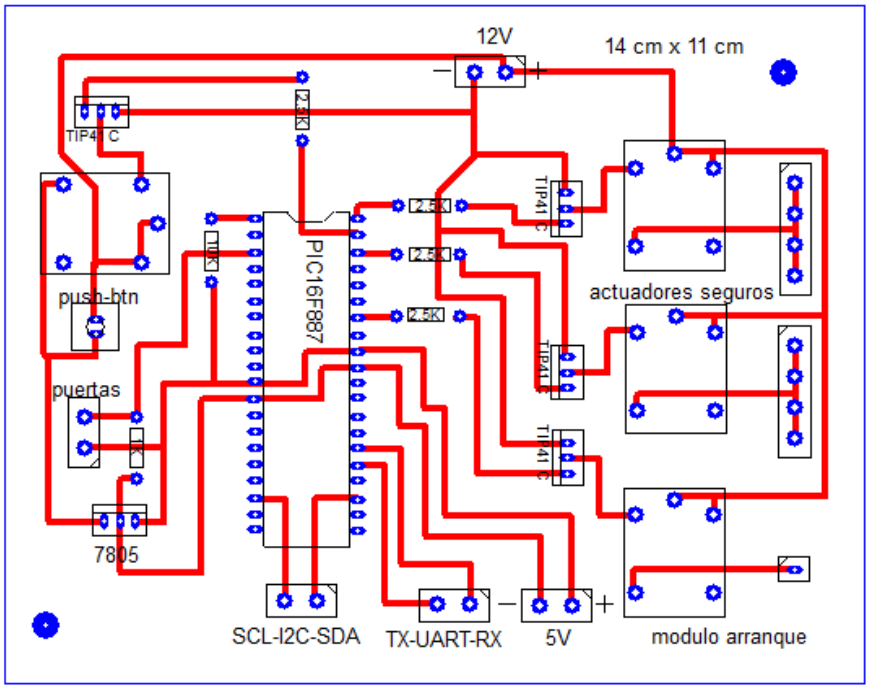
Dado que el proyecto requiere una comunicación interna de todos los elementos incorporados en el sistema, su comunicación está basada en una red I^2C donde se tiene un sólo elemento maestro central quien es el que interactúa con el usuario para la toma de decisiones y manda órdenes a los demás elementos esclavos. Dicho sistema de seguridad inteligente embebido requiere del uso de 4 microcontroladores en red, los cuales han sido diseñados para etapas específicas de control.

1.1. Primera etapa control de seguridad de puertas

Para las restricciones y la seguridad de usuarios en el proyecto, se tiene la detección de huellas digitales registradas en el sistema. De esta manera sólo las huellas autorizadas son capaces de acceder. Al accionar un botón de encendido (puede ser por un switch, sensor de proximidad o pulsador con enclavado), el único módulo que se va a iniciar será un controlador (pic16f887) con un lector y para poder pasar a los siguientes procesos se debe verificar primeramente que el usuario se encuentra habilitado.

Para un mismo automóvil es necesario tener un módulo lector para el control de seguros de las puertas y un segundo para el arranque del vehículo (el lector de huellas digitales hace uso del protocolo UART para la comunicación con su controlador por lo que requiere los puertos “RX” y “TX”).

Figura 19: Diseño de tarjeta electrónica para el control de seguridad de las puertas

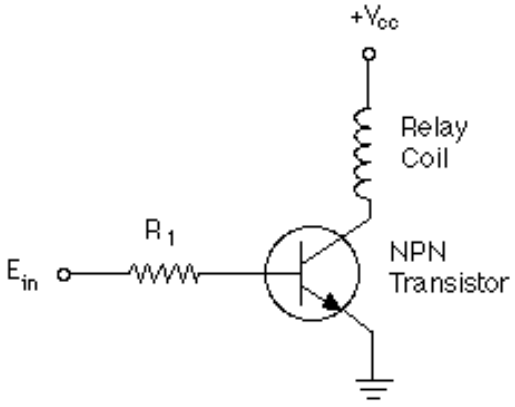


Fuente: diagrama realizado en PCB wizard.

El diseño del módulo de control de seguros de las puertas se basa en un mecanísmo de encendido por pulsador y enclavado (figura 23). Para encender esta parte del sistema se debe presionar el pulsador que energiza al microcontrolador (pic16f887) junto con el detector de huellas digitales y una vez encendido sin dejar de presionar se debe acceder una huella digital en el lector.

Si la huella digital es reconocida, el seguro de puertas se desactivará y el microcontrolador activará un relevador para enclavar el voltaje de alimentación del módulo, otro más para encender el módulo de arranque y así dejar de presionar el botón de inicio sin riesgo de apagar el sistema. Para activar de nuevo los seguros basta con abrir y cerrar la puerta del conductor o apagar el sistema desde la pantalla. El módulo cuenta con 1 push-botón para energizar el sistema, 1 regulador “7805” para alimentar los elementos internos, 1 salida de 5V y comunicación UART para el envío de información entre el pic16f887 y el lector de huellas digitales, comunicación I^2C para el control en red desde el dispositivo maestro, 1 entrada digital para el sensor de puertas abiertas y 4 salidas digitales a transistor (figura 24) para el control de relevadores de enclavado para este módulo, para el encendido del módulo de arranque, para la activación y la desactivación de seguros de las 4 puertas de un automóvil.

Figura 20: Configuración del TIP41C colector común



Fuente: imagen tomada de la hoja de datos del TIP41C.

Mediante las siguientes fórmulas se puede determinar la resistencia en base (R1) que debe llevar el transistor para saturarse y activar el relevador (considérese el relevador como una resistencia conectada al colector del transistor "tip41C").

$Ic = B * Ib$	$V = R * I$
Ic = corriente en colector	V = voltaje
B (hfe) = beta	R = resistencia
Ib = corriente en base	I = corriente

Primero se calcula la corriente que pasa por el relevador sabiendo que este tiene una resistencia de 400 Ω alimentado a 12V.

$$I = \frac{12V}{400\ \Omega} = 30\ mA$$

Ahora con este dato se calcula la corriente que pasa por la base del transistor si el datasheet indica una ganancia beta mínima de 15 y la corriente del colector (Ic) es igual a la del relevador.

$$Ib = \frac{30\ mA}{15} = 2\ mA$$

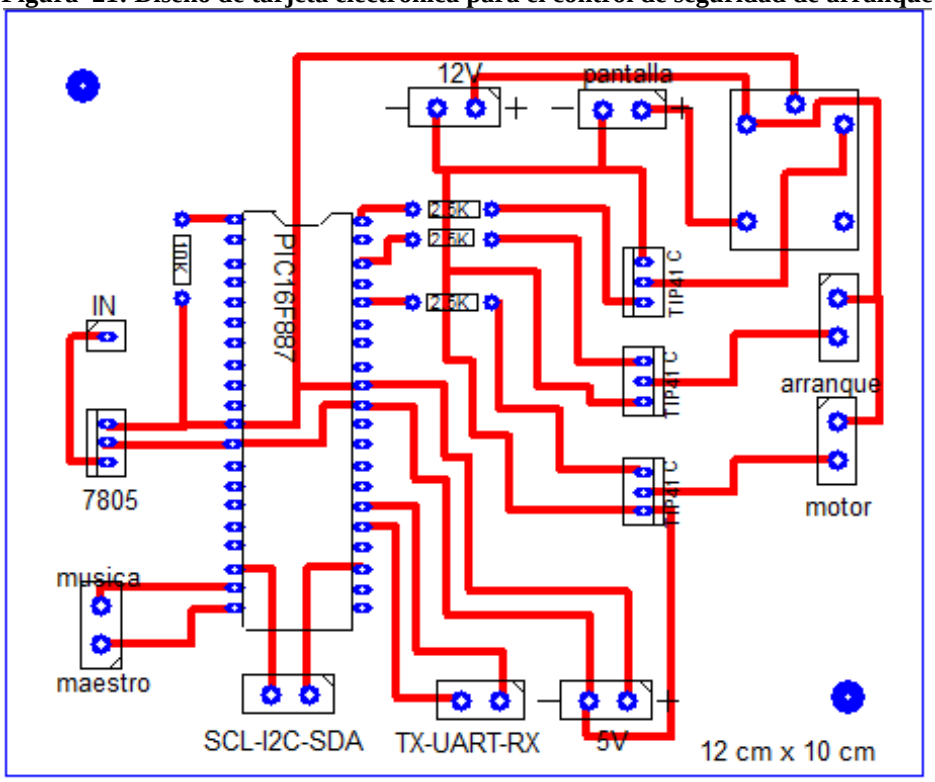
Por último se calcula la resistencia de la base del transistor donde su voltaje es de 5V ya que viene de una salida del microcontrolador (pic16f887).

$$R = \frac{5V}{2\ mA} = 2.5K\Omega$$

1.2. Segunda etapa arranque del vehículo

El módulo de arranque es muy similar al módulo de los seguros de las puertas, debido a que también hace uso de un pic16f887 y un lector de huellas digitales con salidas a transistores.

Figura 21: Diseño de tarjeta electrónica para el control de seguridad de arranque



Fuente: diagrama realizado en PCB wizard.

Después de que el módulo de los seguros de las puertas acepte un usuario y active sus salidas a transistores, el módulo de arranque se iniciará con el mismo mecanísmo de búsqueda de huellas digitales habilitadas para seguir con el protocolo de seguridad. Una vez que se encuentre algún usuario registrado, todas las salidas digitales que permiten el control de todo el sistema automatizado se activarán. Este tarjeta posee 1 regulador "7805" para todos los elementos internos, 1 salida a 5V y comunicación UART para la comunicación entre el microcontrolador y el lector de huellas digitales, comunicación I^2C para el control en red desde el dispositivo maestro, 2 salidas digitales para iniciar el módulo de música y el módulo maestro, 2 salidas digitales a transistores para las bujías, para el botón de arranque del motor y 1 salida digital a transistor con relevador para encender la pantalla touch. (figura 25).

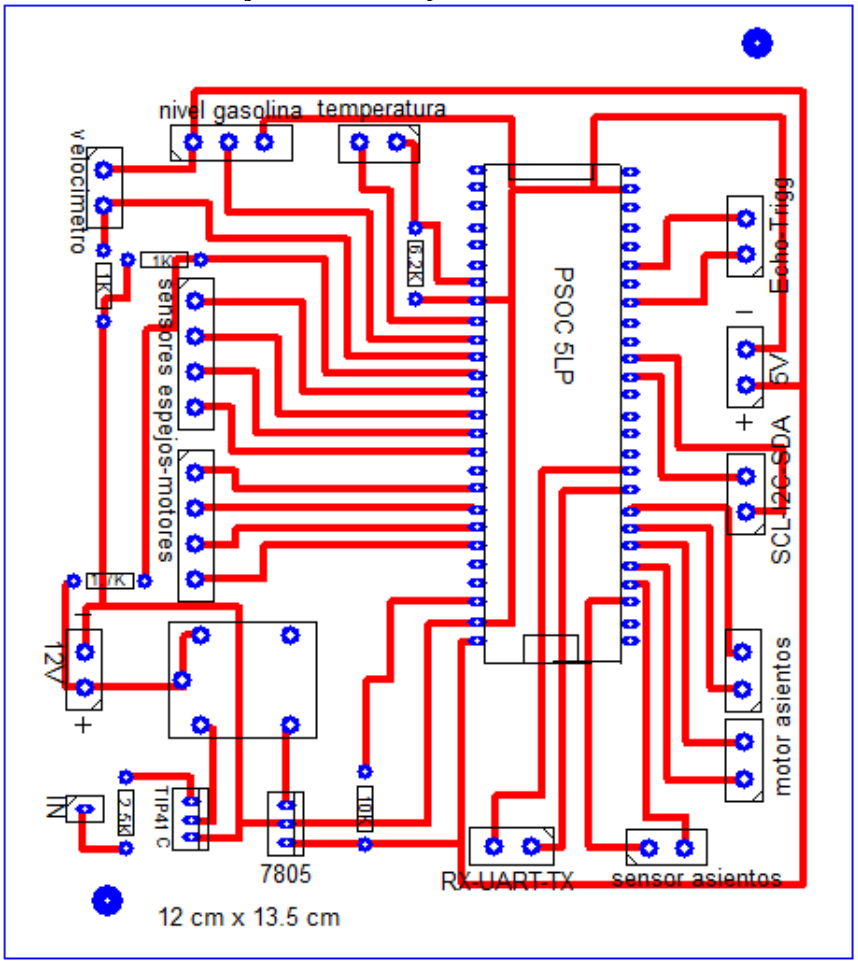
1.3. Tercera etapa control maestro

Este módulo es el módulo central, él se encarga de controlar todo el sistema, hacer consultas a las memorias y mandar instrucciones a los otros microcontroladores dentro de la red, realiza la lectura de los diferentes sensores para después mostrarlos en pantalla, manipula los motores del asiento y espejos. Para estos requerimientos enumerados, el microcontrolador PSOC5 es ideal para el proyecto, porque alcanza velocidades más rápidas de procesamiento y tiene menos pérdida de información al leer los sensores con sus convertidores de 12 y 16 bits. Tiene una comunicación UART con una pantalla touch a través de los puertos "RX" y "TX", desde la cual el usuario puede manipular un menú con diferentes opciones de procesos y visualizar las lecturas de todos los sensores. También tiene una comunicación I^2C para controlar los otros módulos.

El proceso es iniciado por el módulo de arranque después de validar una huella digital y al igual que el módulo controlador de música, es encendido por un transistor con relevador. Su función principal es leer cada cierto tiempo todos los sensores conectados y tener un intercambio de datos constante con la pantalla touch para mostrar información al usuario. Al presionar un botón gráfico, ejecuta una acción programada o manda una orden asignada a otro módulo dentro de la red I^2C.

Este tarjeta se enciende desde el módulo de arranque al aceptar una huella digital. Cuenta con 1 regulador "7805" para todos los elementos internos y salidas de alimentación a 5V para los sensores, comunicación UART para el intercambio de información entre el microcontrolador PSOC5 y la pantalla touch nextion, comunicación I^2C para controlar todos los elementos de la red, 2 entradas digitales para los sensores de posición del asiento del conductor, 4 salidas digitales para el puente H controlador de los motores del asiento, 4 entradas analógicas para los sensores de los espejos, 4 salidas digitales para los puentes H controladores de los motores de los espejos, 1 entrada digital para el velocímetro, 1 salida digital y 1 entrada analógica para la medición de distancia por medio de un sensor ultrasónico, 3 entradas analógicas para medir la temperatura, la capacidad de la batería y el nivel de gasolina por medio de un divisor de voltaje con 5V de referencia (figura 26).

Figura 22: Diseño de tarjeta electrónica para el control maestro de todo el sistema

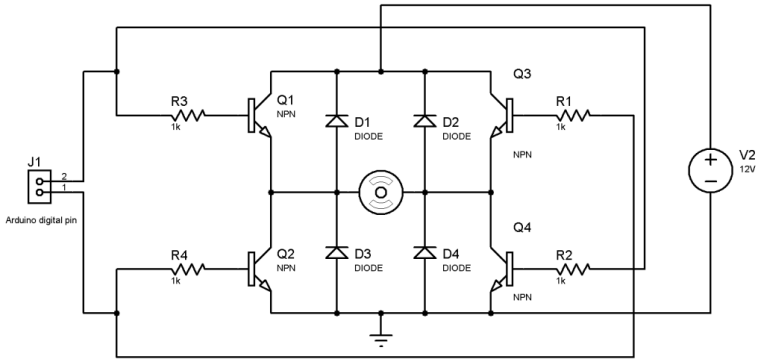


Fuente: diagrama realizado en PCB wizard.

Las salidas del microcontrolador hacia los motores tienen que ir primero a un puente H antes de llegar a ellos. De no ser así, los motores nunca se moverán porque funcionan con 12V y consumen más amperaje del que puede entregar una salida del microcontrolador.

Un puente H parte de un arreglo de resistencias y transistores como interruptores (figura 27). En la parte de los colectores de los 2 transistores primarios ingresa el voltaje que requieren los motores, (en este caso 12V) con sus emisores hacia el motor en polos opuestos. Estos a su vez se conectan a los colectores de otros 2 transistores con sus emisores a tierra. Para activar cada transistor, las bases se conectan con su respectiva resistencia a las salidas del microcontrolador de forma invertida, de manera que si se activa “J1”, este activará “Q1” Y “Q4” obligando a la corriente a pasar obligatoriamente a través del motor. Si no se conectan las salidas de forma invertida, la corriente seguirá el camino con menor resistencia evitando pasar por el motor.

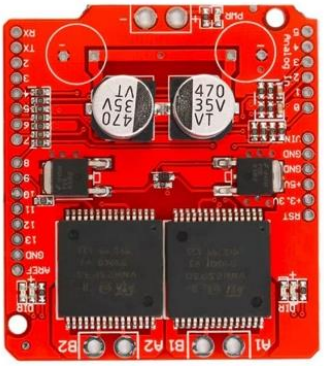
Figura 23: Diseño del funcionamiento de un puente H con transistores



Fuente: imagen tomada de la página http://panamahitek.com para proyectos con arduino.

El consumo de corriente de los motores eléctricos del asiento puede llegar a ser muy alto ya que el peso del usuario puede variar. De ser mucho peso, los motores requerirían gran consumo eléctrico para mover la carga. Aunque para este prototipo es un peso estándar de 70 kg, para evitar problemas se recomienda usar un puente H de 2 canales "vnh2sp30" (figura 28).

Figura 24: Puente H VNH2SP30



Fuente: imagen tomada de vendedores en mercado libre.

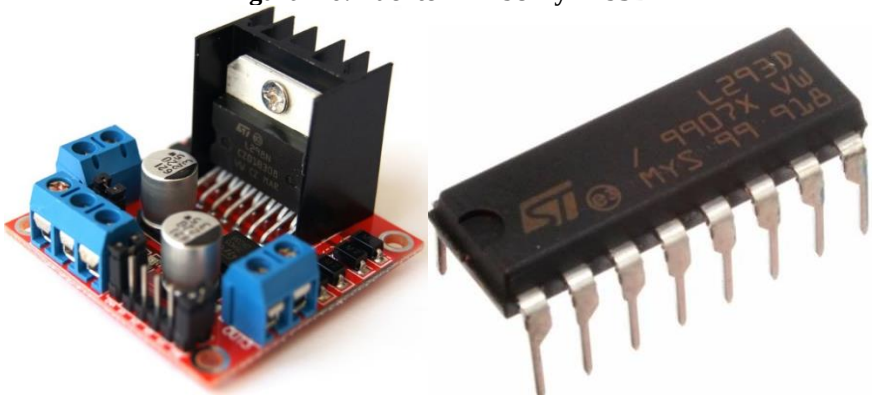
Este elemento cuenta con capacitores de aluminio, transistores mosfet y 2 puentes H de 2 canales cada uno que soportan hasta 30 A de consumo. Cuenta con 2 entradas por cada puente para el control en ambas direcciones de cada motor. El movimiento de los motores es establecido por tiempos. Mediante un sensor inductivo (detector de materiales metálicos) se establece la posición a donde deberá llegar el asiento al iniciar el proceso de ajuste (posición 0) y de ahí se empezará a contar en segundos hasta alcanzar la posición deseada (figura 29).

Figura 25: Sensor inductivo para control del asiento

Fuente: fotografía tomada por el autor.

Para el control de los motores de los espejos, el tipo de puente H que se llegue a usar dependerá del material de la estructura del diseño mecánico, ya que al ser un peso constante y muy pequeño a comparación con el de una persona, los motores requerirán menor torque y menor consumo de energía. Teniendo en cuenta esto, se puede usar el puente H "L298N" que soporta 2 A por canal, el puente H "L293D" con un arreglo de diodos que soporta 600 mA por canal (figura 30) o un diseño de transistores y diodos como ejemplifica la hoja de datos de un puente H genérico.

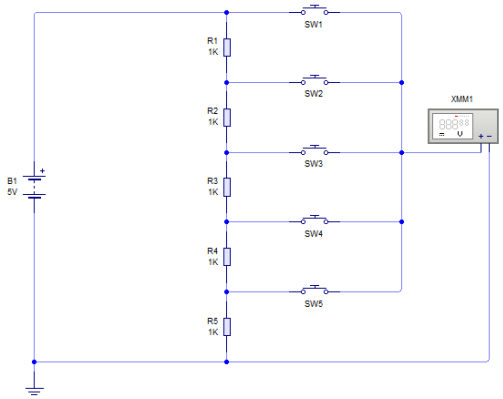
Figura 26: Puente H L298N y L293D



Fuente: imagen tomada de vendedores en mercado libre.

Los modelos de detección a base de interruptores magnéticos para el control de espejos que se encuentran en el mercado, no son los más viables para este proyecto. Ellos manejan 5 interruptores magnéticos por motor y 5 entradas digitales individuales del microcontrolador, haciendo un total de 10 por cada espejo. Un mejor modelo es con divisores de varios voltajes para cada motor conectados a una misma entrada del microcontrolador (figura 31).

Figura 27: Diseño de divisores de voltajes para las posiciones de los espejos



Fuente: esquema realizado en Live Wire.

De este modo se reducen los 20 pines digitales necesarios a sólo 4 entradas analógicas con 1V de diferencia entre cada interruptor magnético. El voltaje de diferencia se puede calcular con la siguiente fórmula.

$$\text{Vout} = \frac{\text{R2} * \text{Vin}}{\text{r1} + \text{r2}}$$

Por ejemplo, para este caso son 5 resistencias de 1 KΩ cada una. Con esa información se puede calcular el voltaje que existe en la resistencia R5 para comprobar la fórmula si se tiene Vin=5V, R1=4 KΩ y R2=1 KΩ.

$$\text{Vout} = \frac{1\ \text{K}\Omega * 5\text{V}}{4\ \text{K}\Omega + 1\ \text{K}\Omega} = 1\text{V}$$

El sensor de temperatura por excelencia es la sonda “NTC 3470” (figura 32) por su facilidad de uso. Este sensor termistor de cabeza de acero inoxidable funciona a base de un divisor de voltaje, con un rango de temperatura de -30 °C a +120 °C y una resistencia de 25 KΩ.

Figura 28: Sonda de temperatura NTC 3470

Fuente: imagen tomada de vendedores en mercado libre.

Dado que el sensor entrega temperaturas negativas con 150 °C de diferencia entre la más alta y la más baja en una escala de 0V a 5V, se tiene que el valor 0 °C equivale a 30 grados expresados en voltaje (1V).

$$Vout = \frac{5\text{V} * 30°\text{C}}{150\ °\text{C}} = 1\text{V}$$

Entonces, para calcular la resistencia R2 se tiene la fórmula despejada del divisor de voltaje.

$$R2 = \frac{\text{Vout} * \text{R1}}{\text{Vin} - \text{Vout}}$$

Por lo tanto R2 tiene su equivalencia en las resistencias comerciales R2=6.2 KΩ.

$$R2 = \frac{1\text{V} * 25\ \text{K}\Omega}{5\text{V} - 1\text{V}} = 6.25\ \text{K}\Omega$$

De igual manera el sensor de nivel de gasolina (figura 33) funciona a base de un divisor de voltaje que ya se encuentra con un potenciómetro interno de un 1 KΩ.

Figura 29: Sensor de nivel de gasolina de motocicleta honda

Fuente: fotografía tomada por el autor.

El divisor de voltaje también es usado para medir la carga de la batería de 12V, pero antes de conectarse con una entrada analógica se debe reducir el voltaje a 5V. Si se escoge una resistencia R1=1 KΩ entonces se tiene R2=1.7 KΩ.

$$R2 = \frac{12\text{V} * 1\ \text{K}\Omega}{12\text{V} - 5\text{V}} = 1.71\ \text{K}\Omega$$

Los coches también tienen un sistema de frenos ABS que comparten para medir la velocidad (figura 34). Este sistema cuenta con un mecanismo magnético de un sensor de efecto Hall sobre un disco giratorio de imanes y metal. Cada que el sensor pasa sobre un imán, se cierra un interruptor magnético y el microcontrolador va contabilizando las veces que se realiza este proceso por minuto. De esta manera se obtienen las RPM y se pueden calcular los Km/hra.

Figura 30: Sistema de frenos ABS y balatas

Fuente: fotografía tomada por el autor.

El prototipo del vehículo go-kart posee un mecanismo muy similar pero de mayor simplicidad ya que al no tener caja de velocidades las RPM dependen de un sólo disco. Con este sensor, un interruptor magnético y un imán en la llanta delantera (figura 35), se hacen las pruebas de velocidad (30 km/hra) para el cálculo de los engranes de transmisión (1:5).

Figura 31: Interruptor magnético del velocímetro

Fuente: fotografía tomada por el autor.

Cada vez que el imán en la llanta pasa sobre el interruptor que está sobre el chasís, la entrada digital del microcontrolador a la que se conecta el sensor, aumenta el valor de las RPM en 1.

Por último para la medición de distancia se usa el sensor ultrasónico “Hc-sr04" (figura 36). Este sensor aparte de detectar objetos, es capaz de medir la distancia en la que se encuentran desde los 2 cm hasta los 4 m en un angulo de 30^{o} según su hoja de datos. El sensor ultrasónico tiene una ventaja frente a los sensores infrarrojos en relación a su uso en exteriores ya que no le afectan los cambios de luz para las mediciones y cuenta con un ángulo más amplio de detección de objetos, pero tiene la desventaja de ser más lento en su proceso debido a que la velocidad del sonido es más lenta que la velocidad de la luz y le cuesta trabajo detectar objetos de materiales suaves como son algunas fibras sintéticas.

Figura 32: Sensor ultrasónico HC-SR04

Fuente: imagen tomada de vendedores en mercado libre.

Para su funcionamiento el microcontrolador activa el pin “TRIGGER” y este a su vez genera una señal que viaja a la velocidad del sonido (343 m/s), después de ello se cuenta el tiempo que tarda en regresar al receptor “ECHO” y dependiendo de este valor se conoce la distancia a la que se encuentra un objeto. El controlador recibe un valor equivalente al tiempo que el sensor tarda en captar la señal, este valor debe ser dividido entre 2 ya que la distancia recorrida es el camino de ida entre el pin “TRIGGER” y el objeto, más el camino de vuelta entre el objeto y el pin “ECHO”. Como ejemplo supongamos que el sensor entrega 1.47 milisegundos, por lo tanto un objeto cualquiera está situado a una distancia de 25.21 cm.

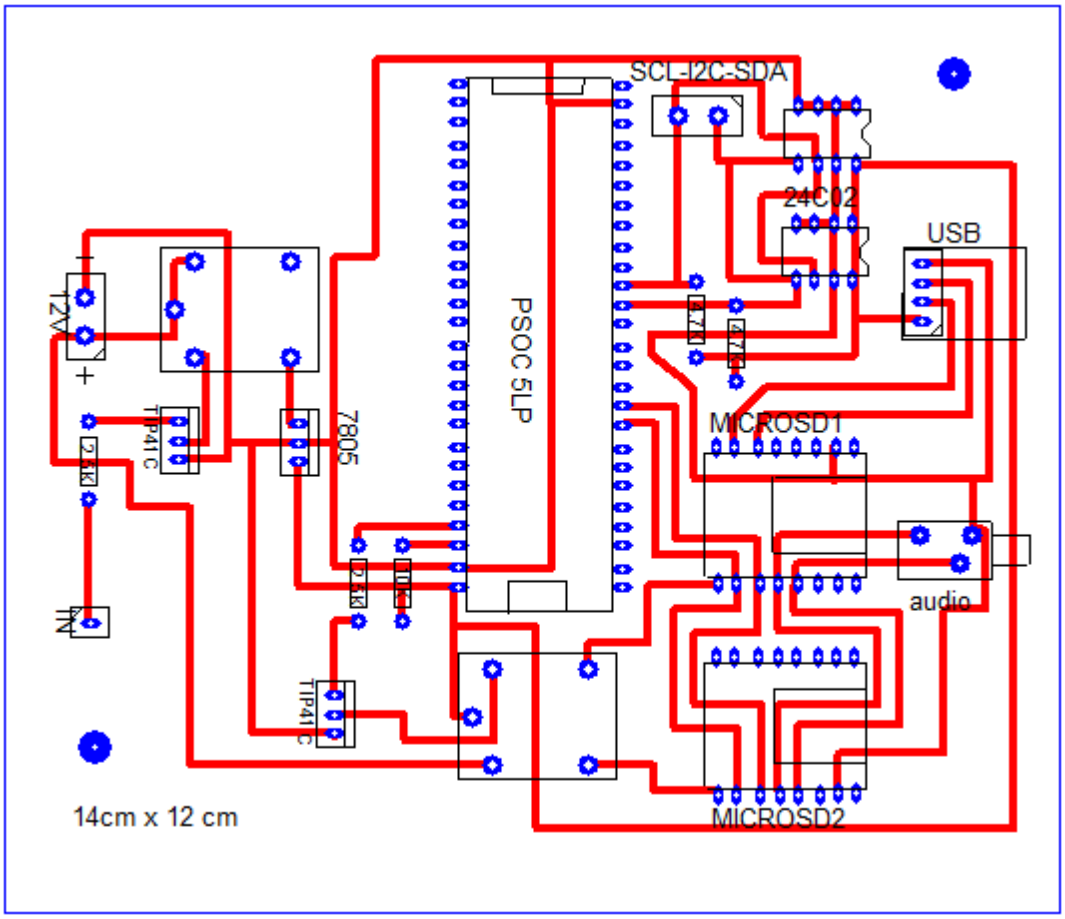
$$\mathrm{D} = 343\frac{\mathrm{m}}{\mathrm{s}} * 1.47\ \mathrm{ms} = \frac{504.2\ \mathrm{mm}}{2} = 252.1\ \mathrm{mm} = 25.21\ \mathrm{cm}$$

1.4. Cuarta etapa control de música

Al mismo tiempo que se enciende el módulo maestro y la pantalla touch, se inicia el módulo de música. Este módulo no hace ninguna tarea al encender, sólo se quede preparado para recibir indicaciones del maestro y controlar la música que se encuentra en las memorias microSD.

Este tarjeta se enciende desde el módulo de arranque al aceptar una huella digital. Cuenta con 1 regulador "7805" para todos los elementos internos, comunicación UART para controlar los reproductores de las memorias microSD, comunicación I^2C para su manipulación desde el dispositivo maestro a través de la red, 2 memorias eeprom "24C02" para el almacenado de información personal de los usuarios, 1 conector de salida de audio y 1 salida digital a transistor con relevador para la selección del reproductor de música (figura 37).

Figura 33: Diseño de tarjeta electrónica para el control de música

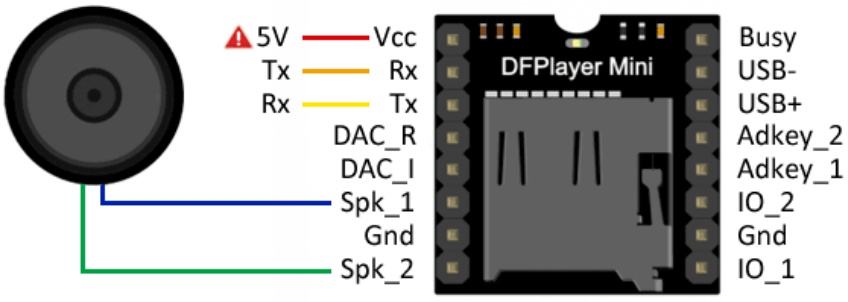


Fuente: diagrama realizado en PCB wizard.

Una vez que el módulo se enciende, este se conecta a la red I^2C donde recibirá órdenes del elemento maestro para gestionar el reproductor de música indicado. Ya que todos los reproductores de música “WTV020” vienen con la misma dirección incorporada desde su fabricación, no es posible tener más de uno conectado directamente al microcontrolador. Para solucionar este problema, existe un relevador con un estado para cada reproductor, pero si se deseará tener múltiples reproductores, se tendría que aumentar el número de relevadores por cada reproductor o buscar otro tipo de dispositivo capaz de reproducir música en una memoria.

El reproductor de música “WTV020” (figura 38) se comunica con un PSOC5 de quien recibe los comandos “play”, ”pause”, ”next” y ”previous” mediante el protocolo UART, quien a su vez recibe instrucciones dentro de la red I^2C por el módulo maestro y este por el usuario desde la pantalla touch. El reproductor cuenta con una forma de conectarse directamente a una bocina por los puertos “spk_1” Y “spk_2” o con salida de audio para soportar bocinas de mayor potencia a través de los puertos “DAC_R”, “DAC_L” y “GND”.

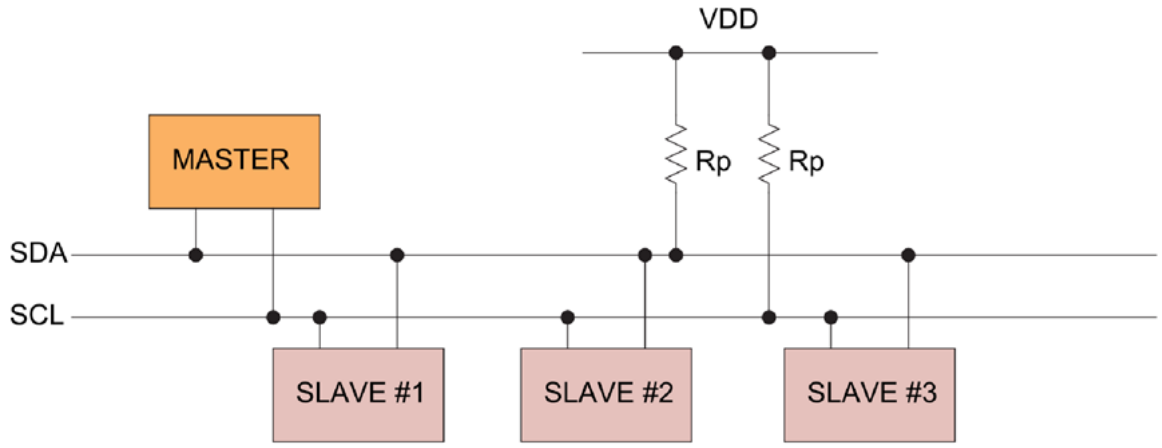
Figura 34: Configuración del módulo MP3 WTV020



Fuente: imagen tomada de la hoja de datos del reproductor WTV020.

Dentro de la misma tarjeta electrónica también se encuentran las memorias eeprom “24C02” que guardan la información de la posición de los asientos, espejos y fechas de servicios. Estas memorias tienen una dirección única de acceso dentro del sistema que se escoge al momento de conectar sus entradas, “A0” para la memoria que conecta todas las entradas “AD” a tierra y “A2” para la memoria que cambia “AD0” a 5V. Para este proyecto se requieren sólo 5 direcciones de almacenamiento por memoria de las 8 que tienen disponibles ya que no se cuenta con espejos ajustables. La red I^2C soporta 8 memorias como máximo de esta matrícula y debe tener 2 resistencias de 4.7 KΩ conectadas a 5V para las líneas “SDA” y “SCL” (figura 39).

Figura 35: Diagrama del protocolo I²C



Fuente: imagen tomada de la documentación oficial de la empresa CYPRESS.

Capitulo 8 Diseño de programación

La programación está basada en lenguaje C y hace uso de 2 plataformas, MPLAB para los pic16f887 y PSOC creator para los PSOC5 LP. Al tener similitudes en código y relojes internos de alta precisión, es viable hacer la comunicación I²C entre estos 2 fabricantes.

1.1. PIC16F887

Lo primero que se debe realizar es establecer las configuraciones generales de los microcontroladores para hacer uso de sus periféricos. La siguiente configuración es para los pic16f887 utilizados con oscilador interno de 8 MHz, comunicación UART a 9600 baudios, comunicación I²C a 100 Kbps e interrupciones.

```
#pragma config FOSC = INTRC_NOCLKOUT                    // activa el oscilador interno
#define delay_ms(x) _delay((unsigned long)((x)*(8000000/4000)))    // fuente de reloj a 8-MHz
OSCCON=0B01110101;      // velocidad del oscilador interno 8-MHz
   SPBRG=12;            // baudaje -> 9600
   SYNC=0;              // comunicación asíncrona
   SPEN=1;              // puerto serial activado
   TXEN=1;              // transmisión activada UART
   CREN=1;              // recepción activada UART
   GIE=1;               // interrupciones activadas
   PEIE=1;              // interrupción por periférico activada
   RCIE=1;              // interrupción por recepción activada
```

El registro “OSCCON” determina las características que tendrá el oscilador y para poder modificar los valores se utiliza la siguiente tabla de configuraciones de bits (figura 40).

Figura 36: Configuración de los bits del registro OSCCON

OSCCON	-	IRCF2	IRCF1	IRCF0	OSTS	HTS	LTS	SCS
Características	R/W (1)	R/W (1)	R/W (0)	R (1)	R (0)	R (0)	R/W (0)	
Nombre de bit	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

IRCF2	IRCF1	IRCF0	FRECUENCIA
1	1	1	8 MHz

Fuente: imagen tomada de la página del fabricante microchip https://www.mikroe.com/.

Bit 7: bit sin función.
Bit 6 (IRCF2): configuración de frecuencia del oscilador.
Bit 5 (IRCF1): configuración de frecuencia del oscilador.
Bit 4 (IRCF0): configuración de frecuencia del oscilador.
Bit 3 (OSTS): determina cual oscilador estará activo (1 para externo y 0 para interno).
Bit 2 (HTS): 1 activa el oscilador interno de alta frecuencia (8 MHz -125 kHz).
Bit 1 (LTS): 1 activa el oscilador interno de baja frecuencia (125 kHz -31 kHz).
Bit 0 (SCS): determina cual oscilador (1 para interno y 0 para externo) se usará como fuente de reloj.

También se debe activar el puerto serial (SPEN=1), la recepción (CR=1) y la transmisión (TXEN=1), las interrupciones globales (GIE=1), las interrupciones por periféricos (PEIE=1), las interrupciones por recepción (RCIE=1), establecer la comunicación asíncrona (SYNC=0) para el protocolo UART y el baudaje a 9600 baudios (SPBRG=12). Para determinar el baudaje a 9600 baudios se calcula el valor de la función “SPBRG” con la siguiente fórmula.

$$\text{SPBRG} = \frac{\text{F\_OSC}}{\text{BAUD} * 64} - 1$$

BAUD = baudaje

F_OSC = cristal

Por lo tanto, para un cristal de 8 MHz a 9600 baudios se calcula el siguiente valor.

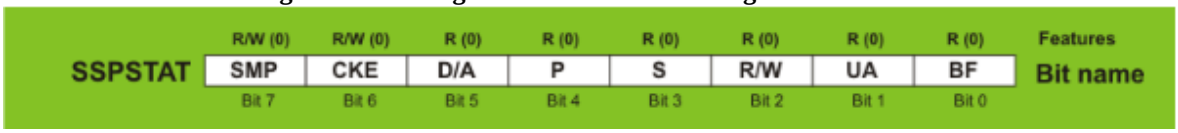
$$SPBRG = \frac{8\ \text{MHz}}{9600 * 64} - 1 = 12$$

Como siguiente paso para la conexión de la red I^2C a 100 Kbps, se configuran ambos microcontroladores como esclavos y se activan las interrupciones como las que anteriormente se ejemplificaron.

```
SSPSTAT = 0B10000000;   //  establece una velocidad estándar de 100 Kbps.
SSPADD =  0XB0;          //dirección de esclavo 0XB0 para un dispositivo y 0XB2 para el otro.
SSPCON = 0B00110110;   // síncrono, activa los puertos scl-sda, inicia el reloj, modo esclavo de 7 bits.
SSPCON2 = 0B00000001; //inicia protocolo I2C.
SSPIF = 0;                //limpia la bandera de interrupción I2C.
SSPIE = 1;                //habilita las interrupciones para I2C.
```

Los registros utilizados generan una configuración específica de trabajo para este protocolo. En el caso de este proyecto son necesarias las interrupciones para el protocolo I^2C (SSPIE=1), limpiar la bandera de interrupción I^2C (SSPIF=0), asignar la dirección "0XB0" para la escritura de un dispositivo y "0XB1" para su lectura, "0XB2" para la escritura del otro microcontrolador y "0XB3" para la lectura de este. Finalmente para los demás registros se asignan configuraciones de trabajo en base a las siguientes tablas (figura 41, figura 42 y figura 43).

Figura 37: Configuración de los bits del registro SSPSTAT



Fuente: imagen tomada de la página del fabricante microchip https://www.mikroe.com/.

Bit 7 (SMP): 1 para para velocidad estándar de 100 Kbps y 0 para alta velocidad de 400 Kbps.
Bit 6 (CKE): cambio de transmisión de flanco de subida = 1 a flanco de bajada = 0.
Bit 5 (D/A): 1 si el byte es un dato y 0 si el byte es una dirección.
Bit 4 (P): 1 cuando se ha detectado el STOP.
Bit 3 (S): 1 cuando se ha detectado el START.
Bit 2 (R/W): 1 para modo lectura y 0 para modo escritura.
Bit 1 (UA): 1 cuando se indica que se requiere actualizar la dirección.
Bit 0 (BF): en recepción (1 completa y 0 en proceso) o en transmisión (1 en proceso y 0 completa).

Para este registro sólo se requiere establecer el bit 7 = 1.

Figura 38: Configuración de los bits del registro SSPCON

SSPCON	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	Features
	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	Bit name
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	

SSPM3	SSPM2	SSPM1	SSPM0	MODE
0	1	1	0	I²C slave mode, 7-bit address used

Fuente: imagen tomada de la página del fabricante microchip https://www.mikroe.com/.

Bit 7 (WCOL): 1 si al escribir las condiciones no son válidas y falla la escritura.
Bit 6 (SSPOV): 1 si un nuevo dato quiere ser recibido pero no existe espacio de memoria.
Bit 5 (SSPEN): 1 para iniciar el modo síncrono y los puertos scl-sda.
Bit 4 (CKP): 0 para mantener el reloj a nivel bajo.
Bit 3 (SSPM3): selecciona el modo esclavo de 7 bits.
Bit 2 (SSPM2): selecciona el modo esclavo de 7 bits.
Bit 1 (SSPM1): selecciona el modo esclavo de 7 bits.
Bit 0 (SSPM0): selecciona el modo esclavo de 7 bits.

Figura 39: Configuración de los bits del registro SSPCON2

SSPCON2	R/W (0)	R (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	Features
	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	Bit name
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	

Fuente: imagen tomada de la página del fabricante microchip https://www.mikroe.com/.

Bit 7 (GCEN): 1 para activar una interrupción cuando existe un llamado general.
Bit 6 (ACKSTAT): 0 cuando ACK fue recibido del esclavo en modo maestro transmitiendo.
Bit 5 (ACKDT): 0 cuando ACK fue recibido del esclavo en modo maestro recibiendo.
Bit 4 (ACKEN): 1 cuando se está solicitando ACK.
Bit 3 (RCEN): 1 para activar la recepción.
Bit 2 (PEN): 1 para iniciar la condición de STOP.
Bit 1 (RSEN): 1 para reiniciar la condición de START.
Bit 0 (SEN): 1 para iniciar la condición de START.

Para este registro sólo se requiere establecer el bit 0 = 1.

Lo primero que el usuario hará para ingresar al vehículo es presionar el push botón de la puerta del conductor (figura 44) y al mismo tiempo sin dejar de presionarlo ingresará la huella digital.

Figura 40: Sensor de huellas digitales para seguros de puertas Hyundai 2019

Fuente: imagen tomada de la revista https://www.motorpasión.com.mx.

Esta es la primera parte del código que se ejecutará al iniciar el módulo de seguros de puertas. Sin dejar de presionar el push botón de encendido, el microcontrolador entra en un ciclo "while" que genera la función de búsqueda de huellas digitales y para salir del ciclo debe encontrar una huella registrada. Para la interacción entre el lector de huellas digitales y el pic16f887, se mandan y reciben líneas de código a través de los puertos "RX" y "TX". Los siguientes comandos son los encargados de la comunicación entre estos elementos que deberán ser enviados en un arreglo.

```
char header[]=  {0XEF,0X01,0XFF,0XFF,0XFF,0XFF,0X01,0X00};
char collect[]= {0X03,0X01,0X00,0X05};
char buffer[]=  {0X04,0X02,0X00,0X00,0X00};
char search[]=  {0X08,0X04,0X00,0X00,0X00,0X00,0X00,0X00,0X00};
char templt[]=  {0X03,0X05,0X00,0X09};
char store[]=   {0X06,0X06,0X00,0X00,0X00,0X00,0X00};
char data[20];
```

Además de estos arreglos, se debe crear un método de escritura para el envío de las funciones hacia el lector de huellas digitales y un método de lectura con interrupción de procesos para la recuperación del número de usuario o la determinación de algún error que haya ocurrido. Dentro de estos métodos existe una variable llamada "lenght" para determinar el tamaño del arreglo que se va a mandar o se va a recibir y un arreglo llamado "data[]" que será llenado con los valores del arreglo "header[]" más el comando a enviar.

1.1.1. Método escritura

```
void finger_write(void){
   flag=0;                          // bandera de estado - inicio escritura
   lenght=8+(data[8]);
   for(i=0;i<=lenght;i++){
      TXREG=data[i];                // transmite lo contenido en data[]
      while(TRMT==0){}
   }
}
```

Con el método de escritura se pondrá la variable “flag=0” para determinar que se va a mandar información al lector de huellas digitales. Previamente se ha llenado el arreglo “data[]” con la función que se desea enviar y se ha determinado la longitud del arreglo con la variable “lenght” para posteriormente enviar dato por dato al buffer de salida “TXREG” a través de un ciclo “for”. El bit “TRMT” se pondrá en 1 cuando el buffer haya transmitido todo lo contenido.

1.1.2. Método lectura

```
void finger_read(void){
   if(RCREG==0XEF)              // byte inicial
      i=0;
   data[i]=RCREG;               // guarda en data[] lo recibido
   lenght=8+(data[8]);
   if(i==lenght){
      code=data[9];             // código de respuesta
      id=data[11];              //  ID
      flag=1;                   // bandera de estado - fin de lectura
   }
   i++;
}
```

Una vez que se ha mandado el arreglo y el método escritura ha terminado, el microcontrolador espera recibir otro arreglo de respuesta. Cuando esa respuesta empieza a llegar al buffer de entrada “RCREG”, se genera una interrupción e inicia el método de lectura. Lo primero que se verifica es que el primer valor recibido sea “0XEF” y si esto es correcto, se llenará el arreglo “data[]” con los valores que recibió el buffer “RCREG”. Por último se almacena el tamaño del arreglo recibido, el valor de la posición “data[9]” para saber que no existieron errores de transmisión, el valor de la posición “data[11]” para conocer el número de usuario que ingresó al sistema y finalmente se pone la variable “flag=1” para indicar que este proceso ha finalizado.

Con estas funciones como cabecera, el microcontrolador pic16f887 puede gestionar el método de búsqueda de huellas dentro del ciclo “while” mencionado anteriormente. Este método hace uso de los métodos de escritura y lectura para mandar y recibir en un orden estructurado los comandos que han sido previamente transferidos al arreglo “data[]”. El orden en el que se deben mandar los comandos es “collect[]”-“buffer[]”-“search[]”.

1.1.3. Método de búsqueda

```
void busqueda(void){
   delay_ms(50);
                              finger_collect();          while(flag==0){}
   if(code==0){               finger_buffer(1);          while(flag==0){}
      if(code==0){            finger_search(1,0,162);    while(flag==0){}
         if(code==0){
            break();
  }  }  }
   if(code!=0){id=0;}
}
```

El método de búsqueda genera primero un retardo de tiempo para dar un breve espacio a la estabilización del sensor de huellas digitales. Posteriormente se inicia el proceso con el comando “collect[]” que será guardado junto con el comando “header[]” en el arreglo “data[]” para transmitirse. Este comando ordena tomar una muestra por el sensor de huellas digitales y esperar hasta que la variable ”flag” sea igual a 1, lo que significa que ha terminado el método lectura.

Después de ello, se verifica que no haya existido ningún error al tener la variable “code” igual a 0 o de lo contrario el proceso se repetirá y la variable “id” será igual a 0. Con el mismo esquema de proceso, el método seguirá avanzando al comando “buffer[]” más el arreglo “header[]” que se encargan de asignar la huella tomada por el sensor a la plantilla 1 y finalmente el comando “search[]” más el arreglo “header[]” buscarán la plantilla 1 en las localidades 0-162.

```
   while(cicloInfinito==0){
     busqueda();
     if(id==2||id==1){
       ID_huella=id;
       cicloInfinito=1;
     }else{
      cicloInfinito=0;
     }
  }
```

Si hubo éxito en la búsqueda y no se generó ningún error (code=0), se comparará la variable "id" que almacena el número de huella encontrada, con el valor que identifica la misma huella en el sistema y en caso de que coincidan ambos números, se guardará ese valor en la variable "ID_huella" para finalmente salir del ciclo "while".

Después se ponen en alto las salidas de los seguros para desactivarse, el pin de autoenclavado propio del módulo y el pin que enciende el módulo de arranque. Para volver a activar los seguros existen 3 métodos. En el primero, el microcontrolador realiza una segunda búsqueda de huellas digitales, si llegase a encontrar alguna registrada, activará los seguros y apagará el sistema. También si dectectase con el sensor de puertas abiertas un cambio de estado 2 veces (abrir y cerrar la puerta) activará los seguros de nuevo o si recibiese del maestro a través de la red I^2C alguna de las siguientes letras, realizará la función indicada.

A= desactivar seguros de puertas.
C= activar seguros de puertas.
N= apagar sistema.
B= salir del método de búsqueda de huella.
R= nuevo registro.

1.1.4. Interrupción I^2C

```
void interrupt I2C_Slave_Read(){
  if(SSPIF == 1){                                      //bandera de interrupción
    SSPCONbits.CKP = 0;                                //cambia a flanco de bajada
      if(!SSPSTATbits.D_nA && !SSPSTATbits.R_nW){      //RECEPCIÓN DE DATOS
      z = SSPBUF;                                      // lee el antiguo valor en el registro
      while(!BF);                                      //espera recepción completa
      valor1 = SSPBUF;                                 //recibe datos del registro
      SSPCONbits.CKP = 1;                              //cambia a flanco de subida
    }
    else if(!SSPSTATbits.D_nA && SSPSTATbits.R_nW){    //ENVIO DE DATOS
      z = SSPBUF;                                      //lee el antiguo valor en el registro
      BF = 0;                                          //transmisión de envio completa
      SSPBUF = valor2;                                 //envio de datos al registro
      SSPCONbits.CKP = 1;                              //cambio a flanco de subida
      while(SSPSTATbits.BF);                           //espera bit de reconocimiento
    }
    SSPIF = 0;                                         //baja bandera de interrupción
  }
}
```

Como el modo esclavo no puede solicitar información de otro dispositivo en red, este dispositivo sólo cuenta con una interrupción. Cuando se detecta un valor en el buffer "SSPSR", se genera una interrupción I^2C y se pone el bit "SSPIF=1" con una detección por flanco de bajada. El método I^2C se estructura de la siguiente manera.

Modo escritura: manda dirección de escritura, manda dato.
Modo lectura: manda dirección de ecritura, manda dirección de lectura, lee dato.

Si el módulo de arranque va a recibir un dato, este entra en modo de recepción con la interrupción del primer valor que es la dirección del dispositivo (0XB2), espera nuevos datos del buffer "SSPSR" en el registro "SSPBUF" para almacenar el comando a ejecutar en la variable "valor1" y cambia al flanco de subida. Si el dispositivo va a enviar un dato, este repite el proceso de recepción y luego entra en modo de envío con la interrupción del segundo valor que es la nueva dirección del dispositivo (0XB3), forza la transmisión completa, envía el valor solicitado de la variable "valor2", cambia al flanco de subida y espera que la transmisión sea completada. Finalmente se pone el bit "SSPIF=0" para indicar que se ha completado la transmisión.

El módulo de arranque necesita ambos métodos para mandar el número de huella reconocida a todo el sistema y para recibir las letras que le indican que proceso realizar. El módulo de seguridad de puertas sólo requiere el modo de recepción para los comandos. Si se llegase a recibir la letra "R" en cualquiera de los 2 módulos, estos entrarán a un ciclo "while" para guardar una nueva huella digital con el siguiente método.

1.1.5. Método de registro

```
void registro(void){
   delay_ms(50);
                              finger_collect();                  while(flag==0){}
   if(code==0){               finger_buffer(1);                  while(flag==0){}
      if(code==0){            finger_collect();                   while(flag==0){}
         if(code==0){         finger_buffer(1);                  while(flag==0){}
            if(code==0){      finger_template ();                while(flag==0){}
               if(code==0){   finger_store (1, ID_huella);       while(flag==0){}
                  if(code==0){
                    cicloInifinito=1;
                    break();
  } } } } } }
   if(code!=0){ cicloInfinito=0;}
}
```

Este método al igual que el método de búsqueda, toma una lectura por el sensor y lo asígna a la plantilla 1, luego vuelve a tomar otra lectura por el sensor y lo asígna a la plantilla 2, compara ambas plantillas y de ser iguales guarda la plantilla 1 en la dirección que contiene la variable "ID_huella" para salir del ciclo "while". De no ser iguales las plantillas, manda un código de error y vuelve a iniciar el proceso de registro.

El método anterior se usa en el módulo de seguridad de puertas y en el módulo de arranque (figura 45). El módulo de arranque se enciende exactamente después de la primera búsqueda con éxito del módulo de seguridad de puertas e inicia también con una búsqueda de huellas digitales. Si la huella es encontrada, activa los puertos para encender la pantalla touch, el módulo de música y el módulo maestro, conecta las bujías al motor, permite el uso del botón de arranque para encender el vehículo e inicia una segunda búsqueda de huellas digitales para desactivar el botón de arranque una vez puesto el vehículo en marcha.

Figura 41: Sensor de huellas digitales para encendido de automóvil Hyundai 2019

Fuente: imagen tomada de la revista https://www.motorpasíon.com.mx.

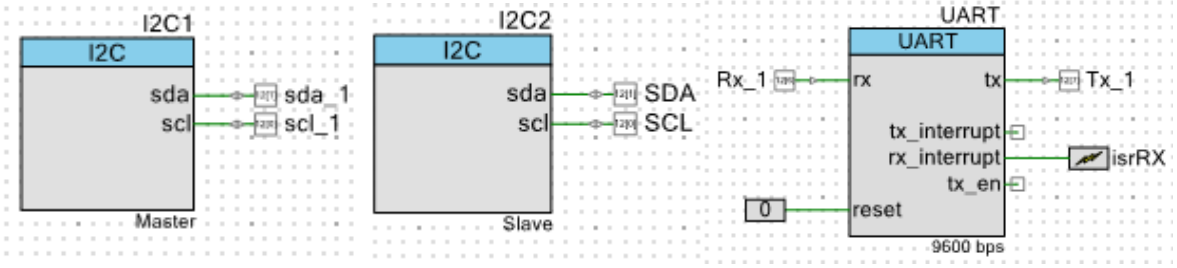
Este microcontrolador tiene las mismas configuraciones generales para el pic16f887, los comandos del lector de huellas digitales, el método de escritura, el método de lectura, el método de búsqueda, el método de registro, la interrupción por periférico I^2C (dirección "0XB2") con función de recepción y transmisión de datos. Dentro de sus procesos, el módulo maestro le puede pedir el número de usuario que ingresó al sistema, mandarle la letra "R" para iniciar el método de registro o la letra "I" para volver a activar el botón de arranque.

1.2. PSOC 5LP

El módulo de control de música y el módulo maestro se basan en la programación de un PSOC5 y al igual que el pic16f887, se deben crear las configuraciones generales junto con la inicialización de los protocolos de comunicación. El PSOC5 tiene un reloj interno estándar de 8 MHz por lo que no necesita ajustarse a no ser que se desee trabajar a diferente frecuencia. Además de código, este microcontrolador permite una programación a bloques que facilita su desarrollo.

Los bloques de los protocolos (figura 46) se configuran haciendo doble clic sobre ellos en la ventana emergente, I^2C modo maestro a 100 kbps y UART con interrupción por recepción a 9600 baudios para los canales "RX" y "TX" sin paridad para el módulo maestro. I^2C modo esclavo con interrupción y UART con las mismas características para el módulo de control de música.

Figura 42: Bloques de comunicación I^2C (master-slave) y UART para PSOC5



Fuente: imagen tomada de la plataforma programable PSOC creator.

Una vez creadas las configuraciones iniciales para el módulo maestro, corresponde crear los métodos de escritura y lectura para cada uno de los elementos existentes. La estructura de comunicación es la siguiente.

1.2.1. Método escritura

```
void escribir(){
I2C1_Start();
I2C1_MasterSendStart("dirección",0);
I2C1_MasterWriteByte(0);
I2C1_MasterWriteByte("dato");
I2C1_MasterWriteByte(1);
I2C1_MasterSendStop();
I2C1_Stop();
}
```

El método de escritura contiene la iniciación del protocolo I^2C, después la dirección del dispositivo con el que se va a comunicar separando el bit menos significativo (LSB) por una coma (esto indicará si es modo escritura o modo lectura). Luego se manda 1 byte=0 y el dato a escribir (para el caso de las memorias eeprom esta sentencia se repetirá tantas veces sea necesaria), otro byte=1 y finalimente el "stop" del protocolo I^2C para finalizar la comunicación.

1.2.2. Método lectura

```
void lectura(){
I2C1_Start();
I2C1_MasterSendStart("dirección",0);
I2C1_MasterWriteByte(0);
I2C1_MasterSendRestart("dirección",1);
"variable" = I2C1_MasterReadByte(I2C1_ACK_DATA o I2C1_NAK_DATA);
I2C1_MasterSendStop();
I2C1_Stop();
}
```

El método de lectura contiene de igual manera la estructura inicial de las primeras 3 líneas de código del método de escritura, pero después de mandar el byte=0, se mandará la dirección del dispositivo en modo lectura separando el bit menos significativo (LSB) por una coma para hacer un cambio de modo escritura a modo lectura. Se leerá el valor que se desea conocer en una variable y en caso de esperar más valores se escribirá "I2C1_ACK_DATA", lo que significa que habrá otra petición para almacenar un nuevo valor en una variable distinta. Cuando se llegue a la última petición, se escribirá "I2C1_NAK_DATA", lo que significa que ya no se requerirán más datos y finalmente se mandará el "stop" del protocolo I^2C. Los métodos, las direcciones y el código a utilizar para el proyecto es el siguiente.

Módulo	Método	Dirección	Código	Instrucciones
Seguros	Escritura	0XB0	(0b1011000,0)	A= desactivar seguros. C= activar seguros. N= apagar sistema. B= salir del método de búsqueda. R= nuevo registro.
Seguros	Lectura	0XB1	(0b1011000,1)	No se requiere este método.
Arranque	Escritura	0XB2	(0b1011000,0)	R= nuevo registro. I= Activar arranque.
Arranque	Lectura	0XB3	(0b1011000,1)	Número de usuario.
Música	Escritura	0X08	(0b0000100,0)	A= play. S= stop. N= next. P= previous. Número de usuario.
Música	Lectura	0X09	(0b0000100,1)	No se requiere este método.
Eeprom 1	Escritura	0XA0	(0b1010000,0)	Posición de los asientos (2). Posición de los espejos (2 o 4). Fecha de servicio (3).
Eeprom 1	Lectura	0XA1	(0b1010000,1)	Posición de los asientos (2). Posición de los espejos (2 o 4). Fecha de servicio (3).
Eeprom 2	Escritura	0XA2	(0b1010001,0)	Posición de los asientos (2). Posición de los espejos (2 o 4). Fecha de servicio (3).
Eeprom 2	Lectura	0XA3	(0b1010001,1)	Posición de los asientos (2). Posición de los espejos (2 o 4). Fecha de servicio (3).

El caso de la búsqueda de información dentro de las memorias eeprom, se hace para recolectar los datos que controlan los motores del asiento y los espejos. Estos valores controlan por tiempo la posición e inclinación del asiento de 0 a 15 segundos, tomando como valor inicial la posición en la que el sensor inductivo es igual a 0V (SensorBase-SensorRespaldo) y de 1 a 5V para controlar por voltaje el ajuste de los espejos, tomando como posición inicial el lugar equivalente a 1V (SensorEspejoL1-SensorEspejoL2). Una vez que se conocen estos valores, se lleva cada motor a la posición inicial y se empieza a controlar por tiempo (método asiento) o por voltaje (método espejos) el moviento de cada uno de los motores.

1.2.3. Método asiento

```
void asiento(){
  while(SensorBase==1){ Rbase_Write(1);}
  Rbase_Write(0);
  for(i=0; i<=PosiciónBase; i++){
    Sbase_Write(1);
    CyDelay(1000);              }
  Sbase_Write(0);
  while(SensorRespaldo==1){ Rrespaldo_Write(1);}
  Rrespaldo_Write(0);
  for(i=0; i<=PosiciónRespaldo; i++){
    Srespaldo_Write(1);
    CyDelay(1000);              }
  Srespaldo_Write(0);
}
```

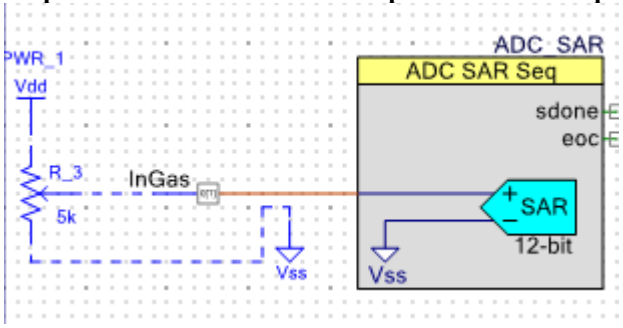
1.2.4. Método espejos

```
void espejos(){
  while(SensorEspejoL1!=1){REspejoL1 _Write(1);}
  REspejoL1_Write(0);
  while(SensorEspejoL1<Posicionespejo1){ SEspejoL1_Write(1); }
  SEspejoL1_Write(0);
  while(SensorEspejoL2!=1){REspejoL2 _Write(1);}
  REspejoL2_Write(0);
  while(SensorEspejoL2<Posicionespejo2){ SEspejoL2_Write(1); }
  SEspejoL2_Write(0);
}
```

1.2.5. Sensores

También este módulo controla las lecturas de los sensores que muestran al operador información acerca del sistema. Dependiendo del tipo de sensor será la programación y el tratamiento de sus valores. Como modelo estándar, el sensor de nivel de gasolina, el sensor de posición de espejos y el sensor de la carga de la batería, están basados en un divisor de voltaje y un convertidor "ADC SAR" (figura 47).

Figura 43: Bloque del convertidor ADC SAR para medir el tanque de gasolina



Fuente: imagen tomada de la plataforma programable PSOC creator.

En el caso del sensor de nivel de gasolina como ya cuenta con un variador de resistencia interno, sólo se conectará la toma central a una entrada analógica del micrcontrolador y mediante código se llevará a cabo el tratado de la infomación.

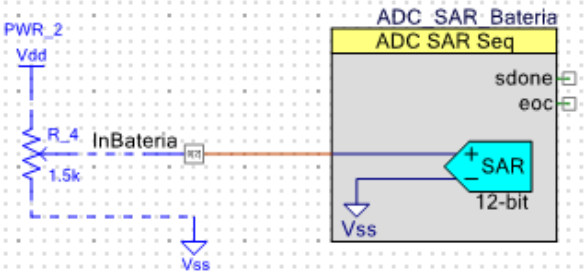
```
ADC_SAR_Start();
ADC_SAR_StartConvert();
ADC_SAR_IsEndConversion(ADC_SAR_WAIT_FOR_RESULT);
 nivelGas=ADC_SAR_GetResult16(0);
ADC_SAR_StopConvert();
 resolucionGas=nivelGas*100/4095;
```

Primero se inicia el bloque “ADC_Start()” y luego se activa el convertidor “ADC_StartCovert()” que cambia el voltaje de entrada a valores expresados en 12 bits (4095). Posteriormente se espera un tiempo hasta que la conversión haya terminado, se guarda ese valor en una variable y se detiene el convertidor “ADC_StopCovert()”. Este valor está expresado en bits, por lo que se debe convertir en porcentaje y almacenarse en una nueva variable antes de mostrarse en pantalla. Como ejemplo se supondrá que el sensor entrega 3V a su salida en una escala de 0-5V.

$$3\text{V} = \frac{3\text{V} * 4095\text{ bits}}{5\text{V}} = 2457\text{ bits} = \frac{2457\text{ bits} * 100\%}{4095\text{ bits}} = 60\%$$

El bloque convertidor ADC es el mismo a utilizar para la carga de la batería (figura 48) y para el sensor de la posición de los espejos, pero en lugar de expresarse en porcentaje, la conversión es de bits a voltaje (figura 49).

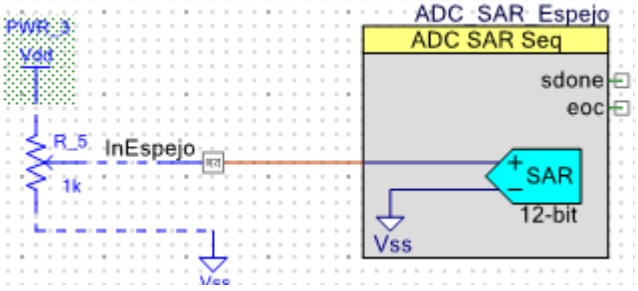
Figura 44: Bloque del convertidor ADC SAR para medir la batería



Fuente: imagen tomada de la plataforma programable PSOC creator.

```
ADC_SAR__Batería_Start();
ADC_SAR_Batería_StartConvert();
ADC_SAR_Batería_IsEndConversion(ADC_SAR_Batería_WAIT_FOR_RESULT);
porcentajeBatería=ADC_SAR_Batería_GetResult16(0);
ADC_SAR_Batería_StopConvert();
resolucionBatería=porcentajeBatería*100/4095;
```

Figura 45: Bloque del convertidor ADC SAR para medir la posición de los espejos

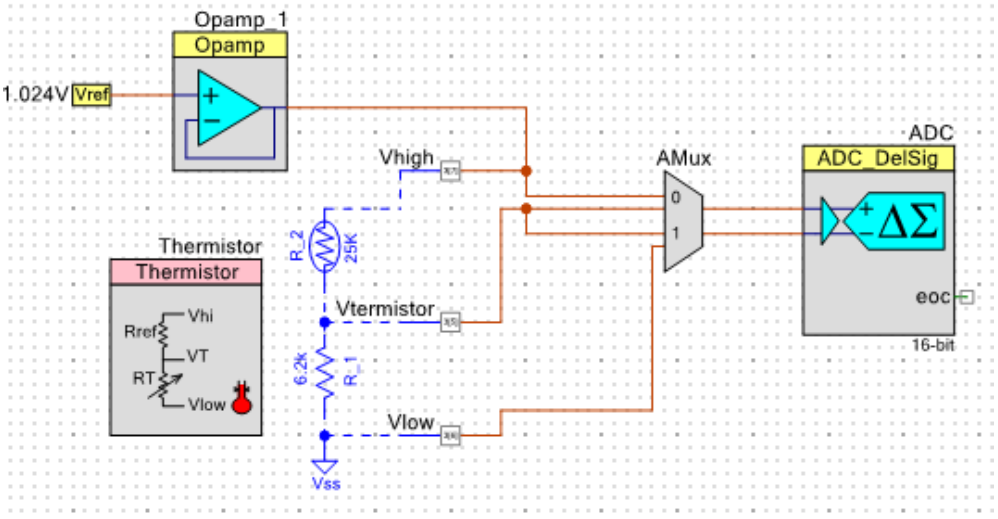


Fuente: imagen tomada de la plataforma programable PSOC creator.

ADC_SAR__Espejo_Start();
ADC_SAR_Espejo_StartConvert();
ADC_SAR_Espejo_IsEndConversion(ADC_SAR_Espejo_WAIT_FOR_RESULT);
voltajeEspejo =ADC_SAR_Espejo_GetResult16(0);
ADC_SAR_Espejo_StopConvert();
*resolucionEspejo=voltajeEspejo*5/4095;*

La lectura de temperatura con un termistor es hecha por un convertidor "ADC Delta-Sigma" (figura 50). Este convertidor tiene una resolución de 16 bits (65535), pero tiene una velocidad de procesamiento más lenta. También puede ser utilizado un convertidor "ADC SAR".

Figura 46: Bloque del convertidor ADC Delta-Sigma para medir la temperatura



Fuente: imagen tomada de la plataforma programable PSOC creator.

La característica principal de este sensor es que los voltajes que van de 0V a 1V corresponden a los grados negativos (-30^{o} a 0^{o} C) y los voltajes que van de 1V a 5V corresponden a los grados positivos (0^{o} a 120^{o} C). Para poder ajustar una escala completa que descarte los valores negativos es necesario un comparador de voltaje "AMux" .

En el canal 0 del comparador “AMux” se conecta un voltaje de referencia de 1.024V dentro del esquemático (los valores del programa no tienen 1V como opción) que pasa por un amplificador seguidor de voltaje para el ajuste de impedancias junto con una entrada del termistor que se compara con la otra entrada de voltaje variante. En el canal 1 se conecta la misma entrada de voltaje variante del termistor junto con una resistencia de 6.2 KΩ que se compara con “GND”. De este modo se crea una escala completa positiva y se ajustan las equivalencias 0^{o} C=1V=0 bits.

```
 ADC_Start();
AMux_Start();
Opamp_1_Start();
AMux_FastSelect(0);
ADC_StartConvert();
ADC_IsEndConversion(ADC_WAIT_FOR_RESULT);
Vtherm=ADC_GetResult32();
ADC_StopConvert();

AMux_FastSelect(1);
ADC_StartConvert();
ADC_IsEndConversion(ADC_WAIT_FOR_RESULT);
Vref=ADC_GetResult32();
ADC_StopConvert();
TermResis=Thermistor_GetResistance(Vref,Vtherm);
TermTemp=Thermistor_GetTemperature(TermResis);
TerTemp=TermTemp/100;
```

En el proceso de toma de temperatura, primero se inicia el convertidor ADC, el comparador de voltaje y el amplificador operacional. Después de ello se toman las lecturas para el canal 0 que determinan el voltaje en el termistor y para el canal 1 que determinan el voltaje en la resistencia de 6.2 KΩ. Con esos valores mediante la función “Thermistor_GetResistance()”, se puede conocer la resistencia que tiene el termistor y con la función “Thermistor_GetTemperature()” se convierte dicho valor a grados centígrados. Las funciones de conversión son generadas por el bloque “ADC_Delta-Sigma” con una resolución de 16 bits (65535).

A su vez y mediante un interruptor magnético de 5V conectado a una de las entradas digitales con interrupción “isr” por flanco de subida se conoce la velocidad a la que el vehículo se está moviendo (figura 51).

Figura 47: Bloque de interrupción para medir la velocidad

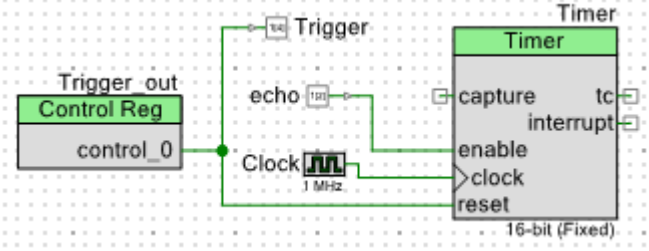
Fuente: imagen tomada de la plataforma programable PSOC creator.

```
CY_ISR(interruptVel){ //interrupción
  countVel++;
}
void velocidad(){
  almacen=countVel*60;
  countVel=0;
  KmHra=revol*1.62;
  KmHra= KmHra*180/50;
}
```

El interruptor magnético contabiliza la velocidad del vehículo cada segundo que pasa. Para ello, se multiplica por 60 el número de vueltas que la llanta ha dado durante un segundo (RPM). Después, se hace la conversión a km/hra al multiplicar las "RPM" por el diámetro de la rueda (1.62 m), por 60, entre 1000. Como el medidor de la pantalla touch tiene una resolución máxima de 50 km/hra distribuidos en 180^{o}, los km/hra calculados se multiplican por 180 y se dividen entre 50 antes de enviarse a la pantalla.

El último sensor del módulo maestro es el sensor ultrasónico de distancia. Este sensor utiliza un bloque "Timer" de 16 bits de resolución (65535) a una frecuencia de 1 MHz y también integra un control de pulsos para resetear el contador del "timer" cada cierto tiempo (figura 52).

Figura 48: Bloque de control del timer para medir distancias con sensor ultrasónico



Fuente: imagen tomada de la plataforma programable PSOC creator.

```
void Sensor_distancia(){
  while(echo_Read()==0){
     Trigger_out_Write(1);
    CyDelay(10u);
    Trigger_out_Write(0);
    CyDelay(1);
  }
  while(echo_Read()==1){}
  value_counter=65535-Timer_ReadCounter();
  distancia=value_counter/58;
}
```

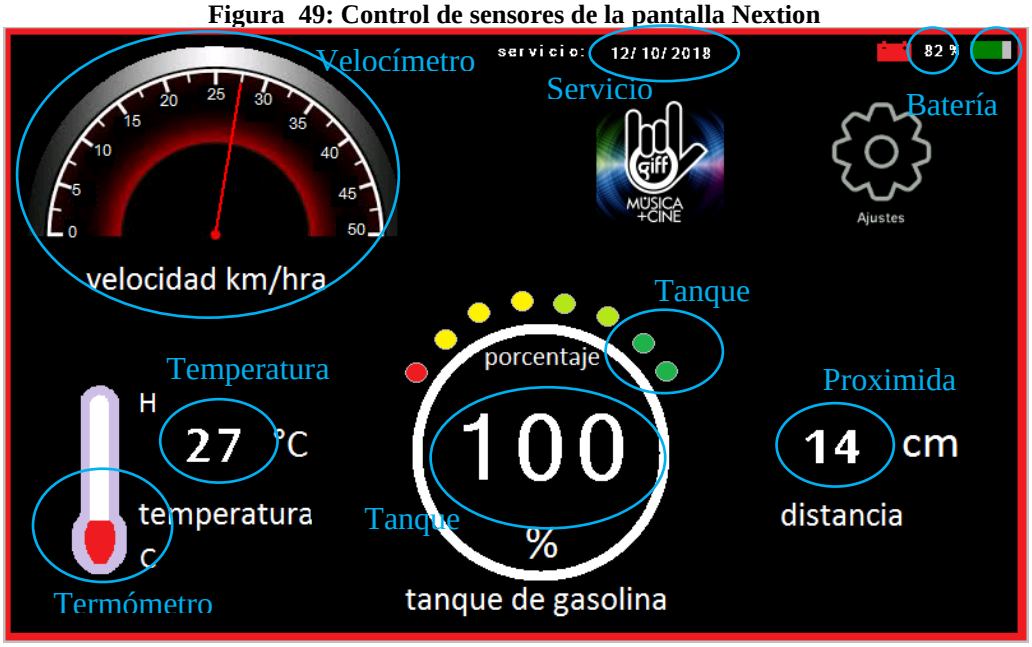
El primer ciclo “while”, que es cuando el puerto “ECHO” se encuentra en espera, activa el puerto “TRIGGER” para mandar una onda de sonido a una frecuencia de 1 MHz durante 10 microsegundos y espera 1 milisegundo de respuesta. Después de ello se activa el puerto de recepción “ECHO” al mismo tiempo que se inicia un contador descendiente de 16 bits (65535). Si se detecta una señal de respuesta, se convierte el valor detectado en un valor ascendente (65535-valor leído). Este valor tiene una equivalencia de 10 uS = 58 cm, por lo que se divide entre 58 para establecer la distancia real. Si el contador llega hasta “0” se determina que no existió ningún objeto.

Lo primero que hace el módulo maestro al encenderse es mandar la instrucción “salir del método de búsqueda” al módulo de seguridad de puertas (método escritura), luego obtiene el número de usuario que ingresó al sistema del módulo de arranque (método lectura), busca la información del usuario en la memoria eeprom correspondiente (método lectura) y finalmente toma lecturas de los sensores de manera consecutiva.

1.3. Pantalla touch Nextion

La pantalla touch de marca nextion tiene una comunicación UART con el microcontrolador PSOC5 del módulo de control maestro a través de los puertos “RX” y “TX”. Para el envío de datos recolectados por los sensores (figura 53), se requiere seguir un protocolo de ordenamiento.

Figura 49: Control de sensores de la pantalla Nextion



Fuente: imagen tomada de la plataforma de diseño Nextion editor.

Los valores de los sensores deben ser separados número por número en variables independientes a través de un ciclo “if” como se muestra a continuación en un ejemplo para la temperatura.

```
TerTemp=27;
centenasTem=0; decenasTem=0; unidadesTem=0;
if(TerTemp>99){centenasTem=TerTemp/100; TerTemp=TerTemp-(centenasTem*100);}
if(TerTemp>9){decenasTem=TerTemp/10; TerTemp=TerTemp-(decenasTem*10);}
if(TerTemp>0){unidadesTem=TerTemp;}
```

En este caso la temperatura es de 27^{o} C. Como número más grande posible están las centenas, por lo que se busca si existe un número mayor a 99 que se divide entre 100 para almacenarse en la variable “centenas”. Después de ello se restan las centenas encontradas a la variable temperatura para dejar un número comprendido entre 0 y 99. Lo mismo sucede con las decenas y las unidades.

Centenas=0; Decenas=2; Unidades=7;

Con esa estructura se mandan los valores en ese orden consecutivo. La pantalla nextion omitirá las variables equivalentes a “0” hasta que reciba un número significativo. En la siguiente tabla se encuentran los elementos a controlar con el código que se debe mandar.

Tipo	Elemento	Nombre	Valores	Estructura de código
Velocidad	Velocímetro	“z0”	0-180	UART_PutString("z0.val="); UART_PutChar(centenas); UART_PutChar(decenas); UART_PutChar(unidades);
Temperatura	Termómetro	“j0“	0-100	UART_PutString("j0.val="); UART_PutChar(centenas); UART_PutChar(decenas); UART_PutChar(unidades);
	Números	“n0”	0-100	UART_PutString("n0.val="); UART_PutChar(centenas); UART_PutChar(decenas); UART_PutChar(unidades);
Distancia	Números	“n1”	0-400	UART_PutString("n1.val="); UART_PutChar(centenas); UART_PutChar(decenas); UART_PutChar(unidades);
Tanque	Porcentaje	“n2”	0-100	UART_PutString("n2.val="); UART_PutChar(centenas); UART_PutChar(decenas); UART_PutChar(unidades);
	Luces	“p3-p10”	0-1	UART_PutString("vis Nombre,Valor");

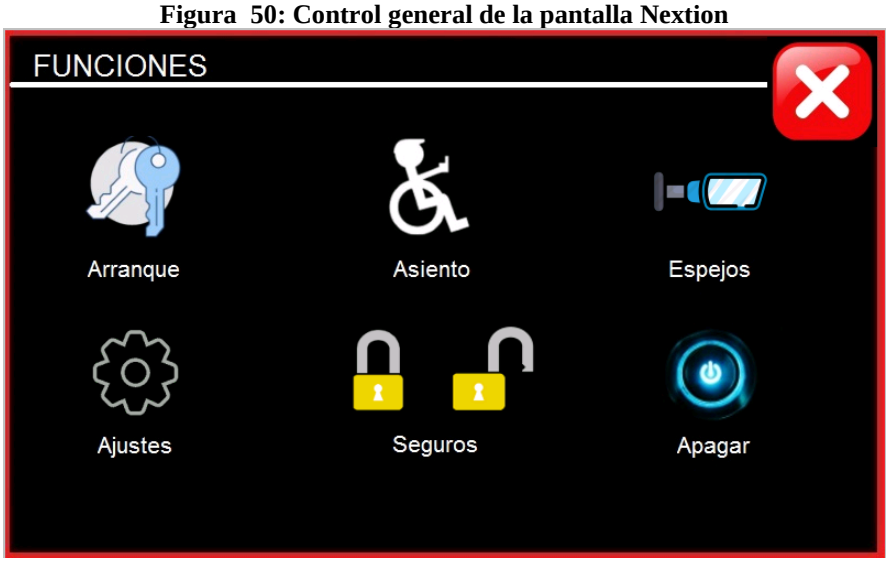
Batería	Números	“n3”	0-100	UART_PutString("n3.val="); UART_PutChar(centenas); UART_PutChar(decenas); UART_PutChar(unidades);
	Batería	“j1”	0-100	UART_PutString("j1.val="); UART_PutChar(centenas); UART_PutChar(decenas); UART_PutChar(unidades);
Servicio	Día	“n4”	0-31	UART_PutString("n4.val="); UART_PutChar(decenas); UART_PutChar(unidades);
	Mes	“n5”	0-12	UART_PutString("n5.val="); UART_PutChar(decenas); UART_PutChar(unidades);
	Año	“n6”	0-9999	UART_PutString("n6.val="); UART_PutChar(u. millar); UART_PutChar(centenas); UART_PutChar(decenas); UART_PutChar(unidades);

El formato de envio de datos debe contener el nombre del elemento que se va a controlar, seguido del comando “.val=” (todo esto dentro de la misma cadena de texto). Después irán los valores a mostrarse en pantalla, separados en unidades de millar, centenas, decenas y unidades, más la sentencia “0xff” repetida 3 veces como se muestra a continuación.

UART_PutChar(0xff);
UART_PutChar(0xff);
UART_PutChar(0xff);

Además de este tipo de elementos, la pantalla nextion cuenta con botones para realizar tareas programadas por los métodos creados en el PSOC5. Las funciones de pantalla de control general (figura 54) son las que controlan los procesos más importantes del proyecto y al presionarse algún botón, también envian una secuencia de código por el protocolo UART de la pantalla al microcontolador.

Figura 50: Control general de la pantalla Nextion



Fuente: imagen tomada de la plataforma de diseño Nextion editor.

Para ello se requiere una interrupción por recepción para el protocolo UART dentro del microcontrolador que recibirá un arreglo de 7 números que indica el proceso a iniciar. Esta serie de datos tiene la siguiente estructura.

Tipo de botón	No. de ventana	No. de elemento	presionar/ soltar	0XFF	0XFF	0XFF

La cadena recibida será almacenada byte por byte por la interrupción en un arreglo de 7 direcciones. Dado que el único valor que se requiere es el contenido en la segunda y la tercera posición del arreglo (número de ventana y valor "id" del botón), serán almacenados en variables para posteriormente realizar sus funciones asignadas.

1.3.1. Interrupción UART

```
CY_ISR(interruptRX){ // interrupción para recepción UART
  datoRX[i]=UART_GetByte();
  i++;
  if(i==7){
    pantalla=datoRX[1];
    estado=datoRX[2];
    i=0;
  }
 }
 if(estado=="no. de elemento" && pantalla=="no. de ventana"){          //opciones de procesos
    pantalla=estado=0;
   dato="código a enviar";
  dirección="dirección del dispositivo al que se mandará";
  escribir();                                                          //método escritura
}
```

En la siguiente tabla están las funciones de control general con su respectivo código.

Nombre	Función	D.	Direc.	Prot.	Método	Código

Arranque	Activa el botón de arranque.	"I"	0XB2	I^2C	Escritura	if(estado==2 && pantalla==3){
						pantalla=estado=0;
						dato="I";
						dirección=0XB2;
						escribir();
						}

Asiento	Ajusta los asientos.	--	--	--	Asiento	if(estado==3 && pantalla==3){
						pantalla=estado=0;
						asiento();
						}

Espejos	Ajusta los espejos.	--	--	--	Espejos	if(estado==4 && pantalla==3){
						espejos();
						}

Ajustes	Pantalla de ajustes.	--	--	--	--	--

Seguros	Poner seguros.	"C"	0XB0	I^2C	Escritura	if(estado==7 && pantalla==3){
						pantalla=estado=0;
						dato="C";
						dirección=0XB0;
						escribir();
						}

	Quitar seguros.	"A"	0XB0	I^2C	Escrutura	if(estado==6 && pantalla==3){
						pantalla=estado=0;
						dato="A";
						dirección=0XB0;
						escribir();
						}

Apagar	Apagar sistema.	"N"	0XB0	I^2C	Escritura	if(estado==8 && pantalla==3){
						pantalla=estado=0;
						dato="N";
						dirección=0XB0;
						escribir();
						}

Estas instrucciones determinan el dato que será mandado a través del protocolo I^2C hacia otro módulo en la red que controle dichos elementos. De esta forma el módulo maestro controlará todo el sistema sin la necesidad de tener todas las salidas y entradas existentes conectadas a él. Con los botones del asiento y los espejos se iniciarán los métodos correspondientes en el mismo módulo de control maestro. El botón de "ajustes" cambiará la pantalla de control general por la pantalla de ajustes (figura 55).

Figura 51: Control de ajustes de la pantalla Nextion

Fuente: imagen tomada de la plataforma de diseño Nextion editor.

En esta interfaz están los controles de movimiento del asiento y de los espejos, cambio de huellas digitales y un botón para almacenar las nuevas posiciones dentro de las memorias eeprom (método escritura).

La primera parte del código del respaldo y de los espejos permite mover los motores a deseo propio, la segunda calcula las nuevas posiciones y la tercera almacena los valores en memoria. Como ejemplo se pondrá un caso de ajuste del respaldo y otro para la posición vertical de los espejos.

1.3.2. Respaldo

```
if(estado==3 && pantalla==1){   // botón aumento en tiempo de respaldo.
   Srespaldo_Write(1);
   CyDelay(1000);
   Srespaldo _Write(0);
}
if(estado==7 && pantalla==1){    // botón guardar con todas las funciones.
    tiempoR=0;
    while(SensorRespaldo!=0){
      Rrespaldo _Write(1);
      CyDelay(1000);
      tiempoR=++;
   }
   Srespaldo_Write(0);
   escritura("dirección","tiempoR");
}
```

Al presionar el botón de avance, el respaldo se moverá un segundo hacia adelante (estas instrucciones son creadas para cada botón de los motores del asiento). Posteriormente el botón "guardar" (estado==7 && pantalla==1) iniciará el contador "tiempoR=0" y en lapsos de 1 segundo moverá el respaldo hasta que sea detectado por el sensor inductivo. Al mismo tiempo el contador "tiempoR" irá en aumentanto y finalmente se mandará llamar el método escritura para almacenar el nuevo valor registrado en las memorias eeprom.

1.3.3. Espejo

```
if(estado==8 && pantalla==1){   // botón aumento en espejo vertical.
  Vespejo1=0;
  If(SensorEspejoL1<5){ Vespejo1=SensorEspejoL1+1;}
  else{ Vespejo1=5;}
  while(SensorEspejoL1<Vespejo1){SEspejoL1 _Write(1);}
  SEspejoL1_Write(0);
  if(estado==7 && pantalla==1){    // botón guardar con todas las funciones
    escritura("dirección"," SensorEspejoL1");
  }
}
```

Si se presiona el botón de aumento vertical de los espejos, se checará si aún existe un valor mayor al actual. De ser así se asignará el valor de la posición actual +1 a una variable o en caso contrario, se asignará a esa variable el valor máximo permitido (5). Después se moverán los motores de los espejos hasta que lleguen a la posición indicada. Finalmente con el botón "guardar" se mandará llamar el método escritura para almacenar el nuevo valor registrado. La siguiente tabla muestra las funciones de control de los motores.

Nombre	Función	Dato	Código

Respaldo	Aumento de tiempo del respaldo	tiempoR	if(estado==3 && pantalla==1){
			Srespaldo_Write(1);
			CyDelay(1000);
			Srespaldo _Write(0);
			}

Respaldo	Desminución de tiempo del respaldo	tiempoR	if(estado==2 && pantalla==1){
			Rrespaldo_Write(1);
			CyDelay(1000);
			Rrespaldo _Write(0);
			}

Asiento	Aumento de tiempo del asiento	tiempoA	if(estado==4 && pantalla==1){
			Sbase_Write(1);
			CyDelay(1000);
			Sbase _Write(0);
			}

Asiento	Desminución de tiempo del asiento	tiempoA	if(estado==5 && pantalla==1){
			Rbase_Write(1);
			CyDelay(1000);
			Rbase _Write(0);
			}

Espejo vetical	Aumento de posición vertical	Vespejo1	if(estado==8 && pantalla==1){
			Vespejo1=0;
			If(SensorEspejoL1<5){
			Vespejo1=SensorEspejoL1+1;
			}else{
			Vespejo1=5;
			}
			while(SensorEspejoL1<Vespejo1){
			SEspejoL1 _Write(1);
			}
			SEspejoL1_Write(0);
			}

Espejo vertical	Disminución de posición vertical	Vespejo1	if(estado==10 && pantalla==1){
			Vespejo1=0;
			If(SensorEspejoL1>1){
			Vespejo1=SensorEspejoL1-1;
			}else{
			Vespejo1=1;
			}
			while(SensorEspejoL1>Vespejo1){
			REspejoL1 _Write(1);
			}
			REspejoL1_Write(0);
			}

Espejo horizontal	Aumento de posición horizontal	Vespejo2	if(estado==11 && pantalla==1){
			Vespejo2=0;
			If(SensorEspejoL2<5){
			Vespejo2=SensorEspejoL2+11;
			}else{
			Vespejo2=5;
			}
			while(SensorEspejoL2<Vespejo2){
			SEspejoL2 _Write(1);
			}
			SEspejoL2_Write(0);
			}

Espejo horizontal	Disminución de posición horizontal	Vespejo2	Vespejo2=0;
			If(SensorEspejoL2>1){
			Vespejo2=SensorEspejoL2-1;
			}else{
			Vespejo2=1;
			}
			while(SensorEspejoL2>Vespejo2){
			REspejoL2 _Write(1);
			}
			REspejoL2_Write(0);
			}

Por su parte, los logos de las huellas digitales mandan la instrucción “R” al módulo de seguridad de puertas (0XB0) o al módulo de arranque (0XB2) para registrar una nueva huella. La siguiente tabla muestra las funciones de cambio de huella y del botón “guardar”.

Nombre	Función	D.	Direc.	Prot.	Método	Código

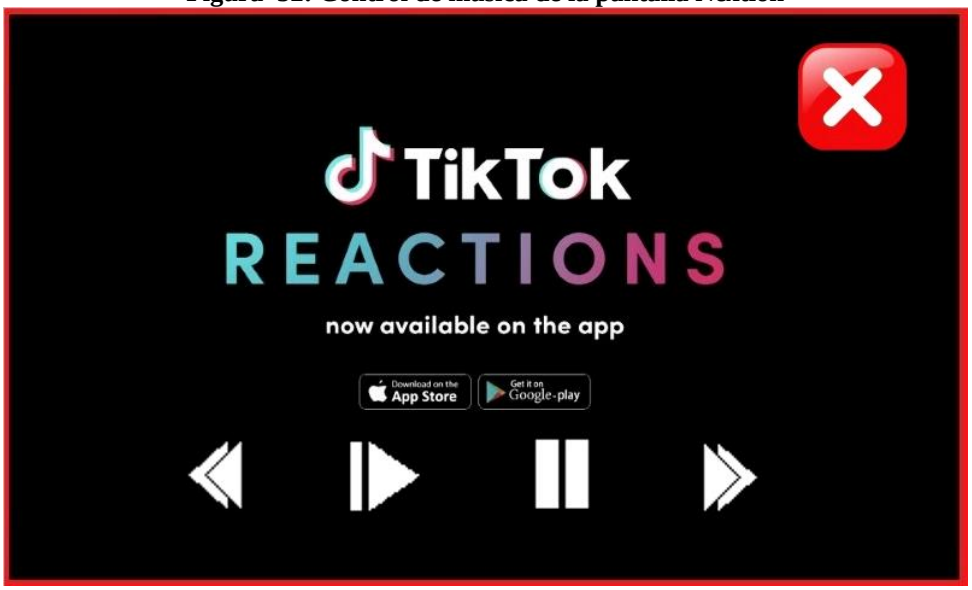
Huella 1	Módulo de seguros	“R”	0XB0	I^2C	Escritura	if(estado==6 && pantalla==1){
						pantalla=estado=0;
						dato=”R”;
						dirección=0XB0;
						escribir();
						}

Huella 2	Módulo de arranque	“R”	0XB2	O2C	Escritura	if(estado==12 && pantalla==1){
						pantalla=estado=0;
						dato=”R”;
						dirección=0XB2;
						escribir();
						}

Guardar	Guardar cambios en memoria	tiempoR
		tiempoA
		Vespejo1
		Vespejo2
			0XA0
			0XA2
				I^2C	Escritura	if(estado==7 && pantalla==1){
						tiempoR=0;
						while(SensorRespaldo!=0){
						Rrespaldo _Write(1);
						CyDelay(1000);
						tiempoR=++;
						}
						Rrespaldo_Write(0);
						tiempoA=0;
						while(SensorBase!=0){
						Rbase _Write(1);
						CyDelay(1000);
						tiempoA=++;
						}
						Rbase_Write(0);
						escritura(“dirección”,”variables”);
						}

La última pantalla tiene los controladores de música. En esta pantalla (figura 56) están todos comandos que se mandan al módulo de control de música para manipular el reproductor MP3.

Figura 52: Control de música de la pantalla Nextion



Fuente: imagen tomada de la plataforma de diseño Nextion editor.

La siguiente tabla muestra el código que cada botón utiliza para controlar la música.

Nombre	Función	D.	Direc.	Prot.	Método	Código

Play	Módulo de música	"A"	0X08	I^2C	Escritura	if(estado==3 && pantalla==2){
						pantalla=estado=0;
						dato="A";
						dirección=0X08;
						escribir();
						}

Stop	Módulo de música	"S"	0X08	I^2C	Escritura	if(estado==4 && pantalla==2){
						pantalla=estado=0;
						dato="S";
						dirección=0X08;
						escribir();
						}

Next	Módulo de música	"N"	0X08	I^2C	Escritura	if(estado==5 && pantalla==2){
						pantalla=estado=0;
						dato="N";
						dirección=0X08;
						escribir();
						}

Previous	Módulo de música	"P"	0X08	I^2C	Escritura	if(estado==2 && pantalla==2){
						pantalla=estado=0;
						dato="P";
						dirección=0X08;
						escribir();
						}

1.3.4. Gestión de música

El módulo de control de música está basado en un microcontrolador PSOC5 con un bloque de comunición I^2C en modo esclavo y un bloque de comunicación UART para controlar el reproductor MP3 "WTV020". Al encender este módulo entrará en un ciclo "while" hasta que el módulo maestro le informe el número de usuario que ingresó al sistema. Para poder recibir los códigos de control es necesario tener una interrupción por el protocolo I^2C.

1.3.5. Interrupción I^2C

```
I2C2_Start();
I2C2_SlaveInitWriteBuf((uint8 *) wrBuf, 1);
if(0u != (I2C2_SlaveStatus() & I2C2_SSTAT_WR_CMPLT)){
    I2C2_SlaveClearWriteStatus();
    if(wrBuf[0]==”valor recibido”){
        “método de envio”
    }
    wrBuf[0]=”0”;
    I2C2_SlaveClearWriteBuf();
}
```

Se debe iniciar el bloque I^2C y la interupción del modo esclavo junto con la variable donde se guardará el dato recibido. Esta interrupción puede ir dentro del método "main" ya que no existen muchos procesos que desarrollar para este módulo. Una vez que se recibió un dato del módulo maestro, la variable recibida será comparada con las posibles letras permitidas por el sistema para escoger alguno de los comandos del reproductor de música MP3 antes de limpiar el buffer. Los posibles comandos a enviar al reproductor MP3 son las siguientes.

```
char buffer_data[10];
char playInicio1[10]={0x7E,0xFF,0x06,0x3F,0x00,0x00,0x00,0xFE,0xBC,0xEF};
char pausar[10]=    {0x7E,0xFF,0x06,0x0E,0x00,0x00,0x00,0xFE,0xED,0xEF};
char next[10]=      {0x7E,0xFF,0x06,0x01,0x00,0x00,0x01,0xFE,0xF9,0xEF};
char previous[10]=  {0x7E,0xFF,0x06,0x02,0x00,0x00,0x01,0xFE,0xF8,0xEF};
```

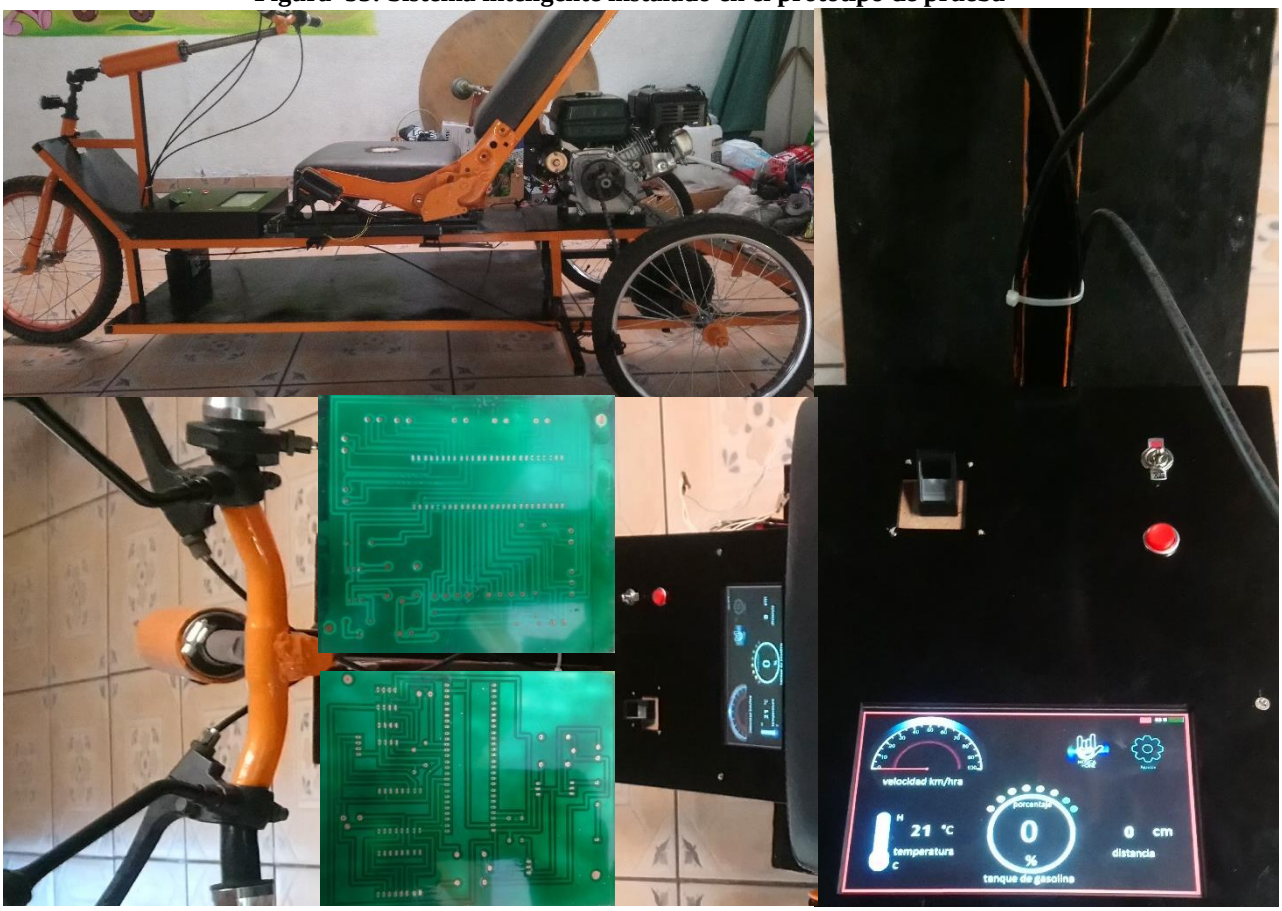
Estos valores llenarán el arreglo “buffer_data[]” y serán transmitidos byte por byte en un ciclo “for” por los puertos “RX” y “TX” al reproductor de música.

1.3.6. Método de envio

```
void send_buffer(){
   for(i=0;i<10;i++){
      UART_PutChar(buffer_data[i]);
   }
}
```

Finalmente se ha montado y programado sobre el vehículo go-kart todo el sistema embebido con sus respectivos protocolos, sensores y métodos establecidos que engloban al sistema de control personalizado e inteligente (figura 57).

Figura 53: Sistema inteligente instalado en el prototipo de prueba



Fuente: fotografía tomada por el autor.

Conclusiones

Los sistemas embebidos son tecnologías que brindan grandes beneficios de rapidez y precisión gracias a sus diseños, generan una programación muy sencilla, los errores ocasionados por una mala estructura se pueden identificar fácilmente y las rutinas establecidas se realizan con más frecuencia. Algunos aspectos importantes que se deben de tomar en cuenta a la hora de su implementación son las características propias de cada microcontrolador, ya que de ellos dependerá la velocidad de trabajo, la precisión, la resolución de valores y la compatibilidad con otros elementos.

El modelo perfecto para desarrollar un sistema embebido es con el uso de una FPGA por sus propiedades de trabajo. Este tipo de tarjeta de desarrollo tiene la característica de poder encapsular la estructura de muchos microcontroladores dentro de ella. También cuenta con velocidades de trabajo más rápidas que las velocidades de un microcontrolador, permite llevar a cabo varios procesos al mismo tiempo y brinda una configuración personalizada de puertos especiales. Las desventajas de estos modelos son que aún no se conoce el algoritmo para encapsular a los microcontroladores, su costo es muy caro y no cuentan con un lenguaje estandarizado. De momento las FPGA son configuradas de forma compleja y poco dúctil en un lenguaje "HDL" (Hardware Description Language).

Aunque los microcontroladores no cuentan con todas las propiedades de las FPGA, ha sido viable la implementación de este proyecto como sistema embebido ya que cumple con los requerimientos esenciales de trabajo: protocolos de comunicación, velocidades de trabajo adecuadas, sencillez de programación, confianza de trabajo y compatibilidad entre elementos. El proyecto ha generado un buen ahorro económico y con la creación de una red interna de comunicación I^2C entre diversos fabricantes de microcontroladores, se han igualado las ventajas de llevar a cabo procesos en paralelo de una FPGA.

En la cuestión práctica se cumplieron todos los objetivos establecidos por el proyecto, pero cabe mencionar que por cuestiones de rendimiento y seguridad no se recomienda cambiar todos los botones y manivelas existentes, ya que ciertos elementos deben de ser de acceso rápido para el conductor en caso de emergencia. Otra situación a considerar es el uso de una llave, aunque esta puede ser eliminada, debe de seguir existiendo un cortacorriente para cuidar la carga de la batería.

Aparte de todo el conocimiento que este proyecto generó, se deben mencionar otros aprendizajes importantes obtenidos en el transcurso de su investigación.

Para las primeras pruebas de trabajo se utilizaron dispositivos AVR con la plataforma de programación CodeVision (lenguaje C). Desafortunadamente estos microcontroladores no pudieron ser integrados en la misma red I^2C junto a otros fabricantes. Para empezar, el reloj interno de los AVR es muy inestable y aunque la plataforma permite configurar su velocidad de trabajo, esta no es tan precisa y por lo tanto las frecuencias no son las correctas. Para solucionar este problema es indispensable un oscilador externo que se configure con los bits "CKSEL" en el programador. Aún solucionando este problema, no es posible integrar la red I^2C, ya que el fabricante AVR utiliza el registro "TWDR" tanto para mandar un dato como para recibir otro y el registro "TWAR" para mandar la dirección del esclavo. En cambio los fabricantes Microchip y Cypress usan el mismo registro para mandar las direcciones o mandar datos y usan otro registro independiente para recibir valores. Debido a esto, no ha sido posible tener un entendimiento en la comunicación de los 3 fabricantes.

Otro aspecto notorio fue al usar un arduino. Esta plataforma es una herramienta de gran ayuda para principiantes que permite realizar proyectos de manera muy sencilla y que ahorra mucho tiempo de estudio. Desde un punto de vista personal la programación de arduino tiene muchos puntos a favor, pero en este caso, su aplicación no fue viable a pesar de sus grandes ventajas. La razón es que este proyecto involucra demasiados elementos como son los sensores, las pantallas, los módulos y los protocolos que consumen muchos recursos con información innecesaria debido a sus librerías. Una librería de arduino facilita la programación, pero a cambio de esto, consume mucho espacio de memoria ya que contiene todos los controladores de todos los modelos existentes de un mismo dispositivo (tal fue el caso del lector de huellas digitales).

Bibliografía

Arenas Mas, M. (Junio de 2008). Diseño y implementación de un sistema de adquisición de aceleraciones con procesamiento mediante microcontrolador. Sevilla, España: Universidad de Sevilla.

Crespo, W. (9 de Febrero de 2011). *¿Qué es la automatización industrial?* Obtenido de Automatización Industrial : https://automatizacionindustrial.wordpress.com/2011/02/09/queeslaautomatizacionindustrial/

FENERCOM. (2007). *La domótica como solución del futuro.* Madrid, España: Consejo de Economía e Innovación Tecnológica.

Galiana Linares, A. N. (junio de 2005). *Sistemas embebidos.* Obtenido de Universidad Politecnica de Valencia: http://server-die.alc.upv.es/asignaturas/paeees/2005-06/A07%20-%20Sistemas%20Embebidos.pdf

Garcia, V. (2 de mayo de 2012). *Introduccion al i2c bus*. Obtenido de (EPA) Electronica practica aplicada: http://www.diarioelectronicohoy.com/blog/introduccion-al-i2c-bus

Guadarrama Fortoul, D. (25 de Octubre de 2012). Sistema de acceso al personal mediante la autentificación de huella dactilar. *Tesis de licenciatura*. CDMX, México: Instituto Politécnico Nacional.

Instituto Nacional de Tecnologías de la Comunicación. (1 de Febrero de 2013). *Seguridad en las contraseñas.* Obtenido de Universidad de la Rioja: https://www.unirioja.es/servicios/si/seguridad/difusion/politica_contrasenas.pdf

Kharsansky, A. (31 de mayo de 2013). Diseño e implementación de un sistema embebido de control de actitud para aeronaves no tripuladas. *Tesis de licenciatura*. Buenos Aires, Argentina: Universidad de Buenos Aires.

Lopéz Pérez, E. (2008). Protocolo SPI (Serial Peripherical Interface). *Ingeniería en microcontroladores*. CDMX, México: i-micro.

Lopéz Pérez, E. (2008). Protocolo USB (UNIVERSAL SERIAL BUS). *Ingeniería en microcontroladores*. CDMX, México: i-micro.

Mayné, J. (Febrero de 2009). Sistemas de comunicación. *Documento de trabajo* . Silica.

PAC. (2011). *Controladores industriales de diseño de alto nivel.* Obtenido de Ingenieria de los sistemas embebidos (Hardware-Hr):

http://www.ieec.uned.es/investigacion/Dipseil/PAC/archivos/Informacion_de_referencia_ISE5_1_1.pdf

Parra Reynada, L. (2012). *Microprocesadores.* Estado de Mexico: Red tercer milenio.

periodismo independiente. (27 de Agosto de 2017). *Animal politico*. Obtenido de http://www.animalpolitico.com/2017/04/autos-robos-modelos-mexico/

Quispe, O. (16 de Abril de 2017). *Tarjetas para desarrollo de hardware*. Obtenido de Lightpath: http://www.lightpath.io/tarjetas-de-desarrollo/

R. Ganorkar, S. (16 de Febrero de 2007). *Iris Recognition: An Emerging Biometric Technology.* Obtenido de wseas: http://www.wseas.us/e-library/conferences/2007corfu/papers/540-292.pdf

Raspberry pi foundation. (2017). *Raspberrypi*. Obtenido de https://www.raspberrypi.org/

Salas Arriarán, S. (2015). *Todo sobre sistemas embebidos.* Lima, Perú: Universidad Peruana de Ciencias Aplicadas.

Siemens . (2017). *Siemens México*. Obtenido de https://www.siemens.com/mx/es/home.html

Valencia González, A., Berríos Miranda, C., & Carreño Bonilla, C. (5 de Octubre de 2012). Apuntes del Taller de microcontroladores ARM7TDMI LPC2148. *Documento de trabajo*. Valparaíso, Chile: Pontificia Universidad Católica de Valparaíso.

Valencia Puentes, D. E. (2010). Desarrollo de sistema embebido en tiempo real. *Tesis de maestria*. Bogotá, Colombia: Universidad Nacional de Colombia.

Valencia, D. E. (2010). Desarrollo de sistema embebido en tiempo real. *Tesis de maestria*. Bogota, Colombia: Universidad Nacional de Colombia.

Vásquez Gómez, J. B. (2012). *Arquitectura de computadoras I.* Tlanepantla, Estado de México: Red tercer Milenio.

Índice de figuras