TECNOLÓGICO NACIONAL DE MÉXICO CAMPUS JIQUILPAN

INGENIERÍA EN SISTEMAS COMPUTACIONALES

MATERIAS:

GRAFICACIÓN

SISTEMAS PROGRAMABLES

ALUMNOS:

JESUS EDUARDO CERVANTES SÁNCHEZ

ALBERTO MENDOZA CARDENAS

LUIS DAVID RAMIREZ SÁNCHEZ

HUGO GUADALUPE ROJAS CAMARENA

CRISTIAN DANIEL SÁNCHEZ SÁNCHEZ

MAESTROS:

ERICK DALET VILLANUEVA MASCORT

FRANCISCO ARMANDO PAYÁN GUERRERO

MONTACARGAS REMOTO (MONTACARGUITAS)

JIQUILPAN, MICHOACÁN, 7 DE MAYO DEL 2024

I Definición del Problema

Antecedentes de la Investigación

El manejo eficiente de cargas es fundamental para garantizar la productividad y la seguridad en el lugar de trabajo. Los montacargas son una herramienta comúnmente utilizada en entornos industriales para el transporte y la manipulación de mercancías pesadas. Sin embargo, el proceso tradicional de operación de montacargas puede presentar desafíos en términos de eficiencia y seguridad.

Los avances en tecnología wifi y bluetooth han permitido el desarrollo de sistemas remotos cada vez más sofisticados y confiables. Esto permite controlar toda la maquinaria desde una distancia segura previniendo accidentes.

Antecedentes del problema

La seguridad laboral es una preocupación clave en cualquier entorno industrial. El manejo de cargas pesadas conlleva riesgos de lesiones para los trabajadores si no se implementan medidas de seguridad adecuadas. La implementación de sistemas de montacargas remoto puede ayudar a reducir estos riesgos al permitir que los operadores controlen la maquinaria desde una ubicación segura.

Planteamiento del problema

Las empresas necesitan mejorar la seguridad en sus operaciones logísticas mediante la implementación de un sistema de montacargas remoto controlado por Arduino en su entorno industrial.

II. Objetivos de la Investigación

General

Implementar un sistema de montacargas remoto controlado por Arduino y Unity en entornos industriales con el fin de mejorar la seguridad y la eficiencia en las operaciones con montacargas.

Específicos

- Investigar y seleccionar los componentes necesarios para la construcción del sistema de montacargas remoto.
- Diseñar y desarrollar el sistema de control remoto utilizando tecnología WiFi asegurando la comunicación entre el dispositivo de control y el montacargas.
- Evaluar el impacto del sistema de montacargas remoto en las operaciones de la empresa, analizando la reducción de riesgos laborales.
- Realizar ajustes y mejoras en el sistema de montacargas remoto según sea necesario.

III. Justificación

La implementación de un sistema de montacargas remoto controlado por Arduino en entornos industriales es una solución innovadora y estratégica para abordar los desafíos críticos de las operaciones de manejo de cargas pesadas actualmente. Esta tecnología no solo aborda las preocupaciones de seguridad laboral, sino que también tiene el potencial de transformar la eficiencia operativa y la competitividad empresarial.

En primer lugar, el riesgo asociado con el manejo manual de cargas pesadas constituye una amenaza constante para la integridad física de los trabajadores. Lesiones en los músculos, huesos, atropellamientos y otros accidentes graves son una realidad en muchos entornos industriales. Al permitir que los operadores controlen los montacargas desde una ubicación remota y segura, elimina efectivamente la exposición directa de los trabajadores a estos peligros, reduciendo drásticamente la probabilidad de accidentes laborales graves. Además, al cumplir con las normativas y regulaciones de seguridad laboral más estrictas, las empresas pueden evitar costosas sanciones y procesos legales.

Más allá de los beneficios en materia de seguridad, la implementación de este sistema de montacargas remoto controlado por Arduino también ofrece oportunidades significativas para mejorar la eficiencia operativa. Además, al proporcionar una visión panorámica del entorno de trabajo, los operadores remotos pueden planificar y coordinar tareas de manera más efectiva, lo que se traduce en una mayor eficiencia y rendimiento.

Los beneficios económicos tangibles de esta inversión son igualmente notables. Los costos asociados con accidentes laborales, daños a la mercancía se reducen significativamente con la implementación de este sistema. Además, el aumento de la productividad y la eficiencia operativa resultante se traduce directamente en mayores ganancias y una ventaja competitiva en el mercado.

Pero los beneficios de adoptar esta tecnología innovadora van más allá de los aspectos puramente operativos y financieros. La implementación de este sistema

demuestra el compromiso de la empresa con la innovación y la mejora continua, lo que puede mejorar su imagen corporativa.

IV Diseño del Marco Teórico

El diseño del marco teórico es donde se mencionan los temas teóricos para comprender el proyecto y su clasificación.

01. Automatización y control remoto en la industria

01.1 ¿Qué es la automatización?

El concepto de automatización está enfocado en una disciplina de control que se basa en el uso de sistemas electromecánicos para controlar de forma automatizada diversos procesos industriales. Abarca control, sistemas digitales, supervisión, gestión de datos, accionamientos, instrumentación, comunicaciones, producción, interacciones y muchos otros.

La automatización incorpora elementos y dispositivos tecnológicos que aseguran tener un control específico sobre los procesos y sus evidentes comportamientos.

La automatización es muy usada en diferentes áreas de trabajo. Los elementos y características de la automatización han tenido importante impacto en el área industrial, mecánica, informática, máquinas y programables.

La alta competitividad empresarial y la necesidad de aumentar eficazmente los procesos de producción mediante la incorporación de la robótica, los robots y automatizar los procesos, han implicado un mayor nivel de integración entre los sistemas productivos y la decisión política empresarial en áreas de fabricación, gestión de procesos, servicios y la gestión de la información.

01.2 Importancia y Objetivos de la Automatización

- Incrementar la Productividad y Reducir Costos:

La automatización busca aumentar la eficiencia de los procesos de producción, lo que resulta en una reducción de costos y una mejora en la calidad de los productos.

- Optimizar Condiciones Laborales:

Mejora las condiciones de trabajo al eliminar tareas tediosas y riesgosas, incrementando la seguridad laboral y la satisfacción de los empleados.

- Ejecutar Nuevas Actividades:

Permite realizar actividades y funciones que no podrían llevarse a cabo manualmente, simplificando procesos complejos.

- Aumentar Producción y Disponibilidad:

Incrementa la producción y disponibilidad de productos o servicios según las demandas empresariales, eliminando desperdicios mediante sistemas como el lean manufacturing.

- Integración de Procesos:

Facilita la integración eficiente de los procesos operativos y de producción, racionalizando los insumos y los procedimientos.

- Retorno de Inversión:

Genera un retorno de inversión más rápido y productivo, mejorando la trazabilidad y reduciendo errores potenciales.

- Simplificación de Actividades:

Simplifica las actividades de modo que los trabajadores no necesiten conocimientos extensivos para operar los sistemas automatizados.

- Menor Inversión y Mayor Retorno:

Reduce la inversión necesaria para implementar sistemas automatizados específicos, aumentando la producción y los retornos de inversión.

- Reducción de la Participación Humana:

Permite que los sistemas de producción realicen importantes actividades de procesamiento, ensamblaje, manejo de materiales e inspección con mínima intervención humana.

1.3 ¿Qué es el control remoto?

El control remoto implica la operación y monitoreo de sistemas, dispositivos o procesos desde una ubicación distinta a donde estos se encuentran físicamente. Esto se logra mediante la transmisión de señales a través de diversas tecnologías de comunicación, como redes de radiofrecuencia, infrarrojos, internet, o redes móviles. El control remoto es fundamental en la automatización moderna y en sistemas distribuidos, permitiendo a los operadores interactuar con máquinas y sistemas sin necesidad de estar físicamente presentes.

02. Protocolo de comunicación

02.1 TCP/IP

TCP/IP es un protocolo de enlace de datos que se usa en Internet para que los ordenadores y otros dispositivos envíen y reciban datos. TCP/IP son las siglas en inglés de Transmission Control Protocol/Internet Protocol (protocolo de control de transmisión/protocolo de Internet). Posibilita que los dispositivos conectados a Internet se comuniquen entre sí en varias redes.

02.2 ¿Qué hace TCP/IP?

Para garantizar que cada comunicación llegue intacta al destino deseado, el modelo TCP/IP divide los datos en paquetes y luego los vuelve a juntar para formar el mensaje completo en el destino. Enviar los datos en paquetes pequeños hace que sea más fácil mantener la exactitud que enviando todos los datos a la vez.

Después de dividir un mensaje individual en paquetes, estos pueden recorrer diversos caminos en caso de congestión.

02.3 ¿Cómo funciona?

Cuando se envía algo por Internet, ya sea un mensaje, una foto o un archivo, el modelo TCP/IP divide esos datos en paquetes según un procedimiento de cuatro capas. Los datos primero atraviesan estas capas en un sentido, y luego lo hacen en sentido contrario cuando los datos se vuelven a juntar en el destino.

02.4 Capas de TCP/IP

Capa 1: capa de acceso a la red

La capa de acceso a la red, también conocida como la capa de enlace a los datos, gestiona la infraestructura física que permite a los ordenadores comunicarse entre sí por Internet. Esto abarca, entre otros elementos, cables Ethernet, redes

inalámbricas, tarjetas de interfaz de red y controladores de dispositivos en el ordenador.

La capa de acceso a la red también incluye la infraestructura técnica, como el código que convierte datos digitales en señales transmisibles, que hacen posible una conexión.

Capa 2: Capa de Internet

La capa de Internet, también llamada la capa de red, controla el flujo y el enrutamiento de tráfico para garantizar que los datos se envían de forma rápida y correcta. Esta capa también es responsable de volver a juntar el paquete de datos en el destino. Si hay mucho tráfico en Internet, esta capa puede tardar un poco más en enviar un archivo, pero es menos probable que el archivo se dañe.

Capa 3: Capa de transporte

La capa de transporte es la que proporciona una conexión de datos fiable entre dos dispositivos de comunicación. Es como enviar un paquete asegurado: la capa de transporte divide los datos en paquetes, confirma los paquetes que ha recibido del remitente y se asegura de que el destinatario confirme los paquetes recibidos por su parte.

Capa 4: Capa de aplicaciones

La capa de aplicaciones es el grupo de aplicaciones que permite al usuario acceder a la red. Para la mayoría de nosotros, esto significa el correo electrónico, las aplicaciones de mensajería y los programas de almacenamiento en la nube. Esto es lo que el usuario final ve y con lo que interactúa al recibir y enviar datos.

03. Unity

03.1 ¿Qué es?

Unity es una plataforma de desarrollo 3D en tiempo real para compilar aplicaciones 2D y 3D, como juegos y simulaciones, con .NET y el lenguaje de programación C#.

Permite crear videojuegos habilitados para web, móviles, consolas, smart TV e incluso dispositivos de realidad aumentada, de manera que es un motor de desarrollo prácticamente universal que ahorra costos significativamente a programadores y desarrolladores independientes.

03.2 Versiones de Unity

Unity 3.5

Esta versión añade características como pathfinding, renderización en calidad HDR y multihilo, evasión de obstáculos, sondas de luz, perfilador de GPU, lightmaps direccionales, entre otras varias.

Unity 4

Lanzada en 2012, implementa Adobe Flash Player y actualiza las herramientas anteriores, mejorando también la interfaz gráfica. Añade sombras en tiempo real, creación de instancias, mapa interactivo, malla de piel y soporte de texturas 3D.

Unity 4.3

Mejora la animación con la herramienta Dopesheet para la creación de animaciones faciales.

Unity 5.6

Presenta la mejora de 2D Toolkit, implementación de Vulkan y mejoras en realidad virtual y realidad aumentada.

03.3 Características principales

Motor Gráfico: Ofrece un motor gráfico potente capaz de renderizar gráficos 2D y 3D con alta fidelidad, permitiendo la creación de entornos visuales complejos y detallados.

Interfaz de Usuario Intuitiva: Unity cuenta con una interfaz gráfica de usuario (GUI) amigable que facilita la manipulación de objetos, escenas y componentes, incluso para aquellos con poca experiencia en desarrollo.

Component-Based Architecture: Utiliza una arquitectura basada en componentes, donde los desarrolladores pueden añadir diferentes componentes a los objetos del juego para definir su comportamiento y propiedades.

Lenguajes de Programación: Principalmente utiliza C# como lenguaje de programación, proporcionando un entorno robusto y flexible para escribir scripts y lógica del juego.

Multiplataforma: Permite desarrollar y exportar aplicaciones para una amplia variedad de plataformas, incluyendo Windows, macOS, Android, iOS, WebGL, y consolas de videojuegos.

Asset Store: Cuenta con una tienda de activos (Asset Store) donde los desarrolladores pueden encontrar y comprar recursos como modelos 3D, texturas, plugins, y herramientas adicionales que facilitan el desarrollo.

04. Tecnología Wifi

04.1 Fundamentos Técnicos del WiFi

El Wifi es una tecnología que se ha establecido hoy como una de las herramientas más importantes en comunicación, para nuestra vida diaria y en entornos industriales, facilitando la conexión de los dispositivos en redes locales y a internet sin utilizar cables. En este capítulo se tratarán los fundamentos técnicos sobre el wifi.

Principios del Funcionamiento del Wifi

El Wifi es una tecnología que permite la conexión inalámbrica entre los dispositivos de una red. Esta conectividad se basa en ondas de radio. Se utilizan dos frecuencias distintas, 2.4 GHz y 5GHz.

Para el correcto funcionamiento se necesita un router, este está conectado mediante cables a internet, pero gracias a él los dispositivos se pueden conectar a internet. Para tener una buena conexión entre los dispositivos y el router los dispositivos deberán estar al alcance de la señal del router.

04.2 Componentes de una Red WiFi

Para comprender como opera una red wifi es fundamental conocer los componentes que la conforman, En esta sección se mencionaran y explicaran los componentes necesarios para una red wifi.

Modém: se encarga, de interpretar y transformar la señal que le llega, y proporciona una salida mediante un cable estándar RJ45, que es el comúnmente utilizado para realizar el cableado de red. Este cable ya podría conectarse directamente a un PC y tendríamos conexión a internet.

Router: Se encarga de la conexión de los múltiples dispositivos que podemos tener conectados: portátiles, ordenadores de sobremesa, tablets, smartTVs, impresoras, servidores, NAS (Newtork Attached Storage, o Almacenamiento Conectado en Red), smartphones, radios IP, cámaras IP, consolas de videojuegos, etc.

Extensor de Red: Cuando la señal generada por el router no es suficiente nos vemos en la necesidad de utilizar un dispositivo que amplifique la señal.

Punto de Acceso: Un punto de acceso es un dispositivo que crea una red de área local inalámbrica (WLAN, o Wireless Local Area Network). Se emplean para gestionar el acceso WiFi de un gran número de equipos de forma óptima, pues pueden gestionar unas 60 conexiones simultáneas por dispositivo (un extensor de red típico no es capaz de manejar más de unas 20 conexiones simultáneas, ya que no incrementa su ancho de banda.

04.3 Protocolos de Comunicación en WiFi

Los protocolos de comunicación son el esqueleto de la red, estos definen como los dispositivos deben intercambiar la información para tener un funcionamiento eficiente y confiable.

TCP/IP: El protocolo TCP descompone los datos en paquetes y los reenvía a la capa del protocolo de Internet (IP) para garantizar que cada mensaje llegue a su ordenador de destino. Esto ayuda a evitar problemas y a mantener la eficiencia durante el proceso.

UDP: Es un protocolo ligero de transporte de datos que funciona sobre ip, proporciona un mecanismo para detectar datos corruptos en paquetes, pero no intenta resolver otros problemas que surgen con paquetes, como cuando se pierden o llegan fuera de orden.

HTTP: HTTP es un protocolo de capa de aplicación en el modelo de comunicación de red de interconexión de sistemas abiertos (OSI). Define varios tipos de solicitudes y respuestas. Por ejemplo, cuando desea ver algunos datos de un sitio web, envía la solicitud HTTP GET. Si quiere enviar información, como rellenar un formulario de contacto, debe enviar la solicitud HTTP PUT.

MQTT: Se utiliza para la comunicación de un equipo a otro. Los sensores inteligentes, los dispositivos portátiles y otros dispositivos de Internet de las cosas (IoT) generalmente tienen que transmitir y recibir datos a través de una red con recursos restringidos y un ancho de banda limitado. Estos dispositivos IoT utilizan MQTT para la transmisión de datos,

04.4 Rendimiento y Cobertura del WiFi

Es fundamental comprender dos aspectos fundamentales del WiFi para que funcione de manera óptima y satisfaga nuestras necesidades de comunicación y acceso a Internet. Exploraremos en detalle los diversos aspectos que influyen en el rendimiento y la cobertura del WiFi.

Factores que afectan el rendimiento:

Colocación del dispositivo de difusión: Una línea de visión clara desde su dispositivo de transmisión (como un router, mesh nodo o extensor de rango) a sus dispositivos cliente proporciona la mejor señal WiFi. La intensidad de la señal disminuye a medida que pasa a través de paredes y otros objetos. Algunos objetos, como gabinetes de metal, pueden bloquear la señal por completo.

Interferencias de otros dispositivos: Muchas tecnologías habilitadas para WiFi y otros dispositivos domésticos utilizan la banda de 2,4 GHz, incluidos microondas, comandos de puertas de garaje y monitores para bebés. Algunos dispositivos WiFi, como los altavoces inteligentes, también utilizan la banda de 2,4 GHz para transferir audio en tiempo real. Cuando varios dispositivos intentan utilizar el mismo espacio de radio, se produce una aglomeración.

Uso compartido del ancho de banda: Compartir canales con sus vecinos, podría, por ejemplo, experimentar una velocidad de conexión más lenta durante aquellas horas en las que sus vecinos suelen acceder más a Internet.

Bucles de red: Un bucle de red se produce cuando una red tiene más de una ruta activa que transporta información desde el mismo origen al mismo destino. Puede crear un bucle de red conectando a través de Ethernet y WiFi al mismo tiempo o conectando dos cables Ethernet al mismo dispositivo.

Prácticas para optimizar la cobertura y calidad de la señal:

Evitar los obstáculos: Las paredes y los techos son obstáculos naturales para las ondas. El router inalámbrico no debería colocarse en el sótano, pero tampoco en

una habitación en medio de la casa. La ubicación ideal es cerca de las puertas, pues cualquier obstáculo, incluso la pared más fina, puede degradar la señal de la red.

Alineación óptima: Muchos routers tienen tres antenas externas que pueden mejorar la señal de la red de forma significativa. Una adecuada alineación de estas es clave: la primera antena debería estar completamente vertical, la segunda, hacia adelante en posición horizontal y la tercera, hacia un lado en posición horizontal. Todas las antenas deben estar orientadas en distinta dirección.

Tener en cuenta la altura: Colocar el router en un lugar elevado es fundamental para que las ondas se transmitan sin interferencias. Las estanterías o los armarios son perfectos en este sentido, salvo aquellos que estén cerca de objetos de metal.

Evitar fuentes de interferencias: Otras fuentes de interferencias son los dispositivos electrónicos como los teléfonos inalámbricos, los microondas e incluso las lavadoras, ya que transmiten ondas de radio en la misma banda de frecuencia, la de 2,4 GHz, que algunos routers. Para evitar estas interferencias, los usuarios pueden utilizar routers que operen en la banda de 5 GHz.

Utiliza repetidores: En casas unifamiliares con varias alturas, los repetidores inalámbricos son una estupenda opción. Reciben la señal wireless y la extienden a otras áreas de la casa en las que antes no había cobertura.

05. Arduino

05.1 ¿Qué es Arduino?

Arduino es una plataforma de código abierto que combina hardware y software fácil de manejar para construir proyectos electrónicos. Está compuesta por una placa de circuito programable (generalmente llamada microcontrolador) y un software o entorno de desarrollo integrado (IDE) que se instala en tu computadora para escribir y cargar el código en la placa física.

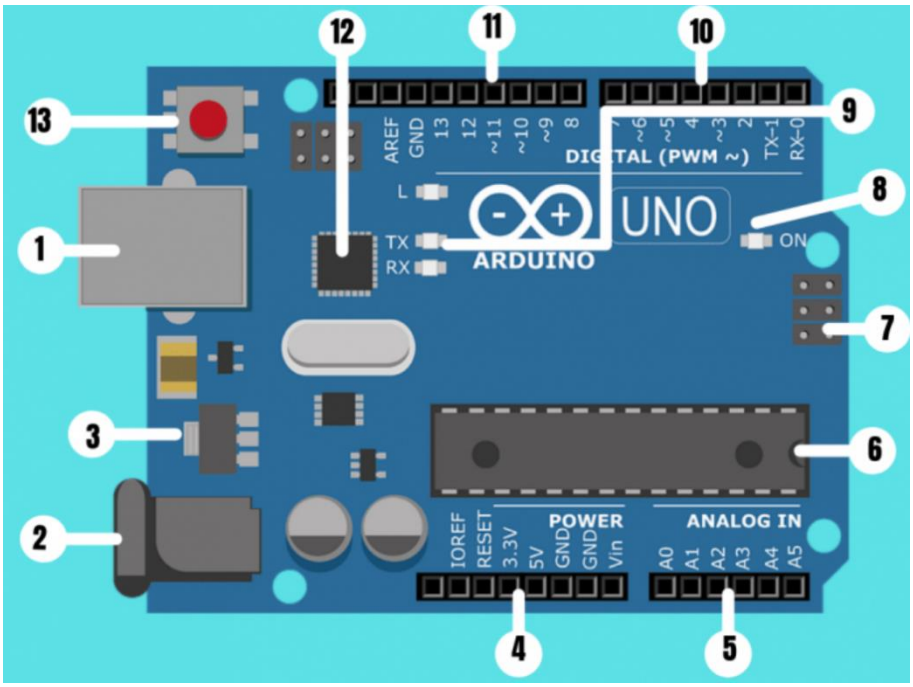
Las placas de Arduino son capaces de leer entradas (como la luz de un sensor o la pulsación de un botón) y convertirlas en salidas (como activar un motor o encender un LED).

05.2 ¿Cómo funciona?

La placa Arduino se conecta a un ordenador mediante un cable USB, permitiendo la interacción con el entorno de desarrollo Arduino (IDE). El usuario escribe el código en el IDE y luego lo sube al microcontrolador, que ejecuta el código e interactúa con las entradas y salidas, como sensores, motores y luces.

Su código abierto es muy amigable tanto para usuarios novatos como para expertos. Hay miles de ejemplos de código disponibles en línea que facilitan el aprendizaje y la experimentación.

05.3 Partes de la placa



1 y 2) Alimentación USB/5VDC

Arduino UNO puede ser alimentado desde un cable USB tipo B o mini conectado a tu ordenador, o desde una fuente de alimentación entre 6V y 18V. En la imagen de arriba, la conexión USB está etiquetada (1) y el conector de la fuente de alimentación (2).

Ademas, la conexión USB sirve para cargar código en la placa de Arduino desde donde se pueden enviar datos de la programación e instrucciones a la placa.

(3) Regulador de voltaje

Este componente controla la cantidad de voltaje que entra en la placa de Arduino, evitando que un voltaje excesivo dañe el circuito.

(4) Conexiones

Los pines de Arduino se utilizan para conectar los cables necesarios para construir un circuito. Este tipo de conexiones tiene varios pines, cada uno de los cuales está impreso en la placa y se utilizan para diferentes funciones:

- Reset: Permite el reinicio del microcontrolador.
- 5V y 3.3V: la clavija de 5V suministra 5 voltios de energía, y la clavija de 3.3V suministra 3.3 voltios de energía. La mayoría de los componentes simples usados con el Arduino funcionan bien con 5 o 3.3 voltios.
- GND: Hay varios pines GND en Arduino, se usan para conectar a tierra el circuito.
- VIN: Se usa para conectar la alimentación de la placa con una fuente externa de entre 6 y 12VDC.

(5) Puertos de entrada Analógicos

Estos pines bajo la etiqueta ‘Analog In’ (A0 a A5 en la UNO) pueden leer la señal de un sensor analógico y convertirla en un valor digital que podemos interpretar.

(6) Micro-controlador Atmega 328

Es el circuito integrado que actúa como cerebro o procesador de la placa de Arduino, donde se implementa la programación.

(7) Entrada ICSP (In Chip Serial Programmer)

Permite grabar programas en el circuito directamente desde el ordenador sin necesidad de utilizar el puerto USB.

(8) Indicador LED de alimentación

Este LED se enciende para indicar que el microprocesador está activo.

(9) LED TX RX

TX es la abreviatura de transmisión de datos y RX es la abreviatura de recepción de datos. Estas marcas comunes aparecen con regularidad en la electrónica para indicar los pines responsables de la comunicación serie. Estos LED se activan visualmente cuando la placa recibe o transmite datos.

(10) Puertos Digitales

Estos pines se pueden utilizar tanto para la entrada digital (como para detectar la pulsación de un botón) como para la salida digital (como para alimentar un LED).

(11) Puerto de conexiones

- 5 entradas o salidas auxiliares (de la 8 a la 12).
- 3 salidas 9, 10 y 11 que permiten la modulación por ancho o de pulso.
- Salida 13 que sirve para conectar un led directamente a tierra.
- Salida a tierra GND.
- Pin AREF que se utiliza para fijar una tensión de referencia externa (entre 0 y 5 voltios) como límite superior de las clavijas de entrada analógica.

(12) Chip de Arduino

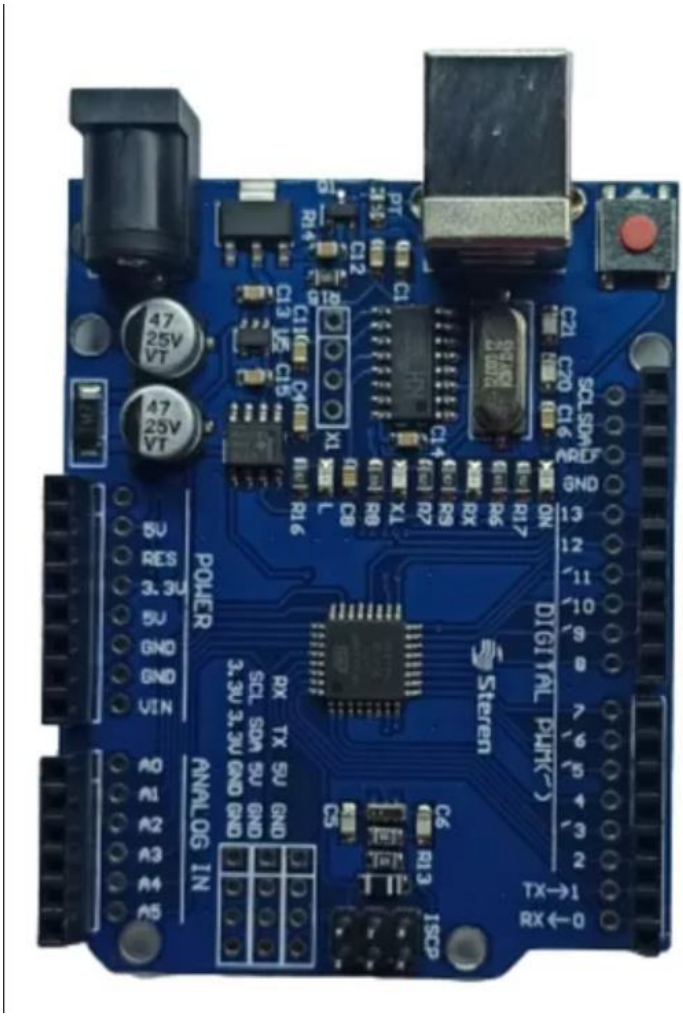
Permite identificar un dispositivo USB por el ordenador, es como su tarjeta de identificación o D.N.I. personal

(13) Botón de RESET

Al presionarlo conectará temporalmente el pin de reset a tierra y reiniciará cualquier código que esté cargado en el micro-controlador de Arduino.

06.Componentes necesarios para el proyecto

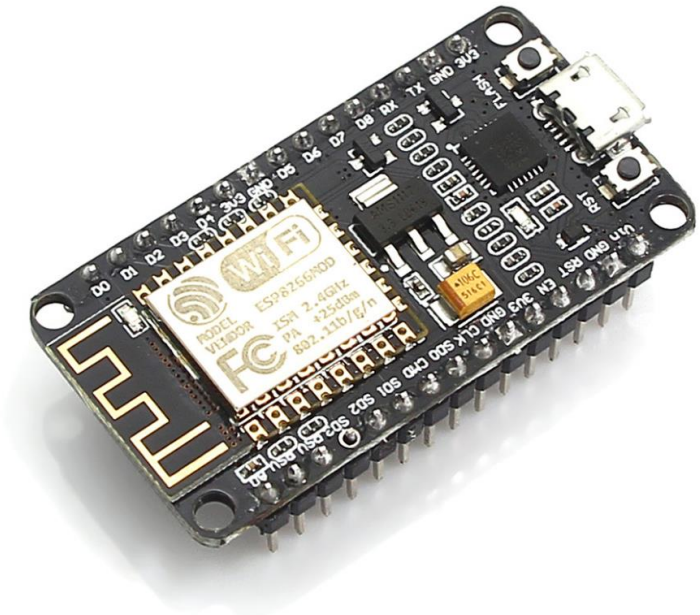
Arduino UNO



Arduino es una plataforma de desarrollo basada en una placa electrónica de hardware libre que incorpora un microcontrolador re-programable (ATMega328P) y una serie de pines hembra que permiten establecer conexiones.

En el montacargas su función es coordinar y controlar varios componentes, a el están conectados el módulo de relés para la solicitud de movimiento hacia los motores, al puente H para la solicitud de movimiento de los actuadores, al módulo ESP 8266 y será alimentado por una PowerBank.

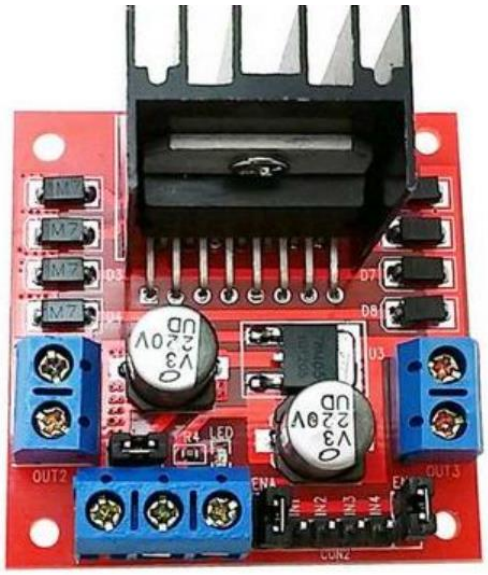
ESP 8266



La función del módulo ESP 8266 es facilitar la comunicación inalámbrica, porque permite la conexión WiFi. En el montacargas estará configurado como cliente, de esta manera recibirá comandos para mover el montacargas de forma remota.

El lenguaje de programación que se utiliza el ESP8266 es el lenguaje de programación C++.

Puente H



Para controlar el movimiento de los actuadores se utilizó un puente H y Arduino, su principal función es permitir que los motores de engranaje (encargados del movimiento de los actuadores) giren en ambas direcciones.

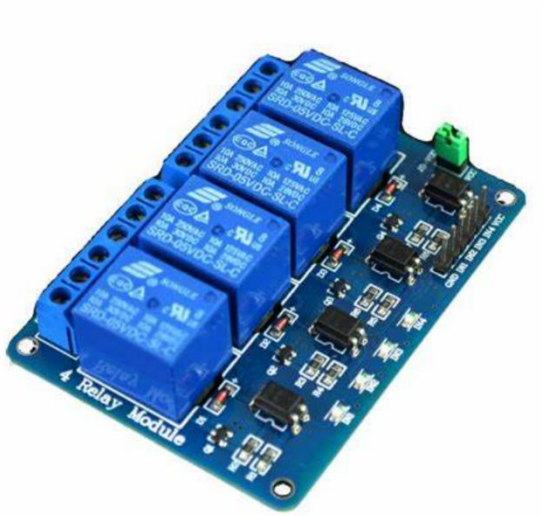
A través de la placa Arduino se envían comandos para controlar ese movimiento.

ESP-32 CAM

Combina una cámara OV2640, un microcontrolador ESP32 y una antena Wi-Fi.

En el montacargas fue implementada para tener visión en tiempo real, su imagen es enviada a una computadora para su visualización.

Módulo Relay



El módulo de relays en el montacargas se activa/desactiva a través de comandos programados en la placa Arduino, es alimentado por 6 baterias de iones de litio de 3.7V cada una.

Su principal función en el montacargas es controlar el movimiento de los dos motores asociados al sistema de transmisión, controla la dirección de giro de estos para poder realizar el movimiento hacia adelante, atrás, derecha, o izquierda, según sea el caso.

Baterías de Litio de 3,7V

Encargadas de la alimentación de los 6 motores.

PowerBank Adata 20000 mah

Encargada de alimentar la placa Arduino y la ESP-32 CAM.

Correa de distribución de goma de bucle cerrado de 10mm x 240mm

Correa de distribución de goma de bucle cerrado de 10mm x 250mm

Las correas de distribución en el modelo del minicargador tienen la función de sincronizar el movimiento, están encargadas de transmitir potencia a las orugas del minicargador, por lo que su labor es crucial para el movimiento.

Motor cepillado 540 impermeable 45T para coche teledirigido

Motor que utiliza escobillas para transferir la energía eléctrica al rotor, son comunes en vehículos RC por su simplicidad y durabilidad.

45T indica el número de vueltas del motor, a mayor número de vueltas mayor torque, pero menor velocidad, para el minicargador elegimos 45T por su equilibrio entre estas dos características.

Printfly-polea de sincronización

También conocidas como poleas síncronas tienen dientes alrededor de su circunferencia, ideales para nuestras correas de sincronización que se enganchan a estos dientes para poder realizar su función con precisión y fuerza.

Mini Motor de engranaje N20 DC 3V-12V de velocidad lenta

Motor de engranaje, específicamente motor de reducción de velocidad, diseñados para proporcionar un alto par de torsión a velocidades más lentas Los motores que

son parte del minicargador son de 60RPM para el funcionamiento de los actuadores, sus características son:

- Voltaje nominal: DC 5V
- Corriente sin carga: 50mA
- Velocidad sin carga: 60RPM
- Voltaje adecuado: DC 3-9V
- Velocidad: 30RPM-108RPM

Le brindan al minicargador un movimiento preciso.

Tornillos y tuercas

- M2x6 - 35 piezas
- M2x6 - 12 piezas (se recomienda cabeza hueca, pero se puede utilizar cualquier tipo de cabeza)
- Rodamientos 5x10x4 - 28 piezas
- M2x8 - 4 piezas
- Rodamientos 8x16x5 - 4 piezas
- Rodamientos 15x24x5 - 4 piezas
- M2x10 - 8 piezas
- M3x5 - 3 piezas (cabeza hueca)
- M3x8 - 6 piezas
- Eje roscado M3x20mm – 2uds
- M3x10-12-2uds
- M3x12-20uds

- Varilla de acero 8x75mm – 2uds
- Varilla de acero 5x50mm – 2uds
- M3x16- 24 piezas M3x20 - 18 piezas
- M3x25 - 4 piezas
- M4x16-4 piezas
- M4x20 –4 piezas
- M4x25 - 6 piezas
- M5x25 - 2 piezas
- M5x45 - 12 piezas
- Tuerca M3 - 8 piezas
- Contratuerca M5 - 4 piezas
- Arandela M5 - 8 piezas

07. Piezas y ajustes de impresión

Leyenda: L = izquierda, R = derecha, FL = delantera izquierda, FR = delantera derecha, RL = trasera izquierda, RR = trasera derecha, CL = central izquierda, CR = central derecha

Muchos detalles del lado derecho son espejos del lado izquierdo. Si el nombre de la parte está coloreado en rojo, entonces es una copia reflejada de la parte del lado izquierdo.

Nombre de Parte	Pza	Ajustes	g	m	Color
CHASSIS					
SSL Chassis F	1	Paredes 6, relleno 20%, Borde 6 líneas, Altura de capa 0,2, Optimizar orden de impresión de paredes, sin soportes	99	33.2	
SSL Chassis side cap L	1				
SSL Chassis side cap R	1				
SSL Chassis R	1	Paredes 6, relleno 20%, Borde 6 líneas, Altura de capa 0.2, Optimizar el orden de impresión de la pared, Ángulo de voladizo de soportes (aquí y en todas las partes): 70 grados, Densidad de soporte 7%, Habilitar techo de soporte (algunos soportes bloqueados)	267	89.7	
SSL Bearing holder 1	2	Paredes 6, relleno 20%, Borde 6 líneas, Altura de capa 0,2,	27	8.9	
SSL Bearing holder 2	2				
SSL Bearing holder 3	2				
SSL Bearing holder 4	1				
SSL Tensioner wheel	4				
SSL Pulley spacer	2				
Pulley 100T(bore **5mm**, width 10mm)	2	Paredes 6, relleno 20%, Borde 6 líneas, Altura de capa 0,2,	137	45.8	
Pulley 100T(bore **8mm**, width 10mm)	2				
SSL Track body L	1	Paredes 4, capas inferiores 6, capas superiores 6, relleno 20%, Borde 6 líneas, Altura de capa 0,2,	216	72.4	
SSL Track body R	1				
SSL Tracks Road wheel small	8	Paredes 6, relleno 20%, Borde 6 líneas, Altura de capa 0,2,	224	75.2	
SSL Tracks Road wheel big	2				
SSL Tracks Idler wheel	2				

SSL Tracks Idler wheel tube	2				
SSL Tracks drive sprocket	2				
SSL Tracks drive sprocket hub	2				
SSL Tracks drive sprocket cap	2				
SSL Track (print with **FLEX** filament)	2	Paredes 4, relleno 15%, Borde 6 líneas, Altura de capa 0,2	196	66	
SSL Motor space cap	1	Paredes 4, relleno 10%, Borde 4 líneas, Altura de capa 0,2, sin soportes	51	17	

BODY					
SSL Body F	1	Paredes 6, relleno 20%, Borde 6 líneas, Altura de capa 0,2+-0,04 (capas adaptables), sin soportes	71	24	B
SSL Body RL	1	Paredes 6, relleno 20%, Borde 6 líneas, Altura de capa 0,2+-0,04 (capas adaptables), sin soportes	189	63.4	B
SSL Body RR	1				B
SSL Body frame rear	1	Paredes 6, relleno 20%, Borde 6 líneas, Altura de capa 0,2	33	11.2	B
SSL Body frame top	1				B
SSL Body top cap	1	Paredes 4, relleno 10%, Borde 6 líneas, Altura de capa 0,2+-0,04 (capas adaptables)	35	11.7	
SSL Body corner part RL	1				
SSL Body corner part RR	1				
SSL Body side grill L	1				
SSL Body side grill R	1				
SSL Tailgate	1	Paredes 4, relleno 10%, Borde 6 líneas, Altura de capa 0,2+-0,04 (capas adaptables)	62	20.7	
SSL Tailgate C-hook	1				

CAB					
SSL Cab	1	Paredes 4, relleno 10%, Borde 6 líneas, Altura de capa 0,2+-0,04 (capas adaptables), sin soportes	205	68.8	
SSL Seat	1	Paredes 4, relleno 10%, Borde 6 líneas, Altura de capa 0,2, sin soportes	34	11.3	
SSL Cab C-hook	2		9	3	

SSL Top window grill	1	Paredes 4, relleno 10%, Borde 6 líneas, Altura de capa 0,2			
SSL Window grill L	1				
SSL Window grill R	1				

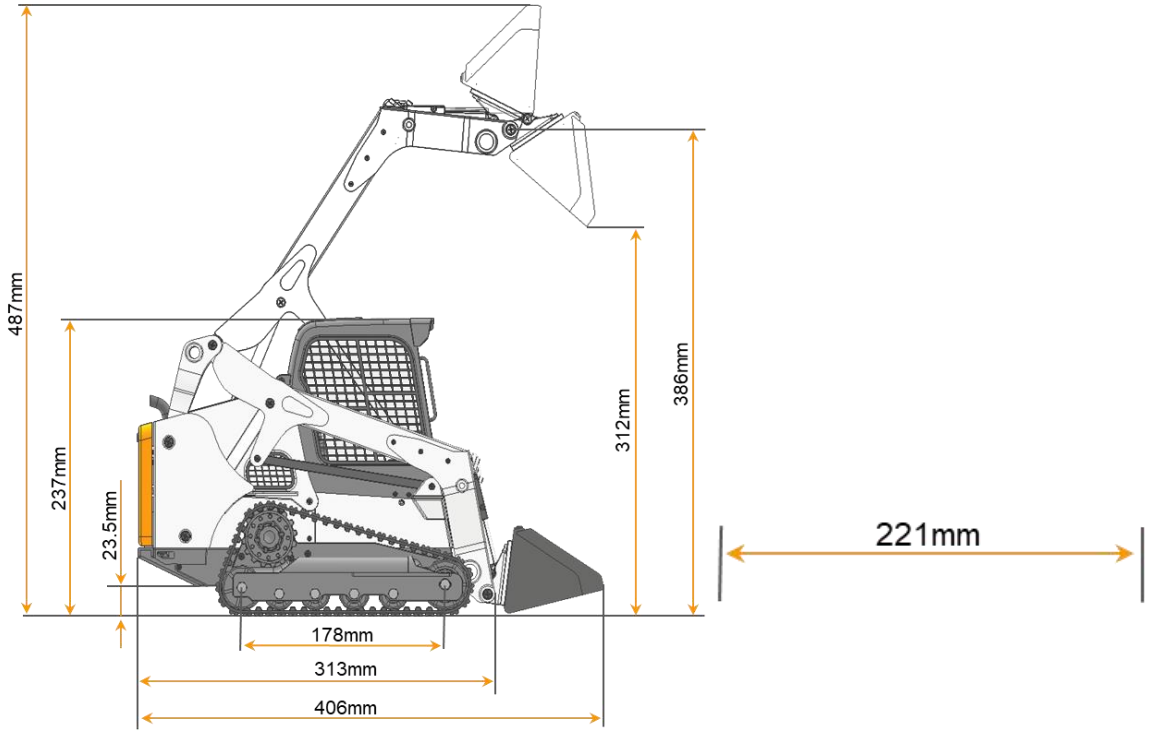
LIFT ARM					
SSL Lift arm F	1	Paredes 6, relleno 20%, Borde 6 líneas, Altura de capa 0,2	154	51.5	B
SSL Lift arm rear frame	1				B
SSL Lift arm pipes connectors	1				B
SSL Lift arm RL	1	Paredes 6, relleno 20%, Borde 6 líneas, Altura de capa 0,2	142	47.5	B
SSL Lift arm RR	1				B
SSL Lift arm link	2	Paredes 6, relleno 20%, Borde 6 líneas, Altura de capa 0,2	12	3.9	
SSL Bucket mount plate	1	Paredes 6, relleno 20%, Borde 6 líneas, Altura de capa 0,2	194	65	
SSL Bucket	1				

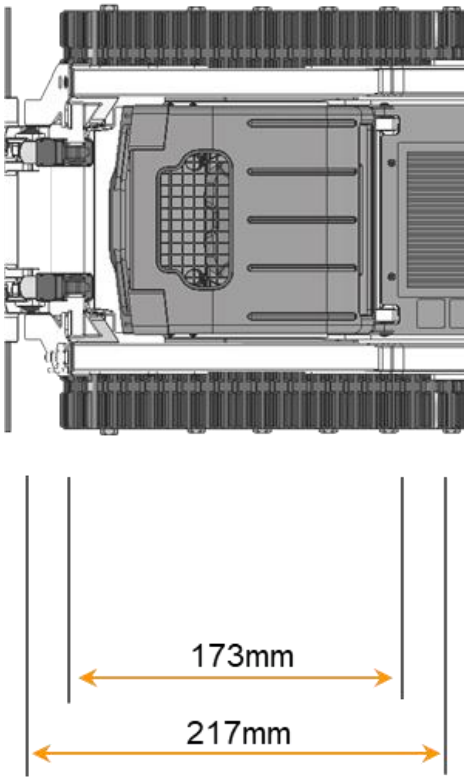
TOTAL, pza	74

2357	**Gramos**
790	**Metros**

08. Guía de montaje del Montacarguitas

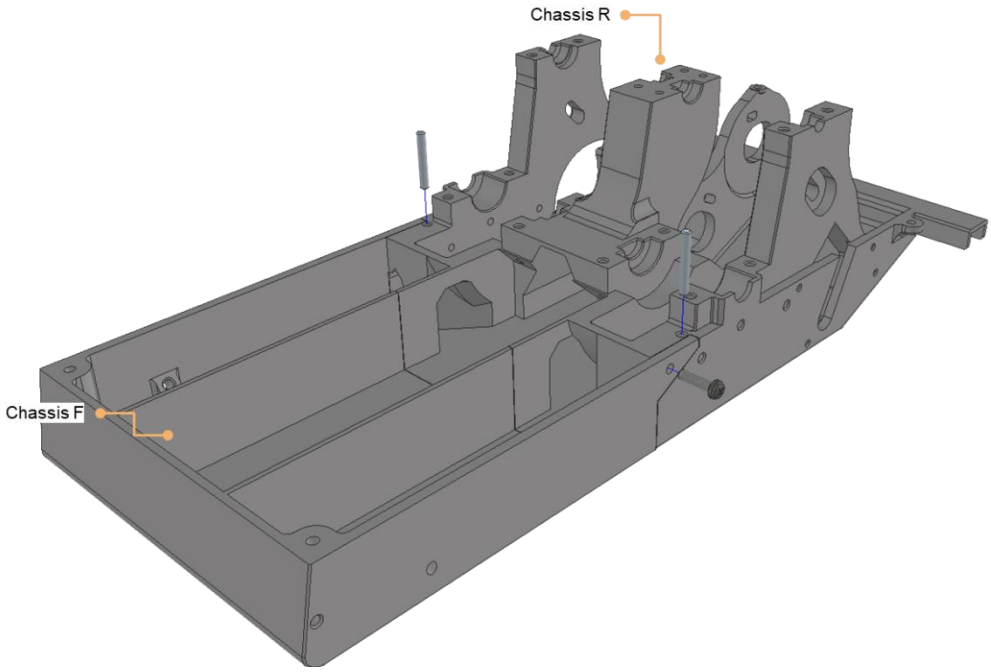
08.1 Medidas



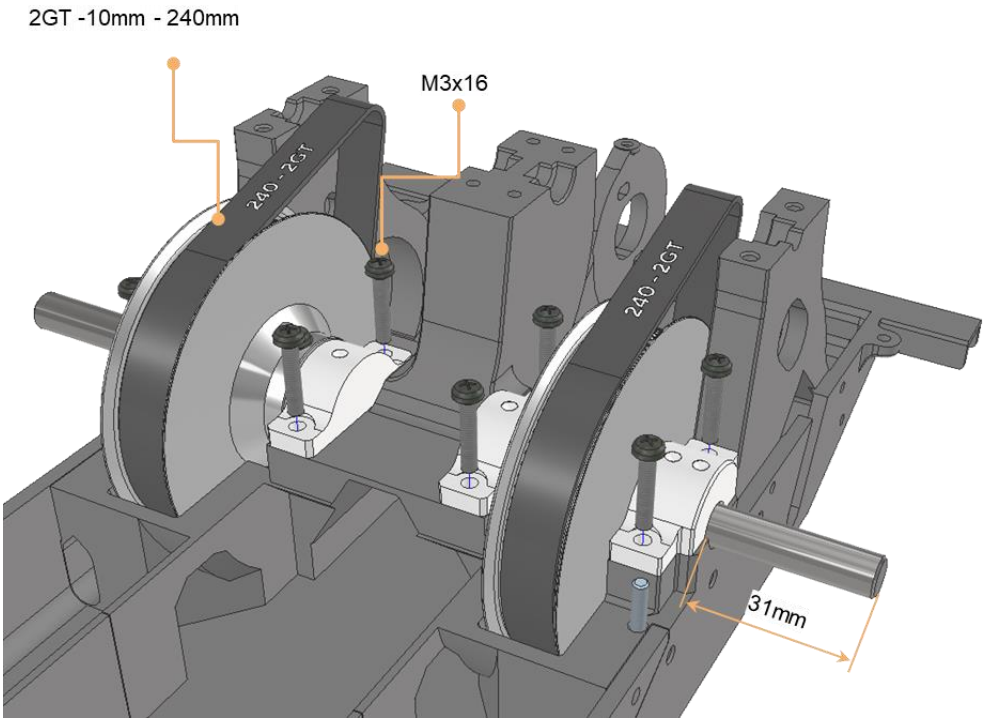


08.2 Guía de montaje

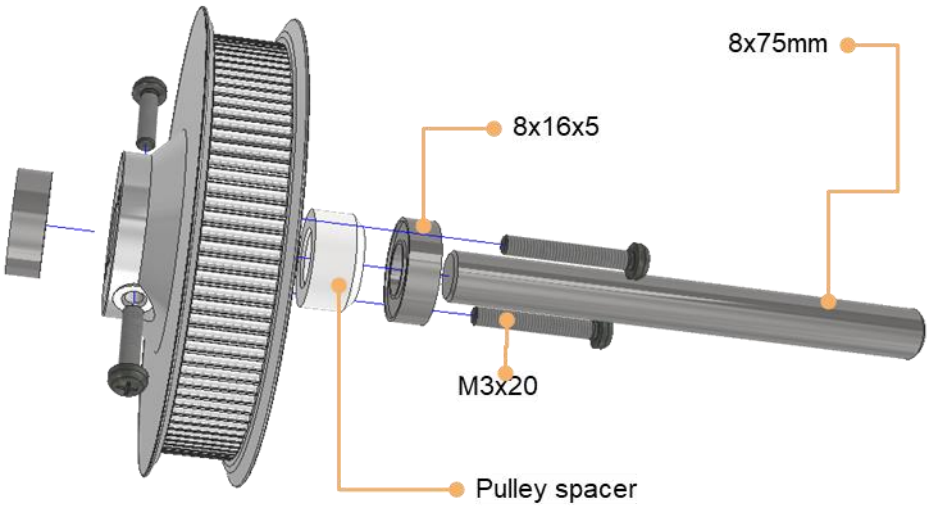
Paso 1:



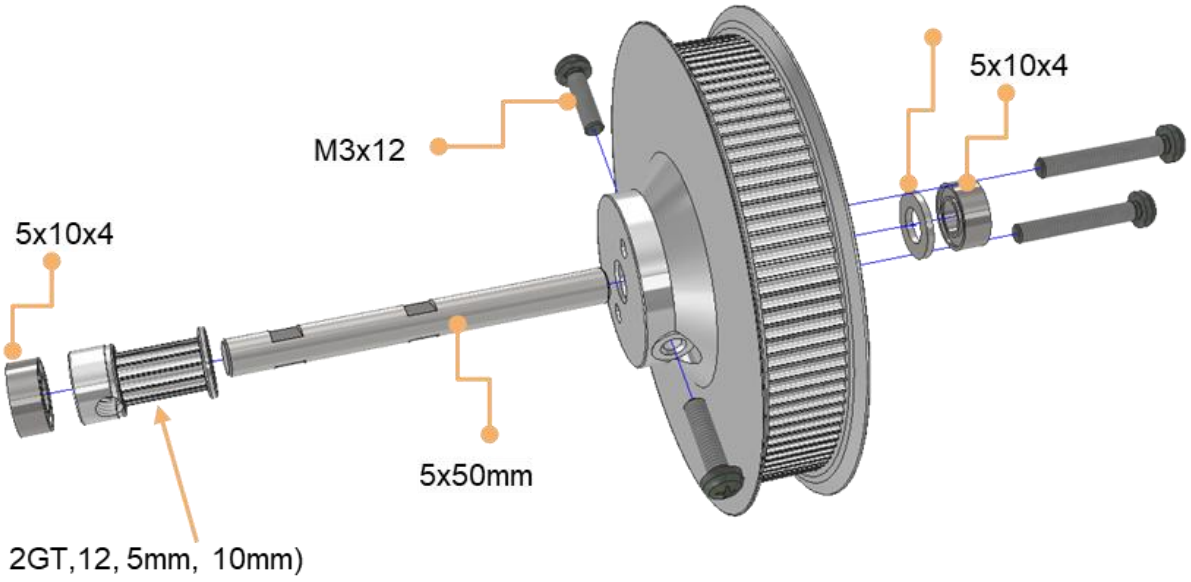
Paso 2:



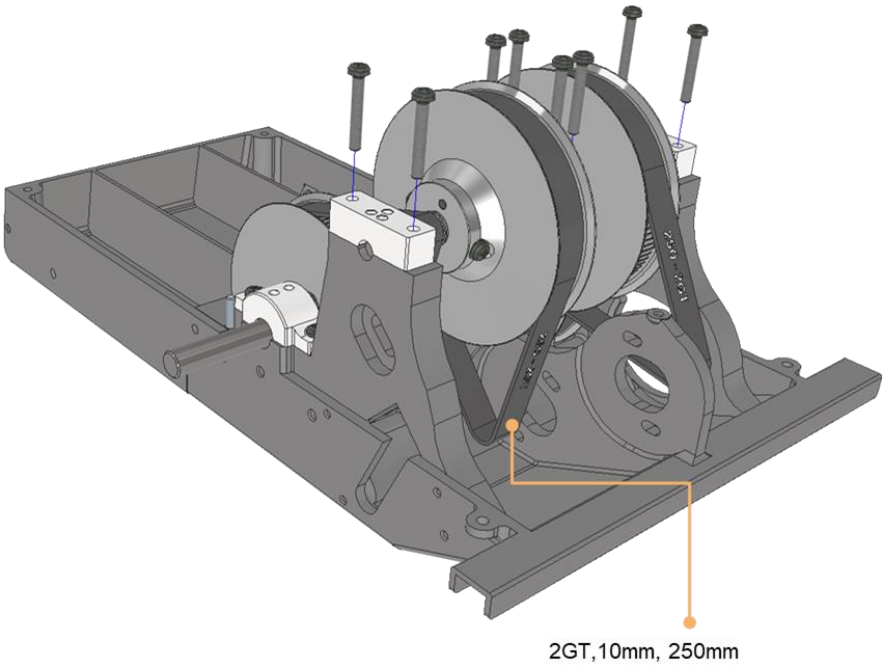
Paso 3:


8x75mm
8x16x5
M3x20
Pulley spacer

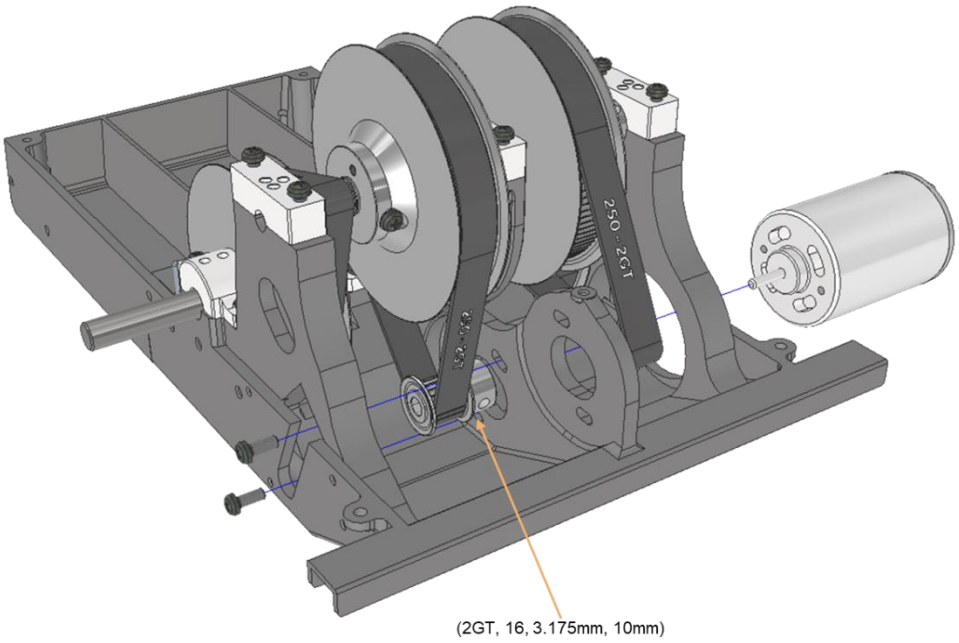
Paso 4:


5x10x4
M3x12
5x10x4
5x50mm
2GT,12, 5mm, 10mm)

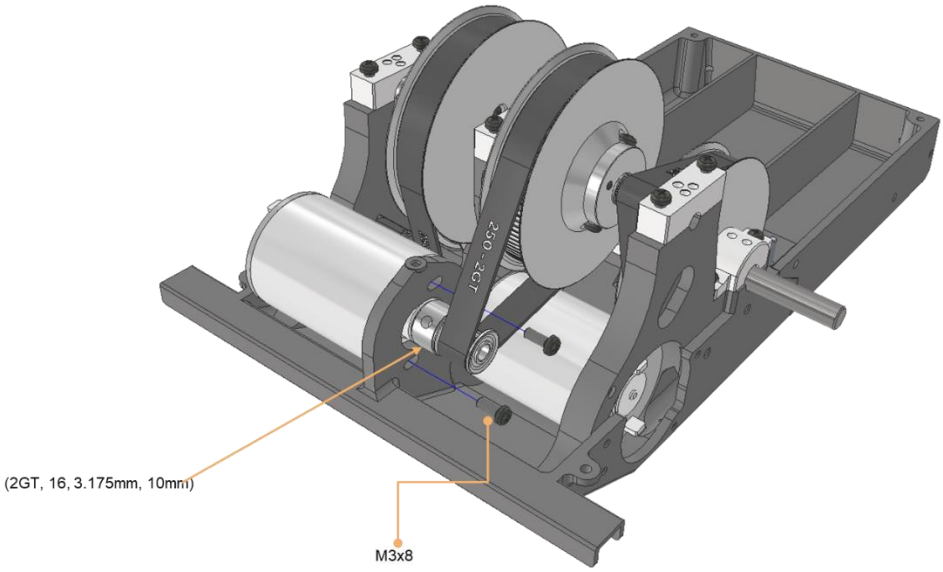
Paso 5:



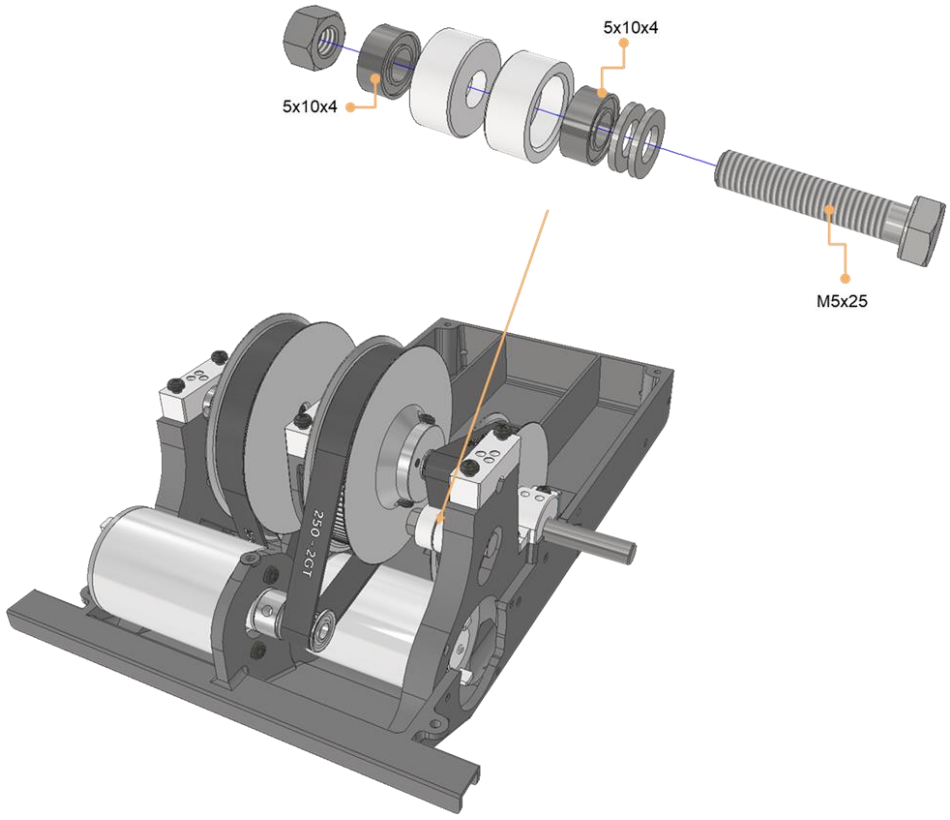
Paso 6:



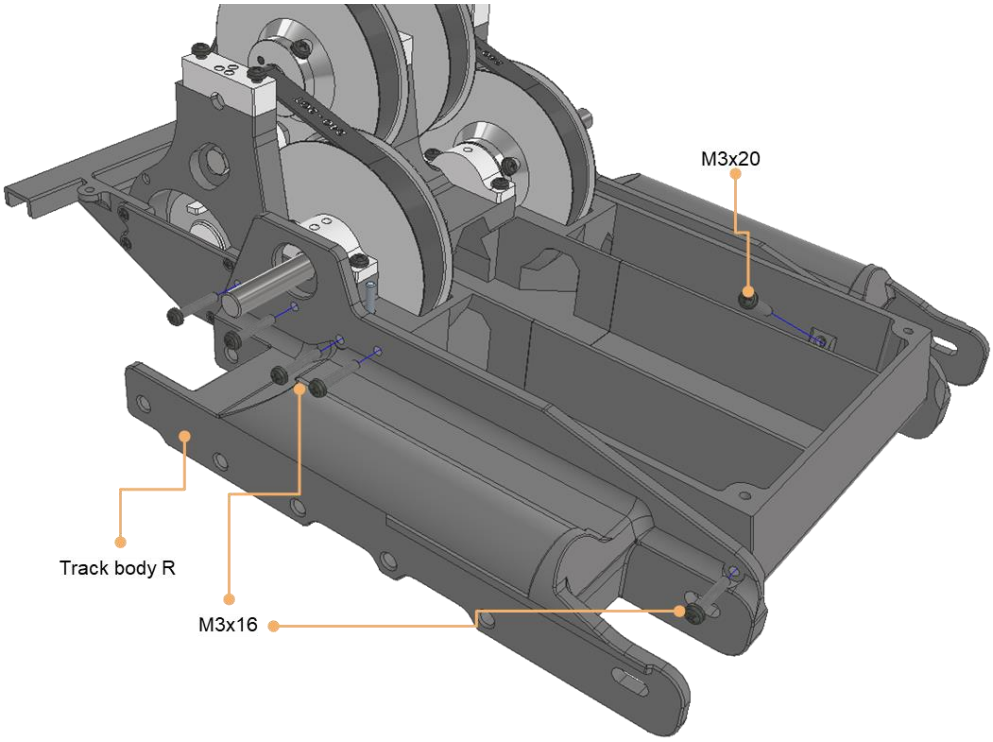
Paso 7:



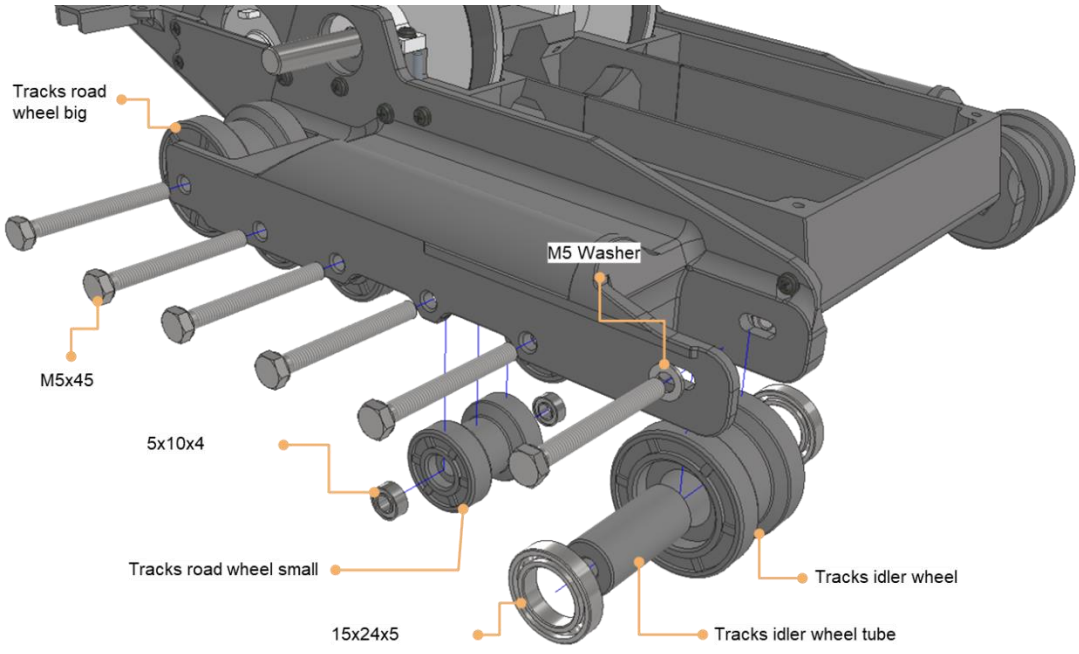
Paso 8:



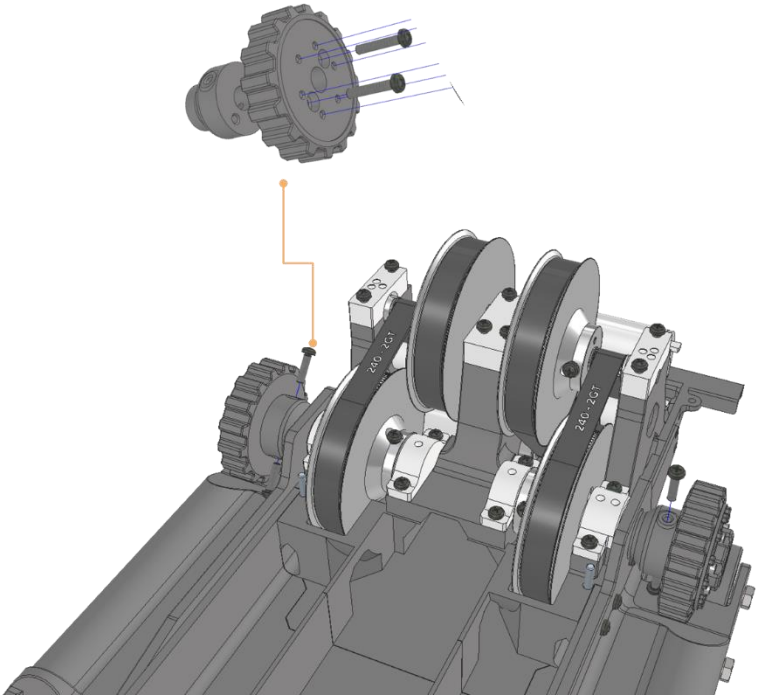
Paso 9:



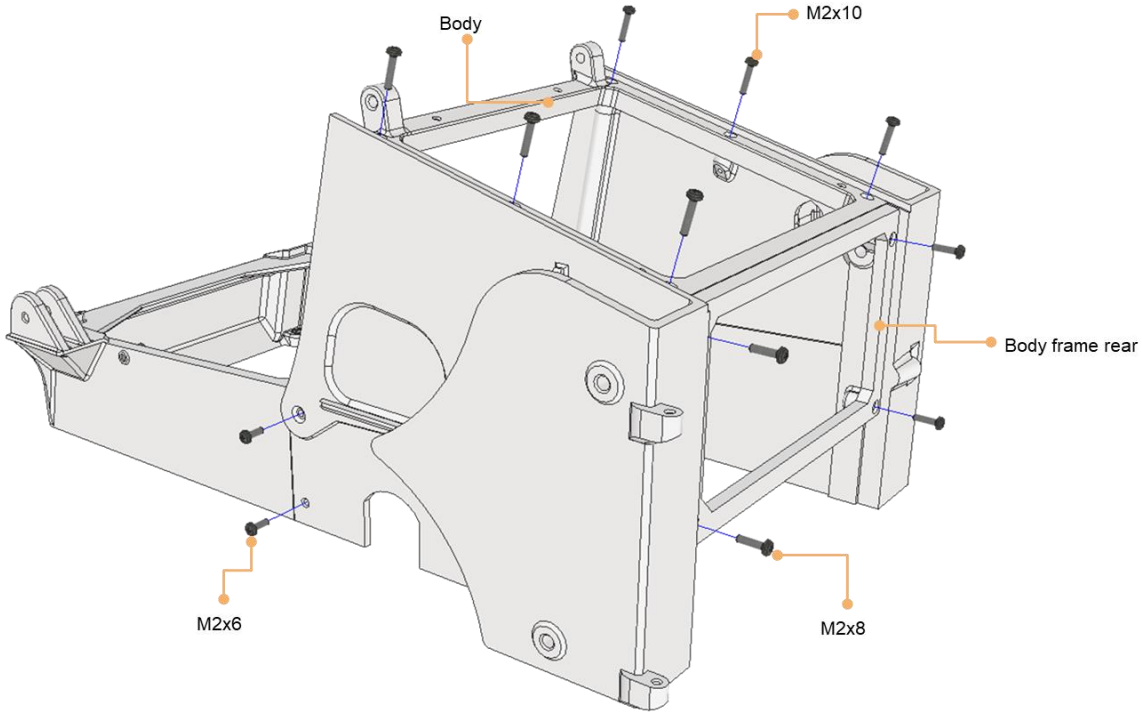
Paso 10:



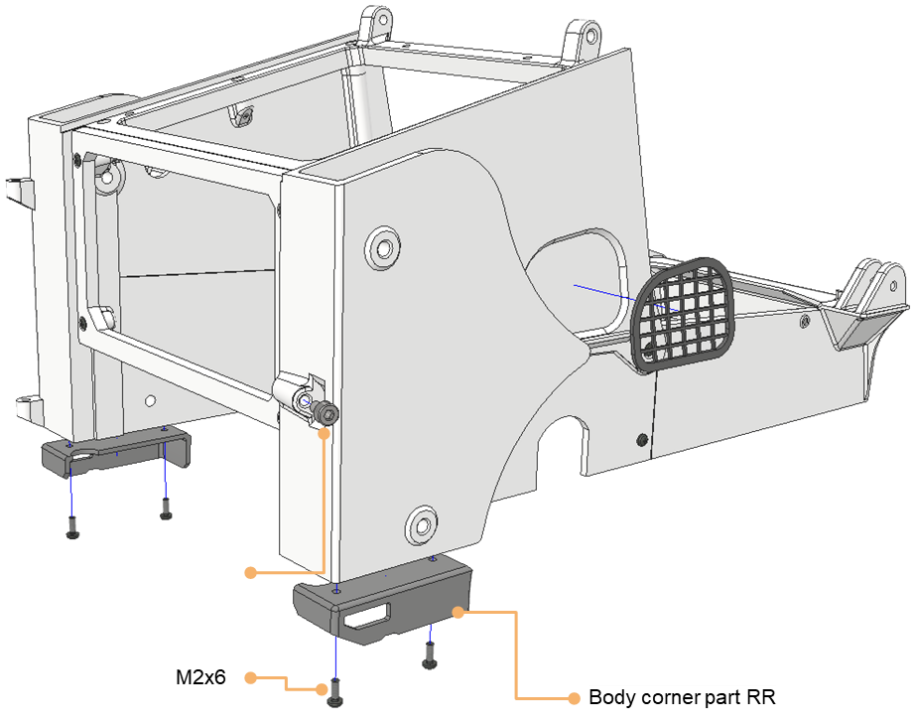
Paso 11:



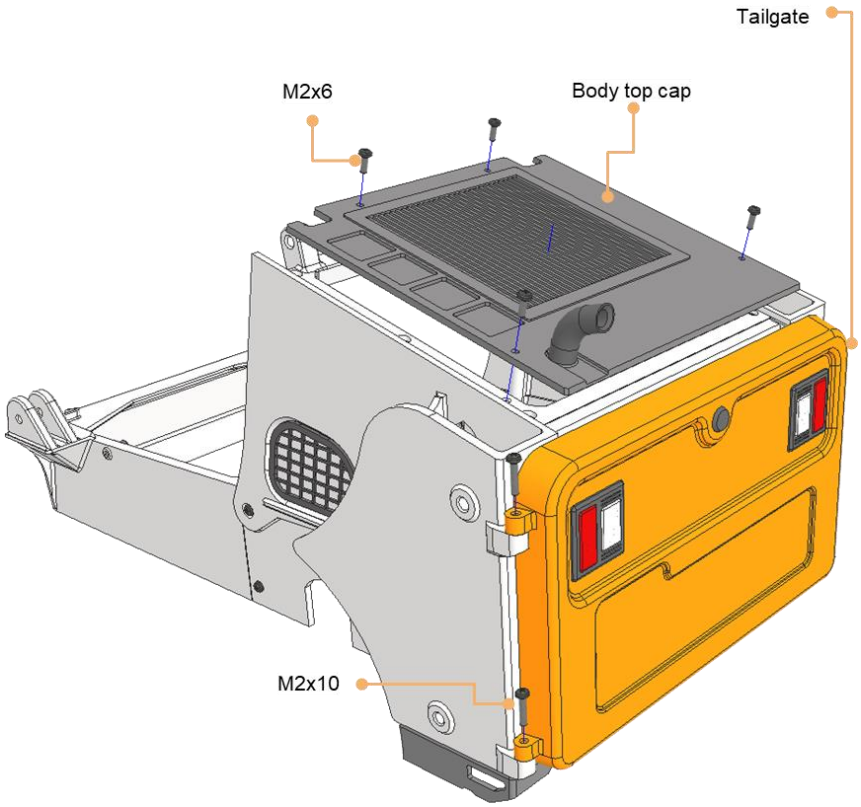
Paso 12:



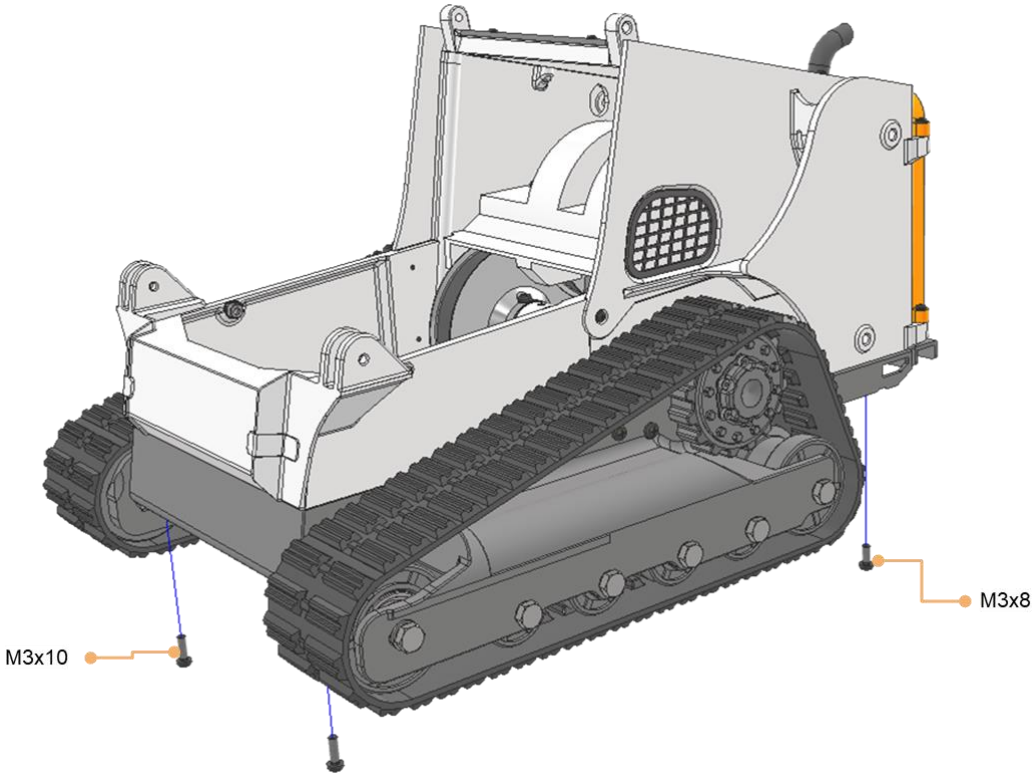
Paso 13:



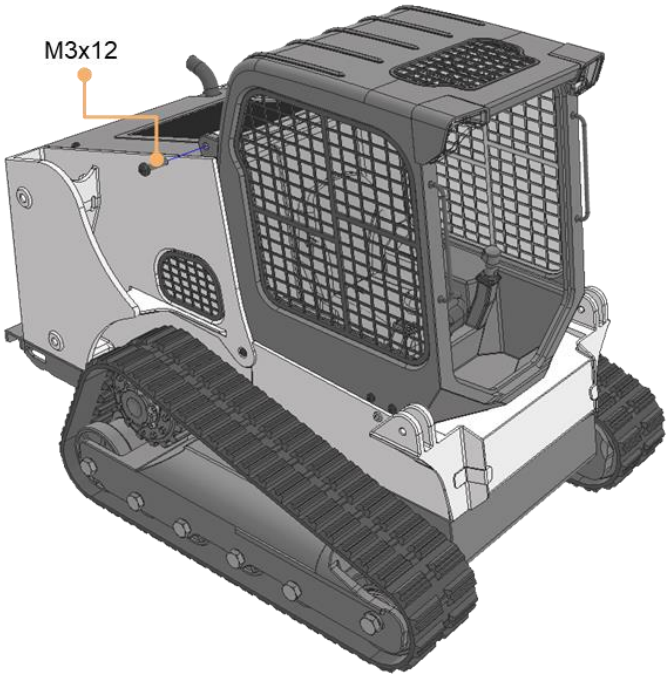
Paso 14:



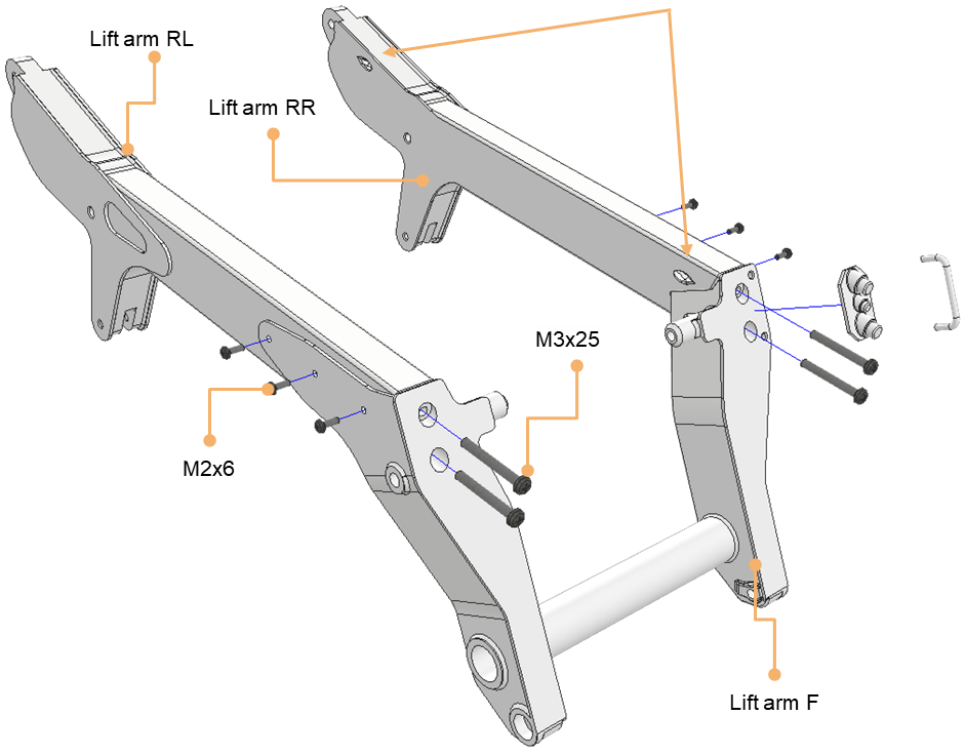
Paso 15:



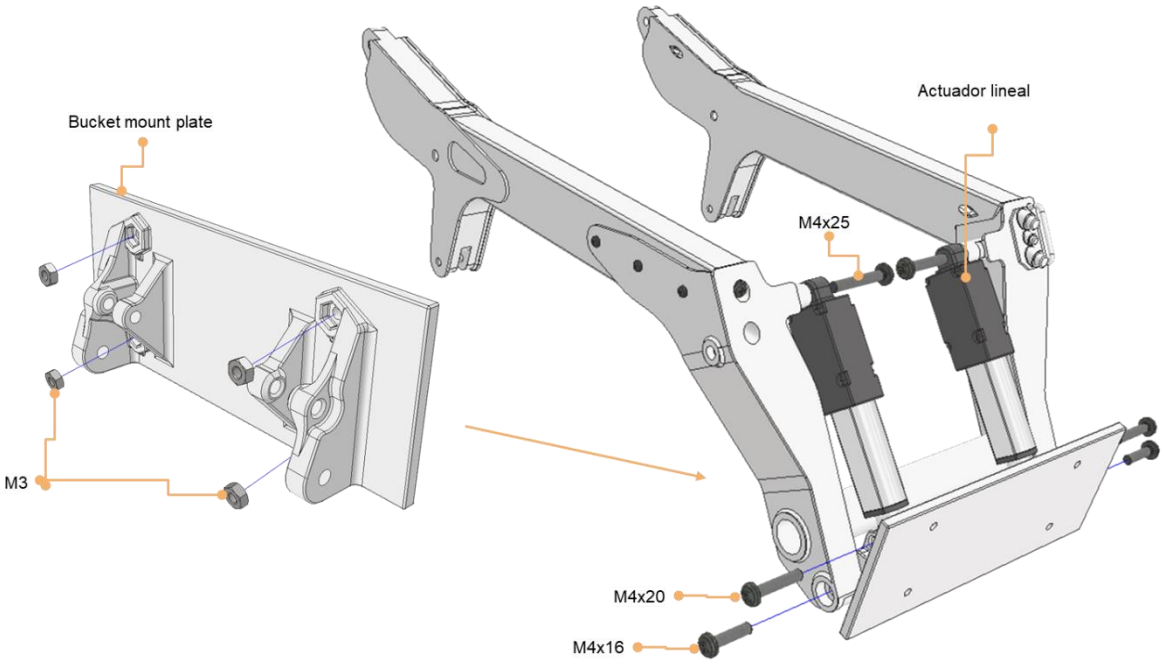
Paso 16:



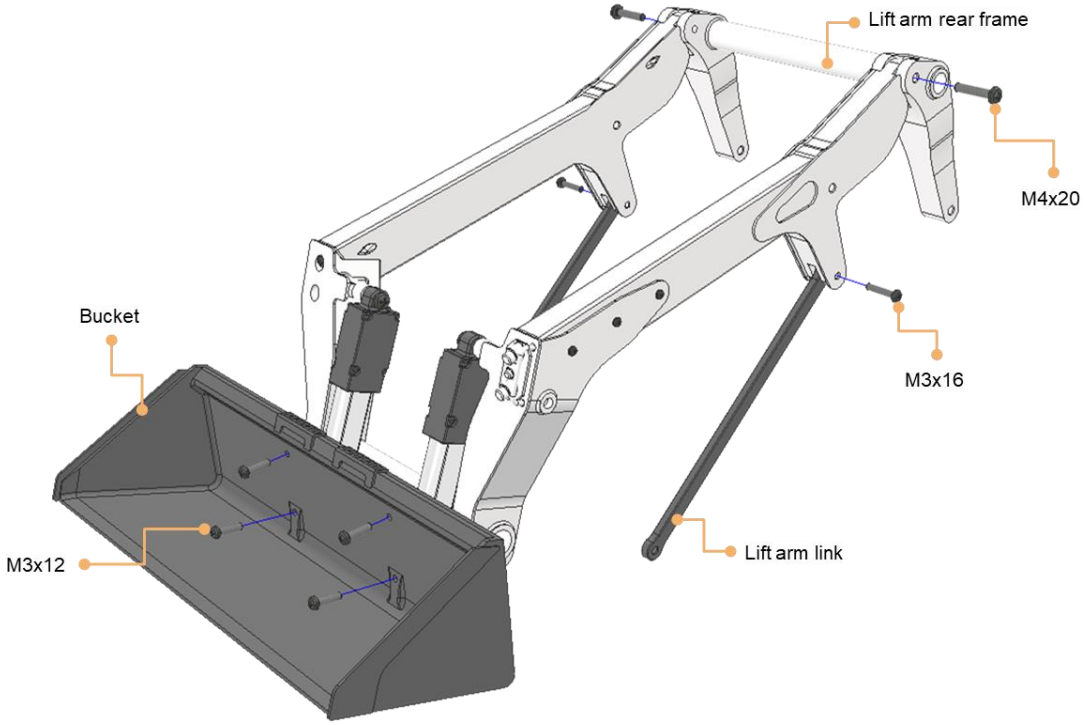
Paso 17:



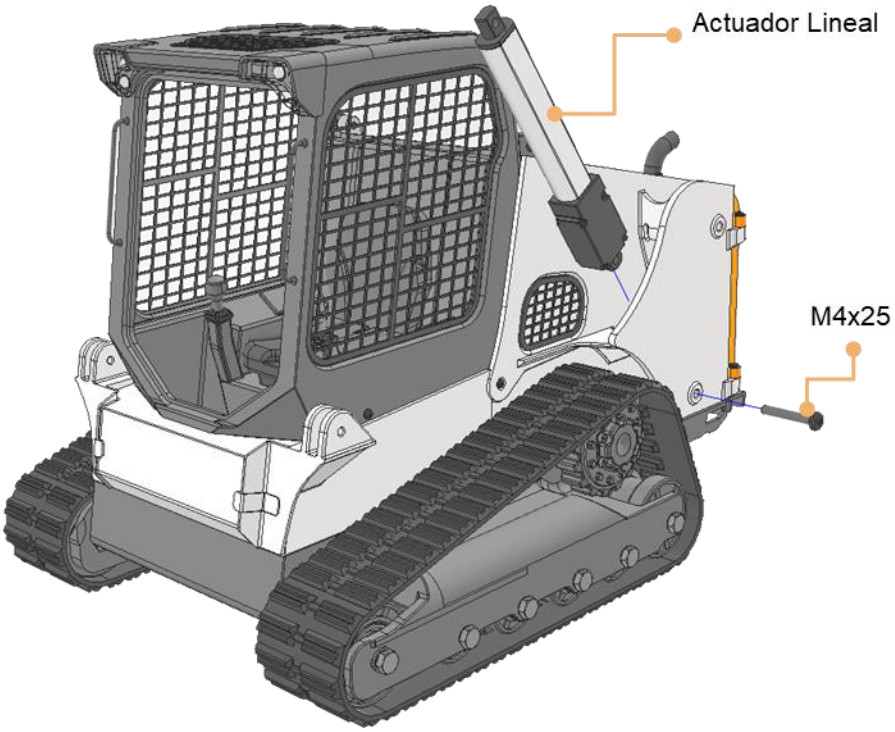
Paso 18:



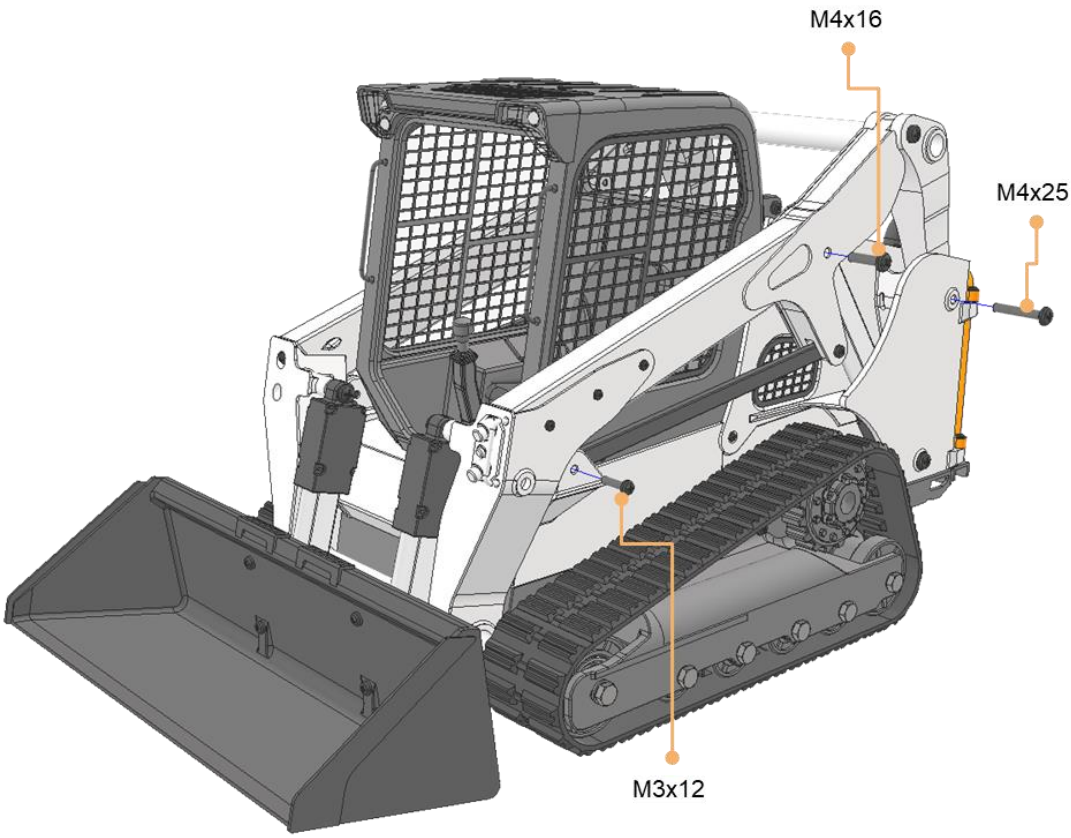
Paso 19:



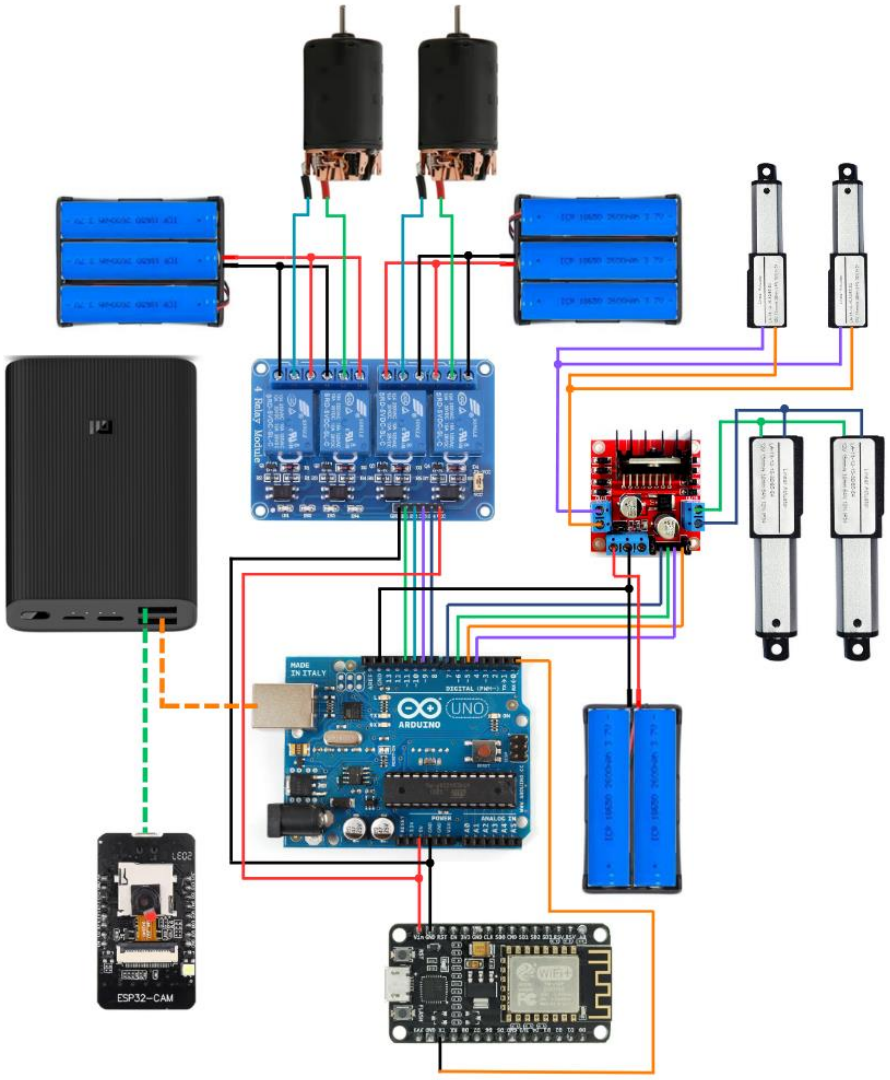
Paso 20:



Paso 21:



08.3 Diagrama eléctrico



09. Codificación.

09.1 Módulo ESP32 CAM

En este apartado se describirá brevemente el proceso de codificación que se tuvo que realizar para poder integrar la señal de video de un módulo ESP32 CAM en la plataforma Unity.

Para comenzar, es necesario aclarar que para poder llevar a cabo este proceso necesitaremos del entorno NodeJS, 3 lenguajes de programación, Arduino, JavaScript y C# , el lenguaje de etiquetas HTML y 3 IDE’s como lo son Arduino IDE, Visual Studio Code y Visual Studio 2022.

09.1.1 Server

Se inicia con la creación de un servidor, para la programación de este es necesario contar con NodeJS instalado además antes de programarlo necesitamos instalar las siguientes librerías con los siguientes comandos:

- npm install –save path
- npm install –save express
- npm install –save ws

```
JS server.js  ×  client.html
MiniCargador > JS server.js > ...
1  const path = require('path')
2  const express = require('express')
3  const webSocket = require('ws');
4  const app = express();
5
6  const WS_PORT = 8888;
7  const HTTP_PORT=8000;
8
9  const wsServer = new webSocket.Server({port: WS_PORT}, ()=>console.log(`WS server esta escuchando en ${WS_PORT}`));
10
11 let connectedClients=[];
12 wsServer.on('connection', (ws, req)=>{
13     console.log('Conectado');
14     connectedClients.push(ws);
15
16     ws.on('message', data=>{
17         connectedClients.forEach((ws, i)=>{
18             if(connectedClients[i]==ws && ws.readyState === ws.OPEN){
19                 ws.send(data);
20             }else{
21                 connectedClients.splice(i,1);
22             }
23         })
24     })
25 });
26
27 app.get('/client', (req, res)=>res.sendFile(path.resolve(__dirname, './client.html')));
28 app.listen(HTTP_PORT, ()=> console.log(`HTTP server escuchando en ${HTTP_PORT} `));
```

En este código básicamente creamos un servidor del tipo webSocket y un servidor Http los cuales escuchan a través de los puertos 8888 y 8000 respectivamente, aunque aquí el servidor de mayor importancia es el de tipo webSocket.

```
JS server.js  client.html  ×
MiniCargador > client.html > html
1  <html>
2      <head>
3          <title>Cliente</title>
4      </head>
5      <body>
6          <img src="">
7          <script>
8              const img = document.querySelector('img');
9              const WS_URL='ws:///192.168.0.105:8888';
10             const ws = new WebSocket(WS_URL);
11             let urlObject;
12             ws.onopen = () => console.log(`Conectado a ${WS_URL}`);
13             ws.onmessage= message => {
14                 const arrayBuffer = message.data;
15                 if(urlObject){
16                     URL.revokeObjectURL(urlObject);
17                 }
18                 urlObject = URL.createObjectURL(new Blob([arrayBuffer]));
19                 img.src=urlObject;
20             }
21         </script>
22     </body>
23 </html>
```

Aqui creamos un cliente HTML con fines de prueba y es por esta razón que se creó el servidor Http con el fin de poder acceder al servidor WS desde el navegador con este cliente.

0.9.1.2 ESP32 CAM

Ahora es turno del módulo ESP 32 CAM, para el cual necesitaremos la librería de este y tomaremos como base el ejemplo que nos proporciona de nombre “CameraWebServer”. Además, instalaremos y haremos uso de la librería ArduinoWebSocket quedando el código de la siguiente manera:

```
#include "esp_camera.h"
#include <WiFi.h>
#include <ArduinoWebsockets.h>

#define CAMERA_MODEL_AI_THINKER // Has PSRAM

#include "camera_pins.h"

const char* ssid = "MERCUSYS";
const char* password = "25876314";
const char* websocket_server_host = "192.168.0.105";
const uint16_t websocket_server_port = 8888;

using namespace websockets;
WebsocketsClient client;

void setupLedFlash(int pin);
```

```
camera_config_t config;
config.ledc_channel = LEDC_CHANNEL_0;
config.ledc_timer = LEDC_TIMER_0;
config.pin_d0 = Y2_GPIO_NUM;
config.pin_d1 = Y3_GPIO_NUM;
config.pin_d2 = Y4_GPIO_NUM;
config.pin_d3 = Y5_GPIO_NUM;
config.pin_d4 = Y6_GPIO_NUM;
config.pin_d5 = Y7_GPIO_NUM;
config.pin_d6 = Y8_GPIO_NUM;
config.pin_d7 = Y9_GPIO_NUM;
config.pin_xclk = XCLK_GPIO_NUM;
config.pin_pclk = PCLK_GPIO_NUM;
config.pin_vsync = VSYNC_GPIO_NUM;
config.pin_href = HREF_GPIO_NUM;
config.pin_sccb_sda = SIOD_GPIO_NUM;
config.pin_sccb_scl = SIOC_GPIO_NUM;
config.pin_pwdn = PWDN_GPIO_NUM;
config.pin_reset = RESET_GPIO_NUM;
config.xclk_freq_hz = 10000000;
config.frame_size = FRAMESIZE_SVGA;
config.pixel_format = PIXFORMAT_JPEG; // for streaming
config.grab_mode = CAMERA_GRAB_WHEN_EMPTY;
config.fb_location = CAMERA_FB_IN_PSRAM;
config.jpeg_quality = 12;
config.fb_count = 1;
```

En este parte hacemos uso de las librerías que se muestran, definimos el modelo de la cámara e importamos el archivo de la configuración de los pines de la cámara, además, otorgamos las credenciales de nuestra red y de nuestro servidor WS.

Después creamos la configuración de la cámara.

```
  WiFi.begin(ssid, password);
  WiFi.setSleep(false);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");


  Serial.print("Camera Ready! Use 'http://");
  Serial.print(WiFi.localIP());
  Serial.println("' to connect");

  while(!client.connect(websocket_server_host, websocket_server_port, "/")){
    delay(500);
    Serial.print(".");
  }
  Serial.println("WebSocket conectado");
}
```

Nos conectamos a la red e intentamos conectarnos a nuestro servidor.

```
void loop() {
  camera_fb_t *fb = esp_camera_fb_get();
  if(!fb){
    Serial.print("Camera capture failed");
    esp_camera_fb_return(fb);
    return;
  }

  if(fb->format != PIXFORMAT_JPEG){
    Serial.print("Non JPEG data not implemented");
    return;
  }

  client.sendBinary((const char*) fb->buf, fb->len);
  esp_camera_fb_return(fb);
}
```

Y por último si no surge ningún error enviamos la señal a nuestro servidor.

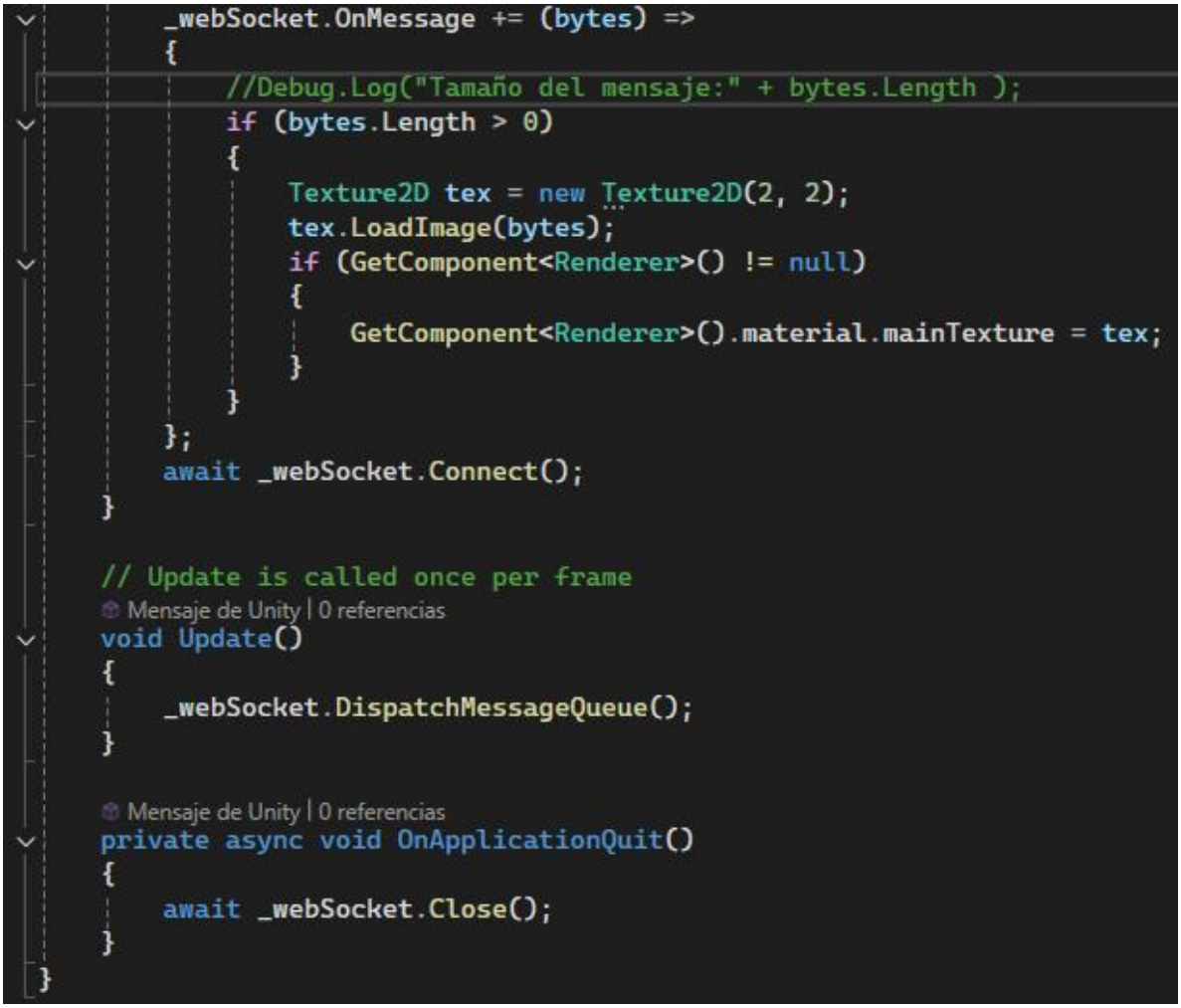
09.1.3 Interfaz Unity

Una vez que comprábamos a través de nuestro cliente HTML que nuestra cámara está enviando la imagen al servidor es momento de introducir esta imagen a nuestra interfaz en unity para eso necesitamos importar las siguientes dos carpetas:


```
using System;
using System.Collections;
using System.Collections.Generic;
using NativeWebSocket;
using UnityEngine;

Script de Unity (1 referencia de recurso) | 0 referencias
public class TextureESP32CAM : MonoBehaviour
{
    private WebSocket _webSocket;
    // Start is called before the first frame update
    Mensaje de Unity | 0 referencias
    async void Start()
    {
        _webSocket = new WebSocket("ws://192.168.0.105:8888");
        _webSocket.OnOpen += () =>
        {
            Debug.Log("Conexion abierta");
        };
        _webSocket.OnError += (e) =>
        {
            Debug.Log("Error"+e);
        };
        _webSocket.OnClose += (e) =>
        {
            Debug.Log("Conexion cerrada");
        };
```

Dentro del código usamos la librería NativeWebSocket, creamos una variable de tipo WebSocket y le brindamos la dirección del servidor WS y colacamos mensajes cuando se ejecuten los eventos OnOpen, OnError y OnClose.



```
        _webSocket.OnMessage += (bytes) =>
        {
            //Debug.Log("Tamaño del mensaje:" + bytes.Length );
            if (bytes.Length > 0)
            {
                Texture2D tex = new Texture2D(2, 2);
                tex.LoadImage(bytes);
                if (GetComponent<Renderer>() != null)
                {
                    GetComponent<Renderer>().material.mainTexture = tex;
                }
            }
        };
        await _webSocket.Connect();
    }

    // Update is called once per frame
    Mensaje de Unity | 0 referencias
    void Update()
    {
        _webSocket.DispatchMessageQueue();
    }

    Mensaje de Unity | 0 referencias
    private async void OnApplicationQuit()
    {
        await _webSocket.Close();
    }
}
```

En el evento onMessage recibimos a través de bytes la imagen del servidor, por lo que si estamos recibiendo bytes creamos una textura a la cual la cargamos la imagen y preguntamos si hay un objeto renderer para poder asignarle esta textura. En el método Update le decimos básicamente que siga recibiendo mensajes y por último cuando cerremos la aplicación cerraremos de igual manera la conexión.

Y solo por último asignamos este script a un plano para que en este se refleje el video.

09.2 ESP8266

Para poder codificar el módulo ESP 8266 se requiere el uso de la librería <ESP8266WiFi.h>, necesaria para la funcionalidad WiFi del módulo.

Después es necesario especificar las credenciales de la red:

- SSID
- Password

Se crea una instancia de WiFiServer, una clase de la librería <ESP8266WiFi.h> que permite que el ESP8266 actúe como un servidor TCP, esperara conexiones de clientes para escuchar por un puerto especifico (80).

Con WiFi.begin(ssid, password) se inicia la conexión WiFi, el ciclo while no se interrumpirá hasta que el ESP8266 se conecte a la red, para finalizar con la configuración, se inicializa el servidor.

```
#include <ESP8266WiFi.h>

const char* ssid = "MERCUSYS";
const char* password = "25876314";

WiFiServer server(80);

void setup() {
  Serial.begin(9600);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {}
  server.begin();
}

void loop() {
  WiFiClient client = server.available();

  if (client) {
    Serial.begin(9600);
    String currentLine = "";
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        if (c == 'w') { Serial.write('w'); }
        if (c == 's') { Serial.write('s'); }
        if (c == 't') { Serial.write('t'); }
        if (c == 'a') { Serial.write('a'); }
        if (c == 'd') { Serial.write('d'); }
        if (c == 'k') { Serial.write('k'); }
        if (c == 'l') { Serial.write('l'); }
        if (c == 'i') { Serial.write('i'); }
        if (c == 'o') { Serial.write('o'); }
        if (c == 'n') { Serial.write('n'); }
        if (c == 'm') { Serial.write('m'); }
      }
    }
    client.stop();
    Serial.end();
  }
}
```

En el loop, se verifica si hay un cliente conectado, si es el caso, mientras que exista la conexión se entra a un bucle en el que el servidor lee caracteres enviados por el cliente y los transmite por el puerto serie. Cuando el cliente se desconecta termina la comunicación.

09.3 Motores

Para controlar el movimiento de los motores encargados del sistema de transmisión y los motores de los actuadores se definen los pines del módulo relay y del puente H, se inicializa la comunicación serial y se configuran los pines como salidas.

```
#define motorR8 8
#define motorR9 9
#define motorL10 10
#define motorL11 11

int in3 = 4;
int in4 = 5;
int in2 = 6;
int in1 = 7;

void setup() {
  Serial.begin(9600);

  pinMode(motorR8, OUTPUT);
  pinMode(motorR9, OUTPUT);
  pinMode(motorL10, OUTPUT);
  pinMode(motorL11, OUTPUT);

  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in1, OUTPUT);
}
```

Se verifica si hay datos disponibles en el puerto serie, si los hay, lo lee y ejecuta acciones específicas de acuerdo con el comando:

Comandos para control de motores (sistema de transmisión):

- 'w': Mueve los motores hacia adelante.
- 's': Mueve los motores hacia atrás.
- 't': Detiene ambos motores.
- 'd': Gira a la derecha.
- 'a': Gira a la izquierda.

Comandos para control del puente H (actuadores):

- 'k': Baja los actuadores traseros.
- 'l': Sube los actuadores traseros.
- 'm': Desactiva los actuadores traseros.
- 'i': Sube los actuadores delanteros.
- 'o': Baja los actuadores delanteros.
- 'n': Desactiva los actuadores delanteros.

```
void loop() {
  if (Serial.available() > 0) {
    char command = Serial.read();

    if (command == 'w') {
      digitalWrite(motorR8, HIGH);
      digitalWrite(motorR9, LOW);
      digitalWrite(motorL10, LOW);
      digitalWrite(motorL11, HIGH);
    }
    else if (command == 's') {
      digitalWrite(motorR8, LOW);
      digitalWrite(motorR9, HIGH);
      digitalWrite(motorL10, HIGH);
      digitalWrite(motorL11, LOW);
    }

    if (command == 't') {
      digitalWrite(motorR8, LOW);
      digitalWrite(motorR9, LOW);
      digitalWrite(motorL10, LOW);
      digitalWrite(motorL11, LOW);
    }

    if (command == 'd') {
      digitalWrite(motorR8, LOW);
      digitalWrite(motorR9, HIGH);
      digitalWrite(motorL10, LOW);
      digitalWrite(motorL11, HIGH);
    }
    else if (command == 'a') {
      digitalWrite(motorR8, HIGH);
      digitalWrite(motorR9, LOW);
      digitalWrite(motorL10, HIGH);
      digitalWrite(motorL11, LOW);
    }

    if (command == 'k') {
      digitalWrite(in1, LOW);
      digitalWrite(in2, HIGH);
    }
    if (command == 'l') {
      digitalWrite(in1, HIGH);
      digitalWrite(in2, LOW);
    }
    if (command == 'm') {
      digitalWrite(in1, LOW);
      digitalWrite(in2, LOW);
    }

     if (command == 'i') {
      digitalWrite(in3, HIGH);
      digitalWrite(in4, LOW);
    }
    if (command == 'o') {
      digitalWrite(in3, LOW);
      digitalWrite(in4, HIGH);
    }
    if (command == 'n') {
      digitalWrite(in3, LOW);
      digitalWrite(in4, LOW);
    }
  }
}
```

09.4 Unity ESP8266

```
using UnityEngine;
using System.Net.Sockets;
using System.Text;

Script de Unity | 0 referencias
public class Chat : MonoBehaviour
{
    private TcpClient client;
    private NetworkStream stream;
    public string ipAddress = "192.168.0.100"; // IP ESP8266
    public int port = 80;

    Mensaje de Unity | 0 referencias
    void Start()
    {
        ConnectToESP8266();
    }
```

Se crea una instancia de TcpClient que se utiliza para conectarse a un servidor remoto a través del protocolo TCP. Una instancia de NetworkStream que proporciona una interfaz para enviar y recibir datos a través de una conexión de red. Se especifica la dirección ip del ESP8286 y el puerto 80.

ConnectToESP8266() es un método que crea una instancia de TcpClient para conectarse al ESP8266 utilizando la dirección IP y el puerto especificado.

```
void ConnectToESP8266()
{
    try
    {
        client = new TcpClient(ipAddress, port);
        stream = client.GetStream();
        Debug.Log("Conectado al ESP8266");
    }
    catch (SocketException e)
    {
        Debug.Log("No se pudo conectar al ESP8266: " + e.Message);
    }
}
```

El método SendMessageToESP(string message) se encarga de tomar un mensaje para codificarlo en bytes y enviarlo al ESP8266 por la conexión establecida previamente.

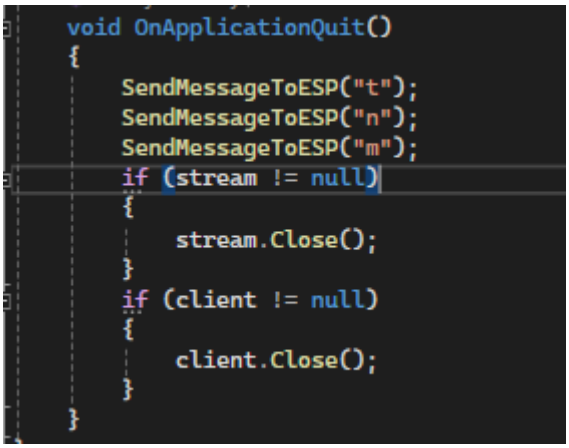
```
void SendMessageToESP(string message)
{
    if (stream != null)
    {
        byte[] data = Encoding.ASCII.GetBytes(message);
        stream.Write(data, 0, data.Length);
        Debug.Log("Mensaje enviado: " + message);
    }
}
```

En el método Update se lleva a cabo el control y manejo de comandos en base a distintas teclas para el control del montacargas, cuando se presiona una tecla se envia un mensaje mediante el método SendMessageToESP hacia el ESP8266.

```
void Update()
{
    if (Input.GetKeyDown(KeyCode.W))
    {
        SendMessageToESP("s");
    }

    if (Input.GetKeyUp(KeyCode.W))
    {
        SendMessageToESP("t");
    }
    if (Input.GetKeyDown(KeyCode.S))
    {
        SendMessageToESP("w");
    }
    if (Input.GetKeyUp(KeyCode.S))
    {
        SendMessageToESP("t");
    }
    if (Input.GetKeyDown(KeyCode.A))
    {
        SendMessageToESP("a");
    }
    if (Input.GetKeyUp(KeyCode.A))
    {
        SendMessageToESP("t");
    }
    if (Input.GetKeyDown(KeyCode.D))
    {
        SendMessageToESP("d");
    }
    if (Input.GetKeyUp(KeyCode.D))
    {
        SendMessageToESP("t");
    }
    if (Input.GetKeyDown(KeyCode.K))
    {
        SendMessageToESP("k");
    }
    if (Input.GetKeyUp(KeyCode.K))
    {
        SendMessageToESP("m");
    }
    if (Input.GetKeyDown(KeyCode.L))
    {
        SendMessageToESP("l");
    }
    if (Input.GetKeyUp(KeyCode.L))
    {
        SendMessageToESP("m");
    }
    if (Input.GetKeyDown(KeyCode.I))
    {
        SendMessageToESP("i");
    }
    if (Input.GetKeyUp(KeyCode.I))
    {
        SendMessageToESP("n");
    }
    if (Input.GetKeyDown(KeyCode.O))
    {
        SendMessageToESP("o");
    }
    if (Input.GetKeyUp(KeyCode.O))
    {
        SendMessageToESP("n");
    }
}
```

OnApplicationQuit() se encarga de enviar mensajes finales al ESP8266, cerrar la conexión y liberar los recursos antes de que la aplicación se cierre por completo.



```
void OnApplicationQuit()
{
    SendMessageToESP("t");
    SendMessageToESP("n");
    SendMessageToESP("m");
    if (stream != null)
    {
        stream.Close();
    }
    if (client != null)
    {
        client.Close();
    }
}
```

10. Plan estratégico del modelo SCRUM para el desarrollo del Montacarguitas

10.1 ¿Qué es Scrum?

Scrum es un marco de trabajo ágil para el desarrollo de proyectos, especialmente en software. Fue diseñado para mejorar la capacidad de las empresas para adaptarse rápidamente a los cambios, generar valor continuamente y facilitar la colaboración de equipos efectiva. Scrum se basa en iteraciones cortas y cíclicas llamadas sprints, en las cuales se entregan incrementos funcionales del producto.

10.2 Fases de Scrum

Planificación del Sprint

En esta fase de debe definir el alcance del trabajo que se realizará durante el Sprint y desarrollar una estrategia para lograrlo. Esto implica analizar los elementos prioritarios del Backlog de productos, calcularlos y elegir los que el equipo se compromete a completar en el Sprint actual. El "Objetivo del Sprint" define el objetivo principal, que es tener una visión clara del incremento del producto que se entregará al final del Sprint. Además, el equipo puede planificar de manera eficiente

cómo abordar el trabajo y distribuir las tareas al crear el Sprint Backlog con las tareas necesarias.

Ejecución del Sprint

Alcanzar el objetivo del Sprint establecido y completar las tareas planificadas en el inventario del sprint. El objetivo del equipo es completar las tareas de manera colaborativa y autónoma, sin interferencias externas. Las Daily Scrums facilitan la transparencia sobre el progreso y los obstáculos, lo que permite ajustar planes y resolver problemas a tiempo. El objetivo principal es entregar un incremento del producto "Terminado" al final del Sprint, cumpliendo con los criterios "Hecho" acordados por el equipo.

Revisión del Sprint

Evaluar el aumento del producto producido durante el Sprint y obtener comentarios de los interesados. El propósito de presentar el trabajo finalizado es verificar que se han cumplido los requisitos y expectativas de los usuarios y partes interesadas. El feedback ayuda a identificar posibles mejoras o cambios en el backlog de productos. El objetivo es garantizar que el producto continúe avanzando y satisface las necesidades del cliente.

Retrospectiva del Sprint

Reflexione sobre el proceso de trabajo y la dinámica del equipo durante el Sprint anterior para identificar áreas de mejora. El objetivo es promover un entorno de revisión y aprendizaje constante en el equipo. El equipo puede definir acciones específicas para optimizar su forma de trabajar al analizar lo que funcionó bien y lo que no. El objetivo es mejorar continuamente la productividad, la cooperación y la calidad del trabajo del equipo.

Refinamiento del Backlog

Mantenga el backlog de productos actualizado, claro y priorizado, con componentes listos para ser seleccionados en Sprints posteriores. Se trata de garantizar que los elementos del backlog estén claramente definidos, estimados y priorizados de

acuerdo con las necesidades del cliente y del negocio. Al dividir tareas grandes en tareas más pequeñas, se facilita la planificación y ejecución del trabajo en Sprints posteriores. El objetivo es tener un inventario de productos "listo para trabajar", lo que facilita la planificación de los próximos Sprints.

10.3 Roles

Product Owner: Es el responsable de tratar con el cliente y decir que se necesita. (Luis Ramírez)

Scrum Master: Ayuda al equipo a seguir las prácticas de Scrum y elimina obstáculos. (Eduardo Cervantes)

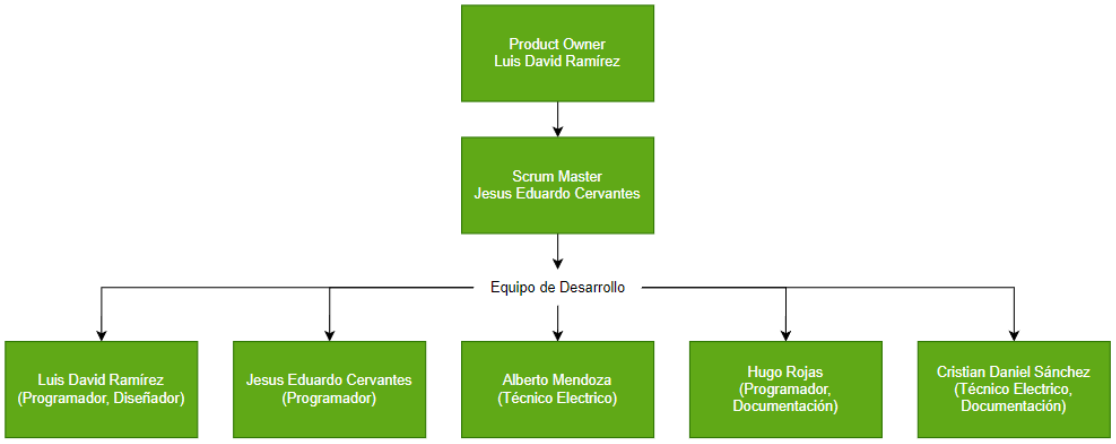
Equipo de desarrollo es un grupo de personas que cumplen múltiples funciones y trabajan para garantizar la entrega del producto.

Programadores: Luis Ramírez, Eduardo Cervantes y Hugo Rojas

Técnicos Eléctricos: Alberto Mendoza, Cristian Sánchez

Diseñador: Luis Ramírez

Documentación: Cristian Sánchez, Hugo Rojas



Bibliografía

BUA. (s/f). Qué es un punto de acceso. Moodle2022-23.ua.es. Recuperado el 24 de mayo de 2024, de https://moodle2022-23.ua.es/moodle/pluginfile.php/11267/mod_resource/content/16/tema/qu_es_un_punto_de_acceso.html

Fernández, C. (2022, marzo 30). ¿Qué es la tecnología Wifi? Tecnología Wifi para empresas. ABAMobile. https://abamobile.com/web/tecnologia-wifi-que-es-y-caracteristicas/

Gallego, G. (2019, junio 19). Qué es el WiFi y cómo funciona para conectar todo a Internet. ADSLZone. https://www.adslzone.net/reportajes/tecnologia/que-es-wifi-como-funciona/

Informatica, I. (s/f). FACULTAD DE INGENIERÍA DE SISTEMAS. Edu.ec. Recuperado el 24 de mayo de 2024, de https://bibdigital.epn.edu.ec/bitstream/15000/9003/1/CD-6017.pdf

¿Qué factores pueden afectar al rendimiento WiFi? (s/f). NETGEAR KB; NETGEAR. Recuperado el 24 de mayo de 2024, de https://kb.netgear.com/es/000037824/Qu%C3%A9-factores-pueden-afectar-al-rendimiento-WiFi?language=es

Seis pequeños trucos de TP-Link para mejorar la señal inalámbrica. (s/f). Tp-link.com. Recuperado el 24 de mayo de 2024, de https://www.tp-link.com/es/press/news/15273/

Vargas, D. (2023, marzo 6). Protocolo TCP: definición y funcionamiento. Tutoriales Hostinger. https://www.hostinger.mx/tutoriales/protocolo-tcp

(S/f-a). Amazon.com. Recuperado el 24 de mayo de 2024, de https://aws.amazon.com/es/compare/the-difference-between-https-and-http/#:~:text=de%20transferir%20datos.-

,%C2%BFC%C3%B3mo%20funciona%20el%20protocolo%20HTTP%3F,env%C3%ADa%20la%20solicitud%20HTTP%20GET.

(S/f-b). Amazon.com. Recuperado el 24 de mayo de 2024, de https://aws.amazon.com/es/what-is/mqtt/#:~:text=La%20comunicaci%C3%B3n%20MQTT%20utiliza%20el,mediante%20certificados%20SSL%20y%20contrase%C3%B1as.

(S/f-c). Khanacademy.org. Recuperado el 24 de mayo de 2024, de https://es.khanacademy.org/computing/ap-computer-science-principles/the-internet/x2d2f703b37b450a3:transporting-packets/a/user-datagram-protocol-udp#:~:text=El%20Protocolo%20de%20datagrama%20de,o%20llegan%20fuera%20de%20orden.

Cómo aplicar la metodología Scrum y qué es el método Scrum. (2024, abril 9). APD España; APD. https://www.apd.es/metodologia-scrum-que-es/

(S/f). Ilimit.com. Recuperado el 31 de mayo de 2024, de https://ilimit.com/blog/metodologia-scrum/