UNCUYO
UNIVERSIDAD
NACIONAL DE CUYO

FACULTAD
DE INGENIERÍA

Programación Orientada a Objetos
Ingeniería en Mecatrónica

ROBOT TANQUE

Profesor: Mg. Ing. César Omar Aranda

Fecha: 9 de Marzo de 2023

Alumno: Mario Stefano Papetti Funes

Legajo: 11807

Tabla de contenido

INTRODUCCIÓN

El presente informe corresponde en el abordaje de un sistema mecatrónico desde el punto de vista de la programación orientada a objetos, con el fin de poder aplicar los conceptos relacionados a la asignatura Programación Orientada a Objetos dictada.

En el informe se toma cómo problemática el control de un robot móvil por medio de conexión bluetooth, el mismo debe de ser controlado tanto de forma local cómo de forma remota por diversos usuarios asignados, para ello, se propone cómo solución un sistema del estilo servidor cliente en donde se envíen los comandos necesarios para controlar el movimiento del robot.

Diseño de solución

El enfoque que se le da en el actual estado del proyecto será bajo el concepto de servidor cliente, donde el servidor (*en Python*) estará funcionando desde una computadora portátil y el cliente (*en C++*) en otro dispositivo del mismo tipo, ambos estarán funcionando bajo la misma red que en este caso será la red doméstica inalámbrica del presente autor, cabe aclarar que tanto el servidor cómo el cliente también podría funcionar bajo el mismo dispositivo y por medio de la comunicación local o *LocalHost* (127.0.0.1).

Se propone acompañar tanto del lado del servidor cómo del cliente con interfaces gráficas realizadas con los frameworks PyQT5 y QT6 respectivamente. También se propone utilizar el protocolo de comunicación XML-RPC para ambos programas.

Se aplican en los siguientes sub ítems modelado UML para el entendimiento del proyecto, tanto por el lado del servidor como el del cliente. Además, se aplicó el método de diseño de software del estilo modelo vista controlador (MVC) que consta de ciertos pasos para poder crear un sistema más ordenado y prolijo, brindando así su posibilidad en escalar en futuras actualizaciones.

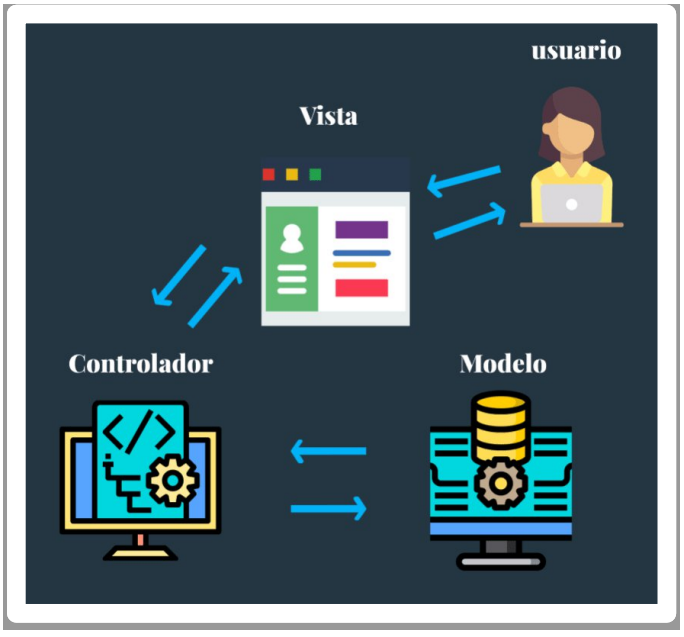


Figura n°1.

Si bien en el modelo usualmente se realiza con integración con base de datos relacionales o documentales, para el presente proyecto proponer dicha implementación sería interesante aunque queda fuera del alcance del mismo propuesto por la asignatura.

Diagrama de clase

Se procede a realizar un diagrama de clases para el lado del servidor y del cliente realizado en UML. También se dejan adjuntos al archivo comprimido para mayor claridad en su revisión.

Diagrama de clase del servidor

Se presenta a continuación el siguiente diagrama correspondiente al programa de servidor realizado con la herramienta en línea llamada Drawdio.

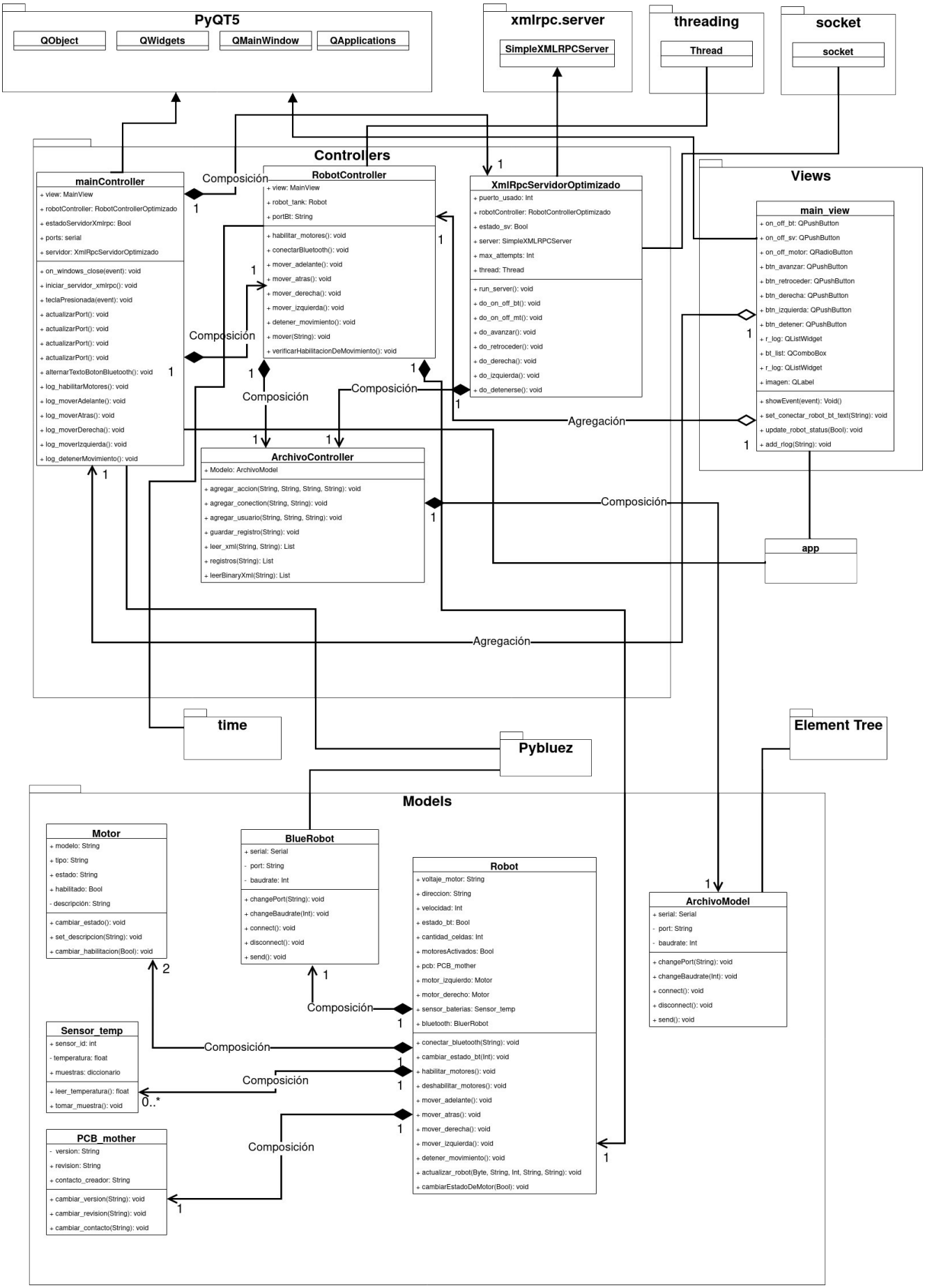

PyQT5
QObject
QWidgets
QMainWindow
QApplications
xmlrpc.server
SimpleXMLRPCServer
threading
Thread
socket
socket
Controllers
mainController
+ view: MainView
+ robotController: RobotControllerOptimizado
+ estadoServidorXmlrpc: Bool
+ ports: serial
+ servidor: XmlRpcServidorOptimizado
+ on_windows_close(event): void
+ iniciar_servidor_xmlrpc(): void
+ teclaPresionada(event): void
+ actualizarPort(): void
+ actualizarPort(): void
+ actualizarPort(): void
+ actualizarPort(): void
+ alternarTextoBotonBluetooth(): void
+ log_habilitarMotores(): void
+ log_moverAdelante(): void
+ log_moverAtras(): void
+ log_moverDerecha(): void
+ log_moverIzquierda(): void
+ log_detenerMovimiento(): void
RobotController
+ view: MainView
+ robot_tank: Robot
+ portBt: String
+ habilitar_motores(): void
+ conectarBluetooth(): void
+ mover_adelante(): void
+ mover_atras(): void
+ mover_derecha(): void
+ mover_izquierda(): void
+ detener_movimiento(): void
+ mover(String): void
+ verificarHabilitacionDeMovimiento(): void
XmlRpcServidorOptimizado
+ puerto_usado: Int
+ robotController: RobotControllerOptimizado
+ estado_sv: Bool
+ server: SimpleXMLRPCServer
+ max_attempts: Int
+ thread: Thread
+ run_server(): void
+ do_on_off_bt(): void
+ do_on_off_mt(): void
+ do_avanzar(): void
+ do_retroceder(): void
+ do_derecha(): void
+ do_izquierda(): void
+ do_detenerse(): void
Views
main_view
+ on_off_bt: QPushButton
+ on_off_sv: QPushButton
+ on_off_motor: QRadioButton
+ btn_avanzar: QPushButton
+ btn_retroceder: QPushButton
+ btn_derecha: QPushButton
+ btn_izquierda: QPushButton
+ btn_detener: QPushButton
+ r_log: QListWidget
+ bt_list: QComboBox
+ r_log: QListWidget
+ imagen: QLabel
+ showEvent(event): Void()
+ set_conectar_robot_bt_text(String): void
+ update_robot_status(Bool): void
+ add_rlog(String): void
ArchivoController
+ Modelo: ArchivoModel
+ agregar_accion(String, String, String, String): void
+ agregar_conection(String, String): void
+ agregar_usuario(String, String, String): void
+ guardar_registro(String): void
+ leer_xml(String, String): List
+ registros(String): List
+ leerBinaryXml(String): List
app
Composición
Agregación
time
Pybluez
Element Tree
Models
Motor
+ modelo: String
+ tipo: String
+ estado: String
+ habilitado: Bool
- descripción: String
+ cambiar_estado(): void
+ set_descripcion(String): void
+ cambiar_habilitacion(Bool): void
BlueRobot
+ serial: Serial
- port: String
- baudrate: Int
+ changePort(String): void
+ changeBaudrate(Int): void
+ connect(): void
+ disconnect(): void
+ send(): void
Robot
+ voltaje_motor: String
+ direccion: String
+ velocidad: Int
+ estado_bt: Bool
+ cantidad_celdas: Int
+ motoresActivados: Bool
+ pcb: PCB_mother
+ motor_izquierdo: Motor
+ motor_derecho: Motor
+ sensor_baterias: Sensor_temp
+ bluetooth: BluerRobot
+ conectar_bluetooth(String): void
+ cambiar_estado_bt(Int): void
+ habilitar_motores(): void
+ deshabilitar_motores(): void
+ mover_adelante(): void
+ mover_atras(): void
+ mover_derecha(): void
+ mover_izquierda(): void
+ detener_movimiento(): void
+ actualizar_robot(Byte, String, Int, String, String): void
+ cambiarEstadoDeMotor(Bool): void
ArchivoModel
+ serial: Serial
- port: String
- baudrate: Int
+ changePort(String): void
+ changeBaudrate(Int): void
+ connect(): void
+ disconnect(): void
+ send(): void
Sensor_temp
+ sensor_id: int
- temperatura: float
+ muestras: diccionario
+ leer_temperatura(): float
+ tomar_muestra(): void
PCB_mother
- version: String
+ revision: String
+ contacto_creador: String
+ cambiar_version(String): void
+ cambiar_revision(String): void
+ cambiar_contacto(String): void
Composición
Composición
Composición
Composición
Composición
Composición
Agregación
Composición
Agregación

Figura n°2.

Se puede observar cómo en el diagrama de clase vemos cómo se implementa el orden de modelo vista controlador con sus respectivas clases.

Diagrama de clase del cliente

Se presenta a continuación el siguiente diagrama correspondiente al programa del cliente realizado con la misma herramienta mencionada

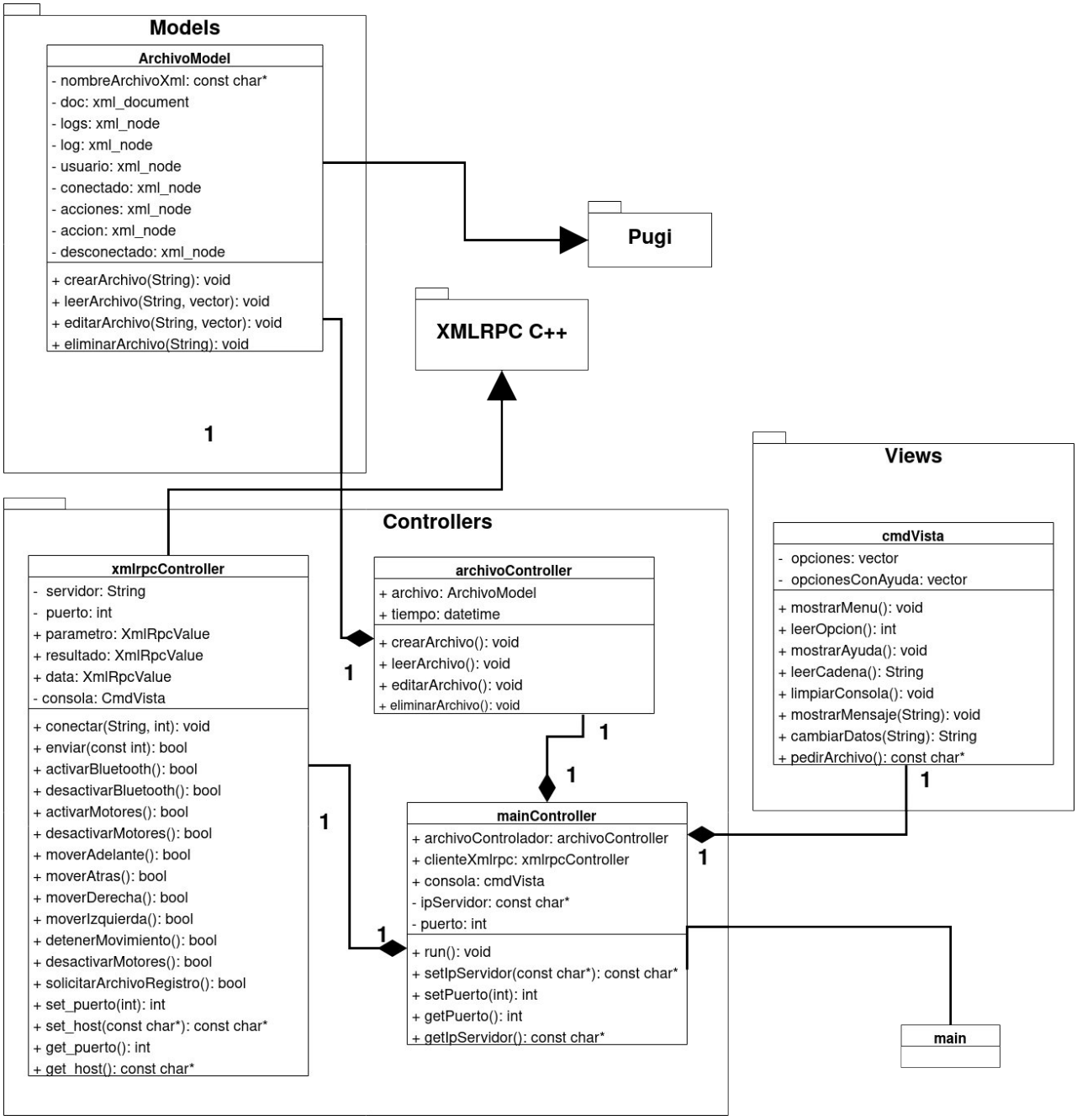


Figura n°3.

Al igual que el anterior se deja la posibilidad de acceso tanto al diagrama de clases del servidor cómo del cliente en el repositorio del proyecto.

Diagrama de secuencia

Se presenta también un diagrama de secuencia del lado del servidor que proporcione información del flujo del programa del lado del servidor con una leve aproximación temporal de los procesos y de manera ordenada.

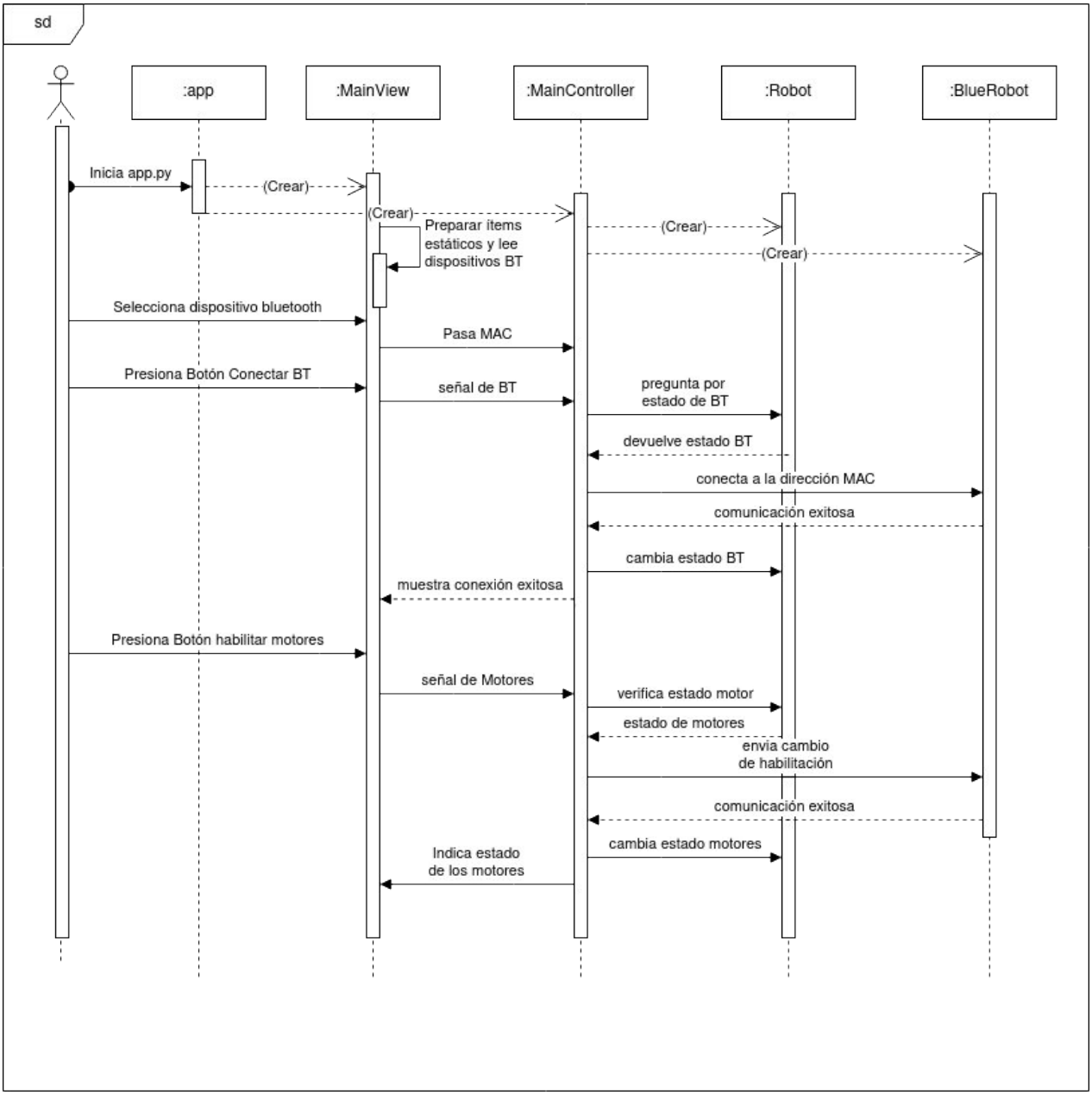


Figura n°4.

Firmware del robot

El robot móvil posee un programa instalado capaz de solamente recibir ciertos comandos, esta parte estará en la placa controladora en el microcontrolador ATmega328p. La misma se desarrolla bajo el lenguaje C++ modificado con el IDE de Arduino en la versión 1.8.12.

En esta parte se establece la comunicación con el módulo bluetooth HC-06 para enlazar una comunicación vía bluetooth con el servidor, este recibirá cómo parámetros los correspondientes al movimiento básico y la habilitación de los motores, los mismos se detallan a continuación:

- Al recibir el dato ‘A’ el robot habilita los motores.
- Al recibir el dato ‘B’ el robot deshabilita los motores.

A partir de ahora en adelante se consideran que los casos siguientes los motores se encuentran habilitados.

- Al recibir el dato ’8’ el robot comienza a avanzar hacia adelante.
- Al recibir el dato ’2’ el robot comienza a avanzar hacia atrás.
- Al recibir el dato ’6’ el robot comienza a girar hacia la derecha.
- Al recibir el dato ’4’ el robot comienza a girar hacia la izquierda.
- Al recibir el dato ’5’ el robot detiene el movimiento de los motores.

Servidor Python

Para la parte del servidor cómo anteriormente se dijo, fue realizado en Python en su versión 3.8.4, utilizando las librerías que proporciona el framework de interfaces QT pero funcionales dentro del entorno de Python ya que el framwork QT en si es en C++, para esta ocasión se usó PyQT5. Por otro lado, también se hizo uso de la librería xmlrpc para Python.

En el apartado del proyecto se encuentran las carpetas de cliente y servidor, donde en está última se presentan los siguiente elementos que integran al servidor

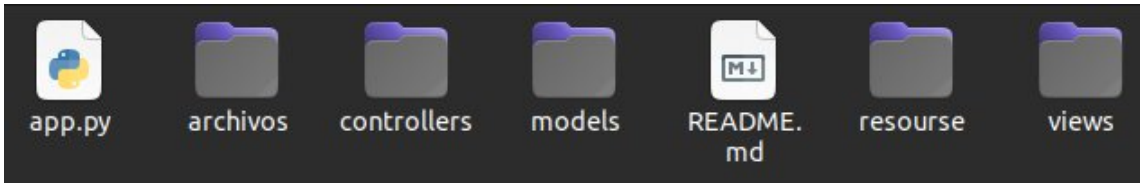


Figura n°5.

Estos elementos se ordenaron de esa manera para poder tener un proyecto de carácter más limpio y ordenado los cuales se describen a continuación:

- Archivo app.py: Es el archivo encargado de lanzar la aplicación del servidor con todos objetos y relaciones.
- Carpeta archivos: Es la carpeta contenedora del registro de actividades que se realizan en el robot, guardada en formato xml.
- Carpeta controllers: Carpeta en la cual se posicionan las clases que manejan la lógica e interacción de la aplicación.
- Carpeta models: Carpeta en la cual se encuentran las clases que hacen al manejo de registros ya sean estáticos o dinámicos.
- Archivo README.md: Es el archivo de descripción referente al proyecto.
- Carpeta resourse: Carpeta la cual almacena los archivos necesarios para poder levantar la interfaz gráfica del lado del servidor.

- Carpeta views: Carpeta que contiene la clase que permite la carga y pequeña interacción de la interfaz de usuario.

A continuación se deja un árbol de archivo con los contenidos de cada una de las carpetas:

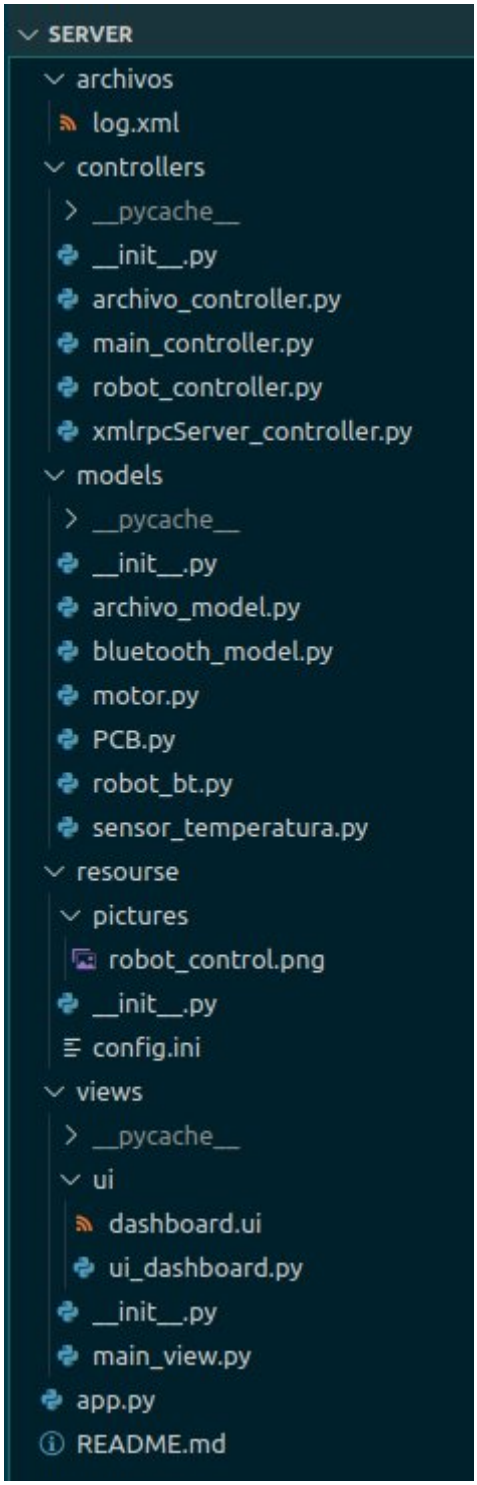


Figura n°6.

Para la clase ventana se toma en cuenta que el objeto que la misma instancia es la interfaz visual que el usuario en la parte del servidor con la cual podrá interactuar

directamente con el robot e iniciar el servicio XML-RPC, la misma está basada de PyQt5 y hereda funcionalidades de la clase QMainWindow para el funcionamiento del mismo.

En la siguiente figura (*figura N°7*) se puede observar la interfaz del servidor.

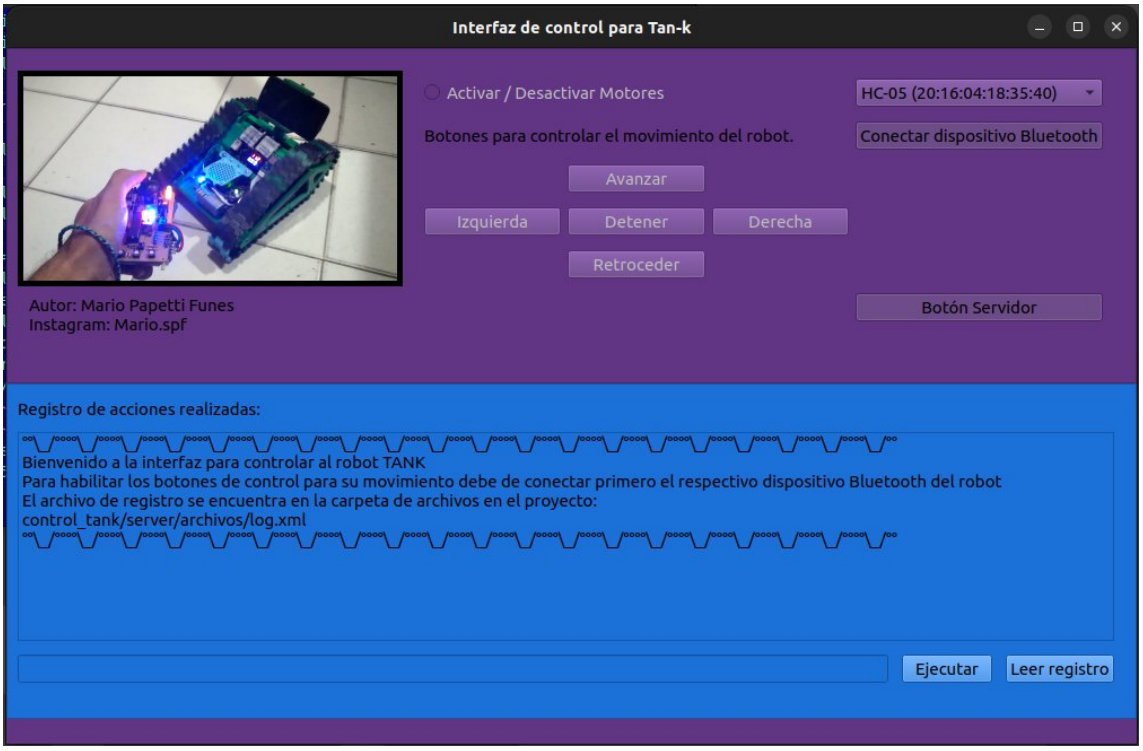


Figura n°7.

A continuación, se presenta en la *figura N°8* marcado los ítems que conforman la interfaz que interactuará con el usuario, el robot y los diversos clientes:

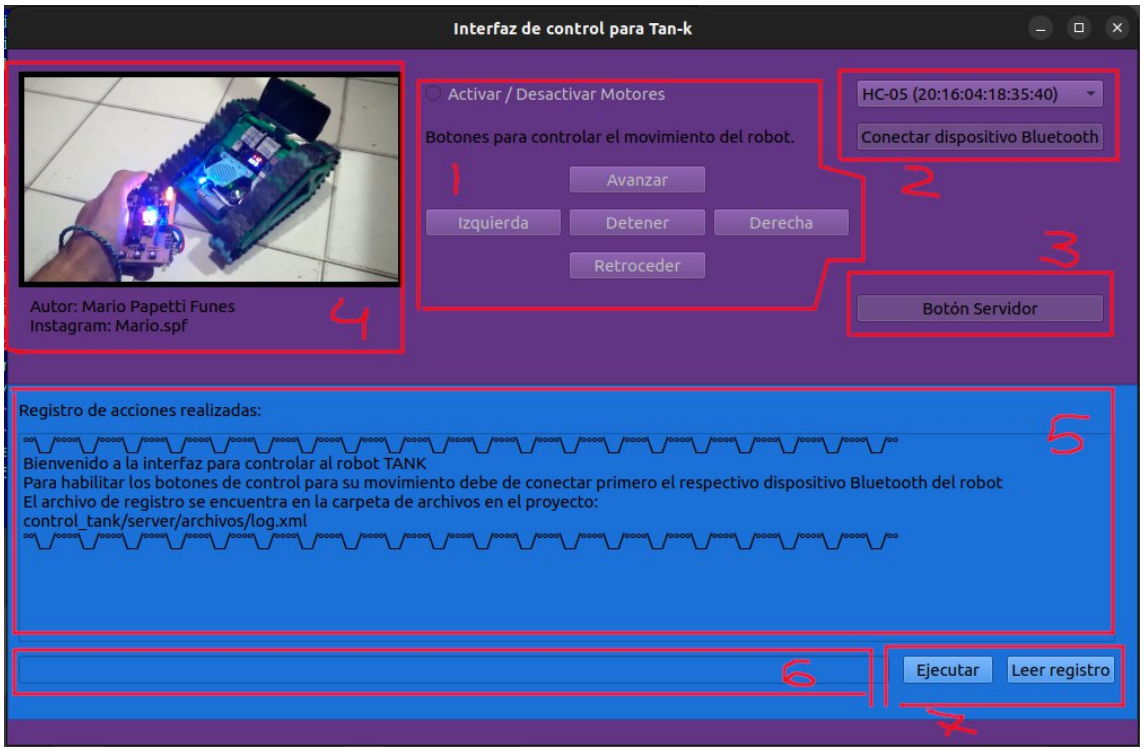


Figura n°8.

A continuación, se detallan los elementos que se visualizan en orden numérico:

1. Sección de botones para habilitar los motores y de control direccional del robot, tales son: Avanzar, Retroceder, Derecha, Izquierda y Detener, los mismos al igual que el botón de habilitación de motores estarán inhabilitados hasta tanto y en cuanto no se establezca la conexión con el robot, o se encuentre ejecutando una tarea desde un archivo xml y también si se activa la comunicación remota por XmlRpc.
2. Sección en donde se encuentra el desplegable para seleccionar el dispositivo bluetooth a conectar y su correspondiente botón para conectar al elemento seleccionado del desplegable.
3. Botón para activar y desactivar el servidor XmlRpc.
4. Sección informativa referente al autor del proyecto y una imagen representativa del robot que se desea controlar.
5. Sección que corresponde a la muestra de información del sistema, en esta parte estarán todas las acciones que se realizan y sus estados correspondientes.
6. Sección para ingresar la ruta con el archivo de registro en formato xml (incluyendo el mismo ".xml") para su tratamiento e interacción de opciones.
7. Sección de botones en donde se encuentran el botón para ejecutar la tarea del archivo xml previamente seleccionado y el botón para poder visualizar el contenido de dicho archivo en la misma app.

Por el lado del servidor aún queda pendiente la posibilidad de generar, enviar y ejecutar un archivo de instrucciones ya ordenadas y la generación y envío de un archivo de reporte.

Cliente en C++

Como anteriormente se mencionó el cliente fue desarrollado en C++ en conjunto con las librerías XmlRpc++ y varias otras más propias del sistema, tales como iostseam, vector, string, fstream.

La fuente de dónde se consiguió la librería XML-RPC se han obtenido desde el siguiente enlace: http://sourceforge.net/projects/xmlrpcpp/

En reglas generales la idea del cliente es poder fundamentalmente iniciar una conexión del tipo cliente con el servidor para poder enviar los comandos de movimiento al robot móvil, activar o desactivar tanto su conexión bluetooth cómo sus motores, además de la posibilidad de extraer un archivo de carácter de registro de actividad por parte del usuario remoto que estaría en la parte del cliente como de también enviarle un archivo de instrucciones que el robot deba de realizar sin intervención o control manual del usuario remoto ni local.

Acontinuación se muestra dentro de la capeta contenedora de la aplicación cliente los elementos que la integran para su correcto funcionamiento:

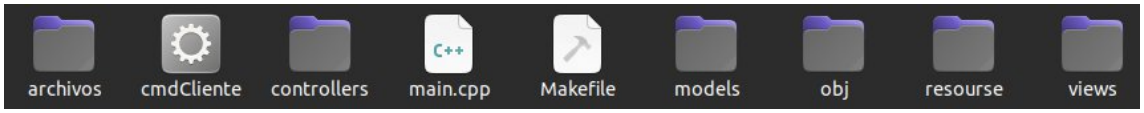
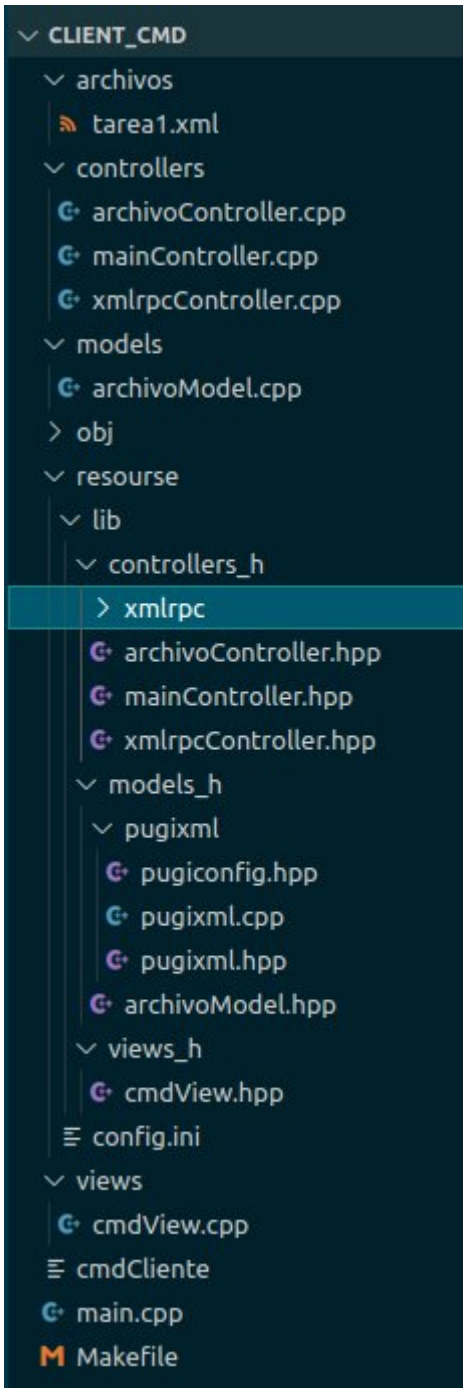


Figura n°9.

Estos elementos se ordenaron de esa manera para poder tener un proyecto de carácter más limpio y ordenado, al igual que del lado del servidor. Estos mismos se describen a continuación:

- Archivo main.cpp: Es el archivo encargado de lanzar la aplicación del cliente con todos objetos y relaciones que lo componen.
- Carpeta archivos: Es la carpeta contenedora del registro de actividades que se realizan en el robot, guardada en formato xml.
- Carpeta controllers: Carpeta en la cual se posicionan las clases que manejan la lógica e interacción de la aplicación.
- Carpeta models: Carpeta en la cual se encuentra la clase encargada de la gestión del archivo de registro.
- Archivo README.md: Es el archivo de descripción referente al proyecto.
- Carpeta resourse: Carpeta la cual almacena los archivos de cabecera de las clases y las librerías que se requieren para la correcto funcionamiento del cliente.
- Carpeta views: Carpeta que contiene la clase que permite la interacción con el usuario implementando una consola básica con su menú de opciones.



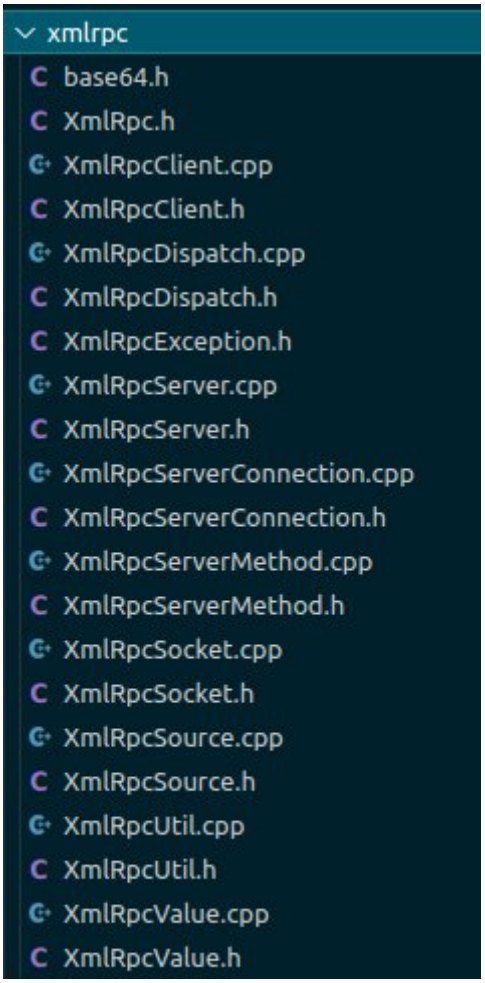


Figura n°10 y 11.

El mismo es ejecutado mediante un archivo ejecutable llamado cmdCliente que proporciona una interfaz muy parecida a la de una consola con el siguiente menú de opciones:

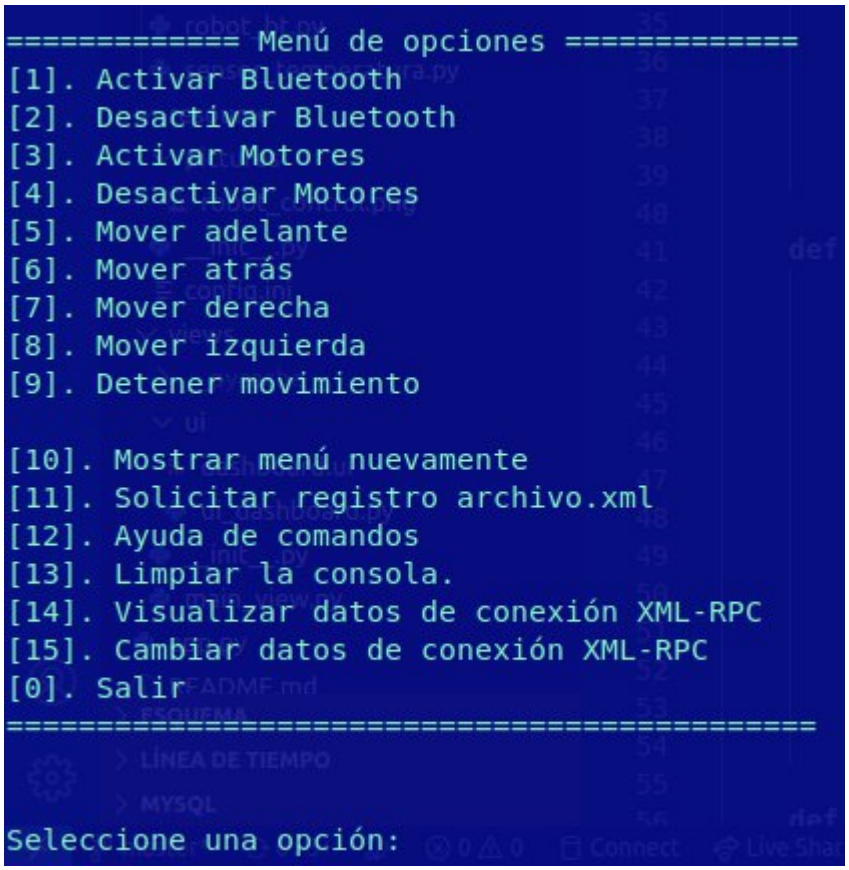


Figura n°12.

Para poder acceder a opción deseada se debe de colocar el número que se encuentra entre los corchetes, caso contrario, saldrá un mensaje de advertencia relacionado a la opción inesperada que se solicita. Para tener más información acerca de las acciones de cada opción del menú está la posibildiad de mostrar la ayuda correspondiente al menú que se muestra a continuación.

```
Seleccione una opción: 12
[1]. Activar Bluetooth
  └──> Se procede a activar la comunicación bluetooth al robot móvil del lado del servidor.
[2]. Desactivar Bluetooth
  └──> Se procede a desactivar la comunicación bluetooth al robot móvil del lado del servidor.
[3]. Activar Motores
  └──> Se procede a activar ambos motores del robot móvil del lado del servidor.
[4]. Desactivar Motores
  └──> Se procede a desactivar ambos motores del robot móvil del lado del servidor.
[5]. Mover adelante
  └──> Se procede a enviar el comando para que el robot móvil del lado del servidor comience a AVANZAR hasta nuevo comando.
[6]. Mover atrás
  └──> Se procede a enviar el comando para que el robot móvil del lado del servidor comience a RETROCEDER hasta nuevo comando.
[7]. Mover derecha
  └──> Se procede a enviar el comando para que el robot móvil del lado del servidor comience a GIRAR A LA DERECHA hasta nuevo comando.
[8]. Mover izquierda
  └──> Se procede a enviar el comando para que el robot móvil del lado del servidor comience a GIRAR A LA IZQUIERDA hasta nuevo comando.
[9]. Detener movimiento
  └──> Se procede a enviar el comando para que el robot móvil del lado del servidor DETENGA SU MOVIMIENTO hasta nuevo comando.

[10]. Mostrar menú nuevamente
   └──> Se imprimirá nuevamente el menú de opciones del lado del cliente.
[11]. Solicitar registro archivo.xml
   └──> Se procede a pedir del lado del servidor un archivo de formato XML con todas las acciones realizadas por los diferentes usuarios
 que han usado al robot móvil.
        El mismo será guardado en la carpeta archivos/ con el nombre de log.xml.
[12]. Ayuda de comandos
   └──> VISTA ACTUAL
[13]. Limpiar la consola
   └──> Se procede a limpiar la pantalla para una mejor claridad en la visualización de los datos.
[14]. Visualizar datos de conexión XML-RPC
   └──> Se procede a mostrar los datos actuales con los que se conecta el cliente XmlRpc (datos del servidor XmlRpc).
[15]. Cambiar datos de conexión XML-RPC
   └──> Se procede a cambiar los datos con los que se conecta el cliente XmlRpc al servidor XmlRpc.
        Para ello se requiere ingreso de la nueva dirección IP del host y del puerto de conexión referentes al servidor XmlRpc.
[0]. Salir
   └──> Se procede a cerrar la aplicación del cliente.

RECUERDE QUE SOLO SE ADMITEN LOS NÚMEROS ENTRE CORCHETES.
Seleccione una opción:
```

Figura n°13.

La estructura de los archivos de registro de actividades y de ejecución de tareas cuyo formato es xml poseen la siguiente semejanza con el ejemplo de un registro:

archivos > log.xml

You, ayer | 1 author (You)

```xml
<?xml version="1.0"?>
<logs>
    <log date="2023-02-22">
        <usuario>local</usuario>
        <conectado>11:20:30</conectado>
        <acciones>
            <accion type="bluetooth" action="on">activar bluetooth</accion>
            <accion type="motores" action="on">activar motores</accion>
            <accion type="movimiento" duration="3">detenerse</accion>
            <accion type="motores" action="off">desactivar motores</accion>
            <accion type="bluetooth" action="off">desactivar bluetooth</accion>
        </acciones>
        <desconectado>11:21:02</desconectado>
    </log>
    <log date="2023-02-23">
        <usuario>remote</usuario>
        <conectado>11:30:50</conectado>
        <acciones>
            <accion type="bluetooth" action="on">activar bluetooth</accion>
            <accion type="motores" action="on">activar motores</accion>
            <accion type="movimiento" duration="0.73">movimiento adelante</accion>
            <accion type="movimiento" duration="0.53">movimiento derecha</accion>
            <accion type="movimiento" duration="0.43">movimiento adelante</accion>
            <accion type="movimiento" duration="1">detenerse</accion>
            <accion type="motores" action="off">desactivar motores</accion>
            <accion type="bluetooth" action="off">desactivar bluetooth</accion>
        </acciones>
        <desconectado>11:50:02</desconectado>
    </log>
</logs>
```

Figura n°14.

CONCLUSIONES Y COMENTARIOS

Se encontraron muchas dificultades a la hora de abordar el proyecto pero las más críticas han sido resueltas satisfactoriamente dando una entrega de producto de poco valor de momento, pero, con la gran capacidad de ser ampliado al ser diseñado bajo el concepto de modelo-vista-controlador. Por otra parte, al proyecto le faltan algunas consignas que hacen del mismo no logre concretar de ciertas maneras ciertos casos de usos.

Otro comentario a tener en cuenta es que, si bien el apartado de Anexos se encuentra con algunos errores, no es de vital importancia en cuanto al proyecto en sí, ya que es para profundizar más en la problemática en sí.

No he logrado poder cerrar el servicio XmlRpc del lado del servidor lo cual genera un conflicto entre los puertos y se terminan abriendo varios servicios XmlRpc en varios puertos y el cliente se congela. Además ha existido problemas al intentar cambiar las credenciales de conexión del servidor con respecto al cliente debido a este problema.

Otro comentario que se debería de considerar es una opción para salir del estado cuando se intenta ejecutar un archivo de registro xml.

Realmente el proyecto tiene potencia para poder seguir creciendo y mejorando, la dificultad del mismo se tornó bastante difícil al ser solo encarado por un solo integrante en el proyecto ya que cuando se trabaja en grupo existe siempre la posibilidad de abordar un problema desde diversos puntos de vista y opiniones.

BIBLIOGRAFÍA

Durante el desarrollo del proyecto se han consultado diversas fuentes de información tales como:

- StackOverflow
- Documentación de la catedra Programación Orientada a Objetos – Fing – UNCuyo.
- GitHUB
- genbeta

Anexos

En los anexos se deja información más pertinente al proyecto del robot móvil en si que del enfoque de POO, es para poder darle un entendimiento más detallado del problema que se abordó.

Control RF1

Resumen

La idea principal era poder verificar la viabilidad mecánica del robot de una forma controlada por el usuario para luego poder llevarlo desde otra perspectiva automatizada, se

podría decir que este fue el primer paso dónde se pudieron ver las falencias mecánicas y corrección de las mismas en conjunto con el hardware de control que posee.

A continuación, se presenta tanto algunas fotos representativas del mismo cómo también se detallará el esquema del control.

PCB

La placa ha sido diseñada con el programa de software libre llamado KiCAD en el cual a continuación se presentan 3 representaciones graficas del control, las cuales son:

- Esquema electrónico *Figura A n°1.*
- Diseño del PCB *Figura A n°2.*
- Representación 3D *Figura A n°3.*

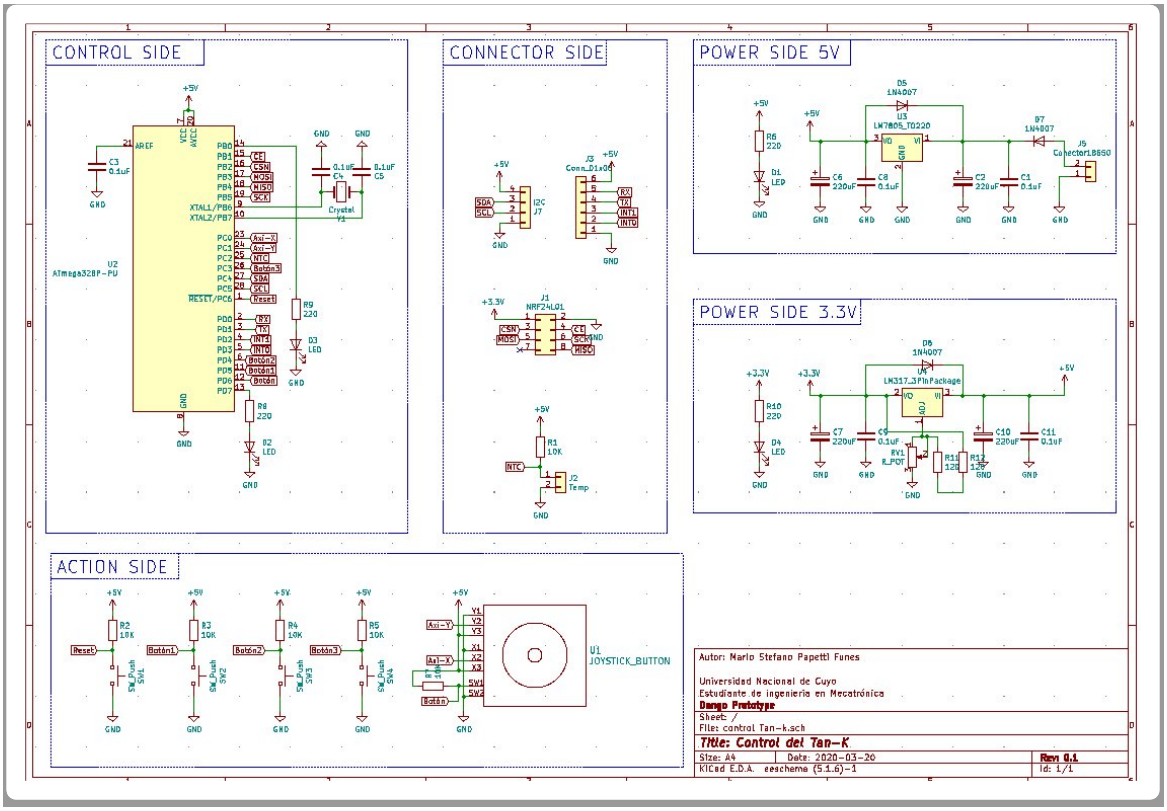


Figura A n°1.

A continuación, se presenta una imagen constructiva de la disposición de cada elemento representativo del esquema anteriormente presentado.

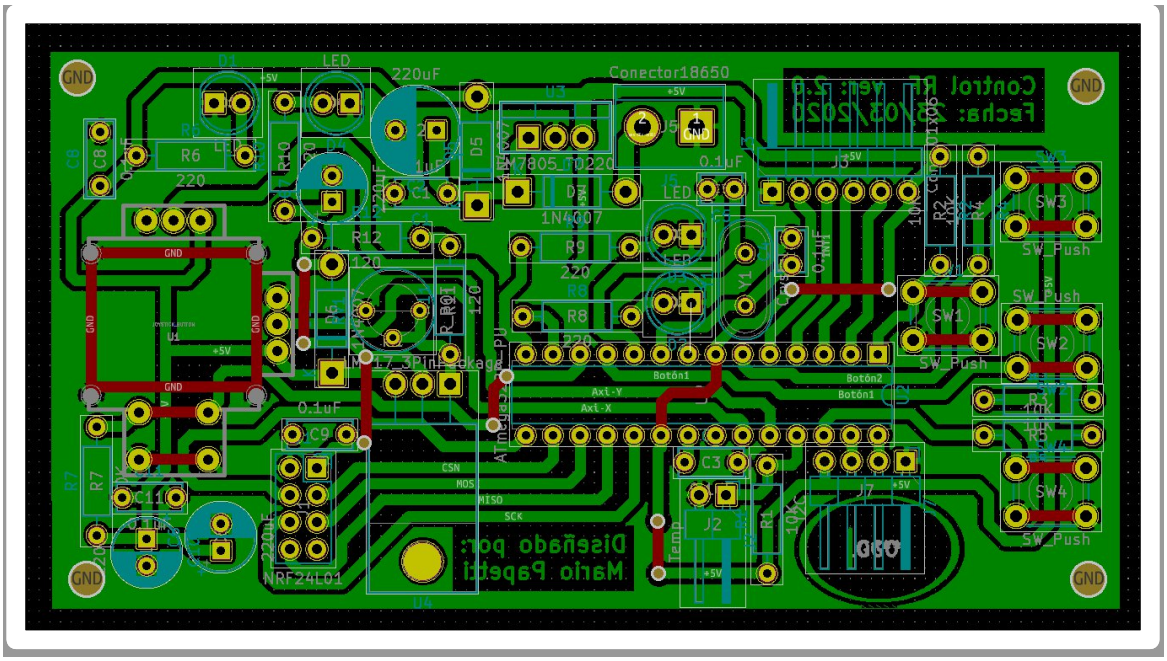


Figura A n°2.

En la siguiente imagen se presenta el modelo 3D en conjunto con el motor grafico de FreeCAD, otro software libre. En otra versión del informe posiblemente se agregue las modificaciones correspondientes en la asociación de los modelos 3D cómo por ejemplo en el analógico, aunque al estar descontinuada esta versión no se le harán cambios.

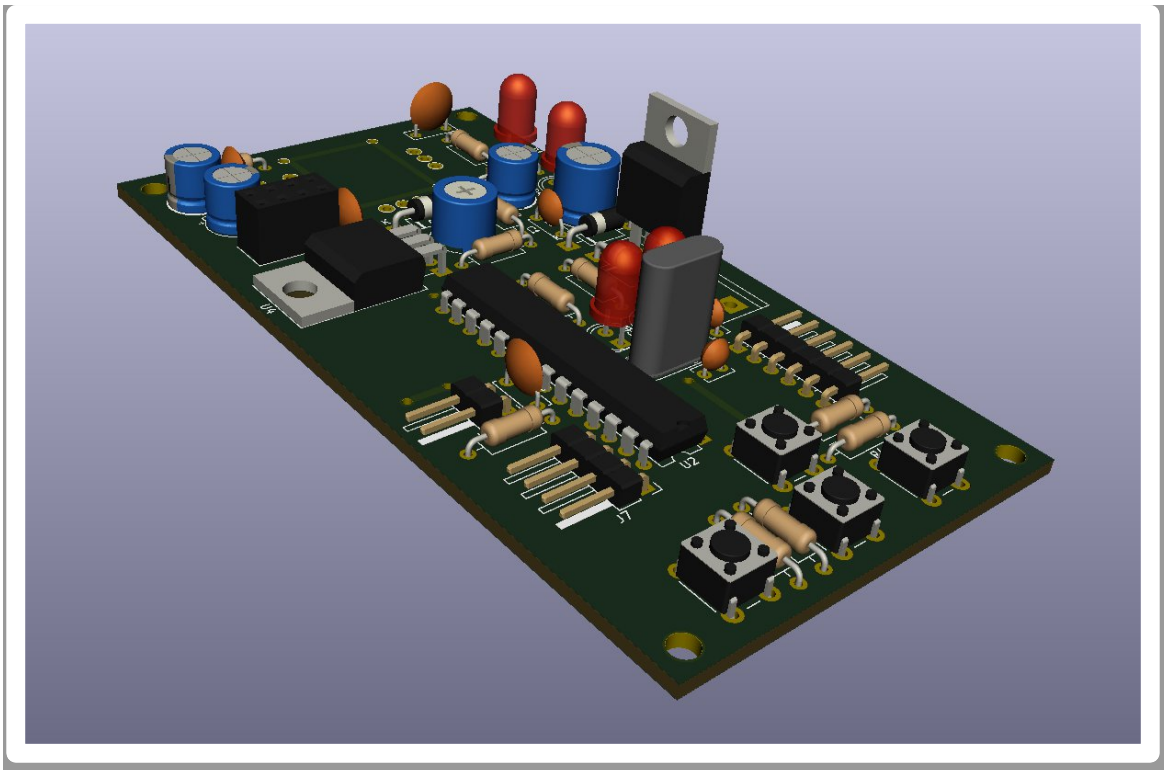


Figura A n°3.

Conclusión

Si bien la idea de aplicar un control remoto es interesante de varios puntos de vista, queda totalmente limitada al dispositivo físico y la comunicación con el mismo. Por lo que se opta por otro tipo de enlace, es en ese momento que entra importancia la solución propuesta en el presente informe.

ESTADO DEL ARTE

Idea general

El robot tanque puede desenvolverse en vario ámbitos cómo vigilancia, seguridad, reconocimientos perimetrales, aplicaciones de machine learning, robot interactivo de entretenimiento. El mismo logra llevar a cabo dichas tareas gracias a la vinculación de diversos dispositivos electrónicos, eléctricos, mecánicos y algoritmos.

Estado actual

En la instancia actual, el robot se controla mediante una aplicación y un módulo bluetooth, en la versión anterior funcionaba mediante un control de radiofrecuencia dónde solo tiene habilitado los movimientos en el plano (Ver sección de Anexos para mayor detalle de la solución por radiofrecuencia).

El robot está conformado por una estructura impresa en 3D de ABS, internamente funciona con un pack de celdas 18650 y su respectivo módulo de carga para 3 en serie (BMS 3S)

A continuación, se presenta una foto (*figura A n°4 y figura A n°5*) del robot siendo controlado por un control a radiofrecuencia, cómo ya se mencionó anteriormente se deja la presentación de la imagen cómo representativa del robot ya que actualmente ha tenido ligeros cambios físicos y respecto a su control, que actualmente funciona mediante Bluetooth con una aplicación realizada con APP Inventor.

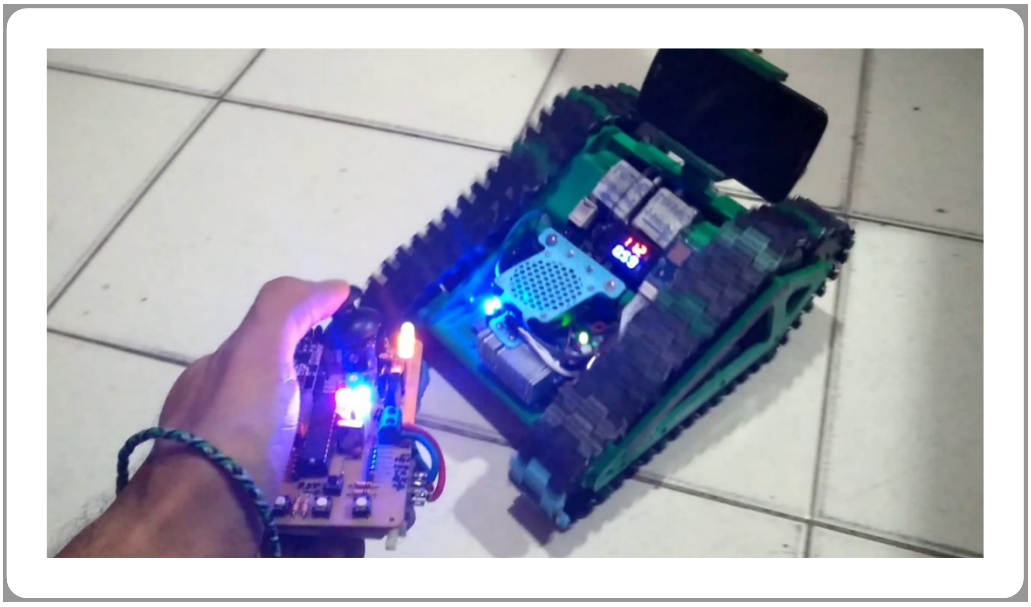


Figura A n°4.

Como anteriormente se mencionó, el robot es capaz de realizar movimientos dentro del plano, lo cual consiste en la capacidad de poder recorrer este mismo, sin pendientes pronunciadas mayores a 30° o pi/6.

En este último se aclara que puede desplazarse en dicho plano con 4 movimientos básicos, lo cuales son: avanzar, retroceder, girar sobre su mismo eje en sentido horario y en sentido antihorario.

Figura A n°5.

ASPECTO MECÁNICO

En la parte donde la mecánica predomina se hace inferencia tanto la transmisión que utiliza cómo la etapa electromecánica que requiere el mismo para sus acciones.

En la siguiente foto (*figura n°4*) se puede observar la estructura del robot que ha sido diseñada por el usuario Staindis sustraído desde la página Thingiverse. A continuación, se deja el link del creador del proyecto original y la imagen (*figura n°3*) representativa bajo la licencia que dicho proyecto se encuentra. https://www.thingiverse.com/thing:2414983

Figura A n°6.

En este aspecto se verá la estructura pura del robot móvil, cómo se muestra en la *figura n°4* en donde se observa el robot sin más que la estructura, sacando de lado el celular de la fotografía y demás sutiles componentes que ayudan a la fijación del mismo podemos decir que el tanque está íntegramente formado por piezas impresas en 3D en especial están hechas en material ABS.

Figura A n°7.

En cuanto a la dinámica vemos que el mismo es traccionado por dos motores paso a paso que han sido reciclados de alguna impresora en desuso, estos motores poseen en su eje un piñón que engrana a una pequeña caja reductora (ambos impresos en ABS) ya que el modelo estaba pensado para ser usado solamente motores brushless de aeromodelismo, los cuales poseen una elevada velocidad pese a su peso. La pequeña etapa de reducción conecta con un rodillo engranado el cual termina transmitiendo el movimiento del robot a las orugas, la cual está conformada por pequeños eslabones impresos en 3D y unidos con alambre.

UNIDAD ALIMENTACIÓN

La fuente de energía capaz de darle la potencia que requiere tanto para la etapa mecánica, cómo para poder sostener la unidad de control y las comunicaciones, son las celdas 18650 de Ion-Li.

Para el robot tanque se han implementado un “pack” de 6 celdas conformadas por 3 pares en serie de las mismas, ya que cada celda posee unos 4,2 v dando en total una diferencia de potencial total de 12,6 v.

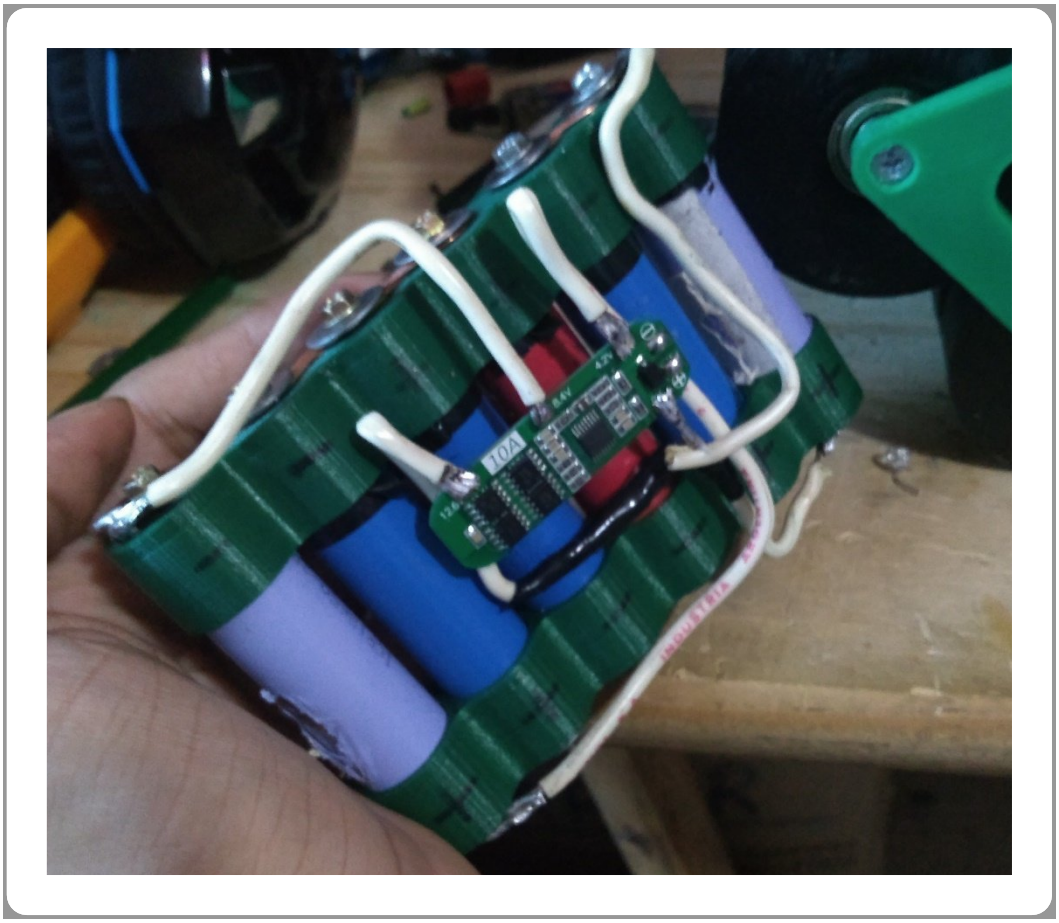


Figura A n°8.

En la foto representada en la *figura A n°8* se puede observar el módulo que anteriormente denominamos cómo “pack”, cabe rescatar que dichas celdas 18650 Ion-Li son recicladas de diversas baterías donde fallaban por una celda aislada o simplemente por la obsolescencia programada que poseen en las placas que cargan dichas celdas.

UNIDAD DE CONTROL

La unidad de control del sistema se ve conformada por una placa basada en el modelo de Arduino donde se utiliza cómo eje controlador al microcontrolador ATmega328p, dicha placa ha sido diseñada y fabricada por el presente autor del informe (*figura A n°9, n°10 y n°11*). En cada una se detallará tanto las conexiones de todos los periféricos que intervienen en el sistema cómo también aquellos aspectos que conciernen a características electrónicas o de control.

PCB ATmega328p

La plaqueta electrónica como bien indica su nombre se basó en el microcontrolador ATmega328p para su diseño bajo el software libre de diseño llamado KiCAD.

A continuación, se presenta el esquemático del mismo en la *figura A n°9*, el diseño propiamente dicho de la placa en la *figura A n°10* y el modelo 3D del mismo en la *figura A n°11*.

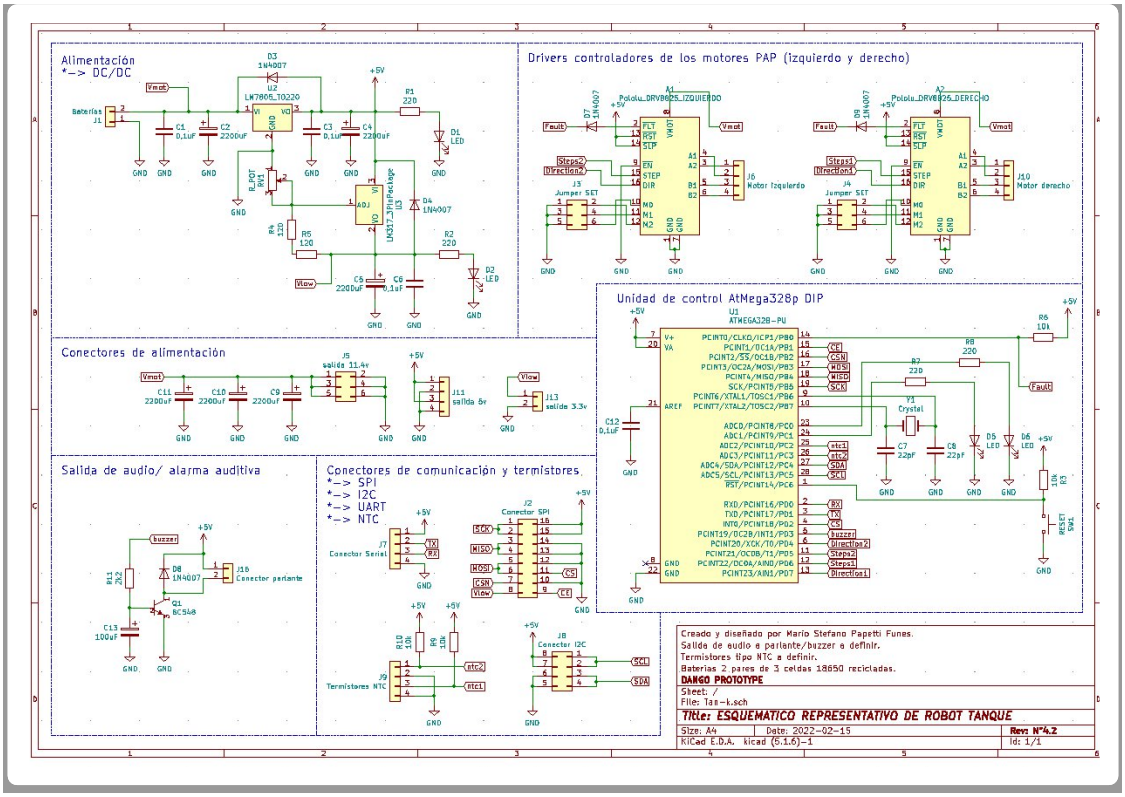


Figura A n°9.

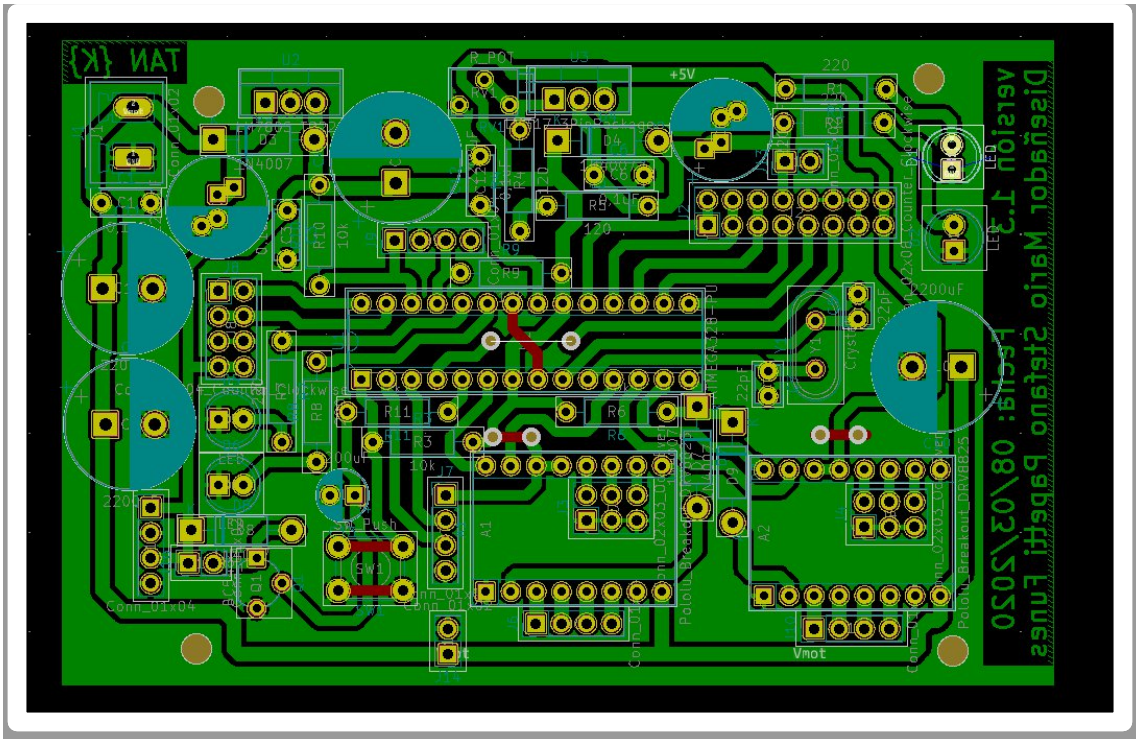


Figura A n°10.

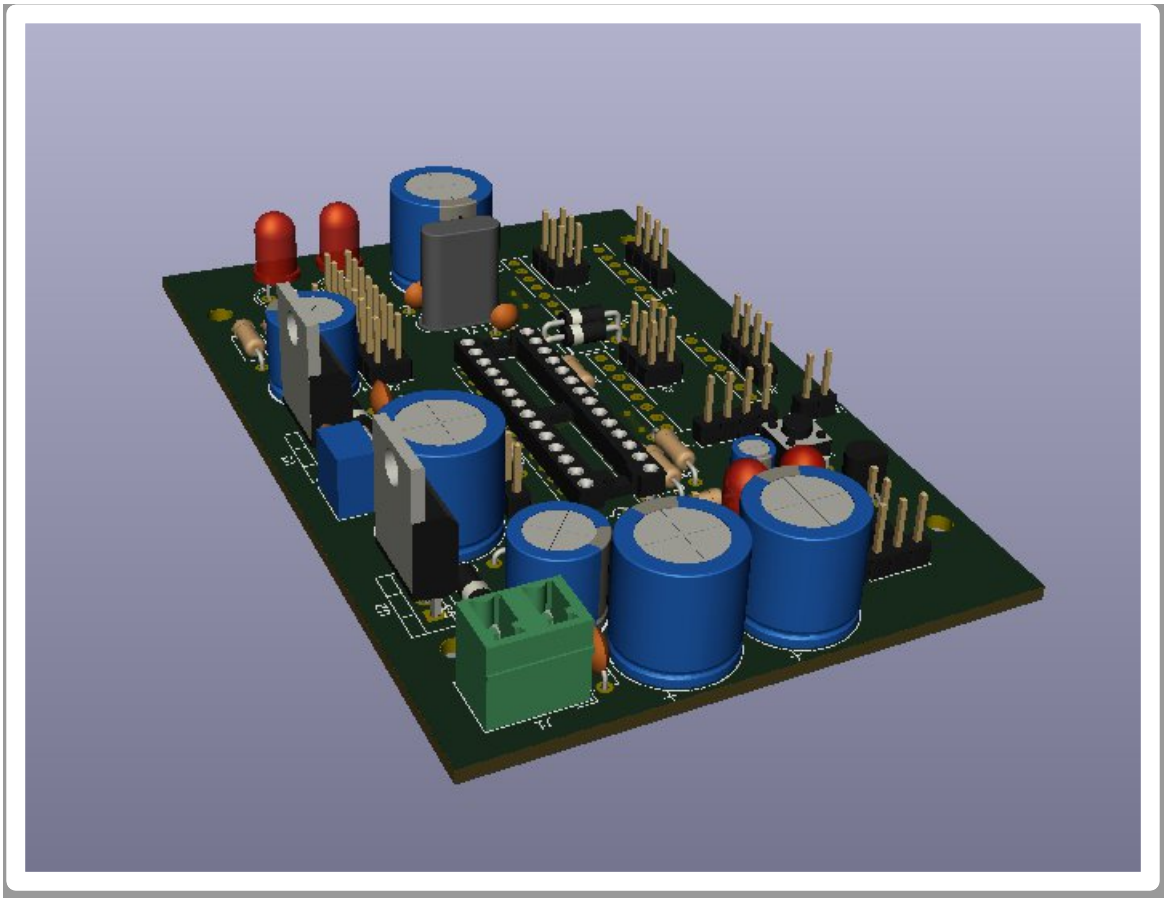


Figura A n°11.