UNIVERSIDAD TECNOLÓGICA EQUINOCCIAL

FACULTAD DE CIENCIAS DE LA INGENIERÍA

CARRERA DE INGENIERÍA AUTOMOTRIZ

ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE RECONOCIMIENTO DE VOZ INTELIGENTE, PARA MEJORAR LA INTERACTIVIDAD Y CONFORT DE UN ACOMPAÑANTE CUADRIPLÉJICO, AL CONTROLAR LOS DIFERENTES ACCESORIOS DE UN VEHÍCULO.

TRABAJO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO AUTOMOTRIZ

JOSÉ LUIS NAVARRETE NAVARRETE

DIRECTOR: ING. ALEXANDER PERALVO - MSC

Quito, Febrero, 2015

DECLARACIÓN

Yo **JOSÉ LUIS NAVARRETE NAVARRETE**, declaro que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

La Universidad Tecnológica Equinoccial puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normativa institucional vigente.

_____

José Luis Navarrete Navarrete

C.I. 1719817734

CERTIFICACIÓN

Certifico que el presente trabajo que lleva por título "**Análisis, diseño e implementación de un sistema de reconocimiento de voz inteligente, para mejorar la interactividad y confort de un acompañante cuadripléjico, al controlar los diferentes accesorios de un vehículo**", que para aspirar al título de Ingeniero Automotriz fue desarrollado por José Luis Navarrete Navarrete, bajo mi dirección y supervisión, en la Facultad de Ciencias de la Ingeniería; y cumple con las condiciones requeridas por el reglamento de Trabajos de Titulación artículos 18 y 25.

_____

Ing. Alexander Peralvo – Msc.

DIRECTOR DEL TRABAJO

C.I. 1718133448

DEDICATORIA

El presente trabajo de investigación está dedicado a mis padres Rosa y Gustavo, quienes con su amor y compresión me dieron la oportunidad de alcanzar esta anhelada profesión.

A mis hermanos quienes me motivaron al fascinante mundo de la computación.

A mis sobrinos quienes son la alegría del hogar.

Con mucho cariño se los dedico...

José Luis

AGRADECIMIENTOS

A Dios por ser mi guía y fortaleza en todo momento especialmente en aquellos días difíciles; así como también en los mejores. Gracias Señor por ayudarme a cruzar este río sin desfallecer en el trayecto, alcanzando así una de mis metas más anheladas y llenándome de orgullo al ver que me voy sintiendo realizado.

A mis padres, aquellos seres humanos maravillosos que con gran esfuerzo y sacrificio me dieron la oportunidad de estudiar y ser una persona de bien. Se los agradezco eternamente porque no hubiese llegado hasta aquí sin su apoyo y afecto incondicional.

A mi hermano Gustavo quien con sus sabios consejos y absoluto apoyo ha sido un segundo padre en mi vida. Y como olvidarme de Galo quien con su papel de hermano mayor siempre ha demostrado ser una persona digna de mi confianza y admiración.

Un especial agradecimiento a mi cuñada Verito, quien con su ayuda se me otorgó la oportunidad de obtener una BECA en esta distinguida universidad y de esta manera salir adelante.

A mi director y profesor Alexander Peralvo quien con su conocimiento y apoyo incondicional me ha formado como un profesional de bien.

A la prestigiosa Universidad Tecnológica Equinoccial.

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE ANEXOS

RESUMEN

El presente proyecto de investigación tiene como finalidad diseñar e implementar un sistema de reconocimiento de voz basado en la utilización de un hardware y software libre, para mejorar la interactividad y confort de una persona discapacitada física y/o visualmente, al controlar los diferentes actuadores electrónicos de un vehículo por medio de comandos de voz. Además es importante indicar que este sistema tiene la capacidad de incrementar en cierta medida la seguridad y el confort del conductor así como la de los acompañantes, al evitar la manipulación directa con los diferentes accesorios en el interior del vehículo. El primer apartado se refiere a la introducción del proyecto, donde se describe brevemente la importancia que tiene la domótica en el área automotriz al mejorar la seguridad, el confort, la accesibilidad y la interacción de las personas con las máquinas. Además se plantea el propósito de la investigación así como los objetivos, el planteamiento del problema, la justificación, el alcance y el método a utilizar por la misma. En el segundo apartado se estudiarán los conceptos básicos en lo que respecta a la electrónica, electricidad y programación C; de manera que los conocimientos adquiridos en base al marco teórico son empleados durante el desarrollo del sistema. En el tercer apartado se expone la metodología utilizada, donde se da a conocer las herramientas analíticas empleadas así como la arquitectura domótica utilizada durante el desarrollo del proyecto, partiendo desde el diseño experimental del sistema hasta su implementación final en el vehículo. En el cuarto apartado se realiza un análisis de resultados obtenidos en base a los costos efectuados, los antecedentes encontrados, las actividades distractoras durante la conducción y los beneficios obtenidos por este sistema al mejorar la interactividad y el confort de una persona con discapacidad. Finalmente en el quinto apartado se enuncian las conclusiones a las que se llegó con el estudio, respondiendo al objetivo general y específicos del proyecto; además se da a conocer las recomendaciones a tomar en cuenta por el lector, con la finalidad de evitar algún tipo de contratiempo durante la investigación.

ABSTRACT

The present project of investigation aims to design and implement a speech recognition system based on the use of a free hardware and free software, to enhance interactivity and comfort of a disabled person physical and/or visually, to control the different electronic actuators of a vehicle through voice commands. Moreover is important to indicate that this system has the ability to increase to some extent safety and driver comfort as well as the passengers, to avoid direct handling with the different accessories in the vehicle. The first section refers to the introduction of the project, where is briefly described the importance of the domotic in the automotive area to improve safety, comfort, accessibility and interaction of people with machines. Furthermore is posed the purpose of the research as well as the objectives, the problem statement, the justification, the scope and the method used for the same. In the second section will be studied the basic concepts in relation to the electronic, electricity and programming C; so that the knowledge acquired based on the theoretical framework are used during system development. The third section presents the methodology used, where it discloses the analytical tools used as well as the domotic architecture used during the project, starting from the experimental design of the system until its final implementation in the vehicle. The fourth section is performed an analysis of results in based on the costs incurred, the background found, the distracting activities while driving and the benefits obtained by this system to enhance interactivity and comfort of a disabled person. Finally in the fifth section is enunciated the conclusions it reached with the study, responding to the general and specific objectives of the project; also it discloses the recommendations to consider by the reader, in order to prevent some mishap during the investigation.

1. INTRODUCCIÓN

En una sociedad donde la tecnología avanza a pasos agigantados, al parecer es posible colaborar en esta carrera contra el tiempo; puesto que la investigación da lugar a que se descubran nuevos campos, se mejoren otros o simplemente se logre fomentar el conocimiento individual. Conocer el funcionamiento de las cosas es algo que el hombre se ha planteado desde el inicio de los tiempos; hoy en día nos enfrentamos a una realidad donde abundan la automatización, la interacción de las personas con las máquinas, la electrónica, la mecánica y la programación.

El término domótica proviene del latín “domus” que significa casa y del griego “tica” que funciona por sí sola, la primera definición de este concepto nació en los años setenta en Francia con la palabra “domo tique”, que hacía referencia al progreso conjunto de tres grandes áreas de la tecnología (informática, electrónica y telecomunicaciones).

Durante los últimos años la domótica ha evolucionado de tal manera que hoy en día podemos observar edificios inteligentes capaces de detectar la presencia de personas, temperatura, nivel de luz, humedad, entre otros estímulos que dan lugar a que la vivienda reaccione por sí sola regulando la calefacción, la iluminación, conectado la alarma, etc. Y al mismo tiempo es capaz de comunicarse e interactuar con nosotros mediante algunos medios como la PC, teléfonos móviles, pantalla táctil, entre otros dispositivos; mejorando de esta manera la seguridad, el confort, la accesibilidad, las comunicaciones y el ahorro energético.

A pesar de que la tecnología domótica ha avanzado en gran medida en la última década; se puede afirmar que su visión no va más allá de brindar servicios de bienestar a las personas dentro de las viviendas (domótica), edificios (inmótica) y sistemas de riego en el campo; poco o nada se conoce la implementación de este sistema en el área automotriz y los beneficios que se obtendrían del mismo.

Es por esta razón que el propósito u objetivo de esta investigación es el de diseñar e implementar un sistema de reconocimiento de voz inteligente, mediante la utilización de un software y hardware libre, para mejorar la interactividad (comunicación) y el confort (bienestar) de una persona cuadripléjica al controlar los diferentes accesorios de un vehículo; cabe recalcar que para este propósito es necesario involucrar a tres grandes áreas de la ingeniería como es la informática, la electrónica y la mecánica automotriz; así como es importante señalar que la implementación de este sistema dará lugar al desarrollo de nuevas tecnologías automotrices.

Hoy en día la mayoría de personas en el mundo nos encontramos en una constante interacción con la tecnología; es así que un ordenador, una tablet o un teléfono inteligente son herramientas que las utilizamos en nuestro diario vivir. Esta estrecha relación entre personas y aparatos tecnológicos (computación física) se ha fortalecido en la última década, debido a que los seres humanos hemos utilizado dichos aparatos con el propósito de mejorar nuestra calidad de vida.

Sí bien el término “computación física” refiere a la construcción de sistemas interactivos físicos mediante el uso de un software y hardware, donde se abarcan aspectos muy interesantes como la automatización de viviendas, edificios inteligentes, sistemas de riego, control del tráfico, automatización de fábricas, en un sentido más amplio la computación física se refiere a la relación de los seres humanos en el mundo virtual.

La utilización de herramientas tales como Arduino juegan un papel muy importante dentro del mundo de la computación física; ya que nos permiten elaborar prototipos (diseños) interactivos basados en un software y hardware (libre), flexibles y fáciles de utilizar, de manera que basta con tener algunos conceptos básicos en electrónica y programación como para emprender con nuestro proyecto de investigación. Además cabe indicar que la plataforma de Arduino cuenta con una gran comunidad donde se comparte todo el

conocimiento, siendo así una gran ventana para el desarrollo de la tecnología.

Por otra parte podemos recalcar que tanto el hardware y software utilizado provienen de la filosofía “libre”, es así que la información científica (especificaciones y diagramas esquemáticos refiriéndonos al hardware; y del código fuente en el caso del software) es de acceso público, lo cual facilita al investigador encontrar el material; así como también motiva a que este no pierda el interés.

Es por esta razón que opté por implementar un sistema inteligente domótico basado en un software y hardware libre, el mismo que tiene como finalidad el mejorar la interactividad y el confort de aquellas personas que padecen de algún tipo de discapacidad física (visual o motora), las mismas que podrán controlar los diferentes accesorios electrónicos de un vehículo (desde el copiloto) mediante un comando de voz. Además es importante resaltar que la utilización de este sistema es totalmente accesible para cualquier persona ya que por su bajo costo y su fácil comprensión, es posible desarrollar desde sistemas domóticos básicos para aficionados; hasta sistemas un poco más desarrollados, los mismos que fomentarán sin lugar a duda al desarrollo de nuevas tecnologías automotrices e impulsarán el mundo de la computación física en el área automotriz.

En la actualidad para importar un vehículo diseñado acorde a su discapacidad, es necesario la autorización por el consejo nacional de discapacidades (CONADIS); para lo cual se necesita un sinnúmero de requisitos con el fin de conseguir este mencionado permiso. A pesar de que el Ecuador por disposición del gobierno se exonera a personas con discapacidades del pago total de derechos arancelarios, impuestos adicionales e impuestos al valor agregado (IVA); este tipo de vehículos están al alcance de unas pocas personas debido a su elevado costo de fabricación, de manera que resulta poco rentable para las concesionarias del

país e inaccesible para muchas personas.

Por otra parte en cuanto a seguridad se refiere, muchas veces hemos podido escuchar de que por algún descuido o por falta de cuidado muchas personas se distraen mientras conducen; ya sea para mover el retrovisor, encender la radio, prender la calefacción o simplemente buscar las luces de salón han sido una de las causas de accidentes automovilísticos más comunes; es por esto que la implementación de dicho sistema nos permitirá controlar los diferentes accesorios electrónicos mediante un comando de voz, evitando de esta manera la manipulación directa con mencionados dispositivos y por ende reduciendo el índice de accidentes automovilísticos como consecuencia.

Cabe indicar que el método utilizado en el presente proyecto de investigación es de tipo exploratorio, ya que para el cumplimiento de los objetivos es necesario indagar en el tema de investigación, el mismo que al parecer únicamente existen ideas vagamente relacionadas; lo que a su vez pretende ser un proyecto nuevo y novedoso al no encontrar estudios o trabajos previos a la investigación.

Además es importante señalar que este estudio permitirá abrir nuevos horizontes de investigación, resolver nuevos problemas, llevar a cabo una investigación más completa en relación a un contexto en particular, sugerir afirmaciones, etc. De manera que se podrá contribuir en el desarrollo de esta línea de investigación denominada nuevas tecnologías automotrices con el fin de favorecer y mejorar nuestra calidad de vida.

Finalmente es necesario recalcar que desde el diseño hasta su implementación final es un proceso rápido y accesible desde el punto de vista científico y económico, ya que el hardware y software a utilizar no es difícil de encontrar y su costo es relativamente bajo, por lo que es posible concluir que esta investigación es totalmente viable y realizable.

2. MARCO TEÓRICO

2.1. ELECTRICIDAD

La electricidad es un fenómeno físico producido por las cargas eléctricas (positivas y negativas) en reposo o en movimiento. Este fenómeno conocido como electricidad puede ser definido como el movimiento de partículas eléctricas negativas (electrones) provenientes de los átomos que constituyen la materia. Existen básicamente seis formas de generar electricidad ya sea por fricción, deformación (presión), calor, luz, reacción química y magnetismo.

- La electricidad por fricción es aquella donde un cuerpo cede electrones al otro, un típico ejemplo de este fenómeno se lo puede observar al pasar un peine por el cabello seco y luego atraer con este (peine) partículas muy pequeñas.

- La electricidad por presión es posible observar generalmente en los materiales piezoeléctricos, quienes al ser sometidos a una fuerza (presión) son capaces de liberar electrones y desplazarlos hacia el extremo opuesto de dicha fuerza aplicada.

- La electricidad por calor conocida también como termoelectricidad es aquella que se origina cuando dos metales de diferente composición (Cu+, Zn-) se ponen en contacto, y son sometidos a una determinada temperatura; uno de ellos termina por ceder electrones al otro debido al aumento del movimiento cinético de sus átomos.

- La electricidad por luz también conocida como efecto fotoeléctrico consiste en la liberación de electrones de un material, cuando sobre este es aplicado una radiación electromagnética o luz. Generalmente este fenómeno lo podemos apreciar en las celdas fotovoltaicas, las mismas que transforman la energía lumínica en energía eléctrica.

- La electricidad por reacción química consiste en la utilización de una

sustancia química conocida también como electrolito, la misma que reacciona con un electrodo de carga positiva y otro de carga negativa; de tal manera que la diferencia de potencial existente entre ambos electrodos permite la circulación de electrones mediante dicho conductor iónico (electrolito).

- La electricidad por magnetismo es aquella que se produce cuando un campo magnético producido generalmente por un imán, es interrumpido por un embobinado giratorio conformado por un material conductor como el cobre, el mismo que corta cientos de veces las líneas de fuerza magnética, produciendo de esta manera una liberación de sus electrones de valencia y por ende una acumulación de carga capaz de generar electricidad.

2.1.1. ANTECENDENTES HISTÓRICOS

La electricidad proviene del griego "*elektron*" que significa ámbar, en honor al descubrimiento realizado por el filósofo griego Tales de Mileto (600 AC); quien observó que al frotar una varilla de ámbar con un pedazo de lana, este obtenía una cierta carga capaz de atraer a ciertos objetos ligeros.

Posteriormente en los años 1600 el físico William Gilbert realizó los primeros estudios científicos acerca de los fenómenos eléctricos, llegando a la conclusión de que el efecto de atracción que ejercían ciertos cuerpos al ser frotados se debía principalmente por la interacción existente entre dos tipos de cargas eléctricas, las cuales denominó como positivas (+) y negativas (-). Utilizando por primera vez el término "electricidad" y estableciendo la diferencia entre las acciones magnéticas y eléctricas.

No obstante las bases de la física en ese entonces aun no estaban bien acentuadas, por lo que se tuvo que esperar hasta el siglo XVII donde Isaac Newton estableció por primera vez los conceptos básicos de la física, permitiendo de esta manera el surgimiento de las bases teórico-prácticas

establecidas posteriormente por los científicos Agustín de Columb, Alejandro Volta, Andre Marie Ampere, Georg Simon Ohm, Michael Faraday, James Joule, entre otros investigadores quienes contribuyeron a la generación, aprovechamiento y distribución de la electricidad.

2.1.2. EL ÁTOMO

La palabra átomo proviene del griego que significa invisible y es la unidad más pequeña de un elemento, está constituido por el núcleo y corteza. En la figura 1 se puede observar que el núcleo está compuesto por protones de carga positiva y neutrones (sin carga); mientras que en la corteza es posible visualizar las partículas denominadas electrones de carga negativa.

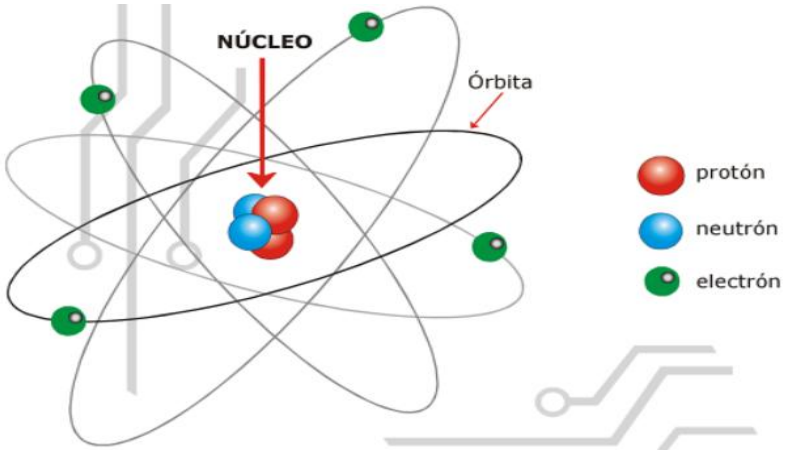


Figura 1. Estructura interna del átomo (Arboledas, 2009)

2.1.3. CORRIENTE CONTINUA “DC”

La corriente continua es aquella en la que los electrones circulan en un solo sentido; es decir aquella que circula a través de dos puntos de diferente potencial, donde los electrones van desde un punto de mayor potencial, al de menor potencial (en una sola dirección).

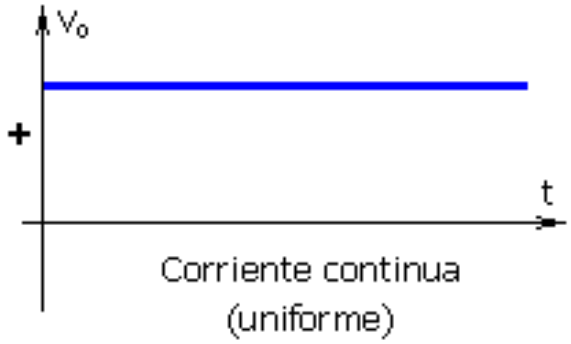


Figura 2. Corriente continua (Pastrana, 2009)

2.1.4. CORRIENTE ALTERNA "AC"

La corriente alterna es aquella donde la intensidad y la tensión cambian de valor y sentido durante un periodo de tiempo determinado, en otras palabras una corriente alterna se caracteriza por cambiar o alternar sus polaridades desde cero hasta su máximo positivo, para luego decrecer desde este punto hasta cero, llegando alcanzar posteriormente su máximo negativo y decreciendo nuevamente a cero. A esta variación de onda senoidal se la conoce como ciclo.

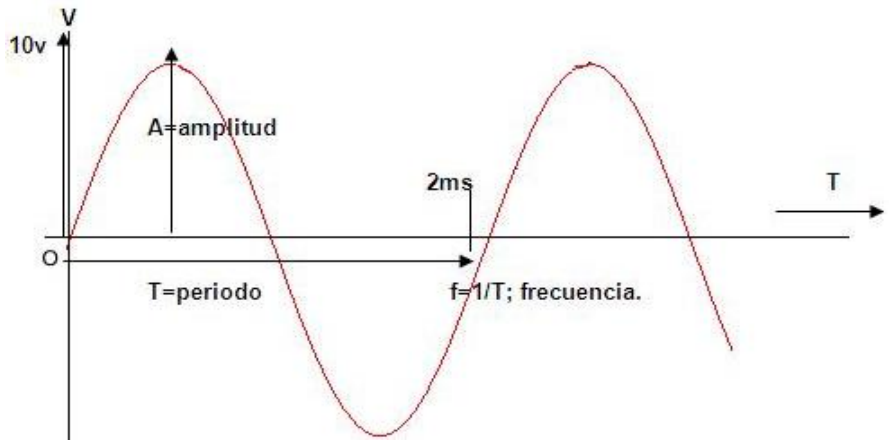


Figura 3. Onda Senoidal (Onubaelectronica, 2014)

2.1.5. AISLANTES Y CONDUCTORES

2.1.5.1. Aislantes

Son aquellos materiales generalmente no metales los cuales bajo condiciones normales carecen o poseen muy pocos electrones libres, por lo que no puede existir flujo de electrones, impidiendo así el paso de la corriente eléctrica. Son materiales aislantes aquellos constituidos por cerámica, mica, porcelana, plástico, hule, baquelita, gomas entre otros materiales los cuales se caracterizan por mantener unidos todos sus electrones a los átomos que los conforman.

2.1.5.2. Conductores

Son aquellos materiales o elementos que permite en flujo de electrones y se caracterizan por ser de la familia de los metales. Estos conductores se caracterizan por tener menos de cuatro electrones de valencia; es así que aquellos que poseen un solo electrón de valencia son considerados como buenos conductores.

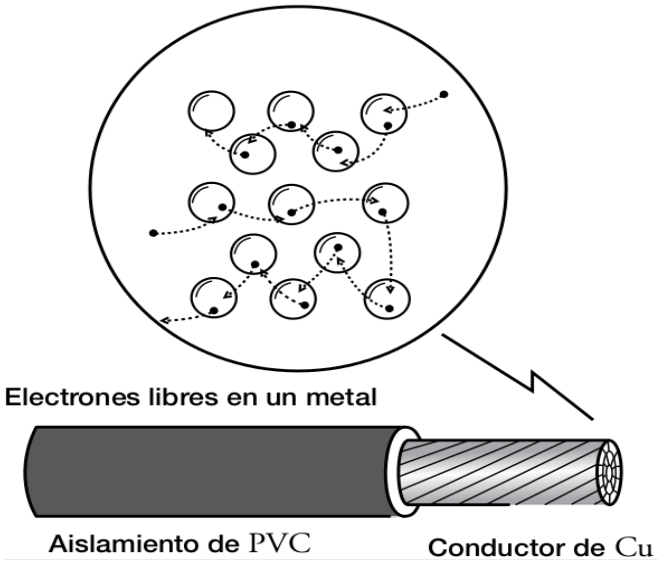


Figura 4. Conductores (Condumex, 2009)

Es por esta razón que materiales como la plata, cobre, oro y aluminio son excelentes conductores eléctricos debido a sus propiedades resistivas. Estos materiales (metales) están constituidos por paquetes de átomos con poco espacio entre ellos pero con una característica en especial y es que poseen electrones libres flotando entre dichos espacios, los mismos que viajan (flujo de corriente) a través del conductor ocupando un espacio tras otro.

2.1.6. TENSIÓN O VOLTAJE

Se refiere a la diferencia de potencial eléctrico existente entre dos puntos, donde la carga de mayor potencial eléctrico (negativo "-") fluye a la de menor potencial eléctrico (positivo "+"). Es decir que el voltaje es la fuerza que impulsa a los electrones o dicho de otra manera es la cantidad de energía necesaria para mover los electrones de un lugar a otro a través de un circuito; su unidad de medida es el Voltio (V) y se expresa en Joules/coulomb.

Instrumento de medición: Voltímetro (V)

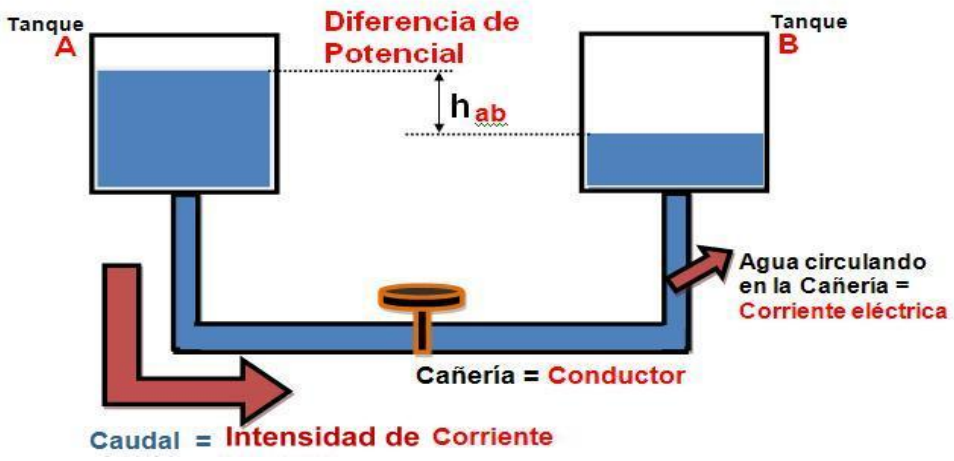


Figura 5. Diferencia de potencial (Caro, 2009)

2.1.7. INTENSIDAD O CORRIENTE

La corriente eléctrica se puede definir como el número de electrones libres que pasan a través de un circuito en un tiempo determinado; en otras palabras la intensidad o corriente de un circuito, se refiere a la cantidad de carga eléctrica transportada por dichos electrones durante un intervalo de tiempo considerado. Su unidad de medida es el amperio y se representa con la letra “A” en honor al físico francés André Ampere.

Instrumento de medición: Amperímetro (A)

2.1.8. RESISTENCIA

Es la propiedad del material a la oposición del flujo de electrones que circulan a través de un conductor. La resistencia a este flujo de electrones se debe a los choques existentes entre los electrones libres de la corriente y los iones positivos del metal, produciéndose de esta manera el calentamiento del conductor, el mismo que a su vez transmite dicha energía al medio en forma de calor; su unidad de medida es el ohmio y se representa con la letra omega “Ω”.

Instrumento de medición: Ohmnímetro (Ω)

2.1.9. LEY DE OHM

La ley de ohm indica la relación existente entre la tensión, la corriente y la resistencia de un circuito; donde después de varios experimentos realizados por el físico alemán OHM, se llegó a la conclusión de que la corriente es directamente proporcional a la tensión aplicada e inversamente proporcional a la resistencia. Es decir que a mayor o menor tensión aplicada, la corriente es directamente proporcional a dicho aumento o decremento de voltaje, siempre y cuando la resistencia se mantenga constante; mientras que si en un circuito cerrado el voltaje de una fuente es constante y la resistencia

aumenta o disminuye (varía), la corriente también cambiará de manera inversamente proporcional a dicha resistencia.

Matemáticamente lo enunciado es expresado de la siguiente manera:

$$I = \frac{V}{R}$$

Donde:

I = Intensidad o Corriente
V = Tensión o Voltaje
R = Resistencia

2.1.10. LEY DE JOULE

La ley de joule manifiesta que la energía eléctrica puede convertirse en energía térmica, debido a los constantes choques que sufren los electrones con los átomos durante su recorrido a través de un conductor. Esta energía térmica o calor producido por cada segundo se mide en watt (W) y un ejemplo muy claro de este efecto Joule lo podemos apreciar en las bombillas de luz de nuestros hogares; así como en los resistores electrónicos los cuales liberan dicha energía (térmica) al medio.

2.1.11. POTENCIA ELÉCTRICA

La potencia eléctrica no es más que la cantidad de energía entregada a un consumidor en un tiempo determinado. Donde de acuerdo a la ley de la conservación de la energía; esta puede transformarse en forma de luz, calor, movimiento, o en cualquier otra forma de energía capaz de realizar un trabajo.

De esta definición podemos deducir que la potencia se mide en Joules por segundo, es decir la energía consumida en un segundo; lo que comúnmente conocemos como vatios o watt (W).

$$P = VxI$$

2.1.12. CIRCUITO SERIE

Los circuitos serie son aquellos donde la energía eléctrica fluye en una sola trayectoria desde una fuente de alimentación, pasando a través de uno o varios consumidores para finalmente regresar a dicha fuente. Es decir que cuando una resistencia (consumidor) está conectada con otra de extremo a extremo (una a continuación de otra) se dice que un circuito está dispuesto en serie.

Un circuito serie se caracteriza por que la corriente que circula a través de las resistencias es constante (igual) para cualquier punto del circuito; mientras que en el caso de la resistencia equivalente y la tensión equivalente (total) corresponde a la sumatoria de todos los valores dados para cada consumidor, por lo que se puede deducir de la siguiente manera:

$$I_{TOTAL} = I_1 = I_2 = I_3 = I_n$$

$$V_{TOTAL} = V_1 + V_2 + V_3 + V_n$$

$$R_{TOTAL} = R_1 + R_2 + R_3 + R_n$$

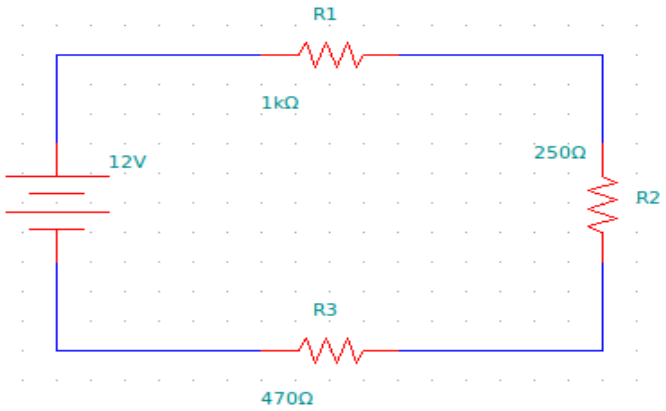


Figura 6. Circuito serie

Donde:

$$V_{TOTAL} = 12\text{V}$$

$$R_{TOTAL} = 1000\Omega + 250\Omega + 470\Omega$$

$$R_{TOTAL} = 1720\Omega$$

$$I_{TOTAL} = \frac{12V}{1720\Omega}$$

$$I_{TOTAL} = 0{,}00697\ A$$

2.1.13. CIRCUITO PARALELO

Un circuito en paralelo se caracteriza porque la corriente que fluye a través del conductor tiene más de una trayectoria, es decir que en un circuito paralelo existen tantos caminos como consumidores incluidos en este último. En la figura 24 se puede observar tres resistencias (R1, R2, R3) conectadas en paralelo, donde la corriente circula a través de ellas gracias a la diferencia de potencial (tensión) proporcionada por la fuente; esta tensión o voltaje se caracteriza por ser equivalente en cualquier punto del circuito.

$$V_{TOTAL} = V_1 = V_2 = V_3 = V_n$$

$$I_{TOTAL} = I_1 + I_2 + I_3 + I_n$$

$$\frac{1}{R_{TOTAL}} = \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} + \frac{1}{R_n}$$

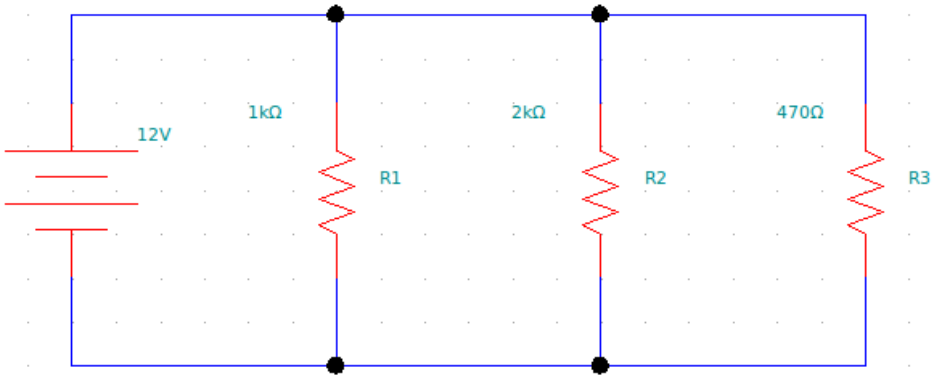


Figura 7. Circuito paralelo

Donde:

$$V_{TOTAL} = 12V$$

$$\frac{1}{R_{TOTAL}} = \frac{1}{1000\Omega} + \frac{1}{2000\Omega} + \frac{1}{470\Omega}$$

$$\frac{1}{R_{TOTAL}} = 0{,}001\,\Omega + 0{,}0005\,\Omega + 0{,}002127\,\Omega$$

$$\frac{1}{R_{TOTAL}} = 0{,}003627\,\Omega$$

Despejando: R_{TOTAL}

$$R_{TOTAL} = 275{,}71\,\Omega$$

$$I_{TOTAL} = \frac{12\text{V}}{275{,}71\,\Omega}$$

$$I_{TOTAL} = 0{,}0435\,A$$

2.1.14. CIRCUITO MIXTO

Un circuito mixto resulta de la combinación de un circuito serie y un circuito paralelo, donde para aplicar las leyes de Ohm es necesario descomponer la conexión en circuitos elementales, es decir que para cada caso se aplicaran las leyes de ohm correspondientes ya sea para un circuito en serie o en paralelo; con la finalidad de obtener un solo circuito asociado y simplificado que nos permita dar con la resolución total.

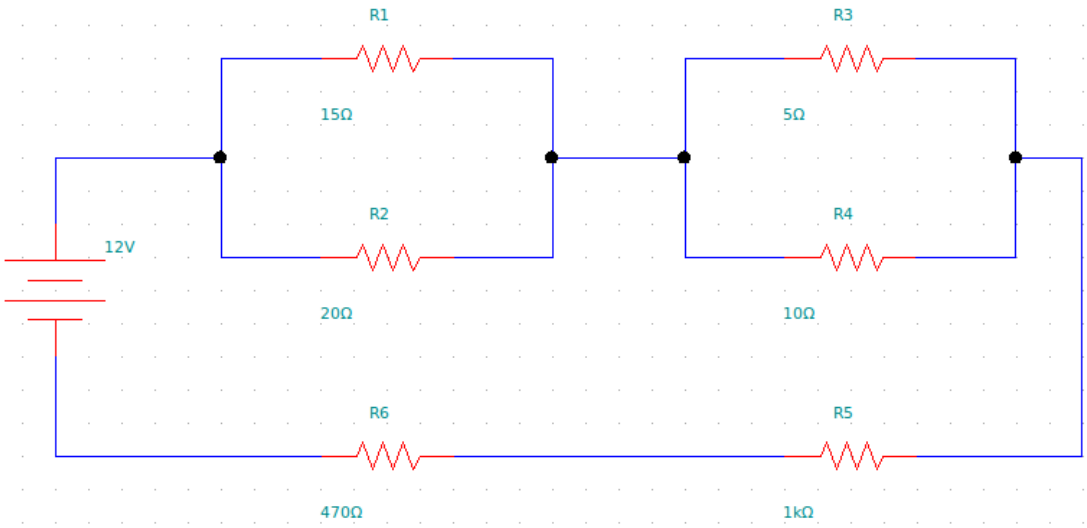


Figura 8. Circuito mixto

Simplificando el circuito:

$$R_A = \frac{R_1 x R_2}{R_1 + R_2}$$

$$R_A = \frac{15\Omega x 20\Omega}{15\Omega + 20\Omega}$$

$$R_A = 8{,}57\,\Omega$$

$$\frac{1}{R_B} = \frac{1}{R_3} + \frac{1}{R_4}$$

$$\frac{1}{R_B} = \frac{1}{5\Omega} + \frac{1}{10\Omega}$$

$$\frac{1}{R_B} = 0{,}20\,\Omega + 0{,}10\,\Omega$$

$$R_B = 3{,}33\,\Omega$$

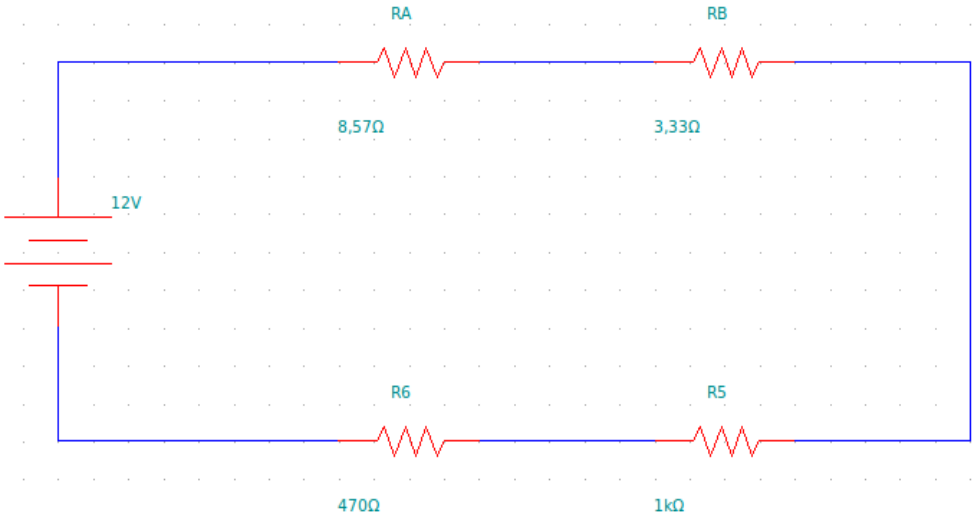


Figura 9. Circuito mixto simplificado

$$V_{TOTAL} = 12\text{V}$$

$$R_{TOTAL} = 8{,}57\,\Omega + 3{,}33\,\Omega + 1000\Omega + 470\Omega$$

$$R_{TOTAL} = 1481{,}9\,\Omega$$

$$I_{TOTAL} = \frac{12\text{V}}{1481{,}9\,\Omega}$$

$I_{TOTAL} = 0{,}008097\,A$

2.2. ELECTRÓNICA

La electrónica es una rama de la física y parte de la ingeniería la misma que se encarga de estudiar el comportamiento de los electrones que circulan a través de un circuito, con la finalidad de controlar estos últimos (electrones). De esta definición es posible afirmar que la electrónica está estrechamente relacionada con la electricidad, ya que todo sistema electrónico por simple o complicado que parezca se alimenta de la energía eléctrica y por ende se rige a los principios básicos de la electricidad.

2.2.1. ANTECEDENTES HISTÓRICOS

Debido a los avances en la evolución de la tecnología eléctrica se dio paso al surgimiento de la electrónica como tal, la misma que hoy en día es considerada como una de las disciplinas más sobresalientes e incluyentes de la ingeniería.

La electrónica tuvo sus orígenes a principios del siglo XX con el descubrimiento del efecto termoiónico por Thomas Edison, donde se pudo observar que los cuerpos incandescentes (metales) eran capaces de emitir electrones libres al dejarlos en el espacio vacío.

En 1904 Fleming inventó el primer diodo de vacío basándose en los principios de la emisión termoiónica, para posteriormente complementarse en 1906 con el descubrimiento del triodo por Lee de Forest. Dando lugar de esta manera al surgimiento de la electrónica como tal y de los primeros amplificadores, televisores y receptores de radio frecuencia.

Un diodo de vacío tiene la particularidad de estar formado por una placa o lámina metálica polarizada positivamente (ánodo) y un filamento denominado cátodo; cuando el filamento recibe una cierta cantidad de

energía en forma de calor, este permite la emisión de electrones libres (efecto termoiónico) los mismos que atraviesan el espacio vacío y son atraídos por la placa metálica de carga positiva.

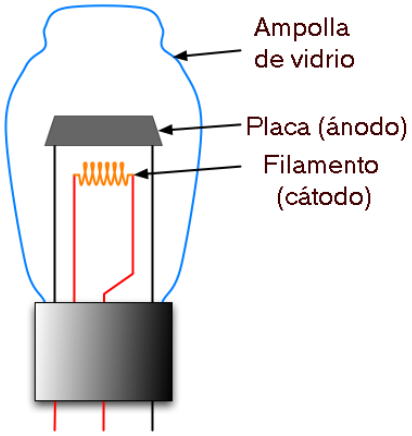


Figura 10. Diodo de vacío **(**Pastrana, 2009)

2.2.2. SISTEMAS ELECTRÓNICOS

Se denomina sistema electrónico al conjunto de componentes de un circuito que interactúan entre sí por medio de un procesador y que como resultado de una entrada; se obtiene una salida. Cabe destacar que tanto las entradas como las salidas pueden ser de tipo análogas o digitales.

Figura 11. Sistemas electrónicos

2.2.3. ENTRADAS (INPUTS)

Las entradas son aquellos componentes electrónicos que transforman las variables del mundo físico (temperatura, presión, humedad, luz, caudal, entre otros) en señales eléctricas para posteriormente ser procesadas. Es decir que dichos dispositivos se encargan de ingresar datos (información)

hacia la unidad central de procesamiento, donde son almacenados y evaluados (procesados) para dar como resultado una salida.

2.2.4. SALIDAS (OUTPUTS)

Son aquellos componentes electrónicos que utilizando las señales enviadas por un microprocesador son capaces de desarrollar determinadas acciones. En otras palabras las salidas son todos aquellos dispositivos que nos permiten representar los resultados de un proceso de datos provenientes de las entradas.

2.2.5. PROCESAMIENTO DE SEÑAL

Las señales emitidas por una entrada como es el caso de los sensores, son procesadas mediante una unidad de control electrónico, la misma que tiene como función el recibir, interpretar y enviar dichas señales como una respuesta hacia los diferentes dispositivos electrónicos (salidas) para realizar una función determinada.

2.2.6. SEÑALES ELECTRÓNICAS

Las señales o impulsos eléctricos son magnitudes físicas de corriente, tensión o carga. Las mismas que pueden ser modificadas en el tiempo (variables) y se caracterizan por ser de dos tipos: variables análogas y digitales.

2.2.6.1. Señal análoga

Una señal análoga o continua es aquella que varía en el tiempo, es decir que dentro de un límite establecido esta puede tomar cualquier valor y se la representa como una onda sinusoidal. En la actualidad este tipo de señal (análoga) sigue siendo muy utilizada por muchos componentes electrónicos; aun cuando la tecnología digital presenta grandes ventajas a comparación

con una señal analógica, es difícil saber si algún día la era digital será capaz de remplazar por completo dicha tecnología análoga.

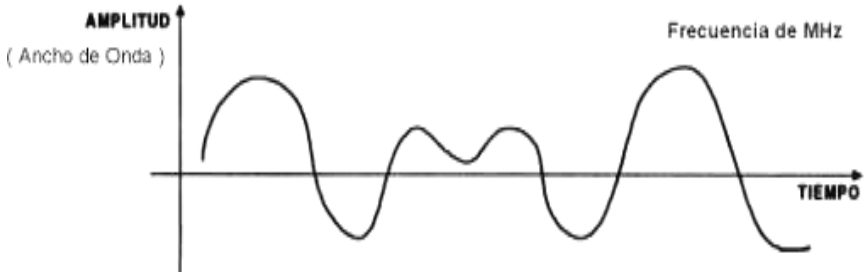


Figura 12. Señal Análoga (Tecnopatoo, 2013)

2.2.6.2. Señal digital

Una señal digital o discreta es aquella que varía en intervalos determinados y puede tomar únicamente dos valores comprendidos entre 0 y 1; es decir que este tipo de señales no pueden tomar un valor intermedio, por lo que se las considera como señales binaras al estar comprendidas entre un nivel lógico alto (1) y un bajo (0).

Una de las principales ventajas que presenta una señal de tipo digital es que esta se ve menos afectada por los ruidos externos (ambientales) a comparación con una señal análoga; por lo que es considerada como una señal mucho más estable (fiable) al no presentar pérdidas de información.

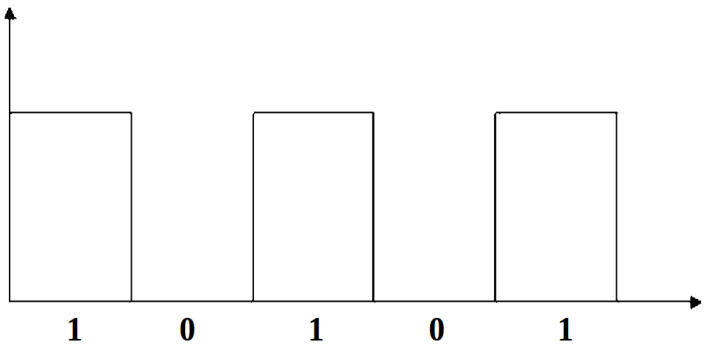


Figura 13. Señal Digital

2.2.7. CONVERSOR A/D

Un convertidor A/D es un circuito electrónico integrado el mismo que convierte una señal de tipo análoga en su entrada, en una señal digital como salida. Para esto es necesario que dicho conversor interprete y transforme una señal variable continua (análoga) en una señal variable binaria; es decir comprendida entre ceros y unos o también conocida como señal digital binaria.

Un ejemplo muy claro se puede observar en el caso de un termopar (sensor de temperatura), donde la señal analógica continua procedente del exterior es amplificada y transformada en una señal de tipo digital mediante un conversor A/D, el mismo que se encarga de enviar hacia la unidad de procesamiento una señal (digital) mucho más legible y fácil de procesar.

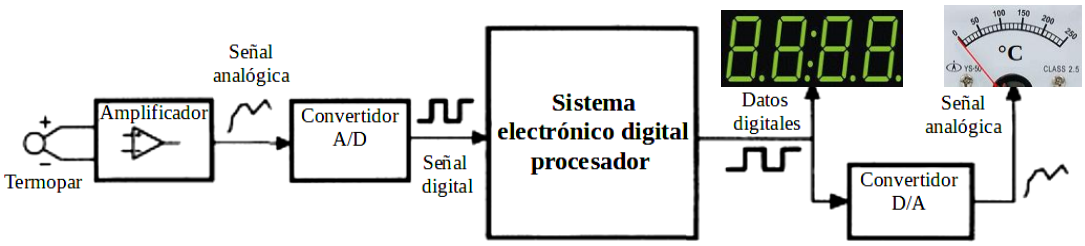


Figura 14. Convertidor A/D – D/A (Hermosa, 2004)

2.2.8. MODULACIÓN POR ANCHO DE PULSO PWM

La modulación por ancho de pulso (PWM) es una técnica que nos permite variar y controlar los impulsos eléctricos en las salidas; logrando de esta manera simular una señal de tipo análoga en una salida digital, por medio de la variación del tiempo (anchura) en que permanece encendido o apagado un consumidor.

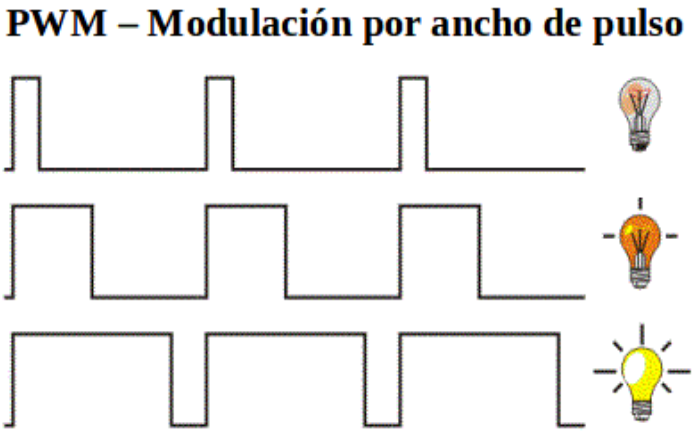


Figura 15. Modulación por ancho de pulso (Banshee, 2014)

2.2.9. COMPONENTES ELECTRÓNICOS

2.2.9.1. Microcontrolador

Un microcontrolador es un “computador” dentro de un circuito integrado programable, el cual posee un CPU (Unidad Central de Procesamiento), una memoria interna (ROM/RAM) y la unidad de entradas y salidas.

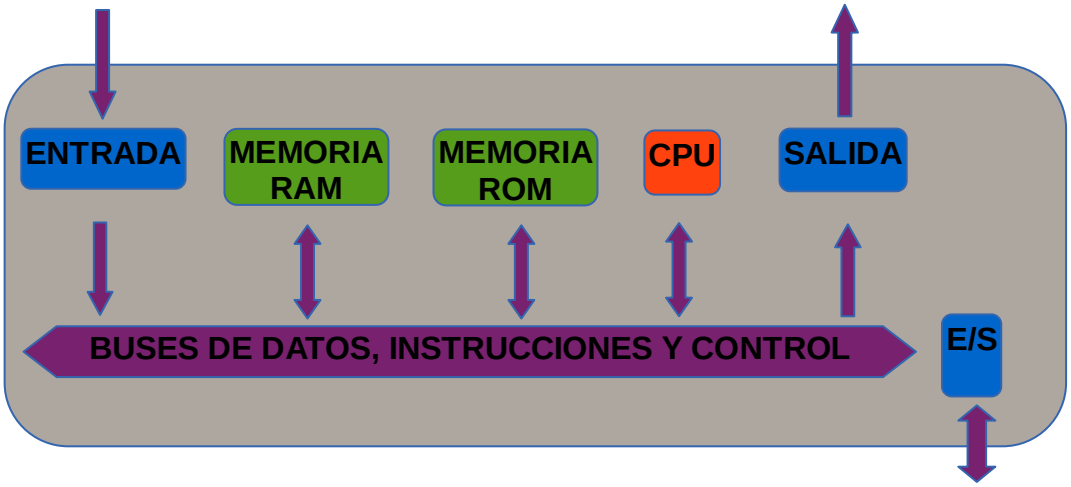


Figura 16. Microcontrolador

Al igual que en un ordenador el cerebro del microcontrolador viene a ser la CPU o unidad central de procesamiento, la misma que se encarga de ejecutar las instrucciones previamente grabadas en su memoria, en base a los datos obtenidos por medio de sus periféricos.

La memoria RAM (Random Access Memory) es una memoria de lectura y escritura volátil, debido a que los datos o información almacenada ocupa gran cantidad de espacio en la memoria ROM por lo que es temporalmente guardada. Una memoria ROM (Read Only Memory) se caracteriza por guardar los datos de manera permanente, debido a que dicha información corresponde a la ejecución del programa en si, por lo que es una memoria de solo lectura y no de escritura.

2.2.9.2. Resistor

Un resistor es un elemento electrónico fabricado con un cierto margen resistivo, el mismo que tiene como finalidad oponerse al flujo de electrones que pasan a través de un conductor; de tal manera que la corriente eléctrica es distribuida adecuadamente hacia los diferentes componentes de un circuito, mientras que la energía excedente consumida (potencia) por dicha resistencia es liberada en forma de calor al medio. Existen varios tipos de resistores como los fabricados a base de carbón, alambre y aquellos que poseen una película metálica en su interior.

Figura 17. Resistor (Fajardo, 2012)

Los resistores de carbón generalmente son muy utilizados en el campo de la electrónica debido a su eficiencia y su bajo costo; además que posee valores resistivos muy altos comprendidos entre un ohmio y veinte y dos millones de ohmios. Estos componentes pasivos poseen un código de color impreso en su superficie el cual nos permite calcular su valor resistivo así como la tolerancia del mismo.

Tabla 1. Código de colores para los resistores

Color	1ª Cifra	2ª Cifra	Multiplicador	Tolerancia
Negro	-	0	1	-
Marrón	1	1	10	± 1%
Rojo	2	2	100	± 2%
Naranja	3	3	1000	-
Amarillo	4	4	10000	± 4%
Verde	5	5	100000	± 0,5%
Azul	6	6	1000000	± 0,25%
Violeta	7	7	10000000	± 0,1%
Gris	8	8	100000000	± 0,05%
Blanco	9	9	1000000000	-
Dorado	-	-	0,1	± 5%
Plateado	-	-	0,01	± 10%
Ninguno	-	-	-	± 20%

2.2.9.3. Capacitor o condensador

Los capacitores son componentes electrónicos de dos terminales que básicamente tienen como función almacenar una carga eléctrica, donde la tensión entre dichos terminales es proporcional a la carga. En otras palabras un capacitor se asemeja mucho a una batería pero con la diferencia de que la carga y descarga de energía es de forma instantánea en un condensador. Son utilizados generalmente para acondicionar señales y proteger circuitos integrados.

Figura 18. Condensador (Fajardo, 2012)

Los capacitores son muy utilizados en los aparatos eléctricos de manera que los podemos encontrar fácilmente en radios, televisores, ordenadores, teléfonos, refrigeradores, lavadoras, y demás artículos utilizados en nuestro diario vivir. Existen varios tipos de capacitores como son los cerámicos, electrolíticos, poliéster y mica; los cuales dependiendo del material dieléctrico utilizado adquieren sus características capacitivas. La unidad de medida de la capacidad de carga o capacitancia es el "*Faradio"* (F) y se lo puede obtener mediante la relación existente entre la carga almacenada (Q) y la tensión aplicada (V).

$$C = \frac{Q}{V}$$

Tres son los submúltiplos del faradio más utilizados en electrónica debido a sus dimensiones, ya que un capacitor de un solo faradio sería demasiado grande para un circuito electrónico donde las tensiones y cargas son relativamente pequeñas así como también sus componentes.

Tabla 2. Submúltiplos del faradio

Submúltiplos	**Símbolo**	**Valor Equivalente**
Microfaradio	(µF)	0,000001F
Nanofaradio	(nF)	0,000000001F
Picofaradio	(pF)	0,000000000001F

2.2.9.4. Diodo

El diodo es un elemento semiconductor el cual tiene como función permitir la circulación de la corriente en un solo sentido. Existen varios tipos de diodos y son utilizados generalmente para proteger circuitos integrados; así como para transformar la corriente alterna en continua.

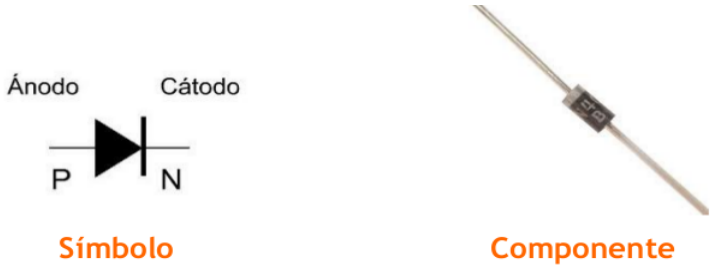


Figura 19. Diodo (Fajardo, 2012)

Un diodo se caracteriza por tener dos secciones o polaridades, donde la sección “P” corresponde al ánodo (+) y la sección “N” al cátodo (-); esto se logra gracias a la cantidad de impurezas (dopaje) existentes en cada sección del material (silicio), las mismas que permitirán el exceso o déficit de electrones en dichas secciones dándoles de cierta manera una polaridad.

Se dice que un diodo es semiconductor debido a que este se comporta como un conductor y como un aislante a la vez, es decir que si se aplica una tensión externa mayor en la sección “P” que en la “N”, se reduce la barrera de potencial y por ende existe circulación de corriente; mientras que si se aplica una tensión externa mayor en la sección “N” que en la sección “P”, se incrementa la barrera de potencial evitando así la circulación de corriente y por ende comportándose como un aislante.

2.2.9.5. Transistor

Un transistor es un elemento semiconductor el cual tiene como función amplificar una señal o a su vez puede comportarse como un interruptor; está formado por tres terminales denominados Emisor, Base y Colector (E, B, C), donde mediante la utilización de elementos químicos semiconductores como el silicio y el germanio es posible controlar una gran potencia a partir de una pequeña. Existen dos tipos de transistores: NPN y PNP. En la figura 11 se puede apreciar la simbología y nomenclatura de sus terminales.

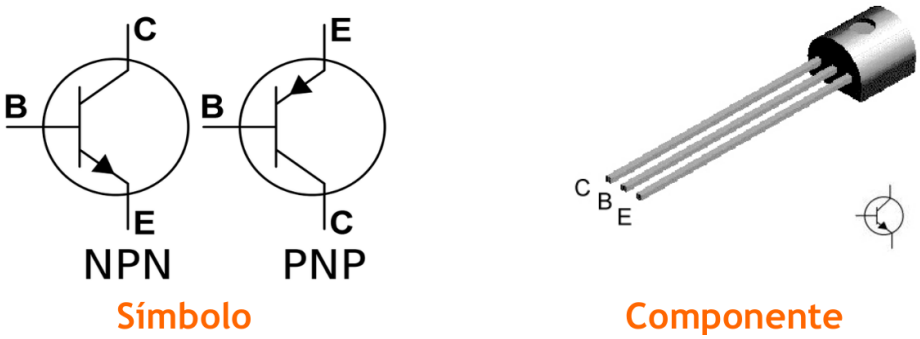


Figura 20. Transistor (Fajardo, 2012)

Un transistor NPN (negativo, positivo, negativo) se caracteriza porque la flecha impresa en el terminal del emisor apunta hacia fuera; mientras que un transistor del tipo PNP (positivo, negativo, positivo) la flecha del terminal del emisor apunta hacia dentro. Estos transistores son denominados bipolares ya que son muy parecidos a un diodo pero con una sección más; cabe indicar que un transistor del tipo NPN no puede ser sustituido por otro del tipo PNP, ya que las uniones colector-base del transistor deben ser polarizadas adecuadamente. Es decir que en un transistor NPN, el colector ha de ser positivo con respecto a la base; mientras que en un transistor del tipo PNP, el colector será negativo con respecto a la base.

2.2.9.6. Diodo luminoso

Un LED es un diodo emisor de luz o semiconductor que emite luz y se caracteriza por poseer dos regiones o polaridades denominadas “P” y “N”, es decir que si a un diodo se lo polariza de manera directa, los electrones pasan de la sección “N” a la sección “P”; dicho de otra manera los electrones libres de la sección “N” son atraídos a los huecos existentes en la sección “P”, lo que da como resultado una liberación de energía electromagnética o emisión de fotones que pueden ser observados en forma de luz.

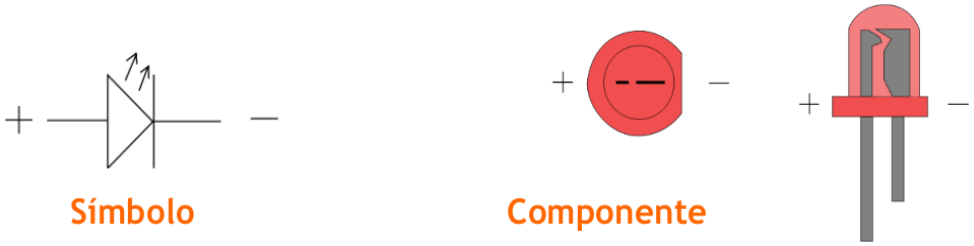


Figura 21. LED (Fajardo, 2012)

Este componente (LED) es muy utilizado en los sistemas electrónicos como un indicador visual; así como una fuente de luz debido principalmente a su bajo consumo, tamaño y durabilidad. Existen varios tipos de LED como los RGB, de iluminación ultravioleta, encapsulados, etc. En la figura superior se puede apreciar la polaridad (ánodo (+) - cátodo (-)) existente en cada uno de sus terminales así como su simbología; cabe indicar que si dicho componente es conectado de manera errónea (polaridad) este puede quemarse.

2.2.9.7. Pulsador

Un pulsador no es más que un interruptor manual utilizado generalmente para activar o desactivar dispositivos eléctricos, este interruptor permite enviar pulsos o señales eléctricas binarias al tener dos estados. Dicho de otra manera un pulsador tiene como función permitir o interrumpir el paso de la corriente eléctrica; a diferencia de un interruptor común, un pulsador cumple con su función únicamente cuando este es presionado por una fuerza externa (normalmente con un dedo).

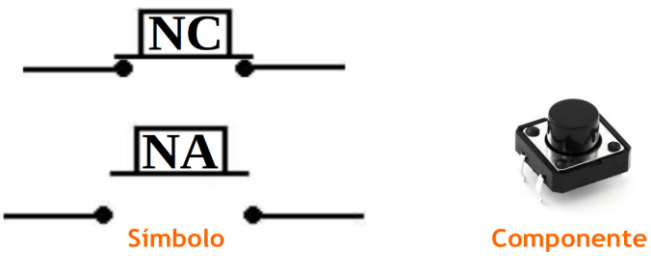


Figura 22. Pulsador (Fajardo, 2012)

Un pulsador se caracteriza por estar en un solo estado inicial, es decir normalmente abierto “NA” o normalmente cerrado “NC”. Cuando un interruptor normalmente abierto (NA) es presionado, este cierra el circuito, permitiendo de esta manera el flujo de corriente; mientras que si un pulsador del tipo NC (normalmente cerrado) es presionado, este se abre, evitando así el paso de la corriente eléctrica.

2.2.9.8. Potenciómetro

Un potenciómetro no es más que una resistencia de tipo variable la misma que mediante un eje manual es posible desplazar el cursor a través de una pista resistiva, donde en cuyos extremos dicho cursor tiene contacto con los terminales variando de esta manera su resistividad; y por ende controlando la intensidad de corriente que fluye a través del circuito.

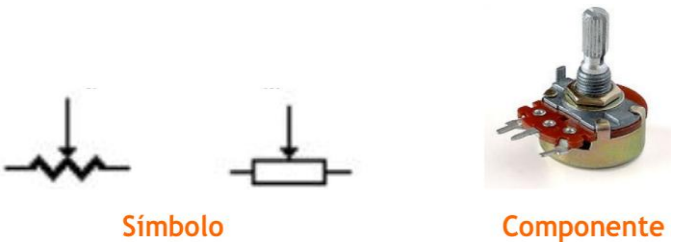


Figura 23. Potenciómetro (Fajardo, 2012)

2.2.9.9. Fotocelda

Una fotocelda o fotorresistencia es un componente electrónico formado por un semiconductor capaz de generar corriente eléctrica al ser sometido a la luz, es decir que mediante la radiación electromagnética (luz) el semiconductor es capaz de reducir su resistencia al paso de electrones (resistencia variable), debido principalmente al efecto fotoeléctrico aplicado sobre este último.

Figura 24. Fotocelda (Fajardo, 2012)

2.2.9.10. Motor DC

Un motor de corriente directa o continua es aquel que transforma la energía eléctrica en mecánica (movimiento); este motor está compuesto principalmente por una carcasa que en cuyo interior contiene un imán fijo o estator y un electroimán giratorio o rotor. Cuando una corriente eléctrica (DC) es aplicada al embobinado de dicho electroimán, este genera un campo electromagnético capaz de interactuar con el campo magnético del imán; de modo que si sus polos (norte - sur) llegan a coincidir, se genera una repulsión entre estos, obligando de esta manera a que gire dicho rotor.

Figura 25. Motor DC (Fajardo, 2012)

El sentido de giro de un motor de corriente directa dependerá de la manera en que este se encuentre conectado a la fuente, es decir que si se cambia sus polaridades, este comenzará a girar en sentido contrario al primero.

Cabe indicar que este componente es muy utilizado en los aparatos de funcionamiento eléctrico (corriente alterna), los mismos que gracias a la

utilización de algunos semiconductores, es posible convertir la corriente alterna en directa. Por otro lado también los podemos encontrar en dispositivos electrónicos así como en el campo automotriz.

2.2.9.11. Protoboard

Un protoboard o breadboard es una tarjeta o placa de pruebas, la cual nos permite montar componentes electrónicos con la finalidad de diseñar prototipos o pruebas experimentales de manera rápida y sencilla sin la necesidad de soldadura. Esta tarjeta está compuesta por una serie de orificios conectados entre sí de manera horizontal (filas) y vertical (columnas) como se puede apreciar en la figura 17.

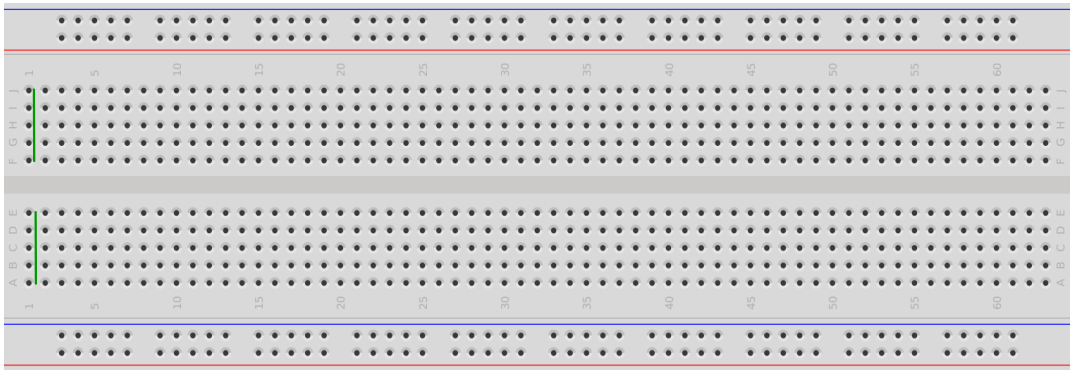


Figura 26. Protoboard

En particular la imagen muestra un protoboard de 63 filas y 14 columnas, donde se puede conectar directamente los componentes electrónicos, así como las líneas o cables que permitirán la alimentación y el diseño del circuito.

2.3. PROGRAMACIÓN

Se conoce como programación a todo el proceso que implica diseñar, codificar, depurar y ejecutar el código fuente. Cabe indicar que uno de los procesos más relevantes durante la programación corresponde a la depuración del código, ya que es aquí donde se identifican los errores o “bugs” de un programa, los mismos que son corregidos con la finalidad de ejecutar el código sin ningún tipo de inconveniente.

Para escribir código es necesario tener ciertos conocimientos en varias disciplinas, así como el dominio del lenguaje de programación a utilizar. Es así que dentro del lenguaje de programación C, uno de los requisitos fundamentales para programar es el conocimiento en lo que se refiere a lógica matemática, algoritmos especializados, funciones matemáticas, variables, constantes, operadores, condicionales, entre otros dominios como la estructura y sintaxis misma del lenguaje usado.

2.3.1. LENGUAJE DE PROGRAMACIÓN

Un lenguaje de programación es aquel que nos permite realizar notaciones útiles para un ordenador; en otras palabras cuando se habla de lenguaje de programación nos referimos a un lenguaje artificial, donde se puede escribir instrucciones u órdenes capaces de realizar un determinado proceso. A estas instrucciones se las conoce normalmente como “código fuente” o “lenguaje fuente”; las mismas que son escritas (gramática) de acuerdo al lenguaje de programación utilizado.

Se puede destacar dos tipos de lenguajes de programación de acuerdo a su nivel o categoría: Los lenguajes de alto nivel (compiladores) y los lenguajes de bajo nivel (lenguaje máquina, lenguaje ensamblador).

2.3.1.1. Lenguaje máquina

El lenguaje máquina es aquel que puede entender cualquier ordenador y se caracteriza por ser código binario; es decir un procesador únicamente puede entender variaciones de voltaje, por lo que utiliza las cifras comprendidas entre cero y uno (apagado y encendido respectivamente) para interpretar las instrucciones y realizar una acción.

Figura 27. Código binario (Fajardo, 2012)

En sus inicios los ordenadores eran programados directamente en lenguaje máquina por lo que resultó muy complicado dicha tarea, ya que una letra o un símbolo tenía su propio código binario (101001101...). Es así que con el pasar del tiempo surgieron herramientas que permitirían a los programadores desarrollar instrucciones mucho más complejas y fáciles de interpretar por el ser humano.

En la actualidad estos estamentos o instrucciones conocidos también como “código fuente” son escritos en un compilador o en lenguaje ensamblador donde la utilización de variables, funciones, constantes, caracteres, etc., permiten construir un algoritmo funcional, el cual posteriormente será interpretado por un computador al ser transcrito a código máquina (código binario).

2.3.1.2. Lenguaje ensamblador

El lenguaje ensamblador se caracteriza por ser una representación simbólica del lenguaje máquina y se lo considera como un lenguaje de bajo nivel en el mundo de la programación. El código fuente escrito en lenguaje ensamblador es traducido a código binario (lenguaje máquina) por medio de un programa informático conocido también como assembler.

Una de las principales ventajas del lenguaje ensamblador es que se caracteriza por ser más compacto y mucho más rápido en comparación de un lenguaje de alto nivel, por lo que es muy recomendable utilizarlo en aquellos casos donde las instrucciones son demasiado extensas.

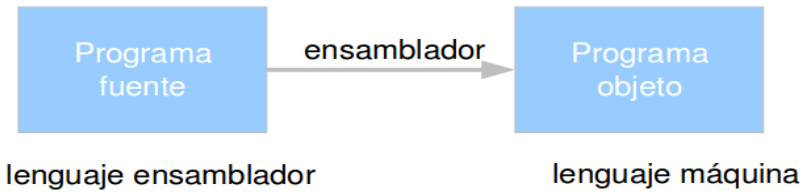


Figura 28. Lenguaje ensamblador

2.3.1.3. Compilador

Un compilador es un programa informático el cual se encarga de traducir un lenguaje escrito de alto nivel (C, C++, javascript, Ruby, python, etc) a un lenguaje entendido por el ordenador (lenguaje máquina) o bien a un lenguaje intermedio como puede ser el lenguaje ensamblador.

Hoy en día el lenguaje de programación de alto nivel es muy aceptado y utilizado por los programadores, debido a que las reglas sintácticas y semánticas que este posee, son muy parecidas al lenguaje humano y por ende es mucho más fácil en términos de complejidad y tiempo llevar a cabo una programación. Un ejemplo muy claro es el entorno de programación integrado de la plataforma Arduino (véanse páginas 63 y 66), conocido también como IDE (Integrated Development Environment) así como lo

muestra la figura inferior.

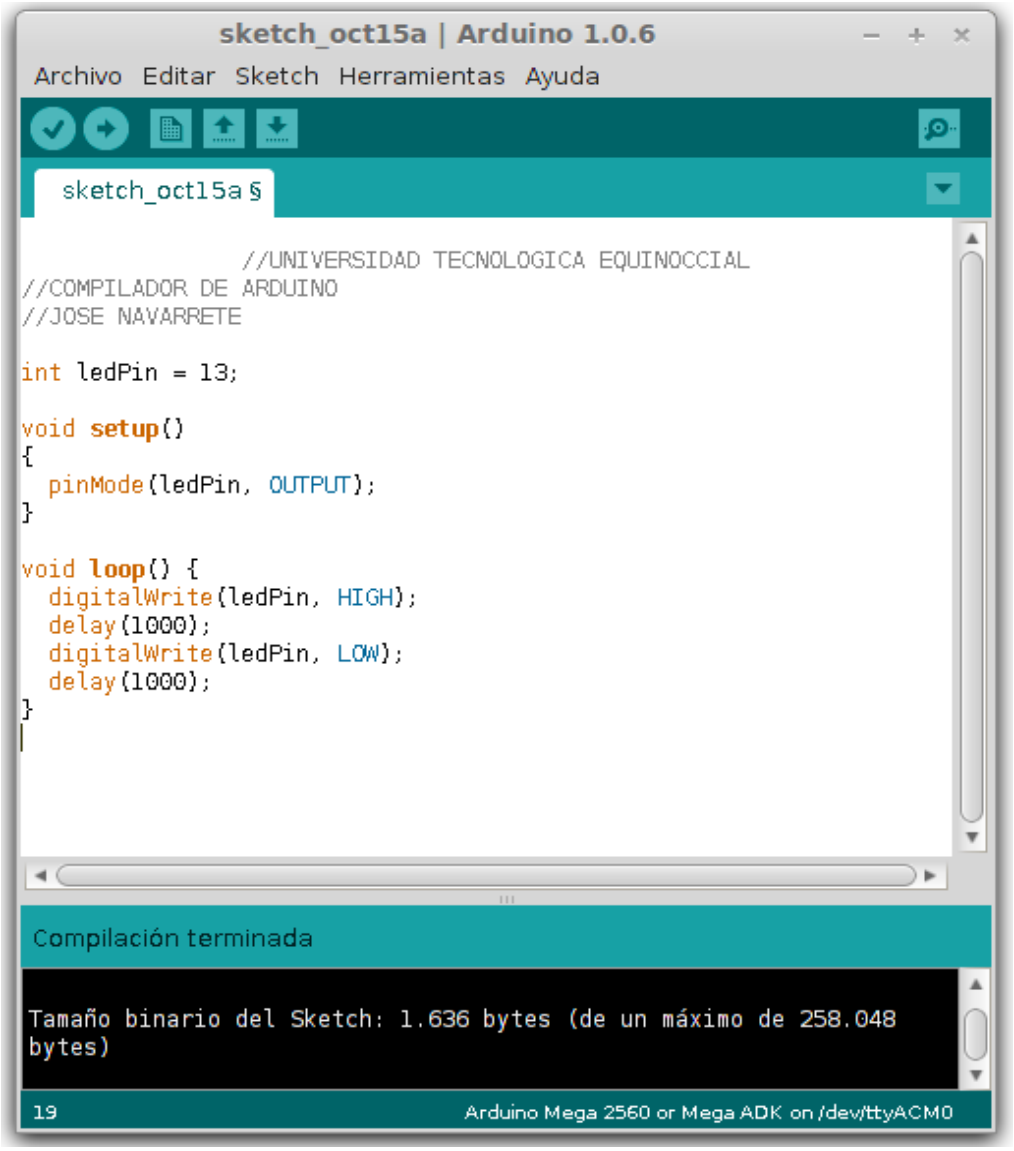


Figura 29. Compilador IDE de Arduino

2.3.2. ALGORITMO

Un algoritmo es una secuencia lógica de pasos o tareas finitas que permiten resolver un determinado problema, por lo que no necesariamente un algoritmo está ligado a la informática como tal; sino más bien para cualquier tipo de problema en la vida cotidiana. Es así que se puede realizar tareas como la preparación de un pastel, cambiar el neumático de un vehículo, construir un edificio, lavar la ropa, etc., por medio de una serie de pasos

lógicos (algoritmo) los cuales se caracterizan por tener una entrada en su inicio y una salida en la solución del problema.

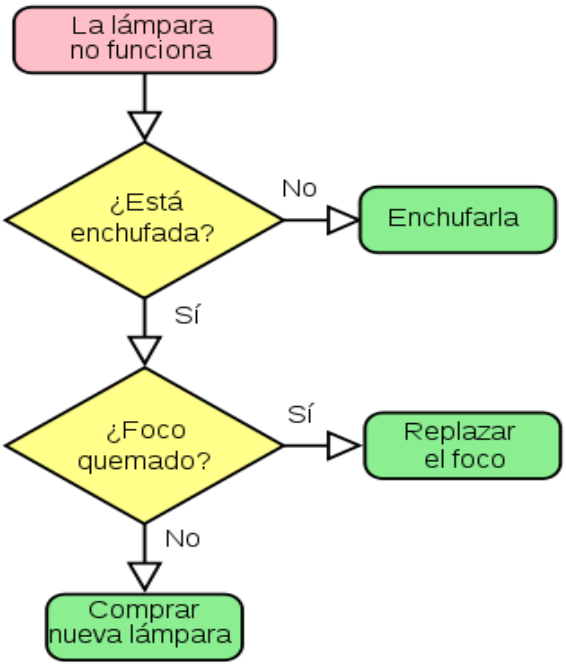


Figura 30. Algoritmo

2.3.3. LENGUAJE DE PROGRAMACIÓN ARDUINO IDE

Arduino utiliza un lenguaje de programación de alto nivel basado en C y C++, donde por medio de estructuras, variables, operadores (matemáticos, lógicos y booleanos), controladores de flujo (estructuras de control) y funciones. Se tiene la capacidad de escribir una serie de instrucciones lógicas, las mismas que pueden ser cargadas directamente al microprocesador de Arduino mediante el compilador IDE.

2.3.4. ESTRUCTURA DEL SKETCH

El entorno de programación en Arduino está formado principalmente por dos funciones denominadas **void setup** y **void loop**.

2.3.4.1. Void setup

Esta es la parte donde se carga la configuración inicial del programa y se ejecuta por una sola vez, es decir que dentro de la función “void setup” se debe declarar tanto los pines de entrada como los de salida (análogos y digitales); así como la comunicación serial entre Arduino y la PC. Cabe indicar que los pines analógicos del controlador Arduino vienen por defecto como entradas, por lo que no es necesario declararlas en dicha función (void setup).

Estructura:

```
void setup()
{
//Configuración inicial
}
```

2.3.4.2. Void loop

La función “void loop” es aquella que se ejecuta de manera cíclica (continua) y es donde escribimos la mayor parte de código, es decir que dentro de esta función se realizan las notaciones de una serie de instrucciones u órdenes capaces de realizar un determinado fin; manteniendo de esta manera la ejecución de la función (void loop) durante todo el tiempo en que la placa permanece energizada.

Estructura:

```
void loop()
{
//Instrucciones u órdenes
}
```

2.3.4.3. Funciones

Una función en el lenguaje de programación C se refiere a un bloque de código el mismo que está compuesto por un identificador (nombre) y una

serie de instrucciones capaces de resolver un determinado problema informático. De manera que la utilización de una función dentro de la programación, se debe principalmente a la capacidad que tiene dicha función en descomponer un gran problema en una tarea muy simple; así como la reducción notable del código escrito. Son funciones las ya mencionadas estructuras “void setup” y “void loop”, las mismas que pueden contener un sinnúmero de funciones en el interior de su estructura.

Cuando una función es llamada, esta devuelve un valor, por lo que es necesario indicar el tipo de dato al que pertenece, como por ejemplo un dato numérico de tipo entero “*int*”; mientras que sí la función no devuelve ningún valor entonces se dice que es una función vacía, por lo que se utiliza la palabra “void” al inicio de la misma. Por otra parte también es necesario indicar el nombre de dicha función, así como los parámetros a ejecutarse.

Sintaxis

```
Tipo nombreFunción(parámetros)
{
Instrucciones;
}
```

2.3.4.4. Entre llaves “{}”

Las llaves son utilizadas para indicar el principio y final de un bloque de instrucciones o estamentos, es decir que durante la llamada de una función o la veracidad de una condición (setup(), loop(), if, for, while…, etc.) el código fuente contenido en el interior del conjunto “{}” es ejecutado.

Estructura lógica:

```
Tipo de función()
{
Instrucciones;
}
```

2.3.4.5. Punto y coma ";"

El punto y coma es utilizado para indicar el final de una línea de código, es decir nos permite separar instrucciones con el fin de facilitar la comprensión del código escrito; así como para mantener una estructura mejor organizada.

Ejemplo:

int ledPin = 13**;** ***//***Declarar a la variable "ledPin" como un entero de valor 13

2.3.4.6. Bloque de comentarios "/*...*/"

Un bloque de comentarios es utilizado para describir en detalle una parte de código escrito o bien para realizar cualquier tipo de aclaración con el fin de facilitar la lectura de dicho código. Es decir que un bloque de comentarios es equivalente a una nota, la misma que es ignorada por nuestro copilador durante su ejecución y no ocupa recursos en la memoria interna por lo que puede ser usada con liberalidad.

/*Este es un bloque de comentarios donde podemos aclarar o describir una parte de código escrito con el fin de facilitar al programador su comprensión****/**

2.3.4.7. Línea de comentarios "//"

Una línea de comentarios es utilizada para realizar cualquier tipo de explicación, descripción o documentación acerca del código escrito; es decir que es equivalente a un bloque de comentarios (*véase bloque de comentarios*), con la única diferencia de que un bloque de comentarios es utilizado cuando existe la necesidad de escribir mucha información; mientras que en una línea de comentarios únicamente se escribe un titulillo de información o aclaración.

digitalWrite(ledPin, HIGH); ***//***Enciende el LED

2.3.5. VARIABLES

Una variable en programación se refiere a una estructura de datos que pueden cambiar de valor durante la ejecución de un programa. Estas variables se caracterizan por ocupar un espacio determinado en la memoria según el tipo de dato a utilizar; es así que durante su declaración es necesario indicar el tipo de variable, el nombre o etiqueta y el valor asignado a la misma.

El tipo de dato se refiere a un conjunto de valores permitidos que para su utilización dependerá de la necesidad del programador y de los recursos (memoria) que dispone. Es así que al utilizar un tipo de dato como por ejemplo “int”, se declara entonces que la variable es de tipo entero de 16 bits comprendido entre un intervalo de -32768 y +32767; mientras que en una variable numérica de tipo extendida “long” (32 bits) podemos encontrar un conjunto de enteros comprendidos entre -2147483648 y +2147483647. Es por esto que es recomendable utilizar un tipo de variable adecuada ya que de esta dependerá la precisión, velocidad y recursos ocupados en la memoria.

Durante la declaración de una variable es necesario identificarla con un nombre de carácter alfanumérico, el mismo que debe estar relacionado con la información que guardará; ya que de esta manera podremos evitar inconvenientes durante la programación. El lenguaje de programación C posee la peculiaridad de que durante la identificación de una variable (etiqueta), la utilización de mayúsculas o minúsculas puede crear muchas variables con un mismo nombre como por ejemplo la variable ledPin no será igual a la variable ledpiN, ledpin o leDpin. Por otra parte es importante aclarar que está prohibido emplear signos y acentos ortográficos, así como la utilización de números (al comienzo) en el nombre o etiqueta de una variable.

Además de declarar el tipo de dato y el nombre de la variable, es necesario

asignarle un valor inicial a la misma lo que comúnmente se lo conoce como inicialización del dato. Este valor será equivalente al tipo de dato utilizado como por ejemplo si el tipo de dato corresponde a *"float"* (punto flotante), el valor asociado a este será un decimal o por lo contrario si utilizamos un dato de tipo *"int"* su valor corresponderá a un entero.

int unaVariable = **12**;
float unaVariable = **4.3**;

2.3.6. TIPOS DE DATOS

Existen varios tipos de datos los cuales tienen como función definir las características que posee una entidad, es decir que el tipo de dato de una variable indica el tamaño en bits a utilizar, el rango de los valores y la interpretación de los mismos. Cabe indicar que los tipos de datos más utilizados corresponden a los datos booleanos, enteros, decimales y caracteres como se detalla en la siguiente tabla.

Tabla 3. Tipos de datos (Benjumea, 2013)

Tipo de dato	Descripción	Tamaño en bits	Rango	
			Min	Max
Bool	Dato de tipo booleano	8 bits	0	1
Byte	Almacena un número entero sin signo	8 bits	0	255
int	Números enteros	16 bits	-32,768	32,767
long	Números enteros de tipo extendido	32 bits	-2,147,483,648	2,147,483,647
float	Números decimales	32 bits	-3.4028235E+38	3.4028235E+38
double	Números decimales de tipo extendido	64 bits	$\pm 1{,}7*10^{-308}$	$\pm 1{,}7*10^{308}$
char	Almacena un carácter	8 bits	-128	127
array	Conjunto finito de datos	8 bits + X	Colección de variables	

2.3.7. OPERADORES

2.3.7.1. Operadores matemáticos

Los operadores aritméticos binarios están compuestos por la suma, resta, multiplicación, división y el operador módulo o residuo (+, -, *, /, %) respectivamente. Estos operadores aritméticos nos permiten realizar operaciones entre operados: números, constantes o variables. Cuyo resultado siempre es un número, el mismo que depende del tipo de dato elegido para los operandos (int, double, float, etc.). En la tabla 4 se puede observar el resultado de una operación matemática de acuerdo al tipo de dato utilizado: donde "x" corresponde a una variable de tipo entero *int*; mientras que "v" es una variable de tipo real *float*.

Tabla 4. Operadores aritméticos (Cairó, 2006)

Operador aritmético	Operación	Ejemplos	Resultados
+	Suma	X =4.5 + 3; V =4.5 + 3;	X = 7 V = 7.5
-	Resta	X =4.5 - 3; V =4.5 - 3;	X = 1 V = 1.5
*	Multiplicación	X = 4.5 * 3; V = 4.5 * 3; V = 4 * 3;	X = 12 V = 13.5 V= 12.0
/	División	X = 4 / 3; X = 4.0 / 3.0; V = 4 / 3; V = 4.0 / 3; V = (float) 4 / 3; V = ((float) 5 + 3) / 6;	X = 1 X = 1 V = 1.0 V = 1.33 V = 1.33 V = 1.33
%	Módulo (residuo)	X = 15 % 2; V = (15 % 2) / 2; V = ((float) (15 % 2)) / 2;	X = 1 V = 0.0 V = 0.5

2.3.7.2. Operadores de comparación

Los operadores de comparación como su nombre lo indica comparan dos operandos ya sean variables o constantes, que generalmente son utilizados en estructuras condicionales del tipo *if*, dando como resultado valores booleanos, es decir verdadero (true) o falso (false). Que de acuerdo a la operación (condición) indicada se verificará su veracidad o falsedad.

Tabla 5. Operadores de comparación

Símbolo	Significado	Ejemplo	
==	igual que	x == y	// x es igual a y
! =	diferente de	x ! = y	// x no es igual a y
<	menor que	x < y	// x es menor que y
>	mayor que	x > y	// x es mayor que y
<=	menor o igual a	x <= y	// x es menor o igual que y
>=	mayor o igual a	x >= y	// x es mayor o igual que y

2.3.7.3. Operadores compuestos

Los operadores compuestos son propios del lenguaje de programación C y permiten combinar una operación aritmética con una variable asignada, con el fin de simplificar y clarificar la estructura del código fuente.

Tabla 6. Operadores Compuestos (Fajardo, 2012)

Símbolo	Significado	Ejemplo	
++	Incremento	x ++	// incremento de x en +1
--	Decremento	x --	// decremento de x en -1
+=	Compuesto Adición	x += y	// incremento de x en +y
-=	Compuesto Substracción	x -= y	// decremento de x en -y
*=	Compuesto Multiplicación	x *= y	// multiplica x por y
/=	Compuesto División	x /= y	// divide x por y

Es importante indicar que el lenguaje de programación C proporciona dos operadores para incrementar (++) y decrementar (--) variables, y se caracterizan por ser un poco especiales ya que se los puede utilizar antes de la variable como prefijos y después de la variable como postfijos; así como lo demuestra la siguiente tabla:

Tabla 7. Operadores de incremento y decremento (Benjumea, 2013)

Sintaxis	**Ejemplo**	
++ <unaVarible>	y = ++x	//pre-incremento en x
<unaVariable> ++	y = x++	//post-incremento en x
-- <unaVarible>	y = --x	//pre-decremento en x
<unaVariable> --	y = x--	//post-decremento en x

2.3.7.4. Operadores lógicos

Los operadores lógicos o booleanos son generalmente utilizados para desarrollar condicionales complejos a partir de condiciones simples, y se caracterizan por comparar dos expresiones para devolver un valor (verdadero o falso) como resultado.

Existen tres tipos de operadores lógicos como son: la negación “NO” **(!)**, la disyunción “O” **(||)** y la conjunción “Y” **(&&)**, los mismos que son utilizados por lo general en el condicional de tipo “if” así como se muestra en la siguiente tabla.

Tabla 8. Operadores Lógicos

Operador	**Operación**	**Ejemplo**	
&&	Conjunción “Y”	If(x ≥ 1 && x ≤ 10)	//Es cierto si las dos expresiones son ciertas
||	Disyunción “O”	If(x > 0 || y > 0)	//Es cierto si una de las dos expresiones es cierta
(!)	Negación “NO"	If(!x = 0)	//Es cierto si la expresión es falsa

2.3.8. CONSTANTES

Una constante es un nombre o identificador cuyo valor no puede cambiar durante la ejecución de un programa. Este valor puede ser asignado en el caso de ser una constante de tipo simbólica; o puede estar incluido en el compilador al tratarse de una constante literal.

2.3.8.1. Constantes simbólicas

En el lenguaje de programación C una constante de tipo simbólica se declara mediante la utilización de la palabra reservada "*const",* el tipo de dato a utilizar*,* el identificador o nombre para dicha constante y el valor asignado a la misma. Cabe indicar que el identificador de una constante generalmente utiliza letras mayúsculas como en el siguiente ejemplo:

Sintaxis

Const <Tipo de dato> <Identificador> = Valor

Ejemplo:

Const int MAXIMO = 5000;

2.3.8.2. Constantes literales

Una constante de tipo literal se caracteriza por tener un valor predeterminado o ya asignado en un identificador, y tiene la finalidad de facilitar al programador la escritura y lectura del código fuente. Dentro de este tipo de constantes (literales) podemos destacar las más utilizadas:

2.3.8.2.1. Constantes booleanas "bool"

Como su nombre lo indica son constantes lógicas donde puede existir un solo valor TRUE or FALSE, uno (1) o cero (0) respectivamente.

Tabla 9. Constantes booleanas

1 Lógico	0 Lógico
TRUE	FALSE
HIGH	LOW

2.3.8.2.2. Constantes de carácter “char”

Una constante del tipo “char” es un entero representado por un carácter delimitado entre comillas simples (’a’, ’/n’, ’+’...) perteneciente a un grupo de caracteres (ASCII) del ordenador. Estos caracteres se caracterizan por ser alfabéticos, numéricos y especiales.

Tabla 10. Constantes de carácter

Caracteres	Constantes (char)
Caracteres alfabéticos	A, B, C,...,Z, a, b, c,...,z
Caracteres numéricos	1, 2, 3...,9
Caracteres especiales	/,+,-,<,...

A continuación se puede observar los caracteres de escape más utilizados o reservados dentro del lenguaje de programación C; y por ende en el entorno de programación IDE de Arduino.

Tabla 11. Caracteres de escape (Benjumea, 2013)

’\n’: salto de línea (newline)
’\r’: retorno de carro (carriage-return)
’\b’: retroceso (backspace)
’\t’: tabulador horizontal
’\v’: tabulador vertical
’\f’: avance de página (form-feed)
’\a’: sonido (audible-bell)
’\0’: fin de cadena

2.3.8.2.3. Constantes enteras

Son constantes de tipo numérico donde no puede existir espacios en blanco u operadores, y son expresadas en sistema decimal (base 10), hexadecimal (base 16) y octal (base 8). El sufijo “L” es utilizado para expresar un tipo de dato long (20000L); mientras que el sufijo “UL” expresa una constante sin signo (unsigned) del tipo long (20000UL).

2.3.8.2.4. Constantes reales

Son constantes de tipo punto flotante y se caracterizan por tener un punto decimal (123,5) o un exponente (1e-3), donde el sufijo “F” corresponde a float (3,47e-1F); mientras que el sufijo “L” especifica long double (2,56e300L).

2.3.9. ESTRUCTURAS DE CONTROL DE FLUJO

2.3.9.1. Condicional “if”

El condicional “if” (SI) en el lenguaje de programación C es utilizado para verificar la veracidad o falsedad de una expresión (condición) establecida, y de esta manera ejecutar o no las instrucciones u operaciones escritas dentro de una proposición compuesta o bloque “{}”.

Sintaxis

```
If (expresión)
{
Instrucciones o proposiciones
}
```

2.3.9.2. Condicional “if-else”

La condición “If-else” a diferencia del condicional “If” es que si la expresión de la proposición 1 no se cumple; entonces se ejecuta la proposición 2

escrita en “else”. Es decir cumple con la expresión “sí esto no se cumple, haz lo otro”.

Sintaxis

```
If(expresión)
{
Proposición 1          //Si es VERDAD ejecuta proposición 1
}
else
{
Proposición 2          //Si es FALSO ejecuta proposición 2
}
```

Ejemplo:

```
If(valor >= 600)                    //Si valor es mayor o igual a 600
{
digitalWrite(ledPin, HIGH);         //Enciende el LED en la variable ledPin
}
Else                                //Sino
{
digitalWrite(ledPin, LOW);          //Apaga el LED en la variable ledPin
}
```

Es importante indicar que el condicional “if-else” puede estar precedido por otra condición es decir una dentro de otra “else-if”, formando de esta manera una cadena o estructura con un número ilimitado de condicionales así como se lo muestra en el siguiente ejemplo:

```
If(inputPin < 400)
{
InstruccionesA;
}
else If(inputPin >= 800)
{
InstruccionesB;
}
else
{
IntruccionesC;
}...
```

2.3.9.3. Sentencia “for”

La declaración “for” (PARA) en el lenguaje de programación C es utilizada generalmente para repetir un número definido de veces una secuencia de instrucciones, donde el bucle “for” se encuentra definido por tres argumentos los mismos que se encuentran separados por un punto y coma “;” y son: sentencia inicial, condición e incremento.

Sintaxis del bucle for

for(*sentencia inicial; condición; incremento*)
{
Instrucciones
}

Ejemplo:

```
for(int i=0; i<10; i++)          //Declara la variable “i” y prueba sí es
{                                  menor que 10; incrementa uno en “i”.
digitalWrite(ledPin, HIGH);      //Enciende el LED
delay (1000);                    //Espera un segundo
digitalWrite(ledPin, LOW);       //Apaga el LED
delay(1000);                     //Espera un segundo
}
```

Como puede verse en el ejemplo anterior la variable “i” controla el número de veces deseado para que se ejecute el bucle, donde la asignación “i=0” viene a ser el valor asignado a la variable o sentencia inicial; mientras que la condición “i<10” nos permite ajustar el número de veces en cumplirse dicha condición, para posteriormente incrementar en uno la variable “i++” cada vez que se ejecuten las instrucciones (bucle).

Al asignar a la variable “i” un valor inicial de cero y una condición donde los valores deben ser menores a diez, la sentencia “for” se ejecutará desde un principio ya que cumple con dicha condición durante su comprobación; la misma que es verificada antes de cada iteración con el fin de aseverar su valor de verdad.

2.3.9.4. Sentencia “while”

La declaración “*while*” (CUANDO) es utilizada al igual que el condicional *if* para verificar el valor de verdad de una expresión o condición. La misma que si resulta ser cierta se ejecutará la sentencia o grupo de sentencias dentro del bucle; por otra parte si la condición resulta ser falsa, la ejecución del programa saltará a la siguiente instrucción tras el bucle.

Sintaxis de la sentencia while

```
while(condición)                //Ejecución repetida de una sola sentencia
{
sentencia;
}

while(condición)                //Ejecución repetida de un grupo de sentencias
{
sentencias;
}
```

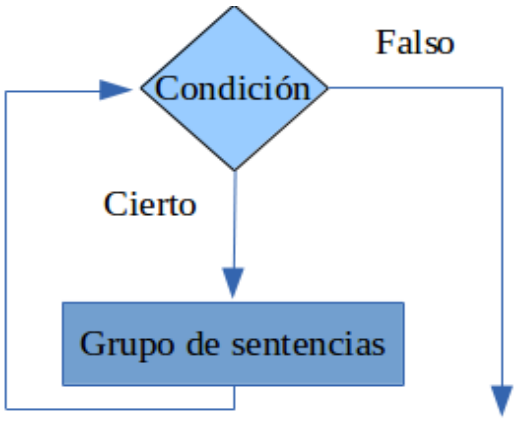


Figura 31. Algoritmo condicional

Ejemplo:

```
while(sensor < 400)         //Testea sí la variable sensor es menor que 400
{
digitalWrite(ledPin, HIGH);         //Enciende el LED
delay (1000);                       //Espera un segundo
digitalWrite(ledPin, LOW);          //Apaga el LED
delay(1000);                        //Espera un segundo
}
```

En este ejemplo podemos observar que la condición únicamente se cumple cuando el valor de la variable *sensor* es menor al indicado (400), es decir que el condicional while puede o no cumplirse desde un principio por lo que el bucle o bloque de instrucciones puede ejecutarse cero o más veces. Sí la condición resulta ser cierta, entonces un indicador visual en este caso alertará al usuario.

2.3.9.5. Sentencia "do-while"

La sentencia "*do-while"* al igual que el condicional *while*, se repite mientras la condición sea verdadera; caso contrario el bucle termina. La diferencia de esta estructura con la anterior radica en que la condición se prueba al final del bucle; por lo que el bloque de instrucciones se habrá ejecutado al menos una vez.

Sintaxis de la sentencia do-while

```
do
{
Instrucciones;
}
while(condición);
```

Ejemplo:

```
do
{
digitalWrite(ledPin, HIGH);          //Enciende el LED
delay (3000);                        //Espera tres segundo
}
while(potenciometro == 1023);        //Cuando la variable potenciometro
                                       es igual a 1023
```

Como es posible observar primero se ejecutan las instrucciones escritas entre llaves "do{...}", para posteriormente comprobar la condición "while" que en este caso la variable "potenciometro" debe ser igual a 1023. Es decir que por lo menos una vez se ejecutará el bucle al inicio y luego si la condición es cierta volverá a ejecutarse hasta que esta resulte ser falsa.

2.3.10. ENTRADAS-SALIDAS DIGITALES “E/S”

2.3.10.1. pinMode()

La instrucción “pinMode” en el compilador IDE de Arduino es utilizada para declarar el estado inicial de los pines, por lo que dicha instrucción es expresada dentro de la función “void setup”. Es decir que el comando *pinMode* permite configurar las entras o salidas de un determinado pin, donde una entrada es declarada como INPUT; mientras que a una salida se la considera como OUTPUT.

Sintaxis de la instrucción pinMode

pinMode(número de pin, estado);

Ejemplo:

```
pinMode(13, OUTPUT);            //Declara al pin 13 como una salida
```

Por otra parte también es posible asignar a una variable el valor del pin deseado antes de la función *“void setup”*, donde lógicamente la variable será un tipo de dato entero (int); ya que no existe un pin 10.5 como para utilizar un dato del tipo punto flotante (float).

Ejemplo:

```
int ledPin = 10;            //Asigna a la variable ledPin el valor entero 10
int sensorDigital = 7;      //Asigna a la variable sensorDigital el valor entero 7

void setup()
{
pinMode(ledPin, OUTPUT);            //Declara al pin 10 como una salida
pinMode(sensorDigital, INPUT);      //Declara al pin 7 como una entrada
}
```

Como se puede observar en el ejemplo las variables son asignadas antes de

la función *void setup*, para posteriormente declarar las entradas y salidas utilizando dichas variables en el interior de comando *pinMode().*

2.3.10.2. digitalRead()

Esta instrucción debe ser declarada en el interior del bucle *"void loop",* la cual tiene como función leer el estado de una entrada digital en un pin definido con anterioridad. Es decir que por medio de este comando se puede conocer el valor dado por dicha entrada, la misma que puede estar en un único estado HIGH (ALTO, 5v) o LOW (BAJO, 0v).

Sintaxis

digitalRead(pin)

Ejemplo:

```
int pulsador = 5;              //Asigna a la variable pulsador el valor entero 5
int ledPin = 10;               //Asigna a la variable ledPin el valor entero 10


void setup()
{
pinMode(pulsador, INPUT);      //Declara a la variable pulsador como entrada
pinMode(ledPin, OUTPUT);       //Declara a la variable ledPin como una salida
}


void loop()
{
if(digitalRead(pulsador) == HIGH)     //Si el pulsador se encuentra oprimido
{
digitalWrite(ledPin, HIGH);           //Enciende el LED
delay(1000);                          //Espera un segundo
digitalWrite(ledPin, LOW);            //Apaga el LED
delay(1000);                          //Espera un segundo
}
}

//Fin del Programa
```

2.3.10.3. digitalWrite()

La instrucción digitalWrite es uno de los comandos más utilizados dentro de la función *"void loop"*, ya que por medio de este recurso podemos enviar el valor HIGH o LOW (1 o 0 respectivamente) a un pin definido con anterioridad como una salida (OUTPUT).

Sintaxis

digitalWrite(Pin, Valor);

Ejemplo:

```
void setup()
{
pinMode(13, OUTPUT);          //Define al pin 13 como una salida
}

void loop()
{
digitalWrite(13, HIGH);       //Situar al pin 13 en estado HIGH (alto o 1)
delay (1000);                 //Espera un segundo
digitalWrite(13, LOW);        //Situar al pin 13 en estado LOW (bajo o 0)
delay(1000);                  //Espera un segundo
}
```

2.3.11. ENTRADAS - SALIDAS ANÁLOGAS "E/S"

2.3.11.1. analogRead()

El comando *"analogRead"* tiene como función leer el valor de tensión de una entrada análoga definida con anterioridad; este valor estará comprendido entre 0 y 1023 ya que el convertidor análogo-digital del hardware a utilizar (Arduino) posee una resolución de 10 bits.

Sintaxis

analogRead(pin)

Ejemplo:

```
int potenciometro = A3;      //Define al pin análogo A3 como potenciometro
int ledPin = 13;             //Define al pin 13 como ledPin
int valorPot = 0;            //Asigna a la variable valorPot en valor de inicio

void setup()
 {
pinMode(ledPin, OUTPUT);    //Declara a la variable ledPin como salida

/*Nota NO es necesario declarar el pin análogo (A3) como una entrada ya que por defecto de fábrica todas las entradas análogas son definidas como entradas IN*/
}

void loop()
{
valorPot = analogRead(potenciometro);
// Lee el valor del potenciómetro y almacenar dicho dato en la variable valorPot
digitalWrite(ledPin, HIGH);
//Enciende el LED
delay(valorPot);
//Detén el programa durante <valorPot> en milisegundos
digitalWrite(ledPin, LOW);
//Apaga el LED
delay(valorPot);
//Detén el programa durante <valorPot> en milisegundos
}
//Fin del programa
```

En este ejemplo un LED se encenderá y apagará proporcionalmente a la lectura obtenida (analogRead) de la señal variable, es decir que si variamos la resistencia del potenciómetro utilizado; también variará la salida digital que en este caso es un indicador visual (LED).

2.3.11.2. analogWrite()

La instrucción *"analogWrite"* tiene la función de simular una salida análoga en una salida digital, mediante la variación del tiempo de encendido (ON) y apagado (OFF), conocido también como "modulación por ancho de pulso" o PWM; esta opción puede ser utilizada dentro del hardware libre de Arduino

(ATmega168 o ATmega328) en los pines 3, 5, 6, 9, 10 y 11. Los valores enviados por dichos pines de salida análoga pueden ser variables o constantes pero siempre con un intervalo comprendido entre 0 y 255, donde dicho margen equivale a 0 y 5 voltios respectivamente.

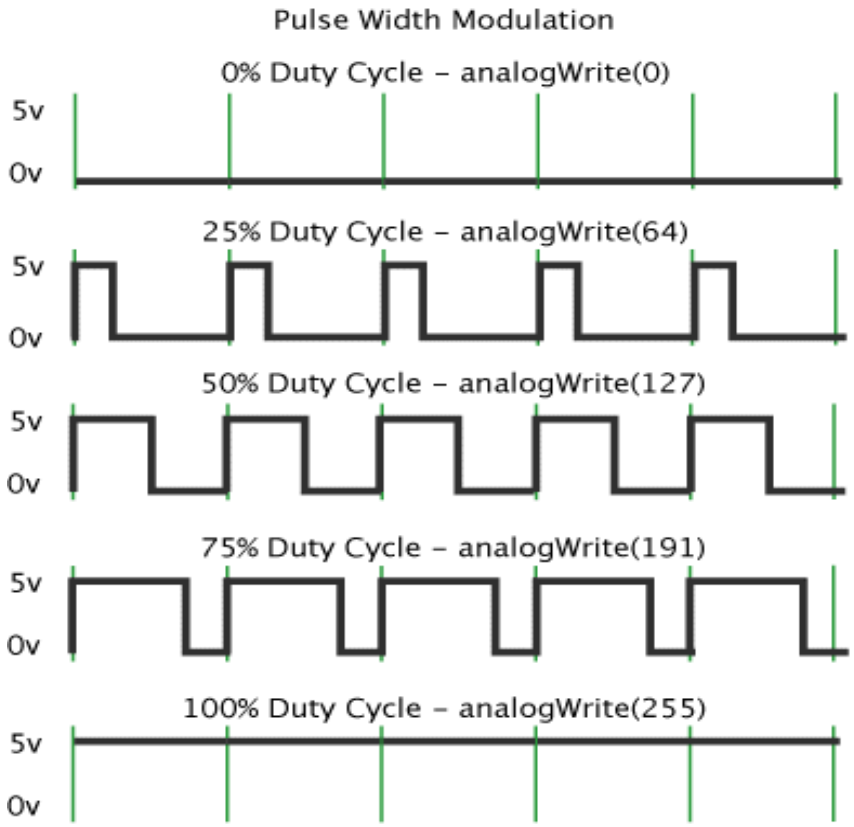


Figura 32. Modulación por ancho de pulso (arduino.cc, 2014)

Sintaxis

analogWrite(pin, valor);

Ejemplo:

```
int ledPin = 10;                //LED conectado al pin digital 10
int potenciometro = A3;         //Potenciómetro conectado al pin análogo A3
int valorPot = 0;               //Variable en el que se almacena el dato leído

void setup()
{
  pinMode(ledPin, OUTPUT);      //Declara al pin 10 como salida
}
```

```
void loop()
{
valorPot = analogRead(potenciometro);
//Asigna a la variable valorPot el valor leído en el pin A3

analogWrite(ledPin, valorPot / 4);
//Simula una salida análoga (PWM) en el pin 10, de acuerdo al valor leído en el pin análogo (A3) dividido para 4.
}
```

Como es posible analizar en este ejemplo el LED se encenderá y apagará de acuerdo al ancho o tiempo modificado en un ciclo de trabajo; el mismo que dependerá del valor leído en la entrada análoga (A3), donde dicho valor es dividido para 4 ya que la instrucción *"analogRead"* tiene un intervalo entre 0 y 1023; mientras que el comando *"analogWrite"* posee un valor comprendido entre 0 y 255.

2.3.12. CONTROL DEL TIEMPO

2.3.12.1. delay()

La instrucción *"delay"* es definida como un retardador del programa en ejecución por un tiempo determinado en milisegundos. Es decir que por medio de dicho comando se puede manipular el tiempo de espera (pausa) deseado durante la ejecución del bucle, donde 1000 milisegundos equivale a 1 segundo.

Sintaxis

delay(ms)

Ejemplo:

```
void loop()
{
  digitalWrite(ledPin, HIGH);      //Enciende el LED
  delay(1000);                     //Espera durante un segundo
  digitalWrite(ledPin, LOW);       //Apaga el LED
  delay(1000);                     //Espera durante un segundo
}
```

2.3.13. FUNCIONES MATEMÁTICAS

2.3.13.1. min()

La instrucción *"min"* nos permite calcular el mínimo entre dos números (x, y) para cualquier tipo de dato, donde "x" corresponde al primer número e "y" al segundo número declarado entre paréntesis "()", los mismos que serán evaluados con la finalidad de obtener el menor valor entre ellos.

Sintaxis

min(x, y)

Ejemplo:

valor = **min(i, 200);** //Asigna a valor el mínimo entre los dos números especificados

2.3.13.2. max()

La instrucción *"max()"* nos permite calcular el máximo entre dos números (x, y) especificados para cualquier tipo de dato, es decir que nos devuelve el valor mayor entre ambos parámetros.

Sintaxis

max(x,y)

Ejemplo

valor = **max(valSensor, 500);**
//Asigna a valor el máximo entre los dos números especificados

2.3.14. COMUNICACIÓN SERIE

2.3.14.1. Serial.begin(rate)

Este comando permite establecer la velocidad de transmisión entre la comunicación del controlador Arduino y la PC. Cabe indicar que la velocidad

típica o por defecto es de 9600 baudios; aunque es posible establecer otras velocidades de transmisión.

```
void setup()
{
Serial.begin(9600);       //Abre el puerto serie y establece la comunicación a
}                           9600 baudios por segundo (bps)
```

2.3.14.2. Serial.printIn(data)

Este comando es muy utilizado sobre todo cuando de depurar el código se trata, ya que por medio del Monitor Serie de Arduino (IDE), es posible leer los datos enviados (impresos) al puerto serie; con el fin de analizar y verificar el correcto funcionamiento del programa.

En el siguiente ejemplo los valores tomados del pin analógico A0 son enviados al ordenador (Monitor Serie) cada segundo.

```
void setup()
{
Serial.begin(9600);       //Abre el puerto serie y establece la comunicación a
}                           9600 bps

void loop()
Serial.printIn(analogRead(A0));    //Envía el valor leído en el pin análogo cero
delay(1000);                       //Espera un segundo
```

2.3.14.3. Serial.print(data, data type)

El siguiente comando imprime un número o una cadena de caracteres en el Monitor Serie de acuerdo al formato utilizado, es decir que si se utiliza un tipo de dato binario (BIN) para un valor “X”, este último pasará a tomar el valor binario correspondiente. Donde DEC (decimal), OCT (octal), BIN (binario), HEX (hexadecimal).

```
int a = 79   //el valor del carácter “a” es 79
```

```
Serial.print(a, BIN);  //envía el valor “1001111”
Serial.print(a, HEX); //envía el valor “4F”
```

3. METODOLOGÍA

3.1. SOFTWARE Y HARDWARE LIBRE

3.1.1. SOFTWARE LIBRE

Cuando se habla de software libre nos referimos a la libertad que tienen los usuarios para ejecutar, copiar, distribuir, estudiar, modificar y mejorar el código fuente; dicho de otra manera el software libre promueve el derecho a la libertad que tiene toda persona, al permitir el total acceso y control (código fuente) de un programa basado en la filosofía libre. Cabe indicar que el software libre no es sinónimo de gratis como se lo confunde por su terminología “free software”.

Figura 33. GNU (Fajardo, 2012)

Se dice que un programa no es libre (privativo) cuando este limita el conocimiento al restringir el libre acceso al programa; obligándonos aceptar una serie de términos y condiciones (licencias) que prohíben la libre expresión de estudiar, modificar, distribuir, mejorar y acceder al conocimiento (código fuente). Frenando de cierta manera a la tecnología y sobre todo controlando al usuario por medio del programa, el que a su vez es controlado por una entidad (corporación o estado).

Para que un programa pueda ser libre se debe cumplir con cuatro libertades esenciales.

- La libertad de ejecutar el programa con cualquier fin o propósito bajo

su responsabilidad.

- La libertad de estudiar al programa, para lo cual se tiene el total acceso al código fuente.

- La libertad de distribuir copias del programa sin ningún tipo de restricciones.

- La libertad de mejorar el programa y distribuir copias de dicho código modificado.

3.1.2. GNU/LINUX

El término “Linux” proviene de su creador Linus Torvalds quien basándose en UNIX fue capaz de desarrollar el kernel o núcleo central (Linux) de un sistema operativo, el mismo que tiene como función asignar los recursos de un ordenador hacia los diferentes programas ejecutados por el usuario. De esta definición se puede decir que Linux por sí solo es inútil debido a que este necesita complementarse con un sistema operativo libre “GNU”; de allí que es posible encontrar un amplia variedad de sistemas operativos libres y libres-privativos como Trisquel (libre), LinuxMint, Fedora, Debian, Parabola (libre), Dragora (libre), Ubuntu, entre otros sistemas operativos basados en Linux.

3.1.3. HARDWARE LIBRE

Se denomina hardware libre a todos aquellos dispositivos informáticos tangibles que al igual que el software libre nos permiten estudiar, modificar, reutilizar, mejorar y compartir los planos utilizados para su construcción, de manera que cualquier persona es libre de crear sus propios prototipos (diseños) al tener un total acceso a dichos diagramas y especificaciones técnicas.

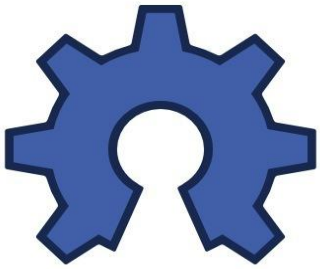


Figura 34. Hardware Libre.
(Fajardo, J. 2012)

Por otra parte la utilización y reutilización de diseños basados en la filosofía libre a facilitado en gran medida el trabajo realizado por los ingenieros de hardware, ya que al disponer de dichos diseños a la mano es posible reducir el tiempo y costos de fabricación; convirtiéndose por ende (hardware libre) en una herramienta versátil y accesible para cualquier persona que desee llevar a cabo una investigación.

3.1.3.1. Arduino

Arduino es una plataforma de hardware libre constituida fundamentalmente por una placa electrónica capaz de recibir, interpretar y enviar señales por medio de un microprocesador; este dispositivo se caracteriza por contar con una serie de entradas y salidas tanto análogas como digitales. Dicho de otra manera, Arduino tiene la capacidad de recoger información proveniente del mundo exterior (entradas), para posteriormente ser enviada hacia a la unidad central de procesamiento, dando como resultado una salida.

Arduino es considerado hardware libre debido a que sus ficheros esquemáticos son de acceso público bajo la licencia de *Creative Commons*, la misma que permite la libertad de citar, crear y reproducir obras derivadas de otras, con la finalidad de reducir las barreras legales de la creatividad. Teniendo de esta manera la capacidad y libertad de crear nuestra propia placa Arduino, ya sea para fines personales o comerciales.

Figura 35. Arduino (arduino.cc, 2014)

Por otra parte es necesario indicar que Arduino es programado mediante el lenguaje de programación *Wiring,* el mismo que es muy similar al lenguaje de programación C, C++ tanto en su estructura como en su sintaxis; mientras que su entorno de desarrollo está basado en “Processing”, una herramienta desarrollada para estudiantes, artistas, diseñadores y aficionados capaces de escribir código por medio de este editor de texto o comúnmente denominado compilador (IDE).

3.2. DESARROLLO DEL PROYECTO

3.2.1. ARQUITECTURA DOMÓTICA UTILIZADA

La arquitectura domótica centralizada fue el modelo a seguir para este diseño, ya que se necesitó de un dispositivo central máster o controlador (Arduino) capaz de procesar la información proveniente del mundo exterior, por medio de una tarjeta de reconocimiento de voz (EasyVR) de tipo shield para Arduino. De manera que los comandos de voz interpretados por el controlador en base a su programación son enviados a los diferentes actuadores con el fin de realizar una determinada acción.

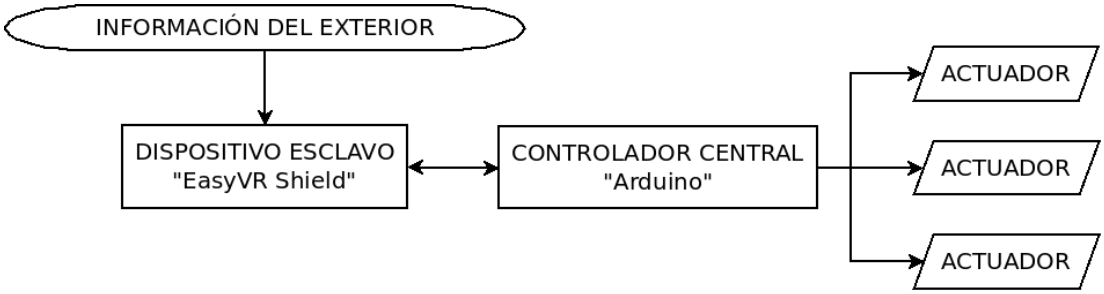


Figura 36. Arquitectura domótica centralizada

3.2.2. DISEÑO EXPERIMENTAL

Para llevar a cabo el siguiente proyecto de investigación se realizó una serie de pruebas experimentales con el fin de garantizar el correcto funcionamiento del sistema. Es importante aclarar que el controlador utilizado para este proyecto corresponde a la tarjeta *Arduino Mega 2560 R3*, la misma que fue programada mediante un ordenador portátil *Dell Inspiron 3421* y un sistema operativo GNU/Linux.

3.2.2.1 Arduino mega 2560 R3

La tarjeta Arduino mega 2560 R3 se destaca por tener mayor espacio en memoria, así como mayor cantidad de pines tanto análogos como digitales a comparación con la tarjeta Arduino UNO. Esta mejorada versión cuenta con un poderoso microcontrolador ATmega 2560 capaz de controlar 54 pines digitales, de los cuales 14 pueden ser utilizados como salidas PWM (modulación por ancho de pulso), 16 entradas análogas y 4 puertos UART. Además posee un oscilador de cristal de cuarzo de 16 MHz y su voltaje de funcionamiento (alimentación) es de 5 voltios por lo que basta con conectarlo mediante el puerto USB; en caso de conectar una fuente externa el voltaje de funcionamiento está comprendido entre los 6 y 20 voltios.

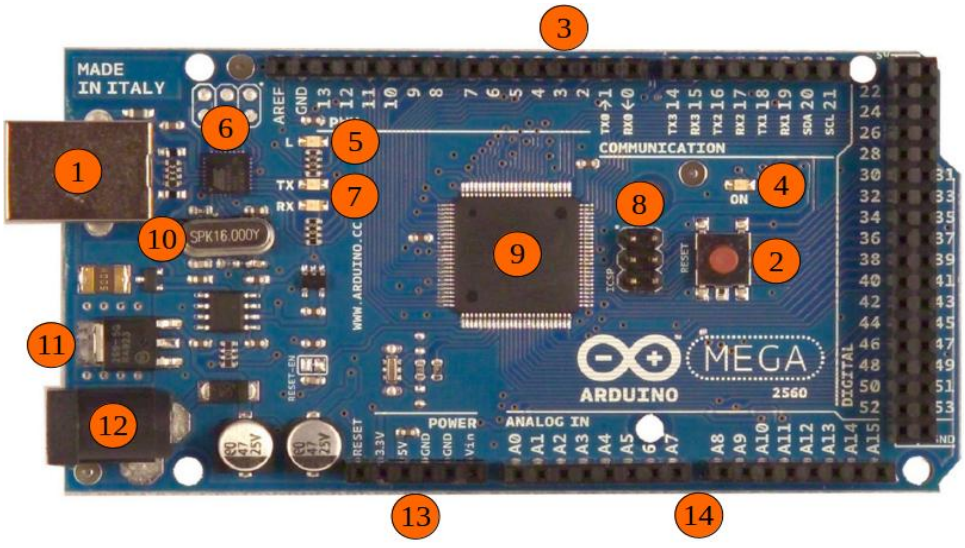


Figura 37. Arduino Mega 2560 R3 y sus partes

1.- Conector USB para el cable Tipo AB

2.- Pulsador Reset

3.- Pines de E/S digitales y PWM

4.- LED verde de placa encendida

5.- LED naranja conectado al pin13

6.- ATmega 16U2 encargado de la comunicación con el PC

7.- LED TX (Transmisor) y RX (Receptor) de la comunicación serial

8.- Puerto ICSP para programación serial

9.- Microcontrolador ATmega 2560

10.- Cristal de cuarzo de 16Mhz

11.- Regulador de voltaje

12.- Conector hembra 2.1mm con centro positivo

13.- Pines de voltaje y tierra

14.- Entradas análogas

3.2.2.2. Instalación del IDE de Arduino

Para proceder a instalar el software que se va a utilizar durante el diseño del código fuente, es necesario descargar el compilador desde la página oficial de Arduino “http://arduino.cc/en/Main/Software”, que de acuerdo al sistema operativo utilizado (GNU/Linux, Mac o Windows) se optará por el correcto.

Cabe indicar que en este caso el sistema operativo utilizado es “Linux Mint 16 petra a 64 bits”, por lo que se procede a descargar el archivo correspondiente (Linux: 64 bits).

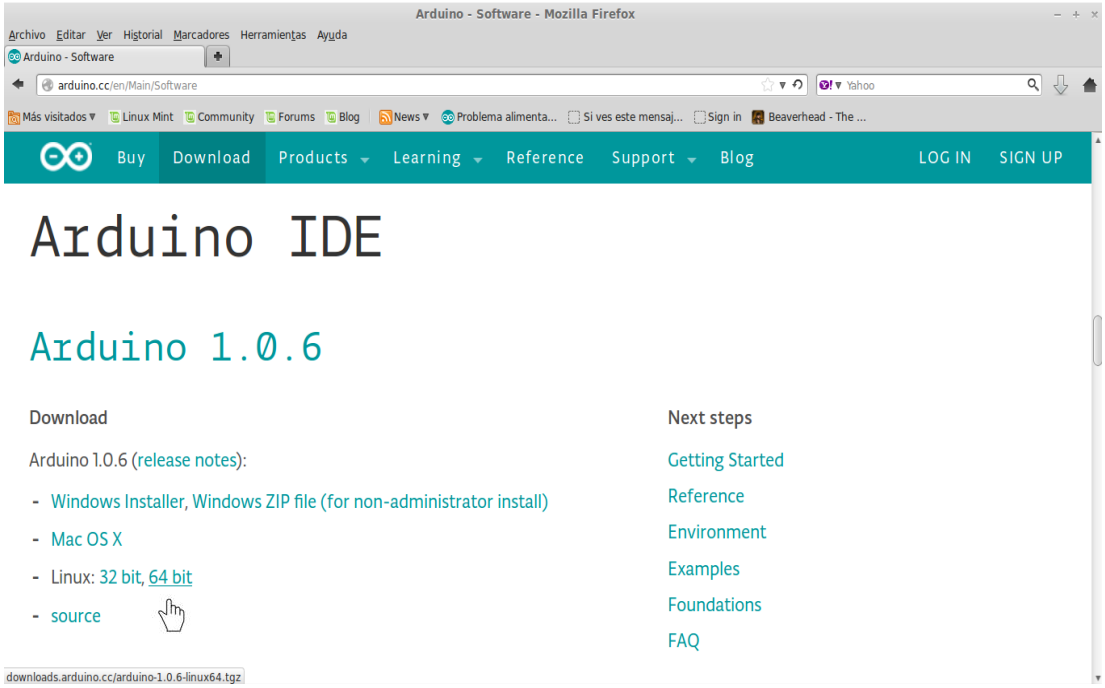


Figura 38. Descarga del compilador Arduino IDE

Una vez que la descarga haya finalizado, se procede abrir la terminal del sistema con el fin de ingresar a la carpeta contenedora del archivo descargado (.tgz), el mismo que en este caso se encuentra en la carpeta "Descargas" por lo que se escribe el directorio "cd /home/*usuario*/Descargas".

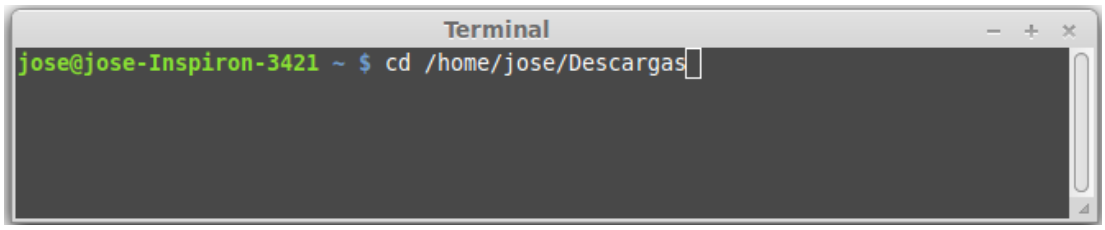


Figura 39. Carpeta contenedora del archivo descargado (.tgz)

A continuación se descomprime el archivo descargado (arduino-1.0.6-linux64.tgz) con el siguiente comando "tar -zxvf nombredelarchivo.tgz" así como lo indica la figura 40.

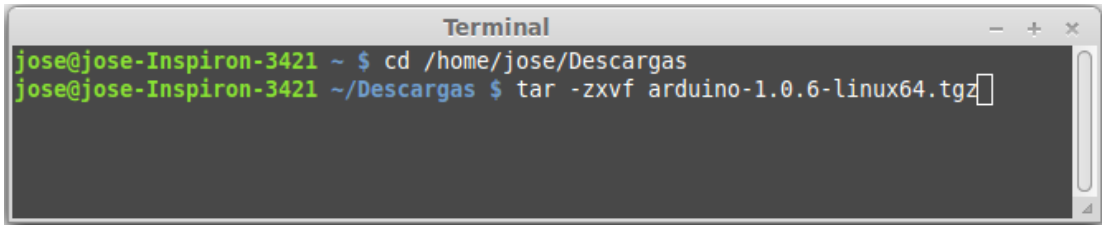


Figura 40. Descompresión del archivo descargado (.tgz)

Ahora ya se tiene el compilador listo para usarse; de manera que en el interior de la carpeta contenedora (Descargas) es posible observar una nueva carpeta denominada “arduino” con la versión más reciente, donde en cuyo interior contiene el archivo de texto ejecutable por medio de la terminal.

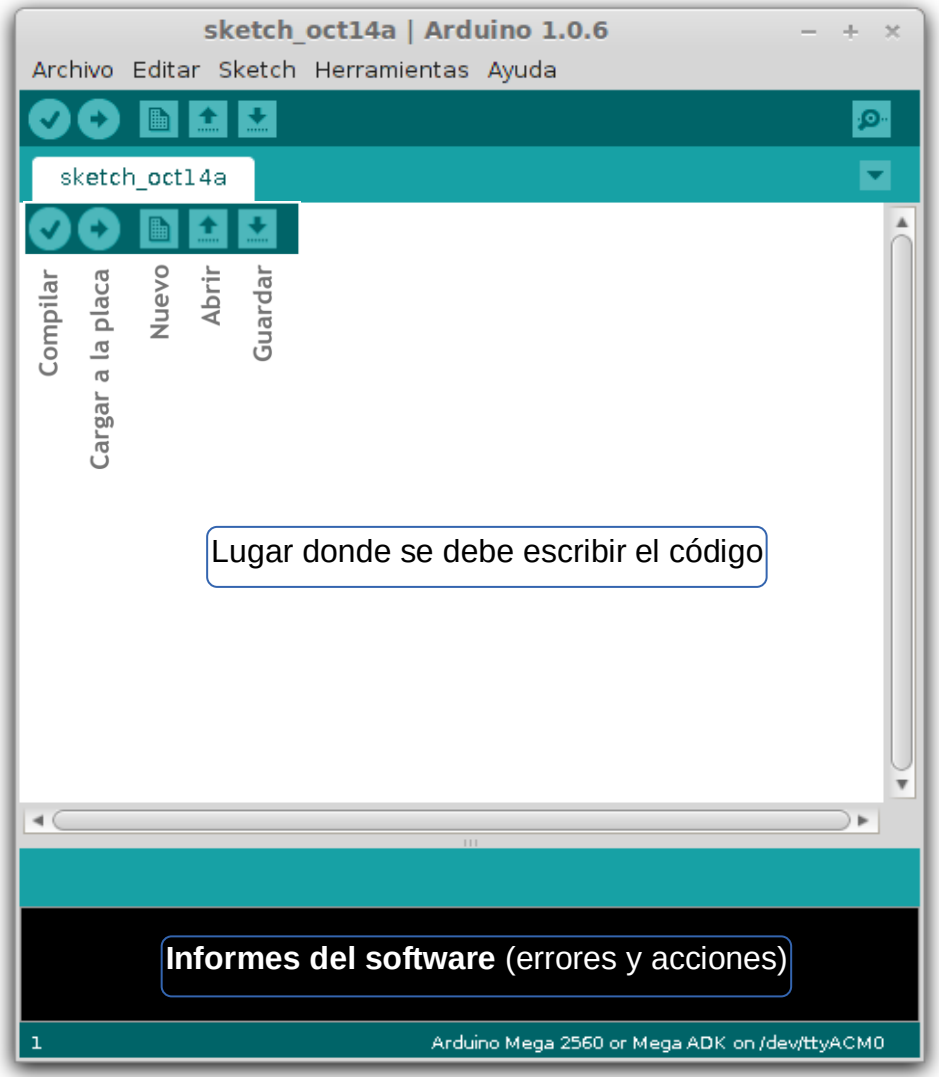


Figura 41. Estructura del compilador de Arduino (IDE)

Cabe indicar que antes de cargar el código escrito al controlador, es importante seleccionar el puerto COM correcto del ordenador; por lo que se procede a conectar la tarjeta al puerto serie por medio del cable USB de tipo AB. En la terminal del sistema se verificará si el controlador a sido reconocido en el puerto USB, para lo cual se procede a listar todos los dispositivos conectados en dicho puerto mediante el comando “lsusb”.

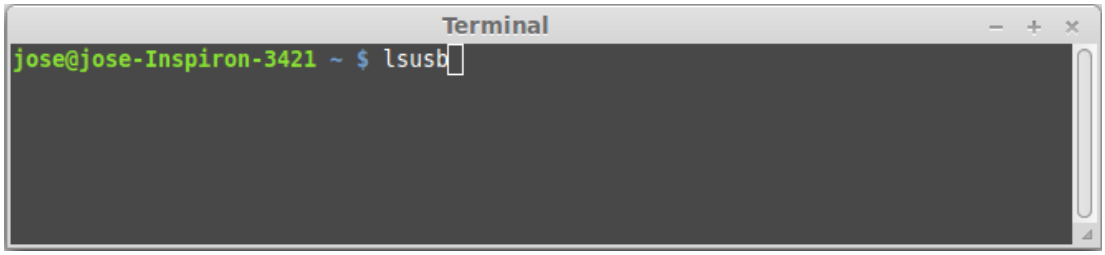


Figura 42. Dispositivos conectados puerto USB

Pulsamos la tecla *“Intro”* y será posible observar al controlador Arduino.

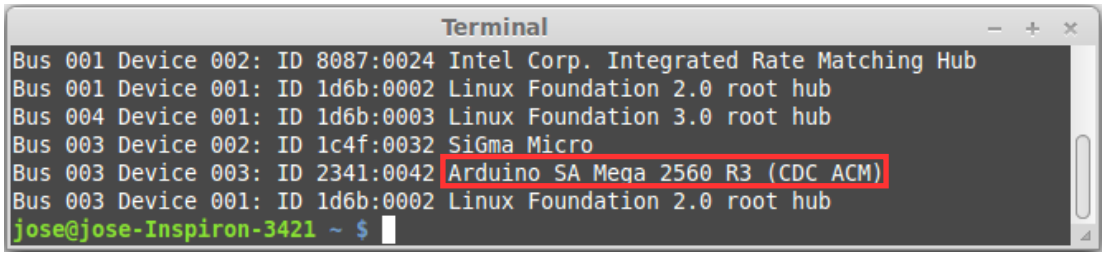


Figura 43. Controlador Arduino Mega 2560 R3 conectado

Ahora se debe determinar el puerto serie (COM) utilizado por dicho controlador, que por lo general es ACM0 aunque puede cambiar; para lo cual se escribe el comando “dmesg” como lo muestra la figura inferior.

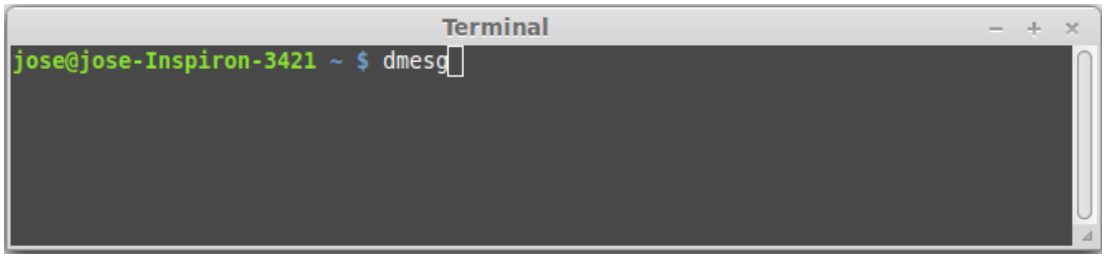


Figura 44. Comando a determinar el puerto COM utilizado

Se presiona *"Intro"* en la terminal y como se puede observar en este caso el puerto COM utilizado corresponde al "ttyACM0" o "COM0" el cual lo anotaremos para no olvidarnos.

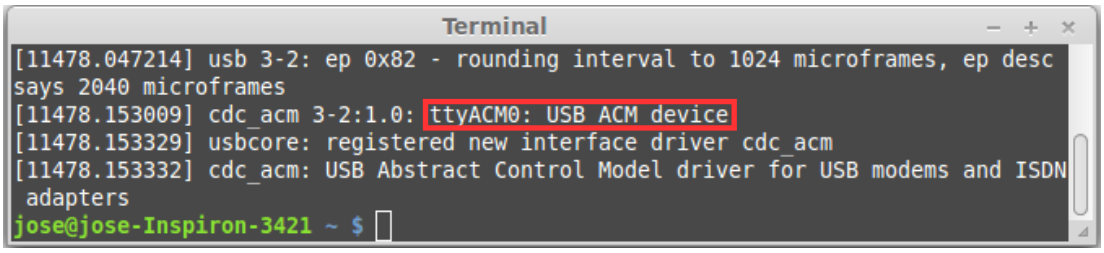


Figura 45. Puerto COM utilizado por el controlador

Finalmente se dan los permisos necesarios al puerto para evitar de esta manera posteriores inconvenientes durante la configuración del compilador (Arduino IDE) por medio del siguiente comando "sudo chmod a+rw /dev/ttyACM0".

Una vez introducido dicho comando en la terminal, presionamos "*intro"* y posteriormente se pedirá autenticarse por medio de la contraseña del sistema, la colocamos y listo ahora solo se tiene que seleccionar el puerto serie correcto en el software utilizado por Arduino.

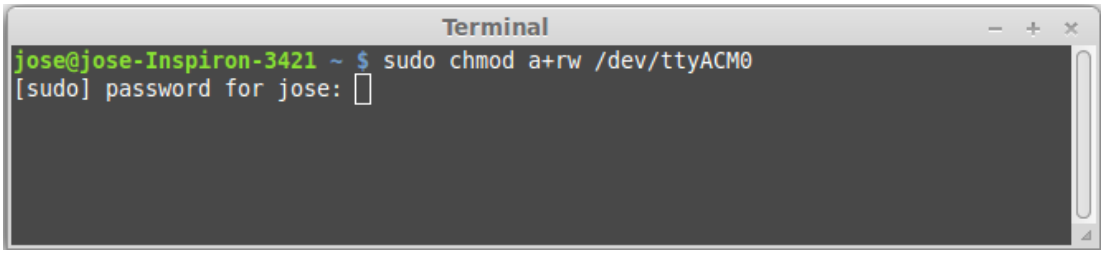


Figura 46. Contraseña del sistema (ordenador)

A continuación se ejecuta el IDE o compilador de Arduino y se procede a configurarlo, para lo cual es necesario dirigirse a la opción *Herramientas, tarjeta* y se elegirá el tipo de placa utilizada; que como sabemos para esta investigación corresponde a la tarjeta *Arduino Mega 2560.*

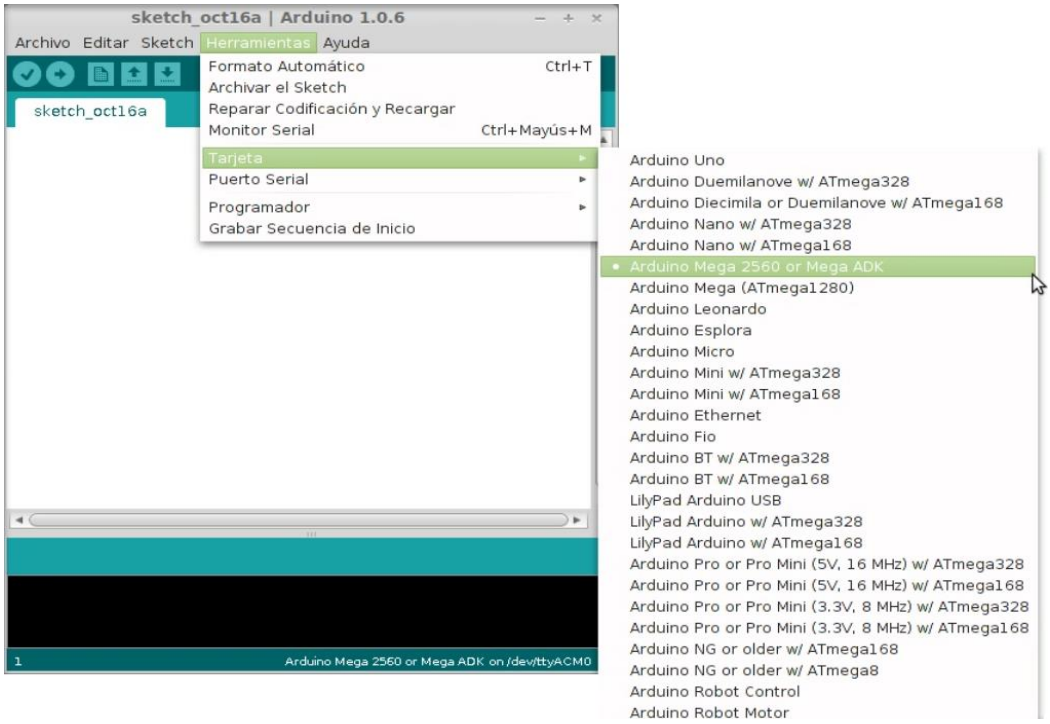


Figura 47. Tipo de tarjeta utilizada

Posteriormente se configura el puerto serie, para lo cual se debe dirigir a la opción *Herramientas, Puerto Serial* y se selecciona la opción correcta de acuerdo al puerto COM utilizado por el controlador, que en este caso resultó ser “ttyACM0”. Por otra parte es importante recalcar que durante todo el proceso de configuración, la tarjeta debe estar conectada al ordenador por medio del cable USB de tipo AB.

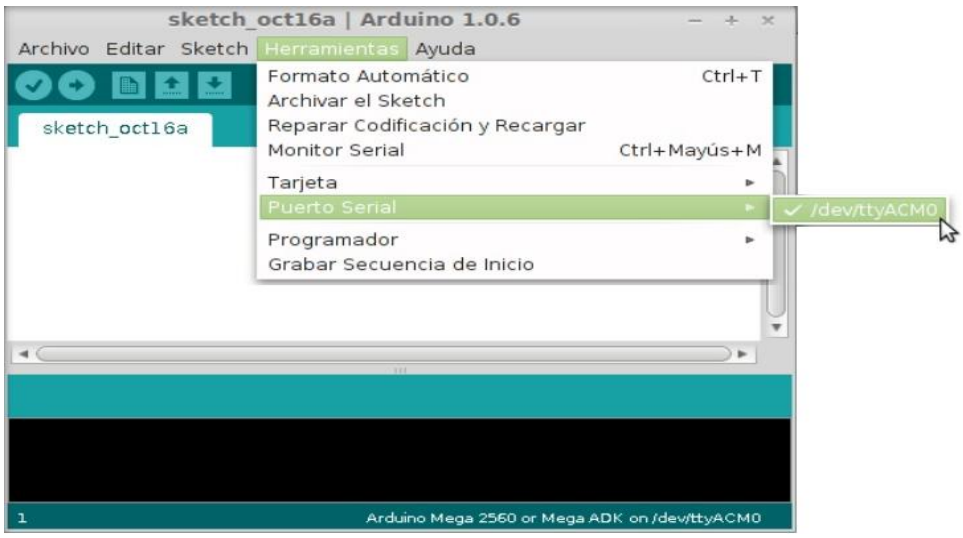


Figura 48. Puerto serie utilizado

3.2.2.3. Comprobando Entradas y Salidas Digitales “E/S”

Una vez instalado y configurado el compilador *IDE de Arduino* en el ordenador, es necesario llevar a cabo las primeras pruebas con la tarjeta o placa (*Arduino Mega 2560 R3*), de manera que sea posible corroborar el correcto funcionamiento de esta; y con el fin de familiarizarnos con el lenguaje de programación (C, C++) aprendido en el apartado anterior.

3.2.2.3.1. Salida digital

Para comprobar las salidas digitales del controlador, basta con utilizar un diodo emisor de luz (LED), una resistencia de 1kΩ, conectores mm y una protoboard para realizar el montaje. El objetivo de esta prueba consiste en que el LED se prenda de manera intermitente (HIGH – LOW) cada segundo utilizando constantes de tipo booleanas.

Para este fin se procede a ejecutar el software de Arduino (IDE), donde se compilará y depurará el código escrito de manera que sea posible cargarlo a la memoria interna del controlador.

- **Código de programación**

```
int ledPin = 10;                //Asigna a la variable ledPin el valor entero 10

void setup()                    //Parámetros Iniciales
{
pinMode(ledPin, OUTPUT);        //Declara al pin digital 10 como salida
}

void loop()                     //Estamentos o instrucciones
{
digitalWrite(ledPin, HIGH);     //Enciende el LED
delay(1000);                    //Espera un segundo
digitalWrite(ledPin, LOW);      //Apaga el LED
delay(1000);                    //Espera un segundo
}

//Fin del programa
```

Una vez escrito el código en el compilador (IDE), se procede a verificarlo (compilarlo) dando un click en el icono de la parte superior izquierda del software, el cual está representado por una señal de cotejo (✓). Sí el código fuente escrito en términos lógicos es correcto, el compilador dará un informe del tamaño binario del sketch a cargar; caso contrario se retornará un error de compilación.

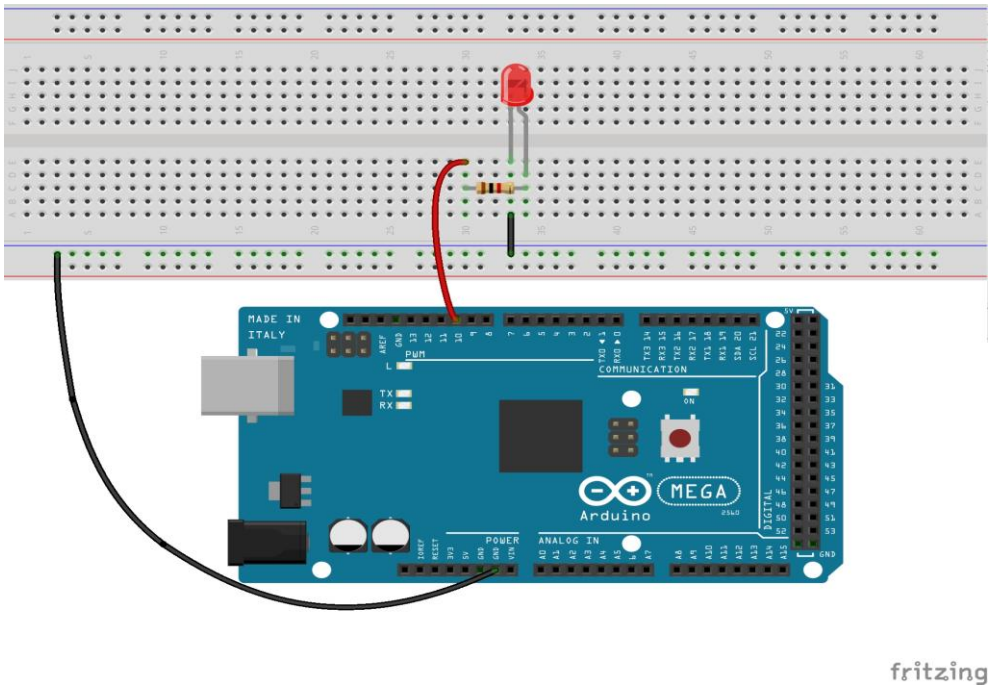


Figura 49. Vista protoboard salida digital

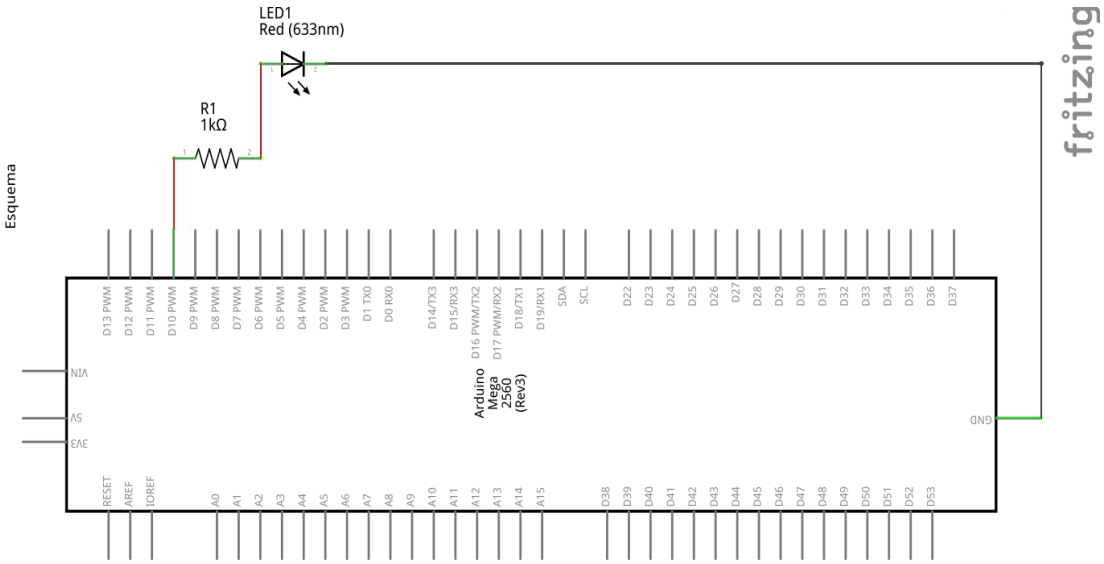


Figura 50. Diagrama eléctrico salida digital

Como se puede analizar el montaje del diseño irá acorde al código de programación escrito, por lo que el pin 10 corresponde a la salida digital declarada anteriormente (OUTPUT), la misma que está conectada mediante un terminal mm (macho-macho) al positivo del LED utilizado (cable rojo). Por otra parte se debe tener en cuenta que el voltaje suministrado por la tarjeta a los pines digitales declarados como salidas oscila alrededor de los 5 voltios, por lo que es necesario utilizar una resistencia (1KΩ) capaz de proteger el LED. Cabe indicar que el controlador Arduino posee pines GND (tierra), los cuales tienen como función cerrar el circuito; de manera que se procede a conectar el negativo del actuador (LED) al negativo del controlador (cable negro).

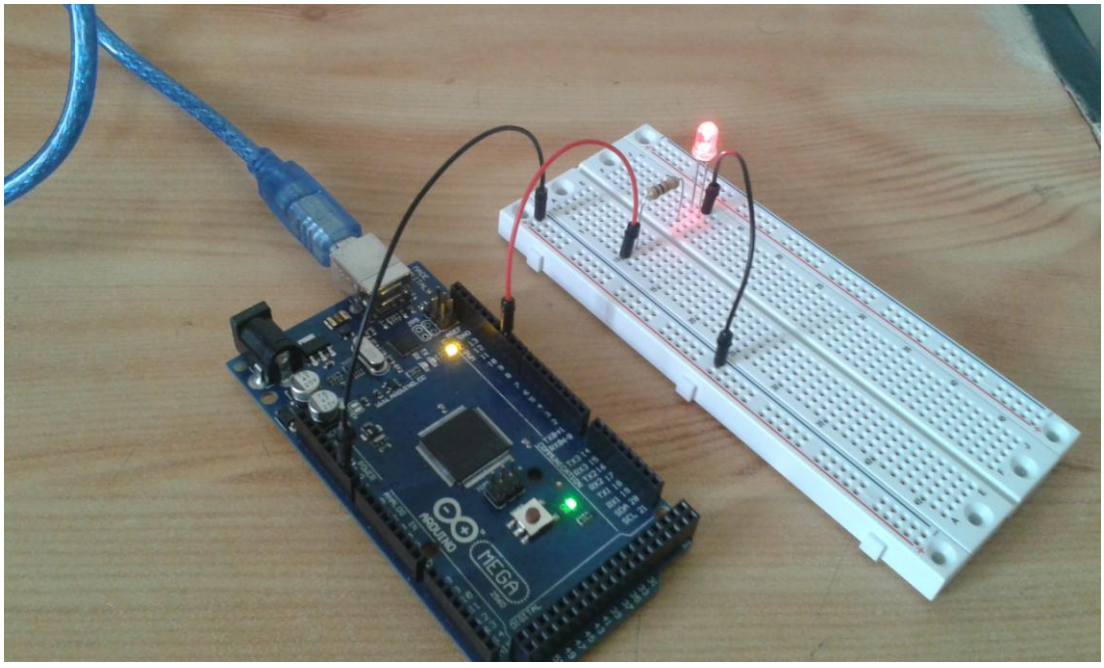


Figura 51. Montaje del circuito (salida digital)

Una vez realizado el montaje físico basado en el diseño esquemático y su programación, se procede a conectar el puerto serie mediante el cable USB de tipo AB con la finalidad de cargar el código fuente al controlador (memoria); para lo cual se debe recordar que tanto la tarjeta como el puerto (COM) a utilizar deben estar correctamente seleccionados en el compilador IDE de Arduino.

Es necesario indicar que para proceder a cargar el código fuente basta con dar un click en la flecha (→) de la parte superior izquierda del compilador. Una vez cargado el código a la memoria de Arduino, se podrá observar el parpadeo intermitente del LED; comprobando de esta manera el correcto funcionamiento del controlador y sus salidas digitales.

3.2.2.3.2. Entrada digital

Ahora se verificarán las entradas digitales del controlador, para lo cual es necesario conseguir un LED, el mismo que tendrá la función de un indicador visual, dos resistencias de 1kΩ, un pulsador NA (normalmente abierto), conectores mm y un protoboard. El objetivo de esta prueba consiste en enviar una señal al controlador cuando se presione el pulsador, con el fin de que se encienda el LED cuando el valor leído en el pin digital declarado como entrada sea alto (HIGH).

- **Código de programación**

```
int ledPin = 10;        //Asigna a la variable ledPin el valor entero 10
int pulsador = 5;       //Asigna a la variable pulsador el valor entero 5

void setup()
{
  pinMode(ledPin, OUTPUT);       //Declara al pin digital 10 como salida
  pinMode(pulsador, INPUT);      //Declara al pin digital 5 como entrada
}

void loop()
{
  if(digitalRead(pulsador)==HIGH)   //Sí el valor leído en el pin digital 5 es alto
  {
    digitalWrite(ledPin, HIGH);              //Enciende el LED
  }
  else                                       //Sino
  {
    digitalWrite(ledPin, LOW);               //Apaga el LED
  }
}
//Fin del programa
```

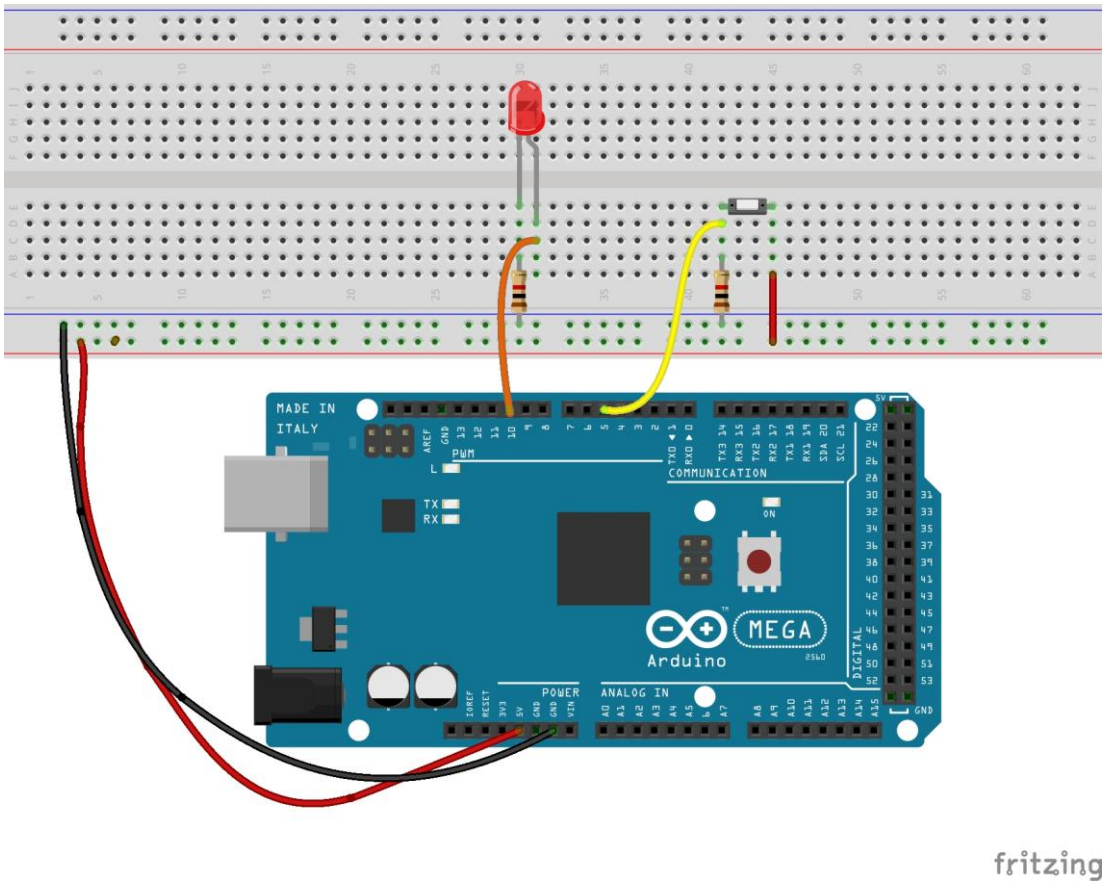


Figura 52. Vista protoboard entrada digital

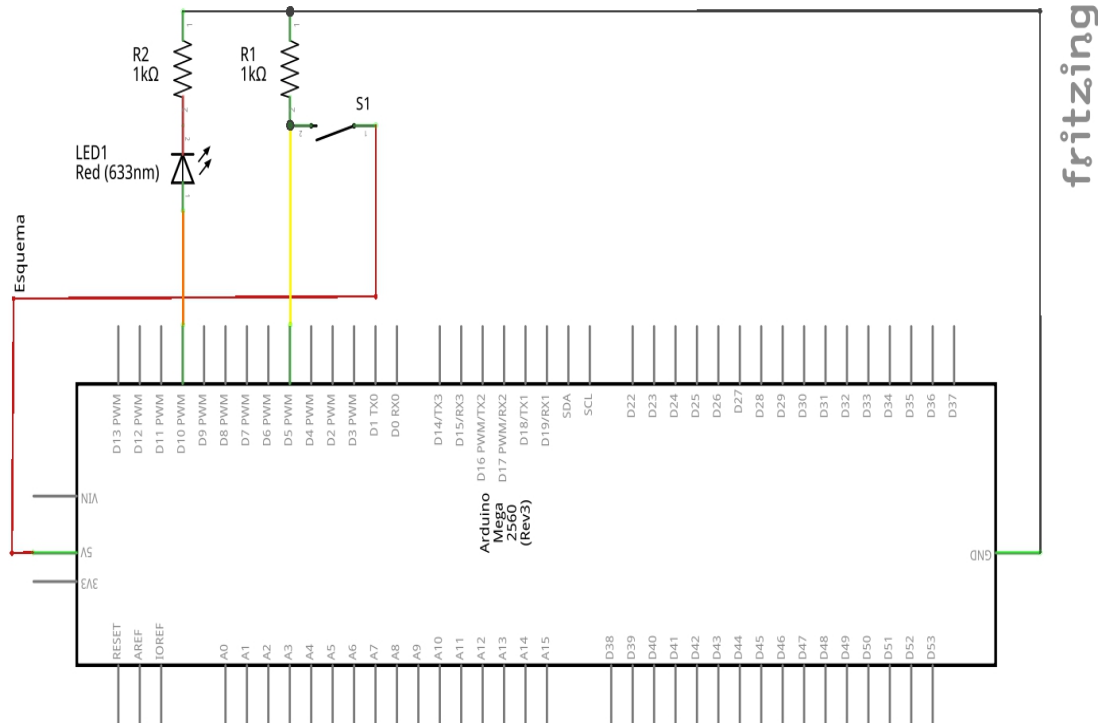


Figura 53. Diagrama eléctrico entrada digital

Como se puede apreciar en la figura 52, el pulsador (normalmente abierto) se encuentra conectado por un extremo al positivo de 5V o VCC suministrado por el controlador (cable rojo); mientras que por el otro extremo del pulsador se encuentra conectado el cable de señal (amarillo) y el negativo por medio de la resistencia de mil ohmios. Cuando el pulsador es accionado, la señal es enviada al pin digital 5 declarado como entrada (INPUT), donde dicha señal es leída por el controlador como un valor alto (HIGH); cumpliendo de esta manera la condición y por ende ejecutándose las instrucciones entre llaves.

Cabe indicar que si la condición resulta ser verdadera (pulsador presionado), el diodo emisor de luz conectado al pin digital 10 declarado como una salida es encendido hasta cuando el valor leído en la entrada digital 5 sea bajo (LOW); es decir que cuando el pulsador deje de ser accionado, la condición resultará ser falsa y por ende el LED es apagado ya que que las instrucciones del condicional “if” no son ejecutadas al tener un valor de verdad *falso*.

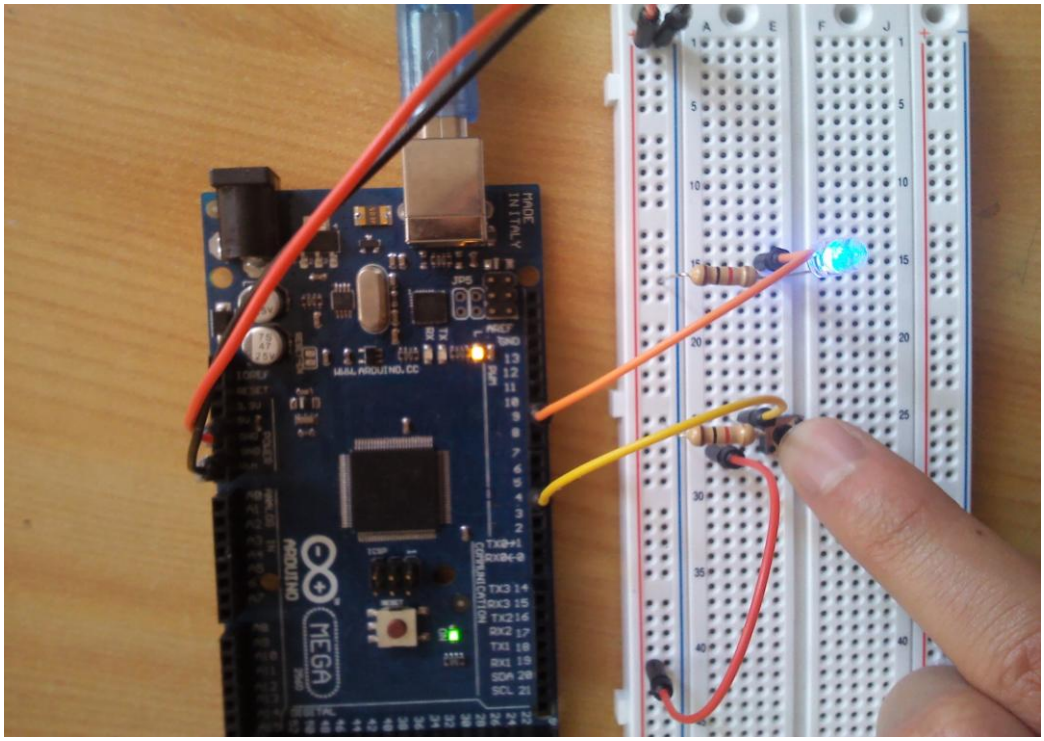


Figura 54. Montaje del circuito (entrada digital)

3.2.2.3.3. Entrada Análoga

Para verificar las entradas analógicas del controlador, se puede utilizar algún tipo de sensor análogo como por ejemplo aquellos que miden la temperatura, humedad, luz, presión, etc., o simplemente se utilizará un potenciómetro de 10 kΩ, un diodo emisor de luz, conectores mm y un protoboard. El objetivo de esta prueba consiste en que a medida que varíe la señal enviada por el sensor (potenciómetro), el indicador visual (LED) informará el estado de este último al encenderse y apagarse de manera intermitente con mayor o menor frecuencia.

- **Código de programación**

```
int potenciometro = A3;      //Define al pin análogo A3 como potenciometro
int ledPin = 13;             //Define al pin 13 como ledPin
int valorPot = 0;            //Asigna a la variable valorPot en valor de inicio

void setup()
 {
pinMode(ledPin, OUTPUT);    //Declara a la variable ledPin como salida

/*Nota NO es necesario declarar el pin análogo (A3) como una entrada ya que por defecto de fábrica todos los pines análogos son definidos como entradas IN*/
}

void loop()
{
valorPot = analogRead(potenciometro);
//Leer el valor del potenciómetro y almacenar dicho valor en la variable valorPot

digitalWrite(ledPin, HIGH);
//Enciende el LED
delay(valorPot);
//Detenemos el programa durante <valorPot> en milisegundos

digitalWrite(ledPin, LOW);
//Apaga el LED
delay(valorPot);
// Detenemos el programa durante <valorPot> en milisegundos
}
//Fin del programa
```

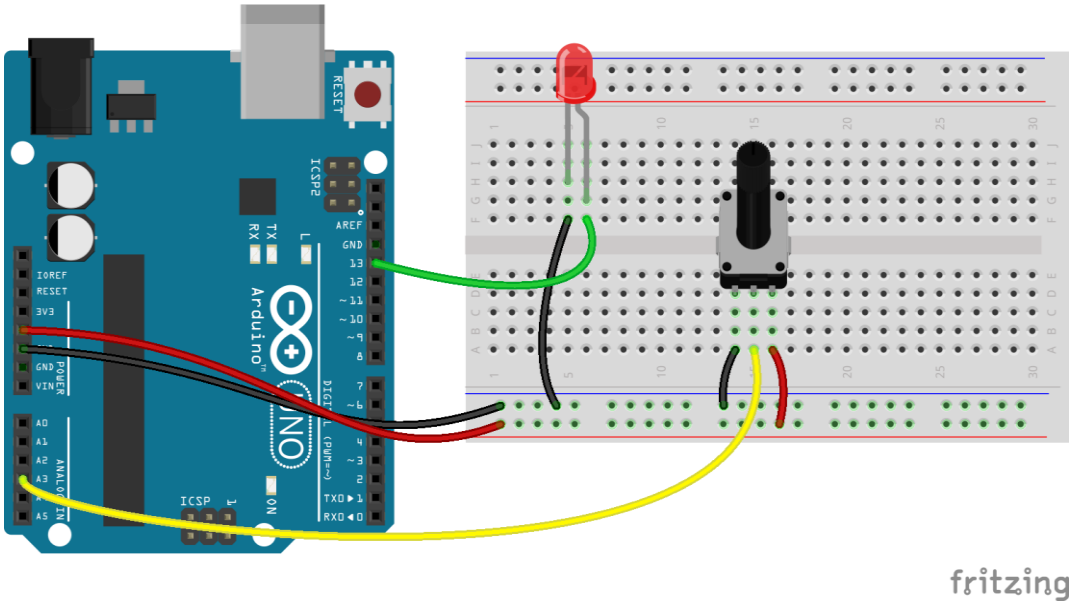


Figura 55. Vista protoboard entrada análoga

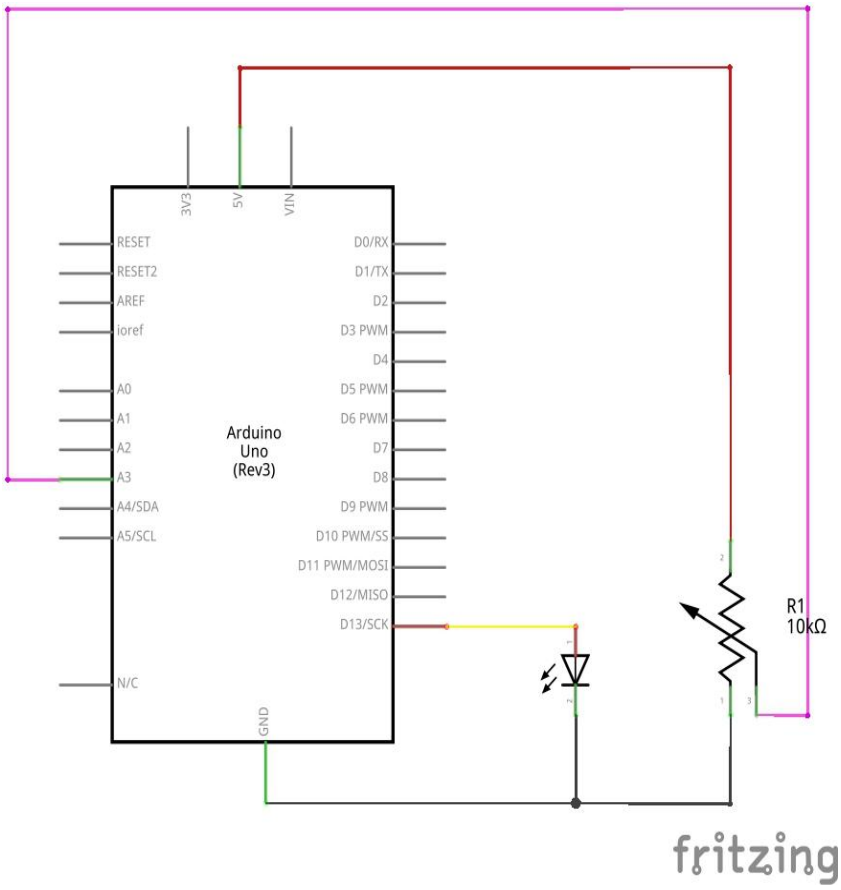


Figura 56. Diagrama eléctrico entrada análoga

Como se puede observar la entrada análoga (potenciómetro) posee tres pines conectados al controlador Arduino, donde el cable rojo corresponde a la alimentación de 5 voltios, el cable negro a GND o tierra y el cable amarillo como una señal, la misma que se encuentra conectada al pin “A3” para enviar el valor o tensión correspondiente (0 y 5 voltios) y convertirlo de esta manera en un dato de tipo entero comprendido entre 0 y 1023. Por otra parte es posible visualizar un LED conectado al pin digital 13, el mismo que ha sido declarado como una salida en el código anteriormente escrito; y que tiene como función indicar el nivel o valor enviado por el potenciómetro mediante una señal luminosa, la misma que se encenderá y apagará proporcionalmente a la lectura obtenida (analogRead) de la señal variable. Cabe indicar que dicha señal o corriente es controlada por la resistencia interna que posee nuestra entrada (potenciómetro) la cual varía según la acción del usuario.

NOTA: El controlador Arduino posee una resistencia interna en el pin digital 13, por lo que no es necesario conectar otra resistencia adicional al LED.

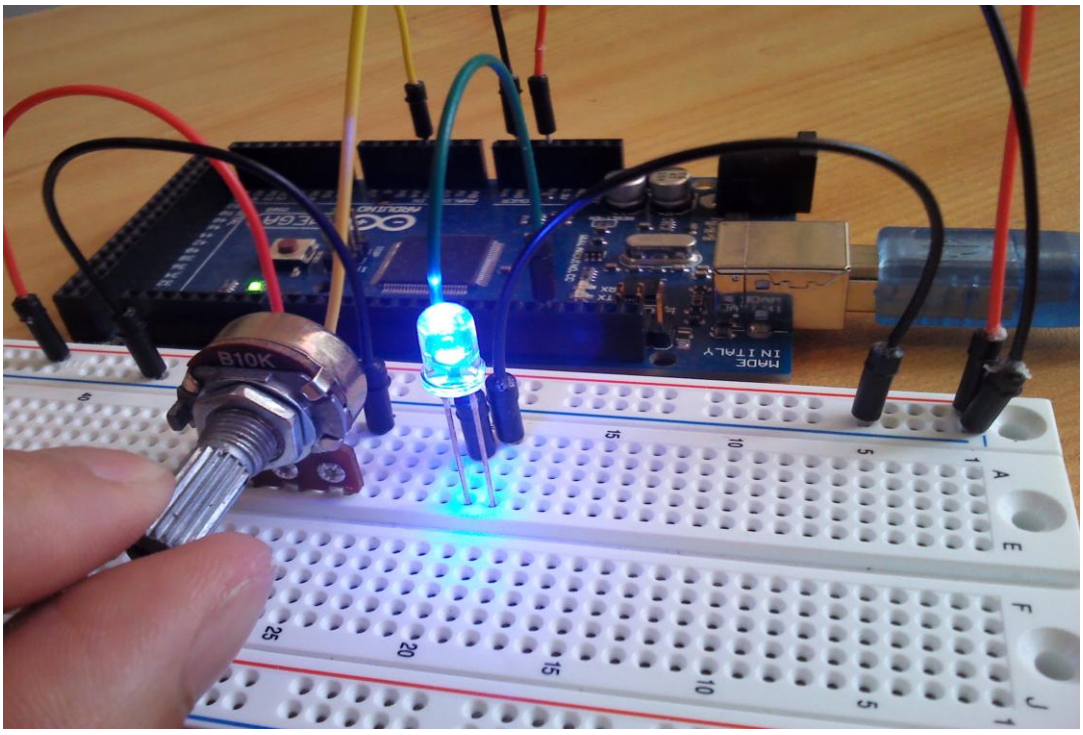


Figura 57. Montaje del circuito (entrada análoga)

3.2.2.3.4. Salida Análoga

El controlador Arduino es capaz de simular una salida análoga por medio de la modulación por ancho de pulso o PWM, para lo cual se va a necesitar un potenciómetro (tipo b) de 10kΩ, un diodo emisor de luz, conectores mm y un protoboard. El objetivo de esta prueba consiste en simular una salida análoga (PWM) en un pin digital utilizando un LED, el mismo que cambiará su estado (HIGH - LOW) en milisegundos a medida que varíe la señal del sensor (potenciómetro).

- **Código de programación**

```
int ledPin = 13;                //LED conectado al pin digital 13
int potenciometro = A3;         //Potenciómetro conectado al pin análogo A3
int valorPot = 0;               //Variable en el que se almacena el dato leído

void setup()
{
  pinMode(ledPin, OUTPUT);        //Declara al pin 13 como salida

/*Nota NO es necesario declarar el pin análogo (A3) como una entrada ya que por defecto de fábrica todos los pines análogos son definidos como entradas IN*/
}

void loop()
{
valorPot = analogRead(potenciometro);
//Asigna a la variable valorPot el valor leído en el pin A3

analogWrite(ledPin, valorPot / 4);
//Simula una salida análoga (PWM) en el pin 13, de acuerdo al valor leído en el pin análogo (A3) dividido para 4.
}

//Fin del programa
```

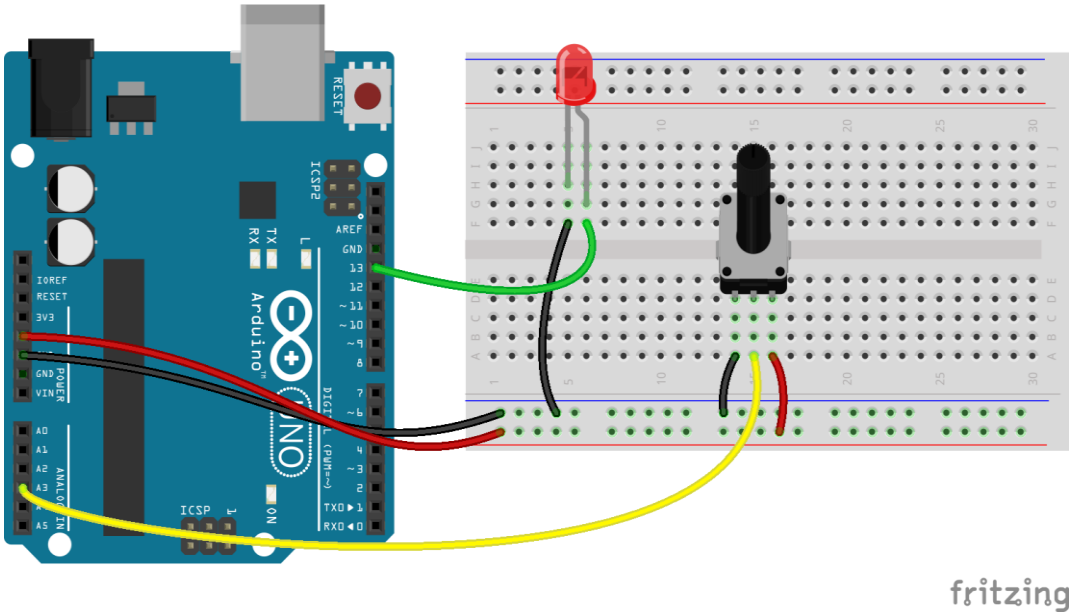


Figura 58. Vista protoboard simulación análoga (PWM)

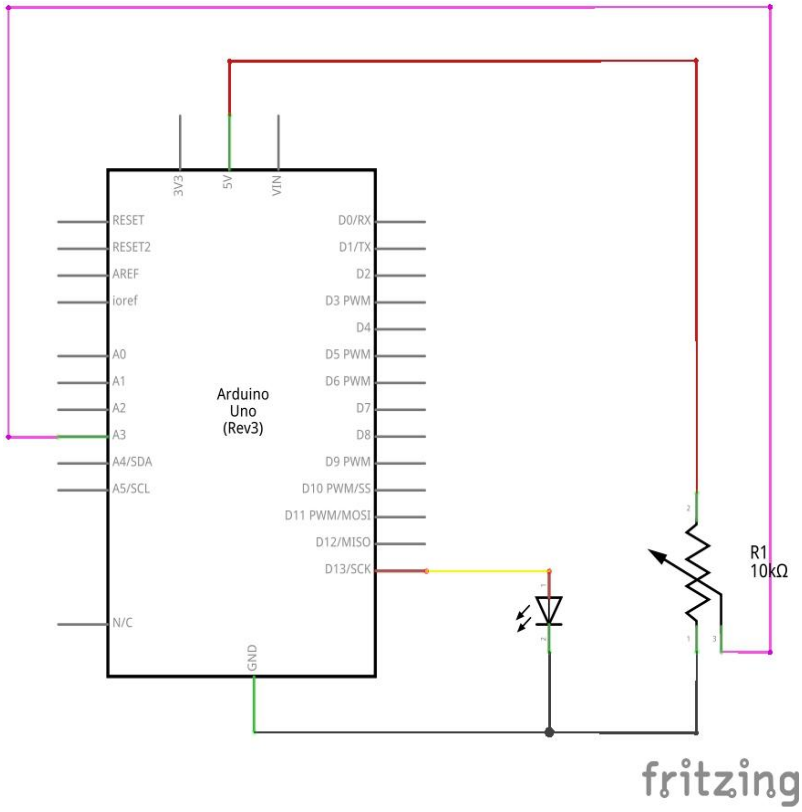


Figura 59. Diagrama eléctrico simulación análoga (PWM)

Como se puede visualizar en esta prueba, el LED utilizado se mantendrá encendido de acuerdo al ancho (PWM) o tiempo modificado (cuando la señal es positiva) con respecto al período, dando como resultado un ciclo de trabajo. Esto dependerá del valor leído en la entrada análoga (A3), donde se encuentra conectada la señal proveniente del potenciómetro.

Cabe indicar que el valor leído en el pin analógico (A3) es almacenado en la variable "*valorPot*", con el fin de simular una señal de tipo análoga (PWM) en la salida del pin digital 13 utilizando el comando "analogWrite". Es importante aclarar que dicho valor almacenado en la variable "*valorPot*" es dividido para 4 ya que la instrucción *"analogRead"* (potenciómetro) tiene un intervalo entre 0 y 1023; mientras que el comando *"analogWrite"* (ledPin) posee un valor comprendido entre 0 y 255.

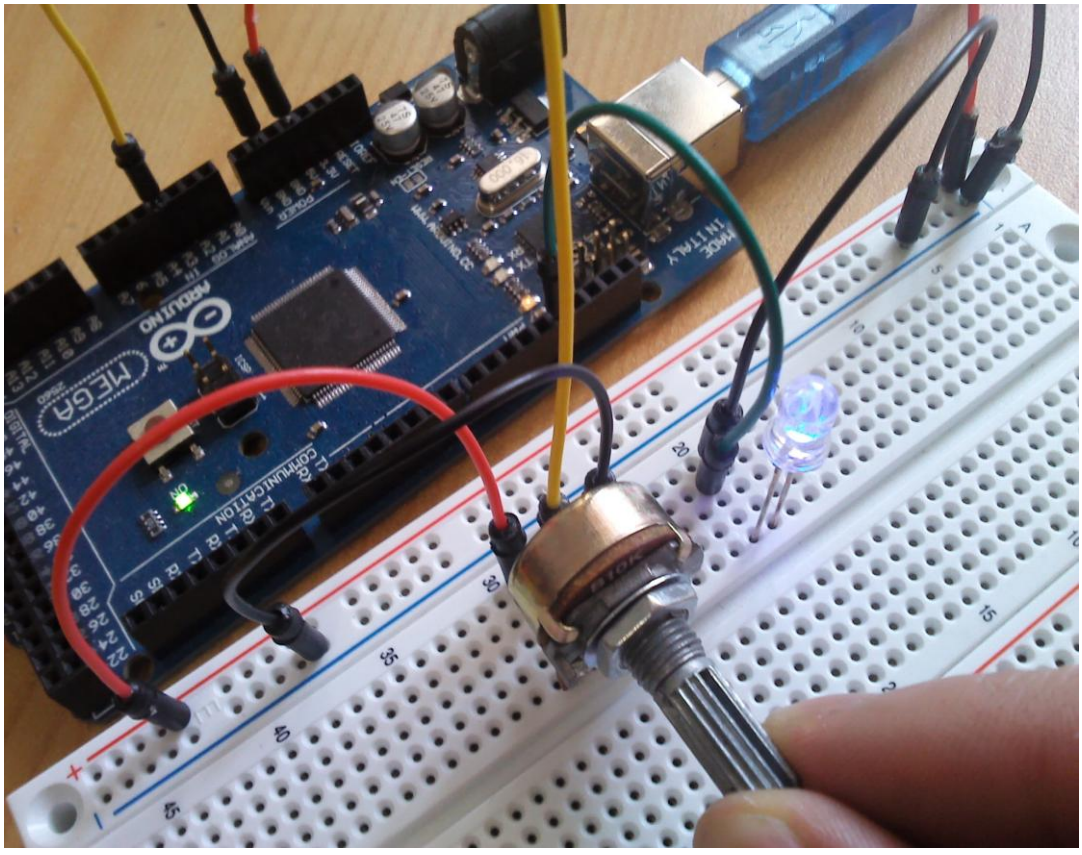


Figura 60. Montaje del circuito (salida análoga)

3.2.3. DISEÑO DEL SISTEMA

3.2.3.1. EasyVR Shield 2.0

El módulo de reconocimiento de voz integrado EasyVR 2.0 es una tarjeta de tipo “Shield” para Arduino, la cual permite expandir las funcionalidades de este (Arduino) al tener la capacidad de procesar los comandos provenientes de nuestra voz y convertirlos en pulsos eléctricos (señales) útiles dentro del mundo de la automatización.

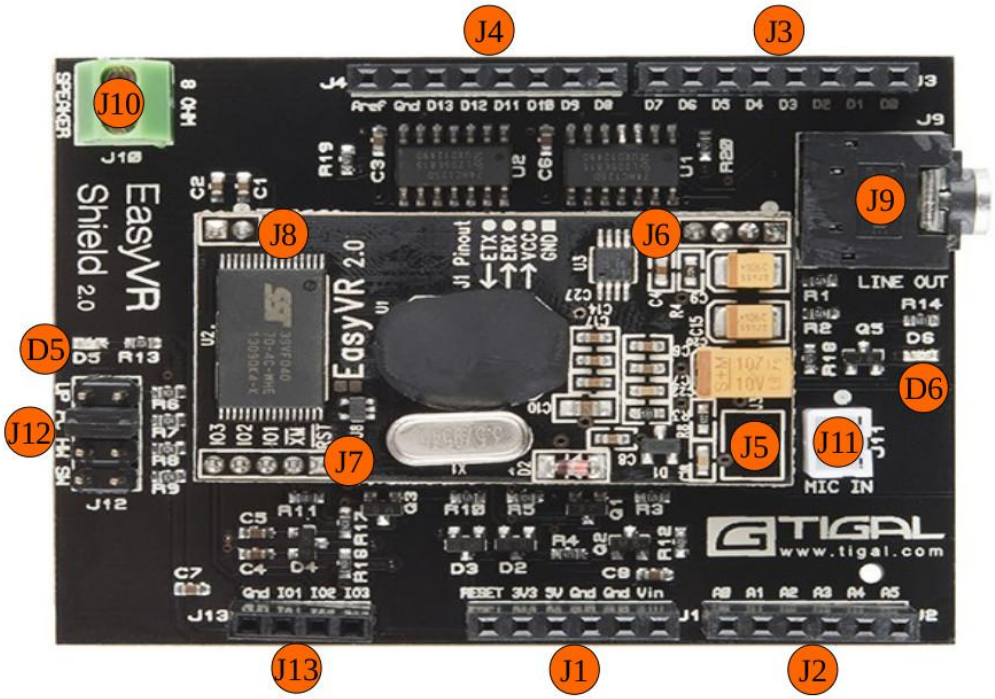


Figura 61. EasyVR Shield 2.0

J1.- Pines de voltaje y tierra

J2.- Entradas análogas

J3, J4.- Pines de E/S digitales, PWM y comunicación serie (ETX y ERX)

J5.- MIC

J6.- VCC (alimentación), GND (tierra), Transmisión de datos (ETX y ERX)

J7.- Reset (RST), Modo de arranque XM, Línea E/S de propósito general IO.

J8.- PWM

J9.- Toma de auriculares 3.5mm stereo/mono

J10.- Salida de audio (Altavoz de 8 Ω)

J11.- Entrada de señal del micrófono

J12.- Jumper (Selecciona el modo de operación de la EasyVR)

J13.- GND, Línea E/S de propósito general 3 voltios (IO1, IO2, IO3).

D5.- LED rojo (Modo actualización).

D6.- LED verde (EsyVR encendida).

3.2.3.2. Montaje de la tarjeta EasyVR Shield 2.0

Para proceder con el montaje entre el controlador máster Arduino Mega 2560 y el dispositivo esclavo EasyVR Shield 2.0, se debe tener muy en cuenta que los pines “A5” y “D0 RX0” estén correctamente alineados uno del otro así como lo muestra la figura 62, de manera que todos los pines deberán coincidir sin ningún problema; ya que si se llega a conectarlos de manera errónea se quemará la tarjeta.

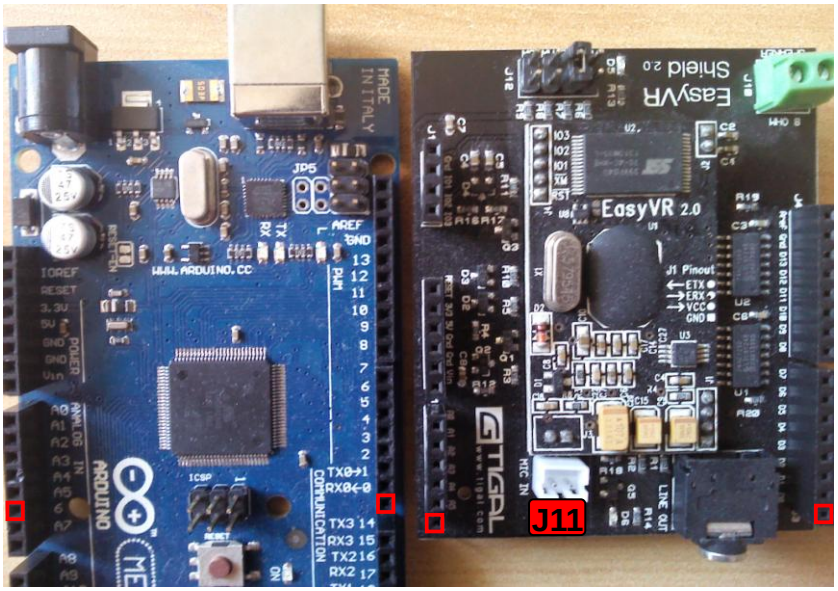


Figura 62. Pines a tener en cuenta durante el montaje

Posteriormente se procede a realizar el montaje de la tarjeta “Shield” sobre el controlador Arduino, donde con mucho cuidado se introducirán los pines correspondientes sin doblar o romper alguno de ellos. Por otra parte también se conectará el micrófono a la entrada J11 (*véase figuras 61-62*).

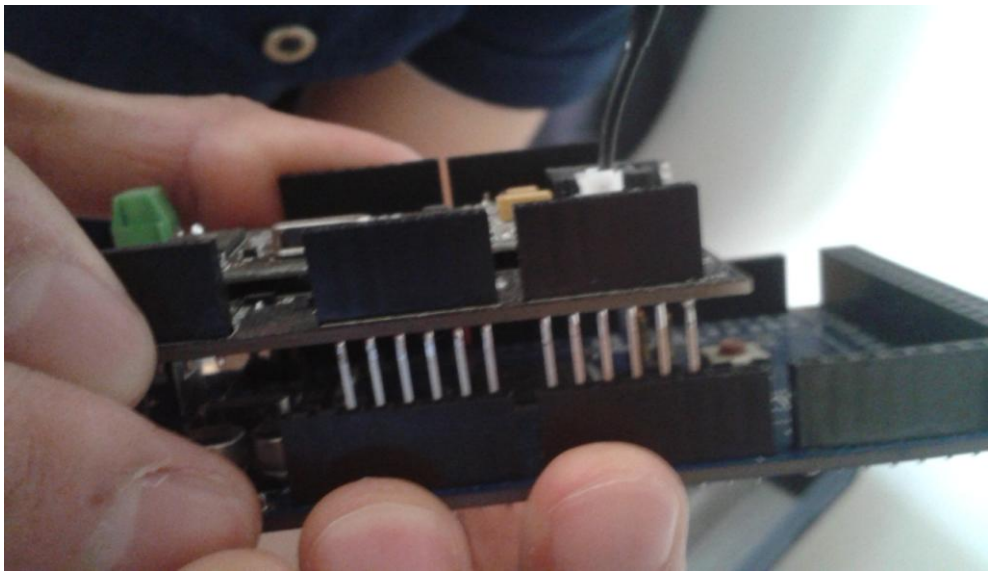


Figura 63. Montaje de la tarjeta EasyVR Shield 2.0

Cabe indicar que para manipular dichos dispositivos, se debe utilizar un brazalete antiestático con el fin de evitar algún tipo de descarga eléctrica que pueda dañar los componentes internos de las tarjetas.

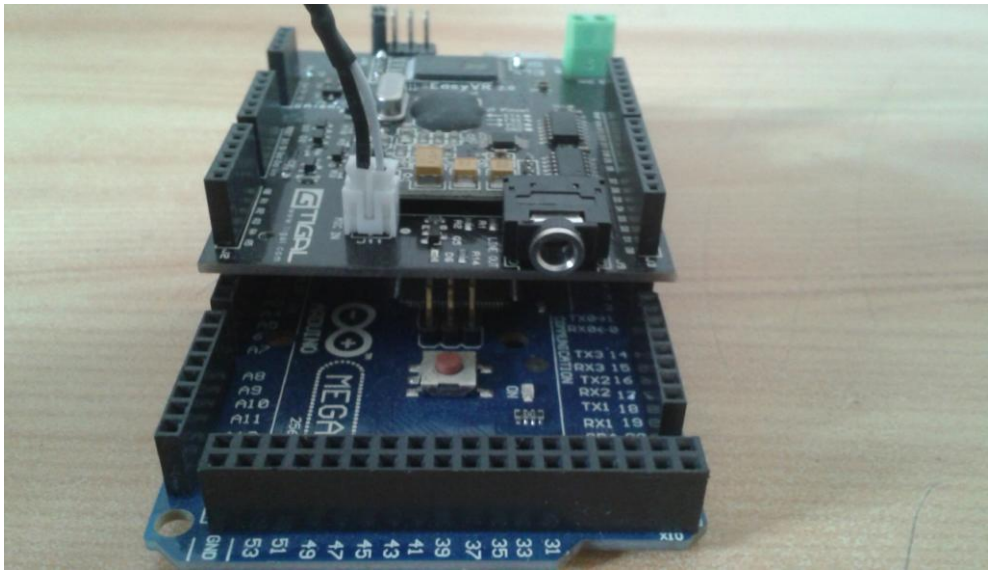


Figura 64. Dispositivo acoplado

3.2.3.3. EasyVR Commander Software

La tarjeta de reconocimiento de voz EasyVR Shield 2.0 utiliza una interfaz gráfica denominada “EasyVR Commander”, la misma que puede ser descargada desde la página oficial de “veear” (http://www.veear.eu/downloads/); para instalar dicho software basta con abrir el archivo ejecutable (EasyVRCommander.exe) y aceptar todos los términos y condiciones de este último. Cabe indicar que durante este proceso se instalará el programa “QuickSynthesis”, el mismo que nos servirá para subir la tabla de sonidos y que se verá más adelante.

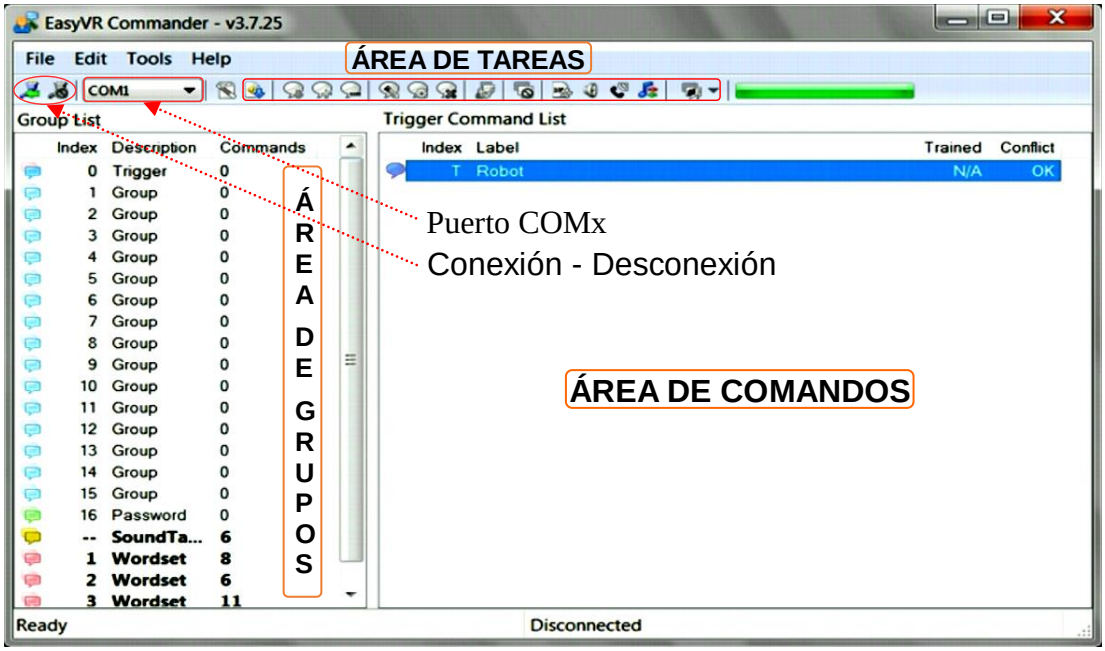


Figura 65. EasyVR Commander Software

Una vez realizado el montaje e instalado el software utilizado por la tarjeta de reconocimiento de voz EasyVR Shield, se procede a descargar las librerías necesarias para Arduino, las cuales se encuentran en la misma página de “veear” (http://www.veear.eu/downloads/) en la sección descargas así como lo muestra la figura 66.

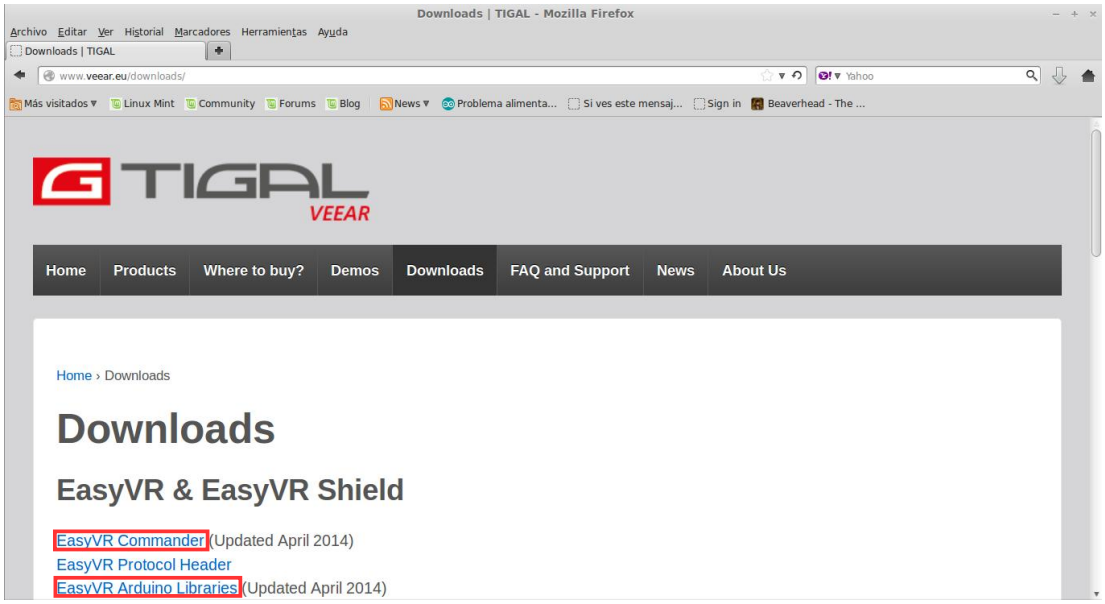


Figura 66. Zona de descarga EasyVR Commander/Arduino Libraries

Posteriormente se abre el archivo descargado (EasyVR Arduino Library) donde se podrá observar una carpeta denominada EasyVR, la misma que se copiará y pegará en el interior de la carpeta “libraries” del software de Arduino (donde se instaló el IDE); que en este caso si recordamos se encuentra en el directorio “Descargas/arduino-1.0.6/libraries”

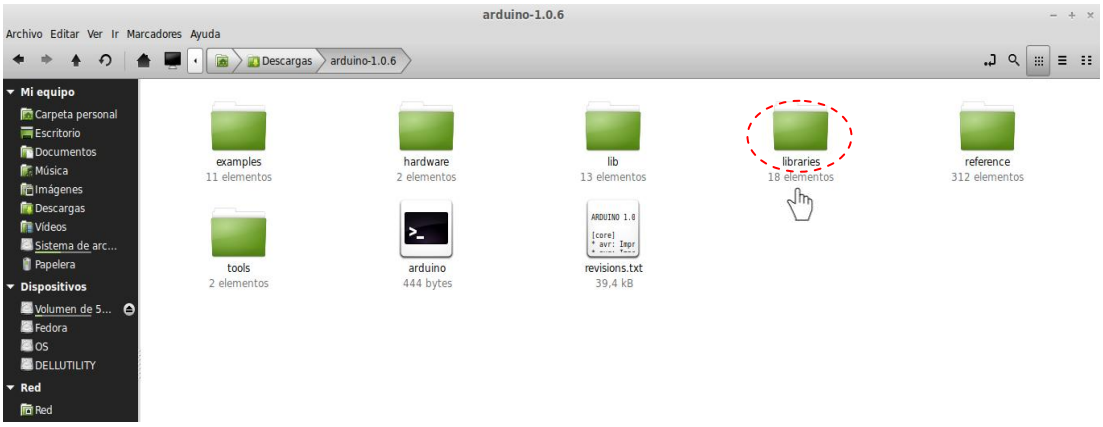


Figura 67. Instalando librerías para Arduino

Una vez copiada y pegada la carpeta “EasyVR” que contiene las librerías necesarias para Arduino, se procede a ejecutar el compilador (IDE) y verificar que se hayan importado correctamente las librerías de la EasyVR; para esto es necesario ir a la pestaña archivo, ejemplos y finalmente se

podrá observar la nueva pestaña añadida (EasyVR).

Figura 68. Librerías necesarias instaladas

Ahora que ya se encuentra listo el compilador de Arduino para proceder a la programación, se procederá a realizar la primera prueba con el fin de que el ordenador pueda reconocer sin ningún problema a la tarjeta EasyVR Shield al ser colocada en modo “Bridge”. Para lo cual se debe ejecutar el compilador de Arduino y a continuación se abrirá una vez más la pestaña archivo, ejemplos, EasyVR, para elegir de esta manera la opción *“TestEasyVR”* (*ver figura anterior*), donde aparecerá una ventana nueva con código escrito en su interior.

Posteriormente se verificarán que las tarjetas estén conectadas correctamente como se lo describió anteriormente en el montaje y a continuación se procede a colocar el jumper de la tarjeta EasyVR Shield en

la posición “***SW***” o modo software.

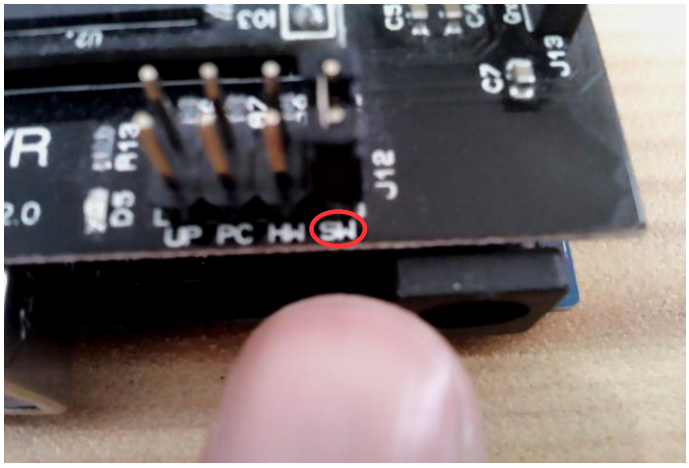


Figura 69. Jumper posición SW

Finalmente se conecta el controlador máster (Arduino) mediante el cable USB de tipo AB al ordenador y a continuación se procede a cargar el código abierto en la ventana del compilador (TestEasyVR).

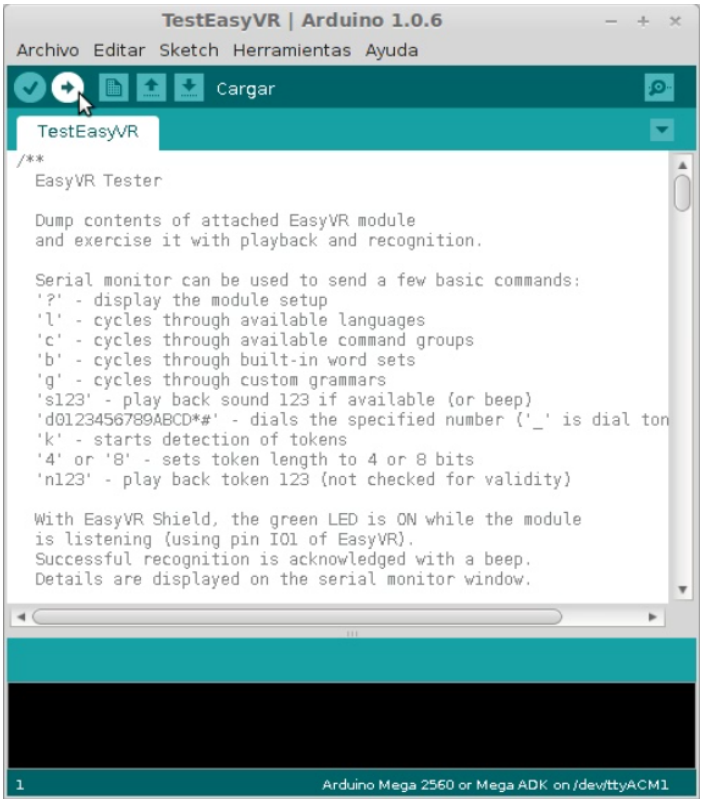


Figura 70. TestEasyVR

Aparentemente no se verá nada fuera de lo común, pero es de vital importancia llevar a cabo este último paso ya que de esta manera el dispositivo esclavo EasyVR se pondrá en modo puente (Bridge) al cargar el código (TestEasyVR) y apagarse el LED de color verde “D6”.

3.2.3.4. Entrenando Comandos

Una vez instalada la interfaz gráfica *“EasyVR Commander”* y posteriormente haber realizado las configuraciones necesarias en el IDE de Arduino (Librerías para Arduino), se procede a conectar el jumper de la tarjeta EasyVR en la posición ***“PC”*** de forma inmediata en el momento en que dicha tarjeta se encuentre en modo “Bridge” (LED “D6” apagado). Cabe indicar que para este fin (cambiar posición del jumper), el controlador siempre debe estar desconectado de la alimentación de energía (USB).

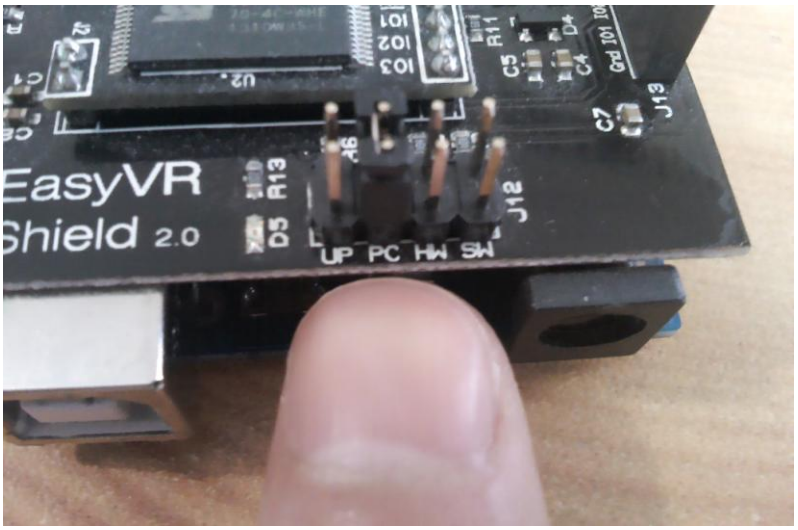


Figura 71. Jumper posición PC

A continuación se conecta el dispositivo al ordenador por medio del cable USB (tipo AB) y se ejecuta la interfaz gráfica *“EasyVR Commnader”*. En la ventana abierta se seleccionará el puerto COM correcto que resulta ser el mismo utilizado por el IDE de Arduino (*véanse páginas 69-70)* y damos click en conectar; en caso de utilizar Windows se puede encontrar el puerto COM utilizado dentro de la opción “Administrador de dispositivos”.

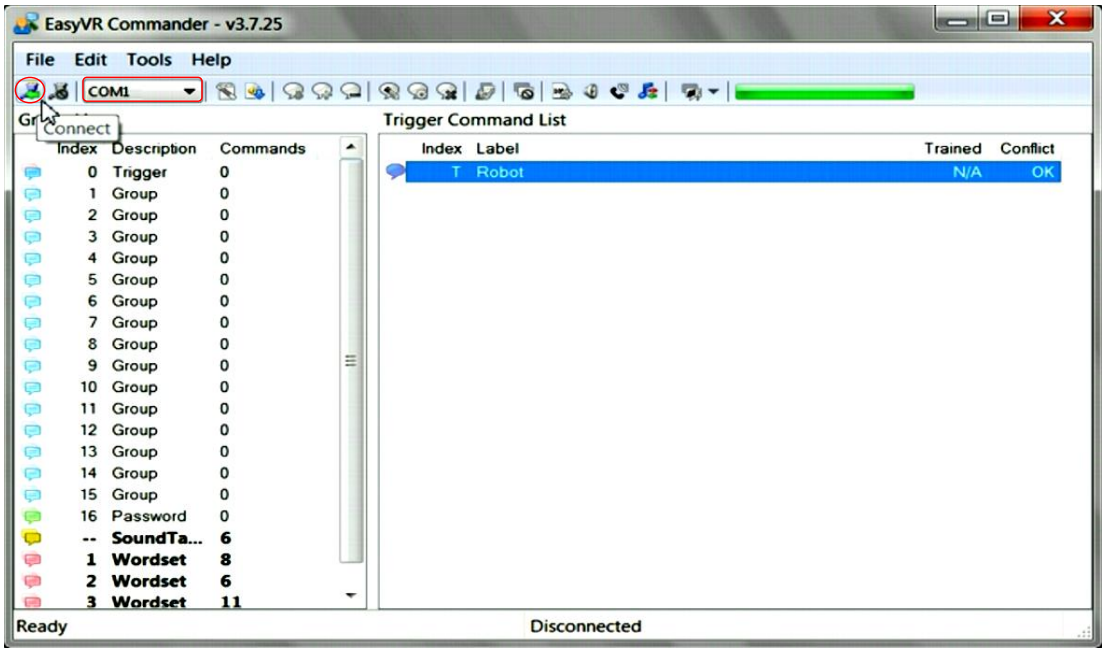


Figura 72. Conexión de la interfaz gráfica EasyVR Commander

Cabe indicar que el puerto COM normalmente utilizado, puede cambiar al conectar otros dispositivos en el puerto serie, por lo que se debe estar pendiente del puerto (COM) empleado por el controlador Arduino; que en este caso resultó ser el puerto COM1.

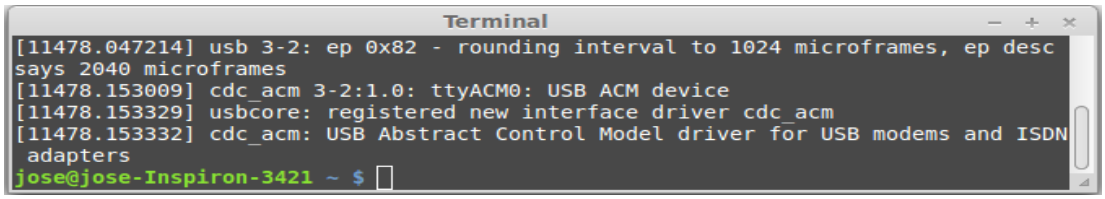


Figura 73. Puerto COM utilizado por el controlador Arduino

Ahora se comenzará a entrenar los comandos deseados, para lo cual nos guiaremos mediante un diagrama de flujo (algoritmo), el mismo que ha sido diseñado en base al problema de investigación. Es decir que mediante este sistema se tendrá la capacidad de controlar los diferentes accesorios electrónicos (luces de salón, ventanas y puertas eléctricas) de un vehículo por medio de la voz. Además es importante indicar que dicho diagrama a emplear será de gran utilidad durante el diseño del código fuente, ya que se trata de la estructura lógica misma del programa.

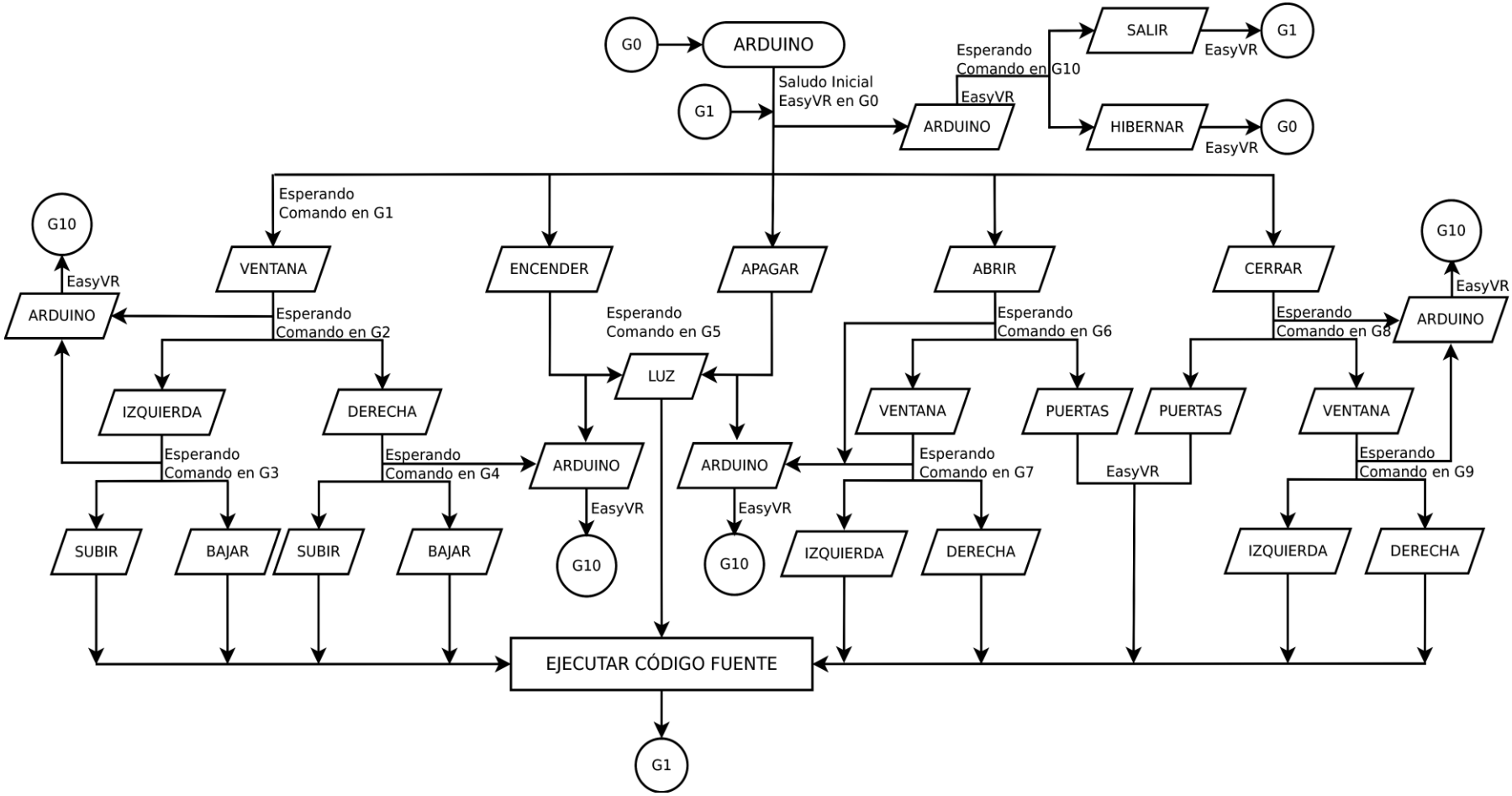


Figura 74. Diagrama de flujo

Una vez conectada la interfaz gráfica (EasyVR Commander) con el dispositivo (EasyVR + Arduino), se procede a verificar que el idioma elegido es el correcto; para lo cual es necesario dirigirse a la pestaña definir idioma (Set Language) y escoger dicho lenguaje deseado (Spanish).

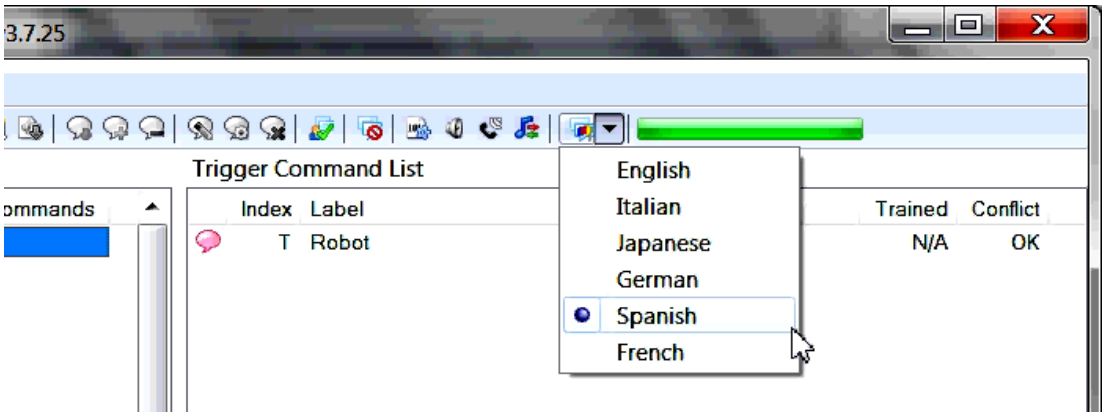


Figura 75. Definiendo un idioma en EasyVR Commander

A continuación se crea el primer comando en el grupo cero (G0) o Trigger (gatillo), para lo cual en el área de grupos se debe seleccionar al disparador conocido también como gatillo (Trigger) y posteriormente se dará click en la opción añadir comando (Add Command) así como lo muestra la figura inferior.

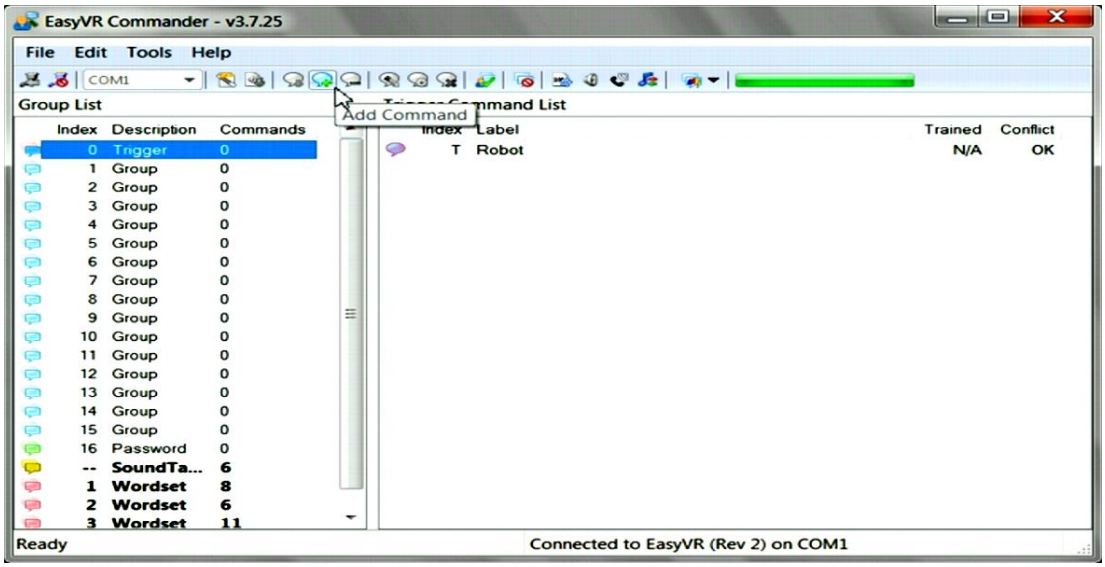


Figura 76. Opción “Añadir Comando”

La palabra o comando a entrenar en este grupo (G0) será el inicio o punto de partida del programa, por lo que es necesario basarse en el diagrama de flujo diseñado (ver figura 74), donde el inicio es la palabra clave “ARDUINO”.

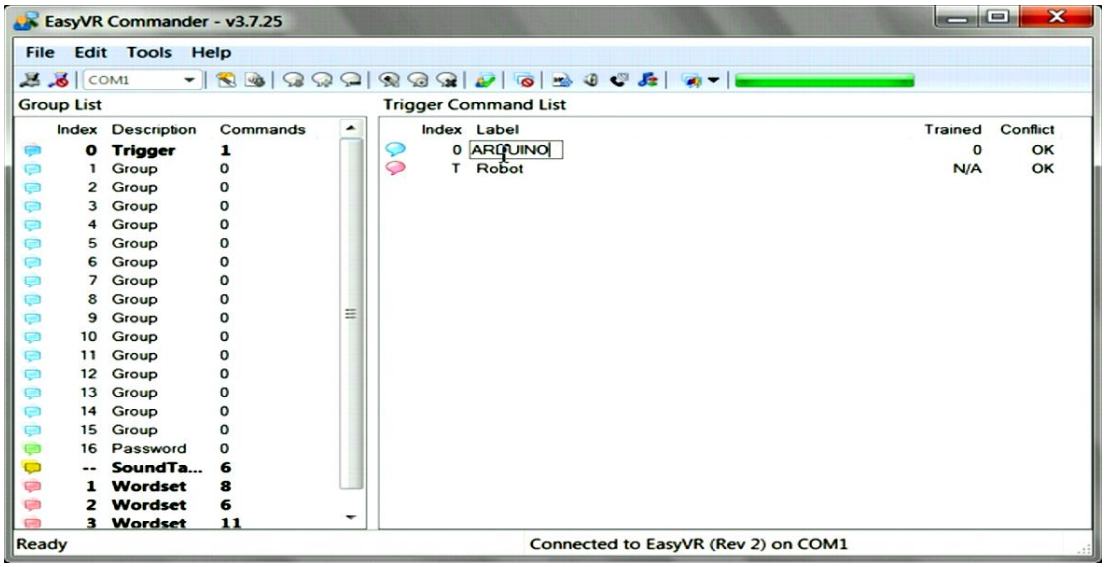


Figura 77. Añadiendo el gatillo del programa en G0

Seguidamente se escribe el comando deseado en el disparador (grupo 0) y a continuación se lo entrena; para este fin se debe seleccionar dicho comando añadido (ARDUINO) y posteriormente se elegirá la opción “entrenar comando” (Train Command) en la barra de tareas.

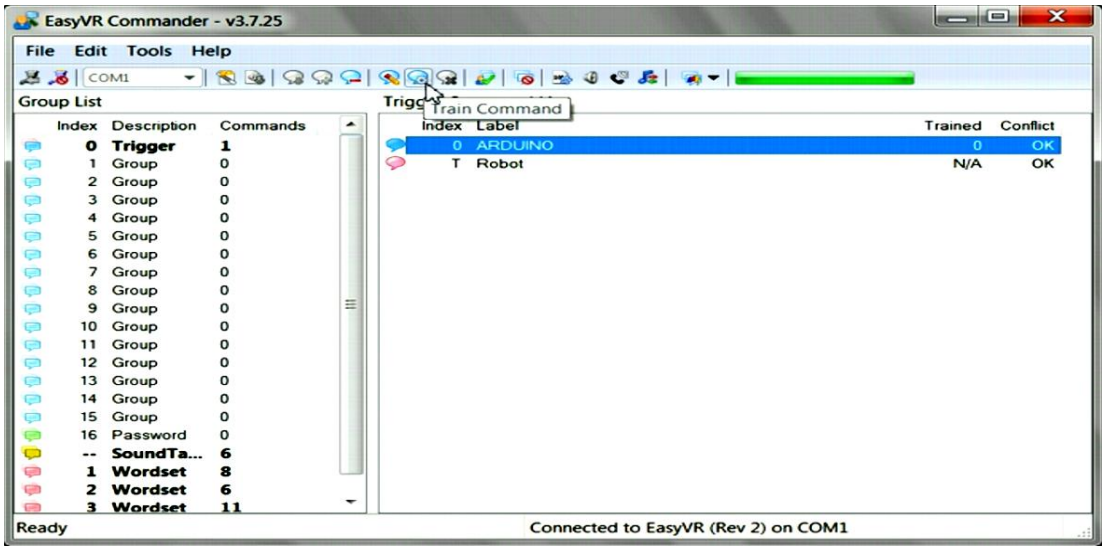


Figura 78. Opción “Entrenar Comando”

Se abrirá una ventana donde se pedirá elegir la opción “Phase 1” y a continuación se deberá pronunciar el vocablo o palabra a relacionarse con el comando en selección. Cabe indicar que durante este proceso es necesario acercarse al menos 10 centímetros al micrófono unidireccional J11.

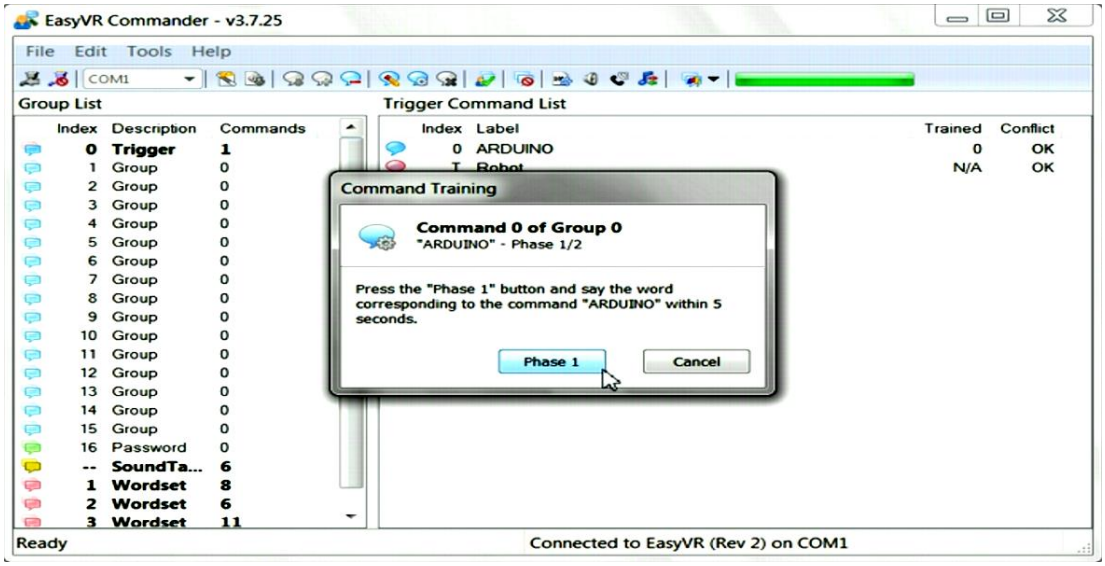


Figura 79. Entrenando el comando “ARDUINO” en G0 (fase uno)

Si el comando de voz fue correctamente pronunciado, se pedirá un segundo intento; por lo que se deberá dar click en la opción “Phase 2” y a continuación se pronunciará una vez más el comando en entrenamiento.

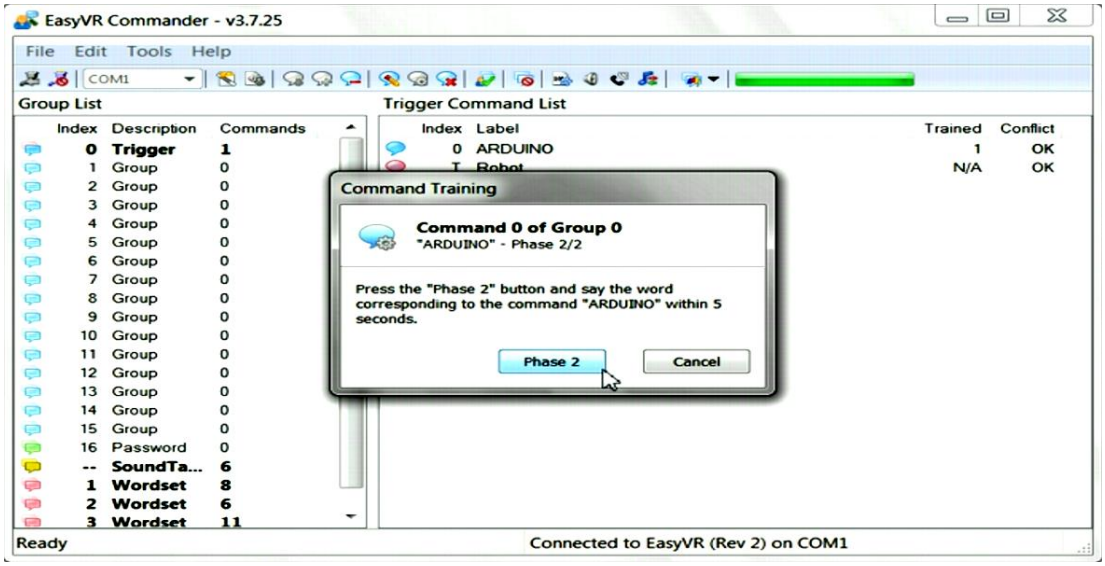


Figura 80. Entrenando el comando “ARDUINO” en G0 (fase dos)

Como se puede observar el comando “ARDUINO” ha sido entrenado dos veces (2) de manera satisfactoria (OK).

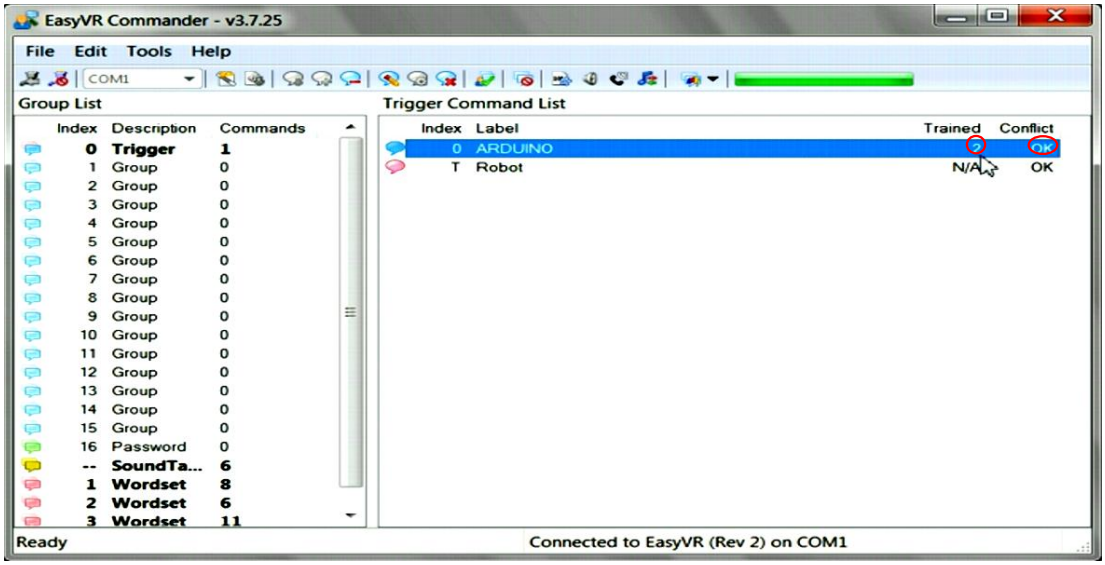


Figura 81. Comando entrenado correctamente

A continuación en el grupo 1 (G1) se procede añadir los comandos pertenecientes a este grupo (*véase diagrama de flujo*).

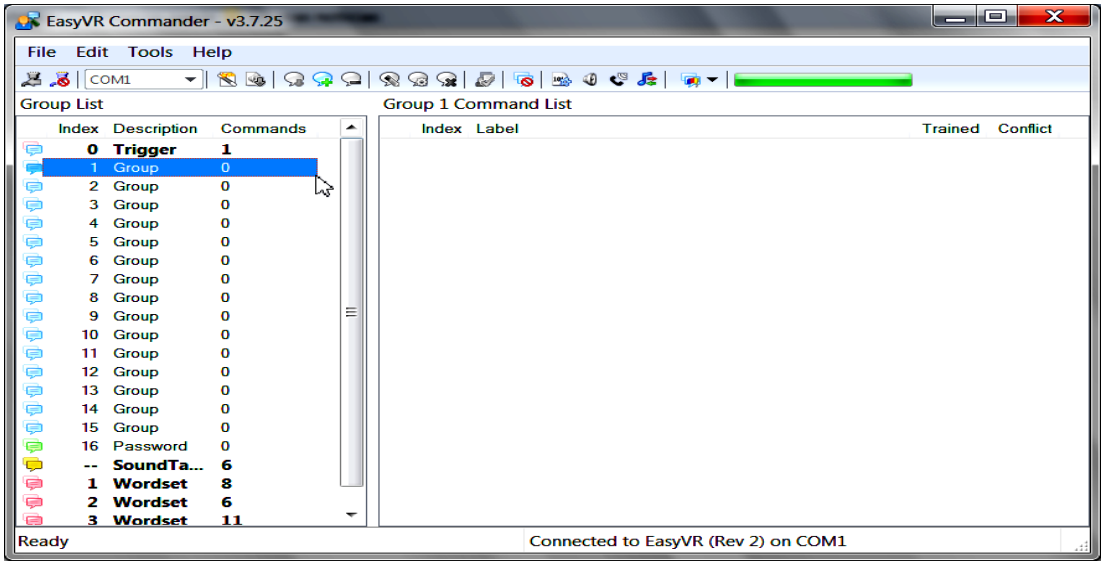


Figura 82. EasyVR Commander grupo 1 (G1)

En la figura 83 se puede observar los comandos pertenecientes a dicho grupo (G1).

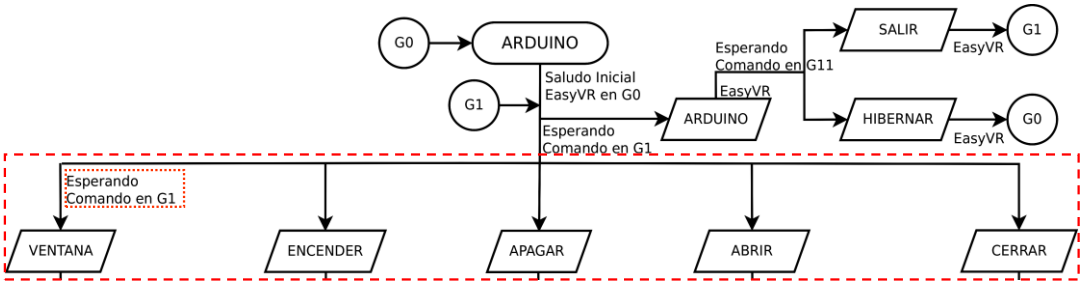


Figura 83. Diagrama de flujo perteneciente a G1

De acuerdo al diagrama de flujo empleado, las entradas pertenecientes al grupo uno (G1) corresponden a los comandos **ventana, encender, apagar, abrir y cerrar**. De manera que se procede añadir y entrenar cada uno de ellos de igual manera como se lo hizo en el grupo 0 (Trigger) con el comando de inicio "*ARDUINO*".

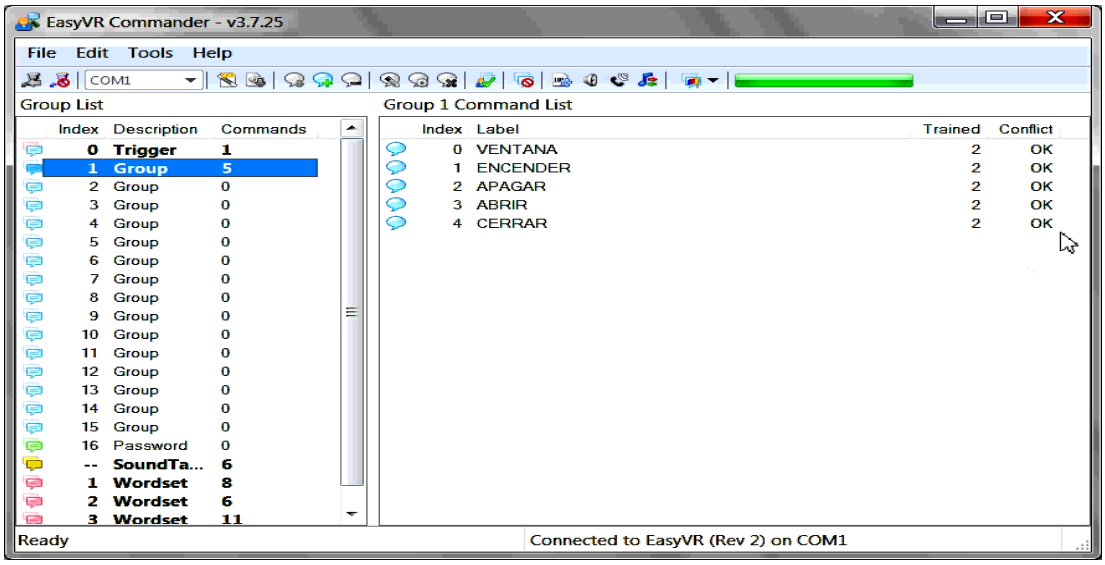


Figura 84. Añadiendo y entrenando comandos en grupo 1 (G1)

Una vez añadido y entrenado los comandos pertenecientes al grupo uno, se continúa con el siguiente grupo que corresponde al *"index 2"* o grupo 2. Cabe indicar que todos los grupos están muy bien enunciados en el diagrama de flujo, de manera que su funcionamiento es fácil de comprender. Por ejemplo en el caso de enunciar el comando "ventana" perteneciente al grupo 1, se tendrá dos opciones (comandos) a elegir en el grupo 2 (G2) izquierda (hacia el grupo 3) o derecha (hacia el grupo 4).

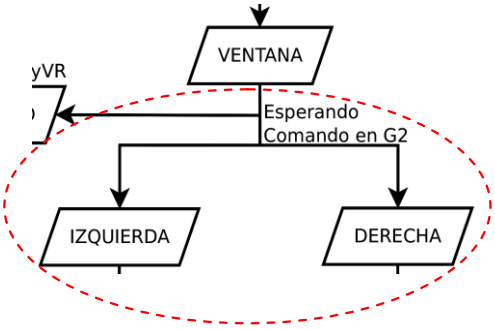


Figura 85. Diagrama de flujo perteneciente a G2

Posteriormente se introducen los comandos correspondientes al grupo 2 y se los entrena.

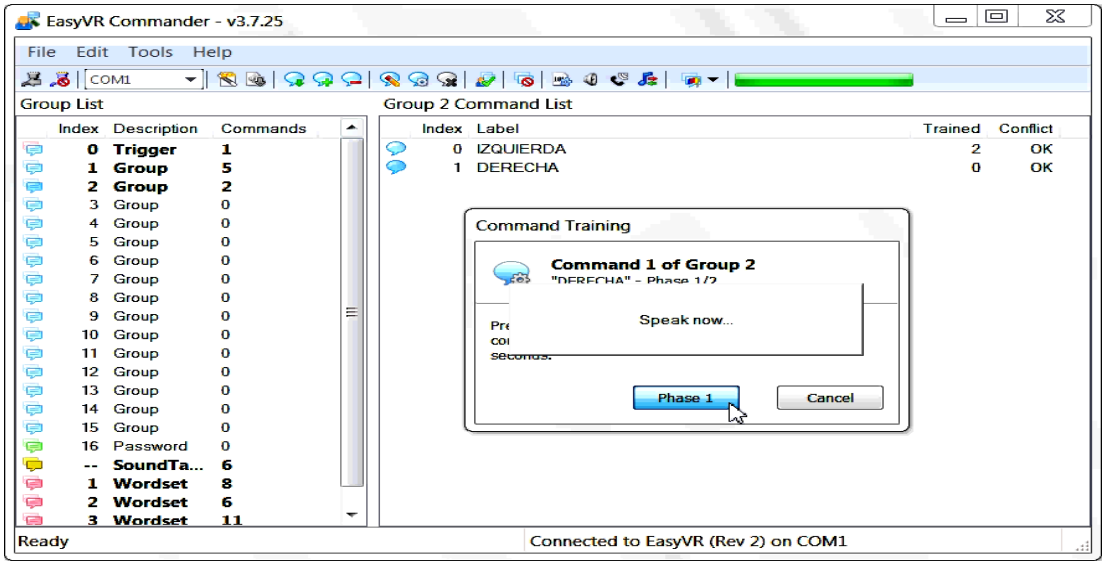


Figura 86. Añadiendo y entrenando comandos en grupo 2 (G2)

En caso de elegir la opción “izquierda” del grupo 2, entonces se tendrá dos opciones más que corresponden a los comandos subir o bajar en el grupo 3.

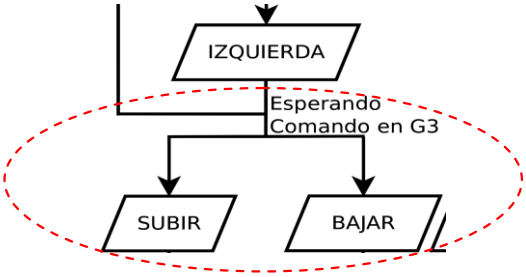


Figura 87. Diagrama de flujo perteneciente a G3

Cualquiera de las opciones a elegir (subir o bajar) ejecutarán el código de programación correspondiente (*véase página 129*), el mismo que lo veremos más adelante en el diseño del código fuente. A continuación se añaden y entrenan los comandos pertenecientes al grupo 3 (G3).

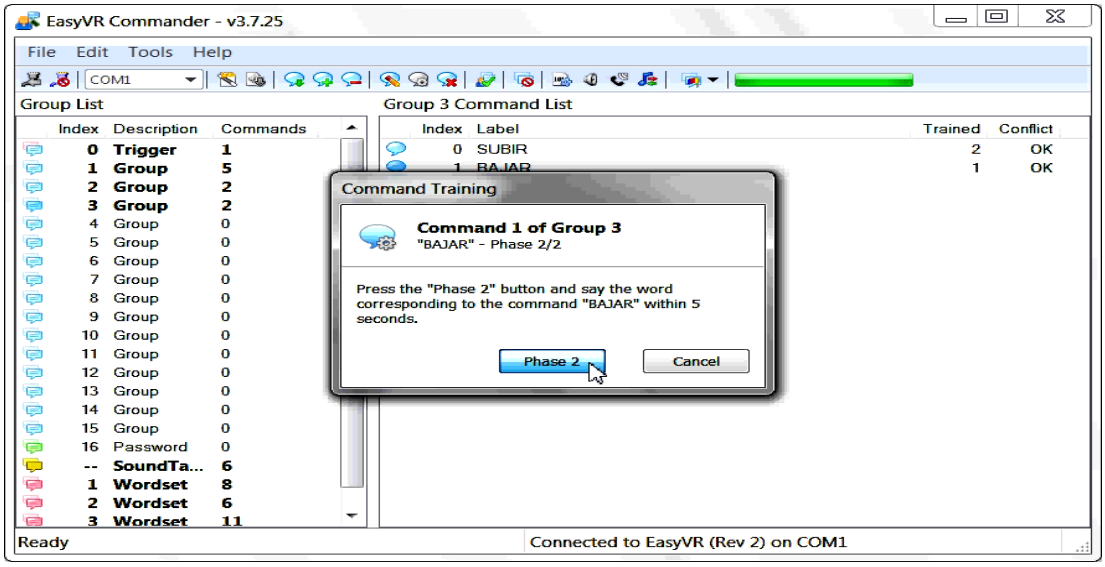


Figura 88. Añadiendo y entrenando comandos en grupo 3 (G3)

Por otra parte si se elige la opción “derecha” del grupo 2, los comandos a elegir en el grupo 4 (G4) son idénticos a los del grupo 3 (subir o bajar); pero con la diferencia de que las constantes booleanas en el código de programación (véase página 129), serán definidas hacia las salidas digitales que controlan los actuadores correspondientes (relés) a las ventanas (derecha e izquierda) del automóvil. Logrando de esta manera subir o bajar una ventana de forma proporcional al tiempo programado (delay) mediante el estado lógico alto o bajo (HIGH – LOW).

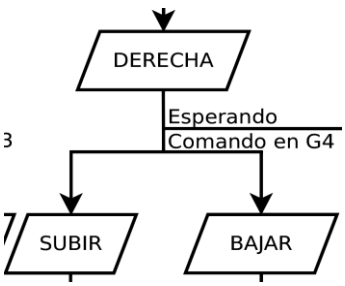


Figura 89. Diagrama de flujo perteneciente a G4

Se añaden y entrenan los comandos pertenecientes al grupo 4 (G4).

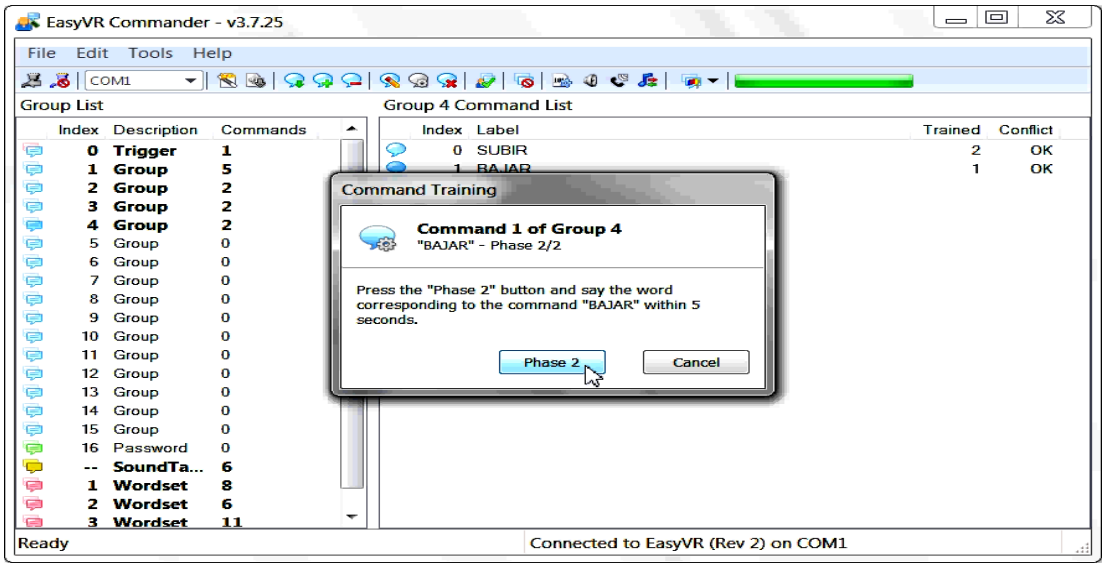


Figura 90. Añadiendo y entrenando comandos en grupo 4 (G4)

En el caso que se necesite encender las luces de salón, simplemente se llamará (pronunciará) al comando “encender” en el grupo 1, y a continuación el comando “luz” perteneciente al grupo 5 (G5), el mismo que ejecutará el código de programación (*véase página 130*) poniéndolo en un estado alto (HIGH) por un tiempo indefinido.

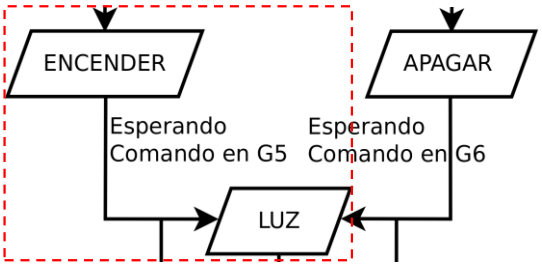


Figura 91. Diagrama de flujo perteneciente a G5

Cabe indicar que tras cada comando entendido por el controlador Arduino, se emitirá un “beep” de sonido hacia los parlantes, con el fin de poderse guiar con mayor facilidad mediante el sentido del oído.

A continuación se añade el comando “LUZ” perteneciente al grupo 5 y se lo

entrena.

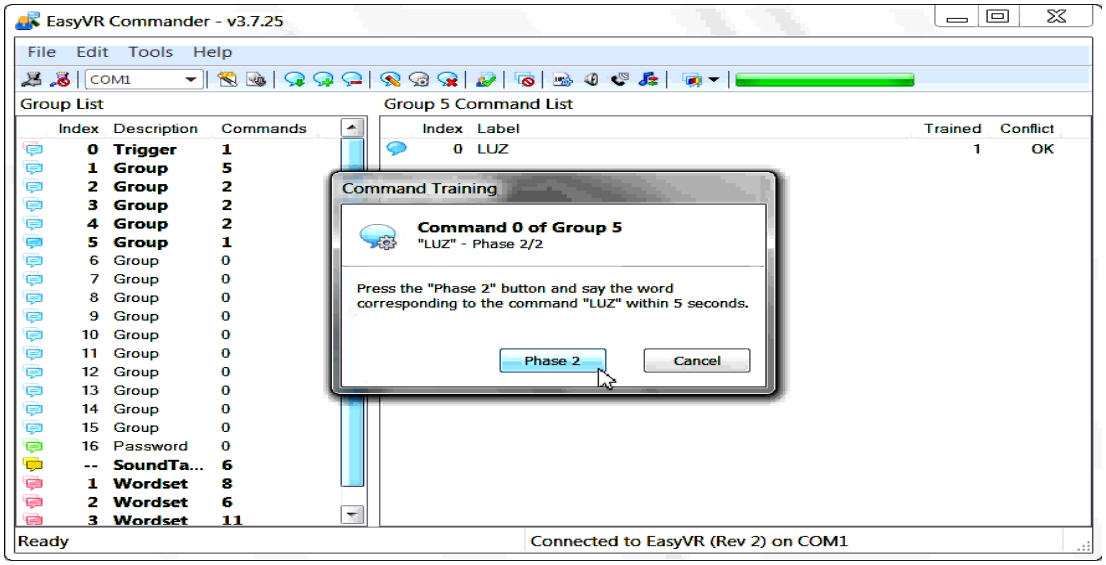


Figura 92. Añadiendo y entrenando comandos en grupo 5 (G5)

Por otra parte si el objetivo es apagar las luces del habitáculo, entonces se enunciará el comando “apagar” en el grupo 1 y posteriormente el comando “luz” perteneciente al grupo 6 (G6), el mismo que ejecutará el código de programación (*véase página 130*) poniéndolo en un estado bajo (LOW) por un tiempo indefinido.

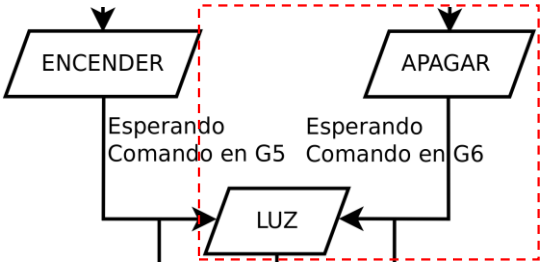


Figura 93. Diagrama de flujo perteneciente a G6

Posteriormente se procede añadir el comando “LUZ” perteneciente al grupo 6 (G6) y se lo entrena.

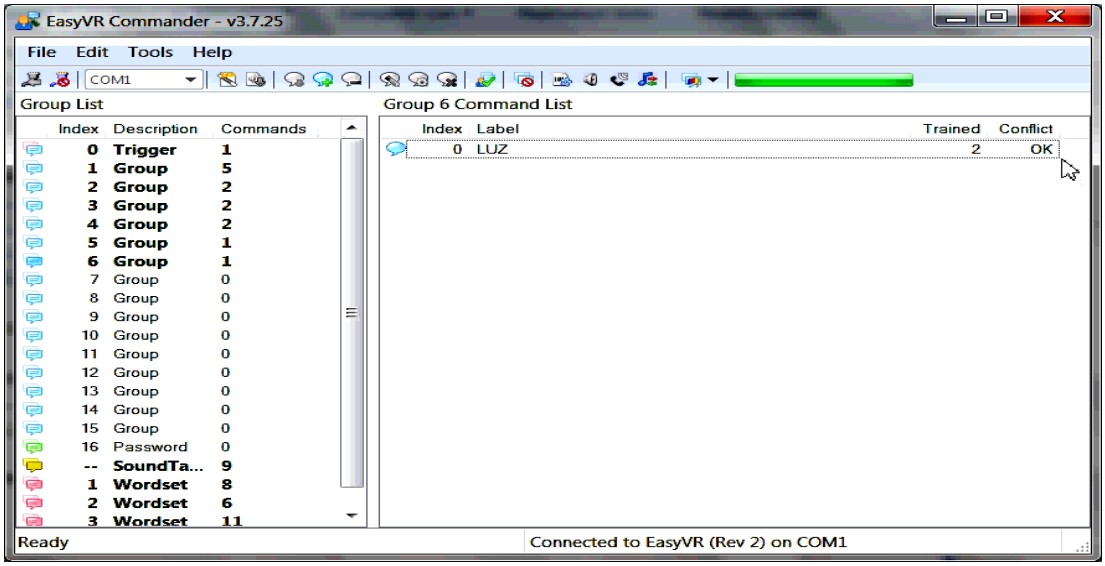


Figura 94. Añadiendo y entrenando comandos en grupo 6 (G6)

Si fuera el caso en que se necesita abrir la ventana (completamente) o las puertas, entonces se procede a mencionar el comando “abrir” en el grupo 1 y a continuación se elegirá la opción deseada (ventana o puertas) en el grupo 7 (G7).

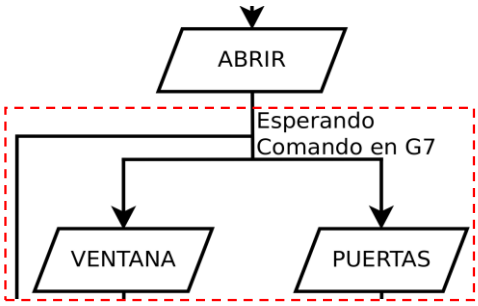


Figura 95. Diagrama de flujo perteneciente a G7

Si la palabra manifestada corresponde a la opción “puertas” (abrir – puertas), entonces el código de programación concerniente a dicho comando es ejecutado de manera inmediata (véase página 131).

A continuación se añaden y entrenan los comandos referentes (ventana - puertas) al grupo 7.

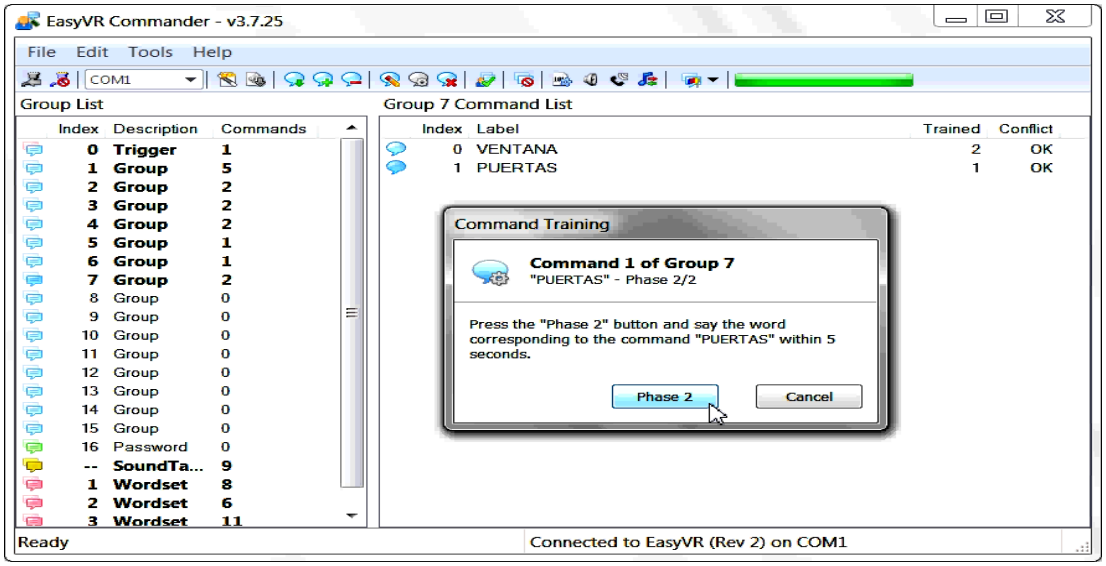


Figura 96. Añadiendo y entrenando comandos en grupo 7 (G7)

Ahora si el comando elegido pertenece al vocablo “ventana” (abrir – ventana) en el grupo 7, entonces se tendrá dos posibilidades más a elegir (izquierda o derecha) en el grupo 8 (G8), las mismas que ejecutarán el código de programación correspondiente (véase página 132).

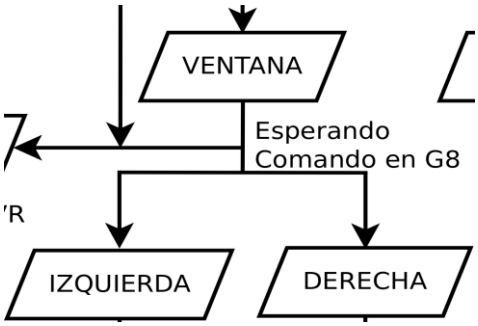


Figura 97. Diagrama de flujo perteneciente a G8

Cabe indicar que el código de programación utilizado en el caso de abrir una ventana (completamente), se mantendrá ejecutado durante el tiempo estimado que lleva realizar dicha acción. Es decir que si la acción de abrir la ventana por completo requiere de 5 segundos aproximadamente, entonces la constante booleana (HIGH) en el código de programación tomará dicho valor o tiempo (delay).

Se añaden los comandos pertenecientes al grupo 8 (G8) y se los entrena.

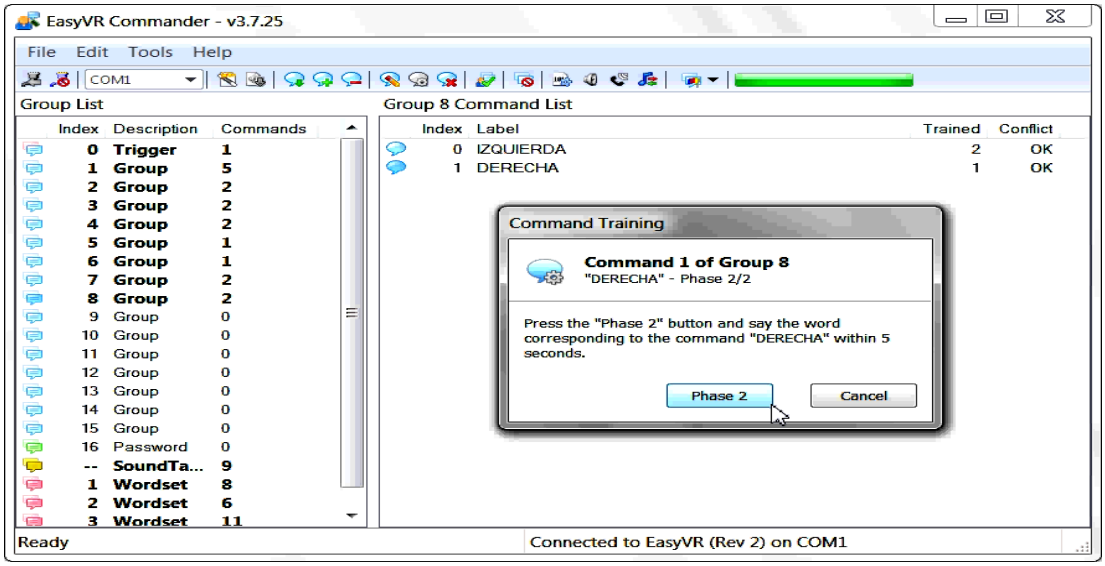


Figura 98. Añadiendo y entrenando comandos en grupo 8 (G8)

De igual manera ocurre con la opción “cerrar” del grupo 1, donde se tiene dos probabilidades idénticas a las del grupo 7 (puertas o ventana). Claro está que a comparación con este último grupo (G7) encomendado a abrir las ventanas y puertas; el grupo 9 (G9) se encargará de cerrar las mismas al ejecutar el código fuente pertinente para dicha acción (véase página 133).

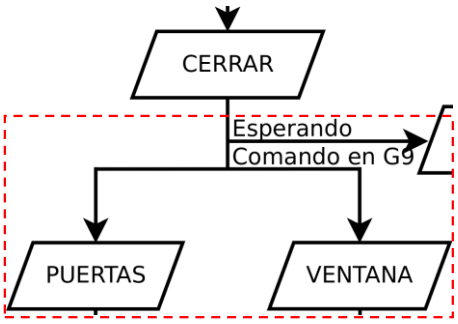


Figura 99. Diagrama de flujo perteneciente a G9

Por otra parte tanto la acción de “abrir” como la de “cerrar” una ventana puede ser utilizada por propósito general, es decir que si utilizamos los comandos “subir” o “bajar” (ventana) en los grupos 3 y 4, es posible abrir o cerrar dicha ventana por completo partiendo de esta última posición tomada.

Se añaden y entrenan los comandos correspondientes al grupo 9 (G9).

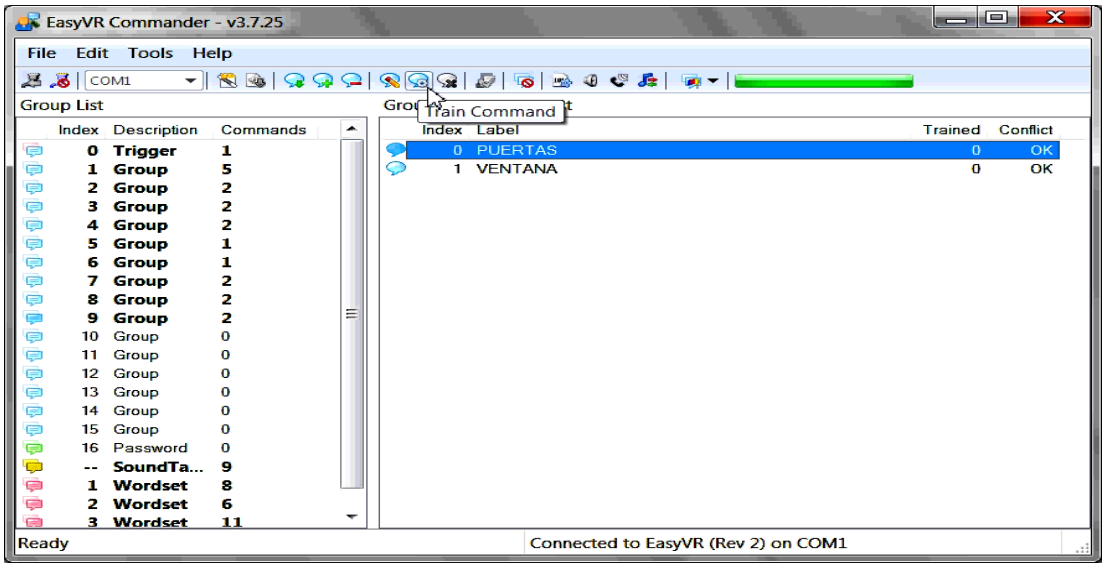


Figura 100. Añadiendo y entrenando comandos en grupo 9 (G9)

En el caso de elegir la opción “ventana” (cerrar – ventana), entonces las posibilidades a elegir son dos (izquierda o derecha) en el grupo 10 (G10), las mismas que definirán la ventana a ser cerrada por medio del código de programación ejecutado (véase página 133).

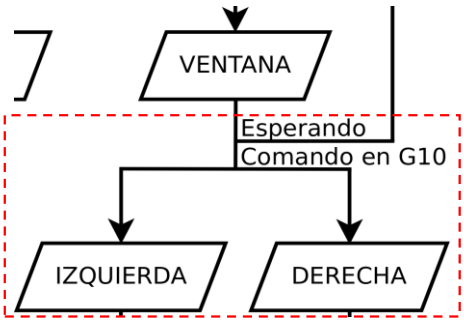


Figura 101. Diagrama de flujo perteneciente a G10

A continuación se añaden y entrenan los comandos correspondientes al grupo 10 (G10), los mismos que permitirán cerrar las ventanas de manera completa acode al lado elegido.

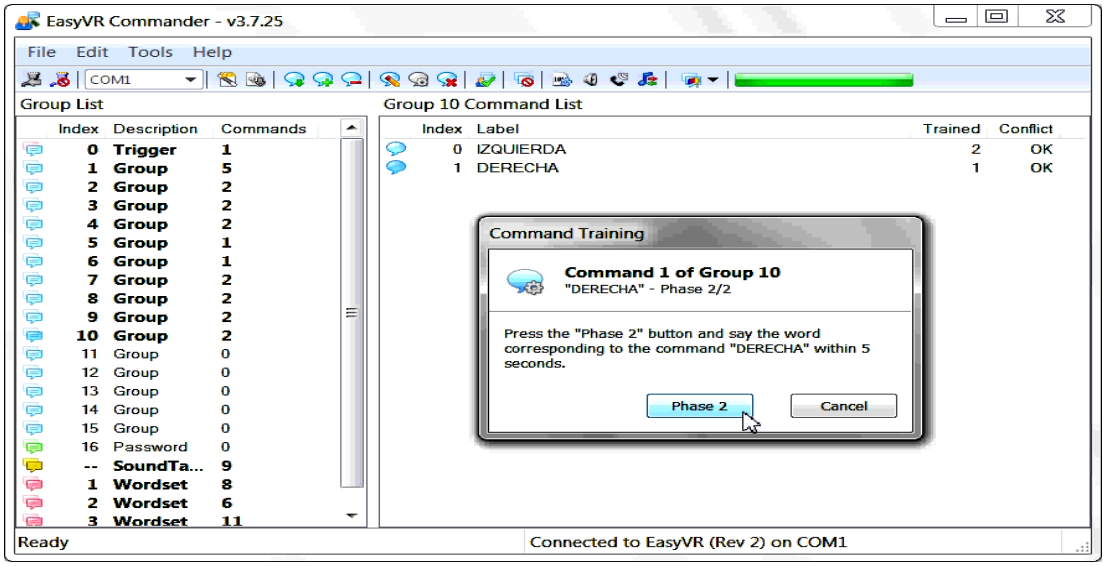


Figura 102. Añadiendo y entrenando coamndos en grupo 10 (G10)

Finalmente se entrenan los comandos o caracteres de escape correspondientes al grupo 11 (G11), los cuales permitirán salir de cualquier grupo en que se encuentren los usuarios en el supuesto caso de equivocarse. Es decir que mediante el comando “salir”, se tendrá la capacidad de regresar nuevamente al encabezado principal o grupo uno (G1).

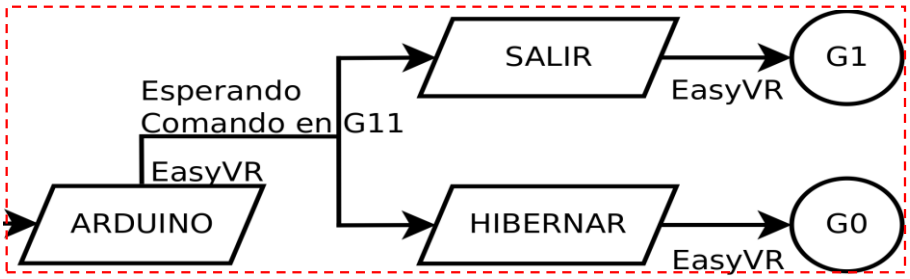


Figura 103. Diagrama de flujo perteneciente a G11

Por otra parte también se añadirá el comando “hibernar” el mismo que permitirá congelar el sistema desde cualquier grupo (G1-G10), es decir que se lo pondrá en modo “inicio” al ser enviado al grupo cero (G0); necesitando por ende de la palabra clave “ARDUINO” en dicho grupo (G0) para arrancar una vez más el sistema en el grupo cabecera (G1).

Se añaden y entrenan los comandos pertenecientes al grupo 11 (G11).

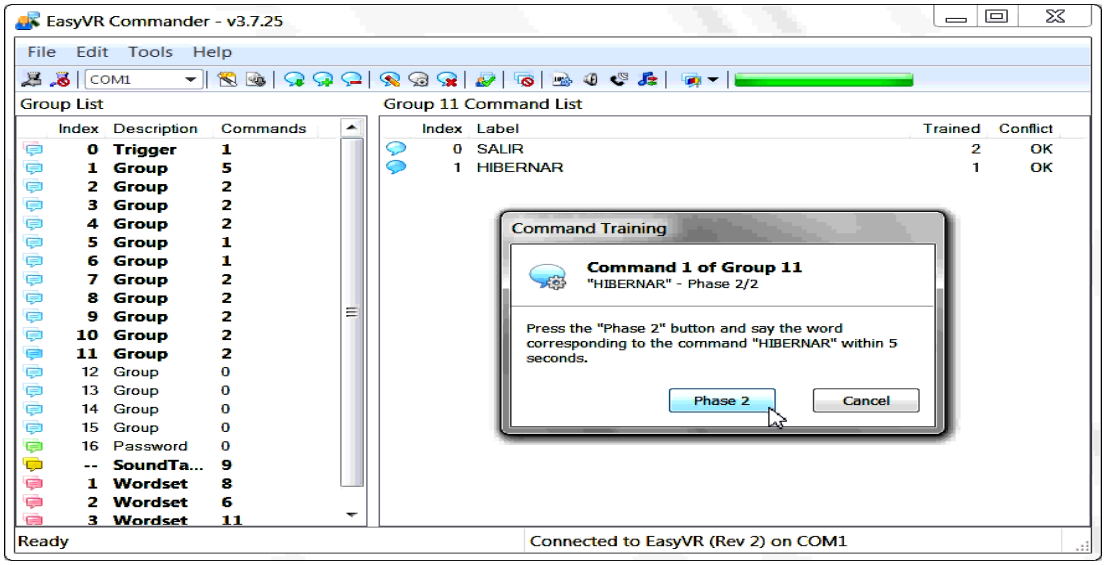


Figura 104. Añadiendo y entrenando comandos en grupo 11 (G11)

Ahora únicamente resta añadir un comando más que permita llamar al grupo 11 (G11) en cualquier grupo que nos encontremos (G1-G10), para lo cual se utilizará una vez más la palabra “ARDUINO” (puede ser cualquier otro comando) y se lo añadirá en todos los grupos (G1-G10) a excepción del grupo cero (G0) y el grupo 11 (G11) ya que no es necesario por razones lógicas (véase diagrama de flujo).

Para comprender de una mejor manera, en el diagrama de flujo (ver figura 74) se puede observar que los grupos G1, G2, G3, G4, G5, G6, G7, G8, G9 y G10 poseen una salida o mejor dicho un comando más a ser elegido (enunciado) en dichos grupos. Este comando no es más que la palabra anteriormente escogida “ARDUINO”, la misma que permitirá enlazarse con el “grupo de escape” (G11) al ser mencionada.

Se añade y entrena el carácter de escape “ARDUINO” en los grupos correspondientes del uno al diez (G1-G10).

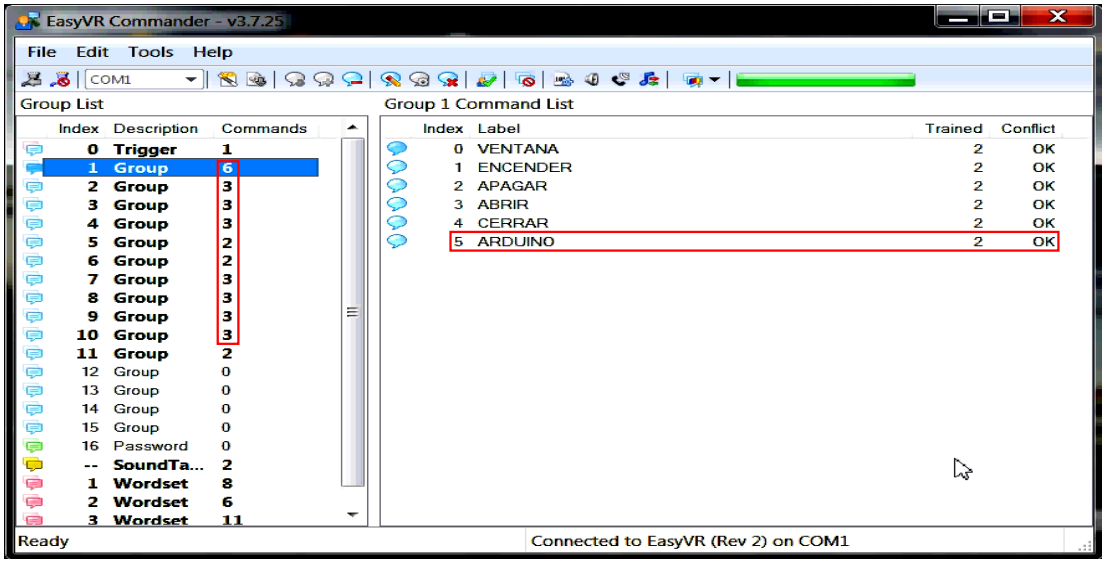


Figura 105. Carácter de escape “ARDUINO” añadido en G1-G10

Como se puede observar en el área de grupos, se ha añadido un comando más (ARDUINO) para cada uno de los grupos anteriormente citados. Por otra parte se debe indicar que si se necesita comprobar cualquier comando dentro de un grupo, basta con dar click en la pestaña “Test Group” y a continuación se pronuncia el comando deseado, de manera que si fue entrenado correctamente se marcará de color verde; caso contrario se puede volverlo a entrenar marcando la opción “Erase Training” en la barra de tareas para luego dar click una vez más en la opción “Train Command”.

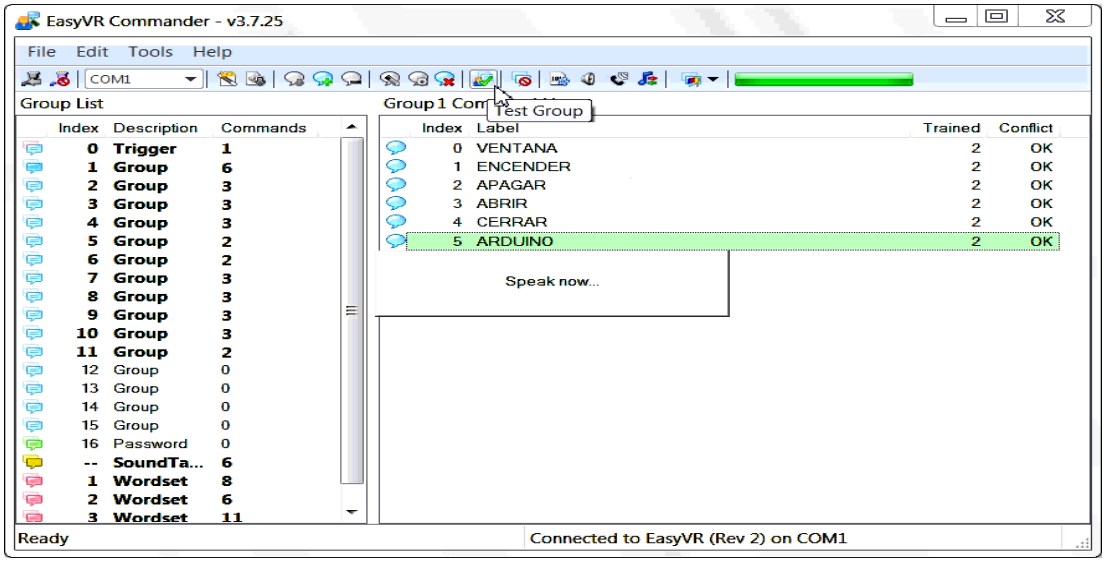


Figura 106. Comprobando comandos entrenados

Finalmente se procede a dar click en la pestaña “Generate Code”.

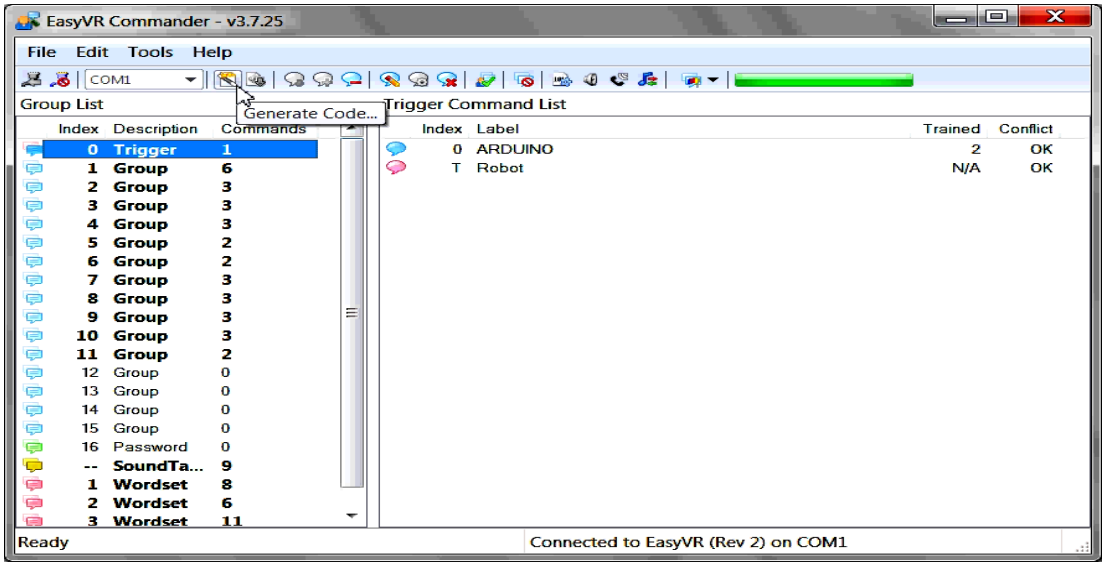


Figura 107. Botón “Generar Código”

Y a continuación se pedirá el directorio donde se desea guardar el código generado, para lo cual es muy importante crear una carpeta con el mismo nombre del archivo a guardar (.pde); ya que de otra manera no se abrirá el código generado con los comandos entrenados en el IDE de Arduino.

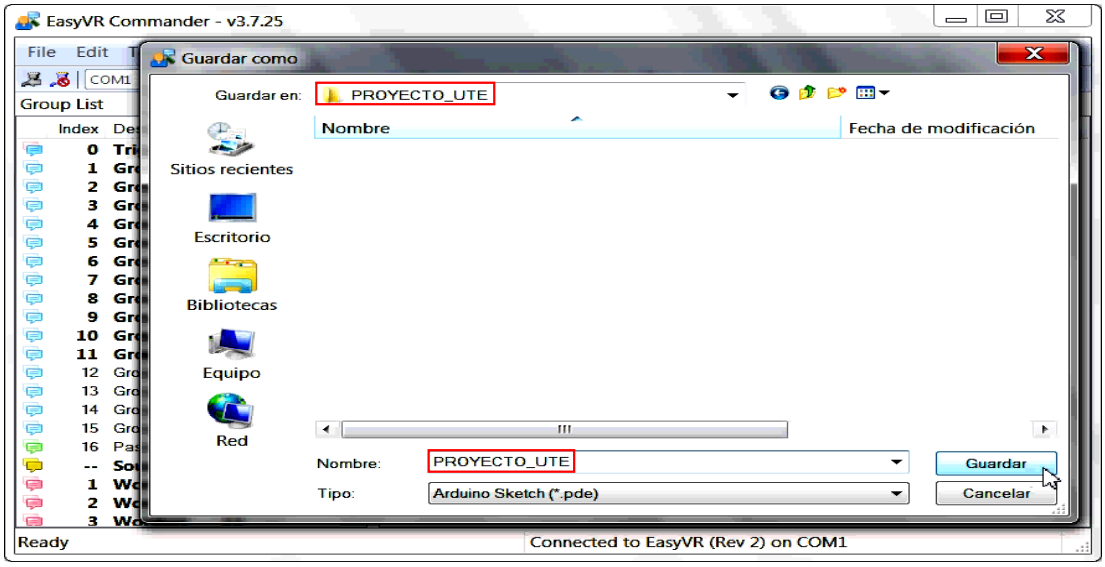


Figura 108. Guardando el archivo del proyecto

Cabe aclarar que dicho código ahora generado únicamente corresponde a las dependencias necesarias para la comunicación serie entre los dispositivos, es decir los fabricantes de la línea “TIGAL Veear” nos facilitan el código para que el dispositivo esclavo “EasyVR” pueda comunicarse con el controlador “Arduino” sin ningún tipo de problema, esto incluye librerías necesarias, velocidad de transmisión de datos, comunicación serie, tasas de refresco de la memoria RAM, idioma elegido, reconocimiento del comando de voz grabado, etc. Debido principalmente a que los productos de “veear” son diseñados para aquellas personas (estudiantes, artistas, diseñadores, aficionados y profesionales) que desean llevar a cabo un proyecto de investigación sin la necesidad de poseer conocimientos avanzados más que los enseñados en el presente trabajo de titulación; mientras que su alcance únicamente está limitado por nuestra imaginación.

Por otra parte se debe recordar que antes de interrumpir al dispositivo de la fuente de alimentación (USB), se debe desconectar primero del software en ejecución (EasyVR Commander) así como lo indica la figura 109.

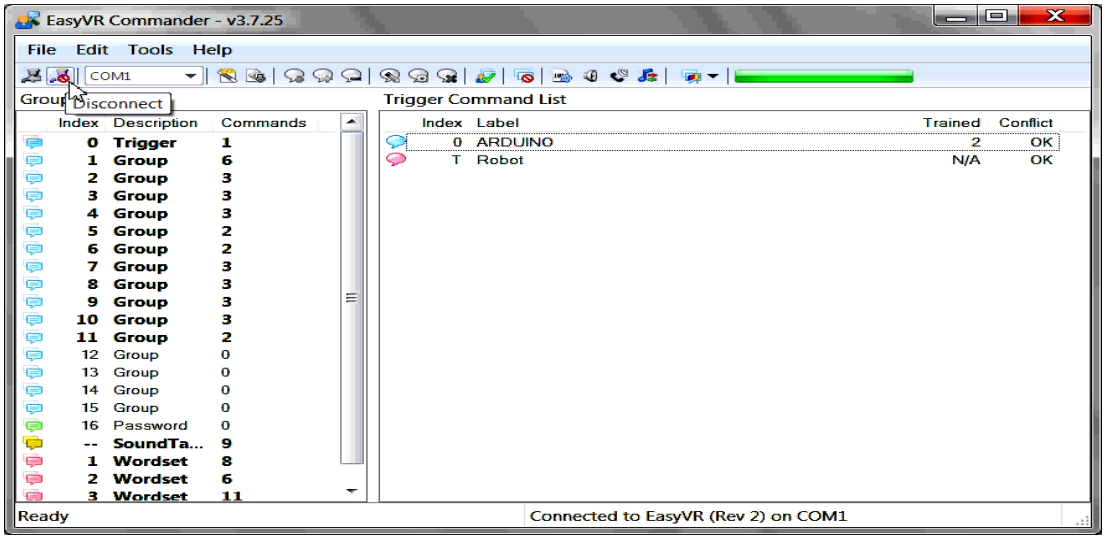


Figura 109. Botón “Desconectar”

A continuación se creará y subirá la tabla de sonidos a reproducir con la finalidad de pulir aún más el presente proyecto de investigación y con el objetivo de dar paso al siguiente tema denominado “diseño del código

fuente”.

3.2.3.5. Creando la tabla de sonidos

La tarjeta de reconocimiento de voz EasyVR tiene la capacidad de reproducir sonidos previamente grabados en su memoria interna. Estos sonidos son organizados en lo que se llama una tabla de sonidos, la misma que es obtenida mediante el programa “*QuickSynthesys*”, una aplicación que se instala por “default” durante el proceso de instalación del EasyVR Commander y es la que permitirá comprimir la tabla de sonidos deseada en un formato de tamaño apropiado (reducido) para el dispositivo esclavo EasyVR.

Para construir dicha tabla de sonidos lo primero que debemos hacer es conseguir los sonidos deseados en los formatos apropiados, es decir deben tener un formato ***WAV*** a ***16 bits sin compresión*** y deben estar en un sólo ***canal mono a 22050 Hz***. Para esto se puede utilizar un sinnúmero de programas los cuales serán de gran utilidad en el proceso de conversión.

En este caso se utilizó un lector de texto online (http://www.lumenvox.com/espanol/products/tts/) capaz de reproducir las palabras escritas en diferentes idiomas y acentos (voz de robot); además de un grabador de audio y vídeo llamado “*RecordMyDesktop*”, el cual permitió grabar el escritorio (ordenador) en formato ogv (software libre), el mismo que se lo convirtió a un formato de tipo WAV (sólo audio) con las características anteriormente descritas; para este fin se utilizó el programa *“Total Video Converter”* y el editor de audio *“Audacity”.*

Una vez que se obtienen los sonidos deseados a reproducir por la tarjeta Shield EasyVR en el formato indicado, se procede a ejecutar el programa “*QuickSynthesys 5*” instalado anteriormente (default); a continuación se dará click en la pestaña archivo (File) y se elegirá la opción crear nuevo proyecto (New).

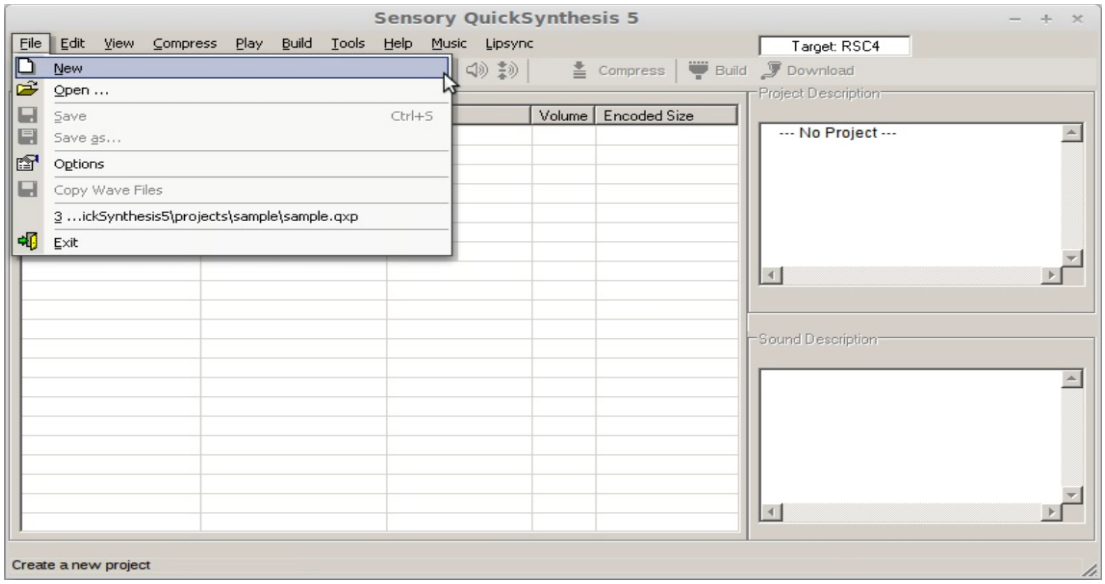


Figura 110. Creando un nuevo proyecto

Se abrirá una ventana donde se elegirá a la familia RSC4.

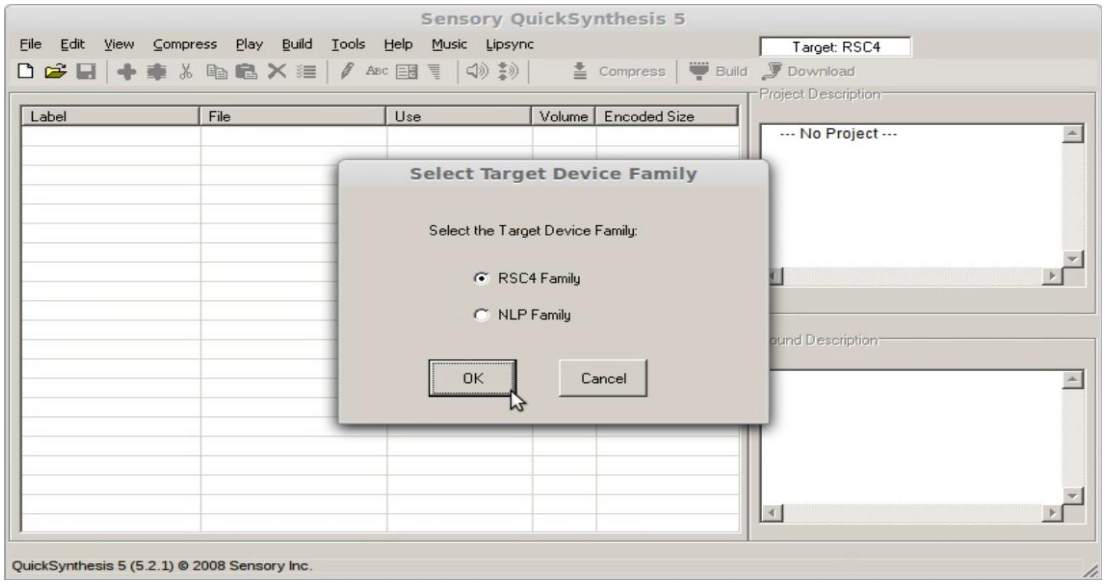


Figura 111. Eligiendo la familia RSC4

A continuación se dará un nombre al archivo sin olvidarse de escribir su extensión (.qxp) que en este caso quedo de la siguiente manera “Tabla_Sonidos_UTE.qxp”. Damos click en abrir y se aparecerá una tabla con la fecha y nombre del proyecto; ahora únicamente resta añadir los sonidos WAV creados anteriormente.

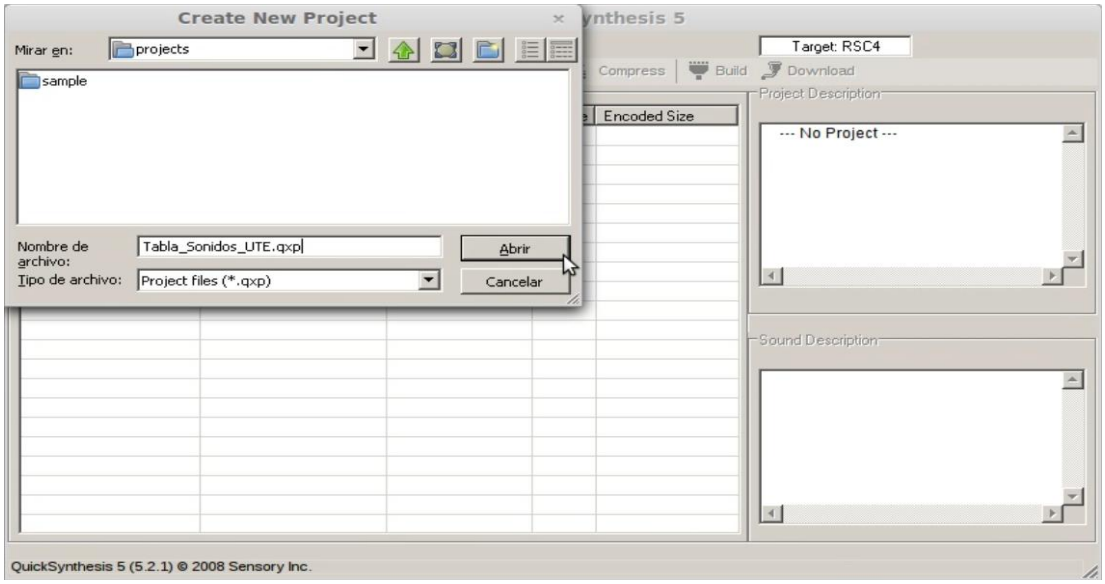


Figura 112. Eligiendo un nombre al archivo con extensión ".qxp"

Para añadir dichos sonidos se debe dirigirse a la pestaña *editar* (Edit) y elegir la opción *añadir archivo WAV* (Add Wav file).

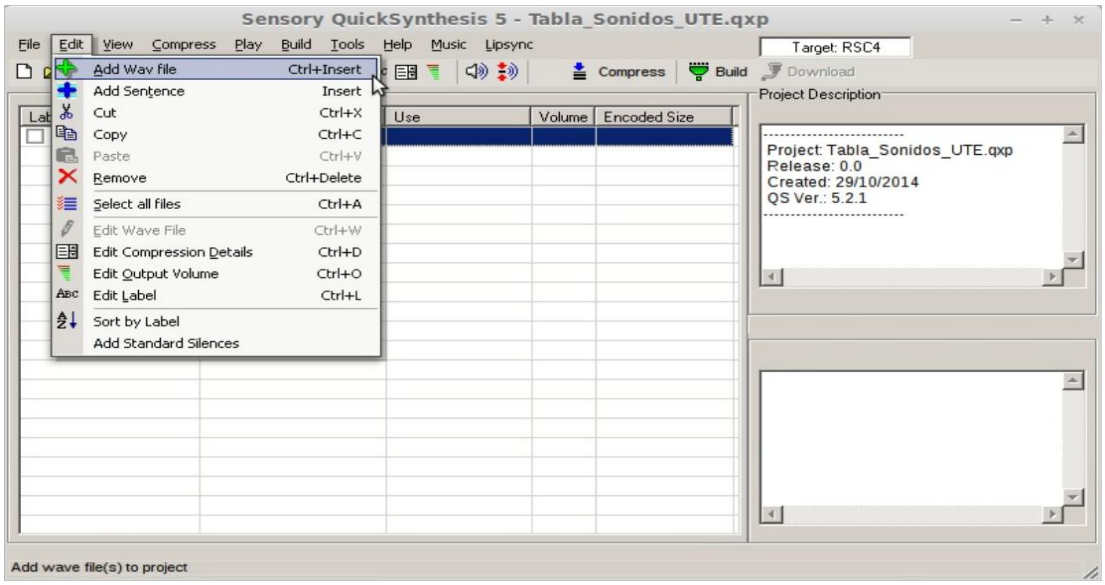


Figura 113. Botón "Añadir Archivo WAV"

Se mostrará una ventana donde se procede a buscar la carpeta que contiene los sonidos a reproducir deseados (WAV). Una vez encontrados dichos sonidos, los seleccionamos y damos click en el botón abrir.

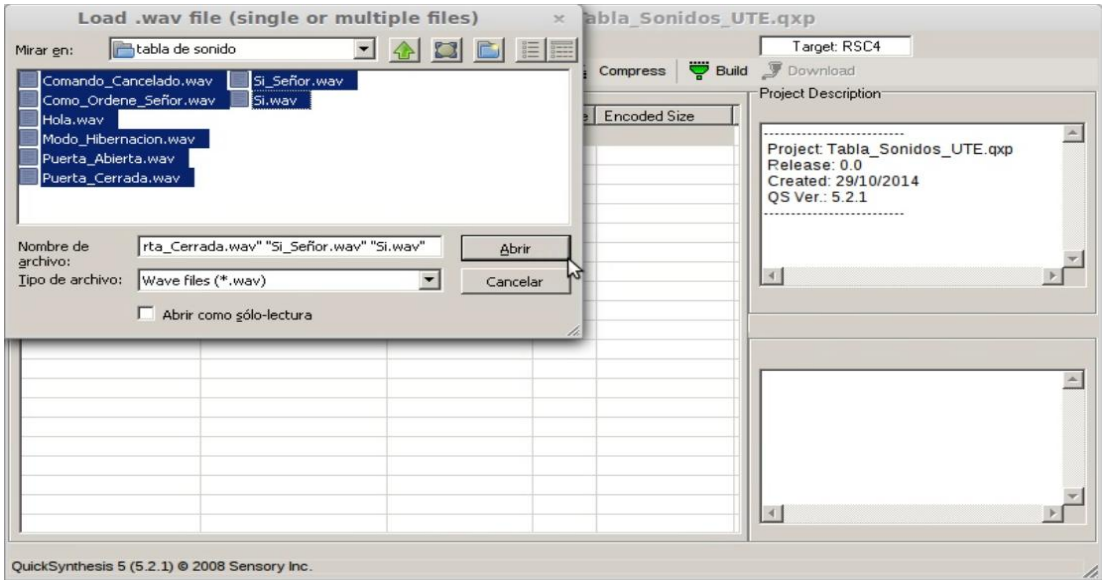


Figura 114. Abriendo los archivos de sonido en formato WAV

Posteriormente se abrirá una ventana donde se pedirá indicar el tipo de compresión a realizar. En la tabla 12 se puede observar el tiempo disponible de la tarjeta EasyVR para cada tipo de compresión. Cabe indicar que a menor compresión mejor será la calidad de sonido; aunque el precio a pagar se refleja en el espacio ocupado (memoria) por este.

Tabla 12. Tipos de compresión

Compresión	Tiempo disponible (8KHz)	Tiempo disponible (9.3KHz)
SX-2	8.7 minutos	7.5 minutos
SX-3	7.6 minutos	6.6 minutos
SX-4	6.8 minutos	5.9 minutos
SX-5	6.1 minutos	5.2 minutos
SX-6	5.6 minutos	4.8 minutos
ADPCM de 4 bits	87 segundos	No disponible
OCM de 8 bits	45 segundos	38 segundos

Debido a que para este proyecto se utilizó un número de archivos de sonido relativamente pequeños, se elegirá el tipo de compresión más común “SX-6 8k” que viene por “default” y luego se dará click en el botón “OK For All”.

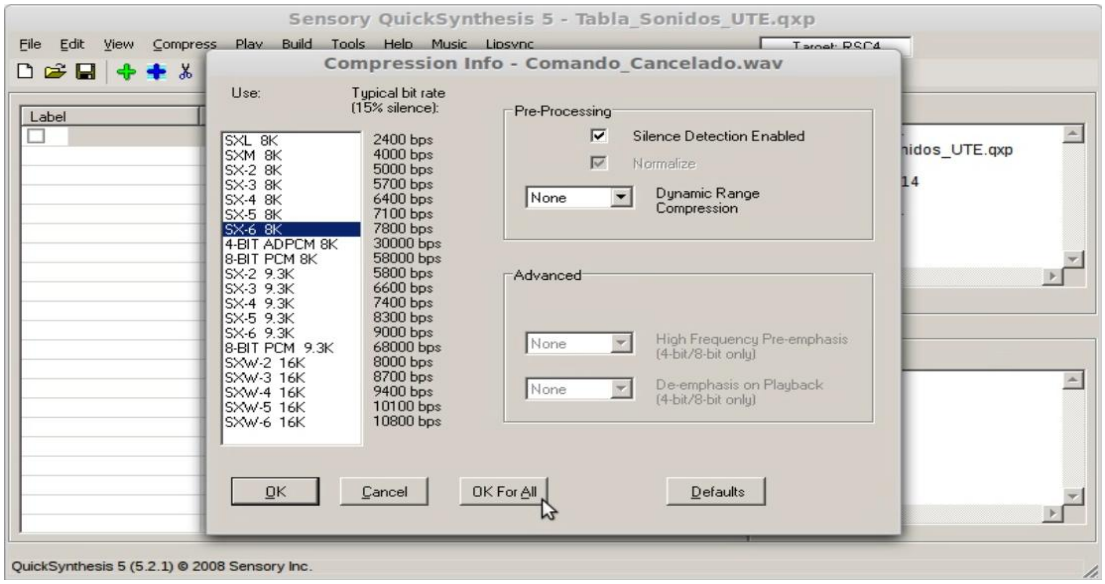


Figura 115. Eligiendo el tipo de compresión deseado

A continuación se indicará un cuadro para cambiar el nombre del archivo o etiqueta (opcional), damos click en “OK” en caso de cambiar la etiqueta o en “Auto Label All” (recomendado) si se desea que el archivo tome el nombre del original. Ahora la tabla se encuentra lista para ser comprimida, de manera que se procede a seleccionar todos los archivos para finalmente dar click en la pestaña “Compress” así como lo indica la figura 116.

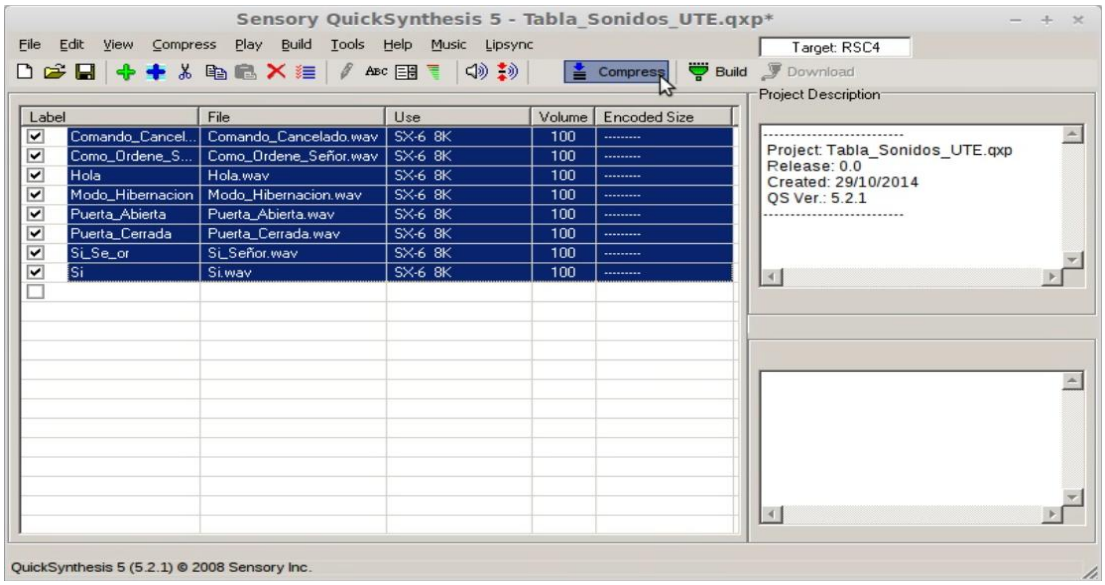


Figura 116. Comprimiendo la tabla de sonidos

Al finalizar la compresión de la tabla de sonidos se indicará un mensaje como el siguiente.

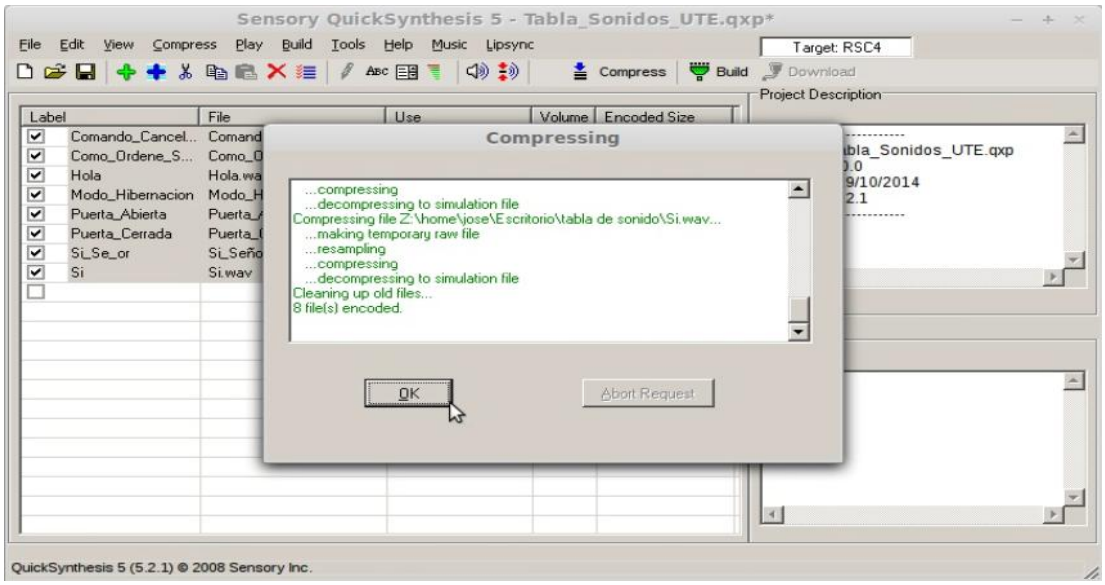


Figura 117. Compresión realizada exitosamente

Luego se da click en “OK” y finalmente se deberá dirigirse a la pestaña “Build”, verificando que todos los archivos sigan seleccionados.

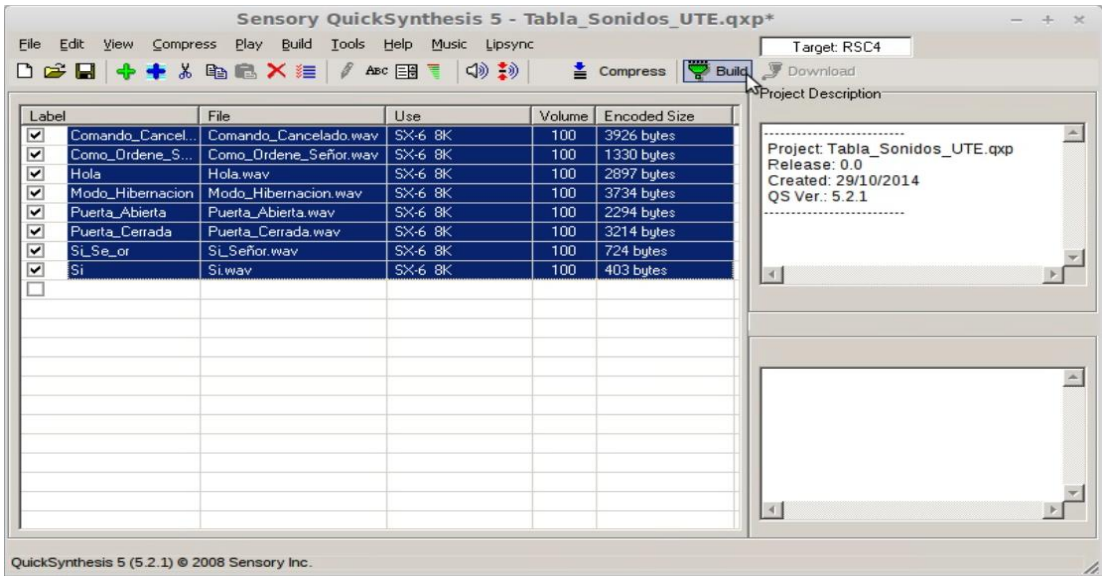


Figura 118. Construyendo la tabla de sonidos

Se aparecerá una ventana durante la construcción del proyecto (Build),

donde utilizando los ajustes por defecto (default): “Build linkable module”, “Load in CONST space”, “Load above or at: **0**” y “Load below or at: **1**”

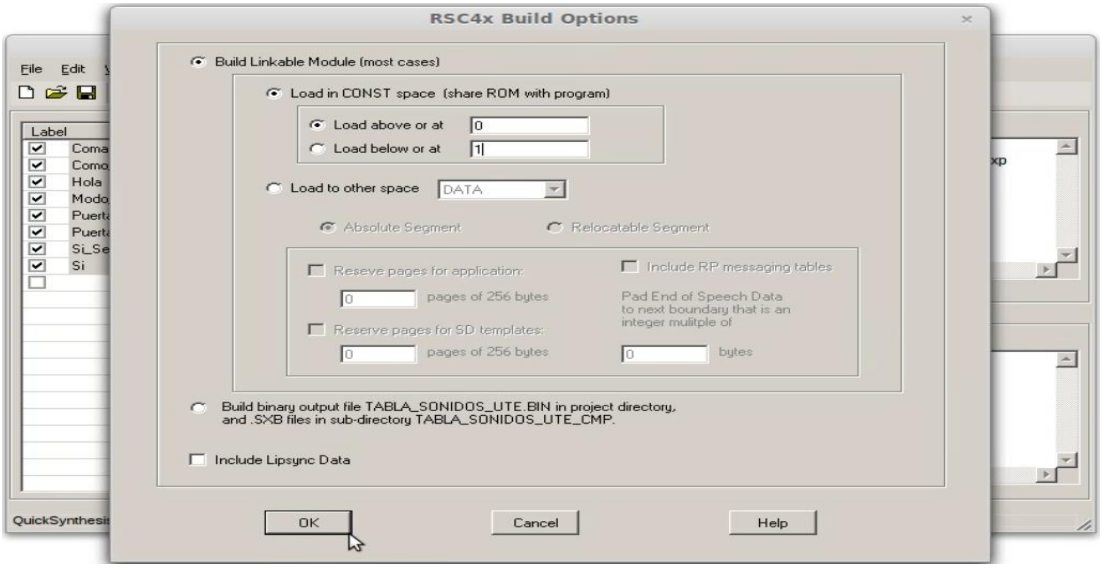


Figura 119. Ajustes por defecto en la construcción de la tabla de sonidos

Se procede a dar click en “OK” y al finalizar la construcción surgirá un mensaje como el siguiente.

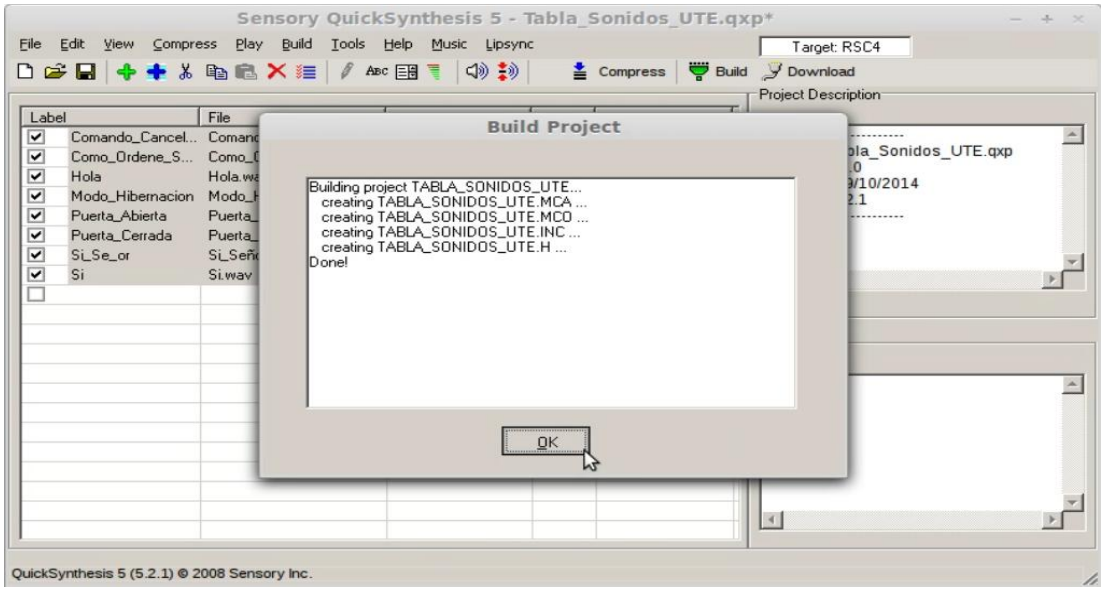


Figura 120. Construcción de la tabla de sonidos exitosa

Se acepta (OK) y listo la tabla de sonidos ha sido creada satisfactoriamente,

ahora solo queda guardar el proyecto, el mismo que se lo utilizará posteriormente a la hora de subir los sonidos (tabla de sonidos) a la memoria de la tarjeta EasyVR.

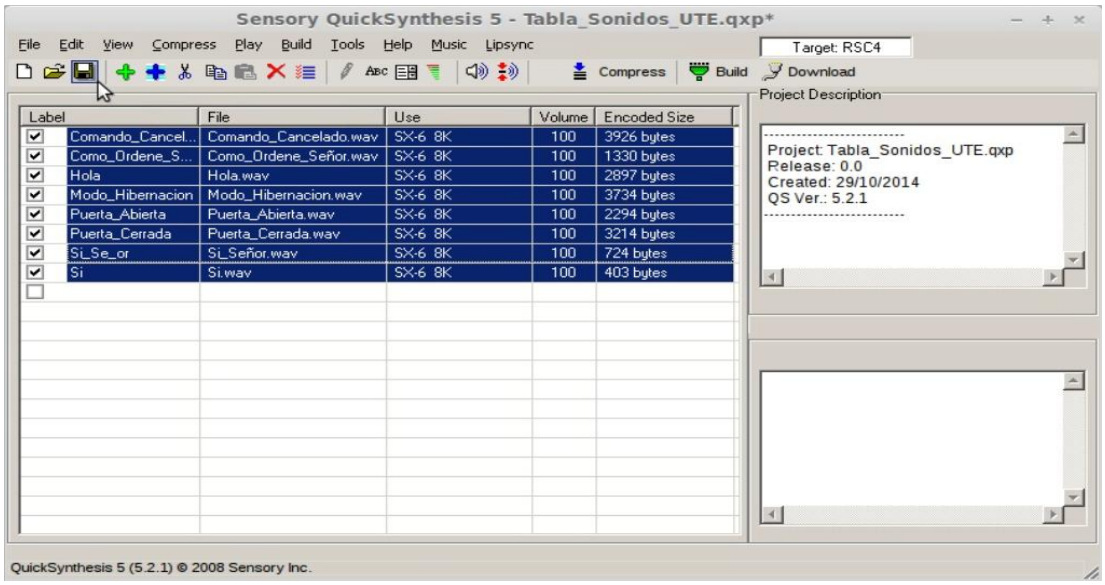


Figura 121. Guardando la tabla de sonidos

3.2.3.6. Subiendo la tabla de sonidos

Una vez creada la tabla de sonidos, es tiempo de subir dicho archivo (Tabla_Sonidos_UTE.qxp) a la memoria interna de la tarjeta "EasyVR" Shield, para lo cual se colocará el "jumper" en la posición "***UP***".

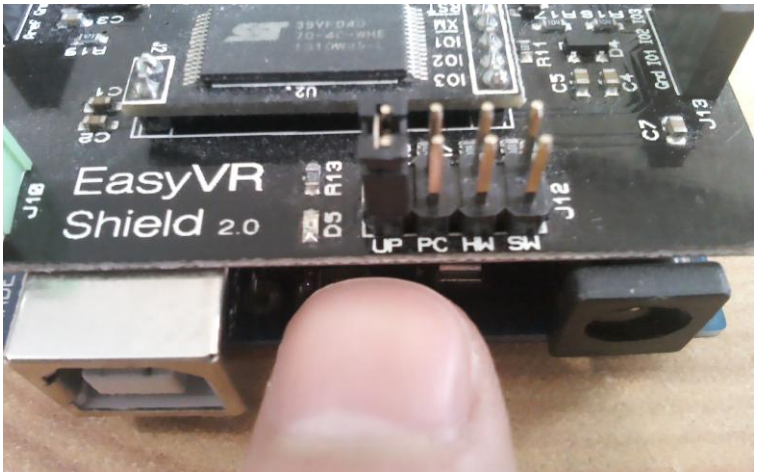


Figura 122. Jumper EasyVR en posición UP

A continuación se conecta el dispositivo (Arduino + EasyVR Shield) al ordenador y se ejecuta la interfaz gráfica “EasyVR Commander”. Para luego seleccionar el puerto COM correspondiente y dar click en la pestaña “Update Custom Data” así como lo indica la figura inferior.

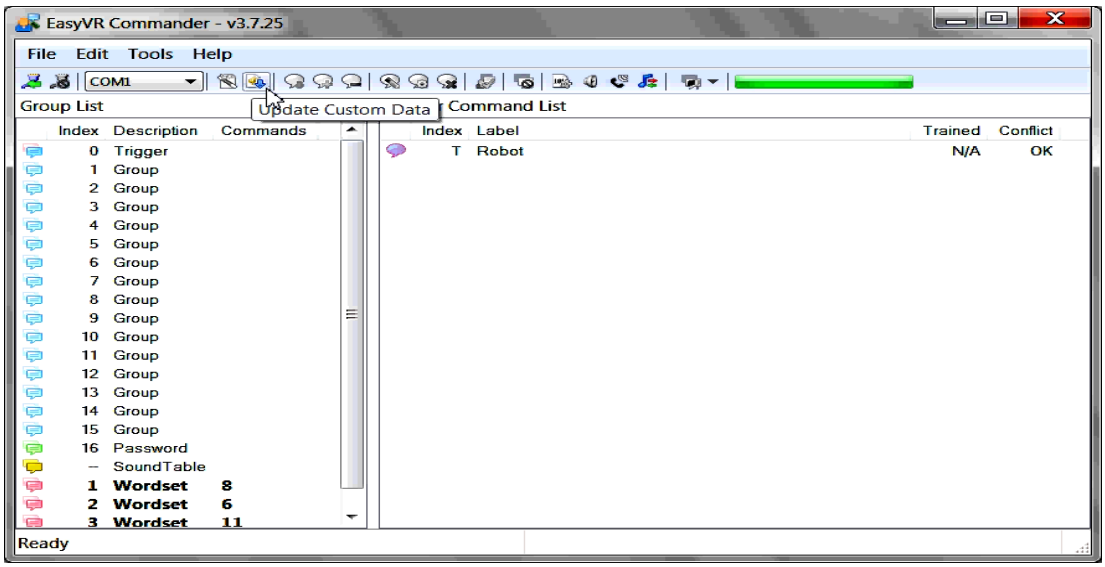


Figura 123. Botón “Subir Tabla de Sonidos”

Se abrirá un cuadro donde se importará el proyecto creado (guardado) anteriormente, para lo cual se dará click en la opción “Import”.

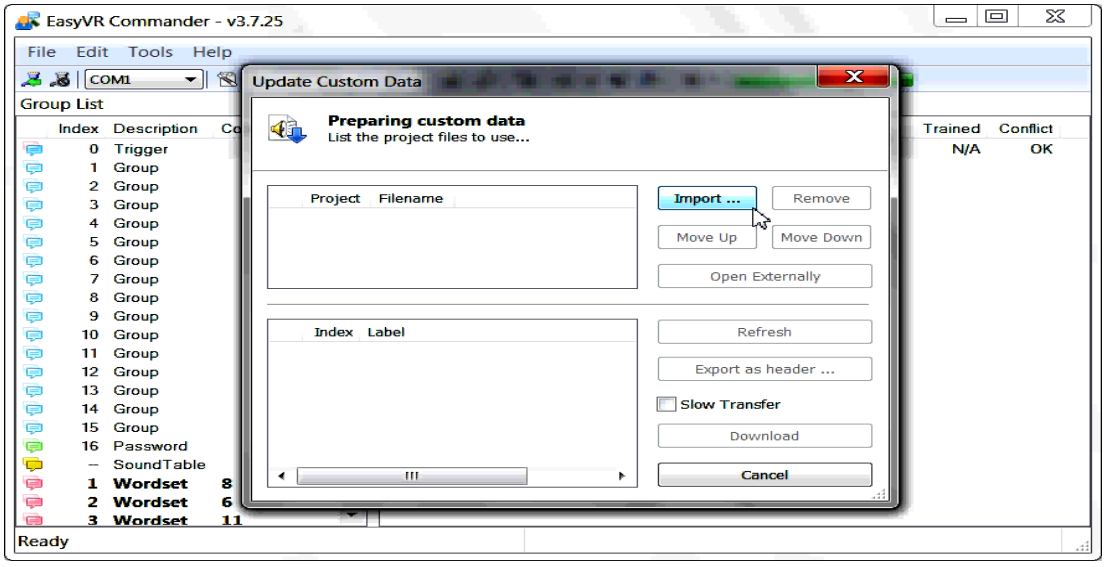


Figura 124. Importando tabla de sonidos creada

A continuación se abre la tabla de sonidos creada anteriormente, la misma que si recordamos se la denominó “Tabla_Sonidos_UTE.qxp”.

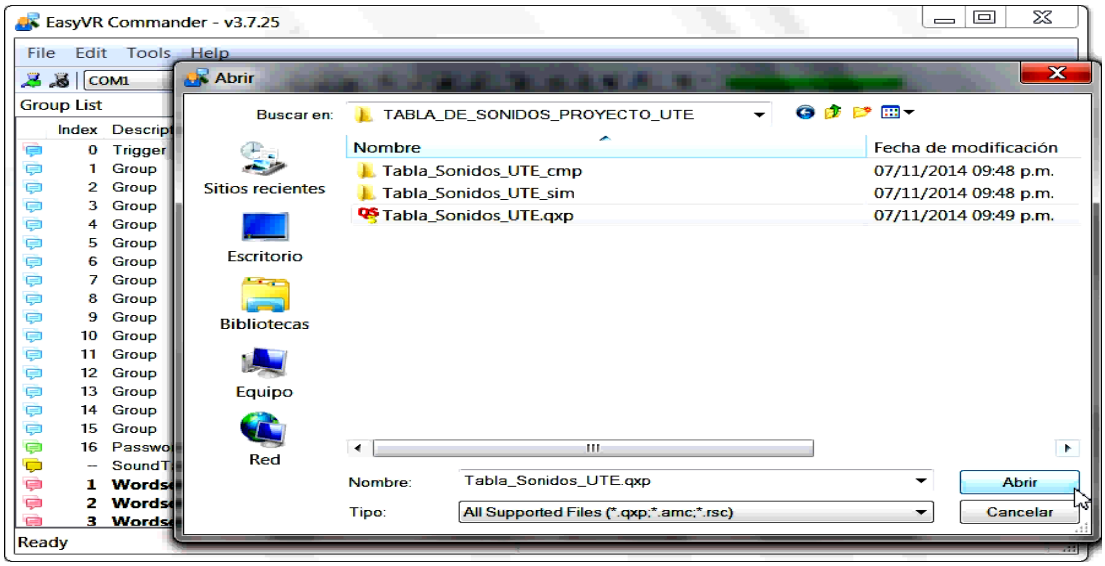


Figura 125. Abriendo el proyecto “Tabla_Sonidos_UTE.qxp”

Una vez importados los sonidos a reproducir deseados (tabla de sonidos), se selecciona la casilla “Slow Transfer” y a continuación se procede a cargar la tabla de sonidos a la memoria interna del dispositivo. Para lo cual se debe dar click en la opción “Download” y listo la tabla de sonidos comenzará a subir (cargar).

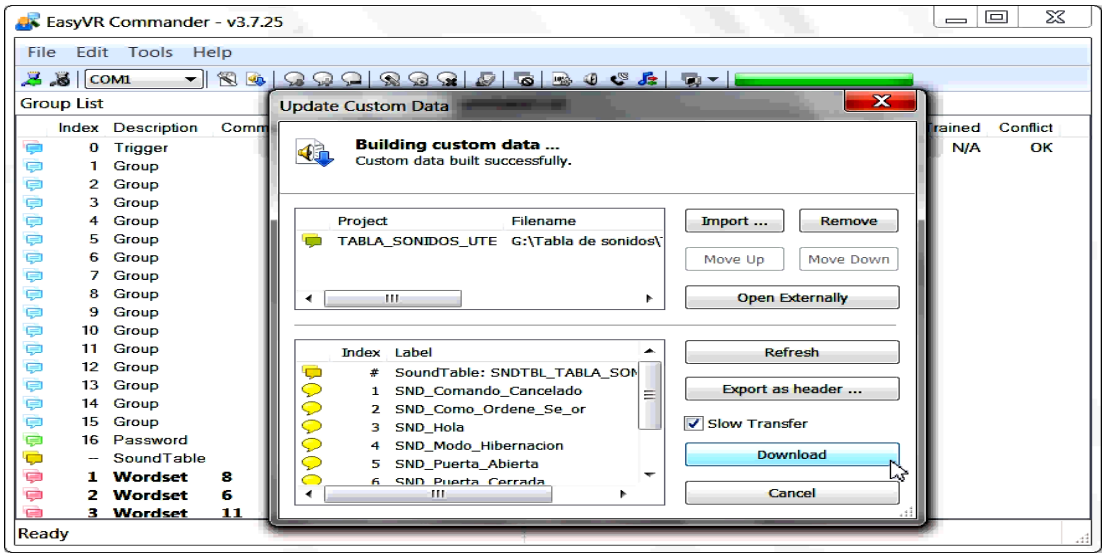


Figura 126. Subiendo la tabla de sonidos a la memoria interna

Para constatar de que la tabla de sonidos ha sido subida exitosamente a la memoria interna de la tarjeta de reconocimiento de voz “EasyVR”, es necesario colocarla en modo puente o “Bridge” con el fin de establecer la conexión entre el dispositivo y el software EasyVR Commander una vez más (*véanse páginas 89, 90, 91 y 92)*. A continuación se cargarán todos los grupos y comandos entrenados anteriormente así como los sonidos a reproducir deseados dentro de la opción “SoundTable” (área de grupos).

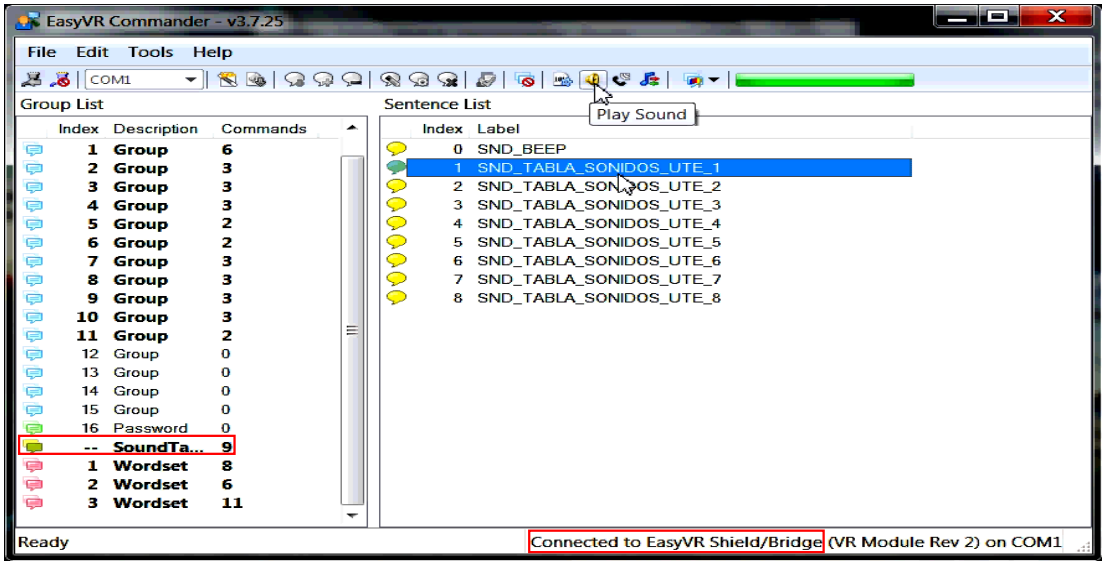


Figura 127. Tabla de sonidos cargada

Es muy importante anotar el número o index correspondiente a cada sonido, ya que estos serán declarados en el código de programación para su posterior ejecución. De manera que es necesario conectar los auriculares (3.5mm) a la toma “J9” de la tarjeta Shield EasyVR (*véase figura 61*), con la finalidad de poder escuchar cada uno de los sonidos cargados mediante su selección y posterior reproducción gracias al icono “Play Sound” situado en el área de tareas en forma de una pequeña bocina (ver figura anterior).

3.2.3.7. Diseño del código fuente

Una vez entrenados todos los comandos y de haber subido la tabla de sonidos, se comenzará a diseñar el código fuente perteneciente a este

proyecto de investigación. Para lo cual se ejecuta el compilador IDE de Arduino y a continuación se elegirá la opción "*Archivo - Abrir",* donde se buscará y abrirá el archivo guardado anteriormente (PROYECTO_UTE.pde), el mismo que contiene el código de programación correspondiente a la comunicación serie entre los dispositivos (véanse páginas 110 y 111), así como la enumeración de grupos y comandos.

En el interior del código abierto se encontrará algunas dependencias y librerías necesarias como es el caso de #include "Arduino.h", #include "SoftwareSerial.h", #include "Wprogram.h", #include "NewSoftSerial.h" e #include "EasyVR.h". Por otra parte es posible observar que los pines digitales 12 y 13 serán ocupados durante la comunicación serie (SoftwareSerial port(12,13);) entre los dispositivos (jumper en posición "SW"), por lo que no se utilizarán dichos pines al momento de declarar las salidas (pinMode) en la función "void setup()".

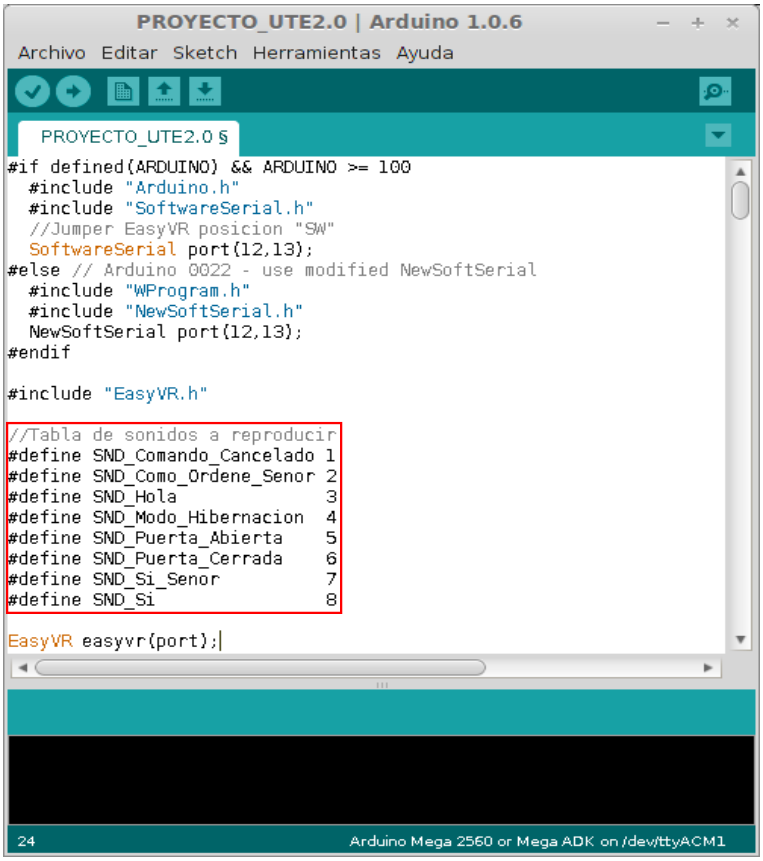


Figura 128. Definiendo la tabla de sonidos

En la cabecera del programa se definirán los sonidos a reproducir (tabla de sonidos) mediante la directiva **#define**, seguido de la constante “SND_” (sound) y posteriormente del nombre o etiqueta con su respectivo número (*véase página 122*) así como lo muestra la figura 128. Cabe indicar que el sonido “SND_BEEP” perteneciente al index (número) 0 no es necesario declararlo ya que este viene por “default”; mientras que por otra parte está prohibida la utilización de signos (acentos) o letras (ñ) que no correspondan al idioma inglés.

Además es importante recordar que se debe tener cuidado con la sintaxis del lenguaje de programación C, C++; ya que si olvidamos escribir un punto y coma (fin de línea) o simplemente se pasa por alto el cierre de llaves que contienen los estamentos (instrucciones), a la hora de compilar se emitirá un error de programación el cual se deberá corregir durante su depuración.

A continuación se procede a declarar los pines digitales como salidas (OUTPUT) dentro de la función “void setup()”, para lo cual se utilizará 8 pines ya que el módulo de relés a utilizar es de 8 canales (ocho relés). Por otra parte es importante recalcar que los pines digitales 0, 1, 12 y 13 son ocupados por el controlador Arduino y el dispositivo esclavo EasyVR durante la transmisión (ETX) y recepción (ERX) serie de datos, por lo que no se los declarará en dicha función.

Tabla 13. Acciones a realizar por los pines y relevadores correspondientes

Controlador	Módulo	Acción
Pin digital 10	Relevador 1	Abrir/Bajar Ventana Izquierda
Pin digital 9	Relevador 2	Cerrar/Subir Ventana Izquierda
Pin digital 8	Relevador 3	Abrir/Bajar Ventana Derecha
Pin digital 7	Relevador 4	Cerrar/Subir Ventana Derecha
Pin digital 6	Relevador 5	Encender/Apagar Luz
Pin digital 5	Relevador 6	Abrir Puertas
Pin digital 4	Relevador 7	Cerrar Puertas
Pin digital 3	Relevador 8	-

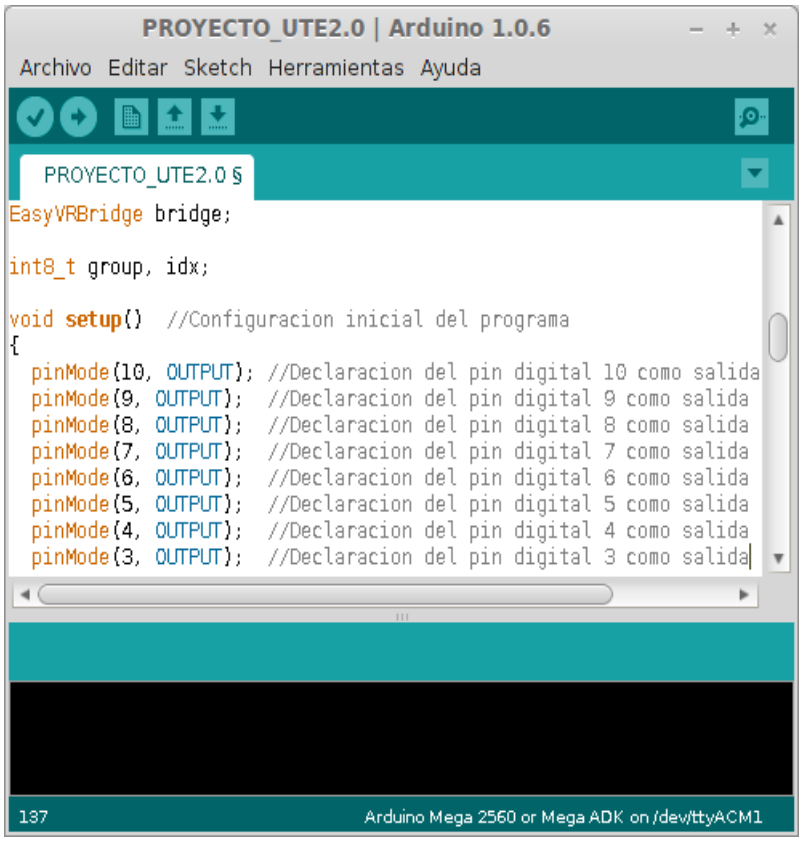


Figura 129. Declaración de los pines digitales como salidas

Como se puede observar en la figura anterior, los ocho pines digitales declarados como salidas corresponden a los puertos (E/S digitales) 10, 9, 8, 7, 6, 5, 4, y 3 del controlador Arduino, los mismos que enviarán las señales (pulsos eléctricos) hacia los relevadores para que estos finalmente controlen los actuadores (accesorios electrónicos del vehículo) así como lo indica la tabla 13.

Posteriormente se comenzará a programar cada uno de los comandos mediante la utilización de constantes de tipo booleanas dentro de la función “void action()”, la misma que se encuentra a continuación de la función “void loop” y que es donde se empezará a escribir las “acciones” (código) a realizar en caso de que un comando haya sido llamado o pronunciado.

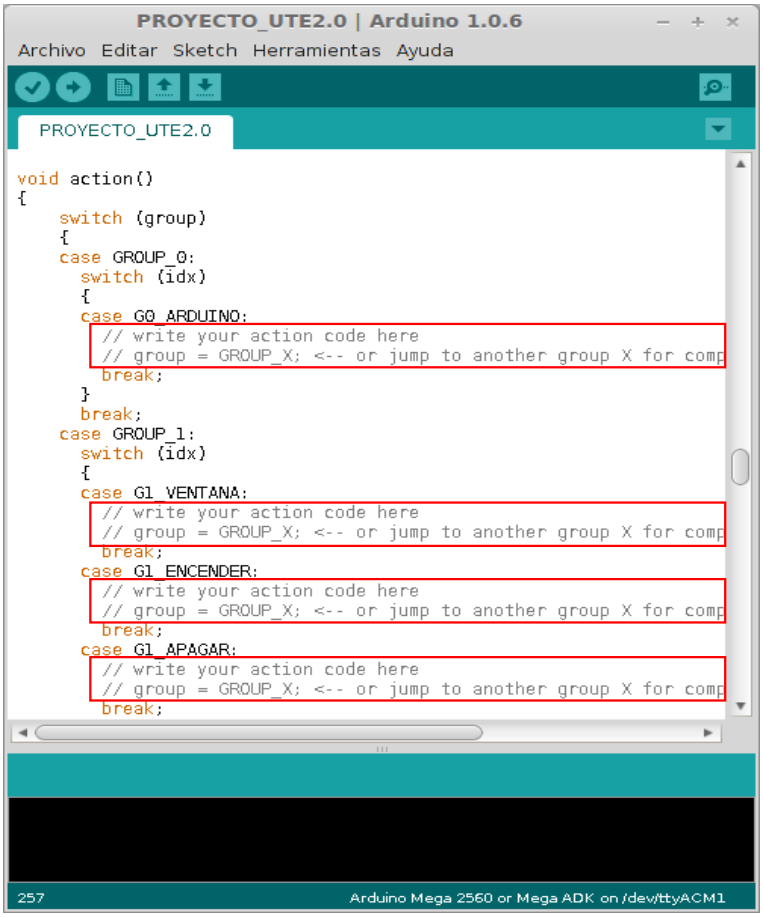


Figura 130. Acciones a realizar durante el llamado de un comando

Como se puede observar en la figura 130 todos los comandos necesitan ser programados de acuerdo a la necesidad del proyecto, por lo que se comenzará con el primer caso que corresponde al “Grupo_0” (G0) o gatillo (TRIGGER).

Sí el comando “ARDUINO” en “G0” es llamado (pronunciado), entonces se ejecutará el código “***easyvr.playSound***(SND_Hola, EasyVR::VOL_FULL);”, el mismo que tiene como función llamar a la constante “SND_Hola” definida anteriormente (véase figura 128), con la finalidad de reproducir el sonido correspondiente a dicha constante en volumen alto. A continuación el grupo actual (G0) tomará el valor del grupo 1 (***group = GROUP_1;***), para finalmente dar un salto (***break***) en el bucle, finalizando de esta manera el grupo 0 (G0) y pasando por ende al siguiente grupo (G1).

```
void action()
{
   switch (group)
   {
   case GROUP_0:
     switch (idx)
   {
     case G0_ARDUINO:
      //Reproduce el sonido SND_Hola
      easyvr.playSound(SND_Hola, EasyVR::VOL_FULL);
      group = GROUP_1;  //Salta al grupo 1 (toma el valor G1)
       break;
     }
 break;
```

Una vez en el grupo 1 (G1) las opciones a elegir son “VENTANA”, “ENCENDER”, “APAGAR”, “ABRIR”, “CERRAR” Y “ARDUINO”. De manera que en base al diagrama de flujo diseñado (*ver figura 74*), se enviará a cada uno de los comandos (en caso de ser llamados) a los grupos que los conforman.

```
   case GROUP_1:
     switch (idx)
     {
     case G1_VENTANA:
       group = GROUP_2;     //Salta al grupo 2 (toma el valor G2)
       break;
     case G1_ENCENDER:
       group = GROUP_G5;  //Salta al grupo 5 (toma el valor G5)
       break;
     case G1_APAGAR:
       group = GROUP_6;     //Salta al grupo 6 (toma el valor G6)
       break;
     case G1_ABRIR:
       group = GROUP_7;     //Salta al grupo 7 (toma el valor G7)
       break;
     case G1_CERRAR:
       group = GROUP_9;     //Salta al grupo 9 (toma el valor G9)
       break;
     case G1_ARDUINO:
       group = GROUP_11;   //Salta al grupo 11 (toma el valor G11)
       break;
     }
     break;
```

En el caso de que la opción elegida resulta ser el comando “VENTANA”, entonces se saltará al grupo 2 (G2) como se puede analizar en el código anteriormente escrito. A continuación se tendrán los comandos “IZQUIERDA”, “DERECHA” Y “ARDUINO”, los mismos que serán enviados mediante un salto de bucle (break) a sus grupos correspondientes como lo muestra la figura 74 perteneciente al diagrama de flujo.

```
case GROUP_2:
    switch (idx)
    {
    case G2_IZQUIERDA:
      group = GROUP_3;    //Salta al grupo 3 (toma el valor G3)
      break;
    case G2_DERECHA:
      group = GROUP_4;    //Salta al grupo 4 (toma el valor G4)
      break;
    case G2_ARDUINO:
      group = GROUP_11;  //Salta al grupo 11 (toma el valor G11)
      break;
    }
    break;
```

Ahora si el comando a elegir en el grupo 2 (G2) corresponde a la opción “IZQUIERDA”, entonces se direccionará al grupo 3 (G3) y se tendrá tres opciones más que son “SUBIR”, “BAJAR” Y “ARDUINO”. Sí nos fijamos en el diagrama de flujo las opciones “SUBIR” y “BAJAR” ejecutarán el código fuente correspondiente, por lo que de acuerdo a la necesidad de cada usuario se ajustará el tiempo en milisegundos en que la ventana elegida (IZQUIERDA) suba o baje de manera proporcional a dicho tiempo (relé energizado), es decir cuando los pines digitales (10 subir, 9 bajar) se encuentren en un estado alto (HIGH).

Cabe recalcar que el comando de escape “ARDUINO” utilizado en la mayoría de grupos (G1-G10), permitirá cancelar y salir de un comando así como congelar el sistema al ponerlo en modo “hibernación” por medio del grupo 11 (G11).

```
case GROUP_3:
    switch (idx)
    {
    case G3_SUBIR:
      digitalWrite(9, HIGH);   //Envía al pin digital 9 el valor HIGH (alto)
      delay(1500);             //Espera 1500 milisegundos
      digitalWrite(9, LOW);    //Envía al pin digital 9 el valor LOW (bajo)
      group = GROUP_1;     //Salta al grupo 1 (toma el valor G1)
      break;
    case G3_BAJAR:
      digitalWrite(10, HIGH);  //Envía al pin digital 10 el valor HIGH (alto)
      delay(1500);              //Espera 1500 milisegundos
      digitalWrite(10, LOW);   //Envía al pin digital 10 el valor LOW (bajo)
      group = GROUP_1;      //Salta al grupo 1 (toma el valor G1)
      break;
    case G3_ARDUINO:
      //Reproduce el sonido SND_Si
     easyvr.playSound(SND_Si, EasyVR::VOL_FULL);
      group = GROUP_11;      //Salta al grupo 11 (toma el valor G11)
      break;
    }
    break;
```

Por otra parte si la opción elegida en el grupo 2 (G2) pertenece al comando “DERECHA”, entonces los vocablos y el código escrito en el grupo 4 (G4) serán exactamente iguales a los del grupo 3 (G3); pero con la única diferencia que las señales correspondientes a este grupo (G4) saldrán hacia los pines digitales 7 (subir) y 8 (bajar), debido principalmente a que el lado elegido corresponde al derecho y no al izquierdo.

```
 case G4_SUBIR:
      digitalWrite(7, HIGH); //Envía al pin digital 7 el valor HIGH (alto)
      delay(1500);           //Espera 1500 milisegundos
      digitalWrite(7, LOW);  //Envía al pin digital 7 el valor LOW (bajo)
      group = GROUP_1;   //Salta al grupo 1
      break;
    case G4_BAJAR:
      digitalWrite(8, HIGH); //Envía al pin digital 8 el valor HIGH (alto)
      delay(1500);           //Espera 1500 milisegundos
      digitalWrite(8, LOW);  //Envía al pin digital 8 el valor LOW (bajo)
      group = GROUP_1;   //Salta al grupo 1
      break;
    case G4_ARDUINO:
      //Reproduce el sonido SND_Si
```

```
      easyvr.playSound(SND_Si, EasyVR::VOL_FULL);
       group = GROUP_11;  //Salta al grupo escape 11 (toma el valor G11)
       break;
     }
     break;
```

En caso de que exista la necesidad de “ENCENDER” o “APAGAR” la luz del habitáculo, entonces se llamará a uno de los comandos mencionados (encender o apagar) en el grupo 1 (G1) y a continuación el vocablo “LUZ” perteneciente a los grupos 5 y 6, el cual ejecutará el código de programación correspondiente, poniendo de esta manera al pin digital 6 en un estado alto (encender) o bajo (apagar) respectivamente.

```
case GROUP_5:
     switch (idx)
     {
    case G5_LUZ:
    digitalWrite(6, HIGH);    //Envía al pin digital 6 el valor alto indefinidamente
       group = GROUP_1;    //Salta al grupo 1 (toma el valor G1)
       break;
   case G5_ARDUINO:
       //Reproduce el sonido SND_Si
      easyvr.playSound(SND_Si, EasyVR::VOL_FULL);
       group = GROUP_11;  //Salta al grupo escape 11 (toma el valor G11)
       break;
     }
     break;
case GROUP_6:
     switch (idx)
     {
    case G6_LUZ:
    digitalWrite(6, LOW);  //Envía al pin digital 6 el valor bajo indefinidamente
       group = GROUP_1; //Salta al grupo 1 (toma el valor G1)
       break;
     case G6_ARDUINO:
       //Reproduce el sonido SND_Si
      easyvr.playSound(SND_Si, EasyVR::VOL_FULL);
       group = GROUP_11;  //Salta al grupo escape 11 (toma el valor G11)
       break;
     }
     break;
```

Por otra parte si la necesidad es abrir las puertas o ventanas

completamente, entonces se llamará al comando “ABRIR” y a continuación se elegirá a una de las dos opciones en el grupo 7 (G7) “VENTANA” o “PUERTAS”. Sí la opción elegida es “ventana”, entonces saltará al grupo 8 (G8); mientras que si el comando elegido corresponde a “puertas”, entonces pondrá en un estado alto al pin digital 5 por un tiempo de 500 milisegundos suficientes para abrir los seguros (puertas) por medio del relevador 6 (*véanse tabla 13 y figura 147*).

```
case GROUP_7:
    switch (idx)
    {
    case G7_VENTANA:
      group = GROUP_8;    //Salta al grupo 8 (toma el valor G8)
      break;
    case G7_PUERTAS:
      digitalWrite(5, HIGH); //Envía al pin digital 5 el valor HIGH (alto)
      delay(500);            //Espera 500 milisegundos
      digitalWrite(5, LOW);  //Envía al pin digital 5 el valor LOW (bajo)
      //Reproduce el sonido SND_Puerta_Abierta
      easyvr.playSound(SND_Puerta_Abierta, EasyVR::VOL_FULL);
      group = GROUP_1;    //Salta al grupo 1 (toma el valor G1)
      break;
    case G7_ARDUINO:
      //Reproduce el sonido SND_Si
     easyvr.playSound(SND_Si, EasyVR::VOL_FULL);
      group = GROUP_11;  //Salta al grupo 11 (toma el valor G11)
      break;
    }
    break;
```

Una vez en el grupo 8 (G8) se tendrá prácticamente dos opciones sin contar el carácter de escape “ARDUINO”, las mismas que pertenecen a los comandos “IZQUIERDA” o “DERECHA”; sí la opción elegida corresponde al vocablo “IZQUIERDA”, entonces se reproducirá el sonido deseado y a continuación se pondrá el pin digital 10 en un estado alto (HIGH) durante el tiempo que lleve abrir dicha ventana (izquierda). De igual manera sucede con el comando “DERECHA” (abrir - ventana - derecha), con la única diferencia de que la señal será enviada en este caso hacia el pin digital 8, el mismo que tiene como función cerrar el circuito en el relevador 3 así como lo indican la tabla 13 y la figura 144.

```
case GROUP_8:
  switch (idx)
  {
  case G8_IZQUIERDA:
    //Reproduce el sonido SND_Como_Ordene_Señor
    easyvr.playSound(SND_Como_Ordene_Senor, EasyVR::VOL_FULL);
    digitalWrite(10, HIGH);  //Envía al pin digital 10 el valor HIGH (alto)
    delay(5000);             //Espera 5000 milisegundos
    digitalWrite(10, LOW);   //Envía al pin digital 10 el valor LOW (bajo)
    group = GROUP_1;       //Salta al grupo 1 (toma el valor G1)
    break;
  case G8_DERECHA:
    //Reproduce el sonido SND_Como_Ordene_Señor
    easyvr.playSound(SND_Como_Ordene_Senor, EasyVR::VOL_FULL);
    digitalWrite(8, HIGH);  //Envía al pin digital 8 el valor HIGH (alto)
    delay(5000);            //Espera 5000 milisegundos
    digitalWrite(8, LOW);   //Envía al pin digital 8 el valor LOW (bajo)
    group = GROUP_1;     //Salta al grupo 1 (toma el valor G1)
    break;
  case G8_ARDUINO:
    //Reproduce el sonido SND_Si
   easyvr.playSound(SND_Si, EasyVR::VOL_FULL);
    group = GROUP_11;   //Salta al grupo 11 (Escape)
    break;
  }
  break;
```

Ahora si la necesidad es cerrar las puertas o ventanas completamente, entonces se llamará al comando “CERRAR” en el grupo 1 (G1) y a continuación se elegirá a una de las dos opciones en el grupo 9 (G9) “PUERTAS” o “VENTANA”. Sí el comando elegido corresponde al vocablo “PUERTAS”, entonces se pondrá en un estado alto al pin digital 4 por un tiempo de 500 milisegundos con el fin de cerrar los seguros de las puertas mediante el relevador 7 (véase figura 147); por otra parte sí el comando a elegir corresponde a la palabra “VENTANA”, entonces saltará al grupo 10 (G10) con el fin de especificar el lado a cerrar deseado.

```
case GROUP_9:
  switch (idx)
  {
  case G9_PUERTAS:
    digitalWrite(4, HIGH);  //Envía al pin digital 4 el valor HIGH (alto)
```

```
  delay(500);      //Espera 500 milisegundos
  digitalWrite(4, LOW);  //Envía al pin digital 4 el valor LOW (bajo)
  //Reproduce el sonido SND_Puerta_Cerrada
 easyvr.playSound(SND_Puerta_Cerrada, EasyVR::VOL_FULL);
 group = GROUP_1;     //Salta al grupo 1 (toma el valor G1)
  break;
case G9_VENTANA:
 group = GROUP_10;   //Salta al grupo 10 (toma el valor G10)
  break;
case G9_ARDUINO:
  //Reproduce el sonido SND_Si
 easyvr.playSound(SND_Si, EasyVR::VOL_FULL);
  group = GROUP_11;  //Salta al grupo 11 (toma el valor G11)
  break;
}
break;
```

Una vez en el grupo 10 (G10) se tendrán las opciones “IZQUIERDA” o “DERECHA”; sí la opción elegida corresponde al vocablo “IZQUIERDA”, entonces se reproducirá el sonido deseado y a continuación se pondrá el pin digital 9 en un estado alto (HIGH) durante el tiempo que lleve cerrar la ventana (izquierda). De igual manera sucede con el comando “DERECHA” (cerrar - ventana - derecha), con la única diferencia de que la señal será enviada en este caso hacia el pin digital 7 al tratarse del lado opuesto.

```
case GROUP_10:
  switch (idx)
  {
  case G10_IZQUIERDA:
  //Reproduce el sonido SND_Si_Señor
  easyvr.playSound(SND_Si_Senor, EasyVR::VOL_FULL);
    digitalWrite(9, HIGH); //Envía al pin digital 9 el valor HIGH (alto)
    delay(5000);           //Espera 5000 milisegundos
    digitalWrite(9, LOW);  //Envía al pin digital 9 el valor LOW (bajo)
    group = GROUP_1;   //Salta al grupo 1 (toma el valor G1)
    break;
  case G10_DERECHA:
  //Reproduce el sonido SND_Si_Señor
  easyvr.playSound(SND_Si_Senor, EasyVR::VOL_FULL);
    digitalWrite(7, HIGH); //Envía al pin digital 7 el valor HIGH (alto)
    delay(5000);           //Espera 5000 milisegundos
    digitalWrite(7, LOW);  //Envía al pin digital 7 el valor LOW (bajo)
    group = GROUP_1;   //Salta al grupo 1 (toma el valor G1)
    break;
```

```
    case G10_ARDUINO:
      //Reproduce el sonido SND_Si
     easyvr.playSound(SND_Si, EasyVR::VOL_FULL);
      group = GROUP_11;  //Salta al grupo 11 (toma el valor G11)
      break;
    }
    break;
```

Finalmente se programará los comandos de escape en el grupo 11 (G11), los mismos que serán llamados mediante la palabra clave “ARDUINO” en cada uno de los grupos (G1-G10) anteriormente citados. Estos comandos corresponden a los vocablos “SALIR” e “HIBERNAR”; sí es el caso en que se requiere cancelar un comando ya sea por un error humano, entonces se llamará a la palabra clave “ARDUINO” y a continuación el comando “SALIR”, el mismo que tiene como función saltar al grupo cabecera (G1); mientras que sí la necesidad recae en congelar al sistema (inhabilitarlo), entonces se pronunciará la palabra clave (ARDUINO) seguido del comando “HIBERNAR”, el mismo que tiene como función tomar el valor G0 (TRIGGER) en el grupo 11.

```
case GROUP_11:
    switch (idx)
    {
    case G11_SALIR:
      //Reproduce el sonido SND_Comando_Cancelado
     easyvr.playSound(SND_Comando_Cancelado, EasyVR::VOL_FULL);
      group = GROUP_1;    //Salta al grupo 1 (toma el valor G1)
      break;
    case G11_HIBERNAR:
      //Reproduce el sonido SND_Modo_Hibernacion
     easyvr.playSound(SND_Modo_Hibernacion, EasyVR::VOL_FULL);
     group = GROUP_0;     //Salta al grupo 0 (toma el valor G0)
      break;
    }
    break;
  }
}

//Fin del Programa
```

3.2.3.8. Montaje del Sistema

Una vez terminada la programación en el IDE de Arduino, se procede a verificar el código escrito mediante la opción “compilar” (véase figura 41) representada por una señal de cotejo (✓). Sí el código fuente fue correctamente escrito, entonces se mostrará un informe acerca del tamaño en bytes a ocupar por el Sketch; mientras que si el código tiene alguna falencia, entonces se emitirá un error de programación el mismo que deberá ser corregido (depurado).

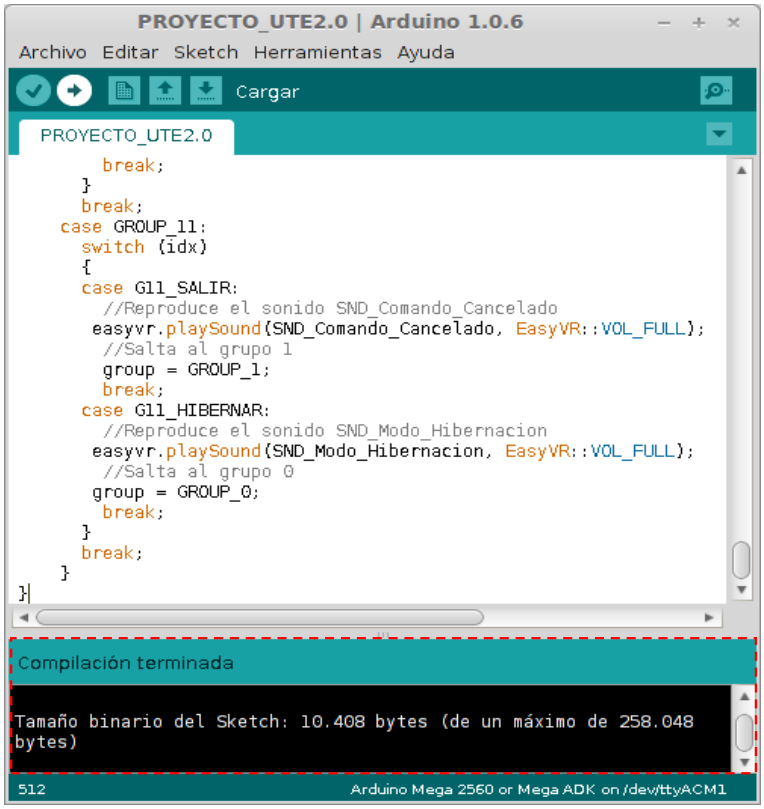


Figura 131. Compilación del programa terminada

A continuación se cargará el código fuente a la memoria interna del controlador Arduino (jumper en la posición “**SW**”), con la finalidad de comprobar el funcionamiento del sistema previo a su implementación. De manera que se realizará el montaje correspondiente con el objetivo de simular el correcto control de los diferentes actuadores, mediante los comandos de voz entrenados y programados con anterioridad.

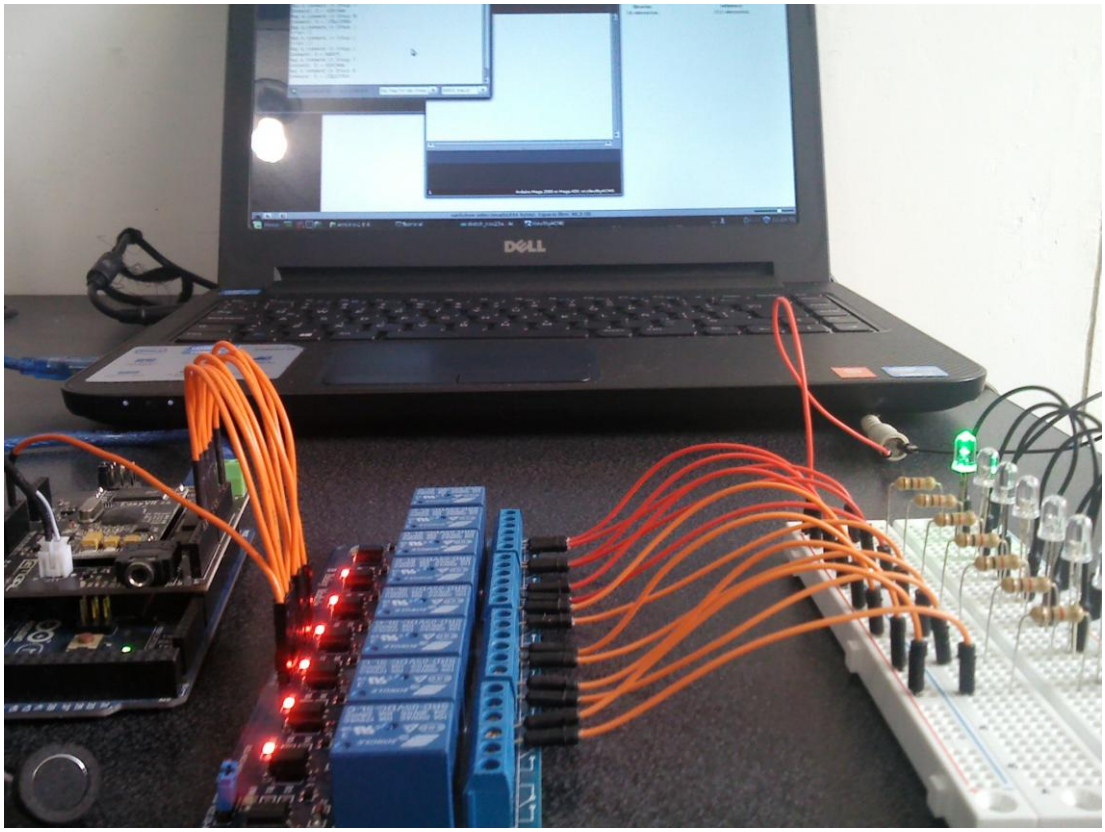


Figura 132. Montaje del sistema

Una vez realizado el montaje correspondiente al diagrama esquemático representado en las figuras 133 y 134; así como haber cargado el código fuente al microcontrolador, se procederá a pronunciar la palabra clave “ARDUINO” en el grupo 0 con la finalidad de arrancar el sistema (gatillo) al entrar en espera de un comando en el grupo cabecera G1. A continuación se llamará a cada uno de los comandos especificados en el diagrama de flujo, con la finalidad de corroborar la fiabilidad del sistema y su programación.

Además es importante indicar que para monitorizar a cada uno de los comandos llamados en sus respectivos grupos, basta con abrir el IDE de Arduino y a continuación en la pestaña “Herramientas” se elegirá la opción “Monitor Serial”, el mismo que abrirá una ventana en la cual se podrá monitorear el estado del sistema. Es decir el grupo actual en el que se encuentra, el comando escuchado, tiempo de espera terminado (time out), detección de errores, detección de la tarjeta Shield, etc.

Cuando el código de programación correspondiente a una acción en particular es ejecutado, entonces se enviará una señal de tipo digital hacia el pin perteneciente a dicha acción (véase diseño del código fuente), el mismo que pondrá en un estado alto (HIGH) o bajo (LOW) al relevador. Cabe indicar que el módulo de relés (8 canales) utilizado funciona con una alimentación de 5 voltios suministrados por el controlador; mientras que las entradas In1, In2, In3, In4, In5, In6, In7 e In8 de dicho módulo se encuentran conectadas a los pines digitales D10, D9, D8, D7, D6, D5, D4 y D3 respectivamente en base al código de programación.

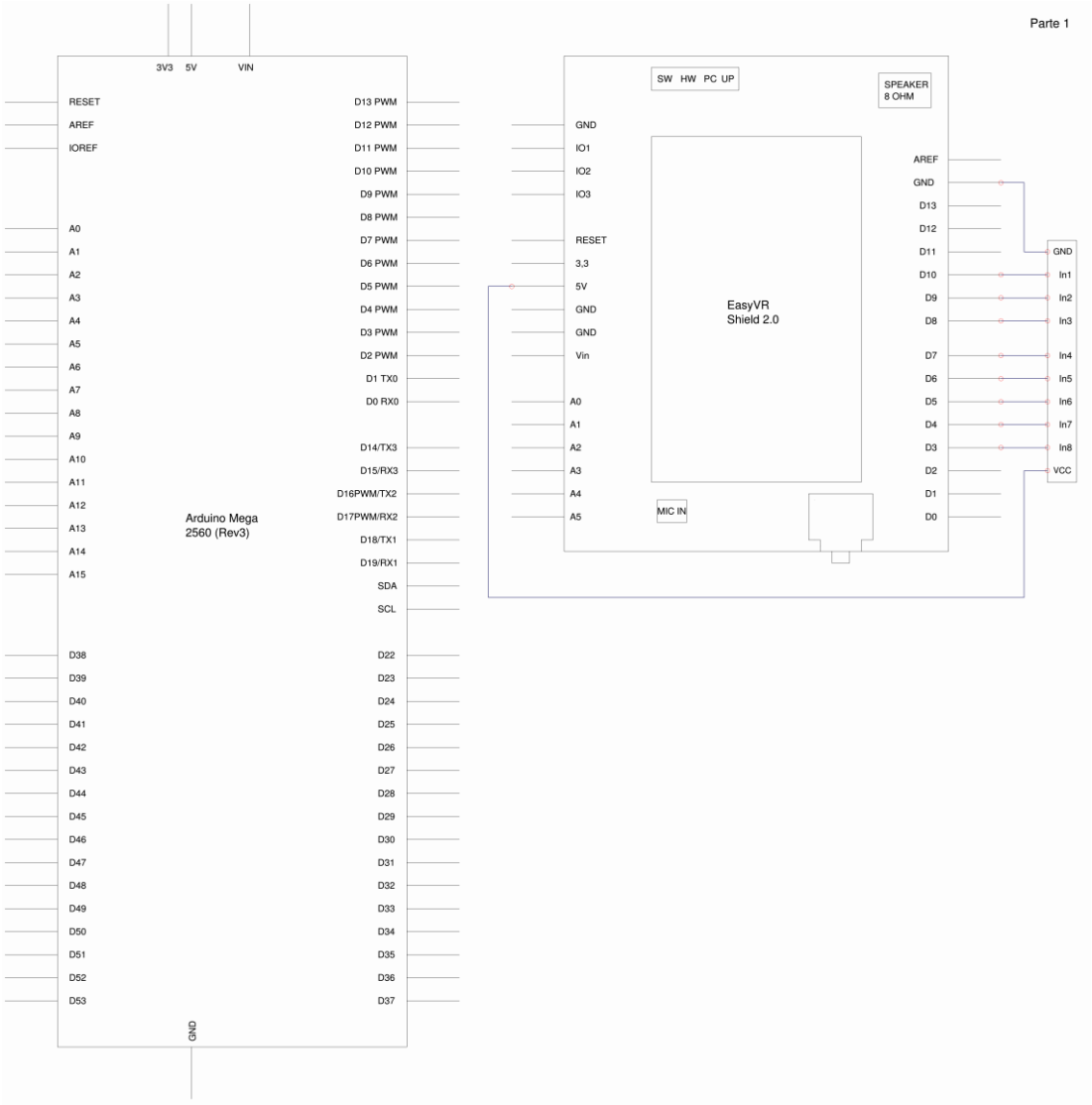


Figura 133. Diagrama esquemático del sistema (parte 1)

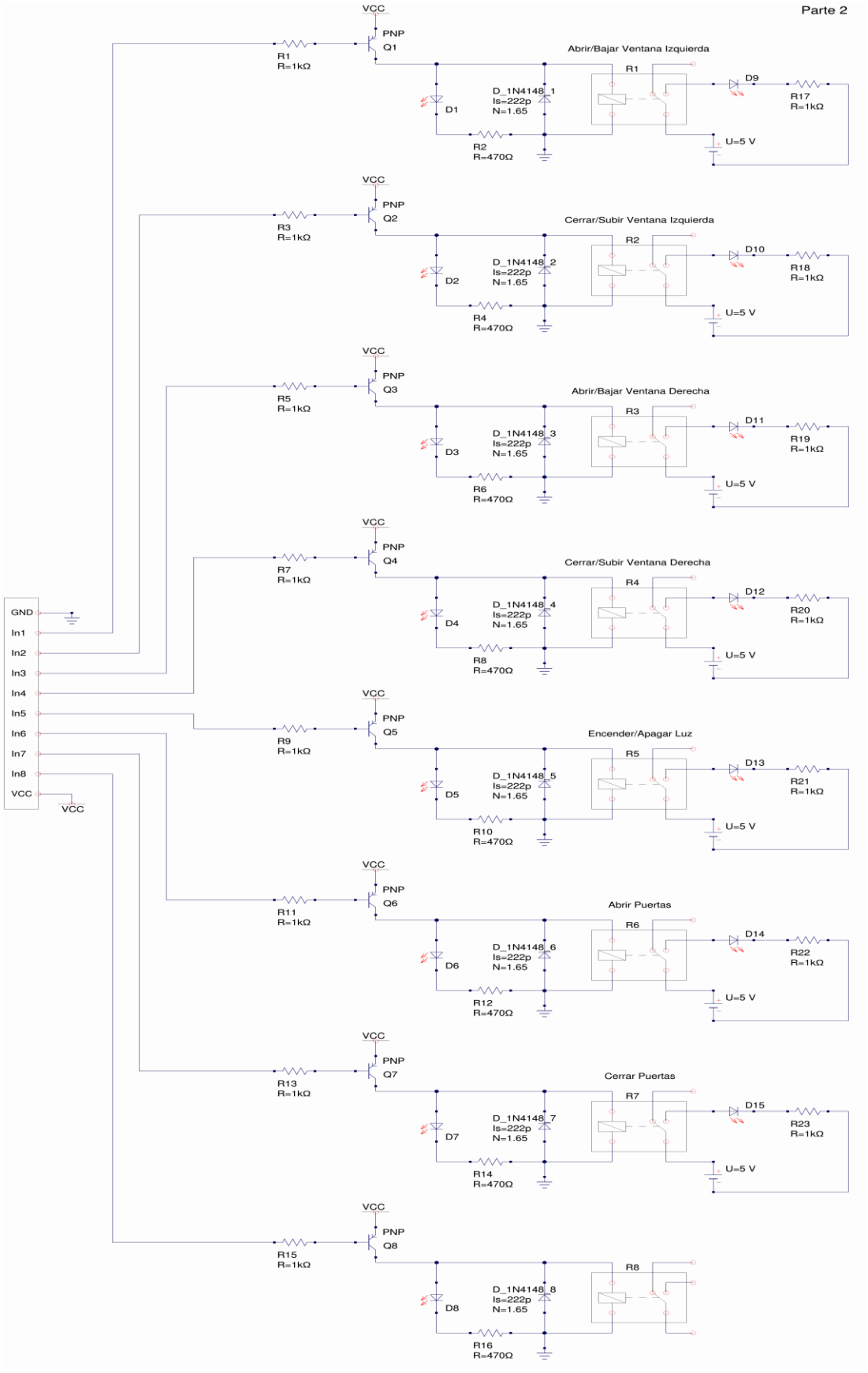


Figura 134. Diagrama esquemático del sistema (parte 2)

Como se puede analizar en el diagrama esquemático de la figura 134 perteneciente al módulo de relés y sus actuadores (LEDs); las señales (digitales) enviadas por el controlador hacia los puertos o entradas (In) del módulo son conducidas a través del circuito (figura 134), donde previo a una resistencia (1kΩ), la señal llega a la base del transistor "Q" el mismo que tiene como función abrir (LOW) o cerrar (HIGH) el circuito entre el emisor y el colector. Permitiendo de esta manera el paso de la corriente suministrada por el mismo controlador (VCC) hacia la bobina del relevador, la cual a su vez se encuentra conectada a la terminal tierra (masa), cerrando así el circuito y por ende activando al relevador debido al campo magnético generado por la bobina.

Una vez cerrado el terminal normalmente abierto "NA" del relevador (relé activado), el actuador (LED) correspondiente a dicho relé es accionando por medio de una fuente externa de 5 voltios, la misma que vendría a ser la simulación de la batería del automóvil. Cabe indicar que el relevador 8 perteneciente al pin digital 3 no tiene ninguna función ya que el proyecto únicamente necesitó de 7 salidas digitales (véase tabla 13).

3.3. IMPLEMENTACIÓN DEL SISTEMA

Con el fin de implementar el sistema, es necesario tener instalado previamente los accesorios electrónicos a ser controlados; por lo que se comenzó incorporando los elevadores de vidrio de marca "Nemesis" (puertas delanteras) así como los seguros eléctricos de las puertas. Cabe indicar que el vehículo utilizado en este proyecto corresponde al chevrolet corsa 1.8 de tipo hatchback.

A continuación se retira el tapizado de las puertas, con la finalidad de colocar los actuadores eléctricos con sus respectivas líneas (cables), las mismas que permitirán enviar las señales o pulsos eléctricos provenientes de la fuente (batería) por medio del controlador Arduino, el mismo que tiene como función activar y desactivar los relevadores del módulo gracias a su

programación.

Figura 135. Implementación de seguros y elevalunas izquierdo

Como se puede observar en la figura superior el elevador de vidrios constituido por un motor de corriente continua y un sistema de engranajes enganchados a la manija, es fijado por medio de remaches en una posición adecuada, es decir evitando que el cable que transmite la energía mecánica proveniente del motor eléctrico, se enrede durante su funcionamiento. Una vez fijado el sistema se procede a cablear la conexión para lo cual se utilizó un cable de tipo TFF 18 AWG, el mismo que se lo puede observar en la figura anterior de color café perteneciente al elevador de vidrio del lado izquierdo; mientras que el cable de color blanco es utilizado para los seguros de las puertas. Es importante indicar que tanto el funcionamiento de los motores eléctricos de las puertas como el de las ventanas son exactamente iguales, es decir que si enviamos de cierta manera la polaridad (positivo – negativo) por los conductores (dos cables), el motor girará en un sentido; mientras que si invertimos la polaridad en sentido contrario al anterior (negativo – positivo) por dichas líneas, el motor también cambiará su sentido de giro.

Figura 136. Implementación de seguros y elevalunas derecho

De igual manera se procede a cablear e instalar los motores DC (puerta y ventana) en el lado derecho del conductor. Teniendo en cuenta que el cableado pertenecientes a los elevadores de vidrio (cables cafés), son enviados hacia el tablero central del vehículo donde se realizarán las respectivas conexiones (véase figura 142); mientras que el cableado concerniente a los seguros eléctricos de las puertas (cables blancos) se encuentran ubicados por debajo del panel de instrumentos (tablero) así como lo indica la figura 145.

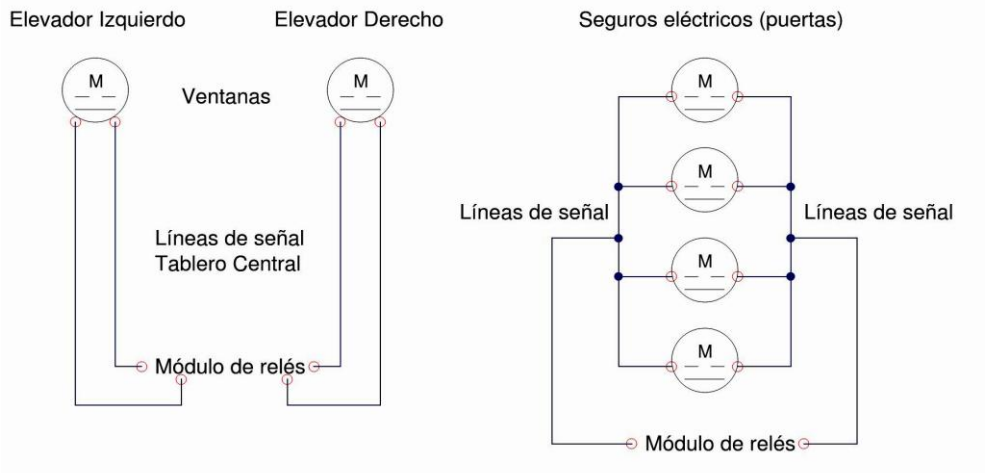


Figura 137. Esquema del sistema de puertas y ventanas (cableado)

Por otra parte se ha realizado la conexión correspondiente a la fuente de alimentación, la misma que permitirá el accionamiento de los actuadores implementados en el vehículo. Cabe indicar que la línea positiva conectada directamente a la batería posee un fusible, el cual tiene como función cortar la fuente de alimentación en caso de que exista algún tipo de sobrecarga o cortocircuito que pueda poner en peligro la integridad de los elementos en cuestión.

Figura 138. Línea positiva de la fuente de alimentación

Cuando el switch de encendido es activado (puesto en contacto), el relevador principal (GT3501C) situado por debajo de la consola central del automóvil (véase figura 140) es accionado, permitiendo de esta manera el paso de la corriente hacia el módulo de relés, el mismo que controlará una vez más el flujo de corriente (alimentación) dirigida hacia los diferentes accesorios electrónicos del vehículo instalados anteriormente. Cabe resaltar que dicho módulo es controlado por la tarjeta central Arduino, la cual ejecutará el código de programación correspondiente al comando escuchado por medio de la interfaz EasyVR; además es importante aclarar que la masa o tierra utilizada en el circuito fue obtenida de la estructura metálica del vehículo.

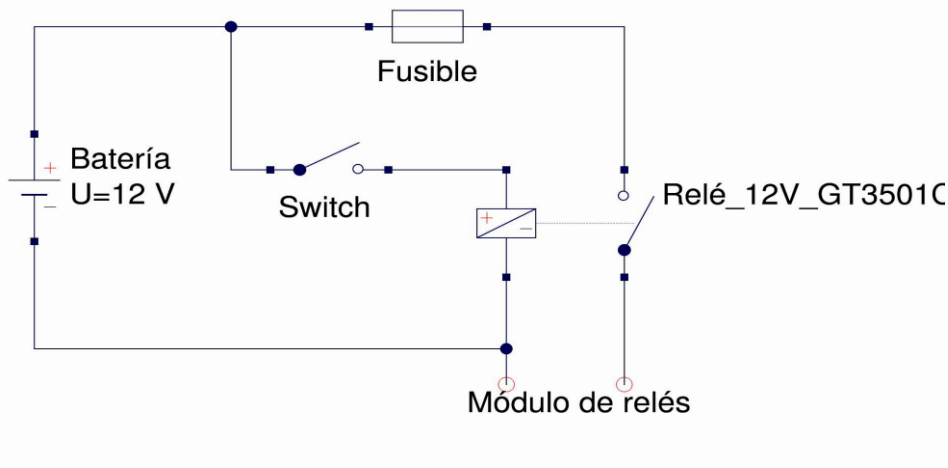


Figura 139. Diseño esquemático de la fuente de alimentación

Como se puede observar el relevador principal es activado mediante el switch de encendido, permitiendo de esta manera el paso del polo positivo (+) hacia el módulo de relés; mientras que el polo negativo (-) conectado a masa (chasis del vehículo) también es compartido con dicho módulo.

Figura 140. Relevador principal de la fuente de alimentación

A continuación se realizan las conexiones pertenecientes a las líneas de alimentación por debajo de la consola central del vehículo, para lo cual se utilizó conectores (macho - hembra) con la finalidad de facilitar el trabajo al momento de cambiar el modo de empleo (manual) de los elevalunas.

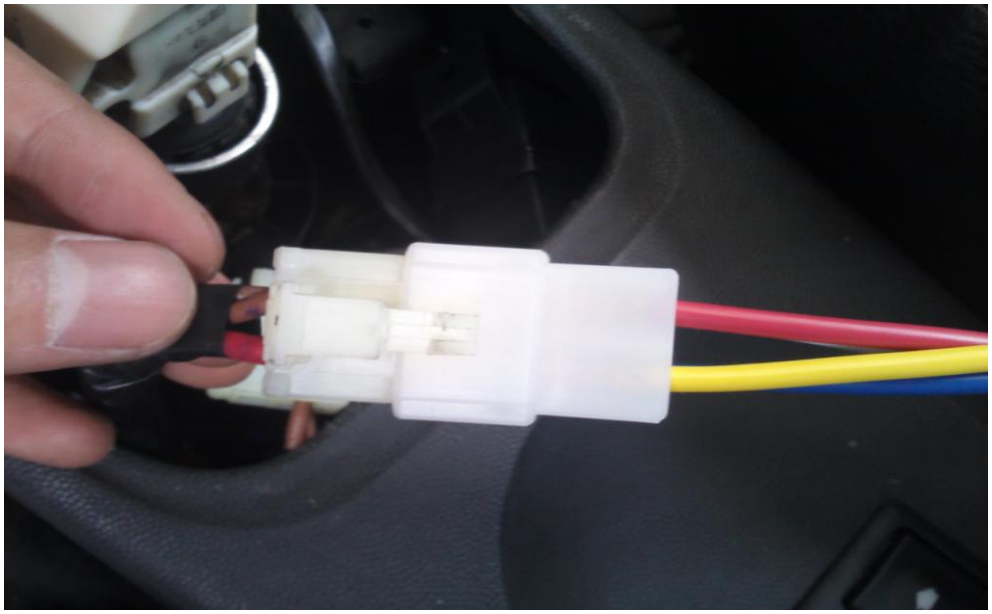


Figura 141. Líneas de alimentación y señal del elevalunas izquierdo

Como se puede observar en la figura superior, se dispone de cuatro cables un positivo (rojo), un negativo (negro) y dos cables de señal (cafés). Los cables rojo y negro pertenecen a la alimentación general del sistema (véase figura 139); mientras que los cables cafés corresponden a las líneas del elevador izquierdo (ventana).

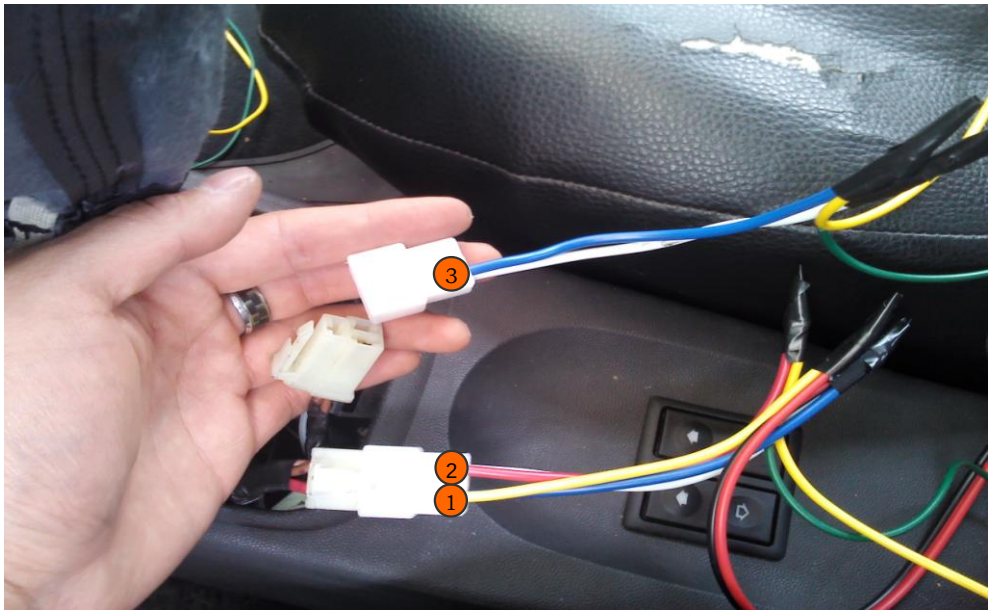


Figura 142. Líneas del elevador de vidrios derecho

De igual manera se procede a conectar las líneas pertenecientes al elevador de cristales del lado derecho del conductor así como lo muestra la figura 142; las mismas (líneas cafés) que posteriormente tomarán los colores amarillo y verde, debido a que las polaridades en este caso son invertidas cuando de abrir o cerrar la ventanilla se trata, por lo que no es necesario colocar un color en específico.

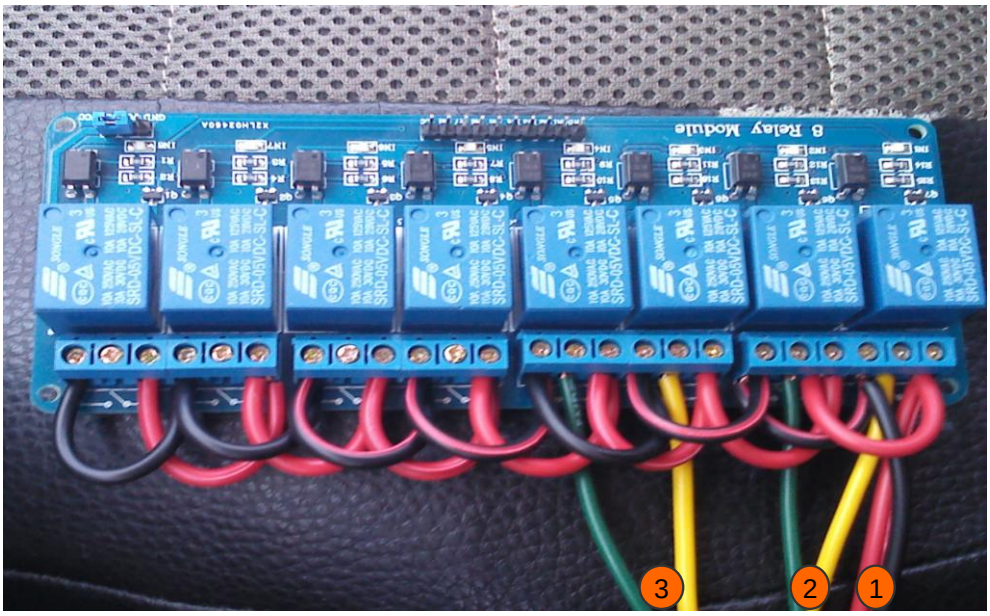


Figura 143. Líneas de alimentación y señal de los elevalunas

A continuación se conectan los cables correspondientes a las figuras 141 y 142, las mismas que pertenecen a la alimentación del sistema (1) y a las señales de los elevadores de vidrios izquierdo (2) y derecho (3). Por otra parte es posible observar que para subir o bajar la ventana es necesario utilizar 2 actuadores (relés) del módulo, debido a que para cambiar el sentido de giro del motor DC (elevador) es indispensable cambiar su polaridad. De manera que las señales correspondientes al elevador de vidrios izquierdo (2) serán conectadas a los relevadores R1 y R2 del módulo; mientras que las señales del lado derecho (3) son conectadas a continuación, es decir en los actuadores R3 y R4 así como lo indican las figuras 143 y 144. Cabe recalcar que dichos relevadores (módulo) comparten la misma masa (tierra), así como su alimentación de 12 voltios proveniente de la batería.

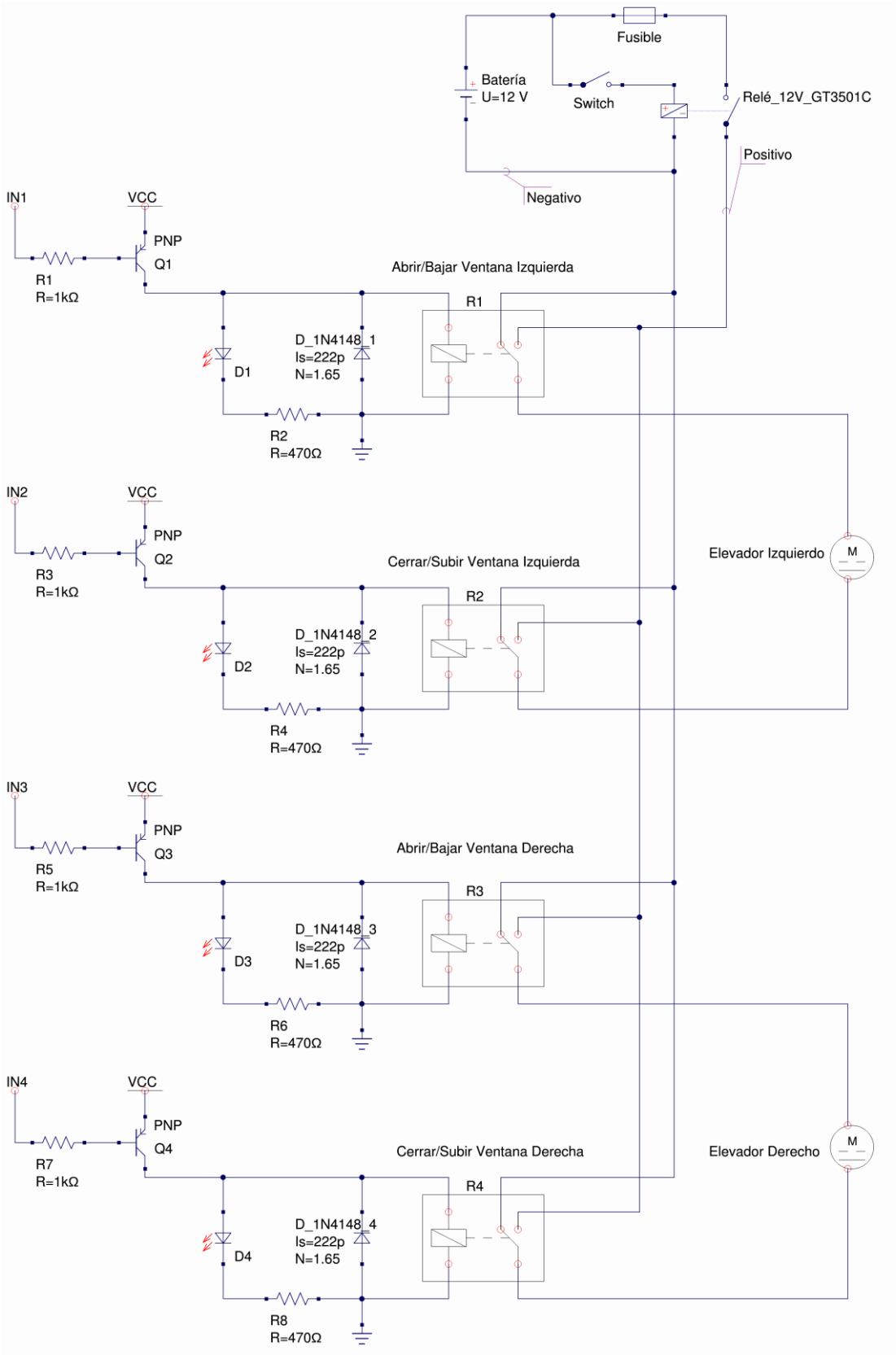


Figura 144. Esquema eléctrico de los elevadores de vidrio (D/I)

Posteriormente se procede a conectar las señales correspondientes a los seguros eléctricos de las puertas, que en este caso se encuentran por debajo del tablero de instrumentos (cables blancos). Además es importante indicar que dichos seguros han sido previamente comprobados (polaridad) durante su instalación, con la finalidad de que todas las puertas se abran o cierren a la vez.

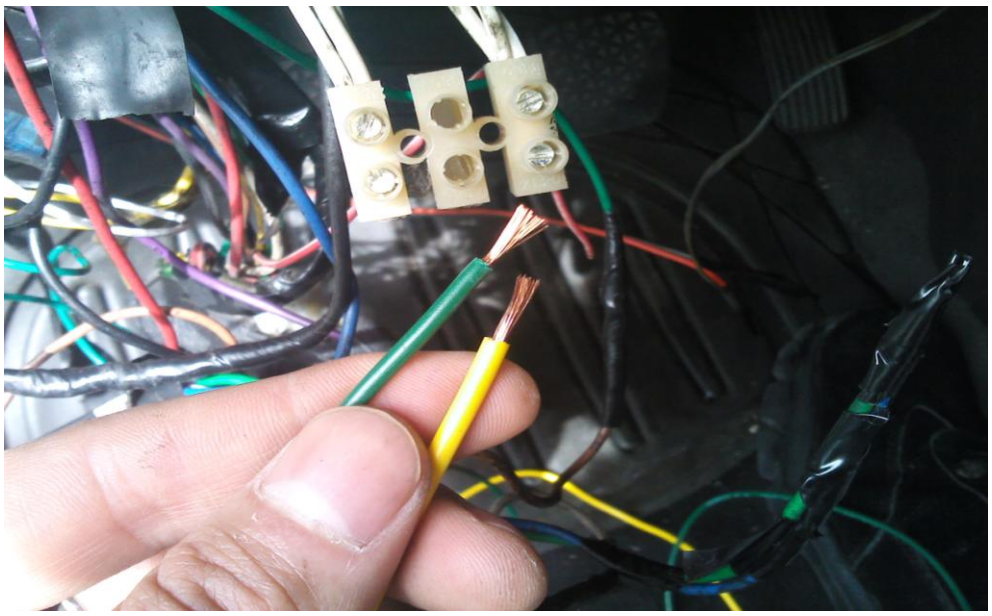


Figura 145. Líneas de alimentación de los seguros eléctricos (puertas)

Utilizando dos cables (TFF 18 AWG) de aproximadamente 2 metros de longitud se conectan las señales correspondientes a los seguros de las puertas; teniendo en cuenta de que el principio de funcionamiento de dichos seguros eléctricos es exactamente el mismo al de los elevadores de vidrio, ya que es necesario invertir la polaridad con la finalidad de abrir o cerrar las puertas (seguros) al cambiar el sentido de giro de los motores DC.

Una vez conectadas las líneas por debajo de la consola principal (tablero), se procede acoplar los terminales en el extremo opuesto de los cables así como lo muestran las figuras 146 y 147, donde utilizando los relevadores R6 y R7 del módulo, se tendrá la capacidad de controlar la polaridad enviada hacia los actuadores de las puertas.

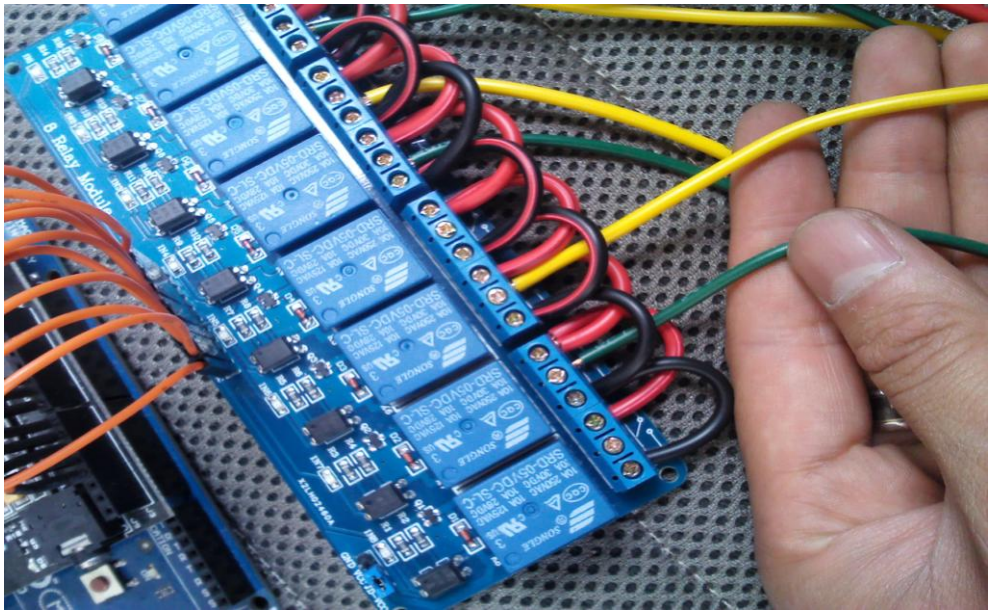


Figura 146. Relevadores pertenecientes a los seguros eléctricos (puertas)

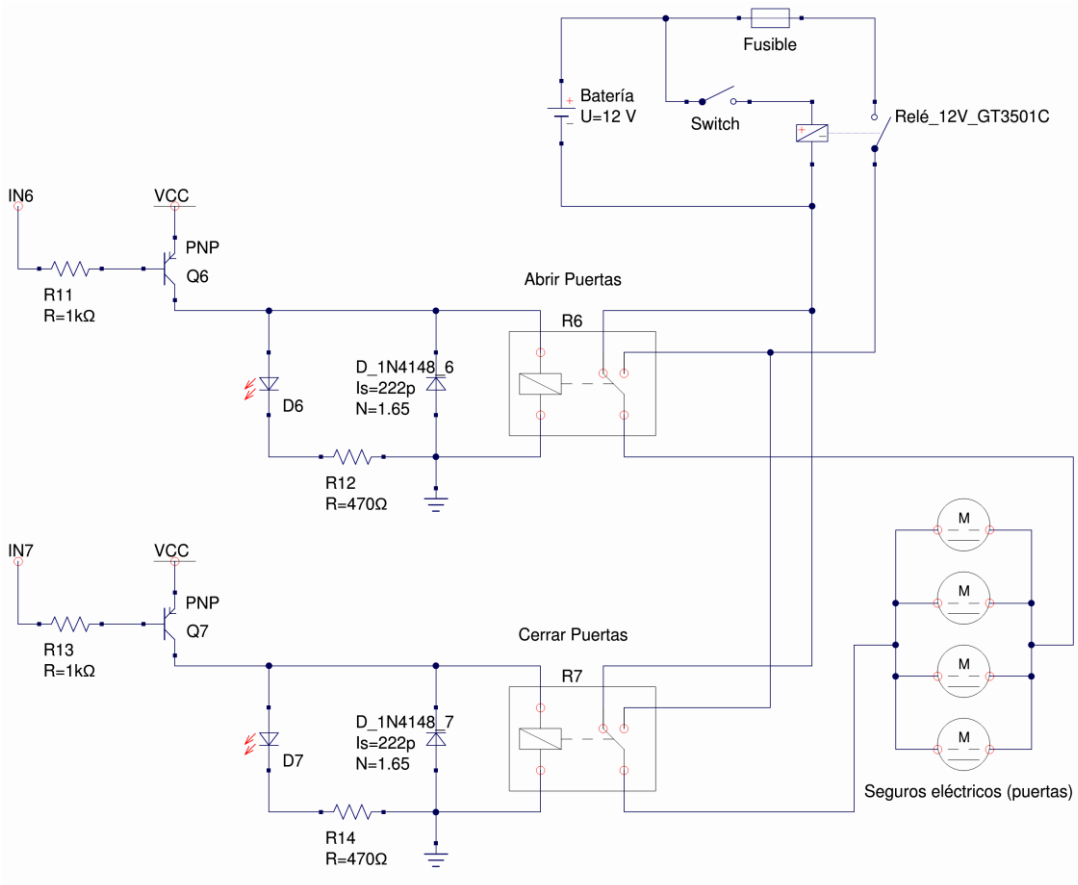


Figura 147. Diagrama esquemático de los seguros eléctricos (puertas)

A continuación se realizan las conexiones pertenecientes a los pines digitales del controlador y el módulo, para lo cual es necesario recordar la tabla 13, la misma que se utilizó a la hora de diseñar el código fuente.

Tabla 13. Acciones a realizar por los pines y relevadores correspondientes

Controlador	Módulo	Acción
Pin digital 10	Relevador 1	Abrir/Bajar Ventana Izquierda
Pin digital 9	Relevador 2	Cerrar/Subir Ventana Izquierda
Pin digital 8	Relevador 3	Abrir/Bajar Ventana Derecha
Pin digital 7	Relevador 4	Cerrar/Subir Ventana Derecha
Pin digital 6	Relevador 5	Encender/Apagar Luz
Pin digital 5	Relevador 6	Abrir Puertas
Pin digital 4	Relevador 7	Cerrar Puertas
Pin digital 3	Relevador 8	-

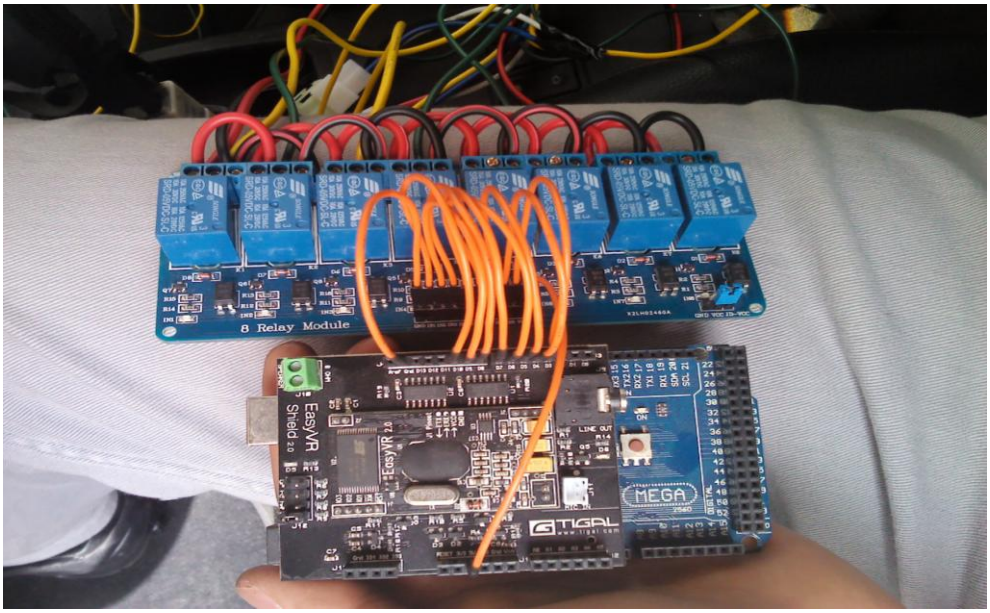


Figura 148. Conexión de los pines digitales del controlador y el módulo

Es necesario aclarar que el relevador 8 se lo utilizó únicamente como el negativo del relevador 5 concerniente a la acción de encender y apagar la luz del habitáculo como se lo detalla a continuación.

Una vez realizadas las conexiones eléctricas correspondientes a los elevadores de vidrio y seguros de puertas, se procede a conectar el último actuador que permitirá encender y apagar la luz de salón por medio de la voz; para lo cual se retiran los terminales positivo y negativo de la bombilla de luz, con la finalidad de conectar las señales provenientes del módulo.

Figura 149. Desconexión de los terminales pertenecientes a la bombilla

Con un par de cables (rojo y negro) de aproximadamente 3 metros de longitud se conectan dichos terminales.

Figura 150. Conexión de las líneas de alimentación para la luz de salón

A continuación se ajustan las líneas de alimentación de la bombilla en el módulo, para lo cual se utiliza el relevador 8 (R8) como masa del circuito ya que durante la operación del módulo todas las salidas son colocados en una posición normalmente cerrada correspondiente al negativo.

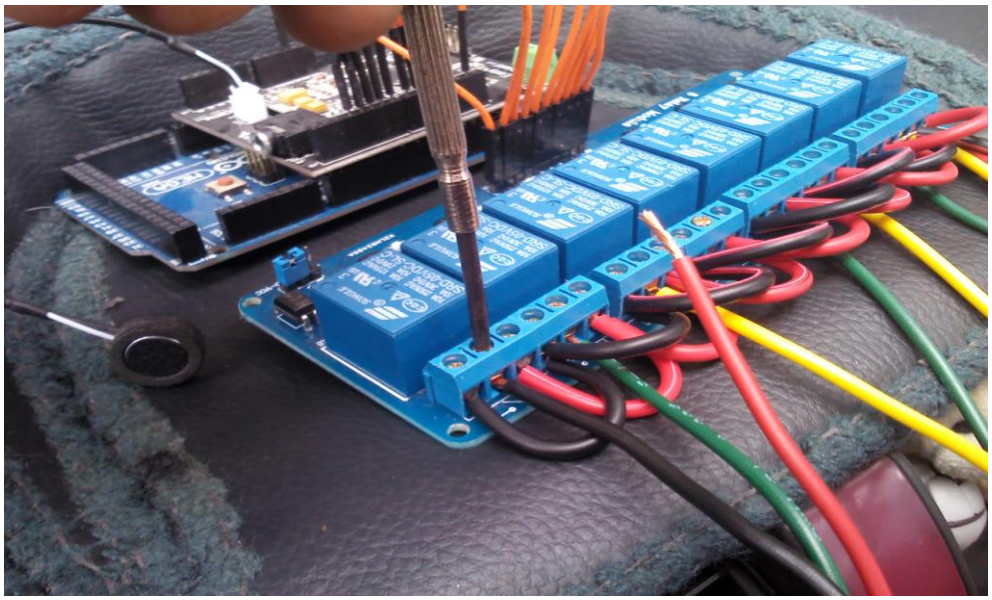


Figura 151. Relevador concerniente a la señal negativa de la bombilla de luz

Finalmente se conecta el positivo de la bombilla en el relevador 5 (R5) controlado por la salida digital 6 (pin) de Arduino acorde a la tabla 13.

Figura 152. Relevador concerniente a la señal positiva de la bombilla de luz

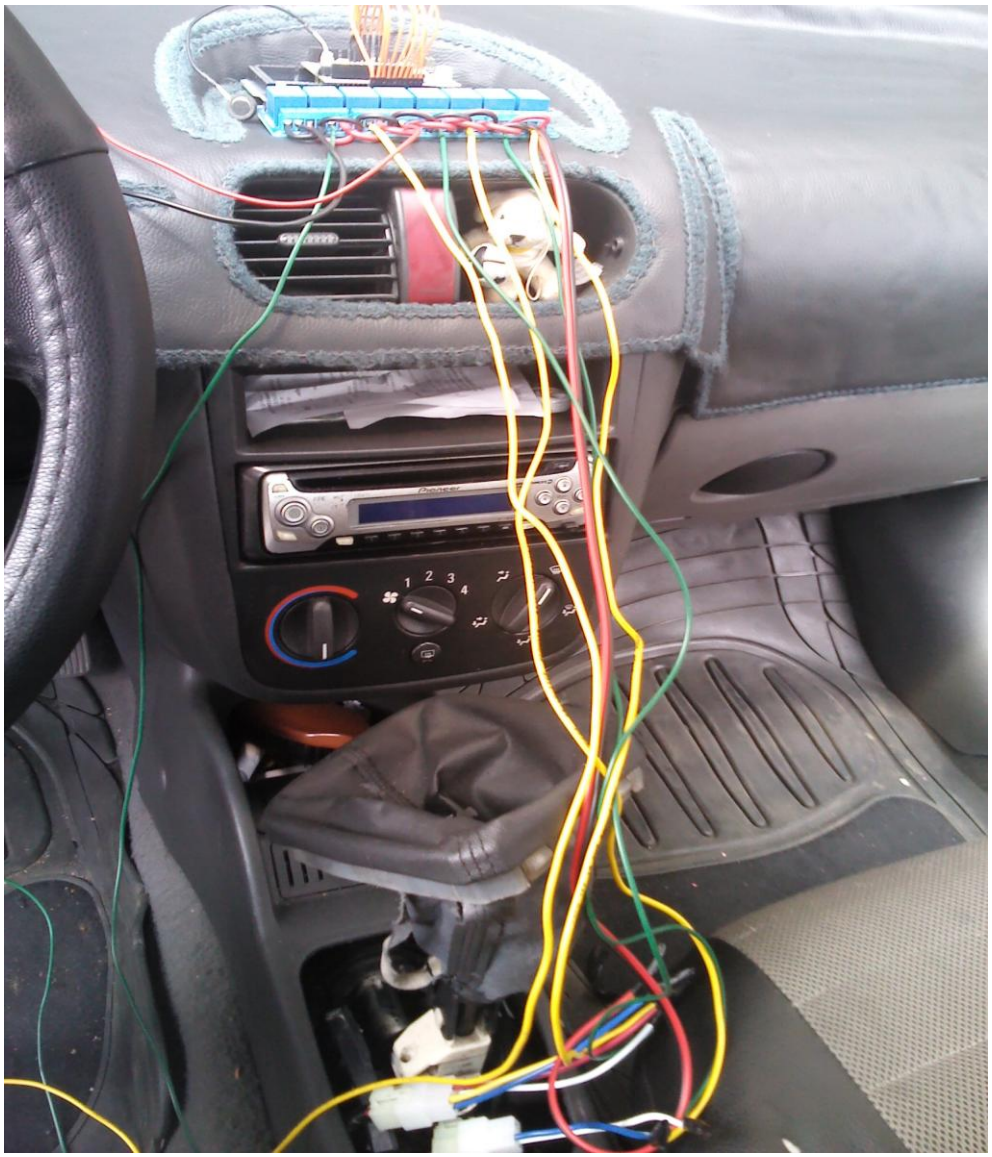


Figura 153. Sistema de reconocimiento de voz implementado

Ahora el sistema se encuentra implementado y listo para funcionar, de manera que se procede alimentar la placa Arduino por medio del cable USB al ordenador o simplemente mediante un adaptador USB de 5V conectado a la toma del mechero (vehículo) será suficiente para alimentar a la placa (Arduino + EasyVR) y por ende al módulo. En este caso se utilizó el ordenador debido a que durante la grabación de la implementación del proyecto, fue necesario abrir el monitor serie del compilador IDE de Arduino con la finalidad de poder analizar y monitorizar el código de programación ejecutado durante el llamado de un comando.

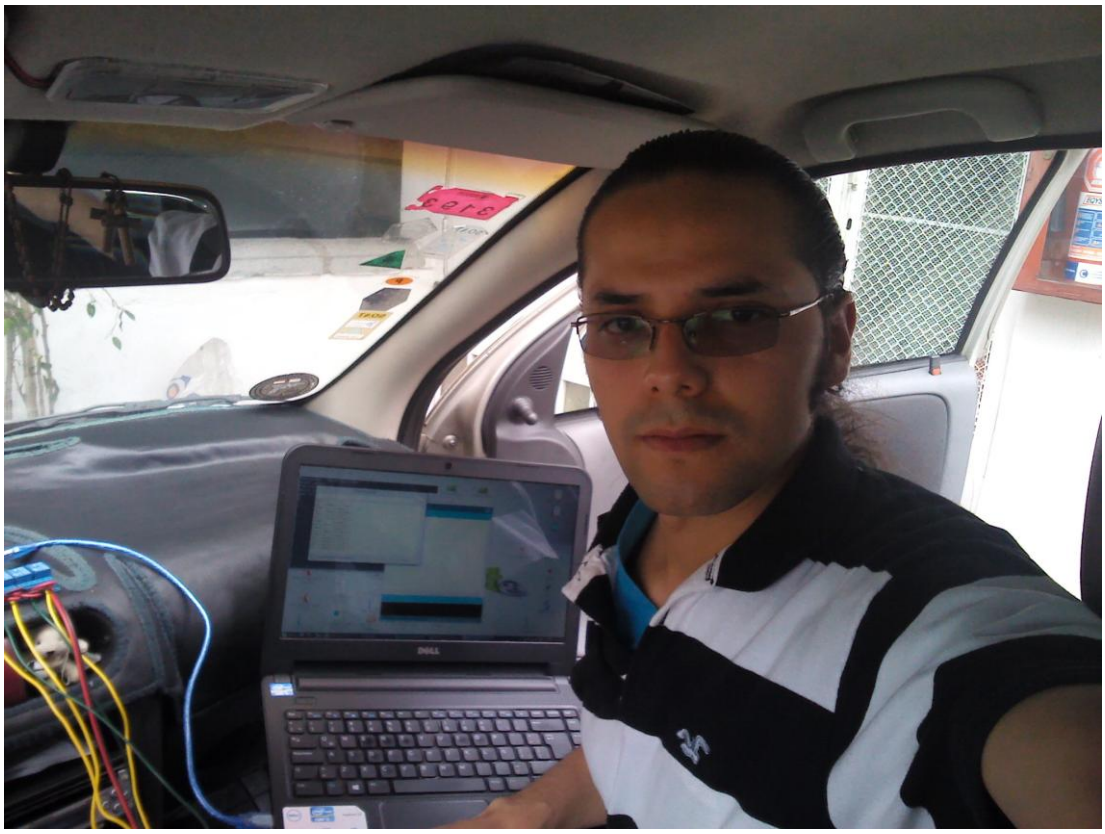


Figura 154. Análisis y monitoreo del sistema implementado

Una vez alimentado el sistema de reconocimiento de voz (cable USB) y de tener la certeza de que las conexiones realizadas anteriormente se encuentran acorde a la lógica utilizada durante el diseño (diagrama de flujo, código fuente, diagrama esquemático del sistema, etc), se procede a colocar el switch de encendido en la posición de contacto, teniendo en cuenta de que el código fuente diseñado debe encontrase cargado en la memoria interna del controlador Arduino, lo cual ya se lo realizó durante el montaje del sistema (véase página 135).

Cuando la llave de encendido es colocada en la posición de contacto, el relevador principal permite el paso del positivo de la batería hacia el módulo; pero como los contactos (positivos) de dicho módulo se encuentran normalmente abiertos, no existirá ningún tipo de acción en los actuadores (accesorios) así como lo indica el diagrama esquemático del sistema (véase figura 156). A continuación se deberá pronunciar los comandos entrenados

acorde al diagrama de flujo con la finalidad de activar el relevador concerniente a la acción deseada, el mismo que permitirá el paso del positivo hacia el actuador eléctrico (accesorio) por medio del cable de señal durante el tiempo programado (delay); al mismo tiempo el circuito es cerrado por un segundo cable al tomar el negativo perteneciente al relevador en conjunto del cambio de polaridad (excepto R5) así como se lo puede apreciar a continuación en el diagrama esquemático perteneciente a la implementación del sistema.

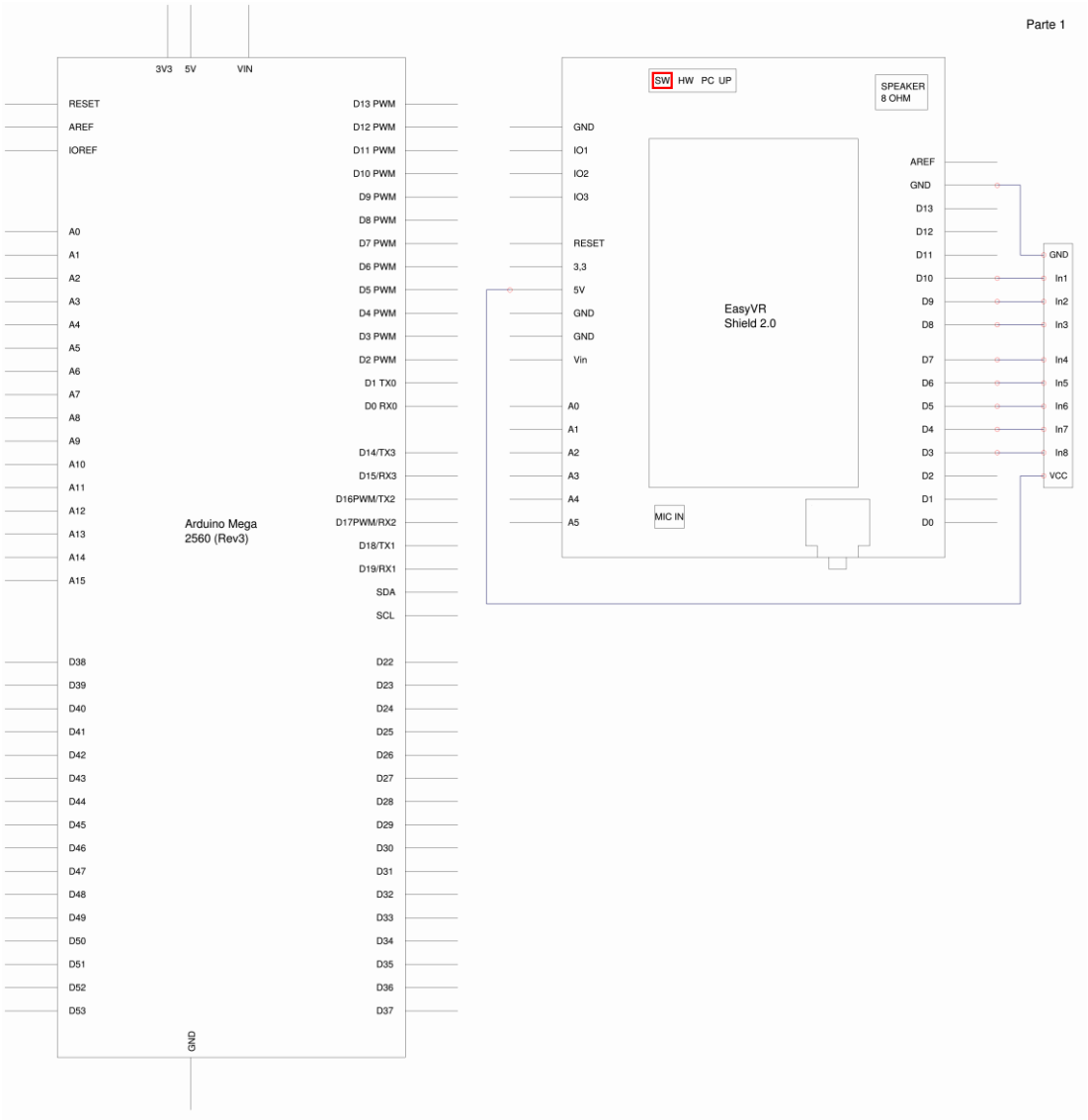


Figura 155. Diagrama esquemático de la implementación del sistema

Figura 156. Diagrama esquemático de la implementación del sistema

Como se puede analizar en el esquema tanto el positivo (batería) como el negativo (chasis) perteneciente a la alimentación de los actuadores (accesorios) es compartida para todos los relevadores en el módulo. Por otra parte cada relevador es activado (energizado) cuando la base de su transistor es cerrada por medio del pulso eléctrico alto (HIGH o 5V) enviado por el controlador (Arduino) hacia los pines digitales correspondientes; logrando de esta manera el total control de cada uno de los relevadores (acorde al código fuente diseñado) y por ende el de los actuadores implementados.

Cabe aclarar que el relevador 5 (R5) concerniente a la acción de encender o apagar la luz no tiene la necesidad invertir su polaridad, por lo que únicamente abre o cierra el paso del positivo R5 al ser llamado el comando "encender – luz"; y toma el negativo del relevador 8 (R8) el cual si recordamos no tiene ninguna función dentro del algoritmo (véase figura 156). Por otra parte los sonidos a reproducir durante la ejecución del sistema son tomados por medio de la salida universal J9 de la tarjeta Shield (véase figura 61) hacia los parlantes del vehículo; o sí el caso lo requiere, es posible colocar un parlante circular (altavoz) de 8 ohmios en la salida J10 de dicha Shield (EasyVR).

El sistema de reconocimiento de voz implementado funcionará sin ningún tipo de inconveniente siempre y cuando el lector haya acatado a plenitud la metodología utilizada en este estudio así como los conceptos, teorías, técnicas y recomendaciones otorgadas durante la investigación. Si bien el mundo de la computación física es muy extenso, se ha demostrado que es posible colaborar en el desarrollo de nuevas tecnologías por medio del conocimiento y que gracias a este recurso ilimitado la calidad de vida de los seres humanos ha mejorado.

4. ANÁLISIS DE RESULTADOS

4.1. ANÁLISIS EN BASE A LOS COSTOS EFECTUADOS

En la tabla 14 es posible analizar los costos efectuados durante el desarrollo del proyecto, donde al parecer el costo total sobrepasa el presupuesto inicial para una persona promedio. Sin embargo si se analiza con más detalle cada rubro asumido por el investigador, es posible darse cuenta que el costo más relevante llevado a cabo durante la investigación corresponde al costo del diseño; el mismo que concierne a la compra de un ordenador portátil Dell Inspiron 3421 core i5. Hoy en día la mayoría de personas disponen de un computador en casa por lo que no es necesario llevar a cabo dicha compra; pero si el caso así lo amerita, es posible encontrar en el mercado ordenadores a precios muy accesibles para cualquier persona ya que el software a utilizar (compilador IDE de Arduino, EasyVR Commander, QuickSynthesis 5, Fritzing, Qucs, dia, etc) no necesita de un computador de última generación; basta con un procesador de 3.0 GHz, una gigabyte de memoria RAM y un sistema operativo (GNU/Linux o Windows) de 32 bits suficientes para proceder con el diseño en lo que respecta al código fuente, diagramas de flujo, diagramas esquemáticos, montajes electrónicos, entre otros diseños llevados a cabo durante el proyecto.

Tabla 14. Costos efectuados durante la investigación

Rubro	***US$***	***Financiamiento***
Material bibliográfico	150,00	Investigador
Total Costo Análisis e Investigación	**150,00**	
Ordenador Portátil	670,00	Investigador
Total Costo Diseño	**670,00**	
Tarjeta Arduino Mega 2560 R3	40,00	Investigador
Tarjeta Shield EasyVR 2.0	70,00	Investigador

Módulo de relés 8 canales	15,00	Investigador
Seguros eléctricos	50,00	Investigador
Elevalunas eléctrico	150,00	Investigador
Bombilla de salón (luz)	-	Investigador
Total Costo Implementación	**325,00**	
COSTO TOTAL	**1145,00**	***Investigador***

Por otra parte el material bibliográfico (libros, artículos científicos, revistas, etc) utilizado es posible encontrarlo fácilmente en la web con mucho más detalle, debido a que existen grandes comunidades a nivel mundial que permiten infundir el conocimiento a través de tutoriales, vídeos y foros de ayuda como lo es Arduino, Fritzing, Veear, sparkfun, Xbee, etc. Además existen páginas web donde se puede encontrar millones de libros gratuitos como lo es “OpenLibra”, “issuu.com”, “google books”, entre otras páginas gratuitas que facilitan llevar a cabo el desarrollo de la investigación.

Finalmente se puede analizar el costo asumido durante la implementación, donde el hardware utilizado perteneciente a la tarjeta Arduino Mega 2560 R3, la tarjeta Shield EasyVR 2.0 y el módulo de relés 8 canales no sobrepasan los 125 dólares (en cualquier tienda online), los mismos que vendrían a ser el costo real del sistema (accesible para cualquier persona) sin contar claro con el precio de los actuadores a controlar deseados, ya que esto depende del usuario; sin embargo los costos efectuados en este estudio concernientes a los elevadores de vidrio y los seguros de puertas eléctricos valorados en 200 dólares hacen un total de 325 dólares al costo total de la implementación, precio que puede ser mejorado si se buscan en la web los actuadores apropiados a un costo sumamente bajo; o sí es el caso de tener los actuadores previamente instalados, únicamente se necesitará realizar la compra del hardware utilizado.

4.2. ANÁLISIS EN BASE A LOS ANTECEDENTES ENCONTRADOS

Según el estudio de nuevas tecnologías realizado por la consultora estadounidense J.D Power, el 60% de personas buscan en su nuevo automóvil una conexión inalámbrica con sus dispositivos móviles, es decir que acciones como abrir o cerrar las puertas de un vehículo no es necesario utilizar las llaves; esto se lo puede hacer a través de un botón en una tableta o un smartphone. En la actualidad grandes empresas como Apple ya han desarrollado herramientas para el control del automóvil por medio de dispositivos externos. CarPlay integra la pantalla del iPhone o iPad con el sistema de información y entretenimiento de un vehículo para recibir órdenes de voz mediante Siri (asistente virtual de Apple). Audi anunció a finales de junio que a partir del 2015, los autos que sean fabricados por esta empresa tendrán la tecnología CarPlay de Apple, que permite vincular los dispositivos de Apple con el vehículo. Esto será un servicio extra a los conductores, porque ayudará a saber los niveles de bencina, la duración de la batería del auto así como bloquear o abrir las puertas, según la publicación realizada el 1 de julio de 2014 por el diario informativo “El Comercio” y el portal web ohmygeek.net.

Al parecer todo lo anteriormente citado se escucha extraordinariamente favorable para el avance de las nuevas tecnologías en el área automotriz, y de cierta forma lo es; sin embrago la pregunta que todos se plantean es la siguiente “¿es accesible para todos?”, teniendo en cuenta que en el Ecuador no todas las personas tienen la capacidad o los recursos necesarios para adquirir un vehículo de alta gama (Audi) que disponga de este sistema denominado “CarPlay”. Además el código fuente de dicha tecnología es cerrado debido a que es una propiedad exclusiva de la empresa (Apple) y por ende no puede ser revelado a escusa de la competencia, lo que genera a su vez una desconfianza sobre todo cuando Google Now, Google Maps y Microsoft también se encuentran desarrollando aplicaciones de código cerrado (privativo) así como son “Android Car” y “Windows in the Car”,

dando paso de esta manera a las muy conocidas puertas traseras cuando de espionaje se trata.

Es por esta y otras razones que se ha diseñado e implementado un sistema de reconocimiento de voz capaz de interactuar con los usuarios del vehículo, el mismo que puede ser modificado (personalizado) acorde a las necesidades individuales de cada persona, limitado únicamente por la imaginación del ser humano. Este sistema está basado en un software y hardware libre flexible y fácil de utilizar, debido principalmente a que el conocimiento o información científica (especificaciones y diagramas esquemáticos refiriéndonos al hardware; y del código fuente en el caso del software) es de acceso público. Por otra parte Arduino cuenta con un sinnúmero de herramientas que permiten potencializar a un más el alcance de este controlador, así como lo son las tarjetas shields (EasyVR, Ethernet, GPS, WiFi, Bluetooth, Xbee, Celular, etc) y los sensores diseñados para Arduino (temperatura, humedad, ultrasonido, inclinación, magnético, luz, barométrico, alcoholímetro, velocidad, distancia, entre otros) que permiten no solamente la interacción de las personas en el mundo virtual (computación física); sino que también dan paso al desarrollo de nuevas tecnologías automotrices sin dejar un lado las demás disciplinas de la ingeniería. Proyectos como implementar un sensor de alcoholemia en el vehículo o un sensor de ultrasonido que indica la proximidad de un objeto durante el estacionamiento ya se los ha desarrollado con Arduino; por otra parte se debe indicar que Arduino cuenta con una serie de tarjetas controladoras (tipos) que facilitan aún más llevar a cabo cualquier tipo de proyecto en mente así como es la tarjeta “Arduino Mega ADK”, la misma que permite la comunicación con el sistema operativo “Android” utilizado actualmente por la mayoría de celulares móviles y que aún no se a implementado dicha tecnología en el área automotriz, dando paso de esta manera al surgimiento de nuevos campos de investigación.

4.3. ANÁLISIS DE LAS ACTIVIDADES DISTRATORAS DURANTE LA CONDUCCION E INCLUSION DE LAS PERSONAS CON DISCAPACIDAD

Sin lugar a duda el sistema de reconocimiento de voz mejora en gran medida la seguridad activa del conductor así como la de los usuarios al evitar la manipulación directa con los diferentes actuadores electrónicos del vehículo, debido principalmente a que la vista es el sentido más utilizado durante la conducción, al depender de esta en un 90% a comparación de los demás sentidos. Este sistema evita la distracción visual del conductor y por ende se reduce en cierta medida los accidentes de tránsito ocasionados por las actividades distractoras durante la conducción así como lo indica el estudio realizado por la RACC Automóvil Club, donde acciones simples como encender la calefacción, manipular los seguros de las puertas, abrir o cerrar las ventanillas, etc., resultan ser actividades muy peligrosas debido a la alta frecuencia en que se realizan.

Tabla 15. Actividades distractoras de duración y frecuencia elevadas

Actividades	Complejidad	Duración	Frecuencia
Mirar objetos en el exterior del vehículo	Media	Baja/Media	Alta
Insertar o retirar un CD del equipo de música	Media	Baja	Alta
Hablar por el teléfono móvil	Media	Alta	Media
Comer o beber	Media	Alta	Media/Alta
Fumar	Baja	Alta	Alta entre fumadores
Hablar con otro pasajero	Baja	Alta	Alta
Manipular controles en el interior del vehículo (climatización, ventanillas, etc.)	Baja	Baja	Alta
Encender la radio	Baja	Baja	Alta

Por otra parte es posible mejorar la interactividad y el confort de una persona discapacitada física y/o visualmente, ya que por medio de esta interfaz se descarta la necesidad de utilizar nuestros propios medios (excepto la voz) con la finalidad de accionar uno o varios accesorios electrónicos en el interior del vehículo. Según las estadísticas obtenidas en septiembre de 2014 por el consejo nacional de discapacidades CONADIS, en el Ecuador existen 193905 personas discapacitadas físicamente; mientras que 46435 personas tienen algún tipo de discapacidad visual. Por lo que este sistema resulta ser de mucha utilidad para aquellas personas que debido a su condición, llevar a cabo actividades como abrir las ventanas o encender la calefacción no es una tarea fácil de realizar; es por estas razones que se ha aportado en el desarrollo de la tecnología automotriz, con el único objetivo de mejorar la calidad de vida de los seres humanos a través del conocimiento.

5. CONCLUSIONES Y RECOMENDACIONES

5.1. CONCLUSIONES

- Se determinó que mediante la implementación del sistema de reconocimiento de voz, se ha logrado impulsar el mundo de la computación física en el área automotriz, mejorando de esta manera la interacción entre los usuarios y el vehículo por medio de dicha interfaz.

- Se determinó que mediante la utilización de este sistema, la seguridad durante la conducción mejora, ya que se evita la manipulación directa con los accesorios electrónicos del vehículo y por ende la distracción del conductor.

- Se concluye que la interacción y el confort de una persona discapacitada de sus facultades motoras y/o visuales mejora en gran medida, debido a que el control de los actuadores en el vehículo son ahora accionados mediante un comando de voz.

- Se concluye que el costo asumido durante el desarrollo de la investigación es accesible para cualquier persona, ya que el costo total del hardware utilizado no sobrepasan los 125 dólares americanos, lo cual lo convierte a su vez en una herramienta viable para el control integrado de proyectos de diseño e interacción.

- Se concluye que la plataforma Arduino juega un papel muy importante en el desarrollo de las nuevas tecnologías automotrices, debido principalmente a la gran comunidad de ayuda que cuenta dicha plataforma a nivel mundial así como el sinnúmero de sensores, módulos, shields, entre otros componentes diseñados para Arduino que permiten potencializar a aún más las funcionalidades y por ende el alcance de este microcontrolador.

5.2. RECOMENDACIONES

- Se recomienda que el comando a entrenar sea una palabra específica a la acción a realizar, debido a que podrían existir inconvenientes durante el reconocimiento de voz, ya que el tiempo en que se lleva vocalizar una oración no siempre es el mismo.

- Se recomienda instalar el micrófono en una posición adecuada a su patrón polar unidireccional; ya que una inadecuada colocación del micrófono reduciría notablemente la precisión durante el reconocimiento de voz.

- Es recomendable utilizar como fuente de alimentación el borne positivo directo de la batería, ya que sí lo tomamos de un derivado como lo es el switch de encendido, la corriente suministrada por este no es la misma debido a que existen otros consumidores en dicha línea.

- Se recomienda utilizar una fuente externa de 12 voltios DC, con la finalidad de conocer el sentido de giro de los motores eléctricos pertenecientes a los elevadores de vidrio y los seguros de puertas al momento de cambiar la polaridad.

- En el supuesto caso de haber realizado una conexión de polaridad errónea, se recomienda cambiar esta última ya sea en el interior del código de programación, en la entrada de los pines digitales del módulo o simplemente en las líneas de señal correspondientes a los elevadores de vidrio y seguros de puertas eléctricos.

GLOSARIO DE TÉRMINOS

PWM	Modulación por ancho de pulso (Pulse Width Modulation).
Conversor A/D	Conversión analógica-digital.
Conversor D/A	Conversión digital-analógica.
RAM	Memoria de acceso aleatorio (Random Access Memory).
ROM	Memoria de solo lectura (Read Only memory).
CPU	Unidad central de procesamiento (Central Processing Unit).
LED	Diodo emisor de luz (Light Emitting Diode).
Motor DC	Motor de corriente continua.
Corriente DC	Corriente continua o corriente directa (Direct Current).
Corriente AC	Corriente alterna (Alternating Current).
IDE	Entorno de desarrollo integrado (Integrated Development Environment).
ASCII	Código estándar estadounidense para el intercambio de información (American Standard Code for Information Interchange).
Sketch	Palabra utilizada en el lenguaje de programación “Processing”, la misma que hace referencia a un programa de computadora o software.
GNU	Sistema operativo libre, cuyo código fuente está basado en UNIX (Gnu's Not Unix).
Puerto COM	Interfaz para la comunicación serie de datos (digitales) por medio de un único cable.
UART	Transmisor-Receptor Asíncrono Universal (Universal Asynchronous Receiver-Transmitter), este dispositivo

	tiene como función controlar los puertos y dispositivos serie.
VCC	Voltaje en corriente continua (positivo).
GND	Masa o tierra (negativo).
E/S Análoga	Entrada-salida análoga.
E/S Digital	Entrada-salida aigital.
USB	Conector de serie universal (Universal Serial Bus).

BIBLIOGRAFÍA

- Alcalde, P. (2010). Electrónica Aplicada. (1ª Edición). Madrid - España: Editorial Paraninfo.

- Alcalde, P. (2009). Electrónica - Instalaciones eléctricas y automáticas. (1ª Edición). Madrid - España: Editorial Paraninfo.

- Arboledas, D. (2009). Electrónica para la educación secundaria. (1ª Edición). España: Editorial BUBOK.

- Benjumea, V., Roldán, M. (2013). Fundamentos de programación con el lenguaje de programación C++. España: Universidad de Málaga.

- Cairó, O. (2006). Fundamentos de programación. Piensa en C. Pearson Educación de México. Litográfica ingramex S.A.

- Condumex. (2009). Manual técnico de instalaciones eléctricas en baja tensión. (5ª Edición). México: Servicios Condumex S.A.

- Fajardo, J. (2012). Guía Básica de Arduino. Bogotá - Colombia: Editorial Robótica Colombia SAS.

- García, E. (2007). Principios básicos de informática. (1ª Edición). Madrid - España: Editorial DYKINSON.

- Harvey, M., Deitel., Paul, J. (2004). Cómo programar en C/C++ y Java. (4ª Edición). México: Editorial: Pearson.

- Hermosa, A. (2012). Electrónica aplicada CF Instalaciones de telecomunicaciones. (1ª Edición). Barcelona - España: Editorial Marcombo S.A.

- Hermosa, A. (2004). Electrónica digital fundamental. (3ª Edición). Barcelona - España: Editorial MARCOMBO S.A.

- Horacio, D. (2012). El mundo de la electrónica. Edición especial: 3001. Argentina: Grupo Enigma Electrónica.

- Landa, N. (2008). Curso de programación C#. (1ª Edición). México: Manuales USERS

- Marzal A., Gracia, I. (2008). Introducción a la programación con C. (1ª Edición). España: Editorial Universitat Jaume I.

- Moro, M. (2011). Instalaciones domóticas. (1ª Edición). Madrid - España: Editorial Paraninfo.

- Muñoz, J., Palacios, R. (2006). Fundamentos de programación utilizando el lenguaje C. (1ª Edición). España: Universidad Pontificia Comillas.

- Pérez, I. (2005). Lenguaje y Compiladores. (1ª Edición). Caracas - Venezuela: Editorial Impresos Minipres C.A.

- Ruiz, A., Ros, F., Rico, J. (2010). Guía práctica de sensores. (1ª Edición). España: Centro español de derechos reprográficos.

- Torrente, O. (2013). Arduino. Curso práctico de formación. (1ª Edición). Madrid - España: Grupo RC Libros.

- Agencia nacional de tránsito. (2014). Siniestros por causas probables a nivel nacional. Disponible en: http://www.ant.gob.ec/index.php/descargable/file/2680-siniestros-diciembre-2014

- Arduino. (2014). Modulación por ancho de pulso. Obtenido de:

http://www.arduino.cc

- Banshee, K. (2014). Modulación por ancho de pulso. [Imagen en línea]. Obtenido de: http://555timer0167.blogspot.com/

- Caro, J. (2009). Diferencia de potencial. [Imagen en línea]. Obtenido de: http://electronesenmovimiento.blogspot.com/

- Consejo nacional de igualdad de discapacidades “CONADIS”. (2014). Registro nacional de discapacidades. Obtenido de: http://www.consejodiscapacidades.gob.ec/wpcontent/uploads/downloads/2014/11/registro_nacional_discapacidades.pdf

- Gordón, A. (2014). Los dispositivos móviles facilitan el control vehicular. Diario informativo “El Comercio”. Obtenido de http://www.elcomercio.com/tendencias/dispositivos-moviles-facilitan-control-vehicular.html

- Onubaelectronica. (2014). Onda Senoidal. [Imagen en línea]. Obtenido de: http://www.onubaelectronica.es/osciloscopio.htm

- Pastrana, S. (2009). Corriente continua. [Imagen en línea]. Obtenido de: http://enciclopedia.us.es/index.php/Corriente_continua

- Pastrana, S. (2009). Diodo de vacío. [Imagen en línea]. Obtenido de: http://enciclopedia.us.es/index.php/Válvula_termoiónica

- RAAC Automóvil Club. (2008). La distracción de los conductores: un riesgo no percibido. Obtenido de: http://imagenes.racc.es/pub/ficheros/adjuntos/adjuntos_esp_distraccions_web_jzq_62fb66d0.pdf

- Tecnopatoo. (2013). Señal Análoga. [Imagen en línea]. Obtenido de: https://tecnopatoo.wordpress.com/2013/01/31/senal-analogica-y-senal-digital-2/

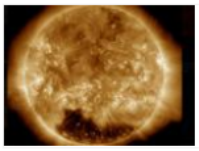
ANEXOS

ANEXO I: Artículo científico relacionado al proyecto de investigación

EL COMERCIO.COM ▸ TENDENCIAS (I)

INICIO ACTUALIDAD TENDENCIAS DEPORTES DATA OPINIÓN SERVICIOS

Temas a seguir: 2015 · Emelec · Quito · Cuba



Las auroras boreales son el resultado de los agujeros...

La contaminación por micropartículas aumenta la

Tendencias · tecnología

Los dispositivos móviles facilitan el control vehicular

Andrea Gordón y GDA agordon@elcomercio.com (I) · 1 de julio de 2014 18:54

Para bloquear o abrir las puertas de un vehículo no es necesario **utilizar las llaves.** Esto se puede realizar a través de un botón en una tableta o un '**smartphone**'.

El objetivo del control vehicular es que los conductores tengan una experiencia de uso similar al que tienen con los otros dispositivos móviles.

Diego Yépez, especialista de **Level 3**, indica que la tendencia actual es que los autos tengan conectividad a Internet. "Los autos van a enviar información básica a los celulares como ubicación, velocidad y posiblemente parámetros de control del motor y las estadísticas del uso de las vías de la ciudad", dice **Yépez.**

Según el estudio de nuevas tecnologías realizado por la consultora estadounidense **J.D Power**, el 60% de los consumidores buscará en su nuevo automóvil una conexión inalámbrica con sus dispositivos móviles.

Durante la última semana de junio, en Google I/O, conferencia celebrada en San Francisco, EE.UU., se presentó Android Car, un sistema que permite sincronizar otros dispositivos Android e integrar aplicaciones como el control de voz de Google Now y Google Maps en los autos. Según el diario español ABC, la empresa ya tiene acuerdos con 40 fabricantes de autos para equiparlos con estos sistemas.

RELACIONADAS

- **Llantas inteligentes se 'adueñan' de la Fórmula E**
- **Los robots copan más actividades de los humanos**
- **EE.UU. aprueba tecnología para que los autos 'hablen' entre sí y evitar accidentes**

Yépez indica que estos proyectos están iniciando y todavía no se puede obtener información especializada sobre la aplicación y la eficacia de los dispositivos en el control de los vehículos.

Sin embargo, otras empresas como Apple ya han desarrollado herramientas para el control del carro por medio de dispositivos externos**. CarPlay** integra la pantalla del iPhone o iPad con el sistema de información y entretenimiento de un vehículo para recibir órdenes de voz mediante Siri (asistente virtual de Apple).

Audi anunció a finales de junio que a partir del 2015, los autos que sean fabricados por esta empresa tendrán la tecnología **CarPlay de Apple**, que permite vincular los dispositivos de **Apple con el vehículo.**

Esto será un servicio extra a los conductores, porque ayudará a saber los niveles de bencina, la duración de la batería del auto y bloquear o abrir las puertas, según la publicación hecha por el portal web ohmygeek.net.

Microsoft también quiere estar en los autos con ‘Windows in the Car’, ya que a finales de abril presentó su plataforma para conectar dispositivos móviles a un auto; es similar al de Apple.

Estas innovaciones no se quedan atrás en el país. Estudiantes de la Escuela Politécnica Nacional (EPN) también realizan proyectos de comunicaciones inalámbricas para el acceso y control de los equipos vinculados a la conducción por medio de aplicaciones móviles para el sistema **Android.**

Jeaneth Acero, estudiante de la EPN, realizó una 'app' para el acceso a parqueaderos, que permiten levantar el brazo mecánico de un estacionamiento por medio de una aplicación instalada en un dispositivo móvil, que se conecta con '**bluetooth**'. Para ello, se instalaron sensores en el estacionamiento.

ANEXO II: Exoneración tributaria para las personas con discapacidad

Resolución Nro. SENAE-DGN-2014-0111-RE

Guayaquil, 11 de febrero de 2014

e Inversiones, **RESUELVE** establecer los siguientes:

REQUISITOS GENERALES PARA LA IMPORTACIÓN DE MERCANCÍAS CON EXONERACIÓN TRIBUTARIA PARA USO DE PERSONAS CON DISCAPACIDAD

TÍTULO I
IMPORTACION DE BIENES

Artículo 1: Importación de bienes: Las personas con discapacidad y las personas jurídicas encargadas de su atención, podrán realizar importaciones de bienes para su uso exclusivo, exentas del pago de tributos al comercio exterior, impuestos al valor agregado e impuestos a los consumos especiales, de acuerdo a la clasificación constante en el artículo 74 de la Ley Orgánica de Discapacidades.

Artículo 2: Autoridad competente, procedimiento: La autoridad competente es el Director Distrital o su delegado, quien a solicitud de la persona con discapacidad o la persona jurídica encargada de su atención, procederá a emitir el acto administrativo correspondiente de exención de tributos al comercio exterior, de conformidad con la normativa vigente. Para el efecto deberá presentarse junto a la solicitud los siguientes documentos:

a. Para la persona natural, el Certificado Único de Calificación de Discapacidad, emitido por el Ministerio de Salud Pública y suscrito por los profesionales autorizados. Las personas naturales sólo podrán importar los bienes que sean recomendados por los profesionales en este certificado.
b. Para el caso de la persona jurídica, el certificado de su registro ante el Ministerio de Salud Pública.

El acto administrativo constituirá documento de soporte para el proceso de despacho. El beneficiario de la exención presentará en el distrito de arribo de la mercancía una declaración aduanera a la que deberá adjuntar los documentos de soporte y de acompañamiento que correspondan.

Artículo 3.- Porcentaje de exoneración.- El porcentaje de exención de tributos para estos casos es del 100%, de conformidad con lo dispuesto en inciso final del artículo 6 de la Ley Orgánica de Discapacidad. Sin embargo, para acceder al mismo, la persona natural deberá poseer, al menos, el mínimo porcentaje de discapacidad definido en el Reglamento a la Ley Orgánica de Discapacidades.

TÍTULO II
IMPORTACIÓN DE VEHÍCULOS ORTOPÉDICOS Y NO ORTOPÉDICOS

Artículo 4.- Importación y compra de vehículos ortopédicos y no ortopédicos.- La importación de vehículos destinados al uso o beneficio particular o colectivo de personas con discapacidad, a solicitud de éstas, de las personas naturales y jurídicas que tengan legalmente bajo su protección o cuidado a la persona con discapacidad, gozarán de exenciones del pago de tributos al comercio exterior, impuestos al valor agregado e impuestos a los consumos especiales, con excepción de las tasas portuarias y de almacenaje, únicamente hasta por un monto equivalente a ciento veinte (120) remuneraciones básicas unificadas del trabajador privado en general del precio FOB, en transporte personal, y hasta por un monto equivalente a doscientas seis (206) remuneraciones básicas del trabajador privado en general del precio FOB, en transporte colectivo, y en los casos detallados en el artículo 80 de la Ley Orgánica de Discapacidades. El vehículo a importarse podrá ser nuevo o de hasta 3 años de fabricación, sin contar el de la importación.

Las personas jurídicas encargadas de la atención y cuidado de discapacitados, únicamente podrán importar vehículos de transporte colectivo de personas. Para efectos de la aplicación de este artículo, se entenderá que son vehículos destinados al transporte colectivo los que estén diseñados al menos para 10 personas, incluido el conductor.

Artículo 5: Valoración del vehículo: Tanto para los vehículos nuevos como usados, se tomará el valor reflejado en la factura comercial emitida por el concesionario de la marca, conforme lo emita el país exportador, como base imponible para el cálculo de tributos al comercio exterior, este valor deberá comprobarse al momento del control concurrente. Sin embargo para el caso de vehículos usados solo se aceptarán facturas que al momento de presentar la declaración aduanera de importación, tengan hasta un año de haber sido emitidas, caso contrario se rechazará el primer método de valoración y se procederá a determinar el valor por parte de la administración aduanera.

Para determinar el valor del vehículo, se debe considerar el valor total neto sin tomar en cuenta los costos adicionales que no guardan relación directa con el vehículo importado, como por ejemplo, costos de matriculación, costos de transporte, costos por gastos administrativos, costos por impuestos internos del país de exportación, etc.

No excluye la posibilidad de que la administración en el control concurrente genere procesos de "duda razonable" respecto del valor declarado y ejerza sus prerrogativas para determinar el valor de la mercancía. Tampoco ninguna de las disposiciones de la presente norma podrá interpretarse en un sentido que restrinja el derecho de la Administración Aduanera en un control posterior de comprobar la veracidad o exactitud de toda información, documento o declaración presentados dentro del trámite de nacionalización.

Artículo 6.- Autoridad competente, procedimiento.- La autoridad competente es el Director Distrital o su delegado, quien a solicitud de la persona con discapacidad o la persona jurídica encargada de su atención, procederá a emitir el acto administrativo correspondiente de exención de tributos al comercio exterior, de conformidad con la normativa vigente. Para el efecto deberá presentarse junto a la solicitud los siguientes documentos:

a. Para la persona natural el Certificado Único de Calificación de Discapacidad, emitido por el Ministerio de Salud Pública y suscrito por los profesionales autorizados.
b. Para el caso de la persona jurídica, el Certificado de su Registro ante el Ministerio de Salud Pública.

El acto administrativo constituirá documento de soporte para el proceso de despacho. El beneficiario de la exención presentará en el distrito de arribo de la mercancía una Declaración Aduanera a la que deberá adjuntar los documentos de soporte y de acompañamiento que correspondan.

Artículo 7.- Porcentaje de exoneración.- El porcentaje de exención de tributos al comercio exterior, se realizará de acuerdo al grado de discapacidad del beneficiario o de la persona a quien sustituye, según el caso, de conformidad con lo dispuesto en el artículo 6 de la Ley Orgánica de Discapacidades, en concordancia con el artículo 6 del Reglamento a la Ley Orgánica de Discapacidades.

De ser el caso, que el valor del vehículo importado exceda de los montos establecidos en la Ley Orgánica de Discapacidades, se permitirá la nacionalización del mismo, pagando los tributos que correspondan por la diferencia sin considerar el grado de discapacidad, liquidación que será efectuada de forma manual.

Artículo 8.- Uso de vehículos importados exentos.- Para efectos de la aplicación y control del presente beneficio, se entenderá como uso particular o colectivo para las personas con discapacidad, lo siguiente:

Uso particular: es aquel que cubre las necesidades propias del transporte personal y privado de la persona con discapacidad; el propietario del vehículo debe ser la persona con discapacidad, y podrá ser conducido por la persona con discapacidad beneficiaria o por los miembros de su núcleo familiar, integrado por los padres, los hijos, dependientes y el cónyuge o conviviente en unión de hecho. También podrá ser conducido por un tercero extraño a su núcleo familiar, siempre que la persona con discapacidad se encuentre en el vehículo.

Uso Colectivo: es aquel que cubre las necesidades propias del transporte colectivo de

Artículo 9.- Control posterior.- La autoridad aduanera nacional efectuará el control posterior de las importaciones, a fin de verificar el uso debido de los bienes importados, para lo cual podrá requerir toda la información que considere necesaria a las instituciones públicas relacionadas, sin perjuicio de la debida reserva que la autoridad aduanera deberá guardar en los casos señalados en la ley.

De constatarse transgresión de lo dispuesto en el artículo precedente, se presumirá el uso indebido del vehículo, debiendo sancionarse de conformidad con el artículo 81 de la Ley Orgánica de Discapacidades.

Artículo 10.- Transferencia de dominio y levantamiento de gravamen.- Los vehículos importados bajo estas condiciones no podrán ser transferidos a favor de terceras personas, salvo en casos debidamente comprobados de imposibilidad absoluta de uso del bien exento, como el siniestro del bien que haya sido asegurado mediante póliza, de conformidad con lo dispuesto en el artículo 22 del Reglamento a la Ley Orgánica de Discapacidades.

En caso de verificarse el fallecimiento del beneficiario directo de la exención, previo el pago de las alícuotas por parte de los interesados, la autoridad aduanera solicitará a la autoridad de tránsito el levantamiento de la prohibición de enajenar el vehículo para que los herederos o asignatarios testamentarios puedan disponer de dicho bien; así también, una vez transcurridos los cuatro (4) años constados desde la fecha en que dicho bien fue nacionalizado, podrá solicitarse el levantamiento del gravamen. n n En cualquiera de los casos señalados en los párrafos precedentes, serán los distritos aduaneros donde se realizó la importación los competentes para autorizar la transferencia de dominio o el levantamiento de gravamen correspondiente.

Artículo 11.- Prohibición.- Los bienes importados al amparo de este beneficio no podrán ser objeto de enajenación ni de cualquier acto jurídico entre vivos que signifique la transferencia de su dominio, posesión, tenencia o uso a terceras personas distintas del destinatario, salvo que haya transcurrido el plazo de cuatro (4) años contados desde la fecha en que dichos bienes hayan sido nacionalizados o adquiridos, o la excepción prevista para los vehículos asegurados que hayan sufrido un siniestro que signifique su pérdida total.

Artículo 12.- Sanción.- De incumplirse las disposiciones determinadas en la Ley Orgánica de Discapacidades y su Reglamento, el Director Distrital o su Delegado, sancionará a la persona con discapacidad o al representante legal de la persona jurídica

con el pago del monto total de la exención tributaria de la que fue beneficiado; sin perjuicio del pago de los tributos correspondientes y demás responsabilidades que pudieren determinarse conforme a las disposiciones legales que sancionen los ilícitos contra la administración aduanera.

DISPOSICIÓN GENERAL

PRIMERA: Los discapacitados y las personas jurídicas encargadas de su protección, tomarán como referencia el documento anexo a la presente resolución para presentar sus solicitudes de exención tributaria ante la administración aduanera.

DISPOSICIÓN TRANSITORIA

PRIMERA.- Hasta que se desarrollen las herramientas informáticas necesarias para efectos de lo establecido en el Artículo 11 de la Ley Orgánica de Discapacidades, se adjuntará a la solicitud de exoneración de tributos al Comercio Exterior, el certificado o documento que acredite la calificación de la discapacidad, emitido por la autoridad competente.

SEGUNDA.- Los trámites que se presenten ante la autoridad aduanera cuyos actos administrativos expedidos por el CONADIS sean anteriores al 17 de diciembre del 2013 y que contemplen beneficios tributarios por el 100% sin considerar el porcentaje de discapacidad, serán atendidos por la autoridad aduanera, salvo que dichas autorizaciones hayan caducado, por cuanto sólo puede considerarse como derecho adquirido mientras dichos actos tenga vigencia.

Según lo indicado anteriormente, la autoridad aduanera no aceptará autorizaciones de exención emitidas por el CONADIS desde el 17 de diciembre del 2013. Las solicitudes que no hubiesen sido atendidas por dicha entidad hasta esa fecha, deberán someterse al trámite respectivo, de conformidad a las disposiciones de la Ley Orgánica de Discapacidades, su reglamento de aplicación y la presente resolución.-

DISPOSICIÓN FINAL

Esta resolución entrará en vigencia a partir de su expedición sin perjuicio de su publicación en el Registro Oficial.

ANEXO III: Siniestros por causas probables a nivel nacional

SINIESTROS POR CAUSAS PROBABLES A NIVEL NACIONAL A DICIEMBRE -2014

CAUSAS PROBABLES	ENE	FEB	MAR	ABR	MAY	JUN	JUL	AGO	SEP	OCT	NOV	DIC	TOTAL A DICIEMBRE - 2014	REPRESENTACIÓN	%
IMPERICIA E IMPRUDENCIA DEL CONDUCTOR	1.321	1.457	1990	1736	1.467	952	836	663	703	716	713	780	13.334		34,49
NO RESPETA LAS SEÑALES DE TRÁNSITO	248	286	238	219	333	905	846	779	943	1129	1019	1082	8.027		20,76
CAUSA DESCONOCIDA	248	58	65	379	644	687	390	182	0	0	0	0	2.653		6,86
EXCESO DE VELOCIDAD	193	215	246	375	283	393	417	350	379	344	346	351	3.892		10,07
EMBRIAGUEZ DEL CONDUCTOR	192	199	239	197	245	223	223	188	150	196	175	211	2.438		6,31
IMPRUDENCIA DEL PEATÓN	179	157	185	206	155	190	197	145	148	179	137	137	2.015		5,21
INVADIR CARRIL	151	0	30	149	168	80	173	100	168	327	286	292	1.924		4,98
MAL REBASAMIENTO	47	59	69	51	66	150	135	225	38	92	76	69	1.077		2,79
OTRAS CAUSAS	73	284	69	118	33	31	12	5	0	0	27	6	658		1,70
CASO FORTUITO	45	21	36	54	55	35	60	29	27	76	75	94	607		1,57
DAÑOS MECÁNICOS	20	17	31	16	37	38	40	19	31	22	35	87	393		1,02
FACTORES CLIMÁTICOS	38	14	50	20	43	23	11	15	36	89	88	69	496		1,28
CANSANCIO AL CONDUCIR	6	0	0	0	9	14	87	50	29	70	64	63	392		1,01
MAL ESTADO DE LA VÍA	21	8	11	9	21	11	12	15	19	10	38	19	194		0,50
SALIDA DE ANIMALES A LA VÍA	12	2	3	9	25	23	20	7	0	0	0	0	101		0,26
MAL ESTACIONAMIENTO	5	7	2	1	10	10	8	20	12	6	25	3	109		0,28
CALZADA RESVALADIZA	3	2	9	0	6	11	5	10	13	11	39	16	125		0,32
OBSTÁCULOS EN LA VÍA	2	0	1	0	12	5	9	7	0	0	0	0	36		0,09
EXCESO DE PESO Y VOLUMEN	2	0	0	0	8	3	10	11	3	15	18	20	90		0,23
EMBRIAGUEZ DEL PEATÓN	2	2	6	5	1	4	2	6	10	10	5	7	60		0,16
FALLAS DE ILUMINACIÓN	2	0	0	0	-	4	6	11	0	1	0	0	24		0,06
ENCANDILAMIENTO	2	0	1	0	3	1	3	3	0	0	0	0	13		0,03
TOTAL	2.812	2.788	3281	3544	3624	3.793	3.502	2.840	2.709	3.293	3.166	3306	38.658		100,00

Fuente: DNCTSV, CTE, EMOV - Cuenca, Gobierno Autónomo Descentralizado Municipal de Loja, Agencia Metropolitana de Tránsito - Quito, Gobierno Autónomo Descentralizado de Manta, Municipio de Ambato, Municipio de Ibarra.

Elaboración: ANT, DEP; Quito, 06/01/2015

ANEXO IV: Personas con discapacidad a nivel nacional

REGISTRO NACIONAL DE DISCAPACIDADES (CONADIS)

Fecha: Septiembre 2014

PROVINCIA CANTÓN	TIPO DE DISCAPACIDAD							
	AUDITIVA	FISICA	INTELECTUAL	LENGUAJE	MENTAL	PSICOLOGICO	VISUAL	TOTAL
CASCALES	38	152	94	9	13	6	59	371
CUYABENO	16	84	49	9	11	1	14	184
GONZALO PIZARRO	32	92	51	6	4	3	27	215
LAGO AGRIO	278	1272	680	68	122	59	349	2828
PUTUMAYO	11	70	38	4	2	5	31	161
SHUSHUFINDI	91	454	221	35	35	18	152	1006
SUCUMBÍOS	19	76	23	2	4	5	19	148
TUNGURAHUA	**2461**	**4841**	**2763**	**181**	**239**	**197**	**1154**	**11836**
AMBATO	1401	3113	1430	97	159	142	806	7148
BAÑOS DE AGUA SANTA	119	251	120	17	11	4	48	570
CEVALLOS	58	85	75	3	7	2	15	245
MOCHA	47	99	82	2	6	4	15	255
PATATE	119	156	119	5	3	7	32	441
QUERO	170	181	166	26	15	2	33	593
SAN PEDRO DE PELILEO	255	511	324	10	19	24	80	1223
SANTIAGO DE PÍLLARO	194	308	336	10	14	12	90	964
TISALEO	98	137	111	11	5		35	397
ZAMORA CHINCHIPE	**413**	**1575**	**886**	**62**	**48**	**50**	**428**	**3462**
CENTINELA DEL CÓNDOR	33	93	70	5	5	3	38	247
CHINCHIPE	32	194	125	7	4	11	34	407
EL PANGUI	38	156	89	13	2	2	43	343
NANGARITZA	18	78	61	3	5	2	26	193
PALANDA	32	121	72	5	2	8	27	267
PAQUISHA	7	52	30	1		1	13	104
YACUAMBI	15	101	39	2	2	3	23	185
YANTZAZA (YANZATZA)	72	291	148	9	7	5	72	604
ZAMORA	166	489	252	17	21	15	152	1112
Total general	**48308**	**193905**	**87581**	**5510**	**6757**	**8737**	**46435**	**397233**

5/5

ANEXO V: Familia de controladores “Arduino”

Arduino UNO

Arduino Mega2560

Arduino Pro

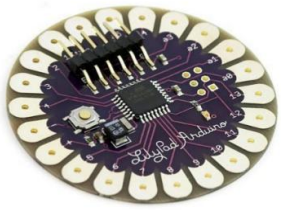
Arduino Fio

Arduino Mini

Arduino Bluetooth

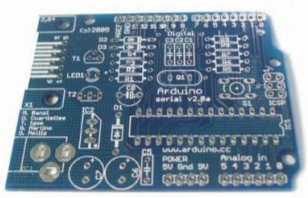
Arduino Mega ADK

Arduino LyliPad

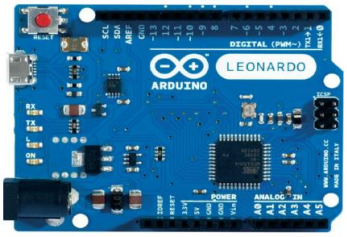


Arduino Nano

Arduino Serial



Arduino Leonardo



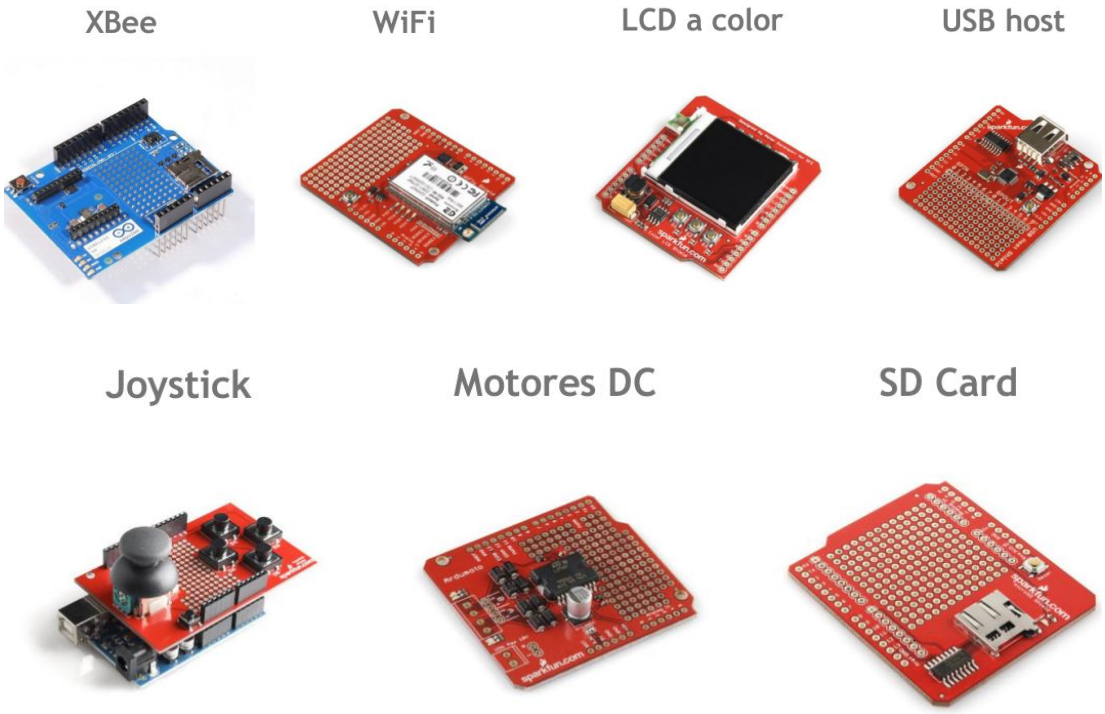
ANEXO VI: Tarjetas de tipo "Shield" para Arduino

Celular

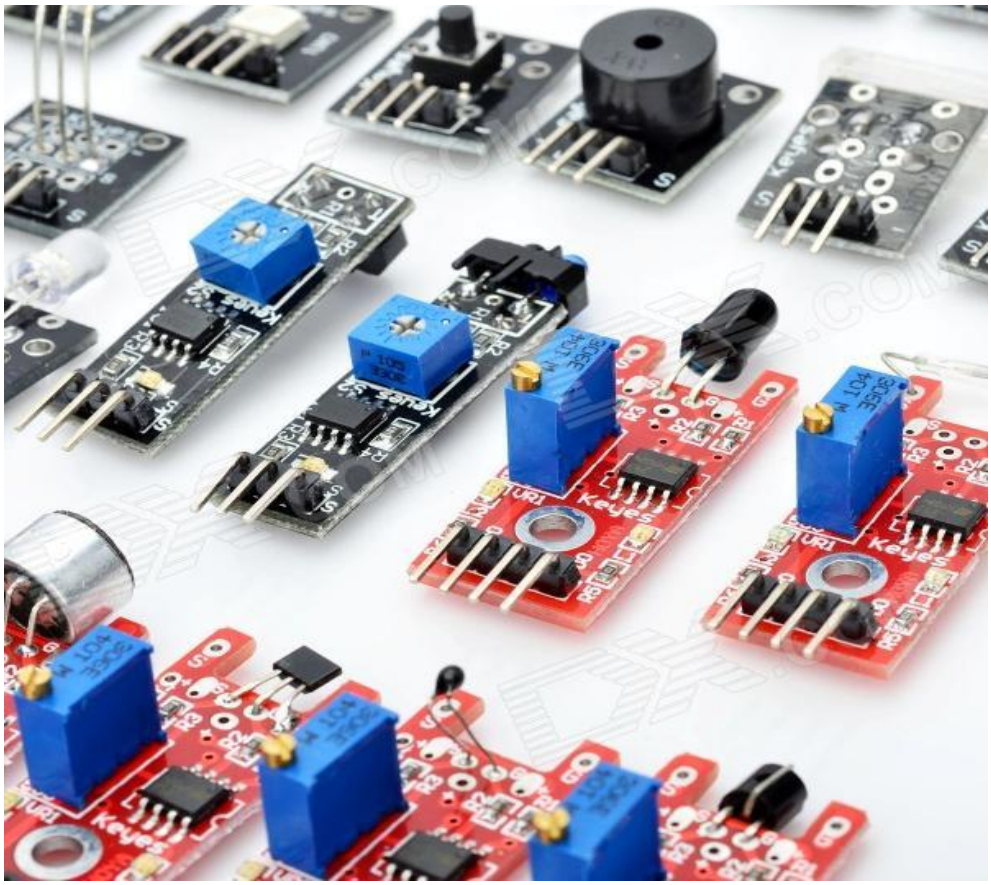
Ethernet

Proto

GPS



ANEXO VII: Sensores para Arduino



“Las obras de conocimiento deben ser libres, no hay excusas para que no sea así”

Richard Stallman

“Si piensas que los usuarios de tus programas son idiotas, sólo los idiotas usarán tus programas”

Linus Torvalds