UNIVERSIDAD SAN SEBASTIAN

SEDE SANTIAGO, BELLAVISTA

FACULTAD DE INGENIERÍA Y TECNOLOGÍA

ESCUELA DE INGENIERÍA CIVIL EN INFORMÁTICA

Diseño, programación y construcción de un prototipo de Sistema de Vehículo de Guiado Autónomo

Memoria para optar al Título de Ingeniero Civil Informático.

Estudiantes: **Erickson Francisco Gallardo Cerda**

José Antonio López Hernández

Profesor Guía: **Carlos Escobar Zepeda**

Santiago, Chile

2018.

CALIFICACIÓN DE LA MEMORIA

En Santiago, el…… de…….……..de…….., los abajo firmantes dejan constancia que el alumno……..……...…… de la carrera……..ha aprobado la memoria para optar al Título……..…….con una nota de……..…….

Profesor
…….
Profesor
…….
Profesor
…….

Dedicatoria

Quiero dedicarle este trabajo a mi mamá Albita, que siempre ha estado conmigo en los momentos más complicados de mi vida, dándome ánimo para seguir adelante, y enseñándome que existe un camino de bien, responsabilidad y dedicación. Ha sido mi apoyo fundamental durante toda mi vida, y con su increíble bondad, alegría, amor y optimismo me dio las fuerzas para nunca rendirme y poder llegar a cumplir mi sueño contra toda adversidad.

A mis hermanos Pablo y Cristóbal, los amo con todo mi corazón y siempre estaré ahí cuando lo necesiten. Espero que puedan cumplir todos sus sueños y convertirse en personas de bien, que logren el éxito y la felicidad plena en sus vidas.

A mi polola Cony, que me ha sabido escuchar y soportar en los momentos más complicados, y me ha amado sinceramente cada día de nuestra relación.

Al profesor Carlos Escobar, que desde que nos conocimos en el taller de Arduino confió en mis capacidades y siempre supo con su buena onda y simpatía enseñarme a ser mejor persona y mejor profesional.

A mi compañero y amigo Erickson, con el que juntos hemos pasado por este gran desafío.

A todos los que siempre han estado a mi lado alentándome a seguir adelante.

José Antonio López.

Quiero dejar expresado en las siguientes palabras, la gran emoción que me llena el poder cumplir con esta gran etapa en mi vida, una etapa que mis padres no pudieron alcanzar por priorizar su bienestar familiar en los duros momento de su juventud. Es por ello, que les dedico este gran logro a ellos.

En primer lugar, a mi padre Francisco Gallardo Arzola, quien con gran esfuerzo y disciplina se ha sacrificado día y noche desde muy joven para sacar adelante nuestra familia, teniendo incluso que dejar de lado ver el crecimiento de sus hijos por mucho tiempo para poder otorgar el sustento necesario para que nosotros pudiéramos tener una mejor vida de la que él pudo vivir.

Y, en segundo lugar, pero no menos importante, a mi madre Jeannette Cerda Herrera, quien con su gran compasión y paciencia estuvo a nuestro lado en cada momento de nuestra vida, siendo nuestro principal apoyo emocional, aconsejándonos y animándonos en todo momento y siempre que lo necesitábamos.

Quiero darles las gracias, y decirles que estoy muy feliz, agradecido y orgulloso de tenerlos como padres, por todo el sacrificio que han hecho para que yo y mi hermano pudiéramos vivir una vida tranquila y para que yo pudiera llegar a este punto de la vida, nos ha tocado vivir momentos difíciles, pero gracias a su determinación y su tenacidad, nunca se rindieron y supieron sacarnos adelante, es por eso y mucho más, que este triunfo no es solo mío, también es de usted.

Ahora podré decirles con toda tranquilidad que es momento que se tomen un descanso, que ahora es mi turno de apoyar en los momentos duros de la vida.

Erickson Francisco Gallardo Cerda.

Agradecimientos

Queremos agradecer el enorme apoyo del profesor, amigo y mentor Carlos Escobar, que desde el comienzo de nuestra carrera fue un apoyo y guía para nuestra formación profesional, enseñándonos no solo en el ámbito académico, si no **tambien**, como desenvolvemos en el mundo laboral, formándonos con habilidades multidisciplinarias, dándonos la seguridad de que podremos enfrentar cualquier desafío laboral y personal que se nos interponga. Gracias por dejarlo todo en la creación del laboratorio Make It, donde fue posible desarrollar este trabajo de título y muchos otros proyectos que sirvieron para lograr esta meta de convertirnos en profesionales.

Queremos agradecer **tambien** a Felipe Flores, que siempre tuvo toda la disponibilidad y buena voluntad para ayudarnos en todo lo que necesitamos durante la creación de este proyecto.

Tabla de contenido

Índice de Tablas

Índice de Figuras

Resumen

El presente informe de tesis tiene por objetivo diseñar, desarrollar e implementar el prototipo de un sistema de vehículos de guiado autónomo utilizado para el transporte de materiales y que responda a pedidos realizados por los usuarios. Este es de bajo costo para apoyar y optimizar los procesos de logística interna de una organización de pequeña envergadura.

El sistema contempla la creación de vehículos autónomos que mediante diversos sensores y placas de prototipado electrónico podrá desplazarse de un lugar a otro siendo controlados por un servidor central que le enviará las instrucciones necesarias para que cumpla con su objetivo. Estos vehículos tienen la capacidad de transportar peso, adaptando la potencia que le entrega a sus motores para moverse a una velocidad constante.

Se desarrolló también un servidor que será el encargado de controlar los pedidos de materiales realizados por los usuarios, se calculan las instrucciones que serán enviadas a los vehículos para seguir una ruta hasta su destino, se evitan posibles colisiones y se encarga de registrar información importante para el buen funcionamiento del sistema.

El sistema es controlado a través de una interfaz web donde el usuario puede realizar los pedidos de materiales a una posición determinada, esta interfaz es sencilla de utilizar y muestra información relevante del estado del pedido como la posición actual del vehículo. También se cuenta con una interfaz encargada de administrar el sistema, donde se pueden registrar vehículos, los materiales que contienen y se configuran los caminos que deben seguir.

Este documento contempla **tambien** como estos sistemas de vehículos de guiado autónomo resuelven problemas en las organizaciones y su impacto en su desarrollo económico, la investigación de alternativas disponibles de diversas

organizaciones en el mercado, la investigación y comparación de diversas alternativas de desarrollo para el sistema y su prototipado, el diseño del sistema con el apoyo de técnicas de diagramación como por ejemplo, diagramas UML de casos de uso, de entidad relación y diagramas de flujo, también se presenta un plan de pruebas necesarias para evaluar y validar las funcionalidades del sistema.

El desarrollo del sistema informático del proyecto descrito en el siguiente documento se realizó con base en la metodología de desarrollo de Proceso Unificado Racional (RUP, por sus siglas en inglés de Rational Unified Process), que permitió crear el prototipo del sistema desde funcionalidades básicas de operación e ir agregando nuevas funcionalidades luego de aprobar el estado del incremento anterior.

Este proyecto finaliza en un prototipo funcional que es implementado en el laboratorio Make It de la Universidad San Sebastián. Se dan sugerencias para continuar el proyecto y agregar funcionalidades que aumenten las capacidades del sistema.

Abstract

The objective of this thesis report is to design, develop and implement the prototype of a self-guided vehicle system used to transport materials and respond to orders placed by users. This is low cost to support and optimize the internal logistics processes of a small organization.

The system contemplates the creation of autonomous vehicles that, through various sensors and electronic prototyping boards, will be able to move from one place to another, being controlled by a central server that will send the necessary instructions so that it fulfills its objective. These vehicles have the ability to carry weight, adapting the power that is delivered to their engines to move at a constant speed.

A server was also developed that will be in charge of controlling the orders of materials made by users, calculating the instructions that will be sent to vehicles to follow a route to their destination, avoiding possible collisions and responsible for recording important information for the proper functioning of the system.

The system is controlled through a web interface where the user can place the material orders to a certain position, this interface is simple to use and shows relevant information of the status of the order as the current position of the vehicle. There is also an interface in charge of administering the system, where vehicles can be registered, the materials they contain and the paths they must follow are configured.

This document also contemplates how these self-guided vehicle systems solve problems in organizations and their impact on their economic development, research alternatives available from various organizations in the market, research and compare various development alternatives for the system and its prototyping, the design of the system with the support of diagramming techniques such as, for

example, UML diagrams of use cases, entity relationship and flow diagrams, a test plan necessary to evaluate and validate the functionalities of the system is also presented.

The development of the computer system of the project described in the following document was made based on the Rational Unified Process (RUP) development methodology, which allowed the creation of the prototype of the system from basic functionalities of operation and add new features after approving the status of the previous increase.

This project ends in a functional prototype that is implemented in the Make It laboratory of San Sebastian University. Suggestions are given to continue the project and add features that increase the capabilities of the system.

Introducción

Cuando se habla del concepto de almacenamiento, lo primero que se viene a la mente es un almacén, donde se pueda aprovisionar diversas materias primas y productos, según lo requiera la industria. Pero la realidad de este concepto ha ido evolucionando a lo largo del tiempo, y en la actualidad, corresponde a una unidad de servicio y soporte a la estructura orgánica y funcional de una empresa, sea esta comercial o industrial (Salazar Lopez, s.f.). Este concepto, que antes se caracterizaba como un simple espacio dentro de la organización destinado simplemente al custodio de mercancías, hoy es una estructura clave que provee elementos físicos y funcionales capaces de entregar valor agregado a los procesos influyentes del negocio.

Esta importancia que se le atribuye a la gestión de almacenes requiere de un conjunto de procesos logísticos que velen por la recepción, el almacenamiento y el traslado de insumos, al interior del almacén, hasta sus puntos de tratado o despacho, junto al tratamiento de los datos generados en cada una de las etapas que lo constituyan.

Estos procesos logísticos se ven enfrentados a la problemática de que las cadenas de suministros son cada vez más grandes y complejas, haciendo necesario crear nuevos mecanismos para gestionar los nuevos desafíos de las industrias modernas, lo que dificulta la gestión y eleva los costos de gran manera (Ballou, 2004).

Es por ello por lo que la gestión de almacenes tiene como principal propósito optimizar su área de logística funcional para así garantizar el suministro continuo y oportuno de los insumos necesarios para una entrega de servicios y/o productos de forma ininterrumpida. En esta área se pueden identificar dos etapas que juntas constituyen a las actividades de mayor importancia en las cadenas de suministros, estas etapas son: Abastecimiento y distribución física (Salazar Lopez, s.f.).

Con el avance de las tecnologías se han desarrollado un gran conjunto de herramientas que apoyan a la gestión de almacenes. Muchas de ellas se encuentran enfocadas principalmente a la etapa de abastecimiento, como los sistemas de gestión de almacenes, planificación de recursos empresariales y otros, y por otra parte los sistemas enfocados a la distribución física como son los sistemas de almacenamiento automático o los sistemas de vehículos de guiado automático (AGV)

CAPÍTULO 1: ANTECEDENTES DEL PROBLEMA

1.1. Formulación del problema

Hoy en día, en un mercado globalizado y altamente competitivo, donde las personas tienen cada vez más opciones de obtener un producto desde cualquier lugar del mundo, se hacen notorios cada vez más los efectos del libre mercado y la autorregulación de los precios. Esto conlleva a que las empresas no puedan fijar precios arbitrarios, sino que, deben establecer precios cercanos a los de la competencia por un mismo producto. Como consecuencia de este fenómeno, es de vital importancia para las organizaciones buscar opciones que reduzcan sus costos para poder aumentar sus utilidades. (Salazar Carvajal, 2005)

El gobierno de Alemania, en el año 2011 presentó el concepto de “Industria 4.0”, el que representa una cuarta revolución industrial, donde las máquinas están conectadas entre ellas y también a sistemas informáticos, que les permiten tomar decisiones acertadas. Esta iniciativa del gobierno alemán pretende potenciar las industrias manufactureras mejorando sus cadenas de valor y modelos de negocios, gracias a la digitalización de los procesos y la interconexión de estos. Se espera que para el año 2020, esta estrategia tenga un impacto positivo en la economía del país europeo (European Commision, 2017), por esta razón otros países del mundo con gran presencia en la industria manufacturera se están adaptando a este cambio tecnológico (Kurfuss, 2014).

A pesar de los beneficios económicos y la reducción de costos esperados en la nueva industria 4.0, solo el 10% de las compañías están integradas completamente a estas nuevas tecnologías, siendo las grandes empresas las que lideran en este cambio y dejando atrás a las medianas y pequeñas, observando una clara relación directa entre el tamaño de la compañía y la adopción de la cuarta revolución industrial (Schröder, 2017).

Al ser una estrategia muy reciente y no haber un historial grande de casos de éxito, aún las medianas y pequeñas empresas presentan incertidumbre sobre los beneficios que puede traer adoptar estas tecnologías, lo que conlleva a no asumir el riesgo de invertir grandes cantidades de dinero en esta transformación (Sundblad, 2018).

1.2. Justificación e importancia del trabajo

La optimización de la línea de suministros representa una ventaja competitiva en las etapas de logística y operaciones internas de la cadena de valor de Michael Porter. Agregando valor a los productos y servicios que la organización entrega, esto se debe a que un producto o un servicio no tiene valor si no se encuentra disponible para el cliente en el momento y el lugar que él desee. Y cabe mencionar, que los clientes desean una respuesta cada vez más rápida, estando dispuestos a pagar un precio mayor por tener el producto o servicio lo antes posible (Ballou, 2004).

La integración de los sistemas de vehículos de guiado autónomo (AGV) implican un apoyo y una mejora para la optimización de la línea de suministros de una organización, ya que con su implementación se puede:

- Reducir riesgos al operar maquinarias.
- Optimizar los tiempos de traslado de activos.
- Integrar sistemas de información que permitan obtener datos precisos para una mejor toma de decisiones.

La incertidumbre generada a las medianas y pequeñas empresas a implementar estos sistemas radica que las tecnologías disponibles desde el comienzo de la cuarta revolución eran aún costosas para la mayoría de las empresas, suponiendo un alto riesgo para aquellas que no cuentan con un flujo de capital que pudiera soportar dicho cambio. Pero en el transcurso del tiempo, se han desarrollado tecnologías más baratas que permitirán implementar un sistema de vehículos de guiado autónomo (AGV) que no supusiera un alto riesgo a mediana y pequeñas empresas.

Por ello, crear una solución de bajo costo que permita a las organizaciones de tamaño mediano y sin un poder adquisitivo elevado, implementar estos sistemas de una forma flexible y fácil le otorgaría grandes beneficios para la logística de sus almacenes y para la organización.

1.3 Elementos a considerar

Para el desarrollo de este proyecto, será necesario investigar las tecnologías disponibles en el mercado para el diseño del sistema de vehículo de guiado autónomo, ya que hay una amplia gama de sensores y tecnologías que permiten el desarrollo de estos sistemas. Por ende, se realizará una investigación de las posibles soluciones del vehículo y se evaluará según su factibilidad y su eficiencia al momento de cumplir con el objetivo de este proyecto.

1.4 Resultados esperados

Al finalizar este trabajo, se espera principalmente obtener un prototipo funcional que cumpla con los requerimientos esenciales de un sistema de vehículo de guiado autónomo (AGV), estos requerimientos corresponden al traslado de un activo o insumo a un lugar determinado por el usuario tantas veces sea necesario. Además del prototipo funcional se espera obtener los diagramas, planos y la documentación del sistema diseñado para su reconstrucción y posible desarrollo en nuevos proyectos para su mejora y escalabilidad.

1.5. Objetivo del proyecto

El desarrollo de este proyecto estará conformado por los siguientes objetivos.

1.5.1 Objetivo General

El objetivo general de este proyecto es implementar un prototipo de un sistema de guiado autónomo de vehículos para el transporte de activos en el laboratorio Make It de la Universidad San Sebastián.

1.5.2 Objetivos Específicos

Para cumplir con el objetivo general, se deben cumplir con los siguientes objetivos específicos:

- Construir un prototipo de vehículo que permita ser controlado mediante comunicación web.
- Desarrollar un servidor capaz de gestionar pedidos y enviar instrucciones a los vehículos.
- Desarrollar una aplicación amigable e intuitiva para realizar pedidos y administrar el sistema.

1.6. Alcances y Limitaciones

Este proyecto contempla los siguientes alcances y limitación.

1.6.1. Alcances

- La construcción de un vehículo funcional que pueda ser controlado remotamente a través de una comunicación web.
- Desarrollar una aplicación que gestione los pedidos de los usuarios y calcule instrucciones para ser enviadas a los vehículos.

- Desarrollar aplicaciones web que sirvan para hacer pedidos de los vehículos y administrar el sistema.
- Solo se considerarán para este proyecto requerimientos funcionales tales como el recorrido automático del vehículo y el transporte de un peso acorde a las capacidades del hardware.

1.6.2. Límites

- El sistema se desarrollará con los materiales disponibles que se encuentran en el laboratorio Make It.
- Los caminos que podrán ser recorridos por los vehículos estarán dispuestos según el tamaño y configuración del laboratorio Make It.
- El peso máximo que soporta el prototipo estará limitado a la potencia de los motores utilizados, este peso se estimará mediante pruebas realizadas al prototipo.
- No se considerarán aspectos de seguridad en el software, ni en la operación del vehículo, tales como detección de obstáculos, informe de desvío o salida del camino.

CAPÍTULO 2: MARCO TEÓRICO

2.1 Enfoque del proyecto

Este proyecto tiene como finalidad la confección de un prototipo funcional que represente el sistema de vehículos de guiado autónomo (AGV) diseñado. Para este propósito se considerará un prototipo funcional como:

Aquel prototipo que contenga los componentes básicos de un sistema AGV y se encuentre capacitado para recibir peticiones de múltiples usuarios, calcular y comunicar la ruta que el vehículo deberá realizar, la ejecución y finalización de la ruta generada del pedido en curso.

A continuación, se describirán algunos conceptos y tecnologías que se han utilizado para el desarrollo de este sistema.

2.1.1 Sistema de vehículo de guiado autónomo (AGV)

Los sistemas de vehículo de guiado autónomo (por sus siglas en inglés AGV), consisten básicamente en una serie de vehículos que se conducen de manera autónoma sin un conductor que estuviera encargado de guiarlos, fueron pensados para la realización de trabajos de transporte y traslado de materiales, que son trabajos de alta repetitividad. Su objetivo es garantizar el transporte de materiales mediante caminos designados con anterioridad sin la necesidad de supervisión directa.

Estos sistemas están formados principalmente por vehículos autónomos que ejecutan órdenes. Estas órdenes vienen dadas por un servidor principal que puede estar conectado a los sistemas principales de gestión de almacenes (WMS) o de planificación de recursos (ERP). Estas órdenes se deben gestionar de tal manera que se cumplan los objetivos de cada pedido y se optimice al

máximo la operación de la organización, esta gestión puede involucrar la priorización, cancelación y modificación de pedidos, entre otros. Los vehículos autónomos también deben poder administrar las rutas que lo lleven a su destino, considerando los mejores tiempos de respuesta y evitando colisiones con otros vehículos, esto se puede realizar ya sea utilizando sensores sofisticados que reconozcan la presencia de obstáculos u otros vehículos o con un sistema automatizado que controle las rutas de forma centralizada.

2.1.2 RFID

RFID es una tecnología utilizada para la identificación de objetos mediante radiofrecuencia normada en el estándar ISO/IEC 18000-3. Consiste en un dispositivo activo que básicamente corresponde a un microcontrolador y una antena que operan a 13.56 MHz induciendo un campo magnético sobre la antena de una etiqueta pasiva para lograr una comunicación entre ambos dispositivos. De esta forma es posible obtener un número que cumple la función de identificador único, con el cual se puede obtener información desde una base de datos similar a un código de barras.

Esta tecnología actualmente es usada en variedad de industrias como lo son el transporte público, plantas de fabricación, almacenes, entre otros.

2.1.3 Sensor de línea

Los sensores de líneas son dispositivos simples pero útiles a la hora de automatizar la circulación de un vehículo. Se compone de un emisor y un receptor de frecuencias infrarrojas.

El funcionamiento de este dispositivo se debe a las propiedades de absorción de frecuencias de los colores, donde el color negro es producido al absorber la mayor parte de las frecuencias y el color blanco al no absorberlas haciendo que todas las frecuencias reboten en el material. De esta forma se puede determinar

si el sensor se encuentra sobre un material de color negro al no recibir la luz infrarroja del emisor reflejada por el material.

Al tener por lo menos dos de estos sensores se puede determinar la desviación con respecto a la línea y ajustar el curso de un vehículo. Mientras más sensores existan más precisamente se puede ajustar esta desviación.

2.1.4 WebSocket's

La tecnología WebSocket's publicada en el año 2011 en el rfc6455, actualmente es un protocolo estándar para la comunicación bidireccional entre el cliente y el servidor. Esta tecnología es útil en aplicaciones donde es necesario que el cliente inicie la comunicación con el servidor y suple las actuales problemáticas que surgen al utilizar técnicas como "long polling" (Fette & Melnikov, 2011).

Existen frameworks que utilizan esta tecnología y facilitan el desarrollo de las aplicaciones web modernas con características de comunicación en tiempo real, ofreciendo la posibilidad de habilitar grupos donde los clientes se pueden suscribir y esperar eventos que emita tanto el servidor como los demás clientes de ese grupo (SocketIo, s.f.).

2.2 Soluciones similares

En esta sección se mencionan algunas empresas que utiliza y han desarrollado sistemas AGV con algunas de las tecnologías mencionadas en el punto anterior. Las empresas que se mencionan a continuación fueron seleccionadas por la disponibilidad de su información en la red, tanto en documentos como en sus respectivos sitios web.

2.2.1 Amazon Robotics

Anteriormente conocido como Kiva Systems, es la empresa de robótica guiada subsidiaria de Amazon que es la encargada de desarrollar los productos de AGV que esta multimillonaria empresa utiliza en sus operaciones de gestión de almacenes. Estos robots llamados Kiva han resultado favorecer en gran medida las operaciones de negocio de Amazon, reduciendo tareas de traslado que tardaban en promedio una hora y media, y con la implementación de los robots Kiva este tiempo de traslado se ha reducido a 15 minutos en promedio, dando pie al aumento a la cantidad de demanda que puedan manejar en un día (Tam, 2014)

Los robots Kiva cuentan con una gran cantidad de sensores que les permite comunicarse entre ellos y evitar las colisiones contra operarios, estanterías u otros vehículos Kiva. Además, sus capacidades mecánicas le permiten levantar cerca de media tonelada, posibilitando trasladar estanterías completas de un lugar a otro a una velocidad de 5 km/h (Steiner, 2009).

2.2.2 SSI Schaefer

Según su página oficial en español (SSI Schaefer, s.f.), SSI Schaefer es una empresa alemana especializada en logística interna de almacenamiento, donde destacan productos para almacenamiento, transporte, preparación de productos y sistemas de manipulación.

En un principio comenzó como una industria de productos de chapas, que luego de su primer desarrollo Lager-Fix, se enfocó en las soluciones logísticas hasta convertirse en la gran industria implementadora de soluciones logísticas.

SSI Schaefer entrega una amplia gama de AGV’s que están al servicio y necesidad de sus clientes, entre ellos se reconoce al AGV llamado WEASEL, que es ideal para transportar unidades de cargas pequeñas en zonas de poco espacio y 2MOVE, que es ideal para productos pesados provenientes de líneas de transportadores modulares.

Estos sistemas se basan en la navegación por guiado láser óptico, o sensores que permitan escanear el ambiente para garantizar la fiabilidad de sus productos y tag RFID para su rastreo y posicionamiento en las plantas de operaciones.

2.2.3 CEIT

En un principio llamada Slovak, un centro de productividad, y posteriormente consolidada como CEIT, es una empresa eslovaca creada por la universidad de Žilina en 1998, es una empresa enfocada principalmente en la investigación y desarrollo de soluciones únicas en logística y robótica. Son reconocidos por ser pioneros en el campo de Digital Factory y ser unos de los líderes en industria 4.0 (CEIT, s.f.).

Esta empresa desarrolló su primer AGV en el año 2007 llamado CEITruck Aurora, que cuenta con 4 modelos con capacidad de carga desde los 500 kilogramos hasta los 3000 kilogramos. Utiliza un sistema de guiado con franjas magnéticas y tag RFID.

CAPÍTULO 3: METODOLOGÍA

Debido a que la naturaleza del proyecto puede dar lugar a cambios en las especificaciones por algún requerimiento no considerado y la necesidad de tener avances funcionales en poco tiempo, se opta por utilizar una metodología iterativa en la que se puedan entregar incrementos que cumplan con un subconjunto de los requerimientos totales, pero que demuestran el correcto avance del proyecto.

Por lo anteriormente dicho, se utilizará RUP como framework de trabajo ya que define pasos concretos para cumplir con las necesidades mencionadas anteriormente. Este framework contempla las siguientes actividades por cada iteración:

1. Modelo de negocios
2. Requisitos
3. Análisis
4. Diseño
5. Implementación
6. Prueba e integración.

Este proyecto no tiene contemplado un modelo de negocio, sino más bien solo contempla el desarrollo de un prototipo funcional y escalable, es por ello por lo que la fase de modelo de negocio no se considerará en el desarrollo de este informe.

3.1 Planificación del proyecto

Con base en la metodología mencionada anteriormente, se planificará la realización de dos iteraciones con inicio y fin correspondientes a los inicios y términos de actividades académicas de la Universidad San Sebastián, esto

quiere decir, que la primera iteración se planificará con el tiempo permitido desde marzo hasta finales de junio, de igual manera, la segunda iteración tendrá una planificación en el tiempo correspondiente desde agosto hasta finales de diciembre.

La primera iteración se centró en el desarrollo de un sistema funcional con un solo vehículo y que consta de las funciones mínimas de ejecución, las cuales permite al vehículo desplazarse de forma independiente de un lugar a otro mediante la llamada desde la aplicación web del usuario. Y la segunda iteración se centró en la mejora del sistema, incorporando la característica de auto regular la velocidad, el funcionamiento de más de un vehículo, mejoras de la aplicación de administración y cliente, entre otras tareas que se detallarán en los cuadros de planificación.

A continuación, se mostrará una tabla con las actividades y tiempos estimados para el desarrollo de la primera y segunda iteración, cabe recordar que la planificación de la segunda iteración se realiza posteriormente de concluir la primera iteración.

3.1.1 Actividades Primera Iteración

N°	Nombre Tarea	Duración	Inicio	Termino
1	**Primera etapa del proyecto**	**79 días**	**12-03-2018**	**28-06-2018**
2	Toma y análisis de requerimientos	4 días	12-03-2018	15-03-2018
3	Planificación del trabajo	3 días	13-03-2018	16-03-2018
4	Búsqueda de alternativas de solución (Vehículo)	5 días	15-03-2018	22-03-2018
5	Análisis de alternativas de solución (Vehículo)	15 días	22-03-2018	12-04-2018
6	Diseño y modelado de piezas 3D	5 días	20-04-2018	27-04-2018
7	Impresión piezas 3D	3 días	30-04-2018	02-05-2018
8	Ensamblaje y armado de vehículo	2 días	03-05-2018	04-05-2018
9	**Programación de vehículo**	**13 días**	**07-05-2018**	**23-05-2018**
10	Programación del seguidor de línea	5 días	07-05-2018	11-05-2018
11	Programación del comunicador	3 días	14-05-2018	16-05-2018
12	Programación ejecución de instrucciones	5 días	17-05-2018	23-05-2018
13	Pruebas funcionamiento del vehículo	5 días	30-05-2018	05-06-2018
14	Diseño de algoritmo cálculo de instrucciones	10 días	27-03-2018	10-04-2018
15	Búsqueda de alternativas de solución (Servidor)	3 días	20-03-2018	23-03-2018
16	Análisis de alternativas de solución (Servidor)	2 días	23-03-2018	27-03-2018
17	**Desarrollo del servidor**	**26 días**	**27-03-2018**	**02-05-2018**
18	Programación del gestor de rutas	6 días	27-03-2018	04-04-2018
19	Programación del calculador de instrucciones	10 días	18-04-2018	02-05-2018
20	Pruebas del servidor	5 días	30-05-2018	05-06-2018
21	Implementación primera etapa	2 días	06-06-2018	07-06-2018
22	Pruebas generales del sistema	15 días	08-06-2018	28-06-2018
23	Corrección de errores primera etapa	15 días	08-06-2018	28-06-2018

Tabla 1 - Actividades primera iteración

3.1.2 Actividades Segunda Iteración

N°	Nombre Tarea	Duración	Inicio	Termino
1	**Segunda etapa del proyecto**	**60 días**	**24-09-2018**	**14-12-2018**
2	Toma y análisis de requerimientos	2 días	02-10-2018	03-10-2018
3	**Cambios Vehículo**	**14** días	**09-10-2018**	**26-10-2018**
4	Programar módulo control de velocidad	7 días	09-10-2018	17-10-2018
5	Programar cambios del comunicador	2 días	18-10-2018	19-10-2018
6	Prueba de cambios realizados	5 días	22-10-2018	26-10-2018
7	Corrección de errores	5 días	22-10-2018	26-10-2018
8	**Cambios de servidor**	**7** días	**09-10-2018**	**17-10-2018**
9	Programar aplicación web de administrador	2 días	09-10-2018	10-10-2018
10	Diseño de algoritmos para operar más de un vehículo	3 días	11-10-2018	15-10-2018
11	Programación cambios del servidor	2 días	16-10-2018	17-10-2018
12	**Construcción de segundo vehículo**	**5** días	**29-10-2018**	**02-11-2018**
13	Impresión de piezas 3D	3 días	29-10-2018	31-10-2018
14	Ensamblaje y armado de vehículo	1 días	01-11-2018	01-11-2018
15	Prueba de funcionamiento del segundo vehículo	1 días	02-11-2018	02-11-2018
16	Pruebas generales del sistema con dos vehículos	10 días	05-11-2018	16-11-2018
17	Corrección de errores	10 días	05-11-2018	16-11-2018
18	**Primer estado de avance**	**6** días	**24-09-2018**	**01-10-2018**
19	Desarrollo de capítulo 1	3 días	24-09-2018	26-09-2018
20	Desarrollo de capítulo 2	2 días	24-09-2018	25-09-2018
21	Corrección observaciones	3 días	27-09-2018	01-10-2018
22	Entrega primer estado de avance	1 días	01-10-2018	01-10-2018
23	**Segundo estado de avance**	**8** días	**22-10-2018**	**31-10-2018**
24	Desarrollo de capítulo 3	4 días	22-10-2018	25-10-2018
25	Desarrollo de capítulo 4	4 días	22-10-2018	25-10-2018
26	Corrección de observaciones	3 días	26-10-2018	30-10-2018
27	Entrega segundo estado de avance	1 días	31-10-2018	31-10-2018
28	**Tercer estado de avance**	**10** días	**19-11-2018**	**30-11-2018**
29	Desarrollo de capítulo 5	5 días	19-11-2018	23-11-2018
30	Desarrollo de capítulo 6	5 días	19-11-2018	23-11-2018
31	Corrección de observaciones	4 días	26-11-2018	29-11-2018
32	Agregar anexos	4 días	26-11-2018	29-11-2018
33	Entregar tercer estado de avance	1 días	31-11-2018	30-11-2018
34	Fecha límite con holgura	1 días	14-12-2018	14-12-2018

Tabla 2 - Actividades segunda iteración

3.2 Plan de pruebas para funcionamiento del sistema

En esta sección se detallarán las pruebas realizadas para comprobar el correcto funcionamiento de cada componente del sistema (pruebas unitarias) y el correcto funcionamiento de cada una de ellas como un único sistema funcional (pruebas de integración). Las pruebas mencionadas a continuación son las planificadas para ejecutarlas en la fase de desarrollo cuando corresponda.

3.2.1 Elementos de pruebas

Los elementos, componente y módulos que son objeto de pruebas serán los siguientes:

- Seguidores de línea
- Lectura RFID
- Ejecución de instrucciones
- Regulador de velocidad
- Conexión al servidor
- Envío de instrucciones desde el servidor
- Notificar RFID al Servidor
- Módulo Cálculo de instrucciones
- Módulo Gestión de pedidos
- Gestor de bloqueos
- Página de administrador
- Página cliente

3.2.2 Funcionalidades a probar

Las funcionalidades primordiales para evaluar son:

- Desplazamiento del vehículo de un punto a otro.
- Comunicación entre el servidor y el vehículo
- Cálculo de instrucciones desde el servidor
- Gestión de pedidos.
- Configuración de mapa, vehículo y materiales en administrador
- Solicitud de pedido desde el cliente
- Notificación de posición y estado de cola hacia el cliente
- Priorización de un vehículo sobre otro para el paso en un mismo punto.

3.2.3 Enfoque de pruebas

Se comenzará realizando pruebas del funcionamiento de cada elemento individualmente en un entorno controlado, realizando las tareas de elementos externo de forma manual para llevar una ejecución controlada y detectar los posibles fallos de ejecución. Luego de realizar las pruebas de funcionamiento individual se comenzarán a ejecutar las mismas incorporando un nuevo elemento a la prueba, de esta manera se irán realizando pruebas del sistema general de forma paulatina, agregando cada vez un nuevo elemento. Se realizarán un mínimo de 10 iteraciones de cada prueba para comprobar el índice de éxito del sistema y poder determinar en qué parte se podría estar produciendo el error, de ser necesario, se realizarán más pruebas hasta poder determinar qué es lo que provoca dicha falla.

3.2.4 Criterios de aceptación o rechazo

El objeto de prueba será aprobado cuando se obtenga un índice de éxito del 90% en las pruebas realizadas, en caso contrario se rechaza su estado actual y se deberá realizar las modificaciones necesarias para su aprobación. Una prueba exitosa será considerada como la correcta ejecución de instrucciones y el alcance de su objetivo sin intervención alguna.

3.2.4.1 Criterios de suspensión

La ejecución de las pruebas se detendrá bajo una de las siguientes condiciones.

- La ejecución de la prueba no es ejecutada por el sistema, evitando realizar más iteraciones.
- Fallo en la primera iteración de prueba, que ocasione un funcionamiento totalmente opuesto al esperado.
- Tres iteraciones realizadas consecutivamente por el mismo fallo.
- Salto de alerta por fuente de alimentación descargada.
- Fallo persistente posterior a cambios realizados.

3.2.4.2 Criterios de reanudación

Las pruebas se reanudarán cuando:

- Corrección de errores en la sección donde fue detectado el fallo.
- Reemplazar la fuente de alimentación por una cargada.

3.2.5 Requerimientos de entornos – Hardware

- Uno o más vehículos completamente ensamblados.
- Un computador con acceso a la red
- Un punto de acceso para crear una red local mediante wifi.
- Uno o más dispositivos móviles que puedan solicitar un pedido.

CAPÍTULO 4: DESARROLLO

4.1 Requerimientos

En un paso previo al desarrollo, se deben analizar las necesidades fundamentales que se desean resolver (Requerimientos funcionales) y las necesidades deseables por los usuarios finales o las restricciones a las que está sometido el proyecto (Requerimientos no funcionales). Por lo tanto, a continuación, se detallan los requerimientos funcionales y no funcionales que se deben contemplar.

4.1.1. Requerimientos Funcionales

Se considerarán los siguientes requerimientos funcionales:

1. Se debe diseñar un vehículo capaz de desplazarse autónomamente.
2. El vehículo debe poder recibir órdenes desde un servidor.
3. Se deben construir al menos dos vehículos.
4. Se debe desarrollar una aplicación donde se puedan realizar pedidos de materiales.
5. En la aplicación se debe seleccionar la estación de trabajo a la que se desea realizar el pedido.
6. La aplicación debe mostrar la posición actual del vehículo mientras existe un pedido activo.
7. Se debe notificar al usuario cuando el vehículo llega a su destino.
8. El usuario debe poder desocupar el vehículo presionando un botón en la aplicación.
9. Si existen más pedidos para el mismo vehículo, los usuarios deben tener un tiempo máximo de espera para desocupar el vehículo.

10. El usuario debe poder cancelar un pedido.
11. Cuando un vehículo no tiene pedidos pendientes, se debe dirigir a una posición específica que se denominará “Home del vehículo”.
12. El sistema debe evitar colisiones de vehículos.
13. Debe existir un sistema de colas, en el que se registren los pedidos pendientes para cada vehículo.
14. Se debe desarrollar una aplicación de administración donde se puedan configurar los aspectos principales del sistema.
15. En la aplicación de administración se deben poder registrar las estaciones de trabajo, vehículos y materiales.
16. En la aplicación de administración se deben listar los pedidos realizados con su estado de finalización.

Los requerimientos anteriormente señalados se visualizan en el siguiente diagrama de casos de uso:

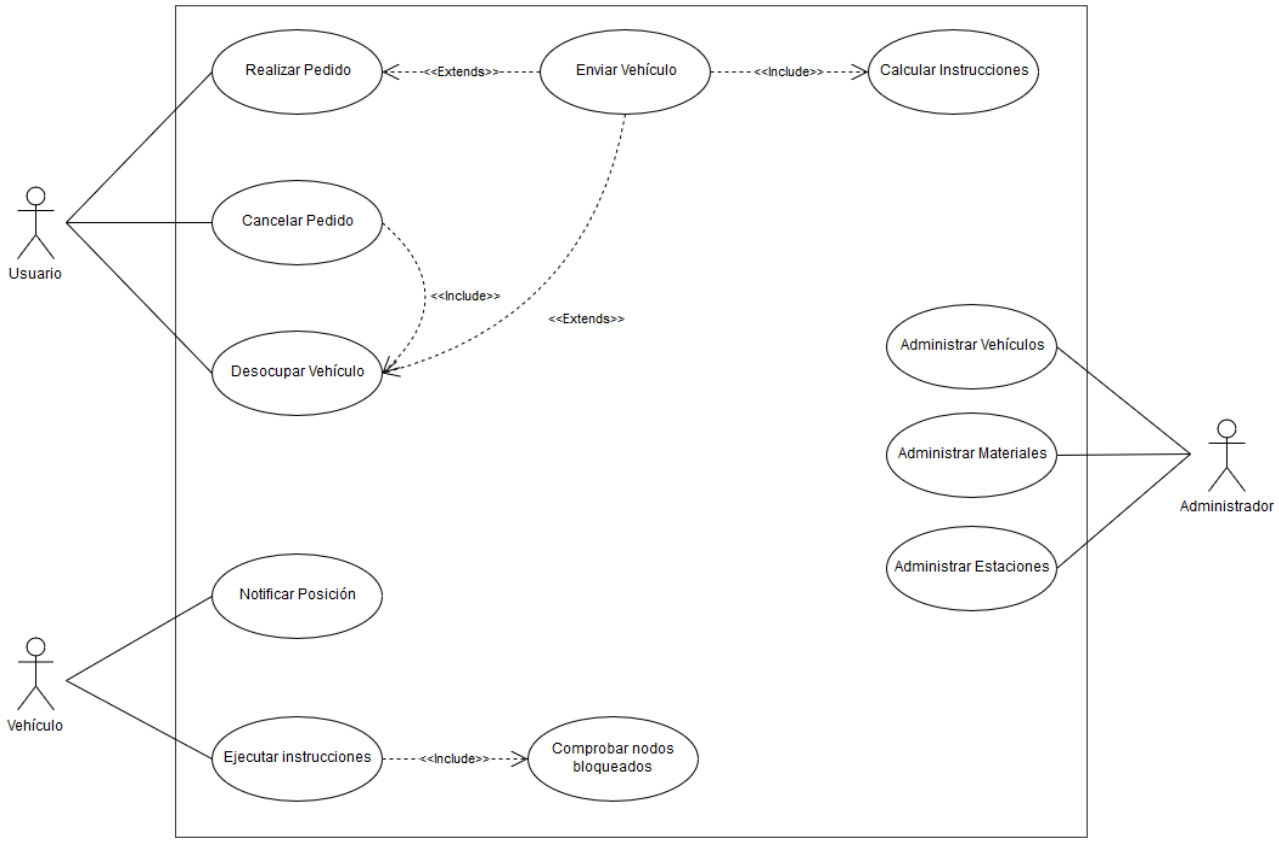

Figura 1 - Diagrama casos de uso

4.1.2. Requerimientos No Funcionales

Se considerarán los siguientes requerimientos no funcionales:

1. El servidor debe ser desarrollado en una tecnología multiplataforma que funcione a lo menos en Windows y Linux.
2. Las aplicaciones del cliente y de administración deben ser web y funcionar tanto en dispositivos móviles como de escritorio.
3. Las interfaces gráficas deben ser sencillas y de fácil uso

4.2 Actividades

A continuación, se explicarán las actividades realizadas en el desarrollo del proyecto, estas actividades principalmente se dividen en el diseño, construcción y programación de los vehículos y el diseño y desarrollo del servidor y las aplicaciones de cliente y administración.

4.2.1. Vehículo

Para el desarrollo de este sistema y su correcto funcionamiento es necesario, en primer lugar, desarrollar el vehículo que se encargará de ejecutar las instrucciones necesarias para poder entregar los materiales al destino solicitado. Es por ello por lo que se deben considerar las alternativas disponibles y condiciones necesarias para su desarrollo.

Las condiciones iniciales para el desarrollo de este prototipo consisten en que los vehículos operarán con un Arduino modelo UNO R3, que es una plataforma de prototipado electrónico (Arduino, s.f.) y estará encargado de ejecutar las instrucciones de desplazamiento y un módulo de conexión wifi ESP8266 desarrollado por SparkFun (Sparkfun, s.f.) que se encargará de la comunicación con el servidor y le entregará las instrucciones al Arduino.

Figura 2 - Sparkfun ESP8266 Thing

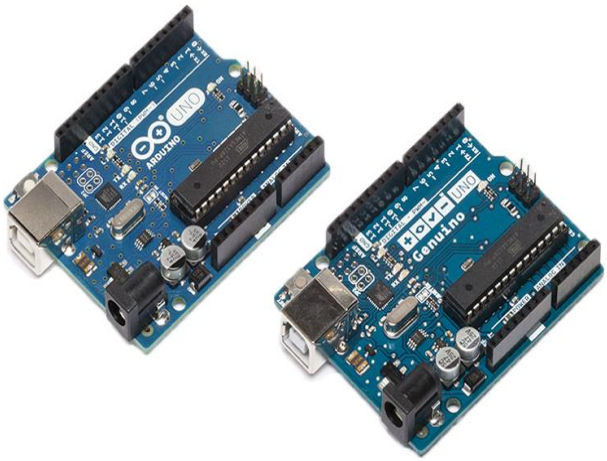

Figura 3 - Arduino UNO R3

A continuación, se detalla el análisis de las alternativas disponibles y diseño de soluciones para el desarrollo del vehículo.

4.2.1.1. Análisis de alternativas y diseño de solución

Sistema de localización

El principal aspecto que se debe analizar para el desarrollo de este vehículo es su localización en el recinto. Este aspecto es un punto crucial para el desarrollo del sistema, ya que, si no se puede conocer la posición actual del vehículo, difícilmente se le puede dar indicaciones para su desplazamiento sin que se presente algún obstáculo que provoque su inmovilización o un mal funcionamiento. Para este aspecto se tiene las siguientes alternativas:

- **Sistema de navegación inercial**: También conocido como Inertial Navigation System (INS) es un sistema de apoyo de navegación usado por barcos, aeronaves, submarinos, misiles y naves espaciales, utilizando sensores que permitan describir su movimiento (acelerómetro) y rotación (giroscopio) para estimar su posición, orientación y velocidad sin la necesidad de una referencia externa (Woodman, 2007). Esto permitirá que el vehículo fuera totalmente autónomo al momento de su desplazamiento en el área de trabajo, ya que solo se le debería indicar cuánta distancia deberá recorrer en una dirección y sentido específico.

- **Localización mediante Wifi**: Es un análogo al sistema de geolocalización GPS ya que utiliza el mismo método para determinar la posición, sin embargo, el sistema GPS tiene grandes dificultades para determinar la posición en ambientes cerrados. La localización mediante Wifi se consigue midiendo el tiempo que tarda un dato salir del punto de acceso (AP) hasta llegar al dispositivo, como la señal se transmite por el aire a la velocidad de la luz se puede determinar la distancia del objeto con respecto al AP, pero es necesario tener una alta precisión de tiempo (en el orden de los nanosegundos) y una cantidad mínima de 3 AP para conocer la posición exacta del objeto. Con todos estos requisitos, se ha llegado a obtener un error de precisión de 1 metro (VIllena Román, 2009) (González Blázquez, Mulas Gómez, & Rivera Retamar, 2006).

- **Sistema de coordenadas cartesiano**: Consiste en la representación del espacio mediante diversos puntos que permiten definir unívocamente la posición de cualquier punto de un espacio euclídeo (Martínez Lendech). De esta forma se puede determinar la posición del vehículo de forma discreta y su historial de posición nos revelaría su dirección y sentido. Para que este sistema de localización pueda funcionar el vehículo debería tener la capacidad de reconocer los puntos desplegados en el área de operación y comunicar su posición, además deberá contar con un sistema de guiado entre punto y punto para que no haya peligro de desvío.

- **Reconocimiento de imagen**: Consistiría en la instalación de diversas cámaras de tal forma que se consiga una vista total del recinto y mediante inteligencia artificial el servidor pueda reconocer al vehículo y la posición en la que se encuentre.

Para la elección del sistema de localización se evaluarán cada uno con una puntuación de 1 a 5, siendo 1 una calificación mala y 5 una calificación excelente, en los siguientes aspectos:

- **Facilidad de implementación**: Consiste en la facilidad de implementación en el laboratorio Make It.

- **Factibilidad**: El nivel de factibilidad que tiene el sistema en el recinto, es decir, que no cuente con demasiados impedimentos o dificultades para su implementación.

- **Tiempo de desarrollo**: Que tanto tiempo se le debe entregar al desarrollo del sistema, se le puntuará más alto si el tiempo de desarrollo es menor.

- **Costo de implementación**: Se trata del costo monetario que se tendría que realizar para el desarrollo de dicho sistema de localización, mientras menos costoso sea su implementación, tendrá una mayor calificación.

Categoría	Facilidad de implementación	Factibilidad	Tiempo de desarrollo	Costo	Total
Importancia	20%	30%	20%	30%	100%
INS	5	2	2	4	3,2
Localización Wi-Fi	4	3	3	2	2,9
Sistema Cartesiano	4	3	3	5	3,8
Reconocimiento de Imagen	2	1	2	1	1,4

Tabla 3 - Matriz de decisión sistema de localización

Análisis de la tabla comparativa (Tabla 3)

El sistema de reconocimiento de imagen queda descartado de forma casi inmediata, debido que el recinto donde se instalará el sistema AGV contará con muebles y personas trabajando en determinadas zonas, generando posibles problemas cuando se quiera encontrar la posición del vehículo y que se necesitará una máquina de alto rendimiento para el procesamiento de las imágenes proveniente de las cámaras, esto supondría un alto costo para el proyecto.

La localización mediante Wifi resulta una propuesta atractiva, pero su mayor inconveniente es su requisito de alta precisión del tiempo para una localización más exacta, aun así, teniendo esa precisión en el tiempo se contaría un margen de error de 1 metro, lo suficiente para poder confundir un punto de trabajo con otro.

El sistema INS es una buena idea para el desarrollo del vehículo, ya que le brindaría autonomía total al vehículo para que se pudiera desplazar, pero mediante pruebas empíricas realizadas, se pudo comprobar que su factibilidad se ve afectada por la calidad de los sensores disponibles para su desarrollo, estos sensores cuentan una inestabilidad y margen de error lo suficientemente grande como para generar errores significativos en el cálculo del movimiento. La implementación de mejores sensores supondría un aumento considerable del costo y desarrollo para el control de los errores tomaría más tiempo del que se dispone.

El sistema cartesiano resulta ser la opción más factible para el desarrollo del sistema, no corresponde al sistema de localización más innovador y es propenso a los errores de desvío entre los puntos, pero no supone grandes costos y dificultades de implementación, es por estas razones que el sistema de localización será implementado de esta manera.

Análisis de implementación Sistema Cartesiano

Para este sistema de localización cada uno de los puntos que se vaya a ubicar en el área de implementación corresponderá a una estación de trabajo, pero hay que tener en cuenta que no se podrán ubicar puntos en toda el área disponible, ya que se cuenta con muebles y personas trabajando en ciertas zonas del recinto, colocar puntos en esos lugares podría generar la inmovilización o deterioro del vehículo. Por ello, solo se implementarán puntos en los cuales el vehículo pueda desplazarse sin mayores dificultades. También se debe ubicar un punto inicial que corresponde a la estación de espera del vehículo, también llamado “Home del vehículo” en los requerimientos funcionales.

Para el correcto funcionamiento de este sistema de localización, es necesario resolver algunos problemas con respecto al reconocimiento y notificación de cada estación de trabajo por parte del vehículo y el desvío de este durante su desplazamiento entre cada estación de trabajo.

Para solucionar el primer problema mencionado se puede optar por una de las siguientes opciones:

- **Marca óptica**: Similar a los códigos de barra, o códigos de respuesta rápida, también conocidos como QR por sus siglas en inglés (Quick Response), cada punto debería tener un código que lo identificara en el plano para que el servidor pudiera entregarle información al vehículo.

- **Tag RFID**: Al igual que la marca óptica anterior, cada punto en el plano correspondería a un tag RFID, la diferencia es que cada uno de estos tags ya contienen un identificador único que lo haría fácilmente reconocible.

Figura 4 - Ejemplo marca óptica QR

Figura 5 - Tags RFID

Se consideró utilizar tags RFID por las ventajas que tiene sobre las marcas ópticas. Las marcas ópticas dependen de su legibilidad para su correcto reconocimiento, estando situadas en el suelo, son propensas a mancharse y a dañarse con mayor facilidad, haciendo más difícil esta tarea y a su vez, aumentando la frecuencia de mantenimiento. Mientras que los tags RFID no se ven afectados por su deterioro visual y solo dependen de la cercanía y calidad del sensor para su correcto funcionamiento.

Luego de solucionar el problema de identificar las estaciones de trabajo por parte del vehículo, queda resolver el problema de desvío entre cada uno de ellos. Este problema surge debido a que los motores utilizados tienen variaciones de desempeño a la misma potencia entregada, esto sumado a las irregularidades del suelo, el vehículo tiende a desviarse de su camino.

Para remediar este problema, entre cada estación de trabajo se trazará una línea blanca, que representará el camino que el vehículo deberá recorrer para llegar a la siguiente estación. Para que el vehículo siga este camino se le implementarán tres sensores seguidores de línea en la parte frontal del chasis, estos sensores entregaran la información correspondiente cuando el vehículo se encuentre en el camino correcto (señal del sensor medio), o si se está desviando hacia la derecha o izquierda (señal de los sensores laterales), estos sensores están diseñados para captar líneas de color negro, pero como el suelo donde se implementará este sistema es de un tono oscuro, se optó a trabajar con su lógica inversa y seguir líneas de color blanco, manteniendo así, el mismo contraste que si fuera a seguir líneas negras. La explicación más detallada de cómo funciona esta lógica se detalla en apartado “Programación del vehículo” correspondiente al punto 4.2.1.3.

4.2.1.1 Plan de implementación Sistema Cartesiano

La implementación de los tags RFID junto a las líneas blancas se debe regir bajo ciertas condiciones que son necesarias para facilitar el cálculo de instrucciones y la operatividad del vehículo. Las condiciones para su implementación son las siguientes:

- La unión de un tag RFID con una línea blanca se denomina, conexión
- Cada tag RFID no podrá tener más de 4 conexiones.
- Las conexiones deben estar posicionadas de tal forma que entre dos conexiones haya un ángulo de 90° o de 180°, formando dos rectas perpendiculares en su totalidad, teniendo el tag RFID en el centro.
- Si un tag RFID tiene un solo camino o dos caminos que forman un ángulo de 90°, su largo se debe extender no más de 5 cm desde el tag RFID.
- Entre cada tag RFID debe haber una separación mínima de 35 cm o la magnitud correspondiente al largo del vehículo, solo en caso de que este sea más largo.

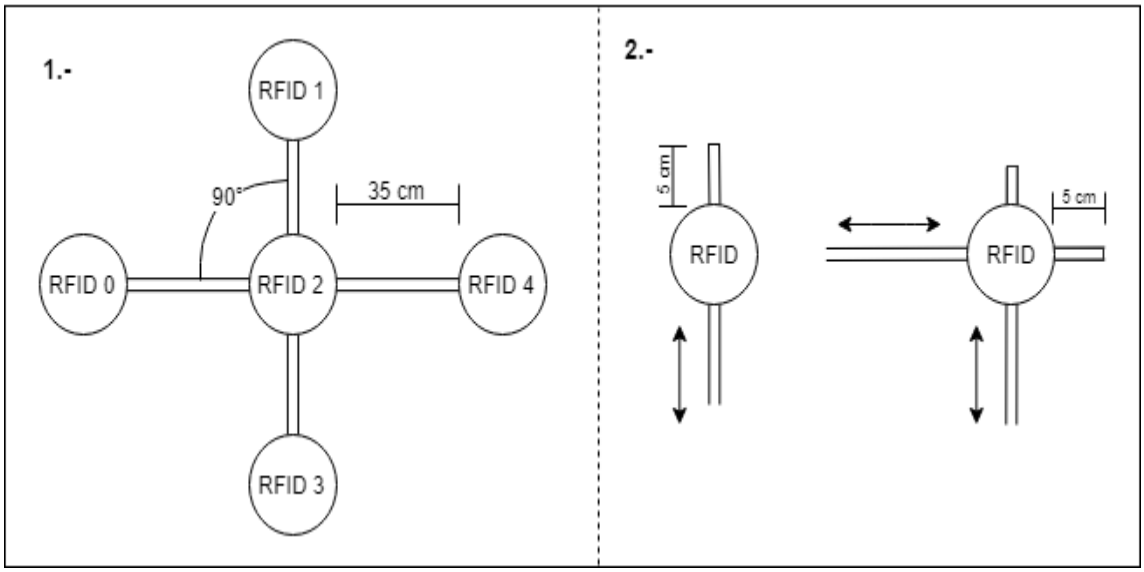

Figura 6 - Ejemplo de implementación de líneas y tags

Funcionamiento del vehículo en el Sistema Cartesiano

Ya sabiendo cómo localizar y guiar el vehículo entre cada una de las estaciones de trabajo, es necesario aclarar cómo será el funcionamiento o el modo de operar del vehículo en este sistema. Con la forma de situar las estaciones de trabajo descrita anteriormente, limitamos al vehículo a ejecutar solo 5 instrucciones de movimiento: Ir hacia delante, girar a la izquierda, girar a la derecha, girar hacia atrás y detenerse. Estas instrucciones solamente serán ejecutadas cada vez que se detecte que el vehículo está en una estación de trabajo, ejecutarlas entre una estación y otra no tendría sentido ya que en este lugar el vehículo solo debe seguir el camino descrito por la línea blanca. Cabe mencionar que los movimientos de giro que el vehículo realizará son movimientos de rotación en su propio eje.

Forma de desplazamiento

Al tener solucionado los problemas mencionados anteriormente sobre localización y desvío, es momento de determinar cómo se desplazará el vehículo. Los motores disponibles en el laboratorio son de corriente continua y para su control se necesitará un controlador L293d, también conocido como “puente H”, que permite alternar el flujo de corriente que se dirige hacia los motores y así cambiar su dirección de giro. Este controlador solo admite un máximo de 2 motores, teniendo en cuenta esto, las posibilidades disponibles son:

- **Desplazamiento con 4 ruedas**: El vehículo contaría con un motor por cada rueda (Figura 7), el desarrollo de este modelo permitiría tener más fuerza de desplazamiento, pero se necesitaría crear un chasis resistente que pudiera albergar los cuatro motores y sus respectivos controladores, también implica un mayor consumo de energía y una mayor cantidad de conexiones desde el Arduino hasta los dos controladores.

- **Sistema de tracción por orugas:** El vehículo se confeccionará sobre un chasis prefabricado (Figura 8) con este sistema, el cual le otorgará una mayor flexibilidad para enfrentar pequeños obstáculos que interfieran en su camino. Este chasis cuenta con solo dos motores, pero esto implica que realizarán el doble de trabajo con respecto a un vehículo con 4 ruedas y podría reducir su vida útil si se le exige demasiado.

Figura 7 - Motor utilizado en vehículos de 4 ruedas

Figura 8 - Chasis tanque

El vehículo será desarrollado sobre el chasis con sistema de tracción por oruga, si bien ambas propuestas son viables para el desarrollo, se escogió esta opción por su facilidad de modificación e implementación de sensores, además siendo un producto comercial, sus medidas son estándar y permite realizar réplicas de manera más sencilla que si se desarrolla un nuevo chasis.

Fuente de Alimentación

La fuente de poder debe entregar la potencia y energía suficiente para que cada componente del vehículo funcione correctamente, en las primeras pruebas realizadas al prototipo se utilizaron 2 baterías en serie de iones de litio (Li-ion) de 3.7v 4200 mAh (Figura 9). El rendimiento observado con estas baterías en las primeras pruebas generales del sistema fueron bastantes aceptables, pero luego de un tiempo de uso, comenzaban a producirse fallas en su funcionamiento tales como: Reinicio de los microcontroladores, inmovilidad de los motores y

funcionamiento deficiente de los sensores de líneas. Luego de realizar reiteradamente estas pruebas y obtener resultados similares en cada iteración, se decidió cambiar está baterías por una de polímero de litio (Li-po) de 12v 2200 mAh (Figura 10). Con esta batería el tiempo de uso continuo aumentó considerablemente y durante este tiempo no se presentaron fallas en el funcionamiento de los componentes del vehículo.

Figura 9 - Baterías Li-ion 3.7v 4200mAh

Figura 10 - Batería Li-po 12v 2200 mAh

Regulador de potencia

Es sabido que mientras mayor masa tenga un objeto, mayor fuerza se le debe aplicar para poder sacarlo de su estado de reposo **(ESTO NO ES VERDAD F=m*a, con la misma fuerza se puede sacar del reposo mayores masas a aceleraciones mas pequeñas).** En tal caso, la potencia que se le deberá entregar a los motores dependerá del peso que tenga que cargar, es por esta razón que el vehículo deberá tener la capacidad de autorregular la potencia que le entregue a los motores, para que estos generen la fuerza necesaria para romper la inercia y para evitar que el vehículo se desplace demasiado rápido y se desvíe en algún punto del camino.

Para que el vehículo pueda ajustar la potencia de los motores es necesario que reciba una retroalimentación de que el motor ya ha superado la inercia y se está moviendo a una velocidad constante, para ello, se le agregaran dos sensores de líneas en la parte trasera del vehículo (uno por cada motor) estos sensores se encuentran al interior del vehículo y enviarán una señal cada vez que capten una de las 4 franjas blancas equidistantes que se encuentran en la cara interior del engranaje principal.

Con el flujo de señales que envíen los sensores, se podrá calcular el tiempo entre cada una de ellas y por consiguiente podríamos calcular su frecuencia. Con esta medida se podrá retroalimentar el algoritmo que se encargue de ajustar la potencia de los motores ya que la relación entre esta y la frecuencia es directamente proporcional, es decir, si los motores no se encuentran en movimiento, la frecuencia será 0, haciendo que la potencia aumente para llegar a la frecuencia esperada y, al contrario, si la frecuencia es más alta que la esperada, la potencia de los motores deberá disminuir.

Una vez analizada las diversas alternativas para la confección del prototipo, se comenzará a explicar detalladamente su ensamblaje.

4.2.1.2. Modelado 3D y ensamblado

Para facilitar la implementación de sensores y componentes que se mencionaron en el punto anterior, fue necesario diseñar y modelar piezas que se ajustaran a las necesidades de cada componente. Con esto se busca tener una implementación más limpia y compacta, que permita mantener cada componente en su sitio, asegurando su correcto funcionamiento y utilizando de mejor manera el espacio disponible. Las piezas fueron diseñadas y modeladas en Autodesk Inventor (licencia académica) y fabricadas en una impresora 3D. A continuación, se detalla brevemente las piezas modeladas para cada componente, los planos de cada pieza se podrán encontrar de forma más detallada en el ANEXO D.

Soporte Arduino y ESP8266

En primer lugar, hay que designar las posiciones de los elementos que se encargaran de controlar el vehículo, como el espacio disponible es compacto e irregular, el Arduino se posicionará por encima de la base del chasis, permitiendo esconder el cableado de los sensores y colocar otros componentes bajo él.

Como se puede apreciar en las Figura 11 y Figura 12, el soporte del Arduino consiste básicamente en una placa plana con el mismo ancho que el interior del chasis, para así poder fijarlo a sus paredes, esta cuenta con una geometría especifica que se debió tener en cuenta para que la pieza se adapte correctamente al chasis del vehículo. El Arduino queda situado exactamente al centro de la pieza y para evitar que se mueva de su lugar, se le añadieron paredes en los laterales y en la parte trasera, además se le añadió un orificio a cada lado para el paso de cableado que va desde el Arduino hasta los demás sensores y componentes.

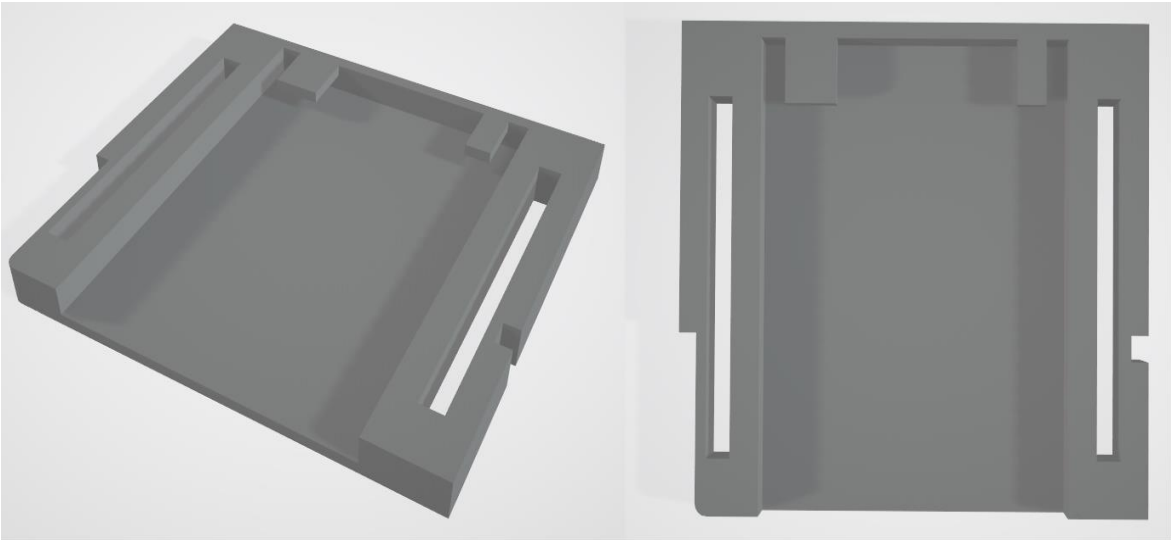

Figura 11 - Modelo 3D Pieza de soporte del Arduino

Figura 12 - Arduino y su soporte impreso

Como el ESP8266 es considerablemente más compacto que el Arduino UNO, su posicionamiento dentro del vehículo es más sencillo, este se situará en una de las paredes del chasis en frente del Arduino. Como se muestra en las Figura 13 y Figura 14, su soporte consiste en una especie de canastilla, cuenta con una cara totalmente plana que se fijará en la pared y una cara que deja al descubierto la zona de conexión del ESP8266.

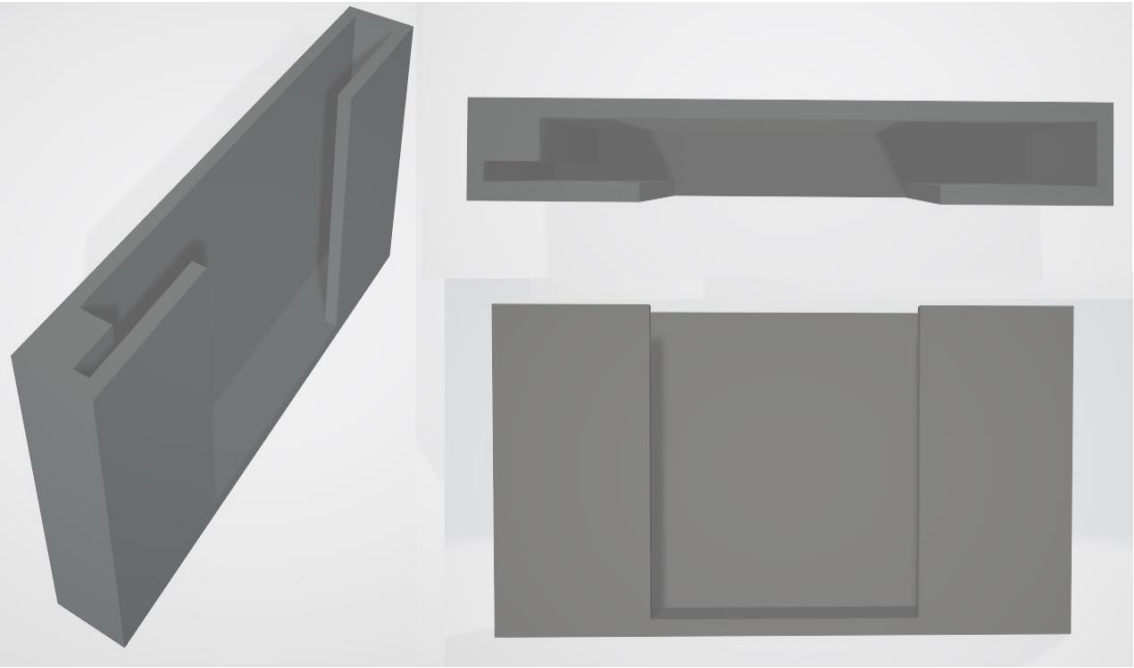

Figura 13 - Modelo 3D soporte ESP8266

Figura 14 - ESP8266 en pieza impresa fijada al chasis

Lector RFID

El lector RFID se debe situar en la parte inferior del chasis para que pueda reconocer correctamente los tags de cada estación de trabajo, además es necesario que este se encuentre en el centro de todo el vehículo. Esta particularidad se debe a que el vehículo solo ejecutará una acción cuando se encuentre en una estación de trabajo, teniendo en cuenta esto, si el lector se sitúa en cualquier otro lugar y si la acción a ejecutar corresponda a girar sobre su propio eje, al término de ejecutar esta acción el vehículo puede quedar mal posicionado para seguir su camino y esto conlleva a un mal funcionamiento.

El chasis cuenta con un compartimiento en la parte inferior, que en un principio fue diseñado para ubicar las baterías. pero debido que se encuentra en el centro del chasis, se utilizará para instalar el lector. Para que quede bien acoplado a este lugar, se tomaron las medidas de la tapa del compartimiento existente y se le realizó una modificación para poder fijar el lector en ella y facilitar la lectura del tag.

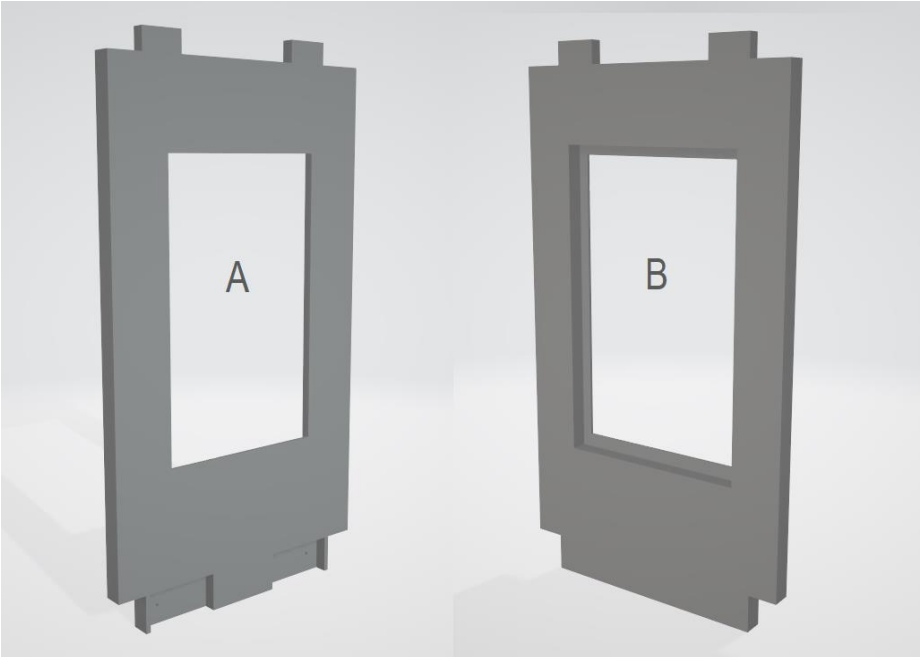

Figura 15 - Modelo 3D soporte lector RFID. Vista frontal (A), Vista trasera (B)

Como se muestra en la Figura 16, el lector se acopla en el orificio diseñado de la pieza mostrada, de tal forma que sus conectores queden hacia el interior del vehículo, estos pasan por un orificio realizado en el compartimiento de las baterías para poder realizar su conexión al Arduino.

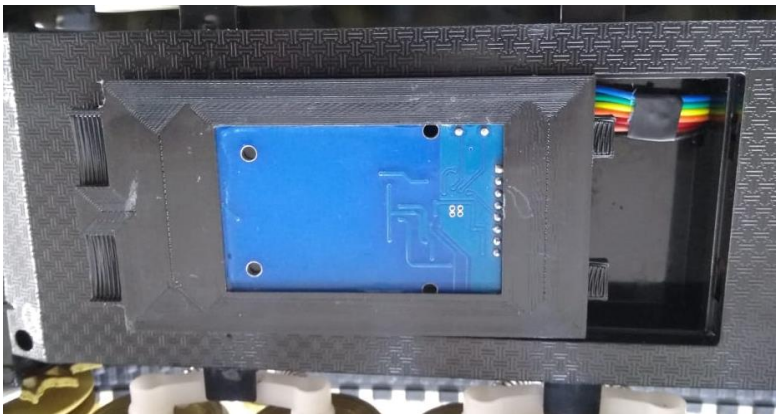

Figura 16 - Soporte lector RFID impreso

Seguidores de línea

Se continúa con la instalación de los seguidores de líneas, que del mismo modo que el lector RFID, estos deben ir ubicados en la parte inferior del vehículo, pero con la diferencia que se deben posicionar en la parte frontal del chasis, para así reconocer de forma más precisa la dirección del vehículo.

Los sensores de líneas se deben encontrar a la misma distancia entre ellos y la distancia entre los dos sensores laterales debe ser por lo menos del mismo ancho que la línea blanca. La parte frontal del chasis es prácticamente lisa y cuenta con una ligera inclinación por estética. Realizar cortes directamente en esta parte del chasis no le daría un acabado adecuado y prolijo, sin mencionar que los sensores podrían no quedar posicionados a la misma distancia. Como se muestra en las Figura 17 y Figura 18, la pieza que se diseñó para los sensores de línea, corresponde a una guía para que estos se encuentren a la misma distancia entre ellos y queden de forma horizontal al suelo adaptándose a la inclinación del chasis.

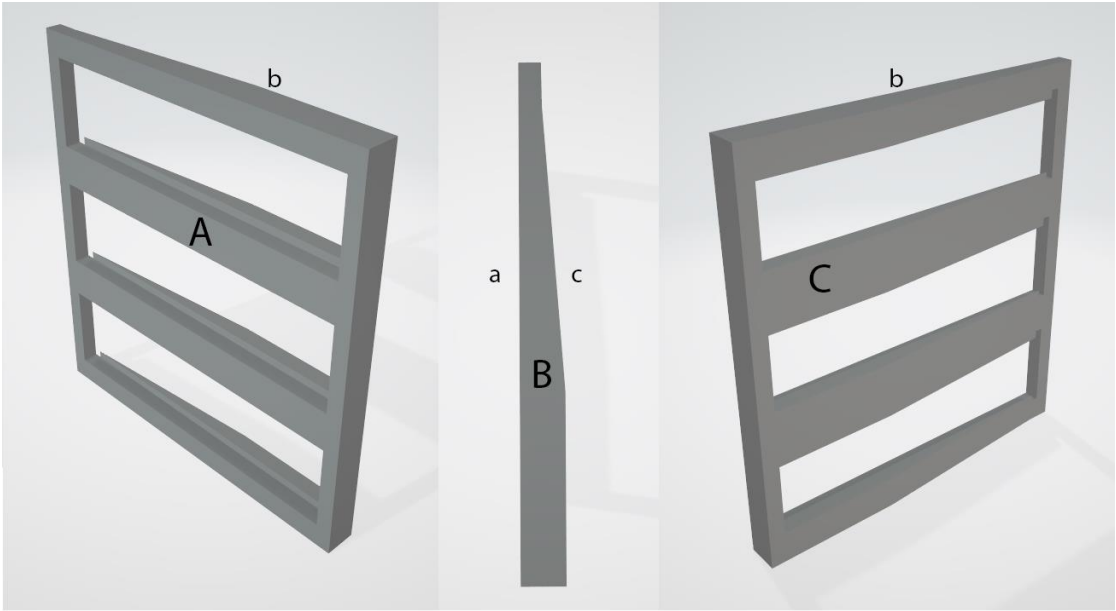

Figura 17 - Modelo 3D soporte seguidores de línea. Cara frontal (A), Cara lateral (B), Cara trasera (C)

Figura 18 - Soporte sensores de línea impreso y ensamblado

Contador de Velocidad

Los sensores con los cuales se medirá la frecuencia de giro de las ruedas no cuentan con un soporte firme que los mantenga en su posición, debido a que el lugar en donde se deben ubicar se encuentran los motores y forman una forma bastante irregular. El diseño que se ve en las Figura 19 y Figura 20, se debe a la forma del espacio libre entre el sensor y la caja de engranajes que forman parte de los motores, de esta forma la pieza busca llenar este espacio vacío para otorgarle mayor área de soporte al sensor y así quede totalmente fijo en el lugar que corresponda.

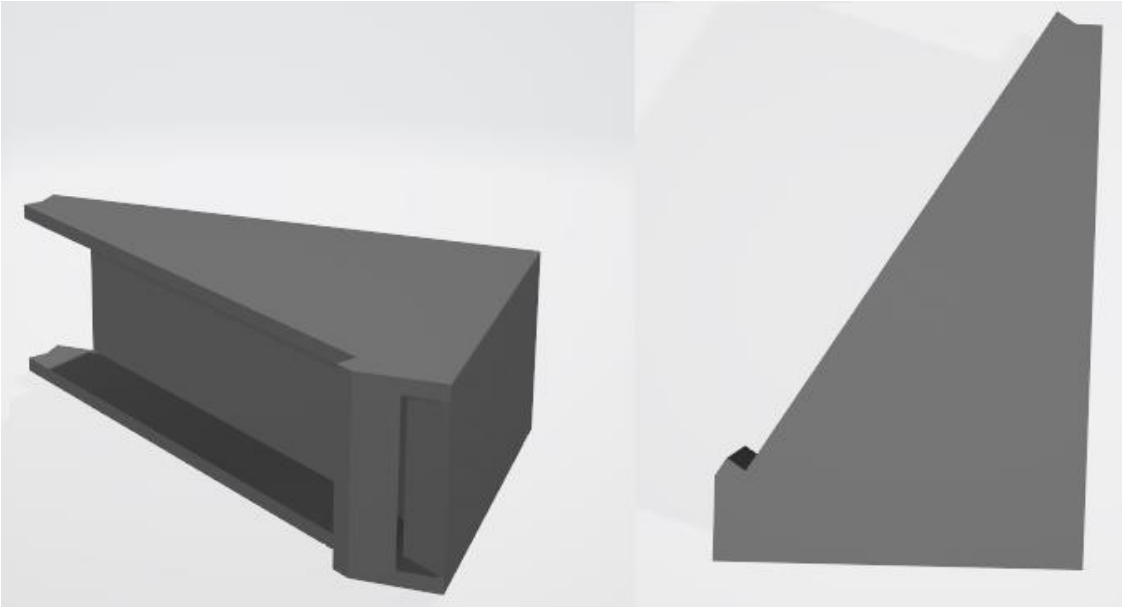

Figura 19 - Modelo 3D soporte sensores de velocidad

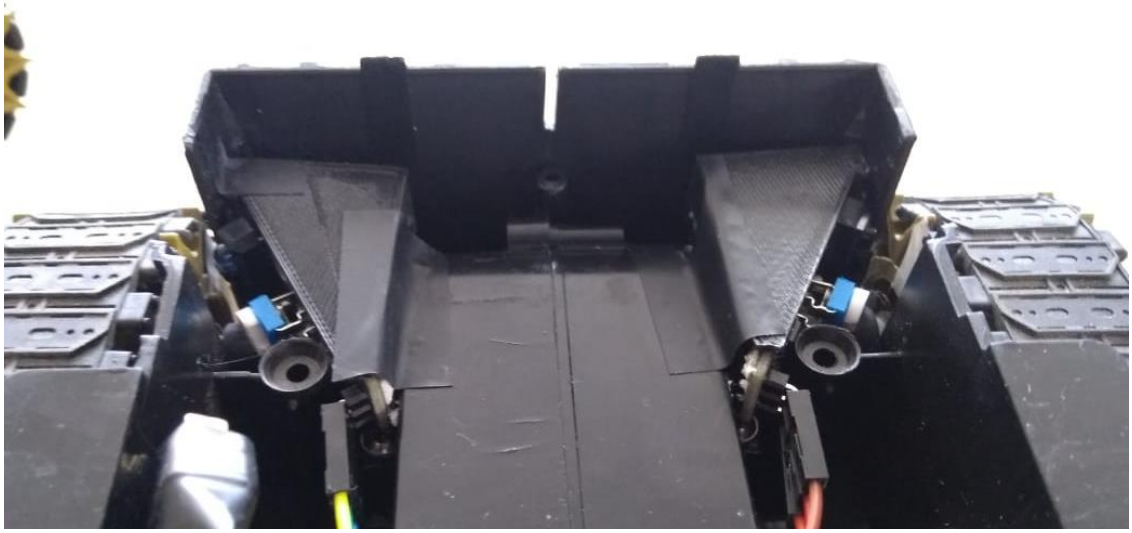

Figura 20 - Soporte sensores de velocidad impreso y ensamblados

Se consideraron también, piezas que irán unidas al chasis del vehículo (Figura 21 y Figura 22) con el fin de soportar las cargas que estos transporten. Estas piezas irán ubicadas en la parte trasera y frontal del vehículo, para nivelar la diferencia de altura entre estos dos puntos, se modelaron piezas de distintos tamaños para compensar esta diferencia.

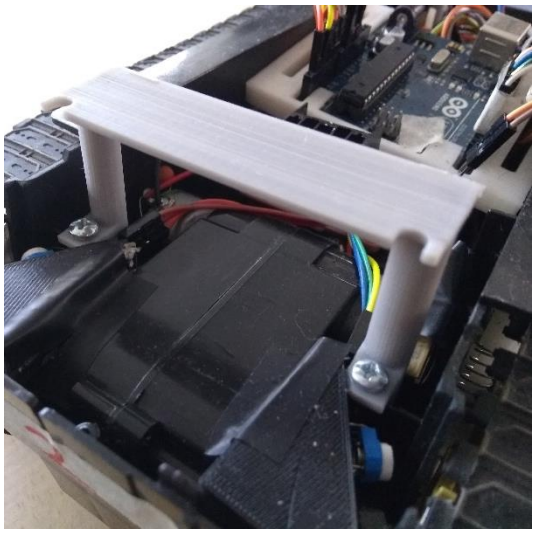

Figura 21 - Soporte de carga trasero

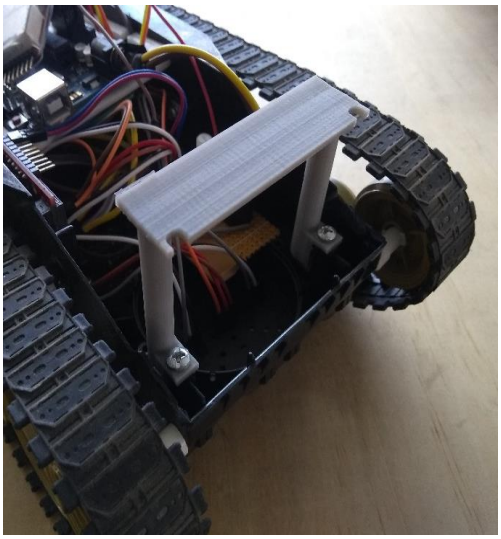

Figura 22 - Soporte de carga delantero

Posición de los componentes

Ya explicadas las piezas de soporte para la mayoría de los componentes, a continuación, se mostrará un par de ilustraciones que describen a grandes rasgos las posiciones de todos los componentes dentro del chasis.

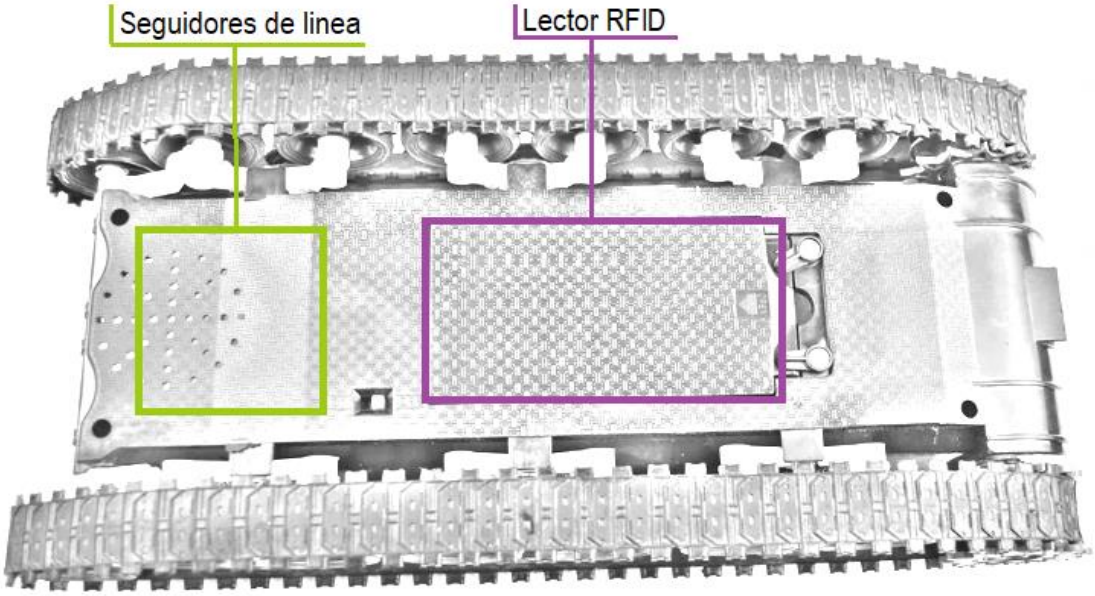

Figura 23 - Vista inferior del vehículo. Se indica la posición del seguidor de línea y el lector RFID

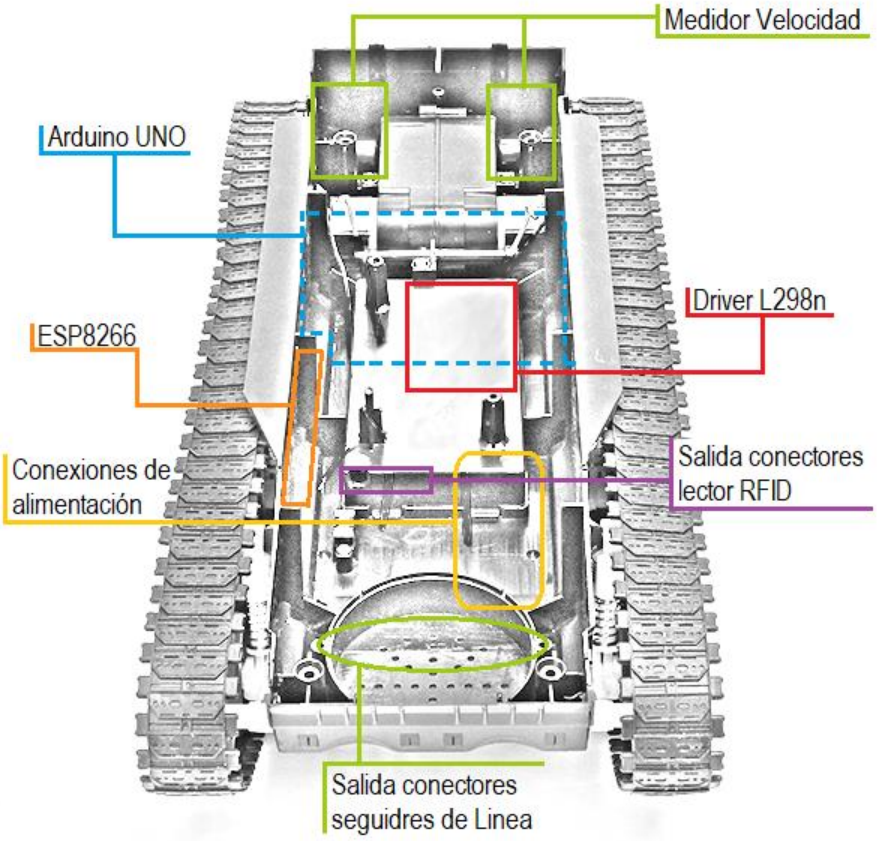

Figura 24 - Vista superior del vehículo, se indican la posición de los componentes

Conexiones Arduino

A continuación, se muestra un diagrama de las conexiones proveniente desde el Arduino hasta los sensores y componentes.

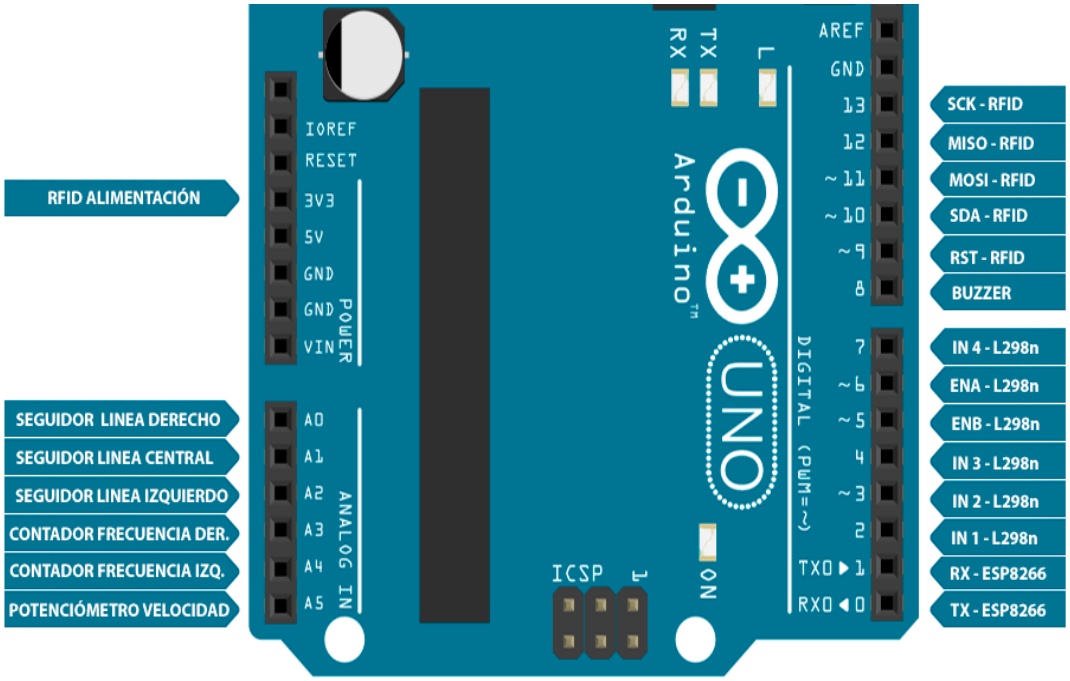

Figura 25 - Diagrama de conexiones del Arduino

4.2.1.3. Programación del vehículo

El vehículo cuenta con dos entidades principales que permiten su funcionamiento, estas entidades están formadas por: una placa Arduino UNO y una placa ESP8266, las cuales trabajan en conjunto para cumplir las indicaciones entregadas por el servidor y hacer que el vehículo pueda llegar a su destino.

El ESP8266 sabe cuáles son las instrucciones que se deben realizar para llegar al destino, debido que es el único que puede entablar comunicación con el servidor, y es el encargado de entregar estas instrucciones al Arduino. El Arduino solo se encarga de realizar los movimientos que le ha indicado el ESP8266 y seguir el camino que se encuentra frente a él.

A continuación, se explicará de forma más detallada cuales son las tareas que realiza cada entidad y sus algoritmos más relevantes.

Arduino UNO

Esta entidad se encarga principalmente de obtener el identificador RFID de la estación de trabajo en la que se encuentra y el control de los motores, que mediante los seguidores de líneas y el medidor de velocidad obtiene una retroalimentación del movimiento del vehículo, permitiendo regular la potencia y sentido de los motores para que pueda desplazarse de forma correcta y sin desviarse. Comenzaremos explicando los tipos de instrucciones que puede realizar el vehículo, estos son:

- ***Seguir adelante***: Consiste en colocar el movimiento del motor izquierdo en sentido antihorario[1] y el motor derecho en sentido horario[1], las señales de los 3 sensores de líneas en la parte frontal regulan la potencia de los motores para que estos no se desvíen del camino, la lógica del seguimiento de líneas es explicada más adelante en este mismo apartado.

- ***Girar derecha:*** Los dos motores giran en sentido antihorario[1], para este movimiento el vehículo entra en un primer bucle de giro hasta que los 3 sensores de línea no envíen señales altas (Se encuentran fuera de la línea), luego se entra a un segundo bucle de giro hasta que los sensores detectan nuevamente una línea blanca. El primer bucle de giro es para dejar de reconocer el camino del cual viene el vehículo y el segundo es para encontrar el nuevo camino que debe seguir.

- ***Girar Izquierda:*** Es la misma lógica descrita en el movimiento anterior, con la diferencia que ambos motores giran en sentido horario[1].

- ***Girar atrás 1:*** Esta instrucción es para cuando el vehículo debe regresar por el mismo camino del que proviene y la estación de trabajo en la que se encuentra, no cuenta con un camino perpendicular hacia la derecha, esta instrucción ejecuta solo una vez la instrucción Girar Derecha.

[1] Para identificar el sentido del giro, se debe observar la cara exterior del chasis

- ***Girar atrás 2:*** Esta instrucción es para el mismo propósito que la descrita anteriormente, pero con la particularidad de que la estación de trabajo en la que se ejecutará cuenta con un camino perpendicular hacia la derecha, esta instrucción ejecuta dos veces la instrucción Girar Derecha.

- ***Detener***: Detiene los motores entregando potencia igual a 0.

Para que el vehículo no se desvíe de su camino entre dos nodos, se utilizan sensores infrarrojos que permiten identificar si el vehículo se encuentra centrado en la línea o si se ha desviado de esta. Cada sensor entrega el valor 0 o 1 dependiendo si se encuentra en una línea negra o blanca respectivamente. El vehículo irá alineado al camino cuando solo el sensor central está encendido, esta situación solo requiere que el vehículo siga avanzando, cuando el vehículo se desvíe hacia un lado, se encenderá el sensor del lado contrario, en esta situación se debe detener el motor que corresponda al lado que se está desviando el vehículo, de esta forma girará levemente para lograr estar centrado nuevamente en la línea.

A continuación, se muestra una imagen que explica la acción de los motores según la posición de los sensores sobre la línea, las flechas en la imagen indican el movimiento o detención de los motores.

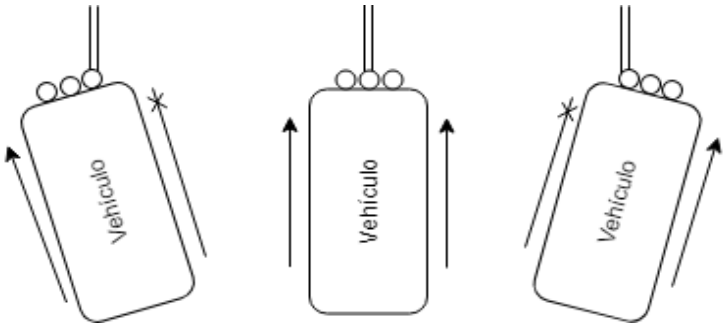

Figura 26 - Diagrama del movimiento de los motores según la posición de los sensores de línea

En la tabla se muestran detalladamente las acciones del vehículo según los valores de cada sensor.

Sensor Izquierdo	Sensor Central	Sensor Derecho	Acción
0	1	0	Ambos motores encendidos
1	0	0	Motor izquierdo con potencia 0
0	0	1	Motor derecho con potencia 0
1	1	0	Motor izquierdo con potencia 0
0	1	1	Motor derecho con potencia 0
1	1	1	Ambos Motores encendidos

Tabla 4 - Descripción de acciones a realizar según la combinación de señales de los sensores de línea

Otras de las funciones que debe realizar el Arduino, es que debe regular la potencia que se le entrega a los motores, para que el vehículo tienda a desplazarse a una velocidad constante y no se produzcan errores o desvíos por los diferentes desempeños de cada uno de los motores con una potencia similar.

El ajuste que se le realiza a la potencia de cada motor corresponde a un incremento, o una disminución, de un porcentaje arbitrario de la diferencia que hay entre la velocidad deseada (ajustada manualmente por un potenciómetro), y la velocidad real de cada motor. El cálculo de la potencia se ve más claramente en la siguiente ecuación.

$$P_{t+1} = P_t + \alpha(v_d - v_r)$$

Ecuación 1 - Aumento de potencia

Donde:

$P: Potencia, \alpha: Porcentaje\ de\ incremento, v_d: Velocidad\ deseada, v_r: Velocidad\ actual$

Para obtener la velocidad de cada motor, se utiliza la información entregada por el medidor de velocidad que se ubica en la parte trasera del vehículo. Estos sensores entregan un flujo de señales correspondientes a la lectura de una de las 4 franja blancas situada en la cara interna de cada rueda (Figura 27). Se calcula el tiempo entre cada una de las señales y se obtiene la velocidad correspondiente a cada motor.

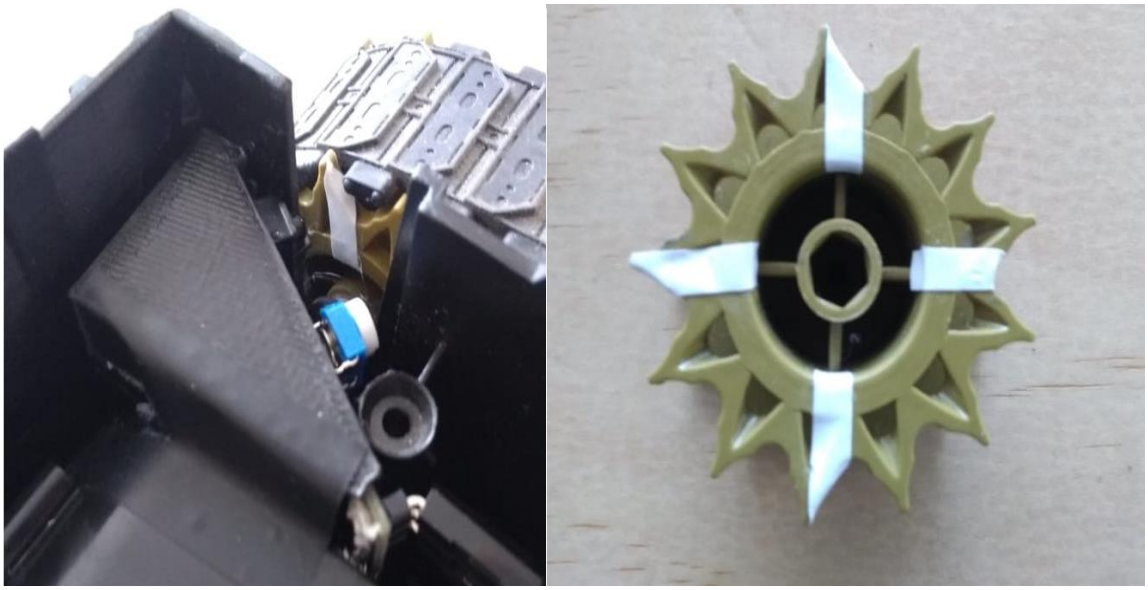

Figura 27 - Posición del sensor y la división de la cara interior del engranaje principal con las franjas blancas.

Utilizando la función ***millis()*** incorporada en la plataforma Arduino, nos entrega el tiempo de ejecución en milisegundos desde que la placa se ha encendido y para obtener el tiempo entre cada señal de los sensores, se obtiene el tiempo de ejecución entre cada una de ellas y se calcula la diferencia entre el tiempo actual y el tiempo obtenido en la lectura anterior. Si calculamos la velocidad con la unidad de tiempo en milisegundos, esta nos dará un valor muy pequeño que para algunas operaciones el microcontrolador podría omitir valores o simplemente considerar el valor como 0, por ser demasiado pequeño, es por ello por lo que se realiza la conversión a decisegundos.

Obtenido ya el tiempo entre cada intervalo ya se puede calcular la velocidad de giro de cada rueda, pero como no se tiene una medida exacta de cuál es el tamaño de la rueda por sus irregularidades, se **trabajara** con una unidad de un cuarto de vuelta, ya que las franjas blancas dividen la rueda en 4 partes iguales, en otras palabras, la velocidad que se obtendrá de este cálculo corresponde a cuantos cuartos de vuelta se realizan en una décima de segundo.

Por último y no menos importante, el Arduino es el único que puede obtener el identificador de la estación de trabajo en que se encuentra, es por eso por lo que cada vez que obtiene el identificador de alguna estación de trabajo, este se lo debe comunicar al ESP8266, para que le entregue la instrucción correspondiente a ese punto.

ESP8266

El ESP8266 es el único que tiene la capacidad de conectarse a una red y establecer una comunicación con el servidor, y solo estará encargado de indicarle al Arduino **cuales** son las instrucciones que debe realizar cuando llegue a una determinada estación de trabajo. El ESP8266 se comunica por WebSocket con el servidor mediante la librería WebSocketsClient, de esta manera el ESP8266 podrá obtener la información de la ruta sin tener que estar preguntando constantemente (long polling) al servidor y evitando la saturación de la red.

La información que el ESP8266 recibe desde el servidor viene estructurada en formato JSON, que con la ayuda de la librería *ArduinoJson* el ESP8266 es capaz de decodificar esta información para el Arduino. La información enviada es un listado de instrucciones junto al identificador de la estación de trabajo donde se deben ejecutar, estas instrucciones son una codificación numérica de los movimientos descritos anteriormente en la explicación del funcionamiento del Arduino.

Una vez que el ESP8266 haya recibido el conjunto de instrucciones este lo guardará en memoria y cada vez que reciba un identificador de una estación de trabajo por parte del Arduino, este le entregara a él la instrucción que debe ser ejecutada en ese punto y posteriormente notificará este identificador al servidor quien guardará su historial de desplazamiento.

4.2.2 Servidor y aplicaciones

Luego de detallar las fases de construcción del vehículo se procederá a explicar el análisis de alternativas y desarrollo del servidor y de las aplicaciones del usuario y de administración. Esto además contempla la explicación de los algoritmos desarrollados para el cálculo de las instrucciones que se le deben enviar al vehículo, y los algoritmos encargados de la gestión de los pedidos.

4.2.2.1 Alternativas de solución

Para el desarrollo del servidor se seleccionará un framework de desarrollo web que se ajuste a los requerimientos anteriormente mencionados. Este framework debe tener como principales características la abstracción de los aspectos más básicos del desarrollo web, con el fin de centrarse en desarrollar las funcionalidades fundamentales del proyecto sin preocuparse demasiado por preparar entornos de desarrollo o construir desde cero componentes comunes del desarrollo web. Es deseado también que se abstraiga la creación de bases de datos, por lo que se tendrá gran consideración si el framework incluye un sistema ORM con el que no se tengan que hacer consultas SQL y se abstraiga la migración de las tablas de la base de datos. Otra característica fundamental para cumplir con los requerimientos del sistema es que el framework soporte la comunicación en tiempo real y bidireccional con el cliente, esta característica permite que el servidor le envíe instrucciones al vehículo de forma proactiva sin necesidad de utilizar técnicas como long polling que podrían saturar la red o no responder a tiempo. Además, utilizar comunicación bidireccional permite notificar al usuario sobre eventos que puedan ocurrir como la desconexión, posición actual o límites de tiempo para ocupar el vehículo, la tecnología de preferencia para este propósito es WebSockets. El framework debe ser multiplataforma y sin licencias de pago.

Las alternativas consideradas fueron las siguientes:

- **Flask**: Flask es un micro framework web multiplataforma desarrollado en Python que se caracteriza por su simplicidad, especial para proyectos pequeños donde las consultas al servidor son sencillas, no cuenta con más funcionalidades que las básicas de un framework web y se pueden utilizar ORM instalando módulos extras. Flask no está diseñado para la comunicación en tiempo real (Flask, s.f.).

- **Django**: Django es un framework web multiplataforma desarrollado en Python y es especial para desarrollar proyectos completos en poco tiempo debido a que trae incluidos componentes que realizan las tareas más comunes del desarrollo web, como un administrador de contenido 100% configurable, un ORM robusto que se puede utilizar con los motores de bases de datos más comunes, middlewares que se encargan de proteger las aplicaciones de las vulnerabilidades más comunes, entre muchas otras características (Django, s.f.). Django recientemente puso a disposición el módulo channels, utilizado para la comunicación en tiempo real, esta librería requiere de un servidor de colas de mensajes externo como Redis (Django Channels, s.f.).

- **Express**: Express en un framework web multiplataforma desarrollado en Javascript sobre NodeJs que permite realizar aplicaciones web sencillas, ofrece las funcionalidades básicas del desarrollo web, se integra fácilmente con las bases de datos más comunes, pero no tiene un ORM incluido (Express, s.f.). Se puede integrar fácilmente con la biblioteca Socket.IO que permite la comunicación en tiempo real y bidireccional con el cliente (SocketIo, s.f.).

- **Sails**: Sails es un framework web multiplataforma desarrollado en Javascript sobre NodeJs que permite realizar aplicaciones web completas

en poco tiempo, ya que posee funcionalidades incluidas que apoyan el desarrollo de aplicaciones web modernas, en las que destacan la creación automática de una API REST que se ajusta a los modelos creados en su potente ORM. El ORM incluido ofrece la abstracción completa de la base de datos, permitiendo utilizar ya sea bases de datos relacionales como no relacionales. Tiene integración con la tecnología WebSockets por defecto y automáticamente emite eventos cuando se modifica la base de datos, permitiendo que sin mayores configuraciones se logran aplicaciones de tiempo real en poco tiempo (Sails, s.f.).

Luego de haberse analizado las opciones consideradas desde el punto de vista de las necesidades del proyecto y las características deseadas para un desarrollo rápido, se puede realizar la siguiente matriz de alternativas con la que se tomará una decisión objetiva. Cada característica será ponderada de acuerdo a su importancia en la decisión y se evaluará cada alternativa con una nota de un 1 a un 4 siendo, muy malo, malo, bueno y muy bueno respectivamente.

	Rapidez de desarrollo	ORM	WebSockets	Total
Importancia	25%	25%	50%	100%
Flask	2	1	1	1.25
Django	4	4	1	1.5
Express	2	2	3	1.6
Sails	4	4	4	4

Tabla 5 - Matriz de decisión alternativas servidor

En la tabla se ve claramente cómo es que Sails es el que cumple de mejor manera según las características consideradas. Por lo tanto, se utilizará en el desarrollo del servidor.

4.2.2.2 Algoritmo gestión de pedidos

El servidor debe poder gestionar los pedidos de forma automática, por lo tanto, existirá un módulo que registrará los pedidos realizados por los usuarios y se encargará de enviar los vehículos en el orden en el que fueron llamados. También controlará el tiempo máximo que un vehículo puede estar ocupado por un usuario y deberá enviar las instrucciones correspondientes al vehículo, tanto para ir al destino del pedido siguiente o para regresar a la estación base del vehículo cuando este haya realizado todos los pedidos.

Cuando el usuario realiza un pedido, el gestor de peticiones lo agrega a una cola correspondiente al vehículo que haya sido asignado al pedido, con el fin de ser ejecutadas en el mismo orden en el que se pidieron, esta cola se compone de pedidos como se muestra en la Figura 28, el primer pedido de cada cola es el pedido activo, es decir, el pedido que está ejecutando el vehículo.

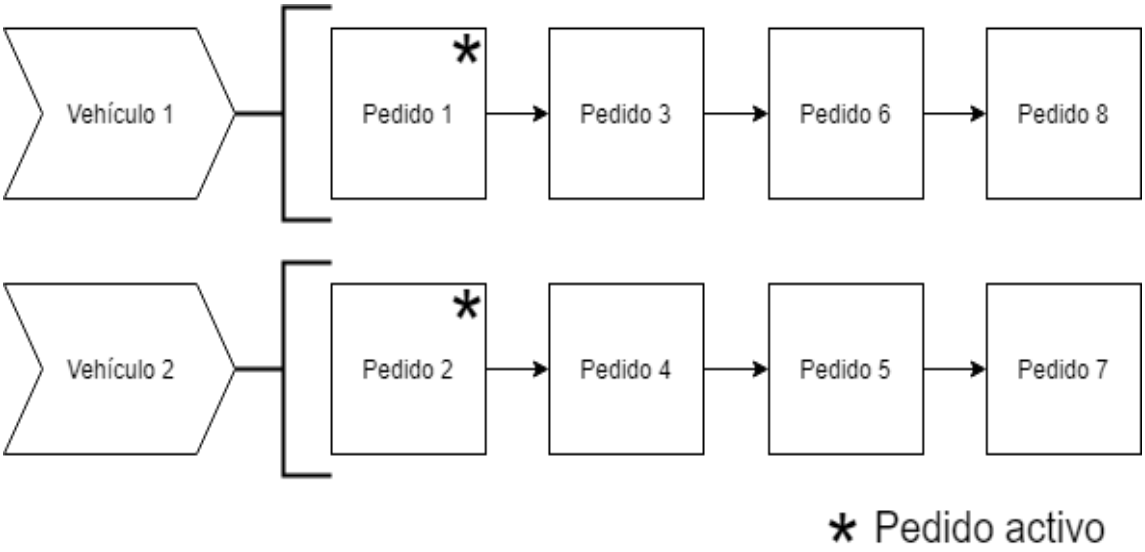

Figura 28 - Diagrama cola de pedidos por vehículo

Cada pedido contiene la información necesaria para poder ejecutarse cuando sea su turno, los atributos de cada pedido son:

- **Destino**: Se guarda el código del nodo RFID al que se debe dirigir el vehículo.
- **Ruta**: Se guarda la ruta calculada que debe seguir el vehículo en el grafo para llegar al destino, la ruta es calculada desde el nodo en el que se encontraba el vehículo en el momento en el que el pedido pasa al primer lugar de la cola, solo el pedido activo tiene calculada la ruta.
- **Req**: Guarda el socket de conexión del usuario que realizó el pedido para poder enviarle mensajes posteriormente.
- **Cancelado**: En el caso que se cancele el pedido, este valor se hace verdadero, luego es comprobado cada vez que el vehículo notifique que llegó a un nodo, en caso de que sea verdadero en el momento en el que el vehículo está sobre un nodo, entonces se desencola el primer pedido, y se ejecuta el siguiente.
- **EnDestino**: Indica que el vehículo llegó a su destino, este atributo es ocupado principalmente para hacer correr el tiempo máximo que el vehículo puede ser ocupado por un usuario.
- **GoHome**: Este atributo es verdadero cuando el pedido es de tipo “GoHome”, este es un pedido especial que es encolado por el sistema cuando no existen pedidos de un usuario pendiente, que tiene siempre por destino la estación base del vehículo. Si el pedido activo es de tipo “GoHome”, cuando se agrega un nuevo pedido realizado por el usuario, automáticamente el pedido “GoHome” se cancela.

Una vez que el vehículo llegó al destino, el usuario debe poder liberar el vehículo, es decir, finalizar el pedido manualmente para indicar que el usuario terminó de utilizar el vehículo y poder ejecutar los siguientes pedidos de la cola. Si existen más pedidos pendientes el usuario tendrá un tiempo máximo de uso del vehículo, al cumplirse este tiempo el vehículo se liberará automáticamente. El tiempo máximo de uso es configurado por el administrador, y sólo se considerará este tiempo cuando existan más pedidos en la cola. En el caso que exista un usuario ocupando un vehículo, y otro pedido se agregue a la cola, el gestor de pedidos notificará al usuario que existe un tiempo máximo de uso del vehículo.

Si el usuario cancela un pedido, ya sea voluntariamente a través de la aplicación o por su desconexión, el gestor de pedidos deberá simplemente eliminar el pedido de la cola, en caso de que el pedido cancelado no esté activo y si el pedido está siendo ejecutado, el gestor de pedidos deberá cancelarlo poniendo el atributo cancelado del pedido en verdadero, para luego cuando el vehículo llegue al siguiente nodo, se proceda a efectuar el siguiente pedido en la cola. En caso de que el vehículo se desconecte el gestor de pedidos elimina todos los pedidos de la cola del vehículo desconectado, además notifica a todos los usuarios a los que se le hayan asignado el vehículo desconectado, que su pedido no podrá ser realizado.

Cuando hayan finalizado todos los pedidos de los usuarios, el gestor de pedidos agrega un pedido especial llamado “GoHome”, que llevará el vehículo desde su última posición a una estación del vehículo previamente configurada. Los pedidos “GoHome” se diferencian de los pedidos normales, ya que son cancelados automáticamente por el gestor de pedidos en caso de que el usuario agregue un pedido nuevo a la cola.

Una función importante del gestor de pedidos es asegurar que sólo un vehículo ocupe un nodo al mismo tiempo, claramente si esto no ocurriera, se produciría una colisión. Para resolver este problema cada vez que se realiza un pedido, el

gestor de pedidos, agrega todos los nodos de su ruta a una estructura de datos de tipo clave-valor, donde las claves principales corresponden a cada nodo y el valor es una lista de bloqueos. Cada bloqueo se conforma de un atributo vehículo, que identifica el vehículo que realizó el bloqueo del nodo y un atributo que indica el tiempo en el que fue realizado el bloqueo. Cuando se da el caso que dos vehículos tienen bloqueado un nodo, se compara el tiempo en el que fue realizado el bloqueo, el primero en haber realizado el bloqueo tiene la preferencia de ocupar ese nodo.

Cada vez que la lista de nodos bloqueados es actualizada, esta se les envía a todos los vehículos conectados, estos, deben detenerse cada vez que el siguiente nodo en su ruta pertenezca al conjunto de nodos bloqueados, y reanudar su marcha cuando esta situación deje de cumplirse. Cuando el vehículo responsable del bloqueo de un nodo notifique que llegó a esa posición, el nodo se desbloqueará para los demás vehículos que desean dirigirse a ese nodo (Figura 29), si el pedido se cancela se eliminan todos los bloqueos de ese vehículo. Además, deberá enviarse al algoritmo de cálculo de instrucciones (Ver 4.2.2.3 Algoritmo cálculo de instrucciones), un conjunto de nodos correspondientes a los destinos de los pedidos activos, con el fin de no considerar estos puntos en el cálculo de la ruta y preferir otro trayecto.

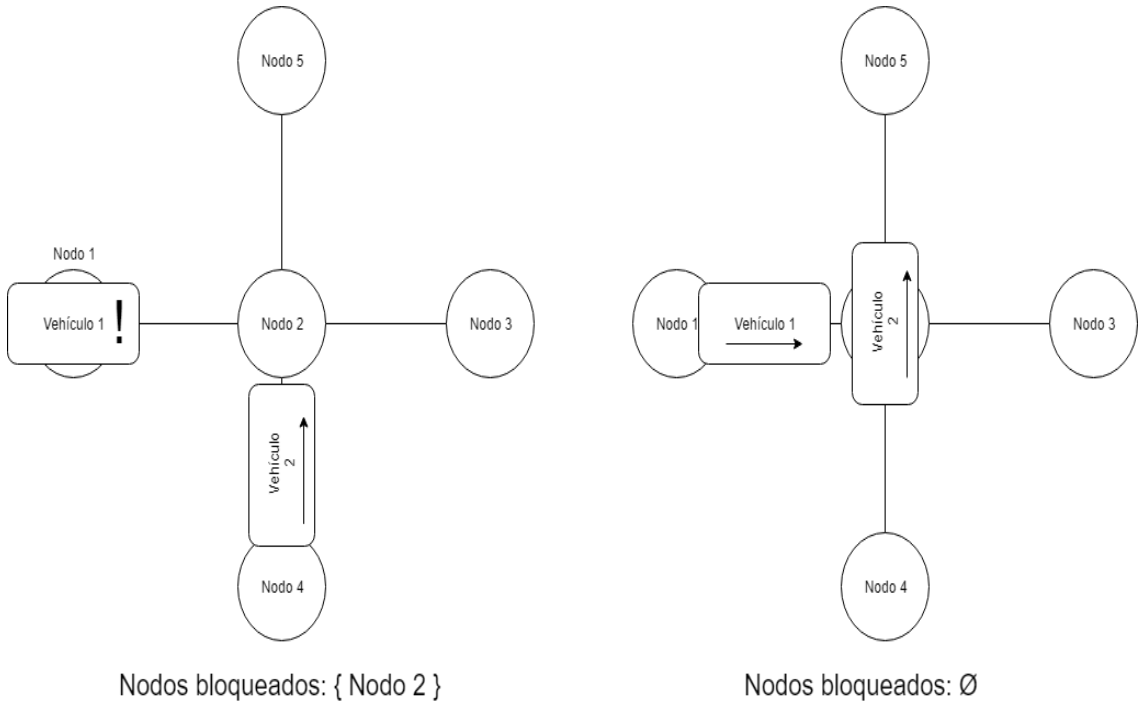

Figura 29 - Diagrama nodos bloqueados

El gestor de pedido también es responsable de comunicar eventos a los distintos actores del sistema, esta comunicación debe ser en tiempo real y bidireccional, por lo que se utiliza el protocolo WebSockets que cumple con este propósito. En este sistema se utiliza el concepto de “Salas”, a las que los clientes se pueden suscribir a la espera de eventos, luego se puede enviar de forma proactiva mensajes a todos o algunos de los miembros de la sala. Sails por defecto crea salas automáticamente, para cada uno de sus modelos, así como también para cada conexión de los clientes, también es posible crear salas personalizadas para darles el uso que se estime conveniente. A continuación, se detallan las salas utilizadas por el gestor de pedidos y sus funciones principales:

- **Sala del vehículo**: Al conectarse un vehículo, este debe ser unido a una sala propia. Cuando el usuario realice un pedido, y este esté activo, debe ser unido a la sala del vehículo. Las principales funciones de esta sala son las de notificar al usuario cuando el vehículo se dirige a su destino, la posición actual del vehículo, si el vehículo llegó a su destino, el tiempo máximo asignado al usuario para ocupar el vehículo y notificar al vehículo si el pedido fue cancelado. Siempre que un pedido finalice, el usuario debe ser expulsado de la sala.

- **Sala de espera**: Cuando un usuario realiza un pedido, es unido a una sala de espera con los demás usuarios que hayan pedido el mismo vehículo, existe una sala de espera por vehículo y su principal función es notificar a los usuarios si un vehículo se desconectó y no puede ejecutar sus pedidos. Cuando haya finalizado el pedido de un usuario, este debe ser expulsado de la sala de espera del vehículo.

- **Sala de vehículos**: Esta es una única sala a la que se unen todos los vehículos conectados, la función de esta sala es notificar a todos los vehículos los nodos bloqueados, cada vez que sea necesario.

4.2.2.3 Algoritmo cálculo de instrucciones

Una parte fundamental del sistema es calcular las instrucciones que se deben enviar al vehículo para que éste las ejecute cuando llegue a un nodo. A pesar de que calcular la ruta que se debe seguir en un grafo es un problema clásico de las matemáticas y ampliamente implementado en la informática, en este sistema se deben calcular los movimientos que debe realizar el vehículo, considerando que este no tiene forma de orientarse globalmente. Por lo tanto, se debe contemplar que todos los movimientos se harán considerando la orientación local calculada de la resta de los vectores posición de los últimos dos nodos que notificó el vehículo. Los movimientos posibles del vehículo son: avanzar, girar a la derecha o izquierda, dar una vuelta en 180°, o detenerse.

Para el cálculo de la ruta, se utilizará el algoritmo de Dijkstra, que es un algoritmo para la búsqueda de los caminos más cortos de un grafo. Este algoritmo necesita el grafo en el que se quiere calcular la ruta, con los respectivos costos entre los nodos, y el origen y el destino de la ruta que se quiere calcular. El algoritmo entrega como resultado una lista de nodos en el orden en el que deben ser recorridos.

Con el fin de evitar colisiones entre vehículos, se deben descartar todos los nodos que sean parte de los destinos de los pedidos activos del cálculo de la ruta, para esto simplemente se debe tener un conjunto de nodos no considerados, que deben ser eliminados del grafo, junto a sus conexiones, esto permitirá que se tome una ruta alternativa (Figura 30).

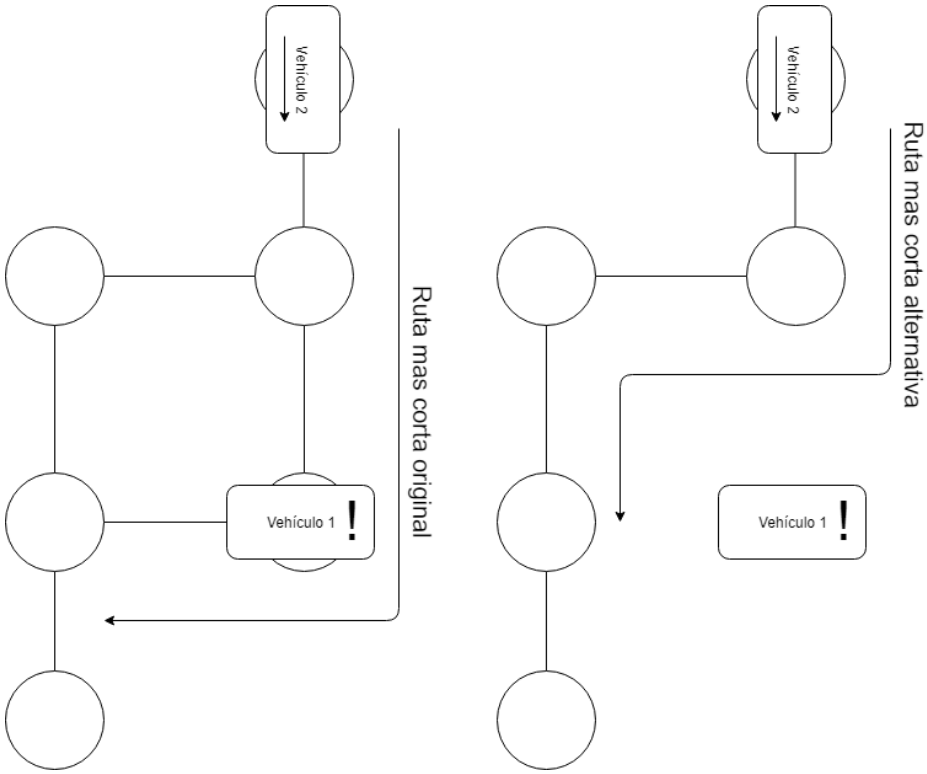

Figura 30 - Diagrama rutas alternativas

Una vez obtenida la ruta que se debe seguir, se deben calcular las instrucciones que serán enviadas al vehículo. Para esto se debe utilizar un sistema de coordenadas, en el que cada nodo del grafo tenga un vector posición. Para el cálculo de las instrucciones no es relevante que la distancia entre los nodos del sistema de coordenadas se ajuste a la realidad, pero si su orientación con respecto a sus nodos adyacentes. Igualmente se debe considerar que cada nodo sólo puede tener 4 nodos adyacentes y el ángulo formado entre ellos debe ser idealmente de 90°.

Antes de comenzar el cálculo de las instrucciones, es importante obtener la orientación del vehículo para luego ocuparla como referencia al calcular las instrucciones. Para esto se deben obtener las dos últimas posiciones del vehículo y restarlas, posteriormente se debe calcular la orientación del vehículo comprobando en qué sector se encuentra el vector restado. Los sectores se pueden ver en la Figura 31.

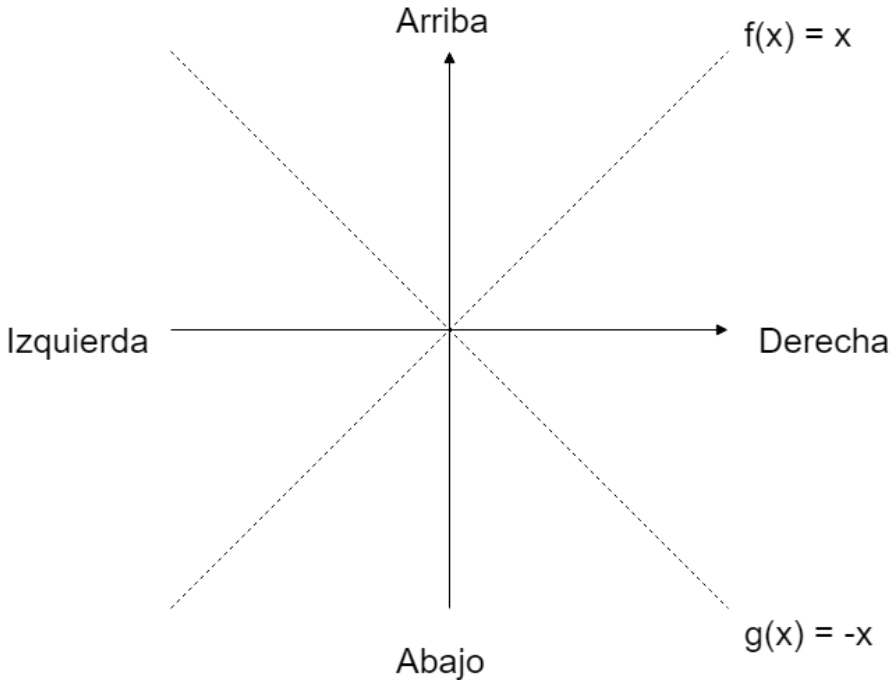

Figura 31 - Sistema de referencia para el cálculo de instrucciones

Los distintos sectores que representan a una determinada orientación se definen según las siguientes condiciones:

- **<u>Arriba</u>**: $(x \geq 0 \wedge y > x) \vee (x < 0 \wedge y > -x)$
- **<u>Abajo</u>**: $(x \geq 0 \wedge y < -x) \vee (x < 0 \wedge y < x)$
- **<u>Derecha</u>**: $(x > 0 \wedge y < x) \wedge (x > 0 \wedge y > -x)$
- **<u>Izquierda</u>**: $(x \leq 0 \wedge y < -x) \wedge (x \leq 0 \wedge y > x)$

Una vez calculada la orientación del vehículo, se itera sobre la ruta obtenida para conseguir las instrucciones correspondientes. Para esto se debe obtener el vector posición tanto del nodo actual de la iteración, como del siguiente, con el fin de obtener la instrucción que corresponde. Se deben restar los vectores de posición de ambos nodos y luego calcular la orientación del nodo siguiente con respecto al nodo actual de la iteración, el cálculo de orientación se hace como se mostró anteriormente en el cálculo de la orientación del vehículo. Luego se compara la orientación del nodo siguiente con la orientación calculada del vehículo, obtener la instrucción que debe ejecutar el vehículo para llegar al siguiente nodo.

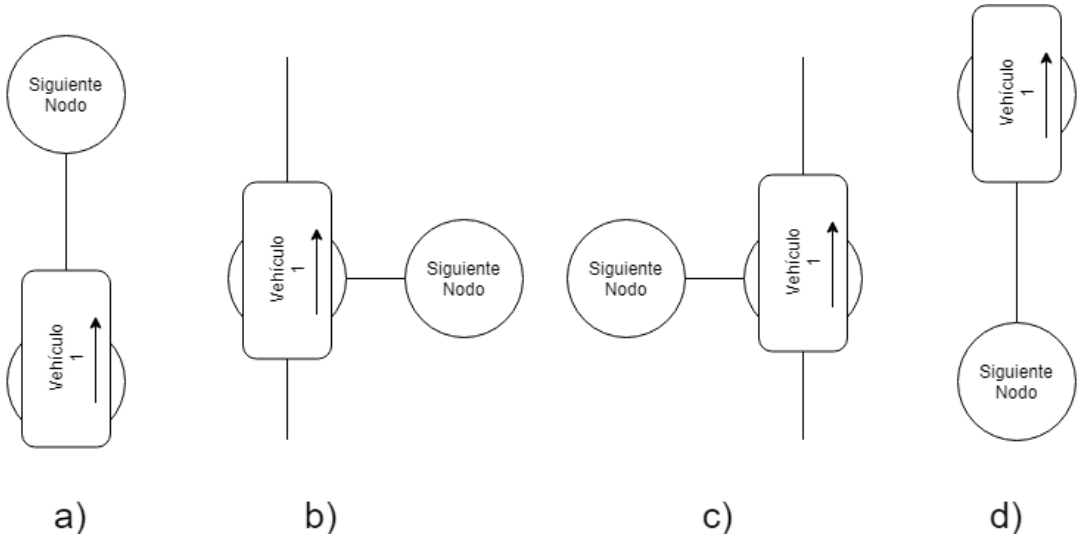

Figura 32 - Diagrama ejemplos de giro

Como se ve en la Figura 32 existen cuatro casos básicos, en el caso a) se debe obtener una instrucción de seguir adelante, en el caso b) el vehículo debe girar a la derecha, en el caso c) el vehículo debe girar a la izquierda, y en el caso d) el vehículo debe dar una vuelta de 180°.

Debido a cómo se encuentra diseñado el sistema de seguimiento de líneas, se debe prestar especial atención a las vueltas en 180°. La función que permite girar el vehículo (ver 4.2.1.3 Programación del vehículo), lo hace primeramente girando hasta dejar de estar sobre la línea y luego seguir girando a hasta encontrarla nuevamente, por lo tanto, si se quiere girar en 180° en un caso como el mostrado en la Figura 32 d) solo basta con realizar un giro a la izquierda o derecha (Por convención en este sistema los giros en 180° siempre se harán a hacia la derecha), pero en el caso en el que el nodo actual tenga adyacente a él otro nodo en la dirección de giro (Figura 33), se deberán hacer dos giros de 90° para cumplir la instrucción.

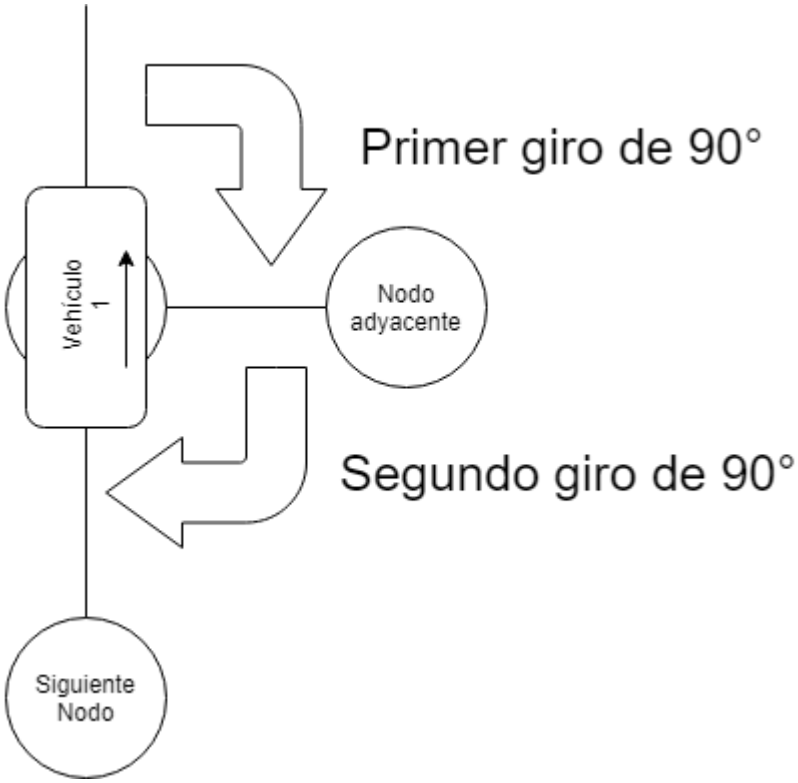

Figura 33 - Ejemplo giro doble

Esta situación requiere que se calcule una instrucción especial que indique que se deben realizar dos giros consecutivos. Para obtener esta información cada vez que la instrucción calculada sea girar atrás, se debe comprobar si existe un nodo adyacente situado en la orientación de la dirección del giro.

Las instrucciones se codifican en forma de números que las representan, el vehículo está preparado para recibir estas instrucciones y ejecutarlas. Las instrucciones y su respectivo código tanto en decimal como en hexadecimal se muestran en la siguiente tabla.

Instrucción	Código Decimal	Código Hexadecimal
Seguir derecho	17	0x11
Girar Derecha	19	0x13
Girar Izquierda	20	0x14
Detenerse	21	0x21
Girar Atrás (Un giro)	22	0x16
Girar Atrás (Dos giros)	23	0x17

Tabla 6 - Instrucciones

Las instrucciones se envían en formato JSON, con una estructura clave-valor, donde la clave es el nodo y el valor la instrucción que debe ser ejecutada. Se envían también claves útiles para la ejecución del pedido, como la clave “Primera” que le indica al vehículo la primera instrucción a ejecutar sin necesidad de pasar sobre un nodo. Además, se envían los nodos posteriores al nodo actual, esto es útil para decidir si detenerse en caso de que el nodo siguiente se encuentre bloqueado. Se puede visualizar un ejemplo de las instrucciones enviadas a un vehículo a continuación.

```
{
    "primera": {
        "instruccion": 22
    },
    "8804D28E": {
        "instruccion": 22,
        "siguiente": "8804E68E"
    },
    "8804E68E": {
        "instruccion": 20,
        "siguiente": "88049E91"
    },
    "88049E91": {
        "instruccion": 19,
        "siguiente": "88042F2F"
    },
    "88042F2F": {
        "instruccion": 17,
        "siguiente": "8804DC8E"
    },
    "8804DC8E": {
        "instruccion": 19,
        "siguiente": "8804CA90"
    },
    "8804CA90": {
        "instruccion": 21
    }
}
```


Figura 34 - Ejemplo de instrucciones en formato JSON

4.2.2.4 Programación del servidor

La programación del servidor como se comentó anteriormente (ver 4.2.2.1 Alternativas de solución) se realizó en el framework Sails, donde principalmente se especifican los modelos que el ORM debe crear en la base de datos y los controladores que se encargaran de responder a los clientes cuando estos hagan una petición a una determinada URL. Por lo tanto, a continuación, se detallarán los modelos creados, la programación de los controladores y las urls que fueron configuradas para interactuar con los clientes.

En primer lugar, se programaron los modelos utilizando el sencillo ORM que Sails ofrece para especificar cómo se debe crear la base de datos. En el siguiente modelo relacional se especifican las entidades de la base de datos, sus atributos y cómo se relacionan entre ellas.

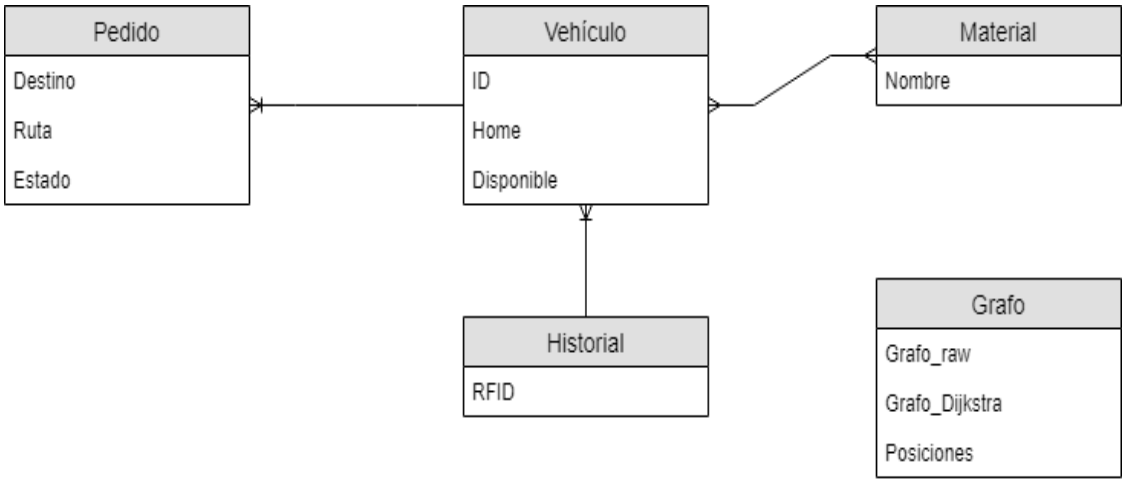

Figura 35 - Modelo relacional de la base de datos

En el modelo anterior se muestran las entidades que definen el sistema y sus atributos, como se ve, es un modelo bastante sencillo ya que solo se consideraron entidades y atributos mínimos para el funcionamiento del sistema, por ejemplo, no se considera la creación de un sistema de inventarios robusto o información que podría ser crítica para la toma de decisiones en una organización que ocupe este sistema, con el fin de centrarse solo en el funcionamiento principal.

En la siguiente lista, se explican los atributos de cada entidad:

1) **Vehículo**:

    a) **ID**: Este atributo es responsable de identificar únicamente un vehículo, las peticiones del vehículo hacia el servidor deben contener el ID correspondiente.

    b) **Home**: Indica el nodo que actúa como estación base, este el punto al que se dirige el vehículo cuando no tiene más pedidos pendientes por atender.

    c) **Disponible**: Indica si el vehículo se encuentra disponible para recibir pedidos.

2) **Pedido**:

   a) **Destino**: Es el nodo al que se debe dirigir el vehículo al ejecutar el pedido.

   b) **Ruta**: Es la ruta calculada para el pedido, los pedidos cancelados pueden no tener una ruta calculada.

   c) **Estado**: El estado en el que terminó el pedido, puede ser: Finalizado o cancelado.

3) **Material**:

   a) **Nombre**: Solo se indica el nombre del material.

4) **Historial**:

   a) **RFID**: Guarda los nodos por los que pasó el vehículo.

5) **Grafo**:

   a) **Grafo_Raw**: Guarda el grafo en forma de string y en formato JSON directamente como se guarda en la aplicación de administración (ver 4.2.2.5 Programación de aplicaciones).

   b) **Grafo_Dijkstra**: Se guarda el grafo en el formato que ocupa la librería responsable de calcular las rutas.

   c) **Posiciones**: Se guardan las coordenadas de los nodos, estas coordenadas son las que se configuraron en la aplicación de administración.

Es importante mencionar que Sails incluye automáticamente atributos a los modelos, estos atributos son comunes en el diseño de bases de datos, y son aplicados de forma automática a los modelos escritos. Los atributos mencionados son los siguientes:

- **createdAt**: Indica la fecha y hora en la que fue creado el registro.
- **updatedAt**: Muestra la fecha y hora en la que el registro fue modificado.
- **id**: En el caso de no especificarlo manualmente Sails crea automáticamente un atributo Id de tipo entero.

Los controladores en Sails son los encargados de responder a las peticiones HTTP de los clientes, estos controladores son funciones que son llamadas cuando el cliente accede a una determinada URL, estas funciones reciben un parámetro “req” encargado de almacenar la petición del cliente, incluyendo principalmente parámetros HTTP, entre otros; y un parámetro “res” encargado de ser el responsable de enviarle la respuesta al cliente. Para este sistema los controladores se separaron en dos módulos distintos: GrafoController, encargado de responder a las operaciones realizadas sobre el grafo y VehiculoController, responsable de responder a todos los eventos de pedidos y vehículos, los controladores se explican en más detalle a continuación:

1) **GrafoController**:
    a) **Crear**: Este controlador se llama al crear o modificar el grafo, es el encargado de asegurarse que solo exista un registro de grafo, además procesa la información para crear los parámetros Grafo_Dijkstra y Posiciones. Calcular estos atributos al crear o modificar el grafo y almacenarlos en la base de datos permiten ahorrar tiempo de procesamiento cada vez que se realiza un pedido.

2) **VehiculoController**:

a) **Conectar**: Este controlador es llamado cuando el vehículo se enciende, con el fin de notificar al servidor que se encuentra disponible y listo para recibir pedidos. Además, se une al vehículo a las salas correspondientes y se notifica a todos los usuarios conectados que se conectó un vehículo nuevo. Para que un vehículo pueda conectarse correctamente previamente debe estar registrado en el administrador.

b) **Llamar**: El controlador llamar es el encargado de recibir los pedidos por parte del usuario. Busca vehículos disponibles que cumplan con las condiciones de material si es que las hubiese y selecciona el mejor para resolver el pedido. Aquí se agrega el pedido al módulo gestor de pedidos, para que este cumpla su función correspondiente (ver 4.2.2.2 Algoritmo gestión de pedidos).

c) **Notificar**: Aquí se recibe la notificación del vehículo cada vez que pasa sobre un nodo, se comprueba que el nodo esté registrado, se crea un registro en el historial del vehículo y se llama a la función “notificarPosición” del gestor de pedidos para que cumpla su función (ver 4.2.2.2 Algoritmo gestión de pedidos).

d) **Terminar**: Este es el controlador encargado de responder a la cancelación de un pedido por parte del usuario, crea un registro de pedido en la base de datos con el atributo de estado en “Cancelado” y llama a la función “cancelarPedido” del gestor de pedidos, para eliminar el pedido de la cola correspondiente.

e) **Liberar**: El controlador liberar, es el encargado de finalizar el pedido, creando un registro pedido con el atributo estado en “Finalizado”, además de notificar al gestor de pedidos que se debe liberar un pedido.

Las URLs que acceden a cada controlador con detalladas en la siguiente tabla:

Verbo HTTP	URL	Controlador
POST	/vehiculo/llamar	VehiculoController.llamar
POST	/vehiculo/conectar	VehiculoController.conectar
POST	/vehiculo/notificar	VehiculoController.notificar
POST	/vehiculo/terminar	VehiculoController.terminar
POST	/vehiculo/liberar	VehiculoController.liberar
POST	/grafo	GrafoController.crear
PATCH	/grafo	GrafoController.crear

Tabla 7 - Rutas del servidor

Sails además trae por defecto una serie de controladores internos que permiten realizar las funciones más comunes de una aplicación web, esta característica se llama Blueprint API (Sails, s.f.) y permiten la obtención, modificación, creación y eliminación de los modelos creados.

En el servidor se incluyen los módulos “gestor-peticiones” y “calculador-rutas” que realizan las funciones de los algoritmos gestión de pedidos (ver 4.2.2.2 Algoritmo gestión de pedidos) y cálculo de instrucciones (ver 4.2.2.3 Algoritmo cálculo de instrucciones) respectivamente.

4.2.2.5 Programación de aplicaciones

Para la interacción con este sistema, se crearon dos aplicaciones orientada a los usuarios, una de ellas es la aplicación cliente, encargada de realizar los pedidos, cancelarlos, y finalizarlos; y la aplicación administración, responsable de la configuración del grafo, el registro de los vehículos y los materiales. Estas aplicaciones fueron desarrolladas con HTML, CSS y Javascript, con el fin de poder utilizarlas en cualquier dispositivo que soporte la visualización de sitios web.

Las aplicaciones utilizan una librería llamada Vis.js (Vis.js, s.f.) encargada de mostrar y administrar el grafo de forma sencilla. El grafo creado puede ser exportado en formato JSON y de esta forma guardarlo en la base de datos en forma de string.

Para realizar un pedido en la aplicación cliente, simplemente se debe seleccionar el material deseado en la lista de materiales, en caso de que se necesite el vehículo sin un material específico se debe seleccionar “Ninguno” en la lista de materiales, luego se debe seleccionar el nodo correspondiente a la estación de trabajo a la que se quiere realizar el pedido, y presionar el botón “Pedir vehículo” (Figura 36). La estación seleccionada se marcará en verde y la posición actual del vehículo se marcará en amarillo. En este momento es posible cancelar el pedido presionando el botón “Cancelar” (Figura 37). Cuando el vehículo llegue al destino aparecerá una notificación y el botón se pintará de verde, para desocupar el vehículo simplemente se debe presionar el botón “Desocupar” (Figura 38).

En caso de que el vehículo asignado esté ejecutando otro pedido en ese momento, se mostrará un mensaje que indicará la posición de la cola y se notificará cuando sea el turno de atender el pedido, en este caso se mostrará el botón de cancelar. Si al llegar el vehículo a su destino, existen más vehículos en la cola de espera, se mostrará un contador en el botón que indicará el tiempo

restante para utilizar el vehículo, cuando el contador llegue a cero, el vehículo se desocupará automáticamente y la aplicación volverá al estado de espera.

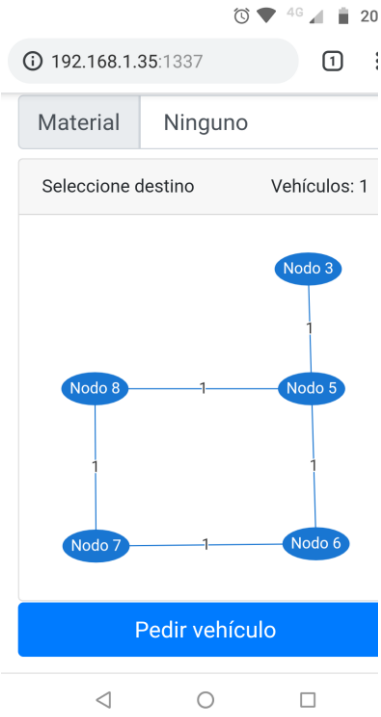

Figura 36 - Pantalla pedir vehículo

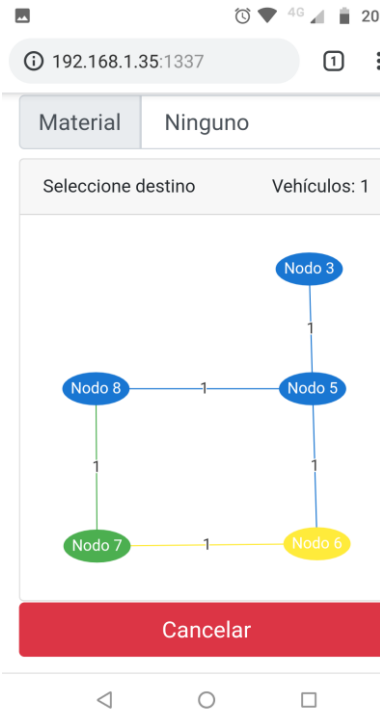

Figura 37 - Pantalla cancelar pedido

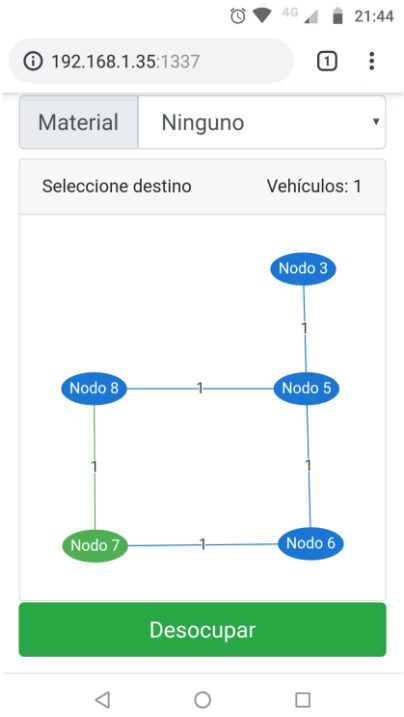

Figura 38 - Pantalla desocupar vehículo

La aplicación de administración cumple la función de configurar el grafo que le será mostrado al usuario, registrar vehículos y sus atributos, asignar materiales y visualizar los pedidos y el historial de cada vehículo. Para esta aplicación se ocupa la misma librería que en la aplicación cliente para mostrar el grafo, pero esta vez con parámetros que permiten editar el grafo de una forma sencilla. En la Figura 39 se pueden apreciar los botones de “Añadir nodo” y “Añadir arista”, con los cuales se podrán realizar dichas acciones. Cuando se añade un nodo, se pregunta por el RFID asignado a ese nodo y cuando se unen dos nodos con una arista se pregunta por el costo de la ruta, es conveniente poner como costo de ruta la distancia real que separa los nodos, sin importar la unidad de medida. Además, se ven otros controles de interacción que hacen más fácil la edición del grafo. Una vez que se hayan realizado los cambios deseados en el grafo,

presionando el botón “Guardar”, la aplicación enviará el grafo al servidor y será cargado en las aplicaciones de los clientes.

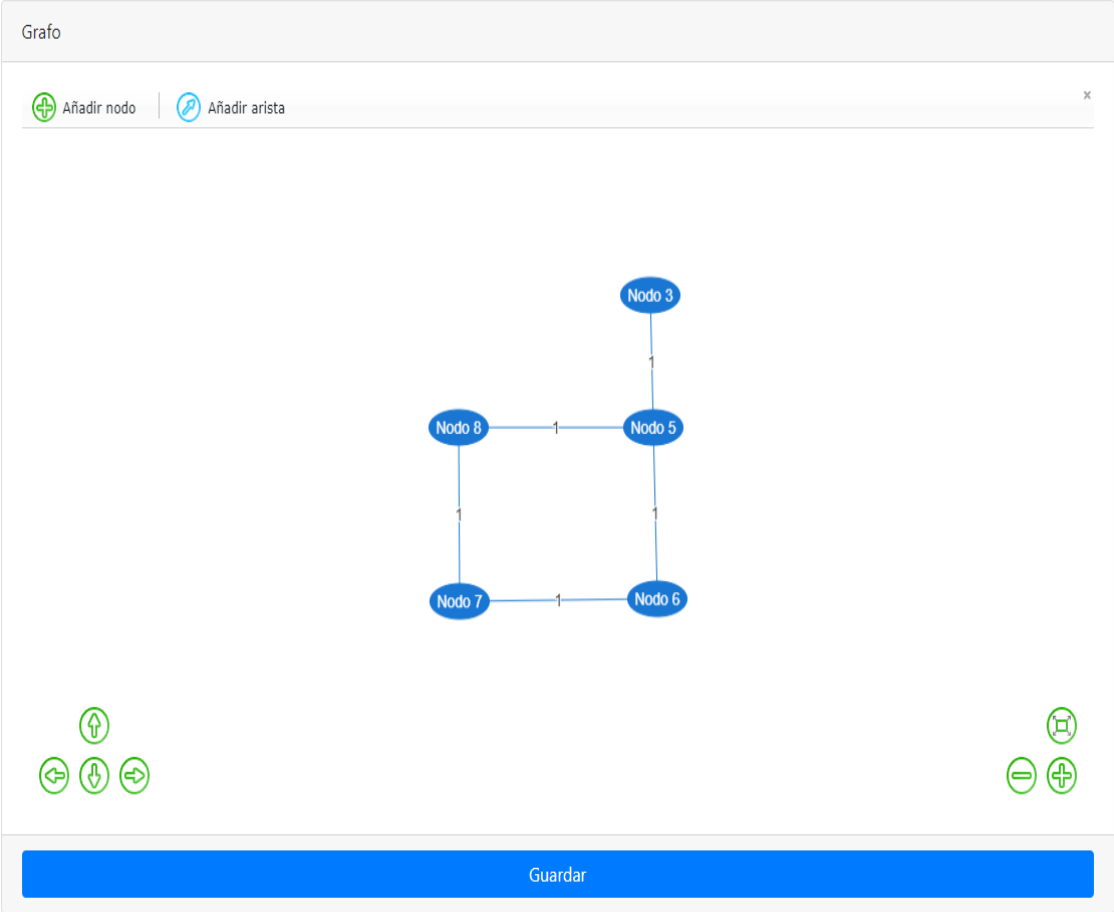

Figura 39 - Pantalla administración de grafo

En la parte inferior de la aplicación se encuentra la administración de los vehículos. En el lado derecho se enlistan los vehículos registrados, con la opción de eliminarlos o seleccionarlos para mostrar el detalle. Para agregar un vehículo se debe presionar el botón “Agregar Vehículo” y se pedirá el ID del vehículo que se está agregando. Al seleccionar un vehículo, el sector izquierdo de la pantalla muestra sus datos correspondientes, aquí se puede configurar el “Home” del vehículo ingresando el RFID correspondiente y se puede modificar también el atributo disponible. Las listas de la derecha muestran información sobre los pedidos, el historial y los materiales de cada vehículo.

En la pestaña pedidos (Figura 40), se enlistan los pedidos que se han asignado al vehículo seleccionado, aquí se pueden apreciar los principales atributos con la fecha en el que fue realizado, el destino al que fue enviado, el estado final del vehículo y al poner el cursor en la palabra “Ruta” se despliega la ruta que fue calculada en cada caso.

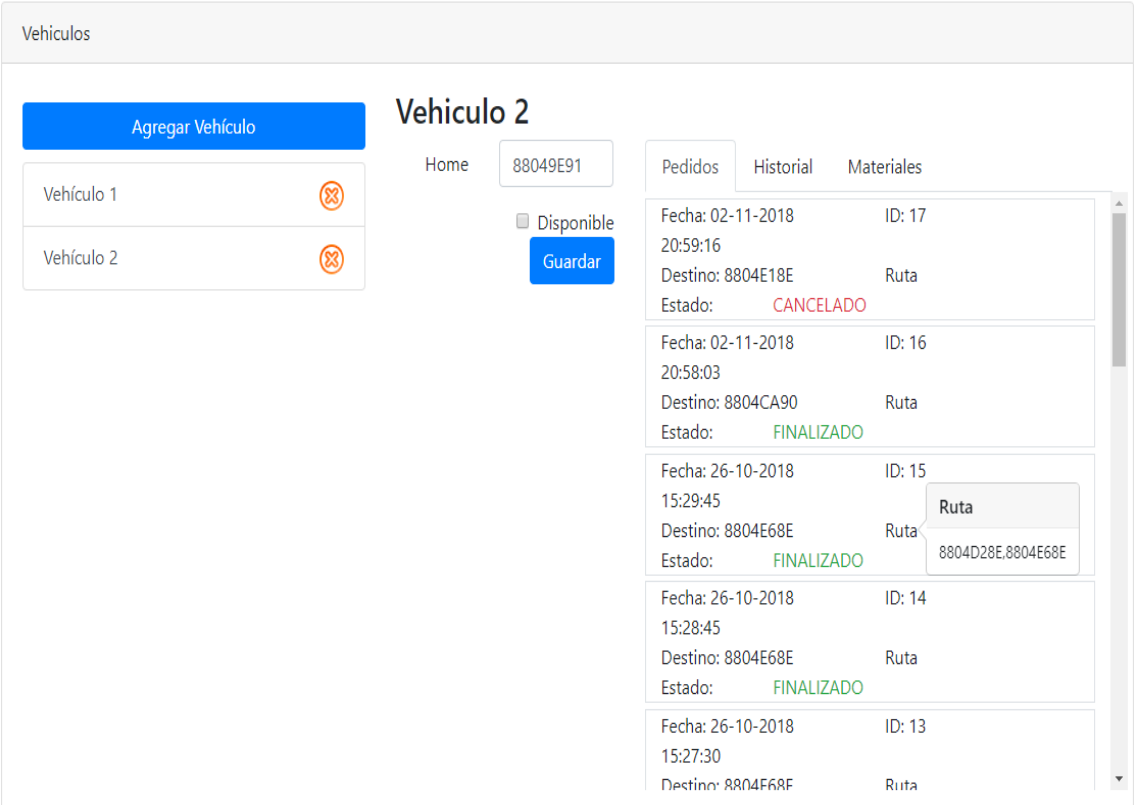

Figura 40 - Pantalla de administración de vehículos

En la pestaña historial (Figura 41) se muestran la lista de nodos por los que ha pasado el vehículo, cada elemento de esta lista se puede eliminar para dar mayor flexibilidad y poder solucionar errores en el cálculo de las instrucciones que puedan ser provocados por historiales defectuosos.

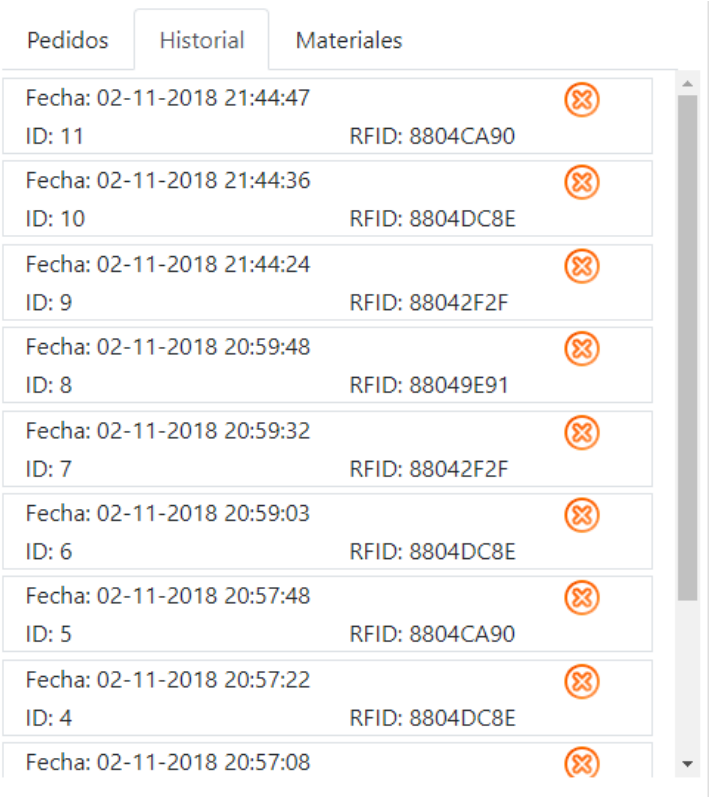

Figura 41 - Pestaña administración de Historial

En la pestaña de Materiales (Figura 42) se enlistan todos los materiales registrados en la aplicación, se muestra el nombre del material y un checkbox que indica si el material está asignado al vehículo seleccionado. También se muestran dos botones para agregar un material, el botón “Agregar sin asignar” que agrega el material a la lista de materiales, pero no lo asigna automáticamente al vehículo; y el botón “Agregar” que automáticamente asigna el material ingresado al vehículo seleccionado. Es posible eliminar un material presionando en el icono “X”

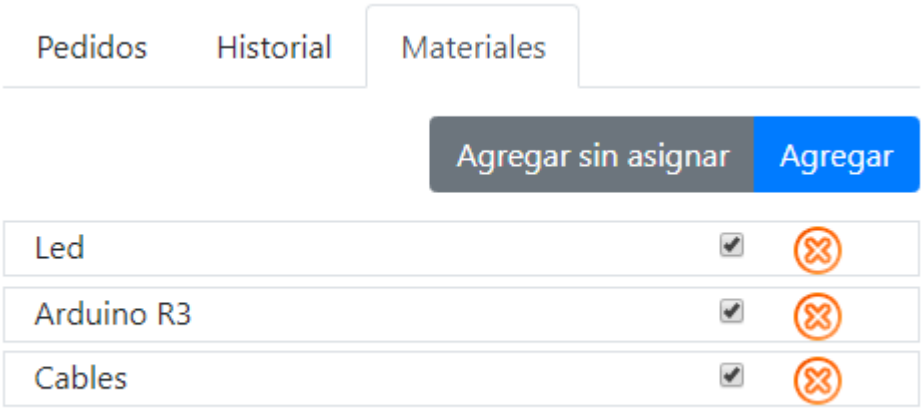

Figura 42 - Pestaña administración de Materiales

CAPÍTULO 5: ANÁLISIS Y DISCUSIÓN DE RESULTADOS

Los resultados que se obtuvieron mediante el desarrollo de este proyecto fueron satisfactorios ya que se cumplió el objetivo general y los objetivos específicos que se propusieron al principio del trabajo, el proyecto pudo finalizar con la implementación exitosa del prototipo del sistema AGV en el laboratorio Make it, este prototipo cuenta con dos vehículos con diferentes carga de materiales y con las funcionalidades básicas que le permitan desplazarse automáticamente hasta su destino, priorizando las peticiones en orden cronológico, gestionando el envío de vehículos dependiendo de los materiales solicitados y gestionando el tráfico generado entre los nodos y caminos de común uso para dos o más vehículos en ejecución. Estas características permiten al sistema cumplir con el propósito primordial de los sistemas AGV, que consisten en el desplazamiento de insumos de un punto a otro de un establecimiento sin la intervención continua de una persona.

Si bien el desarrollo de este sistema terminó con un prototipo funcional que cumplió con los requerimientos principales, esto no implica que el sistema se encuentre en su estado óptimo para su desarrollo a escala y comercialización. En su estado actual el sistema se encuentra limitado solo a sus funcionalidades esenciales, dejando fuera muchos rasgos de seguridad y gestión a gran escala, pero el propósito de este proyecto fue el de entablar las bases del sistema para un posterior desarrollo, que pueda integrar y mejorar el sistema para así llegar a un producto comercial que pueda ser distribuido para las pequeñas y medianas empresas.

A continuación, se muestra el estado final de las configuraciones que se realizaron en la implementación del sistema, además se comentarán algunas situaciones importantes que se debieron considerar y las versiones de las librerías y software utilizadas.

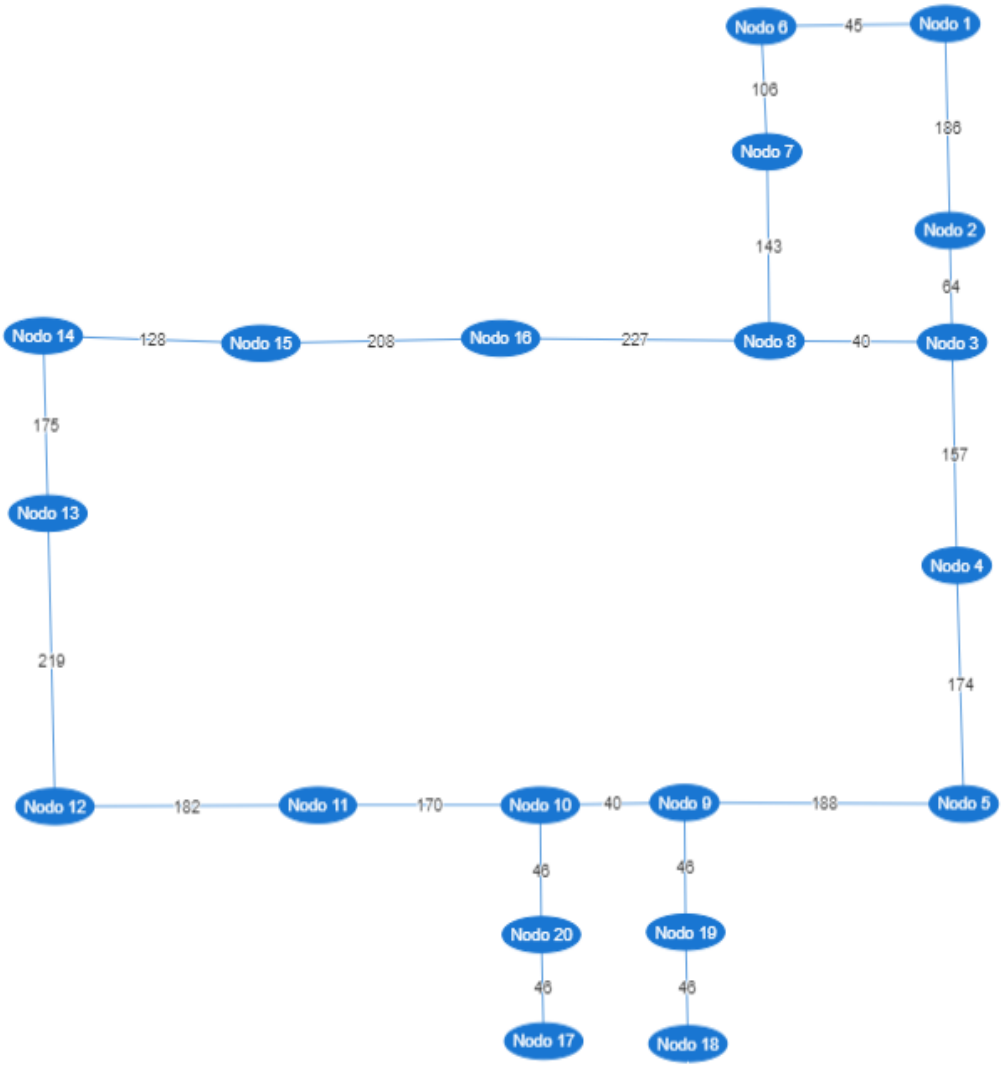

Figura 43 - Configuración final de los caminos

En la Figura 43, se muestra la disposición de las estaciones de trabajo que se configuró para satisfacer las necesidades del laboratorio, cada nodo se encuentra cerca de una estación de trabajo real. También se pueden ver las distancias medidas en centímetros que existen entre cada nodo, estas distancias son usadas por el algoritmo Dijkstra para calcular la ruta más corta. Los nodos 17 y 18 cumplen la función de estaciones “Home” donde los vehículos irán cada vez que no tengan más pedidos pendientes, y los nodos 20 y 19 fueron colocados en ese lugar para evitar una condición de bloqueo que se presentó al enviar ambos vehículos al mismo tiempo a nodos en dirección opuesta al de su nodo “home”. Es importante mencionar que los nodos 1, 2, 3, 6, 7 y 8, cumplen una función importante en la redundancia de los caminos, en el caso que no existieran los nodos 6 y 7, podrían darse casos de bloqueos indefinidos o la imposibilidad de llegar al nodo 1 por la falta de caminos alternativos. Si se desea diseñar otra configuración de caminos, es importante tomar esta consideración en cuenta.

Una vez implementado el sistema se detectaron errores particulares que es importante mencionar y tener en cuenta. A cierta hora del día, el sol se posiciona de forma que su luz entra directamente por las ventanas del laboratorio, iluminando algunos sectores del suelo con mucha intensidad, esta situación puede causar que los sensores infrarrojos capten la luz infrarroja proveniente de la luz del sol, activándolos como si estuvieran sobre una línea blanca incluso cuando esto no sea cierto. Esta situación solo ocurre a hora de la tarde cuando el laboratorio no se encuentra en uso, pero es importante considerar este error a la hora de implementar el sistema en otro espacio, pudiendo ser una posible solución crear una pieza que cubra los sensores infrarrojos de tal forma que la luz lateral no afecte al sensor. También se detectaron errores en el momento en el que el vehículo debe girar sobre un nodo cuando la velocidad del vehículo es demasiado alta, ya que al finalizar el giro la inercia del vehículo lo desplaza dejando los sensores fuera de la línea, esto hace que el vehículo continúe su rumbo desviado, teniendo que manualmente volver a posicionarlo sobre la línea, este error se soluciona ajustando los parámetros de velocidad esperada con el potenciómetro y en caso que esto no sea suficiente se debe cambiar la constante de cambio de velocidad en el giro para que este sea mucho más preciso.

El sistema fue instalado en un microcomputador “Orange Pi” modelo “PC Plus v1.1” con una distribución Linux especial para este tipo de microcomputadores llamado “Armbian “. Se utilizó la versión 8.9.4 del motor de Javascript Node.js y el gestor de paquetes npm en su versión 5.6.0, la versión de Sails utilizada fue la 1.0.2 y el paquete instalado con npm para el cálculo de rutas llamado “node-dijkstra”[2] en su versión 2.5.0. Para la programación del ESP8266 se utilizó el addon para Arduino en la versión 2.4.1, es importante destacar que existe una versión más reciente (v2.4.2) que no es compatible y se registraron errores al

[2] Librería obtenida desde https://github.com/albertorestifo/node-dijkstra

utilizarla; se utilizó la librería “arduinoWebSockets”[3] en su versión 2.1.0 y la librería “ArduinoJson”[4] en la versión 5.13.1.

Es importante destacar que se configuró una IP estática para el servidor desde el panel DHCP del router disponible en el laboratorio que se utiliza como red local. Esto es importante, ya que, si fuera una IP dinámica, al cambiar, los vehículos no se podrían conectar al servidor y se debería cambiar manualmente en el código de los vehículos.

Según las pruebas realizadas el vehículo puede soportar un máximo aproximado de 3 kg dependiendo del estado de las baterías, alrededor de este peso de carga se presentan errores en los giros y los motores deben realizar mucho esfuerzo para moverse, por lo tanto, se recomienda un máximo de 1.5 kg de peso máximo, para aumentar la vida útil de los motores. El peso que pueda soportar el vehículo depende solo de los motores utilizados, por lo tanto, si se desea aumentar el peso máximo que soportan los vehículos es necesario considerar instalar otros motores con mayor potencia.

[3] Librería obtenida desde https://github.com/Links2004/arduinoWebSockets
[4] Librería obtenida desde https://arduinojson.org/

CAPÍTULO 6: CONCLUSIONES Y RECOMENDACIONES

6.1. Conclusiones

El sistema fue implementado con éxito en el laboratorio Make It de la Universidad San Sebastián cumpliendo satisfactoriamente con los requerimientos planteados y los plazos de desarrollo planificados al comienzo de cada etapa, exceptuando las últimas actividades de la segunda etapa, que se fueron extendiendo hasta el último plazo de holgura permitido, pero sin generar grandes inconvenientes en los plazos de entrega, finalizando con un prototipo funcional con las principales características que debe tener un sistema de vehículos de guiado autónomo.

En cuanto al desarrollo físico del sistema cabe destacar, que durante el desarrollo de este prototipo se fueron presentando varios inconvenientes con respecto a los componentes y elementos utilizados para la implementación, tales como:

a. Los sensores seguidores de línea ubicados en la parte inferior del vehículo y los tags RFID que se encuentran en el piso, frecuentemente se deterioraron ocasionando problemas de ejecución y provocando su reemplazo en varias oportunidades.

b. El router con el cual se generaba la red local para el funcionamiento del sistema (Dlink DIR-600), presentaba problemas generando retraso en la comunicación con el vehículo, o desconexiones involuntarias. La solución fue reemplazarlo por uno de mejor calidad (Linksys WRT-54G). Esto se debe tener en cuenta en una futura implementación.

c. Movimientos rápidos y bruscos de los vehículos en su desplazamiento, provocando que se desviaran por la inercia del movimiento.

d. El algoritmo encargado de evitar colisiones presenta problemas cuando se desbloquea un nodo y no le da tiempo suficiente para salir de la posición actual antes que los vehículos que están esperando por entrar, ingresen a esa posición, de esta forma se provoca una colisión. Se recomienda para futuras versiones mejorar el algoritmo para que no dependa de la distancia entre los nodos.

Los inconvenientes mencionados en el punto a fueron resueltos rápidamente debido a la gran disponibilidad de estos componentes en el laboratorio, permitiendo contar con un componente de repuestos en caso de ocurrir una falla. El mencionado en el punto b fue resuelto gracias al apoyo del encargado del laboratorio quien facilitó diversos routers para su utilización en el desarrollo del proyecto y el inconveniente c se puede solucionar cambiando la tasa de ajuste de potencia de los motores.

Por parte del desarrollo del software encargado del control de los vehículos tiene un desempeño aceptable para el propósito del prototipo, cumpliendo con los requisitos mínimos para las operaciones de usuarios y de administración, pero cabe destacar que los algoritmos implementados y el código fuente de este sistema no se encuentran optimizados en su totalidad y no cuentan con elementos de seguridad para mantener la integridad del sistema, ya que no corresponden al objetivo principal de este proyecto y se priorizó el correcto funcionamiento del sistema antes que la eficiencia y seguridad de ejecución.

El sistema actualmente es una versión básica que cumple con los propósitos locales del laboratorio, el cual no es suficiente aún para su implementación en un sistema industrial ya que no cuenta con los estándares de seguridad y calidad esperados por el mercado.

En el siguiente punto, se mencionan recomendaciones y funcionalidades que recomendamos para nuevos proyectos.

6.2. Recomendaciones y Desarrollos pendientes

El proyecto a pesar de que llegó al estado esperado, aún se le pueden ampliar funcionalidades que harían un sistema más robusto. Estas recomendaciones se plantean para un proyecto futuro que tenga como objetivo ampliar el sistema actual.

La incorporación de un nuevo vehículo al sistema debe ser analizado cuidadosamente dependiendo de cómo se encuentre la configuración de las estaciones de trabajo, esto se debe considerar para evitar un bloqueo mutuo entre los vehículos en caso de que deban desplazarse por un mismo punto.

Para proyectos que deseen continuar con este desarrollo, es necesario agregar sistemas de seguridad para la parte operativa del vehículo, tales como la detección de objetos frente a su camino, finalizar la ejecución de instrucciones en caso de desvío, entre otras circunstancias que se pudieran observar; y por parte de aplicación de cliente y servidor, como acceso restringidos, persistencia de la sesión de usuario, entre otros.

En una segunda versión del proyecto, se debe mejorar el sistema de colisiones asegurando que los vehículos esperen hasta que el nodo esté totalmente desocupado. ***Tambien*** se podría solucionar este problema, agregando sensores que permitan detenerse cuando exista un obstáculo en frente.

Incorporar un pequeño sistema de gestión de materiales que se asemeje, de una manera básica, a un sistema gestor de almacenes (WMS) para darle mayor nivel de detalle a los materiales e insumos que el sistema deberá transportar.

Diseñar un sistema de carga automática para las baterías de los vehículos, así se podría dar un uso continuo de estos.

Referencias

Arduino. (s.f.). *What is Arduino?* Obtenido de https://www.arduino.cc/en/Guide/Introduction

Ballou, R. (2004). *Logística, Administracion de la cadena de suministro.* Pearson.

CEIT. (s.f.). *FACTS ABOUT CEIT*. Obtenido de http://ceitgroup.eu/index.php/en/who-we-are/introduction

Django Channels. (s.f.). *Django Channels*. Obtenido de https://channels.readthedocs.io/en/latest/#

Django. (s.f.). *Django*. Obtenido de https://www.djangoproject.com/start/overview/

European Commision. (Enero de 2017). *Germany: industry 4.0.* Obtenido de https://ec.europa.eu/growth/tools-databases/dem/monitor/sites/default/files/DTM_Industrie%204.0.pdf

Express. (s.f.). *Express*. Obtenido de https://expressjs.com/

Fette, I., & Melnikov, A. (Diciembre de 2011). *The WebSocket Protocol*. Obtenido de https://tools.ietf.org/html/rfc6455

Flask. (s.f.). *Flask*. Obtenido de http://flask.pocoo.org/

González Blázquez, A., Mulas Gómez, P. P., & Rivera Retamar, R. (2006). *Localización de dispositivos móviles en interiores usando redes Wireless.* Madrid: Universidad Complutense de Madrid.

Kurfuss, T. (Diciembre de 2014). *Industry 4.0: Manufacturing in the United States.* Obtenido de https://ostaustria.org/bridges-magazine/item/8310-industry-4-0

Martínez Lendech, J. F. (s.f.). *Sistemas de Coordenadas.* Obtenido de http://ri.uaemex.mx/bitstream/handle/20.500.11799/63801/secme-35342.pdf?sequence=1

Restifo, A. (s.f.). *node-dijkstra*. Obtenido de https://github.com/albertorestifo/node-dijkstra

Sails. (s.f.). *Sails*. Obtenido de https://sailsjs.com/

Sails. (s.f.). *Sails Blueprint API*. Obtenido de https://sailsjs.com/documentation/reference/blueprint-api

Salazar Carvajal, R. O. (17 de Febrero de 2005). *Estrategias de marketing en un entorno globalizado*. Obtenido de https://www.gestiopolis.com/estrategias-marketing-entorno-globalizado/

Salazar Lopez, B. A. (s.f.). *Gestión de Almacenes*. Recuperado el Septiembre de 2018, de https://logisticayabastecimiento.jimdo.com/almacenamiento/

Schröder, C. (2017). *The Challenges of Industry 4.0 for Small and medium-sized Enterprises.* Friedrich ebert stiftung.

Socket.io. (s.f.). *Socket.io*. Obtenido de https://socket.io/

SocketIo. (s.f.). *SocketIo*. Obtenido de https://socket.io/

Sparkfun. (s.f.). *ESP8266 Thing Hookup Guide*. Obtenido de https://learn.sparkfun.com/tutorials/esp8266-thing-hookup-guide/hardware-overview

SSI Schaefer. (s.f.). *SSI Schaefer Web Site*. Obtenido de https://www.ssi-schaefer.com/es-es

Steiner, C. (26 de Febrero de 2009). *Bot-In-Time Delivery*. Obtenido de https://www.forbes.com/forbes/2009/0316/040_bot_time_saves_nine.html#1aa70fb4b942

Sundblad, W. (30 de Julio de 2018). *What's At Stake In The Race To Industry 4.0?* Obtenido de https://www.forbes.com/sites/willemsundbladeurope/2018/07/30/whats-at-stake-in-the-race-to-industry-4-0/#680f0aba7d11

Tam, D. (2 de Diciembre de 2014). *Conoce a Kiva: el hacendoso empleado robótico de Amazon*. Obtenido de https://www.cnet.com/es/noticias/conoce-al-robot-kiva-el-hacendoso-empleado-de-amazon/

VIllena Román, J. (2009). *Sistema de Localización de Dispositivos Móviles Basada en Wireless LAN.* Madrid: Universidad Carlos III.

Vis.js. (s.f.). *Vis.js*. Obtenido de http://visjs.org/

Woodman, O. (2007). *An introduction to inertial navigation.* Cambridge: University of Cambridge.

ANEXOS

ANEXO A: Casos de usos en detalle

RQT-01	**Realizar un Pedido**		
Descripción	El sistema deberá permitir al usuario poder realizar una solicitud de material registrado en el sistema según se describe en el siguiente caso de uso		
Precondición	Haber ingresado al sitio web para realizar la solicitud y no contar con una solicitud pendiente o en ejecución.		
Secuencia Normal	**Paso**	**Acción**	
	1	El usuario deberá seleccionar desde una lista de materiales el que desea pedir	
	2	El usuario deberá seleccionar la estación de trabajo para la recepción del material solicitado, de preferencia la estación donde él se encuentre.	
	3	El usuario deberá mandar la solicitud con los datos mencionados anteriormente, presionando el botón “Pedir Vehículo”.	
	4	El sistema deberá seleccionar un vehículo que cumpla con las condiciones especificadas en los pasos anteriores.	
	5	El vehículo puede tener un pedido en ejecución.	
		5.1	Si el vehículo tiene un pedido en ejecución, agregar pedido a la cola de espera del vehículo.
		5.2	Si el vehículo no tiene pedidos en ejecución, realizar caso de uso “Enviar Vehículo”
Postcondición	Pedido realizado		
Excepciones	**Paso**	**Acción**	
	3.1	En el caso de que el usuario intente realizar un pedido sin haber realizado los pasos 1 y 2 el sistema deberá notificar la falta de información para realizar dicha solicitud	
	3.2	Si se selecciona un destino ocupado como “Estación home” de un vehículo se cancelará la solicitud y se informará al usuario	
	4	Si no se encuentra un vehículo que cumpla con las condiciones de solicitud, se cancela la ejecución del caso de uso y se notifica al usuario.	

RQT-02	**Enviar Vehículo**	
Descripción	El sistema deberá gestionar el envío de los vehículos correspondientes al pedido en ejecución.	
Precondición	Haber realizado un pedido y que se encuentre en ejecución.	
Secuencia Normal	**Paso**	**Acción**
	1	Se calcula la ruta, realizar caso de uso “Calcular Instrucciones”
	2	Se notifica al usuario la posición actual del vehículo.
	3	Se bloquea el ingreso a las estaciones de trabajo de la ruta calculada.
	4	Se notifican las estaciones de trabajo bloqueadas a los demás vehículos activos en el sistema.
Postcondición	Vehículo enviado	
Excepciones	**Paso**	**Acción**
	1	No existe una ruta factible para la ejecución de esta solicitud, se le notifica al usuario del problema presentado.

RQT-03	**Calcular Instrucciones**	
Descripción	El sistema deberá entregar las instrucciones de desplazamiento al vehículo según se describe en el siguiente caso de uso:	
Precondición	Pedido en ejecución	
Secuencia Normal	**Paso**	**Acción**
	1	El sistema deberá identificar la última estación de trabajo en el cual se encuentra el vehículo con el material solicitado.
	2	El sistema deberá calcular la ruta de desplazamiento desde la última estación en la que se encuentra el vehículo hasta la estación indicada en la solicitud en ejecución.
	3	El sistema deberá identificar la orientación en la cual se encuentra el vehículo al momento de ejecutar este paso.
	4	El sistema deberá calcular las instrucciones de desplazamiento para el vehículo con la ruta y orientación obtenidos en los pasos 2 y 3.
	5	El sistema deberá enviar las instrucciones obtenidas en el paso 4 al vehículo seleccionado.
Postcondición	Instrucciones Calculadas	
Excepciones	**Paso**	**Acción**
	6	En caso de que el vehículo se encuentre en la misma estación de trabajo indicada en la solicitud, la ejecución del caso de uso se detiene en el paso 2 y se notifica al usuario que el vehículo ya se encuentra en el lugar.
	7	En caso de que la estación de trabajo ingresada en la solicitud no coincida con las estaciones disponibles en el mapa, se cancela la ejecución del caso de uso y se notifica al usuario del fallo al intentar ejecutar su solicitud.

RQT-04	**Cancelar Pedido**	
Descripción	El sistema deberá permitir al usuario poder cancelar un pedido realizado.	
Precondición	Haber realizado un pedido y que se encuentre en cola o en ejecución	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario deberá presionar el botón “Cancelar” en la página web del pedido que haya realizado
	2	El sistema deberá guardar un registro de dicho pedido con estado “Cancelado”
	3	El sistema debe liberar el vehículo cuando llegue a una estación de trabajo, realizar caso de uso Desocupar Vehículo
Postcondición	Pedido Cancelado	
Excepciones	**Paso**	**Acción**
	1	Si el usuario o el vehículo se desconecta del sistema, el pedido deberá ser cancelado automáticamente, comenzando la ejecución de este caso de uso desde el paso número 2.
	3	Si el pedido se encuentra en cola, este paso se omite y se retira dicho pedido de la cola de espera.

RQT-05	**Desocupar Vehículo**		
Descripción	El sistema deberá permitir en cualquier momento desde la solicitud de un pedido, liberar el vehículo por cancelación o finalización de dicho pedido, para la ejecución de los pedidos pendientes según se describe en el siguiente caso de uso:		
Precondición	Se ha realizado una cancelación del pedido realizado o se completó correctamente la ejecución del pedido.		
Secuencia Normal	**Paso**	**Acción**	
	1	Una vez que el usuario haya obtenido los materiales solicitados y terminado el uso del vehículo, el usuario deberá liberar el vehículo presionando el botón "Desocupar", de la página donde se realizó dicho pedido	
	2	El sistema deberá registrar el pedido como Finalizado.	
	3	El sistema deberá retirar el pedido actual de la cola de ejecución	
	4	Existen más pedidos en cola	
		4.1	Si no existen más pedidos en cola, se deberá realizar el caso de uso "Enviar vehículo" con destino a su "Estación Home".
		4.2	Si existen más pedidos en cola, se deberá realizar el caso de uso "Enviar vehículo" con la información del siguiente pedido en la cola.
Postcondición	Vehículo desocupado		
Excepciones	**Paso**	**Acción**	
	1	Si existen pedidos en cola para el vehículo utilizado, el caso de uso se ejecutará después de 5 min desde la llegada al destino.	
	2	Si este caso de uso es ejecutado por la cancelación de un pedido, se omite este paso	

RQT-06	**Notificar Posición**	
Descripción	El sistema debe mantener un seguimiento de las estaciones de trabajo que va recorriendo cada vehículo en cada instante de la ejecución de un pedido.	
Precondición	El vehículo ha pasado por una estación de trabajo	
Secuencia Normal	**Paso**	**Acción**
	1	El vehículo deberá enviar el identificador de cada estación de trabajo por la que pase.
	2	El sistema deberá guardar, por cada vehículo, un registro de los identificadores por el cual hayan pasado.
Postcondición	Posición Notificada	
Excepciones	**Paso**	**Acción**
	3	El sistema deberá ignorar todos los identificadores que sean iguales al último identificador registrado.
	4	El sistema deberá ignorar todo identificador que no se encuentre registrado anteriormente en el sistema por parte del administrador.

RQT-07	**Ejecutar Instrucciones**	
Descripción	El vehículo deberá ejecutar las instrucciones entregadas por el servidor.	
Precondición	El vehículo ha recibido instrucciones nuevas para ejecutar.	
Secuencia Normal	**Paso**	**Acción**
	1	El vehículo siempre deberá ejecutar la primera instrucción automáticamente
	2	El vehículo obtiene el identificador de una estación de trabajo
	3	Si la estación de trabajo en la que se encuentra el vehículo cuenta con un bloqueo de el mismo, el sistema deberá quitar el bloqueo de este vehículo de esta estación.
	3	Se realiza el caso de uso “Comprobar nodos bloqueados”
	4	El vehículo ejecuta la instrucción que se encuentre asociada a esa estación.
	5	El vehículo deberá notificar el identificador de la estación de trabajo por la cual acaba de pasar, realizar caso de uso Notificar Posición
Postcondición	Instrucciones ejecutadas	
Excepciones	**Paso**	**Acción**
	4	Si el vehículo obtiene el identificador de una estación de trabajo que no corresponde a la ruta entregada por el sistema, el vehículo seguirá ejecutando la última instrucción antes de obtener dicho identificador.

RQT-08	**Comprobar nodos Bloqueados**		
Descripción	El vehículo deberá comprobar que la siguiente estación de trabajo a la cual deba dirigirse se encuentre bloqueada por otro vehículo del sistema.		
Precondición	Encontrarse en una estación de trabajo		
Secuencia Normal	**Paso**	**Acción**	
	1	Obtener el identificador de la siguiente estación de trabajo	
	2	Verificar si la estación de trabajo siguiente se encuentra en la lista de estaciones bloqueadas	
		2.1	Si la siguiente estación de trabajo se encuentra en la lista de estaciones bloqueadas, el vehículo debe detenerse y esperar a que dicha estación se desbloquee
		2.2	Si la siguiente estación de trabajo no se encuentra en la lista de estaciones bloqueadas, el vehículo deberá ejecutar las instrucciones con normalidad.
Postcondición	Bloqueo comprobado		

ANEXO B: Casos de prueba en detalle

Código	Objeto de prueba	Descripción
P01	Seguidor de Líneas	La prueba consistirá en comprobar si el vehículo puede desplazarse siguiendo la línea blanca y este pueda ajustar los motores para corregir el desvío correctamente. La línea blanca debe tener un largo igual o superior a 1 metro para esta prueba.
P02	Lector de RFID	El vehículo deberá emitir un sonido mediante un buzzer cada vez que lea un tag RFID.
P03	Ejecución de instrucciones	El vehículo deberá ejecutar correctamente una de las cinco instrucciones según la codificación numérica entregada que se le asignó a dicha instrucción. Si para una codificación numérica el vehículo ejecuta una instrucción diferente o ninguna, se considera un fallo.
P04	Regulador de velocidad	El objetivo de esta prueba es comprobar el ajuste de velocidad que realiza el vehículo variando la velocidad esperada mediante un potenciómetro instalado en él, para comprobar los cambios se mostrará en un monitor serial los datos correspondientes a la velocidad esperada, potencia de cada motor y velocidad de cada motor.
P05	Conexión al servidor	El objetivo es comprobar la conexión del vehículo a la red wifi y al servidor, mediante la ayuda de un escaneado de red se comprobará si el vehículo se ha conectado a la red y el servidor indicará cada vez que un vehículo se haya conectado a él.
P06	Notificar RFID	Comprobar que cada vez que el vehículo pase por un tag RFID este sea enviado al servidor. para comprobar esto el servidor mostrará la información recibida.
P07	Módulo cálculo de instrucciones	Comprobar, según un mapa de pruebas, que el servidor entrega correctamente las instrucciones que serán ejecutadas en cada tag RFID. las instrucciones entregadas se considerarán correctas siempre y cuando se llegue al destino.

P8	Gestión de bloqueos	Cuando se realiza un pedido, todos los nodos que sean parte de la ruta deben ser bloqueados por el vehículo que vaya a ejecutar el pedido. Se debe comprobar que se realice esta tarea y que se les envíe a todos los vehículos la lista de nodos bloqueados.
P09	Gestión de bloqueos Notificar RFID	Cuando un vehículo pase por un nodo y notifique el tag RFID se debe retirar el bloqueo del vehículo de ese tag RFID
P10	Envío de instrucciones desde el servidor	El objetivo es comprobar si las instrucciones calculadas en el servidor son recibidas correctamente por el ESP8266 del vehículo. para comprobar esta prueba, se mostrará por monitor serial los datos que reciba el ESP8266
P11	Aplicación web Cliente, Cálculo de instrucciones	Realizar y cancelar un pedido y que el servidor entregue las instrucciones para llegar al destino, en caso de ser cancelado, el servidor entregará las instrucciones para que el vehículo regrese a casa según su última posición.
P12	Gestión de Pedidos, Cálculo de instrucciones, Aplicación web Cliente	Poder realizar más de un solo pedido, los cuales estarán en sistema de colas hasta que se haya terminado de ejecutar el primer pedido, la aplicación del cliente deberá recibir la notificación si su pedido se encuentra en ejecución o si se encuentre en estado de espera.
P13	Lectura de RFID, Ejecución de instrucciones	Comprobar la ejecución de la instrucción correspondiente al RFID que el vehículo ha leído. Si se lee un tag desconocido, se ignora y se sigue con la ejecución de la instrucción que estaba realizando.
P14	Ejecutar instrucciones, Seguidor de líneas, Regulador de velocidad	Comprobar que las instrucciones de giro se ejecuten correctamente y el vehículo quede posicionado de tal forma que pueda seguir la línea blanco frente a él.
P15	Aplicación web Cliente, Gestión de Pedidos, Cálculo de instrucciones, Ejecución de instrucciones	Cuando se cancele o finalice un pedido el sistema el vehículo deberá realizar ejecutar las instrucciones generadas por el servidor para volver a su punto de origen.

P16	Aplicación web Cliente, Gestión de Pedidos, Cálculo de instrucciones, Ejecución de instrucciones	Poder ejecutar dos o más pedidos al mismo tiempo, uno por cada vehículo disponible, se debe comprobar que el vehículo enviado a un pedido sea el que cuente con el material solicitado en dicho pedido, es decir, tiene que ser el vehículo que tenga asignado el material solicitado.
P17	Ejecución de instrucciones, Gestión de bloqueos	Cuando uno o más vehículos deban pasar por un mismo camino se deberá comprobar que el primero en ejecutar las instrucciones sea el que recorra el camino mientras que el siguiente queda detenido en el tag RFID anterior, hasta que el primero haya pasado por el tag RFID en común.

ANEXO C: Diagramas de flujo

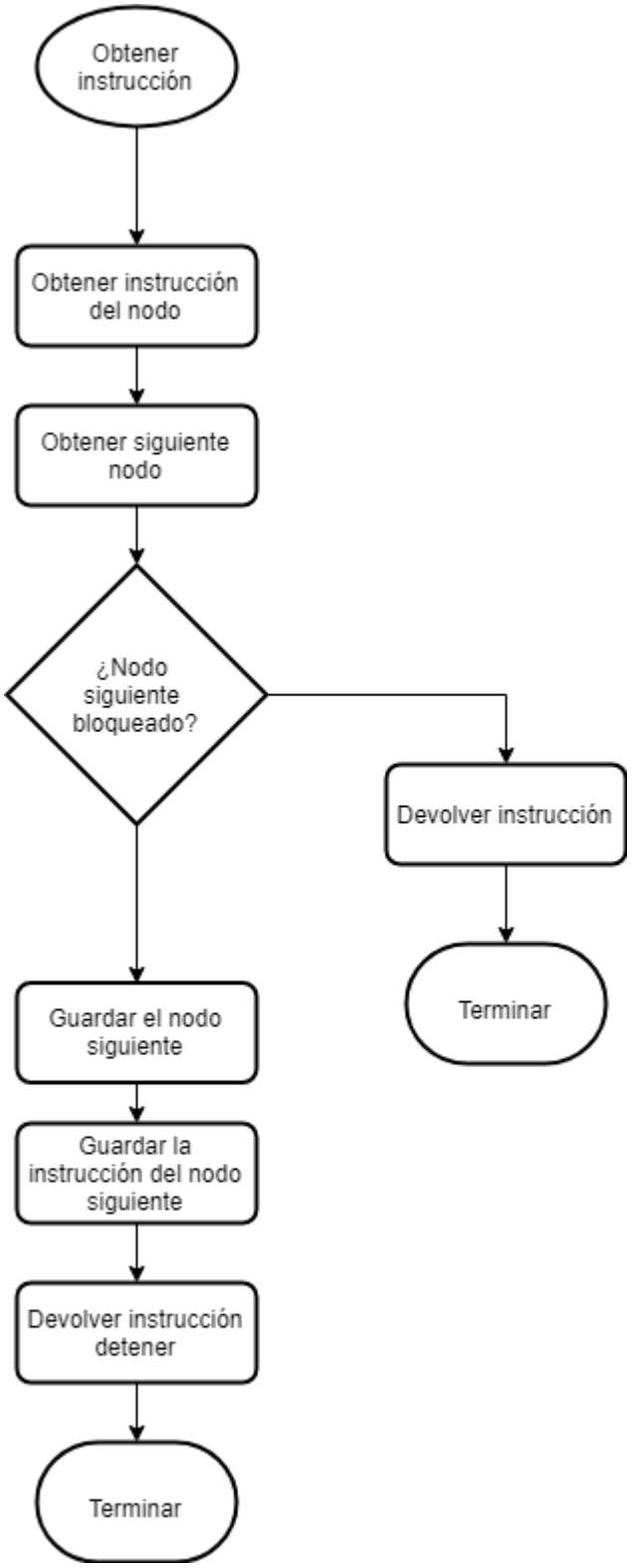

Figura 44 - Diagrama de flujo obtener instrucción

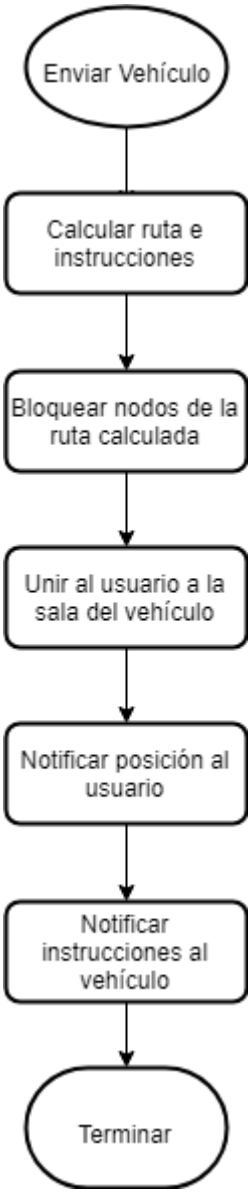

Figura 45 - Diagrama de flujo enviar vehículo

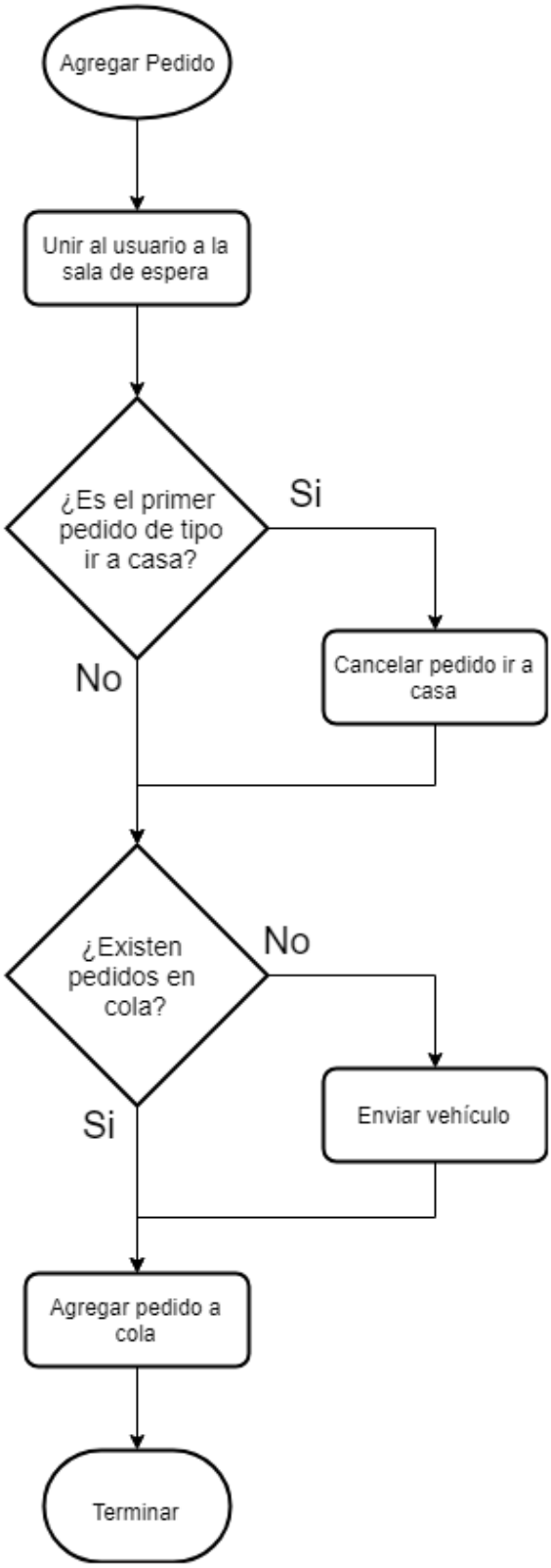

Figura 46 - Diagrama de flujo agregar pedido

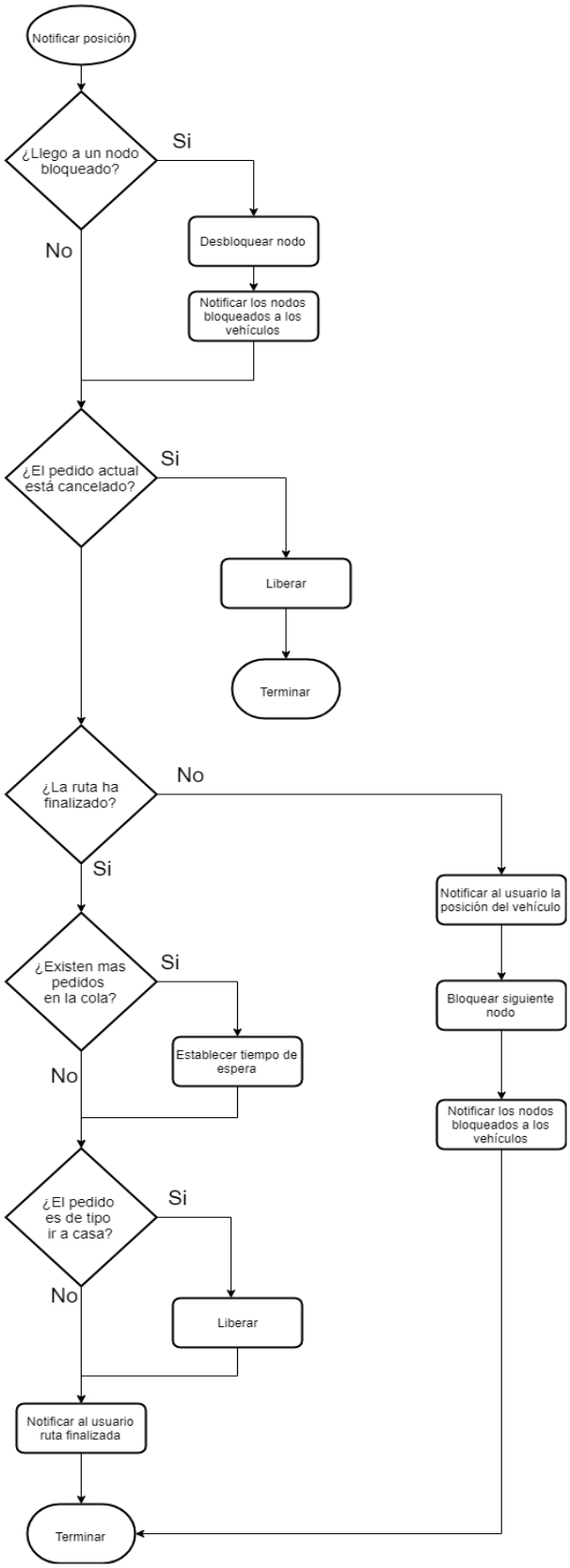

Figura 47 - Diagrama de flujo notificar posición

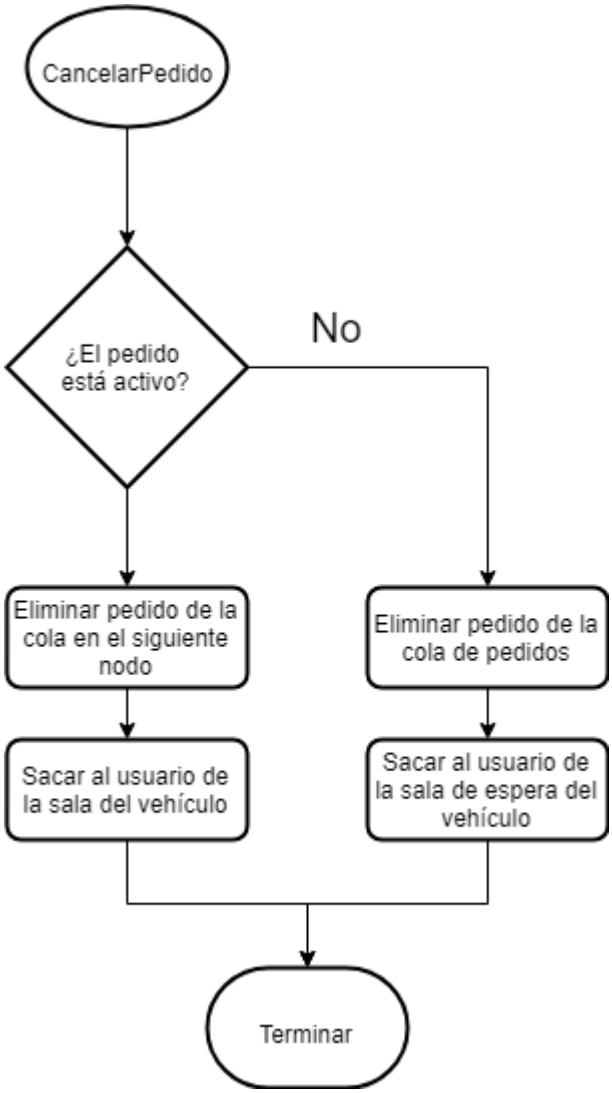

Figura 48 - Diagrama de flujo cancelar pedido

ANEXO D: Planos de piezas

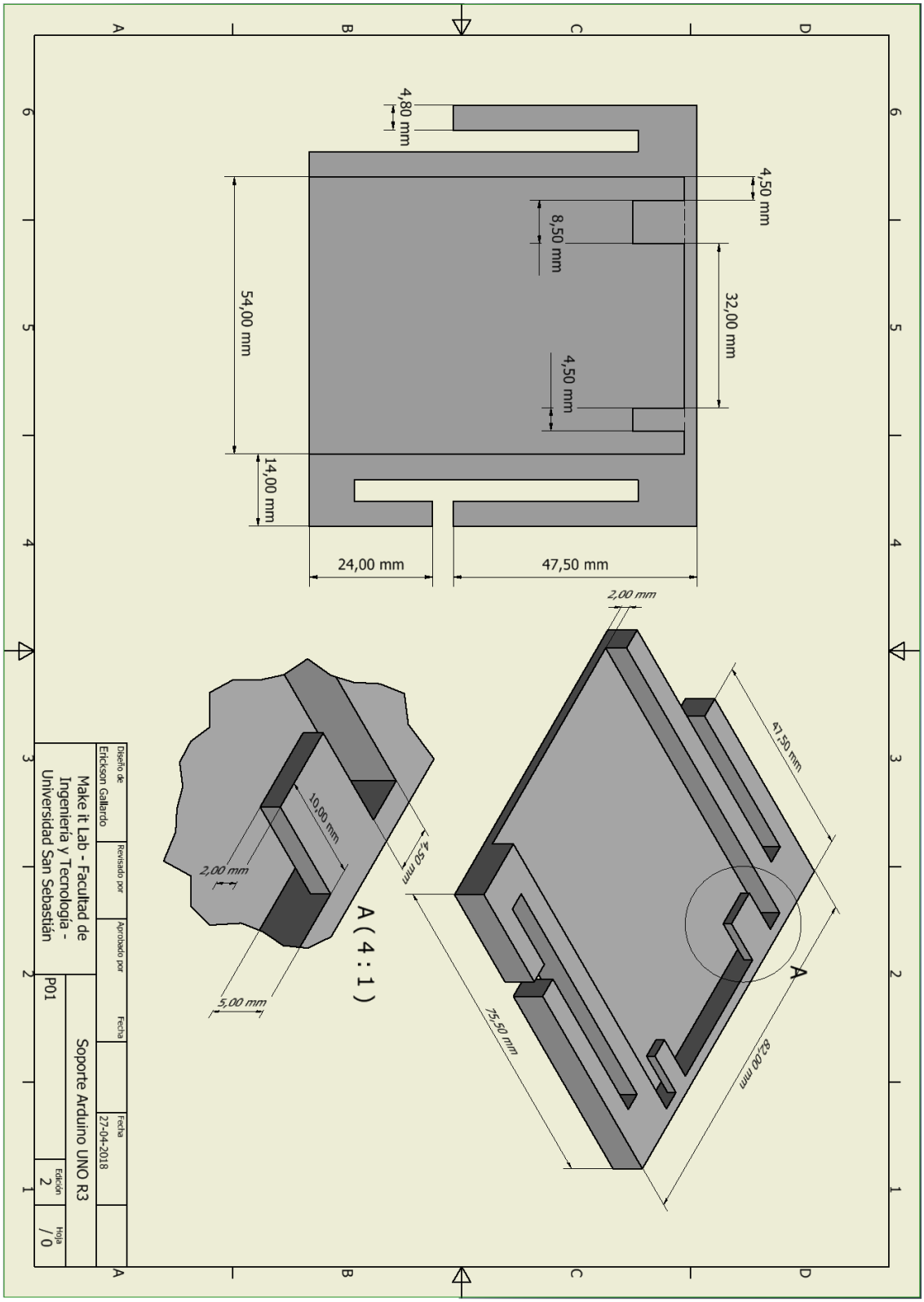

Figura 49 - Plano pieza soporte Arduino

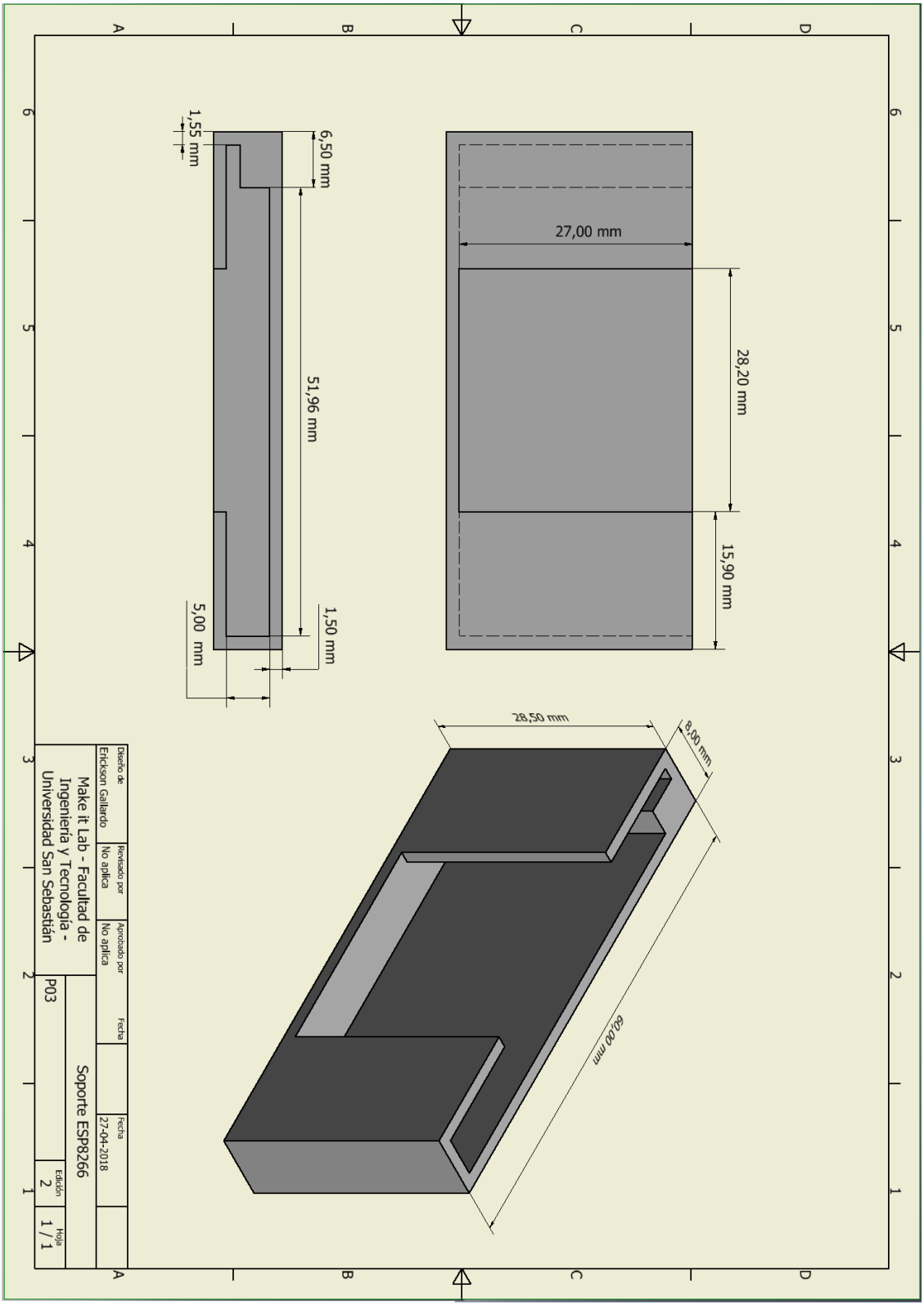

Figura 50 - Plano pieza soporte ESP8266

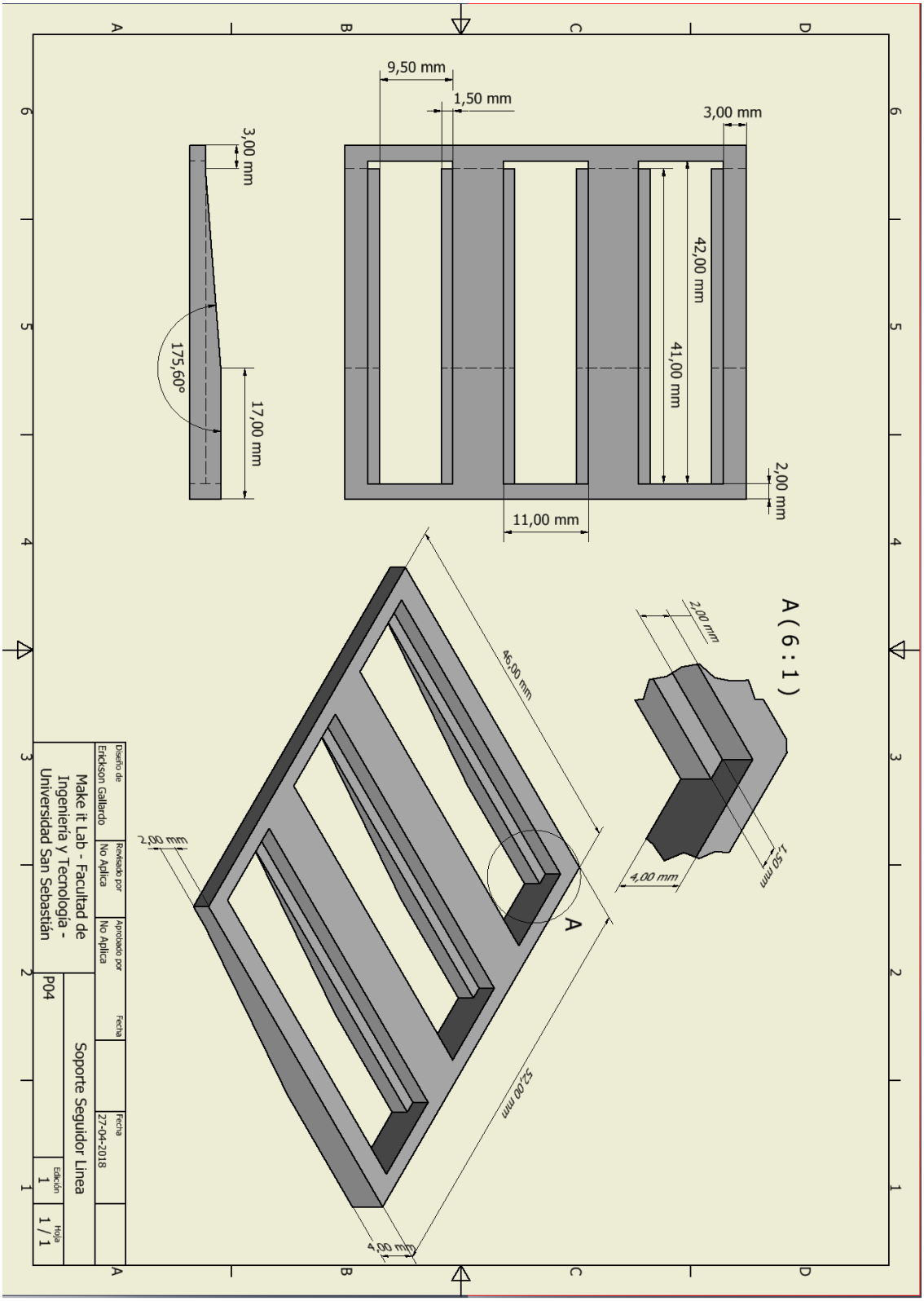

Figura 51 - Plano pieza soporte sensores de línea

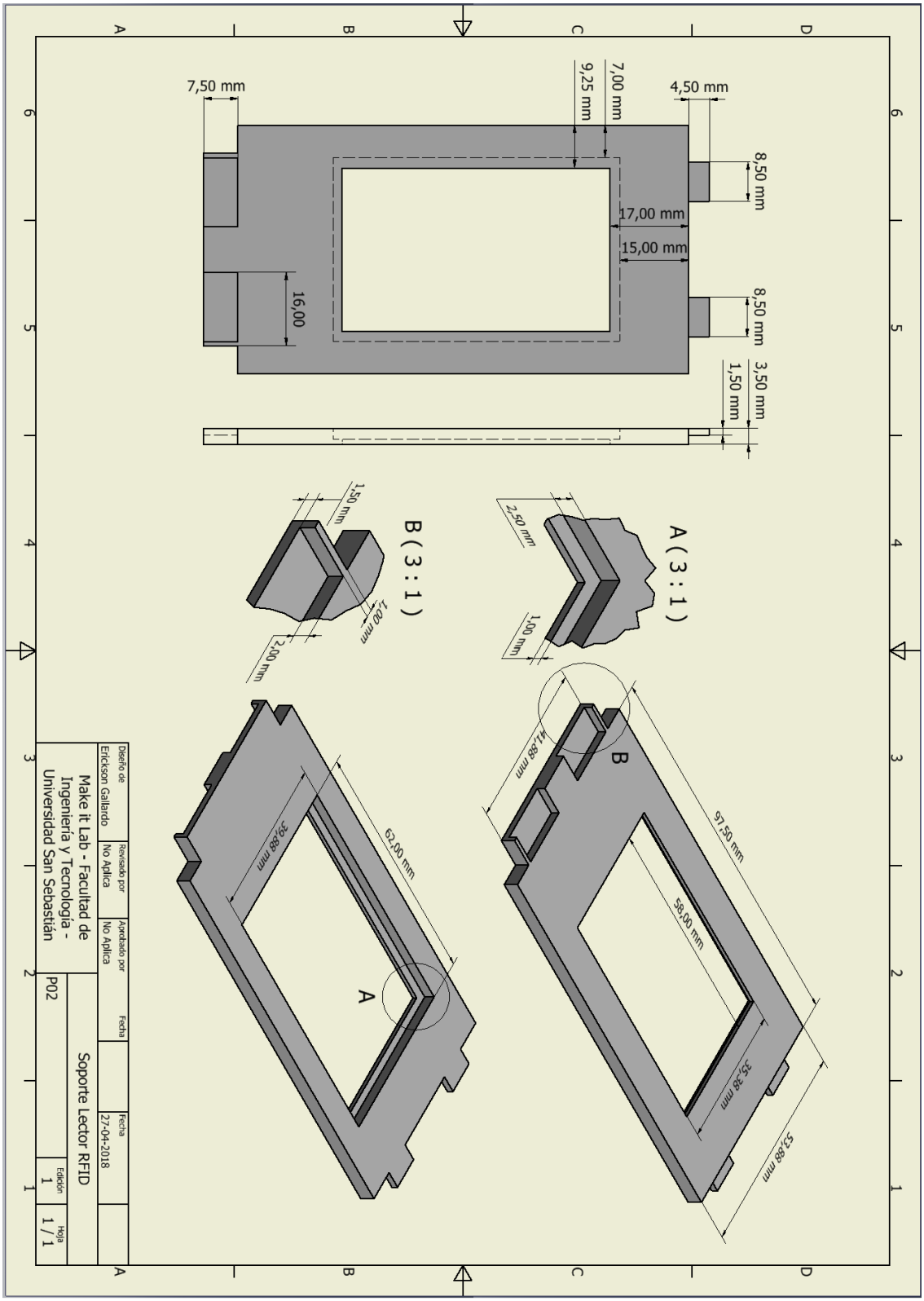

Figura 52 - Plano pieza soporte lector RFID

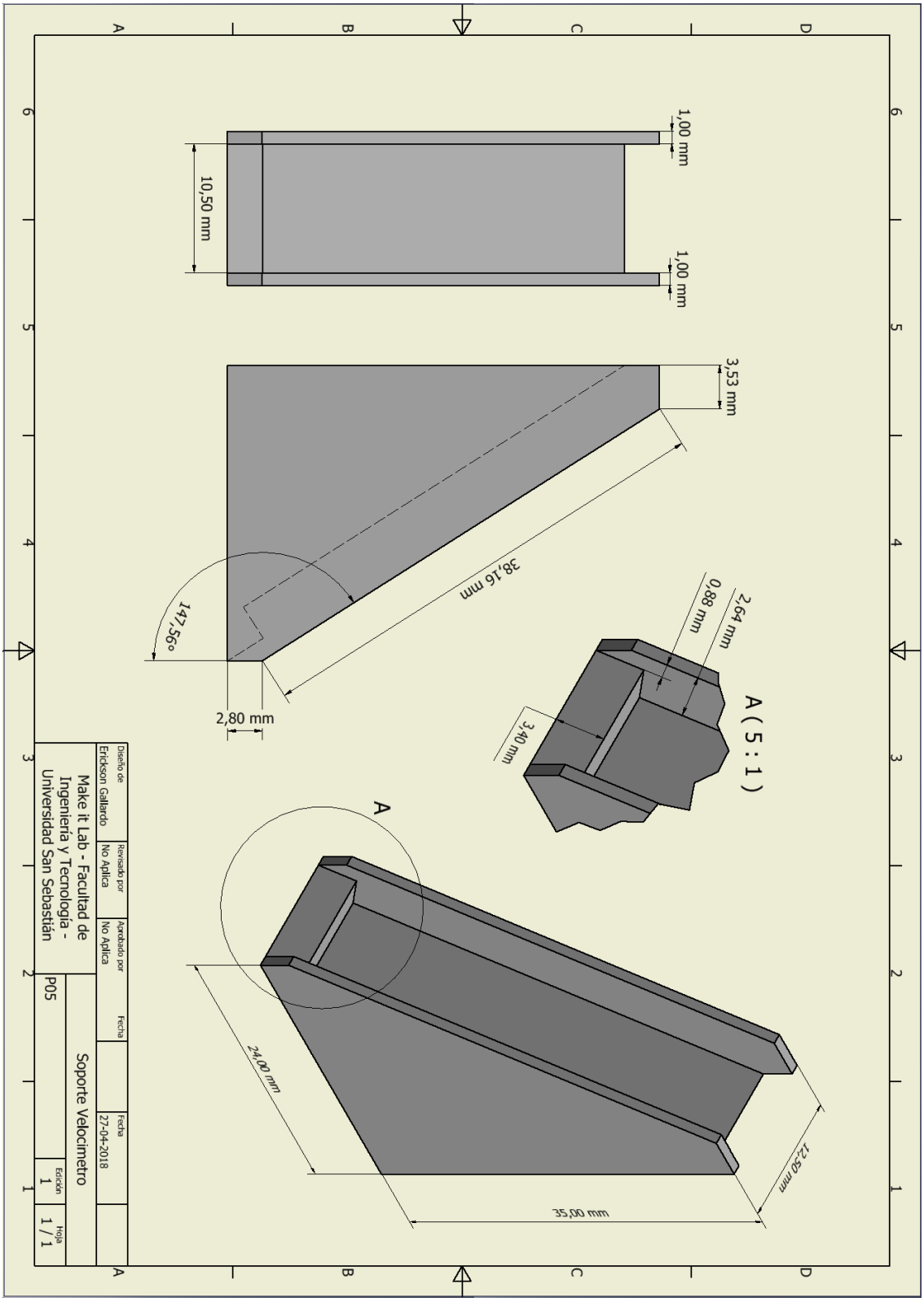

Figura 53 - Plano pieza soporte sensores de velocidad

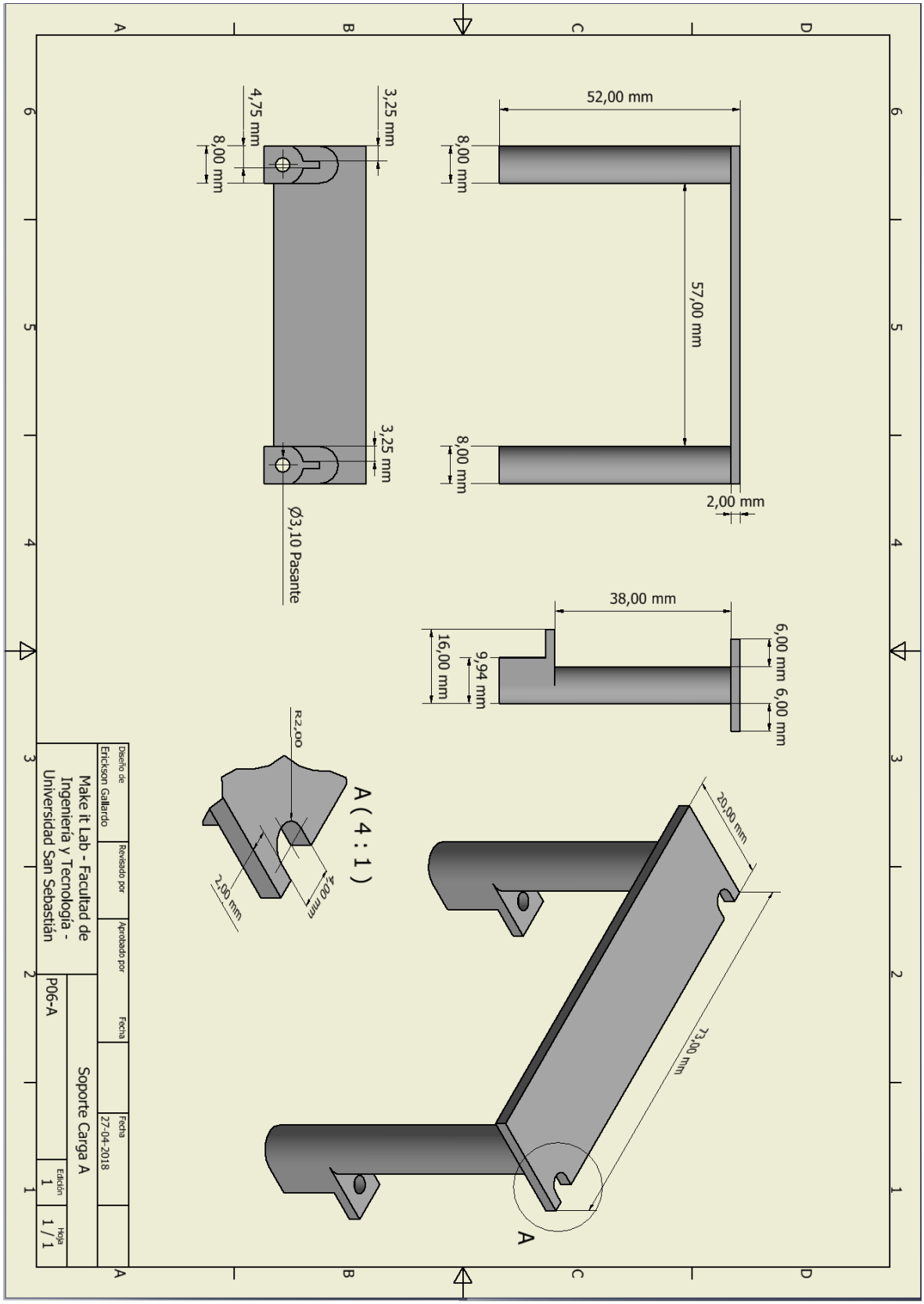

Figura 54 - Plano pieza soporte carga tipo A

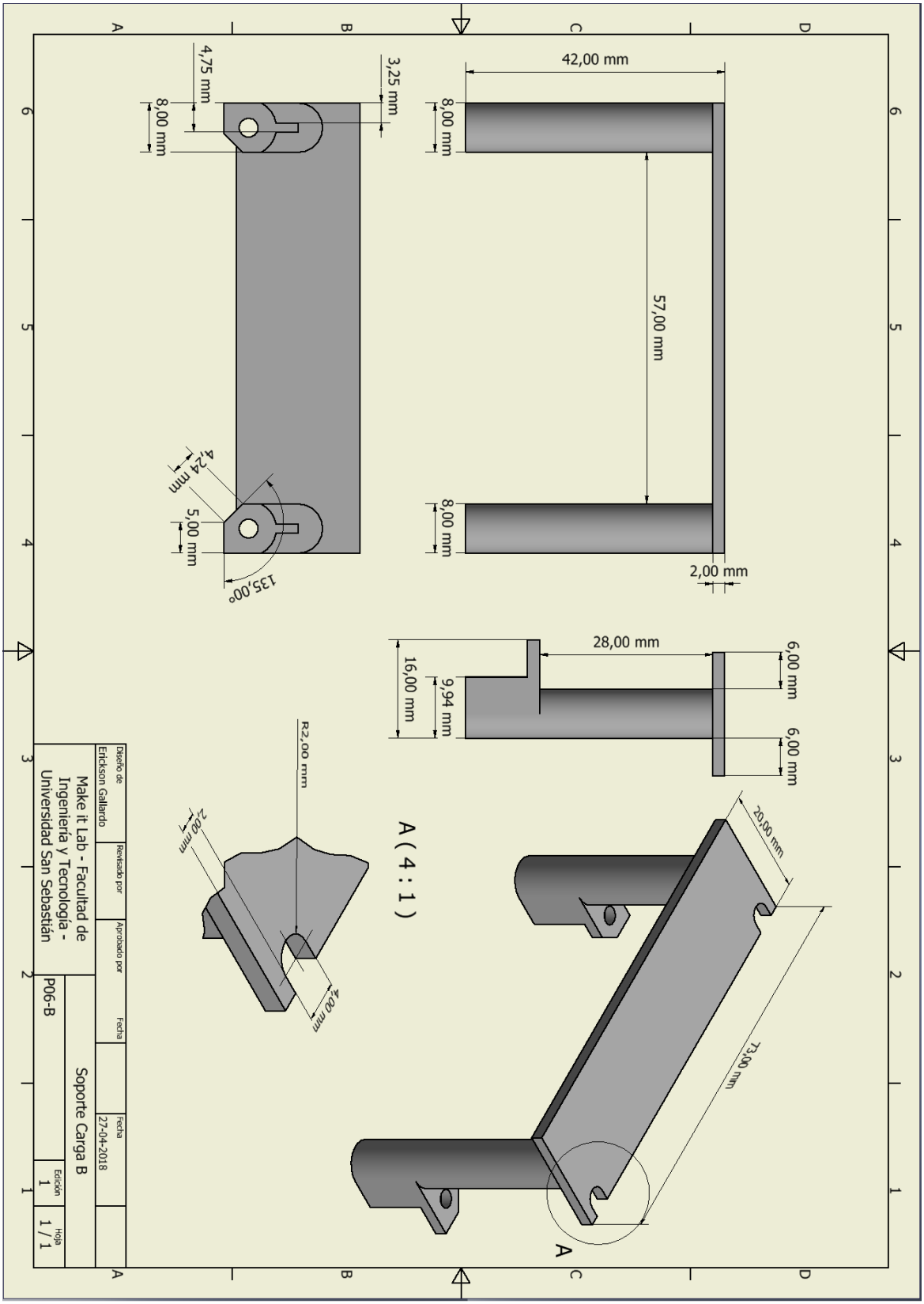

Figura 55 - Plano pieza soporte carga tipo B

ANEXO E: Diagrama de circuito

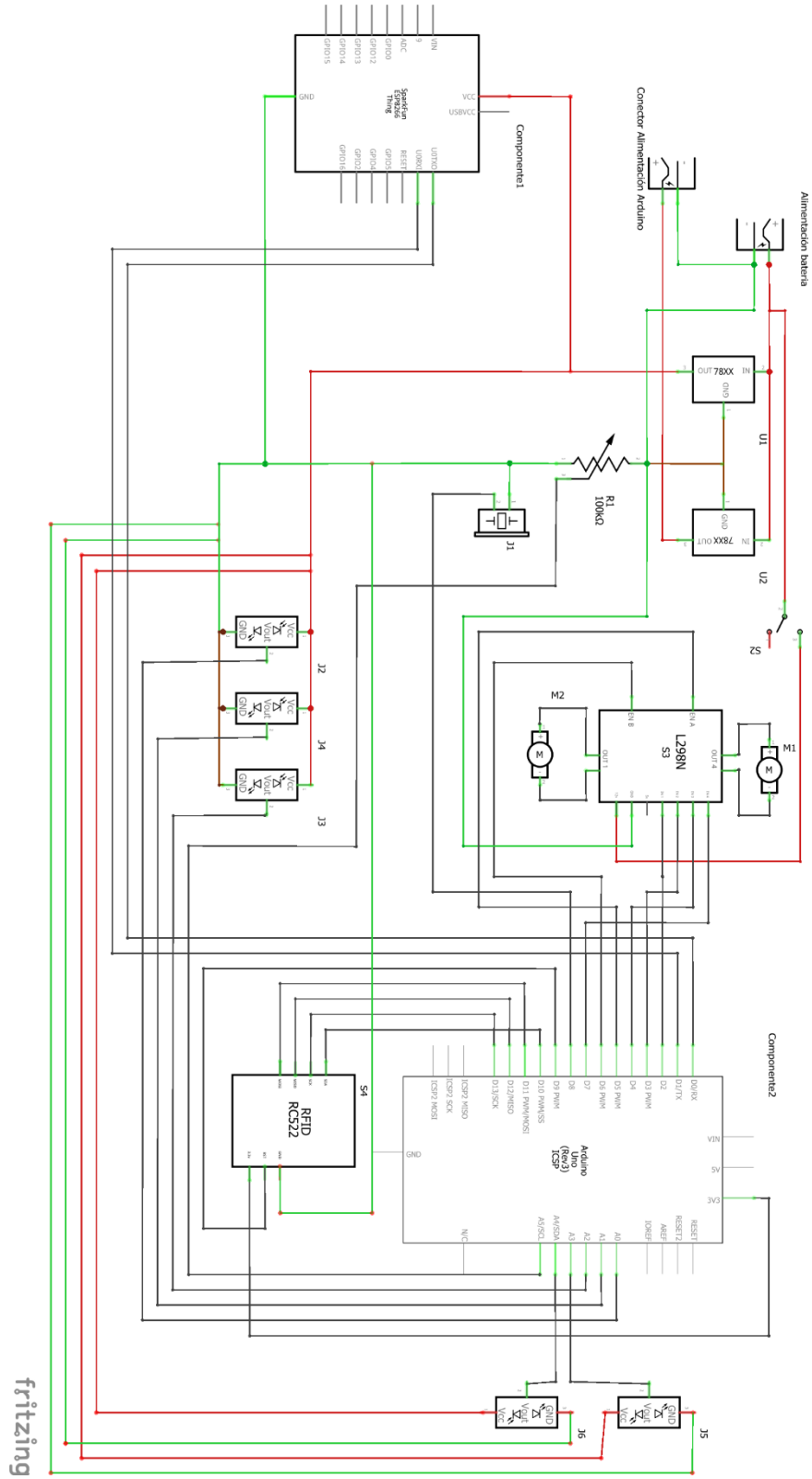

Figura 56 - Diagrama circuito eléctrico del vehículo

ANEXO F: Tabla de costos

MATERIAL	COSTO UNITARIO	CANTIDAD	COSTO TOTAL
CHASIS	$ 14.337	1	$ 14.337
SENSOR INFRARROJO	$ 2.890	5	$ 14.450
SENSOR RFID	$ 2.900	1	$ 2.900
NFC TAGS (50 UNIDADES)	$ 4.964	2	$ 9.928
CINTA AISLADORA BLANCA 3/4" 10M	$ 1.490	5	$ 7.450
PUENTE H	$ 3.390	1	$ 3.390
BATERIA LIPO 12V	$ 28.000	1	$ 28.000
CABLES JUMPER (75 UNIDADES)	$ 6.990	1	$ 6.990
ARDUINO UNO R3	$ 6.700	1	$ 6.700
ESP8266	$ 11.290	1	$ 11.290
ORANGE PI PC PLUS	$ 16.573	1	$ 16.573
ROUTER LINKSYS WRT-54G	$ 37.104	1	$ 37.104
		TOTAL	$ 159.112

Tabla 8 - Tabla de costos