Universidad Católica de Santa María

Facultad de Ciencias e Ingenierías Físicas y Formales

Escuela Profesional de Ingeniería Mecánica,

Mecánica-Eléctrica y Mecatrónica

CONTROL SIMULTÁNEO DE VEHICULOS NO TRIPULADOS UTILIZANDO LA CONFIGURACION MAESTRO-ESCLAVO

Tesis presentada por el Bachiller:

Juárez Valencia, Josué David

Para optar el Título Profesional de

Ingeniero Mecatrónico

Asesor:

Ing. Mestas Ramos, Sergio Orlando

AREQUIPA – PERU

2020

Universidad Católica de Santa María

AREQUIPA-PERÚ

☎ (51 54) 382038 http://www.ucsm.edu.pe facebook.com/u

ESCUELA PROFESIONAL DE INGENIERÍA MECANICA, MECÁNICA ELÉCTRICA Y MECATRÓNICA

INFORME DICTAMINATORIO

V I S T O

EL BORRADOR DE TESIS TITULADO:

"CONTROL SIMULTANEO DE VEHICULOS NO TRIPULADOS UTILIZANDO LA CONFIGURACION MAESTRO – ESCLAVO"

Presentado por el Bachiller:

JUAREZ VALENCIA JOSUE DAVID

Nuestro **DICTAMEN** es:

Aprobado

OBSERVACIONES:

Ninguna

Arequipa, 19 de noviembre 2019

ING. SERGIO MESTAS RAMOS

ING. CHRISTIAM COLLADO OPORTO

AGRADECIMIENTOS

Agradezco a Dios por cada aspecto de mi vida, a mis padres por darme siempre su apoyo y respaldo en mi educación, mi formación y en cada uno de mis proyectos. A mis hermanos por estar conmigo cada momento importante de mi vida, a mis familiares y amigos por ser mi soporte y aliento.

A mis docentes por la enseñanza, la formación y la guía que me otorgaron a lo largo de toda mi carrera.

RESUMEN

La presente tesis está destinada a la mejora de las condiciones operativas de una tecnología que en los últimos años ha crecido a gran escala, me refiero a los vehículos no tripulados, estos pueden desarrollarse en varios terrenos y ambientes para diferentes fines.

Se destinó el proyecto en la mejora del sistema de comunicación que presentan estos vehículos, para mejorar el control cuando se necesitan tener múltiples vehículos a trabajar en conjunto.

Se realizaron diferentes pruebas de comunicación, al igual que de tarjetas de control, se llegó a establecer la comunicación con varios controladores dando paso así a la construcción de pequeños modelos para las pruebas usando materiales simples enfocados en el cumplimiento de los objetivos planteados.

Estos modelos son simples, donde se busca la demostración de la buena comunicación que se pudo establecer entre los vehículos y el método que el usuario destino para controlarlos.

Una mejora que se abordo es la facilidad que debería tener un usuario en maniobrar con estos vehículos, esto se logró elaborando una aplicación para dispositivos inteligentes, más conocida con app.

La aplicación nos permitió tener el control de los vehículos en una sola pantalla, esto permitió poder manejar ambos vehículos al mismo tiempo sin que estos tuvieran interrupción en las órdenes que se les daban.

Finalmente se pudo lograr que las instrucciones dadas por la aplicación fueran correctamente recibidas por las tarjetas de mando seleccionadas y estas convirtieran estas instrucciones en ordenas a diferentes actuadores logrando así el movimiento de los vehículos construidos.

Palabras clave: vehículos no tripulados, dispositivos inteligentes, múltiples vehículos, maestro-esclavo.

ABSTRACT

This thesis is aimed at improving the operating conditions of a technology that has grown on a large scale in recent years, I mean unmanned vehicles, and these can be developed in various terrains and environments for different purposes.

The project was destined to improve the communication system presented by these vehicles, to improve control when multiple vehicles are needed to work together.

Different communication tests were carried out, as well as control cards, communication was established with several controllers, giving way to the construction of small models for tests using simple materials focused on the fulfillment of the proposed objectives.

These models are simple, where the demonstration of the good communication that could be established between the vehicles and the method that the user destined to control them is sought.

An improvement that is addressed is the ease that a user should have in maneuvering with these vehicles, this was achieved by developing an application for smart devices, better known with app.

The application allowed us to have the control of the vehicles in a single screen, this allowed to be able to handle both vehicles at the same time without these having interruption in the orders that were given to them.

Finally, it was possible to achieve that the instructions given by the application were correctly received by the selected control cards and these instructions were converted into orders to different actuators thus achieving the movement of the vehicles built.

Keywords: unmanned vehicles, smart devices, multiple vehicles, master-slave.

INTRODUCCIÓN

El interés por el uso de vehículos no tripulados nació en el área de las fuerzas militares que buscaban tener un armamento preciso en su funcionamiento y la reducción de las bajas humanas. Esta tecnología se basó primeramente en reemplazar a los seres humanos en el control y manejo de los diversos vehículos que se manejaban.

El avance tecnológico empezó a dar características diferentes a estos vehículos, modificaciones en el tamaño necesario para que estos artefactos cumplieran los propósitos para los que fueron creados, mayor maniobrabilidad, otras áreas de desarrollo y materiales menos costosos.

Esta tecnología ha avanzado a grandes pasos en los últimos años, que nos ha permitido interactuar con gran variedad de estos vehículos, ya sea de forma recreativa o de uso en labores y con objetivos más complejos.

La presente tesis es una propuesta al control múltiple de vehículos no tripulados, estos vehículos han tenido gran impacto en diferentes campos de desarrollo, en la actualidad se han convertido en instrumentos se usa más común para las personas y no solo en tareas de orden militar.

El poder de adquirir gran cantidad de vehículos en más sencillo cada vez, pero cada uno de estos vehículos carga a con ellos un sistema de control independiente y en algunos casos único.

Con el control múltiple que se plantea, se propone maximizar el potencial en los diferentes trabajos que los vehículos desarrollen, permitiendo una simplificación considerable en el uso y supervisión de cada vehículo no tripulado.

INDICE

Lista de figuras

CAPÍTULO I

1. MARCO METODOLÓGICO

1.1 Problema

En los últimos años, hemos sido testigos del avance tecnológico que se presenta en distintas partes del mundo, tecnología centrada en el desarrollo de vehículos autónomos capaces de desarrollar tareas complicadas o riesgosas para el ser humano, este avance se ha centrado en especial en la mejora de los vehículos no tripulados entre estos tenemos a los vehículos terrestres no tripulados (UGVs), vehículos submarinos no tripulados (UUVs) y los vehículos aéreos no tripulados (UAV).

Los vehículos no tripulados han demostrado desde sus primeras apariciones ser instrumentos muy versátiles y de gran ayuda para los diferentes campos en los que se los ha puesto a prueba, aunque mostrando también limitaciones en diversos aspectos al momento de realizar sus funciones debido a factores como las condiciones ambientales, problemas de comunicación, fallos en las funciones, etc.

La utilización de un vehículo no tripulado para realizar una tarea específica ha facilitado diversos trabajos para las personas; pero al necesitarse dos o más vehículos a la vez para la realización de una misma tarea la programación, los materiales, la comunicación y la supervisión de estos se vuelve una operación complicada ya que cada vehículo necesita de un sistema propio de operación a distancia y seguimiento.

El fin de estos vehículos es la simplificación de algunas tareas, al tener mayor cantidad de estos debería mejorar aún más el fin para el que sean requeridos, esto no suele suceder debido a la necesidad de vigilancia que requiere cada vehículo en la ejecución de la labor

para que fue asignado, volviendo un problema el uso de más de un vehículo por un solo usuario.

1.2 Descripción

El presente proyecto busca la aplicación de los diferentes conocimientos que proporciona la ingeniería mecatrónica, estos en los diversos avances tecnológicos que surgen en el mundo, su estudio en los campos de la automatización y de control de procesos; para hacer esto posible se incursionara en el campo de los vehículos no tripulados, explorando los diferentes rasgos que presentan y las posibilidades que estos proporcionan por su tecnología con el fin para plantear soluciones para mejorar algunas de sus deficiencias y limitaciones.

Se busca una forma de mejorar los beneficios que proporcionan los vehículos no tripulados a las distintas actividades humanas, esto se obtendrá mediante la opción de controlar varios de estos vehículos de forma simultánea, con el fin de optimizar las diferentes tareas que realizan, para ello se propone usar dos vehículos al mismo tiempo, a su vez se busca mejorar las capacidades en la comunicación y en el desarrollo de sus funciones.

Este mejoramiento será posible mediante la implementación de una configuración muy usada en sistemas computacionales y de control, esta es la configuración maestro-esclavo, la cual permitirá una mejora significativa en la utilización y la supervisión de los drones para optimizar las tareas para los que sean requeridos.

1.3 Estado del arte

"DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE CONTROL DE DISPOSITIVOS MAESTRO-ESCLAVO BASADOS EN LA RED INDUSTRIAL AS-I PARA EL LABORATORIO DE AUTOMATIZACIÓN INDUSTRIAL"

Autor: Bryron José Orellana Proaño

En la tesis se muestra el principio de funcionamiento de la configuración maestro-esclavo, se realiza un de modulo didáctico para que pueda ser utilizado en el laboratorio de automatización de la universidad.

La tesis presenta la implementación de la configuración maestro-esclavo basado en la red AS-I, se realizará un manual de prácticas para de manera progresiva ir capacitando a los estudiantes sobre el dominio de esta red ASI y su interacción con la red Profinet debido a que el controlador lógico programable y la pantalla HMI se comunican a través de esta red.

"VEHÍCULO AÉREO NO TRIPULADO PARA VIGILANCIA EN AMBIENTES CERRADOS CON DETECCIÓN DE PERSONAS Y OBSTÁCULOS A SU ALREDEDOR"

Autor: Sergio Renato Postigo Huanqui

Los diseños más convencionales de drones con aplicaciones específicas no están adaptados para volar en ambientes cerrados, pues no solo corren el riesgo de estrellar sus hélices contra paredes o techos, sino que también corresponderían un riesgo para las personas en su entorno. No obstante, hay tareas que podrían desarrollar en este tipo de

espacios si se adaptara su diseño a uno más seguro. Por ejemplo, podrían ser utilizados para realizar rondas de vigilancia a través de cuartos y pasillos tal y como lo hace el personal de seguridad. La segunda problemática apunta precisamente a este sector, pues dispositivos como cámaras de vigilancia tienen algunas limitaciones como el hecho de no ser disuasivas por no estar a la vista o de no estar necesariamente en los lugares donde se les necesita. El VANT será controlado desde una estación en tierra conformada por una laptop a la cual irá conectado el receptor de video. Desde aquí el operario enviará la trayectoria del vuelo al VANT podrá visualizar el video en vivo.

"SISTEMA HÁPTICO PARA LA OPERACIÓN MAESTRO-ESCLAVO DE UN ROBOT INDUSTRIAL"

Autor: Francisco Javier Mendoza Galindo

Los robots cumplen con una importante función en el área industrial ya que realizan tareas de manera eficiente, sin embargo, han adquirido auge en campos como medicina donde son guiados por un operador para realizar intervenciones quirúrgicas. Este trabajo presenta la operación maestro-esclavo de un robot industrial utilizando un dispositivo háptico de 3 grados de libertad como maestro y un robot PUMA como esclavo. El uso de un dispositivo háptico permite que el usuario realice los movimientos de una manera natural ya que el dispositivo háptico se mueve a través del espacio. Se obtuvieron las ecuaciones de cinemática inversa para robots PUMA con el objetivo de que el movimiento del dispositivo háptico sea replicado lo más rápido posible y del mismo modo. Pese a que se tiene un maestro con menor número de grados de libertad que el esclavo, se mantiene fija la orientación del órgano terminal del robot PUMA.

1.4 Justificación

Esta investigación es necesaria para optimizar las características que poseen los Vehículos no tripulados las cuales les permiten ayudar en diferentes tareas para la población.

Este proyecto tiene como finalidad controlar varios vehículos a través de un solo dispositivo líder que nos permita realizar una buena comunicación entre los vehículos y un mejor control de los trabajos para la realización de una tarea programada, optimizando los componentes que se necesitan para la coordinación de varios de estos vehículos.

Con el propósito de mejorar y hacer más eficiente los diferentes trabajos que pueden realizar estos dispositivos, en aplicaciones como el monitoreo de una zona geográfica, la irrigación de campos de cultivo, búsqueda de personas, movimiento de paquetes de entrega, vigilancia de un área, etc.

También se realiza para poder dejar un antecedente de investigación para una futura indagación en el campo de los vehículos no tripulados y los avances que estos presenten en el tiempo.

1.5 Objetivo

1.5.1 Objetivo General

Controlar simultáneamente dos vehículos no tripulados usando la configuración maestro-esclavo.

1.5.2 Objetivos Específicos

- Controlar de dos a cinco vehículos con un solo medio de mando.
- Determinar el mejor medio de transmisión de datos entre el maestro y los esclavos.
- Diseñar un interfaz de comunicación.
- Lograr una buena comunicación con los vehículos.
- Mejorar el rendimiento de los vehículos.

1.6 Hipótesis

La implementación de la configuración maestro-esclavo en vehículos no tripulados aumenta la eficiencia en el uso de varios de estos vehículos, en múltiples tareas en simultáneo.

CAPÍTULO II

2. MARCO TEÓRICO

2.1 Vehículo No Tripulado (VNT)

Un vehículo no tripulado se puede definir como un conjunto de elementos mecánicos y electrónicos que se desplazan en un entorno terrestre, acuático o aéreo con el fin de cumplir una tarea asignada.

Los vehículos no tripulados actualmente son construidos con la finalidad de realizar trabajos pequeños o que impliquen su fácil transporte y ejecución (Omar Sánchez).

2.1.1 Clasificación

Existen diversos motivos por los que se han desarrollado los vehículos no tripulados, ya sea por cumplir un trabajo específico, asistir en tareas de desarrollo en distintas áreas de trabajo o simplemente por recreación.

Por las distintas características y tareas para los que estos vehículos son diseñados se han separado a los vehículos en tres grandes grupos:

- Vehículos terrestres no tripulados

  Se desplazan en tierra, suelen ser copias en miniatura de vehículos terrestres comunes, usan llantas redondas o tipo oruga según la misión que desarrollaran.

Figura 2.1 Vehículo terrestre no tripulado
Fuente: https://www.swissinfo.ch/spa/desminado-made-in-switzerland-_el-suizo-que-ha-declarado-la-guerra-a-las-minas/43086202

- Vehículos acuáticos no tripulados

  Estos vehículos son los que se mueven en entornos líquidos, se pueden dividir en dos sub categorías: los que se mueven sobre la superficie del líquido y los que se sumergen en el líquido; estos últimos son los que más desarrollo tecnológico tiene debido a la cantidad de usos que se le puede dar.

Figura 2.2 Vehículo acuático no tripulado
Fuente: https://www.zimarobotics.com/areas-de-negocio/vehiculos-no-tripulados-drones/

- Vehículos aéreos no tripulados

  Son vehículos que se desplazan en un medio gaseoso generalmente en el aire, estos vehículos también son conocidos como drones, tienen muchas sub divisiones debido a la gran popularidad que han tenido, algunos son miniaturas de

vehículos conocidos, otros tienen nuevos diseños que mejoran sus capacidades y aprovechan mejor la condición de no ser tripulados.

Figura 2.3 Vehículo aéreo no tripulado
Fuente: https://www.vaiu.mx/implementacion-de-vehiculos-aereos-no-tripulados-para-estudios-tecnicos/

2.1.2 Sistemas de control

Los vehículos no tripulados constan de varios componentes para que puedan moverse y realicen su tarea de forma satisfactoria; entre los componentes tenemos: chasis, armadura o carcasa, microcontroladores, actuadores, sensores, sistema de comunicación, etc.

Para poder distinguir mejor estos componentes debemos separarlos en sus dos partes: el vehículo y el mando.

- Vehículo

  Se refiere a la composición del vehículo a controlar donde tenemos generalmente el chasis, que es el esqueleto del vehículo, la carcasa encargada de proteger los componentes, un controlador o cerebro del vehículo, diversos sensores encargados de alimentar con información al controlador y los actuadores que trasmitirán las órdenes para generar los movimientos del vehículo.

- Mando

  El sistema de mando es el que controla las funciones del vehículo, se encuentra generalmente estático, se encarga de dar las órdenes a distancia, recibe y envía la información necesaria para dirigir correctamente al vehículo en su tarea.

Figura 2.4 PC portátil de control
Fuente: Catalogo Expal

2.1.3 Arquitectura de un VNT

La arquitectura de un VNT abarca los componentes básicos con los que generalmente cuentan estos vehículos, conocer estos elementos nos da un punto de partida para un diseño según el propósito que deseemos darles.

A continuación, se nombrarán algunos de estos componentes que tienen la mayoría de los VNT

- Chasis

Corresponde al armazón o estructura que le da forma al vehículo, también se encarga de sostener los diversos componentes necesarios para el funcionamiento del vehículo.

La forma del chasis estará determinada de acuerdo a la tarea asignada al vehículo, es muy importante tener esto en cuenta debido a que en el chasis se distribuyen lo elementos, debe tener una buena rigidez ya que al fallar el chasis todo el conjunto de componentes fallar.

Algunos materiales de los cuales son fabricados tenemos: fibra de vidrio, fibra de carbono, madera, plástico, aluminio, acero, titano, etc. El material será elegido según la necesidad que tengamos ya que algunos son más ligeros, otros más resistentes y algunos ofrecen mayor rigidez.

El diseño del chasis es muy decisivo en el comportamiento de vehículo no tripulado, por eso es muy importante considerar su forma, la distribución de los componentes, el peso de cada componente y el mejor lugar que tenga cada uno para desarrollar su tarea.

- Actuadores

Son elementos mecánicos que dan una fuerza para poder mover un componente mecánico, puede ser una fuerza rotacional o lineal; existen diferentes tipos de actuadores y diferentes formas de clasificarlos, la división más práctica de dividirlos es en actuadores neumáticos, hidráulicos y electromecánicos.

En los vehículos no tripulados usualmente tenemos actuadores que mueven llantas y/o hélices de diferentes tipos para que el vehículo se mueva según lo requerido, también se pueden usar actuadores que realizar una tarea específica que no corresponde con el movimiento del vehículo.

- Sensores

Los sensores son dispositivos que captan información del entorno en forma de estímulos, para los vehículos no tripulados en muy importante tener una referencia del medio en el que desarrollan su tarea, estos sensores varían de acuerdo a la necesidad de cada vehículo, algunos de estos sensores captan: distancia, giro, velocidad, aceleración, humedad, etc.

Los sensores deben estar ubicación estratégicamente en el chasis del vehículo, esto según la función que cumplan los sensores y de acuerdo a la misión que le debes.

- Microcontrolador

El microcontrolador es el componente más importante del vehículo, podríamos compararlo con el cerebro, se encarga de recibir los datos de los sensores y las órdenes del piloto; esto para luego dar órdenes a los actuadores que compongan al vehículo.

De acuerdo a su complejidad y programación; el microcontrolador se encargará de dirigir al vehículo y de que este llegue a cumplir con la tarea asignada.

- Sistema de comunicación

El sistema de comunicación es la parte más esencial de los vehículos no tripulados debido a que no solo se encarga de trasmitir información con un vehículo común, sino que depende de este sistema que el vehículo se desempeñe según lo deseado.

En los vehículos es muy variado los sistemas de comunicación, encontramos dos grandes tipos de estos sistemas los alámbricos, que son los que mantienen una conexión física entre el mando y el vehículo; y el inalámbrico, que no está físicamente conectado al vehículo, sino que trasmite su información por señales de onda.

La selección del tipo de comunicación depende de donde se desarrollará la tarea del vehículo, debido a que en algunos ambientes las comunicaciones inalámbricas no son muy confiables como en el caso de la exploración de cuevas o exploraciones subacuáticas.

Referencia: *Introducción y Arquitectura de microcontroladores.*

2.2 Configuración Maestro-Esclavo

La configuración maestro-esclavo es usada en diversos sistemas, ya sea en una red de comunicación o en una secuencia de mandos en una industria, esta configuración tiene grandes ventajas en las tareas en simultáneo (National Instruments).

Tenemos dos elementos fundamentales de esta configuración que son el maestro y los esclavos.

2.2.1 El Maestro

Se le conoce así al elemento que controla todo el sistema, su tarea es la recibir la orden principal y encargarse que se cumpla; el maestro se encarga de controlar a los dispositivos esclavos, solo este dispositivo puede interactuar con los otros dispositivos y nada puede saltar sus órdenes. No solo se encarga de dominar a los esclavos, sino que también puede realizar tareas independientes.

2.2.2 Esclavos

Tienen esta denominación los dispositivos que están sujetos a las órdenes del dispositivo maestro que se encarga de darle órdenes para que cumpla su función, los dispositivos esclavos se dedican únicamente a realizar su tarea, están constantemente en comunicación con el dispositivo maestro, los esclavos no pueden interactuar entre ellos, solo pueden transferir datos al maestro.

2.2.3 Configuración Maestro-Esclavo

La configuración se basa en el protocolo de comunicación Modbus, este se basa en el sistema de solicitud-respuesta, donde el dispositivo maestro manda órdenes al dispositivo esclavo con el fin de realizar una función.

El principio de funcionamiento de esta configuración se basa en que los dispositivos esclavos esperar un comando del dispositivo maestro, loes esclavo solo se comunican con el dispositivo maestro y este supervisa el comportamiento de cada esclavo.

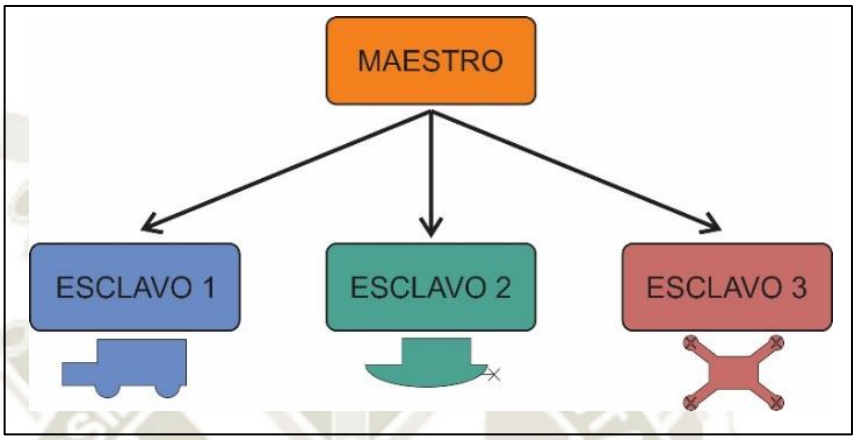


Figura 2.5 Esquema Maestro-Esclavos
Fuente: Propia

2.3 Microcontrolador

Los microcontroladores son circuitos integrados programables se les cargar un programa para la realización de diferentes tareas, para esto los microcontroladores cuentan básicamente con una unidad central de procesos, una unidad de memoria y puertos de entra y salida (Sesanchezv).

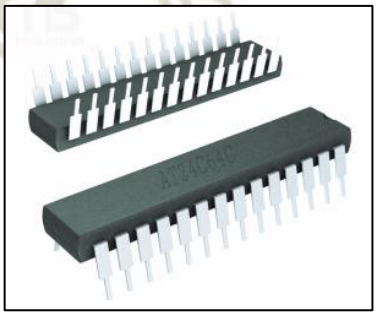


Figura 2.6 Microcontrolador
Fuente: https://teslabem.com/blog/como-elegir-el-microcontrolador-correcto/

2.3.1 Modo de operación

En el mercado existen diversas marcas que desarrollan microcontroladores, cada una suele tener su propio protocolo de funcionamiento, de lenguaje de programación y distribución de elementos.

Aunque muchos aspectos varían de acuerdo a la marca, los microcontroladores suelen tener un mismo principio de operación.

Los microcontroladores son encapsulados que como componentes electrónicos que son requieren ser alimentados con una tensión, también tienen parámetros de funcionamiento con la necesidad de un reloj que se encarga de dar pautas a su funcionamiento, este parámetro de tiempo es dado normalmente por un cristal oscilador.

Los encapsulados están diseñados de modo que protejan su integridad y faciliten al usuario. Esto implica que tienen elementos que deben ser alimentados o necesitan de un dato digital lógico, tenemos componentes con lo enable, reset, programación y algunos otros.

Según sean las características del microcontrolador tenemos pines que se encargan de a comunicación, de programar, entradas analógicas, entrada digitales, salidas analógicas y salidas digitales. Muchas de estas funciones pueden ser cumplidas por un mismo pin de salida y según la programación escoger la función.

2.3.2 Tipos

Existen muchos tipos de microcontroladores, esto debido a los diferentes propósitos que se les encomienda, algunas tareas llegan a ser muy complejas, otras más simples, unas necesitan gran interacción de puertos, otros pocos puertos y mucha memoria; debido a

esto los microcontroladores se suelen dividir en tres tipos: Gama baja o básica, gama media y gama alta.

Hoy en día se usan microcontroladores desde pequeños aparatos caseros hasta para el control de grandes motores y secuencias industriales.

2.3.3 Complementos

Los microcontroladores tienen gran variedad de usos y funciones; sin embargo, muchas veces se quedan cortos para algunas tareas por ello se han ido implementando diferentes componentes que potencian sus cualidades, estos componentes suelen ser independientes ya que no es obligatorio que el microcontrolador lo tenga porque eso aumentaría su capacidad en hardware y software.

Algunos de estos complementos están diseñados para facilitar el programa del microcontrolador ya que mandan señales que el programa puede usar directamente sin la necesidad de filtrarla o adecuarla, tenemos, por ejemplo: transductores de humedad, transductores de temperatura, transductores de luz, controladores de motor, controladores de relés, emisores de señal bluetooh, antena WI-FI, etc.

Figura 2.7 Kit de desarrollo
Fuente: https://store.arduino.cc/usa/arduino-ctc-101-program-sl

2.4 Sistemas de Telecomunicación

La telecomunicación se define como la transmisión y recepción de señales de cualquier tipo información.

También es conocida por ser la disciplina que estudia las señales y los diferentes medios que tiene para ser transmitida

2.4.1 Canal

Los diferentes medios por donde se transmite la comunicación se pueden dividir en dos grupos los de forma alámbrica y los de forma inalámbrica.

- Alámbrico

Se refiere a la comunicación por medio de cables donde el emisor y el receptor mantienen una conexión sólida, esto permite que la transmisión de datos sea mejor, rápida y con un alto grada de eficiencia; algunas desventajas es que su distancia está limitada por la longitud del cable de comunicación, y que depende mucho de la calidad de estos cables.

- Inalámbrico

La conexión inalámbrica es la que se da cuando la comunicación se da sin una conexión solida entre el emisor y el receptor; este tipo de conexión tiene la ventaja de no estar limitada por los parámetros de objetos sólidos, nos permiten grandes distancias, mayor cantidad de puntos de conexión, menor costo de instalación y menor peso en los equipos; tenemos algunas desventajas como la señal que algunas veces no puede atravesar ciertos objetos sólidos, son más inseguras, tienen menor ancho de banda y suelen ser más inestables.

2.4.2 Tipos

- Radio control

Radio control es un sistema que nos da el mando sobre un equipo a distancia y de forma inalámbrica; este control lo hace mediante la emisión de una onda con una forma característica que permite la interacción entre el emisor y el receptor.

El radio control es uno de los métodos más usado por su largo alcance, su facilidad de uso y su bajo costo de producción; algunas desventajas tenemos que solo pueden controlar un aparato de forma confiable a la vez, se puede interferir en su señal, no es muy confiable al trasmitir información.

- Bluetooth

El bluetooth es un dispositivo creado con la finalidad de transmitir datos entre dispositivos mediante un enlace de radiofrecuencia en la banda ISM de los 2.4 GHz.

Como medio de telecomunicación este nos permite crear pequeñas redes, simple configuración y buena conexión entre equipos; aunque su modo de operación es limitado debido a que su alcance es muy bajo, no tiene buena seguridad y su comunicación se limita un solo dispositivo a la vez.

- Wi-Fi

  La palabra Wi-Fi es originalmente una abreviación de la marca Wireless Fidelity, que podemos traducirlo como “Fidelidad inalámbrica”. La tecnología Wi-Fi permite implementar redes de conexión para múltiples usuarios, se asocia el Wi-Fi a la conexión con el internet ya que fue concebido principalmente para facilitar su uso (Significados).

Esta tecnología permite la interconexión inalámbrica de diferentes dispositivos electrónicos por medio de un punto de acceso o mediante la conexión a internet.

La tecnología Wi-Fi está basada en el protocolo IEEE 802, este especifica una frecuencia de operación de 2.4 GHz con velocidades de transmisión de 1 y 2 Mbps. El protocolo se inició en 1997, desde entonces la tecnología a sufrido mejoras para la interacción entre equipos, estas mejoras se dan en la velocidad, la capacidad y el alcance de la señal.

El 802.11 es una arquitectura donde el sistema se divide en celdas. Cada celda es controlada por una estación base denominada AP (Access Point). La mayor parte de las instalaciones están compuestas por un conjunto de celdas formando una red con los APs conectados. La tecnología basa su funcionamiento.

En la actualidad existe una norma para el uso del IEEE 802.11, esto ha generado una gran variedad de protocolos que se usan de acuerdo a las necesidades, a partir de sus características han resaltado:

- 802.11a, lanzado en 1999, trasmite datos en velocidades de 6 Mb/s, 9 Mb/s, 12 Mb/s, 18 Mb/s, 24 Mb/s, 36 Mb/s, 48 Mb/s y 54 Mb/s.
- 802.11g, lanzado en 1999, las velocidades en las que transmite datos son 1 Mb/s, 2 Mb/s, 5,5 Mb/s y 11 Mb/s. Su intervalo de frecuencia es entre 2.4 y 2.4835 GHz, en rango de alcance de transmisión es de 400m en lugares abiertos y de 50m en lugares cerrados; se puede ver afectado por objetos que causan interferencia. Fue el que más popularizo las redes Wi-Fi.
- 802.11b, fue lanzado en el 2003, se lo conoce como el sucesor de la versión 802.11b, tiene similares características y trabaja fácilmente con equipos que cuenten con el protocolo estándar 802.11b y 802.11ª.

- 802.11n, lanzado en 2009, tiene como principal característica el uso de un esquema llamado Multiple-Input Multiple-Output (MIMO), el protocolo 802.11n es capaz de hacer transmisiones en el rango de 300 Mb/s y, teóricamente, puede alcanzar velocidades de hasta 600 Mb/s. En el modo de transmisión más simple, con una vía de transmisión, el 802.11n puede llegar a los 150 Mb/s. En relación a su frecuencia, el estándar 802.11n puede trabajar con las bandas de 2,4 GHz y 5 GHz, lo que lo hace compatible con los estándares anteriores, incluso con el 802.11a. Cada canal dentro de esas pistas tiene, por defecto, ancho de 40 MHz.
- 802.11ac, lanzado entre los años 2011 y 2013. Es el sucesor del 802.11n, su velocidad estimada en hasta 433 Mb/s, es posible hacer que la red supere los 6 Gb/s en un modo más avanzado que utiliza múltiples vías de transmisión, con un máximo de ocho. Se destaca también el uso de un método de transmisión llamado Beamforming (también conocido como TxBF), que en el estándar 802.11n es opcional: se trata de una tecnología que permite al aparato transmisor (como un router) evaluar la comunicación con un dispositivo cliente para optimizar la transmisión en su dirección (WI-FI Alliance).

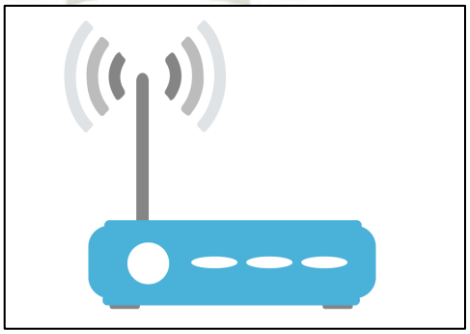


Figura 2.8 Punto de acceso WIFI
Fuente: https://linube.com/blog/wpa3-protocolo-redes-wifi/

2.5 Entorno de Control

Es el instrumento por el cual se podrá interactuar con el vehículo no tripulado para poder enviar información en fin de cumplir la tarea propuesta.

En una red de manejo de distintos equipos siempre se tiene una forma de monitorear los equipos que se estén usando, dependiendo de la complejidad de este entorno de control se los puede observar, diagnosticar y manipular a distintos grados dependiendo de su tecnología.

2.5.1 Tipos de control

- Autónomos

Los vehículos autónomos tienen la característica principal de ser no necesitar de ninguna orden durante la ejecución de su labor, la única orden dada es la de inicio y una orden de interrupción en caso de necesitar suspender su labor.

Sus movimientos dependen de la respuesta del programa que tiene a la información recabada por sus sensores.

La información que suelen transmitir se da al concluir la tarea asignada, no necesita que se vigilen sus movimientos teniendo así un alto nivel de confianza.

- Remota

Son los controlados por un piloto a distancia, este vehículo realiza funciones correspondiendo a las mandadas mediante algún dispositivo remoto, sus sensores le permiten obtener información del entorno de trabajo; sin embargo, sus respuestas dependen de la decisión del operador.

Los programas en estos casos suelen tener acciones para la protección del vehículo en la realización de su tarea, pero se puede ignorar dependiendo de la programación que estos tengan.

- Supervisada

Este tipo de control cumple con parámetros similares a los de un vehículo no tripulado autónomo, la diferencia radica en que su funcionamiento siempre debe ser vigilado por un piloto, esto se debe a que en la realización de su tarea encuentra dificultades que en su programación no tienen respuesta o que al tener diferentes alternativas de comportamiento la decisión de la respuesta debe depender de un usuario.

La mayoría de vehículos en la actualidad cuenta con este tipo de control, esto se debe a su carácter semiautomático que permite que los vehículos puedan cumplir sus objetivos sin requerir un gran conocimiento del usuario del vehículo.

2.5.2 Formas de control

En el manejo de los vehículos no tripulados se cuentan con diferentes dispositivos que nos facilitan él envió de órdenes y la comunicación necesaria para un correcto uso de estos.

Algunos de estos dispositivos son:

- Joystick

Se conoce como Joystick al mando que cuenta con una o varias palancas que al ser movidas generan ordenes, existen una gran variedad de estos dispositivos, sus características varían de acuerdo a la necesidad con la que son diseñados.

- Dispositivos móviles inteligentes

Algunos de los dispositivos inteligentes más conocidos son los teléfonos celulares y las tablets.

Estos dispositivos cuentan con diferentes sistemas de operativos, las diferentes funciones que cumplen se deben a las aplicaciones que estos tienen, estas aplicaciones se las conoce como app.

En la actualidad existen apps para muchas tareas que el usuario realiza, estas pueden llevar al máximo las características de los dispositivos, gracias a su versatilidad se pueden enlazar comandos que permitan el control de los vehículos no tripulados.

- Computador o PC

De todos los dispositivos de control el más complejo es el computador personal o PC.

Estos tienen todo lo necesario para poder ser un instrumento de control óptimo de los vehículos no tripulado, sus limitaciones en cuanto a recursos de software son muy pocos ya que cuentan con grandes características.

La mayor dificultad que poseen es su tamaño y capacidad; para controlar un vehículo no se necesita la gran complejidad de estas máquinas y genera una dificultad más que una ayuda usarlas para las tareas que realizan la mayoría de vehículos.

2.5.3 Importancia

Este entorno de control no solo debe estar enfocado en cumplir una tarea con la de interactuar con los distintos equipos sino también de tener una interfaz amigable y fácil de manipular por el usuario generando un método más óptimo de funcionamiento.

En los últimos años el desarrollo de aplicaciones de forma libre a permitido que los entornos de control se algunos artefactos se vuelvan autónomos y fáciles de manipular para personas menos expertas en temas de informática.

CAPÍTULO III

3. DESARROLLO DE INGENIERÍA

3.1 Diseño del Sistema

El presente proyecto propone el control de dos vehículos de forma simultánea, para lo cual se propone un esquema que permita demostrar el control mediante el uso de dos vehículos no tripulados, un medio de comunicación y un dispositivo de control.

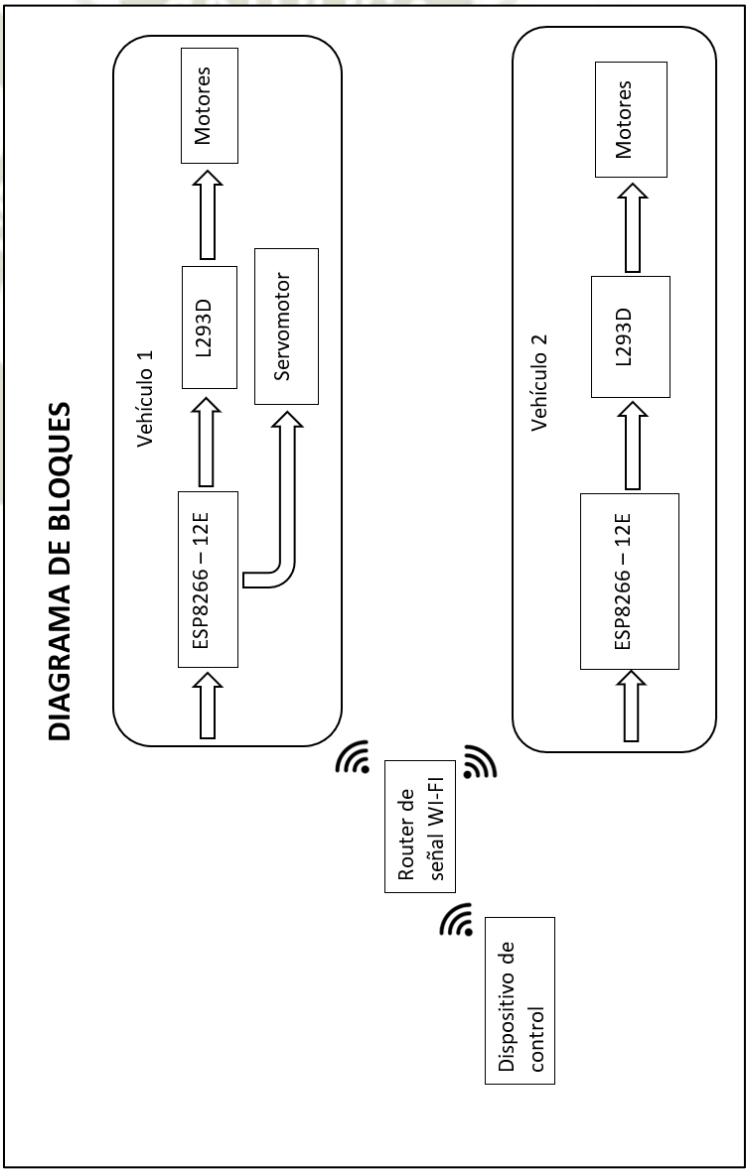


Figura 3.1 Diagrama de bloques
Fuente: Propia

En el diagrama de bloques podemos apreciar como estableceremos la comunicación entre el maestro y los esclavos.

El dispositivo maestro se comunicará con los esclavos mediante una señal inalámbrica emitida por un dispositivo que genere una red de comunicación (router), los vehículos recibirán las instrucciones y solo se comunicarán con el maestro.

Al recibir las ordenes los vehículos ejecutaran su programa y ejecutando los movimientos necesarios mediante los diferentes actuadores; el funcionamiento será de lazo abierto, es decir que no tendrá sensores que retroalimenten su programa, esto debido a que el objetivo de la tesis es el control de estos vehículos y no el comportamiento de estos.

3.2 Microcontrolador

Para el desarrollo del proyecto se consideró diversos microcontroladores que pudieran realizar el objetivo planteado, se tenía la necesidad que cumpliera con los siguientes requisitos:

- Procesador de 32 bit
- Comunicación inalámbrica
- Entradas y salidas digitales
- Salidas analógicas o pwm
- Tamaño ligero

Se realizaron pruebas de comunicación y cumplimiento de órdenes con combinaciones usando módulos de bluetooth con tarjetas Arduino, se consideró los módulos rc para la trasmisión de órdenes y módulos de comunicación wifi.

Al finalizar las pruebas de funcionamiento se llegó a la conclusión que la tarjeta que mejor cumplía con los requisitos necesarios para la tarea era la ESP8266.

La tarjeta ESP8266 es muy versátil, de lenguaje sencillo de programar, cuenta con una antena wifi que nos permite conectarnos a una red inalámbrica de nuestra elección, también podemos conectarnos a la red de internet mediante una página web (WI-FI Alliance).

Este microcontrolador se encuentra contenido en una tarjeta es el Tensilica L106 , de 32-bits con arquitectura RISC que funciona a una velocidad de 80Mhz, con una velocidad máxima de 160Mhz..

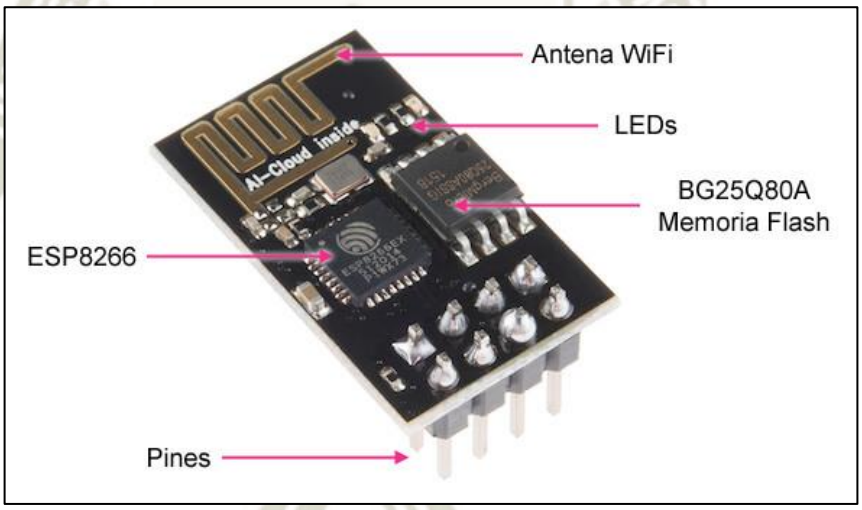


Figura 3.2 Tarjeta ESP8266
Fuente: https://programarfacil.com/podcast/esp8266-wifi-coste-arduino/

La variación de la tarjeta ESP8266 que se usara es el ESP8266-12E que cuenta con 12 pines programables para ser usados de entrada o salida, algunos como salida pwm y una memoria de tamaño adecuado para su programación. Algunas de las características por las que se escogió este microcontrolador son:

- Protocolo: 802.11 b/g/n (HT20)
- Frequency Range: 2.4G ~ 2.5G (2400M ~ 2483.5M)
- CPU Tensilica L106 32-bit processor
- Peripheral Interface UART/SDIO/SPI/I2C/I2S/IR Remote Control

  GPIO/ADC/PWM/LED Light & Button
- Operating Voltage: 3.0V ~ 3.6V
- Operating Current Average value: 80 mA
- Operating Temperature Range: –40°C ~ 125°C
- Wi-Fi Mode: Station/SoftAP/SoftAP+Station
- Network Protocols: IPv4, TCP/UDP/HTTP
- User Configuration: AT Instruction Set, Cloud Server, Android/iOS App

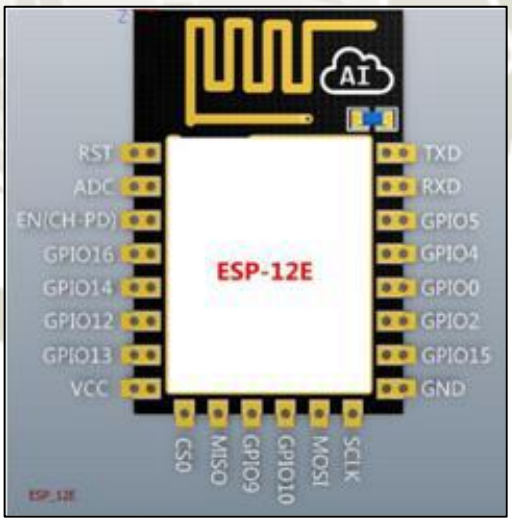


Figura 3.3 Tarjeta ESP8266 – 12E
Fuente: http://www.ai-thinker.com

En el anexo E, se encuentra el Datasheet del ESP8266-12E

3.3 Actuadores

Para la demostración del buen funcionamiento del proyecto debemos mostrar la comunicación lograda mediante acciones, para esto necesitamos usar motores que nos permitan ser controlados con eficiencia, estos motores de preferencia deben ser de corriente continua y como el fin es demostrar la comunicación simultánea los motores pueden ser de pequeños.

Se necesita un buen control de los motores por lo cual usaremos un controlador que nos permita suministrar energía externa a la del microcontrolador ya que esta es muy baja.

Después de una investigación se encontró que los motorreductores dc de 3v a 12v contienen las características necesarias para el proyecto y que se pueden adquirir fácilmente en el mercado.

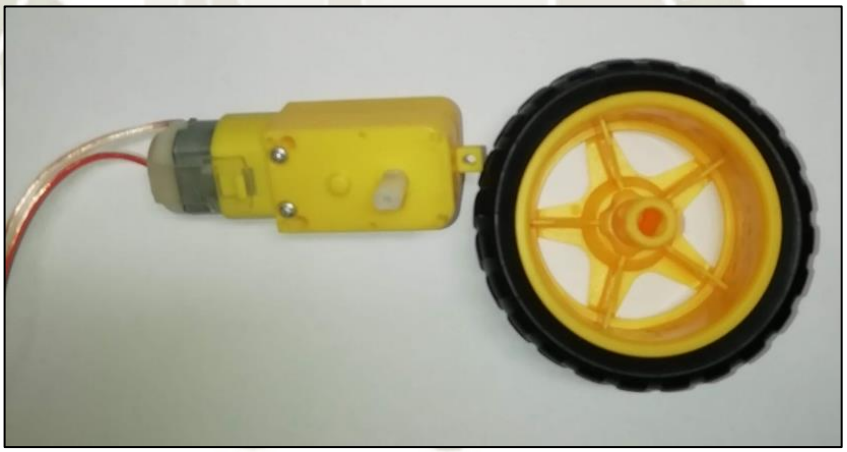


Figura 3.4 Motorreductor
Fuente: Propia

Adicional a los motores se consideró un driver que nos permitiera aprovechar al máximo las características del motor elegido, se seleccionaron dos drivers el L293D y el L293B; ambos de características similares que permitirán junto al microcontrolador ESP8266-12E obtener un control correcto del funcionamiento de los motores.

Figura 3.5 Driver L293D
Fuente: Propia

Para demostrar la versatilidad del microcontrolador se usará en uno de los vehículos un servomotor que permita maniobras más complejas, para la demostración se optó por el servomotor SG90 debido a que sus características de relación de fuerza, peso, tamaño y voltaje de alimentación cumplen adecuadamente con lo requerido en el proyecto.

El servomotor SG90 tiene una rotación de 180 grados, un torque de 2.5 KgCm, una velocidad de 0.1s y un voltaje de funcionamiento entre 4.8-6 voltios.

Figura 3.6 Servomotor SG90
Fuente: Propia

3.4 Programa de vehículos

Para el programa que los vehículos desarrollaran, se planteó que estos tengan una respuesta a las órdenes que el dispositivo de mando proporcione, con el fin de comprobar la buena comunicación, solo se usaran actuadores en los vehículos, por lo que el programa solo recibirá información del dispositivo y actuara según las ordenes enviadas sin ningún sensor externo.

El programa contara con un envió de respuesta hacia el dispositivo de mando por el mismo canal que recibe las ordenes con el propósito de comprobar la recepción de las órdenes.

Para probar una variedad de los usaremos dos tipos de vehículos, el primero solo tendrá motores que permitan su movimiento y será guiado por una rueda.

El segundo vehículo será dirigido por un servomotor, que realizará las funciones de un timón para dos ruedas delanteras con las que se dará la dirección al vehículo.

Al momento de programar los vehículos, se debe tener en consideración los actuadores que tendrán y como deben ser las órdenes para realizar sus movimientos.

Los vehículos tendrán la posibilidad de realizar secuencias de movimiento preestablecidas, para esto el dispositivo de control tendrá la posibilidad de elegir una de estas secuencias que según sean los vehículos, estarán establecidas.

A continuación, presentaremos los diagramas de flujo para cada vehículo.

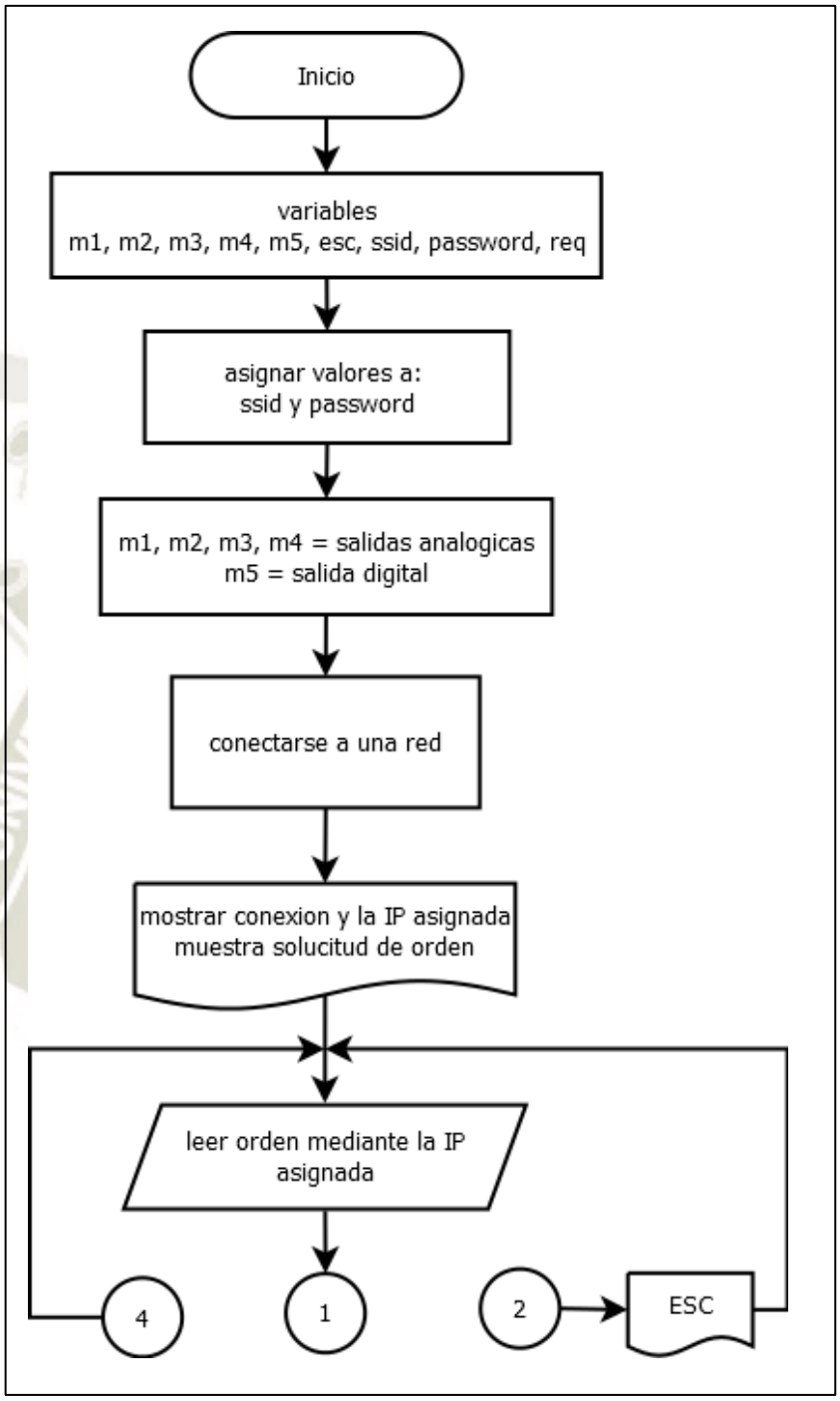


Figura 3.7 Diagrama de flujo vehículo 1 parte 1
Fuente: Propia

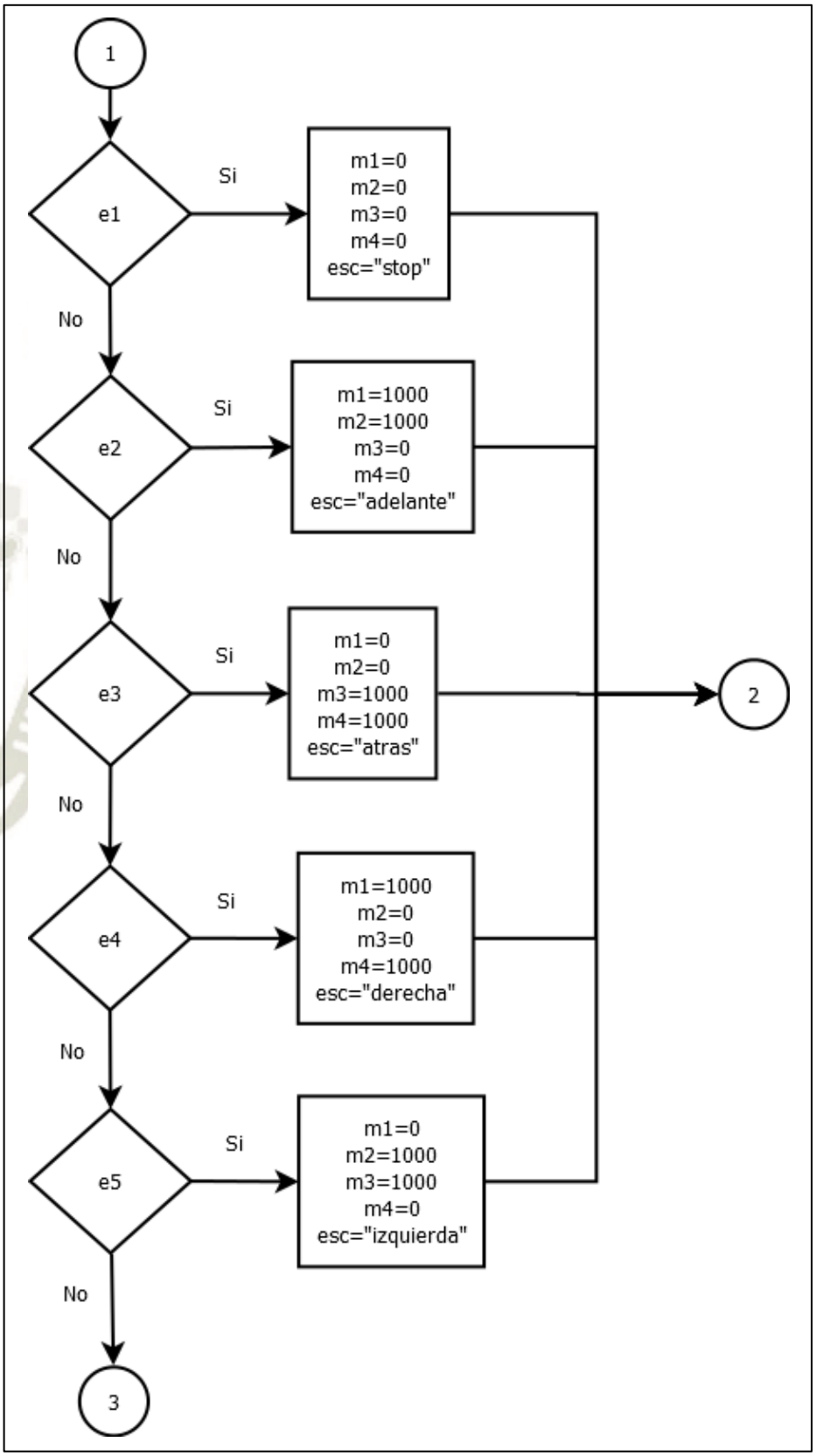


Figura 3.8 Diagrama de flujo vehículo 1 parte 2
Fuente: Propia

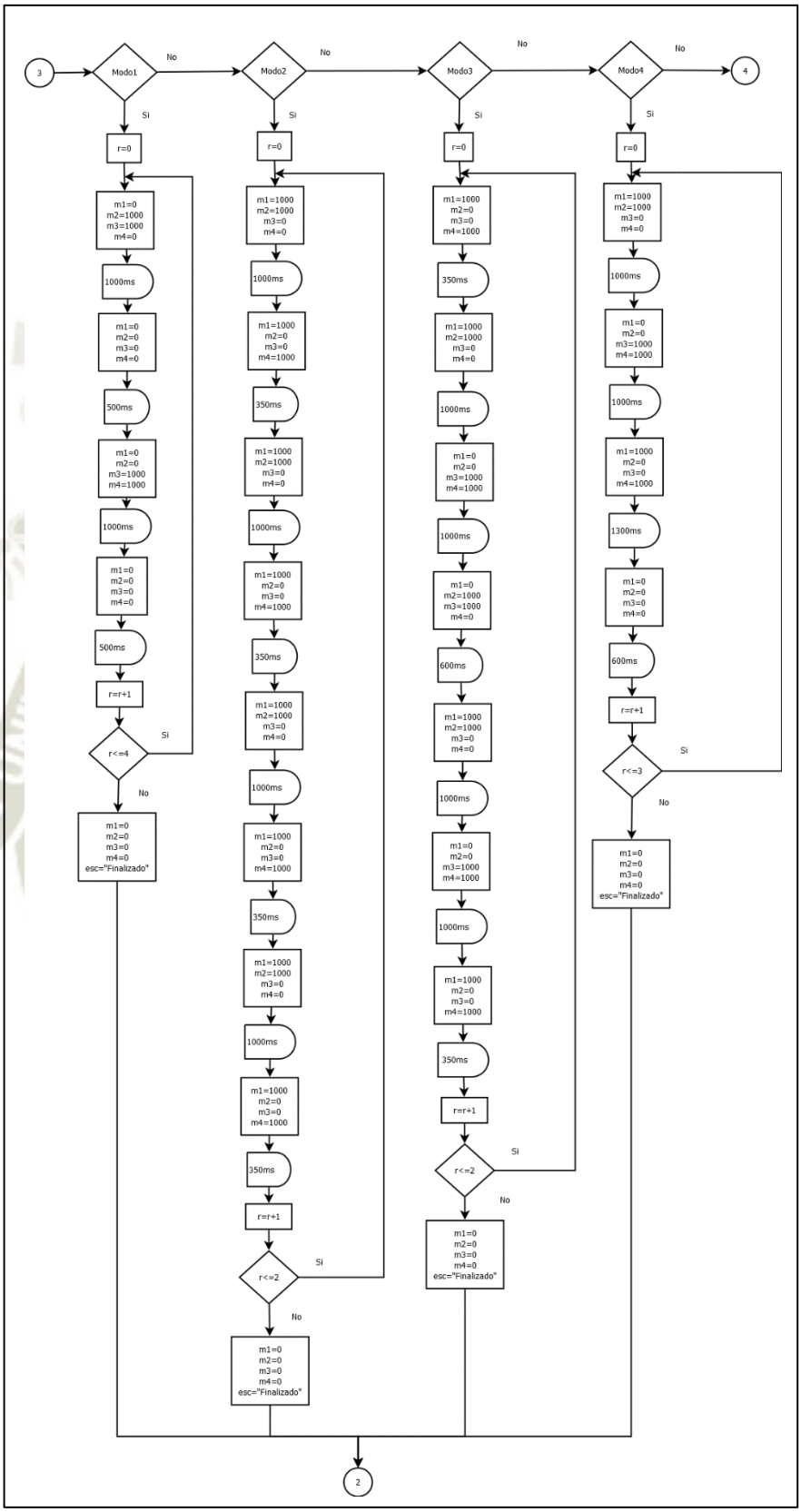


Figura 3.9 Diagrama de flujo vehículo 1 parte 3
Fuente: Propia

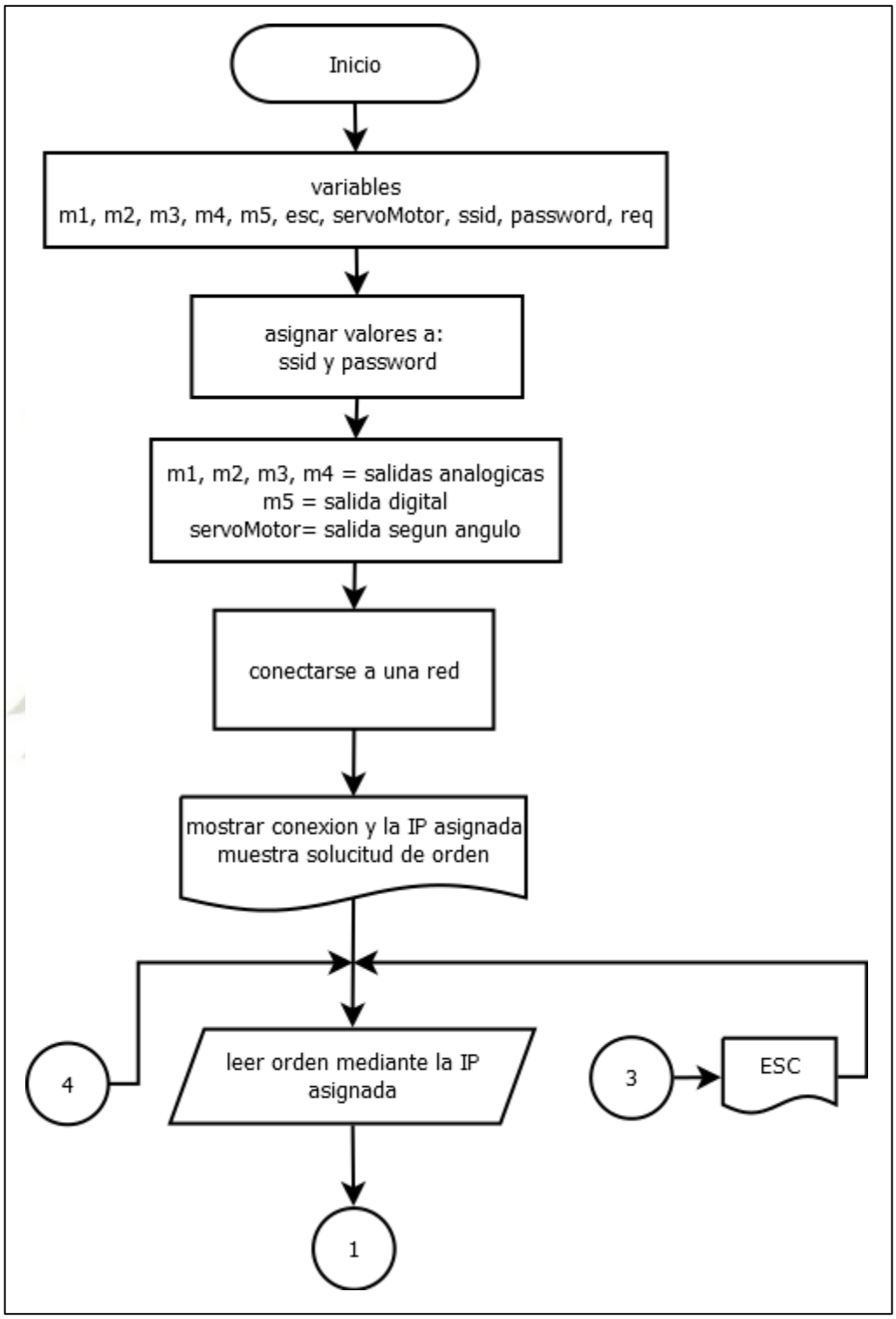


Figura 3.10 Diagrama de flujo vehículo 2 parte 1
Fuente: Propia

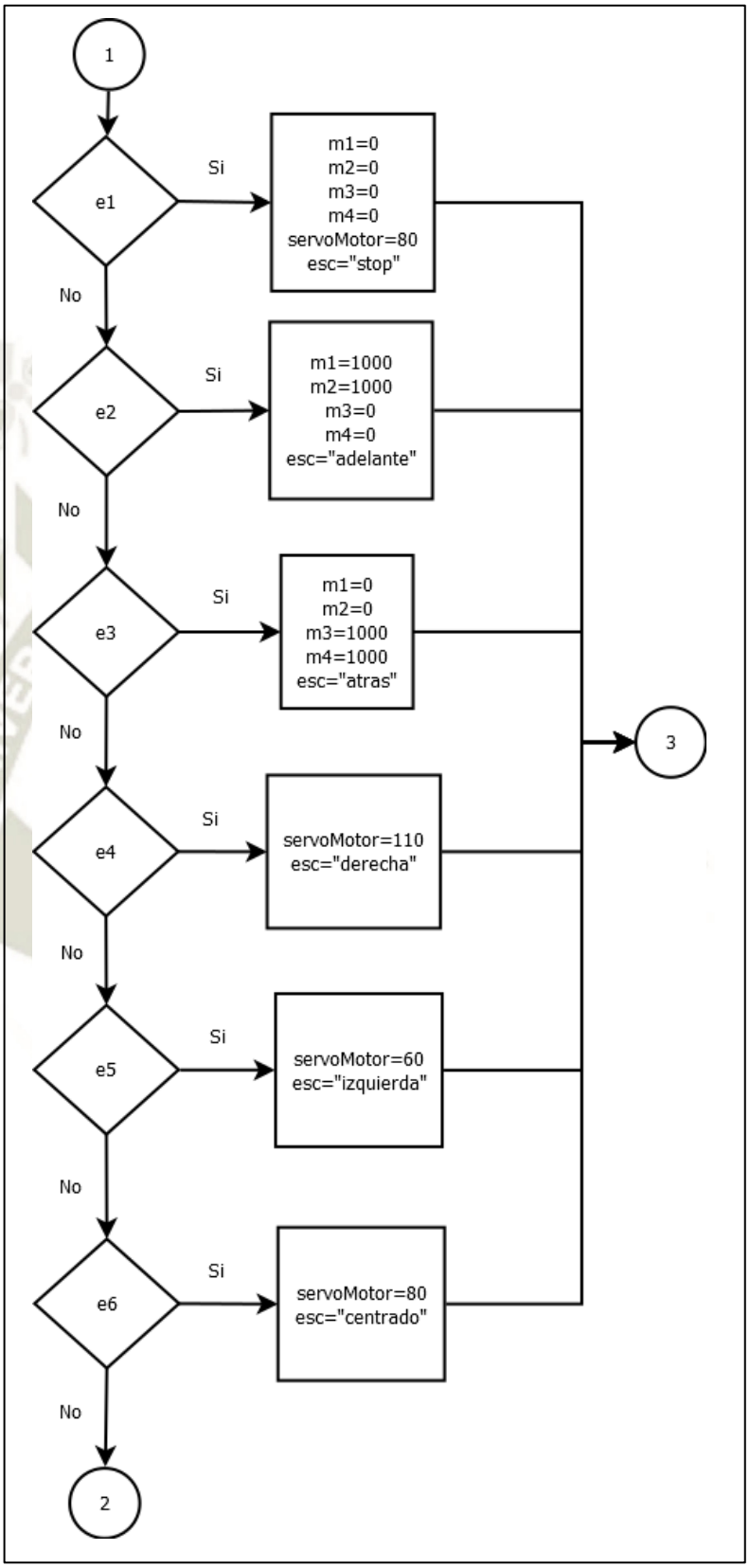


Figura 3.11 Diagrama de flujo vehículo 2 parte 2
Fuente: Propia

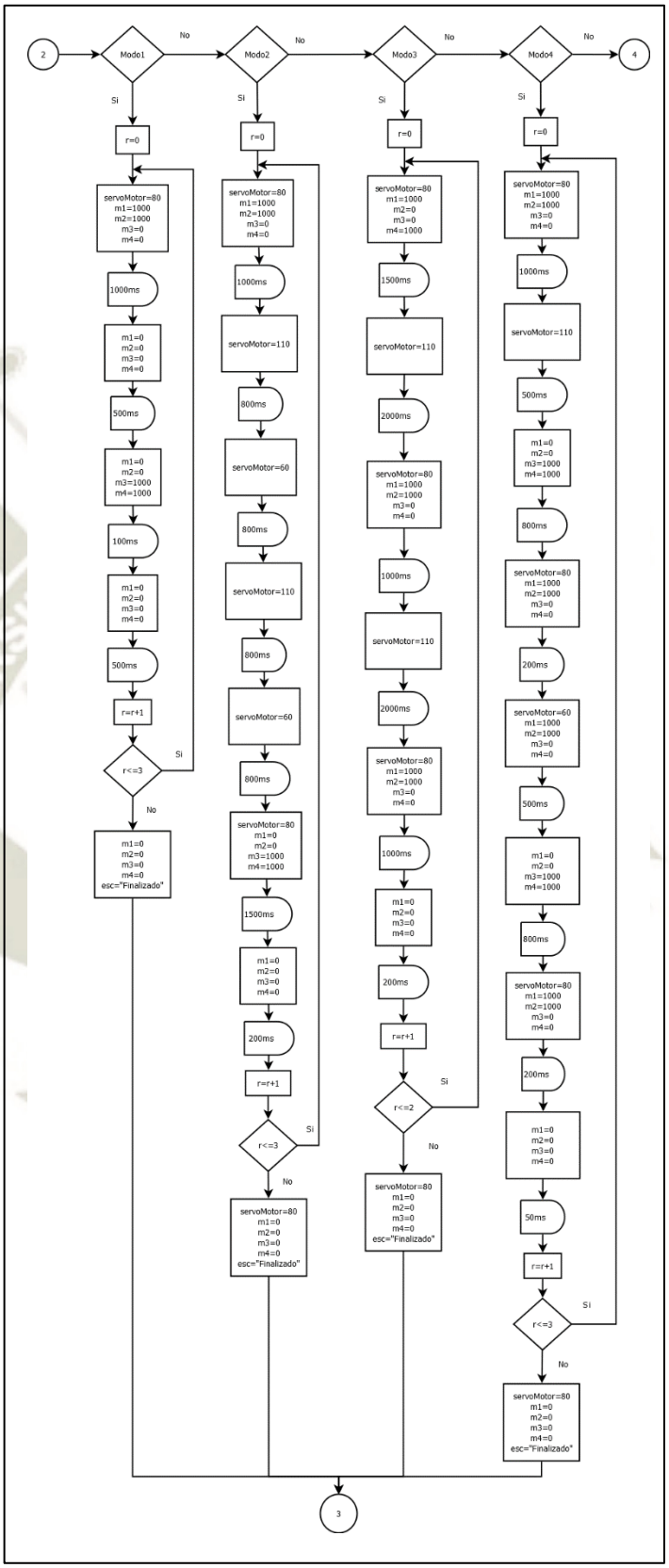


Figura 3.12 Diagrama de flujo vehículo 2 parte 3
Fuente: Propia

En el diagrama de flujo del vehículo 1, se declaran las variables m1, m2, m3, m4, m5, esc, ssid, password y req.

En variable ssid se colocará el nombre de la red a la que nos conectaremos y en la variable password colocaremos la clave de la red.

Las variables m1, m2, m3 y m4 serán usadas para la dar el mando al controlador de los motores, por eso se los declarara como salidas analógicas (pwm).

La variable m5 la usaremos como led indicador para saber si el vehículo se conectó a una red, la variable esc servirá para colocar el mensaje que el vehículo enviara al dispositivo de mando y la variable req será la que tomara el valor enviado por el dispositivo de mando para que el programa pueda ordenar a los actuadores.

Para que el programa comience sus funciones debe conectarse a una red, para esto usara las variables ssid y password.

Al empezar a correr el programa, este se conectará a una red y nos mostrara la ip por el cual le daremos ordenes, esta ip será constante cada vez que la tarjeta se conecte a la red, se debe anotar para usarlo cada vez que se ponga en funcionamiento el vehículo.

Una vez que el programa esté conectado a una red, esperará a que se le dé una orden, el programa recibirá una orden del dispositivo de mando y mediante un sistema de condicionales “If” el programa responderá a las órdenes.

Cada opción que pueda ser seleccionada dará una señal al controlador de los motores para realizar los movimientos, estos movimientos serán cinco, el paro total, adelante, atrás, derecha e izquierda.

Los vehículos cuentan con cuatro modos automáticos, estos se pueden seleccionar, al empezar alguno de estos, deberá terminar toda su secuencia para poder ejecutar otra orden.

Después de ejecutar una de las opciones, el programa esperara recibir la siguiente orden.

En el segundo vehículo tenemos un programa similar, con la diferencia que se usar un servomotor, para el control del servo usaremos una variable más que será "servoMotor" esta variable dará la señal para poder mover el servo los grados necesarios para los distintos movimientos.

Tendremos una opción más que nos permitirá centrar el servo, y los movimientos del segundo vehículo dependerán de la posición del servo.

3.5 Software de programación

Después de seleccionar el microcontrolador que usaremos para desarrollar el sistema de comunicación, debemos escoger un programa que nos permita desarrollar un programa y quemar este en la memoria para poder ser ejecutado.

Para el microcontrolador ESP8266-12E existen una variedad de Software para poder programarlo de acuerdo a las funciones que deseemos que desarrolle en la tarea indicada.

Entre los diferentes programas de desarrollo Arduino IDE, es una plataforma de código abierto que nos permite realizar programas en líneas de código usando el lenguaje C++ como base, este software se emplea principalmente las tarjetas de desarrollo de la marca Arduino, pero tienen una librería que permite la programación de la tarjeta ESP8266-12E por lo que se convierte en una herramienta útil para el correcto desarrollo del proyecto (Arduino).

Figura 3.13 Arduino IDE
Fuente: https://www.arduino.cc/en/Trademark/CommunityLogo

3.6 Mando de usuario

En la configuración maestro-esclavo, los esclavos reciben órdenes solo del maestro, se comunican solo con este y no entre ellos.

En el proyecto el dispositivo maestro es el dispositivo que el usuario controlara para ordenar a los vehículos que serán los esclavos.

El maestro será un dispositivo móvil, para poder desarrollar nuestros objetivos se requiere una app, la app es como se le conoce a las aplicaciones para dispositivos móviles, con esta app se debe poder controlar los dos vehículos que se propone.

Se desarrollará una aplicación que contendrá diferentes ventanas y botones para el manejo de los vehículos.

Actualmente se tienen diferentes programas que nos permiten desarrollar aplicaciones para una variedad de sistemas operativos móviles, para fines prácticos y por la relevancia en el mercado local desarrollaremos una aplicación en el sistema Android.

Para el desarrollo de esta aplicación conocida como app, se optó por usar el programa “MIT App Inventor 2” este es un software de desarrollo que nos permite crear apps para

dispositivos que cuenten con Android en su sistema operativo (Instituto Tecnológico de Massachusetts).

Figura 3.14 Programa de desarrollo de aplicaciones MIT APP Inventor
Fuente: https://www.tuappinvetorandroid.com/2017/08/12/mit-app-inventor-cambia-su-logo/

El programa consta de dos etapas:

La primera dedicada al diseño gráfico de los elementos visibles en la app y todo con lo que podrá interactuar el usuario.

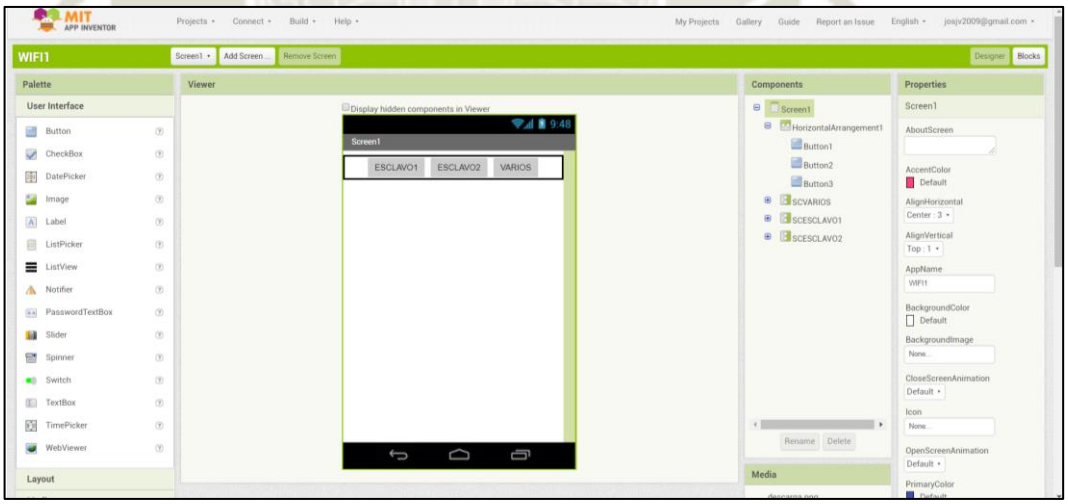


Figura 3.15 Ventana de diseño
Fuente: http://ai2.appinventor.mit.edu

En la segunda tenemos la programación de los elementos colocados en el entorno de trabajo, esta programación se da mediante bloques de programación que representan a las funciones básicas basadas en el lenguaje de programación C+.

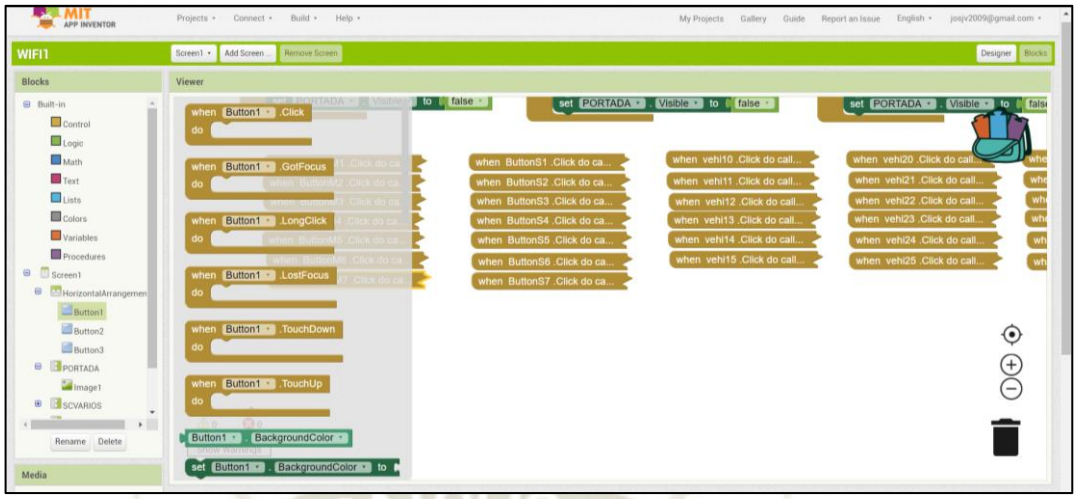


Figura 3.16 Ventana de programación
Fuente: http://ai2.appinventor.mit.edu

3.7 Modulo de prueba

A fin de poder establecer los parámetros que cumplirán los vehículos que se desarrollaran en el cumplimiento de lo planteado en esta tesis, se experimentó en la tarjeta ESP8266-12E, para comprobar todas sus capacidades se armó un módulo experimental en un protoboard de pruebas, una vez instalado de procedió a comprobar sus distintos modos de uso.

Se enlazo la tarjeta al pc mediante el módulo “Arduino Uno Rev3”, este nos permitió tener una comunicación con la tarjeta y poder programarla para los fines de la tesis.

Para poder probar el funcionamiento de la tarjeta mediante órdenes dadas por el dispositivo de control se necesitó generar una red para que se pudieran comunicar, esta red debe ser una red wifi, es decir que cumpla con el protocolo de comunicación, no es necesario que la red se conecte a internet, ya que lo que se necesita es que los dispositivos se puedan comunicar, la red se generó con un router.

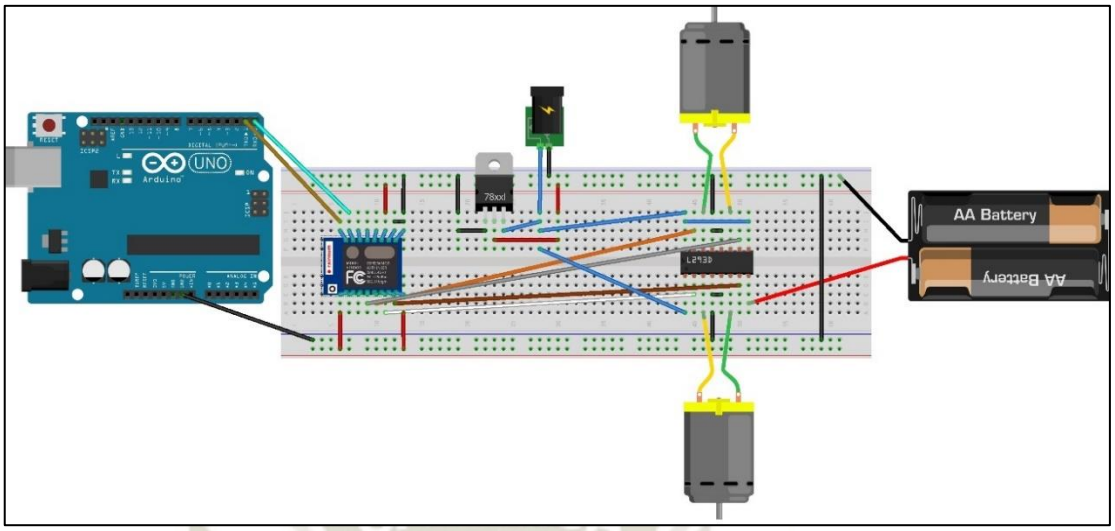


Figura 3.17 Esquema de módulo de pruebas.
Fuente: Propia

Una vez que se instaló la conexión de los componentes necesarios para el funcionamiento del proyecto se empezó a realizar las pruebas necesarias para llegar a nuestros objetivos. Las pruebas nos permiten observar el comportamiento de los diferentes componentes, sus tiempos de repuesta, el comportamiento a diferentes estímulos y el consumo de recursos. Por ello se pudo establecer que la alimentación requerirá por la tarjeta de mando que es de bajo consumo de voltaje a diferencia de los motores que con un mayor voltaje alcanzarían su mayor rendimiento, pero la tarjeta requiere un elevado consumo de corriente a diferencia de otras tarjetas de mando.

Estos desbalances en las necesidades de los componentes, provocaban que las baterías externas se desgastaran a rápidamente y algunos componentes no funcionaban correctamente.

Para mejorar el rendimiento de los vehículos se independizó la alimentación de la placa de mando y de los motores encargados de dar la potencia al movimiento.

3.8 Diseño de placa

A fin de tener un control sólido de los vehículos se optó por fabricar una placa donde todos los componentes electrónicos estén correctamente enlazados.

Este componente contendrá el microcontrolador ESP8266-12E, el controlador de motores y otros componentes electrónicos para el correcto funcionamiento del microcontrolador.

Para el diseño de esta placa electrónica usaremos el software de diseño de placas Eagle.

El software contiene librerías con imágenes de componentes electrónicos que permite la creación de placas con dimensiones reales y estandarizadas de los elementos que existen en el mercado.

Para complementar este software se instalaron librerías creadas de domino libre, estas librerías contendrán esquemas de los diferentes componentes que se usaron en el módulo de investigación y se colocaran en la tarjeta final.

Centraremos el diseño en la tarjeta ESP8266-12E

Según las pruebas debemos alimentar a la tarjeta con 3.3v en los pines Vcc y dar un uno lógico en los pines CH-PD y GPIO2.

Se debe poner el pin GND al negativo de la alimentación y el pin GPIO15 a un cero lógico.

Direccionaremos 4 pines hacia el controlador de motores, estos pines deben ser programados como salidas.

El pin GPIO0 permite que la tarjeta entre en modo programación cuando se le da un cero lógico, para poder tener esta opción disponible, se le colora una salida para poder ser usado.

En los demás pines de la tarjeta de dará la opción de ser utilizados, para esto en el diseño se colocará la opción de poder soldar pines macho de salida.

Para obtener el voltaje requerido usaremos el regulador de voltaje AMS1117, este según su hoja de información nos entregara 3.3 voltios.

Figura 3.18 Regulador de voltaje a 3.3 voltios
Fuente: Propia

El controlador de motores debe tener una estructura de conexión que nos permite controlar el giro de los motores y además nos permita alimentarlos de manera independiente.

En la conexión tendremos una alimentación de 5 voltios en el pin 16, los pines 4,5,12 y 13 serán conectados a tierra, los pines 1 y 9 que son los enables se les pondrá un uno lógico que es aproximadamente 5 voltios, los pines 2, 7, 10 y 15 se conectaran a los pines de mando de la tarjeta ESP8266-12E, los pines 3, 6, 11 y 14 serán las salidas para los motores, esto estarán emparejado los pines 3 y 6 serán para el primer motor y los pines11

y 14 para el segundo motor; por ultimo tenemos el pin 8 que será la alimentación con lo que se alimentaran a los motores y este puede variar de 5 a 12 voltios.

Por último, se colocarán diferentes componentes para complementar los controladores como switchs para la energía, capacitores para estabilizar los voltajes, leds que nos señalizarán la energización de los componentes, borneras para la salida de voltaje hacia los motores y diferentes pines libres de la tarjeta ESP8266-12E para su programación y control.

El programa nos permite buscar y colocar los diferentes componentes que usaremos en la placa

Después de tener todos en la pantalla de esquema, procederemos a conectarlos de la manera más conveniente considerando los pines de conexión.

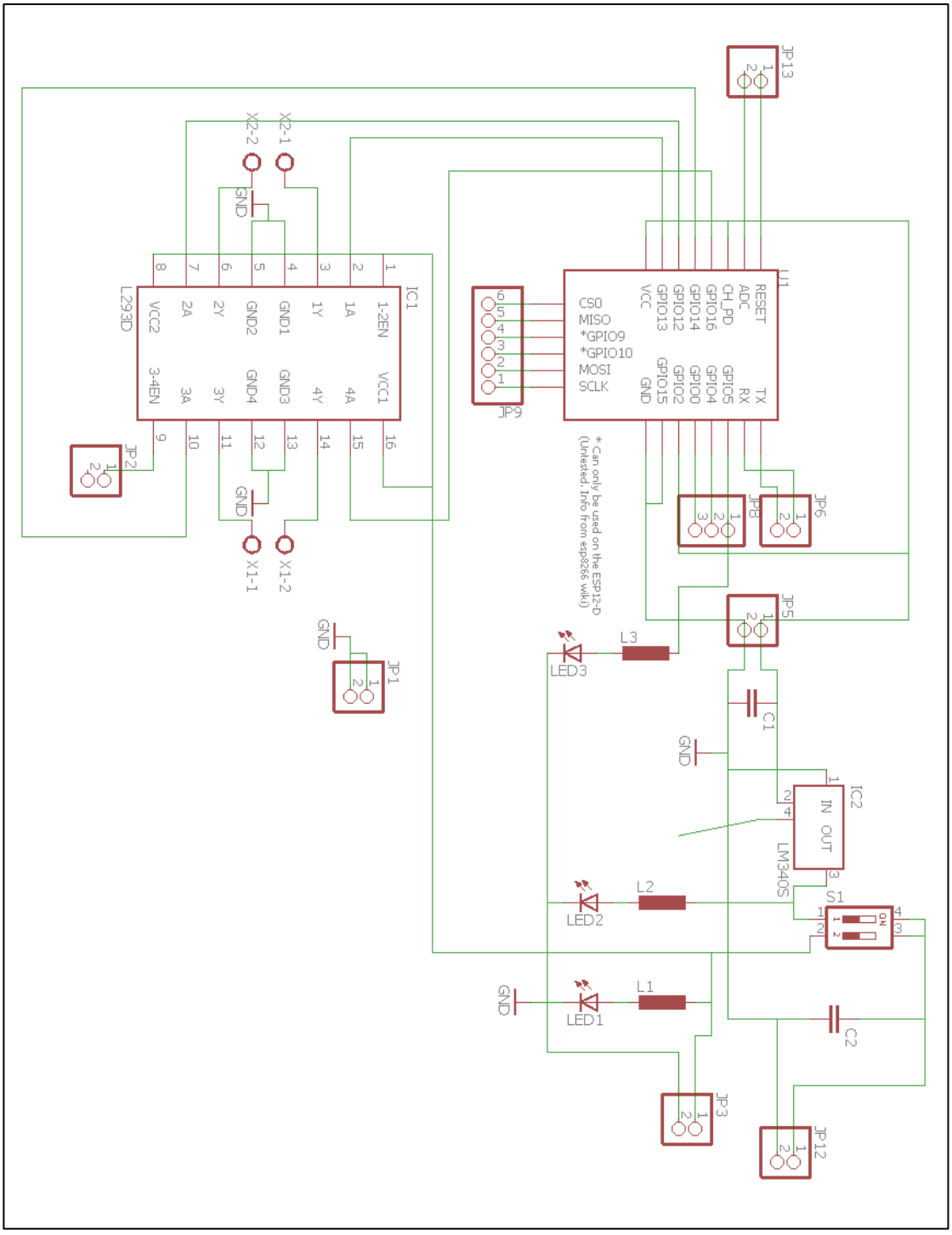


Figura 3.19 Esquema de desarrollo de placa
Fuente: Propia

Una vez conectados pasaremos al tablero donde podremos ver una representación de los componentes en tamaño real, en este espacio distribuiremos los componentes según se vea convenientes, después de distribuirlos pasaremos a dar las rutas de conexión que representaran la pista de cobre en la placa terminada y al final se pueden agregar detalles de acuerdo a los requerimientos.

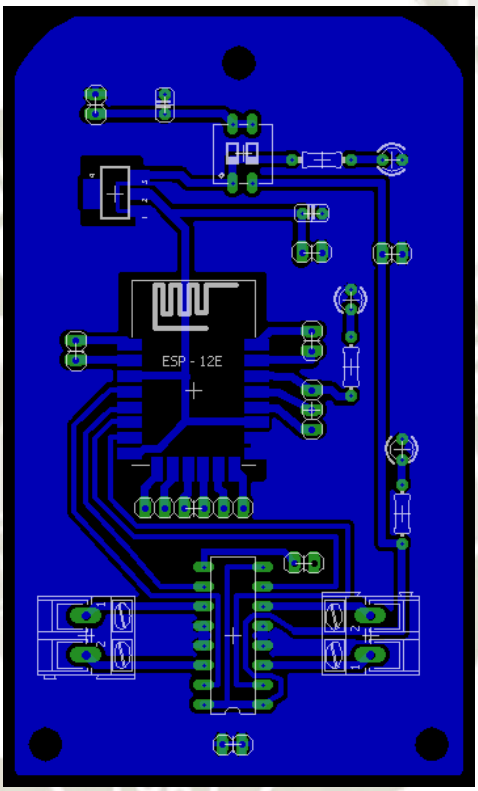


Figura 3.20 Vista de placa con dimensiones de componentes reales
Fuente: Propia

CAPÍTULO IV

4. IMPLEMENTACIÓN

4.1 Vehículos

En la demostración de funcionamiento del sistema de comunicación maestro-esclavo que se plantea en este proyecto, tenemos como opción la utilización de vehículos terrestres, aéreos o acuáticos.

Como nuestra principal tarea es el control simultáneo se decidió el diseño y construcción de vehículos terrestres.

4.1.1 Diseño

De los diversos programas de diseño que se tiene, se decidió usar el programa de diseño "Inventor", este programa de nos permite usar diferentes herramientas para el dibujo de los elementos y su extensión en compatible con diversos programas.

El formato de este software nos permite una buena interacción con equipos de uso comercial como las cortadoras laser y cnc en el mercado.

Figura 4.1 Logo Software Autodesk Inventor
Fuente: https://www.imagine-dsuk.com/wp-content/uploads/2017/05/Autodesk-Inventor-Logo.jpg

Un primer vehículo contará con un controlador puente h para los motores, de una rueda universal que adaptará su movimiento según el giro de los motores, esto implicará que los giros y el control del vehículo dependerá de las órdenes que se dé a los motores.

El armazón del vehículo debe contar con espacio para sostener los motores, la placa y el soporte para la alimentación.

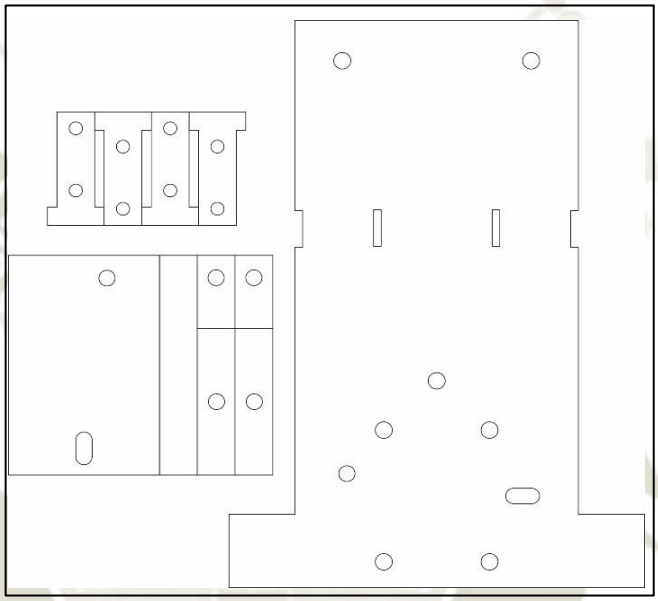


Figura 4.2 Diseño de piezas del vehículo 1
Fuente: Propia

En el segundo vehículo tendremos la fuerza dada a los motores mediante al controlador puente h y se incorporara un servomotor que direccionara el destino del vehículo.

Se probará la versatilidad de la tarjeta escogida al controlar el sentido de giro de los motores y el mando al servo de dirección.

La distribución de los elementos considerara los motores, el espacio para la placa, el soporte del servomotor, las ruedas delanteras conectadas al servomotor y el soporte para la alimentación.

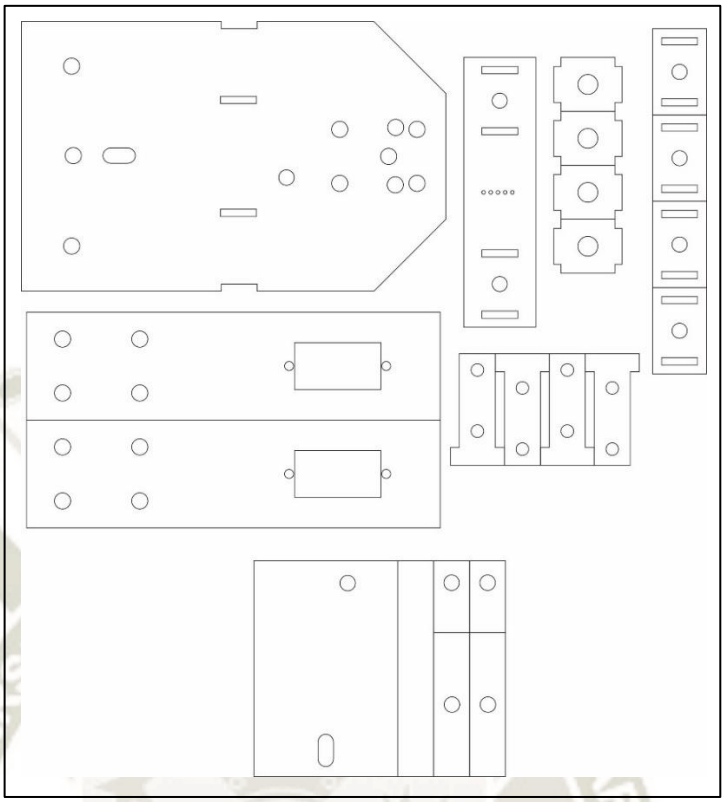


Figura 4.3 Diseño de piezas del vehículo 2
Fuente: Propia

4.1.2 Construcción

Para la construcción de los vehículos de prueba el material que se usara es el acrílico de 2 mm de espesor, este material posee las características más convenientes para el desarrollo de pruebas, tiene una buena rigidez y es ligero.

Una vez terminado el diseño en computadora, pasaremos la plancha de acrílico a una cortadora laser, esta máquina nos da la precisión esperada en la elaboración de cada pieza diseña para los vehículos, esto es necesario para poder ensamblar los elementos correctamente.

El diseño debe ser preciso, esto debido a que la cortadora laser tiene poco margen de error y las dimensiones que nosotros designemos a cada parte en el diseño computarizado será la dimensión real que la maquina nos otorgará.

Figura 4.4 Cortadora laser
Fuente: Propia

El primer vehículo consta de una placa principal que sostiene los componentes del vehículo, se consideró las dimensiones de los actuadores y los elementos de sujeción que se utilizaran.

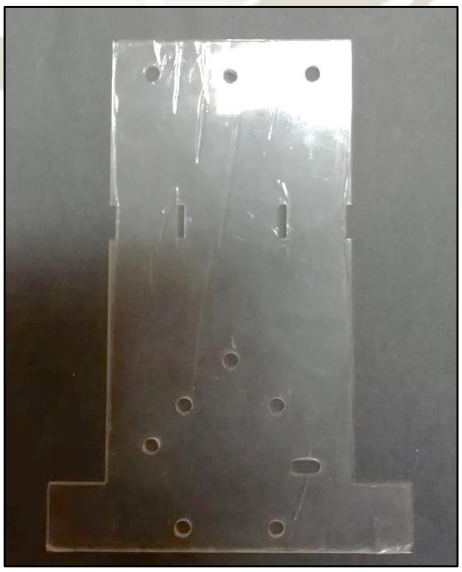


Figura 4.5 Placa principal del vehículo 1
Fuente: Propia

Los motores son alineados con soportes que encajaran en las ranuras de la placa principal y serán sujetados con pernos para que se mantengan rígidos.

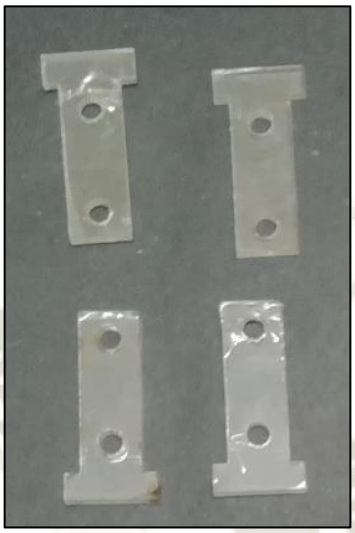


Figura 4.6 Soportes de motor
Fuente: Propia

En la parte delantera se colocará una rueda universal o rueda loca, para soporte y dirección del vehículo.

Figura 4.7 Rueda universal
Fuente: Propia

Por encima de la rueda universal colocaremos el soporte de la batería que alimentará a placa.

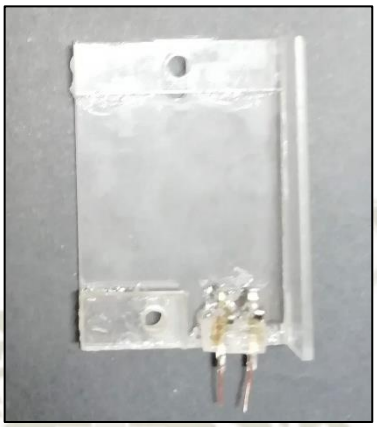


Figura 4.8 Soporte para batería de litio
Fuente: Propia

Debajo del vehículo en la parte posterior colocaremos el soporte para la batería que alimentará los motores.

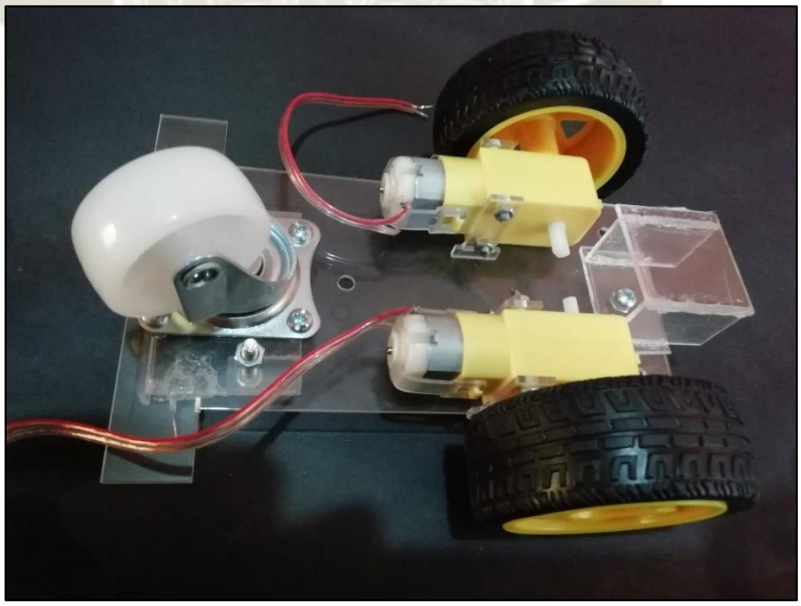


Figura 4.9 Ensamble Vehículo 1
Fuente: Propia

En el segundo vehículo el chasis estará compuesto de cuatro elementos que darán forma y rigidez al vehículo.

En el primer elemento se colocarán los motores, la batería de alimentación para la placa y la batería de alimentación para los motores.

Figura 4.10 Placa principal del vehículo 2
Fuente: Propia

El segundo elemento está compuesto por dos placas que formaran un brazo, este sostendrá el servomotor y dará espacio para el giro de las ruedas.

Figura 4.11 Placas de soporte del vehículo 2
Fuente: Propia

El tercer elemento de sostendrá en la salida del servomotor y mediante unas estructuras sostendrá dos ruedas para dar estabilidad y dirección al vehículo.

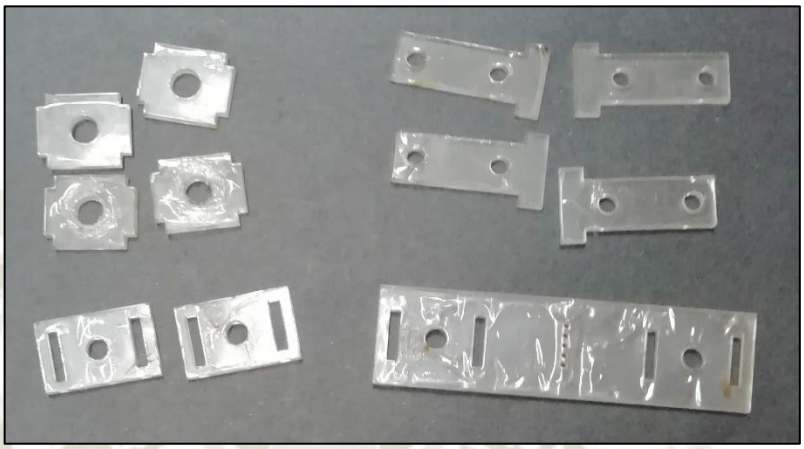


Figura 4.12 Soporte de motores y piezas del soporte direccional
Fuente: Propia

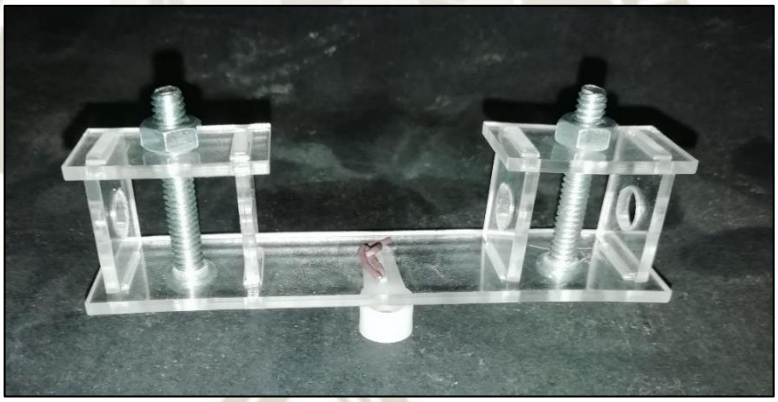


Figura 4.13 Soporte de ruedas direccionales
Fuente: Propia

Los componentes son sostenidos mediante pernos para lograr una sujeción mecánica fuerte que nos permita regular distancias para lograr los objetivos.

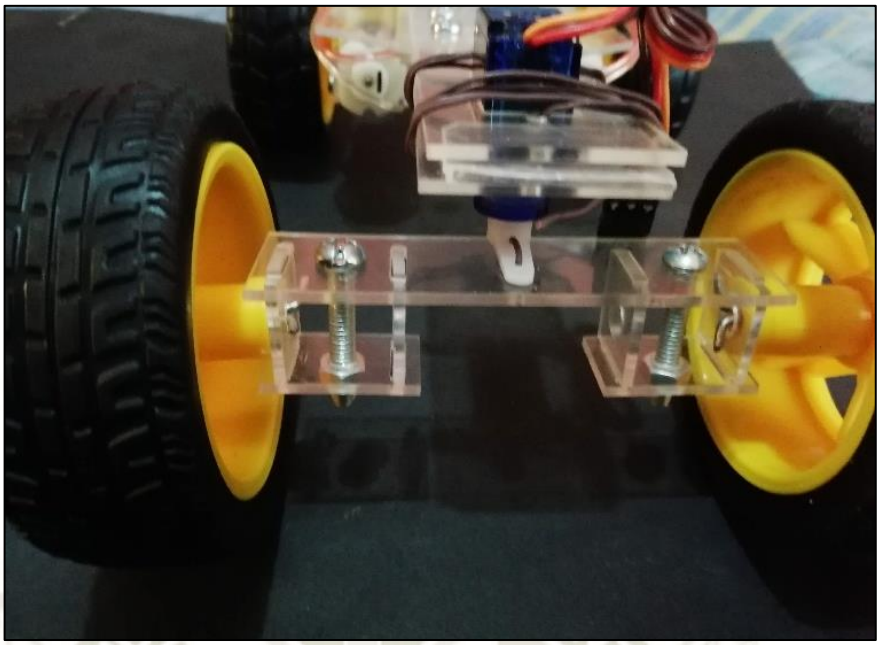


Figura 4.14 Ensamble de direccional con servomotor
Fuente: Propia

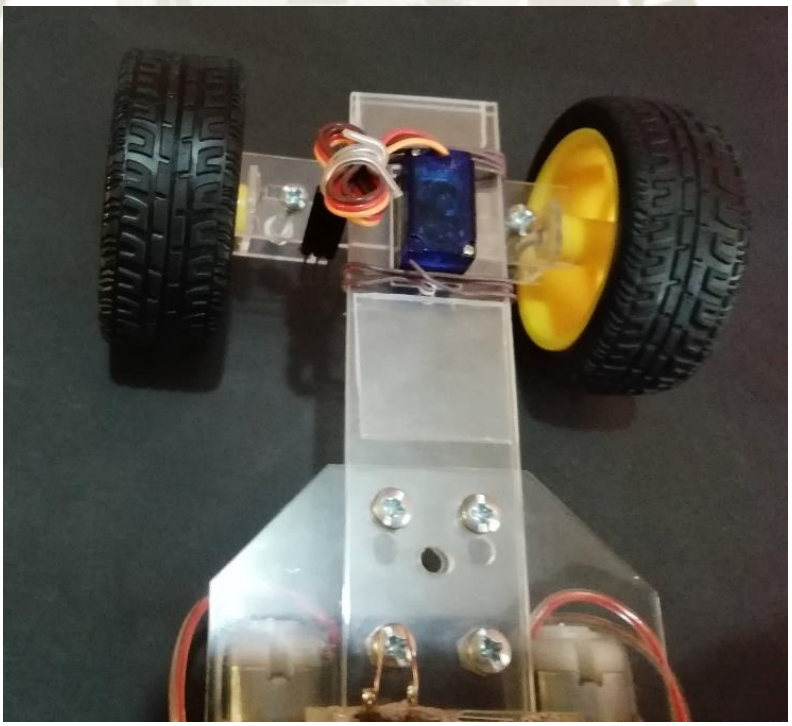


Figura 4.15 Conexión de brazo y dirección con placa principal
Fuente: Propia

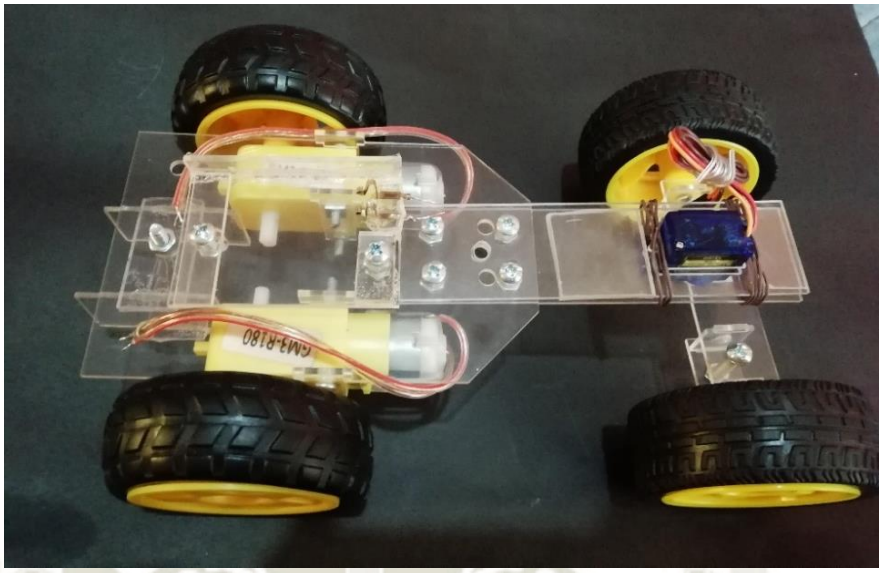


Figura 4.16 Vehículo 2 ensamblado
Fuente: Propia

4.2 Construcción de placa de mando

Una vez que tengamos el diseño de la placa de mando, pasaremos a la construcción de esta para poder ensamblarla a los vehículos.

En la elaboración del circuito impreso usaremos una placa de baquelita para electrónica, esta consta de una capa de cobre en una de sus caras.

Después de tener listo el diseño de la placa en el programa Eagle procederemos a imprimir la imagen con una impreso laser en un papel brillante para que podamos pasar la imagen a la placa de cobre.

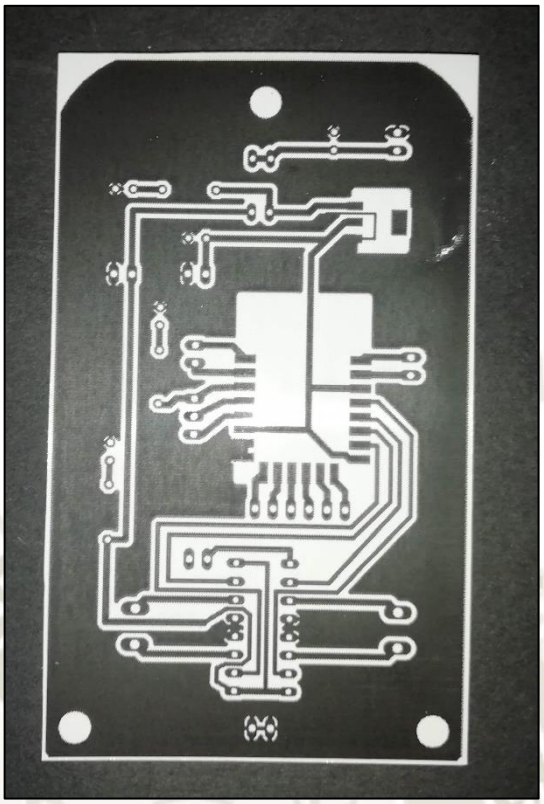


Figura 4.17 Impresión en papel fotográfico de la placa
Fuente: Propia

Una vez impreso el circuito procederemos a trabajar la placa de cobre, primeramente colocaremos la cara impresa del circuito sobre la placa de cobre, seguidamente se procede a aplicar calor para que el circuito impreso se pegue a la placa de cobre, el calor se aplica mediante una plancha caliente, después de una prolongada exposición a la temperatura alta, el polvo fino de la impresión laser estará impregnado en la cara de cobre, se debe revidar que todas las pistas estén bien definidas en la placa, si no han quedado bien marcadas, debemos completarlo con un marcador de tinta permanente.

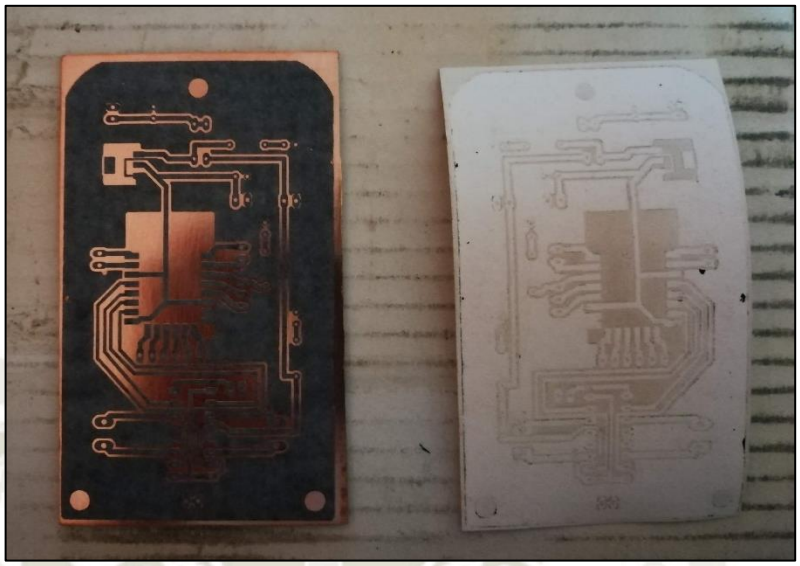


Figura 4.18 Placa de cobre impregnada del polvo de impresión por exposición a alta temperatura
Fuente: Propia

Cuando el circuito ha quedado correctamente impregnado en la placa de cobre, procederemos a darle un baño en ácido férrico, este disolverá todo el cobre que no esté protegido ya sea por el polvo fino o por la tinta indeleble.

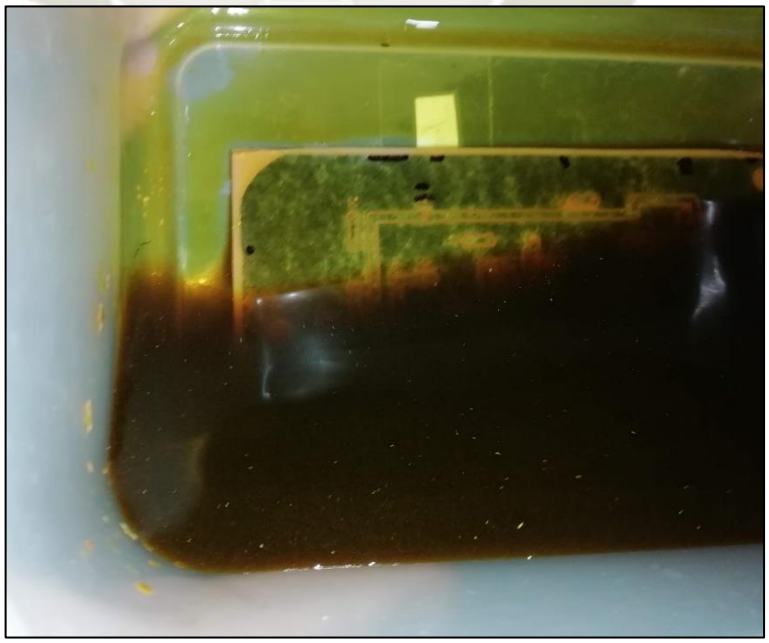


Figura 4.19 Placa sumergida a ácido férrico
Fuente: Propia

Al retirar la placa del ácido férrico debemos revidar que el cobre en las partes no cubiertas ha sido removido en su totalidad, pasaremos a limpiar la placa con una esponja de metal removiendo la protección de la placa para exponer el cobre no disuelto.

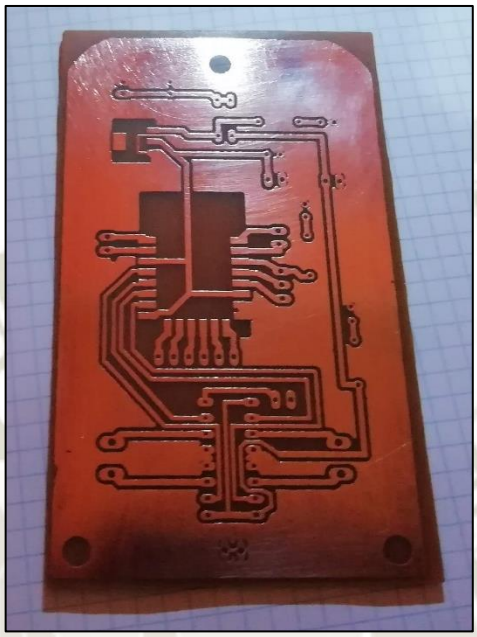


Figura 4.20 Placa limpia después del baño en ácido
Fuente: Propia

Con el circuito impreso en cobre, procederemos perforar los agujeros donde encajaran los componentes electrónicos, se debe tener en cuenta el tamaño necesario de los agujeros para los componentes y respetar las distancias establecidas en el diseño de la tarjeta.

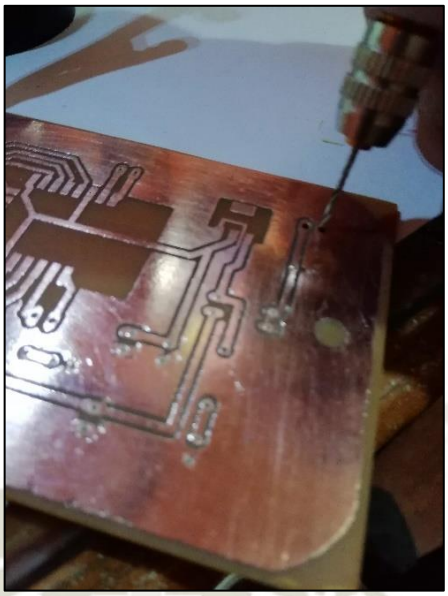


Figura 4.21 Perforación de placa
Fuente: Propia

Para una mejor presentación y ubicación de los componentes se pegó un esquema de la placa en la cara de la baquelita que no tiene cobre.

El esquema nos permite la colocación correcta de los componentes y nos permite apreciar las conexiones que se establecieron para funcionamiento deseado en los componentes.

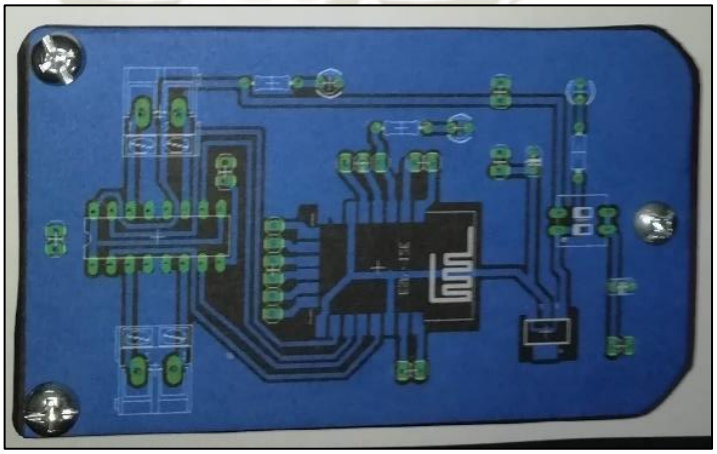


Figura 4.22 Placa con vistas de representación de componentes
Fuente: Propia

Se procedió a soldar cada uno de los componentes según lo establecido en el diseño de la placa, que se basó en el módulo de pruebas, para esto se utilizó estaño y un cautín tipo lápiz.

Se debe tomar en cuenta que las tarjetas ESP8266-12E y el regulador de voltaje LM1117 son encapsulados que se sueldan directamente sobre la placa de cobre.

Figura 4.23 Soldadura de componentes a la placa
Fuente: Propia

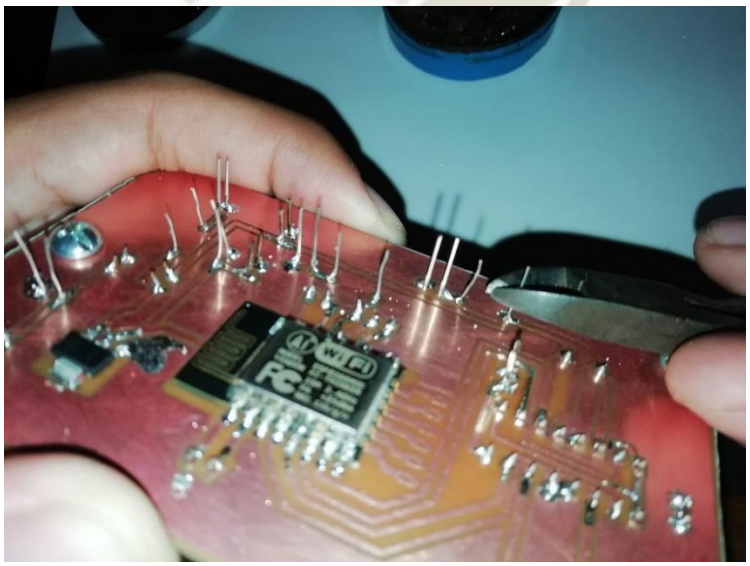


Figura 4.24 Eliminación de sobrantes en la soldadura
Fuente: Propia

Placa terminada serán sujetada con pernos a los vehículos construidos, es por ello que se perforo tres agujeros de acuerdo al diseño final de los vehículos ensamblados.

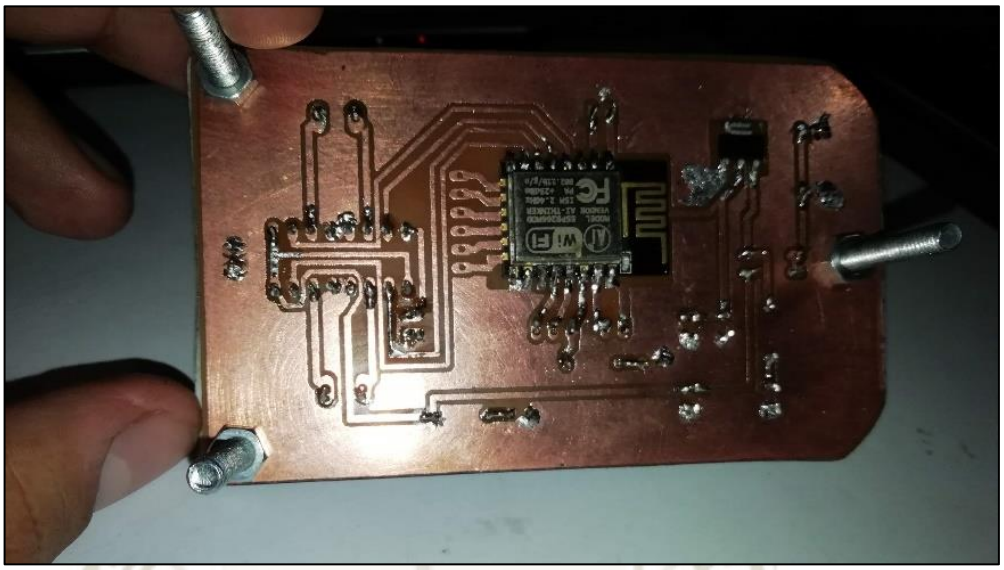


Figura 4.25 Vista inferior de la palca terminada
Fuente: Propia

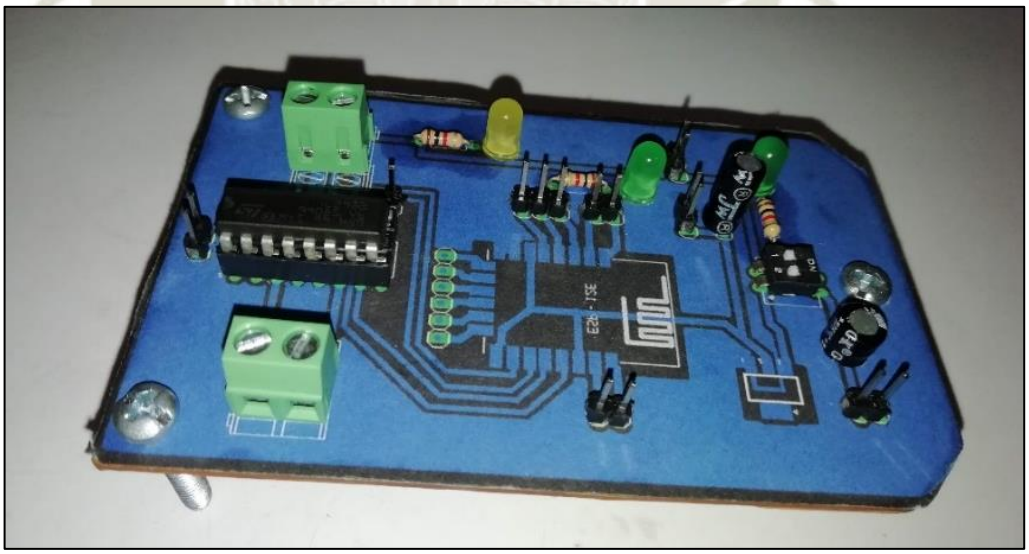


Figura 4.26 Vista superior de la palca terminada
Fuente: Propia

4.3 Programación

El primer paso en la programación es establecer las funciones que se utilizaran, para esto la tarjeta permite utilizar los comandos básicos de las tarjetas Arduino, por ello se programara utilizando los comandos conocidos.

Se utilizó la librería “ESP8266WiFi.h”, esta librería de desarrollo para la programación de la tarjeta ESP8266, esta librería contiene diferentes comandos que nos permiten una mayor rapidez para llamar funciones y poder llegar a cumplir los parámetros establecidos.

A continuación, se declararon las variables que se usarían para facilitar la programación, a estas variables se le asignaron los nombres de los puertos que usaran.

Les daremos la función que cumplirá cada puerto que usaremos para el control de los vehículos, estos puertos deben ser programados como salidas pwm para el control de los motores.

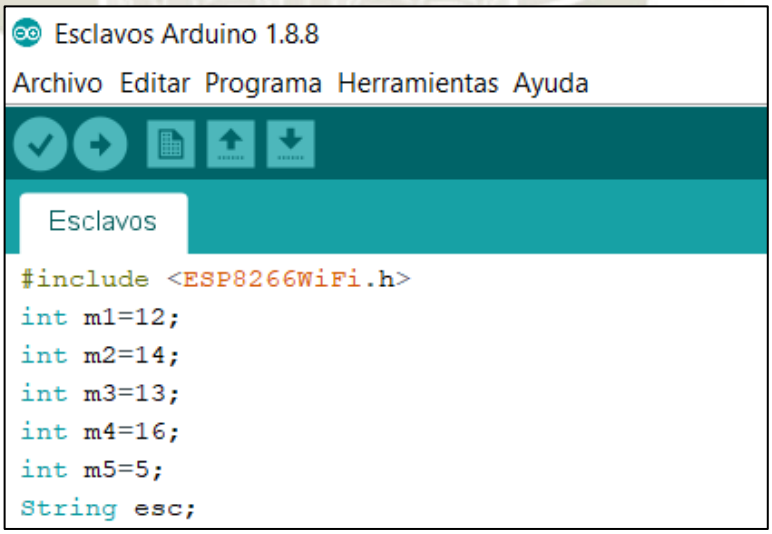


Figura 4.27 Librería y variables
Fuente: Propia

Para conectar la tarjeta a una red, usaremos el comando “wifi.begin()” donde colocaremos el nombre de la red y el password para su conexión.

Escogeremos el puerto por el cual se transmitirá la comunicación, para ello se estableció el puerto 80, ya que es el que se asigna por defecto.

```
int m4=16;
int m5=5;
String esc;
const char* ssid = "HUAWEI";
const char* password = "12345678";
WiFiServer server(80);
void setup()
   {  Serial.begin(115200);
      delay(1000);

      pinMode(m1, OUTPUT);
      analogWrite(m1, 0);
      pinMode(m2, OUTPUT);
      analogWrite(m2, 0);
      pinMode(m3, OUTPUT);
      analogWrite(m3, 0);
      pinMode(m4, OUTPUT);
      analogWrite(m4, 0);
      pinMode(m5, OUTPUT);
      digitalWrite(m5, 0);
```

Figura 4.28 Nombre de la red y determinación de funciones
Fuente: Propia

Una vez establecidos estos parámetros, pasaremos a conectarnos a la red escogida, para esto usaremos el comando “WiFi.status()” que conecta a red seleccionada, esta conexión suele necesitar varios intentos, es por eso que lo introduciremos en un comando “while” y una vez conectado termine la repetición.

Después de ser lograda la conexión, mostraremos la IP que corresponderá a la tarjeta, esto nos permitirá dar las órdenes para el cumplimiento de las funciones.

```
Serial.print("Connecting to ");
Serial.println(ssid);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED)
   {    delay(500);
        Serial.print(".");

   }
Serial.println("WiFi connected");
server.begin();                          // Iniciamos el servidor
Serial.println("Server started");
Serial.println(WiFi.localIP());          // mostramos la IP
digitalWrite(m5,1);
```

Figura 4.29 Secuencia de conexión a red wifi
Fuente: Propia

Ya establecidos las funciones para la conexión a la red wifi, procederemos al programa principal que permitirá el funcionamiento de los vehículos no tripulados.

Pondremos al microcontrolador en estado de pendiente para que este en reposo esperando una orden.

Las órdenes deben ser enviadas por una plataforma web teniendo en cuenta la IP asignada y el puerto de comunicación.

```
if (!client)
     return;
Serial.println("new client");
while(!client.available())
      delay(1);
String req = client.readStringUntil('\r');
Serial.println(req);
client.flush();
```

Figura 4.30 Lectura de datos recibidos
Fuente: Propia

Una vez recibido un dato el programa guardara este dato en una variable y usara esta información para la ejecución de instrucciones de acuerdo a lo requerido para cumplir los requerimientos de la tarea propuesta.

Al terminar una instrucción el programa enviara una respuesta para indicar la acción que se ha cumplido, esta respuesta se mostrara en la misma ubicación de donde se produjo la orden.

El programa termina la orden y se pone en estado de espera para recibir el siguiente dato.

```
if (req.indexOf("e1") != -1){
    analogWrite(m1, 0);
    analogWrite(m2, 0);
    analogWrite(m3, 0);
    analogWrite(m4, 0);
    esc="stop";
 }
```

Figura 4.31 Órdenes dadas en uno de los posibles pedidos
Fuente: Propia

```
String s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!DOCTYPE HTML>\r\n<html>\r\nGPIO<br /> estado ";
s += "<br />";
s += (esc);
s += "</html>\n";

client.print(s);  //
delay(1);
esc="o";
Serial.println("Client disonnected");
}
```

Figura 4.32 Respuesta al finalizar una orden
Fuente: Propia

Uno de los vehículos que se diseñó con el objetivo que en sus movimientos estén incluidos los movimientos de un servomotor, para el control de este usaremos un pin adicional por el cual enviaremos la señal de mando, la plataforma de Arduino nos provee con una

librería especial para el control de un servomotor, esta librería nos permite introducir la posición deseado en grados y trasforma esta información en la señal pwm necesaria para el movimiento del actuador.

```
#include <ESP8266WiFi.h>
#include <Servo.h>
Servo servoMotor;
int m1=12;
int m2=14;
int m3=13;
int m4=16;
int m5=5;
String esc;
const char* ssid = "HUAWEI";
const char* password = "12345678";
WiFiServer server(80);
void setup()
   {  Serial.begin(115200);
      delay(1000);

      pinMode(m1, OUTPUT);
      analogWrite(m1, 0);
      pinMode(m2, OUTPUT);
      analogWrite(m2, 0);
      pinMode(m3, OUTPUT);
      analogWrite(m3, 0);
      pinMode(m4, OUTPUT);
      analogWrite(m4, 0);
      pinMode(m5, OUTPUT);
      digitalWrite(m5, 0);
servoMotor.attach(4);
```

Figura 4.33 Modificaciones para uso de servomotor
Fuente: Propia

4.4 Sistema de mando

4.4.1 Interfaz

Lo primero que debemos crear es el interfaz con la que el usuario podrá interactuar para el control de los vehículos.

El software nos muestra un entorno de trabajo basado en las dimensiones de un smartphone, para poder desarrollar la aplicación tenemos una paleta de componentes en donde se encuentran diferentes elementos que podemos agregar para crear nuestra aplicación.

Se armaron cuatro diferentes ventanas, la primera simplemente de presentación, la segunda y tercera ventana fueron diseñadas para controlar un vehículo diferente cada una; la cuarta ventana contiene elementos que nos permiten controlar hasta tres vehículos de manera simultánea.

Para poder cambiar de ventanas se colocó en la parte superior tres botones, al hacer clic en alguno de estos, se pondrá de color verde y nos mostrará la ventana que se le asigno.

Figura 4.34 Portada principal de la aplicación
Fuente: Propia

En nuestra aplicación el diseño se basó en tres ventanas.

Cada ventana corresponde a un modo de uso para los vehículos, las dos primeras ventanas nos permiten controlar un vehículo con un entorno de mando distinto.

Al oprimir el botón “VEHICULO1”, se activa la primera ventana en donde encontraremos en la parte superior una casilla donde deberemos ingresar la ip correspondiente al vehículo en esta ventana, en el lado izquierdo de esta casilla tendremos un letrero que tienen escrito “IP” para indicar lo que se debe colocar, debajo de esta casilla

tendremos diferentes botones alineados con diferentes distribuciones que realizaremos con los componentes en el Layout ubicado en la paleta de elementos, todos estos botones tendrán un nombre o una imagen que nos indique su función para la conducción del vehículo, debajo de los botones de movimiento tendremos el botón de "STOP" para detener en su totalidad el vehículos y a su costado el botón "AUTO" que dará una instrucción prestablecida al vehículos, esta función se puede seleccionar de las opciones que se pueden desplegar de la lista que se presenta a lado izquierdo del botón.

En la parte inferior de la ventana tendremos un espacio donde nos indicara la respuesta del vehículo a cada instrucción que le demos, para esto se colocara un "WebViewer" que nos mostrara la página asignada al vehículo, del mismo modo al oprimir el botón "VEHICULO2" encontraremos un entorno igual al desarrollado, que a la vez es independiente del primero, esto con la finalidad de poder controlar un segundo vehículo independiente con una ip diferente a la del primero.

Figura 4.35 Diseño final una ventana de control para la aplicación
Fuente: Propia

El tercer botón de nombre "MULTIPLE" nos mostrara una ventana en la cual podremos controlar hasta tres vehículos de manera simultánea, para activar cada control debemos

ingresar la IP en los textbox colocados uno debajo del otro, tendremos la distribución de diferentes botones para el control de cada uno de los vehículos en una solo ventana.

Debajo de cada grupo de botones de control nos mostrara el estado de funcionamiento de cada vehículo con un “WebViewer” para cada vehículo.

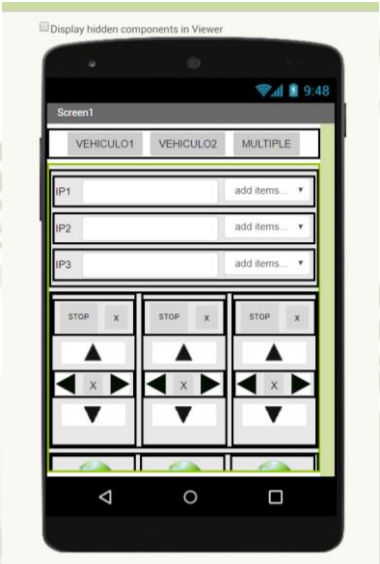


Figura 4.36 Diseño final de la ventana de control múltiple
Fuente: Propia

4.4.2 Programación

Una vez que se acomodaron los elementos en el interfaz del usuario, pasaremos a programar las funciones de cada elemento.

El programa AppInventor2 tiene un sistema de programación mediante bloques, estos bloques contienen el nombre de funciones y comandos usados en programación.

La programación debe coincidir con lo que la tarjeta interpretara como un orden para que los vehículos realicen las diferentes tareas y objetivos.

Los elementos que nos permite usar el programa tienen diferentes bloques para su programación, algunos permiten ejecutar órdenes al ser accionados, otros solo se modifican como resultado de alguna orden.

Cada botón creado tiene diversas posibilidades en su funcionamiento, ya que los botones pueden tener respuesta si son accionados con un solo toque, un toque prolongado, al ser accionados o al ser soltado; es por eso que debemos seleccionar el tipo de accionamiento queremos que tenga y según ese accionamiento cumpla una serie de instrucciones.

Dentro del bloque que indica el estímulo que ha recibido el botón tendremos que poner las diferentes funciones que este desarrollara, para estas funciones debemos seleccionar los bloques del elemento que vallamos a alterar al accionar el botón, estos bloques nos indican la característica que se alterara y nos permitirá colocar un bloque para modificarlo.

Los tres botones superiores tienen una connotación similar ya que estos al ser oprimidos abren la ventana que se les asigno, ocultan las otras dos ventanas y colocar su fondo de color verde dejando a los otros dos botones de color gris.

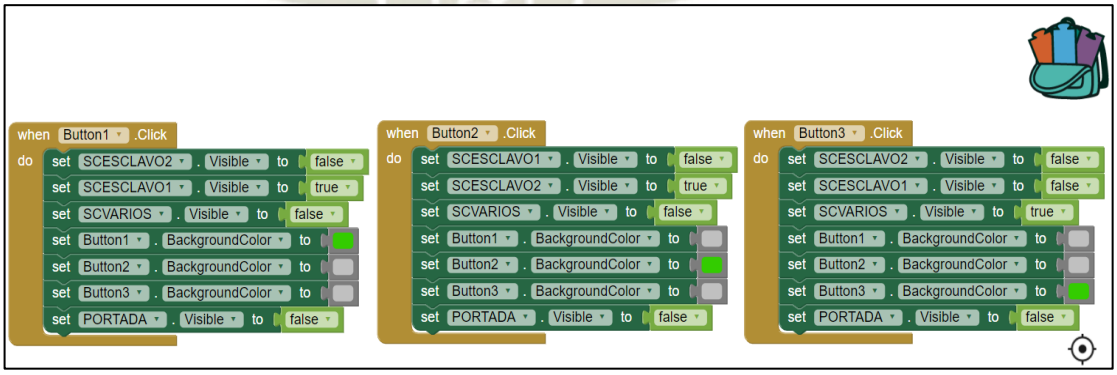


Figura 4.37 Bloques de programación de los botones de navegación
Fuente: Propia

La primera ventana enlazada al botón “VEHICULO1” contiene 7 botones, para cada botón se configurará su estado de funcionamiento y las acciones que realizará.

Cada botón debe mandar una instrucción mediante la red wifi, para esto al hacer clic el botón, se usará el WebViewer, en esta pantalla web se escribirá la ip pedida, se dará una orden que la tarjeta reconocerá y transformará en un movimiento.

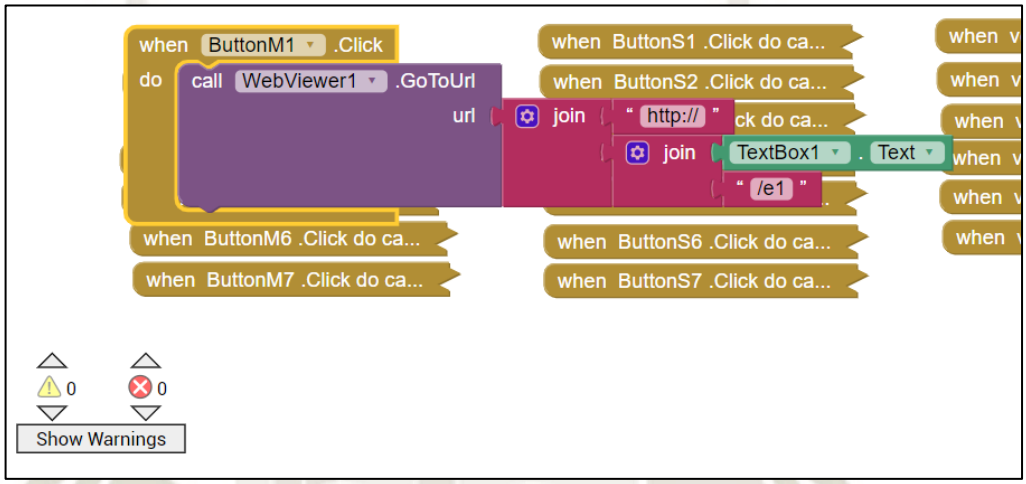


Figura 4.38 Bloques de programación de un botón de mando
Fuente: Propia

La programación de los botones de mando en cada ventana es similar, siempre debemos tener en consideración que WebViewer usaremos, la IP que llamaremos y la orden que mandaremos de acuerdo a lo propuesto en la interfaz.

Al finalizar la programación de la aplicación, pasaremos a generar su apk, este nos permitirá instalar la aplicación en un dispositivo que tenga Android como su sistema operativo y poder probar todas sus funciones.

CAPÍTULO V

5. PRUEBAS Y RESULTADOS

En este capítulo se realizarán diferentes pruebas para poder demostrar lo planteado en los objetivos y en la hipótesis.

Para dar un mayor rendimiento a los vehículos, se independizaron la alimentación de la tarjeta de mando y de los actuadores; esto con el fin de tener una mayor durabilidad de las baterías usadas, debido a que el microcontrolador ESP8266 usa un voltaje bajo y una alta cantidad de corriente, a diferencia de los motores que necesitan de un voltaje más alto para lograr su mayor rendimiento.

Al independizar la alimentación de los actuadores, se puede dar la facilidad de variarlos según la necesidad, sin que la alimentación de la tarjeta se vea afectada, logrando así poder experimentar con diferentes actuadores de características diversas.

Una vez ensamblados los vehículos junto a la tarjeta de mando, los actuadores y las baterías de alimentación, procedemos a ponerlos en funcionamiento.

Primero necesitamos una red WI-FI donde nuestros vehículos se desarrollarán, para esto necesitamos generar un punto de acceso, se generó una red con un smartphone, se le dio un nombre y una contraseña a la red generada.

Se procederá a cargar los programas diseñados para los dos diferentes vehículos a las placas de mando fabricadas, para esto conectaremos los pines txd y rxd a un dispositivo que nos permita conectarnos a la CPU.

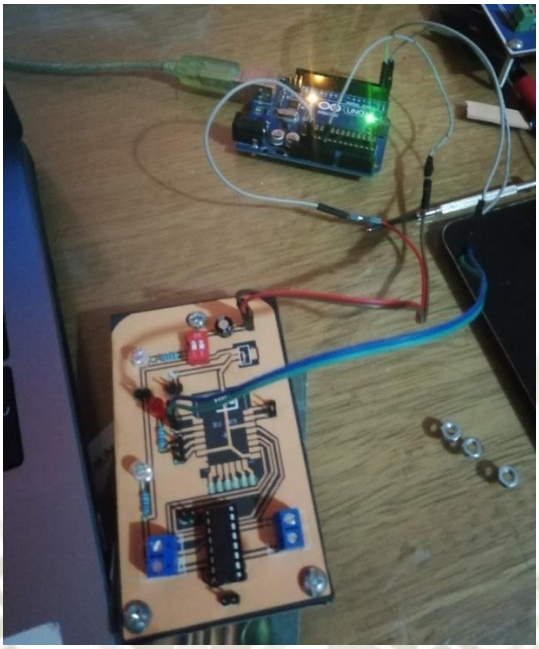


Figura 5.1 Conexión de comunicación entre el ESP8266-12E y la CPU
Fuente: Propia

Una vez conectados los pines de comunicación, alimentaremos a la placa, al ser lograda la comunicación, pasaremos a colocar la placa en modo programación, para esto llevaremos el pin GPIO0 a un cero lógico, resetearemos la placa para que esté lista para recibir el programa y daremos la orden de cargar el programa.

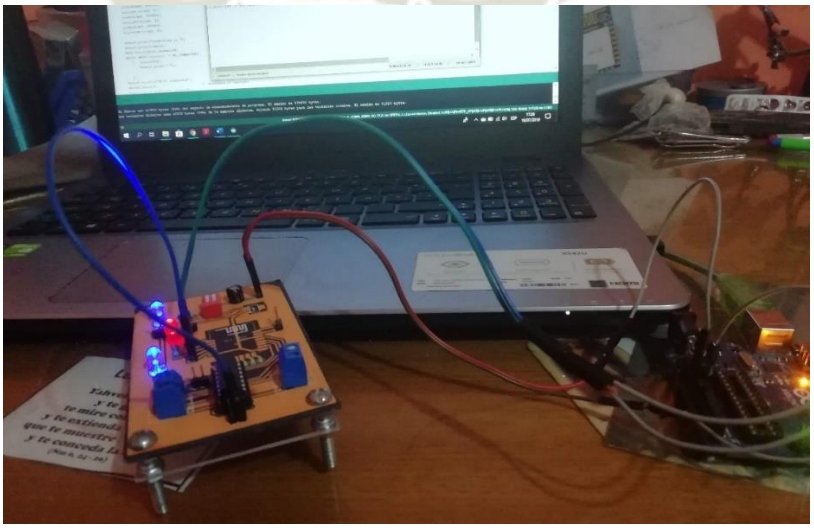


Figura 5.2 Placa puesta en modo programación para cargar programa
Fuente: Propia

Terminada la carga del programa, desconectaremos el GPIO0, reiniciaremos la tarjeta, una vez que empiece a funcionar la tarjeta buscara conectarse a la red que se generó, al establecer un contacto nos mostrara la IP que se le asignó a la tarjeta, debemos recordar esta IP ya que la usaremos cada vez que usemos el vehículo.

Después de comprobar el funcionamiento de la placa, instalaremos la placa en el vehículo, conectaremos los motores y la alimentación para la placa y para los motores.

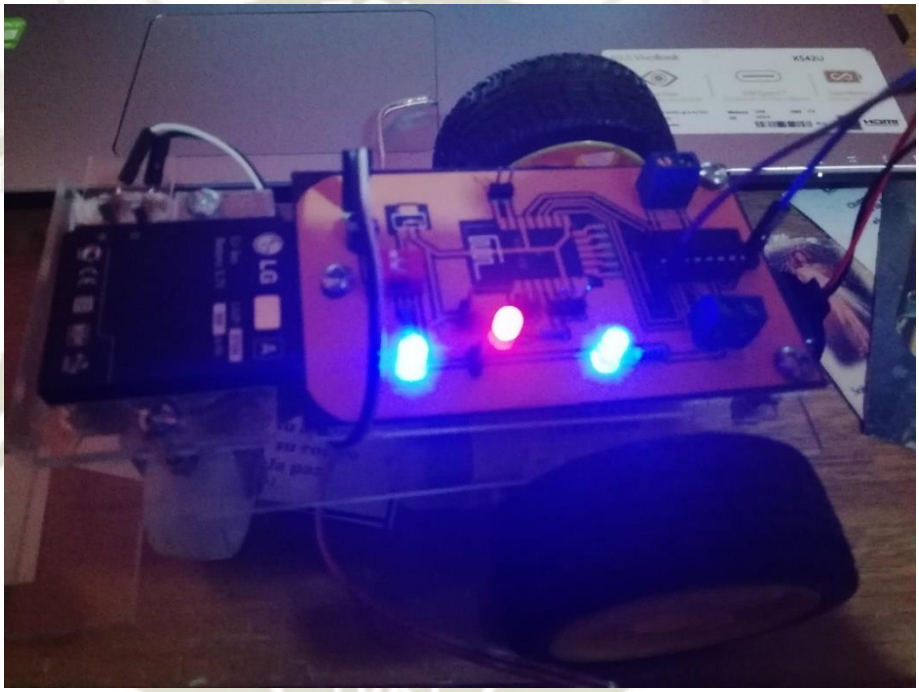


Figura 5.3 Vehículo uno con placa instalada
Fuente: Propia

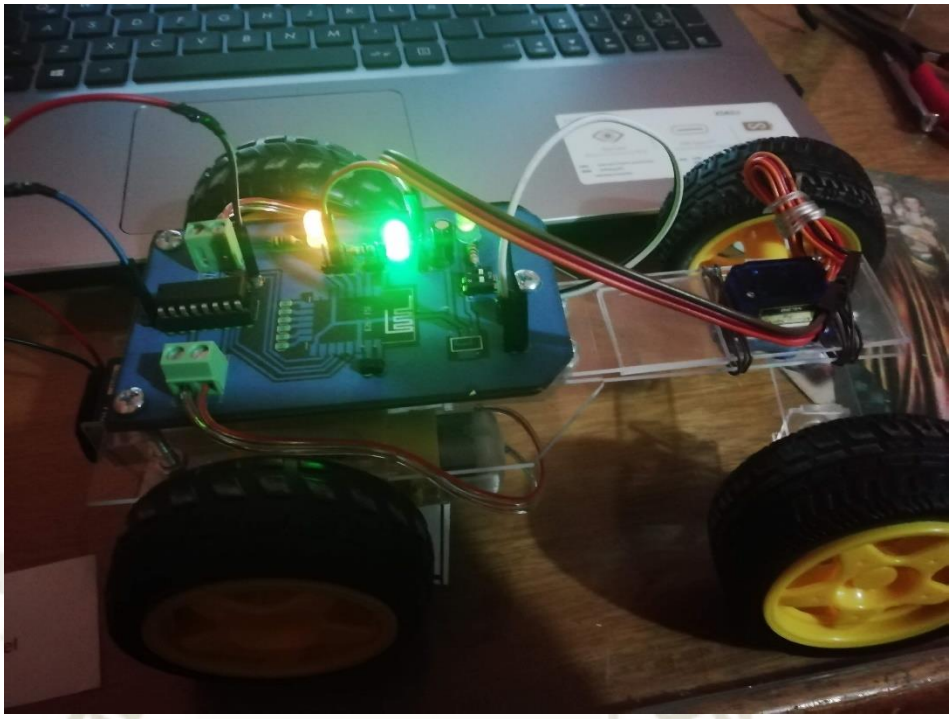


Figura 5.4 Vehículo dos con placa instalada
Fuente: Propia

Teniendo la red, instalaremos la aplicación creada en un dispositivo móvil, para esto pasaremos el instalador generado al dispositivo seleccionado, conectaremos el dispositivo a la red wifi, una vez enlazados a la red podremos acceder a la aplicación y verificar todas las funciones creadas.

Encenderemos los vehículos, un led se encenderá cuando se conecte a la red como se ve en las figuras 5.3 y 5.4.

El encendido de los leds nos permitirá empezar a controlar cada vehículo.

Ingresaremos a la aplicación, nos ubicaremos en una de las ventanas de control, ingresaremos la IP correspondiente al vehículo y empezaremos a realizar funciones.

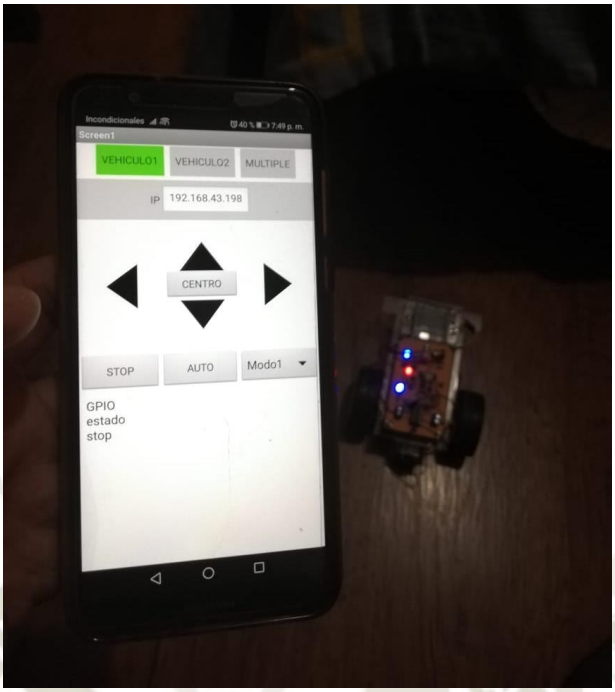


Figura 5.5 Uso de la App con un vehículo.
Fuente: Propia

Procederemos a repetir el proceso para el segundo vehículo, considerando que el comportamiento de este incluye un servomotor, cargaremos el programa correspondiente.

Entraremos a la segunda ventana de la aplicación donde ingresaremos la IP que se asignó al segundo vehículo y comprobaremos el funcionamiento de este.

Una vez que ambos vehículos funcionaron correctamente, pasaremos al control simultaneo de ambos, para esto pasaremos a la tercera ventana de la aplicación donde ingresaremos la IP de los vehículos en los espacios asignados y empezaremos a dar instrucciones a ambos vehículos para comprobar su funcionamiento simultaneo.

Figura 5.6 Uso de la App para el control de múltiples vehículos.
Fuente: Propia

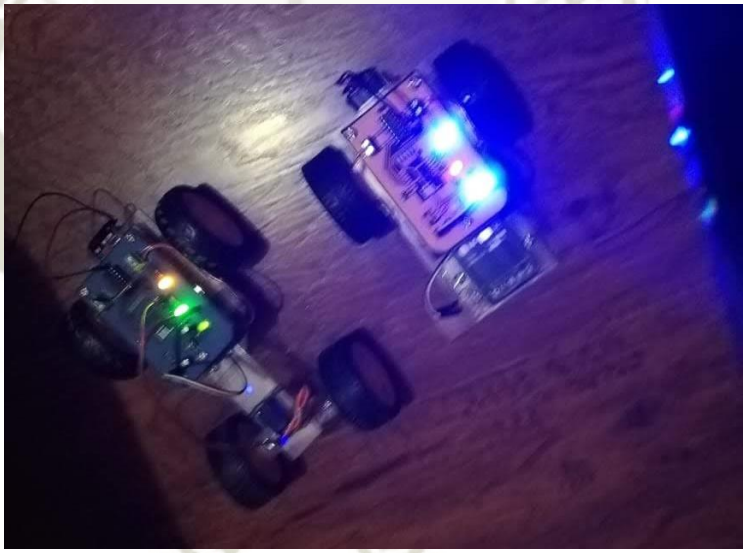


Figura 5.7 Vehículos funcionando en simultaneo.
Fuente: Propia

Para terminar de comprobar el funcionamiento múltiple, usaremos la selección de tareas prestablecidas para ambos vehículos y procederemos a activarlos.

CONCLUSIONES

1. Se logró controlar más de un vehículo no tripulado al mismo tiempo con un solo dispositivo que se estableció como el dispositivo maestro, controlando los vehículos como dispositivos esclavos.
2. Se logró establecer el control y la comunicación entre un solo dispositivo de mando maestro y varios dispositivos de acción o esclavos de manera simultánea mediante el uso del protocolo de comunicación IEEE 802.11 conocido comercialmente como Wi-Fi.
3. Se determinó que el mejor medio de transmisión de datos para el uso de varios dispositivos fue la señal Wi-Fi, que a diferencia de otros medios nos permite una conectar varios dispositivos en una sola red.
4. Se diseñó e implemento un interfaz para dispositivos móviles, fácil de manejar para los usuarios, que nos permitía el control simultaneo de hasta tres vehículos para el cumplimiento de diversas tareas que se planteaban.
5. Se logró una buena comunicación con cada vehículo, con una buena velocidad de respuesta entre la orden dada por el dispositivo de mando y los vehículos controlados.
6. Se mejoró el rendimiento de los vehículos al establecer el control múltiple, la eficiencia de los vehículos sube, esto debido a que se pueden manejar una mayor cantidad de vehículos por un solo usuario, utilizando un solo aparato de control que nos permite tener una supervisión en tiempo real del estado de los vehículos

RECOMENDACIONES

1. La tesis fue propuesta para dar un paso a estas nuevas tecnologías, se sugiere seguir investigando los medios de comunicación y los diferentes microcontroladores del mercado para aprovechar al máximo los objetivos obtenidos.
2. La aplicación creada se basa en un programa básico, existen otros softwares de diseño donde se puede mejorar la apariencia del interfaz para ser más amigable con el usuario.
3. El principal objetivo fue el controlar dos vehículos a la vez, para esto solo se usaron órdenes de mandos en actuadores, se descubrió que se pueden aprovechar sensores en los vehículos, pero este punto no se tomó para los modelos finales.
4. La tecnología del microcontrolador se usó para los vehículos no tripulados, se sugiere aprovechar los diferentes beneficios en el desarrollo de otros campos de ingeniería.

BIBLIOGRAFIA

Arduino. (Enero, 2018). What is Arduino? Recuperado de: https://www.arduino.cc/en/guide/introduction

Barrientos, A., del Cerro, J., Gutiérrez, P., San Martin, R., Martínez, A. y Rossi, C. (2007). Vehículos aéreos no tripulados para uso civil. Tecnología y aplicaciones. Universidad politécnica de Madrid.

Boltom, W. (2010). Mecatrónica. Sistemas de control electrónico en ingeniería mecánica y eléctrica. 4ta Edición. México: Alfaomega.

Creus, A. (2011*). Instrumentación Industrial. 8va Edición. México: Alfaomega*.

Del Valle, L. (2015). *Guía de configuración y programación ESP01-ESP8266.* Recuperado de: https://programarfacil.com/podcast/esp8266-wifi-coste-arduino/

Galeano, G. (2009). Programación de Sistemas Embebidos en C. Colombia: Alfaomega.

Instituto Tecnológico de Massachusetts. (2016, Noviembre). *The MIT App Inventor Library: Documentation & Support.* Recuperado de: http://appinventor.mit.edu/explore/library.html

Katsuhiko, O. (2003). Ingeniería de Control Moderna, Madrid: Pearson Educación.

Martin Del Brío, B. (2007). Redes Neuronales y Sistemas Borrosos. 3ra. Edición. México: Alfaomega.

Microcontroladores y Sus Aplicaciones. (2013). *Introducción y Arquitectura de microcontroladores*. Recuperado de: https://microcontroladoressesv.wordpress.com/arquitectura-de-los-microcontroladores/

National Instruments. (2015). *Patrones de diseño de aplicaciones: Maestro/Esclavo.* Recuperado de: http://www.ni.com/tutorial/3022/es/

Network world. (2018). 802.11: estándares de Wi-Fi y velocidades. Recuperado de: https://www.networkworld.es/wifi/80211-estandares-de-wifi-y-velocidades

Robayo, E. (2007). Control Difuso Fundamentos y Aplicaciones. 1ra. Edición. Universidad del Norte Editorial.

Rusell, S. y Norving, P. (2004). Inteligencia Artificial un Enfoque Moderno. 2da. Edición. Madrid: Pearson Educación.

Sánchez, O. (septiembre, 2009). *Vehículos no tripulados Aplicaciones de la robótica* Recuperado de: https://es.slideshare.net/omarspp/vehiculos-no-tripulados.

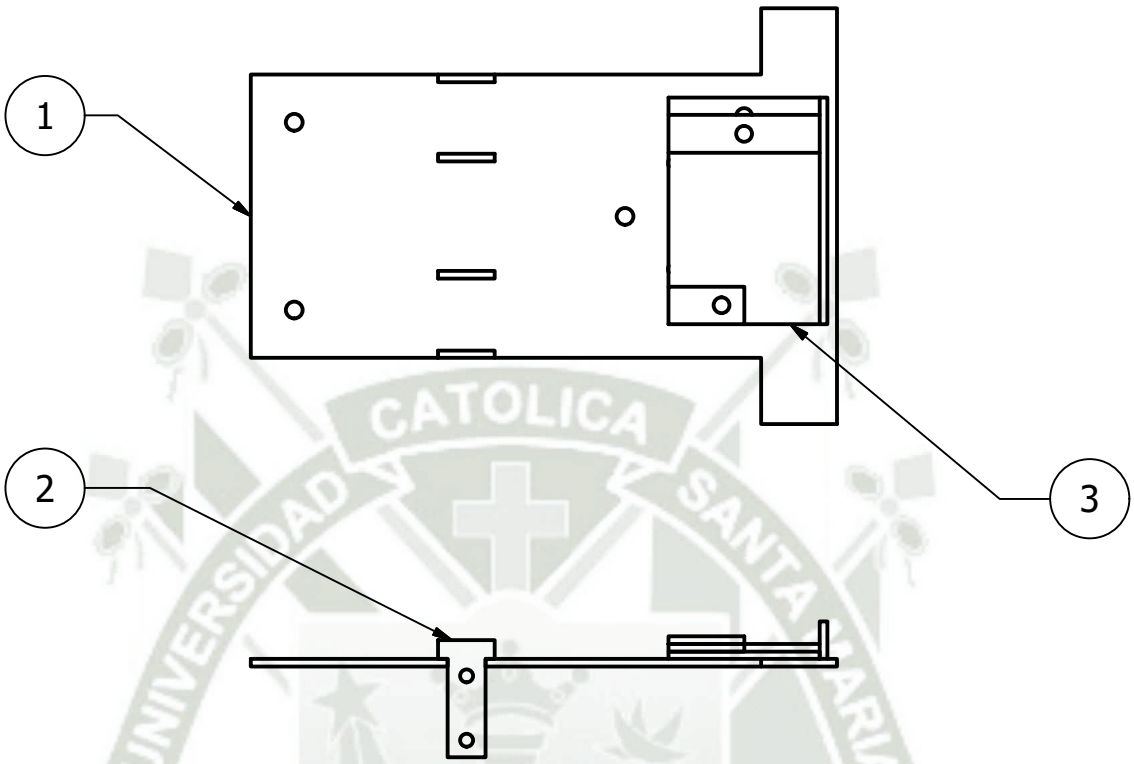
Significados. (2018). *Significado de Wifi*. Recuperado de: https://www.significados.com/wifi/

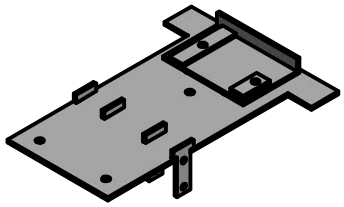
WI-FI Alliance. (2018). *Wi-Fi Generations*. Recuperado de: https://www.wi-fi.org/discover-wi-fi

ANEXOS

ANEXO A

PLANOS VEHÍCULO 1

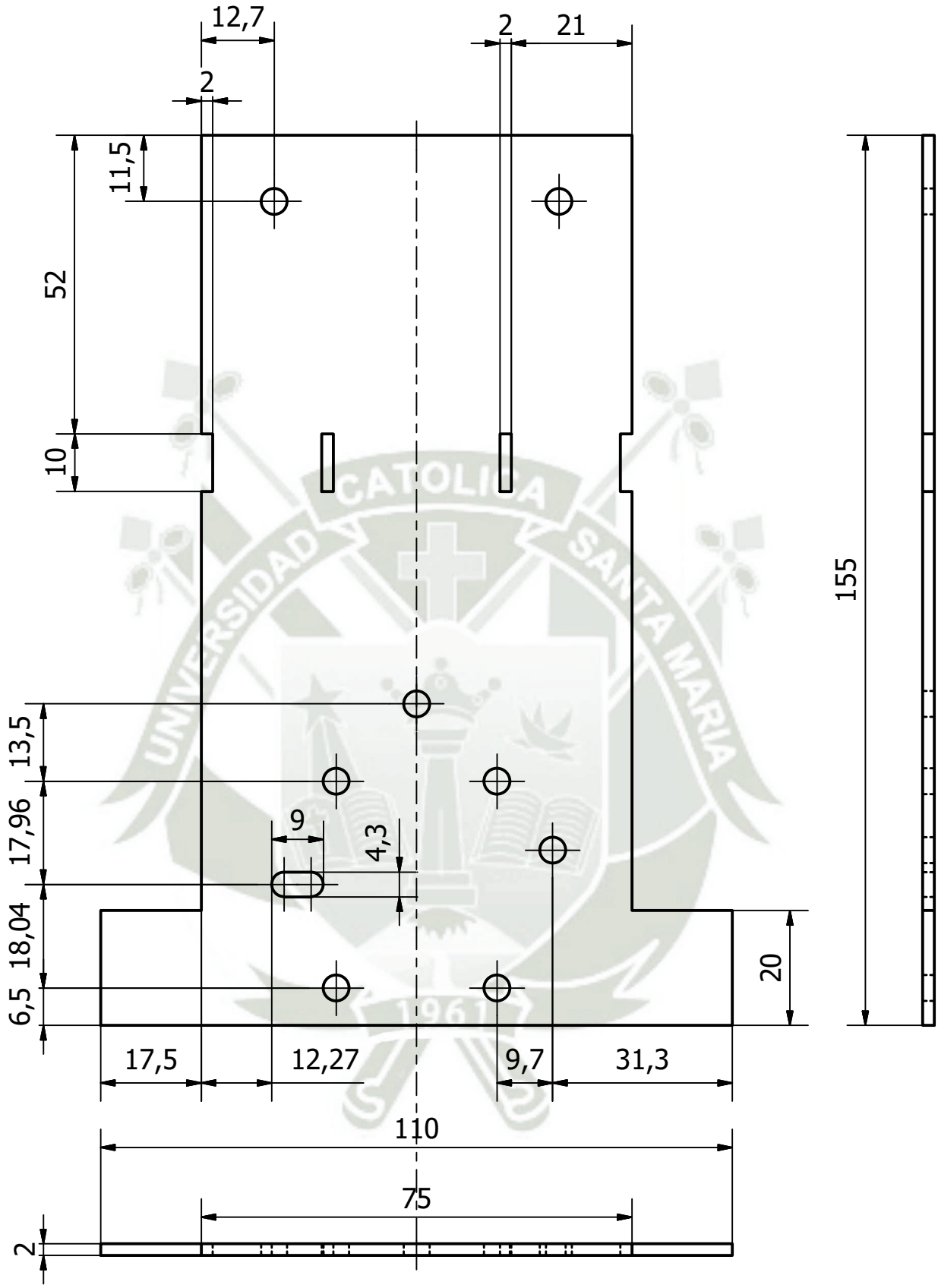




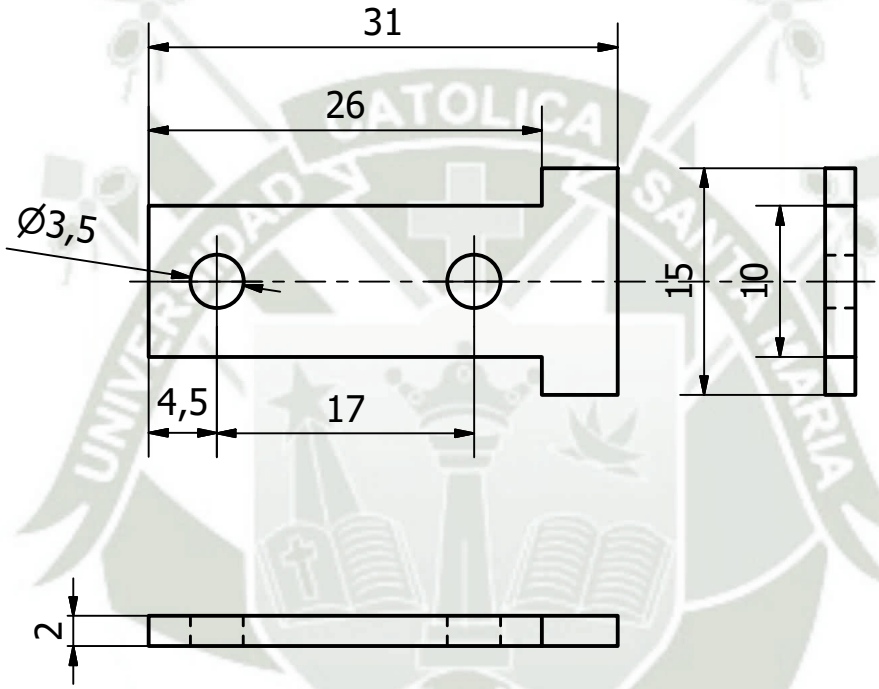
CANT.	DENOMINACION	MARC.	NORMA	MATERIAL	OBSERVACIONES
1	SOPORTE DE BATERIA	3		PMMA	
4	SOPORTE DE MOTOR	2		PMMA	
1	BASE VEHÍCULO UNO	1		PMMA	

	FECHA	NOMBRE	UNIVERSIDAD CATOLICA SANTA MARIA
DIBUJADO	15/01/2019	JOSUÉ DAVID JUÁREZ VALENCIA	

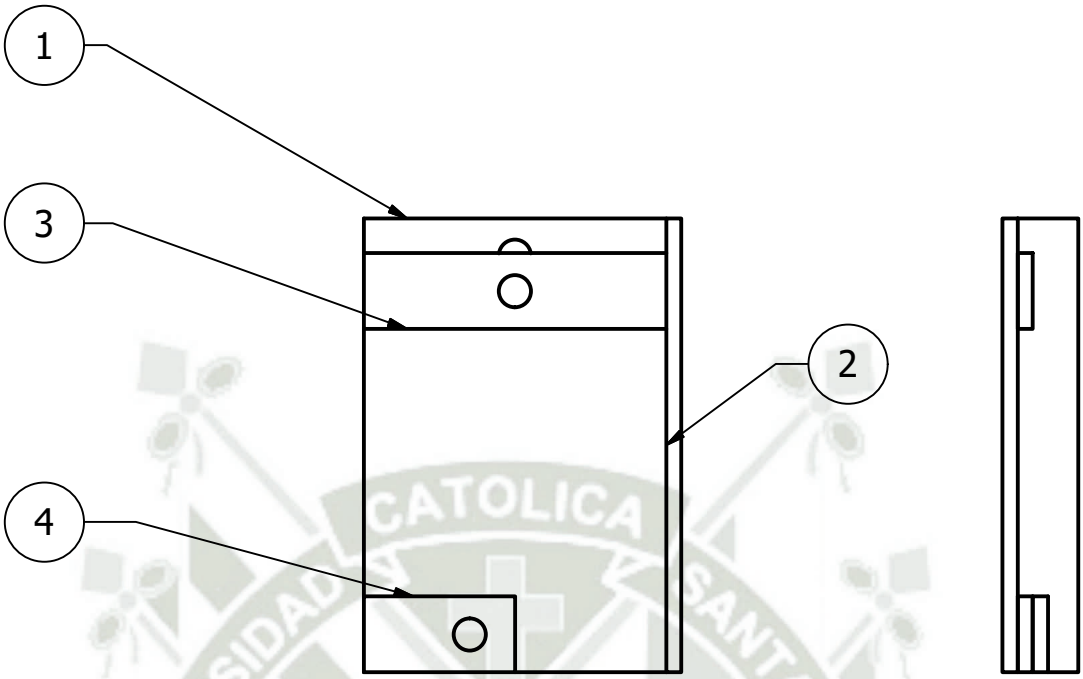
ESCALA		NUMERO
1:2	VEHÍCULO 1	-

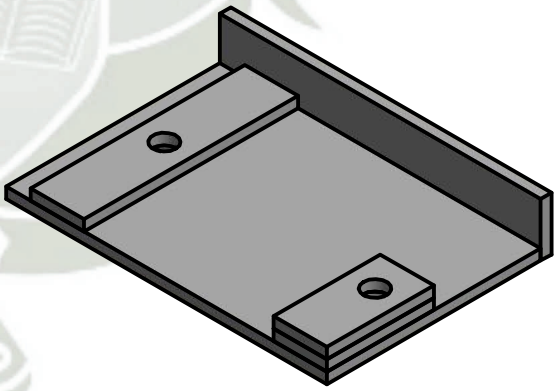


	FECHA	NOMBRE	UNIVERSIDAD CATOLICA SANTA MARIA
DIBUJADO	25/07/2019	JOSUÉ DAVID JUÁREZ VALENCIA	
ESCALA 1:1	BASE VEHÍCULO UNO		NUMERO 1/2



	FECHA	NOMBRE	UNIVERSIDAD CATOLICA SANTA MARIA
DIBUJADO	25/07/2019	JOSUÉ DAVID JUÁREZ VALENCIA	
ESCALA 2:1	SOPORTE DE MOTOR		NUMERO 2/2

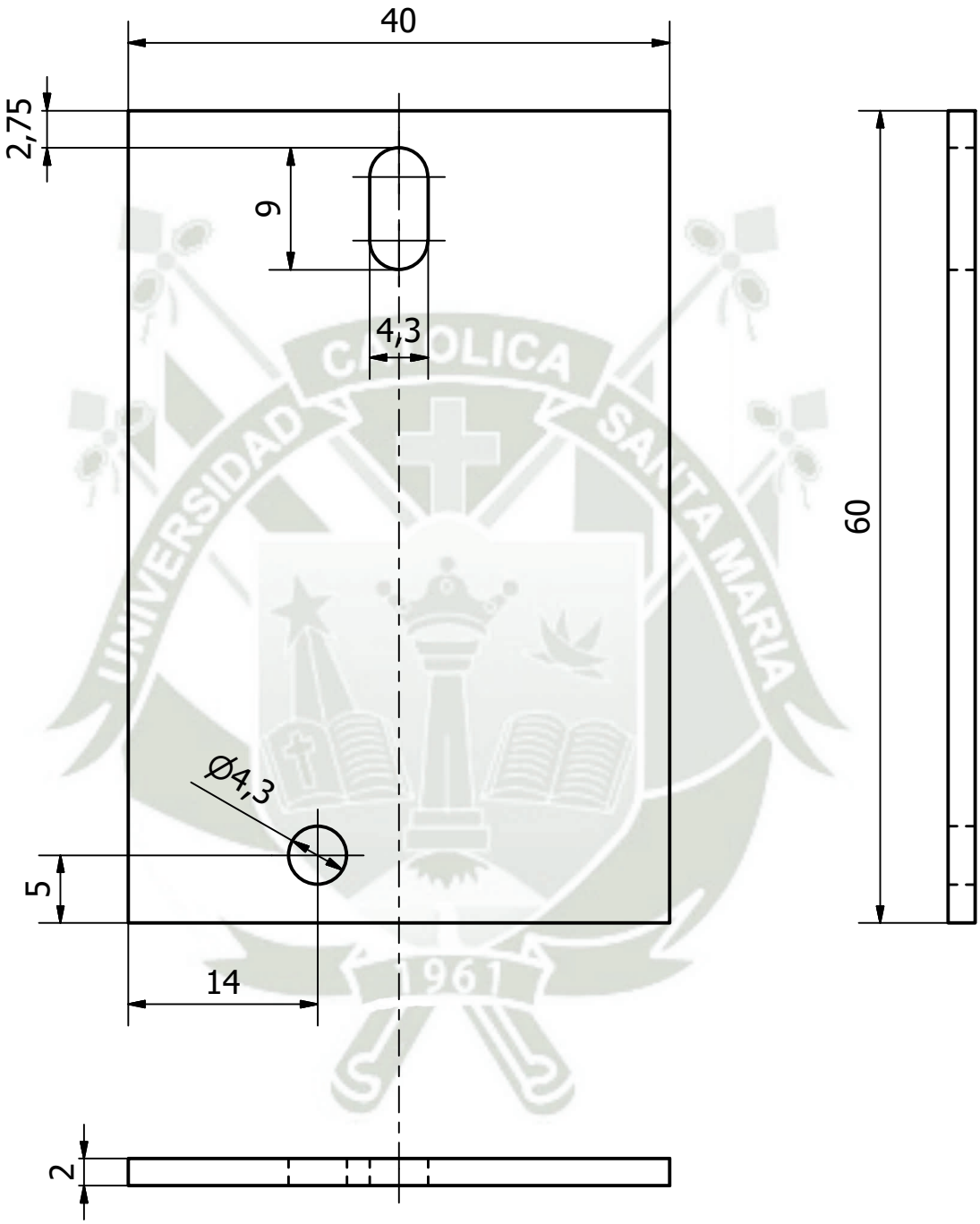




CANT.	DENOMINACION	MARC.	NORMA	MATERIAL	OBSERVACIONES
2	LIMITADOR DE POSICIÓN PARA BATERÍA	4		PMMA	
1	AJUSTE DE BATERÍA	3		PMMA	
1	PIEZA DE CONTENCION PARA BATERÍA	2		PMMA	
1	BASE SOPORTE DE BATERÍA	1		PMMA	

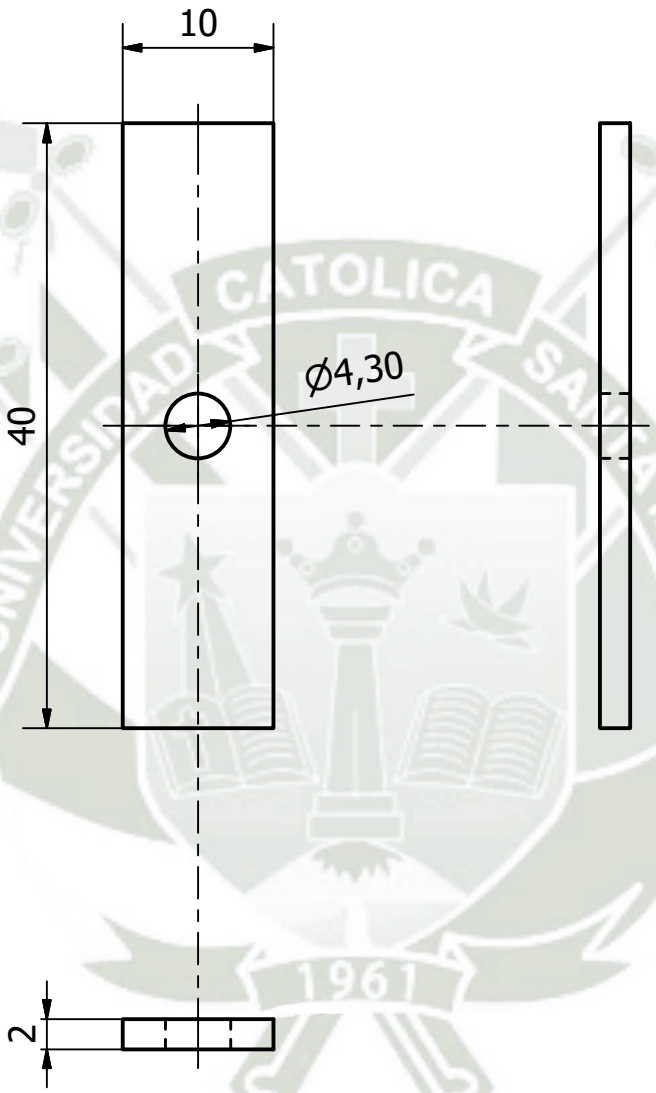
	FECHA	NOMBRE	UNIVERSIDAD CATOLICA SANTA MARIA
DIBUJADO	15/01/2019	JOSUÉ DAVID JUÁREZ VALENCIA	

ESCALA	SOPORTE DE BATERIA	NUMERO
1:1		-

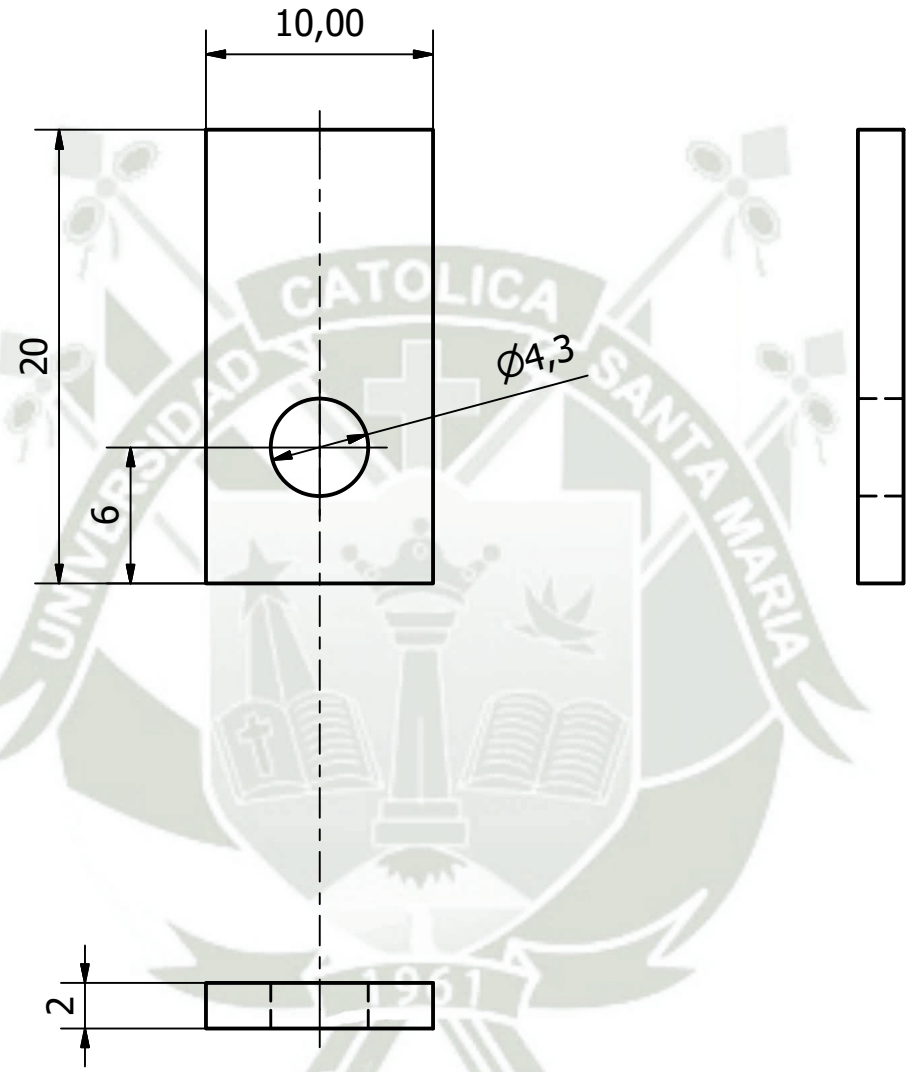


	FECHA	NOMBRE	UNIVERSIDAD CATOLICA SANTA MARIA
DIBUJADO	25/07/2019	JOSUÉ DAVID JUÁREZ VALENCIA	
ESCALA 2:1	BASE SOPORTE DE BATERIA		NUMERO 1/4

	FECHA	NOMBRE	UNIVERSIDAD CATOLICA SANTA MARIA
DIBUJADO	25/07/2019	JOSUÉ DAVID JUÁREZ VALENCIA	
ESCALA 2:1	PIEZA DE CONTENCION PARA BATERIA		NUMERO 2/4



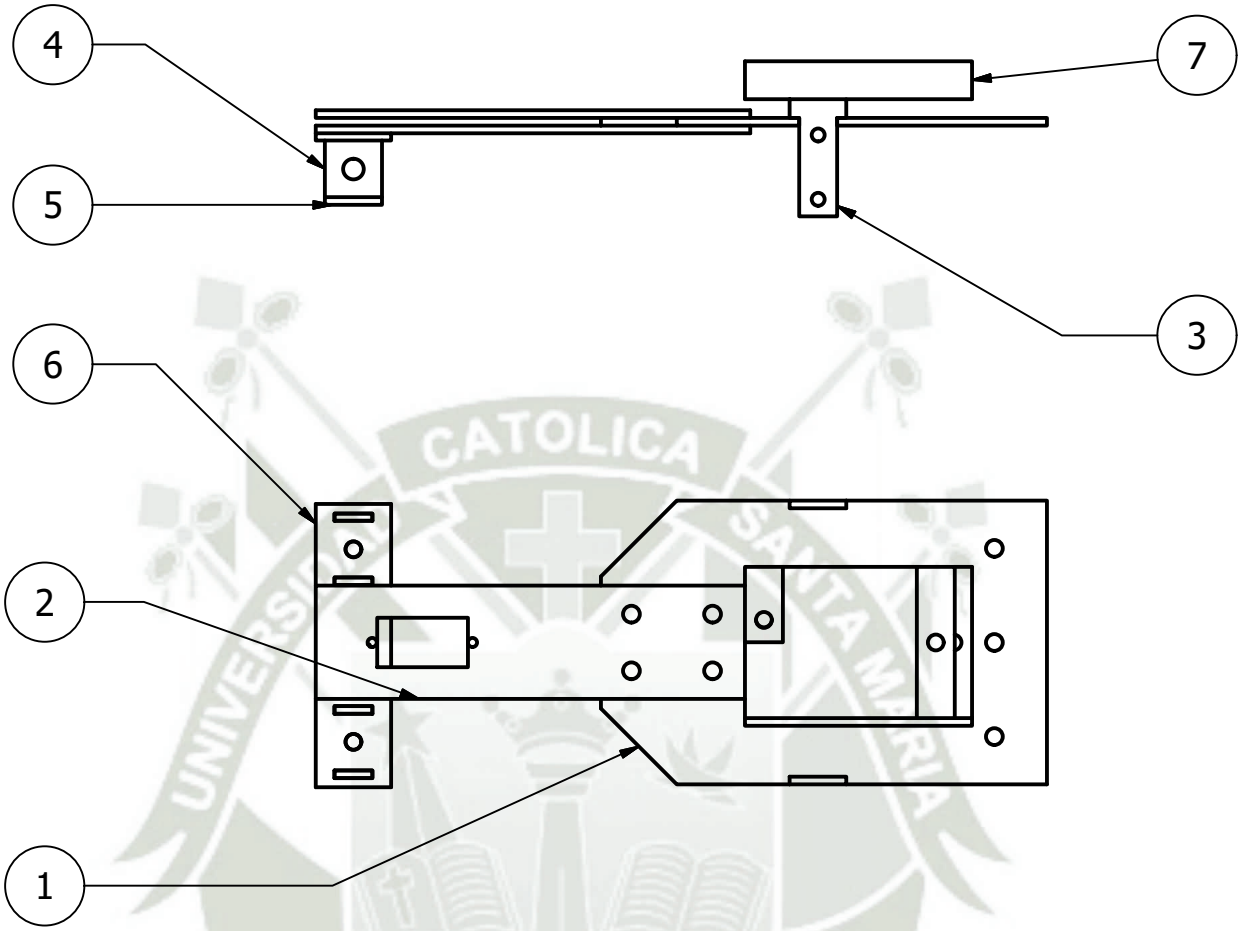
	FECHA	NOMBRE	UNIVERSIDAD CATOLICA SANTA MARIA
DIBUJADO	25/07/2019	JOSUÉ DAVID JUÁREZ VALENCIA	
ESCALA 2:1	AJUSTE DE BATERIA		NUMERO 3/4

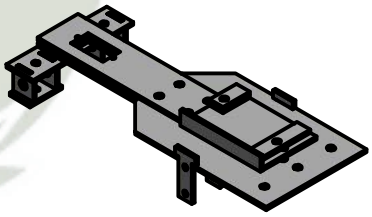


	FECHA	NOMBRE	UNIVERSIDAD CATOLICA SANTA MARIA
DIBUJADO	25/07/2019	JOSUÉ DAVID JUÁREZ VALENCIA	
ESCALA 3:1	LIMITADOR DE POSICION PARA BATERIA		NUMERO 4/4

ANEXO B

PLANOS VEHÍCULO 2

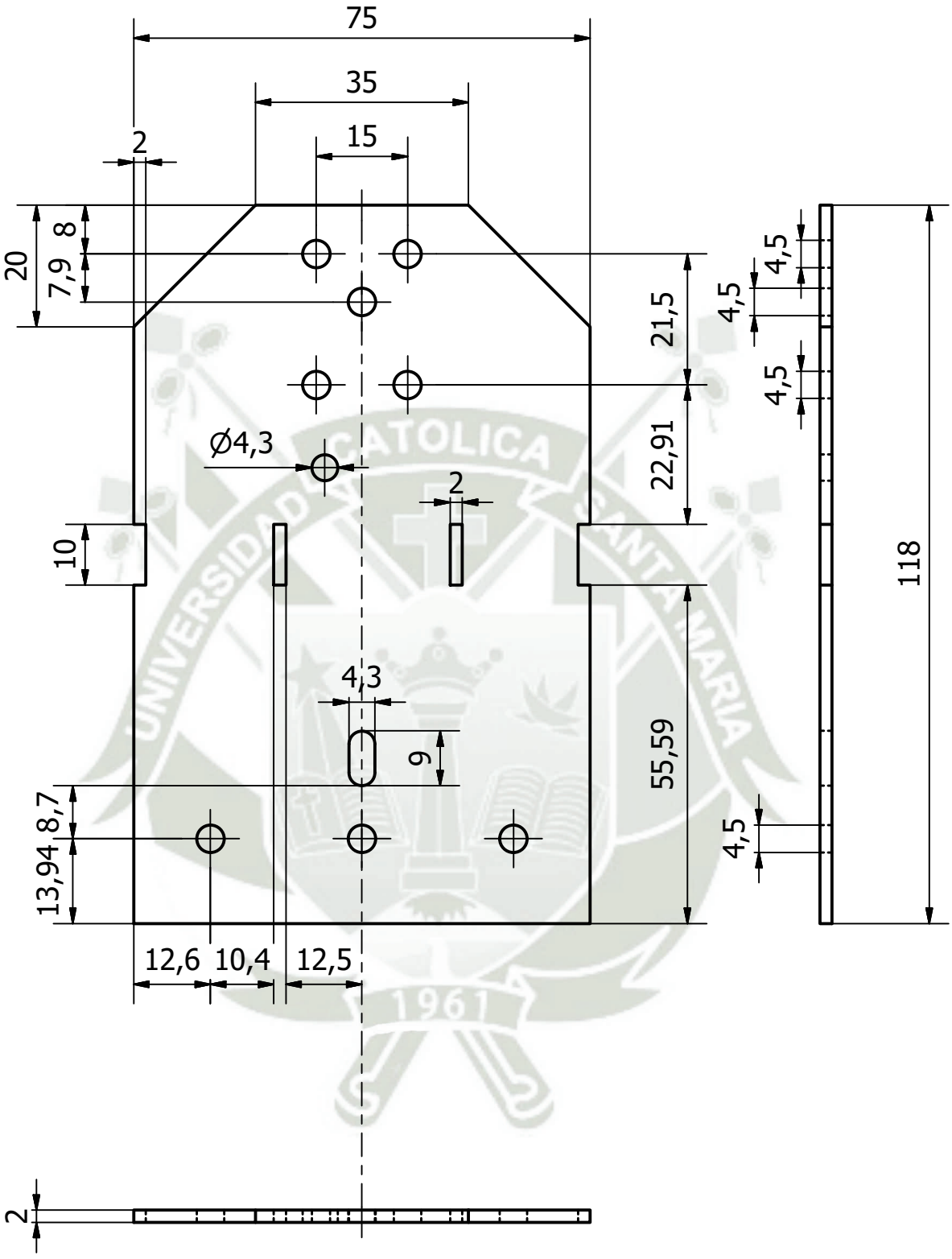




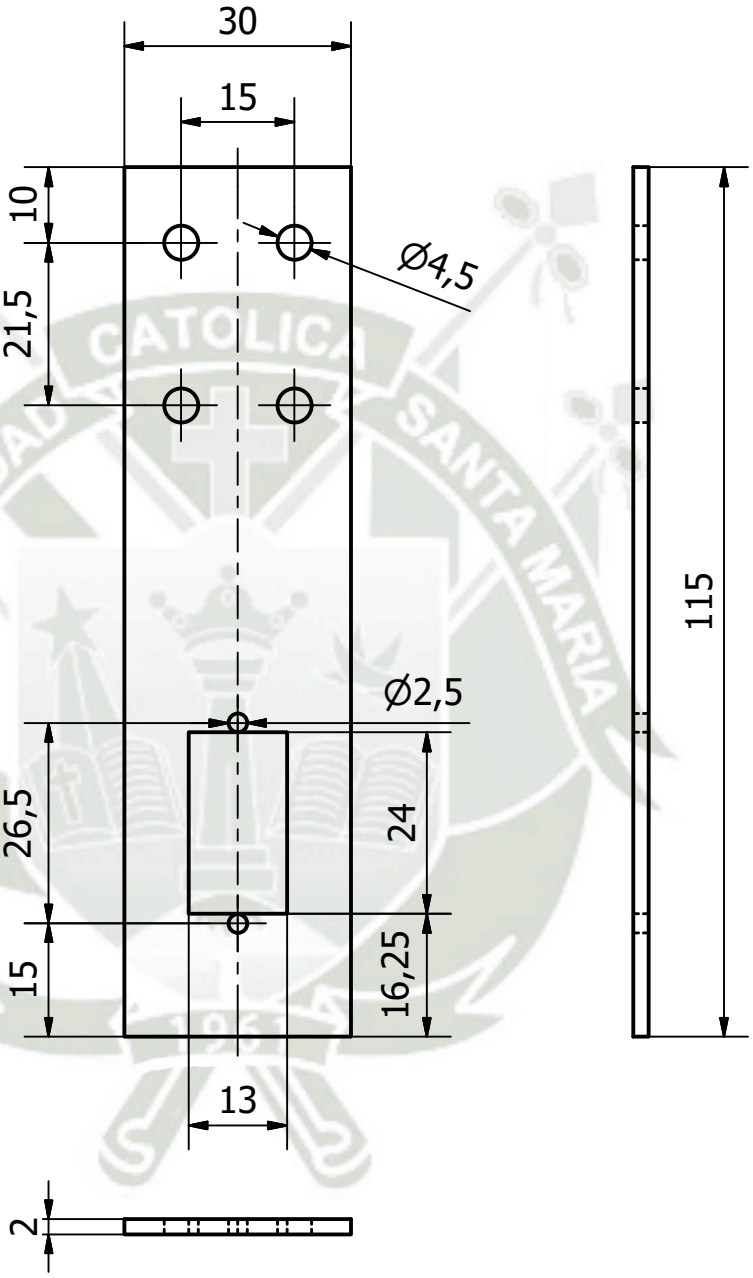
CANT.	DENOMINACION	MARC.	NORMA	MATERIAL	OBSERVACIONES
1	SOPORTE DE BATERIA	7		PMMA	
1	PIEZA EJE 3	6		PMMA	
2	PIEZA EJE 2	5		PMMA	
4	PIEZA EJE 1	4		PMMA	
4	SOPORTE DE MOTOR	3		PMMA	
2	BRAZO DE CONEXION	2		PMMA	
1	BASE VEHICULO DOS	1		PMMA	

	FECHA	NOMBRE	UNIVERSIDAD CATOLICA SANTA MARIA
DIBUJADO	15/01/2019	JOSUÉ DAVID JUÁREZ VALENCIA	

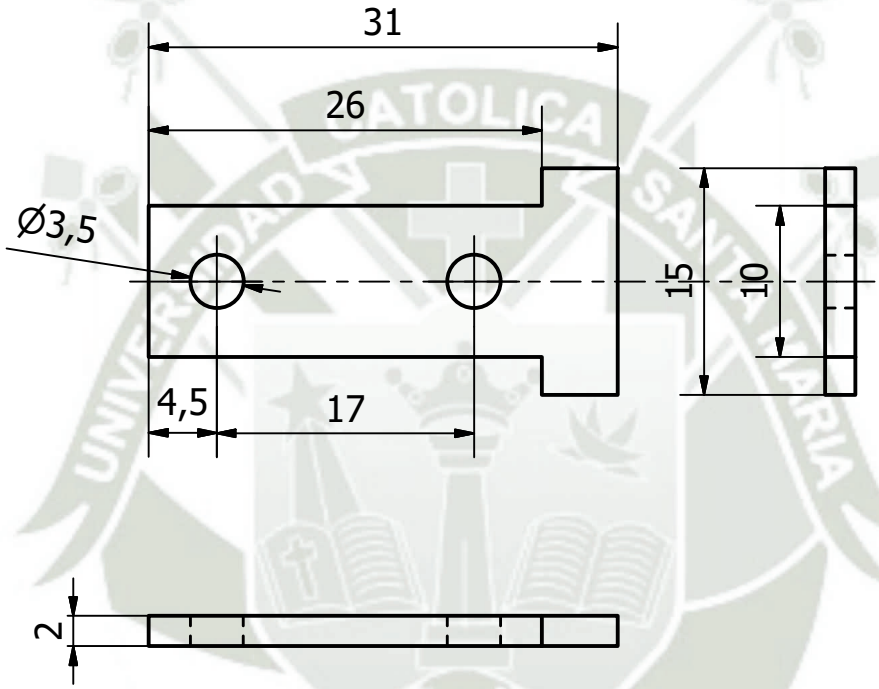
ESCALA	VEHÍCULO 2	NUMERO
1:2		-



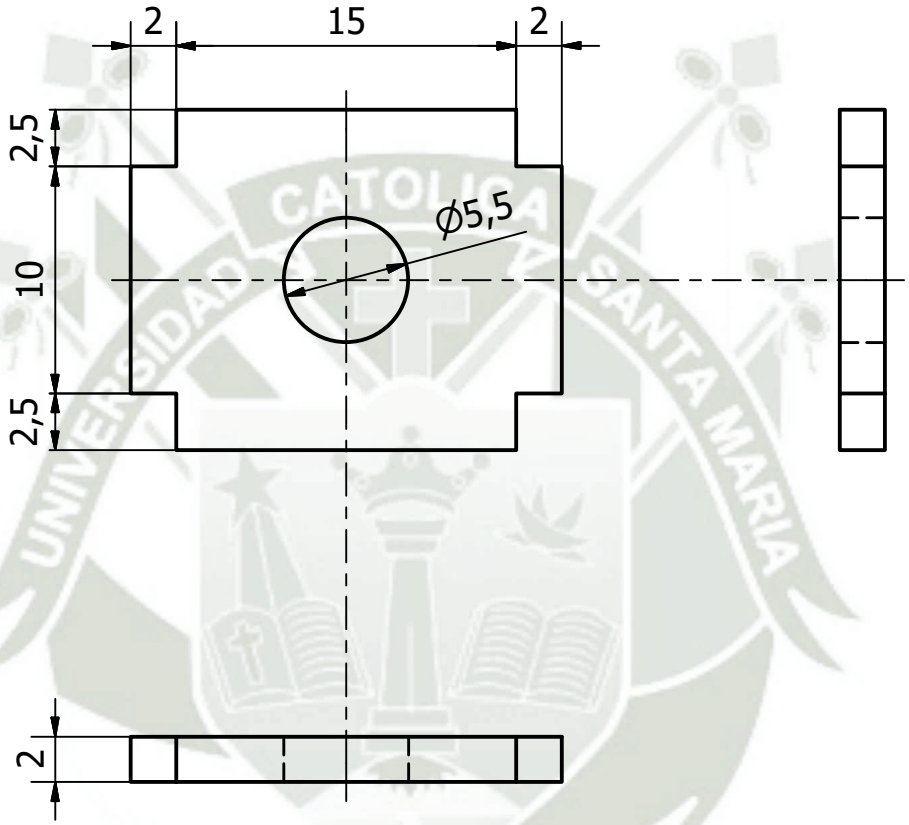
	FECHA	NOMBRE	UNIVERSIDAD CATOLICA SANTA MARIA
DIBUJADO	25/07/2019	JOSUÉ DAVID JUÁREZ VALENCIA	
ESCALA 1:1	BASE VEHICULO DOS		NUMERO 1/6



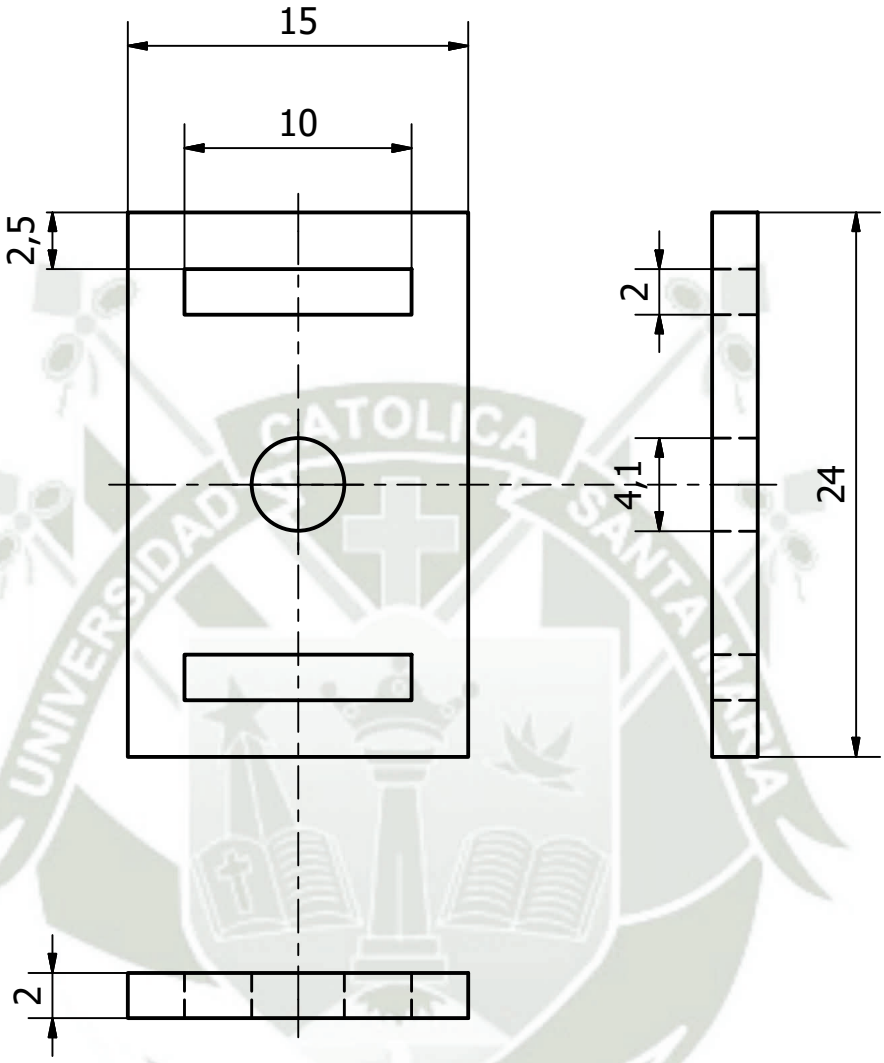
	FECHA	NOMBRE	UNIVERSIDAD CATOLICA SANTA MARIA
DIBUJADO	25/07/2019	JOSUÉ DAVID JUÁREZ VALENCIA	
ESCALA 1:1	BRAZO DE CONEXION		NUMERO 2/6



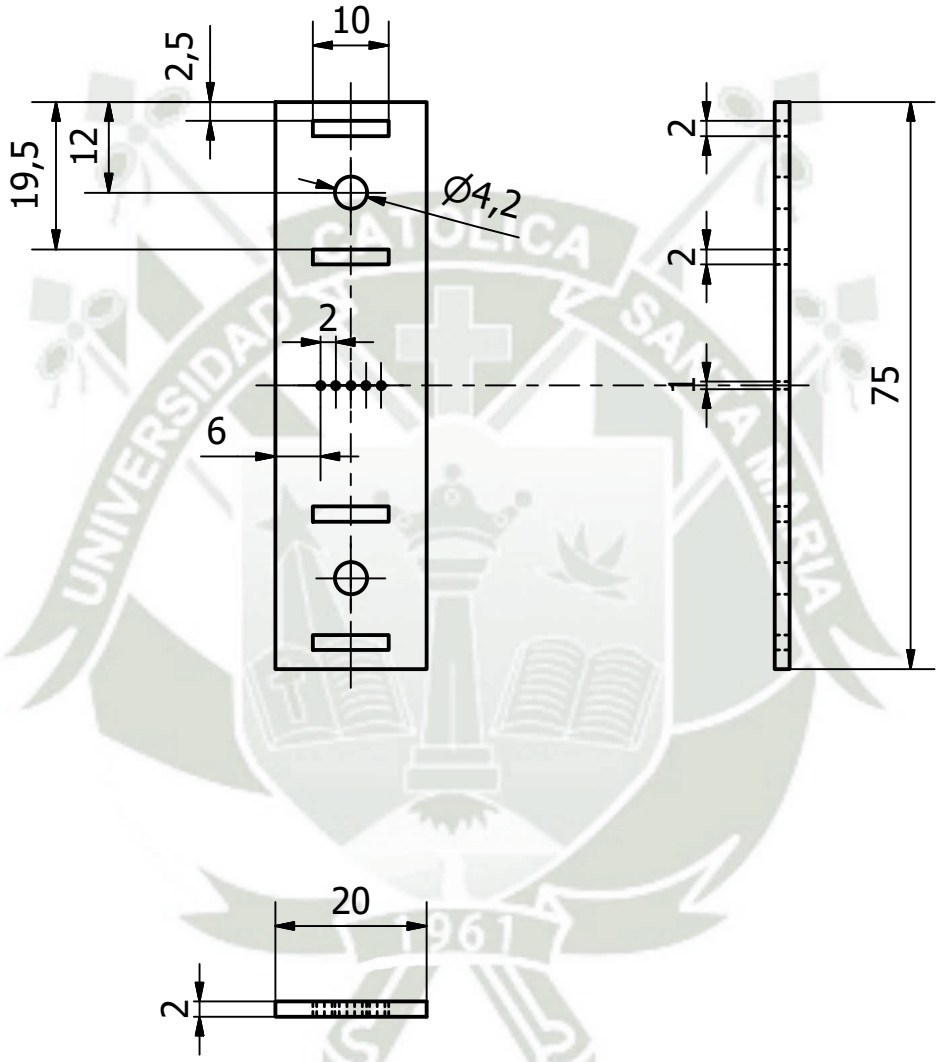
	FECHA	NOMBRE	UNIVERSIDAD CATOLICA SANTA MARIA
DIBUJADO	25/07/2019	JOSUÉ DAVID JUÁREZ VALENCIA	
ESCALA 2:1	SOPORTE DE MOTOR		NUMERO 3/6



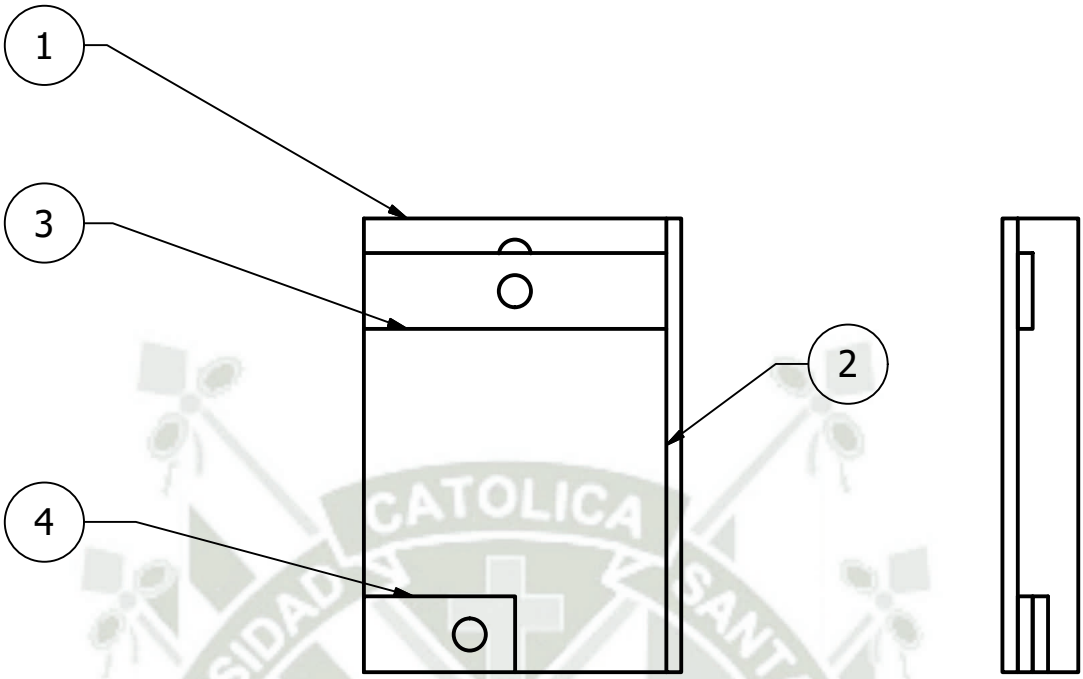
	FECHA	NOMBRE	UNIVERSIDAD CATOLICA SANTA MARIA
DIBUJADO	25/07/2019	JOSUÉ DAVID JUÁREZ VALENCIA	
ESCALA 3:1	PIEZA EJE 1		NUMERO 4/6

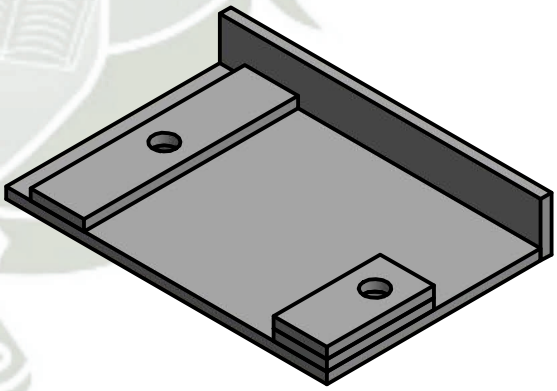


	FECHA	NOMBRE	UNIVERSIDAD CATOLICA SANTA MARIA
DIBUJADO	25/07/2019	JOSUÉ DAVID JUÁREZ VALENCIA	
ESCALA 3:1	PIEZA EJE 2		NUMERO 5/6



	FECHA	NOMBRE	UNIVERSIDAD CATOLICA SANTA MARIA
DIBUJADO	25/07/2019	JOSUÉ DAVID JUÁREZ VALENCIA	
ESCALA 1:1	PIEZA EJE 3		NUMERO 6/6

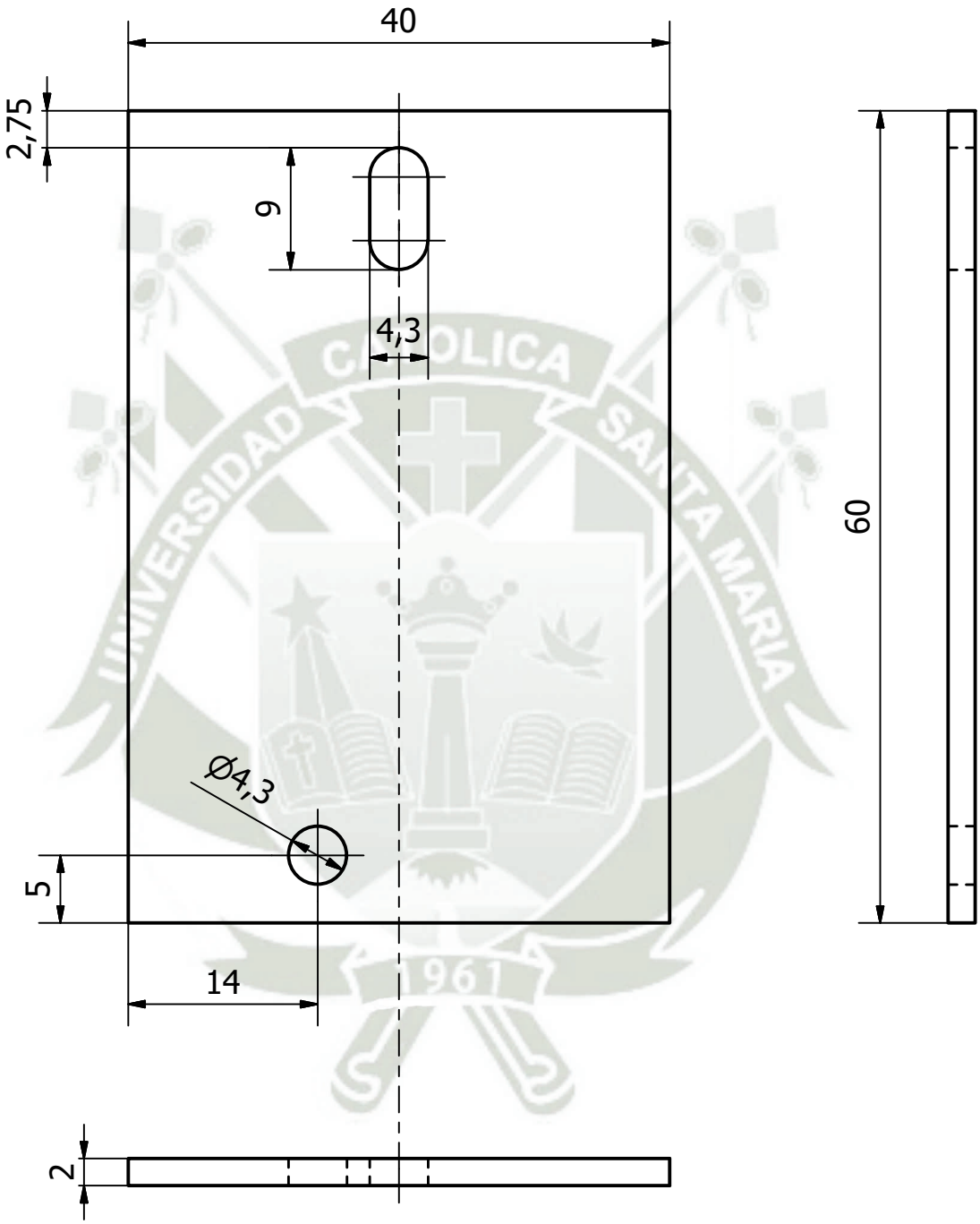




CANT.	DENOMINACION	MARC.	NORMA	MATERIAL	OBSERVACIONES
2	LIMITADOR DE POSICIÓN PARA BATERÍA	4		PMMA	
1	AJUSTE DE BATERÍA	3		PMMA	
1	PIEZA DE CONTENCION PARA BATERÍA	2		PMMA	
1	BASE SOPORTE DE BATERÍA	1		PMMA	

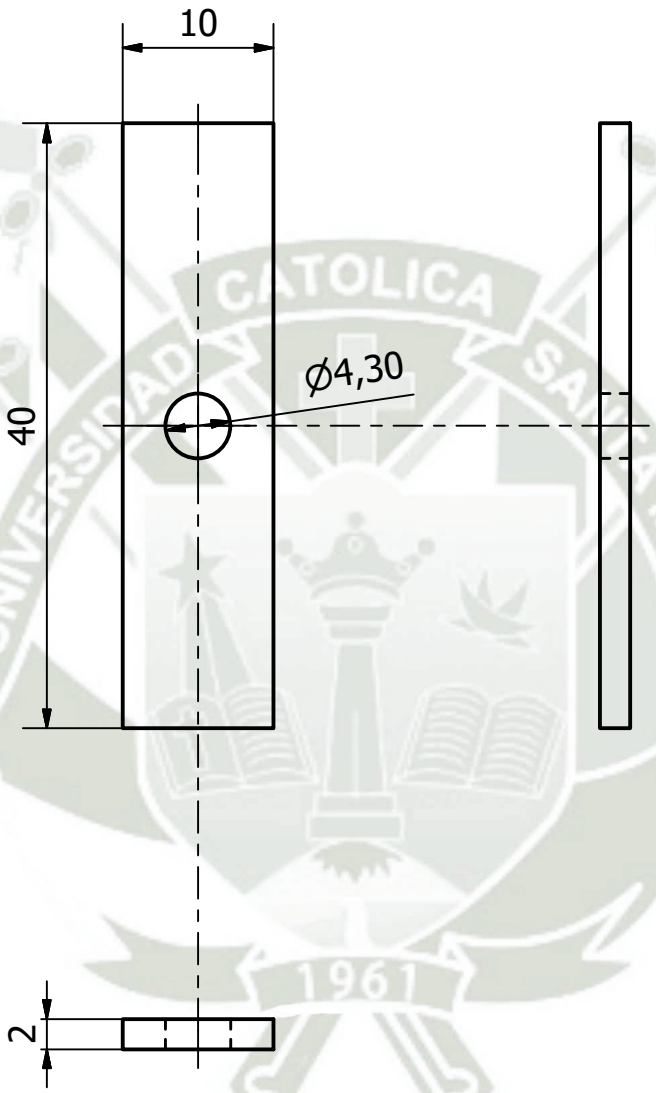
	FECHA	NOMBRE	UNIVERSIDAD CATOLICA SANTA MARIA
DIBUJADO	15/01/2019	JOSUÉ DAVID JUÁREZ VALENCIA	

ESCALA	SOPORTE DE BATERIA	NUMERO
1:1		-

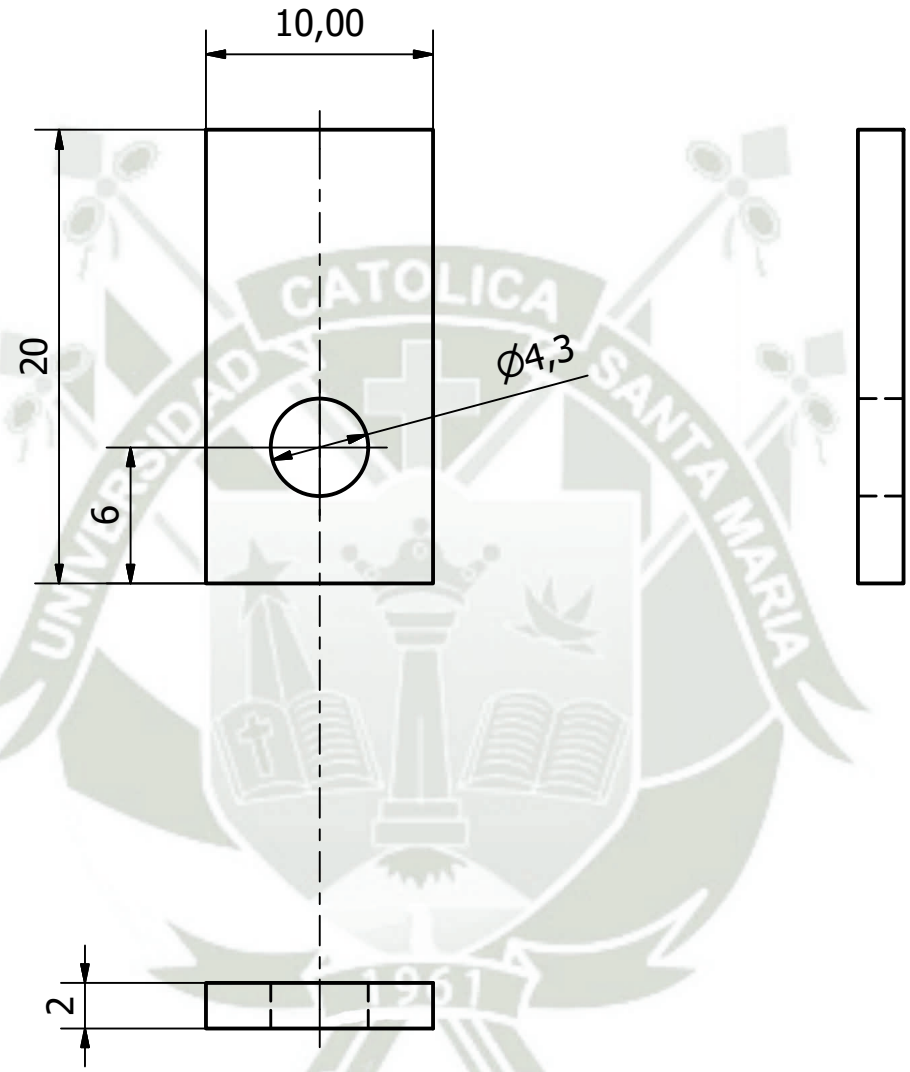


	FECHA	NOMBRE	UNIVERSIDAD CATOLICA SANTA MARIA
DIBUJADO	25/07/2019	JOSUÉ DAVID JUÁREZ VALENCIA	
ESCALA 2:1	BASE SOPORTE DE BATERIA		NUMERO 1/4

	FECHA	NOMBRE	UNIVERSIDAD CATOLICA SANTA MARIA
DIBUJADO	25/07/2019	JOSUÉ DAVID JUÁREZ VALENCIA	
ESCALA 2:1	PIEZA DE CONTENCION PARA BATERIA		NUMERO 2/4



	FECHA	NOMBRE	UNIVERSIDAD CATOLICA SANTA MARIA
DIBUJADO	25/07/2019	JOSUÉ DAVID JUÁREZ VALENCIA	
ESCALA 2:1	AJUSTE DE BATERIA		NUMERO 3/4



	FECHA	NOMBRE	UNIVERSIDAD CATOLICA SANTA MARIA
DIBUJADO	25/07/2019	JOSUÉ DAVID JUÁREZ VALENCIA	
ESCALA 3:1	LIMITADOR DE POSICION PARA BATERIA		NUMERO 4/4

ANEXO C

PROGRAMA VEHÍCULO 1

PROGRAMA DEL VEHÍCULO 1 PARA ARDUINO IDE

```
#include <ESP8266WiFi.h>
int m1=12;
int m2=14;
int m3=13;
int m4=16;
int m5=5;
String esc;
const char* ssid = "HUAWEI";
const char* password = "12345678";
WiFiServer server(80);
void setup()
  {  Serial.begin(115200);
     delay(1000);

     pinMode(m1, OUTPUT);
     analogWrite(m1, 0);
     pinMode(m2, OUTPUT);
     analogWrite(m2, 0);
     pinMode(m3, OUTPUT);
     analogWrite(m3, 0);
     pinMode(m4, OUTPUT);
     analogWrite(m4, 0);
     pinMode(m5, OUTPUT);
     digitalWrite(m5, 0);

     Serial.print("Connecting to ");
     Serial.println(ssid);
     WiFi.begin(ssid, password);
     while (WiFi.status() != WL_CONNECTED)
       {   delay(500);
          Serial.print(".");

       }
     Serial.println("WiFi connected");
     server.begin();                  // Iniciamos el servidor
     Serial.println("Server started");
     Serial.println(WiFi.localIP());      // mostramos la IP
     digitalWrite(m5,1);
  }
void loop() {
```

```
  WiFiClient client = server.available();

if (!client)
    return;
Serial.println("new client");
while(!client.available())
    delay(1);
String req = client.readStringUntil('\r');
Serial.println(req);
client.flush();

if (req.indexOf("e1") != -1){
   analogWrite(m1, 0);
   analogWrite(m2, 0);
   analogWrite(m3, 0);
   analogWrite(m4, 0);
   esc="stop";
 }
else if (req.indexOf("e2") != -1){
   analogWrite(m1, 1000);
   analogWrite(m2, 1000);
   analogWrite(m3, 0);
   analogWrite(m4, 0);
   esc="adelante";
}
else if (req.indexOf("e3") != -1){
   analogWrite(m1, 0);
   analogWrite(m2, 0);
   analogWrite(m3, 1000);
   analogWrite(m4, 1000);
   esc="atras";
}
else if (req.indexOf("e4") != -1){
   analogWrite(m1, 1000);
   analogWrite(m2, 0);
   analogWrite(m3, 0);
   analogWrite(m4, 1000);
   esc="derecha";
}
else if (req.indexOf("e5") != -1){
   analogWrite(m1, 0);
   analogWrite(m2, 1000);
```

```
   analogWrite(m3, 1000);
   analogWrite(m4, 0);
   esc="izquierda";
}
else if (req.indexOf("Modo1") != -1){
int r=0;
do{
   analogWrite(m1, 1000);
   analogWrite(m2, 1000);
   analogWrite(m3, 0);
   analogWrite(m4, 0);
delay(1000);
   analogWrite(m1, 0);
   analogWrite(m2, 0);
   analogWrite(m3, 0);
   analogWrite(m4, 0);
delay(500);
   analogWrite(m1, 0);
   analogWrite(m2, 0);
   analogWrite(m3, 1000);
   analogWrite(m4, 1000);
delay(1000);
   analogWrite(m1, 0);
   analogWrite(m2, 0);
   analogWrite(m3, 0);
   analogWrite(m4, 0);
delay(500);
r=r+1;
}while (r<=4);
   analogWrite(m1, 0);
   analogWrite(m2, 0);
   analogWrite(m3, 0);
   analogWrite(m4, 0);
esc="Finalizado";
}

else if (req.indexOf("Modo2") != -1){
int r=0;
do{
   analogWrite(m1, 1000);
   analogWrite(m2, 1000);
   analogWrite(m3, 0);
```

```
    analogWrite(m4, 0);
delay(1000);
    analogWrite(m1, 1000);
    analogWrite(m2, 0);
    analogWrite(m3, 0);
    analogWrite(m4, 1000);
delay(350);
    analogWrite(m1, 1000);
    analogWrite(m2, 1000);
    analogWrite(m3, 0);
    analogWrite(m4, 0);
delay(1000);
    analogWrite(m1, 1000);
    analogWrite(m2, 0);
    analogWrite(m3, 0);
    analogWrite(m4, 1000);
delay(350);
    analogWrite(m1, 1000);
    analogWrite(m2, 1000);
    analogWrite(m3, 0);
    analogWrite(m4, 0);
delay(1000);
    analogWrite(m1, 1000);
    analogWrite(m2, 0);
    analogWrite(m3, 0);
    analogWrite(m4, 1000);
delay(350);
    analogWrite(m1, 1000);
    analogWrite(m2, 1000);
    analogWrite(m3, 0);
    analogWrite(m4, 0);
delay(1000);
    analogWrite(m1, 1000);
    analogWrite(m2, 0);
    analogWrite(m3, 0);
    analogWrite(m4, 1000);
delay(350);
r=r+1;
}while (r<=2);
    analogWrite(m1, 0);
    analogWrite(m2, 0);
    analogWrite(m3, 0);
```

```
   analogWrite(m4, 0);
esc="Finalizado";
}
else if (req.indexOf("Modo3") != -1){
int r=0;
do{
   analogWrite(m1, 1000);
   analogWrite(m2, 0);
   analogWrite(m3, 0);
   analogWrite(m4, 1000);
delay(350);
   analogWrite(m1, 1000);
   analogWrite(m2, 1000);
   analogWrite(m3, 0);
   analogWrite(m4, 0);
delay(1000);
   analogWrite(m1, 0);
   analogWrite(m2, 0);
   analogWrite(m3, 1000);
   analogWrite(m4, 1000);
delay(1000);
   analogWrite(m1, 0);
   analogWrite(m2, 1000);
   analogWrite(m3, 1000);
   analogWrite(m4, 0);
delay(600);
   analogWrite(m1, 1000);
   analogWrite(m2, 1000);
   analogWrite(m3, 0);
   analogWrite(m4, 0);
delay(1000);
   analogWrite(m1, 0);
   analogWrite(m2, 0);
   analogWrite(m3, 1000);
   analogWrite(m4, 1000);
delay(1000);
   analogWrite(m1, 1000);
   analogWrite(m2, 0);
   analogWrite(m3, 0);
   analogWrite(m4, 1000);
delay(350);
r=r+1;
```

```
}while (r<=2);
   analogWrite(m1, 0);
   analogWrite(m2, 0);
   analogWrite(m3, 0);
   analogWrite(m4, 0);
esc="Finalizado";
}
else if (req.indexOf("Modo4") != -1){
int r=0;
do{
   analogWrite(m1, 1000);
   analogWrite(m2, 1000);
   analogWrite(m3, 0);
   analogWrite(m4, 0);
delay(1000);
   analogWrite(m1, 0);
   analogWrite(m2, 0);
   analogWrite(m3, 1000);
   analogWrite(m4, 1000);
delay(1000);
   analogWrite(m1, 1000);
   analogWrite(m2, 0);
   analogWrite(m3, 00);
   analogWrite(m4, 1000);
delay(1300);
   analogWrite(m1, 0);
   analogWrite(m2, 0);
   analogWrite(m3, 0);
   analogWrite(m4, 0);
delay(600);
r=r+1;
}while (r<=3);
   analogWrite(m1, 0);
   analogWrite(m2, 0);
   analogWrite(m3, 0);
   analogWrite(m4, 0);
esc="Finalizado";
}
String s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!DOCTYPE
HTML>\r\n<html>\r\nGPIO<br /> estado ";
s += "<br />";
s += (esc);
```

```
s += "</html>\n";

client.print(s);  //
delay(1);
Serial.println("Client disonnected");
}
```

ANEXO D

PROGRAMA VEHÍCULO 2

PROGRAMA DEL VEHÍCULO 2 PARA ARDUINO IDE

```
#include <ESP8266WiFi.h>
#include <Servo.h>
Servo servoMotor;
int m1=12;
int m2=14;
int m3=13;
int m4=16;
int m5=5;
String esc;
const char* ssid = "HUAWEI";
const char* password = "12345678";
WiFiServer server(80);
void setup()
  {  Serial.begin(115200);
     delay(1000);

     pinMode(m1, OUTPUT);
     analogWrite(m1, 0);
     pinMode(m2, OUTPUT);
     analogWrite(m2, 0);
     pinMode(m3, OUTPUT);
     analogWrite(m3, 0);
     pinMode(m4, OUTPUT);
     analogWrite(m4, 0);
     pinMode(m5, OUTPUT);
     digitalWrite(m5, 0);
servoMotor.attach(4);

     Serial.print("Connecting to ");
     Serial.println(ssid);
     WiFi.begin(ssid, password);
     while (WiFi.status() != WL_CONNECTED)
       {   delay(500);
          Serial.print(".");

       }
     Serial.println("WiFi connected");
     server.begin();                  // Iniciamos el servidor
     Serial.println("Server started");
     Serial.println(WiFi.localIP());      // mostramos la IP
```

```
     digitalWrite(m5,1);              // encendermos un led para mostrar que la conexion
fue lograda
  }
void loop() {
  WiFiClient client = server.available();

if (!client)
    return;
Serial.println("new client");
while(!client.available())
    delay(1);
String req = client.readStringUntil('\r');
Serial.println(req);
client.flush();

if (req.indexOf("e1") != -1){
   analogWrite(m1, 0);
   analogWrite(m2, 0);
   analogWrite(m3, 0);
   analogWrite(m4, 0);
   servoMotor.write(80);
   esc="stop";
 }
else if (req.indexOf("e2") != -1){
   analogWrite(m1, 1000);
   analogWrite(m2, 1000);
   analogWrite(m3, 0);
   analogWrite(m4, 0);
   esc="adelante";
}
else if (req.indexOf("e3") != -1){
   analogWrite(m1, 0);
   analogWrite(m2, 0);
   analogWrite(m3, 1000);
   analogWrite(m4, 1000);
   esc="atras";
}
else if (req.indexOf("e4") != -1){
   servoMotor.write(110);
   esc="derecha";
}
else if (req.indexOf("e5") != -1){
```

```
   servoMotor.write(60);
   esc="izquierda";
}
else if (req.indexOf("e6") != -1){
   servoMotor.write(80);
   esc="centrado";
}
else if (req.indexOf("Modo1") != -1){
char r=0;
do{
   servoMotor.write(80);
   analogWrite(m1, 1000);
   analogWrite(m2, 1000);
   analogWrite(m3, 0);
   analogWrite(m4, 0);
   delay(1000);
   analogWrite(m1, 0);
   analogWrite(m2, 0);
   analogWrite(m3, 0);
   analogWrite(m4, 0);
   delay(500);
   analogWrite(m1, 0);
   analogWrite(m2, 0);
   analogWrite(m3, 1000);
   analogWrite(m4, 1000);
   delay(100);
   analogWrite(m1, 0);
   analogWrite(m2, 0);
   analogWrite(m3, 0);
   analogWrite(m4, 0);
   delay(500);
   r=r+1;
}while (r<=3);
   analogWrite(m1, 0);
   analogWrite(m2, 0);
   analogWrite(m3, 0);
   analogWrite(m4, 0);
esc="Finalizado";
}
else if (req.indexOf("Modo2") != -1){
char r=0;
do{
```

```
   servoMotor.write(80);
   analogWrite(m1, 1000);
   analogWrite(m2, 1000);
   analogWrite(m3, 0);
   analogWrite(m4, 0);
   delay(1000);
   servoMotor.write(110);
   delay(800);
   servoMotor.write(60);
   delay(800);
   servoMotor.write(110);
   delay(800);
   servoMotor.write(60);
   delay(800);
   servoMotor.write(80);
   analogWrite(m1, 0);
   analogWrite(m2, 0);
   analogWrite(m3, 1000);
   analogWrite(m4, 1000);
   delay(1500);
   analogWrite(m1, 0);
   analogWrite(m2, 0);
   analogWrite(m3, 0);
   analogWrite(m4, 0);
   delay(200);
   r=r+1;
}while (r<=3);
   servoMotor.write(80);
   analogWrite(m1, 0);
   analogWrite(m2, 0);
   analogWrite(m3, 0);
   analogWrite(m4, 0);
esc="Finalizado";
}

else if (req.indexOf("Modo3") != -1){
char r=0;
do{
   servoMotor.write(80);
   analogWrite(m1, 1000);
   analogWrite(m2, 1000);
   analogWrite(m3, 0);
```

```
   analogWrite(m4, 0);
   delay(1500);
   servoMotor.write(110);
   delay(2000);
   servoMotor.write(80);
   analogWrite(m1, 1000);
   analogWrite(m2, 1000);
   analogWrite(m3, 0);
   analogWrite(m4, 0);
   delay(1000);
   servoMotor.write(110);
   delay(2000);
   servoMotor.write(80);
   analogWrite(m1, 1000);
   analogWrite(m2, 1000);
   analogWrite(m3, 0);
   analogWrite(m4, 0);
   delay(1000);
   analogWrite(m1, 0);
   analogWrite(m2, 0);
   analogWrite(m3, 0);
   analogWrite(m4, 0);
   delay(200);
   r=r+1;
}while (r<=2);
   servoMotor.write(80);
   analogWrite(m1, 0);
   analogWrite(m2, 0);
   analogWrite(m3, 0);
   analogWrite(m4, 0);
esc="Finalizado";
}
else if (req.indexOf("Modo4") != -1){
char r=0;
do{
   servoMotor.write(80);
   analogWrite(m1, 1000);
   analogWrite(m2, 1000);
   analogWrite(m3, 0);
   analogWrite(m4, 0);
   delay(1000);
   servoMotor.write(110);
```

```
    delay(500);
    analogWrite(m1, 0);
    analogWrite(m2, 0);
    analogWrite(m3, 1000);
    analogWrite(m4, 1000);
    delay(800);
    servoMotor.write(80);
    analogWrite(m1, 1000);
    analogWrite(m2, 1000);
    analogWrite(m3, 0);
    analogWrite(m4, 0);
    delay(200);
    servoMotor.write(60);
    analogWrite(m1, 1000);
    analogWrite(m2, 1000);
    analogWrite(m3, 0);
    analogWrite(m4, 0);
    delay(500);
    analogWrite(m1, 0);
    analogWrite(m2, 0);
    analogWrite(m3, 1000);
    analogWrite(m4, 1000);
    delay(800);
    servoMotor.write(80);
    analogWrite(m1, 1000);
    analogWrite(m2, 1000);
    analogWrite(m3, 0);
    analogWrite(m4, 0);
    delay(200);
    analogWrite(m1, 0);
    analogWrite(m2, 0);
    analogWrite(m3, 0);
    analogWrite(m4, 0);
    delay(50);
    r=r+1;
  }while (r<=3);
    servoMotor.write(80);
    analogWrite(m1, 0);
    analogWrite(m2, 0);
    analogWrite(m3, 0);
    analogWrite(m4, 0);
  esc="Finalizado";
```

```
}

String s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!DOCTYPE
HTML>\r\n<html>\r\nGPIO<br /> estado ";
s += "<br />";
s += (esc);
s += "</html>\n";

client.print(s);  // Enviar el resultado de val al cliente
delay(1);
Serial.println("Client disonnected");
}
```

ANEXO E

DATASHEET ESP8266-12E

ESP-12E WiFi Module

Version1.0

Disclaimer and Copyright Notice.

Information in this document, including URL references, is subject to change without notice.
THIS DOCUMENT IS PROVIDED AS IS WITH NO WARRANTIES WHATSOEVER, INCLUDING ANYWARRANTY OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATIONOR SAMPLE. All liability, including liability for infringement of any proprietary rights, relating to use of information in this document is disclaimed. No licenses express or implied, by estoppel or otherwise, to any intellectual property rights are granted herein.
The WiFi Alliance Member Logo is a trademark of the WiFi Alliance.
All trade names, trademarks and registered trademarks mentioned in this document are property of their respective owners, and are hereby acknowledged.

Notice

Product version upgrades or other reasons, possible changes in the contents of this manual. AI-Thinker reserves in the absence of any notice or indication of the circumstances the right to modify the content of this manual. This manual is used only as a guide, Ai-thinker make every effort to provide accurate information in this manual, but Ai-thinker does not ensure that manual content without error, in this manual all statements, information and advice nor does it constitute any express or implied warranty.

Table of Contents

1. Preambles

ESP-12E WiFi module is developed by Ai-thinker Team. core processor ESP8266 in smaller sizes of the module encapsulates Tensilica L106 integrates industry-leading ultra low power 32-bit MCU micro, with the 16-bit short mode, Clock speed support 80 MHz, 160 MHz, supports the RTOS, integrated Wi-Fi MAC/BB/RF/PA/LNA, on-board antenna.

The module supports standard IEEE802.11 b/g/n agreement, complete TCP/IP protocol stack. Users can use the add modules to an existing device networking, or building a separate network controller.

ESP8266 is high integration wireless SOCs, designed for space and power constrained mobile platform designers. It provides unsurpassed ability to embed Wi-Fi capabilities within other systems, or to function as a standalone application, with the lowest cost, and minimal space requirement.

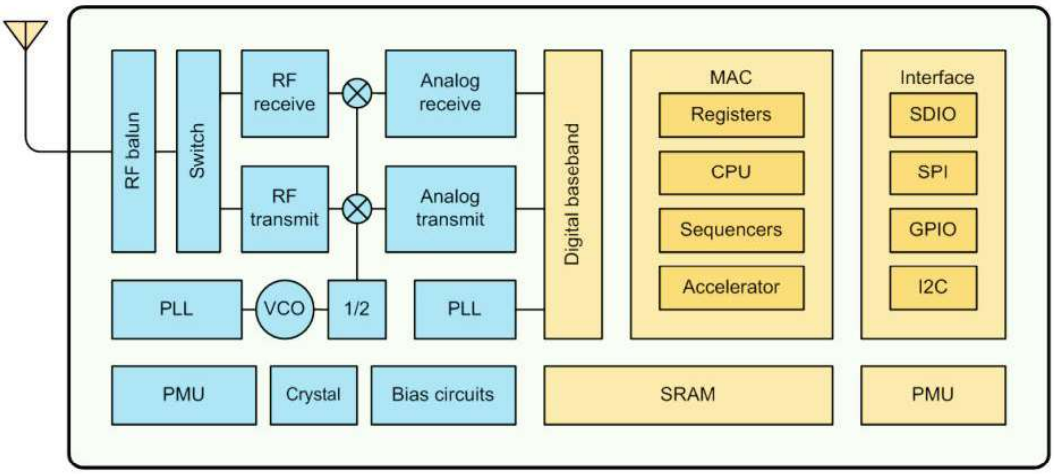


Figure 1 ESP8266EX Block Diagram

ESP8266EX offers a complete and self-contained Wi-Fi networking solution; it can be used to host the application or to offload Wi-Fi networking functions from another application processor.

When ESP8266EX hosts the application, it boots up directly from an external flash. In has integrated cache to improve the performance of the system in such applications.

Alternately, serving as a Wi-Fi adapter, wireless internet access can be added to any micro controllerbased design with simple connectivity (SPI/SDIO or I2C/UART interface).

ESP8266EX is among the most integrated WiFi chip in the industry; it integrates the antenna switches, RF balun, power amplifier, low noise receive amplifier, filters, power management modules, it requires minimal external circuitry, and the entire solution, including front-end module, is designed to occupy minimal PCB area.

ESP8266EX also integrates an enhanced version of Tensilica's L106 Diamond series 32-bit processor, with on-chip SRAM, besides the Wi-Fi functionalities. ESP8266EX is often integrated with external sensors and other application specific devices through its GPIOs; codes for such applications are provided in examples in the SDK.

Espressif Systems' Smart Connectivity Platform (ESCP) demonstrates sophisticated system-level features include fast sleep/wake context switching for energy-efficient VoIP, adaptive radio biasing. for low-power operation, advance signal processing, and spur cancellation and radio co-existence features for common cellular, Bluetooth, DDR, LVDS, LCD interference mitigation.

1.1. Features

- 802.11 b/g/n
- Integrated low power 32-bit MCU
- Integrated 10-bit ADC
- Integrated TCP/IP protocol stack
- Integrated TR switch, balun, LNA, power amplifier and matching network
- Integrated PLL, regulators, and power management units
- Supports antenna diversity
- Wi-Fi 2.4 GHz, support WPA/WPA2
- Support STA/AP/STA+AP operation modes
- Support Smart Link Function for both Android and iOS devices
- Support Smart Link Function for both Android and iOS devices
- SDIO 2.0, (H) SPI, UART, I2C, I2S, IRDA, PWM, GPIO
- STBC, 1x1 MIMO, 2x1 MIMO
- A-MPDU & A-MSDU aggregation and 0.4s guard interval

- Deep sleep power <10uA, Power down leakage current < 5uA
- Wake up and transmit packets in < 2ms
- Standby power consumption of < 1.0mW (DTIM3)
- +20dBm output power in 802.11b mode
- Operating temperature range -40C ~ 125C

1.2. Parameters

Table 1 below describes the major parameters.

Table 1 Parameters

Categories	Items	Values
WiFi Paramters	WiFi Protocles	802.11 b/g/n
	Frequency Range	2.4GHz-2.5GHz (2400M-2483.5M)
Hardware Paramaters	Peripheral Bus	UART/HSPI/I2C/I2S/Ir Remote Contorl
		GPIO/PWM
	Operating Voltage	3.0~3.6V
	Operating Current	Average value: 80mA
	Operating Temperature Range	-40°~125°
	Ambient Temperature Range	Normal temperature
	Package Size	16mm*24mm*3mm
	External Interface	N/A
Software Parameters	Wi-Fi mode	station/softAP/SoftAP+station
	Security	WPA/WPA2
	Encryption	WEP/TKIP/AES
	Firmware Upgrade	UART Download / OTA (via network) / download and write firmware via host
	Ssoftware Development	Supports Cloud Server Development / SDK for custom firmware development
	Network Protocols	IPv4, TCP/UDP/HTTP/FTP
	User Configuration	AT Instruction Set, Cloud Server, Android/iOS App

2. Pin Descriptions

There are altogether 22 pin counts, the definitions of which are described in Table 2 below。

Table 2 ESP-12E Pin design

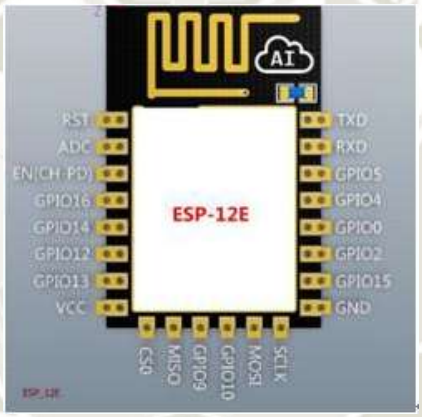


Table 3 Pin Descriptions

NO.	Pin Name	Function
1	RST	Reset the module
2	ADC	A/D Conversion result.Input voltage range 0-1v,scope:0-1024
3	EN	Chip enable pin.Active high
4	IO16	GPIO16; can be used to wake up the chipset from deep sleep mode.
5	IO14	GPIO14; HSPI_CLK
6	IO12	GPIO12; HSPI_MISO
7	IO13	GPIO13; HSPI_MOSI; UART0_CTS
8	VCC	3.3V power supply (VDD)
9	CS0	Chip selection
10	MISO	Salve output Main input

11	IO9	GPIO9
12	IO10	GBIO10
13	MOSI	Main output slave input
14	SCLK	Clock
15	GND	GND
16	IO15	GPIO15; MTDO; HSPICS; UART0_RTS
17	IO2	GPIO2; UART1_TXD
18	IO0	GPIO0
19	IO4	GPIO4
20	IO5	GPIO5
21	RXD	UART0_RXD; GPIO3
22	TXD	UART0_TXD; GPIO1

Table 4 Pin Mode

Mode	GPIO15	GPIO0	GPIO2
UART	Low	Low	High
Flash Boot	Low	High	High

Table 5 Receiver Sensitivity

Parameters	Min	Typical	Max	Unit
Input frequency	2412		2484	MHz
Input impedance		50		Ω
Input reflection			-10	dB
Output power of PA for 72.2Mbps	15.5	16.5	17.5	dBm
Output power of PA for 11b mode	19.5	20.5	21.5	dBm
Sensitivity				
DSSS, 1Mbps		-98		dBm
CCK, 11Mbps		-91		dBm
6Mbps (1/2 BPSK)		-93		dBm
54Mbps (3/4 64-QAM)		-75		dBm
HT20, MCS7 (65Mbps, 72.2Mbps)		-72		dBm
Adjacent Channel Rejection				
OFDM, 6Mbps		37		dB
OFDM, 54Mbps		21		dB
HT20, MCS0		37		dB
HT20, MCS7		20		dB

3. Packaging and Dimension

The external size of the module is 16mm*24mm*3mm, as is illustrated in Figure 3 below. The type of flash integrated in this module is an SPI flash, the capacity of which is 4 MB, and the package size of which is SOP-210mil. The antenna applied on this module is a 3DBi PCB-on-board antenna.

Figure 3 [Module Pin Counts, 22 pin, 16 mm *24 mm *3 mm]

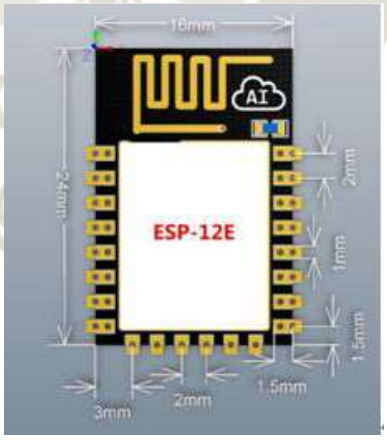


Figure 4 Top View of ESP-12E WiFi Module

Table 5 Dimension of ESP-12E WiFi Modul

Length	Width	Height	PAD Size(Bottom)	Pin Pitch
16 mm	24mm	3 mm	0.9 mm x 1.7 mm	2mm

4. Functional Descriptions

4.1. MCU

ESP8266EX is embedded with Tensilica L106 32-bit micro controller (MCU), which features extra low power consumption and 16-bit RSIC. The CPU clock speed is 80MHz. It can also reach a maximum value of 160MHz. ESP8266EX is often integrated with external sensors and other specific devices through its GPIOs; codes for such applications are provided in examples in the SDK.

4.2. Memory Organization

4.2.1. Internal SRAM and ROM

ESP8266EX WiFi SoC is embedded with memory controller, including SRAM and ROM. MCU can visit the memory units through iBus, dBus, and AHB interfaces. All memory units can be visited upon request, while a memory arbiter will decide the running sequence according to the time when these requests are received by the processor.

According to our current version of SDK provided, SRAM space that is available to users is assigned as below:

▪RAM size < 36kB, that is to say, when ESP8266EX is working under the station mode and is connected to the router, programmable space accessible to user in heap and data section is around 36kB.)

▪ There is no programmable ROM in the SoC, therefore, user program must be stored in an external SPI flash.

4.2.2. External SPI Flash

This module is mounted with an 4 MB external SPI flash to store user programs. If larger definable storage space is required, a SPI flash with larger memory size is preferred. Theoretically speaking, up to 16 MB memory capacity can be supported.

Suggested SPI Flash memory capacity:

▪OTA is disabled: the minimum flash memory that can be supported is 512 kB;

▪OTA is enabled: the minimum flash memory that can be supported is 1 MB.

Several SPI modes can be supported, including Standard SPI, Dual SPI, and Quad SPI.

Therefore, please choose the correct SPI mode when you are downloading into the flash, otherwise firmwares/programs that you downloaded may not work in the right way.

4.3. Crystal

Currently, the frequency of crystal oscillators supported include 40MHz, 26MHz and 24MHz. The accuracy of crystal oscillators applied should be ±10PPM, and the operating temperature range should be between -20°C and 85°C.

When using the downloading tools, please remember to select the right crystal oscillator type. In circuit design, capacitors C1 and C2, which are connected to the earth, are added to the input and output terminals of the crystal oscillator respectively. The values of the two capacitors can be flexible, ranging from 6pF to 22pF, however, the specific capacitive values of C1 and C2 depend on further testing and adjustment on the overall performance of the whole circuit. Normally, the capacitive values of C1 and C2 are within 10pF if the crystal oscillator frequency is 26MHz, while the values of C1 and C2 are 10pF<C1, C2<22pF if the crystal oscillator frequency is 40MHz.

4.4. Interfaces

Table 6 Descriptions of Interfaces

Interface	Pin Name	Description
HSPI	IO12(MISO) IO13(MOSI) IO14(CLK) IO15(CS)	SPI Flash 2, display screen, and MCU can be connected using HSPI interface.
PWM	IO12(R) IO15(G) IO13(B)	Currently the PWM interface has four channels, but users can extend the channels according to their own needs. PWM interface can be used to control LED lights, buzzers, relays, electronic machines, and so on.
IR Remote Control	IO14(IR_T) IO5(IR_R)	The functionality of Infrared remote control interface can be implemented via software programming. NEC coding, modulation, and demodulation are used by this interface. The frequency of modulated carrier signal is 38KHz.
ADC	TOUT	ESP8266EX integrates a 10-bit analog ADC. It can be used to test the power-supply voltage of VDD3P3 (Pin3 and Pin4) and the input power voltage of TOUT (Pin 6). However, these two functions cannot be used simultaneously. This interface is typically used in sensor products.
I2C	IO14(SCL) IO2(SDA)	I2C interface can be used to connect external sensor products and display screens, etc.

Interface	Pin Name	Description
UART	**UART0:** TXD (U0TXD) RXD (U0RXD) IO15 (RTS) IO13 (CTS) **UART1:** IO2(TXD)	Devices with UART interfaces can be connected with the module. Downloading: U0TXD+U0RXD or GPIO2+U0RXD Communicating: UART0: U0TXD, U0RXD, MTDO (U0RTS), MTCK (U0CTS) Debugging: UART1_TXD (GPIO2) can be used to print debugging information. By default, UART0 will output some printed information when the device is powered on and is booting up. If this issue exerts influence on some specific applications, users can exchange the inner pins of UART when initializing, that is to say, exchange U0TXD, U0RXD with U0RTS, U0CTS.
I2S	**I2S Input:** IO12 (I2SI_DATA) ; IO13 (I2SI_BCK); IO14 (I2SI_WS); **I2S Output:** IO15 (I2SO_BCK); IO3 (I2SO_DATA); IO2 (I2SO_WS).	I2S interface is mainly used for collecting, processing, and transmission of audio data.

4.5. Absolute Maximum Ratings

Table 7 Absolute Maximum Ratings

Rating	Condition	Value	Unit
Storage Temperature		-40 to 125	°C
Maximum Soldering Temperature		260	°C
Supply Voltage	IPC/JEDEC J-STD-020	+3.0 to +3.6	V

4.6. Recommended Operating Conditions

Table 8 Recommended Operating Conditions

Operating Condition	Symbol	Min	Typ	Max	Unit
Operating Temperature		-40	20	125	°C
Supply voltage	VDD	3.0	3.3	3.6	V

4.7. Digital Terminal Characteristics

Table 9 Digital Terminal Characteristics

Terminals	Symbol	Min	Typ	Max	Unit
Input logic level low	V_{IL}	-0.3		0.25VDD	V
Input logic level high	V_{IH}	0.75VDD		VDD+0.3	V
Output logic level low	V_{OL}	N		0.1VDD	V
Output logic level high	V_{OH}	0.8VDD		N	V

Note: Test conditions: VDD = 3.3V, Temperature = 20 ℃, if nothing special is stated.

5. RF Performance

Description	Min.	Typ.	Max	Unit
Input frequency	2400		2483.5	MHz
Input impedance		50		ohm
Input reflection			-10	dB
Output power of PA for 72.2Mbps	15.5	16.5	17.5	dBm
Output power of PA for 11b mode	19.5	20.5	21.5	dBm
Sensitivity				
CCK, 1Mbps		-98		dBm
CCK, 11Mbps		-91		dBm
6Mbps (1/2 BPSK)		-93		dBm
54Mbps (3/4 64-QAM)		-75		dBm
HT20, MCS7 (65Mbps, 72.2Mbps)		-72		dBm
Adjacent Channel Rejection				
OFDM, 6Mbps		37		dB
OFDM, 54Mbps		21		dB
HT20, MCS0		37		dB
HT20, MCS7		20		dB

Table 10 RF Performance

6. Power Consumption

Parameters	Min	Typical	Max	Unit
Tx802.11b, CCK 11Mbps, P OUT=+17dBm		170		mA
Tx 802.11g, OFDM 54Mbps, P OUT =+15dBm		140		mA
Tx 802.11n, MCS7, P OUT =+13dBm		120		mA
Rx 802.11b, 1024 bytes packet length , -80dBm		50		mA
Rx 802.11g, 1024 bytes packet length, -70dBm		56		mA
Rx 802.11n, 1024 bytes packet length, -65dBm		56		mA
Modem-Sleep①		15		mA
Light-Sleep②		0.9		mA
Deep-Sleep③		10		uA

Table 11 Power Consumption

❶Modem-Sleep requires the CPU to be working, as in PWM or I2S applications. According to 802.11 standards (like U-APSD), it saves power to shut down the Wi-Fi Modem circuit while maintaining a Wi-Fi connection with no data transmission. E.g. in DTIM3, to maintain a sleep 300mswake 3ms cycle to receive AP's Beacon packages, the current is about 15mA.

❷ During Light-Sleep, the CPU may be suspended in applications like Wi-Fi switch. Without data transmission, the Wi-Fi Modem circuit can be turned off and CPU suspended to save power according to the 802.11 standard (U-APSD). E.g. in DTIM3, to maintain a sleep 300ms-wake 3ms cycle to receive AP's Beacon packages, the current is about 0.9mA.

❸ Deep-Sleep does not require Wi-Fi connection to be maintained. For application with long time lags between data transmission, e.g. a temperature sensor that checks the temperature every 100s ,sleep 300s and waking up to connect to the AP (taking about 0.3~1s), the overall average current is less than 1mA.

7. Reflow Profile

Table 12 Instructions

T_S max to T_L (Ramp-up Rate)	3°C/second max
Preheat Temperature Min.(T_S Min.) Temperature Typical.(T_S Typ.) Temperature Min.(T_S Max.) Time(T_S)	 150°C 175°C 200°C 60~180 seconds
Ramp-up rate (T_L to T_P)	3°C/second max
Time Maintained Above: --Temperature(T_L)/Time(T_L)	 217°C/60~150 seconds
Peak Temperature(T_P)	260°C max. for 10 seconds
Target Peak Temperature (T_P Target)	260°C +0/-5°C
Time within 5°C of actual peak(t_P)	20~40 seconds
T_S max to T_L (Ramp-down Rate)	6°C/second max
Tune 25°C to Peak Temperature (t)	8 minutes max

8. Schematics

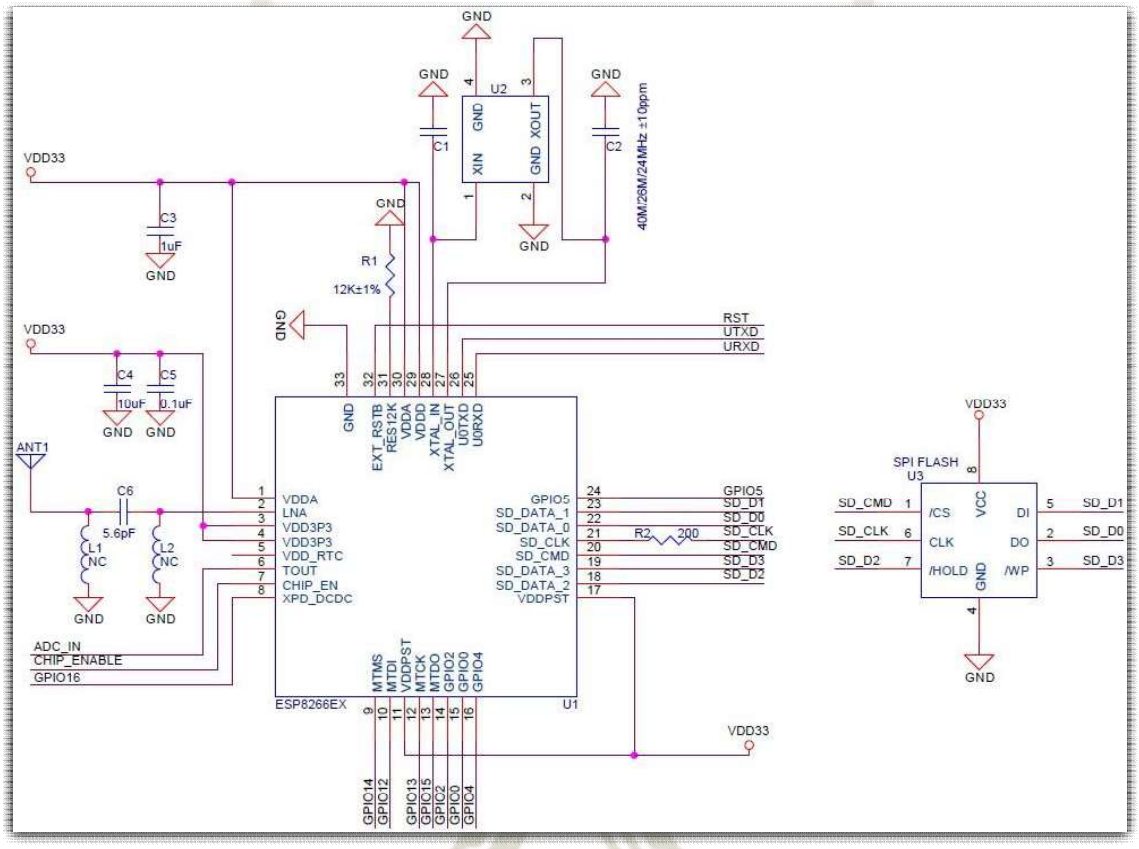


Figure 4 Schematics of Esp-12E WiFi Module

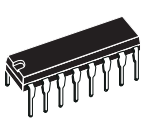
ANEXO F

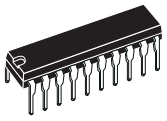
DATASHEET L293D

L293B
L293E

PUSH-PULL FOUR CHANNEL DRIVERS

- OUTPUT CURRENT 1A PER CHANNEL
- PEAK OUTPUT CURRENT 2A PER CHANNEL (non repetitive)
- INHIBIT FACILITY
- HIGH NOISE IMMUNITY
- SEPARATE LOGIC SUPPLY
- OVERTEMPERATURE PROTECTION





DIP16 **POWERDIP(16+2+2)**

ORDERING NUMBERS:

L293B **L293E**

DESCRIPTION

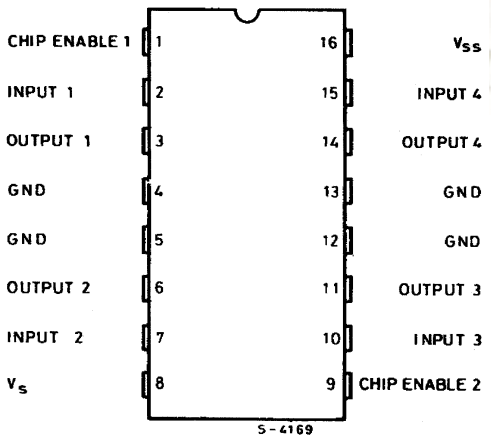
The L293B and L293E are quad push-pull drivers capable of delivering output currents to 1A per channel. Each channel is controlled by a TTL-compatible logic input and each pair of drivers (a full bridge) is equipped with an inhibit input which turns off all four transistors. A separate supply input is provided for the logic so that it may be run off a lower voltage to reduce dissipation.

Additionally, the L293E has external connection of sensing resistors, for switchmode control.

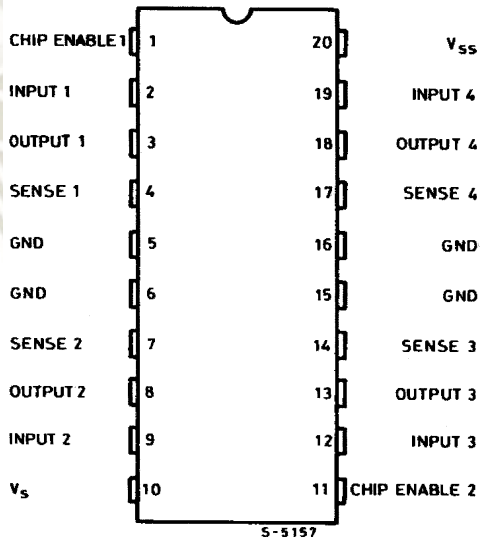
The L293B and L293E are package in 16 and 20-pin plastic DIPs respectively ; both use the four center pins to conduct heat to the printed circuit board.

PIN CONNECTION (Top view)

DIP16 - L293B

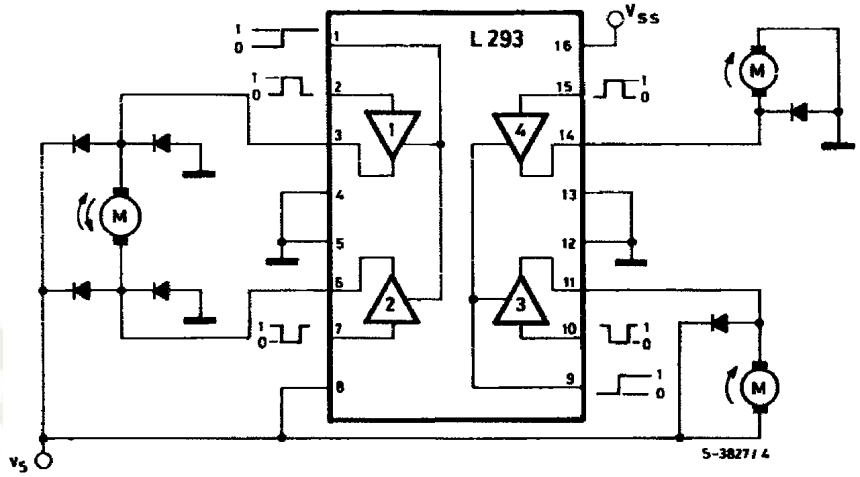


POWERDIP (16+2+2) - L293E

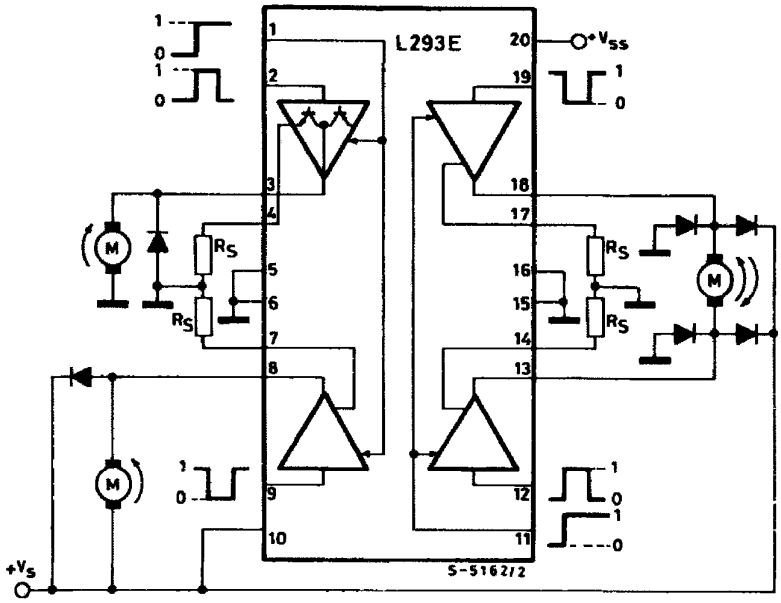


BLOCK DIAGRAMS

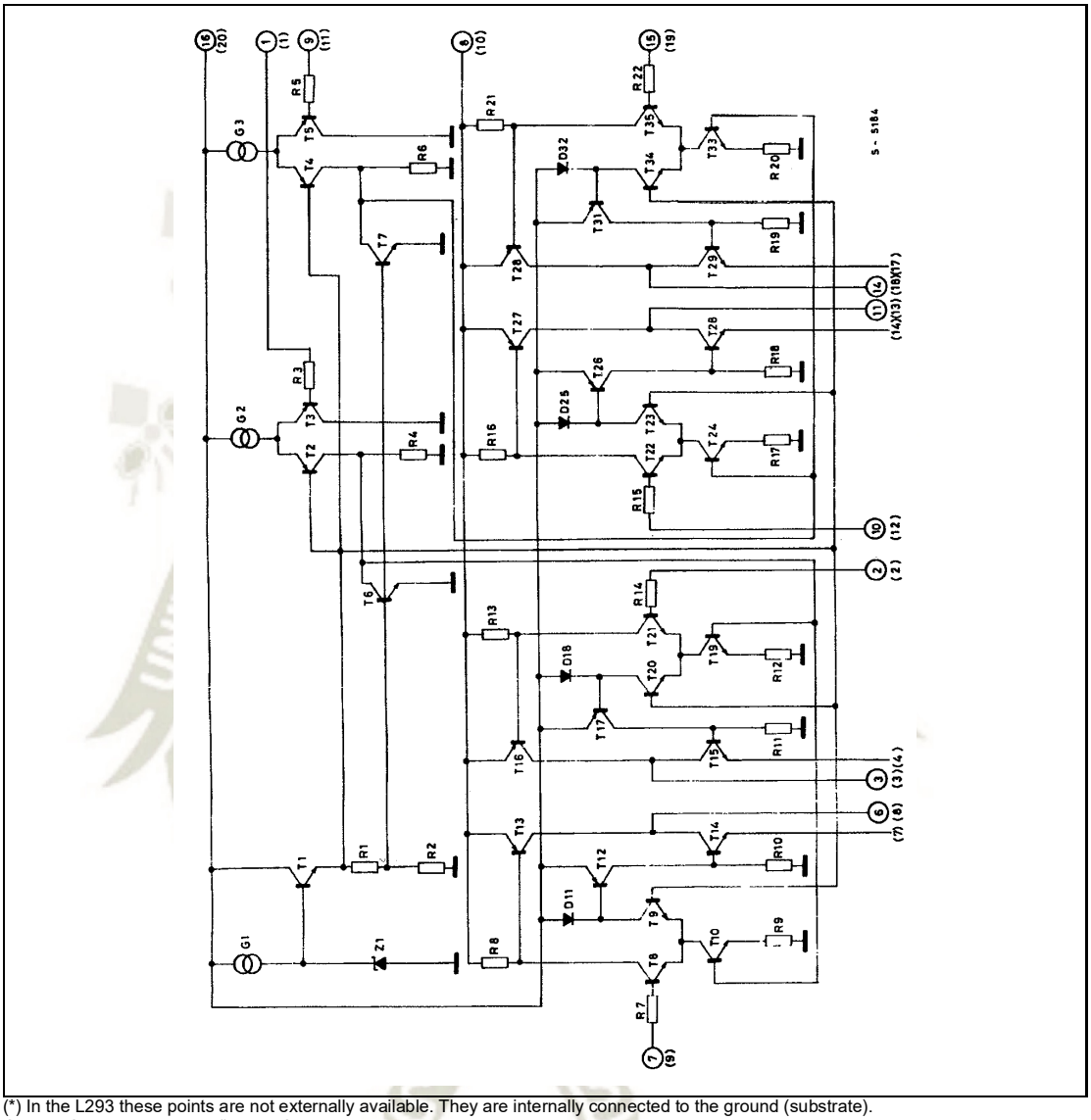
DIP16 - L293B



POWERDIP (16+2+2) - L293E



SCHEMATIC DIAGRAM



(*) In the L293 these points are not externally available. They are internally connected to the ground (substrate).
O Pins of L293 () Pins of L293E.

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_s	Supply Voltage	36	V
V_{ss}	Logic Supply Voltage	36	V
V_i	Input Voltage	7	V
V_{inh}	Inhibit Voltage	7	V
I_{out}	Peak Output Current (non repetitive t = 5ms)	2	A
P_{tot}	Total Power Dissipation at $T_{ground\text{-}pins}$ = 80°C	5	W
T_{stg}, T_j	Storage and Junction Temperature	–40 to +150	°C

THERMAL DATA

Symbol	Parameter		Value	Unit
$R_{th\ j\text{-}case}$	Thermal Resistance Junction-case	Max.	14	°C/W
$R_{th\ j\text{-}amb}$	Thermal Resistance Junction-ambient	Max.	80	°C/W

ELECTRICAL CHARACTERISTCS

Symbol	Parameter	Test Condition	Min.	Typ.	Max.	Unit
V_s	Supply Voltage		V_{SS}		36	V
V_{SS}	Logic Supply Voltage		4.5		36	V
I_s	Total Quiescent Supply Current	V_i = L; I_o = 0; V_{inh} = H		2	6	mA
		V_i = h; I_o = 0; V_{inh} = H		16	24	mA
		V_{inh} = L			4	mA
I_{ss}	Total Quiescent Logic Supply Current	V_i = L; I_o = 0; V_{inh} = H		44	60	mA
		V_i = h; I_o = 0; V_{inh} = H		16	22	mA
		V_{inh} = L		16	24	mA
V_{iL}	Input Low Voltage		-0.3		1.5	V
V_{iH}	Input High Voltage	$V_{SS} \leq 7V$	2.3		V_{ss}	V
		$V_{SS} > 7V$	2.3		7	V
I_{iL}	Low Voltage Input Current	V_{il} = 1.5V			-10	µA
I_{iH}	High Voltage Input Current	$2.3V \leq V_{IH} \leq V_{SS} - 0.6V$		30	100	µA
V_{inhL}	Inhibit Low Voltage		-0.3		1.5	V
V_{inhH}	Inhibit High Voltage	$V_{SS} \leq 7V$	2.3		V_{ss}	V
		$V_{SS} > 7V$	2.3		7	V
I_{inhL}	Low Voltage Inhibit Current	V_{inhL} = 1.5V		-30	-100	µA
I_{inhH}	High Voltage Inhibit Current	$2.3V \leq V_{inhH} \leq$ Vss- 0.6V			±10	µA
V_{CEsatH}	Source Output Saturation Voltage	I_o = -1A		1.4	1.8	V
V_{CEsatL}	Sink Output Saturation Voltage	I_o = 1A		1.2	1.8	V
V_{SENS}	Sensing Voltage (pins 4, 7, 14, 17) (**)				2	V
t_r	Rise Time	0.1 to 0.9 V_o (*)		250		ns
t_f	Fall Time	0.9 to 0.1 V_o (*)		250		ns
t_{on}	Turn-on Delay	0.5 V_i to 0.5 V_o (*)		750		ns
t_{off}	Turn-off Delay	0.5 V_i to 0.5 V_o (*)		200		ns

* See figure 1
** Referred to L293E

TRUTH TABLE

V_i (each channel)	V_o	V_{inh} (**)
H	H	H
L	L	H
H	X (*)	L
L	X (*)	L

(*) High output impedance
(**) Relative to the considerate channel

Figure 1. Switching Timers

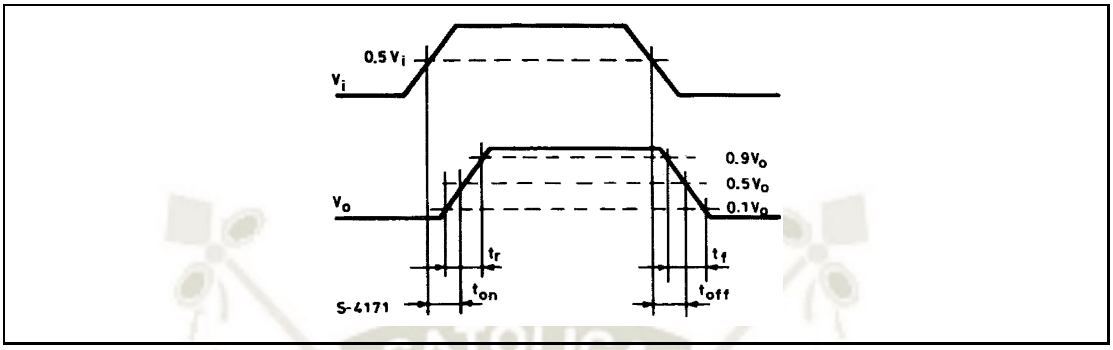


Figure 2. Saturation voltage versus Output Current

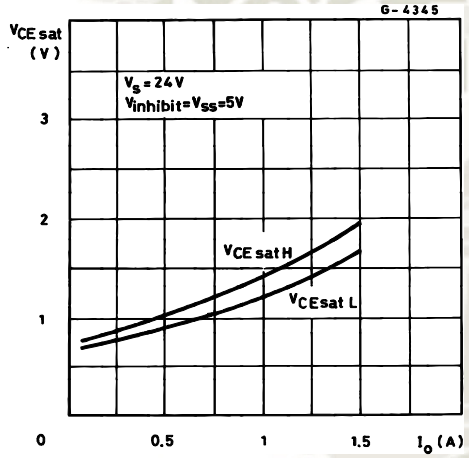


Figure 3. Source Saturation Voltage versus Ambient Temperature

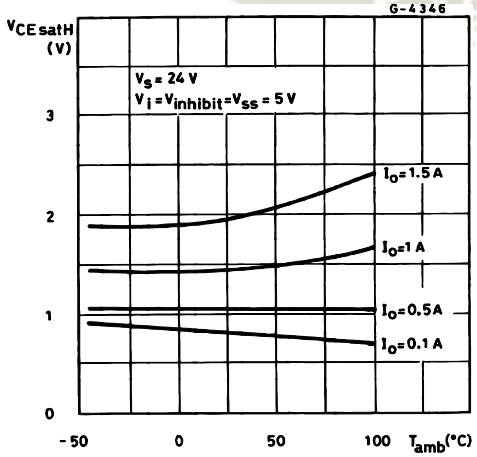


Figure 4. Sink Saturation Voltage versus Ambient Temperature

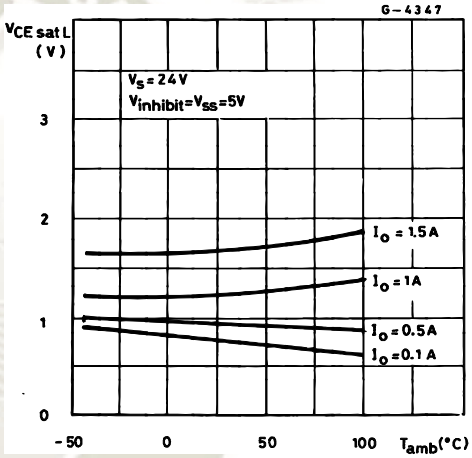


Figure 5. Quiescent Logic Supply Current versus Logic Supply Voltage

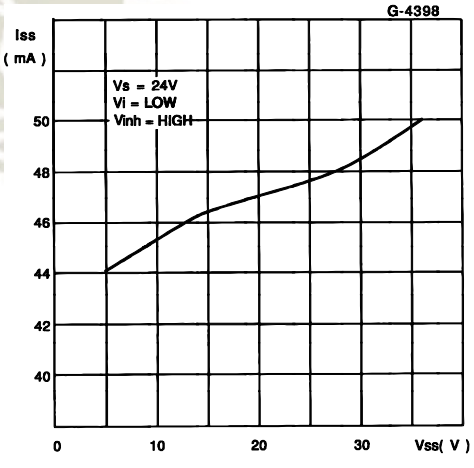


Figure 6. Output Voltage versus Input Voltage

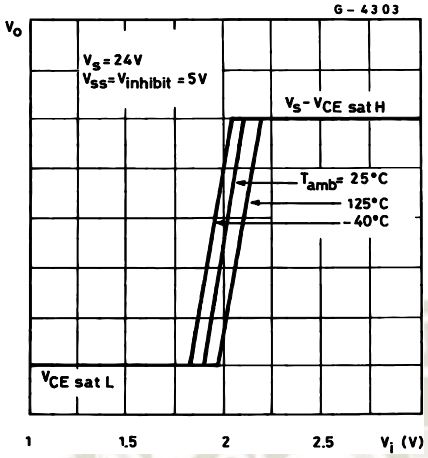
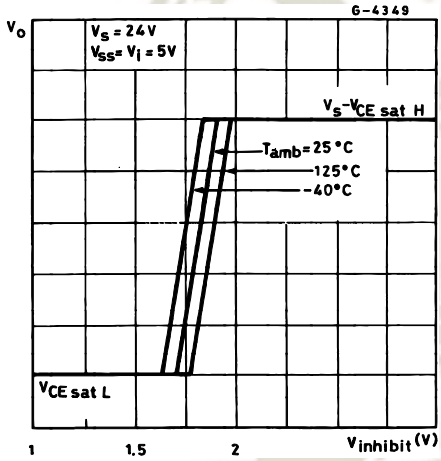
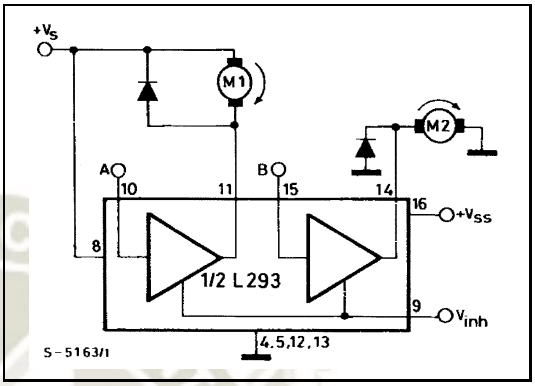


Figure 7. Output Voltage versus Inhibit Voltage



APPLICATION INFORMATION

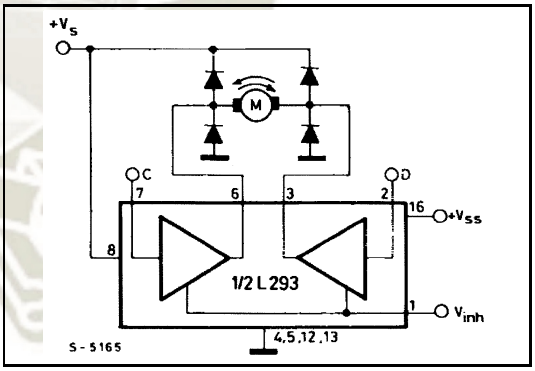
Figure 8. DC Motor Controls (with connection to ground and to the supply voltage)



V_{inh}	A	M1	B	M2
H	H	Fast Motor Stop	H	Run
H	L	Run	L	Fast Motor Stop
L	X	Free Running	X	Free Running
		Motor Stop		Motor Stop

L = Low H = High X = Don't Care

Figure 9. Bidirectional DC Motor Control



Inputs	Function	
V_{inh} = H	C = H ; D = L	Turn Right
	C = L ; D = H	Turn Left
	C = D	Fast Motor Stop
V_{inh} = L	C = X ; D = X	Free Running Motor Stop

L = Low H = High X = Don't Care

Figure 10. Bipolar Stepping Motor Control

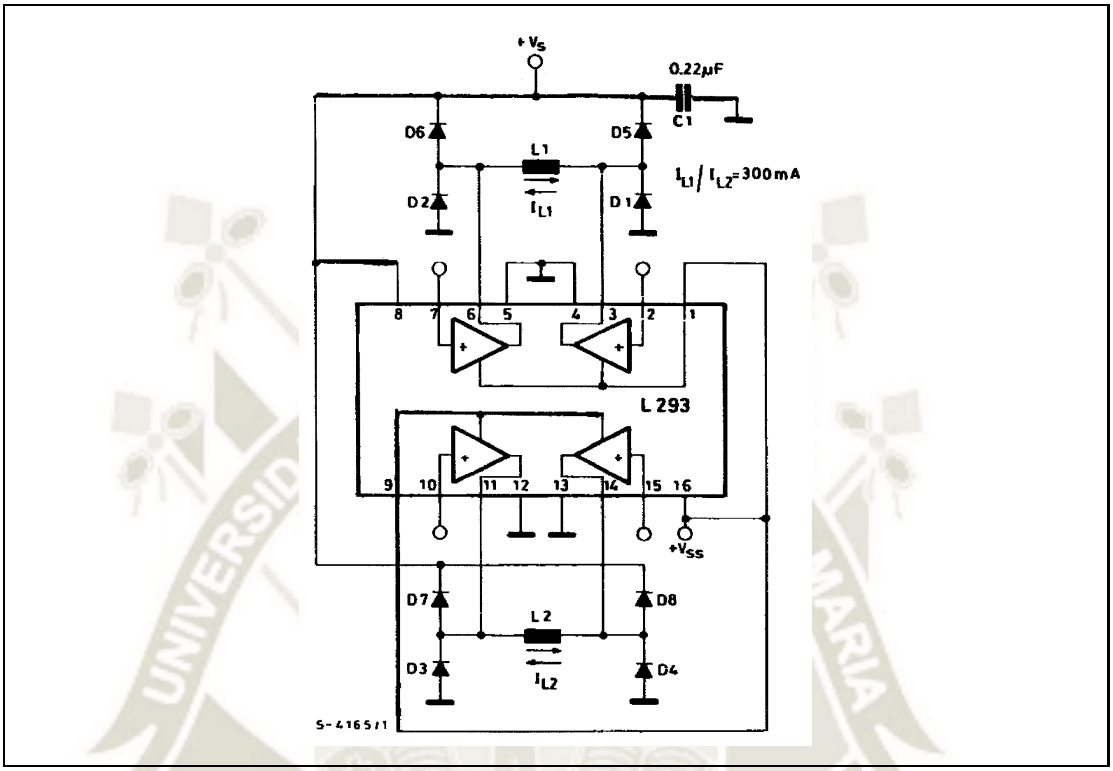
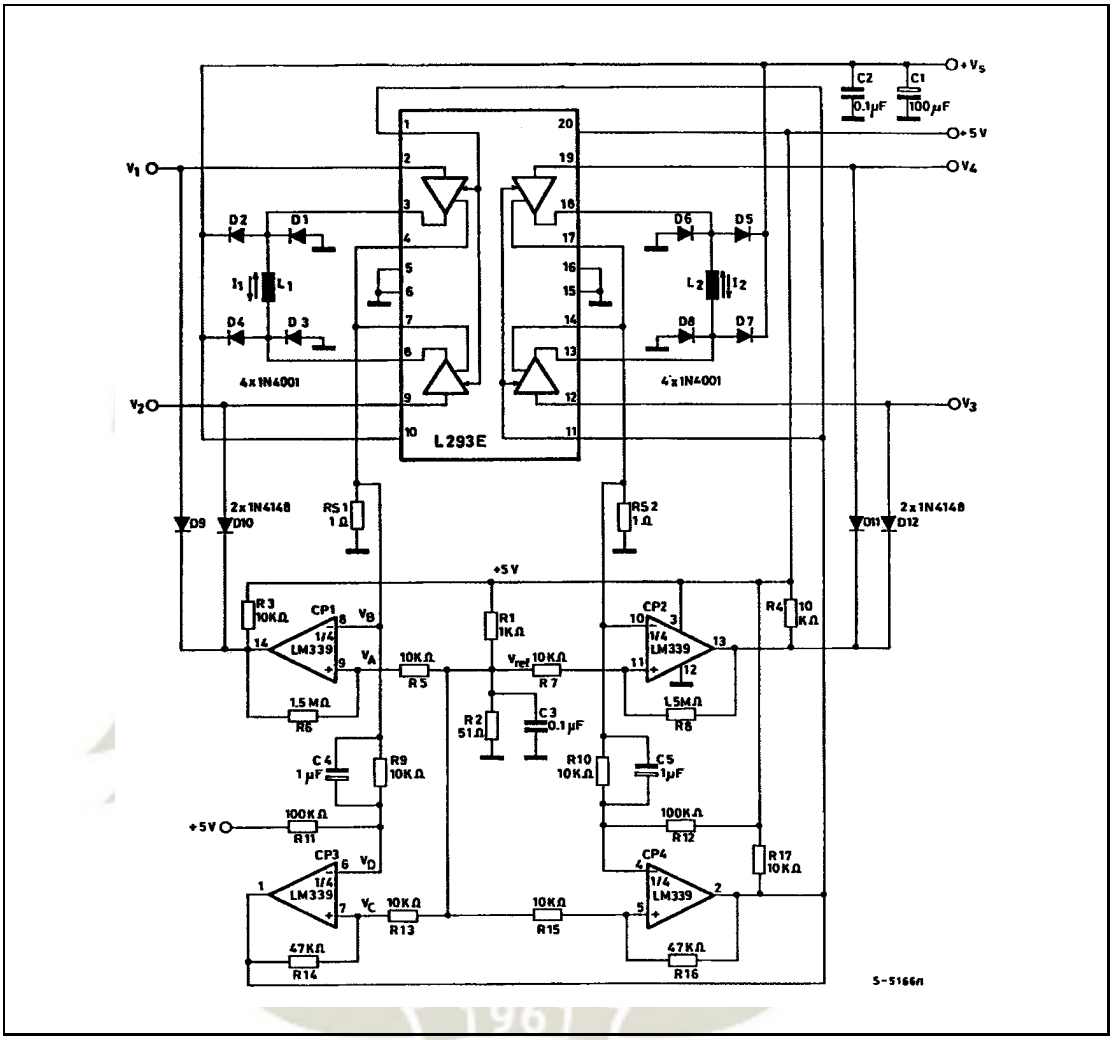


Figure 11. Stepping Motor Driver with Phase Current Control and Short Circuit Protection



MOUNTING INSTRUCTIONS

The $R_{th\ j\text{-}amb}$ of the L293B and the L293E can be reduced by soldering the GND pins to a suitable copper area of the printed circuit board as shown in figure 12 or to an external heatsink (figure 13).

During soldering the pins temperature must not exceed 260°C and the soldering time must not be longer than 12 seconds.

The external heatsink or printed circuit copper area must be connected to electrical ground.

Figure 12. Example of P.C. Board Copper Area which is Used as Heatsink

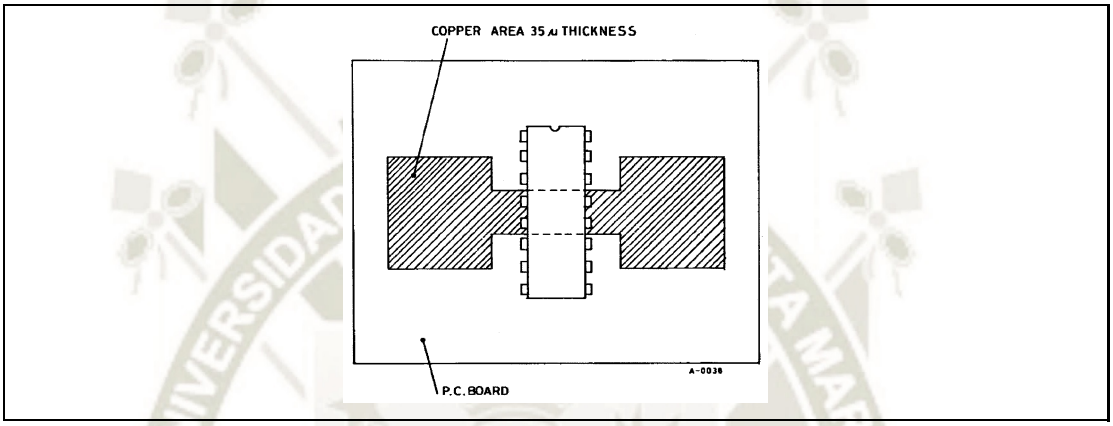
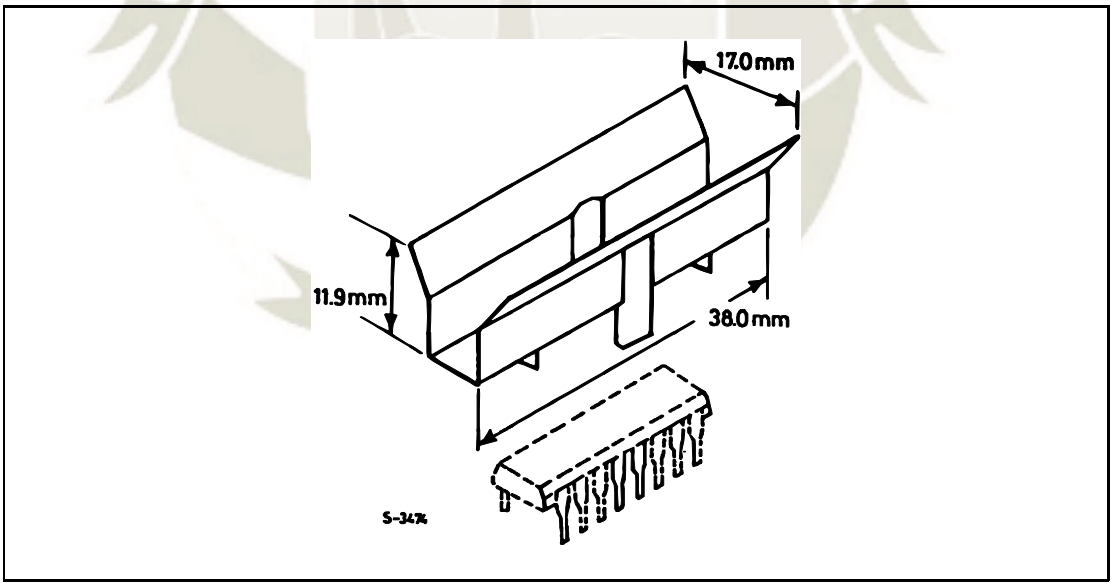
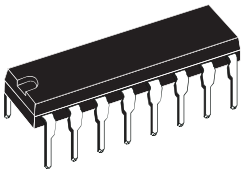


Figure 13. External Heatsink Mounting Example (R_{th} = 30°C/W)

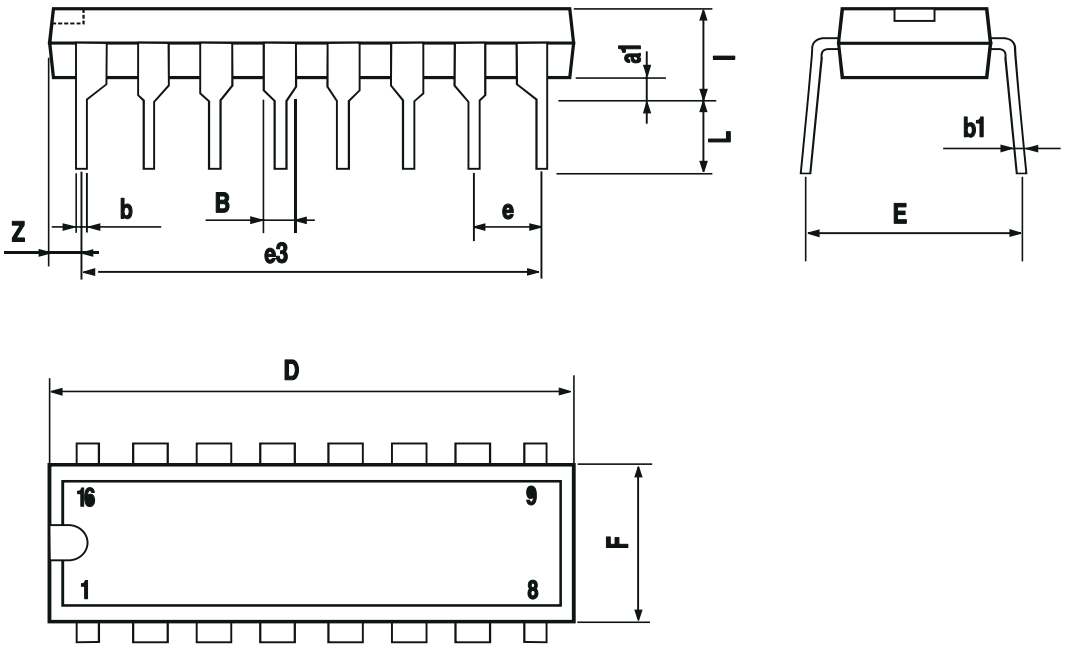


DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
a1	0.51			0.020		
B	0.77		1.65	0.030		0.065
b		0.5			0.020	
b1		0.25			0.010	
D			20			0.787
E		8.5			0.335	
e		2.54			0.100	
e3		17.78			0.700	
F			7.1			0.280
I			5.1			0.201
L		3.3			0.130	
Z			1.27			0.050

OUTLINE AND MECHANICAL DATA

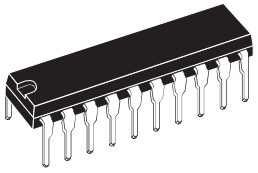


DIP16

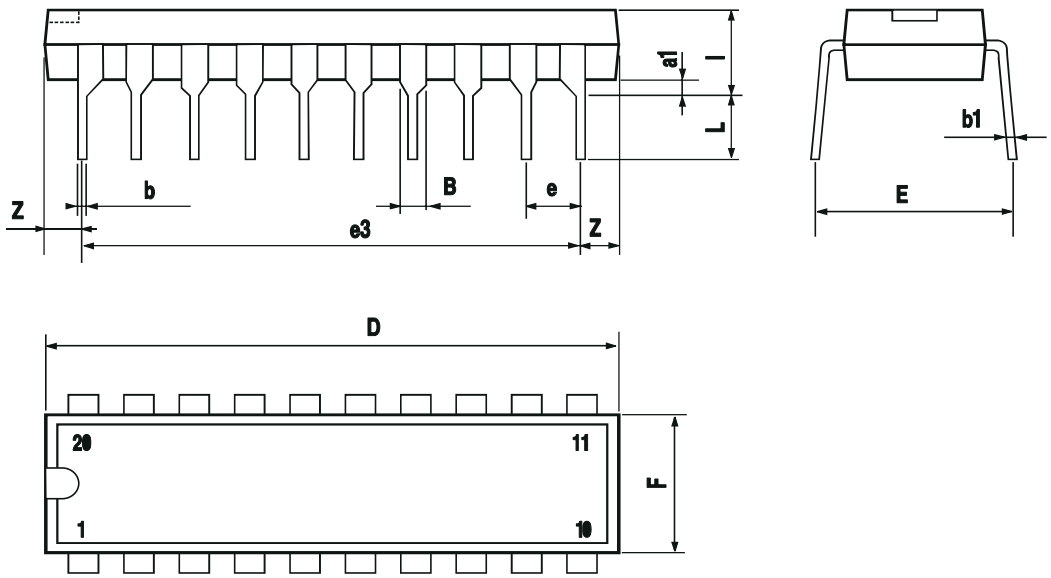


DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
a1	0.51			0.020		
B	0.85		1.40	0.033		0.055
b		0.50			0.020	
b1	0.38		0.50	0.015		0.020
D			24.80			0.976
E		8.80			0.346	
e		2.54			0.100	
e3		22.86			0.900	
F			7.10			0.280
I			5.10			0.201
L		3.30			0.130	
Z			1.27			0.050

OUTLINE AND MECHANICAL DATA



Powerdip 20



ANEXO G

DATASHEET LM1117

Advanced Monolithic Systems

AMS1117

1A LOW DROPOUT VOLTAGE REGULATOR

RoHs Compliant

FEATURES

- **Three Terminal Adjustable or Fixed Voltages* 1.5V, 1.8V, 2.5V, 2.85V, 3.3V and 5.0V**
- **Output Current of 1A**
- **Operates Down to 1V Dropout**
- **Line Regulation: 0.2% Max.**
- **Load Regulation: 0.4% Max.**
- **SOT-223, TO-252 and SO-8 package available**

APPLICATIONS

- **High Efficiency Linear Regulators**
- **Post Regulators for Switching Supplies**
- **5V to 3.3V Linear Regulator**
- **Battery Chargers**
- **Active SCSI Terminators**
- **Power Management for Notebook**
- **Battery Powered Instrumentation**

GENERAL DESCRIPTION

The AMS1117 series of adjustable and fixed voltage regulators are designed to provide up to1A output current and to operate down to 1V input-to-output differential. The dropout voltage of the device is guaranteed maximum 1.3V, decreasing at lower load currents.

On-chip trimming adjusts the reference voltage to 1.5%. Current limit is set to minimize the stress under overload conditions on both the regulator and power source circuitry.

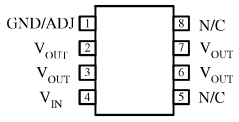
The AMS1117 devices are pin compatible with other three-terminal SCSI regulators and are offered in the low profile surface mount SOT-223 package, in the 8L SOIC package and in the TO-252 (DPAK) plastic package.

ORDERING INFORMATION:

PACKAGE TYPE			OPERATING JUNCTION TEMPERATURE RANGE
TO-252	SOT-223	8L SOIC	
AMS1117CD	AMS1117	AMS1117CS	-40 to 125° C
AMS1117CD-1.5	AMS1117-1.5	AMS1117CS-1.5	-40 to 125° C
AMS1117CD-1.8	AMS1117-1.8	AMS1117CS-1.8	-40 to 125° C
AMS1117CD-2.5	AMS1117-2.5	AMS1117CS-2.5	-40 to 125° C
AMS1117CD-2.85	AMS1117-2.85	AMS1117CS-2.85	-40 to 125° C
AMS1117CD-3.3	AMS1117-3.3	AMS1117CS-3.3	-40 to 125° C
AMS1117CD-5.0	AMS1117-5.0	AMS1117CS-5.0	-40 to 125° C

*For additional available fixed voltages contact factory.

8L SOIC Top View

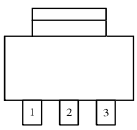


PIN CONNECTIONS

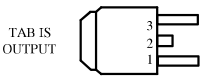
3 PIN FIXED/ADJUSTABLE VERSION

1- Ground/Adjust
2- V_{OUT}
3- V_{IN}

SOT-223 Top View



TO-252 FRONT VIEW



Advanced Monolithic Systems, Inc. www.advanced-monolithic.com Phone (925) 443-0722 Fax (925) 443-0723

ABSOLUTE MAXIMUM RATINGS (Note 1)

Power Dissipation	Internally limited
Input Voltage	15V
Operating Junction Temperature	
Control Section	-40°C to 125°C
Power Transistor	-40°C to 125°C
Storage temperature	- 65°C to +150°C

Soldering information	
Lead Temperature (25 sec)	265°C
Thermal Resistance	
SO-8 package	φ_{JA}= 160°C/W
TO-252 package	φ_{JA}= 80°C/W
SOT-223 package	φ_{JA}= 90°C/W*

* With package soldering to copper area over backside ground plane or internal power plane φ_{JA} can vary from 46°C/W to >90°C/W depending on mounting technique and the size of the copper area.

ELECTRICAL CHARACTERISTICS

Electrical Characteristics at I_{OUT} = 0 mA, and T_J = +25°C unless otherwise specified.

Parameter	Device	Conditions	Min	Typ	Max	Units
Reference Voltage (Note 2)	AMS1117	I_{OUT} = 10 mA 1.5V≤ (V_{IN} - V_{OUT}) ≤ 12V	1.232 **1.2125**	1.250 **1.250**	1.268 **1.2875**	V V
Output Voltage (Note 2)	AMS1117-1.5	V_{IN} = 3V	1.478 **1.455**	1.500 **1.500**	1.522 **1.545**	V V
	AMS1117-1.8	V_{IN} = 3.3V	1.773 **1.746**	1.800 **1.800**	1.827 **1.854**	V V
	AMS1117-2.5	V_{IN} = 4V	2.463 **2.425**	2.500 **2.500**	2.537 **2.575**	V V
	AMS1117-2.85	V_{IN} = 4.35V	2.808 **2.7645**	2.850 **2.850**	2.892 **2.9355**	V V
	AMS1117-3.3	V_{IN} = 4.8V	3.251 **3.201**	3.300 **3.300**	3.349 **3.399**	V V
	AMS1117-5.0	V_{IN} = 6.5V	4.925 **4.850**	5.000 **5.000**	5.075 **5.150**	V V
Line Regulation	AMS1117	1.5V≤ (V_{IN} - V_{OUT}) ≤ 12V		0.015 **0.035**	0.2 **0.2**	% %
	AMS1117-1.5	1.5V≤ (V_{IN} - V_{OUT}) ≤ 12V		0.3 **0.6**	5 **6**	mV mV
	AMS1117-1.8	1.5V≤ (V_{IN} - V_{OUT}) ≤ 12V		0.3 **0.6**	5 **6**	mV mV
	AMS1117-2.5	1.5V≤ (V_{IN} - V_{OUT}) ≤ 12V		0.3 **0.6**	6 **6**	mV mV
	AMS1117-2.85	1.5V≤ (V_{IN} - V_{OUT}) ≤ 12V		0.3 **0.6**	6 **6**	mV mV
	AMS1117-3.3	1.5V≤ (V_{IN} - V_{OUT}) ≤ 12V		0.5 **1.0**	10 **10**	mV mV
	AMS1117-5.0	1.5V≤ (V_{IN} - V_{OUT}) ≤ 12V		0.5 **1.0**	10 **10**	mV mV
Load Regulation (Notes 2, 3)	AMS1117	(V_{IN} - V_{OUT}) =1.5V, 10mA ≤ I_{OUT} ≤ 0.8A		0.1 **0.2**	0.3 **0.4**	% %
	AMS1117-1.5	V_{IN} = 3V, 0 ≤ I_{OUT} ≤ 0.8A		3 **6**	10 **20**	mV mV
	AMS1117-1.8	V_{IN} = 3.3V, 0 ≤ I_{OUT} ≤ 0.8A		3 **6**	10 **20**	mV mV
	AMS1117-2.5	V_{IN} = 5V, 0 ≤ I_{OUT} ≤ 0.8A		3 **6**	12 **20**	mV mV

ELECTRICAL CHARACTERISTICS

Electrical Characteristics at I_{OUT} = 0 mA, and T_J = +25°C unless otherwise specified.

Parameter	Device	Conditions	Min	Typ	Max	Units
Load Regulation (Notes 2, 3)	AMS1117-2.85	V_{IN} = 4.35V, 0 ≤ I_{OUT} ≤ 0.8A		3 **6**	12 **20**	mV mV
	AMS1117-3.3	V_{IN} = 4.75V, 0 ≤ I_{OUT} ≤ 0.8A		3 **7**	15 **25**	mV mV
	AMS1117-5.0	V_{IN} = 6.5V, 0 ≤ I_{OUT} ≤ 0.8A		5 **10**	20 **35**	mV mV
Dropout Voltage (V_{IN} - V_{OUT})	AMS1117-1.5/-1.8/-2.5/-2.85/-3.3/-5.0	ΔV_{OUT}, ΔV_{REF} = 1%, I_{OUT} = 0.8A (Note 4)		**1.1**	**1.3**	V
Current Limit	AMS1117-1.5/-1.8/-2.5/-2.85/-3.3/-5.0	(V_{IN} - V_{OUT}) = 1.5V	**900**	**1,100**	**1,500**	mA
Minimum Load Current	AMS1117	(V_{IN} - V_{OUT}) = 1.5V (Note 5)		**5**	**10**	mA
Quiescent Current	AMS1117-1.5/-1.8/-2.5/-2.85/-3.3/-5.0	(V_{IN} - V_{OUT}) = 1.5V		**5**	**11**	mA
Ripple Rejection	AMS1117	f =120Hz , C_{OUT} = 22µF Tantalum, I_{OUT} = 1A, (V_{IN}-V_{OUT}) = 3V, C_{ADJ} =10µF	**60**	**75**		dB
	AMS1117-1.5/-1.8/-2.5/-2.85	f =120Hz , C_{OUT} = 22µF Tantalum, I_{OUT} = 1A, V_{IN} = 4.35V	**60**	**72**		dB
	AMS1117-3.3	f =120Hz , C_{OUT} = 22µF Tantalum, I_{OUT} = 1A V_{IN} = 4.75V	**60**	**72**		dB
	AMS1117-5.0	f =120Hz , C_{OUT} = 22µF Tantalum, I_{OUT} = 1A V_{IN} = 6.5V	**60**	**68**		dB
Thermal Regulation	AMS1117	T_A = 25°C, 30ms pulse		0.008	0.04	%W
Adjust Pin Current	AMS1117	I_{OUT} =10mA , 1.5V≤ (V_{IN} - V_{OUT}) ≤ 12V		55	 **120**	µA µA
Adjust Pin Current Change	AMS1117	I_{OUT} =10mA, 1.5V≤ (V_{IN} - V_{OUT}) ≤ 12V		**0.2**	**5**	µA
Temperature Stability				**0.5**		%
Long Term Stability		T_A =125°C, 1000Hrs		0.3	1	%
RMS Output Noise (% of V_{OUT})		T_A = 25°C , 10Hz ≤ f ≤ 10kHz		0.003		%
Thermal Resistance Junction-to-Case		All packages			15	°C/W

Parameters identified with **boldface type** apply over the full operating temperature range.

Note 1: Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. For guaranteed specifications and test conditions, see the Electrical Characteristics. The guaranteed specifications apply only for the test conditions listed.

Note 2: Line and Load regulation are guaranteed up to the maximum power dissipation of 1.2 W for SOT-223, 2.2W for TO-252 and 780mW for 8-Lead SOIC. Power dissipation is determined by the input/output differential and the output current. Guaranteed maximum power dissipation will not be available over the full input/output range.

Note 3: See thermal regulation specifications for changes in output voltage due to heating effects. Line and load regulation are measured at a constant junction temperature by low duty cycle pulse testing. Load regulation is measured at the output lead ~1/8" from the package.

Note 4: Dropout voltage is specified up to 0.8A load. For currents over 0.8A dropout will be higher

Note 5: Minimum load current is defined as the minimum output current required to maintain regulation. When 1.5V ≤ (V_{IN} - V_{OUT}) ≤ 12V the device is guaranteed to regulate if the output current is greater than 10mA.

APPLICATION HINTS

The AMS1117 series of adjustable and fixed regulators are easy to use and are protected against short circuit and thermal overloads. Thermal protection circuitry will shut-down the regulator should the junction temperature exceed 165°C at the sense point.
Pin compatible with older three terminal adjustable regulators, these devices offer the advantage of a lower dropout voltage, more precise reference tolerance and improved reference stability with temperature.

Stability

The circuit design used in the AMS1117 series requires the use of an output capacitor as part of the device frequency compensation. The addition of 22µF solid tantalum on the output will ensure stability for all operating conditions.
When the adjustment terminal is bypassed with a capacitor to improve the ripple rejection, the requirement for an output capacitor increases. The value of 22µF tantalum covers all cases of bypassing the adjustment terminal. Without bypassing the adjustment terminal smaller capacitors can be used with equally good results.
To further improve stability and transient response of these devices larger values of output capacitor can be used.

Protection Diodes

Unlike older regulators, the AMS1117 family does not need any protection diodes between the adjustment pin and the output and from the output to the input to prevent over-stressing the die. Internal resistors are limiting the internal current paths on the AMS1117 adjustment pin, therefore even with capacitors on the adjustment pin no protection diode is needed to ensure device safety under short-circuit conditions.
Diodes between the input and output are not usually needed. Microsecond surge currents of 50A to 100A can be handled by the internal diode between the input and output pins of the device. In normal operations it is difficult to get those values of surge currents even with the use of large output capacitances. If high value output capacitors are used, such as 1000µF to 5000µF and the input pin is instantaneously shorted to ground, damage can occur. A diode from output to input is recommended, when a crowbar circuit at the input of the AMS1117 is used (Figure 1).

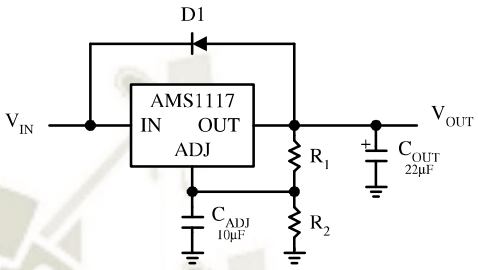
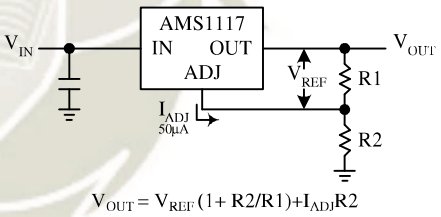


Figure 1.

Output Voltage

The AMS1117 series develops a 1.25V reference voltage between the output and the adjust terminal. Placing a resistor between these two terminals causes a constant current to flow through R1 and down through R2 to set the overall output voltage. This current is normally the specified minimum load current of 10mA. Because I_{ADJ} is very small and constant it represents a small error and it can usually be ignored.



$$V_{OUT} = V_{REF}(1 + R2/R1) + I_{ADJ}R2$$

Figure 2. Basic Adjustable Regulator

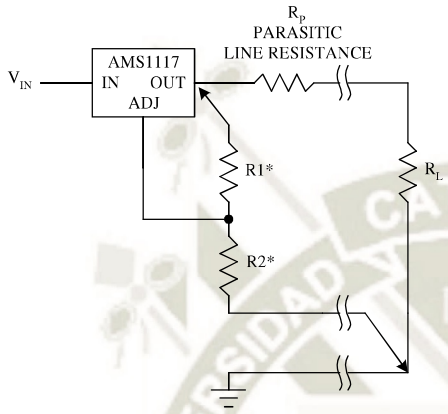
Load Regulation

True remote load sensing it is not possible to provide, because the AMS1117 is a three terminal device. The resistance of the wire connecting the regulator to the load will limit the load regulation. The data sheet specification for load regulation is measured at the bottom of the package. Negative side sensing is a true Kelvin connection, with the bottom of the output divider returned to the negative side of the load.
The best load regulation is obtained when the top of the resistor divider R1 is connected directly to the case not to the load. If R1 were connected to the load, the effective resistance between the regulator and the load would be:

$$R_P \times \left(\frac{R2+R1}{R1}\right), \quad R_P = \text{Parasitic Line Resistance}$$

APPLICATION HINTS

Connected as shown , R_P is not multiplied by the divider ratio



*CONNECT R1 TO CASE
CONNECT R2 TO LOAD

Figure 3. Connections for Best Load Regulation

In the case of fixed voltage devices the top of R1 is connected Kelvin internally, and the ground pin can be used for negative side sensing.

Thermal Considerations

The AMS1117 series have internal power and thermal limiting circuitry designed to protect the device under overload conditions. However maximum junction temperature ratings of 125°C should not be exceeded under continuous normal load conditions.
Careful consideration must be given to all sources of thermal resistance from junction to ambient. For the surface mount package SOT-223 additional heat sources mounted near the device must be considered. The heat dissipation capability of the PC board and its copper traces is used as a heat sink for the device. The thermal resistance from the junction to the tab for the AMS1117 is 15°C/W. Thermal resistance from tab to ambient can be as low as 30°C/W.

The total thermal resistance from junction to ambient can be as low as 45°C/W. This requires a reasonable sized PC board with at least on layer of copper to spread the heat across the board and couple it into the surrounding air.
Experiments have shown that the heat spreading copper layer does not need to be electrically connected to the tab of the device. The PC material can be very effective at transmitting heat between the pad area, attached to the pad of the device, and a ground plane layer either inside or on the opposite side of the board. Although the actual thermal resistance of the PC material is high, the Length/Area ratio of the thermal resistance between layers is small. The data in Table 1, was taken using 1/16" FR-4 board with 1 oz. copper foil, and it can be used as a rough guideline for estimating thermal resistance.
For each application the thermal resistance will be affected by thermal interactions with other components on the board. To determine the actual value some experimentation will be necessary.
The power dissipation of the AMS1117 is equal to:

$P_D = (V_{IN} - V_{OUT})(I_{OUT})$

Maximum junction temperature will be equal to:

$T_J = T_{A(MAX)} + P_D$(Thermal Resistance (junction-to-ambient))

Maximum junction temperature must not exceed 125°C.

Ripple Rejection

The ripple rejection values are measured with the adjustment pin bypassed. The impedance of the adjust pin capacitor at the ripple frequency should be less than the value of R1 (normally 100Ω to 200Ω) for a proper bypassing and ripple rejection approaching the values shown. The size of the required adjust pin capacitor is a function of the input ripple frequency. If R1=100Ω at 120Hz the adjust pin capacitor should be >13µF. At 10kHz only 0.16µF is needed.

The ripple rejection will be a function of output voltage, in circuits without an adjust pin bypass capacitor. The output ripple will increase directly as a ratio of the output voltage to the reference voltage (V_{OUT} / V_{REF}).

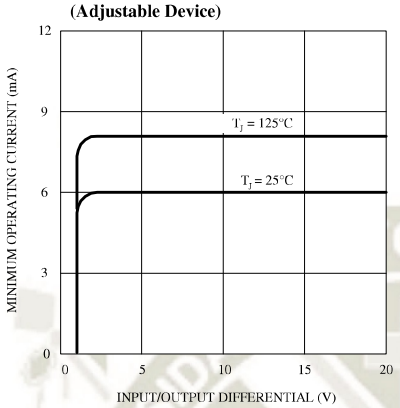
Table 1.

COPPER AREA		BOARD AREA	THERMAL RESISTANCE (JUNCTION-TO-AMBIENT)
TOP SIDE*	BACK SIDE		
2500 Sq. mm	2500 Sq. mm	2500 Sq. mm	55°C/W
1000 Sq. mm	2500 Sq. mm	2500 Sq. mm	55°C/W
225 Sq. mm	2500 Sq. mm	2500 Sq. mm	65°C/W
100 Sq. mm	2500 Sq. mm	2500 Sq. mm	80°C/W
1000 Sq. mm	1000 Sq. mm	1000 Sq. mm	60°C/W
1000 Sq. mm	0	1000 Sq. mm	65°C/W

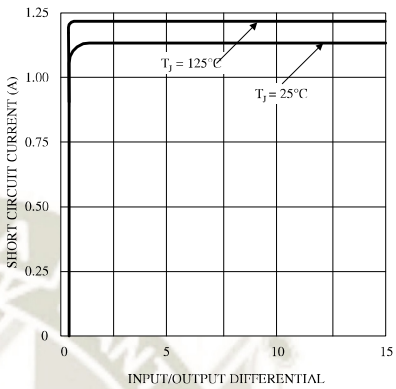
* Tab of device attached to topside copper.

TYPICAL PERFORMANCE CHARACTERISTICS

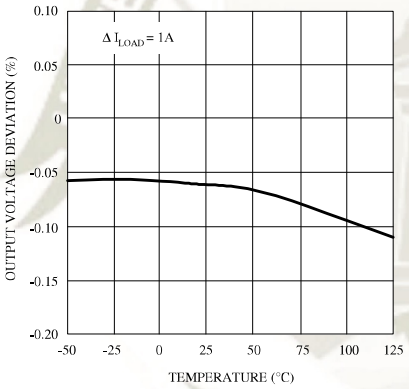
Minimum Operating Current (Adjustable Device)



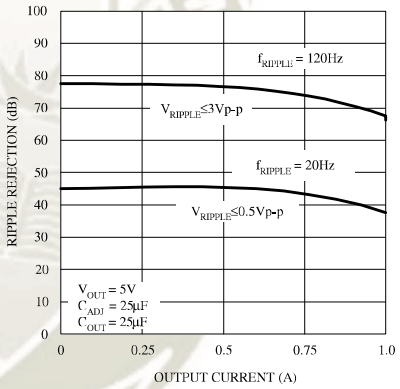
Short-Circuit Current



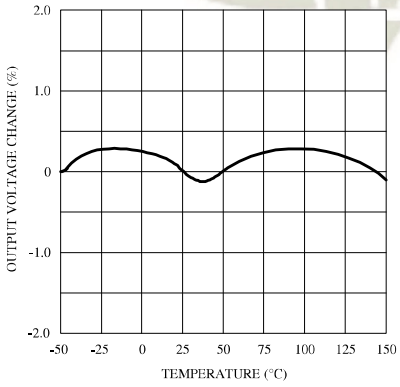
Load Regulation



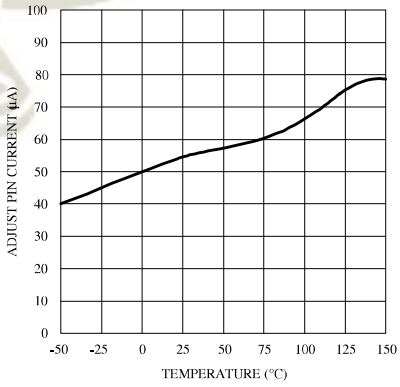
Ripple Rejection vs. Current



Temperature Stability

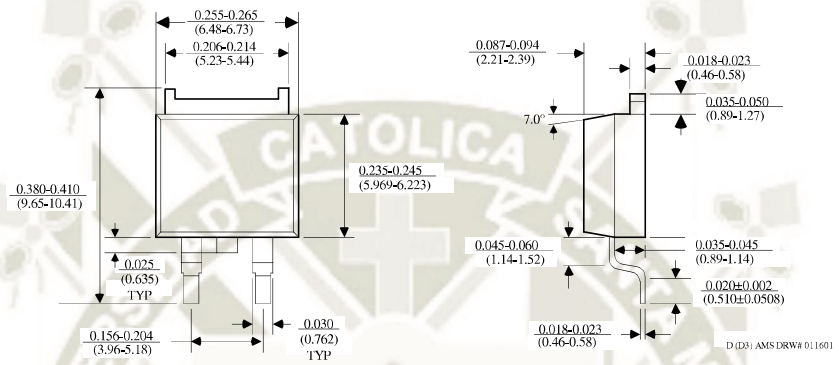


Adjust Pin Current

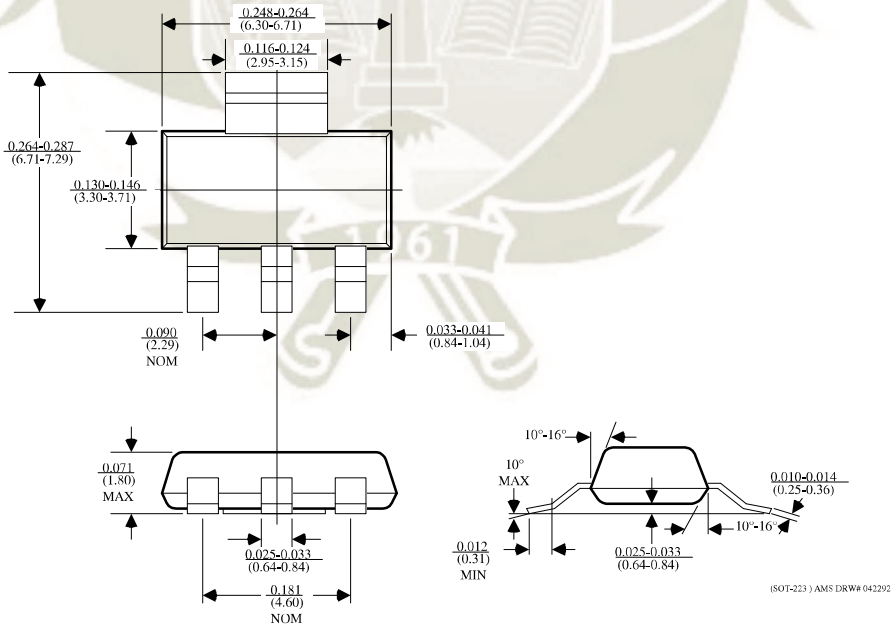


PACKAGE DIMENSIONS inches (millimeters) unless otherwise noted.

TO-252 PLASTIC PACKAGE (D)

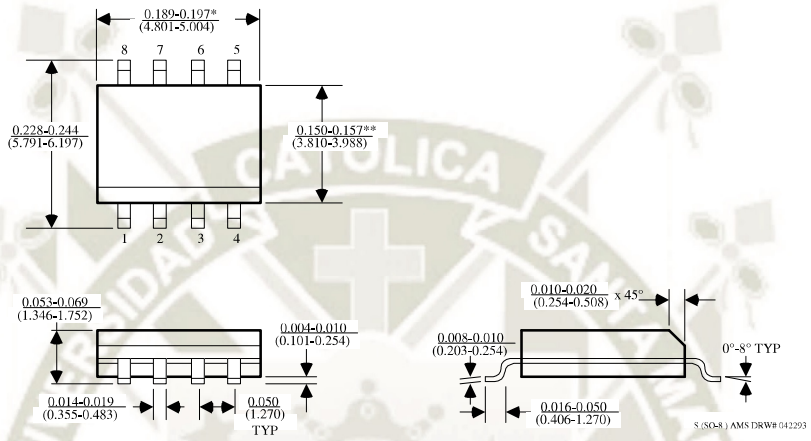


3 LEAD SOT-223 PLASTIC PACKAGE



PACKAGE DIMENSIONS inches (millimeters) unless otherwise noted (Continued).

8 LEAD SOIC PLASTIC PACKAGE (S)

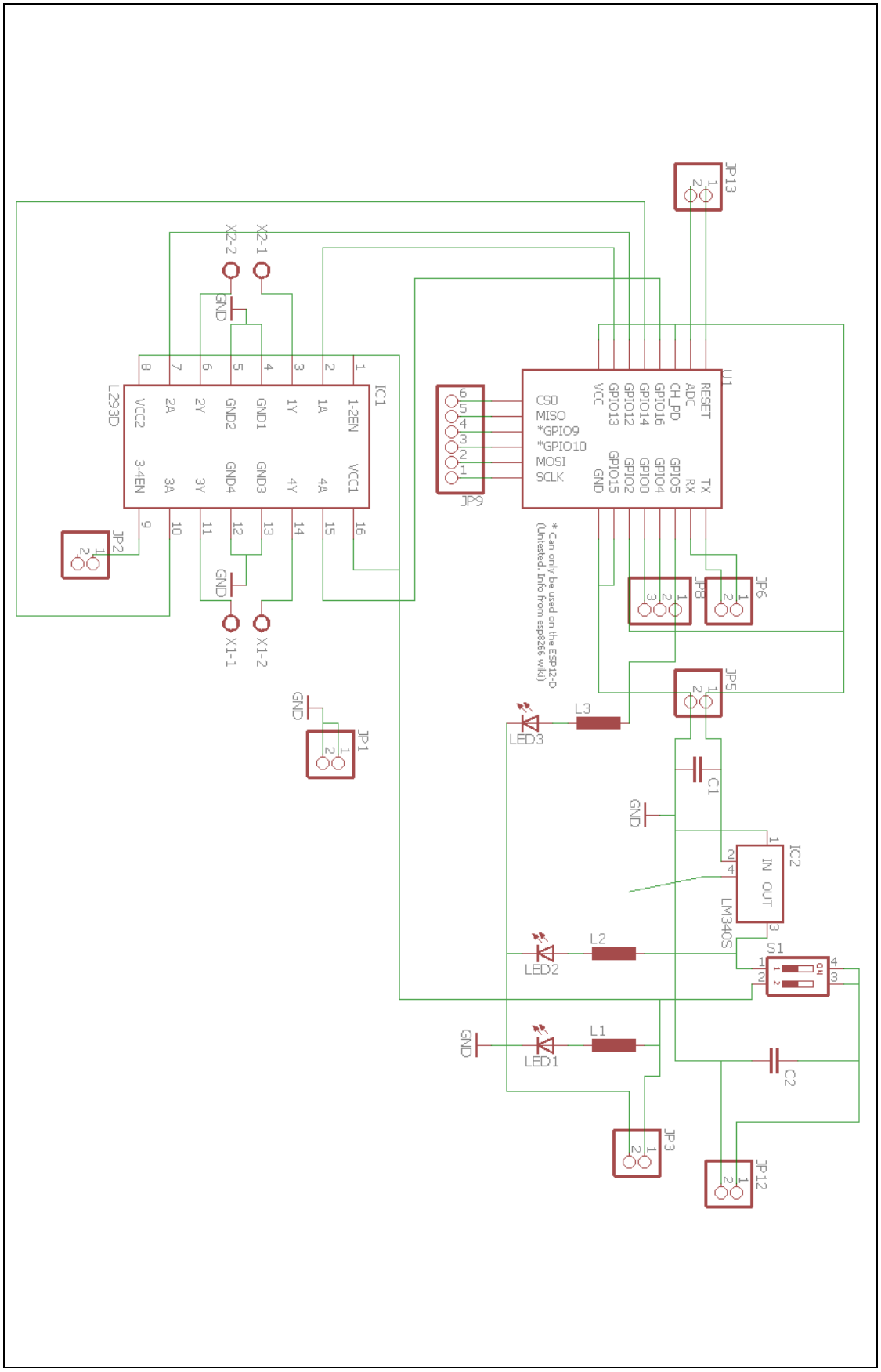


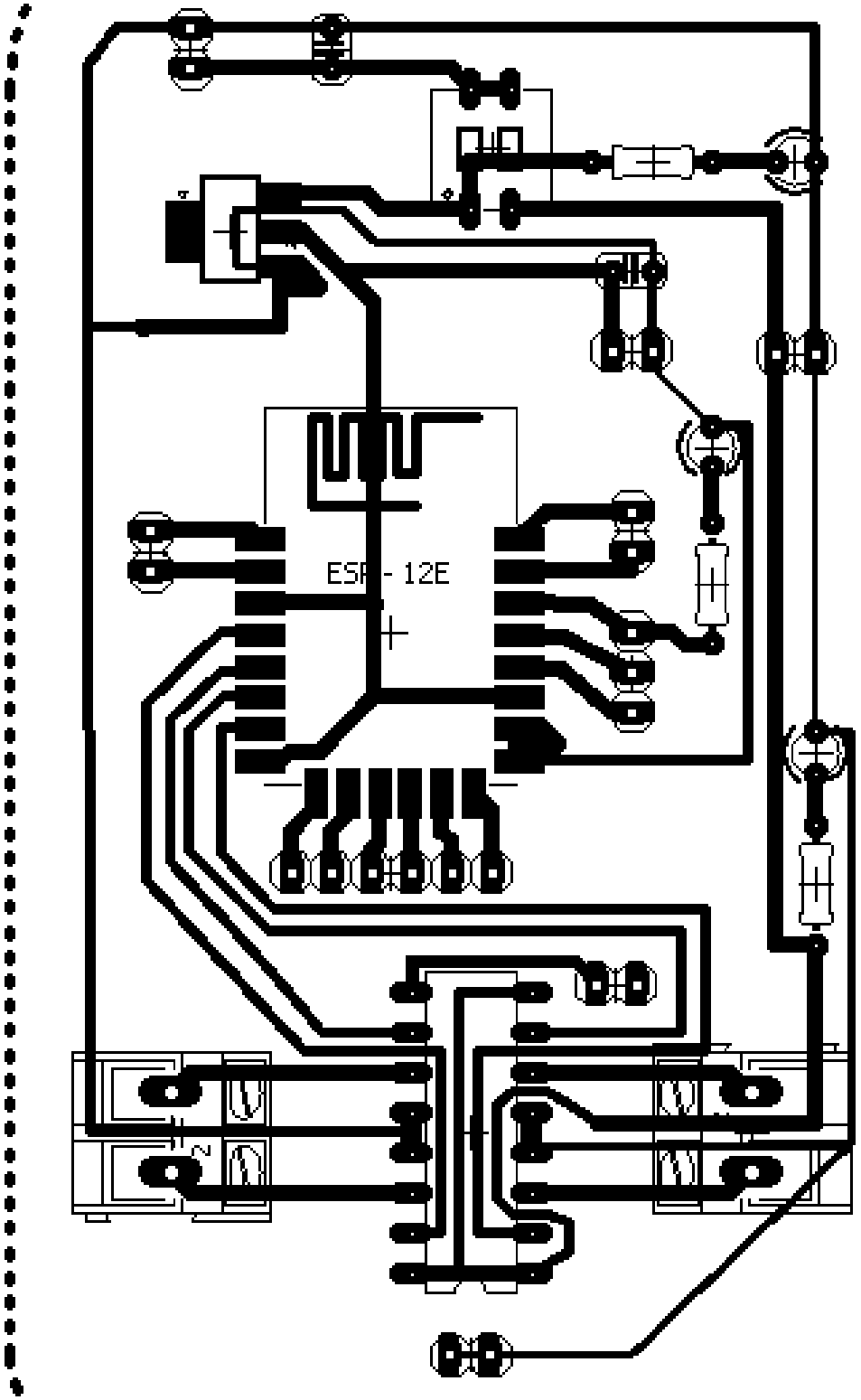
*DIMENSION DOES NOT INCLUDE MOLD FLASH. MOLD FLASH SHALL NOT EXCEED 0.006" (0.152mm) PER SIDE

**DIMENSION DOES NOT INCLUDE INTERLEAD FLASH. INTERLEAD FLASH SHALL NOT EXCEED 0.010" (0.254mm) PER SIDE

ANEXO H

DISEÑO DE PLACA



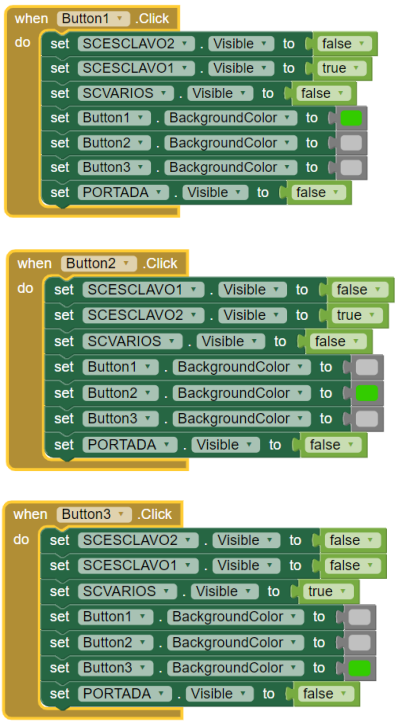


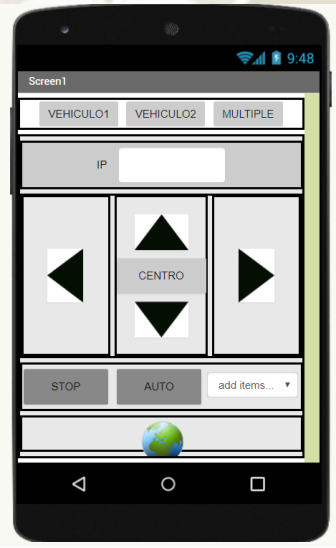
ANEXO I

DISEÑO Y PROGRAMACIÓN DE APP

INTERFAZ

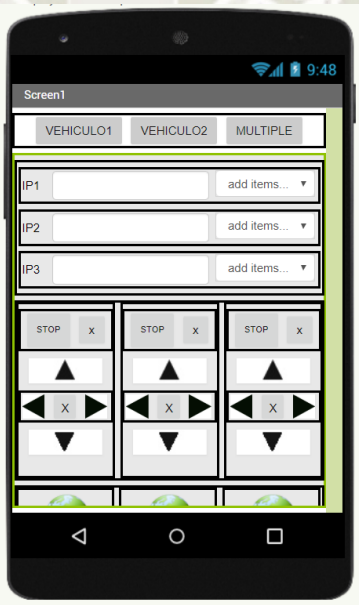
Creación de aplicación para dispositivos móviles inteligentes por medio del programa AppInventor

<table>
<tr><th>Componenete</th><th>Diseño</th><th>Programación</th></tr>
<tr><td colspan="3">Botones de selección de ventanas</td></tr>
<tr><td>Screen1
HorizontalArrangement1
Button1
Button2
Button3</td><td>VEHICULO1 VEHICULO2 MULTIPLE</td><td> 
</td></tr>
<tr><td colspan="3">Portada</td></tr>
<tr><td>PORTADA
Image1</td><td> 
</td><td></td></tr>
</table>

Primera ventana de control para un vehículo		
SCESCLAVO1 HorizontalArrangement2 Label1 TextBox1 HorizontalArrangement3 VerticalArrangement7 ButtonM5 VerticalArrangement8 ButtonM2 ButtonM6 ButtonM3 VerticalArrangement9 ButtonM4 HorizontalArrangement27 ButtonM1 ButtonM7 Spinner1 WebViewer1		
Segunda ventana de control para un vehículo		
SCESCLAVO2 HorizontalArrangement4 Label2 TextBox3 HorizontalArrangement5 VerticalArrangement11 ButtonS5 VerticalArrangement12 ButtonS2 ButtonS6 ButtonS3 VerticalArrangement13 ButtonS4 HorizontalArrangement23 ButtonS1 ButtonS7 Spinner2 WebViewer2		

Ventana de control para múltiples vehículos

- SCVARIOS
  - VerticalArrangement14
    - HorizontalArrangement28
      - Label3
      - TextBox5
      - Spinner3
    - HorizontalArrangement29
      - Label4
      - TextBox7
      - Spinner4
    - HorizontalArrangement30
      - Label5
      - TextBox8
      - Spinner5
    - HorizontalArrangement18
      - VerticalArrangement15
        - HorizontalArrangement24
          - vehi10
          - vehi16
          - vehi11
        - HorizontalArrangement19
          - vehi13
          - vehi15
          - vehi12
          - vehi14
      - VerticalArrangement16
        - HorizontalArrangement25
          - vehi20
          - vehi26
          - vehi21
        - HorizontalArrangement20
          - vehi23
          - vehi25
          - vehi22
          - vehi24
      - VerticalArrangement17
        - HorizontalArrangement26
          - vehi30
          - vehi36
          - vehi31
        - HorizontalArrangement21
          - vehi33



Control 1

Control 2

vehi35
vehi32
vehi34
HorizontalArrangement22
WebViewer3
WebViewer4
WebViewer5

Control 3