EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE TLÁHUAC III

INGENIERÍA INFORMÁTICA

APLICACIÓN CLIENTE SERVIDOR PARA CONTROL Y MONITOREO DE ROBOT INSTITUCIONAL

NAVARRETE CRUZ ANGEL

181120061

TLÁHUAC, CDMX **2023**

Enero - Junio 2023

AGRADECIMIENTO

Con la virtud de nuestra formación y crecimiento personal, agradezco al Instituto Tecnológico de Tláhuac III por brindarme las condiciones apropiadas para desarrollar este trabajo de manera exitosa durante el año 2021. También agradezco a los profesores por ser apoyo durante esta carrera, que son un aporte al proceso de formación profesional, en especial a todos los docentes de esta carrera por el acompañamiento en el desarrollo de este trabajo.

Gracias al mundo por apoyarme siempre en mi postura de trabajar solo en los proyectos que me interesan, aquellos que logran quitarme el aburrimiento, me dan ánimo en mi profesión y me permiten expresarme con honestidad.

Agradezco a los responsables de este contenido por ser de posible ayuda, y gracias por la intervención de las asesorías semanales que llevaron a cabo este puntual proyecto, muchas gracias. Además, quiero dar mención honorífica a 'la nena', que no me dejaba quedarme dormido mientras realizaba este documento. Muchas, muchas gracias.

RESUMEN

Durante el desarrollo del proyecto se explora la creación de un completo sistema cliente-servidor, que será la forma en que se pueda intervenir para el monitoreo y control de un robot con una aplicación remota.

Para esto, el robot se conecta a un servidor para que pueda ser controlado a través de una página web ejecutada desde un computador conectado a una red de área local.

Esta es una herramienta didáctica que permite al estudiante controlar un sistema real desde cualquier punto de conexión dentro de la red de área local.

En un equipo servidor se desarrolla un hardware de control que se ejecuta por medio de la implementación de Arduino y la comunicación de este con el usuario remoto a través de un protocolo llamado; “Firmata” utilizando el lenguaje JavaScript y Node.js. En el computador servidor usaremos express para crear un servidor web y Node.js como el entorno de ejecución, ya que esto nos permitirá el acceso para usar el robot de forma remota.

De esta forma se elimina la interacción directa que siempre ha existido entre el robot y su operario, ejecutando la aplicación por medio de una red de área local.

El robot utilizado en el proyecto ha sido creado por la Universidad Autónoma Metropolitana Unidad Iztapalapa, evitando así acudir a robots elaborados por terceras personas y, por consiguiente, corriéndose el riesgo de encontrar problemas de estrategias de control dado que es casi imposible llegar a modificar el algoritmo de programación con el que ya vienen programados.

INDICE

AGRADECIMIENTO 1
RESUMEN 2
INTRODUCCIÓN 6
Capítulo 1: GENERALIDADES DEL PROYECTO 7
- **1.1 JUSTIFICACIÓN** 7
- **1.2 DESCRIPCIÓN DE LA ORGANIZACIÓN** 7
- **1.3 SITUACIÓN PROBLEMÁTICA** 8
- **1.4 OBJETIVO GENERAL** 8
- **1.5 OBJETIVOS PARTICULARES** 8

Capítulo 2: MARCO TEÓRICO 9
- **2.1 ARQUITECTURA CLIENTE-SERVIDOR** 10
- **2.2 WINDOWS SERVER** 11
- **2.3 LENGUAJE JAVASCRIPT** 12
- **2.4 ENTORNO DE EJECUCIÓN NODE.JS** 13
- **2.5 FRAMEWORK JOHNNY-FIVE** 14
- **2.6 RED ÁREA LOCAL (LAN)** 15
- **2.7 PROTOCOLOS DE COMUNICACIÓN** 16
- **2.7.1 PROTOCOLO TCP/IP** 16
- **2.7.2 PROTOCOLO FIRMATA** 17
- **2.8 PLACA ARDUINO UNO** 19

Capítulo 3: FASES DE DESARROLLO 20
- **3.1 DISEÑO DEL ROBOT** 20
  - 3.1.1 SISTEMA DEL ROBOT 20
  - 3.1.2 ALIMENTACIÓN 21
  - 3.1.3 MOTORES 21
  - 3.1.5 ESTRUCTURA DEL ROBOT 22
  - 3.1.6 PLACA MADRE 23
  - 3.1.7 SOFTWARE Y CÓDIGO 24
  - 3.1.8 DIAGRAMA DE FLUJO DEL PROCESO DEL FUNCIONAMIENTO 27
- **3.2 PREPARACIÓN DEL ENTORNO** 28

- **3.3 Desarrollo Del Programa** 30
  - 3.3.1 Introducción Y Aclaraciones Previas 30
  - 3.3.2 Programa Principal 31
  - 3.3.3 Funcionamiento Del Código 35

Capítulo 4: RESULTADOS 47

Capítulo 5: CONCLUSIONES 51

- **5.1 Recomendaciones** 51
- **5.2 Posibles Mejoras** 52

Capítulo 6: COMPETENCIAS DESARROLLADAS Y/O APLICADAS 53

- **6.1 Dificultades** 53
- **6.2 Experiencia obtenida** 54

Capítulo 7: FUENTES DE INFORMACIÓN 55

TABLA DE ILUSTRACIONES

Ilustración 1. Esquema General del Proyecto 9
Ilustración 2. Modelo Cliente-Servidor 10
Ilustración 3 Sistema general del Robot 20
Ilustración 4. Batería 12 voltios 21
Ilustración 5. L298N (Puente H) 21
Ilustración 6. Motor de corriente continua 21
Ilustración 7. Arduino UNO 22
Ilustración 8. Arduino nano 22
Ilustración 9. Estructura de P.E.P.E 22
Ilustración 10. Placa base 23
Ilustración 11. Proceso de Funcionamiento de P.E.P.E 27
Ilustración 12. Página oficial de Node.js 28
Ilustración 13. Página oficial de Arduino 29
Ilustración 14. Diagrama de Casos de Uso 30
Ilustración 15. Proyecto Control P.E.P.E 31
Ilustración 16. Carpeta Raíz 31
Ilustración 17. Instalación de Jhonny-Five 32
Ilustración 18. Johnny -Five instalado correctamente 32
Ilustración 19. Instalación de Socket.io 32
Ilustración 20. Socket.io instalado correctamente 33
Ilustración 21. Instalación de Express 33
Ilustración 22. Visual Studio Code 33
Ilustración 23. Proyecto Control-PEPE en Visual Studio Code 34
Ilustración 24. package.json 34
Ilustración 25. Diagrama de Secuencia 40
Ilustración 26. Interfaz Web Control P.E.P.E 40
Ilustración 27. Puerto COM8 47
Ilustración 28. Protocolo Firmata 47
Ilustración 29. StandardFirmataPlus 48
Ilustración 30. Protocolo Subido correctamente al Arduino UNO 48
Ilustración 31. Carpeta del Proyecto P.E.P.E 49
Ilustración 32. Terminal y Ubicación Raíz del Proyecto 49
Ilustración 33. Ejecución del Programa 49
Ilustración 34. Control Remoto de P.E.P.E 50
Ilustración 35. Control P.E.P.E en móvil 50

INTRODUCCIÓN

La utilización de herramientas didácticas para la enseñanza en instituciones de educación superior es cada día más común en nuestro medio. Mediante estas herramientas, el estudiante se motiva en el desarrollo de nuevas habilidades y destrezas en su metodología de estudio.

En la Universidad Autónoma Metropolitana unidad Iztapalapa existen algunas de estas herramientas para la enseñanza. Una de ellas es el robot institucional llamado PEPE (Prototipo de Enseñanza de Programación y Electrónica), el cual te permite, con todos sus componentes, el desarrollo del aprendizaje acerca de los principios de informática y de la base de la electrónica. Esto proporciona un enfoque totalmente práctico de las disciplinas mencionadas. El mayor problema con respecto al robot es la delimitación y velocidad de ejecución, así como su practicidad de manejo y monitoreo. En este proyecto se desarrolla un sistema completo basado en una arquitectura cliente-servidor para un mejor control y monitoreo de dicho robot.

Desde el punto de vista de la automatización, un sistema cliente-servidor es una arquitectura bastante robusta, pero con una buena flexibilidad de crecimiento que nos permitirá, de forma certera, una excelente ejecución. El robot P.E.P.E tendrá conectada una mini laptop, la cual funcionará como un servidor web, y estará conectada a una red de área local, teniendo una dirección IP estática. Mientras tanto, el cliente, a través de cualquier navegador web, podrá acceder y ejecutar una página web donde se encontrará el control de P.E.P.E.

En este trabajo se muestra una posible solución de aplicación para el control y monitoreo de un robot. Además, explicaremos en detalle el funcionamiento de la arquitectura cliente-servidor, cómo conectaremos ambas peticiones y abordaremos la creación del control mediante el lenguaje JavaScript.

Capítulo 1: GENERALIDADES DEL PROYECTO

1.1 JUSTIFICACIÓN

En la universidad autónoma metropolitana unidad Iztapalapa, emplean el uso de robots institucionales los cuales son manejados por el grupo de la IEEE de dicha unidad, para promocionar las carreras de CBI (Ciencias básicas e Ingeniería), los cuales son enviados a concursos, eventos, exposiciones, entre otras invitaciones externas e internas, Cuando estuve en mi servicio social como mantenimiento preventivo y correctivo de estos robots institucionales, me percate que la mayoría de complicaciones provenían del manejo y monitoreo de los distintos controles que tienen para su uso, por lo tanto la idea de que cualquier persona pueda manejarlos con el simple hecho de entrar al servidor del robot por medio de algún navegador de cualquier dispositivo, conllevaría a la mejor solución que se le pueda brindar para optimizar el manejo de dicho robot.

1.2 DESCRIPCIÓN DE LA ORGANIZACIÓN

La Universidad Autónoma Metropolitana (UAM) Unidad Iztapalapa es una institución pública de educación superior ubicada en la Ciudad de México. Fue fundada en 1974 con el objetivo de ofrecer una educación de calidad y fomentar la investigación en diversas áreas del conocimiento, la cual brinda varias carreras con tres divisiones disciplinarias, ya sean CBS (Ciencias Biológicas y de la Salud), CBI (Ciencias Básicas e Ingeniería) y CSH (Ciencias Sociales y Humanidades).

La investigación es una de las principales actividades de la UAM Iztapalapa, con un enfoque interdisciplinario que aborda problemáticas relevantes a nivel local, nacional e internacional. Los proyectos de investigación se realizan en colaboración con otras instituciones académicas y organizaciones de la sociedad civil. Este proyecto es manejado gracias al apoyo de la división de ciencias básicas e ingeniería de la institución, así como apoyo del departamento del laboratorio de ingeniería de software y del departamento de energía eléctrica.

1.3 Situación Problemática

En el medio estudiantil de la UAM unidad Iztapalapa, para el manejo y monitoreo del robot institucional PEPE (Prototipo de enseñanza de programación y electrónica), se requieren de una gran cantidad de elementos para su ejecución, teniendo dificultades en la facilidad de uso como en la practicidad de manejo. Es por esto que se plantea que los estudiantes puedan acceder a una aplicación de acceso remoto para monitorear y controlar un sistema permitiendo el fácil acceso y manipulación de dicho robot institucional.

1.4 Objetivo General

Diseñar, implementar y desarrollar un sistema cliente-servidor para monitorear y controlar el manejo de un robot a través de un servidor web.

1.5 Objetivos Particulares

a) Creación de un servidor en el robot.

b) Creación de una aplicación cliente la cual se conectará al servidor.

c) Diseñar una interfaz gráfica para acceder de forma remota al control del robot.

d) Ajustar que el Servidor trate a los clientes de manera independiente. Evitando que si está ejecutando una orden otro cliente pueda acceder a él.

e) Establecer las técnicas de control adecuadas para las diferentes acciones.

f) Realizar un análisis de los diferentes movimientos del robot para establecer el control a utilizar.

Capítulo 2: MARCO TEÓRICO

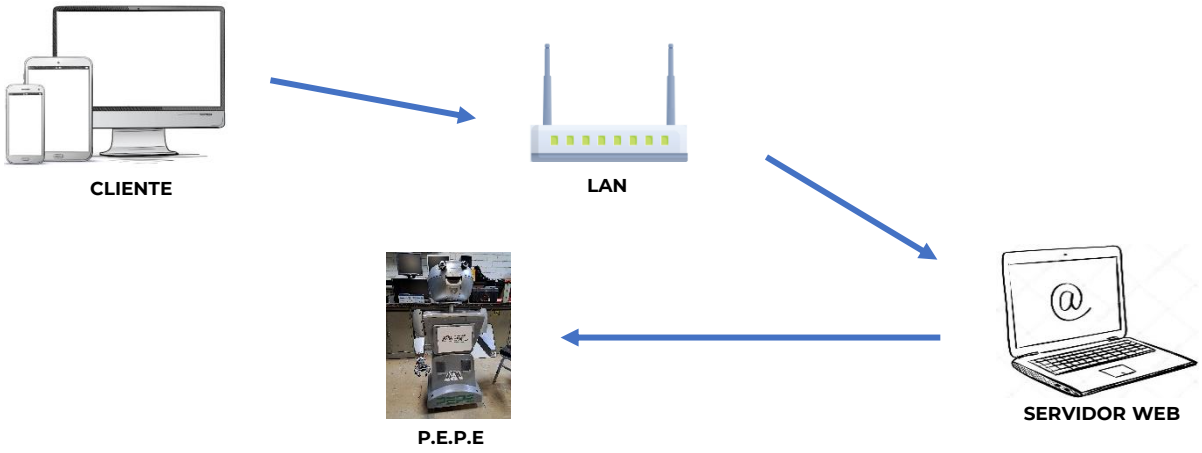

Ilustración 1. Esquema General del Proyecto

En la ilustración número 1, podemos observar un esquema funcional sobre la interacción entre cada uno de los componentes a utilizar en el proyecto, el móvil, ordenador o Tablet (*utilizado para el cliente*), el ordenador (*Utilizado para ser el servidor*), el modem (Donde se alojará la red de área local para todo el sistema cliente-servidor) y el robot a controlar.

Todos estos componentes son necesarios para el objetivo de monitorear y controlar el robot el cual consiste en: El cliente usando cualquier dispositivo, se conecta a la red de área local para poder comunicarse con el servidor, este servidor esta creado de manera local usando express, donde estará alojado el sistema de control y monitoreo del robot. La comunicación entre el servidor y el robot se realiza mediante una placa Arduino UNO conectado al puerto USB del servidor, enlazado con el código de la función del robot y el puerto correspondiente del servidor de la página web.

A continuación, se hace una introducción de conceptos básicos, de los diferentes temas a tratar en el desarrollo de este proyecto, con la finalidad de dar a entender una idea general de lo planteado.

2.1 Arquitectura Cliente-Servidor

La arquitectura cliente-servidor es un modelo de diseño de software en el que las tareas de una aplicación se dividen en dos partes distintas y separadas: el cliente y el servidor. el cliente es la interfaz de usuario que se ejecuta en una computadora local o remota y que interactúa con el usuario final, mientras que el servidor es el software que se ejecuta en una computadora remota y que maneja las solicitudes del cliente.

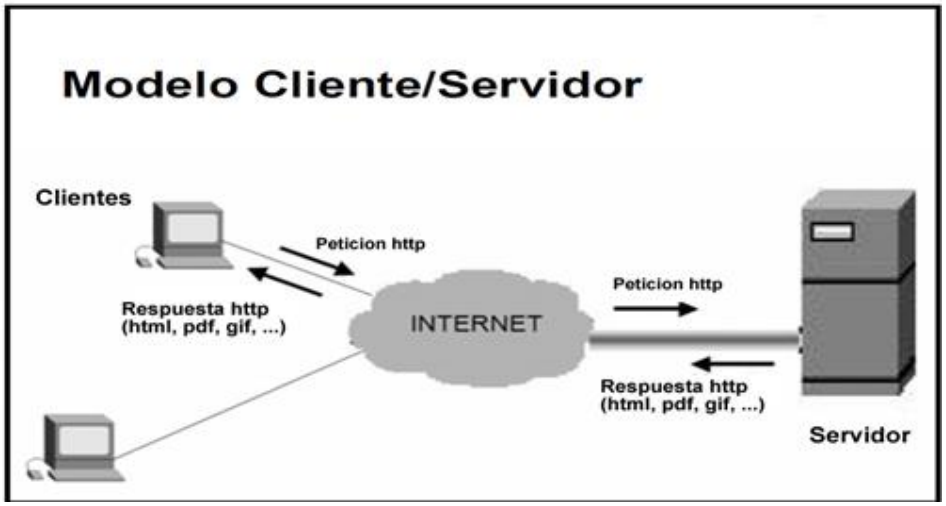

Ilustración 7. Modelo Cliente-Servidor

En la ilustración 2 se muestra la configuración de la arquitectura cliente-servidor, donde el cliente solicita servicios o recursos al servidor, y el servidor responde a estas solicitudes proporcionando los servicios y recursos necesarios. El cliente y el servidor se comunican a través de una red de computadoras, como Internet o una red local, utilizando un protocolo de comunicación específico, como HTTP o TCP/IP.

La arquitectura cliente-servidor es un modelo de diseño de software que se utiliza ampliamente en aplicaciones empresariales y de red debido a su escalabilidad, seguridad, mantenimiento y flexibilidad. Es una opción popular para muchas empresas y organizaciones que necesitan una aplicación que pueda crecer y adaptarse a las necesidades cambiantes del negocio.

2.2 Windows Server

Windows Server es un sistema operativo desarrollado por Microsoft diseñado específicamente para servidores y redes de computadoras. Este sistema operativo es utilizado por empresas y organizaciones para administrar y gestionar una amplia variedad de servicios, aplicaciones y recursos de red.

Windows Server funciona como una plataforma para proporcionar servicios y recursos de red a los usuarios, como, por ejemplo:

- Servicios de directorio y autenticación de usuarios
- Servidores de correo electrónico
- Servidores web y de aplicaciones
- Servicios de almacenamiento en red
- Servicios de virtualización y de nube privada
- Servicios de seguridad y control de acceso a la red

Al utilizar Windows Server, las empresas pueden mejorar la eficiencia y la productividad al compartir recursos y datos de manera centralizada, lo que reduce el costo y la complejidad de la gestión de redes. También permite una mayor seguridad y un control de acceso más granular a los recursos de la red, lo que es importante para proteger la información confidencial de la empresa.

2.3 Lenguaje JavaScript

JavaScript es un lenguaje de programación que se utiliza para crear aplicaciones web dinámicas e interactivas. Fue creado en 1995 por Brendan Eich y es uno de los lenguajes de programación más populares del mundo.

JavaScript se utiliza en una variedad de aplicaciones, incluyendo la programación web del lado del cliente, la programación de aplicaciones móviles y la programación del lado del servidor. Es el lenguaje de programación más comúnmente utilizado para programar el comportamiento de las páginas web en el lado del cliente (en el navegador web del usuario) y para interactuar con elementos HTML y CSS.

Razones por las cual se usará JavaScript:

Interactividad en el lado del cliente: JavaScript permite a los desarrolladores crear aplicaciones web interactivas y dinámicas que pueden responder a las acciones del usuario en tiempo real. Esto incluye la validación de formularios, animaciones y efectos visuales.

Mejora de la experiencia del usuario: Al agregar interactividad en el lado del cliente, los usuarios pueden interactuar con una página web de manera más intuitiva y eficiente, lo que mejora la experiencia del usuario y aumenta la satisfacción.

Bibliotecas y frameworks: Existen una gran cantidad de bibliotecas y frameworks de JavaScript disponibles para los desarrolladores, que facilitan la creación de aplicaciones web complejas y sofisticadas. Estos incluyen Angular, React, Vue.js y Node.js.

JavaScript es un lenguaje de programación que se utiliza para crear aplicaciones web dinámicas e interactivas. Se utiliza para agregar interactividad en el lado del cliente, mejorar la experiencia del usuario, funcionar en múltiples plataformas y aprovechar las bibliotecas y frameworks de JavaScript disponibles.

2.4 Entorno De Ejecución Node.Js

Node.js es un entorno de tiempo de ejecución de JavaScript que permite a los desarrolladores crear aplicaciones en el lado del servidor utilizando el lenguaje de programación JavaScript. Node.js se utiliza comúnmente para crear aplicaciones de red escalables y de alta velocidad, como servidores web, aplicaciones de mensajería y aplicaciones de tiempo real.

Razones por las que se utilizara Node.js:

Eficiencia en el manejo de eventos: Node.js se basa en un modelo de evento no bloqueante que lo hace muy eficiente para manejar múltiples solicitudes simultáneamente. Esto significa que Node.js puede manejar una gran cantidad de conexiones simultáneas sin consumir demasiada memoria del servidor.

Facilidad de uso: Node.js utiliza JavaScript, que es uno de los lenguajes de programación más populares y conocidos. Los desarrolladores que ya están familiarizados con JavaScript pueden aprender Node.js más fácilmente.

Biblioteca de módulos: Node.js tiene una amplia biblioteca de módulos disponibles que se pueden utilizar para agregar funcionalidad adicional a una aplicación. Esto incluye módulos para manejar solicitudes HTTP, interactuar con bases de datos, manejar autenticación y autorización, y mucho más.

Node.js se utiliza para crear aplicaciones en el lado del servidor que son escalables, de alta velocidad y eficientes en el manejo de eventos. Node.js es fácil de aprender para los desarrolladores que ya conocen JavaScript y tiene una amplia biblioteca de módulos disponible. Además, Node.js es muy rápido y tiene una gran comunidad activa de desarrolladores.

2.5 Framework Johnny-five

Johnny-Five es una biblioteca de JavaScript que se utiliza para interactuar con placas y dispositivos electrónicos, especialmente con Arduino. Permite controlar y programar fácilmente dispositivos electrónicos utilizando JavaScript en lugar de tener que usar lenguajes de programación de bajo nivel como C o C++.

Johnny-Five proporciona una API sencilla y amigable que abstrae la complejidad de la comunicación con hardware y permite a los desarrolladores crear proyectos interactivos utilizando JavaScript. La biblioteca incluye soporte para una amplia gama de componentes electrónicos como sensores, actuadores y otros dispositivos, lo que permite crear proyectos de robótica, domótica, Internet de las cosas (IoT) y más.

Razones para implementar Johnny-five:

a) Control de luces y actuadores: Johnny-Five permite encender y apagar luces, controlar motores, servomotores y otros dispositivos físicos conectados a una placa Arduino o compatible.
b) Lectura de sensores: Puedes usar Johnny-Five para leer datos de sensores como sensores de temperatura, sensores de luz, sensores de movimiento, sensores de proximidad, entre otros.
c) Interacción con dispositivos externos: Puedes utilizar Johnny-Five para interactuar con otros dispositivos externos como pantallas LCD, pantallas LED, teclados, joysticks, cámaras, entre otros.
d) Proyectos de robótica: Johnny-Five es muy utilizado en proyectos de robótica, ya que permite controlar los movimientos de robots, interactuar con sensores y crear comportamientos personalizados.

Johnny-Five es una biblioteca de JavaScript que facilita la programación de dispositivos electrónicos y la interacción con hardware utilizando un lenguaje de programación de alto nivel como JavaScript.

2.6 Red Área Local (Lan)

Una red de área local (LAN, por sus siglas en inglés) es una red de computadoras que se encuentran en un área geográfica limitada, como un edificio, una escuela, una empresa o una casa. Las LAN permiten que los dispositivos conectados en la red se comuniquen entre sí y compartan recursos, como archivos, impresoras, aplicaciones y dispositivos de almacenamiento.

Las LAN suelen ser gestionadas por un administrador de red, quien es responsable de asegurarse de que todos los dispositivos estén conectados y funcionando correctamente. Las redes de área local pueden tener diferentes topologías, que son la forma en que los dispositivos están conectados en la red. Las topologías comunes incluyen el bus, la estrella y el anillo.

Las LAN se utilizan para compartir recursos, reducir costos y mejorar la productividad. Por ejemplo, los empleados de una empresa pueden compartir archivos y documentos en la red, acceder a recursos compartidos como impresoras y escáneres, y comunicarse entre sí a través de la red. Las LAN también se utilizan en entornos educativos para compartir recursos de aprendizaje y permitir la colaboración entre estudiantes y profesores.

Una red de área local es una red de computadoras en un área geográfica limitada que permite la comunicación y el intercambio de recursos entre los dispositivos conectados en la red.

2.7 Protocolos De Comunicación

Este proyecto se trabajan dos protocolos de comunicación. El primero de ellos es el protocolo TCP/IP para la transmisión de datos entre los equipos servidor y cliente y el segundo es el protocolo UDP para la transmisión en tiempo real.

2.7.1 Protocolo Tcp/Ip

TCP/IP es un conjunto de protocolos de red que se utiliza para conectar dispositivos y sistemas informáticos en redes de área amplia (WAN) y redes de área local (LAN). El protocolo TCP/IP se ha convertido en el estándar de facto para la comunicación de datos en Internet y se utiliza en muchas aplicaciones y dispositivos, desde servidores web y routers hasta dispositivos móviles y de escritorio.

TCP/IP es un conjunto de protocolos compuestos por dos partes principales: el protocolo de control de transmisión (TCP) y el protocolo de Internet (IP). TCP es responsable de dividir los datos en paquetes y asegurarse de que los paquetes se entreguen en orden y sin errores. IP, por su parte, es responsable de enrutar los paquetes de datos a través de la red hasta su destino final.

TCP es un protocolo orientado a la conexión, lo que significa que establece una conexión entre el emisor y el receptor antes de transmitir datos. Este proceso se conoce como un "apretón de manos" (handshake) y se realiza mediante el intercambio de paquetes de control entre los dispositivos. Una vez que se establece la conexión, TCP divide los datos en paquetes y agrega encabezados de control de flujo y confirmación de recepción para asegurarse de que los paquetes se entreguen correctamente y en orden.

IP es responsable de enrutar los paquetes de datos a través de la red hasta su destino final. Cada dispositivo conectado a la red tiene una dirección IP única que

se utiliza para identificar el dispositivo y enrutar los paquetes. Cuando un paquete se envía a través de la red, IP utiliza la dirección IP de destino para determinar la ruta que debe seguir el paquete a través de la red.

TCP/IP se utiliza en una amplia variedad de aplicaciones y dispositivos, desde servidores web y routers hasta dispositivos móviles y de escritorio. Se utiliza en Internet para conectar dispositivos y sistemas informáticos en todo el mundo y también se utiliza en redes privadas, como redes de área local (LAN), para conectar dispositivos en una red local.

TCP/IP es un conjunto de protocolos de red que se utiliza para conectar dispositivos y sistemas informáticos en redes de área amplia (WAN) y redes de área local (LAN). Está compuesto por dos partes principales: el protocolo de control de transmisión (TCP) y el protocolo de Internet (IP). TCP se encarga de dividir los datos en paquetes y asegurarse de que se entreguen correctamente, mientras que IP es responsable de enrutar los paquetes a través de la red. TCP/IP se utiliza en una amplia variedad de aplicaciones y dispositivos y es el estándar de facto para la comunicación de datos en Internet.

2.7.2 Protocolo Firmata

Firmata es un protocolo de comunicación estándar utilizado para establecer la comunicación entre una computadora y un microcontrolador o dispositivo electrónico, como un Arduino. Permite que la computadora controle y se comunique con el microcontrolador o dispositivo de manera sencilla.

El protocolo Firmata se basa en una serie de mensajes y comandos predefinidos que se intercambian entre la computadora y el microcontrolador. Estos mensajes permiten a la computadora enviar instrucciones al microcontrolador para leer y escribir en los pines digitales y analógicos, controlar actuadores, recibir datos de sensores y realizar otras operaciones.

La ventaja principal de utilizar el protocolo Firmata es que simplifica la programación del microcontrolador. En lugar de tener que escribir un código específico para cada tarea en el microcontrolador, se puede utilizar una biblioteca o un programa en la computadora que se comunique con el microcontrolador a través de Firmata.

De esta manera, se puede utilizar cualquier lenguaje de programación compatible con Firmata para controlar el microcontrolador sin necesidad de escribir código adicional en el dispositivo. Esto hace que el desarrollo de proyectos que involucran microcontroladores sea más rápido y flexible, ya que se puede aprovechar el poder de programación de la computadora para controlar el hardware.

El protocolo Firmata funciona mediante la comunicación serie entre una computadora (anfitrión) y un microcontrolador (cliente) que ejecuta el firmware Firmata. A continuación, se describe el proceso básico de cómo funciona Firmata:

Configuración inicial: El microcontrolador se programa con el firmware Firmata, que implementa el protocolo. Este firmware establece la comunicación serie con la computadora y configura los pines y características del microcontrolador.

Establecimiento de la conexión: La computadora y el microcontrolador se conectan físicamente a través de un puerto serie (como USB) o mediante conexión inalámbrica (Bluetooth, por ejemplo). Se selecciona el puerto adecuado para la comunicación.

2.8 Placa Arduino Uno

Arduino Uno es una placa de desarrollo de microcontrolador de código abierto. Es una de las placas más populares de la plataforma Arduino y se utiliza en una amplia variedad de proyectos electrónicos. La placa se compone de un microcontrolador AVR de Atmel, puertos de entrada/salida, un oscilador de cristal, un conector USB, un conector de alimentación, y un botón de reset.

La placa Arduino Uno se utiliza para prototipar y construir proyectos electrónicos de una manera rápida y sencilla. Los proyectos se programan en el lenguaje de programación de Arduino, que es una versión simplificada de C++. Arduino Uno se puede programar desde un ordenador utilizando el software de Arduino, que es un entorno de desarrollo integrado que permite escribir, compilar y cargar programas en la placa.

Arduino Uno se utiliza en proyectos que van desde sistemas de automatización del hogar, control de robots y drones, hasta proyectos de arte interactivo y wearables. La placa es especialmente popular en proyectos de robótica y domótica, ya que cuenta con una gran comunidad de usuarios y una gran cantidad de documentación y tutoriales disponibles en línea.

La placa Arduino Uno es una placa de desarrollo de microcontrolador de código abierto que se utiliza en una amplia variedad de proyectos electrónicos para prototipar y construir sistemas electrónicos de manera rápida y sencilla. Se programa utilizando el lenguaje de programación de Arduino y se utiliza en proyectos de robótica, domótica, arte interactivo, wearables, y muchos otros.

Capítulo 3: FASES DE DESARROLLO

3.1 Diseño Del Robot

El robot a controlar es un robot móvil controlado por medio de 2 microcontroladores (Arduino UNO y Arduino nano), el Arduino UNO el cual hace todo el seguimiento del movimiento de las llantas (adelante, izquierda, derecha y hacia atrás), y el Arduino nano el cual controla el movimiento de la cabeza (arriba y giro a los lados) que fue diseñado y construido para poder moverse como robot móvil guiado por un control remoto (mando de X box) hacia adelante, hacia atrás y girar a los lados, mover los brazos en un movimiento perpendicular, poder mover las muñecas de las pinzas en un giro de 180° conforme a su eje inicial, abrir las pinzas en un movimiento de 90° grados con forme a su eje inicial y mover la cabeza hacia arriba y hacia ambos lados.

3.1.1 Sistema del robot

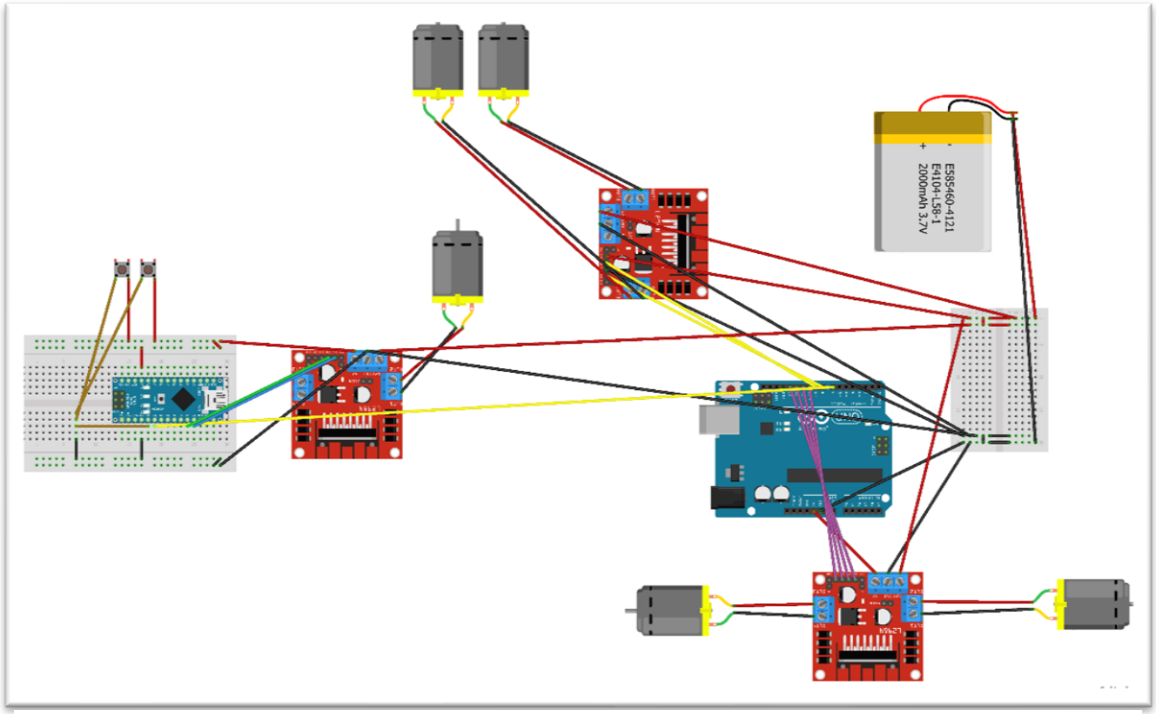

Ilustración 8 Sistema general del Robot

En la ilustración 3 se muestra un esquema de los componentes del sistema completo del robot, este sistema consta de una fuente de alimentación de 12 voltios, 5 motores de corriente continua, 3 puentes H (L298N) y 2

microcontroladores Arduino para el control del mecanismo de movimiento y para la comunicación con el computador.

3.1.2 ALIMENTACIÓN

El robot se encuentra alimentado por una batería de 12 voltios, Su funcionamiento no requiere de tanta corriente, para ello usa varios puentes H (L298n), para impulsar mas corriente a los varios motores que usa tanto en las ruedas, en los brazos y en la cabeza.

Ilustración 10. L298N (Puente H)

Ilustración 9. Batería 12 voltios

3.1.3 MOTORES

Los brazos, las ruedas y la cabeza tienen movimiento por la acción de los motores de corriente continua que emplean. Estos motores de corriente continua son adecuados para la aplicación de alta velocidad y torque que requiere cada parte del robot. El control de sentido de giro de estos motores se hace por medio de varios puentes H conjunto a un programa del microcontrolador.

Ilustración 11. Motor de corriente continua

3.1.4 MICROCONTROLADOR

El sistema funciona con dos microcontroladores Arduino de la serie UNO y la serie nano, estos microcontroladores son los encargados de activar los motores del

robot. Además, que permiten la conexión con la laptop para poder conectarse con el emulador del mando de X box y así poder manipular el robot.

Ilustración 12. Arduino UNO

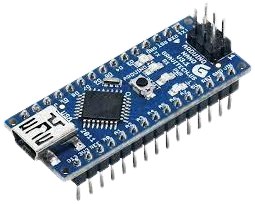

Ilustración 21. Arduino nano

3.1.5 Estructura del robot

En la ilustración 9, se muestra a P.E.P.E (Prototipo de Enseñanza de Programación y Electrónica), el robot móvil utilizado en este proyecto, Su estructura está compuesta por materiales reciclados, en su mayoría composición de plástico, con un esqueleto metálico, sus dimensiones son 50 cm de ancho, 50 cm de largo y 1.20 cm de alto, aproximadamente.

Ilustración 29. Estructura de P.E.P.E

3.1.6 Placa madre

En la ilustración 10, se muestra la placa madre, encargada de impartir las instrucciones a todos los dispositivos conectados a ella; Para esto cuenta con un Arduino UNO donde se conecta a un ordenador para la simulacion del mando de Xbox, el Arduino UNO usa un software que le indica a P.E.P.E que instrucción ejecutar por medio del mando de Xbox, en esta placa se conectan dos puente H (L298N) que se encarga de suministrar la corriente e invertir el sentido del giro de los motores en cada momento determinado, todo alimentad con una batería de 12 voltios.

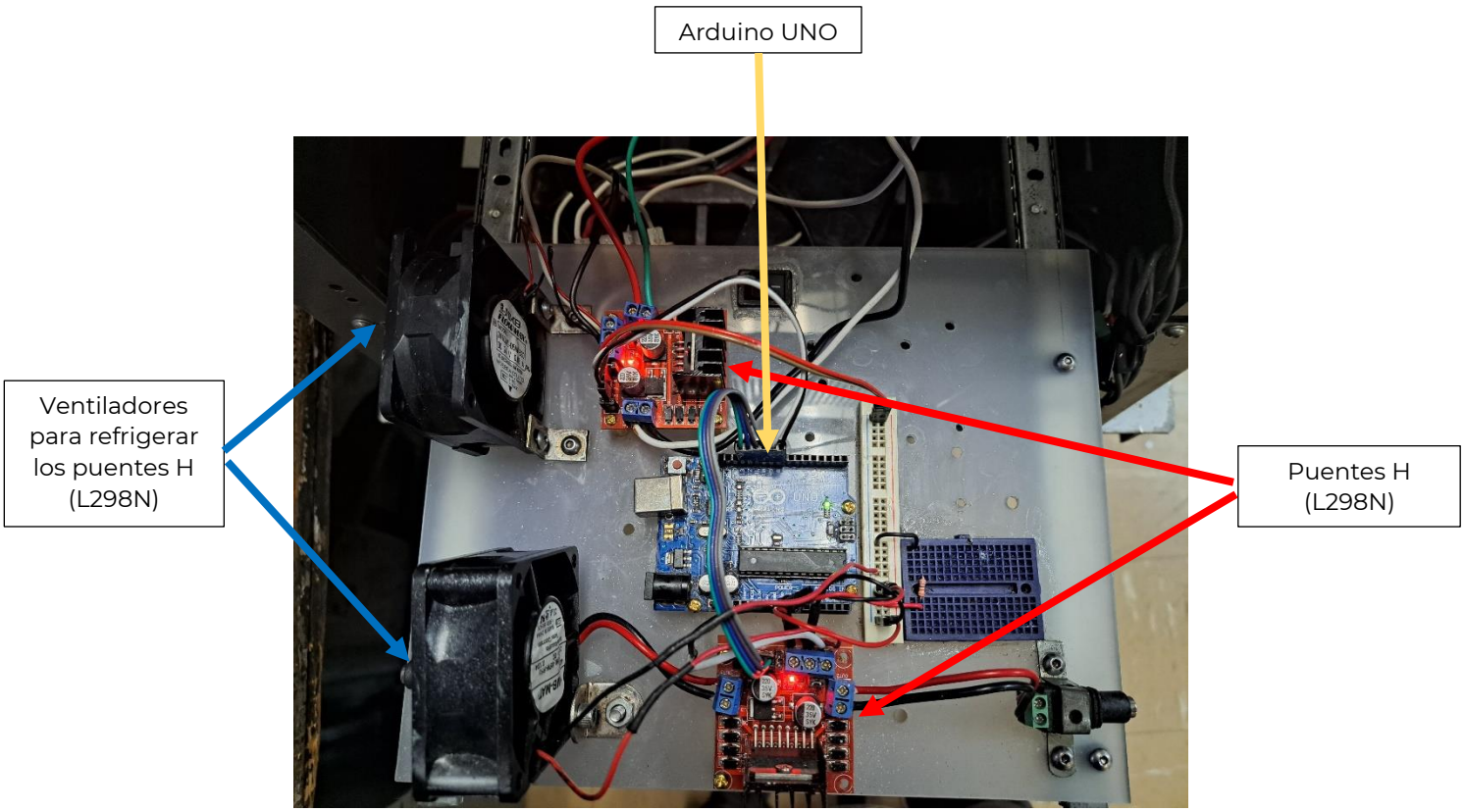

Ilustración 34. Placa base

3.1.7 SOFTWARE Y CÓDIGO

En el siguiente apartado mostramos el código que maneja el microcontrolador de Arduino uno para el total manejo de uso del robot móvil P.E.P.E.

```
/* UNIVERSIDAD AUTONOMA METROPOLITANA

Descripción: Control del Robot PEPE utilizando ARDUINO.

M en C Omar Lucio Cabrera Jiménez        MAY 2017    */

#include<Servo.h>

Servo servoCbzaNo;
Servo servoCbzaSi;

int incrServoCbzaNo = 5;
int incrServoCbzaSi = 5;

unsigned long milisActual = 0;
unsigned long milisAnterior = 0;

long intervalo = 250;

int posServoCbzaNo = 90;
int posServoCbzaSi = 90;

int bzoder = LOW;
int bzoizq = LOW;

int cbz_no = LOW;
int cbz_si = LOW;
int cmdnvo = 64;
int cmdant = 64;

void setup()  {
  Serial.begin(9600);   // Comunicación Serie
  for (int Pin = 8; Pin <= 13; Pin++)
    pinMode(Pin, OUTPUT); // Pines de salida
  //servoCbzaNo.attach(6);
  pinMode(6, OUTPUT);
  servoCbzaSi.attach(5);
}

void vehiculo (int a, int b, int c, int d)  {
  digitalWrite(10, a);
  digitalWrite(11, b);
  digitalWrite(12, c);
  digitalWrite(13, d);
}

void paro(){
  vehiculo(LOW, LOW, LOW, LOW);
  delay(50);
}

void loop()
{
  if (Serial.available() > 0)
  {
```

Derechos Reservados

Importación de la biblioteca Servo: se utiliza para controlar

Declaración de los objetos de los motores

Declaración de variables y pines

En este fragmento de código se configuran los pines del Arduino que se utilizan para controlar el motor del vehículo, los brazos y la cabeza

Esta función se utiliza para controlar el movimiento del vehículo, mediante la activación de los pines correspondientes

Esta función detiene el movimiento del vehículo

```
cmdnvo = Serial.read();    // Lectura de comando:

switch (cmdnvo)
{
  case 'D':  // Avanza
    if (cmdnvo != cmdant)
      paro();
    vehiculo(HIGH, LOW, HIGH, LOW);
    break;

  case 'I':  // Reversa
    if (cmdnvo != cmdant)
      paro();
    vehiculo(LOW, HIGH, LOW, HIGH);
    break;

  case 'R':  // Vuelta a la derecha
    if (cmdnvo != cmdant)
      paro();
    vehiculo(LOW, HIGH, HIGH, LOW);
    break;

  case 'A':  // Vuelta a la izquierda
    if (cmdnvo != cmdant)
      paro();
    vehiculo(HIGH, LOW, LOW, HIGH);
    break;

  case 'Z':  // Brazo izquierdo
    delay(100);
    if (bzoizq == 0)  {
      digitalWrite(9, HIGH);
      bzoizq = 1;
    }
    else  {
      digitalWrite(9, LOW);
      bzoizq = 0;
    }
    break;

  case 'E': // Brazo derecho
    delay(100);
    if (bzoder == 0)  {
      digitalWrite(8, HIGH);
      bzoder = 1;
    }
    else  {
      digitalWrite(8, LOW);
      bzoder = 0;
    }
    break;

  case 'S':  // Cabeza SI
    delay(100);
    if (cbz_si == 0)
    {
      digitalWrite(7, HIGH);
      cbz_si = 1;
    }
    else
    {
      digitalWrite(7, LOW);
```

Se verifica si hay datos disponibles en el puerto serie. Si hay datos disponibles, lee el comando y lo compara con una lista de posibles comandos en un switch

```
        cbz_si = 0;
      }
      break;

    case 'N':  // Cabeza NO
      delay(100);
      digitalWrite(6,HIGH);
      delay(100);
      digitalWrite(6,LOW);
      break;

    default:  {  // Desactivar las salidas:
      paro();
    }
  }
  cmdant = cmdnvo;
}

if ( cbz_no == HIGH)
{
   milisActual = millis();
   if(milisActual - milisAnterior > intervalo)
   {
      posServoCbzaNo += incrServoCbzaNo;
      if(posServoCbzaNo == 150 || posServoCbzaNo == 30)
         incrServoCbzaNo *= -1;
      servoCbzaNo.write (posServoCbzaNo);
      milisAnterior = milisActual;
   }
}

if ( cbz_si == HIGH)
{
   milisActual = millis();
   if(milisActual - milisAnterior > intervalo)
   {
      posServoCbzaSi += incrServoCbzaSi;
      if(posServoCbzaSi == 150 || posServoCbzaSi == 30)
         incrServoCbzaSi *= -1;
      servoCbzaSi.write (posServoCbzaSi);
      milisAnterior = milisActual;
   }
}
}
```

se ejecuta un movimiento en un servo motor conectado a la cabeza del robot, para hacer que la cabeza mire hacia la izquierda y hacia la derecha

Si el comando 'S' se recibe, el servo que mueve la cabeza hacia la posición "SI" comienza

3.1.8 Diagrama de flujo del proceso del funcionamiento

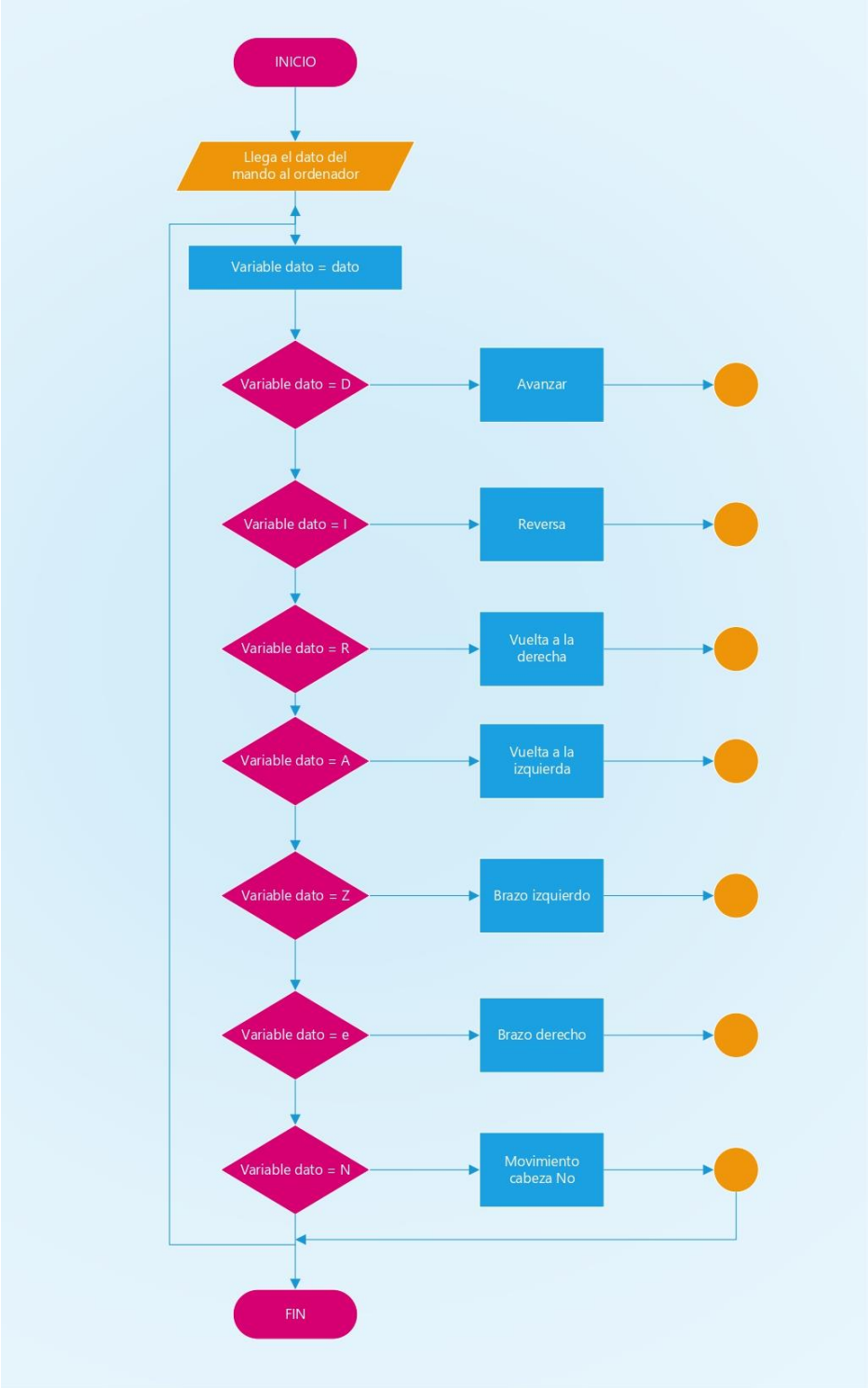

Ilustración 39. Proceso de Funcionamiento de P.E.P.E

3.2 Preparación del entorno

Antes de comenzar el desarrollo del programa, así como la elaboración de la página web, necesitaremos contar con algunas herramientas necesarias.

Node.js: Para la creación del programa que pueda manipular un Arduino usando JavaScript, necesitaremos un entorno de ejecución que este basado en JavaScript para ello contaremos con Node.js, que nos permitirá poder ejecutar el programa por parte del servidor.

(Para descargar node.js, necesitaremos entrar a su página oficial y descargar la opción de LTS)

Página oficial de node.js: Node.js (nodejs.org)

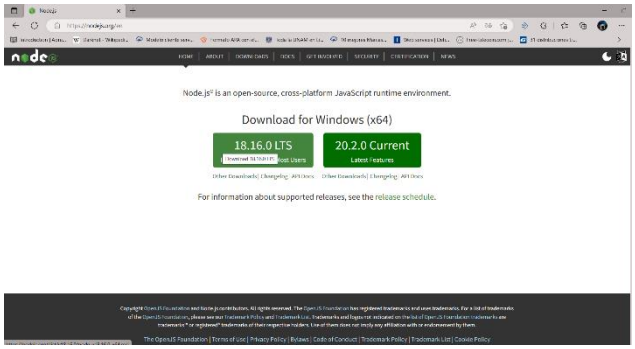

Ilustración 40. Página oficial de Node.js

Firmata: Para hacer posible la comunicación entre el Arduino UNO y el software, necesitaremos usar Firmata; Firmata es una biblioteca que implementa el protocolo de comunicación con el host del equipo. Esto nos permitirá escribir firmware personalizado sin tener que crear protocolos u objetos para el proyecto.

(Para usarlo, basta con descargar el IDE de Arduino)

Arduino IDE: El entrono de desarrollo integrado de Arduino nos permitirá desarrollar, diseñar y comunicar código de Arduino con la placa Arduino UNO, nosotros la usaremos para poder cargar la biblioteca de Firmata, Para ello necesitáremos descargar el IDE y conectar nuestra placa de Arduino.

(Para descargar Arduino IDE, necesitaremos entrar a su página oficial y descargar la opción Legacy IDE (1.8.X))

Página oficial de Arduino: Software | Arduino

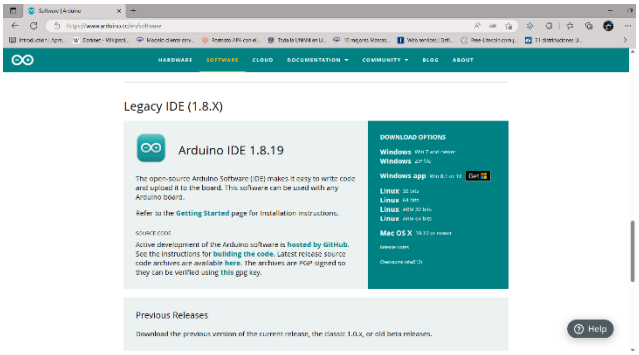

Ilustración 41. Página oficial de Arduino

Johnny-five: Para poder comunicarnos usando JavaScript al microcontrolador de Arduino usando el protocolo Firmata, necesitaremos de una plataforma para el uso de Robótica, Johnny-five es un Framework que usaremos para poder comunicarnos (mandar el código) a la placa de Arduino que utilizaremos y asi comunicarnos usando el lenguaje JavaScript.

(Para usar el Framework jhonny-five, tendremos que instalarlo en la carpeta raíz del documento).

Express: Para la creación de un servidor web local necesitaremos usar Express, este Framework de node.js nos servirá para desarrollar la aplicación web y que nos proporcionará una forma rápida y sencilla para crear un servidor web local.

(Para usar el Framework Express, tendremos que instalarlo en la carpeta raíz del documento).

Socket.io: Para la comunicación de la página web con el código que estará ejecutándose en el microcontrolador de Arduino, necesitaremos emplear de Socket.io la cual es una biblioteca de JavaScript que permite la comunicación bidireccional en tiempo real entre un servidor Node.js y un cliente web.

(Para usar la biblioteca Socket.io, tendremos que instalarlo en la carpeta raíz del documento).

3.3 Desarrollo Del Programa

Antes de la realización del programa, explicaremos de una forma no técnico el funcionamiento del programa, explicaremos en un diagrama basado en lenguaje UML que ayudara a la comprensión.

3.3.1 Introducción Y Aclaraciones Previas

Se ha diseñado una página Web la cual controla los movimientos de P.E.P.E, el cual se representa en el siguiente diagrama de casos de uso; Por un lado, el usuario tiene control del movimiento de las llantas, (Hacia adelante, hacia atrás, hacia la derecha, hacia la izquierda), al igual de la posibilidad de mover los brazos (izquierdo y derecho), cuenta con la posibilidad de mover las garras al mismo tiempo que mover la cabeza haciendo un movimiento de “NO”.

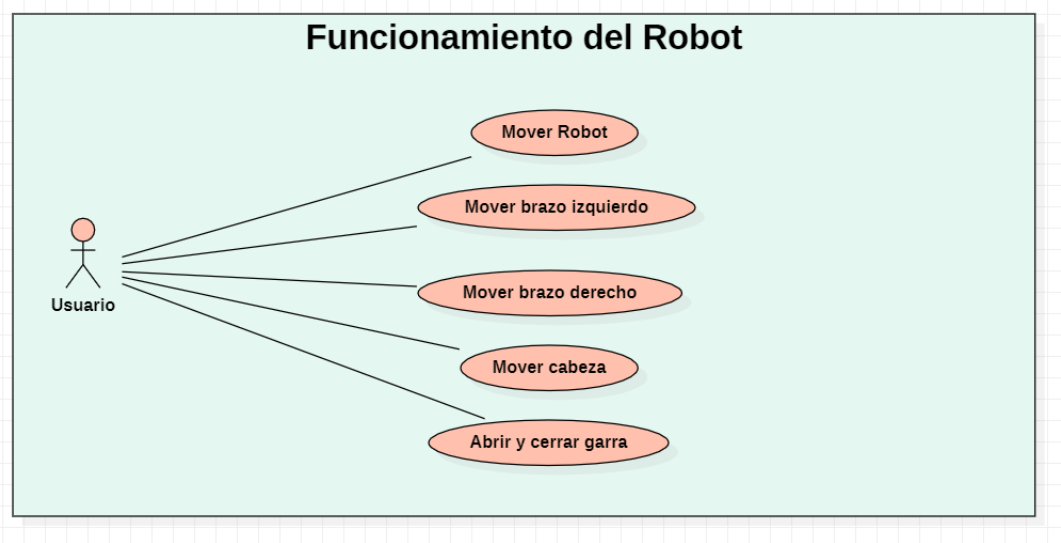

Ilustración 42. Diagrama de Casos de Uso

Estos 4 tipos de movimientos pueden ser controlados desde la página Web, esta forma sigue la misma programación del control, con la diferencia que la elaboración del código está elaborado en JavaScript y conlleva un manejo sencillo desde la web.

3.3.2 Programa Principal

Para iniciar con el programa necesitaremos crear nuestra carpeta principal del proyecto la cual llamaremos “Control-PEPE”.

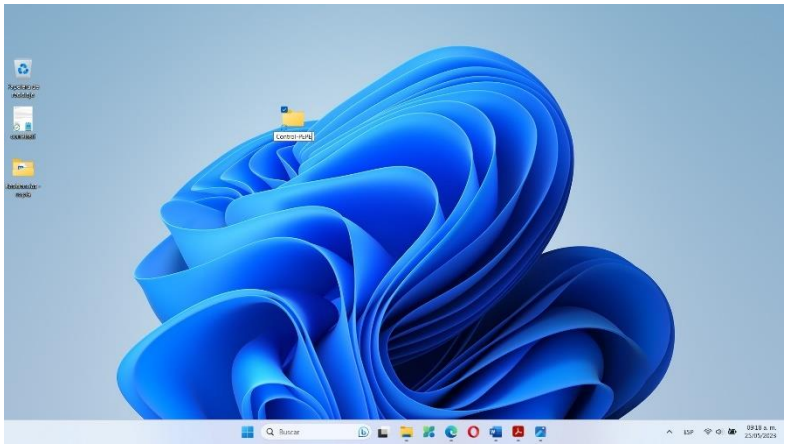

Ilustración 43. Proyecto Control P.E.P.E

La carpeta se encuentra vacía, se trabajará con la terminal por lo tanto necesitaremos abrir la terminal y entrar a la ubicación raíz de la carpeta.

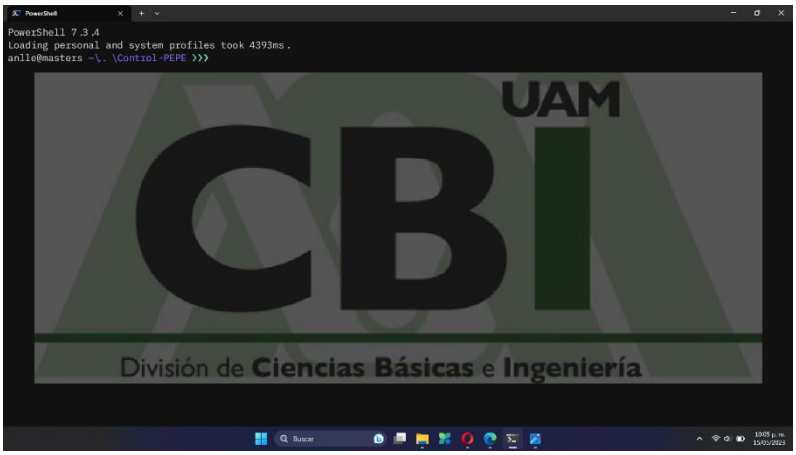

Ilustración 44. Carpeta Raíz

Antes de la creación del código necesitaremos primero instalar las herramientas que ya hemos tratado en este documento, instalaremos una herramienta, una por una adentro de la carpeta raíz Control-PEPE, (Recordemos ya tener instalado node.js).

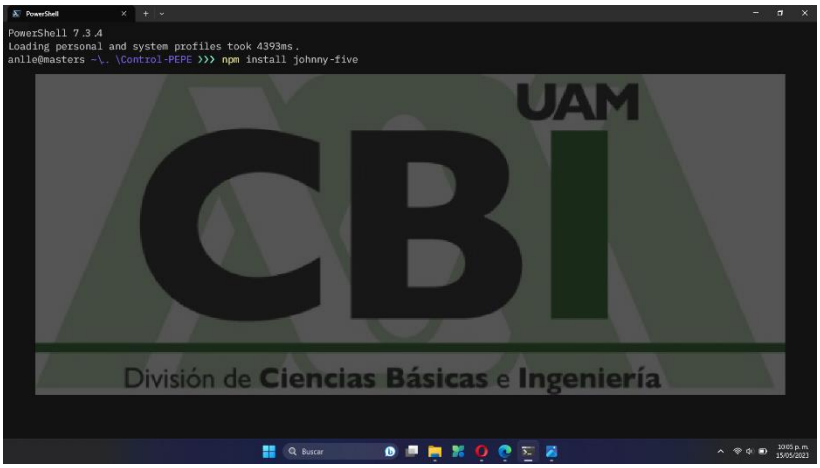

Ilustración 45. Instalación de Jhonny-Five

Ilustración 46. Johnny -Five instalado correctamente

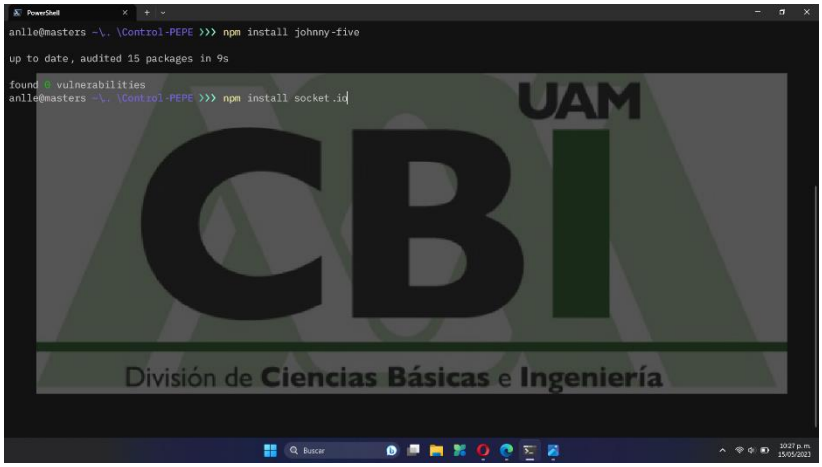

Ilustración 47. Instalación de Socket.io

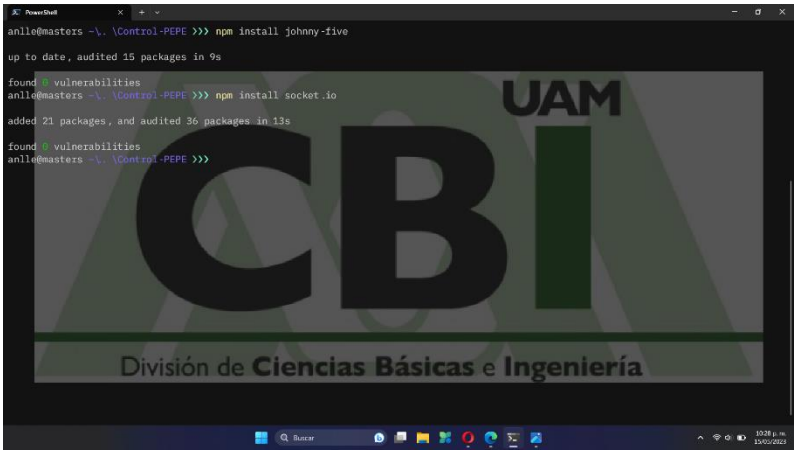

Ilustración 48. Socket.io instalado correctamente

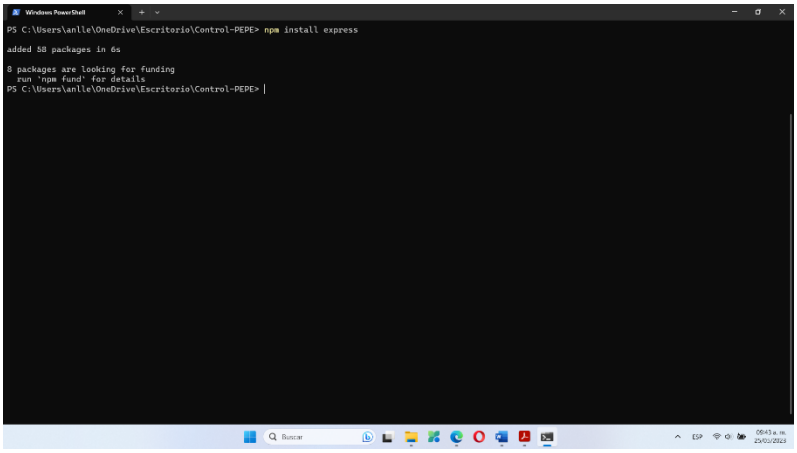

Ilustración 49. Instalación de Express

Teniendo las bibliotecas completamente instaladas, necesitaremos usar un editor de codigo, y abrir el archio .json que se genero, esto para observar las bibliotecas instaladas y sus versiones.

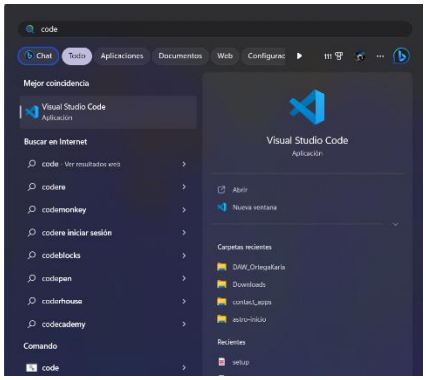

Ilustración 50. Visual Studio Code

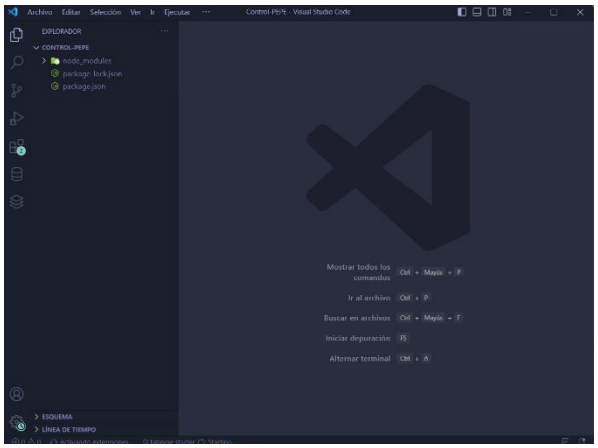

Ilustración 51. Proyecto Control-PEPE en Visual Studio Code

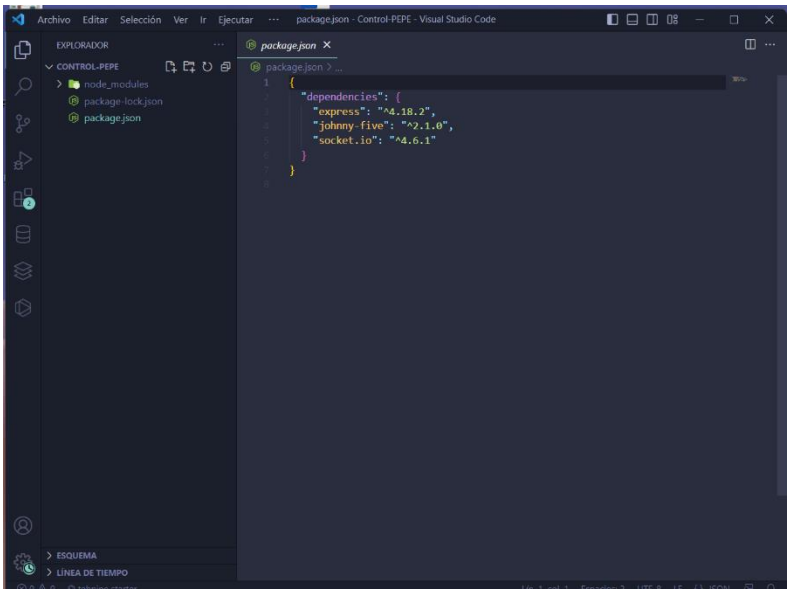

Ilustración 52. package.json

Completando la instalación completa y checando la visualización de las versiones de cada biblioteca, ahora lo que haremos es crear nuestro archivo llamado “app.js”, donde escribiremos el programa completo de la funcionalidad del robot P.E.P.E.

3.3.3 Funcionamiento Del Código

El código de P.E.P.E tiene la función de crear un servidor web que permite controlar los movimientos y el seguimiento de P.E.P.E conectado a un Arduino a través de una interfaz web.

Al ejecutar este código, se inicia un servidor web utilizando Express. Cuando un cliente se conecta al servidor, se establece una conexión Socket.IO entre el servidor y el cliente, lo que permite la comunicación bidireccional en tiempo real.

Cuando el cliente envía comandos a través de la conexión Socket.IO, el servidor responde ejecutando el movimiento indicado conectado al Arduino. Los comandos disponibles incluyen cambiar el movimiento a valores específicos (hacia adelante, hacia atrás, hacia la izquierda, hacia la derecha), mover ambos brazos, mover la cabeza en movimiento de “SÍ” y “NO”.

Este código tiene la finalidad de construir una aplicación web que puede controlar y manipular a P.E.P.E conectado a un Arduino a través de una interfaz de usuario en tiempo real. Esto es útil para el control remoto de los componentes físicos del robot a través de una aplicación web, controlado por computadora o cualquier dispositivo conectado al servidor.

3.3.4 Código JavaScript

```
const { Board, Servo } = require('johnny-five');
const express = require('express');
const http = require('http');
const socketIo = require('socket.io');

const app = express();
const server = http.createServer(app);
const io = socketIo(server);

const board = new Board();

let servoCbzaNo;
let servoCbzaSi;
let incrServoCbzaNo = 5;
let incrServoCbzaSi = 5;
let posServoCbzaNo = 90;
let posServoCbzaSi = 90;
let cmdnvo = 64;
let cmdant = 64;

board.on('ready', () => {
  servoCbzaNo = new Servo({ pin: 6 });
  servoCbzaSi = new Servo({ pin: 5 });

  // Pines de salida
  board.pinMode(8, board.MODES.OUTPUT);
  board.pinMode(9, board.MODES.OUTPUT);
  board.pinMode(10, board.MODES.OUTPUT);
  board.pinMode(11, board.MODES.OUTPUT);
  board.pinMode(12, board.MODES.OUTPUT);
  board.pinMode(13, board.MODES.OUTPUT);

  io.on('connection', (socket) => {
    console.log('A client has connected!');

    socket.on('data', (data) => {
      cmdnvo = data.toString();
      switch (cmdnvo) {
        case '0':
          paro();
          break;
        case '1':
          vehiculo(0, 0, 0, 0);
```

Se importan los módulos necesarios: johnny-five, express, http y socket.io.

Se crea una instancia de la aplicación Express y se configura el servidor HTTP.

Se crea una instancia de socket.io para habilitar la comunicación en tiempo real con los clientes.

Se crea una instancia de Board de johnny-five para interactuar con la placa de Arduino.

Se definen variables y objetos para los servos de los brazos y la cabeza, y se inicializan algunas variables de control.

Se definen variables y objetos para los servos de los brazos y la cabeza, y se inicializan algunas variables de control.

Cuando la placa Arduino está lista (board.on('ready')), se configuran los pines de salida y se establece la lógica para la comunicación con los clientes a través de socket.io.

```
          break;
        case '2':
          vehiculo(1, 0, 0, 1);
          break;
        case '3':
          vehiculo(0, 1, 1, 0);
          break;
        case '4':
          vehiculo(0, 0, 1, 1);
          break;
        case '5':
          vehiculo(1, 0, 1, 0);
          break;
        case '6':
          vehiculo(0, 1, 0, 1);
          break;
        case '7':
          mueveCabezaNo();
          break;
        case '8':
          mueveCabezaSi();
          break;
        default:
          break;
      }
      cmdant = cmdnvo;
    });

    socket.on('disconnect', () => {
      console.log('A client has disconnected!');
    });
  });
});

function delay(ms) {
  return new Promise((resolve) => setTimeout(resolve, ms));
}

function paro() {
  board.digitalWrite(8, board.LOW);
  board.digitalWrite(9, board.LOW);
  board.digitalWrite(10, board.LOW);
  board.digitalWrite(11, board.LOW);
}
```

Dentro de la lógica de comunicación con los clientes, se define la acción a tomar según los comandos recibidos. Los comandos están representados por números (0-8) y se ejecutan diferentes funciones dependiendo del comando recibido.

```
function vehiculo(m1, m2, m3, m4) {
  board.digitalWrite(11, m1 ? board.HIGH : board.LOW);
  board.digitalWrite(12, m2 ? board.HIGH : board.LOW);
  board.digitalWrite(13, m3 ? board.HIGH : board.LOW);
  board.digitalWrite(10, m4 ? board.HIGH : board.LOW);
}

function mueveCabezaNo() {
  posServoCbzaNo += incrServoCbzaNo;
  if (posServoCbzaNo > 180 || posServoCbzaNo < 0) {
    incrServoCbzaNo *= -1;
  }
  servoCbzaNo.to(posServoCbzaNo);
}

function mueveCabezaSi() {
  posServoCbzaSi += incrServoCbzaSi;
  if (posServoCbzaSi > 180 || posServoCbzaSi < 0) {
    incrServoCbzaSi *= -1;
  }
  servoCbzaSi.to(posServoCbzaSi);
}

server.listen(3000, () => {
  console.log('Server is running on port 3000!');
});
```

Las funciones paro, vehiculo, mueveCabezaNo y mueveCabezaSi implementan la lógica específica para controlar el vehículo y los servos.

Finalmente, el servidor se inicia y se escucha en el puerto 3000.

Este código configura un servidor web que sirve un archivo HTML y utiliza Socket.IO para permitir el control de P.E.P.E conectado a un Arduino a través de una interfaz web. Los eventos de Socket.IO se utilizan para recibir comandos desde el cliente y controlar el movimiento del robot.

3.3.5 Diagrama De Secuencia

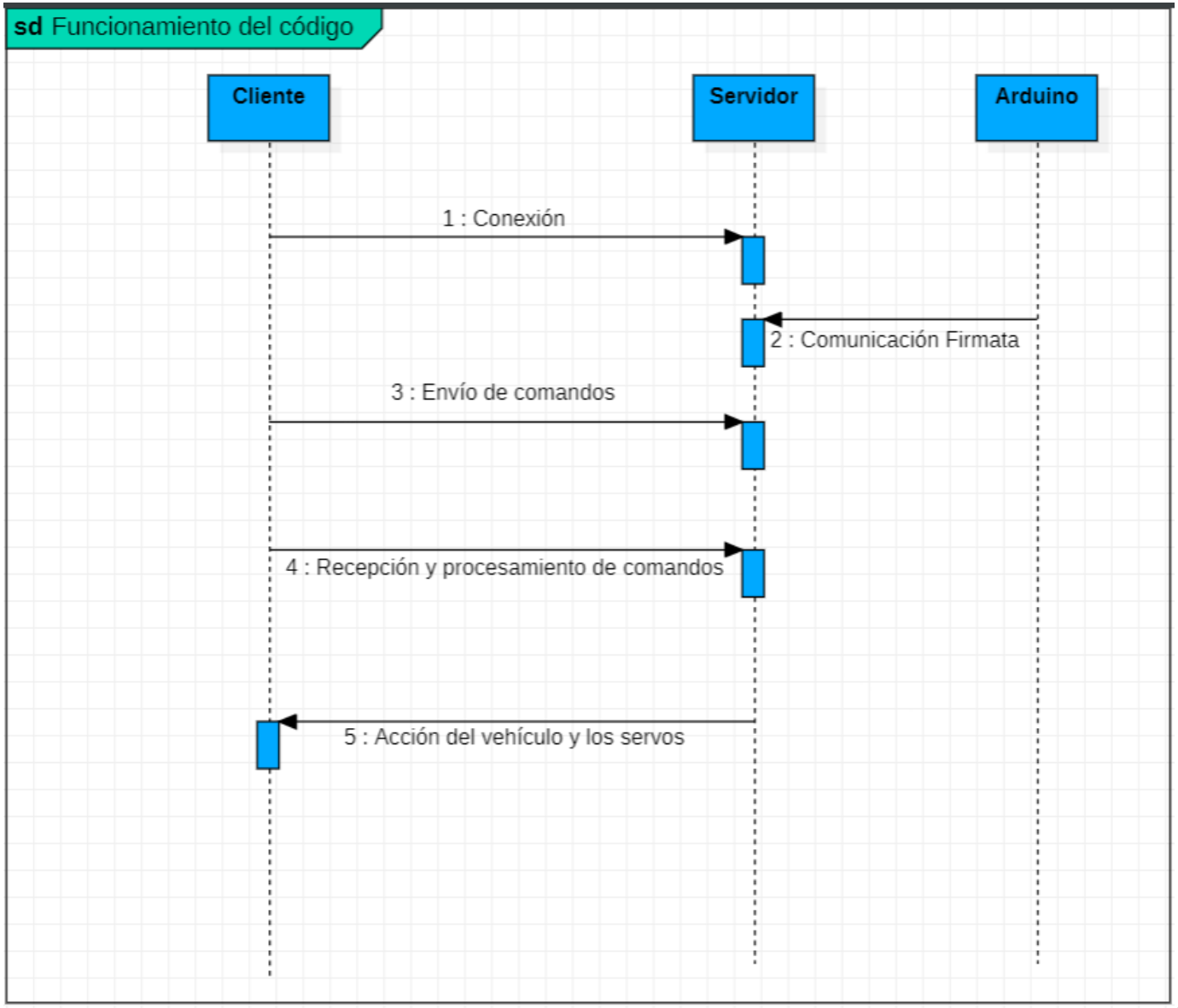

Ilustración 54. Diagrama de Secuencia

3.3.6 Interfaz Web

La página web esta creada en HTML usa hojas en estilo en cascada (CSS), y es una página web estática y está siendo ejecutada mediante un servidor local creado en express; Para acceder a la página web es necesario ejecutar el código en la terminal y entrar en la direccion IP del servidor y usando el número de puerto asignado.

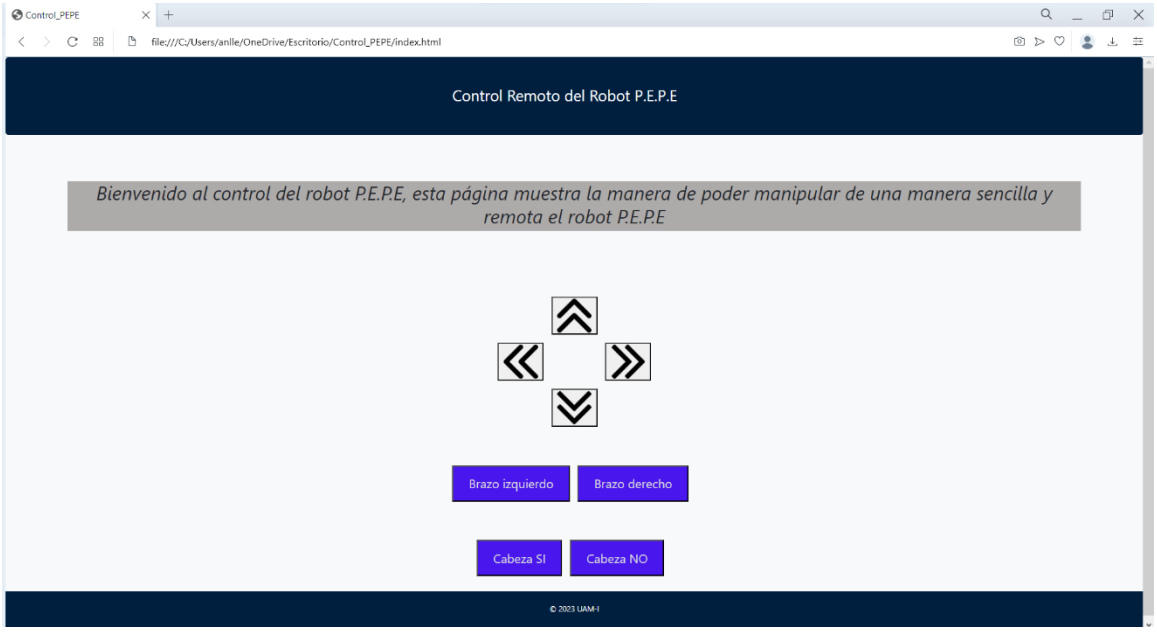

Ilustración 55. Interfaz Web Control P.E.P.E

En el código de JavaScript donde generamos todas las instrucciones de control del robot creamos métodos necesarios usando el framework Johnny-Five, una vez que el código este en ejecución lo que hace es mandarnos la interfaz gráfica.

La interfaz gráfica está elaborada en HTML, también usamos Hojas de estilo en cascada (CSS) para poder darle diseño y formato a los botones.

Los botones mediante scripts son los encargados de llamar a las funciones JavaScript que se utilizan para enviar comandos al robot a través de Socket.IO.

3.3.7 Código de la INTERFAZ WEB

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Control_PEPE</title>
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.cs
s" integrity="sha384-
1q8mTJOASx8j1Au+a5WDVnPi2lkFfwwEAa8hDDdjZlpLegxhjVME1fgjWPGmkzs7"
crossorigin="anonymous">
  <link rel="stylesheet"
href="node_modules/bootstrap/dist/css/bootstrap.min.css">
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/4.4.1/socket.io.js">
</script>

  <script
src="node_modules/bootstrap/dist/js/bootstrap.bundle.min.js"></script>

</head>

<body>
  <header>
    <nav class="navbar">
      <div class="container">
        <a class="navbar-brand" href="imagen/abajo.png">Control Remoto del
Robot P.E.P.E</a>
      </div>
    </nav>
    <div class="header-container">
      <div class="container" style="background-color: rgb(173, 170,
170);"> <!-- Contenedor con color gris -->
        <i><h1>Bienvenido al control del robot P.E.P.E, esta página
muestra la manera de poder manipular de una manera sencilla y remota el
robot P.E.P.E</h1></i>
      </div>
    </div>
  </header>

  <br>
```

Abrimos la etiqueta HTML en su versión 5

Enlaces para vincular el estilo CSS, Ajax y boostrap

Librerias para usar sockets y boostrap

Parte del cuerpo donde indicamos contenedores

```
  <br>
  <br>

  <center>
    <div class="grid-container">
      <div class="grid-item"></div>
      <div class="grid-item up">
        <button id="forwardButton" onclick="moveForward()"><img
src="imagen/arriba.png" alt="Avanzar"></button>
      </div>
      <div class="grid-item"></div>
      <div class="grid-item left">
        <button id="leftButton" onclick="moveLeft()"><img
src="imagen/izquierdo.png" alt="Girar a la izquierda"></button>
      </div>
      <div class="grid-item"></div>
      <div class="grid-item right">
        <button id="rightButton" onclick="moveRight()"><img
src="imagen/derecha.png" alt="Girar a la derecha"></button>
      </div>
      <div class="grid-item"></div>
      <div class="grid-item down">
        <button id="reverseButton" onclick="moveBackward()"><img
src="imagen/abajo.png" alt="Reversa"></button>
      </div>
      <div class="grid-item"></div>
    </div>
  </center>

  <br><br>

  <div class="button-container">
    <button id="armLeftButton" onclick="moveArmNo()">Brazo
izquierdo</button>
    <button id="armRightButton" onclick="moveArmSi()">Brazo
derecho</button>
  </div>

  <br><br>

  <div class="button-container">
    <button id="headYesButton" onclick="sendCommand('S')">Cabeza
SI</button>
    <button id="headNoButton" onclick="sendCommand('N')">Cabeza
NO</button>
```

Botones para controlar los métodos del robot (Hacia adelante, hacia atrás, hacia la izquierda, hacia la derecha)

Botón para controlar el movimiento de los brazos

Botón para controlar el movimiento de la cabeza

```
  </div>

  <footer class="footer">
    <p>© 2023 UAM-I</p>
  </footer>

  <script>
    const socket = io();

    function sendCommand(command) {
      socket.emit('command', command);
    }
  </script>

  <script
src="node_modules/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
</body>

</html>
```

Conexión del socket para mandar eventos de comandos a través del socket

3.3.8 CSS DE LA INTERFAZ WEB

```
<style>
    body {
      background-color: #f8f9fa;
    }

    .container {
      margin-top: 20px;
    }

    h1 {
      text-align: center;
      font-weight: normal;
    }

    .btn {
      background-color: transparent;
      border: none;
      padding: 0;
    }

    .btn img {
```

```
    width: 30px;
    height: 30px;
  }

  .navigation-buttons {
    display: flex;
    justify-content: center;
    align-items: center;
    flex-wrap: wrap;
    margin-top: 20px;
  }

  .footer {
    background-color: #001f3f;
    color: white;
    text-align: center;
    padding: 15px;
    margin-top: 20px;
  }

  .grid-container {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    grid-template-rows: repeat(3, 1fr);
    grid-gap: 10px;
    width: 200px; /* Ajusta el ancho según tus necesidades */
    margin: 0 auto; /* Centra el contenedor */
  }

  .grid-item {
    position: relative;
  }

  .grid-item img {
    width: 100%;
    height: 100%;
    object-fit: cover;
  }

  .center {
    grid-row: 2/3;
    grid-column: 2/3;
  }

  .btn {
```

```
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    background-color: transparent;
    border: none;
    padding: 0;
  }

  .btn img {
    width: 30px;
    height: 30px;
  }

  .button-container {
    display: flex;
    justify-content: center;
    align-items: center;
    margin-top: 20px;
  }

  .button-container button {
    margin-right: 10px;
    background-color: rgb(73, 23, 237); /* Color azul para los botones */
    color: rgb(221, 221, 221); /* Texto en blanco para los botones */
    padding: 10px 20px; /* Ajusta del padding */
    font-size: 16px; /* Ajuste del tamaño de fuente */
  }

  /* Estilos para el navbar */
  .navbar {
    background-color: #001f3f; /* Color azul fuerte */
    padding: 15px;
  }

  .navbar-brand {
    color: white;
    font-size: 20px;
  }

  .header-container {
    background-color: #f8f9fa; /* Color gris para el contenedor del header
*/
    padding: 20px;
    margin-bottom: 20px;
```

```
    }

    .button-container button:hover {
      box-shadow: 0 0 5px rgba(0, 0, 0, 0.3); /* sombra al pasar el cursor
*/
    }

  </style>
```

3.3.9 Scripts de la INTERFAZ WEB

```
<script>
    const socket = io();
    function sendCommand(command) {
      socket.emit('command', command);
    }

    function moveForward() {
      sendCommand('1');
    }

    function moveBackward() {
      sendCommand('2');
    }

    function moveLeft() {
      sendCommand('3');
    }

    function moveRight() {
      sendCommand('4');
    }

    function stop() {
      sendCommand('0');
    }

    function moveArmNo() {
      sendCommand('7');
    }

    function moveArmSi() {
      sendCommand('8');
    }
  </script>
```

Creamos una instancia de socket

Funciones que definen por el un parámetro sendCommand. Toma un parámetro command y dentro de la función, emite un evento llamado 'command' a través del socket creado anteriormente para hacer varias funciones dependiendo el numero indicado

Capítulo 4: RESULTADOS

Para mostrar el resultado mostramos los pasos siguientes de la ejecución:

Cuando conectemos el Arduino al ordenador mediante un cable serial, necesitaremos entrar al IDE de Arduino para corroborar el número del puerto asignado por el ordenador.

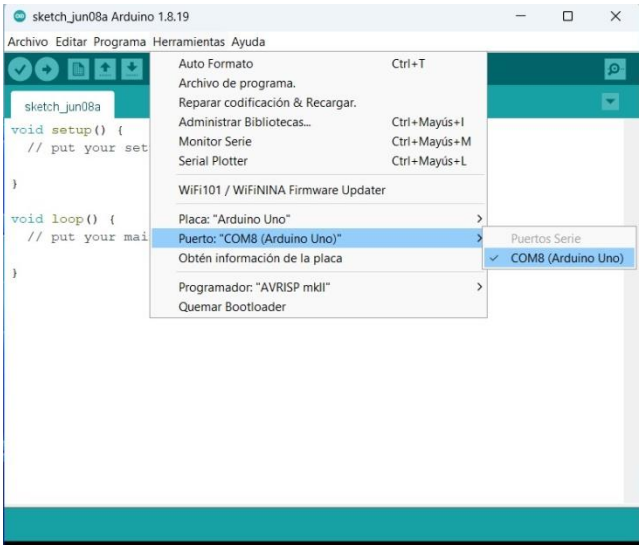

Ilustración 57. Puerto COM8

Ya conociendo el puerto asignado lo que haremos es cargar el protocolo de comunicación Firmata al Arduino, para ello haremos lo siguiente:

1) En el menú Archivo, opción ejemplos, Seleccionar el menú Firmata:

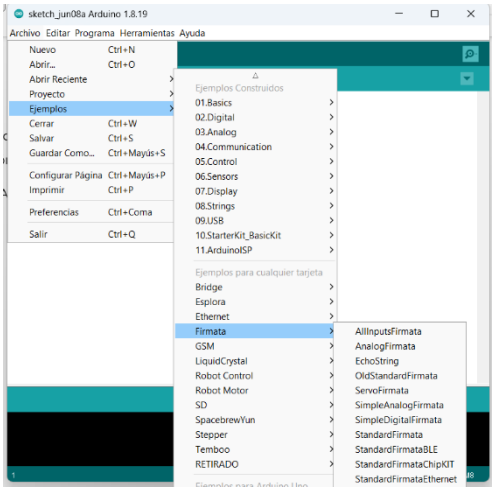

Ilustración 58. Protocolo Firmata

2) De toda la lista de protocolos Firmata Seleccionaremos la Opción 'StandardFirmataPlus'

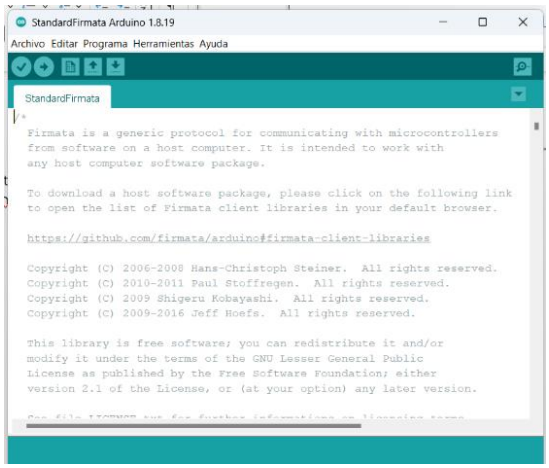

Ilustración 59. StandardFirmataPlus

3) Ahora debemos subir el código al Arduino

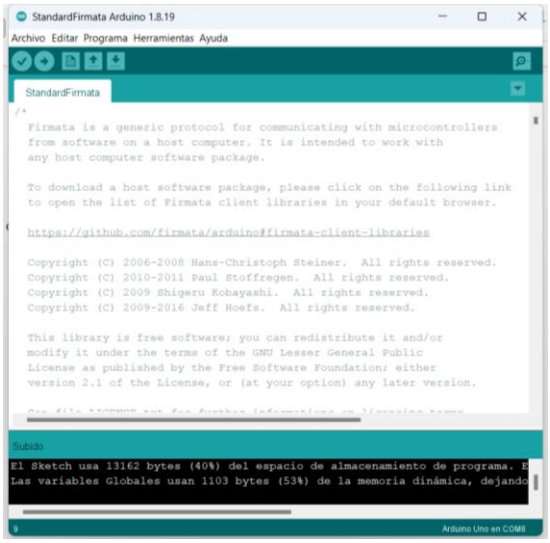

Ilustración 60. Protocolo Subido correctamente al Arduino UNO

Una vez cargado el protocolo de Firmata procederemos a correr el sistema.

Nota: *Este paso solo se realiza una vez, o cada que se cambie el Arduino, Ya que el Arduino retiene el protocolo cuando se apaga.*

Ahora lo que haremos es entrar a la terminal para desde ahí entrar al directorio raíz de nuestro proyecto.

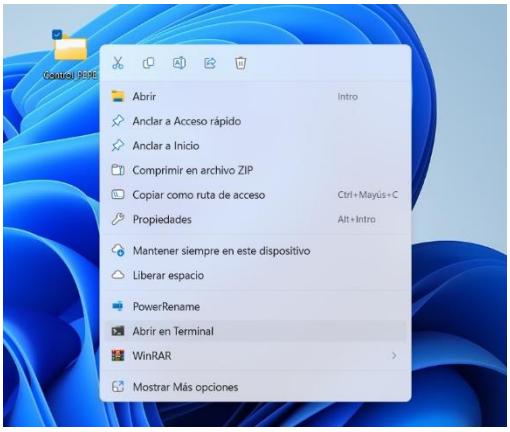

Ilustración 61. Carpeta del Proyecto P.E.P.E

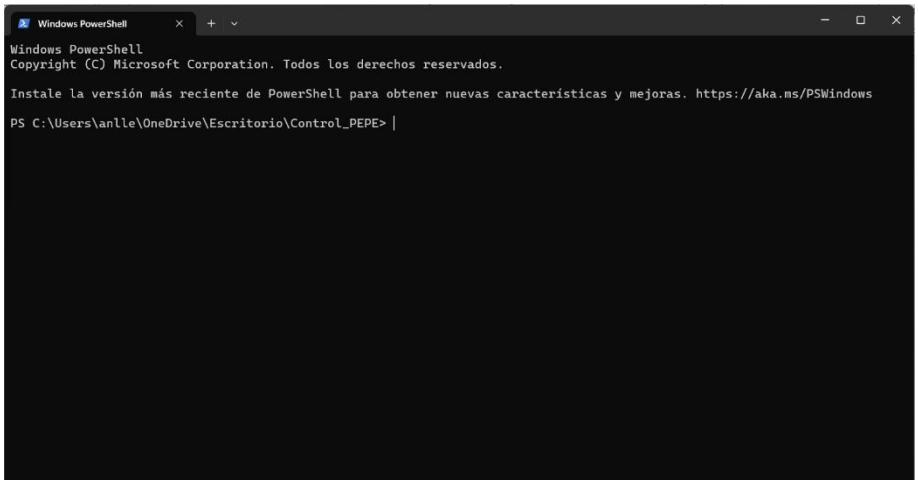

Ilustración 62. Terminal y Ubicación Raíz del Proyecto

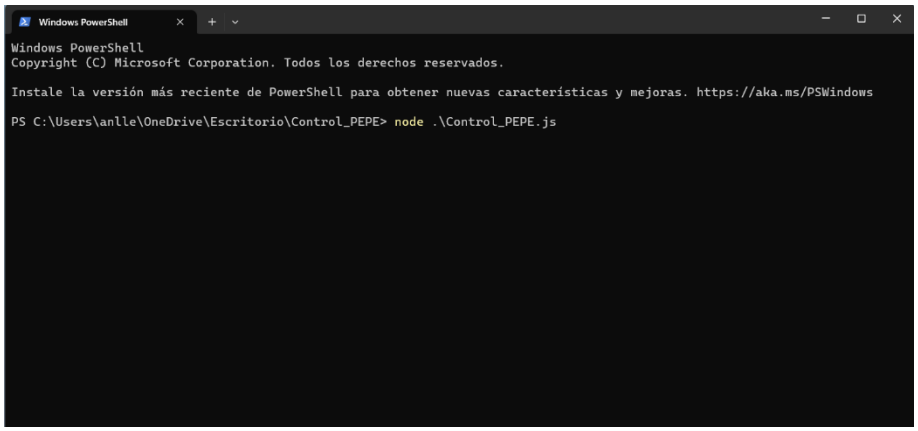

Ilustración 63. Ejecución del Programa

Una vez ejecutemos el programa, vamos a cualquier navegador y lo que tenemos que escribir en el URL es el localhost y el puerto asignado, en este caso es en *http://localhost:5000, y nos llevara directamente al servidor donde este alojado la interfaz gráfica.*

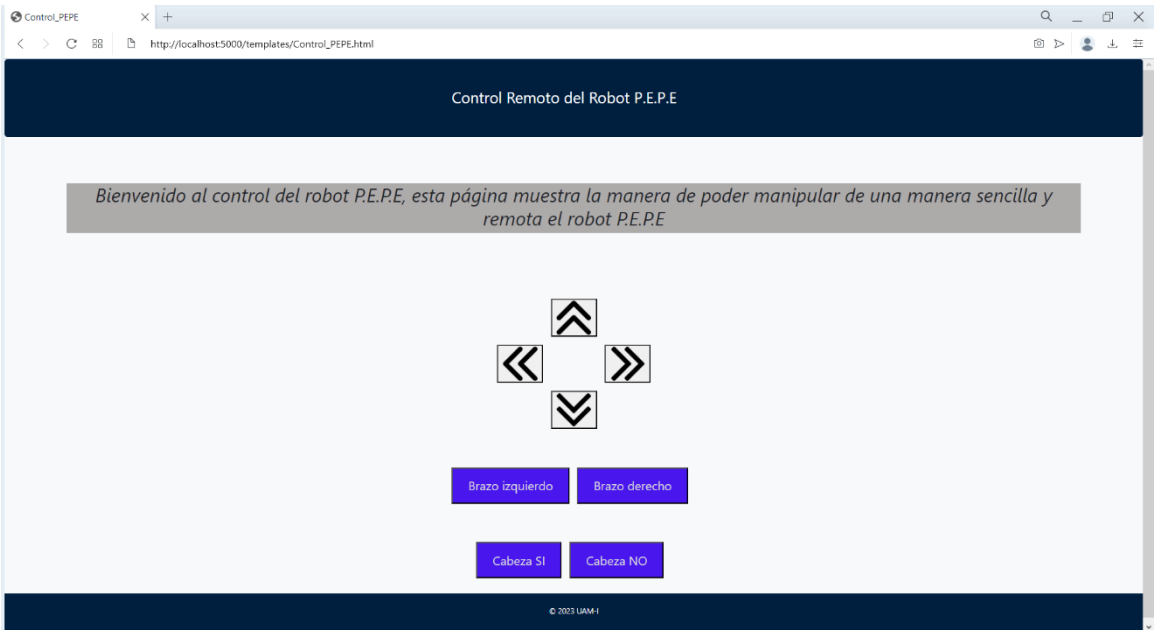

Ilustración 64. Control Remoto de P.E.P.E

Mientras estemos conectado a la misma red local, podemos entrar desde el teléfono o cualquier dispositivo en donde podamos entrar a cualquier navegador.

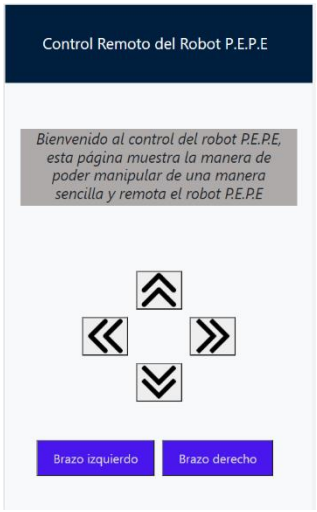

Ilustración 65. Control P.E.P.E en móvil

Capítulo 5: CONCLUSIONES

El proyecto consiguió llegar a la meta de los objetivos establecidos, se consiguió desarrollar un sistema más allá de una aplicación, Un sistema cliente-servidor donde nos permite controlar y monitorear el movimiento y la interacción del robot P.E.P.E con un gran rendimiento y una fortuita eficiencia, así como la gran facilidad para que cualquier usuario que necesite operar a P.E.P.E lo pueda manipular de una manera muy simple.

Se presento la metodología implementada para desarrollar el sistema "Control_PEPE" controlado a través de internet mediante una página web, que termino siendo un éxito así logrando conjuntar varias tecnologías como son el uso de una plataforma de desarrollo Arduino para la comunicación usando Firmata, la integración de Johnny-five y sockets para gestionar la comunicación a través de internet y la programación de la página web en código HTML y el uso de la tecnología Ajax y sockets para actualizar y comunicar en tiempo real los datos en la página web, *como dato importante es totalmente indispensable el estar en una conexión de área local teniendo acceso a internet en todo momento para que pueda funcionar el sistema cliente-servidor de este proyecto.*

5.1 Recomendaciones

Se recomienda siempre checar las versiones de cada herramienta implementada cada cierto tiempo, ya que las actualizaciones de dichas tecnologías usadas en este sistema cliente-servidor, son de prioridad, ya que, si las actualizaciones no corresponden correctamente con las versiones usadas con el protocolo, con los framework y con el lenguaje de programación no son compatibles el sistema no podrá ejecutarse de manera correcta.

5.2 Posibles Mejoras

Se ha mencionado anteriormente que, para la ejecución del sistema, se debe tener acceso al internet para que se pueda usar en tiempo real el control y monitoreo del robot, pero para posibles mejoras del sistema podría implementarse algunos cambios en el uso de las tecnologías, las cuales podrían ser las siguiente, esto en base que no necesite el uso de internet para su control y monitoreo:

Cambio al Protocolo TCP/IP: Para una comunicación donde no se necesite traer bibliotecas de web sockets y así haya una comunicación en tiempo real; Cambiar del protocolo Firmata al uso del protocolo TCP/IP para que pueda existir una conexión directa a través de la red de área local hacia el componente del Arduino usando una conexión directa atreves del Router así también liberándose del problema de búsqueda de los puertos disponibles del ordenador, cambiar el uso de puertos COM a usar directamente la direccion IP de la placa Arduino que está conectada a la red.

Actualización al Arduino UNO WIFI: Para hacer posible la conexión a través del protocolo TCP/IP, se necesitaría que la placa Arduino pueda conectarse directamente a la red de área local, para ello se pueden usar dos cosas, primero el integrarle un módulo WIFI a la placa Arduino ya empleada o cambiar la placa Arduino uno actual por una placa Arduino WIFI.

Capítulo 6: COMPETENCIAS DESARROLLADAS Y/O APLICADAS

En este apartado se especifican las dificultades que se han tenido a lo largo de su desarrollo, así como la experiencia obtenida.

6.1 DIFICULTADES

Una de las dificultades más grandes que se ha tenido en la elaboración del sistema, se centra directaremnete en dos cosas fundamentales:

a) ***Los puertos de comunicación:*** Para la posible conexión entre la placa Arduino uno que se utilizó en el proyecto y el ordenador se realiza mediante un cable USB, el cual ordenador se encarga de determinar el número de puerto COM (ya que se usa Windows) pero este detalle era el que no se lograba completar, ya que se tenía problemas de uso de puertos, puertos usados, puertos que no eran posibles conectarse, El detalle fue arreglado gracias a que en el código de JavaScript se le añadió el número establecido del puerto serie conectado al Arduino uno.

b) ***Las versiones de las herramientas:*** Otro detalle fue la adaptación del sistema empleado en términos de cada herramienta empleada, ya que se usaba un ordenador con Windows, cada herramienta debía acoplarse a las versiones que fueran compatibles entre sí, ya que algunos framework no lograron actualizarse a versiones posteriores con otras herramientas como por ejemplo, El framework de Johnny-five que se usa para poder manejar el Arduino con JavaScript, no recibió soporte para poder usar la versión más actual de node.js, así como el recibimiento del protocolo Firmata a la placa Arduino uno, estos detalles se lograron solucionar al buscar exhaustivamente las versiones compatibles de cada herramienta.

6.2 EXPERIENCIA OBTENIDA

Se ha completado de manera satisfactoria los requerimientos del proyecto, Siendo la primera vez en controlar un Robot usando un lenguaje de script, así como la primera vez en manejar y mandar ordenes vía web, se puede decir que durante la elaboración de este sistema cliente-servidor, se obtuvo nuevos conocimientos, y una profundización sobre las maneras en manipular un robot.

Por último, se destaca que se ha reforzado la habilidad de programación de robots, al crear el flujo de control con los métodos necesarios, implementación de módulos, así como el reconocimiento del empleo de Farmeworks que permiten el controlar y manipular los robots usando lenguajes de programación, así como la adecuación correcta de controlarles vía web para experimentar los distintos protocolos de comunicación que se pueden utilizar para la comunicación de los componentes.

Capítulo 7: FUENTES DE INFORMACIÓN

Arduino - Home. (s. f.). https://www.arduino.cc/

IBM Documentation. (s. f.). https://www.ibm.com/docs/es/aix/7.3?topic=concepts-sockets

Johnny-Five: The JavaScript Robotics & IoT Platform. (s. f.). Johnny-Five. https://johnny-five.io/

Rwaldron. (s. f.). *GitHub - rwaldron/johnny-five: JavaScript Robotics and IoT programming framework, developed at Bocoup.* GitHub. https://github.com/rwaldron/johnny-five

Hernández, O., López, E., & Pérez, V. M. (2016). Diseño y construcción de un brazo robot controlado mediante el módulo Arduino Mega a través de internet. Ciudad Mendoza, Ver.: Tesis de Licenciatura. Universidad Veracruzana.

Medina, J., Castro, N., Mejía, E., & Villafuerte, R. (2016). Aplicación móvil para el control de un brazo robot. Revista Iberoamericana de Producción Académica y Gestión Educativa [publicación en proceso], 1-22.

K. Taylor J. Trevelyan. "Australia's telerobot on the web" in: International Symposium on Industrial Robots, 1995, pp.

García, E. G. (2011). Diseño de un laboratorio robótico remoto para la enseñanza.

LabsLand. (s.f.-b). Robot arduino (código). Descargado de https://labsland.com/es/ labs/arduino-robot