

ROBOTS PARA AGRICULTURA

SISTEMA DE ASPERSIÓN PARA LOS CULTIVOS DE MARACUYÁ MEDIANTE ROBÓTICA APLICADA

Brahyan Marulanda, Henry Carmona, Daniel Tobar, José Swan

En Colombia, la agricultura y los entornos agropecuarios han sido los más afectados en cuanto a su transición y desarrollo normal debido a los subproblemas como lo son los cultivos improductivos, la deforestación acelerada, los grandes subsidios ineficientes, dietas que no son saludables y el aumento de importaciones, además de la perdida de alimentos a causa de los cambios climáticos. Específicamente, en el país se pierden y se desperdician un total de 9.76 millones de toneladas de alimentos, que es lo representativo al 34% del total producido y un 64% de ello corresponde a las pérdidas causadas en las etapas de producción, postcosecha, almacenamiento y procesamiento industrial donde, el sector agropecuario y/o campesino cuenta con un porcentaje del 40.5 % de todo el desperdicio. Lograr reducir el porcentaje de perdida de los alimentos ha sido un reto para los agricultores debido a las técnicas arcaicas y poco tecnificadas tanto de cosechar como de controlar las plagas, enfermedades o afecciones en cada uno de los cultivos. Hasta el momento, los métodos existentes para realizar dicho control son el tratamiento manual con motobomba del fungicida, aspersión de insecticidas, sistemas de goteo, atomizadores caseros, entre otros. Sin embargo, no han sido tan efectivos. Por tal motivo el equipo de trabajo, en compañía de Deivy Mañozca, agricultor del corregimiento de Rozo, Valle del Cauca, el cual es la persona doliente de la problemática, occasionó el surgimiento de la idea del Splinker Robot, un robot aspersor capacitado para pulverizar cultivos de maracuyá en terrenos planos y semiplanos abordando los retos de movilidad y maniobrabilidad con el objetivo de suprir el trabajo pesado de los agricultores, el cual tendrá como objetivo, reducir las pérdidas a partir del control en la reducción del hongo. Por tal motivo, para dicho dispositivo, será realizado la formulación del problema, la motivación del equipo de trabajo y los potenciales impactos del producto desarrollado además de la explicación textual de la solución robótica propuesta y un diagrama esquemático de la misma, así como también la explicación de los modelos cinemáticos directos e inversos y, por último, la visualización del problema y de la solución usando Rviz y de forma simultánea, utilizando Gazebo. Todo ello, organizado en un cronograma en el que se tendrán las actividades realizadas y logros alcanzados, semana a semana, durante el desarrollo del proyecto además de la explicación del trabajo independiente realizado: donde se desatarán las fuentes consultadas y procedimientos realizados en aras de fundamentar la habilidad del equipo de trabajo para identificar, formular y resolver problemas complejos desde la ingeniería.

INTRODUCCIÓN

En Colombia, según Darío Fajardo¹ se tiene que algunos factores que han incidido radicalmente en la agricultura colombiana ha sido el conflicto armado, el narcotráfico, la dificultad en el acceso a la tierra, la desigualdad y la exclusión social, además de la mala gestión y distribución, valoración e importancia de la mano de obra campesina en el país donde se tiene la presencia de la caída en la producción de alimentos en las últimas décadas. Ello, sustentado que para el año 2013, tan sólo el 9.6% de los productores recibieron alguna asistencia técnica para el desarrollo de actividades agropecuarias. Además, la situación actual del país presenta la ineficiente gestión del acceso de alimentos, los incentivos para que las personas cambien sus dietas, el desperdicio de alimentos y la gestión de los ecosistemas naturales. En complemento a lo anterior, la agricultura colombiana se caracteriza por modelos

extensivos de baja productividad. Actualmente, 43 millones de hectáreas son de uso agropecuario, de las cuales, 34 millones, asociadas a la ganadería, actividad que genera graves efectos ambientales, como la deforestación y la emisión de gases de efecto invernadero. No obstante, esta actividad aporta el 1,7% del PIB nacional y el 25% del PIB agropecuario. Sin embargo, la mayoría de los productores están aislados y empobrecidos, con vías terciarias “en condiciones lamentables”. Además, la mayoría no cuenta con asistencia técnica para llevar a cabo su proceso.[1] Uno de esos casos a tratar, es la situación de Deivy Mañozca, agricultor de Rozo, Valle del Cauca, para el cual, el equipo de trabajo (Brahyan Marulanda, José Swan, Daniel Tobar y Henry Carmona), ha propuesto una solución desde la ingeniería.

¹ Antropólogo, profesor de la Universidad Externado de Colombia.

SCIENCE ROBOTICS | SPLINKER ROBOT

La solución corresponde al Splinker Robot, un robot aspersor capacitado para pulverizar cultivos de maracuyá en terrenos planos y semiplanos, el cual aborda los retos de movilidad y maniobrabilidad con el objetivo de suplir el trabajo pesado de los agricultores, que en caso específico es Deivy Mañozca y tiene como objetivo funcional y práctico, reducir las pérdidas a partir del control en la reducción del hongo y optimizar la eficiencia de los cultivos. Así, con el desarrollo del presente documento, será realizado en primer lugar, la formulación del problema, la motivación del equipo de trabajo y los potenciales impactos del producto desarrollado además de la explicación textual de la solución robótica propuesta y detallada, un diagrama esquemático de la misma, y con ello, la explicación de los modelos cinemáticos directos e inversos y por último, la visualización del problema y de la solución usando Rviz y de forma simultánea, utilizando Gazebo. Todo ello, organizado en un cronograma en el que se explicarán las actividades realizadas para el dispositivo y logros alcanzados, semana a semana, durante el desarrollo del proyecto además de la explicación explícita del trabajo independiente realizado: donde se desatacarán las fuentes consultadas y procedimientos realizados en aras de fundamentar la habilidad del equipo de trabajo para identificar, formular y resolver problemas complejos de la ingeniería aplicando principios de ingeniería, ciencias y matemáticas donde se tiene como finalidad, lograr no solo adquirir y aplicar nuevos conocimientos cuando es requerido, usando estrategias apropiadas de aprendizaje sino también, hacer el proceso de realizar la búsqueda de información y emplear adecuadamente la información consultada para llevar a cabo una solución óptima respecto al contexto del problema y realizar todo el proceso de Gestión del innovación, Diseño Mecatrónico y de la asignatura de robótica pues la integración de los diferentes conocimientos adquiridos para llevar a cabo un prototipo funcional, es el procedimiento mediante el cual, muchas industrias o proyectos de investigación es lo que se realiza para la generación de dispositivos electrónicos o robóticos de alta calidad en aras de solucionar problemas desde la ingeniería generando un gran impacto social y a veces, sostenible con los recursos.

FORMULACIÓN DEL PROBLEMA, MOTIVACIÓN Y POTENCIALES IMPACTOS DEL PRODUCTO DESARROLLADO

Para llevar a cabo la *Formulación del Problema*, se tuvo en cuenta la información obtenida en el desarrollo de la fase de Inmersión desde la asignatura de Gestión de la Innovación donde fueron realizadas diferentes encuestas, shadowing y diarios de campo a diferentes agricultores y personas del común del corregimiento de Rozo, Valle del Cauca. Luego de obtener dicha información, se toma la decisión de tener un enfoque en el agricultor Deivy Mañozca, residente del corregimiento, específicamente de La Torre, el cual es el

principal doliente de su problemática. De acuerdo con la información obtenida, la problemática corresponde a la falta de control de la reproducción de los hongos en los cultivos de maracuyá en el corregimiento de Rozo, Valle del Cauca, Colombia en temporadas de invierno, la cual ha ocasionado la perdida de alimentos y de rentabilidad en el proceso de siembra del fruto.

La *motivación* para afrontar dicha problemática social consta de que en el país se pierde y se desperdician en promedio, 9.76 millones de toneladas de alimentos, que representa el 34% del total producido y se tiene la intención de tomar acción en ello, puesto que, de las perdidas, el 64% corresponde a las ocasionadas en las etapas de producción, post cosecha, almacenamiento y procesamiento industrial y, específicamente el sector agropecuario y/o campesino cuenta con un porcentaje del 40.5 % de desperdicio, lo cual requiere que se tenga un enfoque específico con el objetivo de mitigar, reducir o disminuir paulatinamente ese índice o tasa de pedidas en los alimentos debido a que para muchas personas, no es un secreto que gran parte de los agricultores en general, cuyo modelo de negocio es la venta de productos caseros o sin procesamiento excesivo o tecnológico, esta es la forma más humilde de supervivencia. [2]



Delimitando más la motivación del equipo de trabajo, se tiene que el enfoque estará encaminado los Objetivos de Desarrollo Sostenible 11 y 12. Específicamente a las metas 12.2, la cual indica que de aquí a 2030, se pueda lograr la gestión sostenible y el uso eficiente de los recursos naturales y la 12.3 que indica que de aquí a 2030, se pueda reducir a la mitad el desperdicio de alimentos per cápita mundial en la venta al por menor y a nivel de los consumidores y reducir las pérdidas de alimentos en las cadenas de producción y suministro, incluidas las pérdidas posteriores a la cosecha. Así, lo que se quiere lograr es la reducción del porcentaje de pérdidas del fruto, hasta que este sea máximo el 15%, pues se quiere obtener más ganancias y que la utilidad del proceso de cultivación del fruto pueda aumentar su rentabilidad, pues según la inmersión realizada, casi siempre, se pierden en promedio, el 20% o 30% de la cosecha, lo que equivale a un promedio de 45 kilos de maracuyá, ya que el agricultor obtiene 300 kg por cada trimestre del año.

No obstante, debido a que este proceso es algo nuevo para el equipo de trabajo, de forma preliminar, se realizó una investigación para apropiarse más del contexto y según la información de las entrevistas, solo es posible hacer el control

SCIENCE ROBOTICS | SPLINKER ROBOT

del hongo mediante la aspersión de fungicida que no tenga ningún agente contaminante con el ambiente, pues es vital para el agricultor la presencia de polinizadores como lo son las abejas y los abejorros. De esta forma, las personas implicadas en el procedimiento corresponden al dueño del terreno (arrendador), el agricultor, los clientes del agricultor, las personas encargadas de realizar el mantenimiento y la

aspersión del fungicida, la/s persona/s encargada/s de el empaque y la distribución del producto por venta al por menor a empresas o clientes por lotes y, por último, pero no menos importante, la persona encargada del riego del cultivo. Así como también, los proveedores del fungicida y Suministro de Agua, Suministro de empaques y bolsas para frutos y demás.

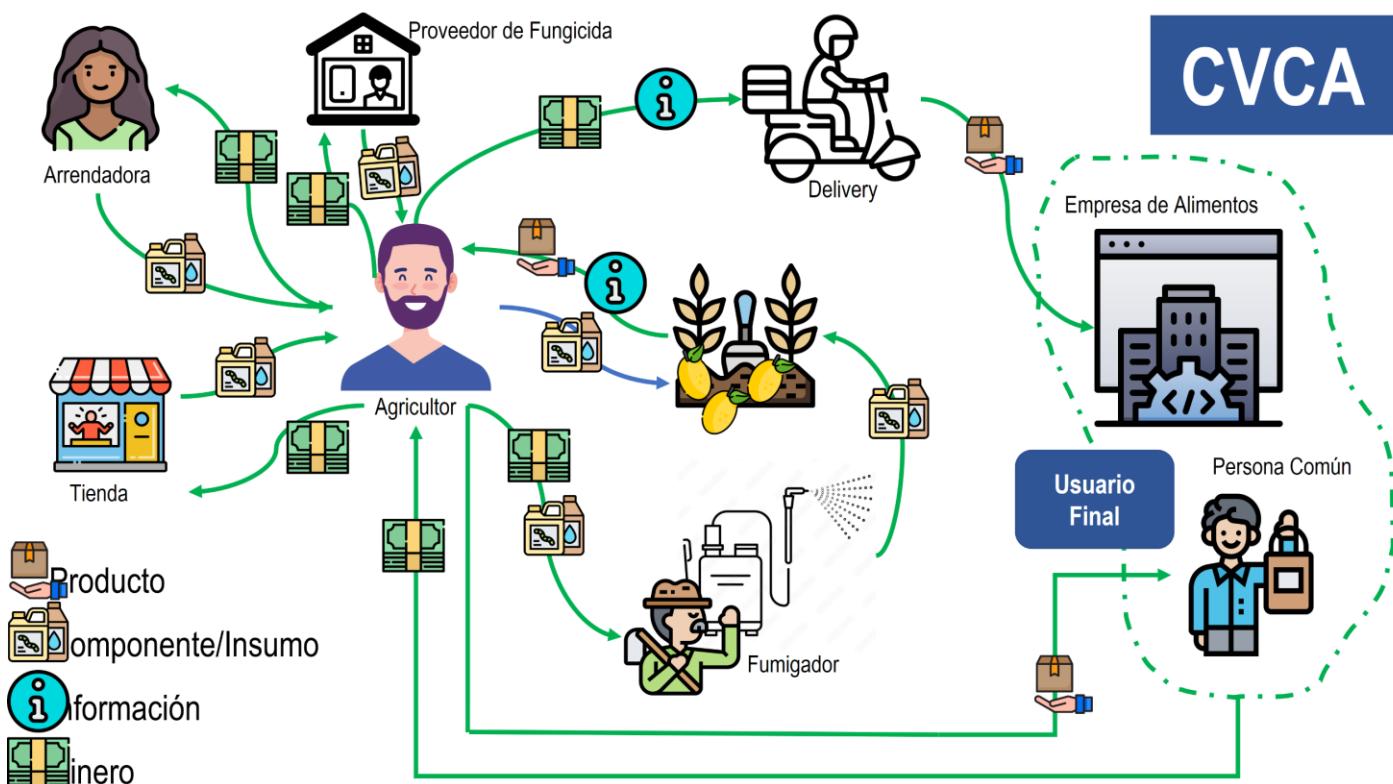


Ilustración 1. CVCA: Customer Value Chain Analysis

Tabla 1. Planteamiento de la misión

Descripción del producto	Mecanismo mediante el cual se realice el riego de los cultivos ubicados terrenos planos
Propuesta de valor	<ul style="list-style-type: none"> - Optimizar el proceso de fumigación/aspersión para los cultivos de Maracuyá. - Reducir el porcentaje de pérdidas de los frutos y hojas durante el proceso de germinación y extracción del fruto.
Mercado primario	Agricultores y Empresas de alimentos
Mercado secundario	El mercado corresponde a Dueños de viveros, Usuarios que tengan una extensa vegetación en sus residencias y Gerentes del sector agrario
Agentes implicados	<ul style="list-style-type: none"> · Clientes nuevos y existentes · Principales minoristas · Marketing, ventas e innovación

SCIENCE ROBOTICS | SPLINKER ROBOT

Teniendo en cuenta la composición que debería tener el robot, el equipo evalúa los impactos sociales del proyecto teniendo en cuenta las 8 categorías presentadas en el desarrollo del curso de Diseño Mecatrónico. Para ello es importante traer a coalición el CVCA mostrado anteriormente donde se destaca las personas más importantes implicadas en el procedimiento de forma directa e indirecta y la relación del contexto del problema para con ellos, además de sus características y la presencia de los proveedores de fungicida y Suministro de Agua, Suministro de empaques y bolsas para frutos y demás.



Ilustración 2. ítems del Impacto Social del Proyecto.

Teniendo en cuenta lo anterior, es posible como equipo de trabajo, indicar que el proyecto estará enfocado a diferentes componentes sociales que permitirán ver, desde diferentes perspectivas, el impacto social generado ya sea desde un componente de desarrollo social, calidad de vida, objeción en pro de la comunidad y demás. Para ello, se tiene:

Employability. En el caso específico del proyecto, se adapta este concepto para afianzar y determinar cuántos empleos está afectando tanto positiva y negativamente en la sociedad, el desarrollo del proyecto. Para ello, el equipo de trabajo se ha enfocado en dos principales situaciones: implementación y puesta en marcha del proyecto (*Ran Up*) y la otra situación es en lo necesario para su *desarrollo*. En el caso de la primera, es necesario la intervención de los siguientes vacantes o personal de trabajo, desde lo económico:

- Diseñador Gráfico (cantidad 1), el cual realizará la planificación, dirección y el control de las producciones para las ventas a empresas digitalmente, además de la edición de contenidos multimediales y la creación del logotipo y el eslogan de la empresa de manera eficiente y acorde con las necesidades del mercado.
- Mano de obra para Equipos de Oficina (cantidad 1), el cual será el encargado de la gestión, distribución

de los espacios de la oficina, así como también de la adecuación de los equipos electrónicos, maquinaria pesada y demás.

- Mercadólogo (cantidad 1), el cual será dispuesto para realizar la búsqueda, identificación y explotación de las oportunidades de la empresa para mejorarla contra la competencia además de llevar a cabo las relaciones públicas y tratar con los clientes directos para crear alianzas estratégicas o potenciales compras.
- Publicista (cantidad 1), el cual será dispuesto para la planificación de campañas de publicidad, supervisión de la creación de anuncios y direccionamiento de las ideas entre los equipos multidisciplinarios.

Ahora, desde el aspecto práctico:

- Ensamblador/es, cantidad 4, para el caso del proyecto serán los mismo estudiantes o diseñadores del concepto de Splinker Robot.
- Proveedor de Insumos electrónicos y agropecuarios (empresa), los cuales corresponden a cada una de las partes y/o los elementos necesarios para llevar a cabo la construcción del mínimo producto viable del proyecto.
- Proveedor de Servicios Públicos (empresa), básicamente para tener presente el aporte que genera la empresa prestadora de servicios de energía e internet.
- Tercerización del Corte láser (empresa), donde el equipo de trabajo recurrirá para realizar el molde, los cortes y la disposición de las diferentes piezas del robot.
- Programador o Ingeniero Informático (cantidad 1), el cual realizará el desarrollo de los scripts en Python o lenguaje C del robot, además de la seguridad informática del dispositivo.

Good Health and Well-Being. En el caso específico del proyecto, se adapta este concepto para analizar cuántas enfermedades a largo plazo se están previniendo con el desarrollo del proyecto. Para ello, el equipo de trabajo tomará en cuenta la información tomada del documento de Enfermedades Profesionales de los Agricultores [3] propuesta por la Comisión Nacional de Seguridad y Salud en el Trabajo de España donde se indica que el ambiente de trabajo de los agricultores conlleva la exposición de estos a riesgos físicos asociados al clima, el terreno, los incendios y la maquinaria; riesgos químicos asociados a los plaguicidas, fertilizantes y combustibles; riesgos biológicos que incluyen la exposición a polvo (orgánico e inorgánico) y alergenos. También se tiene en cuenta el contacto con plantas, animales e insectos; riesgos ergonómicos y psicosociales, como la manipulación manual de cargas, posturas forzadas, movimientos repetidos, y una organización de trabajo con una gran variedad de peligros para la salud, en particular las muchas horas de trabajo. Algunas de dichas características específicas corresponden a:

SCIENCE ROBOTICS | SPLINKER ROBOT

- La mayoría de las tareas se desarrollan al aire libre, exponiendo a los trabajadores a condiciones climáticas de frío y/o calor extremo que, además, hacen muy difícil controlar la seguridad y salud en el trabajo.
- Empleo y contacto con productos químicos y biológicos.
- El contacto con animales y plantas que expone a los trabajadores a mordiscos, envenenamientos, infecciones, enfermedades parasitarias, alergias, toxicidad y otros problemas de salud.
- El tipo de posturas del trabajo y la duración de las tareas a realizar.

Tabla 2. Tabla de enfermedades tomada de la Subdirección General de Estadísticas Sociales y Laborales del Ministerio de Trabajo en España.

Enfermedades	Hombres		Mujeres		Total	
	Nº de casos	incidencia	Nº de casos	incidencia	Nº de casos	incidencia
Infecciosas	10	3,1	0	0,0	10	2,2
Neurológicas	38	11,8	30	23,6	68	15,1
De los órganos de los sentidos	5	1,5	0	0,0	5	1,1
Respiratorias	5	1,5	2	1,6	7	1,6
Cutáneas	35	10,8	27	21,3	62	13,8
Osteomusculares	199	61,6	141	111,0	340	75,5
Total	311	90,3	200	157,5	492	109,3

Teniendo en cuenta la referencia anterior, se realizará un análisis detallado y especificado del proyecto. En este caso, se analizará el comportamiento de la salud de las personas en los cultivos de maracuyá donde su principal doliente, Deivy Mañozca, el cual es el encargado de realizar casi todas las actividades relacionadas con el cultivo, será tomado como referencia de análisis, así como también los demás agricultores del sector de Rozo Valle del Cauca cuyas labores corresponden al trabajo pesado de sus propios cultivos. Para el análisis, se tienen en cuenta los tres principales ítems de enfermedades presentadas anteriormente:

En el caso de las *infecciones*, se tiene que en el momento de reducir la producción de hongos y presencia de insectos en el cultivo lo que se está haciendo directamente es la mitigación disminución o asentamiento de la curva de contacto presentada entre las personas que interactúan con el cultivo y las plagas naturales de las plantas, debido a que en la información tomada en la fase de inmersión se obtuvo el dato de que no todos los participantes activos en el momento de cultivar, poseen guantes o viseras de protección, por lo tanto esto causa enfermedades en la piel o en infecciones bacterianas como lo es la brucelosis cuando se combina frutas con hongos mezcladas con los lácteos, la cual genera una sintomatología evidenciada en dolor de las articulaciones y los

músculos, fiebre, pérdida de peso y fatiga. Algunas personas también presentan dolor de vientre y tos.

En el caso de las *respiratorias*, también existe una relación similar que la del caso anterior, la cual es causada por la exposición de las personas cuando existen temporadas de lluvia excesivas o cambios abruptos en el clima en conjunción con la humedad y debido a ello, la exposición a los cambios rápidos y drásticos entre el calor y frío, lo cual ha causado la generación de asma profesional, que es un trastorno pulmonar en el cual sustancias que se encuentran en el lugar de trabajo provocan que las vías respiratorias de los pulmones se inflamen y se estrechen. Esto lleva a que se presenten ataques de sibilancias, dificultad respiratoria, sensación de opresión en el pecho y tos. Ahora, en el momento en el que el robot empiece a funcionar, la persona no realizará directamente el trabajo así mismo no tendrá que someterse a dichos cambios abruptos.

El trabajo de aspersión es realizado básicamente tres veces por semana, tres secciones, es decir, aproximadamente 9 sesiones por semana donde básicamente se realiza durante una hora, máximo dos, haciendo que la persona deba estar en promedio, 18 a 20 horas semanales bajo un trabajo pesado que consta del transporte en la espalda de una motobomba, sumado a ello el peso del fungicida y la sujeción de la manguera y el aspersor manual. Esto ha causado la generación

SCIENCE ROBOTICS | SPLINKER ROBOT

de enfermedades *osteomusculares* tales como lesiones inflamatorias o degenerativas de los músculos, tendones, articulaciones, ligamentos y nervios. Generalmente localizados en el cuello, espalda, hombros, codos, muñecas y manos. De esta forma, cuando el dispositivo se implemente, estas enfermedades también se reducirán paulatinamente debido a que el robot es el que realizará el trabajo pesado y continuo. Ahora, cómo es una máquina solamente habría que realizar una lubricación y refrigeración de este y su respectiva limpieza, lo cual podría también generar un conflicto porque reduciría el trabajo de cierta cantidad de personas, pero realmente lo que se está beneficiando es la salud del trabajador y directamente su calidad de vida. En este sentido, será un dispositivo que podría implementarse en todos los cultivos de maracuyá o de otras frutas que se den en forma de cascada.

Teniendo en cuenta la información anterior ya delimitada, el impacto social causado por la implementación del proyecto es positivo. Sin embargo, no se da veracidad de algo utópico en cuanto a que solo genera trabajo porque para algunas personas puede reducir labor humana pero también recorte de personal, ya que la trascendencia de realizar el procedimiento de forma arcaica a hacerlo de forma tecnificada es un proceso de cambio no solo tecnológico sino algo que ha estado arraigado culturalmente en Colombia, específicamente a la generación de personas adultas que no están totalmente relacionado con la tecnología; pero, si se observa desde la retribución monetaria y el apoyo en cuestión a la salud que genera el dispositivo, se podría asegurar a priori que su retribución para la sociedad es positiva en cuanto al desarrollo del proyecto.

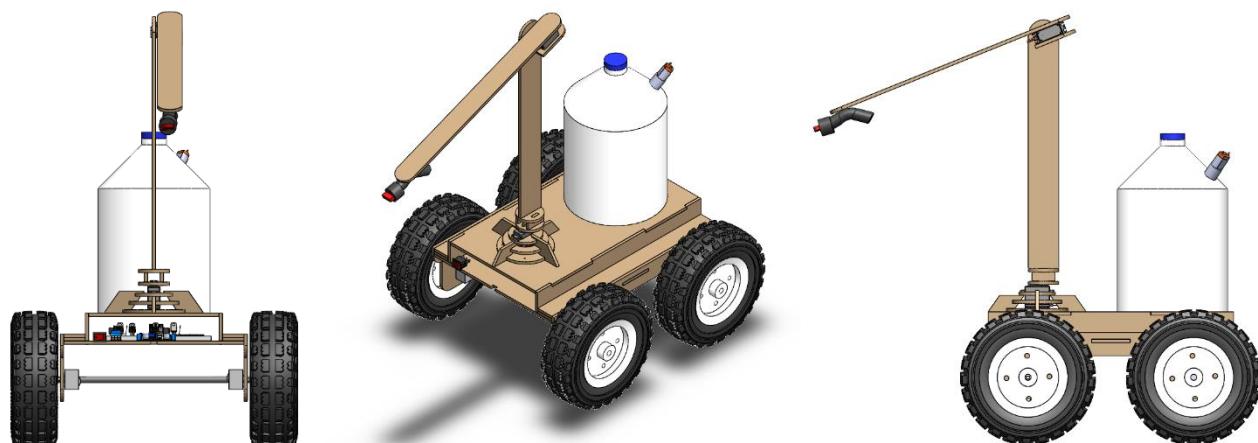


Ilustración 3. Modelo renderizado del Splinker Robot

EXPLICACIÓN TEXTUAL DE LA SOLUCIÓN PROUESTA Y UN DIAGRAMA ESQUEMÁTICO DE LA MISMA

En aras de realizar la explicación de la solución propuesta, se tiene en cuenta la Ilustración 3, donde se declara al Splinker Robot como un dispositivo con Arquitectura Modular. Para justificar ello, son planteados tres módulos generales que cobijan los elementos físicos y sus funciones.

El primer módulo trata sobre el sistema de movimiento, en este se definen los objetos que debe tener el producto para dar cumplimiento con la etapa de cinemática del robot, aquí se presentan las siguientes subfunciones: El chasis, este debe ser robusto, capaz de sostener el tanque, circuitos, motores, además debe brindar estabilidad y seguridad en el sistema. Adicionalmente, se tiene presente las llantas, las cuales deben cumplir ciertos requerimientos debido a la zona a la cual el robot será sometido. Por último, se tiene los motores conectados a las llantas, estos serán controlados con el objetivo de generar el desplazamiento, es decir, ejecutar la cinemática del robot.

El segundo módulo consiste en el sistema de aspersión el cual es fundamental puesto que es el núcleo de lo que se desea

realizar con el producto. Para lo cual es necesario definir las subfunciones, las cuales constan de almacenar el fungicida, distribuir el mismo desde el tanque hasta el pulverizador, brindar la presión necesaria para transportar el fluido y, por último, pulverizar el fungicida, subfunciones que son realizadas mediante el tanque, mangueras, una motobomba casera y el efecto final correspondiente a la boquilla pulverizadora, respectivamente.

El tercer módulo trata sobre las conexiones que existen con el microcontrolador, este será programado para la correcta ejecución de los actuadores y sensores. En este módulo se tiene en cuenta el sensor de proximidad, el cual es capaz de detectar obstáculos en la parte delantera y trasera del robot, sirve para prevenir choques. Por otra parte, se tendrá el sensor de nivel de líquido, que estará ubicado dentro del tanque, este brindará información al usuario sobre la cantidad de líquido y alertará al usuario cuando se esté agotando el fungicida. Adicionalmente, se tendrán los servomotores, estos ayudarán a controlar el brazo robótico por medio de las *articulaciones revolutas*. Por último, en esta etapa entran los componentes eléctricos y electrónicos, como lo son la batería, integrados, módulos, entre otros.

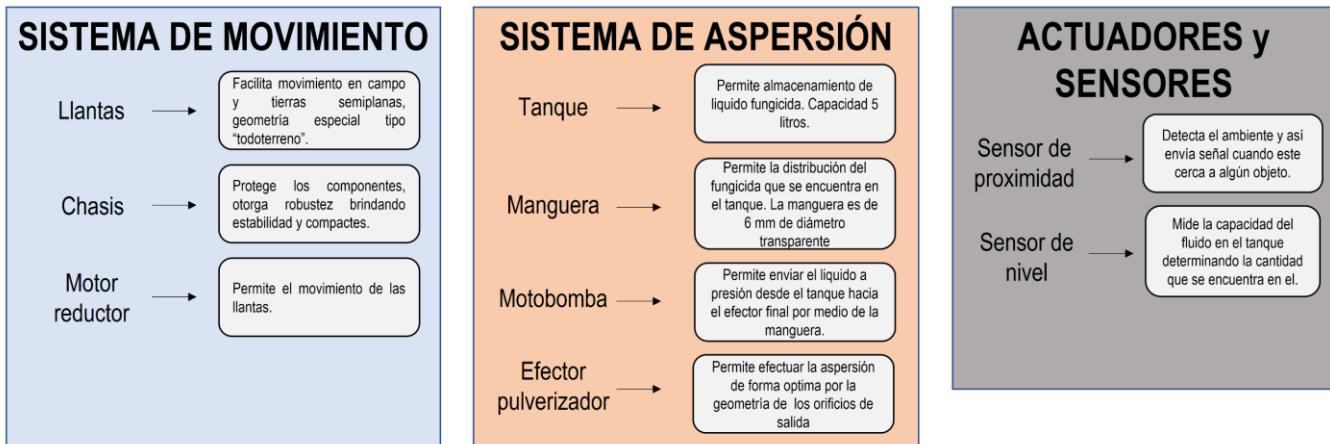


Ilustración 4. Arquitectura de Módulos

Ahora, desde la robótica, el equipo de trabajo compone su solución en dos apartados: Robot Manipulador y Robot Móvil. En el componente manipulador, es utilizado en contexto para reemplazar el trabajo manual y electromecánico que realizan los agricultores en el momento de realizar la aspersión del fungicida. En este caso, será utilizado para alcanzar posiciones y orientaciones por los senderos del cultivo de maracuyá del agricultor. Por dicha razón, también se opta por el uso de un aplicativo móvil para realizar de forma básica y sencilla, la teleoperación simulada del manipulador.



Ilustración 5. Motobomba de aspersión de fungicida

El Splinker Robot estará fabricado, con una perspectiva desde el **Brazo Manipulador** con dos articulaciones y la base del chasis, sus nombres correspondientes son: link_1, link_2 y base_link. Cada eslabón está formado de tablones de MDF y los grados de libertad son dispuestos por servomotores que, a su vez, son controlados por el Arduino.

La unión entre el base_link y en link_1 está dada por la articulación revoluta joint_1. Asimismo, La unión entre el link_1 y el link_2 está dada por la articulación revoluta joint_2. Con ambos eslabones es que se logra la conformación del brazo robótico, el cual puede ejercer, a priori, el mismo movimiento que realiza la sujeción entre el brazo de la motobomba y la mano de la persona. La armadura podrá realizar las tareas que sean indicadas por el agricultor, evitando así las diferentes enfermedades mencionadas en el impacto social tanto para los asuntos climáticos como para las

enfermedades osteomusculares provocadas por el trabajo pesado y continuo. Así, los Joints mencionados del robot permiten tener los dos grados de libertad del dispositivo, siendo el primero el que realiza un movimiento de yaw, y se tiene el segundo (de abajo hacia arriba) que permite realizar el movimiento de pitch. La altura del brazo es aproximadamente 1.05 m y por tal motivo puede tener un alcance para frutos que estén comprendidos entre los 0-1.1 m. Además, cuenta con el aspersor clásico ubicado en la parte final del segundo eslabón que, para efectos del curso, corresponde al Efecto Final. Ahora, para la parte del **Robot Móvil**, se dispone de 4 llantas estándar fijas, ubicadas y alineadas de tal forma, que las llantas traseras sean con tracción y las delanteras, sean dispuestas por individual mediante la sujeción mecánica de los motorreductores para que, a cambios de velocidad, pueda realizar el giro en contra y a favor de las manecillas del reloj y que, además, pueda ir hacia adelante o hacia atrás dependiendo de lo que el usuario desee indicarle. En este caso se tienen las articulaciones front_jr, front_jl, rear_jr y rear_jl, que representan la unión de cada una de las llantas con el chasis o la base del Splinker Robot.

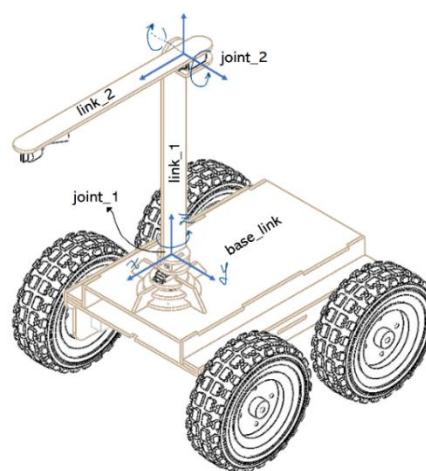


Ilustración 6. Manipulator Arm: Rool, Pitch & Yaw

SCIENCE ROBOTICS | SPLINKER ROBOT

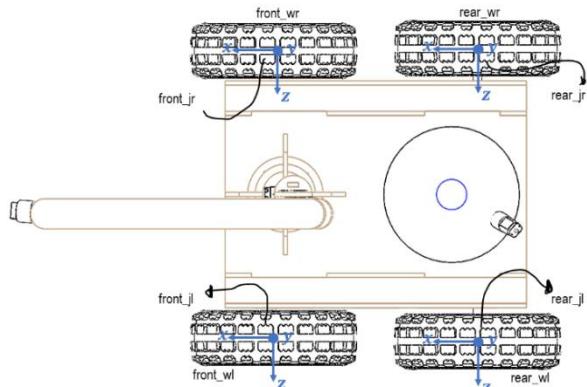


Ilustración 7. Mobile Robot: vista desde planta

Elementos	Cantidad	Precio		Total	
		Mínimo	Máximo	Mínimo	Máximo
Batería	Batería 12 V	1	13500	60000	13500
Lógica	Arduino Atmega 2560	1	0	0	0
Sensores	Sensor de Nivel	1	2000	15000	2000
	Sensor de Proximidad	2	6000	15000	12000
Módulos	Nivel de Batería (LM3915)	1	5000	7000	5000
	Regulador de tensión lineal (LM317)	1	3100	9000	3100
	Controlador de alta corriente (Puente H - L293D -L298N)	1	5000	13000	5000
Actuadores	Motor DC	1	4000	9000	4000
	Motorreductor	2	40000	55000	80000
	Servomotor	2	10000	34000	20000
Comp. Eléctricos	Resistencias	1	5000	7000	5000
	LEDs	1	5000	11000	5000
	Transistores	1	3000	20000	3000
Otros	Mangueras	1	0	0	0
	Boquilla pulverizadora	1	11000	40000	11000
	Tanque	1	0	0	0
	Cableado o Jumpers x40	2	10000	10200	20000
	Material Chasis	1	0	0	0
	Llantas	4	0	0	0
Total				188600	419400

Ilustración 8. Elementos tenidos en cuenta para su construcción para el prototipo funcional en Diseño Mecatrónico.

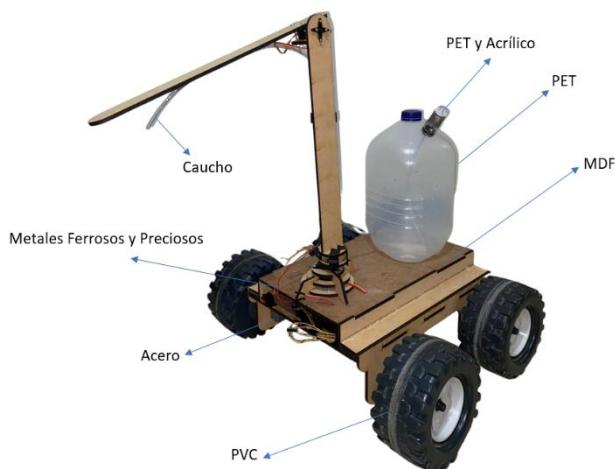


Ilustración 9. Materiales utilizados

La siguiente tabla corresponde a los costos que se han invertido en el desarrollo del modelo físico 3D actualizado, el cual será ilustrado a continuación con una vista isométrica y también, al final del documento con las diferentes dimensiones más significativas y el plano formal del mismo realizado en SolidWorks. La información básica de su composición interna consta de:

SCIENCE ROBOTICS | SPLINKER ROBOT

Para tener una mayor compresión de la solución propuesta, se tiene el funcionamiento del robot en función de las señales de control que digitaría el usuario cuando utilice el aplicativo.

Por dicha razón, la secuencia de pasos a seguir para el usuario corresponde a:

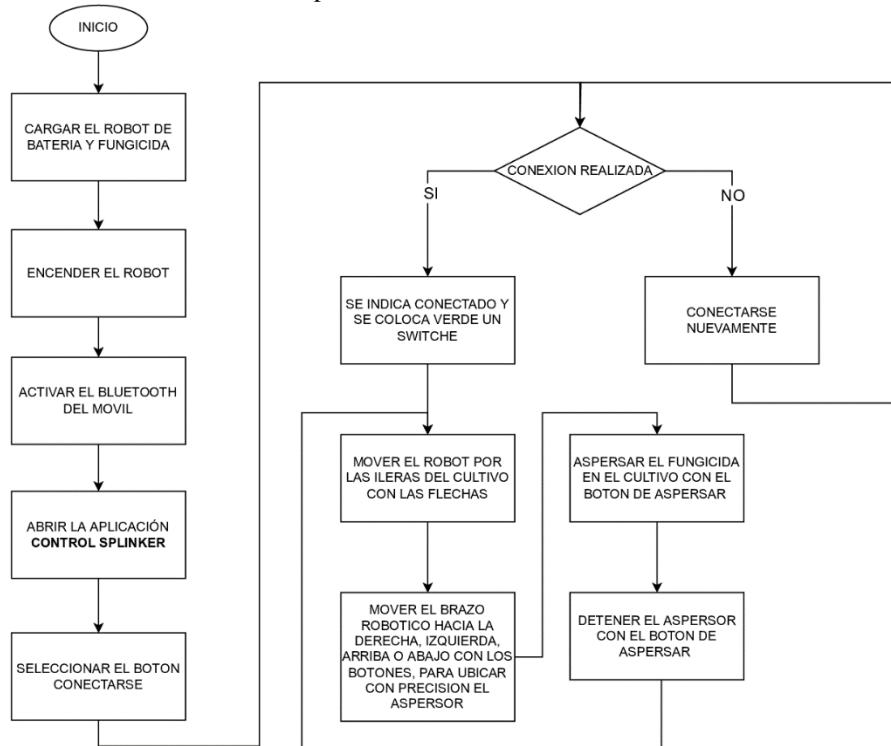


Ilustración 10. ¿Como utilizar el dispositivo?

Luego de ello, se tiene la descomposición funcional de la disposición que será explicada brevemente por la siguiente ilustración, pero también sustentada en el árbol de articulaciones y eslabones correspondientes a la morfología del robot (Anexo 1). En la siguiente imagen, se tiene información relacionada a la construcción del prototipo funcional donde se tiene la disposición comunicativa de cada uno de los componentes más significativos del robot,

incluyendo en él, el control remoto y el aplicativo realizado en App Inventor, que es la forma en la que se le envían, vía bluetooth, las señales de control de movimiento y aspersión. Además, se tiene también la disposición de un interruptor, el cual evita que se tenga que exponer a la intemperie los diferentes componentes electrónicos cada vez que sea necesario ejecutar el sistema.

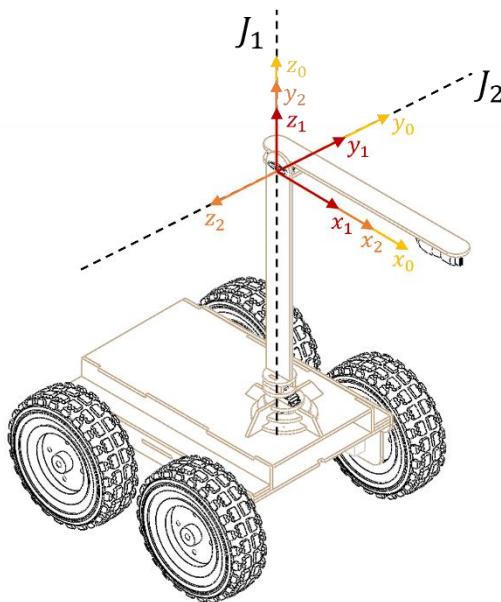


Ilustración 11. Elementos funcionales con descripción

SCIENCE ROBOTICS | SPLINKER ROBOT

MODELOS CINEMÁTICOS DIRECTO E INVERSO

En este inciso se realizará el desarrollo matemático del cálculo de los modelos cinemáticos directos e inversos para el robot, tanto de su brazo manipulador como para la velocidad del chasis. De esta forma, para el cálculo del modelo cinemático directo del brazo manipulador, se inicia por realizar la matriz DH con sus respectivas convenciones vistas durante el desarrollo del curso, todo ello se evidenciará mediante la siguiente grafica.



A partir de este, se realiza el cálculo de los valores de los parámetros de la tabla, presentada a continuación. Estos valores serán remplazados en la siguiente ecuación con el fin de obtener las matrices intermedias, que serán posteriormente multiplicados para obtener la pose del efecto final con respecto a la base. Vale la pena resaltar, que debido a que solo se presentan articulaciones rotacionales, el parámetro que varía es theta.

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	90°	0	0	θ_2

$${}^{i-1}{}_i T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1} d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

De esta manera, mediante Python y con los parámetros ya remplazados, se procede a calcular la multiplicación de las

matrices intermedias, de la cual se obtiene el modelo cinemático directo, correspondiente a la pose del efecto final en función de las variables articulares.

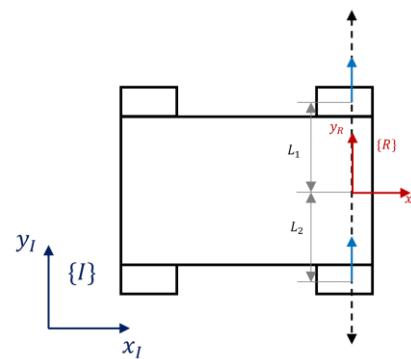
$${}^0 {}^2 T = \begin{bmatrix} \cos \theta_1 \cdot \cos \theta_2 & -\cos \theta_1 \cdot \sin \theta_2 & \sin \theta_1 & 90 \cdot \cos \theta_1 \\ \cos \theta_2 \cdot \sin \theta_1 & -\sin \theta_1 \cdot \sin \theta_2 & -\cos \theta_1 & 90 \cdot \sin \theta_1 \\ \sin \theta_2 & \cos \theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ahora, con el modelo cinemático directo obtenido, se procede a calcular el modelo cinemático inverso, con el cual se obtiene el valor de las variables articulares en función de la pose deseada del efecto final. De este modo, se iguala la matriz de transformación homogénea de 2 con respecto a 0 con la matriz deseada simbólica y se procede a despejar theta1 y theta2 en función de la deseada.

$$\theta_1 = \text{Atan2}\left(\sqrt{1 - \left(\frac{Px}{90}\right)^2}, \frac{Px}{90}\right)$$

$$\theta_2 = \text{Atan2}\left(\sqrt{1 - \left(\frac{r11}{\cos \theta_1}\right)^2}, \frac{r11}{\cos \theta_1}\right)$$

De forma análoga, se realiza el cálculo de la cinemática directa para la parte móvil del robot, la cual entrega o predice la velocidad del chasis a partir de la velocidad de las llantas. Para este, se debe tener en cuenta que el robot a pesar de contar con 4 llantas estándar, se considerara como un robot diferencial puesto que las llantas delanteras presentan tracción, mientras que, las traseras no presentan tracción, ni son direccionables. Dando claridad de lo anterior, se realiza un diagrama el cual permite percibir las convenciones tomadas y el cálculo de los parámetros. Por otra parte, para los valores de L, se realiza la medición de las distancias mediante la aplicación de renderizado SolidWorks.



Teniendo en cuenta lo anterior, se procede a calcular los valores de Alpha y Beta para cada una de las llantas, con el fin de obtener las restricciones de rodadura y de deslizamiento. A partir de estas restricciones se obtienen las matrices de J1, J2 y C1, las cuales serán unidas posteriormente, en una única ecuación y matriz que entregue la cinemática directa.

SCIENCE ROBOTICS | SPLINKER ROBOT

Llanta	α_i	β_i	l_i	r
<i>L</i>	90	0	190	
<i>R</i>	-90	180	190	85

$$j_1 = \begin{bmatrix} 1 & 0 & -190 \\ 1 & 0 & 190 \end{bmatrix}; j_2 = \begin{bmatrix} r & 0 \\ 0 & r \end{bmatrix}; C_1 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Con el cálculo de las matrices se procede a plantear la ecuación única mencionada anteriormente, con la cual a partir del cálculo de la inversa de la matriz de proyecciones y multiplicarla por la matriz de velocidades y de rotación, se obtiene el modelo cinemático directo con respecto al marco de referencia global I, presente en el último apartado.

$$\dot{\xi}_I = \begin{bmatrix} \frac{85 \cdot \varphi_L}{2} + \frac{85 \cdot \varphi_R}{2} \\ 0 \\ \frac{17 \cdot \varphi_L}{76} - \frac{17 \cdot \varphi_R}{76} \end{bmatrix}$$

VISUALIZACIÓN DEL PROBLEMA Y DE LA SOLUCIÓN USANDO RVIZ

Para el desarrollo de este inciso, se debe tener en cuenta que el urdf/xacro que fue utilizado para la primera entrega del proyecto, no ha sufrido cambios. Por lo tanto:

```

import os
from launch import LaunchDescription
from launch.actions import DeclareLaunchArgument
from launch.conditions import IfCondition, UnlessCondition
from launch.substitutions import Command, LaunchConfiguration
from launch_ros.actions import Node
from launch_ros.substitutions import FindPackageShare

def generate_launch_description():

    # Set the path to different files and folders.
    pkg_share = FindPackageShare(package='robotuao2_description').find('robotuao2_description')
    default_launch_dir = os.path.join(pkg_share, 'launch')
    default_model_path = os.path.join(pkg_share, 'models/robotuao2_urdf.xacro')
    robot_name_in_urdf = 'robotuao2'
    default_rviz_config_path = os.path.join(pkg_share, 'rviz/urdf_config.rviz')

    # Launch configuration variables specific to visualization
    gui = LaunchConfiguration('gui')
    model = LaunchConfiguration('model')
    rviz_config_file = LaunchConfiguration('rviz_config_file')
    use_robot_state_pub = LaunchConfiguration('use_robot_state_pub')
    use_rviz = LaunchConfiguration('use_rviz')
    use_sim_time = LaunchConfiguration('use_sim_time')

    # Declare the launch arguments
    declare_model_path_cmd = DeclareLaunchArgument(
        name='model',
        default_value=default_model_path,
        description='Absolute path to robot urdf file')

```

```

# Publish the joint state values for the non-fixed joints in the URDF file.
start_joint_state_publisher_cmd = Node(
    condition=UnlessCondition(gui),
    package='joint_state_publisher',
    executable='joint_state_publisher',
    name='joint_state_publisher')

# A GUI to manipulate the joint state values
start_joint_state_publisher_gui_node = Node(
    condition=IfCondition(gui),
    package='joint_state_publisher_gui',
    executable='joint_state_publisher_gui',
    name='joint_state_publisher_gui')

# Subscribe to the joint states of the robot, and publish the 3D pose of each link.
start_robot_state_publisher_cmd = Node(
    condition=IfCondition(use_robot_state_pub),
    package='robot_state_publisher',
    executable='robot_state_publisher',
    parameters=[{'use_sim_time': use_sim_time,
                 'robot_description': Command(['xacro', model])}],
    arguments=[default_model_path])

# Launch RViz
start_rviz_cmd = Node(
    condition=IfCondition(use_rviz),
    package='rviz2',
    executable='rviz2',
    name='rviz2',
    output='screen',
    arguments=['-d', rviz_config_file])

```

En las imágenes anteriormente ilustradas, se logra apreciar el archivo *launch*, el cual consta de unas direcciones para importar los paquetes. Para el caso del robot es la última modificación con sus respectivos sensores; a su vez, cuenta con el GUI para poder manipular los valores de forma manual publicando en el *joint_state_publisher*. Adicionalmente, presenta la ejecución del Rviz con sus respectivas acciones de descripción.

```
jswan@ubuntu:~/dev_ws$ ros2 topic list
/clicked_point
/clock
/cmd_vel
/dynamic_joint_states
/goal_pose
/initialpose
/joint_states
/joint_trajectory_controller/joint_trajectory
/joint_trajectory_controller/state
/parameter_events
/performance_metrics
/robot_description
/rosout
/tf
/tf_static
/wheel/odometry
```

SCIENCE ROBOTICS | SPLINKER ROBOT

SIMULACIÓN DEL PROBLEMA Y DE LA SOLUCIÓN USANDO GAZEBO

```
<!-- world link -->
<link name="world"/>

<link
  name="base_link">
  <inertial>
    <origin
      xyz="5.8016E-17 -0.0024966 -0.0040213"
      rpy="0 0 1.5" />
    <mass
      value="3.4068" />
    <inertia
      ixz="0.053563"
      ixy="8.889E-19"
      ixz="-1.2755E-18"
      ixy="0.080978"
      iyz="0.00068442"
      izz="0.032409" />
  </inertial>
  <visual>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="$(find robotuao2_description)/meshes/base_link.STL" />
    </geometry>
    <material
      name="" />
  </visual>
<!-- base_link and its fixed joint -->
<joint name="base_joint" type="fixed">
  <parent link="base_link"/>
  <child link="world"/>
</joint>

<link
  name="link_1">
  <inertial>
    <origin
      xyz="4.3399E-17 0.27928 -3.6082E-16"
      rpy="0 0 0" />
    <mass
      value="0.62129" />
    <inertia
      ixz="0.016962"
      ixy="-1.8336E-19"
      ixz="6.6174E-21"
      ixy="0.00029804"
      iyz="-3.4701E-19"
      izz="0.016901" />
  </inertial>
  <visual>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
```

En las imágenes anteriormente mostradas, las cuales hacen parte del xacro (que es donde se encuentra toda la descripción del robot y su configuración), es posible observar que para poder modelar el controlador de la articulación junto con su trayectoria, se debe crear un link el cual estará vacío y se llamará *world* y a su vez, como se logra apreciar en la segunda imagen, se debe codificar una articulación llamada *base_joint*, la cual es de tipo fija vacía pero que involucra las características de que el link padre, es *base_link* dado que el brazo está sujeto al chasis y el hijo, es el mundo (*world*) vacío anteriormente creado.

```
<limit
  effort="200"
  velocity="5" />
```

Cabe destacar que para efectos del controlador en el archivo xacro, todas las articulaciones o los tags joints deben tener límites para así prevenir daños o molestias en la simulación, los límites que estos deben tener son esfuerzo con un valor de 200 y velocidad de 5 en el movimiento del brazo.

```
<!-- ***** SONAR SETUP ***** -->
<link name="sonar_link">
  <inertial>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <mass value="0.125" />
    <inertia ixz="0.001" ixy="0" ixz="0.001" iyy="0.001" iyz="0.001" izz="0.001" />
  </inertial>

  <collision>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <cylinder radius="0.0208" length="0.10" />
    </geometry>
  </collision>

  <visual>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <cylinder radius="0.0208" length="0.10" />
    </geometry>
  </visual>
</link>

<joint name="sonar_joint" type="fixed">
  <parent link="base_link"/>
  <child link="sonar_link"/>
  <origin xyz="0 0 0.20" rpy="0 4.7 0" />
</joint>

<gazebo reference="sonar_link">
  <sensor type="ray" name="sonar">
    <visualize>true</visualize>
    <update_rate>5</update_rate>
    <ray>
      <scan>
        <horizontal>
          <samples>15</samples>
          <resolution>1.0</resolution>
          <min_angle>-0.55</min_angle>
          <max_angle>0.55</max_angle>
        </horizontal>
        <vertical>
          <samples>10</samples>
          <resolution>1</resolution>
          <min_angle>-0.45</min_angle>
          <max_angle>0.45</max_angle>
        </vertical>
      </scan>
      <range>
        <min>0.3</min>
        <max>4.5</max>
        <resolution>0.015</resolution>
      </range>
    </ray>
    <plugin filename="libgazebo_ros_range.so" name="gazebo_ros_range">
      <ros>
        <remapping>/out:=scan</remapping>
      </ros>
      <gaussianNoise>0.005</gaussianNoise>
      <alwaysOn>true</alwaysOn>
      <updateRate>5</updateRate>
    </plugin>
  </sensor>
</gazebo>
```

En la imagen anteriormente colocada se puede observar la configuración y descripción del link donde estará el sensor ultrasónico. En él se puede apreciar sus colisiones, así como también la articulación llamada *sonar_joint* en donde el link padre será el *base_link* y el link hijo será el anteriormente creado llamado *sonar_link*. Este debe contener un origen en donde se visualizará. Por último, en la última parte del fragmento de código, se logra apreciar ya el plugin en donde estará las características del sensor, el cual contiene factores importantes en el eje horizontal como numero de muestras, resolución, ángulos en donde detectará el ancho y así mismo, en el lado vertical, se deben colocar las mismas características. Adicionalmente se adecua un rango que es hasta donde llegará la detección del sensor. Finalizando se aprecia la librería de Gazebo en donde se encuentra el tipo de sensor llamado *sonar* de tipo *ray*.

SCIENCE ROBOTICS | SPLINKER ROBOT

```

<topicName>sonar</topicName>
<frameName>sonar_link</frameName>
<fov>0.5</fov>
<radiation>ultrasound</radiation>
</plugin>
</sensor>
</gazebo>
<gazebo reference="sonar_link">
<mu1>0.01</mu1>
<mu2>0.01</mu2>
<material>Gazebo/Blue</material>
</gazebo>

```

Las últimas líneas de código son muy importantes dado que el sensor publica su comportamiento en sonar. Asimismo, tiene el *radiation* que sería el ultrasónico y presenta una referencia en gazebo en donde estará el material y el color de cómo saldrá en la ejecución del gazebo.

```

<ros2_control name="GazeboSystem" type="system">
<hardware>
  <plugin>gazebo_ros2_control/GazeboSystem</plugin>
</hardware>

<joint name="joint_1">
  <command_interface name="position">
    <param name="min">-10</param>
    <param name="max">10</param>
  </command_interface>
  <state_interface name="position"/>
  <state_interface name="velocity"/>
  <state_interface name="effort"/>
</joint>

<joint name="joint_2">
  <command_interface name="position">
    <param name="min">-10</param>
    <param name="max">10</param>
  </command_interface>
  <state_interface name="position"/>
  <state_interface name="velocity"/>
  <state_interface name="effort"/>
</joint>

<joint name="front_jr">
  <command_interface name="velocity"/>
  <state_interface name="position"/>
  <state_interface name="velocity"/>
</joint>

<joint name="rear_jr">
  <command_interface name="velocity"/>
  <state_interface name="position"/>
  <state_interface name="velocity"/>
</joint>

<joint name="front_jl">
  <command_interface name="velocity"/>
  <state_interface name="position"/>
  <state_interface name="velocity"/>
</joint>

<joint name="rear_jl">
  <command_interface name="velocity"/>
  <state_interface name="position"/>
  <state_interface name="velocity"/>
</joint>
</ros2_control>

<gazebo>
  <plugin filename="libgazebo_ros2_control.so" name="gazebo_ros2_control">
    <robot_sim_type>gazebo_ros2_control/GazeboSystem</robot_sim_type>
    <parameters>/home/jswan/dev_ws/src/robotuao2_control/config/joint_trajectory_p_controller.yaml</parameters>
  </plugin>
</gazebo>

<gazebo>

```

A continuación, se puede apreciar el plugin de las llantas en donde se crea el tópico para que el robot pueda moverse por el mundo y para ello se debe tener en cuenta algunas características como lo es el par de llantas y sus respectivos nombres de articulación, también la separación de las llantas, el diámetro y los límites que pueden tener estas llantas. En dichos parámetros se dispone de la magnitud del torque y aceleración, por último, el tópico, el cual recibirá las velocidades enviadas al cmd_vel.

```

<plugin name="robotuao2_diff_drive" filename="libgazebo_ros_diff_drive.so">
  <update_rate>30</update_rate>
  <!-- WHEEL INFORMATION -->
  <num_wheel_pairs>2</num_wheel_pairs>
  <left_joint>front_jl</left_joint>
  <left_joint>rear_jl</left_joint>
  <right_joint>front_jr</right_joint>
  <right_joint>rear_jr</right_joint>

  <!-- kinematics -->
  <wheel_separation>0.117</wheel_separation>
  <wheel_separation>0.117</wheel_separation>
  <wheel_diameter>0.206</wheel_diameter>
  <wheel_diameter>0.206</wheel_diameter>
  <!-- limits -->
  <max_wheel_torque>20</max_wheel_torque>
  <max_wheel_acceleration>1.0</max_wheel_acceleration>
  <!-- Receive velocity commands on this ROS topic -->
  <command_topic>cmd_vel</command_topic>
  <!-- output -->
  <!-- When false, publish no wheel odometry data to a ROS topic -->
  <publish_odom>true</publish_odom>

```

De las imágenes anteriores, se tiene la declaración del controlador llamado JointTrajectory. Para ello se debe agregar el plugin llamado gazebo_ros2_control/gazeboSystem y a todas las articulaciones se les implementa una interfaz en la cual se le da prioridad a la posición, igualmente se deben modificar unos parámetros, es decir, un rango para que la articulación no presente fallas si es implementado el tópico de states, que es donde estarán las demás restricciones como lo es la velocidad y esfuerzo. Este procedimiento se debe realizar para el brazo robótico, así como también para las llantas que serán controladas. Para finalizar, al plugin anteriormente mencionado, que es en donde se inicializa la librería, se debe implementar el parámetro de joint_trajectory_p_controller.yaml que es en donde este tendrá un controlador gerente, lugar donde estarán algunas características del tópico además de las articulaciones que se desean controlador, las cuales serán explicadas a continuación:

```

controller_manager:
  ros_parameters:
    update_rate: 100 # Hz

  joint_state_broadcaster:
    type: joint_state_broadcaster/JointStateBroadcaster

  joint_trajectory_controller:
    type: "joint_trajectory_controller/JointTrajectoryController"

  joint_trajectory_controller:
    ros_parameters:
      joints:
        - joint_1
        - joint_2

    command_interfaces:
      - position

    state_interfaces:
      - position
      - velocity

    state_publish_rate: 50.0 # Defaults to 50
    action_monitor_rate: 20.0 # Defaults to 20

    allow_partial_joints_goal: false # Defaults to false
    hardware_state_has_offset: true
    deduce_states_from_derivatives: true
    constraints:
      stopped_velocity_tolerance: 0.01 # Defaults to 0.01

```

Para la implementación del yaml, el cual se implementó a partir de la guía del robot Bazu, se tomaron algunas

SCIENCE ROBOTICS | SPLINKER ROBOT

restricciones y valores ya creados por defaults y se le realizaron los cambios en ros_parameter, ubicado los 2 joints en dicho espacio. Por otra parte, se tiene la adecuación necesaria para el control de posición, a partir de la implementación de los dos tópicos que son Joint State Broadcaster y Joint_trajectory_controller

```
from setuptools import setup
from glob import glob
import os
package_name = 'robotuao2_control'

setup(
    name=package_name,
    version='0.0.0',
    packages=[package_name],
    data_files=[
        ('share/ament_index/resource_index/packages',
            ['resource/' + package_name]),
        ('share/' + package_name, ['package.xml']),
        (os.path.join('share', package_name), glob('launch/*.py')),
        (os.path.join('share', package_name), glob('models/*')),
        (os.path.join('share', package_name), glob('meshes/*.stl')),
    ],
    install_requires=['setuptools'],
    zip_safe=True,
    maintainer='jswan',
    maintainer_email='jose.swan@uao.edu.co',
    description='TODO: Package description',
    license='TODO: License declaration',
    tests_require=['pytest'],
    entry_points={
        'console_scripts': [
            'trajectory_generator = robotuao2_control.trajectory_gen:main',
            'controller_generator = robotuao2_control.controller_wheel:main',
        ],
    },
)
```

La imagen anterior evidencia el archivo setup.py. Este archivo contiene las dependencias sobre algunos requerimientos. Posteriormente se implementa la búsqueda de los archivos en donde se encuentra el launch, models y meshes, luego se tiene información del desarrollador y finalmente y lo más importante es que en la parte donde se encuentra el *entry_points*, se debe colocar el nombre exacto como se desea ejecutar los controladores en el terminal. En este apartado se pueden visualizar dos que serían el controlador del brazo para que siga diversas trayectorias y el controlador de las llantas que también se estableció para que siguiera un desplazamiento por el mundo aspersando. Estos dos controladores se pueden ejecutar simultáneamente usando el setup. Ahora, a continuación, se evidencia el archivo trajectory_gen.py. Este archivo contiene toda la información y descripción acerca del controlador. En el def init se coloca el nodo, tópico, tiempo de periodo, el cual ejecutará el timer_Callback y las posiciones, las cuales se desea que se desplace el brazo del robot. Ya para continuar en el siguiente def, se tiene en cuenta que a robotuao2_control_trajectory_msg se le da los valores de Jointtrajectory y a su vez, se le da la información de los joints que en este caso es joint1 y joint 2. Se realiza la publicación por medio de puntos en donde se da valor a Point y en este tópico se almacenarán en float las velocidades, posiciones y aceleraciones y una duración para conocer qué parte comenzar y así sucesivamente, ver las diferentes trayectorias. Al final se realizan unas publicaciones con la línea self.trajectory_publisher.publish(robotuao2_control_trajectory_msg)

```
import rclpy
from rclpy.node import Node
from builtin_interfaces.msg import Duration
from trajectory_msgs.msg import JointTrajectory, JointTrajectoryPoint
from geometry_msgs.msg import Twist, Point
from trajectory_msgs.msg import JointTrajectoryPoint

class Trajectory_publisher(Node):
    def __init__(self):
        super().__init__('trajectory_published_node')
        publish_topic = "/joint_trajectory_controller/joint_trajectory"
        self.trajectory_publisher = self.create_publisher(JointTrajectory,publish_topic, 10)
        timer_period = 14
        self.timer = self.create_timer(timer_period, self.timer_callback)
        self.joints = ['joint_1','joint_2']
        self.goal_positions = [0.5,-0.6]
        self.goal_positions1 = [-0.7,0.5]
        self.goal_positions2 = [-0.8,-0.5]
        self.goal_positions3 = [0.8,0.5]

    def timer_callback(self):
        robotuao2_control_trajectory_msg = JointTrajectory()
        robotuao2_control_trajectory_msg.joint_names = self.joints
        ## creating a point
        point = JointTrajectoryPoint()
        point.positions = self.goal_positions
        point.time_from_start = Duration(sec=2)

        robotuao2_control_trajectory_msg.points.append(point)
        ## creating a point
        point_1 = JointTrajectoryPoint()
        point_1.positions = self.goal_positions1
        point_1.time_from_start = Duration(sec=5)

        robotuao2_control_trajectory_msg.points.append(point_1)
        ## creating a point
        point_2 = JointTrajectoryPoint()
        point_2.positions = self.goal_positions2
        point_2.time_from_start = Duration(sec=7)

        robotuao2_control_trajectory_msg.points.append(point_2)
        ## creating a point
        point_3 = JointTrajectoryPoint()
        point_3.positions = self.goal_positions3
        point_3.time_from_start = Duration(sec=10)

        robotuao2_control_trajectory_msg.points.append(point_3)
        ## adding newly created point into trajectory message
        self.trajectory_publisher.publish(robotuao2_control_trajectory_msg)

def main(args=None):
    rclpy.init(args=args)
    joint_trajectory_object = Trajectory_publisher()
    rclpy.spin(joint_trajectory_object)
    joint_trajectory_object.destroy_node()
    rclpy.shutdown()

if __name__ == '__main__':
    main()
```

```
import rclpy
import rclpy.node
from rcl_interfaces.msg import ParameterDescriptor
from builtin_interfaces.msg import Duration
from geometry_msgs.msg import Twist , Point
from trajectory_msgs.msg import JointTrajectoryPoint

class VelParam(rclpy.node.Node):
    def __init__(self):
        super().__init__('param_vel_node')
        self.timer = self.create_timer(20, self.timer_callback)
        self.publisher = self.create_publisher(Twist, 'cmd_vel', 10)

    def timer_callback(self):
        import time
        speed = 5.5

        Controller_wheel_msg = Twist()

        Controller_wheel_msg.linear.x = 0.0
        Controller_wheel_msg.linear.y = 0.0
        Controller_wheel_msg.linear.z = 0.0

        Controller_wheel_msg.angular.x = 0.0
        Controller_wheel_msg.angular.y = 0.0
        Controller_wheel_msg.angular.z = 1.0*speed

        time.sleep(1)
        self.publisher.publish(Controller_wheel_msg)
```

SCIENCE ROBOTICS | SPLINKER ROBOT

```

Controller_wheel_msg.linear.x = 5.0
Controller_wheel_msg.linear.y = 0.0
Controller_wheel_msg.linear.z = 0.0

Controller_wheel_msg.angular.x = 0.0
Controller_wheel_msg.angular.y = 0.0
Controller_wheel_msg.angular.z = 0.0

time.sleep(69)

self.publisher.publish(Controller_wheel_msg)

Controller_wheel_msg.linear.x = 0.0
Controller_wheel_msg.linear.y = 0.0
Controller_wheel_msg.linear.z = 0.0

Controller_wheel_msg.angular.x = 0.0
Controller_wheel_msg.angular.y = 0.0
Controller_wheel_msg.angular.z = 1.0*speed

time.sleep(0.851)

self.publisher.publish(Controller_wheel_msg)

Controller_wheel_msg.linear.x = 5.0
Controller_wheel_msg.linear.y = 0.0
Controller_wheel_msg.linear.z = 0.0

Controller_wheel_msg.angular.x = 0.0
Controller_wheel_msg.angular.y = 0.0
Controller_wheel_msg.angular.z = 0.0

Controller_wheel_msg.linear.z = 0.0

Controller_wheel_msg.angular.x = 0.0
Controller_wheel_msg.angular.y = 0.0
Controller_wheel_msg.angular.z = 0.0

time.sleep(5.5)

self.publisher.publish(Controller_wheel_msg)

Controller_wheel_msg.linear.x = 0.0
Controller_wheel_msg.linear.y = 0.0
Controller_wheel_msg.linear.z = 0.0

Controller_wheel_msg.angular.x = 0.0
Controller_wheel_msg.angular.y = 0.0
Controller_wheel_msg.angular.z = 1.0*speed

time.sleep(0.876)

self.publisher.publish(Controller_wheel_msg)

def main(args=None):
    rclpy.init(args=args)
    cmd_vel_object = VelParam()

    rclpy.spin(cmd_vel_object)
    cmd_vel_object.destroy_node()
    rclpy.node.shutdown()

```

Las capturas de pantalla tomadas anteriormente permiten ilustrar el controlador llamador *controller_wheel*, el cual fue elaborado por el equipo de trabajo y en el cual, siguiendo el mensaje *geometry_msg*, se publica con *cmd_vel* y el *Twist* que es donde se envian los datos que se quieren visualizar en el desplazamiento del robot. Para ello se inicializa el timer y

el publicador para ya pasar a indicarle las coordenadas dependiendo de lo que se deseé realizar. Primero, en la implementación del robot, se desea que el robot al ejecutarlo se spawnee en el sendero para comenzar a aspersar los cultivos. Por lo tanto, el robot se desplazará linealmente durante 69 segundos que es el tiempo que tarda en pasar de lado a lado. Cuando supera el otro lado, el robot girará 90 grados y avanzara durante 5,5 segundos para prepararse a volver a girar. Es entonces ahí cuando realiza de nuevo el giro y se regresa por la otra acera (o sendero) del cultivo donde, para lograr dicho funcionamiento, fue necesario usar una dependencia de timer y usar un delay para que el robot pueda publicar y hacer por partes el desplazamiento sin tener fallas.

```

@return LaunchDescription([
    ExecuteProcess(
        cmd=['gazebo', '--verbose', '-s', 'libgazebo_ros_factory.so'],
        output='screen',
    ),
    ExecuteProcess(
        cmd=['ros2', 'control', 'load_controller', '--set-state', 'start', 'joint_state_broadcaster'],
        output='screen',
    ),
    ExecuteProcess(
        cmd=['ros2', 'control', 'load_controller', '--set-state', 'start', 'joint_trajectory_controller'],
        output='screen',
    ),
    #]
])

```

Para finalizar, se logra apreciar los cambios que se le realizaron al launch. Dichos cambios corresponden a agregar los procesos anteriormente explicados, donde se ejecutan los controladores y en donde leerá las publicaciones de los topicos *joint_state_broadcastes* y *joint_trajectory_controller*. A su vez, en el archivo launch fue necesario eliminar el nodo *joint_state_publisher* ya que no se requiere su uso debido a que se realizará por código para visualizar las trayectorias deseadas.

CRONOGRAMA DE ACTIVIDADES

En este inciso del documento se tiene la disposición del cronograma que explicara las actividades y los logros alcanzados, semana a semana, para llevar a cabo el desarrollo del proyecto. Comenzará desde la semana 8 e irá desglosando cada avance del prototipo con su nivel de relevancia para tener la construcción final, teniendo en cuenta que en un principio no se sabía que el prototipo físico no iba a ser solicitado de forma obligatoria para la entrega final.

AVANCE/PROTOTIPO	DESCRIPCIÓN	LOGRO ALCANZADO	INCONVENIENTES	
Generación de Conceptos	Sem 8	En este apartado, el equipo de trabajo, teniendo en cuenta la descomposición funcional del dispositivo, las necesidades del contexto del problema y las métricas encontradas a partir de ella, realizó un bosquejo de diferentes alternativas para evaluar la morfología del robot y lo que iba a ser necesario para que cumpliera con la tarea mínima que era la aspersión y el movimiento.	Los bocetos ayudaron a desarrollar un concepto de manera más rápida pero que estuviera acorde con el contexto del problema. De esta forma, se puede tener en cuenta en ellos, los lineamientos generales a partir de la naturaleza visual de los bocetos, en el que incluso es posible explicar de forma gráfica, sus funciones prácticas y simbólicas.	A veces, no solamente con dibujar o bocetar una idea es suficiente para dar a entender los aspectos generales de un diseño. Por tal motivo, explicar mediante bocetos hace que sea necesario un buen dibujo y si es posible, un dibujo técnico.

SCIENCE ROBOTICS | SPLINKER ROBOT

Diseño del Robot en SolidWorks	Sem 9	A partir del concepto seleccionado, el equipo de trabajo realizó las diferentes partes del robot, entre lo que componen de forma más significativa, el chasis, las articulaciones para las ruedas, las ruedas y, por otra parte, la armadura (compuesta de dos eslabones y dos articulaciones).	Obtener un render para el proyecto es muy importante, porque un prototipo 3D puede ser incluido dentro de ROS y es utilizado en todas las etapas de planeación de un prototipo Alpha, un MVP para después, llevar a cabo de una mejor forma su proceso de desarrollo. Además, permite, desde la simulación de SolidWorks, revisar que problemas tiene el diseño antes de ser construido físicamente o incluso tener un Feedback del cliente.	No se tenía conocimiento acerca de cómo transferir archivos desde SolidWorks para ser tenidos en cuenta en la programación de ROS. Por otra parte, en un principio, las dimensiones del robot fueron aleatorias, luego, a partir de la contextualización del problema, se tuvieron ciertos requerimientos de diseño.
Visualización en Rviz	Sem 10	A partir del prototipo virtual del robot realizado en SolidWorks, se realiza la codificación del urdf, el cual es un archivo que puede ser entendido por el Middleware y que, a su vez, describe las características cinemáticas del Splinker. Dicha configuración se dispone dentro del paquete del robot creado en el espacio de trabajo para poder seguirlo desarrollando o codificando.	Para el equipo de trabajo fue importante realizar la visualización puesto que la idea es desarrollar de forma física el robot y la intención es poder llegar a obtener la simulación del Splinker. Con Rviz, se logra observar la morfología tridimensional del Splinker descrito en formato urdf incluyendo sus llantas, el chasis, el almacenador de fungicida y en él, se dispone, mediante la librería tf, los datos de esos datos en el marco de referencia del robot. En él se utiliza el joint_state_publisher para ver el comportamiento de los grados de libertad más que todo del Robot Manipulador.	Hasta ese momento no se sabía cómo hacer el movimiento de las llantas ni la adecuación de sus respectivos ejes. Además, al principio hubo problemas con la adecuación de los colores.
Simulación en Gazebo	Sem 11	El equipo de trabajo modificó el archivo launch.py; también se cargaron dentro de la carpeta meshes, los componentes en formato ST tanto para representar la forma del robot y otros, con una geometría sencilla de recubrimiento con el objetivo de llevar a cabo, el tratamiento y la verificación de las colisiones. Por otra parte, se cargó el mundo agriculture.world	En este inciso, el equipo de trabajo aprendió a actualizar dependencias, también a modificar los archivos de lanzamiento y las coordenadas del Spawn del Robot en el mundo, por tal motivo, se aprendió también a cargar el mundo y a realizar la transferencia de la información desde un urdf a un archivo tipo xacro.	En primera instancia se tuvo problemas con las colisiones debido a que no se había tenido una geometría sencilla para la simulación, sino que el recubrimiento era todo con rectángulos porque se consideró como equipo de trabajo

SCIENCE ROBOTICS | SPLINKER ROBOT

			<p>Por tal motivo, Gazebo permite observar desde la simulación, el comportamiento físico y dinámico del robot en un contexto emulado de la realidad. De esta forma, permite simular con precisión y eficiencia, los interiores y los exteriores que, en el caso del robot permitió la realimentación y verificación de colisiones y también la interacción entre los objetos físicos teniendo en cuenta los cuerpos rígidos establecidos y su movimiento.</p> 	<p>que así fuese. No obstante, se pudo corregir dicho error a tiempo debido a que un rectángulo recubriendo por ejemplo las llantas generaba que el robot se moviera, pero de forma inestable.</p>
Implementación del Controlador y Sensores	Sem 13	<p>La implementación de los controladores se hicieron con el fin de dar cumplimiento a las trayectorias deseadas por el usuario la cual sirve para visualizar y simular el robot en funcionamiento en la herramienta de Gazebo para esto se tuvieron que implementar dos controladores llamados Trajectory_generator y Controller_wheel cada uno de estos consta de un proceso en python, mediante el cual se publica en algunos topics de articulaciones del brazo robótico como también de las llantas y así, es la forma de asumir una trayectoria deseada que es recorrer las aceras de la granja aspersando el líquido (fungicida). El código consta de 3 incisos. Uno donde se genera la inicialización de librerías y nodos. Posteriormente procede al llamado donde se ejecuta la ruta de lo que se desea hacer y a su vez publicarlo y, por último, el código main donde esta las inicializaciones ya anteriormente colocadas y algunas líneas tomadas de proyectos vistos en clase como guías. Para la implementación de los sensores se tuvo en cuenta que se deben crear unos links y joints llamados sonar los cuales tienen un plugin en el urdf y estos se toman en base a el tipo de sensor utilizado lo cual es el sensor</p>	<p>Básicamente, la implementación tanto de sensores como de controladores es la que permite la simulación más real y verídica del funcionamiento del robot en un ambiente de agricultura, logrando con ello, obtener más estatus para su comercialización y más credibilidad en el momento de monetizar y validar la idea de diseño desde la robótica.</p>	<p>El problema se presentó en el momento de colocar los controladores, debido a que se arrojaba un error en el terminal de algunas líneas de código del launch cuando se ejecutaba y tomó bastante tiempo corregirlo. Así como también en el controlador del movimiento de las llantas, cuando se enviaba el Twist para que publicara en línea recta se desviaba provocando fallas en el desplazamiento y, por ende, en la trayectoria.</p>

SCIENCE ROBOTICS | SPLINKER ROBOT

		<p>ultrásónico. Este, envía un rango y al percibir un objeto cerca, la onda se regresa y se visualiza de color más azul. A su vez, involucra un sensor en la parte de adelante como en la trasera para captar a lo mejor, algún objeto en estas posiciones. Para realizar ello, se dispone de las librerías y las características de cada sensor. El controlador se debe ejecutar en el terminal y tiene un periodo de publicación, regido por el periodo colocado en el código</p>		
Compra de materiales para generar el prototipo físico	Sem 14	<p>En este caso se utilizó la tabla de materiales y componentes anteriormente mencionada y el equipo de trabajo se dirigió a los locales de electrónica ubicados en el centro de la ciudad para hacer la compra. Además, se solicitaron algunos componentes en Vistronica.</p>	<p>En este caso, se aprendió a utilizar bien los manuales de las especiaciones técnicas de cada uno de los componentes, pues es de esta, la única forma en la que se pueden tener consideraciones tanto para las funciones prácticas como para el ensamblaje de cada uno de los elementos.</p>	<p>No se encontraban los motorreductores adecuados para poder mover al menos 5 kg que correspondían a los 5 L de fungicida.</p>
Corte laser y ensamble de las partes del robot	Sem 15		<p>En este caso fue importante llevar a cabo el ensamblaje y la distribución de cables de tal forma que se evitara el recalentamiento del regulador de tensión LM317T. Asimismo, intentar tener la suficiente corriente para la manipulación del Servomotor MG995, debido a que necesita 1 A para su funcionamiento y al estar conectado al Arduino, no se cuenta con tal magnitud de corriente. La última consideración, fue que al no disponer de la cortadora laser, se utilizaron las herramientas del taller de prototipado para ajustar el chasis y las llantas, adquiriendo con ello, experiencia en el uso de herramientas cotidianas.</p>	<p>No fue fácil la adaptación de las dimensiones del eje del motor y el espacio disponible para los ejes en la llanta. El módulo de carga al ser soldado se está soltando con facilidad.</p>
Aplicativo móvil para el control del robot y testeo de este en terreno similar en la universidad	Sem 16	<p>Mediante la realización del aplicativo, se tenía la intención de poder crear una conexión más amena del robot con el agricultor y que fuera más intuitivo la funcionalidad de este.</p>	<p>El desarrollo del aplicativo generó un gran impacto emocional para el agricultor porque le permitió desde su utilidad, poder reemplazar el trabajo pesado que anteriormente realizaba cuando tenía la motobomba “cargada en la espalda”, pues ahora sería suficiente con su interacción para así poder tener mayor efectividad en el proceso.</p>	<p>Conexión interrumpida debido al alcance que puede sostener el módulo bluetooth.</p>

TRABAJO INDEPENDIENTE

El primer trabajo independiente que se realizó fue investigar sobre como generar las meshes, URDF teniendo como base un modelado 3D hecho en SolidWorks, para ello fue necesario consultar varios videos en YouTube y un foro en GitHub, donde explicaban que mediante un plugin es posible

realizarlo, pero hay que tener varios aspectos en cuenta como la ubicación de los sistemas coordenados de cada articulación, la ubicación correcta de esos marcos de referencia, ya que si se ubican mal esto afecta demasiado la orientación del robot, además se tuvieron en cuenta las geometrías simples para las colisiones y se eliminaron las complejas con el objetivo de

SCIENCE ROBOTICS | SPLINKER ROBOT

ahorrar recursos. En segundo lugar, ya estando en el sistema operativo de Ubuntu, se dirige al espacio de trabajo/src con el objetivo de crear un paquete tipo ament_cmake con el nombre del proyecto, en este caso robotsprinkler_description, dentro de este paquete se crearon las carpetas launch, meshes, models, por consiguiente, se copiaron y pegaron los archivos generados por el plugin de SolidWorks en cada una de sus respectivas carpetas, posteriormente se hizo colcon build y source para probar ese paquete, con el objetivo de corroborar su correcto funcionamiento, para todo lo anterior fue suficiente consultar en las diapositivas del curso sección "Your first URDF file".

Luego, en el primer lanzamiento en Rviz se analizó si las variables articulares se movían correctamente, verificando los sistemas coordenados y los criterios de ubicación de los ejes donde z sigue la regla de la mano derecha y un giro positivo en contra de las manecillas del reloj, Así mismo, el lanzamiento en Gazebo permitió corroborar la correcta aparición o spawn en la aplicación y además la correcta carga del mundo, en este caso se utilizó un mundo predeterminado de agricultura, donde se tiene similitud con los cultivos convencionales de maracuyá en su fase de crecimiento.

Por último, el trabajo que se realizó fue la implementación de controladores para mover el brazo robótico y calcular las trayectorias a las cuales debe seguir por medio de desplazamiento generados por la rotación de las llantas, vale la pena mencionar que Splinker Robot funciona como un robot tipo diferencial donde las llantas delanteras son de tracción y las llantas traseras son solo de apoyo no son ni de tracción ni direccionables, el tutorial que se utilizó fue el de Bazu, donde el equipo de trabajo aprendió a cómo hacer la implementación del controlador que permitiera la manipulación del brazo robótico, es decir, variar sus valores articulares para obtener una pose deseada. Por último, se obtuvo la tele operación del Splinker robot donde, al ejecutar el archivo de lanzamiento, aparece el robot spawneado en una posición deseada y este a su vez cumple con una trayectoria de aproximadamente 80 seg y que pudiera darle como mínimo una vuelta a los senderos. Aplicando con ello la cinemática inversa para el componente del Brazo Manipulador y cinemática directa para el movimiento de la parte móvil del robot.

DISCUSIÓN Y CONCLUSIONES

Con la realización formal del documento y, teniendo en cuenta el indicador de desempeño que evalúa si el equipo de trabajo tiene la capacidad de resolver problemas complejos de ingeniería aplicando principios de conceptos desde las asignaturas de ingeniería, ciencias y matemáticas, se tiene que con la implementación del Splinker Robot como solución a la problemática correspondiente a la falta de control de la reproducción de los hongos en los cultivos de maracuyá en el corregimiento de Rozo, Valle del Cauca, en temporadas de

invierno, si se está resolviendo de forma sobresaliente y optima debido a que este ha sido un proceso que desde el momento en que inició teniendo presente como punto de partida, la extracción de unas necesidades a partir del contexto del problema y la delimitación del usuario, sus requerimientos y los asuntos latentes y personales que se tuvieron en cuenta, ha sido una idea muy concreta y específica para un grupo definido de personas. Sin embargo, el hecho de haber realizado el Modelado 3D del Robot en SolidWorks, le permitió al equipo de trabajo, no solo fortalecerse en la herramienta CAD sino que permitió tener los fundamentos necesarios para diseñar teniendo en cuenta componentes robóticos, como los ejes de rotación, relaciones de posición de las articulaciones con los eslabones, efectores, sensores y demás, lo cual, cuando se integran dichos elementos, le brinda la oportunidad al ingeniero en proceso de formación, que adquiera experticia y que el día de mañana, no solo se conforme con realizar el renderizado del diseño sino lograr que, desde la robótica, sean aplicados correctamente los principios básicos de la integración de diferentes asignaturas como correspondencia a dicha integración, el Diseño Mecatrónico con su respectivo análisis del impacto social; El Diseño Industrial, con la evaluación de las funciones prácticas y comunicativas del dispositivo. Gestión de la Innovación, con las estrategias de marketing, modelos de negocio e impulso de la creatividad y lo que corresponde al eje central, que es la Robótica, la cual tiene una de las más grandes ventajas, la cual es que el ser humano logre sustituir constantes labores agotadoras y repetitivas gracias a la creación de robots, los cuales incluso han mejorado su capacidad de rapidez, calidad de vida y la optimización de tiempo en cuanto a la efectividad de sus tareas. Por tal motivo, para realizar la discusión acerca de la credibilidad y actualización de las búsquedas de información, el equipo de trabajo inicia haciendo una investigación de escritorio acerca de la problemática en diferentes páginas web. En este momento del documento, existió la tarea de realizar el reconocimiento de la información y, por ende, hacer un filtro de las páginas más fidedignas en aras de poder tener seguridad en el momento de hablar por ejemplo de cifras, porcentajes de perdida, de ganancia o incluso, adquiriendo la habilidad de cómo seleccionar la información concerniente al tópico requerido, reconociendo la calidad del material. Las métricas que surgieron de las necesidades mencionadas en el desarrollo del documento corresponden a la delimitación de la información obtenida por el agricultor Deivi Mañozca. Por dicha razón, en aras de ser pertinentes en la aplicación de información, se podría decir que, a partir de la información obtenida en la fase de inmersión, es posible completar la adecuación y la gestión de la información recopilada, así como también de la capacidad de explicar claramente el procedimiento seguido y puede ayudar a otros.

SCIENCE ROBOTICS | SPLINKER ROBOT

El hecho de haber diseñado, codificado en ROS, visualizarlo y simular el Splinker Robot, fue una de las formas en las que el equipo de trabajo comprendió que si es posible la integración de las diferentes ramas de la ingeniería mecatrónica puesto que, a pesar de no tener mucha experticia, no solo se lleva a cabo el modelo virtual del dispositivo, sino también el modelo físico funcional, que es el Mínimo Producto Viable y al disponerlo en un ambiente similar al del cultivo cumple con las expectativas planteadas al inicio del documento, donde se esperaba que el dispositivo causara en el usuario, la idea de notarse como un dispositivo capaz de optimizar el proceso de fumigación/aspersión para los cultivos de Maracuyá y también de que con él, “a la mano”, se lograra reducir el porcentaje de pérdidas de los frutos y hojas durante el proceso de germinación y extracción del fruto. Todo lo anterior, destacando la importancia de los procesos de diseño en la ingeniería aplicada que en el caso corresponde a la robótica, pues es de esta forma, al final de cuentas, lo que se realiza industrialmente en la solución de los problemas ingenieriles que ha sido el proceso que ha originado ideas significativas para el desarrollo humano, llevando consigo el tratamiento de los ODS en aras de proporcionar elementos para construir Ciudades y Comunidades Sostenibles y para tener un consumo responsable de los aimentos, pues en Colombia, el sector agrario ha sido uno de los sectores más vulnerables y menos tecnificados del país y con la solución propuesta, no solo se está mejorando la calidad de vida del agricultor sino también, de forma indirecta, el impacto social en cuanto a

empleabilidad y reducción de enfermedades respiratorias y osteomusculares, logrando con ello, la validez y la acreditación que toma, el importante papel significativo que juega la robótica en el desarrollo social del ser humano.

REFERENCIAS

1. Retos de la Agricultura en Colombia, centro de los objetivos de desarrollo sostenible para américa latina, [En línea]. Disponible en: <https://cods.uniandes.edu.co/los-retos-de-la-agricultura-colombiana-frente-al-cambio-climatico/#:~:text=Ganader%C3%A1%20extensiva%20cultivos%20improductivos%2C%20deforestaci%C3%B3n,sobre%20agricultura%20y%20cambio%20clim%C3%A1tico>.
2. Mientras 2.7 millones de colombianos sufren hambre, 10 millones de toneladas de alimentos se desperdician anualmente, Noticia disponible en Universidad Nacional de Colombia, [En línea]. Disponible en: <http://ieu.unal.edu.co/medios/noticias-del-ieu/item/mientras-2-7-millones-de-colombianos-sufren-hambre-10-millones-de-toneladas-de-alimentos-se-desperdician-anualmente#:~:text=En%20Colombia%20se%20 pierde%20y,9%2C76%20millones%20de%20toneladas>.
3. ENFERMEDADES PROFESIONALES DE LOS AGRICULTORES, Comisión Nacional de Seguridad y Salud en el Trabajo, Documento PDF [En línea]. Disponible en: <https://www.ucm.es/data/cont/media/www/pag56437/enfermedades%20profesionales%20de%20los%20agricultores.pdf>.

