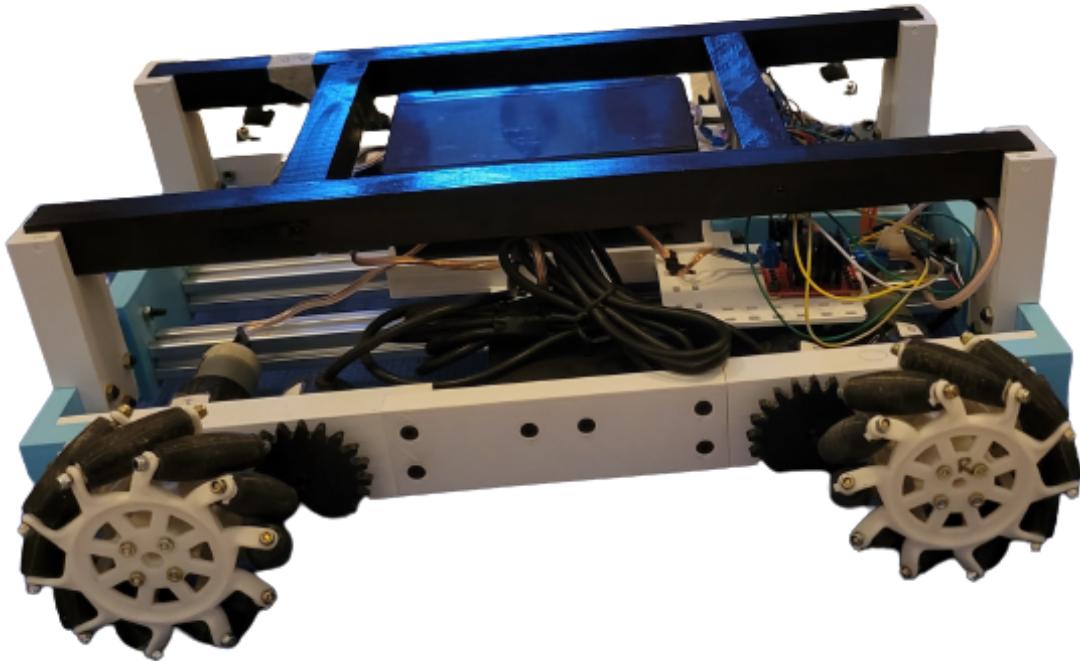


Universidad Austral

Vehículo omnidireccional no tripulado (VONT)

Fecha de inicio: 20/03/2022



Ivan Susnisky

Carrera: Ingeniería Industrial

Ramiro Molin

Orientación: Mecatrónica

Nicolas Leunda

Proyecto Final de Carrera

Índice

Índice	2
Objetivos	4
Principio de funcionamiento	5
Traslado lineal	5
Giros	6
Traslado horizontal	7
Componentes	8
Electrónica	8
Microcontrolador	8
Cable USB-Micro USB	9
Motor	9
Batería	13
Puente H	13
Encoder	14
Bulonería	15
Cableado	15
Diagrama de conexionado	16
Piezas de impresión 3D	17
Otras piezas	17
Montaje	18
Ruedas	18
VONT	18
Soporte de PC y carga	24
Programación	25
C++	25
Lógica de código:	26
Librerías utilizadas	26

Link al código	26
MIT	26
Modo de uso	27
Python	28
Guia instalacion de stella_vslam sus dependencias y Python Bindings	28
Librerías utilizadas	30
Comunicación entre PC y ESP-32	30
Utilización	31
Guía uso de Stella vSlam	31
Uso manual con la aplicación	40
Uso con coordenadas en aplicación	41
Conclusión	42
Limitaciones y mejoras realizables	42
Anexos	44
Anexo I	44
Anexo II	44
Anexo III	44
Anexo IV	44
Anexo V	45
Anexo VI	45

Objetivos

El objetivo del trabajo es diseñar y construir un Vehículo con movimiento **Omnidireccional No Tripulado** (de ahora en más, lo llamaremos VONT). Que sea omnidireccional quiere decir que puede moverse en todas las direcciones y sentidos. Para ello, se utilizarán un tipo de ruedas llamadas *Mecanum* (ver *Imagen 1*), patentadas por Bengt Erland Ilon en Estados Unidos en 1972¹, hoy esa patente ya está expirada.



Imagen 1. Ruedas Mecanum.

El objetivo base del proyecto es lograr manejar el vehículo a distancia con un celular mediante una aplicación específica para este propósito.

El objetivo final de este trabajo es lograr que, utilizando el procesamiento de imágenes en tiempo real, para que el VONT pueda trasladarse de manera autónoma solo recibiendo las coordenadas objetivo a las cuales se debe mover, por medio de la aplicación móvil previamente mencionada.

El alcance del proyecto es que el vehículo pueda transportar carga de una forma rápida. Como nuestro trabajo es un prototipo de montacargas se puso como objetivo que pueda trasladarse a

¹ Llamas, Luis (17/07/2019). *Robot con Mecanum Wheel controlado por Arduino*.

<https://www.luisllamas.es/robot-con-mecanum-wheel-controlado-por-arduino/#:~:text=Las%20ruedas%20Mecanum%20Wheel%20fueron,han%20empleado%20ruedas%20Mecanum%20Wheel>.

la misma velocidad a la que se mueve una real: 6 km/h u 1.67 m/s. En cuanto a la masa apilable sobre el vehículo se definió que sean 1.8 kg, el peso de 1 portátil para videojuegos aproximadamente.

Principio de funcionamiento

Las ruedas Mecanum constan de una serie de rodillos rotados 45 grados respecto de la rueda. Además, estos rodillos son curvados, de forma que la rueda mantiene el perfil circular (ver *Imagen 1*).

A diferencia de las convencionales, al rotar ejercen una fuerza a 45° grados respecto del eje de rotación de la misma (ver *Imagen 3*). Dicha fuerza se orienta perpendicularmente a los rodillos dentro de la rueda (ver *Imagen 3*). Gracias a esto al controlar cada una de ellas de forma independiente se combinan la rotaciones de las cuatro de distintas formas para generar vectores resultantes que muevan el vehículo en cualquier dirección.

A continuación se detallan las distintas combinaciones utilizadas en el proyecto.

Traslado lineal

Para trasladarse hacia adelante y hacia atrás se deben rotar las 4 ruedas en la misma dirección:

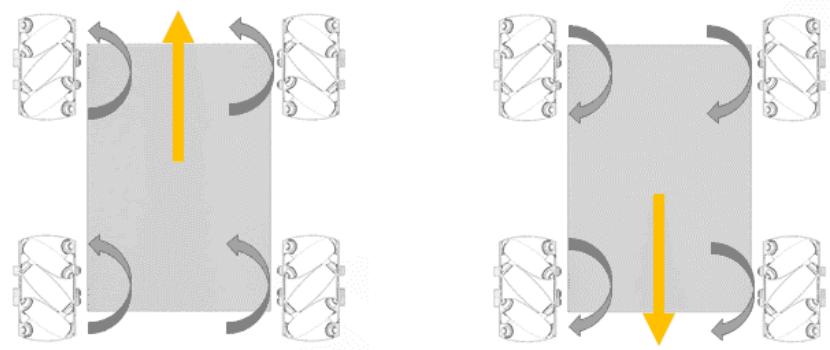


Imagen 2. Rotaciones y vectores resultantes para movimientos lineales.

Al realizar estos movimientos, se generan los siguientes vectores en cada una de las ruedas (vectores azules) y en amarillo se puede observar el vector resultante (ver *imagen 3*).

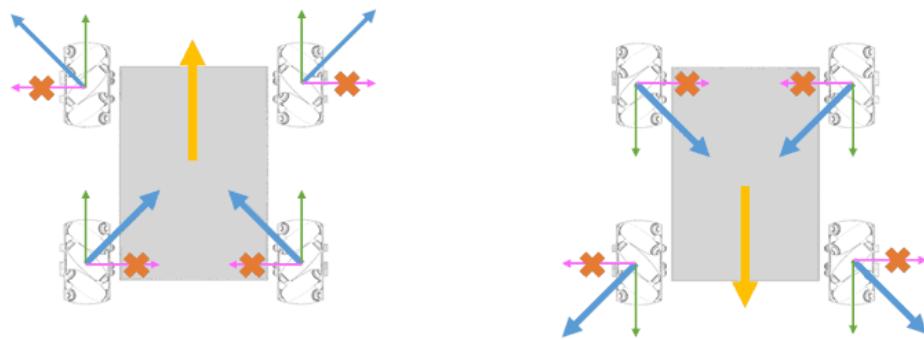


Imagen 3. Rotaciones y vectores para movimientos lineales.

Giros

Otro de los movimientos posibles es girar el vehículo sobre sí mismo. Se pueden observar los sentidos de giro y los vectores en las siguientes imágenes (ver *imagen 4 e imagen 5*).

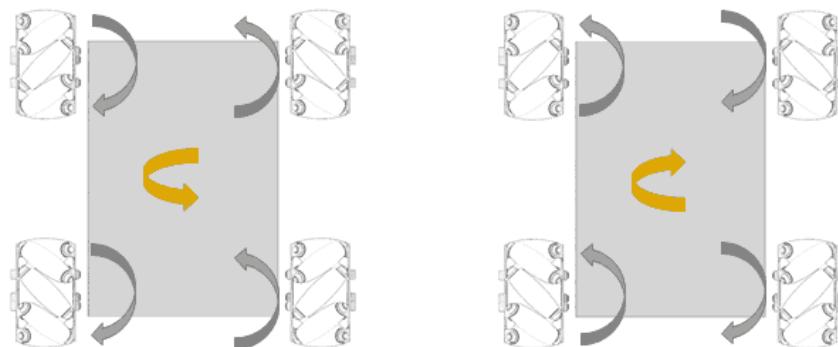


Imagen 4. Rotaciones y vectores para giros.

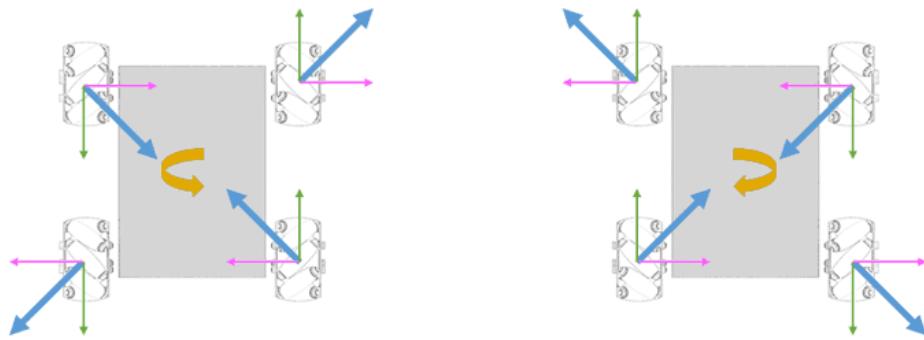


Imagen 5. Rotaciones y vectores para giros.

Traslado horizontal

Por último, el VONT es capaz de trasladarse lateralmente utilizando las combinaciones que se detallan a continuación (ver *imagen 6* e *imagen 7*).

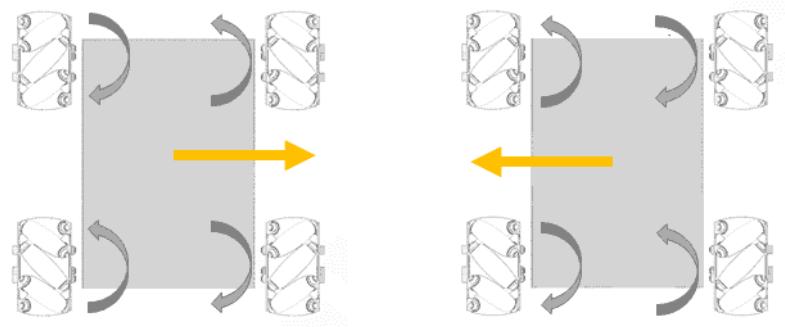


Imagen 6. Rotaciones y vectores para movimientos laterales.

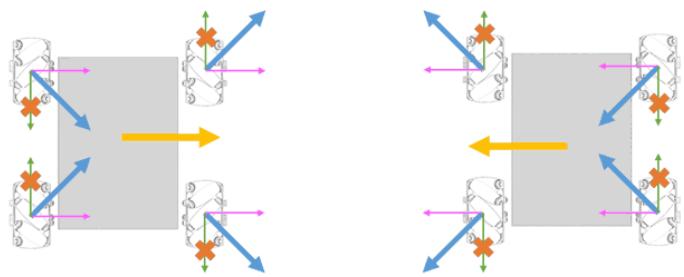


Imagen 7. Rotaciones y vectores para movimientos laterales.

Componentes

Electrónica

Microcontrolador

Un microcontrolador es un circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria². Elegimos el microcontrolador ESP-32 debido a que es una opción económica con bajo consumo de energía que permite trabajar con el framework de Arduino además de tener los módulos de WiFi y Bluetooth ya integrados para el control inalámbrico.



Imagen 8. Microcontrolador ESP-32.

² Microcontrolador. Wikipedia. <https://es.wikipedia.org/wiki/Microcontrolador>

Cable USB-Micro USB

Se utiliza un cable USB-Micro USB para conectar el ESP-32 a la computadora, tanto para alimentar el microcontrolador como para el flujo de información entre ambas.



Imagen 9. Cable USB-Micro USB.

Motor

Antes de elegir un motor específico, se analizó qué tipo de motor eléctrico se iba a utilizar. Se barajaron dos opciones distintas: motor de corriente continua y motor paso a paso. La programación del motor paso a paso resulta más sencilla, pero en un caso industrial real no son utilizados en vehículos de transporte de cargas ya que se especializan en la precisión. En cambio, los motores de corriente continua, en comparación con los paso a paso, mantienen un alto torque a altas velocidades. Debido a esto se optó por los motores de corriente continua para que el proyecto debido a la escalabilidad que le otorga.

Para seleccionar el motor, primero se realizó una estimación del torque necesario para luego seleccionar uno sobredimensionado.

A continuación se detallan los cálculos realizados para la elección del motor CC.

Características VONT	Peso	10	kg
	Fpeso	98	kgf
	Radio Ruedas	5	cm
	Velocidad	1	m/s
Coeficiente de fricción de rodadura	C	0,05	-
Masa del vehículo y carga útil	mt	10	kg
Aceleración gravitacional	g	9,8	m/s^2
Fuerza de fricción	$Fr = C * mt * g$	4,9	N
Torque del motor	Tr*it	0,2722222222	N*M
Ratio de reducción	it		-
Eficiencia de la transmision de energía	nu	0,9	-
Radio de la rueda	r	0,05	M
Fuerza motriz	Fd	4,9	N
Rueda =	Circunferencia	0,3141592654	m
Características para un motor	RPM	190,986	rev/min
	w (rad/s)	20	
	Torque (rueda) = $(Fd + Fr) * r$	0,49	N*m
	Torque motor	49	N*cm
	Torque motor = Peso * r * 0,05	2,5	kgf*cm
	Potencia = Torque (rueda) * w	9,8	Watts (N*m/s)
	Torque por motor (4)	0,625	kgf*cm
	Sobredimensionado	1,25	kgf*cm

RPM	Calculadas a partir de la velocidad objetivo del VONT			
m	1,000	----->	1,000	s
m	0,314	----->	0,314	s
s	0,314	----->	1,000	rev
s	60,000	----->	190,986	rev

Con estos cálculos se seleccionó el motor-reductor marca Ignis de 12V con una relación 36:1 y 177 rpm. el cual otorga una velocidad cercana a la deseada y un torque de 2.09 kgf.cm. Esta opción era la que más se aproximaba al objetivo dentro del catálogo disponible.

MOTORREDUCTOR																																																																																																					
 <p>MR 08 D - [] - [] - []</p>	FAMILIA VERSIÓN	TENSIÓN	POTENCIA	VELOCIDAD	OPCIONALES																																																																																																
	LT		Ø 24 Ø 16 Ø 34		Ø 8 ⁰ _{-0,02} Ø 4 ⁰ _{-0,05}																																																																																																
GENERAL	Esc. de Referencia 1:2																																																																																																				
Juego Libre (Backlash)	Menor a2°		Esfuerzo Radial Máximo	96,8 Kgf																																																																																																	
Temperatura de operación	Ta + 50°C		Esfuerzo Axial Máximo	40 Kgf																																																																																																	
Cupla de arranque / Bloqueo	Cupla Nominal * 4		Momento Torsor Máximo	176 Kgf.cm																																																																																																	
Velocidad Vacío (aprox.)	Velocidad Nominal + 15%		Momento Flexor Máximo	2,0 Kgf.cm																																																																																																	
		Origen		Argentina / China (Motor)																																																																																																	
<table border="1"> <thead> <tr> <th>Modelo</th> <th>MR08D-012004</th> <th>MR08D-024022</th> <th>MR08D-024022-H</th> </tr> </thead> <tbody> <tr> <td>Opcional</td><td colspan="3">- L(eje liso) - P(eje plano) - H(híbrido)</td></tr> <tr> <td>Adicional</td><td colspan="3">- BR08</td></tr> <tr> <td>Material del Reductor</td><td colspan="2">Plástico</td><td>Plas - Metal</td></tr> <tr> <td>Servicio</td><td colspan="3">Normal</td></tr> <tr> <td>Potencia [Watt]</td><td>4,84</td><td colspan="2">22,7</td><td colspan="3"></td></tr> <tr> <td>Tensión nominal [V]</td><td>12 Vcc</td><td colspan="2">24Vcc</td><td colspan="3"></td></tr> <tr> <td>Io . Inom . Is [A]</td><td>0,15 . 0,65 . 5,60</td><td colspan="2" rowspan="2">0,21 . 1,27 . 7,0</td><td colspan="3"></td></tr> <tr> <td>Ruido máx. [dB] (Adicional única etapa 15%)</td><td colspan="3">95</td><td colspan="3"></td></tr> <tr> <td>RPM Nom . RPM Vacío (motor)</td><td>6140 . 7600</td><td colspan="2">7810 . 3100</td><td colspan="3"></td></tr> <tr> <td>Peso . Adicional por etapa [Kg]</td><td>0,235 . 0,005</td><td>0,350 . 0,005</td><td>0,370 . 0,005</td><td colspan="3"></td></tr> <tr> <td>Largo[LT] . Adicional por etapa . Diámetro Motor [DM]</td><td>106 . 5,5 . 27,5</td><td>127 . 5,5 . 37</td><td>132,5 . 5,5 . 37</td><td colspan="3"></td></tr> <tr> <th>Etapas</th><th>Relación</th><th>Engranajes</th><th>Velocidad [RPM]</th><th>Cupla [Kgf.cm]</th><th>Cupla [Kgf.cm]</th><th>Cupla [Kgf.cm]</th></tr> <tr> <td rowspan="3">2</td><td>16:1</td><td>44</td><td>400</td><td>0,92*</td><td>5,08*</td><td>--</td></tr> <tr> <td>24:1</td><td>64</td><td>266</td><td>1,39*</td><td>7,65*</td><td>--</td></tr> <tr> <td>36:1</td><td>96</td><td>177</td><td>2,09*</td><td>11,40*</td><td>--</td></tr> </tbody> </table>	Modelo						MR08D-012004	MR08D-024022	MR08D-024022-H	Opcional	- L(eje liso) - P(eje plano) - H(híbrido)			Adicional	- BR08			Material del Reductor	Plástico		Plas - Metal	Servicio	Normal			Potencia [Watt]	4,84	22,7					Tensión nominal [V]	12 Vcc	24Vcc					Io . Inom . Is [A]	0,15 . 0,65 . 5,60	0,21 . 1,27 . 7,0					Ruido máx. [dB] (Adicional única etapa 15%)	95						RPM Nom . RPM Vacío (motor)	6140 . 7600	7810 . 3100					Peso . Adicional por etapa [Kg]	0,235 . 0,005	0,350 . 0,005	0,370 . 0,005				Largo[LT] . Adicional por etapa . Diámetro Motor [DM]	106 . 5,5 . 27,5	127 . 5,5 . 37	132,5 . 5,5 . 37				Etapas	Relación	Engranajes	Velocidad [RPM]	Cupla [Kgf.cm]	Cupla [Kgf.cm]	Cupla [Kgf.cm]	2	16:1	44	400	0,92*	5,08*	--	24:1	64	266	1,39*	7,65*	--	36:1	96	177	2,09*	11,40*	--	
Modelo	MR08D-012004	MR08D-024022	MR08D-024022-H																																																																																																		
Opcional	- L(eje liso) - P(eje plano) - H(híbrido)																																																																																																				
Adicional	- BR08																																																																																																				
Material del Reductor	Plástico		Plas - Metal																																																																																																		
Servicio	Normal																																																																																																				
Potencia [Watt]	4,84	22,7																																																																																																			
Tensión nominal [V]	12 Vcc	24Vcc																																																																																																			
Io . Inom . Is [A]	0,15 . 0,65 . 5,60	0,21 . 1,27 . 7,0																																																																																																			
Ruido máx. [dB] (Adicional única etapa 15%)	95																																																																																																				
RPM Nom . RPM Vacío (motor)	6140 . 7600	7810 . 3100																																																																																																			
Peso . Adicional por etapa [Kg]	0,235 . 0,005	0,350 . 0,005	0,370 . 0,005																																																																																																		
Largo[LT] . Adicional por etapa . Diámetro Motor [DM]	106 . 5,5 . 27,5	127 . 5,5 . 37	132,5 . 5,5 . 37																																																																																																		
Etapas	Relación	Engranajes	Velocidad [RPM]	Cupla [Kgf.cm]	Cupla [Kgf.cm]	Cupla [Kgf.cm]																																																																																															
2	16:1	44	400	0,92*	5,08*	--																																																																																															
	24:1	64	266	1,39*	7,65*	--																																																																																															
	36:1	96	177	2,09*	11,40*	--																																																																																															

Imagen 10. Especificaciones del motor seleccionado.

Batería

Se utiliza una batería de 12V y 7.5Ah. Cualquier batería de 12V cumple con el cometido, en este caso la empleada fue la Kitzuma MG1275.



Imagen 11. Bateria Kitzuma MG1275.

Puente H

El puente H es un circuito integrado que se utiliza para controlar motores con un microcontrolador. Se seleccionó el puente H L298N. Cada uno de ellos permite controlar dos motores de corriente continua con voltaje de 24V y corriente 2A, como máximos. Como el motor consume una corriente nominal de 0.63A y el voltaje utilizado es de 12V, este driver cumple con las especificaciones requeridas.



Imagen 12. Puente H L298N.

Encoder

Un encoder es un sensor que se utiliza para convertir el movimiento de la posición angular de un eje a una señal eléctrica. Se utilizaron los encoders LPD3806-600BM-G5-24C que tienen una resolución de 600 pasos por vuelta. Este encoder se puede alimentar tanto con 12V como con 5V, por lo cual facilita el conexionado eléctrico de los elementos dando versatilidad en el mismo.



Imagen 13. Encoder LPD3806-600BM-G5-24C.

En el [Anexo I](#) se pueden encontrar las cantidades necesarias para cada uno de los componentes electrónicos.

Bulonería

Para no sufrir inconvenientes por tipos de roscas se decidió utilizar la rosca métrica con cabeza allen en todo el vehículo. Además, este tipo de rosca da mucha versatilidad debido a la gran variedad de largos y diámetros disponibles.

En el [Anexo I](#) se puede encontrar el tipo y las cantidades necesarias para reproducir el VONT.

Cableado

Todas las conexiones realizadas entre el ESP-32 y la alimentación de los encoders utilizan los cables de Arduino (ver *Imagen 14*).



Imagen 14. Cables Arduino.

Para la conexión de la batería con los drivers se utiliza un cable de 1.25mm² de grosor, suficiente para soportar 12V y 0.7A sin riesgos.

Diagrama de conexionado

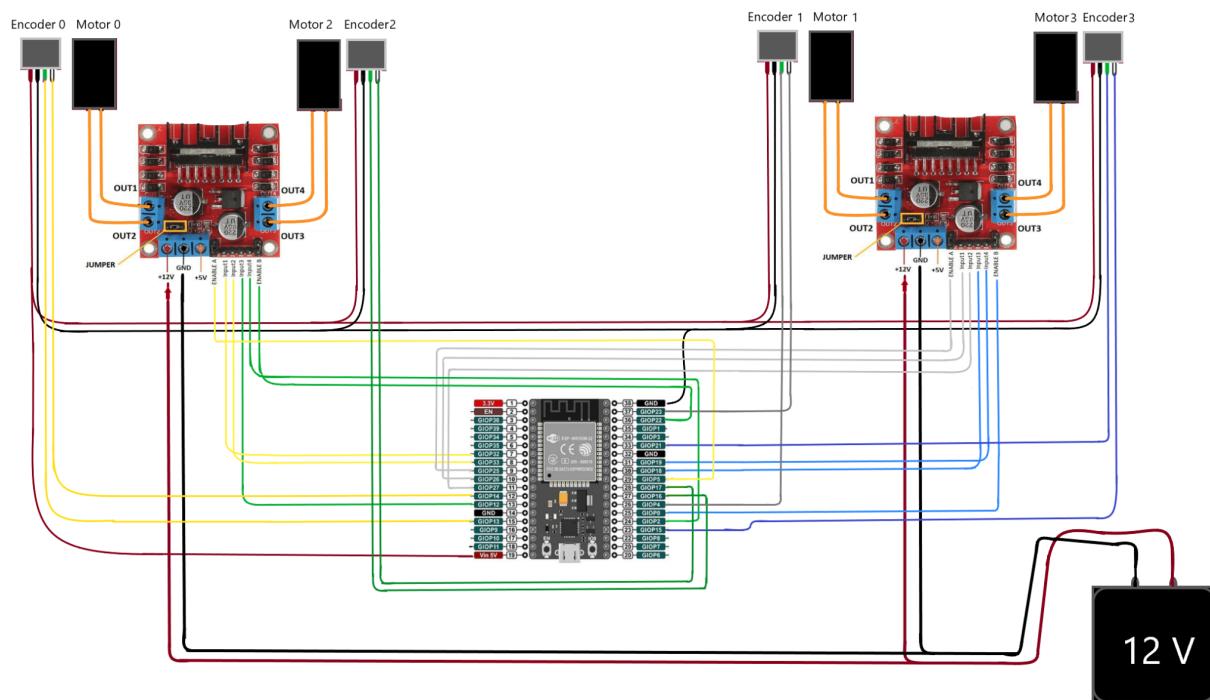


Imagen 15. Diagrama de conexionado eléctrico.

En la imagen 15 se puede ver el conexionado eléctrico ([imagen con mayor resolución en Anexo III](#)). Los cables finos representan los de Arduino y los más gruesos representan los de 1.25mm². El código de colores del diagrama, donde los motores se representan en un tono más claro que los encoders, es el siguiente:

- Conjunto 0: amarillo.
- Conjunto 1: gris.
- Conjunto 2: verde.
- Conjunto 3: azul.
- Voltaje positivo: rojo.
- GND: negro.

Los motores se conectan al puente H con cables de 1.25mm². Estos cables están representados con naranja debido a que no tienen un polo positivo y negativo constante, dado a que varían según el sentido de giro deseado.

Piezas de impresión 3D

La gran mayoría del VONT está diseñado e impreso en 3D. El listado de las piezas se puede encontrar en el [Anexo I](#) así como también el material correspondiente para cada una de ellas y recomendaciones para el perfil de impresión.

Para la impresión se usaron dos materiales diferentes: PLA y Flex. El segundo se utiliza únicamente en los rodillos de las ruedas para que haya una mayor fricción con la superficie, mejorando la tracción del vehículo.

Todas los archivos CAD de estas piezas se pueden encontrar en el [Anexo II](#) del documento, mientras que en el [Anexo I](#) se pueden ver la cantidad, material e información relevante para llevar a cabo la impresión de las mismas.

Otras piezas

Dos unidades de perfil 2020 de 408 mm para la estructura del vehículo, haciendo que pueda soportar mayores cargas.



Imagen 16. Perfil 2020.

Por último, se utilizó madera para la estructura que sostiene la PC y la carga (ver *imagen 26*).

Montaje

Se recomienda en primer lugar tener todos los materiales disponibles antes de comenzar el montaje.

Ruedas

Para el armado de las ruedas, comenzar cortando 40 ejes de 60mm varilla métrica M3. Luego atornillar con cuatro tornillos M3x40 con el siguiente orden: engranaje motor, pieza interna (izquierda o derecha), centro, pieza externa (izquierda o derecha) y cuatro tuercas M3 cada rueda. Luego montar los rodillos colocando la parte impresa en PLA en el interior de la parte impresa en Flex. Por último, pasar los ejes en los agujeros de cada rueda, colocando el rodillo, y poniendo a la varilla tuercas autofrenantes en cada extremo.

Para una demostración del montaje, consultar el siguiente [LINK](#).

Se puede encontrar más información respecto a la rueda utilizada en el siguiente [LINK](#).

VONT

Atornillar los laterales con los agujeros hacia arriba a los soportes de encoder y motor prestando atención a que el agujero para el motor (aquel con cuatro tornillos en cruz) se encuentre en la parte más externa utilizando dos tornillos M4x20 y dos tuercas M4 por lado. Luego colocar tuercas cuatro M4 en las ranuras inferiores de la viga (ver *imagen 17*) y atornillarla a los laterales usando cuatro tornillos M4x35 (ver *imagen 18*).

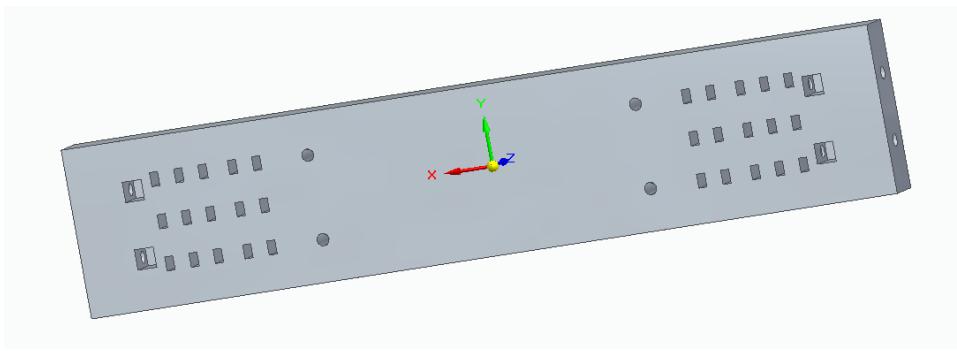


Imagen 17. Viga del VONT.

Una vez realizado el paso anterior, obtendremos la siguiente estructura.

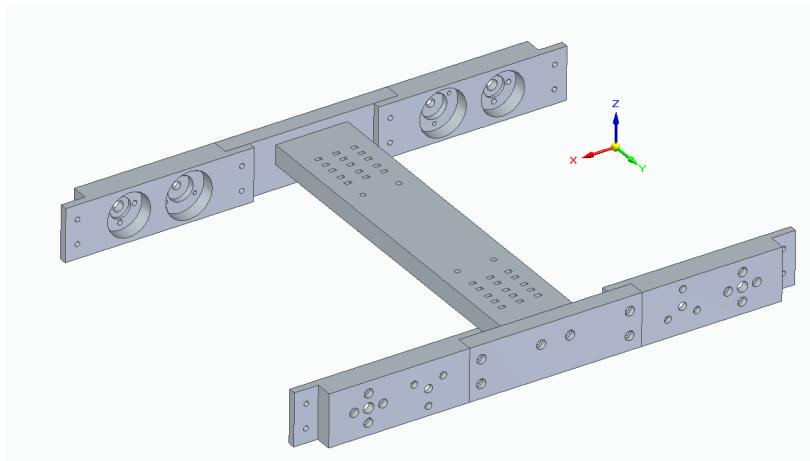


Imagen 18. Viga, laterales y soportes de encoder y motor.

Continuar colocando los paragolpes izquierdo y derecho al soporte de encoder y motor utilizando dos tornillos M4x20 y tuercas M4 para cada uno, así como también a la unión de los paragolpes con los agujeros más grandes en la parte inferior con un tornillo M4x20 y una tuerca M4 para cada uno de ellos (ver *Imagen 19*).

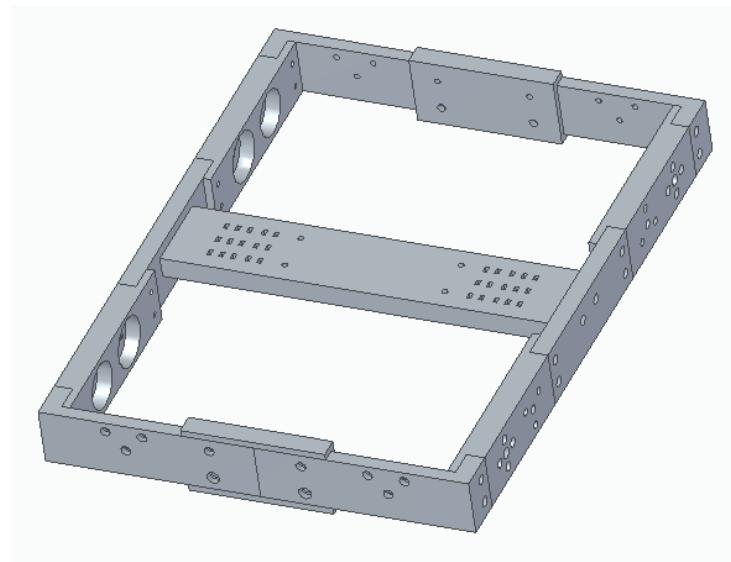


Imagen 19. Marco y viga del VONT.

Luego, cortar los perfiles 2020 a un largo de 408mm y roscar ambos extremos con un macho de roscado de métrica M6. A continuación, atornillarlos a la unión de los paragolpes con tornillos M6x25 para finalizar con la estructura del VONT (ver *imagen 20*).

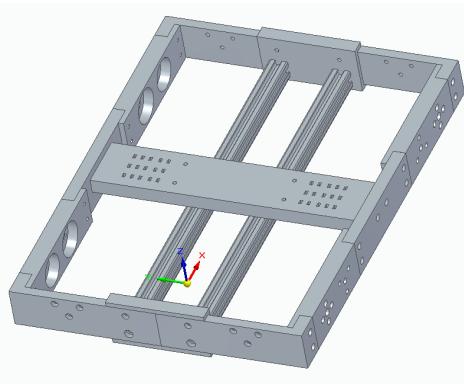


Imagen 20. Estructura del VONT.

Luego colocar los cuatro motores y encoders en cada uno de sus soportes usando cuatro tornillos M4x10 para los motores y tres tornillos M3x10 para los encoders. A continuación colocar en cada eje los engranajes y ruedas correspondientes colocando una tuerca M3 en cada uno de los orificios y tornillos M3x10 para que no haga juego con el eje (ver *Imagen 21*).

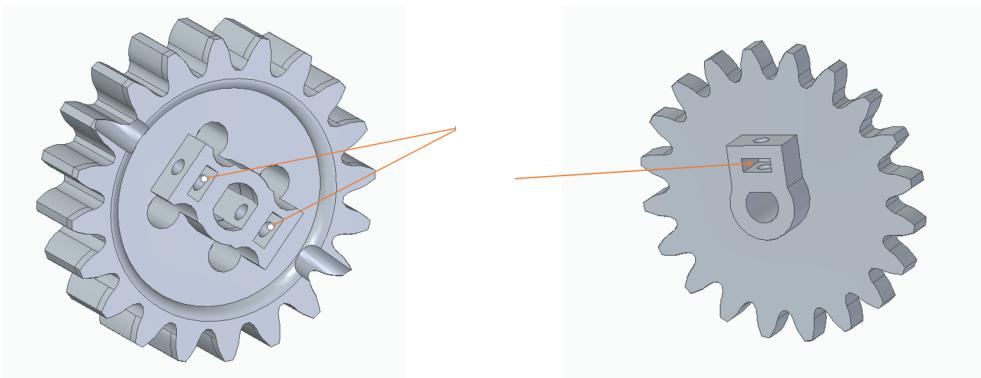


Imagen 21. Orificios para tuercas M3.

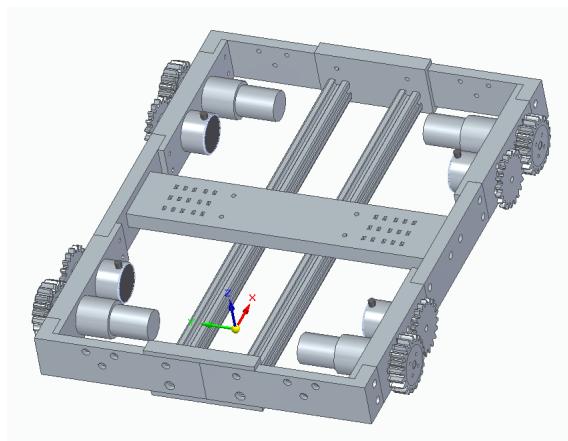


Imagen 22. VONT con motores, encoders y engranajes.

Continuar colocando el soporte de la batería atornillado a la viga con cuatro tornillos M3x25 y tuercas M3. También colocar los cuatro soportes PC atornillados con tres tornillos M4x20 y tres tuercas M4 para cada uno, a cada uno de los paragolpes (ver *imagen 23*).

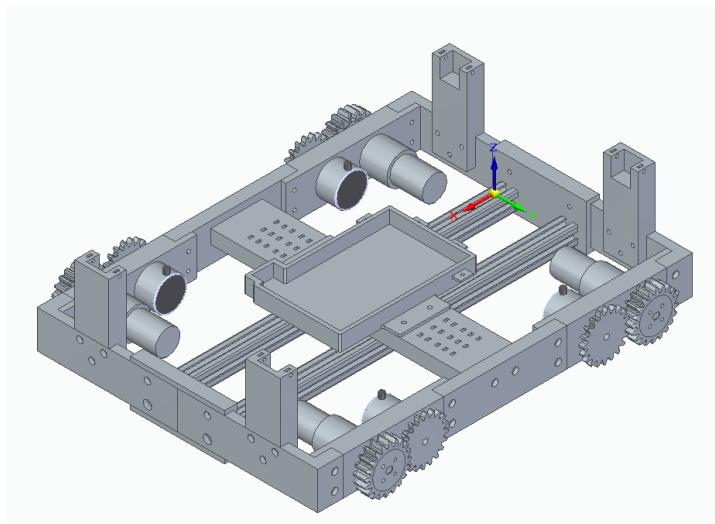


Imagen 23. VONT con soportes de batería y PC.

Para este caso se utilizó una protoboard para realizar el conexionado de el ESP-32 hacia los distintos drivers y encoders. Las conexiones deben hacerse según el diagrama eléctrico provisto. Los cables se conectan en un extremo a la protoboard y luego se pasan por el medio del soporte de la misma como se muestra en la *imagen 24*. Luego debe pasarse el otro extremo por cada uno de los agujeros asignados, para así mantener un control visual organizado y seguimiento de los cables. Se debe respetar el código de colores provisto en el diagrama eléctrico para evitar confusiones.

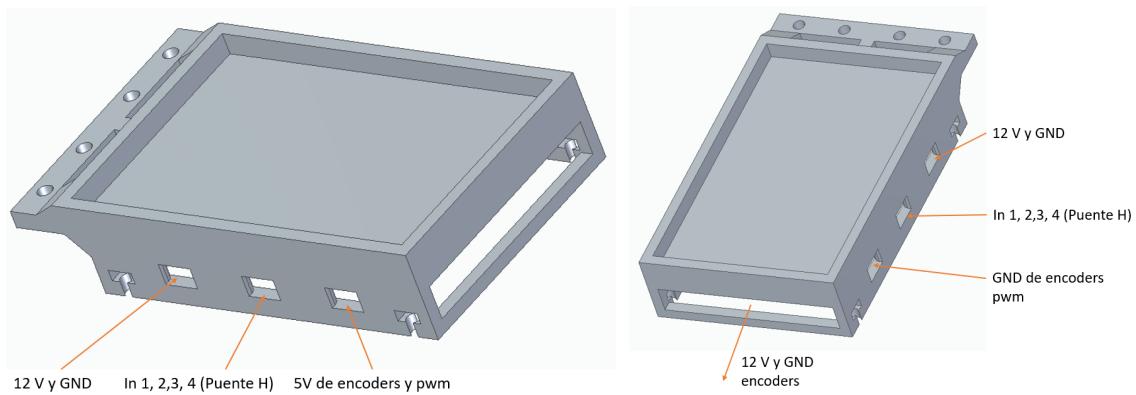


Imagen 24. Soporte de protoboard.

Nota: Conectar a lado izquierdo los cables de los motores 0 y 2. Conectar a lado derecho los cables de los motores 1 y 3.

Una vez realizado el paso anterior se coloca la tapa del soporte de la protoboard con cuatro tornillos M3x10 y cuatro tuercas M3.

A continuación, atornillar los soportes de electrónica al soporte de batería utilizando dos tornillos M3x10 y dos tuercas M3 donde indican las flechas azules (ver *Imagen 25*) y cuatro tornillos M3x12 y cuatro tuercas M3 donde indican las flechas naranjas (ver *Imagen 25*).

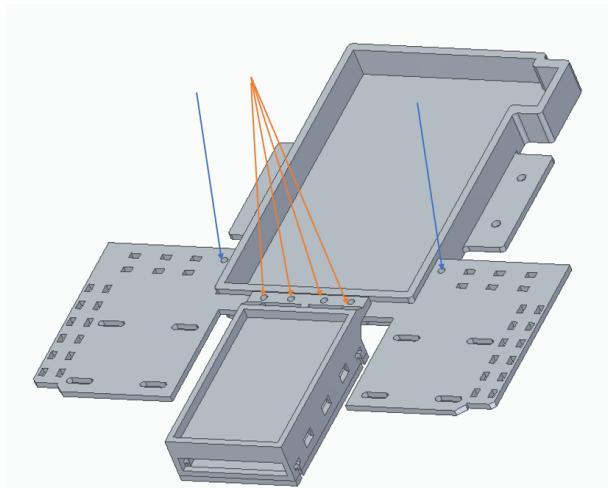


Imagen 25. Soportes de electrónica, de protoboard y de batería.

Una vez con la protoboard con el cableado en la estructura (la protoboard se coloca a presión) se procede a colocar el ESP-32 en la misma, y a colocar los puentes H en los soportes (se utiliza un tornillo M3x10 y 2 tuercas M3 para cada agujero del puente H para fijarlo a los oblongos), para luego hacer el conexionado con el extremo del cable aún no conectado. Luego se conecta cada motor a su pinOUT correspondiente y por último se realiza la conexión de 12V y GND que luego será conectado a la batería a la batería.

Nota: Muchas de las piezas llevan rendijas incorporadas para la colocación de precintos en caso de considerarlo necesario.

Soporte de PC y carga

A partir de maderas de perfil cuadrado de 22 mm, se realiza la estructura que soportara la PC y la carga. La misma calza a presión en los soportes y tiene las medidas que se pueden observar en la siguiente imagen (ver *Imagen 26*).

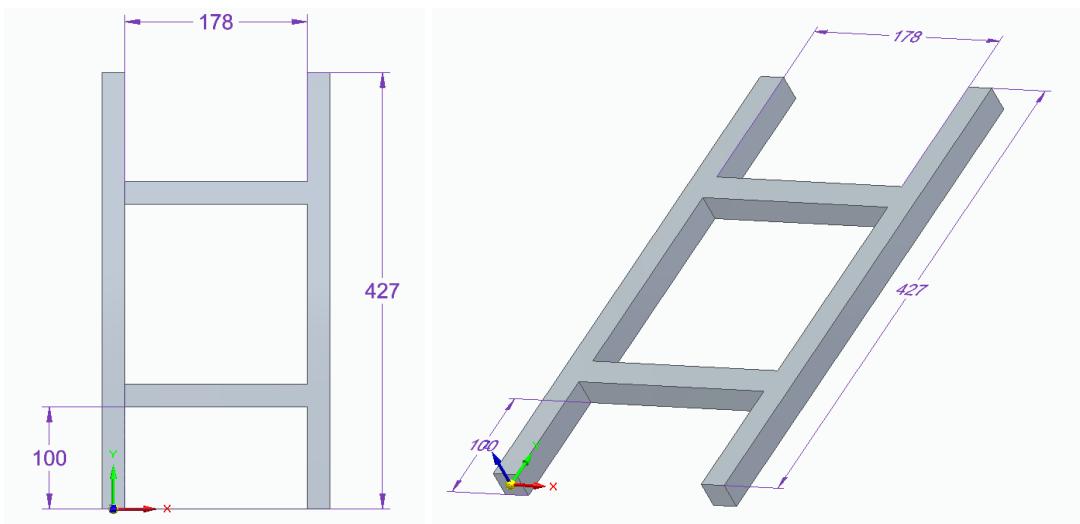


Imagen 26. Estructura que soporta la PC y la carga.

Programación

C++

Se codificó en Arduino y cargando en el ESP-32 lo necesario para el funcionamiento de los actuadores del VONT, es decir los cuatro motores, así como también para el uso de los encoders. Este código activa o bloquea las diferentes ruedas para lograr los movimientos deseados (explicados en la sección [Principio de funcionamiento](#)). Se comunica por puerto serie con la computadora, para recibir la posición donde se encuentra; y para recibir las coordenadas objetivo mediante WiFi, con la aplicación desarrollada (explicada a continuación). También cuenta con la posibilidad de publicar por MQTT³ la posición del VONT para poder hacer un

³ "Protocolo de red ligero, de publicación y suscripción, de máquina a máquina para cola de mensajes/servicio de cola de mensajes. Está diseñado para conexiones con ubicaciones remotas que tienen dispositivos con restricciones de recursos o ancho de banda de red limitado."
MQTT. Wikipedia. <https://en.wikipedia.org/wiki/MQTT>

seguimiento desde cualquier dispositivo con conexión a internet (función deshabilitada en el código para reducir el tiempo de respuesta).

Lógica de código:

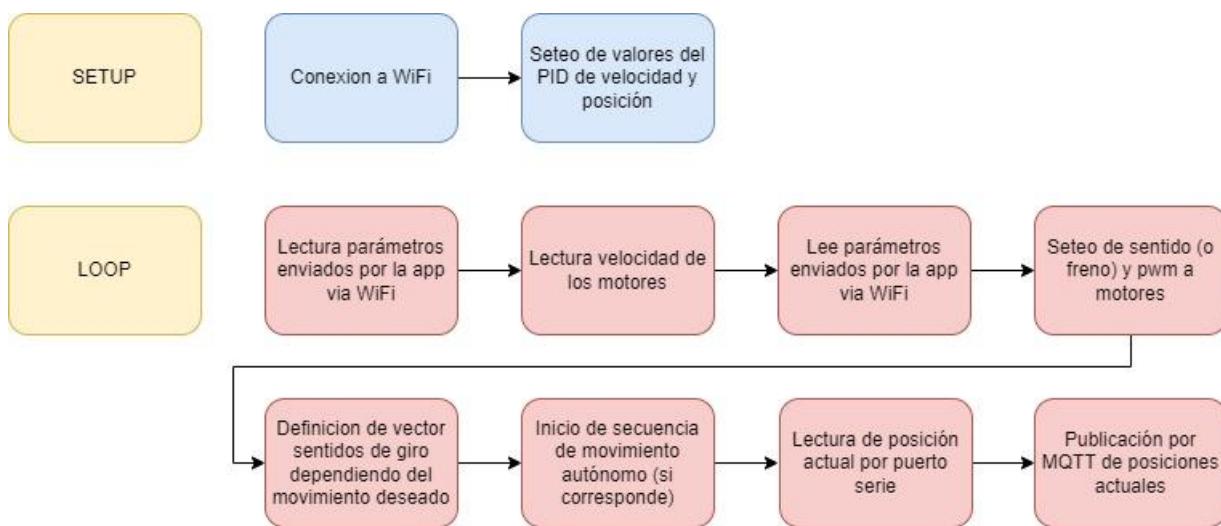


Imagen 27. Esquema de flujo del código cargado en el ESP-32.

Librerías utilizadas

- knollwary/PubSubClient@^2.8
- WiFi.h

Link al código

El código para descargar se puede encontrar en el [Anexo IV](#).

MIT

La aplicación utilizada para el control del VONT se realizó con el entorno de desarrollo de MIT app inventor. Permite realizar aplicaciones de forma sencilla y gratuita mediante la

programación con bloques. En el [Anexo IV](#) se encuentra el link de la aplicación web para desarrollo y la aplicación desarrollada para su comunicación mediante comunicación WiFi.

Modo de uso

Para poder controlar el dispositivo, se debe colocar el IP generado por el código en el cuadro inferior provisto para tal fin, con la misma sintaxis que se muestra en el mismo. El ESP-32 y el celular mediante el cual se controla, deben estar conectados al mismo WiFi. La programación permite dos formas de control del prototipo: manual y mediante coordenadas. En la *Imagen 28* se puede ver la aplicación y la utilidad de cada botón. Cada modo de uso se encuentra explicado detalladamente más adelante en la sección de [Utilización](#).

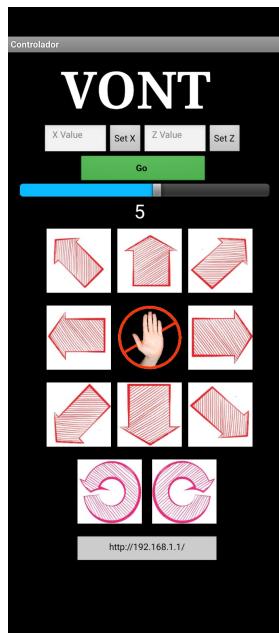


Imagen 28. Aplicación del VONT.

Python

Se utilizó el software [stella_vslam 0.3.9](#) para el desarrollo de la visión artificial en este proyecto. Stella Slam permite el mapeo de un espacio desconocido trazando puntos a través de la triangulación. Esta nube de puntos se guarda como un mapa y luego puede localizarse en el espacio mediante la información adquirida. En otras palabras, puede reconocer dónde se encuentra una imagen analizando el mapa realizado. Alimentando el software con imágenes en tiempo real, obtenemos la localización del vehículo. En este proyecto se utiliza para extraer las coordenadas del plano del suelo y también el ángulo formado por el VONT (ver *imagen 29*).

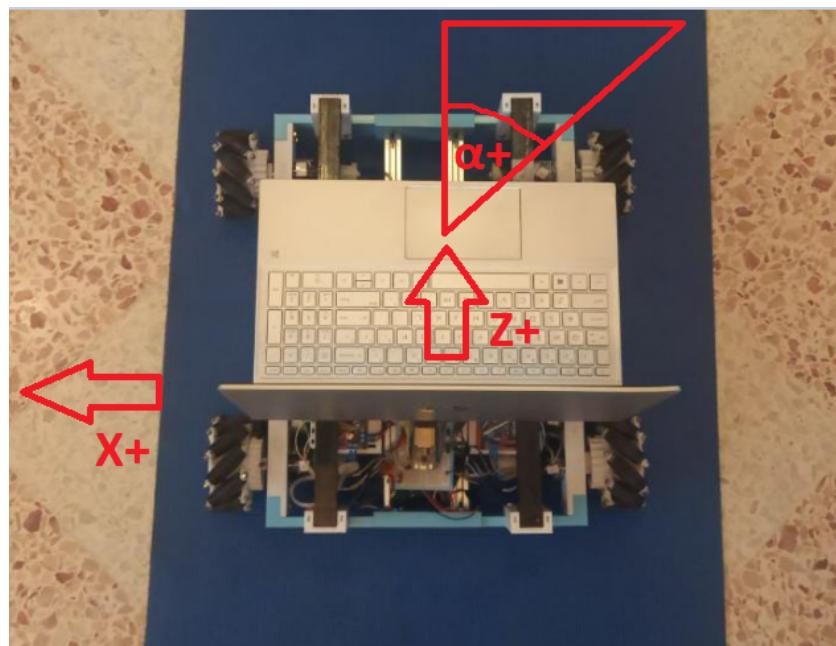


Imagen 29. Ejes de coordenadas del VONT.

Guia instalacion de stella_vslam sus dependencias y Python Bindings

Antes de iniciar con la descarga del Stella se debe realizar la instalación de Linux. Para poder utilizar Stella, se debe ejecutar en Linux debido a la dependencias de este programa. La

instalación del sistema operativo se puede hacer en un disco externo aunque de esta forma la retroalimentación será más lenta, por lo que se optó por la segunda opción que fue hacer de Linux el sistema operativo de la PC.

Linux es un sistema operativo el cual es de código abierto, esto ayuda debido a su gran adaptación y facilidad en las descargas de bibliotecas.

Primero ingresar a la [guía de instalación de Stella_vslam](#), la misma indica paso a paso qué es lo que se debe descargar para utilizar el software.

Durante las siguientes instalaciones se utilizará para todas, el comando *sudo* en el terminal, el cual permite obtener permisos de administrador, los cuales son necesarios para la instalación de Stella.

En la guía, en primer lugar se clona el repositorio de GITHUB, esto se realiza ya que facilita la convivencia de los archivos.

Para que el sistema funcione se deben descargar las siguientes dependencias:

- **Eigen:** librería de álgebra lineal que se utiliza dentro del mapeo.
- **G2o:** permite graficar el mapeo para así poder visualizarlo.
- **SuiteSparse:** requerido por G2o debido a su conjunto de algoritmos de matrices
- **FBoW:** (Fast Bag of Words) modela el conjunto de datos, el cual permite recopilar y administrar los mismos.
- **Yaml-cpp:** permite calibrar la cámara para obtener imágenes anti distorsionadas por la cámara.
- **OpenCV:** biblioteca de visión artificial la cual se utiliza para el procesamiento y análisis de imágenes.

Luego en la instalación se debe decidir entre si utilizar PangolinViewer o SocketViewer. En nuestro caso utilizamos el primero debido a su facilidad en el uso de la herramienta. Ambas son bibliotecas para crear prototipos de algoritmos 3D basados en imágenes. Este programa muestra visualmente el mapeo que se realiza a través de Stella_vslam.



También se debe instalar el módulo para calibrar la cámara y así poder utilizar los parámetros que devuelve el módulo de la misma, para obtener las imágenes anti distorsionadas. Para realizar esta instalación se clona el siguiente [repositorio de Github](#).

Finalmente se instala Python y openvslam bindings desde la [guía de instalación](#), lo cual permite controlar el sistema openvslam desde Python. De esta forma se pueden cargar y guardar distintos mapas, alimentar stella_vlsam con imágenes para obtener la matriz de pose. Dentro de la [guía de instalación](#) se pueden encontrar los pasos a seguir de manera detallada para la descarga de los archivos.

Librerías utilizadas

- Numpy
- Serial
- Math
- Time

Comunicación entre PC y ESP-32

El flujo de información de la PC al ESP-32 se realizó a través del puerto serie. El puerto serie es una conexión alámbrica que envía una secuencia de bits.

Para enviar desde la PC un *string* se utilizó la librería *serial*. La PC envía un *string* con las coordenadas (Z, X y α) en las cuales se encuentra el VONT. El ESP-32 procesa el mensaje y realiza el movimiento autónomo si corresponde.

Utilización

En esta sección se explica cómo utilizar el VONT una vez ya está armado, con los códigos cargados y todo el cableado realizado.

Guía uso de Stella vSlam

Antes de comenzar, se debe realizar, por única vez, la calibración de la cámara. Dicha calibración se encuentra en el [Anexo V](#).

Luego de instalar todo, utilizando el path donde estará instalado el Stella 0.3.9. se entra desde la terminal hasta build donde estarán: el mapa, la calibración de la cámara, el vocabulario y el programa para referenciar a la cámara con el pangolín.

En nuestro caso la ubicación de estos archivos están en: Descargas/VONT/0.3.9stella/build

Para llegar al mismo se utiliza la función cd para así entrar de carpeta en carpeta.

Luego se ejecuta el siguiente comando el cual abrirá el Pangolin y permitirá mapear **sudo ./run_camera_slam -n 0 -v ./orb_vocab.fbow -c ../example/aist/ALE.yaml --map-db-out map.msg**

Aquí se ejecuta con permisos de administrador (sudo), la cámara 0 (webcam), los datos modelados, utilizando la calibración de la cámara, para así mapear y guardar el mapeo en el archivo map.msg.

Al ejecutar el comando comienza el mapeo. Aquí se podrán visualizar los puntos en el espacio del lado derecho y los puntos referenciados en el lado izquierdo.

También se podrá ver donde está ubicada la cámara actualmente (triángulo celeste) y el recorrido que fue realizando durante el mapeo (triángulos verdes) (ver *Imagen 30*).

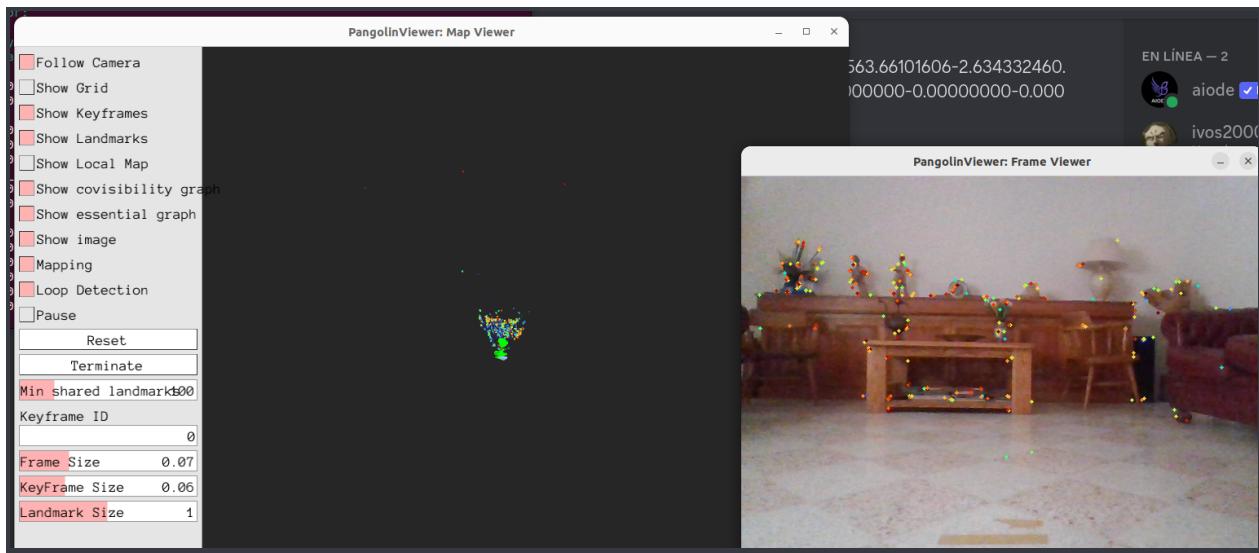


Imagen 30. Mapeo en pangolin.

Para mapear es importante moverse dentro del ambiente controlado completando ciclos (cuadrados, círculos, triángulos, comenzar en un punto y volver el mismo a través de un recorrido), de esta forma se podrá referenciar los puntos llegando desde distintas ubicaciones y así encontrar los puntos desde distintas posiciones. Para finalizar se pulsara **Terminate**, guardando el mapeo efectuado.

Luego se podrá corroborar si el mapeo fue correcto ejecutando `sudo ./run_camera_slam --disable-mapping -v ./orb_vocab.fbow -n 0 -c ../example/aist/ALE.yaml --frame-skip 3 --no-sleep --map-db-in map.msg`

En este caso si es correcto el mapeo, detectará puntos mapeados dentro del *Pangolin* y los pondrá en celeste para hacer referencia a los mismos, podremos movernos dentro del mapeo verificando los mismos (ver *Imagen 31*).

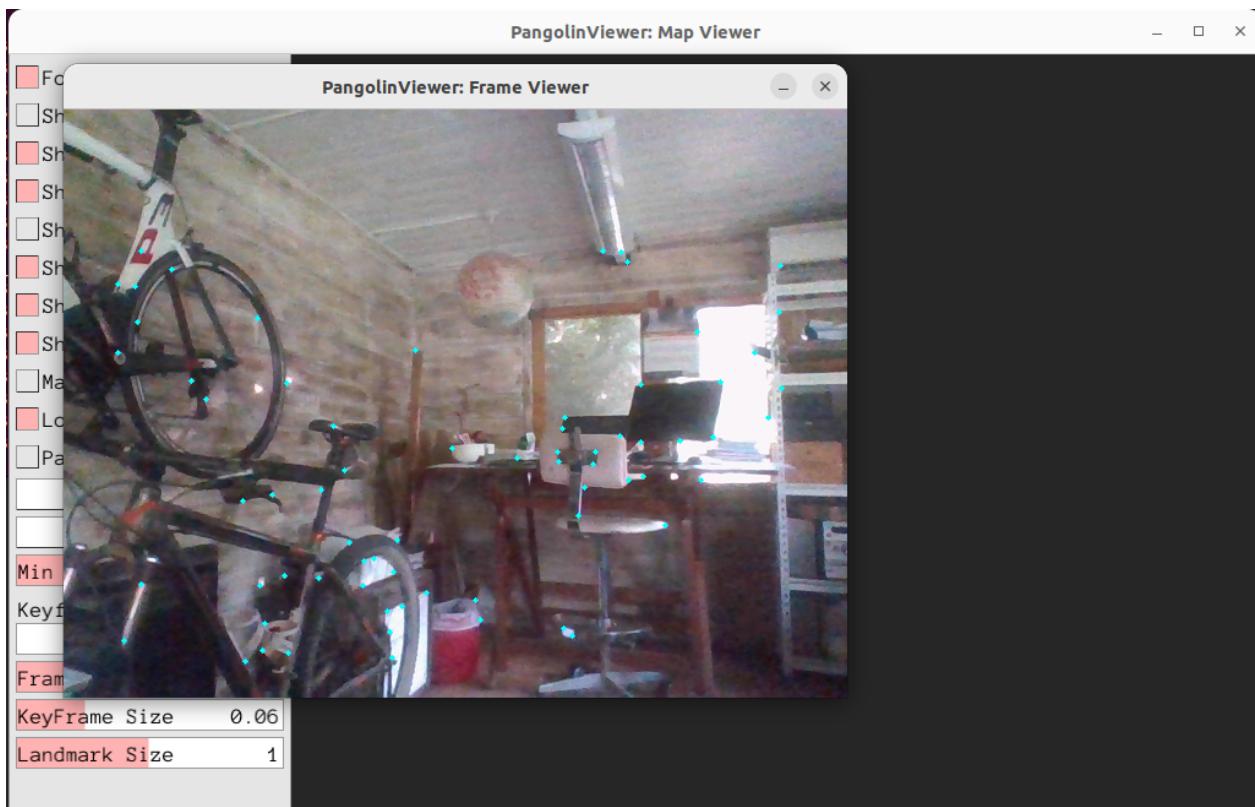


Imagen 31. Chequeo del mapeo

Al finalizar copiar y pegar el mapa dentro de la carpeta de Descargas/VONT/BINDINGS colocando el nombre que deseemos al mismo.

Aquí primero se debe ingresar a los cod1, cod2, cod3 y cod4 copiando dentro del código el mapa nuevo `parser.add_argument("-p", "--map_db", help="store a map database at this path after SLAM", default=".//MAPAREALIZADO.msg")`. Luego se ejecutará en la terminal el comando `python3 cod1.py` donde se deberá localizar gracias al mapeo previamente realizado. Mostrará una matriz de Pose en pantalla (ver *Imagen 32*) que varía dependiendo de la posición de la cámara. En este punto fijaremos el origen, donde pondremos la computadora con la cámara ortogonal al piso para así obtener los ángulos más limpios posibles. También la computadora deberá estar en la posición que va a ser el que se defina como el (0,0) (ver *Imagen 33*).

```
ivan@ivan-HP-Pavilion-Laptop-15-eh0xxx: ~/Descargas/VON... Q E - □ ×
```

```
0.001212300.99996908-0.007770050.00068979  
-0.015186030.007787570.999854360.00156017  
0.0000000000.0000000000.000000001.00000000  
Timestamp 2760 , Pose:  
0.999912170.000442010.013246090.00197181  
-0.000394760.99999355-0.00356915-0.00267716  
-0.013247580.003563610.99990590-0.00126635  
0.0000000000.0000000000.000000001.00000000  
Timestamp 2790 , Pose:  
0.99995018-0.001268340.009901320.00518177  
0.001306060.99999191-0.00380421-0.00199594  
-0.009896420.003816950.99994374-0.00349715  
0.0000000000.0000000000.000000001.00000000  
Timestamp 2820 , Pose:  
0.99987339-0.000931130.015885070.00013837  
0.001153010.99990183-0.013964290.00665461  
-0.015870510.013980840.999776310.00195578  
0.0000000000.0000000000.000000001.00000000  
Timestamp 2850 , Pose:  
0.99981607-0.000072510.01917845-0.00300948  
0.000201940.99997722-0.00674717-0.00036737  
-0.019177520.006749800.999793310.00091584  
0.0000000000.0000000000.000000001.00000000
```

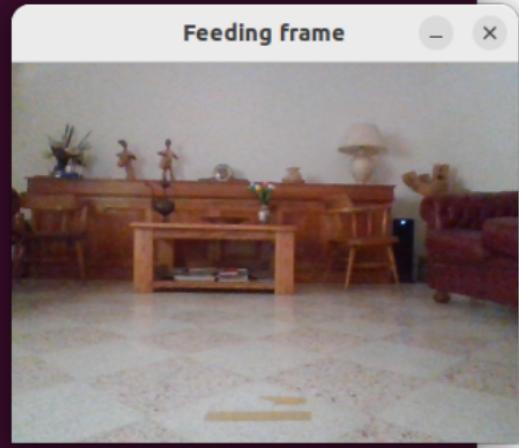


Imagen 32. Matriz de Pose inicial.

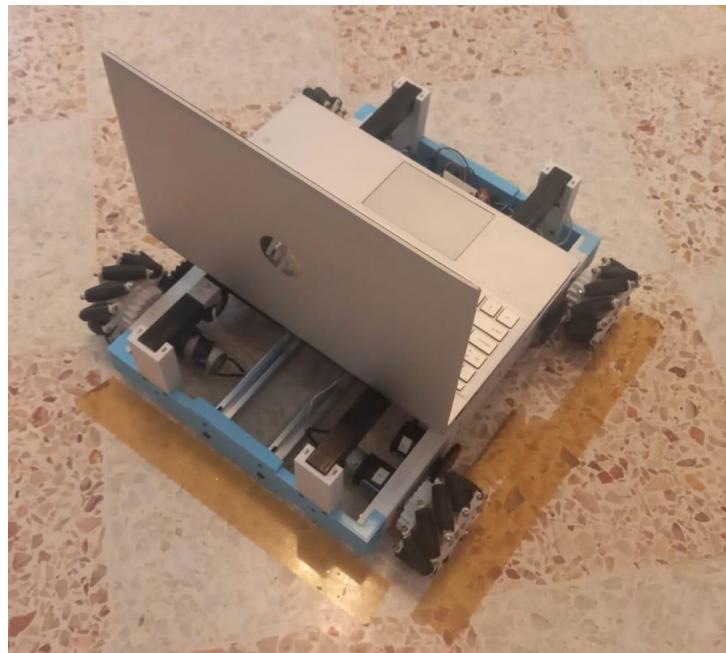


Imagen 33. VONT en origen para guardar matriz de pose.

Para convertir este punto en el origen se deberá copiar la matriz de Pose en este punto, definir la misma como X2 dentro de cod2.py e invertirla para luego multiplicar la matriz actual por la matriz invertida del origen (X1 luego de realizada la inversión de la misma), así obtendremos en el origen la matriz identidad (ver *imagen 34*). Luego ejecutar `python3 cod2.py` para probar mover la computadora y volver al origen obteniendo la matriz identidad nuevamente (ver *imagen 35*).

```
47
48 # matriz origen
49 x2 = np.array([[0.99999697, -0.00151724, 0.00193766, -0.00243044],
50 [0.001498980, 0.99995476, 0.0093931, -0.00065528],
51 [-0.00195182, -0.00939017, 0.99995401, -0.00153721],
52 [ 0.0, 0.0, 0.0, 1.0]])
53
54 x1 = np.linalg.inv(x2) # matriz origen invertida
55
```

Imagen 34. Matriz de pose en origen guardada en X2 y luego invertida.

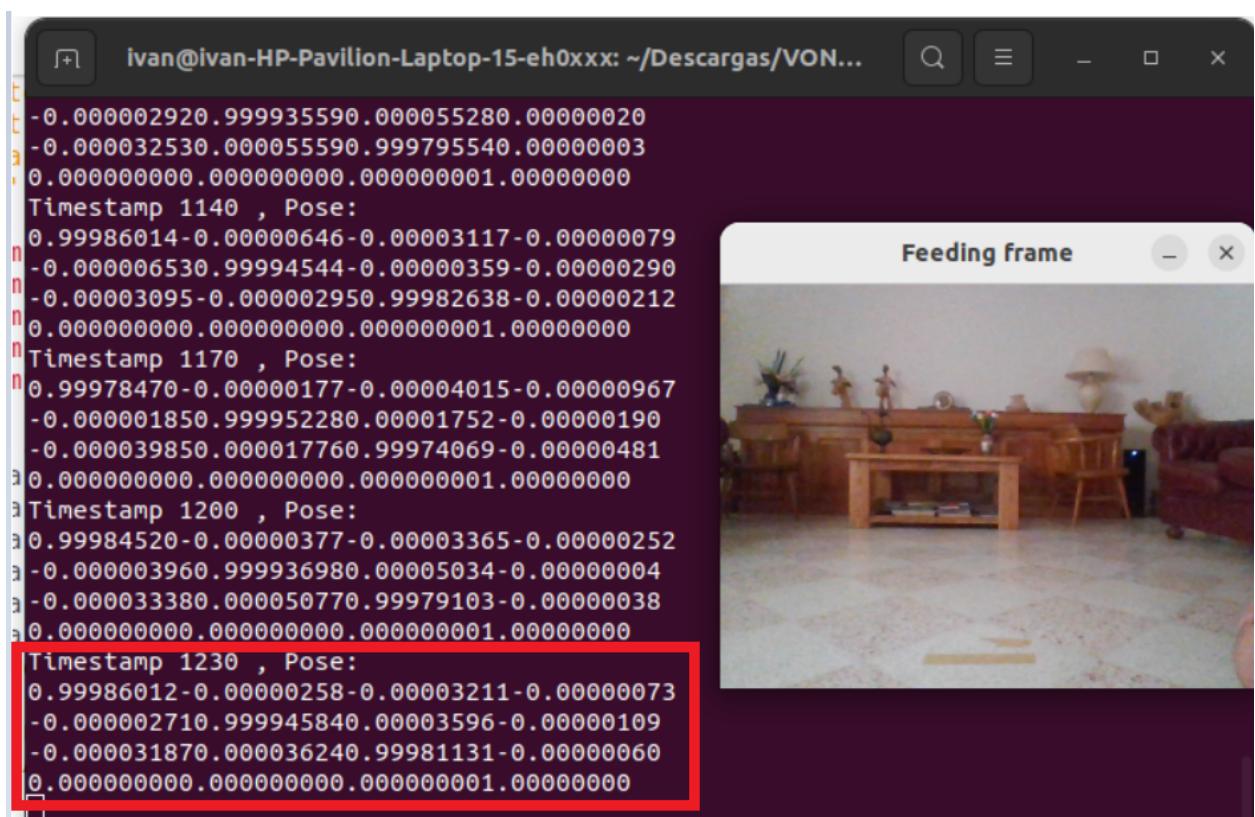
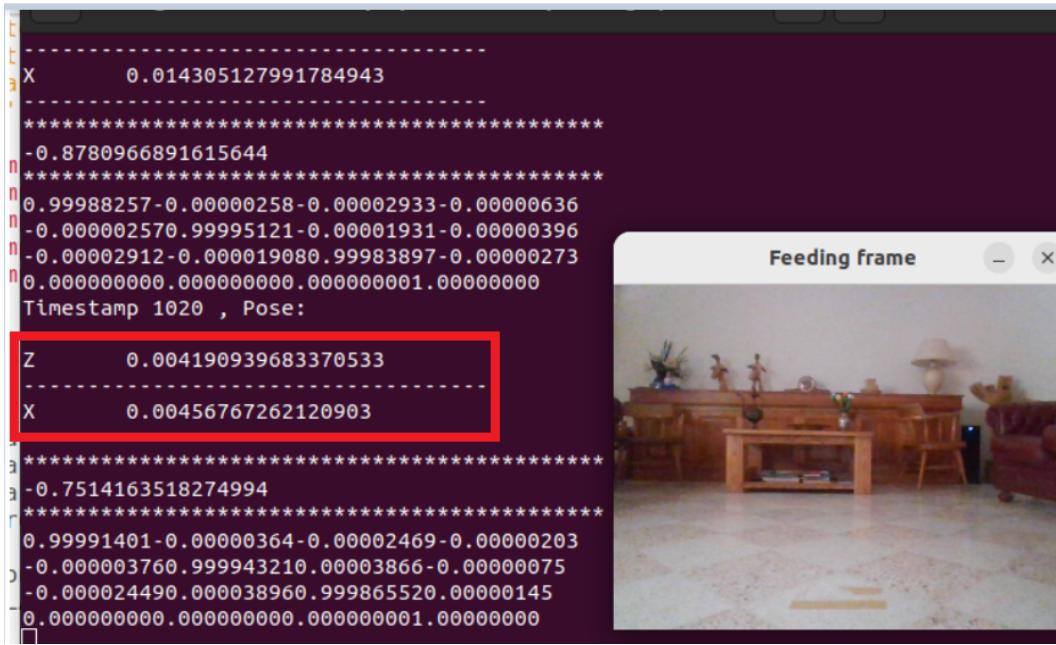


Imagen 35. Matriz identidad en el origen luego de multiplicar la matriz actual con la inversa del origen.

Luego se ejecuta ya con estos datos en terminal `python3 cod3.py`, habiendo cambiado previamente X2 como se realizó anteriormente (ver *imagen 34*). Aquí se convierten los valores de X y Z al sistema métrico y se obtiene el ángulo del VONT. Para realizar esto se debe mover al vehículo 1mt para adelante y 1 mt para la izquierda y guardar estos valores de la matriz. Siendo Z la posición (2,3) y X la posición (0,3). Luego se multiplican los valores guardados por los valores actuales, convirtiendo tanto X como Z al sistema métrico finalmente. Por último se probarán las tres posiciones para detectar que la transformación al sistema métrico se realizó correctamente (siendo las 3 posiciones de (Z, X): (0 ,0), (1 , 0), (0 , 1)) (ver *imagenes 36, 37 y 38*).

$$\text{equis} = (1/\text{Valor de X guardado}) * \text{x}$$

$$z1 = (1/\text{Valor de Z guardado}) * z$$

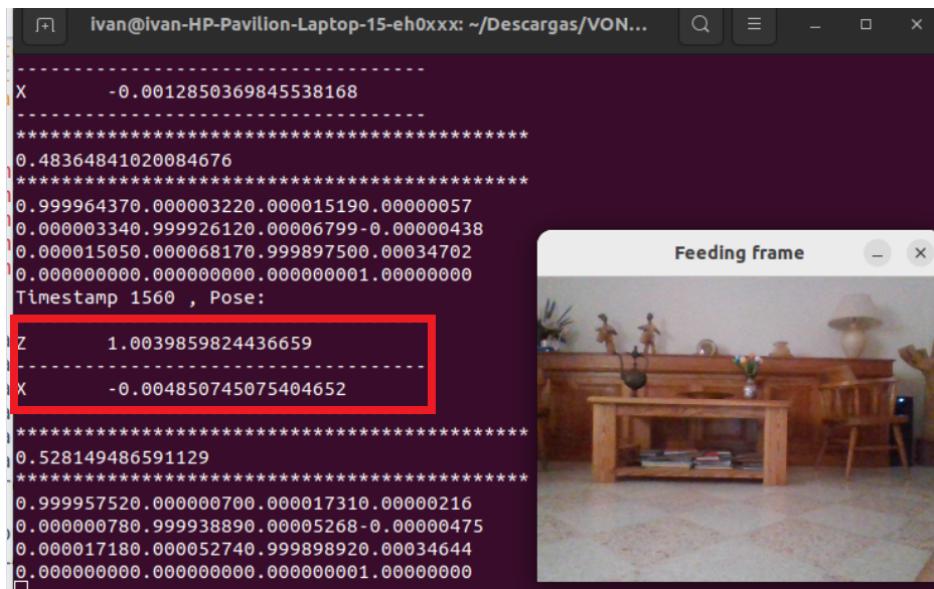


```

X      0.014305127991784943
*****
-0.8780966891615644
*****
0.99988257-0.0000258-0.00002933-0.00000636
-0.000002570.99995121-0.00001931-0.00000396
-0.00002912-0.000019080.99983897-0.00000273
0.000000000.000000000.000000001.000000000
Timestamp 1020 , Pose:
Z      0.004190939683370533
-----
X      0.00456767262120903
*****
-0.7514163518274994
*****
0.99991401-0.00000364-0.00002469-0.00000203
-0.000003760.999943210.00003866-0.00000075
-0.000024490.000038960.999865520.00000145
0.000000000.000000000.000000001.000000000

```

Imagen 36. (Z, X) en sistema métrico en el origen.

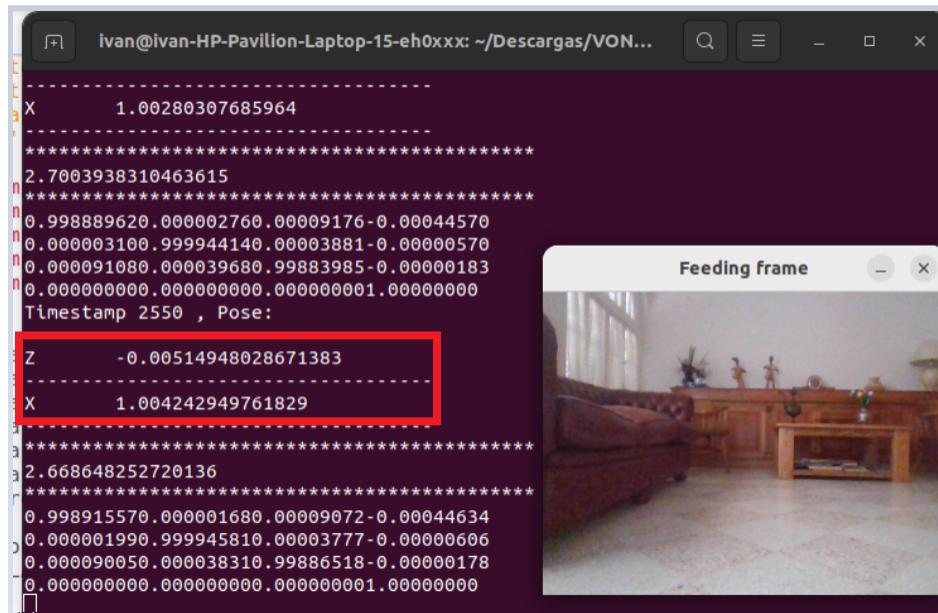


```

ivan@ivan-HP-Pavilion-Laptop-15-eh0xxx: ~/Descargas/VON...
X      -0.0012850369845538168
*****
0.48364841020084676
*****
0.999964370.000003220.000015190.00000057
0.000003340.999926120.00006799-0.00000438
0.000015050.000068170.999897500.00034702
0.000000000.000000000.000000001.000000000
Timestamp 1560 , Pose:
Z      1.0039859824436659
-----
X      -0.004850745075404652
*****
0.528149486591129
*****
0.999957520.000000700.000017310.00000216
0.000000780.999938890.00005268-0.00000475
0.000017180.000052740.999898920.00034644
0.000000000.000000000.000000001.000000000

```

Imagen 37. (Z,X) en sistema métrico en Z = 1 m y X = 0 m.



The terminal window displays the following data:

```
X      1.00280307685964
*****
2.7003938310463615
*****
0.998889620.000002760.00009176-0.00044570
0.000003100.999944140.00003881-0.00000570
0.000091080.000039680.99883985-0.00000183
0.000000000.000000000.000000001.00000000
Timestamp 2550 , Pose:
Z      -0.00514948028671383
X      1.004242949761829
*****
2.668648252720136
*****
0.998915570.000001680.00009072-0.00044634
0.000001990.999945810.00003777-0.00000606
0.000090050.000038310.99886518-0.00000178
0.000000000.000000000.000000001.00000000
```

The camera feed window is titled "Feeding frame" and shows a living room interior.

Imagen 38. (Z,X) en sistema métrico en Z = 0 m y X = 1 m.

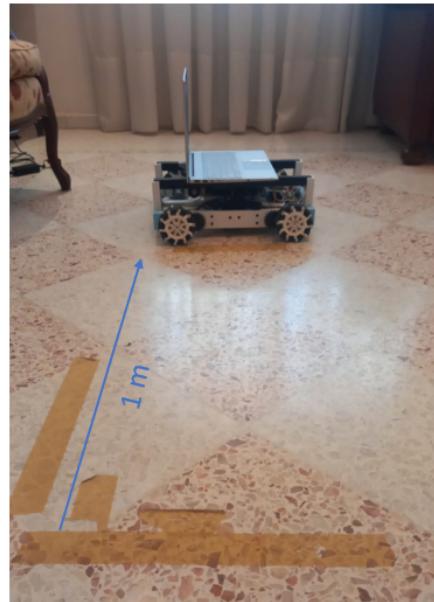


Imagen 39. VONT en X = 1 m y Z = 0 m.

Luego para obtener el ángulo se utiliza (0,0) que lo llamaremos A y (0,2) que lo llamaremos C, estos son los valores del cos α (A) con el cual se obtiene el ángulo (para obtener el mismo se le realiza el arcos a A y luego se pasa el valor al sistema de grados) y sen α (C) que dependiendo su signo sabremos para qué lado se está moviendo, se definió el sentido horario como positivo (ver imágenes 40 y 41).

```

90      # con c obtenemos el signo del angulo
91      # mirando de frente como si fuese la cam, a la derecha es positivo
92      if c > 0:
93          signo = -1
94
95      if c < 0:
96          signo = 1
97
98      # arcos del angulo para asi obtener el angulo real, tambien lo pasamos de radianes a grados
99      arccos = acos(a)*(180/3.1415)
100
101     angulo=arccos*signo #angulo que enviamos a la ESP

```

Imagen 40. Obtenemos el ángulo de la cámara

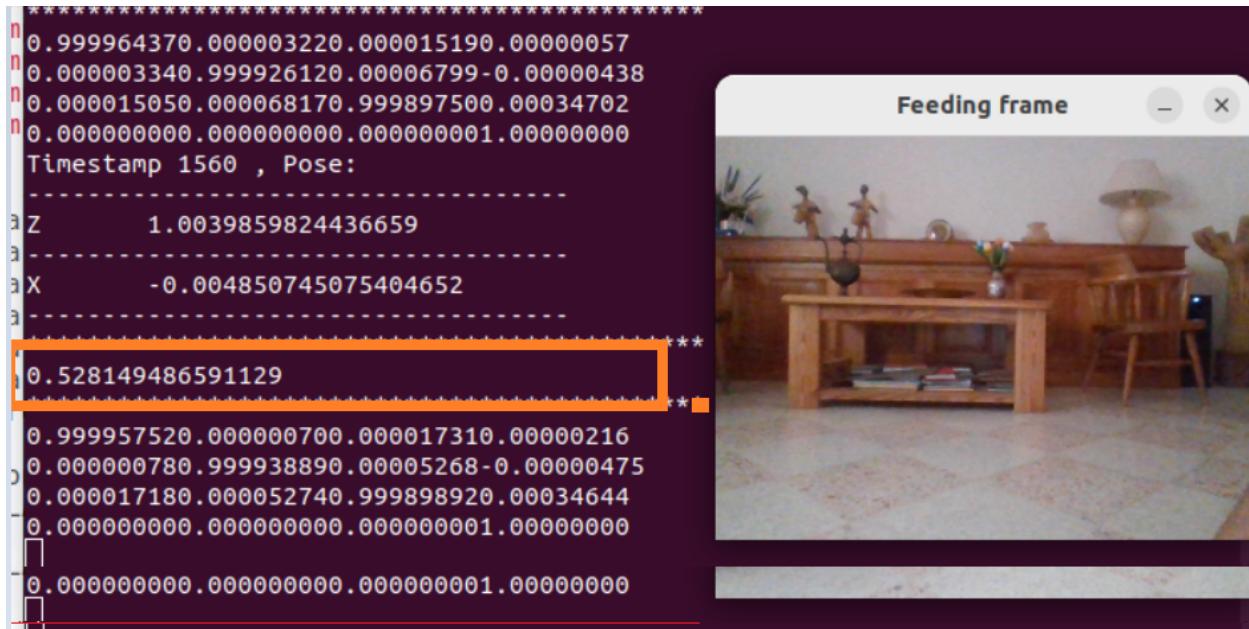


Imagen 41. Ángulo de la cámara.

El ángulo, X y Z son los tres datos que se necesita enviar a el ESP-32 para retroalimentar el sistema y poder hacer uso del mapeo en el VONT.

Por último con los datos que se recolectaron en cod1, cod2 y cod3 se puede ejecutar `python3 cod4.py` y así hacer funcional el trabajo de mapeo con el VONT. Este último, se ejecutará ya con la computadora conectada al ESP y desde Visual Studio Code debido a la comunicación con el ESP.

Antes de ejecutar el código se debe dar permiso al puerto con el comando `sudo chmod` y el puerto a utilizar. En nuestro caso `sudo chmod 666 /dev/ttyUSB0`

Uso manual con la aplicación

En primera instancia se encuentra el control manual. Es decir, que se elige un tipo de movimiento dentro de los 10 posibles o el Stop para dejar de moverse. Además tiene una barra deslizante para setear la velocidad de movimiento.

Antes de comenzar, en el sector inferior de la aplicación se debe colocar el IP del ESP-32. Este IP es enviado por el ESP-32 vía puerto serie a la PC. Dicho IP es siempre el mismo para cada red WiFi, siempre y cuando se mantenga exactamente el mismo nombre y la misma contraseña (si alguno de estos cambian, el IP cambia).

En primer lugar se debe setear la velocidad deseada para luego indicar la dirección de movimiento. En cualquier momento se puede presionar otra dirección o el botón de stop una vez para terminar con el movimiento. También es posible ir ajustando la velocidad con el VONT en movimiento (ver *Imagen 42*).

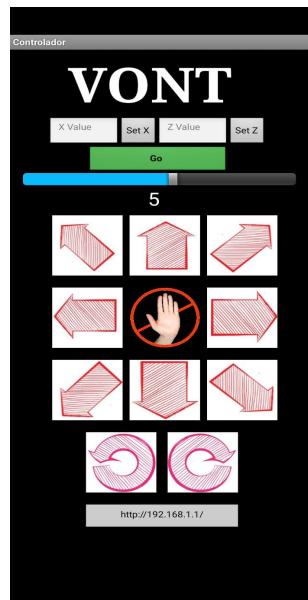


Imagen 42. Aplicación VONT.

Uso con coordenadas en aplicación

Para utilizar el VONT con traslado autónomo (mediante la retroalimentación de visión artificial) seteando coordenadas, se debe poner los valores deseados de las coordenadas X y Z, clickear en “Set X” y “Set Z” para que se guarden los valores en las variables y después de haber seteado la velocidad deseada, presionar “Go”. Se puede detener el movimiento con el botón de “Stop” o variar la velocidad con el slider en cualquier momento. Al llegar a las coordenadas indicadas el VONT frena solo. Es decir detiene todo tipo de movimiento (similar a haber presionado “Stop”).

Conclusión

En cuanto a las especificaciones de velocidad quedó por debajo de lo deseado. No obstante, esto se podría lograr simplemente con un cambio de motor reductor ya que las especificaciones objetivos de carga se superaron ampliamente. El prototipo es capaz de cargar hasta 11 kg, siendo el limitante la estructura de material impreso 3D. La velocidad medida fue de 1,44 m/s con 1.8kg en un suelo liso (deslizante). Con la retroalimentación mediante visión artificial se logró una precisión de 5 mts y 5°.

El proyecto posee algunos limitantes que se describirán a continuación así como también posibles mejoras para mitigarlos.

Limitaciones y mejoras realizables

Una de las limitaciones del prototipo son las vibraciones que se generan al trasladarse. Esto produce que se suelten cables y que los dispositivos eléctricos se deterioren. Dentro de las posibles mejoras están el rediseño de las piezas para que tengan más puntos de anclaje, y un rediseño donde contemple suspensión a las ruedas para poder amortiguar las perturbaciones. Además se podría reemplazar gran parte del cableado con una PCB mecanizada.

Otra posible mejora relacionada al diseño mecánico, son el material y diseño de los rodillos de las ruedas para lograr un mayor agarre en pisos lisos y el rediseño del chasis. Este actualmente está hecho de PLA. También se podrían incorporar rodamientos para las ruedas, aumentando la carga útil del VONT.

En cuanto a la aplicación, posee muchas limitaciones ya que se reduce al control de velocidad y movimientos predeterminados. Desarrollando una app con más funcionalidades, tales como bloqueo del número IP, seteo de distintos puntos (varias coordenadas) o trayectorias predefinidas diferentes a los movimientos ya mencionados, la utilidad se vería aumentada. El bloqueo del número IP se puede evitar implementando el uso de un IP fijo, es decir que no varíe dependiendo a qué red WiFi se conecta.

Actualmente el dispositivo no posee detección de obstáculos, por lo que choca cualquier objeto en el camino. Se le puede incorporar sensores para este fin (por ejemplo un sensor ultrasónico), o bien detección de obstáculos frontales con el uso de visión artificial.

Para que el prototipo sea escalable a la industria, es necesario que no requiera de una PC para funcionar. La PC podría ser reemplazada por una Raspberry Pi con una ESP-32 Cam para poder realizar un producto más afín a las necesidades del mercado.

En este momento, para obtener la matriz en sistema métrico, normalizamos los valores de Z y X con los valores que indica la matriz a 1 mt del origen en ambos sentidos. Para obtener una mejor matriz de pose, se podría realizar a través del método algebraico. Se debería obtener dicha matriz en forma de vectores y hacer un planteo algebraico con N ecuaciones y N incógnitas, de esta forma, se es más preciso en los resultados obtenidos al depender de más variables. Otra mejora sería poder visualizar los landmarks que se están detectando mientras corre el código de python y se imprime la matriz de pose. Esto ayudaría al usuario a entender mejor qué es lo que está detectando (y que no).

Para finalizar con las propuestas de mejora, se encuentra la secuencia de movimientos para llegar a las coordenadas objetivo, optimizando el camino a (Z, X). Por ejemplo, podremos arribar al lugar indicado primero acomodando el ángulo con el método SOHCAHTOA para direccionar el VONT a la coordenada objetivo, y finalmente avanzar derecho.

Anexos

Anexo I

Lista de materiales del VONT donde podemos encontrar cada componente, la cantidad, el material, la categoría y aclaraciones: [+ Lista de Materiales BOM](#)

Anexo II

Link para archivos CAD del VONT. Se encuentran en formato .STL y .par. En la carpeta STL podemos encontrar tres carpetas diferentes: una para los archivos que deben ser impresos en 3D para montar el chasis, otra para los archivos correspondientes a las ruedas que también deben ser impresos en 3D y una ultima carpeta donde están las reproducciones de los componentes que deben ser comprados (motor, encoder, batería, etc) o hechos de otros materiales (soporte de la carcasa, carcasa, etc).

https://drive.google.com/drive/folders/1bwY2njGJ3KmBZs24sobz9KPa-_Nfntu8?usp=share_link

Anexo III

Circuitado del VONT e imágenes auxiliares de pines del ESP-32.

https://drive.google.com/drive/folders/1I5Dh-sRsfscTXqrkfVRq4UQLBxpSYXNn?usp=share_link

Anexo IV

Carpeta con los códigos necesarios para el VONT.

https://drive.google.com/drive/folders/1yjhNJerWsQyt29_n_UnkHD4z6saKuYAB?usp=share_link

La app se puede descargar y editar desde la web de [mit app inventor](#) → Create app → Log in to gallery / Search in Gallery → Search: “VONT_app”.

Anexo V

 Calibracion de camara

Anexo VI

En este anexo podemos encontrar un análisis realizado para el desarrollo sostenible del VONT.

 Equipo 8 - TA- Entrega Final