



PROYECTO FINAL DE INGENIERIA

“Desarrollo e implementación del prototipo para el control del sistema de pulverización de herbicida de un robot desmalezador”

ALUMNOS:

- FOLMER, Fernando
- ISAÍA, Guillermo

DIRECTOR:

- KOFMAN, Ernesto

Rosario, Santa Fe, Argentina
2020

Agradecimientos

Queremos agradecer en primer lugar al Dr. Ernesto Kofman, nuestro director, por confiar en nosotros este proyecto y acompañarnos a lo largo del mismo, brindándonos su asesoramiento y compañía. También a gran parte del equipo que se encuentra abocado al desarrollo de este proyecto integral, por brindarnos su apoyo, en especial a Maximiliano García e Ismael Ait, quienes nos ayudaron en el montaje y modelado del robot respectivamente.

Por último, queremos agradecer al Centro Internacional Franco Argentino de Ciencias de la Información y de Sistemas (CIFASIS) por convertir un proyecto de esta magnitud en uno multidisciplinario y permitir que este sea llevado a cabo, por estudiantes.

Resumen

En el informe que sigue a continuación se trabajó sobre el módulo de pulverización de un prototipo de robot autónomo, diseñado para combatir malezas en diversos cultivos.

Se planteó como objetivo principal el desarrollo del sistema de control de un conjunto de electroválvulas que realizan dicha tarea.

El proyecto abarca desde el diseño de la solución, contemplando los inconvenientes que se presentaron en el mismo, hasta la realización y posterior evaluación de esta.

Finalmente, se presentan los resultados obtenidos.

Índice

Agradecimientos.....	2
Resumen.....	3
1. Introducción	6
1.1 Tecnología agropecuaria	6
1.2 Agricultura de precisión.....	6
2. El Proyecto.....	7
2.1 Motivación del proyecto	8
3. Nuestro Proyecto.....	9
3.1 Alcance del proyecto	9
3.2 Condiciones limites.....	10
3.3 Árbol de tareas	11
4. Marco Teórico.....	13
4.1 Mecánica del sistema de pulverización	13
4.2 Análisis de soluciones similares en el mercado.....	15
4.3 Análisis teórico matemático de la solución.....	19
5. Equipamiento Utilizado	29
5.1 Electroválvulas.....	29
5.1.1 Acción directa	29
5.1.2 Acción indirecta	30
5.1.3 Acción mixta	31
5.2 Microcontrolador (placa de desarrollo FRDM-KL46Z).....	32
5.2.1 Características principales	33
5.2.2 Diagrama de entrada/salida	34
5.3 Computadora a utilizar	35
5.4 Softwares utilizados para el desarrollo	36
5.4.1 Proteus 8 Profesional	36
5.4.2 Mcuxpresso IDE	36
5.4.3 Codeblocks IDE	37
5.4.4 Vmware.....	38
6. Desarrollo	40
6.1 Diagrama de bloques.....	40
6.2 Disposición de los equipos en el carro	41
6.3 Circuito de comando.	42
6.4 Software del microcontrolador	44

6.4.1	Descripción del software	45
6.4.2	Solución código C.....	45
6.5	Software de la computadora	47
6.5.1	Descripción del software de simulación	51
6.5.2	Descripción del software principal	52
7.	Ensayo Y Pruebas De Campo	56
7.1	Testeo de software sobre el micro (con PuTTY)	56
7.2	Testeo PCB y cableado.....	56
7.3	Testeo periodicidad de envío de tramas	57
7.4	Animación en Blender	58
7.4.1	¿Qué es blender?.....	58
7.4.2	Keyframes	58
7.4.3	Animación generada para testeo del software.	59
7.5	Animación en Matlab	63
7.6	Prueba real	64
8.	Conclusiones.....	65
9.	Bibliografía.....	66
10.	Apéndice.....	67
10.1	Hoja de datos transistor TIP29C	67
10.2	Hoja de datos buffer 74HC367N.....	69
10.3	Esquema final de placa PCB.....	70

1. Introducción

1.1 Tecnología agropecuaria

La **tecnología** juega un papel muy importante para optimizar las decisiones en el campo a través de la **información** oportuna, veraz y fiable.

La **agrotecnología** es la **tecnología** aplicada a la **agricultura**. Brinda los métodos y la maquinaria adecuada para **optimizar** la producción, enfocándose en los **procesos** utilizados en el **sector** para eficientizar el uso de los **recursos** y ayudar al **agricultor** en sus actividades. Es decir, la tecnología agropecuaria, hace mención a los conocimientos, técnicas y artefactos que permiten la utilización de elementos tecnológicos en las tareas agrícolas y ganaderas. Esta incluye desde distintos tipos de máquinas hasta trabajos de laboratorio que permiten incrementar la eficiencia de estas actividades.

El proceso y desarrollo de la tecnología en la agricultura son de suma importancia, ya que este sector es el encargado de alimentar al mundo. Se debe dar seguridad alimentaria para todos los países, pues enfrentan problemas de gran impacto como cambios climáticos, escasez de agua, aumento de población y empobrecimiento de esta. Por ende, se busca tener una producción suficiente de alimentos en cantidad y calidad, a bajo costo y con el menor impacto ambiental posible.

Estos son alguno de los motivos por los cuales la tecnología robótica es aplicada al sector agrícola, la cual se encuentra en un estado de desarrollo avanzado, con algunas realizaciones ya plenamente comerciales y otras validadas a nivel de prototipo. La inminente necesidad de aumentar la producción sin aumentar los recursos y minimizando el impacto ambiental, demanda, tal vez de manera inexorable, dar el paso de la mecanización a la automatización de la agricultura, en donde la robótica agrícola tendrá un protagonismo destacado.

1.2 Agricultura de precisión

Cuando de agricultura se trata, la forma convencional considera las condiciones de terreno como homogéneas y aplica la misma cantidad de insumos, por ejemplo, agua, fertilizantes y agroquímicos, a toda la superficie de siembra. Esto incrementa los costos de inversión y aumenta los riesgos de contaminación ambiental. Por estas razones, el mundo le ha dado lugar a la *Agricultura de Precisión*, que crece día a día. Esta brinda un aumento y una notable mejoría al rendimiento del cultivo, como lo pueden ser los fertilizantes y los pesticidas, además de optimizar los recursos económicos.

El principio básico de la agricultura de precisión es aplicar la cantidad justa de insumos en el lugar exacto y en el momento adecuado. Con este objetivo se usan sistemas captores y sensores capaces de detectar, tratar y almacenar información del cultivo con el fin de mantener un monitoreo constante del mismo. Además, se usan dispositivos de comunicación que permiten mantener el flujo de información constante de estos datos. En este tipo de agricultura se utilizan también dispositivos de procesamiento global, imágenes satelitales, herramientas de software y la red mundial (internet) para mejorar la producción, con el menor impacto ambiental y costo posible.

2. El Proyecto

Hace ya unos años, en el Centro Internacional Franco Argentino de Ciencias de la Información y de Sistemas (CIFASIS) se está llevando a cabo el desarrollo de un prototipo de robot pulverizador.

En él, se basaron varios proyectos anteriores de grupos de estudiantes de diferentes ramas de ingeniería de la Facultad de Ciencias Exactas, Ingeniería y Agrimensura (FCEIA) perteneciente a la Universidad Nacional de Rosario.

El prototipo puede verse en la Figura 2.1 y tiene como principal objetivo la eliminación de malezas en cultivos mediante un adecuado tratamiento. El robot debe ser capaz de desplazarse entre los surcos del cultivo siguiendo una determinada trayectoria y diferenciar las malezas del propio cultivo. Una vez detectado un objetivo, debe aplicar una dosis del agroquímico correspondiente sobre la misma de manera local.

Para lograr este objetivo, el proyecto se puede desglosar de la siguiente manera:

- a. Diseño mecánico del robot.
- b. Dimensionamiento y control de los motores eléctricos de tracción y dirección; medición de las variables necesarias para poder conocer la dirección del vehículo y reconstruir, la posición y velocidad del mismo por odometría.
- c. Desarrollo de convertidores de potencia para administrar el sistema de energía y dimensionamiento de las celdas solares para disponer de la energía necesaria en todos los procesos del vehículo.
- d. Procesamiento de imágenes: Por un lado, identificar los surcos y el terreno para poder definir trayectorias. Y por el otro, diferenciar maleza de cultivo para su posterior tratamiento selectivo.
- e. Georreferenciación mediante técnicas de GPS de precisión para poder identificar la posición del vehículo dentro de un terreno agrícola, ayudar en la generación de trayectorias y realizar mapas con ubicación de maleza y cultivo.
- f. Desarrollo e implementación de algoritmos adecuados para el seguimiento de trayectoria.
- g. Diseño y control del sistema de pulverización selectiva inteligente.

Nuestro proyecto corresponde al punto g). El mismo se plantea como la continuación de un proyecto de ingeniería Mecánica, donde se dotó al prototipo del sistema hidráulico necesario para proporcionar a los picos el caudal necesario de herbicida, el cual se encuentra almacenado en tanques sobre el prototipo. Así mismo, se incorporó al prototipo la estructura mecánica para la colocación de las electroválvulas correspondiente, con su conexión al sistema hidráulico.



Figura 2.1 Robot desmalezador en el CIFASIS

2.1 Motivación del proyecto

Las malezas se abren paso en medio de los cultivos y compiten con ellos por el agua del suelo, los nutrientes, el espacio y la luz, reduciendo el rendimiento y en ocasiones portando enfermedades que afectan el cultivo. Además, suelen ser muy competitivas, con una gran capacidad de adaptación a condiciones adversas y un rápido crecimiento. La práctica más utilizada en la agricultura tradicional para el control de malezas es la de aplicar herbicida pulverizando de forma homogénea por todo el terreno. Las grandes dosis de productos químicos y la resistencia de las malezas constituyen un problema serio en la agricultura mundial, lo que provoca que el uso de agroquímicos sin control tenga efectos negativos, entre los que se destacan, el gasto innecesario de herbicida (pérdida económica) y el daño medioambiental (contaminación de suelo y aguas subterráneas). Consecuentemente se hace necesario un tratamiento específico por zonas, mejorando la eficiencia en el control de malezas y reduciendo la dosis de fitosanitario utilizado. A su vez, aplicar el concepto de dosificación selectiva en un vehículo autónomo realza aún más los beneficios antes mencionados.

Finalmente, la virtud de autonomía del vehículo nos independiza de personal abocado a su conducción. Logrando así mayor precisión gracias a que el uso de la tecnología optimiza en gran medida el recorrido a lo largo del terreno cultivado y asegura que no habrá solapamiento entre pasada y pasada (nunca se recorre un mismo surco 2 veces).

3. Nuestro Proyecto

En particular, en nuestro proyecto trabajaremos en el control de las válvulas del sistema de pulverización del robot.

El mismo consiste tanto del análisis de los datos obtenidos mediante el procesamiento digital de imágenes como de la activación de 9 electroválvulas ubicadas en un barral en la zona posterior del chasis.

Este control se llevará a cabo mediante la información obtenida con el tratamiento digital de imágenes del terreno en el que se encuentra el robot, las cuales le permitirán al sistema poder elegir qué electroválvulas deberá activar. De esta manera se busca administrar la cantidad óptima a dosificar para eliminar la maleza detectada sin un uso excesivo de herbicida.

Para alcanzar los objetivos propuestos se deben resolver los siguientes problemas:

- Análisis del protocolo de comunicación a utilizar entre el sistema de actuación y el de control.
- Desarrollo software para el análisis de información obtenida mediante el procesamiento digital de imágenes (DIP).
- Definición de la información enviada al microcontrolador.
- Diseño del circuito y PCB para la activación de las electroválvulas.
- Desarrollo de algoritmo para comandar electroválvulas (Microcontrolador).
- Diseño del circuito del producto final, en el cual incluirá toda la unidad activación (Microcontrolador y etapa de potencia)

3.1 Alcance del proyecto

El alcance del proyecto fija los límites dentro de los que se debe permanecer en todo momento mientras se desarrolla la implementación de la solución al problema planteado.

El proyecto se centra en el análisis de información obtenida por un previo tratamiento de imágenes, a partir del cual, se tomará la decisión de qué electroválvula/s se debe activar para la inyección de herbicida en la zona de maleza previamente detectada.

Para esto se deberá realizar lo siguiente:

- Programación del software de toma de decisiones:
 - Esto implica, el desarrollo de un programa a partir de la información recibida por un previo tratamiento de imágenes del campo donde se encuentra el robot. Allí se analizará los datos obtenidos y se definirá qué electroválvula/s se deberá activar y se lo enviará al microcontrolador.
Este software se llevará a cabo en la computadora principal del robot (en adelante NUC)
- Implementar el Sistema de Control (en adelante SC):
 - Habiendo recibido, vía comunicación externa con la PC, la información necesaria, genere la señal que haga actuar la/s electroválvula/s correspondientes a la/s zona/s afectada/s. Para eso se hará uso de un microcontrolador (en adelante μC) a desarrollar en secciones posteriores.
- Implementar la alimentación del SC descrito en el ítem anterior.

- Aquí se deberá analizar si es suficiente utilizar el conversor ya implementado en el robot o si se deberá elegir uno de mayor capacidad. Además, se utilizará aquí también la referencia de tensión que provee la placa de desarrollo en la que está montado el μC .
- Adecuar las señales:
 - En este apartado se deberá diseñar el PCB para poder adaptar las señales de salida del μC (adecuar tensión y corriente) y a partir de estas, comandar las 9 (nueve) electroválvulas que conformarán el Sistema Pulverizador (las características de las mismas se describen en una sección posterior)

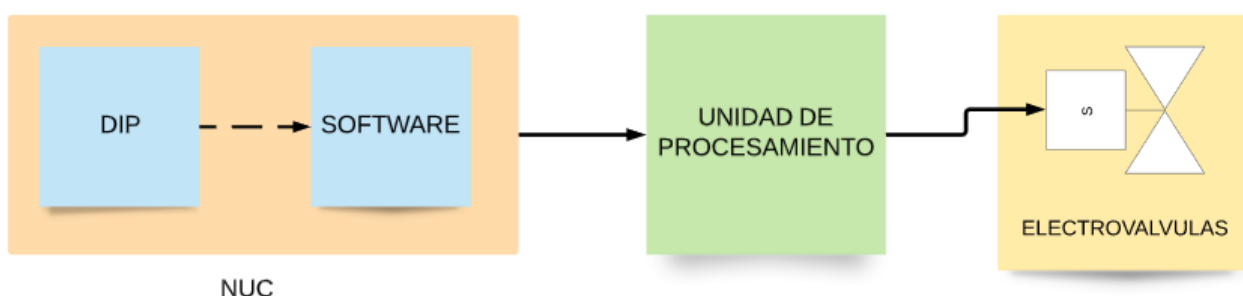


Figura 3.1: Esquema simplificado del proyecto

En la imagen superior se podrá ver los equipos que formarán parte del proyecto y su interconexión.

Aquí se detallará la acción que se realizará en cada uno de los bloques:

NUC: Se diseñará el software a partir de los datos obtenidos mediante DIP (Procesamiento digital de imágenes). Este software será el encargado de analizar la información obtenida y decidir qué electroválvula se deberá accionar para actuar sobre la maleza detectada.

Cabe destacar que el procesamiento digital de imágenes es una tarea que excede nuestro proyecto, siendo el punto inicial de nuestro proyecto la información.

UNIDAD DE PROCESAMIENTO: Este bloque contendrá tanto el microcontrolador como la etapa de potencia.

Aquí se diseñará el software en el microcontrolador el cual, a partir de la información recibida desde la NUC, generará las señales para la activación de las electroválvulas.

Además, se implementará el circuito necesario para la adaptación de las señales provenientes del microcontrolador para que estas puedan comandar las electroválvulas.

3.2 Condiciones limites

En esta sección se describen los puntos de partida que consideramos válidos para el desarrollo del proyecto. Se realizan hipótesis que permiten la ejecución de los alcances del proyecto. Aunque a priori puedan parecer idealidades, se contrastarán luego con los resultados obtenidos de forma tal de validar lo asumido.

Se detalla a continuación la lista de condiciones límite:

- La información/los datos que detallan la ubicación de la maleza y su dimensión (en superficie) se encuentra en la NUC luego de aplicar técnicas de procesamiento de imagen.

Esta cuestión escapa a nuestro alcance, ya que solo utilizaremos la información procesada.

Estos datos serán las coordenadas de la ubicación de la maleza detectada, así como el tamaño de la misma.

- Se garantiza que tanto la presión como el caudal del líquido a pulverizar permanecen constantes a los fines prácticos.
- Se garantiza además que los tanques siempre tendrán el nivel necesario para asegurar el correcto funcionamiento.
- La distribución de las 9 electroválvulas en el barral con los brazos extendidos es equidistante, y 2 electroválvulas contiguas superponen su zona de pulverización, es decir que, cuando una maleza se encuentre en la zona media entre las 2 zonas de pulverización podrá ser pulverizada por una o por ambas válvulas.

3.3 Árbol de tareas

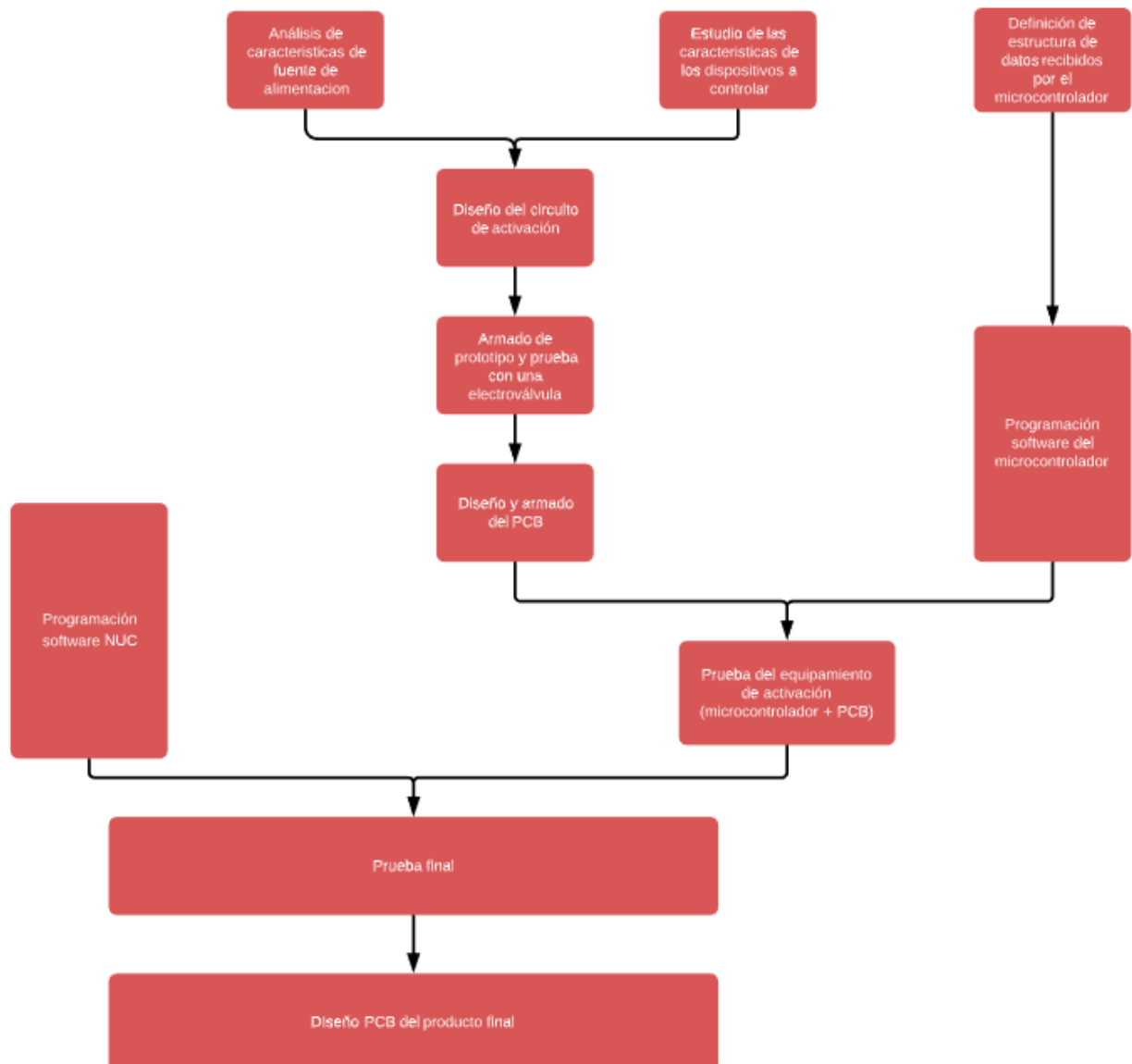


Figura 3.2: Diagrama de tareas del proyecto

La imagen anterior grafica el diagrama de las tareas a realizar, donde se explicita las tareas desarrolladas para poder llevar a cabo y cumplir con los lineamientos del proyecto.

Además, se muestra el orden en que se llevaron a cabo las tareas con el fin de poder organizar las actividades dentro del proyecto. Cumpliendo con el orden establecido, al comenzar las tareas ya se encontraba finalizada la etapa anterior.

El diagrama puede ser dividido en tres grandes rubros, donde dos de ellos se corresponden a la programación, siendo estos la programación tanto del microcontrolador como de la NUC y la tercera el diseño y armado de la placa correspondiente al circuito de potencia para el comando de las electroválvulas.

Aquí pasaremos a especificar el alcance que tenía cada una de las tareas con mayor detalle:

- **Análisis de características de fuente de alimentación:** El robot ya cuenta con conversores a 12 V el cual alimenta distintos componentes del mismo, por lo tanto, se deberá analizar si es necesario elegir una de mayor capacidad, o si la que ya se encuentra está capacitada para alimentar nuestro circuito.
- **Estudio de las características de los dispositivos a controlar:** Análisis de las características de las electroválvulas para la elección correcta de componentes electrónicos.
- **Diseño del circuito de activación:** Elección de la topología y los componentes que conformarán la placa final. Esto se llevará a cabo utilizando una herramienta de simulación de circuitos electrónicos.
- **Armado de prototipo y prueba con una electroválvula:** Se desarrollará un prototipo en una protoboard para la prueba del correcto funcionamiento del mismo.
- **Diseño y armado de PCB:** Se generará un PCB que contendrá el circuito de potencia.
- **Definición de la estructura de los datos recibidos por el microcontrolador:** Se deberá establecer la información que le será brindada al microcontrolador para el correcto funcionamiento del sistema, así como el protocolo utilizado para la comunicación.
- **Desarrollo de software NUC:** Se desarrollará un software que a partir de los datos obtenidos de la maleza analizará y definirá que acción tomar, es decir que electroválvula accionar.
- **Desarrollo de software Microcontrolador:** Se desarrollará un software que, a partir de la información recibida por la NUC, comandará cada una de las electroválvulas.
- **Prueba del equipamiento de activación:** Se probará el dispositivo integrado por el microcontrolador y el PCB para la activación de la electroválvula. Se enviarán datos desde una PC simulando los datos que recibirá desde la NUC esperando que cumpla con los requisitos preestablecidos.

- **Diseño del PCB del producto final:** A modo de conclusión se diseñará el circuito completo de la unidad de procesamiento (Microcontrolador y etapa de potencia), el cual consistirá en el producto final obtenido por el proyecto.

4. Marco Teórico

4.1 Mecánica del sistema de pulverización

El sistema de pulverización está compuesto por subsistemas que cumplen funciones parciales. En conjunto, proporcionan a los picos el caudal de líquido, de forma limpia, homogénea con la presión necesaria para la condición de pulverización que el operador propone. Aquí haremos una breve descripción de cada uno estos.

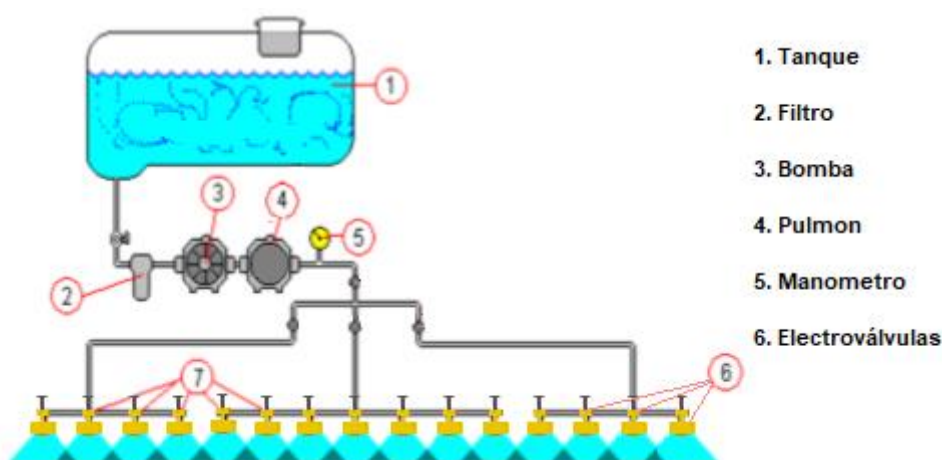


Figura 4.1: Esquema de sistema de pulverización

Tanque: Este componente es el encargado de almacenar el herbicida. Este prototipo tiene dos tanques que poseen una capacidad total de 32 litros cada uno. Este es de material plástico estable a los productos químicos.

Filtros: Son elementos que evitan el paso de impurezas que puedan dañar la bomba, que ocasionaría la reducción de la presión de trabajo por los taponamientos en el circuito y una pulverización deficiente por parte de las pastillas.

Bomba hidráulica: Es uno de los elementos fundamentales del equipo pulverizador, permiten transformar la potencia eléctrica en potencia hidráulica. La potencia mecánica es obtenida de la toma de potencia del tractor en máquinas montadas o de arrastre, y a través del propio motor de una unidad autopropulsada.

La bomba utilizada es de la marca Five Oceans 3.50GPM/45PSI HD 12V. Esta posee las siguientes características principales:

- ✓ Automáticas hasta 45 PSI
- ✓ Autocebantes hasta 183cms verticales
- ✓ Protección Térmica
- ✓ Entrega 13.2 litros por minuto (3.5 GPM)
- ✓ Diseño Compacto
- ✓ Puede trabajar en seco sin dañarse
- ✓ Temperatura máxima de agua: 43° C (110°F)

- ✓ Conectores entrada y salida: 1/2"
- ✓ Voltaje: 12v
- ✓ Fusible: 10amps

Manómetro: Este componente mide la presión del líquido en el punto del circuito donde está conectado. Esto significa que a lo largo del circuito existen diferentes presiones, aunque para uso práctico se tome sólo una como referencia.

Pulmón de agua: Los acumuladores permiten que la bomba alcance la presión de cierre más suavemente, lo que reduce el ciclo de encendido/apagado, lo que proporciona un mejor control de la entrega de agua caliente.

El utilizado es de la marca Five Oceans de 0.75litros cuyas características principales son:

- ✓ Volumen Interno: 0.75L /23.5OZ
- ✓ Carga previa: 10 PSI / 0.7 BAR
- ✓ Operación Máxima: 125 PSI / 8.6 BAR
- ✓ Válvula de llenado de aire: Válvula estilo neumático
- ✓ Diámetro Entrada / Salida: 13mm (1/2")
- ✓ Peso: 360gr

El tanque acumulador de agua (pulmón de agua) Five Oceans contiene un depósito de aire y agua debajo de la bomba. El aire actúa absorbiendo las fluctuaciones de presión, suavizando el flujo y haciendo que el sistema sea más silencioso.

Barral: Es la estructura metálica sobre la cual están montadas las conducciones (caños, mangueras, picos), con sus respectivas pastillas. La misma debe trabajar paralela al terreno y perpendicular al sentido de avance del equipo. Este está compuesto por un barral central, así también como dos brazos rebatibles, tanto del lado izquierdo como del derecho.

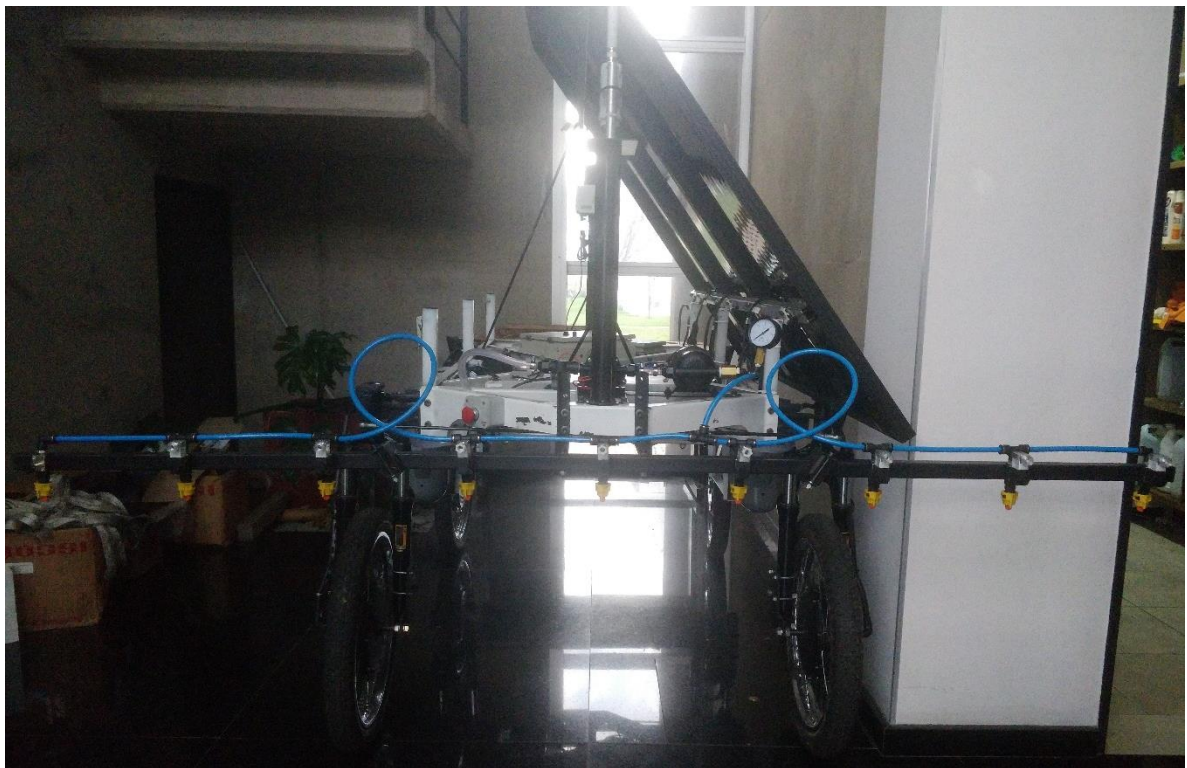


Figura 4.2: Barral del robot desmalezador con brazos rebatidos



Figura 4.3: Barral del robot desmalezador en estado extendido.

Picos: Los picos de una máquina pulverizadora son el conjunto de piezas que tienen como función alojar y posicionar las pastillas pulverizadoras sobre el barral.

4.2 Análisis de soluciones similares en el mercado

Actualmente se desarrollan prototipos y versiones comerciales de equipos que realizan en mayor o menor medida, tareas comparables con las llevadas a cabo por el robot desmalezador sobre el que trabajamos nosotros.

Mencionaremos los siguientes modelos comerciales:

Proyecto TERRAN (desarrollado por Plantium)

Es un vehículo de conducción autónoma, autopropulsado, reconfigurable y modular. Los desarrolladores lo definen como de fácil adaptabilidad al cambio de trabajo, con características que permiten el monitoreo y control del mismo de forma remota. El mismo es comandado por software diseñado específicamente para su uso, y es multiplataforma, admitiendo su uso desde una PC, Tablet o teléfono inteligente.

Puede utilizarse tanto para aplicación en barbecho, es decir, para la detección de malezas durante la etapa de crecimiento (“verde sobre marrón”); como para aplicación durante la etapa de cultivo (“verde sobre verde”).

Cuenta con un chasis construido en aleación de acero, lo que le confiere resistencia y reduce el peso para hacer un uso eficiente de la energía, minimizando la compactación del piso.

Los brazos de suspensión son fabricados de una aleación de aluminio que no se especifica.

Dispone del sistema universal de enganche de 3 puntos, que le permite acoplar los implementos existentes para realizar otro tipo de tareas.

La motorización consiste, tal como en el caso de nuestro robot, de motores brushless de alta eficiencia individuales, acoplados directamente a las ruedas y módulos controladores, también individuales, orientados al uso en agricultura. Hay disponibles dos tipos de tracción:

- una delantera, que desarrolla 16 hp
- una integral, que desarrolla 32 hp

La alimentación de los motores brushless se ofrece también en dos alternativas.

La primera de ellas, 100% eléctrica, formada por un conjunto de baterías que le brindan 8 horas de autonomía, con 4 horas de tiempo de recarga hasta completarse. Las baterías pueden ser cargadas mediante la red eléctrica, o mediante paneles solares.

La segunda, combina un generador (diesel o naftero) con un banco de baterías, para cuando no se dispone de fuentes eléctricas. Igual que en el caso anterior, brinda una autonomía de 8 horas por carga del generador.

Para la navegación autónoma, cuenta con el siguiente instrumental:

- Doble antena GNSS
- 3 giróscopos
- 3 acelerómetros
- Lidar
- Cámaras RGB
- Cámaras 3D
- Radar
- Ultrasonido

En lo que respecta al proceso de pulverización, el desarrollador declara que su diseño cuenta con un botalón de 10 metros de ancho de trabajo, 40 picos, y este fabricado en aleación de aluminio. El producto a pulverizar se almacena en un tanque roto moldeado de 600 litros de capacidad.

Finalmente, se informa que el ahorro potencial por la utilización del pulverizado selectivo, puede llegar al 80%. El conjunto cuenta con iluminación artificial, lo que le permite su uso durante horas sin luz natural.

A continuación, se incluyen las especificaciones encontradas en el sitio del fabricante:

Especificaciones extraídas del sitio web del fabricante (terran.com.ar)

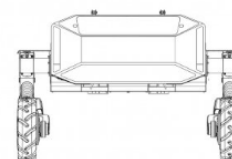
Especificaciones

Vehículo

Dimensiones (L x A x P)
Peso base
Peso cargado
Ancho de trocha
Despegue del suelo
Cubiertas
Llantas

Tracción Delantera

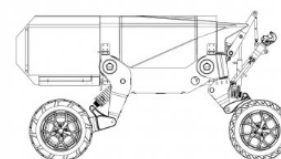
200 x 230 x 140 cm
850 kg
1600 kg
210 cm
70 cm
F-185/65-15 R-180/70-15
15 x 6.5"



Velocidad y performance

Carga máxima
Ángulo de partida
Velocidad máxima
Suspensión
Potencia

750 kg
14°
16 kmh
Monoshock
16 hp

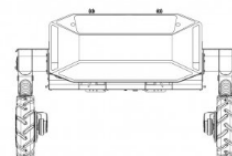


Vehículo

Dimensiones (L x A x P)
Peso base
Peso cargado
Ancho de trocha
Despegue del suelo
Cubiertas
Llantas

Tracción Integral

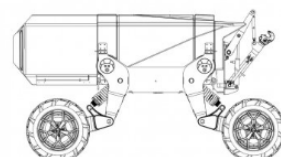
200 x 230 x 140 cm
950 kg
1900 kg
210 cm
70 cm
185/65-15
15 x 6.5"



Velocidad y performance

Carga máxima
Ángulo de partida
Velocidad máxima
Suspensión
Potencia

950 kg
30°
16 km/h
Monoshock
32 hp



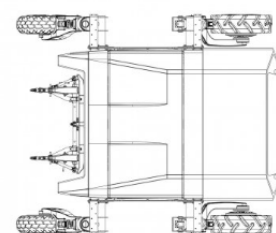
Energía

Composición baterías
Capacidad baterías
Tiempo de carga
Generador
Tiempo de uso a carga completa

Eléctrico

LiFePo
700 Ah
4 h

8 h



Energía

Composición baterías
Capacidad baterías
Tiempo de carga
Generador
Tiempo de uso a carga completa

Híbrido

LiFePo
200 Ah
2 h
Nafta o Diesel
10 h

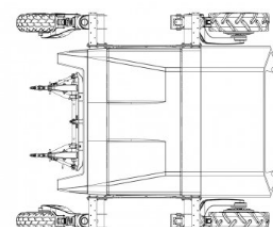
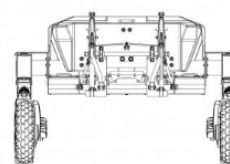


Figura 4.4: Especificaciones del proyecto Terran (Plantium)

Ambiente

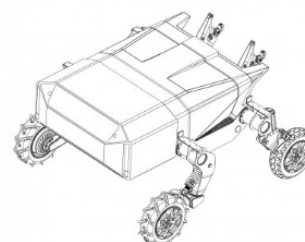
Temperatura operativa	-5° a 50°
Temperatura de almacenamiento	-5° a 60°
Clasificación IP	65



Aplicación

Ancho de trabajo	10 m
Tanque producto	600 l
Cañería	Acero Inoxidable 1/2"
Distancia entre portapicos	25 cm
Pastillas	TeeJet SS-40-020
Bomba	76 l/min a 3,4 bar

Pulverización



Milar Eco Snipper

Es un sistema de detección de malezas y aplicación selectiva desarrollado en Argentina (el primero de estas características en llegar al mercado en el país, según indica el fabricante), que permite aplicar herbicidas específicamente donde se encuentran las malezas, y no en la totalidad del lote.

A diferencia de nuestro sistema, el Eco Snipper no es autopropulsado, sino que se monta en el botalón de cualquier pulverizador terrestre. Cada unidad es autónoma, y se compone de: cámara, procesador, software específico, pico y válvula que abre y cierra donde corresponde.

Se estima que puede lograr ahorros de hasta el 80% en el uso de producto. La detección se realiza en tiempo real a través de cámaras ópticas que van marcando presencia “verde sobre marrón”- período sin cultivo presente.

Posteriormente, la información de todas las unidades se concentra en una central para poder relevar y almacenar los datos recabados durante la aplicación.

No ha sido posible acceder a las especificaciones detalladas del producto.

También fue posible encontrar las siguientes soluciones, pero no se pudo acceder a mayor información sobre las mismas. Por este motivo solo se enumeran las mismas a continuación.

- Greeneye Technology (Tel Aviv – Israel) <https://greeneye.ag/technology>
- Carbon Bee SmartStriker (Francia) <https://carbonbee-agtech.fr/>
- Bilberry (Francia) <https://bilberry.io/#home>
- BlueRiver Technology (California, Estados Unidos)

Es en esencia, un implemento que se añade al tractor, y que permite la detección de malezas y aplicación local de herbicida. Fue desarrollado para plantaciones de algodón en el sur estadounidense. <http://www.bluerivertechnology.com/>

4.3 Análisis teórico matemático de la solución

En esta sección se detallará, en un análisis lógico y matemático, como se logra obtener la combinación de electroválvulas a activar a partir de los siguientes datos de entrada. Para esto se seguirá por una serie de pasos, cumpliendo el siguiente diagrama de flujo.

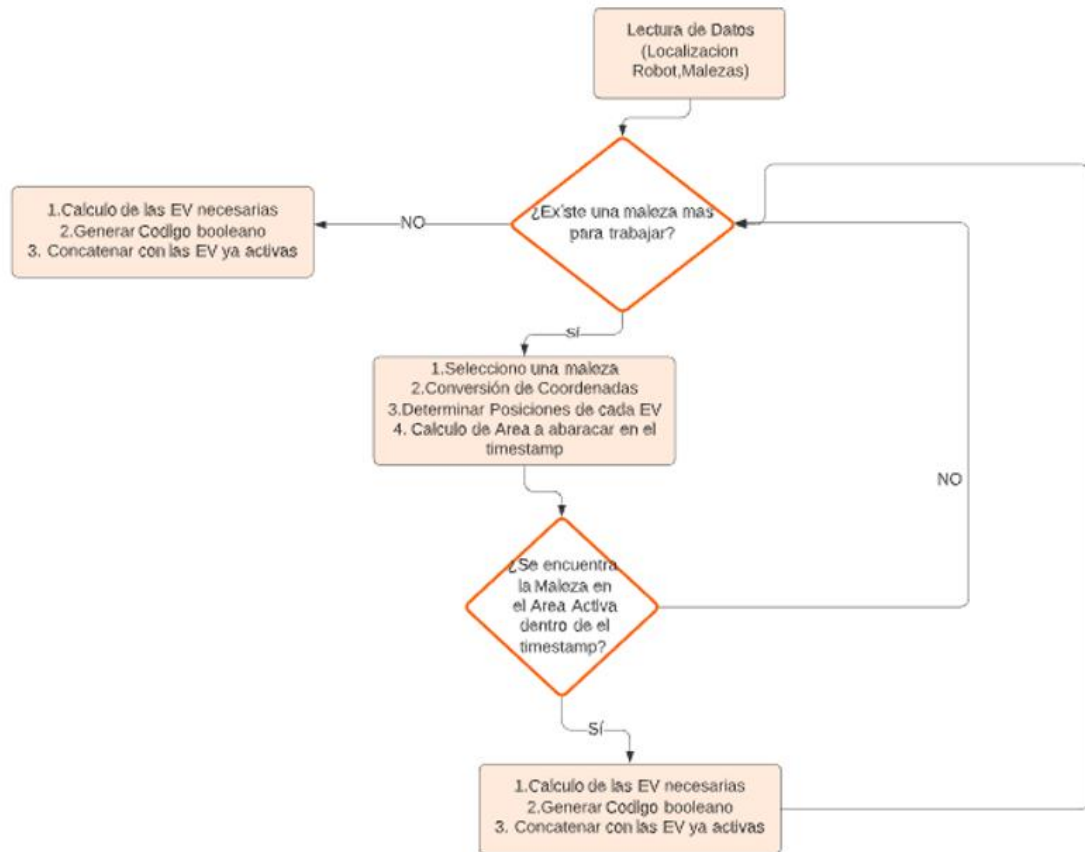


Figura 4.5: Diagrama de flujo de la solución adoptada

Cabe destacar que este proceso deberá realizarse periódicamente a medida que avanza el robot por el campo. En el caso del software se realizará a una tasa de 100ms.

Los datos iniciales de los cual se partirá el análisis son:

- Datos del robot:
 - ✓ Coordenadas X e Y en metros.
 - ✓ Orientación en radianes.
 - ✓ Velocidad.
- Datos de la maleza:
 - ✓ Coordenadas X e Y de su centro en metros.
 - ✓ Radio de la maleza en metros.

Para la definición de las electroválvulas a encender para actuar sobre las malezas, es necesario conocer algunos datos y medidas físicas del prototipo del robot, estos son:

- ✓ Largo del barral.
- ✓ Apertura de las Electroválvulas.
- ✓ Distancia desde el centro de coordenadas hasta el barral.
- ✓ Altura posicionada para el barral (cabe destacar que la estructura provee 3 posiciones distintas).
- ✓ Distancias entre electroválvulas.

Se puede ver todas las medidas físicas del prototipo ubicada en la siguiente figura:

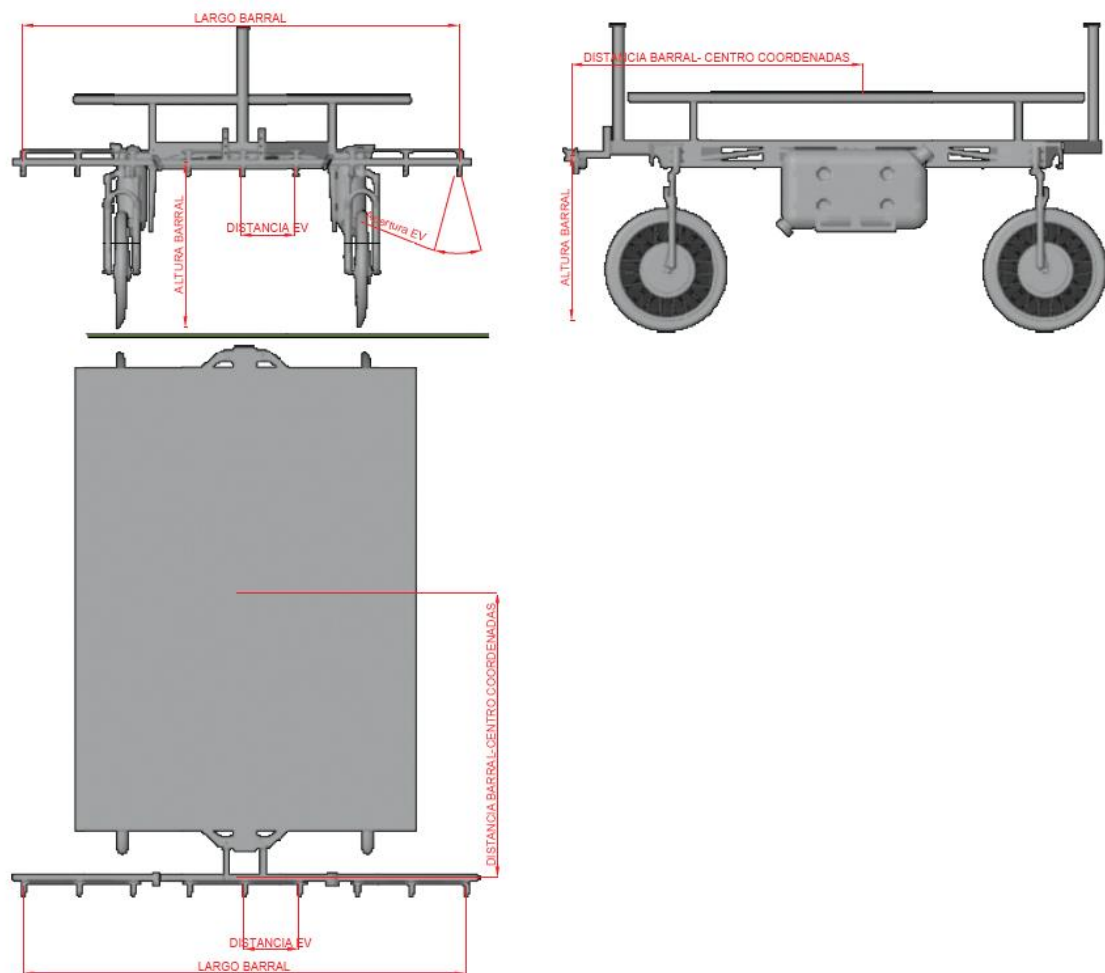


Figura 4.6: Medidas físicas del robot

Si dentro de la información de las malezas recibida, se encuentra más de una, se deberá repetir el siguiente análisis para cada una de estas, donde se tomará la primera maleza disponible y comenzará a evaluarse. Aquí se obtendrá como resultado final, en caso de que la maleza se encuentre dentro de la zona de acción del robot, un código de 1 y 0 correspondientes a las electroválvulas

necesarias para combatir cada maleza, las cuales se irán concatenando para el armado del código final a enviarle al microcontrolador.

Para lograr lo anteriormente planteado, se describirá el análisis y procesamiento que se deberá hacer a cada una de las malezas a actuar.

1. Cambio de coordenadas.

Dada una maleza, ubicada en las coordenadas (XM, YM) dentro del eje cartesiano global, y el robot encontrado en la posición (XR, YR) con un ángulo θ .

Se desea transformar, y obtener las coordenadas de la maleza en relación con el eje coordenado con su centro y orientación coincidentes a la del robot.

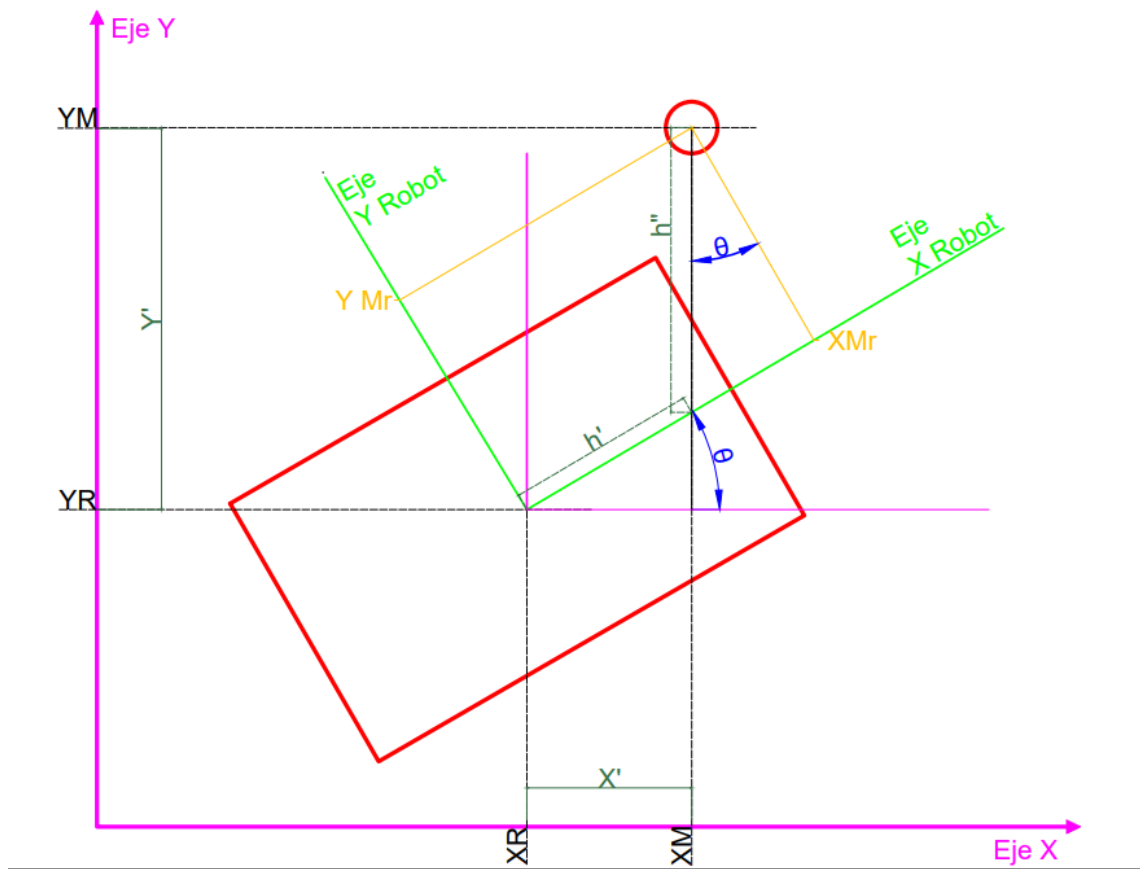


Figura 4.7: Referencias para el cambio de coordenadas entre planos

Por lo tanto, a partir de las relaciones geométricas que se ven en la figura se pueden encontrar las siguientes ecuaciones:

$$\begin{aligned} X' &= XM - XR ; Y' = YM - YR \\ h' \cos(\theta) &= X' \\ Y' - h'' &= h' \sin(\theta) \end{aligned}$$

Estas permiten transformar las coordenadas de la maleza a coordenadas asociadas al nuevo par de ejes (XMr, YMr), las cuales se obtienen de la siguiente forma:

$$\begin{aligned} YMr &= h'' \cos(\theta) = (Y' - h' \sin(\theta)) \cos(\theta) = \left[(YM - YR) - \frac{X' \sin(\theta)}{\cos(\theta)} \right] \cos(\theta) \\ &= (YM - YR) \cos(\theta) - (XM - XR) \sin(\theta) \end{aligned}$$

$$\begin{aligned} XMr &= h' + h'' \sin(\theta) = \frac{X'}{\cos(\theta)} + (Y' - h' \sin(\theta)) \sin(\theta) \\ &= \frac{(XM - XR)}{\cos(\theta)} + (YM - YR) \sin(\theta) - \frac{(XM - XR)}{\cos(\theta)} \sin^2(\theta) \\ &= \frac{(XM - XR)}{\cos(\theta)} (1 - \sin^2(\theta)) + (YM - YR) \sin(\theta) \\ &= \frac{(XM - XR)}{\cos(\theta)} \cos^2(\theta) + (YM - YR) \sin(\theta) \\ &= (XM - XR) \cos(\theta) + (YM - YR) \sin(\theta) \end{aligned}$$

2. Cálculo de las posiciones de cada electroválvula.

Teniendo en cuenta parámetros de características físicas y dimensiones del prototipo, se calculará las coordenadas del área abarcada por cada una de las electroválvulas.

El área abarcada por una electroválvula se determina de la siguiente manera:

$$AreaEV = 2 * AlturaBarral * \tan(AperturaEV/2)$$

A partir de este parámetro, junto con el largo de la barra, y la distancia entre electroválvulas se obtiene las coordenadas del área para cada electroválvula referenciada al sistema de coordenadas del robot.

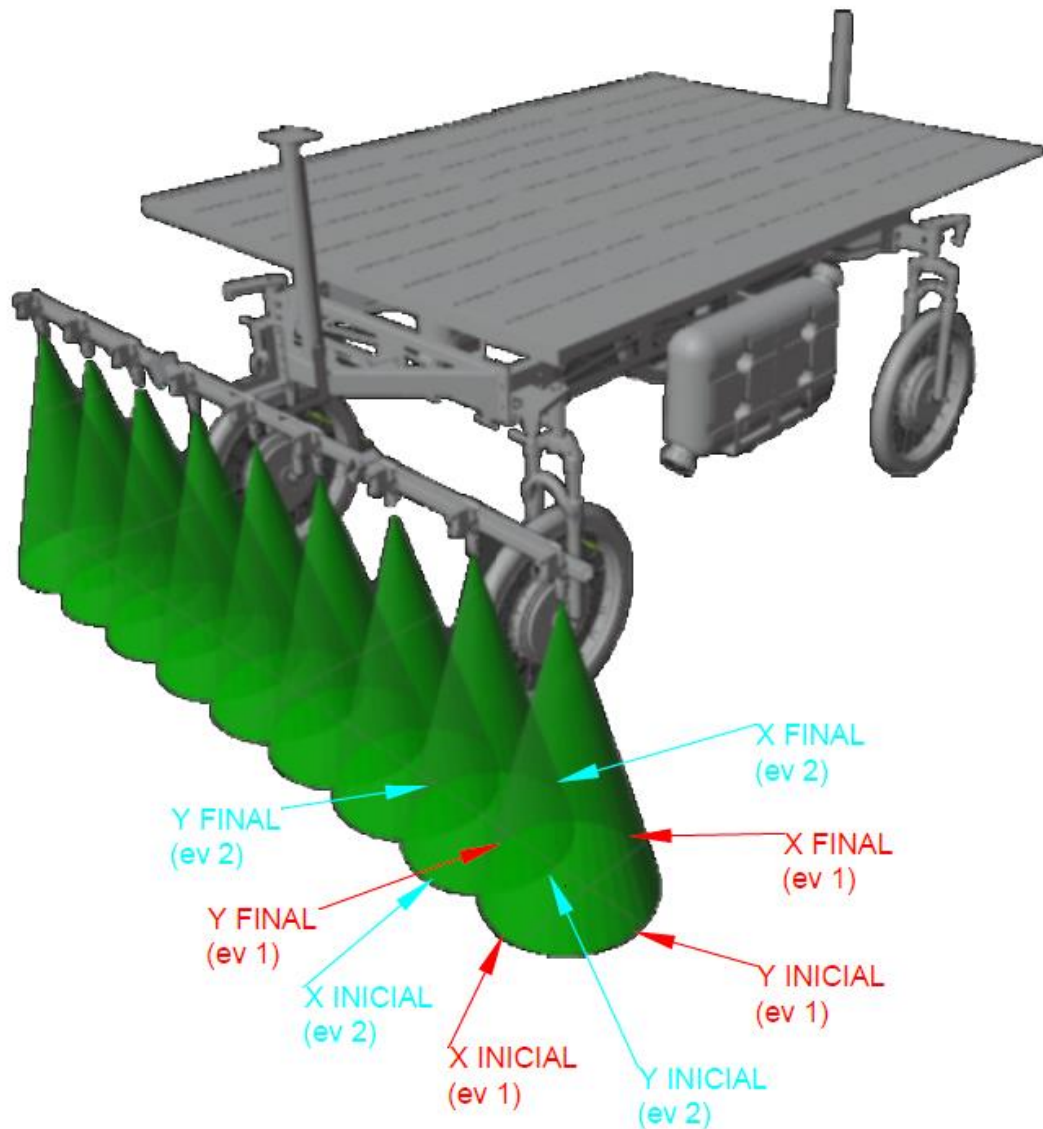


Figura 4.8: Ejemplo. Posición de las coordenadas de las electroválvulas

Estas estarán dadas de la siguiente manera:

❖ Electroválvula 1:

- $X_{INICIAL} = -Offset - \frac{AreaEV}{2}$
- $X_{FINAL} = -X_{INICIAL} + AreaEV$
- $Y_{INICIAL} = -\frac{Largo}{2} - \frac{AreaEV}{2}$
- $Y_{FINAL} = Y_{INICIAL} + AreaEV$

❖ Electroválvula 2:

- $X_{INICIAL} = -Offset - \frac{AreaEV}{2}$
- $X_{FINAL} = -X_{INICIAL} + AreaEV$
- $Y_{INICIAL} = -\frac{Largo}{2} + DistanciaEV - \frac{AreaEV}{2}$
- $Y_{FINAL} = Y_{INICIAL} + AreaEV$

❖ Electroválvula 3:

$$\begin{aligned} \rightarrow X_{INICIAL} &= -Offset - \frac{AreaEV}{2} \\ \rightarrow X_{FINAL} &= -X_{INICIAL} + AreaEV \\ \rightarrow Y_{INICIAL} &= -\frac{Largo}{2} + 2 * DistanciaEV - \frac{AreaEV}{2} \\ \rightarrow Y_{FINAL} &= Y_{INICIAL} + AreaEV \end{aligned}$$

❖ Electroválvula 4:

$$\begin{aligned} \rightarrow X_{INICIAL} &= -Offset - \frac{AreaEV}{2} \\ \rightarrow X_{FINAL} &= -X_{INICIAL} + AreaEV \\ \rightarrow Y_{INICIAL} &= -\frac{Largo}{2} + 3 * DistanciaEV - \frac{AreaEV}{2} \\ \rightarrow Y_{FINAL} &= Y_{INICIAL} + AreaEV \end{aligned}$$

❖ Electroválvula 5:

$$\begin{aligned} \rightarrow X_{INICIAL} &= -Offset - \frac{AreaEV}{2} \\ \rightarrow X_{FINAL} &= -X_{INICIAL} + AreaEV \\ \rightarrow Y_{INICIAL} &= -\frac{Largo}{2} + 4 * DistanciaEV - \frac{AreaEV}{2} \\ \rightarrow Y_{FINAL} &= Y_{INICIAL} + AreaEV \end{aligned}$$

❖ Electroválvula 6:

$$\begin{aligned} \rightarrow X_{INICIAL} &= -Offset - \frac{AreaEV}{2} \\ \rightarrow X_{FINAL} &= -X_{INICIAL} + AreaEV \\ \rightarrow Y_{INICIAL} &= -\frac{Largo}{2} + 5 * DistanciaEV - \frac{AreaEV}{2} \\ \rightarrow Y_{FINAL} &= Y_{INICIAL} + AreaEV \end{aligned}$$

❖ Electroválvula 7:

$$\begin{aligned} \rightarrow X_{INICIAL} &= -Offset - \frac{AreaEV}{2} \\ \rightarrow X_{FINAL} &= -X_{INICIAL} + AreaEV \\ \rightarrow Y_{INICIAL} &= -\frac{Largo}{2} + 6 * DistanciaEV - \frac{AreaEV}{2} \\ \rightarrow Y_{FINAL} &= Y_{INICIAL} + AreaEV \end{aligned}$$

❖ Electroválvula 8:

$$\begin{aligned} \rightarrow X_{INICIAL} &= -Offset - \frac{AreaEV}{2} \\ \rightarrow X_{FINAL} &= -X_{INICIAL} + AreaEV \\ \rightarrow Y_{INICIAL} &= -\frac{Largo}{2} + 7 * DistanciaEV - \frac{AreaEV}{2} \\ \rightarrow Y_{FINAL} &= Y_{INICIAL} + AreaEV \end{aligned}$$

❖ Electroválvula 9:

$$\begin{aligned} \rightarrow X_{INICIAL} &= -Offset - \frac{AreaEV}{2} \\ \rightarrow X_{FINAL} &= -X_{INICIAL} + AreaEV \\ \rightarrow Y_{INICIAL} &= -\frac{Largo}{2} + 8 * DistanciaEV - \frac{AreaEV}{2} \end{aligned}$$

$$\triangleright Y_{FINAL} = Y_{INICIAL} + AreaEV$$

3. Cálculo del área a abarcar en el ciclo

Se debe evaluar si la maleza, la cual se está estudiando se encuentra dentro del área en la que el robot puede atacar en este instante, es decir, el área que abarcará hasta la próxima actualización de los datos descriptos anteriormente como parámetros de entrada.

Esto se hace a partir de las posiciones de las electroválvulas calculadas anteriormente y una estimación de estas en el último instante previo a la lectura de nuevos datos.

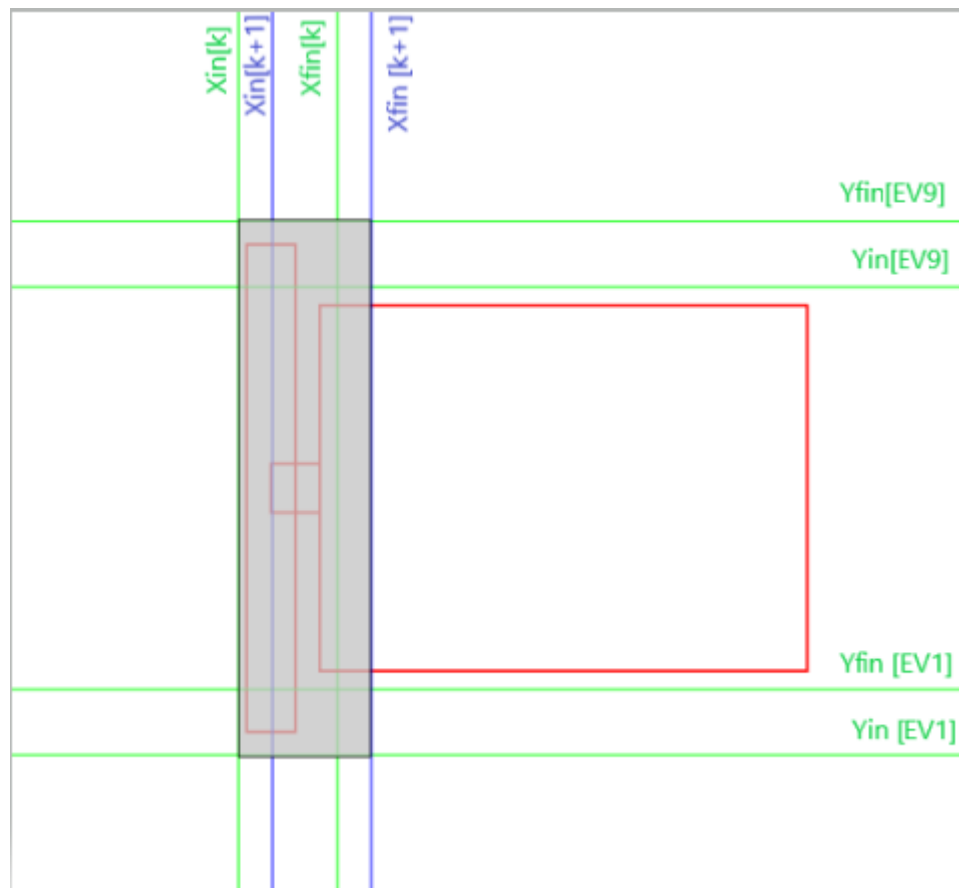


Figura 4.9: Delimitación de la zona de injerencia del conjunto de electroválvulas

El área queda determinada por las rectas $X_{in}[k]$ y $X_{fin}[k+1]$, siendo este salto de tiempo el ciclo con el cual se repetirá el proceso, y por las rectas dadas por Y_{in} de la electroválvula 1 e Y_{fin} de la electroválvula 9. Esta queda representada por la zona sombreada en la figura anterior.

Solo se pasará al siguiente punto en el caso de que la maleza a estudiar se encuentre dentro del área calculada. Esta condición se cumple cuando algún punto de la misma se encuentre en el área que cubrirá el robot, es decir, deberá cumplirse con al menos, alguna de las opciones mostradas en la siguiente figura.

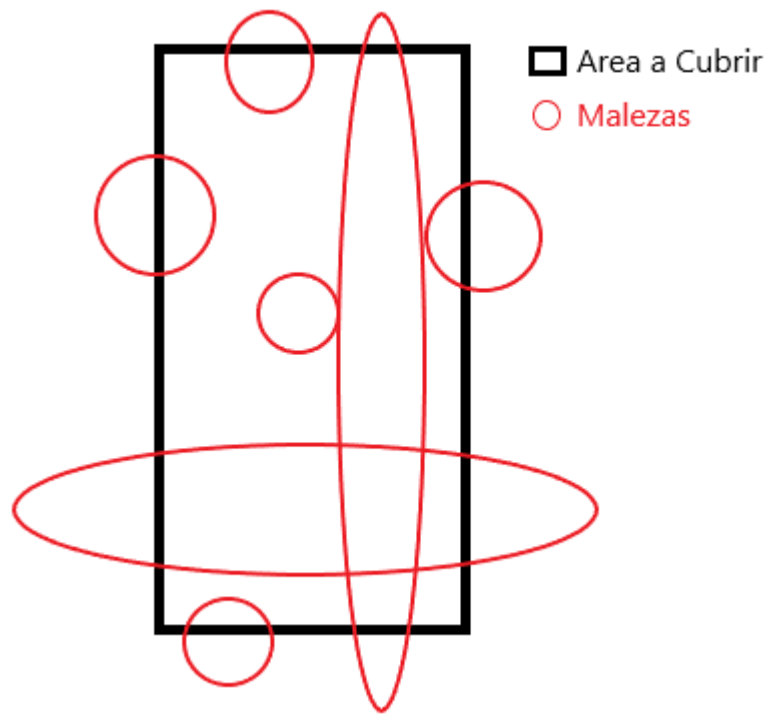


Figura 4.10: Casos de posibles malezas

Transformando estas opciones a ecuaciones matemáticas, se concluye que se podría dividir en ecuaciones, siendo estas (1) y (2) correspondientes al eje X y las ecuaciones (3) y (4) correspondientes al eje Y.

$$(1) \text{AreaEV.}X_{in} \leq (\text{Maleza.}X - \text{Maleza.}r) \&\& (\text{Maleza.}X - \text{Maleza.}r) \leq \text{AreaEV.}X_{fin}$$

$$(2) \text{AreaEV.}X_{in} \leq (\text{Maleza.}X + \text{Maleza.}r) \&\& (\text{Maleza.}X - \text{Maleza.}r) < \text{AreaEV.}X_{in}$$

$$(3) \text{AreaEV.}Y_{in} \leq (\text{Maleza.}Y - \text{Maleza.}r) \&\& (\text{Maleza.}Y - \text{Maleza.}r) \leq \text{AreaEV.}Y_{fin}$$

$$(4) \text{AreaEV.}Y_{in} \leq (\text{Maleza.}Y + \text{Maleza.}r) \&\& (\text{Maleza.}Y - \text{Maleza.}r) < \text{AreaEV.}Y_{in}$$

La ecuación final correspondería a la unión de estas cuatro ecuaciones de la siguiente forma

$$[(1) \text{ or } (2)] \text{ and } [(3) \text{ or } (4)]$$

En caso contrario se descartará esta maleza y se repetirá la secuencia con la siguiente maleza disponible.

4. Cálculo de las electroválvulas de inicio y fin

Dada la información de una maleza que se encuentre dentro del área activa del robot, aquí se determinará que electroválvula deberá actuar.

Este punto se determinará evaluando el eje Y - siempre hablando de las coordenadas referenciadas al sistema de coordenadas del robot-.

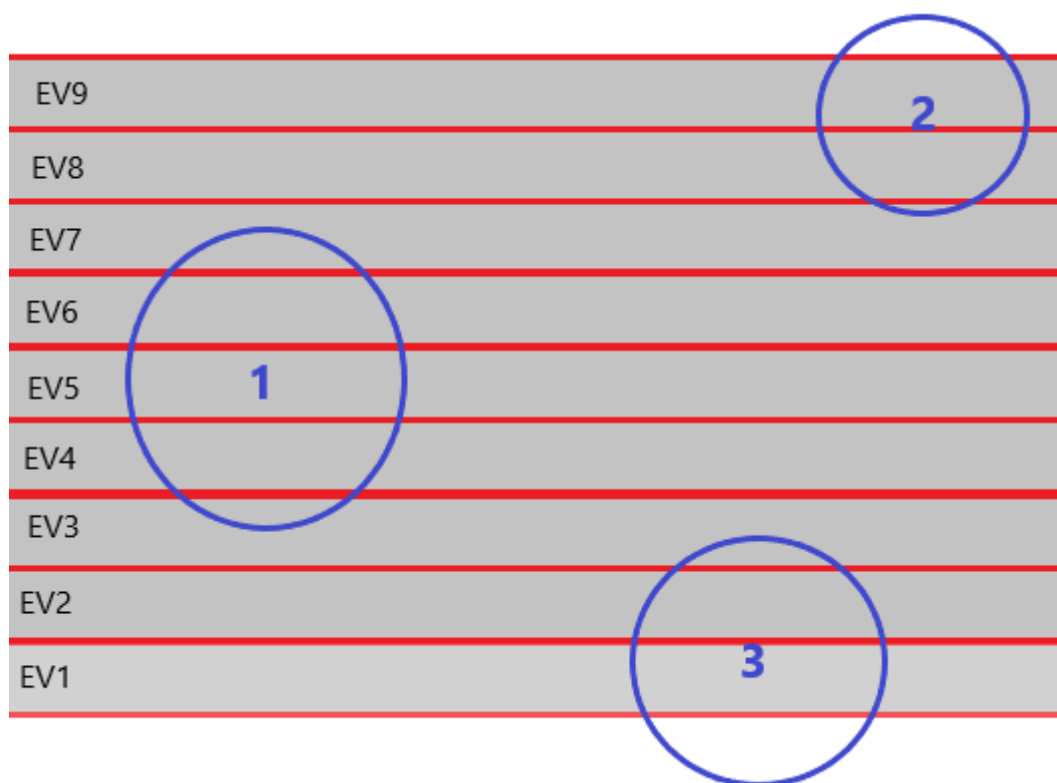


Figura 4.11: Ejemplo de cobertura de malezas por las válvulas

Se determinará la electroválvula inicial, siempre y cuando cumpla alguna de las siguientes ecuaciones

$$(1) \text{ PosEV.Yin}[i] \leq (\text{Maleza.Y} - \text{Maleza.r}) \text{ and } (\text{Maleza.Y} - \text{Maleza.r}) < \text{PosEV.Yfin}[i]$$

$$(2) (\text{Maleza.Y} - \text{Maleza.r}) < \text{PosEV.Yin}[1] \text{ and } \text{PosEV.Yin}[1] \leq (\text{Maleza.Y} + \text{Maleza.r})$$

- La primera electroválvula por activar será aquella donde el comienzo de la maleza, en el eje Y se encuentre entre los valores Yin e Yfin de la misma.
En la figura, la electroválvula numero 3 sería la electroválvula inicial para la maleza 1. Esto se obtiene a partir de la ecuación (1).
- En otro caso, la primera electroválvula se considerará como la inicial en el caso en que el inicio de la maleza sea menor que el inicio de la primera electroválvula, y que el final de la maleza sea mayor al inicio de la electroválvula. Este es el caso representado por la ecuación (2).
En la figura, este caso está representado por la maleza 3, tomándose la electroválvula 1 como la inicial

Para el caso de la electroválvula final, se determinará en el caso de que cumpla las siguientes ecuaciones.

$$(3) \text{ PosEV.Yin}[i] \leq (\text{Maleza.Y} + \text{Maleza.r}) \text{ and } (\text{Maleza.Y} + \text{Maleza.r}) < \text{PosEV.Yfin}[i]$$

$$(4) (\text{Maleza.Y} - \text{Maleza.r}) \leq \text{PosEV.Yfin}[9] \text{ and } \text{PosEV.Yfin}[9] \leq (\text{Maleza.Y} + \text{Maleza.r})$$

- La electroválvula final por activar será aquella donde el final de la maleza, en el eje Y se encuentre entre los valores Yin e Yfin de la misma.
En la figura, la electroválvula numero 7 sería la electroválvula final para la maleza 1. Esto se obtiene a partir de la ecuación (3).
- En otro caso, la última electroválvula se considerará como la final en el caso en que el inicio de la maleza sea menor que el final de la última electroválvula, y que el final de la maleza sea mayor al final de la electroválvula número 9. Este es el caso representado por la ecuación (4).
En la figura, este caso está representado por la maleza 2, tomándose la electroválvula 9 como la final

5. Generación de código numérico y concatenación.

Una vez obtenidas las electroválvulas de inicio y fin, se generará el código numérico, colocando un '1' en las posiciones las cuales se encuentren entre los valores de las electroválvulas inicial y final, incluyendo las mismas.

Es decir, en el caso que el paso anterior devuelva que la electroválvula de inicio es 2 y la electroválvula de fin es 5, el código resultante será:

001111000

Este código se concatenará mediante una función 'OR' con las electroválvulas encendidas para otras malezas dentro del mismo ciclo de acción.

5. Equipamiento Utilizado

5.1 Electroválvulas

Las electroválvulas o válvulas solenoides son dispositivos diseñados para controlar el flujo (On-Off) de determinado fluido. Están diseñadas para poder utilizarse con agua, gas, aire, vapor, entre otros. Estas válvulas pueden ser de dos hasta cinco vías. Pueden estar fabricadas en latón, acero inoxidable o PVC, dependiendo del fluido en el que se vayan a utilizar.

En las válvulas de 2 vías, normalmente se utilizan las que funcionan con tres modalidades diferentes, dependiendo del uso que están destinadas a operar; pueden ser de acción directa, acción indirecta y acción mixta o combinada. Además, cada una de estas categorías puede ser Normalmente Cerrada (N.C.) o Normalmente Abierta (N.A.), dependiendo de la función que va a realizar ya sea que esté cerrada y cuando reciba una señal en el solenoide abra durante unos segundos, o que esté abierta y cuando reciba la señal el solenoide corte el flujo.

5.1.1 Acción directa

El comando eléctrico acciona directamente la apertura o cierre de la válvula, por medio de un embolo.

La diferencia entre la válvula N.C. y la válvula N.A. de acción directa es que, cuando la válvula N.C. no está energizada el embolo permanece en una posición que bloquea el orificio de tal manera que impide el flujo. Cuando se energiza la bobina el embolo es magnetizado de tal manera que se desbloquea el orificio y de esta manera fluye el fluido. En la válvula N.A., cuando la bobina no está energizada mediante la acción de un resorte el embolo se mantiene en tal posición que siempre está abierta y cuando se energiza la bobina la acción es hacia abajo empujando el resorte haciendo que cierre el orificio e impida que fluya el fluido.

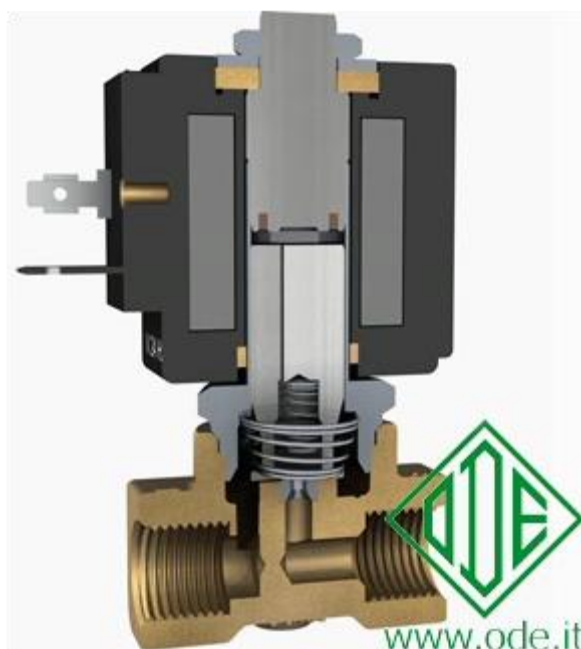


Figura 5.1: Válvula de acción directa, Normal cerrada

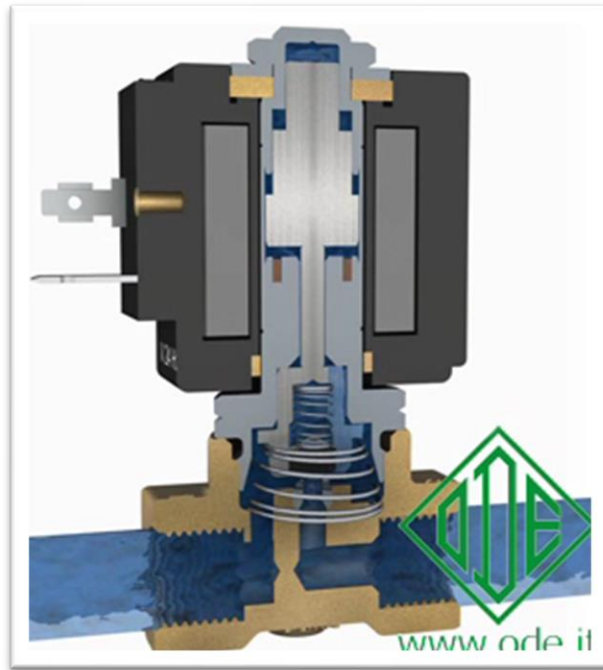


Figura 5.2: Válvula de acción directa. Normal abierta

5.1.2 Acción indirecta

La característica principal de la válvula del tipo acción indirecta es que cuando recibe el comando eléctrico se acciona el embolo el cual permite a su vez como segunda acción, o acción indirecta, que el diafragma principal se abra o se cierre, en una acción indirecta. Esta serie de válvulas necesita una presión mínima para poder funcionar correctamente. También en esta serie de comando indirecto tenemos válvulas normalmente cerradas y válvulas normalmente abiertas.

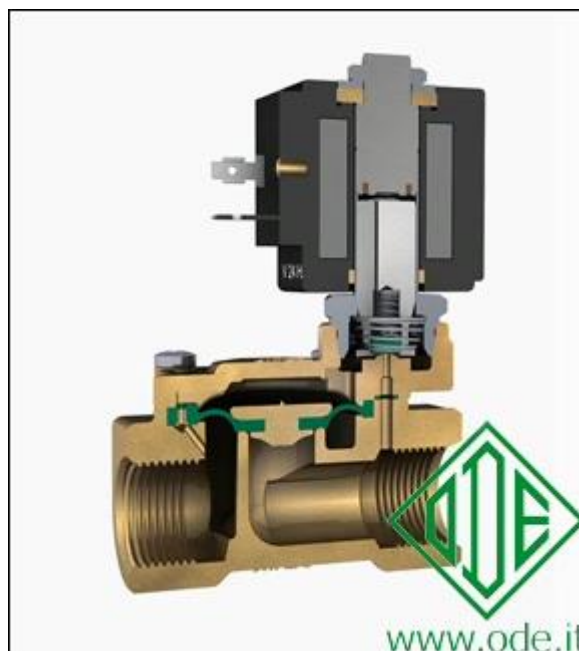


Figura 5.3: Válvula de acción indirecta

5.1.3 Acción mixta

En las válvulas de Acción Mixta o Combinada una característica es que no requieren una presión mínima como las de acción indirecta. Estas válvulas al igual que las de acción indirecta el comando de abertura se hace en 2 tiempos, primero se vacía la presión superior del diafragma grande y después, segunda acción, la presión de abajo del diafragma lo empuja para que se abra. Además, el embolo está sujetado por medio de un resorte al diafragma grande y este resorte acelera la acción de la presión de abajo hacia arriba para abrir el mismo diafragma, esta es la segunda etapa de apertura.

Estas válvulas de acción mixta pueden ser ya sea normalmente abiertas o normalmente cerradas.



Figura 5.4: Válvula de acción mixta

En las válvulas de acción mixta y de acción indirecta, los diafragmas que se utilizan dependen del material que vaya a fluir a través de ellas. Los diafragmas pueden ser de buna, viton o teflón debido a que cada uno de estos diafragmas tiene ciertas características. Por ejemplo, el diafragma de buna soporta temperaturas de -10 a +90 C y es recomendado para agua, aire, gas inerte. En el caso del viton soporta temperaturas más altas (-10 a +140 C) y se recomienda para aceite ligero, gasolina, diesel. En el caso del teflón soporta temperaturas de -10 a +180 C este se recomienda para vapor debido a la temperatura que puede soportar y que el teflón es más resistente.

Las electroválvulas que vamos a utilizar son de tipo acción directa, corresponden a la empresa Intor específicamente el modelo 2P025G02 la cual se puede ver en la siguiente imagen.



Figura 5.5: Electroválvula a utilizar

Las características técnicas de la misma son las siguientes:

- Vías/Posiciones: 2/2
- Posiciones Estables: Monoestable.
- Roscas Principales: 1/4 BSP.
- Función: Normal Cerrado (NC).
- Voltaje: DC 12V.
- Fluido: Aire filtrado 40 m³, con o sin lubricación
- Presión de trabajo : 0 a 10 bar (0 a 145 PSI).
- Tamaño de orificio: 2.5mm (CV 0.23)
- Consumo: 6.0 W
- Tolerancia de voltaje: $\pm 10\%$
- Grado de protección: IP65
- Material del Cuerpo: Plástico

A estas electroválvulas se le agregará una terminación plástica con el objetivo de que la inyección de herbicida sea localizada con una zona de cobertura deseada de 30°.[1]

5.2 Microcontrolador (placa de desarrollo FRDM-KL46Z)

Se utilizó una placa de desarrollo “KL46Z”, siendo esta una plataforma construida bajo la arquitectura ARM Cortex M0+ con un microcontrolador MKL46Z256VLL4MCU de la serie Kinetis L.

Este microcontrolador posee las siguientes características principales:

- Frecuencia máxima de reloj: 48 MHz
- Memoria flash: 256 KB
- Memoria RAM: 32 KB SRAM

Además, posee las siguientes ventajas generales:

- Nueve modos de baja potencia para proporcionar optimización de energía según los requisitos de la aplicación.
- Consumo de energía hasta 50 μ A/MHz en modo VLPR (Very low power run)

- Consumo de energía estático hasta 2 μA con retención de estado completo y activación de 4.5 μs .
- La arquitectura de ahorro de energía está optimizada para bajo consumo de energía con tecnología TFS de 90 nm, técnicas de activación de reloj y alimentación y controlador de memoria flash de estado de espera cero

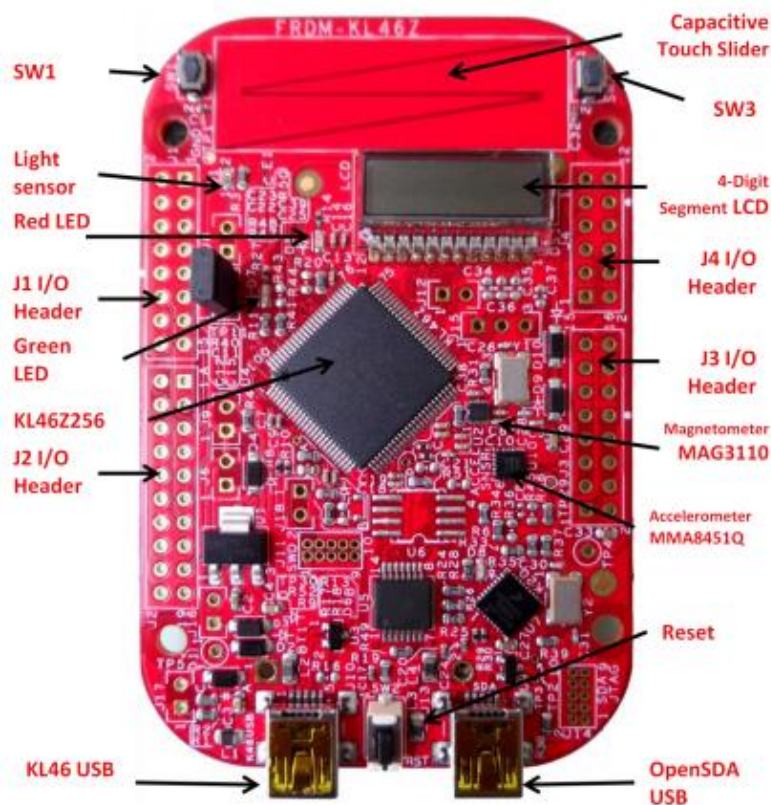


Figura 5.6: Principal componente placa de desarrollo KL46Z

5.2.1 Características principales

Performance.

- ✓ 48 MHz ARM® Cortex®-M0+ core.

Memorias e Interfaces de Memoria.

- ✓ Hasta 256 KB memoria flash programable.
- ✓ Hasta 32 KB SRAM

Periféricos del Sistema.

- ✓ Nueve modos de baja potencia para proporcionar optimización de energía.
- ✓ Controlador DMA de 4 canales, soporta hasta 63 fuentes de request.
- ✓ Unidad de despertador de low leakage.
- ✓ Interfaz de depuración SWD y Micro Trace Buffer.
- ✓ Motor de manipulación de bit (BME)

- ✓ COP Software watchdog.

Interfaces de Comunicación

- ✓ Controlador On-the-Go USB de velocidad alta/baja con transceptor en chip y regulador de 5V a 3.3V.
- ✓ Dos módulos SPI de 16 bits.
- ✓ Módulo I2S (SAI)
- ✓ Un módulo UART de bajo consumo
- ✓ Dos módulos UART
- ✓ Dos módulos I2C

Reloj.

- ✓ Oscilador de cristal de 32kHz a 40kHz o de 3MHz a 32MHz.
- ✓ Fuente de reloj multipropósito.

Características de Operación.

- ✓ Rango de voltaje de 1.71 a 3.6V
- ✓ Rango de voltaje de escritura flash 1.71 a 3.6 V.
- ✓ Rango de temperatura -40 a 105 °C

Módulos Analógicos.

- ✓ ADC de 16-bit (SAR)
- ✓ DAC de 12-bit
- ✓ Comparador analógico (CMP) que contiene un DAC de 6-bits y una entrada de referencia programable.

Módulos de Seguridad e integración.

- ✓ Número de identificación único de 80 bits por chip.

Temporizadores.

- ✓ PMW/Temporizador de 6 canales (TPM).
- ✓ Dos módulos PWM/Temporizadores de 2 canales.
- ✓ Temporizador de interrupción periódica.
- ✓ Temporizador de bajo consumo de 16-bits (LPTMR)
- ✓ Reloj de tiempo real.

5.2.2 Diagrama de entrada/salida

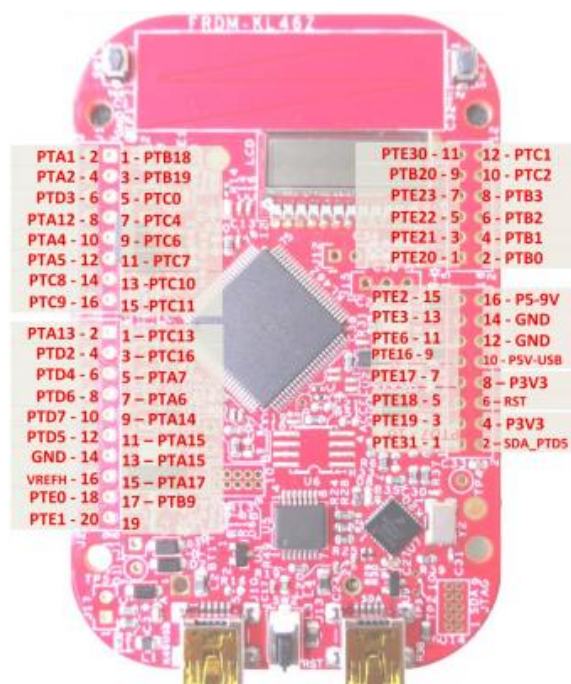


Figura 5.7: Puertos entrada/salida de la placa de desarrollo KL46Z

Por otra parte, en la siguiente tabla, se muestra las entradas salidas del microprocesador necesarias para solventar el control de las electroválvulas correspondientes al sistema de pulverización.

E/S	Tipo	Número	Descripción
Salidas	Digitales	9	Pines para comandar la activación o no de electroválvulas
Salida	Digitales	1	Pin de alimentación de 3.3V
Salida	Digitales	1	GND
Entrada	Serie	1	Comunicación con PC (UART por USB)

Tabla 5.1: Entradas/salidas a utilizar en el proyecto

5.3 Computadora a utilizar

Tanto para el procesamiento digital de imágenes para la detección de la maleza como para la toma de decisiones de la activación de electroválvulas se utilizará la computadora principal de robot desmalezador, una MINI-PC Intel® NUC Kit NUC6CAYH.



Figura 5.8: Mini PC NUC6CAYH.

La computadora NUC6CAYH es de tamaño reducido (aproximadamente 11x6 cm), posee disipador en lugar de cooler (el cual podría traer aparejados problemas con la tierra y el polvo al trabajar en ambientes como en el campo) y dispone de cuatro puertos USB, los cuales serán utilizados en la comunicación con el microcontrolador. Puede alimentarse entre 12 y 19 voltios, haciéndola viable de ser montada sobre el robot desmalezador que posee alimentaciones de 5, 12 y 36 V.

Por último, posee las siguientes características principales:

- ✓ Procesador: Intel® Celeron® J3455
- ✓ Velocidad de CPU: 1,50 GHz
- ✓ Velocidad Turbo: 2,30 GHz
- ✓ Núcleo: 4
- ✓ TDP máx.: 10 W
- ✓ Memoria: 2 sockets DDR3L 1866MHz SODIMM, 1.35V
- ✓ Cantidad máxima admitida: 8Gb
- ✓ Arquitectura: Dual Channel (Doble canal)
- ✓ Conectividad Interna:
 - 2x internal USB 2.0 via header.
- ✓ Conectividad Externa
 - 2x puertos frontales USB 3.0
 - 2x puertos traseros USB 3.0
 - Lector de tarjetas SDXC
 - Jack audio
- ✓ Micro SDXC* slot with UHS-I support
- ✓ One SATA3 port for connection to 2.5* HDD or SSD (up to 9.5 mm thickness)
- ✓ 64 Mb Flash EEPROM with Intel® Platform Innovation Framework for EFI Plug and Play.

5.4 Softwares utilizados para el desarrollo

5.4.1 Proteus 8 Profesional

Proteus es una aplicación para la ejecución de proyectos de construcción de equipos electrónicos en todas sus etapas: diseño del esquema electrónico, programación del software, construcción de la placa de circuito impreso, simulación de todo el conjunto, depuración de errores, documentación y construcción.

Este permite el desarrollo y simulación del equipo electrónico, lo cual permite encontrar errores y problemas en la etapa previa a la construcción de prototipos.

Con Proteus las fases de prueba no suponen la necesidad de volver a construir nuevos prototipos, con el ahorro de costos y tiempo que ello supone.

Por lo tanto, se utilizó este software para la ejecución completa de la placa para adaptar las señales salientes del microcontrolador para que puedan activar las electroválvulas correspondientes, es decir la etapa de desarrollo del esquema electrónico, simulación de la misma y por último el diseño final de la placa de circuito impreso

Principales características:

- La aplicación ISIS permite generar circuitos reales, y comprobar su funcionamiento en un PCB (*printed circuit board*).
- Entorno de diseño gráfico de esquemas electrónicos fácil de utilizar y con efectivas herramientas.
- Entorno de simulación con la tecnología exclusiva de Proteus de modelación de sistemas virtuales (VSM).
- Herramienta ARES para el enrutado, ubicación y edición de componentes, utilizado para la fabricación de placas de circuito impreso.
- Interfaz intuitivo y atractivo estandarizado para todos los componentes de Proteus.

5.4.2 Mcuxpresso IDE

MCUXpresso IDE versión es una plataforma de desarrollo de microcontroladores (MCU) de bajo costo dentro de la familia de NXP. Proporciona una solución integral que permite desarrollar aplicaciones integradas desde la evaluación inicial hasta la producción final.[3]

MCUXpresso IDE se basa en Eclipse IDE e incluye las herramientas dentro del estándar de la industria ARM GNU. Ofrece un entorno de desarrollo de código ilimitado y fácil de usar para microcontroladores NXP basados en núcleos Cortex-M (LPC, Kinetis e iMX RT). El IDE combina lo mejor de los IDE ampliamente populares de LPCXpresso y Kinetis Design Studio, que proporcionan una plataforma para todos los microcontroladores NXP Cortex-M.

Proporciona una interfaz intuitiva y potente con evaluación de perfiles y medición de potencia, en placas compatibles, bibliotecas y herramientas de integración GNU, depurador multinúcleo, funcionalidad de seguimiento y más.

Las conexiones de depuración IDE de MCUXpresso admiten Freedom, Tower, EVK, LPCXpresso y placas de desarrollo personalizadas con código abierto estandarizado en la industria, sondas de depuración que incluyen LPC-Link2, PEmicro y SEGGER.

Principales características:

- Un tamaño de código ilimitado gratuito, IDE fácil de usar.
- Edición, compilación y edición avanzadas con colores de sintaxis, vistas de depuración específicas de MCU, seguimiento de código y creación de perfiles.
- Utilice la herramienta de selección de SDK incorporada o arrastre y suelte paquetes prediseñados creados con SDK Builder.

- Herramientas de configuración integradas, incluidos pines, relojes, periféricos y herramientas TEE.
- Herramientas GNU estándar en la industria con una selección de bibliotecas: biblioteca C optimizada o la biblioteca GNU Newlib / Nano estándar con soporte para semihosting.
- Seguimiento y creación de perfiles SWO (a través de sondas SEGGER J-Link, PE Micro y LPC-Link2).
- Seguimiento de instrucciones (para dispositivos habilitados para ETB / MTB).
- Herramienta de programación Flash GUI integrada.
- Ricas capacidades de visualización para registros periféricos, variables globales, variables en vivo textuales / gráficas
- Nuevos Project Wizards y fácil importación de ejemplos de SDK.
- Vista de uso de montón y pila para realizar un seguimiento del uso de pila y montón de proyectos bare metal.
- La vista de información de la imagen brinda el uso general detallado de la memoria, el contenido de las regiones de memoria y los gráficos de llamadas estáticas, incluida la información de uso de la pila.
- Soporte para depuración consciente de FreeRTOS.
- Altamente extensible utilizando complementos de Eclipse ampliamente disponibles.
- Sistemas operativos de host:
- Microsoft ® de Windows ® (sólo 64 bits) 8/10
- Ubuntu Linux ® 16.04 LTS / 18.04 LTS (solo sistema operativo host de 64 bits)
- macOS 10.11 -> 10.14

5.4.3 Codeblocks IDE

Code Blocks es una opción muy buena para programar en C. Constituye un entorno integrado de desarrollo multiplataforma y de código abierto que soporta la utilización de múltiples compiladores, entre ellos: GCC (MingW / GNU GCC), MSVC, Digital Mars, Borland C 5.5 u Open Watcom. El compilador que viene por defecto con este paquete de Code Blocks es MinGW.

Las funciones de Code Blocks se pueden ampliar mediante la utilización de plugins, posee un rápido sistema de construcción personalizado que no necesita la presencia de makefiles y que permite realizar construcciones paralelas en equipos con una CPU de varios núcleos.

Por último, Code Blocks colorea la sintaxis del código fuente, incluye la función de agrupación de código o 'code folding' para los lenguajes C y XML, posee una interfaz con pestañas, cuenta con un explorador de clases y una función para completar el código fuente.[4]

Principales características:

- **Multiplataforma.** Se ejecuta en Linux, Mac, Windows (usa wxWidgets).
- Escrito en C ++. No se necesitan idiomas interpretados ni libs propietarias.
- Extensible a través de complementos

Compilador

- **Compatibilidad con varios compiladores:**
 - GCC (MingW / GNU GCC) por defecto
 - MSVC ++
 - Clang
 - Borland C ++ 5.5

- Open Watcom
- Sistema de compilación personalizado **muy rápido** (no se necesitan archivos make)
- Soporte para **compilaciones paralelas** (utilizando los núcleos adicionales de su CPU)
- Proyectos de objetivos múltiples
- Espacios de trabajo para combinar múltiples proyectos.
- Dependencias entre proyectos dentro del espacio de trabajo

Debugger:

- Interfaces GNU GDB
- También es compatible con MS CDB (no con todas las funciones)
- **Soporte completo de puntos de interrupción:**
 - Puntos de interrupción del código
 - Puntos de corte de datos (lectura, escritura y lectura / escritura)
 - Condiciones de punto de interrupción (se rompen solo cuando una expresión es verdadera)
 - El punto de interrupción ignora los recuentos (se rompe solo después de cierto número de golpes)
- Mostrar símbolos y argumentos de funciones locales
- Relojes definidos por el usuario (soporte para ver tipos definidos por el usuario a través de secuencias de comandos)
- Pila de llamadas
- Desmontaje
- Volcado de memoria personalizada
- Cambiar entre hilos
- Ver registros de CPU

Interfaz:

- Resaltado de sintaxis, personalizable y extensible
- Código plegable para C, C ++, Fortran, XML y muchos más archivos.
- Interfaz con pestañas
- Completar código
- Explorador de clases
- Sangría inteligente
- Intercambio de una tecla entre archivos .h y .c / .cpp
- Abra la lista de archivos para cambiar rápidamente entre archivos (opcional)
- "Herramientas" personalizables externas
- Gestión de listas de tareas con diferentes usuarios

5.4.4 Vmware

Para poder trabajar sobre una máquina que contenga el mismo sistema operativo que la computadora y desarrollar el código en un ambiente similar, se generó una máquina virtual que tenga como sistema operativo LINUX.

Para esto se utilizó el software VMware Workstation Pro.[5]

VMware Workstation Pro crea máquinas virtuales seguras, totalmente aisladas, que encapsulan un sistema operativo y sus aplicaciones. La capa de virtualización de VMware asigna los recursos de hardware físicos a los recursos de la máquina virtual, por lo que cada máquina virtual cuenta con una CPU, una memoria, unos discos y unos dispositivos de E/S propios, y equivale en su totalidad a una máquina x86 convencional. VMware Workstation Pro se instala en el sistema operativo host y ofrece una amplia compatibilidad de hardware al heredar del host la compatibilidad con los dispositivos.

Cualquier aplicación que funcione en un PC convencional se ejecutará en una máquina virtual de VMware Workstation Pro. VMware Workstation Pro es el equivalente de un PC completo, con todos los dispositivos y redes, donde cada máquina virtual tiene sus propios recursos: procesador, memoria, discos, dispositivos de E/S, etc. Esto le permite ejecutar cualquier aplicación que funcione en los sistemas operativos invitados compatibles, entre las que se incluyen Microsoft Office, Adobe Photoshop, Apache Web Server, Microsoft Visual Studio, depuradores de núcleo, cortafuegos, software VPN, etc.

La ventaja que se obtiene de este modo es desarrollar el código en un ambiente similar, con correspondencia en su sistema operativo, al equipo donde finalmente correrá y trabajará el software.

6. Desarrollo

6.1 Diagrama de bloques

En la siguiente imagen se puede apreciar la interconexión de las distintas etapas y componentes encontrados en el proyecto.

Aquí se encuentra la computadora principal (NUC) la cual tendrá la información recolectada a partir del procesamiento digital de imágenes para la detección de malezas.

Esta informará al microcontrolador mediante una comunicación serie (UART) el id correspondiente a la válvula a activar en el ciclo de ejecución actual.

Luego el microcontrolador procesará estos datos y comandará los pines de salidas para activar o desactivar las electroválvulas.

Estas señales de salida del microcontrolador deberán adaptarse a la potencia que las electroválvulas requieren. Esta acción es realizada por la etapa de potencia, que posee un transistor que actuará como llave electrónica siendo comandada por la señal proveniente del microcontrolador.

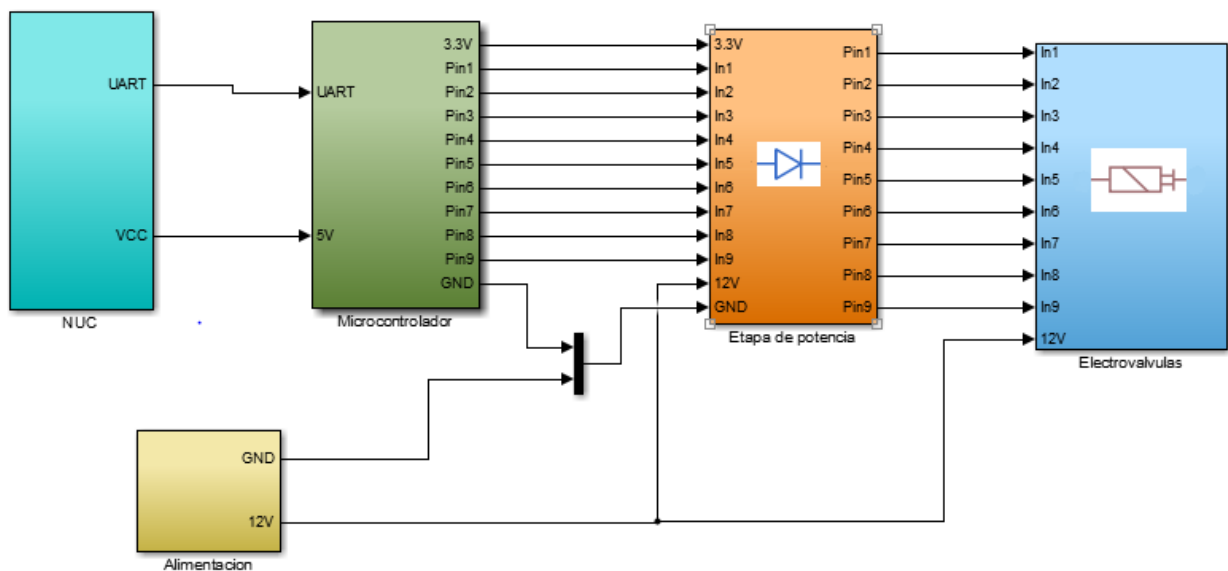


Figura 6.1: Diagrama de bloques del sistema

6.2 Disposición de los equipos en el carro

En la siguiente imagen se mostrará la distribución de los equipos que compondrán el sistema de pulverización dentro del chasis del robot desmalezador.

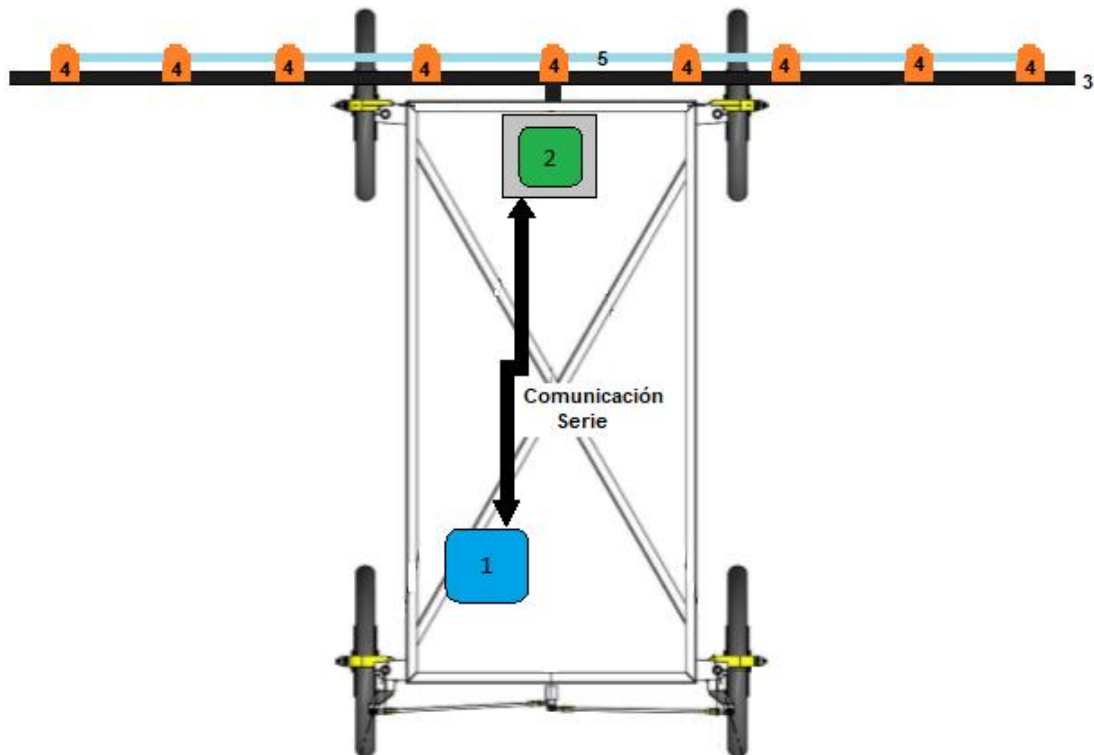


Figura 6.2: Esquema del sistema de control del circuito de la pulverización montado en el carro

1. Gabinete de PC: En este compartimiento se encontrará la computadora principal del robot desmalezador (NUC)
2. Gabinete de Control: Este gabinete tendrá los dispositivos encargados de la activación de las electroválvulas. Aquí se ubicará la placa de desarrollo FRDM-KL46Z que tiene un microcontrolador MK64FN1M0VLL12 y la placa encargada de adaptar las señales del microcontrolador a la potencia necesaria para la activación de electroválvulas. Actualmente en este gabinete se encuentra el sistema de tracción y dirección del prototipo.
3. Barral: Barral rebatible donde se encontrarán posicionadas las electroválvulas
4. Electroválvulas
5. Mangueras

6.3 Circuito de comando.

Para comandar las electroválvulas, es necesario un circuito de adaptación de la salida del microcontrolador a la potencia requerida para poder activar cada una de ellas.

En rasgos generales el microcontrolador proveerá una señal de 3.3V de tensión cuando desea prender una electroválvula, teniendo como limitación 5mA por pin de salida.

La electroválvula, para ser comandada, debe ser alimentada con una tensión continua de 12V y por cada una de ellas deberá circular 0,5A.

Para esto se utilizó el circuito mostrado, aquí se ve el comando de una electroválvula de forma individual.

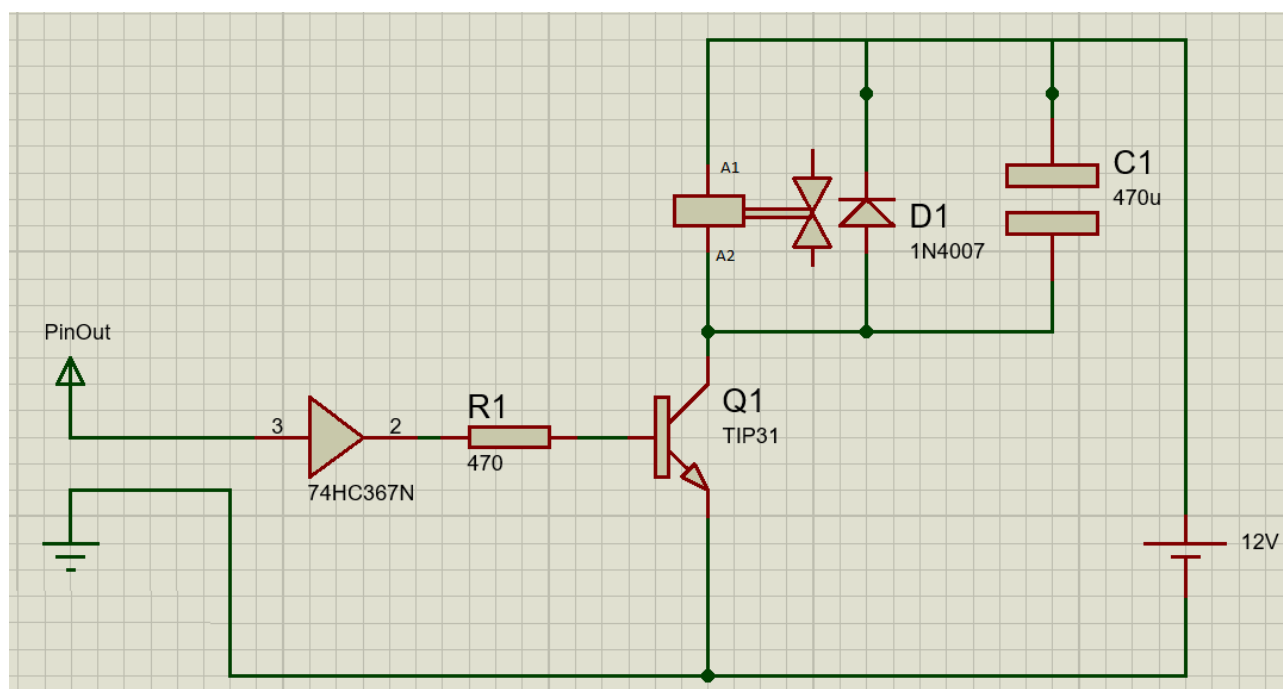


Figura 6.3: Circuito de adaptación de señales- Etapa de potencia.

Para comenzar la entrada correspondiente al circuito, PinOut del microcontrolador, se conectó a la etapa buffer, generando un desacople desde el circuito de potencia al circuito lógico, permitiendo de esta forma presentarle al microcontrolador una impedancia alta para no exigirle grandes corrientes. Además, provee un desacople para que problemas en la etapa de potencia no dañen al microcontrolador.

A la salida del buffer tendremos una copia de la tensión disponible en el pin del microcontrolador.

Luego, esta señal será el comando del transistor, quien actuará como llave electrónica. Se utilizó un transistor TIP29C, de tipo NPN de potencia, que admite valores de corriente I_C hasta 1A, y soporta tensiones de caída en corte V_{CEO} de hasta 100V, ambos valores que se adaptan al uso que le daremos. Cuando esta valga 3.3V el transistor entrará en saturación, generando de esta forma que los 12V caigan sobre la electroválvula y de esta forma será accionada.

Una vez que la señal del microcontrolador caiga, la base del transistor estará con un potencial de masa, haciendo que este se corte y la tensión de la fuente caiga sobre sí mismo ($V_{ce} = 12V$), es decir desactivando la electroválvula.

Como la electroválvula está compuesta mayoritariamente por un inductor, al cortar el transistor, deberá presentarle un camino para permitir la circulación de esta corriente. Este camino está dado por la red impuesta por el diodo y el capacitor.

En la siguiente imagen puede observarse el diagrama esquemático completo, formado por la totalidad de las electroválvulas, la etapa de alta impedancia que le provee el buffer, y el circuito de accionamiento que incluye los transistores, junto a las resistencias de base, y finalmente el circuito de supresión de corriente transitoria, formado por el diodo y capacitor.

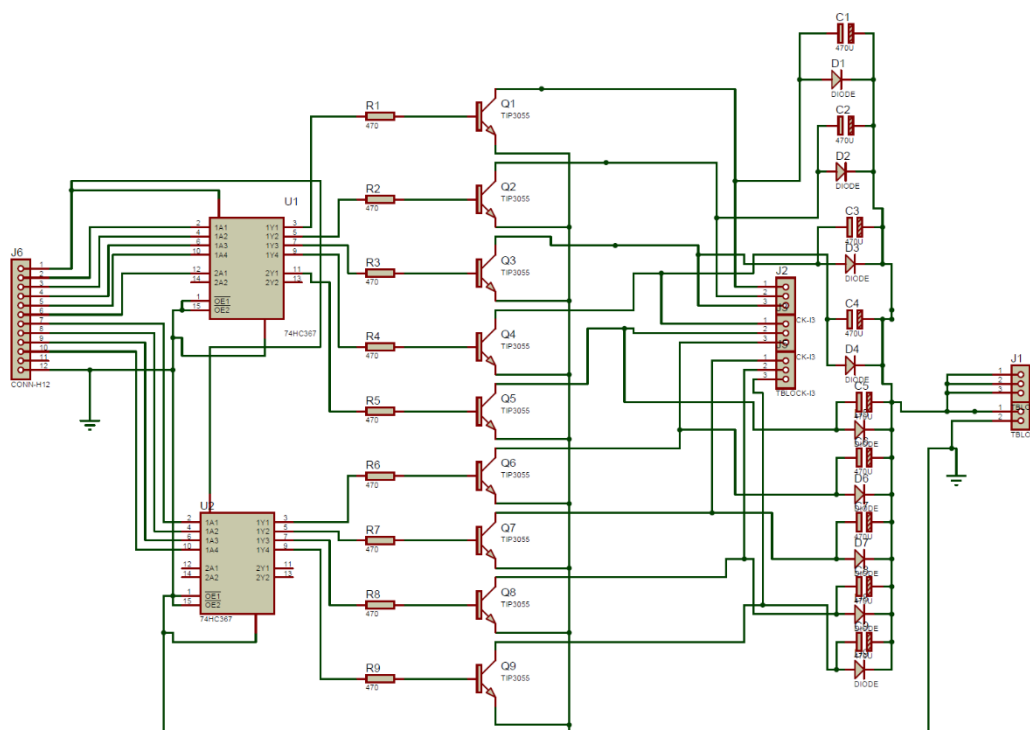


Figura 6.4: Esquemático del circuito de comando de electroválvulas completo

Al final, se agregan las hojas de datos de los componentes aquí utilizados.

6.4 Software del microcontrolador

En esta sección se describirá el funcionamiento del software en el microcontrolador. El mismo es el encargado de recibir una trama por su puerto serie (UART) y, dependiendo de los valores de esta, activar cada una de las salidas que se corresponden con cada EV.

En primer lugar, se definirán la correspondencia con las salidas utilizadas y cableadas con la placa.

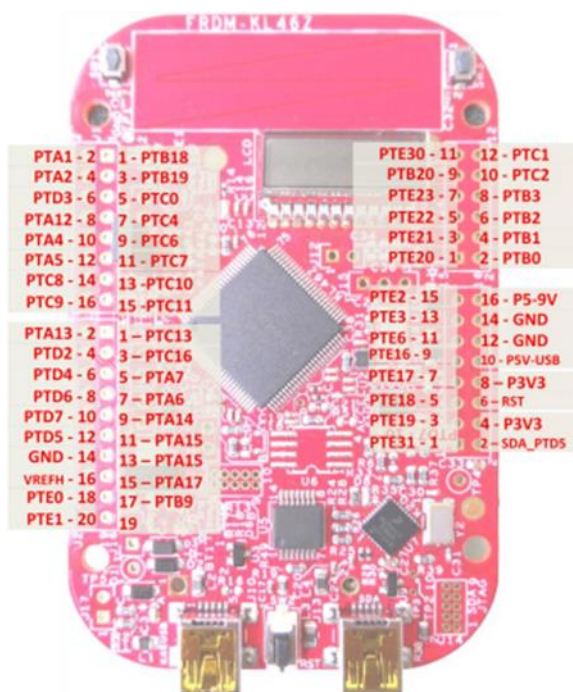


Figura 6.5: Diagrama de salidas del microcontrolador

PUERTO	GPIO	Pin	Uso
PORTE	GPIOE	22	Electroválvula 1
PORTE	GPIOE	21	Electroválvula 2
PORTE	GPIOE	20	Electroválvula 3
PORTE	GPIOE	2	Electroválvula 4
PORTE	GPIOE	3	Electroválvula 5
PORTE	GPIOE	6	Electroválvula 6
PORTE	GPIOE	16	Electroválvula 7
PORTE	GPIOE	17	Electroválvula 8
PORTE	GPIOE	18	Electroválvula 9
		4 o 8	Señal 3.3V
		14 o 12	GND

Tabla 6.1: Salidas utilizadas

6.4.1 Descripción del software

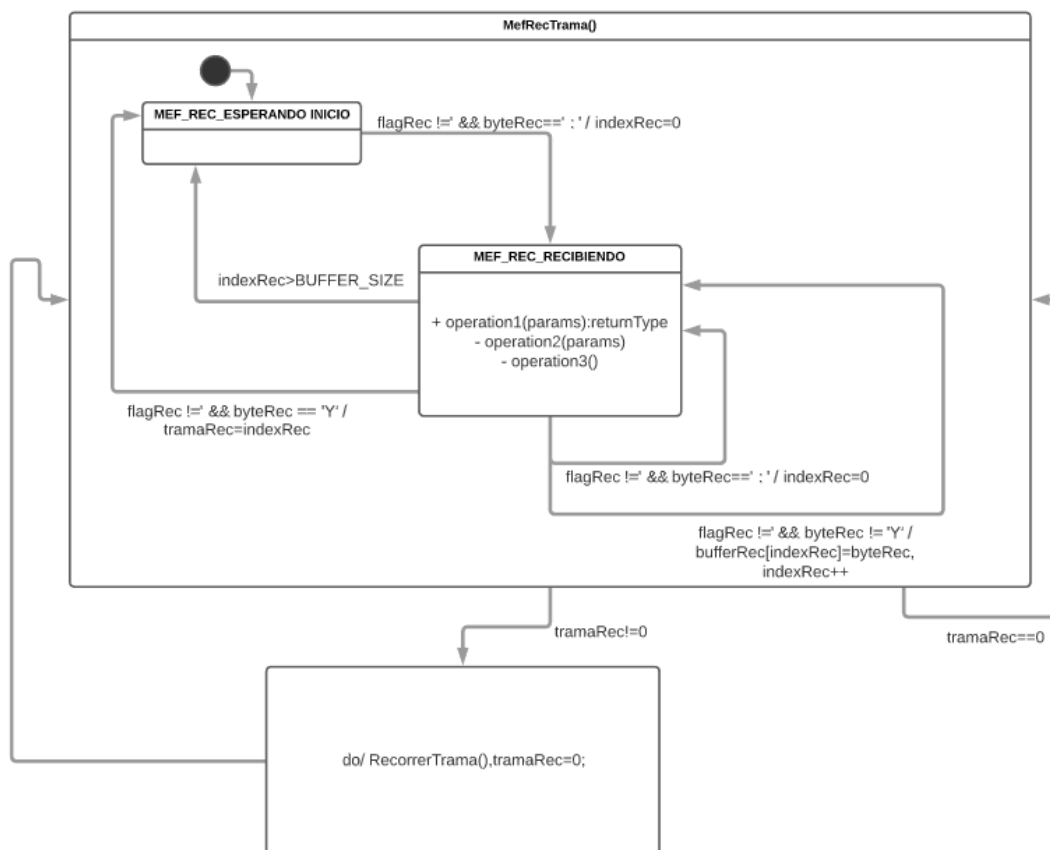
El sistema se puede diferenciar en dos etapas de comportamiento diferenciado. La primera es la encargada de recibir y validar la trama serie mediante el puerto correspondiente a la UART.

La trama es válida cuando cumple con todas las restricciones, las cuáles son:

- Inicio de la trama con el carácter ':'
- Fin de la trama con el carácter 'Y'
- Longitud de la trama menor al valor predefinido por BUFFER_SIZE.

Una vez validada, el sistema evoluciona a la etapa donde se evaluará el contenido de la misma. Aquí el código evaluará para cada posición de la trama, si el carácter que recibió es 'A' o 'E', donde apagará o encenderá respectivamente la salida correspondiente a la electroválvula asociada a la posición del carácter en la trama.

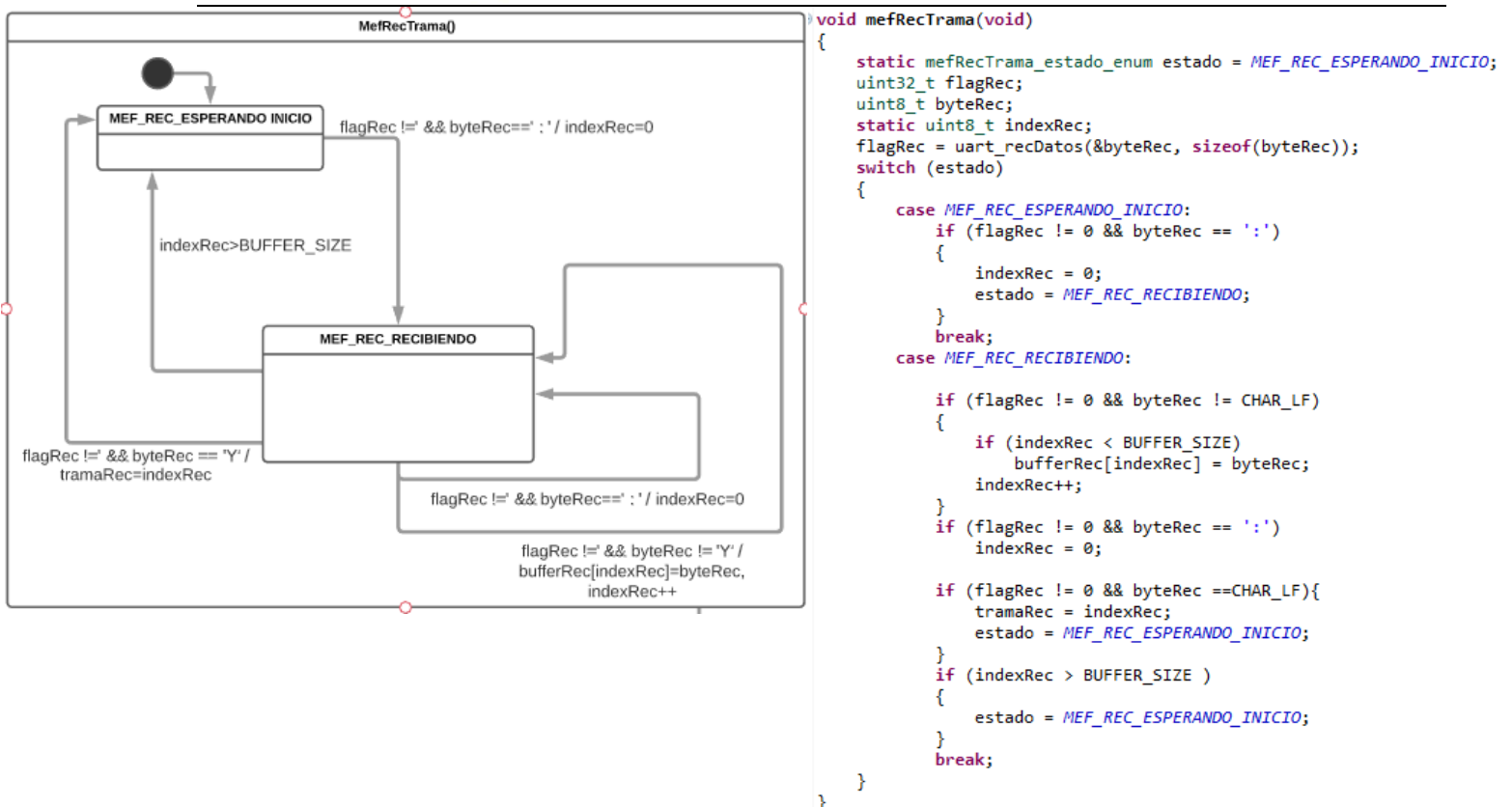
Como se mencionó anteriormente estas etapas poseen comportamiento diferente, por ende, se pueden modelizar por separado. La primera se modeliza en una MEF llamada MefRecTrama, mientras que la segunda se modeliza con el llamado a una función.



6.4.2 Solución código C

En este apartado se mostrará el código en C donde se implementa cada una de las etapas explicadas anteriormente.

Para comenzar, se codificará la MEF destinada a la recepción de la trama.



Aquí se puede observar como funciona esta MEF, la misma espera el inicio de la trama el cual se da con el carácter de inicio ‘:’.

Una vez encontrado este carácter, la MEF pasa al estado MEF_REC_RECIBIENDO en donde cada nuevo carácter, siempre que sea distinto del carácter final, lo irá acumulando dentro de bufferRec e irá sumando el índice, para tener la longitud de la trama recibida.

Si esta suma supera el tamaño maximo de BUFFER_SIZE, se elimina la trama y vuelve al estado inicial.

En caso de que dentro de la trama se vuelva a recibir el carácter de inicio, se elimina la trama anterior y se vuelve a reiniciar la secuencia.

Cuando se recibe el carácter de fin, el carácter CHAR_LF, se pasa la cantidad de caracteres leído al atributo tramaRec con el cual se pasará a la siguiente etapa de análisis de trama. La secuencia se reinicia para poder comenzar la lectura y captar una nueva trama.

Para el caso de la segunda etapa del analisis, se utilizan las siguientes funciones, donde se evalúa para cada carácter de la trama su valor, y con este toma la decision de activar o descativar la salida correspondiente a la electroválvula asociada a la posicion del carácter en la trama.

```
void IOValve(uint8_t charIdValv, int Accion)
{
    switch(Accion){
        case 'E':
            board_setLed(charIdValv,BOARD_LED_MSG_ON);
            break;
        case 'A':
            board_setLed(charIdValv,BOARD_LED_MSG_OFF);
            break;
    }
}

void RecorrerTrama(){
    for (int i=0;i<9;i++){
        IOValve(i,bufferRec[i]);
    }
}
```

6.5 Software de la computadora

A continuación, se documentará el software utilizado en la computadora central. Este es el software principal dentro de la solución, ya que, a partir de información recibida como la ubicación del robot y de las malezas, es el encargado de la toma de decisiones de que electroválvulas deberá abrir. Y como resultado final enviará una trama serie hacia el microcontrolador.

La codificación de este programa se basa en el análisis teórico- matemático desarrollado en la sección 4.3.

Como durante el desarrollo y testeo de este, no se encontraba listo el módulo de geolocalización ni el de detección de malezas, se desarrolló otro código el cual simula la función de estos, generando una trayectoria del robot y una disposición general de malezas en el mapa. En las siguientes secciones, se explicará cómo se llevó a cabo el software propio del proyecto, como así también el programa anteriormente comentado de simulación y la comunicación entre estos.

Para la ejecución de ambos programas simultáneamente, se utilizó la función fork().[6]

La llamada al sistema Fork se usa para crear un nuevo proceso, que se llama proceso hijo, que se ejecuta simultáneamente con el proceso que hace la llamada fork () (proceso padre). Después de crear un nuevo proceso hijo, ambos procesos ejecutarán la siguiente instrucción después de la llamada al sistema fork (). Un proceso secundario usa la misma PC (contador de programa), los mismos registros de CPU, los mismos archivos abiertos que se usan en el proceso primario.

No toma parámetros y devuelve un valor entero. A continuación, se muestran diferentes valores devueltos por fork ().

Valor negativo: la creación de un proceso hijo no tuvo éxito.

Cero: regresó al proceso hijo recién creado.

Valor positivo: devuelto al padre o al llamante. El valor contiene la identificación del proceso hijo recién creado.

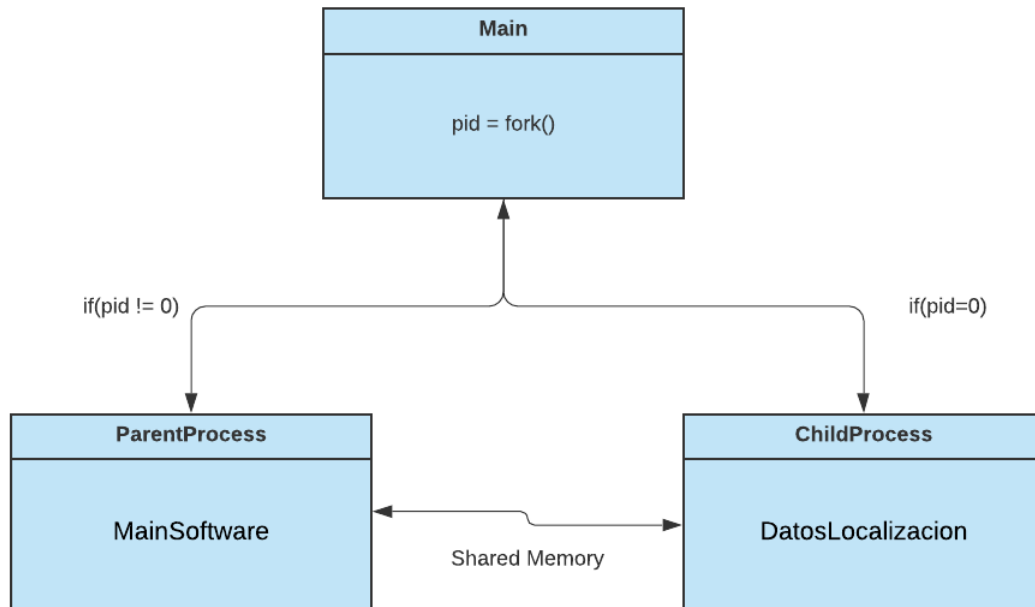


Figura 6.6: Ejecución simultanea de programas, utilizando la función fork()

Una vez generados ambos procesos, a partir del valor devuelto se bifurcara su ejecución hacia el script principal, el cual consiste en la toma de decisiones de que electroválvulas abrir (caso donde la función fork devuelve un valor positivo), como así también, en caso de que el valor devuelto sea cero, se ejecutará la llamada del código correspondiente a “DatosLocalizacion”, el cual se encarga de la creación de la información de Malezas en el campo y también de la trayectoria que el robot tomará.

La llamada a este código se realiza con la función exec().

La familia de funciones exec reemplaza el proceso en ejecución actual por un proceso nuevo. Se puede utilizar para ejecutar un programa en C utilizando otro programa en C. Viene bajo el archivo de encabezado unistd.h. Hay muchos miembros en la familia ejecutiva que se muestran a continuación con ejemplos.

- **execvp:** con este comando, el proceso hijo creado no tiene que ejecutar el mismo programa que el proceso padre. Las llamadas al sistema de tipo ejecutivo permiten que un proceso ejecute cualquier archivo de programa, que incluye un ejecutable binario o un script de shell.

Sintaxis:

int execvp (const char * archivo, char * const argv []);

archivo: apunta al nombre del archivo asociado con el archivo que se está ejecutando.

argv: es una matriz de punteros de carácter terminada en nulo.

- **execv:** Esto también es muy similar a la función execvp () en términos de sintaxis. La sintaxis de execv () es la que se muestra a continuación:

Sintaxis:

int execv (const char * ruta, char * const argv []);

ruta: debe apuntar a la ruta del archivo que se está ejecutando.

argv []: es una matriz de punteros de carácter terminada en nulo.

- **execlp y execl**: estos dos también tienen el mismo propósito, pero la sintaxis de ellos es un poco diferente, como se muestra a continuación:

Sintaxis:

```
int execlp (const char * archivo, const char * arg, ... / * (char *) NULL * /);
```

```
int execl (const char * ruta, const char * arg, ... / * (char *) NULL * /);
```

archivo: nombre de archivo asociado con el archivo que se está ejecutando
const char * arg y puntos suspensivos: describen una lista de uno o más punteros a cadenas terminadas en nulo que representan la lista de argumentos disponible para el programa ejecutado.

- **execvpe y execl**: estos dos también tienen el mismo propósito, pero la sintaxis de ellos es un poco diferente de todos los miembros anteriores de la familia exec. A continuación, se muestran las **sintaxis** de ambos:

Sintaxis:

```
int execvpe (const char * archivo, char * const argv [], char * const envp []);
```

```
int execl (const char * ruta, const char * arg, ... / *, (char *) NULL,
```

```
char * const envp [] * /);
```

La sintaxis mostrada arriba tiene un argumento diferente de todos los miembros exec anteriores, es decir,

char * const envp []: permite que el llamador especifique el entorno del programa ejecutado a través del argumento envp.

envp: este argumento es una matriz de punteros a cadenas terminadas en nulo y debe terminar con un puntero nulo. Las otras funciones toman el entorno para la nueva imagen de proceso de la variable externa entorno en el proceso de llamada.

Comunicación entre los procesos

La comunicación entre procesos a través de la memoria compartida es un concepto en el que dos o más procesos pueden acceder a la memoria común. Y la comunicación se realiza a través de esta memoria compartida donde los cambios realizados por un proceso pueden ser vistos por otro proceso.

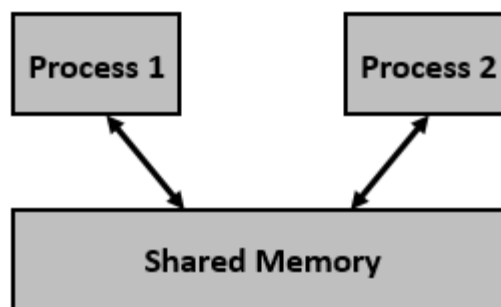


Figura 6.7: Esquema de memoria compartida

Para reiterar, cada proceso tiene su propio espacio de direcciones, si algún proceso quiere comunicarse con alguna información de su propio espacio de direcciones a otros procesos, entonces solo es posible con técnicas de IPC (comunicación entre procesos). La comunicación puede ser entre procesos relacionados o no relacionados.

Por lo general, la comunicación de procesos interrelacionados se realiza mediante tuberías o tuberías con nombre. La comunicación de procesos no relacionados (por ejemplo, un proceso que se ejecuta en un terminal y otro proceso en otro terminal) se puede realizar mediante canalizaciones con nombre o mediante técnicas populares de IPC de memoria compartida y colas de mensajes.

El problema con las canalizaciones (pipes), la fila de mensajes (fifo) y la cola de mensajes (message queue) es que dos procesos intercambian información. La información tiene que pasar por el kernel.

- El servidor lee el archivo de entrada.
- El servidor escribe estos datos en un mensaje utilizando una tubería, un filtro o una cola de mensajes.
- El cliente lee los datos del canal IPC, lo que nuevamente requiere que los datos se copien del búfer IPC del kernel al búfer del cliente.
- Finalmente, los datos se copian del búfer del cliente.

Se requieren un total de cuatro copias de datos (2 de lectura y 2 de escritura). Entonces, la memoria compartida proporciona una forma de permitir que dos o más procesos compartan un segmento de memoria. Con la memoria compartida, los datos solo se copian dos veces: del archivo de entrada a la memoria compartida y de la memoria compartida al archivo de salida.

Las funciones deben ser utilizadas para el uso de la memoria compartida

ftok (): se usa para generar una clave única.

shmget (): `int shmget (key_t, size_tsize, intshmflg)`; una vez completado con éxito, `shmget ()` devuelve un identificador para el segmento de memoria compartida.

shmat (): antes de poder usar un segmento de memoria compartida, debe adjuntarse a él usando `shmat ()`. `void * shmat (int shmid, void * shmaddr, int shmflg)`; `shmid` es una identificación de memoria compartida. `shmaddr` especifica la dirección específica para usar, pero debemos establecerla en cero y el sistema operativo elegirá automáticamente la dirección.

shmdt (): cuando haya terminado con el segmento de memoria compartida, su programa debería separarse de él usando `shmdt ()`. `int shmdt (vacío * shmaddr)`;

shmctl (): cuando se desconecta de la memoria compartida, no se destruye. Entonces, para destruir se usa `shmctl ()`. `shmctl (int shmid, IPC_RMID, NULL)`;

En nuestro caso se utilizó la memoria compartido para que los procesos intercambien la siguiente información, la cual se encuentra en una estructura especial generada para la memoria compartida:

- Robot: Esta consiste en una estructura la cual se conforma de 4 datos del tipo flotante, estos corresponden a la posición X e Y del robot, la orientación y la velocidad.
- Maleza: Este consiste en un array de una estructura llamada maleza, donde se conforma de 3 flotantes, la posición X e Y de la maleza y su radio. Este array permite enviar la información de varias malezas a la vez.
- CantMal: Este es un dato del tipo entero, donde se envía la cantidad de malezas que se encuentran en el array explicado anteriormente.
- Complete: Dato del tipo booleano donde informara el fin de la trayectoria deseada.
- NewDato: Dato del tipo booleano el cual informara que el software debe leer los datos enviados, es decir que se cumplió un ciclo completo de 100ms.

6.5.1 Descripción del software de simulación

Como se explicó anteriormente este programa tiene la función de generar un mapa de malezas aleatorias, luego a partir de una trayectoria previamente definida, el robot ira avanzando y cada 100ms, el código enviará al programa principal los siguientes datos:

- Posición y orientación del robot
- Posición y malezas en una cercanía al robot.

Dentro de este software existen dos funciones donde, en una de ellas se define un área de cercanía, para el cual a medida que el robot avanza, las malezas enviadas al programa principal serán aquellas que se encuentren dentro del área alternativa. Esta función, se llama “Maleza en Área”

La otra función, corresponde a la trayectoria, donde se especifica como ira avanzando paso a paso el robot, generando la trayectoria deseada. Esta función se llama Trayectoria Robot

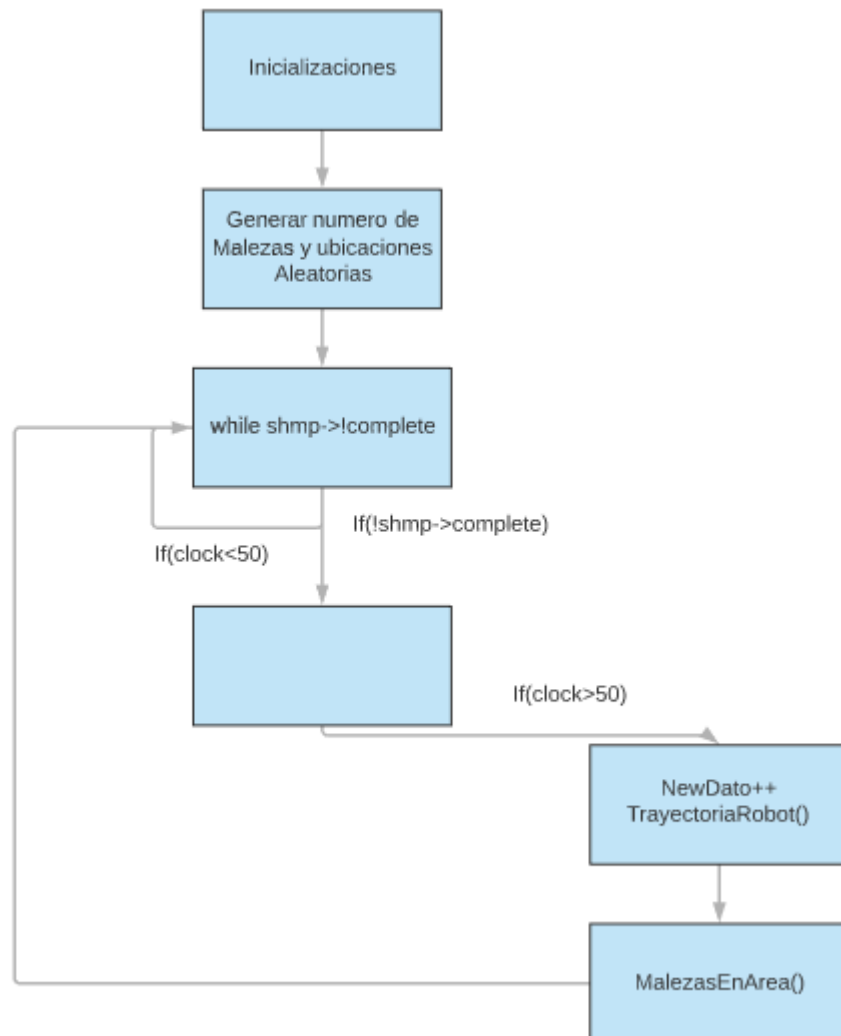


Figura 6.8: Diagrama de flujo software de simulación

El funcionamiento del programa es el siguiente, al inicializarse genera un numero aleatorio de malezas ubicadas en posiciones y de tamaño también aleatorio.

Luego, repetirá de forma cíclica, en este caso cada 50ms, la siguiente secuencia.

Primero calcula la nueva posición del robot mediante la función mencionada anteriormente, “Trayectoria Robot”, además, analiza qué malezas, de las generadas aleatoriamente, se encuentran dentro del área especificada por la función “Maleza en Área”. Esta información, los datos de la posición y orientación del robot, la información de las malezas dentro del área de cercanía al robot y por último la cantidad de malezas ubicadas en esta área se copian a la estructura utilizada para la comunicación entre los procesos hijo y padre, es decir, la comunicación entre los programas Datos Localización y el programa principal.

Dentro de esta estructura se enviarán dos bits más, los cuales especifican el fin de la trayectoria, el cual se activa en el instante donde la trayectoria deseada ya se recorrió. El otro bit corresponde a una bandera donde se le indicara al programa principal que debe leer los datos, ya que se cumplió el ciclo completo desde que leyó el dato anterior, esto ocurre cada 100 ms, es decir, se da en dos ejecuciones del software Datos Localización.

6.5.2 Descripción del software principal

Este es el programa principal dentro de la solución, siendo los explicados anteriormente programas necesarios para la simulación y testeo. Lo explicado anteriormente será sustituido por los bloques de geolocalización y de detección de las malezas – mediante procesamiento digital de imágenes- los cuales le brindarán la información necesaria a este programa para que el mismo defina qué electroválvulas necesita abrir para atacar a las malezas. Por ende, este software necesitara como información externa los datos de las malezas y los datos de geolocalización que serán enviados a través de los módulos correspondientes.

Como se explicó en secciones previas, la información que recibirá de forma externa este programa es la siguiente:

- Robot: Esta consiste en una estructura la cual se conforma de 4 datos del tipo flotante, estos corresponden a la posición X e Y del robot, la orientación y la velocidad.
- Maleza: Este consiste en un array de una estructura llamada maleza, donde se conforma de 3 flotantes, la posición X e Y de la maleza y su radio. Este array permite enviar la información de varias malezas a la vez.
- CantMal: Este es un dato del tipo entero, donde se envía la cantidad de malezas que se encuentran en el array explicado anteriormente.
- Complete: Dato del tipo booleano donde informara el fin de la trayectoria deseada.
- NewDato: Dato del tipo booleano el cual informara que el software debe leer los datos enviados, es decir que se cumplió un ciclo completo de 100ms.

Cabe aclarar que el dato llamado NewDato se utiliza para sincronizar la ejecución de los programas tanto de este como el de DatosLocalizacion. Para la versión final, el programa correrá de manera independiente, y cíclica ejecutándose cada 100 ms., hasta que haya terminado el recorrido. Esto ya se encuentra en el programa, pero mientras funcione a la par del software DatosLocalizacion está desactivado.

Otros parámetros necesarios para la ejecución del programa son los datos físicos y parámetros mecánicos del robot y electroválvulas, los cuales son configurables mediante un archivo txt, llamado “Parámetros” donde se pueden cargar esta información

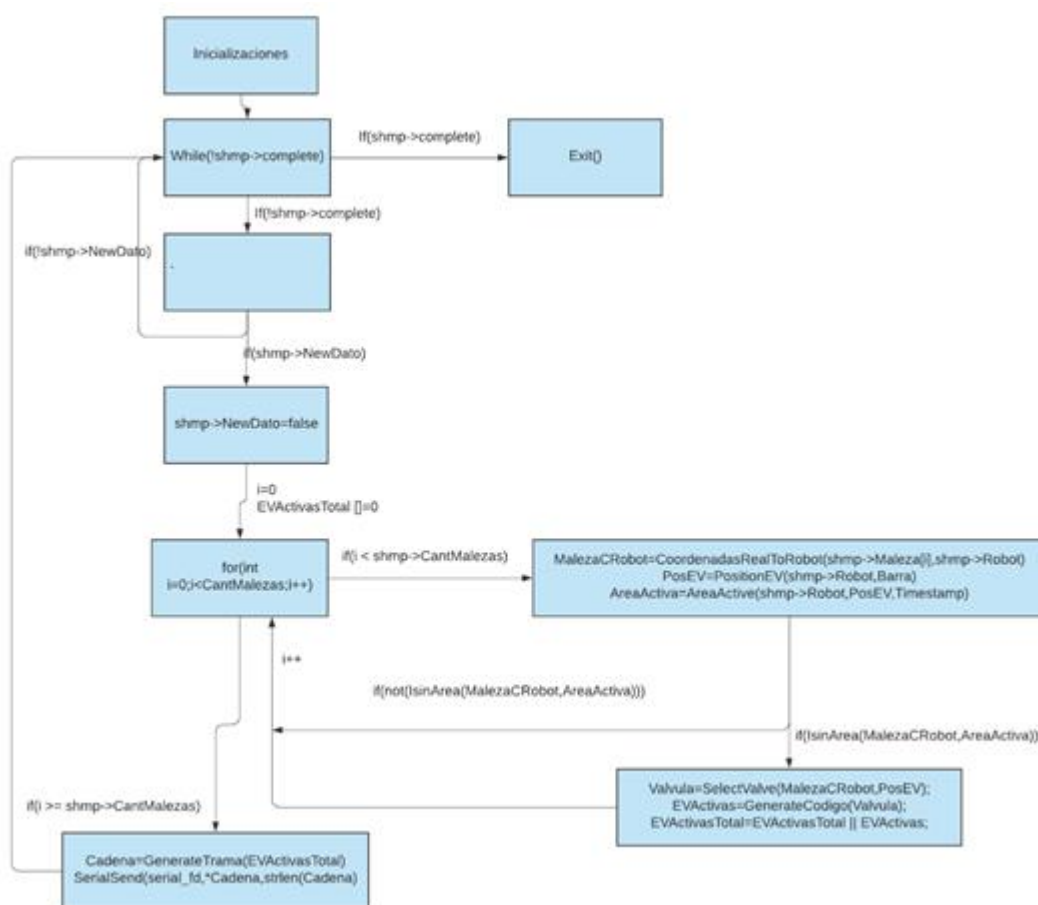


Figura 6.9: Diagrama de flujo software principal

En la imagen anterior se puede ver un esquema de cómo es el proceso de detección que el software sigue.

Para comenzar, luego de ciertas inicializaciones, el programa se ejecutará de manera cíclica mientras la trayectoria no haya finalizado, es decir, mientras el dato “complete” no se encuentre activo.

En estas iteraciones consultará el valor del dato “NewDato” hasta que el mismo tenga el valor true. Cabe destacar que como se comentó anteriormente esta es una forma de sincronización con el programa DatosLocalizacion y será reemplazado, así no evaluará NewDato, sino que mediante un temporizador se ejecutará cada 100ms.

Una vez que NewDato esté activa, se baja, para poder leer la próxima vez que esta bandera tenga el valor de true.

Mientras tanto, repetirá un bucle de iteraciones igual al número de malezas recibida por el dato CantMalezas, donde en cada iteración, trabajará con cada una de las malezas recibida.

Aquí aplicará el análisis teórico-matemático desarrollado en la sección 4.3.

Repasando este análisis, se cambiará de coordenadas a la maleza estudiada, es decir, se modificará las coordenadas de las malezas dada globalmente a un sistema de coordenadas con centro de ejes ubicado sobre el robot. Se convierte a coordenadas de este nuevo sistema de ejes, y se trabajar a partir de aquí en este nuevo de ejes de coordenadas.

Además, se calculará las posiciones de cada una de las electroválvulas, como así también las coordenadas del área que podrá cubrir con el herbicida expulsado por cada una de estas. También se

calculará, a partir del área cubierta por el herbicida expulsado por cada electroválvula, toda el área en el cual se podrá ocupar y actuar durante este ciclo de 100ms.

Para continuar en los siguientes pasos, la maleza deberá estar dentro del área que se abarca y se calculó previamente, por lo menos alguno de los puntos de esta deberá estar dentro de esta área. En caso contrario se descartará la maleza en este scan, y se realizará el mismo proceso con las siguientes malezas recibidas hasta encontrar una que sí cumpla esta característica, es decir, hasta encontrar una que esté dentro del área de trabajo.

A partir de aquí, con la maleza que está dentro del área de este ciclo de iteración, se estudiará qué electroválvulas será necesaria para trabajar en esta, es decir, siguiendo un orden creciente en el eje Y, se buscará la primera electroválvula cuya área de acción coincida con algún punto dentro de la maleza. De esta misma forma se realizará con la última electroválvula.

Como la maleza ocupa una superficie continua, todas las electroválvulas entre la primera y la última encontrada deberán estar activas. Es decir, si la primera y última electroválvulas son la número 2 y 5 respectivamente, las electroválvulas 3 y 4 también deberán estar activas para combatir esta maleza.

A partir de estas electroválvulas se genera un código numérico, colocando un '1' en las electroválvulas que se deben activar para combatir esta maleza y un '0' en las que no son necesarias. Siguiendo el ejemplo del párrafo anterior, el código generado será '011110000' comenzando desde la electroválvula número uno.

Con este código se ira concatenando mediante la función 'Or' con los códigos obtenidos para las otras malezas que se combatirán en este ciclo de iteración.

Una vez que se terminó de ejecutar el bucle de todas las malezas, con el código final se transformará en el código a enviar, aquí se convertirá la trama, reemplazando cada '1' por 'E' y cada '0' por 'A'. Además, se le agregará los caracteres inicial y final de la trama para que el microcontrolador la tome como válida.

Una vez que se tiene la trama lista, se enviará por puerto serie para luego esperar nuevamente que NewDato tenga el valor true y repetir todo este proceso. Esto se repite cada 100ms.

Aquí se muestra un glosario con el nombre de las funciones utilizadas:

CoordenadasRealToRobot()	Esta función, como su nombre lo indica transforma las coordenadas de la maleza de las coordenadas globales al sistema de eje coordenados ubicados sobre el robot. Esta función requiere como parámetros de entrada la información de la maleza a convertir (X, Y y radio) y también la del robot (X, Y y orientación). Devuelve las posiciones X e Y de la maleza en este nuevo sistema de coordenadas.
PositionEV()	Calcula las coordenadas necesarias para delimitar el área cubierta por la pulverización de herbicida de cada electroválvula. Esta función requiere como parámetros de entrada la información del robot (X, Y y orientación) como así también los parámetros de dimensiones y especificaciones mecánicas del barral, para esto existe una estructura llamada barral que agrupa toda la información del mismo. Esta función retorna las coordenadas (Xin,Xfin,Yin,Yfin) del área de cada una de las electroválvulas
AreaActive()	Genera el área que se abarcara durante el ciclo de scan (100ms). Esta área será en la cual se estudiará en esta iteración.

	<p>Necesita como parámetros de entrada la información del área de cada una de las electroválvulas que devuelve la función PositionEV, la información del robot (X, Y, orientación y velocidad) y el timestamp definido en 100ms.</p> <p>Esta función retorna las coordenadas (Xin,Xfin,Yin,Yfin) del área activa.</p>
IsinArea()	<p>Esta función devuelve un valor booleano, dependiendo si la maleza se posee, por lo menos algún punto dentro del área activa.</p> <p>Por lo tanto, requerirá la información de la maleza y del área activa</p>
SelectValve()	<p>Esta función es la encargada de determinar las electroválvulas de inicio y fin necesarias para combatir la maleza.</p> <p>La misma devuelve una estructura que consiste en dos enteros, para señalar la electroválvula de inicio y fin.</p> <p>Para esto necesitara las coordenadas de la maleza y del área que cubre cada una de las electroválvulas</p>
GenerateCodigo()	<p>La misma genera un código de unos y ceros, tomando como parámetros de entrada la estructura de electroválvulas de inicio y fin que devuelve la función SelectValve.</p> <p>Esta retorna el código numérico con unos en las posiciones de las electroválvulas entre la electroválvula de inicio y fin, incluyéndolas</p>
GenerateTrama()	<p>Esta función genera la trama necesaria para enviar al microcontrolador, convirtiendo los '1' en 'E' y los '0' en 'A'.</p> <p>Además, de incluir los caracteres de validación.</p> <p>Necesita como parámetros de entrada el código obtenido como la concatenación (OR) de todos los códigos obtenidos para cada maleza por la función GenerateCodigo</p>
serial_open()	<p>Esta función se encarga de iniciar la comunicación por puerto serie, abriendo el puerto.</p> <p>Para esto se le debe especificar qué puerto se desea abrir y el baud rate que se utilizara.</p> <p>Devolverá -1 en caso de que no haya conseguido abrir el puerto, u otro número entero en caso de que este haya sido satisfactorio, este valor será utilizado para identificar este puerto</p>
Serial_send()	<p>Función usada para envío de la trama por puerto serie.</p> <p>Requiere el identificador del puerto, la trama a enviar y el tamaño de la misma.</p>

7. Ensayo Y Pruebas De Campo

Una vez ensayados todos los elementos por separado y verificado su correcto funcionamiento, se procedió al ensamble de estos y su posterior ensayo para verificar que cumple con las expectativas.

Procedimos a verificar el correcto funcionamiento individual de cada una de las partes, una vez finalizada la etapa de armado correspondiente.

Estas serían:

- Ensayo del funcionamiento del circuito de activación de electroválvulas (En Protoboard).
- Ensayo de la plaqueta final del circuito de adaptación (PCB)
- Verificación del correcto funcionamiento del software de activación de electroválvulas. Este ensayo puede ser realizado junto con la plaqueta para analizar el funcionamiento correcto del actuador final de control.
- Verificación del software de la NUC mediante simulaciones.
- Ensayo final del control del sistema de pulverización completo. (Queda pendiente)

Estas pruebas por etapas se realizaron a fin de comprobar el funcionamiento de cada uno de los elementos por separado. De esta forma, se pudo verificar un correcto funcionamiento del elemento al comenzar la etapa de desarrollo del elemento previo, es decir asegurar que funciona correctamente para la verificación del elemento que se encuentra aguas arriba del mismo.

7.1 Testeo de software sobre el micro (con PuTTY)

Para el testeo del funcionamiento del microcontrolador, en una primera etapa se testeo enviándole la trama por puerto serie y comprobando que las salidas que correspondan activarse lo hayan realizado, y que aquellas salidas que no, se encuentren apagadas.

Para esto, se utilizó el software PuTTY. Este es un emulador de terminal de código abierto y gratuito, el cual admite varios protocolos de red, incluyendo SSH, Telnet, rlogin y conexión de socket sin formato. También se puede conectar a un puerto serie.

Por lo tanto, se utilizó este emulador de terminal para enviar por puerto serie diferentes tramas (validas y no validas) y mediante un tester, comprobar que las salidas, dispuestas como se explicó en la sección 6.4, posean las tensiones correspondientes al dato enviado en la trama, es decir 3.3V en el caso de que esa salida haya recibido una 'E' y 0V en el caso de que haya recibido una 'A' en su posición (siempre y cuando la trama cumpla con los parámetros de validación).

7.2 Testeo PCB y cableado

Una vez que se terminó el desarrollo y armado del PCB, y que se cableo las salidas con las electroválvulas correspondientes y su respectiva alimentación, se realizó una prueba para comprobar el correcto funcionamiento del mismo (PCB y cableado).

Esto se llevó a cabo alimentando las entradas de la placa, simulando la señal que enviaría el microcontrolador, es decir 3.3V en el caso que se desee activar. A partir de allí, alimentando entrada por entrada, se pudo corroborar como iban activándose la electroválvula correspondiente al pin de entrada alimentado.

7.3 Testeo periodicidad de envío de tramas

Otro análisis que se llevó a cabo fue para evaluar que las tramas sean enviadas, tal como se desea cada 100ms.

Para esto, utilizamos el software Serial Port Monitor, la cual es una aplicación profesional para la supervisión de puertos COM, RS232/422/485. Este supervisa, muestra, registra y analiza toda la actividad de un puerto serie en un sistema. Se puede supervisar varios puertos simultáneamente y comparar sus sesiones.

#	Time	Function	Direct...	Status	Data	Data (chars)	D..	R...	Port	Comments
679	12:17:11:737	IRP_MJ_READ	UP	STATUS_SUCCESS	59	Y	1		COM3	
680	12:17:11:737	IRP_MJ_READ	DOWN					1	COM3	
681	12:17:11:836	IRP_MJ_READ	UP	STATUS_SUCCESS	3a	:	1		COM3	Inicio de primer trama
682	12:17:11:836	IRP_MJ_READ	DOWN					1	COM3	
683	12:17:11:836	IRP_MJ_READ	UP	STATUS_SUCCESS	45	E	1		COM3	
684	12:17:11:836	IRP_MJ_READ	DOWN					1	COM3	
685	12:17:11:836	IRP_MJ_READ	UP	STATUS_SUCCESS	45	E	1		COM3	
686	12:17:11:836	IRP_MJ_READ	DOWN					1	COM3	
687	12:17:11:836	IRP_MJ_READ	UP	STATUS_SUCCESS	45	E	1		COM3	
688	12:17:11:836	IRP_MJ_READ	DOWN					1	COM3	
689	12:17:11:836	IRP_MJ_READ	UP	STATUS_SUCCESS	45	E	1		COM3	
690	12:17:11:836	IRP_MJ_READ	DOWN					1	COM3	
691	12:17:11:836	IRP_MJ_READ	UP	STATUS_SUCCESS	41	A	1		COM3	
692	12:17:11:836	IRP_MJ_READ	DOWN					1	COM3	
693	12:17:11:836	IRP_MJ_READ	UP	STATUS_SUCCESS	41	A	1		COM3	
694	12:17:11:836	IRP_MJ_READ	DOWN					1	COM3	
695	12:17:11:837	IRP_MJ_READ	UP	STATUS_SUCCESS	41	A	1		COM3	
696	12:17:11:837	IRP_MJ_READ	DOWN					1	COM3	
697	12:17:11:837	IRP_MJ_READ	UP	STATUS_SUCCESS	41	A	1		COM3	
698	12:17:11:837	IRP_MJ_READ	DOWN					1	COM3	
699	12:17:11:837	IRP_MJ_READ	UP	STATUS_SUCCESS	41	A	1		COM3	
700	12:17:11:837	IRP_MJ_READ	DOWN					1	COM3	
701	12:17:11:837	IRP_MJ_READ	UP	STATUS_SUCCESS	59	Y	1		COM3	
702	12:17:11:837	IRP_MJ_READ	DOWN					1	COM3	
703	12:17:11:936	IRP_MJ_READ	UP	STATUS_SUCCESS	3a	:	1		COM3	Inicio de segunda trama
704	12:17:11:936	IRP_MJ_READ	DOWN					1	COM3	
705	12:17:11:936	IRP_MJ_READ	UP	STATUS_SUCCESS	41	A	1		COM3	
706	12:17:11:936	IRP_MJ_READ	DOWN					1	COM3	
707	12:17:11:936	IRP_MJ_READ	UP	STATUS_SUCCESS	41	A	1		COM3	
708	12:17:11:936	IRP_MJ_READ	DOWN					1	COM3	
709	12:17:11:936	IRP_MJ_READ	UP	STATUS_SUCCESS	41	A	1		COM3	
710	12:17:11:936	IRP_MJ_READ	DOWN					1	COM3	
711	12:17:11:936	IRP_MJ_READ	UP	STATUS_SUCCESS	41	A	1		COM3	

Figura 7.1: Supervisión de trama con Serial Port Monitor

Se puede ver en la imagen anterior como el inicio de una trama, la cual está dada por el carácter ':' y el inicio de la siguiente trama que se recibió por el puerto serie dado por el mismo carácter. Que la diferencia temporal entre estos caracteres fue de 100ms, lo mismo sucede para otras tramas, con lo cual se logra verificar que la tramas son recibidas y por lo tanto enviadas con la periodicidad deseada, es decir, 100ms.

7.4 Animación en Blender

7.4.1 ¿Qué es blender?

Blender es un programa de código abierto creado por la compañía Blender Foundation especializado en animación 3D. Entre los softwares de animación este es un programa multiplataforma que funciona sin problemas en Linux, Windows y Macintosh.[2]

Se puede utilizar para crear visualizaciones en 3D, como imágenes fijas, animaciones en 3D, tomas VFX y edición de video.

Características claves:

- Es una suite de creación de contenido 3D totalmente integrada, que ofrece una amplia gama de herramientas esenciales, incluyendo Modelado, Renderizado, Animación y Rigging, Edición de Video, VFX, Composición, Texturizado, y algunos tipos de Simulaciones.
- Es multiplataforma, con una OpenGL GUI que es uniforme en todas las plataformas principales (y personalizable con scripts de Python).
- Tiene una arquitectura 3D de alta calidad, lo que permite un flujo de trabajo de creación rápida y eficiente.
- Cuenta con un apoyo activo de la comunidad.
- Tiene un pequeño ejecutable, que es opcionalmente portátil.

7.4.2 Keyframes

Un *keyframe* es simplemente un marcador de tiempo que almacena el valor de una propiedad.

Por ejemplo, un keyframe podría definir que la posición horizontal de un objeto está a 3 m en el fotograma 1.

El propósito de un keyframe es permitir la animación interpolada, lo que significa, por ejemplo, que el usuario podría agregar otro keyframe en el fotograma 10, especificando la posición horizontal del objeto a 20 m, y Blender determinará automáticamente la posición correcta del objeto para todos los fotogramas entre el fotograma 1 y 10 en función del método de interpolación elegido (por ejemplo, lineal, Bézier, cuadrático, etc.).

Métodos de interpolación:

- Constante

No hay interpolación en absoluto. La curva contiene el valor de su último fotograma clave, dando una "curva" discreta (escalera). Por lo general, solo se usa durante la etapa inicial de "bloqueo" en los flujos de trabajo de animación de pose a pose.



Figura 7.2: Interpolación constante

- Lineal

Esta simple interpolación crea un segmento recto, dando una línea no continua. Puede ser útil cuando se usan solo dos fotogramas clave y el modo de extensión de extrapolación, para obtener fácilmente una línea recta infinita.

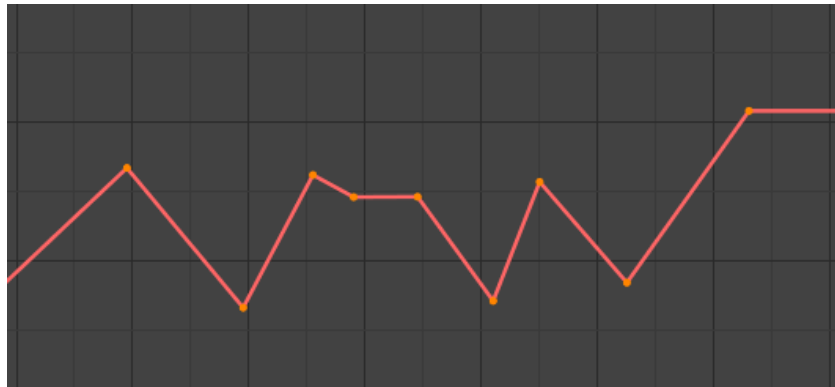


Figura 7.3: Interpolación lineal

- Bézier

La interpolación más poderosa y útil, y la predeterminada. Ofrece curvas muy suaves, es decir, animaciones suaves. Esta es el método utilizado en nuestro caso

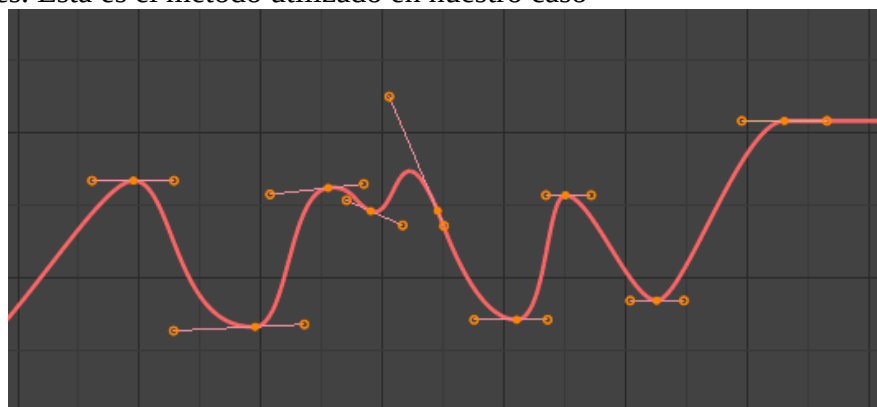


Figura 7.4: Interpolación Bézier

7.4.3 Animación generada para testeo del software.

A continuación, se pasará a explicar la animación generada para el análisis y testeo del software correspondiente a la computadora.

Para esto, se generó un modelado del robot en 3D, respetando las dimensiones reales del mismo, pudiendo aprovechar el modelado previo de las partes del robot en proyectos anteriormente realizados. Una vez asociado todas estas partes del robot, se dotó de una animación a las ruedas para que las mismas giren a medida que el robot avance, dándole así una mayor veracidad a la animación.

El modelado del prototipo final se puede visualizar en la siguiente figura:

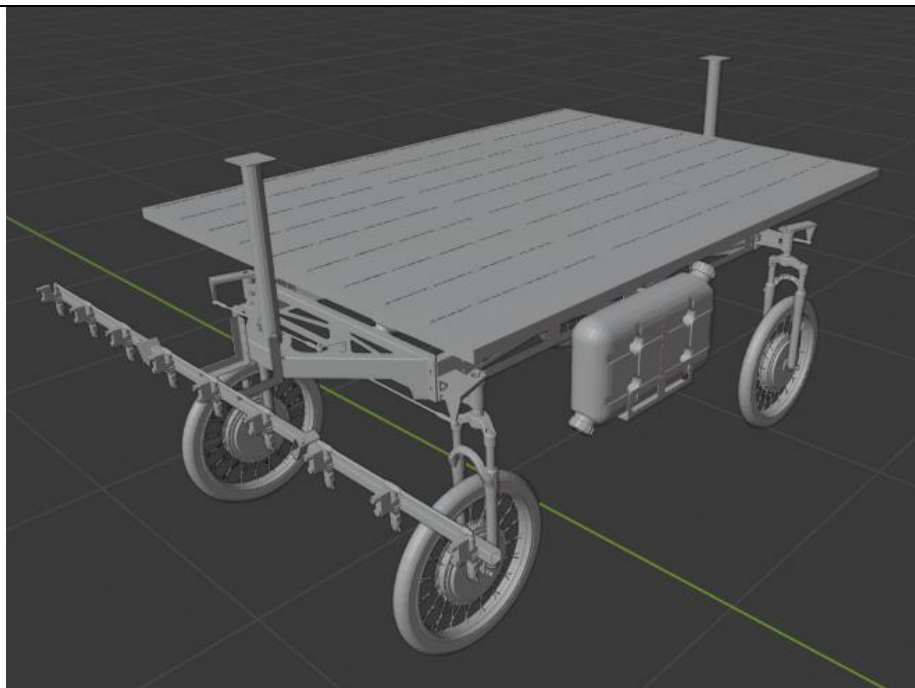


Figura 7.5: Modelado del robot en Blender

A partir de allí, mediante un script en Python se genera la animación, aprovechando los archivos txt que brinda el software de la computadora en modo debug.

Los mismos contienen la información de la posición y orientación del robot en cada scan, la información de las posiciones de todas las malezas y la trama enviada por puerto serie en cada scan.

A partir de esta información, en el script se prepara la escena de animación, configurando la cantidad de frames que tendrá, este valor se corresponde con la cantidad de scans que se obtuvo al correr el software. Además, para cada uno de estos frames, se le insertaran algunos keyframes, que tal como se explicó anteriormente nos permite definir propiedades del robot para cada frame. Allí se establece para cada uno de estos frames, la posición y orientación del robot.

Para finalizar, se modelizó la apertura de las electroválvulas mediante conos de color verde, que representarían el agroquímico pulverizado por cada una de estas.

Por lo tanto, mediante la información de la trama enviada en cada scan, se inserta nuevos keyframes de visibilidad para cada uno de estos conos en cada frame, correspondiente con la trama.

De este modo, se simula la apertura, dejando visibles estos conos solo para aquellas electroválvulas que se encuentran abiertas en este frame.

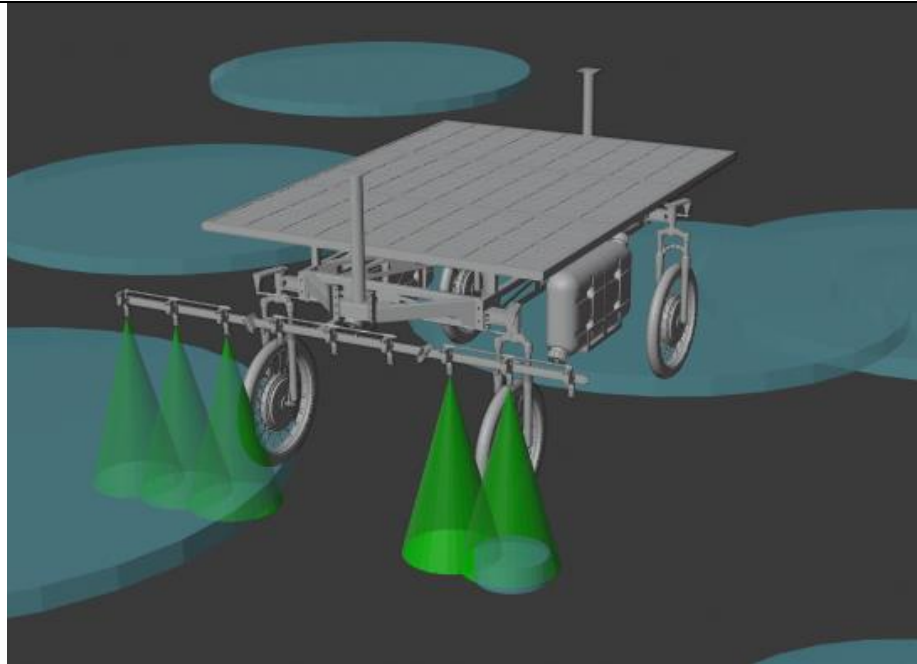


Figura 7.6: Apertura de válvulas según la ubicación de la maleza

En la siguiente figura se podrá ver el código implementado que genera y prepara la animación, identificando algunos de los pasos más importantes.

```

1 import bpy
2 from bpy import ops
3 #KeyFrame Robotpositions
4 RobotPos = open('C:/Users/Guillermo/Desktop/Proyecto Final/DebugFiles/LocalizacionRobotDebug.txt', 'r')
5 lines=len(RobotPos.readlines())
6 RobotPos.seek(0)
7 Trama= open('C:/Users/Guillermo/Desktop/Proyecto Final/DebugFiles/SerialPort Ubuntu.txt')
8 # prepare a scene
9 scn = bpy.context.scene
10 scn.frame_start = 1
11 scn.frame_end = lines
12 #Delete alls keyframes
13 context=bpy.context
14 for ob in context.selected_objects:
15     ob.animation_data_clear()
16 #Set position del carro
17 Trama.seek(0)
18 for Data in RobotPos:
19     Datos=[Data.strip().split('\t')]
20     bpy.data.objects['Empty'].location = (float(Datos[0][0]),float(Datos[0][1]),1.285)
21     bpy.data.objects['Empty'].rotation_euler=(0,0,float(Datos[0][2]))
22     bpy.data.objects['Empty'].keyframe_insert(data_path='location',frame=float(Datos[0][3]))
23     bpy.data.objects['Empty'].keyframe_insert(data_path='rotation_euler',frame=float(Datos[0][3]))
24     TramaVar=Trama.readlines(1)
25     for i in range(0,len(TramaVar[0])):
26         print(TramaVar[0][i])
27         Cone='Cone'+ str(i)
28         print(Cone)
29         if(TramaVar[0][i]=='A'):
30             print("Apagado")
31             bpy.data.objects[Cone].hide_viewport=True
32             bpy.data.objects[Cone].keyframe_insert(data_path='hide_viewport',frame=float(Datos[0][3]))
33         elif(TramaVar[0][i]=='E'):
34             print("Encendido")
35             bpy.data.objects[Cone].hide_viewport=False
36             bpy.data.objects[Cone].keyframe_insert(data_path='hide_viewport',frame=float(Datos[0][3]))
37
38 RobotPos.close()
39 Trama.close()
40
41 #Delete old circles
42 if 'Malezal' in bpy.data.objects:
43     mat=bpy.context.scene.objects['Malezal'].active_material
44     for obj in bpy.data.objects:
45         print(obj.name)
46         print(obj.active_material)
47         print("-----")
48         if obj.active_material==mat:
49             bpy.data.objects.remove(obj)
50
51 #PrintAllCircles
52 CircleFile = open('C:/Users/Guillermo/Desktop/Proyecto Final/DebugFiles/TodasMalezas.txt')
53 contMal=0
54 mat_Maleza=bpy.data.materials.new('Malezas')
55 mat_Maleza.diffuse_color=(0.283,0.861,1,0.89)
56 for x in CircleFile:
57     Circle=[x.strip().split('\t')]
58     # ops.curve.primitive_bezier_circle_add(radius=float(Circle[0][2]),location=(float(Circle[0][0]),float(Circle[0][1]),0.0))
59     bpy.ops.mesh.primitive_cylinder_add(depth=0.1,radius=float(Circle[0][2]),location=(float(Circle[0][0]),float(Circle[0][1]),-0.05))
60     bpy.data.objects['Cylinder'].active_material=mat_Maleza
61     print(Circle)
62     contMal=contMal+1
63     bpy.context.scene.objects['Cylinder'].name=('Maleza' + str(contMal))
64

```

Preparar la escena con la cantidad de frames necesarios

Eliminacion de todos los keyframes previos existentes

Key frames de posicion y orientacion del robot

Key frames de visibilidad para el herbicida pulverizado

Se eliminan todas las malezas previas ya dibujadas

Dibujo de cilindros que representan las malezas

Figura 7.7: Código del script ejecutado desde Blender

Se pueden ver algunas animaciones obtenidas en este software en los siguientes links:

<https://drive.google.com/drive/folders/1DSEH-GYrFMAXRto1WqoSeYSUVQoBHqN7?usp=sharing>

7.5 Animación en Matlab

Se realizó como paso para la comprobación del funcionamiento del software, una animación en Matlab consistente en lo siguiente:

Se cargó en un script el mapa de las malezas generadas aleatoriamente que había sido utilizado con anterioridad a modo de testeo del software principal. Estas se imprimieron en un plano en dos dimensiones, junto a un esquema simplificado del robot y la zona de injerencia de cada electroválvula en un momento determinado. En la figura se puede ver plasmado.

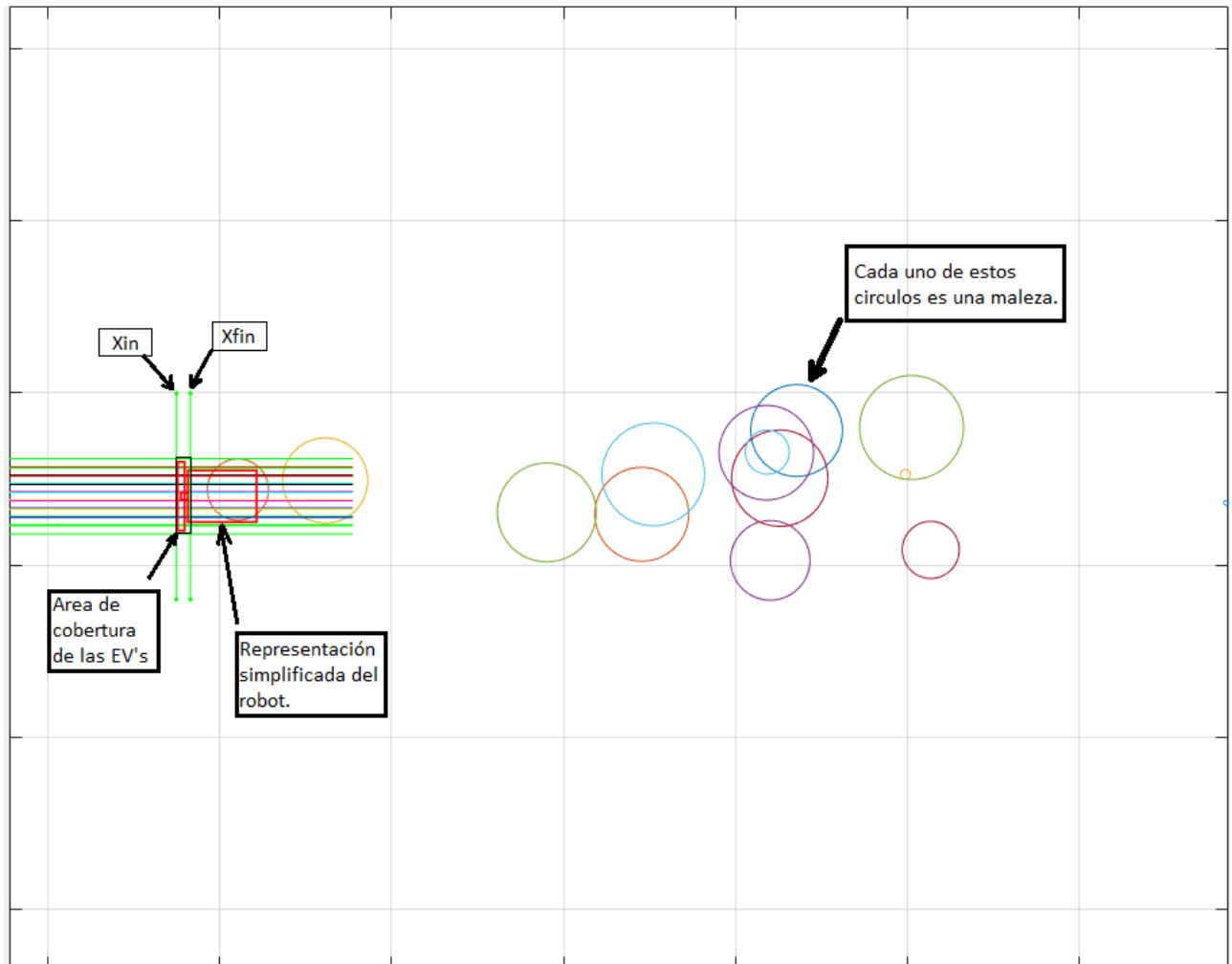


Figura 7.8: Animación en Matlab

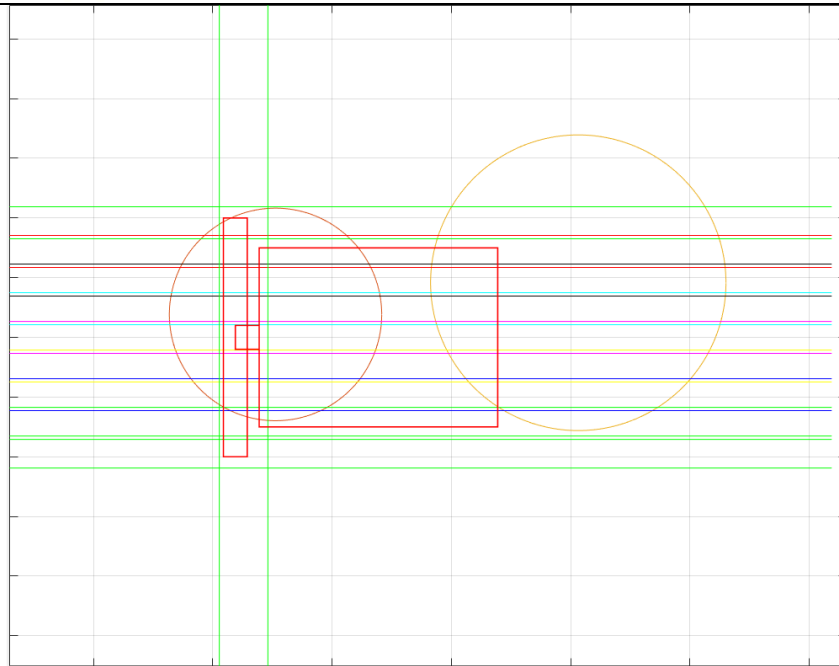


Figura 7.9: Animación anterior con zoom

A continuación, se procedió a ejecutar el script paso a paso, obteniendo como resultado el movimiento del robot en la dirección de avance, en una distancia que se corresponde con la velocidad adoptada para el desplazamiento multiplicada por el timestamp definido para el programa principal.

El resultado se verifica comparando visualmente que para el paso de simulación en que se encuentra, la válvula que corresponde (dada la ubicación de la maleza en el espacio) esta encendida. Esto es posible visualizarlo en el documento que muestra las tramas que se envían al microcontrolador para activar y desactivar las electroválvulas.

De esta forma, es posible verificar que una maleza cuyo centro se encuentre en la posición dada por la dupla (x, y) , con radio r , debe ser atacada con herbicida desde las válvulas correspondientes.

Para ejemplificar como fue utilizada esta herramienta de testeo, en la figura 7.9 se puede ver que las electroválvulas de la 2 a la 9 deberían estar encendidas. Esto se comparó con la trama devuelta por el software para ese instante.

7.6 Prueba real

Por el momento, debido a la situación de aislamiento que genero la pandemia del COVID-19, no fue posible tener acceso al robot, que actualmente se encuentra en el CIFASIS, para realizar las pruebas en campo de todo el sistema conjuntamente

8. Conclusiones

Previo a concluir sobre lo realizado hasta aquí, queremos destacar la importancia del proyecto, que a su vez incluye muchas de las incumbencias profesionales de un ingeniero electrónico. A grandes rasgos destacamos, el diseño de software, programación de un microcontrolador, diseño y armado de una placa en PCB, entre otras. Por otro lado, trabajamos sobre software que procesa y toma las decisiones en tiempo real.

Dadas las circunstancias actuales, de encontrarnos en medio de una situación de aislamiento debido al virus COVID-19, no fue posible realizar parte de las pruebas del sistema completo.

Además, queremos destacar que, al no contar con los desarrollos previos necesarios, como es el caso del sistema de detección de las malezas, y el de geolocalización, se tuvo que desarrollar un programa de simulación capaz de suplir estos módulos faltantes.

Destacamos que dada la modularidad del sistema implementado, se pudo desarrollar y verificar el funcionamiento de cada una de las etapas por separado, permitiendo detectar posibles fallas, individualizarlas, y corregirlas previamente al desarrollo de la próxima etapa. Además, esto nos permitió trabajar en paralelo con más de una etapa, permitiendo la reducción de tiempo de implementación.

Una de las partes importantes del proyecto, fue el armado y la integración de los elementos que componen el sistema de pulverización para que finalmente el robot pueda ejecutar las tareas para las que fue diseñado. Esto es, la correcta aplicación de producto herbicida sobre las malezas presentes entre cultivos.

Finalmente, como mejoras a tener en cuenta para futuras implementaciones, se puede mencionar el hecho de incluir el microcontrolador dentro de la misma placa de adecuación de señales, para la presentación de la solución como un producto final, completo y modular.

9. Bibliografía

- [1] Documentacion Electroválvulas
<https://www.altecdust.com/blog/item/32-como-funcionan-las-electrovalvulas-o-valvulas-solenoides-de-uso-general>

- [2] Documentacion sobre Blender
https://docs.blender.org/manual/es/dev/getting_started/about/introduction.html

- [3] Documentacion MCU Xpresso IDE
https://www.nxp.com/docs/en/user-guide/MCUXpresso_IDE_User_Guide.pdf
<https://www.nxp.com/design/software/development-software/mcuxpresso-software-and-tools-/mcuxpresso-integrated-development-environment-ide:MCUXpresso-IDE>

- [4] Documentacion Code Blocks
<https://code-blocks.uptodown.com/windows>

- [5] Documentacion VMware
<https://www.vmware.com/ar/products/workstation-pro/faqs.html>

- [6] Documentacion funciones librería utilizadas
<https://www.geeksforgeeks.org/fork-system-call/>
<https://www.geeksforgeeks.org/exec-family-of-functions-in-c/#:~:text=The%20exec%20type%20system%20calls,with%20the%20file%20being%20executed.>
<https://www.geeksforgeeks.org/ipc-shared-memory/>

10. Apéndice

10.1 Hoja de datos transistor TIP29C

Absolute maximum ratings

TIP29A TIP29C

1 Absolute maximum ratings

Table 2. Absolute maximum ratings

Symbol	Parameter	Value		Unit
		TIP29A	TIP29C	
V_{CBO}	Collector-base voltage ($I_E = 0$)	60	100	V
V_{CEO}	Collector-emitter voltage ($I_B = 0$)	60	100	V
V_{EBO}	Emitter-base voltage ($I_C = 0$)	5		V
I_C	Collector current	1		A
I_{CM}	Collector peak current ($t_p < ms$)	3		A
I_B	Base current	0.4		A
P_{TOT}	Total dissipation at $T_c \leq 25^\circ C$	30		W
	Total dissipation at $T_{amb} \leq 25^\circ C$	2		W
T_{stg}	Storage temperature	-65 to 150		$^\circ C$
T_J	Max. operating junction temperature	150		$^\circ C$

2 Electrical characteristics

($T_{case} = 25^{\circ}C$; unless otherwise specified)

Table 3. Electrical characteristics

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
I_{CEO}	Collector cut-off current ($I_B = 0$)	for TIP29A $V_{CE} = 30V$ for TIP29C $V_{CE} = 60V$			0.3 0.3	mA mA
I_{CES}	Collector cut-off current ($V_{BE} = 0$)	for TIP29A $V_{CE} = 60V$ for TIP29C $V_{CE} = 100V$			0.2 0.2	mA mA
I_{EBO}	Emitter cut-off current ($I_C = 0$)	$V_{EB} = 5V$			1	mA
$V_{CEO(sus)}^{(1)}$	Collector-emitter sustaining voltage ($I_B = 0$)	$I_C = 30mA$ for TIP29A for TIP29C	60 100			V V
$V_{CE(sat)}^{(1)}$	Collector-emitter saturation voltage	$I_C = 1A$ $I_B = 125mA$			0.7	V
$V_{BE}^{(1)}$	Base-emitter voltage	$I_C = 1A$ $V_{CE} = 4V$			1.3	V
$h_{FE}^{(1)}$	DC current gain	$I_C = 0.2A$ $V_{CE} = 4V$ $I_C = 1A$ $V_{CE} = 4V$	40 15		75	

1. Pulsed duration = 300 ms, duty cycle $\geq 1.5\%$.

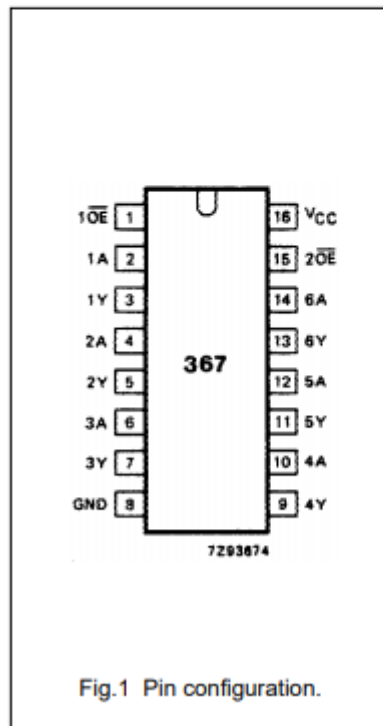
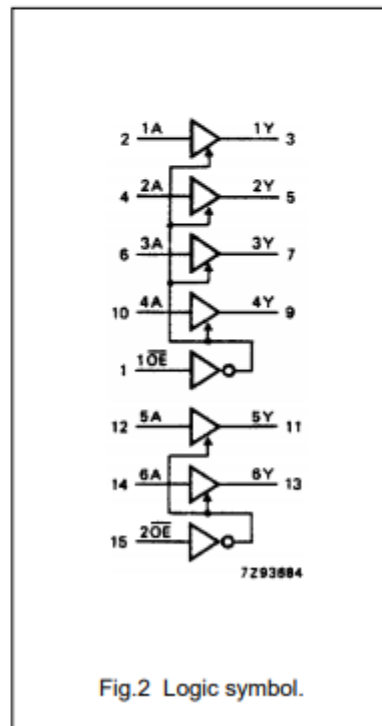
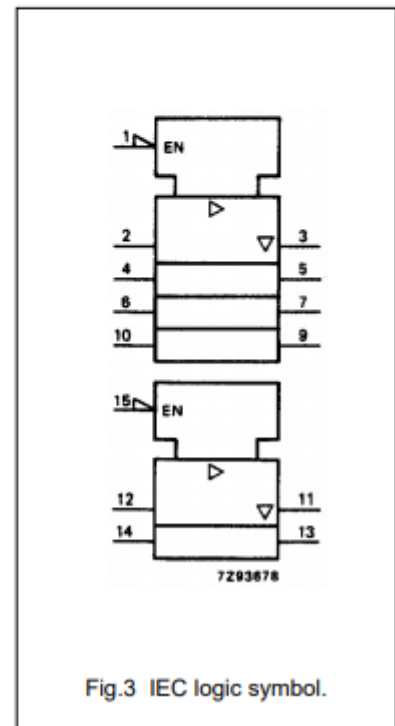
10.2 Hoja de datos buffer 74HC367N

Hex buffer/line driver; 3-state

74HC/HCT367

PIN DESCRIPTION

PIN NO.	SYMBOL	NAME AND FUNCTION
1, 15	$\overline{1OE}, \overline{2OE}$	output enable inputs (active LOW)
2, 4, 6, 10, 12, 14	1A to 6A	data inputs
3, 5, 7, 9, 11, 13	1Y to 6Y	data outputs
8	GND	ground (0 V)
16	V _{CC}	positive supply voltage



10.3 Esquema final de placa PCB

