ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTA DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA DE INGENIERÍA ELECTRÓNICA EN CONTROL Y REDES INDUSTRIALES

"IMPLEMENTACIÓN DE UN PROTOTIPO DE ROBOT SEMBRADOR DE PAPA EN TERRENOS SIN INCLINACIÓN PARA PEQUEÑOS PRODUCTORES"

Trabajo de titulación: PROYECTO TÉCNICO
Presentado para optar al grado Académico de**:**
INGENIEROS ELECTRÓNICOS EN CONTROL Y REDES INDUSTRIALES

AUTORES: HENRY FABRICIO GONZÁLEZ GAVILANES
MARÍA GABRIELA CARRILLO TRUJILLO

TUTOR: ING. JOSÉ MORALES

Riobamba-Ecuador
2019

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA DE INGENIERÍA ELECTRÓNICA EN CONTROL Y REDES INDUSTRIALES.

El tribunal del trabajo de titulación certifica que: El trabajo de investigación:" **IMPLEMENTACIÓN DE UN PROTOTIPO DE ROBOT SEMBRADOR DE PAPA EN TERRENOS SIN INCLINACIÓN PARA PEQUEÑOS PRODUCTORES"**, de responsabilidad de los señores: MARÍA GABRIELA CARRILLO TRUJILLO Y HENRY FABRICIO GONZÁLEZ GAVILANES, ha sido minuciosamente revisado por los Miembros del Tribunal del trabajo de titulación quedando autorizada su presentación.

	FIRMA	**FECHA**
Ing. Washington Gilberto Luna E. **DECANO DE LA FACULTAD DE INFORMÁTICA Y ELECTRÓNICA.**	.....	17/Julio/2019
Ing. Freddy Chávez V. **DIRECTOR DE LA ESCUELA DE INGENIERÍA ELECTRÓNICA EN CONTROL Y REDES INDUSTRIALES.**	.....	17/JULIO/2019
Ing. José Morales **DIRECTOR DEL TRABAJO DE TITULACIÓN.**	.....	17/Julio/2019
Ing. José Guerra **MIEMBRO DEL TRIBUNAL.**	.....	17-JULIO-2019

Nosotros, MARÍA GABRIELA CARRILLO TRUJILLO Y HENRY FABRICIO GONZÁLEZ GAVILANES somos responsables de las ideas, doctrinas y resultados expuestos en esta Tesis y el patrimonio intelectual de la Tesis de Grado pertenece a la Escuela Superior Politécnica de Chimborazo.

María Gabriela Carrillo Trujillo

Henry Fabricio González Gavilanes

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE GRÁFICOS

ÍNDICE DE ANEXOS

RESUMEN

El prototipo de robot sembrador de papas se realizó al observar la necesidad del agricultor al realizar el proceso de siembra. El prototipo se implementó en dos etapas, la primera el montaje mecánico el cual consta de materiales como latón y placas de metal, su peso es de 25kg teniendo un anclaje en la tierra haciendo el proceso más eficiente, cuenta con un sistema de almacenaje tipo tolva para 3lb de papas y un sistema de transporte realizado con una banda de plástico, la cual llevará a la semilla hacia su destino, la segunda etapa fue la implementación electrónica, cuenta con un sistema de potencia, el cual depende de 4 motores capaces de realizar una fuerte tracción para mover la cadena tipo oruga, con un motor en la banda transportadora para movilizar el tubérculo. Los motores son programados a través del microprocesador Arduino en forma de control de velocidad denominado PWM, el sistema de control consta de envío y recepción de datos a la aplicación móvil, los datos son transmitidos inalámbricamente por WIFI a través del microprocesador NodeMCU, cumpliendo con los parámetros de odometria para cumplir el proceso autónomo de siembra. De las pruebas realizadas se determinó que el prototipo cumplió con un tiempo de trabajo de 49min, el proceso de siembra según las especificaciones técnicas tanto en profundidad y distancia comparadas con la siembra artesanal tuvo una precisión de un 40% y un 10.25% de aumento en su productividad. Al evaluar su costo teniendo una comparación entre la maquinaria utilizada para preparar el terreno se obtuvo una diferencia de 50.51% en el ensamblaje del prototipo. Es recomendable la capacitación de los agricultores sobre la aplicación móvil y el funcionamiento del prototipo para el uso correcto en el proceso de siembra.

PALABRAS CLAVE: <TECNOLOGÍA DE CONTROL AUTOMÁTICO>, <AGRICULTURA>, <ROBÓTICA>, <AUTÓMATA>, <ODOMETRIA>, <SIEMBRA>, <PAPA (*Solanum tuberosum*)>, <ARDUINO (SOFTWARE – HARDWARE)>.

ABSTRACT

The prototype potato-sower robot was realized by observing the need of the farmer to carry out the sowing process. The prototype was implemented in two stages, the first the mechanical assembly which consists of materials such as brass and metal plates, its weight is 25kg having an anchor in the ground making the process more efficient, it has a storage system type hopper for 3lb of potatoes and a transportation system made with a plastic band, which will take the seed to its destination, the second stage was the electronic implementation, has a power system, which depends on 4 engines capable of performing strong traction to move the caterpillar-type chain, with a motor on the conveyor belt to move the tuber. The motors are programmed through the Arduino microprocessor in the form of a speed control called PWM, the control system consists of sending and receiving data to the mobile application, the data is transmitted wirelessly via WIFI through the NodeMCU microprocessor, compliance with the Odometry parameters to meet the autonomous process planting. Of the tests carried out it determined that the prototype met a working time of 49min, the sowing process according to the technical specifications both in depth and distance compared to the artisanal sowing had an accuracy of 40% and a 10.25% increase in your productivity. When evaluating its cost having a comparison between the machinery used to prepare the land a difference of 50.51% was obtained in the assembly of the prototype. It is recommended the ability of farmers on the mobile application and the operation of the prototype for the correct use in the sowing process.

KEYWORDS: <AUTOMATIC CONTROL TECHNOLOGY>, <AGRICULTURE>, <ROBOTICS>, < AUTOMATON>, <ODOMETRY>, <SOWING>, < (Solanum tuberosum) POTATOES >, <(SOFTWARE-HARDWARE) ARDUINO >.

INTRODUCCIÓN

ANTECEDENTES

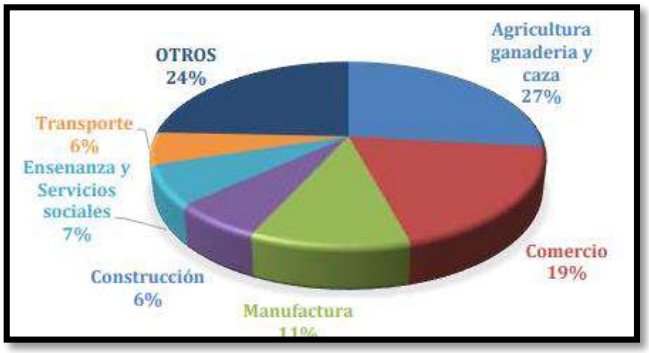
Actualmente la demanda de alimentos está en constante crecimiento como resultado del aumento de la población, la ONU estima un aumento al año 2050 habrá aproximadamente 9100 millones de habitantes los cuales deberán ser también alimentados, además estos alimentos deben ser producidos mediante el uso eficaz de los recursos naturales tomando en cuenta el no perjudicar al medio ambiente en este punto la agricultura tiene un gran desafío el de generar una gran cantidad de alimentos considerando en un futuro la mayoría de la población vivirá en la ciudad o áreas urbanas ,razón por la cual el uso de la tecnología robótica facilitará la producción de dichos alimentos (Morales García, 2013).

Los procesos productivos han sido reemplazados por máquinas inteligentes capaces de sustituir las tareas realizadas por los agricultores como es: la siembra, recolección de frutos, fumigación y preparación de la tierra , por otra parte, uno de los factores influyentes para la creación es la falta de mano de obra y sus elevados costos al tratar de producir una gran cantidad de alimentos, dejando así a los productores sin posibilidades de cumplir con todas sus cosechas por no tener el suficiente personal para realizarlo, como consecuencia la mano de obra se está reduciendo, por tal motivo se ha dado paso a la automatización con el propósito de mejorar los procesos agrícolas tales como el arado, el sembrado y la recolección de frutos (Jorge Velasco Cruz, 2017).

El 85% de superficie sembrada es menor a dos hectáreas en países de bajo y mediano ingreso por lo cual no será factible para un aporte a la economía de estos países, sin embargo, a pesar de ser pequeños productores tienen una mayor productividad de sus tierras (FAO, 2017).

Ecuador es en su mayor parte agrícola, siendo este su principal fuente de ingresos y de empleo con un 27% para las familias como se puede ver en la Grafica 1-1 ,sin embargo en los últimos años se ha dejado a un lado su participación en la economía del país debido al petróleo y comercio, aun así, el sector agrícola es considerado como un sector estratégico, a medida que pasa los años el sector urbano ha tenido un incremento importante en sus zonas mientras en el sector rural se

ha visto un alto índice de abandono conllevando esto a una mano de obra más barata (Luciano.M, 2013).



Grafica 1-1: Porcentaje empleo 2017.
Fuente: (Fiallo.J, 2017)

A partir del año 2008 el Gobierno de Ecuador dividió su territorio por zonas en donde según análisis realizados por el Ministerio de Agricultura y Ganadería en donde se puede observar en Figura 1-1 cada una de las zonas con sus respectivas provincias, así como la cantidad de superficie con aptitudes agropecuarias siendo la zona tres con un 95% de aptitud agropecuaria seguido de la zona cinco con un 91%por lo tanto se determina que el 79% de la superficie del territorio es netamente productiva (MAGAP, 2015).

Zona de planificación	Provincias	Superficie total (km²)	Población a 2014	% Superficie con aptitud agropecuaria
Zona 1	Imbabura, Carchi, Sucumbíos y Esmeraldas	42.584,6	1'408.235	72%
Zona 2	Pichincha, Napo y Orellana	43.770,5	3'157.510	61%
Zona 3	Cotopaxi, Tungurahua, Chimborazo, Pastaza	45.635,6	1'595.581	77%
Zona 4	Manabí y Santo Domingo de losTsáchilas	22.386,3	1'892.949	95%
Zona 5	Santa Elena, Guayas, Los Ríos y Bolívar	30.271,2	5'428.821	91%
Zona 6	Azuay, Cañar y Morona Santiago.	35.515,1	1'220.754	84%
Zona 7	El Oro, Loja, y Zamora Chinchipe	27.413,7	1'257.923	86%
T. país*		**247.576,9**	**15'961.773**	**79%**

Figura 1-1: Superficie con aptitud agropecuaria.
Fuente: (MAGAP, 2015).

FORMULACIÓN DEL PROBLEMA

¿El proceso artesanal de siembra no es establecidos según parámetros técnicos lo que permite el desperdicio de las semillas?

¿Cuantos tubérculos son plantados artesanalmente en una parcela pequeña?

¿Qué diseño es el más adecuado para que el prototipo cumpla con el proceso de siembra?

¿Qué tipo de elementos hardware y que software es el más indicado para cumplir con los requerimientos de diseño?

JUSTIFICACIÓN

"Ecuador se encuentra entre los países que poseen una capacidad de producción de alimentos por encima de las crecientes demandas de su población, en un mundo que demanda cada día más alimentos, especialmente por parte de las economías emergentes, el sector agropecuario ecuatoriano ofrece enormes posibilidades para la población y para la economía en su conjunto es por eso la relevancia y preocupación primordial que genera su análisis y atención prioritaria dentro de las políticas públicas, además de ser un enorme reto, su adecuado tratamiento puede magnificar visiblemente la enorme contribución del sector agropecuario" (MAGAP, 2015).

Al realizar esta investigación se busca implementar un prototipo que siembre en terrenos planos un tipo de producto que tiene mayor alcance de exportación como lo es la papa y de esta manera ayudar a los pequeños productores que son los que dan un mayor aporte al país. El resultado esperado de esta investigación es disminuir el esfuerzo y tiempo del agricultor al realizar este proceso, además mediante el estudio se prevé aumentar la productividad del agricultor debido a que el prototipo tendrá los parámetros técnicos establecidos por estudios realizados.

A nivel mundial existe diversos dispositivos capaces de realizar tareas en la agricultura tales como fumigación, destrucción de malezas, arado, implantación de semillas mediante un brazo robótico, recolectores de frutos, pero hasta en la actualidad no se han comercializado en el Ecuador los cuales serían de ayuda para los agricultores.

Mediante los estudios realizados se pudo observar que hay muy pocos sembradores autónomos por lo cual al desarrollar este prototipo se trata de ayudar al agricultor a sembrar de una manera más fácil y adecuada. El prototipo será capaz de colocar una semilla seleccionada cada cierta distancia, así como el usuario prototipo será capaz de llevar el control de la cantidad de semillas sembradas y enviar mensajes de alerta de falta de semillas y culminación de la siembra al celular del agricultor.

OBJETIVOS

Objetivo General

Desarrollar un prototipo de sembrador terrestre, capaz de colocar las semillas en la tierra y así minimizar el tiempo de sembrado que lleva la papa.

Objetivo específicos

- Realizar una investigación sobre el proceso de siembra de papa.
- Diseñar el prototipo más adecuado para cumplir con los requerimientos de la siembra.
- Determinar los elementos hardware y software que permiten cumplir con los requerimientos del diseño
- Evaluar si el prototipo cumplió con los requerimientos

CAPITULO I

1 MARCO TEÓRICO

En este capítulo se habla de la importancia del cultivo de la papa en el ecuador, el proceso de siembra local, la agricultura robotizada a nivel mundial y se detalla los dispositivos existentes en el mercado para la implementación del prototipo sembrador.

1.1 La importancia de la papa.

Considerada la papa como un alimento para las personas de bajos recursos ya sean en la zona rural o en la zona urbana así como se estima que en los Andes hay alrededor de 3000 tipos de estos tubérculos con altos nutrientes para los seres humanos , además de ser el cuarto alimento más importante en el mundo después del trigo maíz y arroz y por tener una alta producción a nivel mundial mayor a 323 millones (T) toneladas Erik Keuneman menciona a la papa como un tesoro escondido (FAO, 2006) .

1.1.1 La papa en el Ecuador

Según el INIAP (Instituto Nacional de Investigaciones Agropecuarias) en el Ecuador existe aproximadamente 82000 personas que trabajan en el cultivo de papas, principalmente el 81% de este tubérculo es destinado para consumo diario y el 19% es dedicada para el procesamiento en las industrias (INIAP, 2014) .

Actualmente existe en el Ecuador 500 tipos de papas y 50 000 Hectáreas de producto sembrado produciendo 3000 (T) toneladas, el MAGAP (Ministerio de Agricultura y Ganadería) en conjunto con el INIAP (Instituto Nacional de Investigaciones Agropecuarias) buscan incentivar a la población agricultora a tener una mayor producción y de esta forma impulsar a la población para tener un mayor consumo (MAGAP, 2014)

Según análisis realizados por el Ministerio de Agricultura y Ganadería el cultivo de papa se realiza en 12 provincias las cuales son: Cotopaxi Carchi Chimborazo, Tungurahua, Cañar, Pichincha, Bolívar, Azuay, Imbabura, Loja, Sucumbíos, El Oro (Guerrero, 2015).

1.1.1.1 *Rendimiento de la papa*

El Ministerio de Agricultura y Ganadería en el año 2017 informa que el rendimiento promedio nacional de la papa es de 18.9(t/ha) toneladas por hectárea de los cuales las provincias productoras con mayor rendimiento son: Sucumbíos (29.90 t/ha), Tungurahua (25.74 t/ha), Carchi(21.80t/ha) y Chimborazo (19.22t/ha), las provincias con menor rendimiento son: Bolívar, Azuay, Imbabura, Pichincha, Cotopaxi esta con una diferencia de (0.3 ,1.3 ,1.5, 3.1 ,6.6) al promedio nacional como se puede observar en la Grafico 2-1 (MAGAP, 2018).

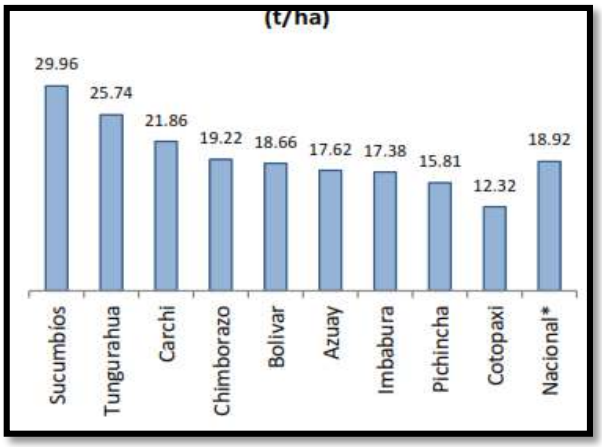


Grafico 2-1: Estudio del rendimiento de la papa por provincias.
Fuente:(MAGAP, 2018)

Según datos estadísticos obtenidos del MAGAP en el año 2015 Carchi(22.43t/ha) era la provincia con mayor rendimiento seguido de Pichincha (14.72t/ha), Tungurahua (14.04), Chimborazo (13.80t/ha) con un 16.2t/ha de promedio nacional, las provincias con mayor rendimiento en el año 2017 es Sucumbíos (29.96t/ha), Tungurahua (25.74t/ha), Carchi (21.86t/ha), Chimborazo (19.22t/ha) y su promedio nacional es de 18.92t/ha en donde se puede observar en un lapso de 3 años ha subido y ha bajado considerablemente su rendimiento en ciertas provincias sin embargo el promedio nacional ha ido en aumento.

1.1.1.2 *Mecanización de Tareas*

Debido al esfuerzo manual que requiere este cultivo los agricultores han optado por utilizar maquinarias en cada tarea como lo observamos en el Gráfico 3-1 en donde el 89% de agricultores utilizan maquinaria para la preparación del suelo, el 3% para el control de malezas y el 1% para la fertilización, además podemos ver el porcentaje de maquinaria para la siembra es nulo (MAGAP, 2018).

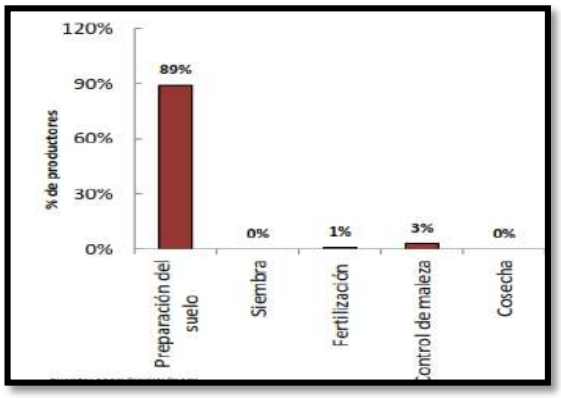


Grafico 3-1: Mecanización por Tareas
Fuente:(MAGAP, 2018)

Cada año la única labor mecanizada con mayor porcentaje a nivel nacional ha sido la preparación del suelo debido a que no existe en el país ningún tipo de maquinaria capaz de cumplir las otras labores tales como siembra, fertilización, control de maleza y cosecha es por eso que mediante el estudio se logra ayudar a incentivar a crear maquinaria autónoma que ayude a cada una de estas tareas tratando de eliminar la mano de obra, esta investigación es basada especialmente para terrenos que sean menores a 4 hectáreas los cuales son trabajados por pequeños productores.

1.2 **Proceso de siembra.**

1.2.1 ***Zonificación y elección del tubérculo***

En una zona de siembra los factores predominantes para realizar el cultivo de papa deben tener una temperatura de 9 a 11°C, un PH acido de 5 a 6, además de existir abundante materia orgánica, debe estar fuera de zonas donde exista heladas y granizadas (Muñoz and Cruz, 1984).

Su variedad depende de las zonas de la sierra como son: norte central y sur, la zona Norte se encuentran la provincia de Carchi, en la zona central las provincias de, pichincha Cotopaxi y Tungurahua y finalmente en la zona sur las provincias de bolívar, Chimborazo, cañar y Azuay (Muñoz and Cruz, 1984).

De acuerdo a las zonas existe una gran variedad de papas, pero las comunes y de mayor consumo a nivel nacional son: Chaucha, Súper chola, Fripapa, única, Gabriela, Leona, Puzza, la Súper chola es la más utilizada por los agricultores con un 55% y con un rendimiento de promedio de 21t/ha, seguida la papa Única, Fripapa y Gabriela con (8%,5%,4%) con un tonelaje de (19,16,23)t/ha como se puede ver en la Grafico 4-1(MAGAP, 2018).

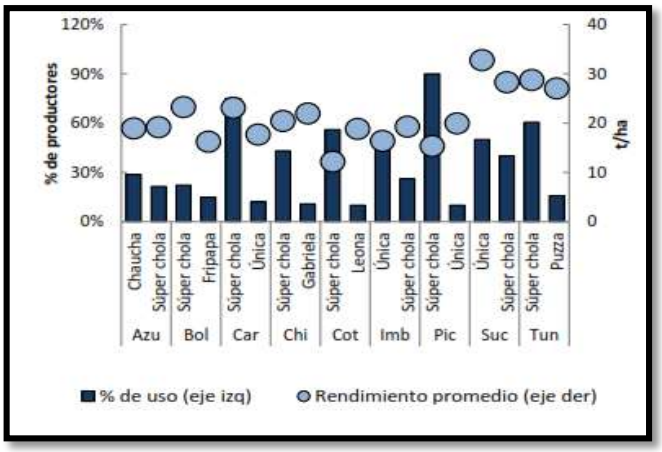


Grafico 4-1: Variedades de Papa en el Ecuador.
Fuente: (MAGAP, 2018)

1.2.2 *Preparación del suelo*

Para una siembra adecuada uno de los papeles más importantes es la preparación del suelo con un tiempo de anticipación debido a la presencia de residuos o en muchos casos de malezas para así tener un tiempo de descomposición de las misma (Miguel Roman y Guillermo Hurtado, 2002), todo esto depende del tipo de suelo elegido y de la rapidez de la descomposición de materias, residuos vegetales teniendo en cuenta con una duración de dos a tres meses. En caso de ocupar suelos con rastrojos es decir haya tenido otros cultivos como cebada, maíz, trigo, el proceso de descomposición de materia verde es de 3 a 4 semanas dependiendo de la humedad. (Muñoz and Cruz, 1984).

Se debe tomar en cuenta la elección del tipo de terreno ya sea ladera o plano con poca inclinación, el primero por tener una inclinación se lo realiza manualmente mediante azadón esta herramienta es encargada para cortar malezas, separar la tierra es decir dejarle sin grandes terrones y finalmente formar surcos; la segunda por ser de más fácil acceso se emplea acarreo animal o maquinaria, cuando prepara el terreno mediante maquinaria o acarreo animal se lo hace a una profundidad de 30cm para levantar las malezas del suelo el siguiente paso es soltar los terrones de tierra y finalmente se realiza el surcado (Miguel Roman y Guillermo Hurtado, 2002) .

1.2.3 Siembra

La siembra se realiza con las denominadas "papas semillas", siendo estos tubérculos con pequeñas fragmentaciones, estos tubérculos se introducen en la tierra con una profundidad aproximada de 5 a 10 cm (FAO, 2008), además de considerar la distancia entre semillas y la distancia entre surcos si estas dichas medidas ocupan una mayor superficie, los tubérculos pueden obtener un mayor tamaño y alcanzar más nutrientes, de lo contrario a menor distancia los tubérculos son de menor tamaño, según estudios realizados se recomienda sembrar una distancia entre tuberculos de 15cm a 25cm y entre surcos una distancia de 90cm (Guillermo Albornoz, 1968).

El número de tubérculos sembrados por los agricultores según estudios realizados por el MAGAP son el 71% siembran dos tubérculos, el 15% un tubérculo y el 14% tres tubérculos, obteniendo como resultado que el sembrío de un tubérculo tiene mejor rendimiento con una diferencia de 1 tonelada más que los otros sembríos como se puede observar en la Grafico 5-1 (MAGAP, 2018).

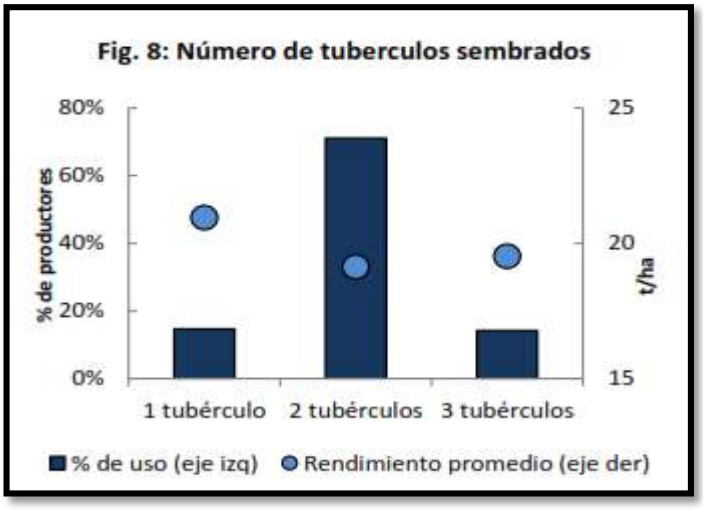


Grafico 5-1: Números de tubérculos sembrados
Fuente:(MAGAP, 2018)

1.2.3.1 *Tipos de Siembra.*

Manual: este se lo realiza abriendo surcos de manera que estén en paralelo mediante azadón o tracción animal, después la semilla es depositada en los surcos al final se tapa la semilla con un rastrón de madera o incluso con el mismo azadón esto se lo realiza en el mismo sentido o de forma transversal al surco (H. Guglielmetti, 1984).

Mecanizado: Dependiendo de la máquina esta puede de realizar pequeños surcos para su sembrado, depositar las semillas y al final taparlas, además es de ayuda en la economía por disminuir el costo de la mano de obra y por perfeccionar la siembra (H. Guglielmetti, 1984).

1.2.4 Características del productor

El promedio general del productor es de 47 años de edad, la mayoría afirman el cultivo de papa es tradición familiar por tres generaciones, además de ser una principal fuente de ingreso para sus hogares, en cuanto a su educación tiene 7 y 8 años enseñanza, además de un 26% de estos productores no pertenecen a una asociación, así como máximo un 6% tienen capacitación de este tipo de cultivo como se puede observar en la Tabla 1-1 (MAGAP, 2018)

Tabla 1-1: Características del productor de las principales provincias productoras de papas.

Provincia	Edad del Productor	Generaciones	Nivel de Educación(Años promedio)	Origen del principal ingreso mensual	Capacitación (%)	Asociatividad (%)
Sucumbíos	46	3	8	Empleo parcial y producción de otro cultivo(60%)	0%	0%
Tungurahua	50	3	7	Producción de cultivo(61%)	21%	3%
Carchi	44	3	8	Producción de cultivo y comercio/negocio propio(64%)	26%	6%
Chimborazo	51	3	5	Producción de otro cultivo(51)	19%	6%

Fuente:(MAGAP, 2018)
Realizado: Gonzalez Henry & Carrillo Maria**,** 2019

1.3 **Robots agrícolas**

La humanidad en estos últimos años se ha preocupado por el medio ambiente, agua, los gases tóxicos generados por las grandes industrias incluyendo para lograr estos cambios (Bugallo, Cortés. and Jaimes, 2016), los robots encargados de facilitar las tareas de exuberantes esfuerzos, alto riesgo y fatiga (Físicas, 2005), en este campo de la robótica hay una rama con mucha fuerza como lo es la robótica agrícola aquella ha evolucionado permitiendo realizar trabajos en la intemperie, invernaderos e incluso en azoteas paredes verticales etc. Estos robots pueden hacer una diversidad de tareas desde el desyerbado, poda, cosecha hasta el empaque de frutos (Bugallo, Cortés. and Jaimes, 2016).

1.3.1 ***Robots manipuladores***

Son máquinas automáticas, reprogramables y multifuncionales, las cuales pueden ser móviles o fijas utilizadas a nivel industrial (Ministerio de España, 2018) , son capaces de realizar tareas peligrosas, repetitivas, tediosas para las personas, tratando de reemplazar la toma de decisiones, la capacidad de trabajar con objetos y herramientas (Díaz, 2017).

En la Figura 2-1 se puede observar un robot japonés automático capaz de realizar el trabajo de jardinería el cual se encarga de recoger y cortar crisantemos, este robot para quitar las hojas de la flor y verifica su entorno lo hace atreves de visión artificial (Físicas, 2005).

Figura 1-1: Robot recogedor de crisantemos
Fuente:(Físicas, 2005)

1.3.1.1 *Robot Fumigador.*

Según estudios realizados por el Instituto de Tecnología Agropecuaria ITA, se creó un robot autómata capaz de realizar la labor de fumigación de la tierra mediante químicos , los cuales son perjudiciales para la salud del ser humano, siendo este robot totalmente eléctrico al cual se lo denomino TRAKÜR que significa niebla por propagar agroquímicos sobre las siembras, de esta manera TRAKÜR controla las plagas de acuerdo a los estudios técnicos antes realizados ,su funcionamiento consta de un cable colocado en el piso como trayectoria el cual genera un campo electromagnético por donde atraviesa corriente, su altura es de 170cm, largo 105cm y ancho 60cm, consta de una batería de 12V y soporta un peso máximo de 100Kilos, como se puede observar en la Figura 3-1, además está pensado para trabajar alrededor de 8 horas, este robot fue creado con el objetivo de ayudar a no afectar la salud de los trabajadores y al mismo tiempo pueda ser de fácil obtención por su bajo costo (PromueveHidroponia, 2014).

Figura 2-1: Robot fumigador TRAKÜR
Fuente:(PromueveHidroponia, 2014)

1.3.1.2 *Robot eliminador de malezas.*

Hortibot es un Robot autónomo coordinado por la Universidad de Aarhus, en el Instituto de Ingeniería Agrícola y el Centro de Investigación Bygholm es el encargado del cuidado de plantas, además de poder transitar sobre filas visibles y reconocer 25 tipos de malezas, para poder eliminarlas, está constituido de un taster encargado de dar una descarga eléctrica a la maleza además de utilizar un software ya existente en el mercado para la localización de las filas como se puede observar en la Figura 4-1 (Bill Christesen, 2007).

Figura 3-1: Robot eliminador de malezas Hortibot
Fuente:(Bill Christesen, 2007)

1.3.1.3 *Robot recolector*

El proyecto CROPS impulso a la creación de un robot capaz de realizar una cosecha selectiva de frutas, hortalizas, uvas, etc., esta máquina tiene como peculiaridad descubrir su grado de madurez para después recogerla con sutileza, como se puede observar en la Figura 5-1 además tiene la capacidad de valoración de obstáculos y otros objetos, este robot puede desplazarse de una forma autónoma y desenvolverse en todo tipo de huertos de árboles frutales (CORDIS, 2016).

Figura 4-1: Robot recolector
Fuente: (CORDIS, 2016)

La robótica en la agricultura a nivel mundial ha evolucionado considerablemente debido a la creación de maquinarias autónomas capaces de realizar ciertas labores como la fumigación, eliminación de malezas, recolección de frutos, sin embargo, la mayoría de estas máquinas

autómatas son de un valor elevado y sus labores son llevadas a cabo en terrenos con grandes superficies de sembríos. Ecuador siendo un país netamente agrícola no cuenta con la maquinaria por sus altos costos además la mayoría de agricultores son pequeños productores sin tener las posibilidades para comprar estas máquinas es por eso que el prototipo implementado buscara la manera de ser de bajo costo.

1.4 Tarjetas de Desarrollo.

Conocidas también como tarjetas para desarrollo de hardware, estas tarjetas contienen placas electrónicas y tiene como particularidad un lenguaje de programación, además de tener una variedad de características en la cual se puede realizar diversas aplicaciones, a medida que va la tecnología creciendo va aumentando diversos tipos de placas de desarrollo, estas palcas tienen entradas y salidas digitales, entradas y salidas analógicas, puerto USB, memoria para el almacenamiento de datos entre otros esto, depende del tipo de placa y de los fabricantes de las mismas, entre las más conocidas tenemos: Raspberry Pi, Intel Galileo, Arduino.

1.4.1 Raspberry Pi.

Conocido como un ordenador de pequeño tamaño contiene un CPU, ARM1176JZF, GPU (procesador gráfico), memoria RAM de 512MB, entradas y salidas analógicas y digitales, lector de tarjeta SD, lector de tarjetas SD, puertos USB entre otros, como se puede observar en la figura 6-1 , esta tarjeta fue creada con el fin de incentivar a los estudiantes de escuelas (Museo Informática, 2013).

Figura 5-1: Tarjeta Raspberry Pi.
Fuente:(Museo Informática, 2013).

1.4.2 Intel Galileo.

Se basa en la arquitectura Intel sistema Intel Pentium de 32 bits, esta tarjeta se creó con el fin soportar voltajes de 3.3V a 5V, es semejante a Arduino y para sí programación se la puede realizar con el software de Arduino, puede comunicarse con computadores, microcontroladores y Arduino, y proporciona una gran cantidad de funciones de fácil manejo para los estudiantes, así como para profesionales como se la pude observar en la Figura 7-1 esta tarjeta contiene físicamente puertos USB, puerto Ethernet ,ranura para una Micro SD conectores de alimentación ,pines digitales (ARDUINO, 2019).

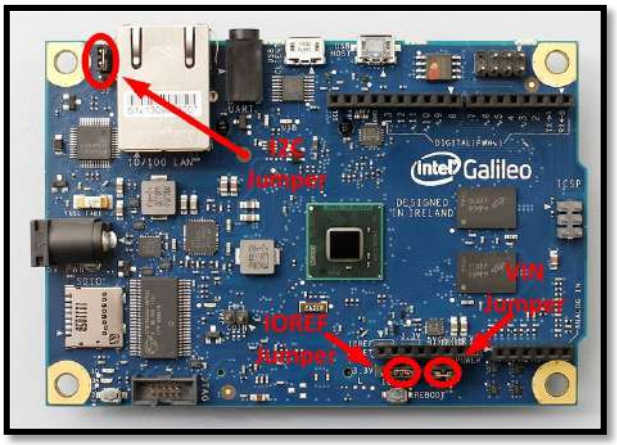


Figura 6-1: Tarjeta Intel Galileo
Fuente:(ARDUINO, 2019)

1.4.3 Arduino.

La creación de esta tarjeta fue con la finalidad de crear prototipos de una manera fácil y rápida para estudiantes sin conocimientos de programación es así Arduino se adopta a las necesidades de cada individuo debido a una plataforma de código abierto de fácil uso, por ejemplo, puede activar o encender un led mediante instrucciones enviadas al microcontrolador, es utilizada para la creación de proyectos de bajo costo, es multiplataforma se ejecuta en Linux, Macintosh OS X y Windows, Arduino está compuesto físicamente por pines digitales y analógicos puerto USB, botón de reset, entrada y salida seriales como se puede observar en la Figura 8-1 (ARDUINO, 2019)

Figura 7-1: Arduino UNO
Fuente:(ARDUINO, 2019)

1.4.4 Tabla comparativa

Luego de haber analizado cada una de las tarjetas de desarrollo se procede a realizar una tabla comparativa entre sus características como se puede observar en la tabla 2-1 Raspberry carece de Memoria Flash y E/S analógicas aso como Arduino UNO no Contiene Ethernet.

Tabla 2-1: Tabla comparativa de las tarjetas de desarrollo.

Características	Raspberry pi (Modelo b)	Intel Galileo (Gen 1)	Arduino Uno
Procesador	ARM11	SoC Quark X100	ATMega328
Voltaje de operación	3.3V/5V	3.3V/5V	5V
Voltaje de entrada	5V	5V	7-12V
RAM	512MB	512KB	2KB
Memoria Flash	-	8MB	32KB
Puerto USB	2	2	1
Velocidad de reloj	700Mhz	400MHz	16MHz
E/s Analógicas	-	6	6
E/S digitales	8	14	14
Ethernet	100	10/100	No contiene

Fuente: Altamirano y otros.
Realizado: Gonzalez Henry & Carrillo Maria**,** 2019

1.4.5 Sensores de distancia.

Para escoger un tipo de sensor se realizó una Tabla 3-1 comparativa de tres diferentes tipos de sensores existentes en el mercado para distancia cada uno posee su principal característica de comparación.

Tabla 3-1: Tabla comparativa de sensores de proximidad.

Sensores	Rango	Señal de salida	Alimentación	Tiempo
Sensor de distancia por rango ajustable	35cm-280cm	4-20mA 0-10V	18-30Vdc	3ms
Sensores Fotoeléctrico Infrarrojos E18-D80NK	3cm-80cm	100mA	5-24Vdc	2ms
Sensor de ultrasonidos con salida analógica	35cm-110cm	0-10V	18-30Vdc	<=500ms
Sensor de distancia HC-SR04	2cm-400cm	15mA	5V	20ms

Fuente: Gonzalez Henry & Carrillo Maria, 2019
Realizado: Gonzalez Henry & Carrillo Maria, 2019

1.4.6 Sensores de desplazamiento.

En la Tabla 4-1 se muestra una tabla comparativa de sensores con la capacidad de medir distancia y transformarlos mediante algoritmos en centímetros recorridos cada uno con sus especificaciones técnicas.

Tabla 4-1: Tabla comparativa de sensores de Desplazamiento.

Sensores	Velocidad máxima	Alimentación	Salidas	Respuesta en frecuencia
Encoder incremental B58N	6000RPM	5 a 26 Vcc	HTL(5-26Vcc) TTL(5Vcc) 40 mA	125kHz
Módulo de velocidad Encoder B83609	4000RPM	4.5V a 5.5 V	200Ma	100kHz
Sensor encoder velocidad herradura	20RPM	3.3V-5V	5V	300kHz

Fuente: Gonzalez Henry & Carrillo Maria, 2019
Realizado: Gonzalez Henry & Carrillo Maria, 2019

1.4.7 Transmisores y receptor de señal.

En la Tabla 5-1 se puede observar distintos módulos de transmisión y recepción de señal con sus principales características tomando en cuenta que solo uno de los módulos es receptor y transmisor como lo es la NodeMCU.

Tabla 5-1: Tabla comparativa de sensores de Desplazamiento.

Sensores	Voltaje de Operación	Corriente	Frecuencia	Alcance
Módulo RF 433MHz TX	3.3V-5V	40mA	433MHz	20m-200m
Módulo RF 433MHz RX	3.3V-5V	4mA	433MHz	20m-200m
NodeMCU SP8266 V12.	3.3V	500mA	80MHz	10m
Modulo Rf emisor con codificador y decodificador	5V-9V	<=2.5mA	433.92MHz	2Km

Fuente: Gonzalez Henry & Carrillo Maria, 2019
Realizado: Gonzalez Henry & Carrillo Maria, 2019

CAPITULO II

2 MARCO METODOLÓGICO.

En este capítulo se detallará el diseño de hardware y software del prototipo de robot sembrador de papa en terrenos sin inclinación para pequeños productores denominado prototipo AGSEM, especificando las etapas que componen el prototipo, sus componentes, esquemas eléctricos y electrónicos, esquemas mecánicos y planos de la estructura realizada.

2.1 Requerimientos del prototipo AGSEM.

Los estudios realizados en el capítulo I se determinó los requerimientos necesarios para obtener un proceso de sembrado óptimo.

El prototipo debe satisfacer los siguientes requerimientos:

- Implementar un prototipo autónomo con la capacidad de reaccionar ante los obstáculos del terreno.
- El sistema implementado sea capaz de controlar los parámetros de trayectoria predefinida.
- Tenga la habilidad de reaccionar a la configuración de parámetros iniciales para cumplir con el proceso de siembra.
- El sistema sea capaz de transmitir y recibir datos en tiempo real inalámbricamente a través de una aplicación móvil.
- Enviar alertas si el sistema implementado así lo necesita.
- Permitir la visualización de datos recolectados en el transcurso del proceso de siembra.

2.1.1 Concepción del prototipo AGSEM.

El diseño del prototipo AGSEM se observa en la Figura 1-2, apreciando el proceso autónomo de siembra y el proceso a cumplir en cada una de sus etapas.

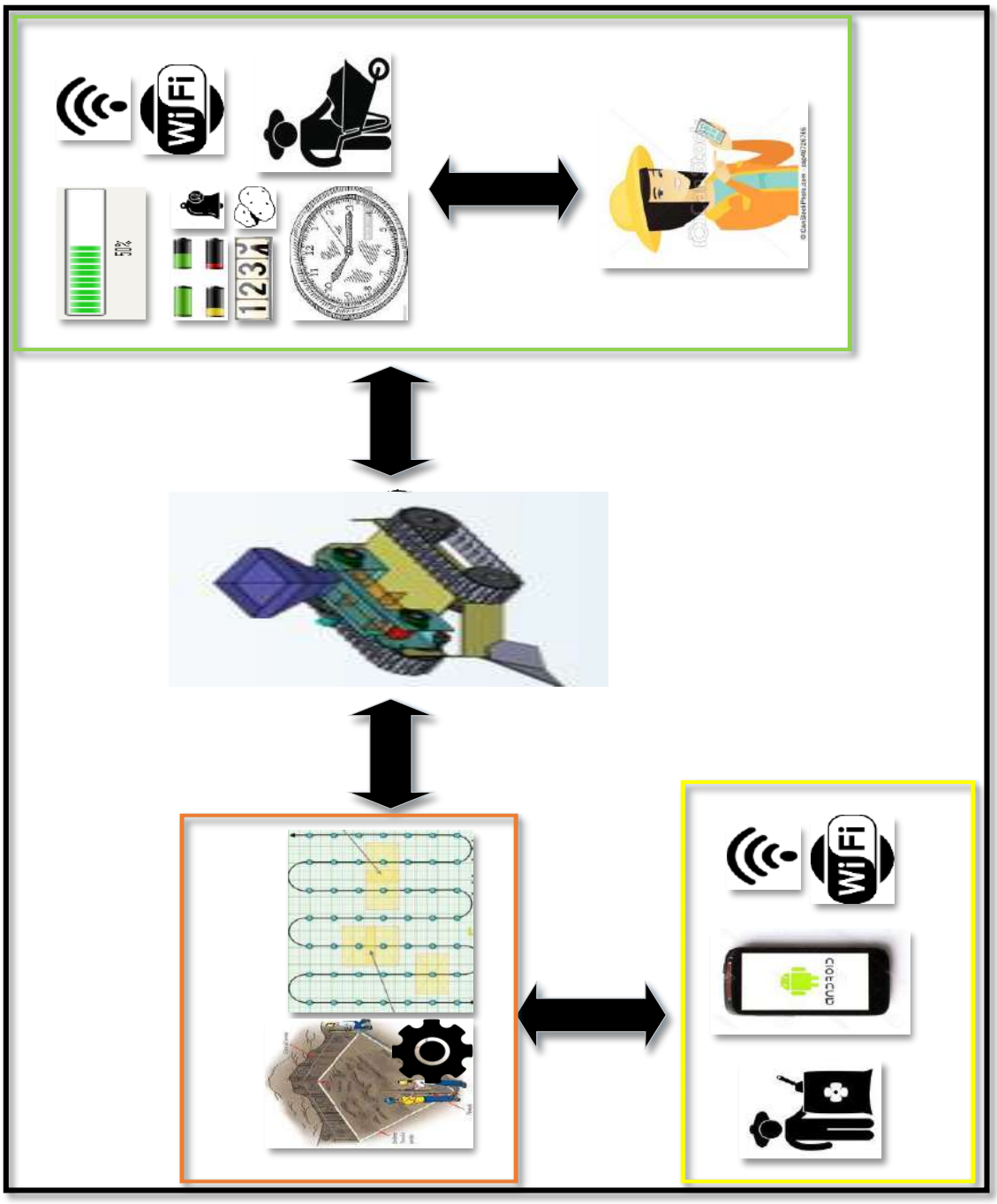


Figura 1-2: Concepción del Prototipo AGSEM.
Realizado por: Gonzalez Henry & Carrillo Maria**.** 2019

Etapa de visualización. Esta etapa realiza el trabajo de observación de datos tales como número de semilla, nivel de carga en la tolva, nivel de energía, distancia recorrida, también sus mensajes de advertencia, obtenidos en tiempo real en una determinada zona, a través de una aplicación móvil mostrando de manera principal la sincronización de la NodeMCU (SP8266 V12), evitando así la perdida de datos, todo esto se realiza a través de un módulo wifi integrada en la NodeMCU y un repetidor de señal wifi el cual deberá ser integrado en el terreno para tener un alcance más efectivo.

Después de la sincronización la visualización pedirá parámetros de inicio como dimensión del terreno y el dibujo de la trayectoria, la NodeMCU trabajando directamente desde su aplicación móvil permitiendo al HMI ser más dinámico en cualquier dispositivo.

Etapa de aviso. Es la encargará de enviar avisos de errores por medio de la NodeMCU los cuales se pueden dar durante el transcurso del proceso de siembra del prototipo AGSEM, los sensores como encoders, sensores infrarrojos, deberán enviar notificaciones al sistema como advertencias de nivel de carga en la tolva, nivel de energía, todos estos sensores son conectados y programados en el Arduino en sus puertos digitales para ser procesados y transmitidos por la NodeMCU hasta el punto de finalización de siembra o recarga de tubérculos.

Etapa de control. Esta etapa interviene el controlador en los motores programado en el Arduino uno de manera de PWM para la tracción y el transporte de los tubérculos desde la tolva a la tierra según técnicas agrónomas predefinidas en el Capítulo I, así como también el control del giroscopio el cual permitirá al robot continúe su trayectoria de una manera recta el cual es colocado y programado en la NodeMCU.

Etapa de almacenamiento. Esta etapa por medio de la comunicación entre la NodeMCU y la aplicación móvil nos permitirá tener el control en tiempo real de los tubérculos entregada por el prototipo AGSEM sembrados hasta la culminación de su trayectoria o recarga de tubérculos, así como el tiempo trascurrido durante su proceso de siembra.

2.1.2 Diseño de la arquitectura del prototipo AGSEM

En la Figura 2-2 se muestra el diagrama de bloques de la arquitectura del prototipo AGSEM según las etapas de la concepción, su bloque principal es la obtención de datos por medio de la lectura de los sensores como sensor tipo encoder, infrarrojo reflectivo y giroscopio del módulo MPU, estos datos ingresan al bloque de procesamiento para ser analizada por la tarjeta de desarrollo arduino y mediante algoritmos permitirá ingresar al bloque de configuración de parámetros de siembra para ser transmitidos en forma inalámbrica por medio del microcontrolador NodeMCU y receptar los datos de forma legible en el módulo de visualización en el dispositivo móvil para ser enviadas al bloque de almacenamiento para el proceso de siembra.

Las alimentaciones de tarjetas de desarrollo, sensores y transmisores serán energizadas por medio de una batería de litio entrega una corriente de 5Ah.

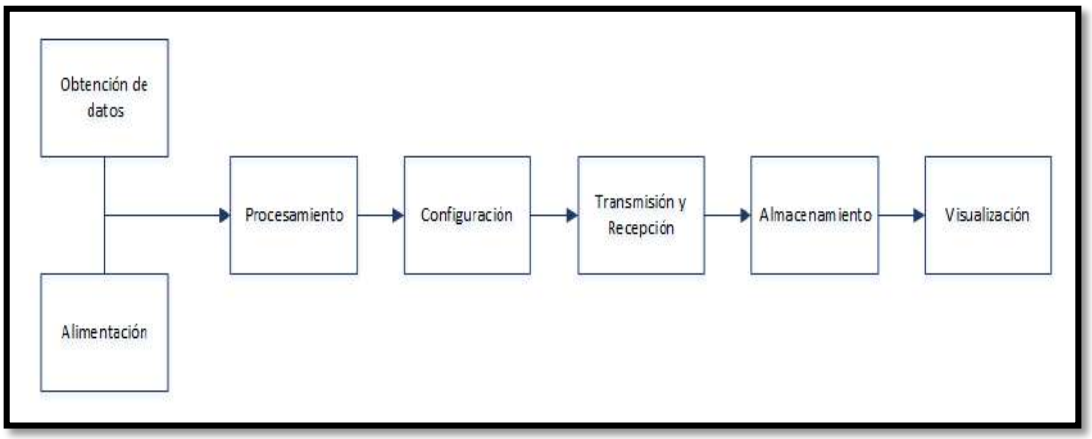


Figura 2-2: Diseño de la Arquitectura del Prototipo AGSEM.
Realizado por: Gonzalez, Henry & Carrillo, Maria**.** 2019

2.1.3 Selección de dispositivos que conforman el prototipo AGSEM.

Para el funcionamiento del prototipo es necesario la utilización de tarjetas de desarrollo para el procesamiento y adquisición de datos emitidos por los sensores, así como de sus elementos de transmisión detallados a continuación estos dispositivos fueron seleccionados gracias a las características estudiadas en el capítulo anterior

Arduino uno. – En la Figura 3-2 se muestra la placa de desarrollo Arduino Uno basada en un microcontrolador ATmega328P, cuenta con 20 terminales programables de entrda y salida, con un cristal de cuarzo 16mhz, conexión usb para subir la programación desde su software de desarrollo Arduino IDE, la placa debe ser alimentada de 7V a 12V con una corriente dc de 20mA y corriente cc de 50mA (Arduino, 2017).

Figura 3-2: Microprocesador Arduino UNO.
Realizado por: Gonzalez, Henry & Carrillo, Maria**.** 2019

CARACTERÍSTICAS PRINCIPALES

- Memoria: 32kb
- Consumo de Corriente: 200ma
- Terminales digitales: 13
- Terminales analógicas: 6
- Terminales de poder: 5

NodeMCU SP8266 V12. – En la Figura 4-2 se muestra el controlador cuenta con 6 pines digitales, capacidad de conectividad inalámbrica incorporada, antena PCB incorporada en el chip con un alcance de hasta 10 metros sin necesidad de repetidor, un módulo de interfaz usb de comunicación serial, compatible con el software de desarrollo Arduino Ide instalada con Phyton, requiere una alimentación de 3.3V con una corriente de 12mA a 200mA (EINSTRONIC, 2017).

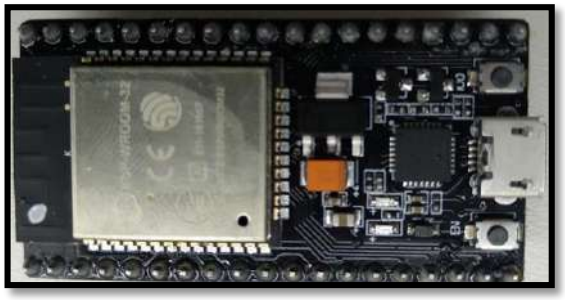


Figura 4-2: Microprocesador NodeMCU
Realizado por: Gonzalez, Henry & Carrillo, Maria**,** 2019

CARACTERÍSTICAS PRINCIPALES

- Memoria: 36kb
- Consumo de Corriente: 170ma
- Temperatura: -40 a 125 °C
- Terminales: 22

Módulo de velocidad Encoder B83609. – En la Figura 5-2 se observa el sensor Encoder permite controlar la posición y velocidad de robot, cuenta con dos leds infrarrojos un emisor un receptor para poder medir la posición angular y transformarlas a pulsos eléctrico, se alimenta de 4.5V a

5.5V con una corriente de 12mA a 200mA, compatible con módulos Arduino y software de desarrollo Arduino Ide (Sensor, 2019).

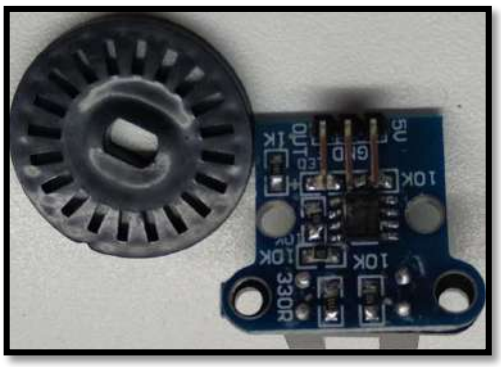


Figura 5-2: Modulo de Velocidad Encoder B83609
Realizado por: Gonzalez, Henry & Carrillo, Maria**,** 2019

CARACTERÍSTICAS PRINCIPALES

- Tamaño: 9*5*2cm
- Consumo de Corriente: 20ma
- Peso: 13g
- Terminales: 3
- Diametro de Disco: 24mm
- Endiduras de Disco: 20

Sensores Fotoeléctrico Infrarrojos E18-D80NK. – En la Figura 6-2 se muestra un sensor óptico reflectivo E18-D80NK capaz de detectar la presencia de objetos con un rango de distancia de 3 a 80cm, trabajo con un voltaje de 5V – 24 V DC con una corriente máxima de 100mA.

Figura 6-2: Sensor Fotoeléctrico Infrarrojo E18-D80NK.
Realizado por: Gonzalez, Henry & Carrillo, Maria**, 2019**

CARACTERÍSTICAS PRINCIPALES

- Tiempo de Respuesta: 32Kb
- Consumo de Corriente: 20ma
- Temperatura: 13 °C

Sensor MPU6050. – En la Figura 7-2 se muestra un sensor giroscopio y acelerómetro, el cual mide aceleraciones lineales y angulares, siendo capaz de trabajar con 6 grados de inclinación con respecto a los 3 ejes, tiene una resolución de 16 bits dividiendo el rango dinámico en 65536 aplicadas a cada eje X, Y, Z, con una alimentación de 2.37V a 3.46V y una corriente de 100mA ,el registro I2C regresa valores en manera de RAW en la Tabla 1-2 se muestra el rango de escala y los valores de Raw (MPU6050 Arduino, Acelerómetro y Giroscopio, 2014).

Tabla 1-2: Rango de valores Sensor MPU6050.

Rango de Escala Giroscopio	Sensibilidad del Giroscopio	Rango de Escala Acelerómetro	Sensibilidad del Acelerómetro
± 250	131	± 2	16384
± 500	65.5	± 4	8192
± 1000	32.8	± 8	4096
± 2000	16.4	± 16	2048

Fuente: (MPU6050 Arduino, Acelerómetro y Giroscopio, 2014)**.**
Realizado por: Gonzalez, Henry & Carrillo, Maria**, 2019**

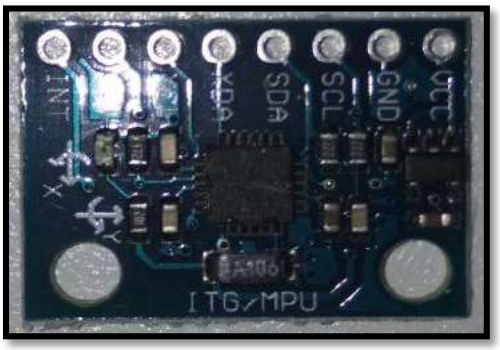


Figura 7-2: Sensor MPU6050.
Realizado por: Gonzalez, Henry & Carrillo, Maria**,** 2019

CARACTERÍSTICAS PRINCIPALES

- Tiempo de Respuesta:30ms
- Consumo de Corriente: 200ma

- Memoria: 16 bits
- Terminales: 8
- Temperatura: -40 a 85 °C

Módulo de control para motor Monster. – Es un controlador para motores, tiene un controlador de lado alto monolítico doble y dos interruptores de lado bajo. El interruptor alto permite una integración eficiente en el mismo torque gracias a un Power MOSFET integrado permite también su protección a sobrecargas, ayuda al encendido o apagado de cada motor trabaja con un rango de voltaje de 5.5V a 16V y una corriente máxima de 30A, una amplitud de frecuencia de 20KHz, se programa gracias a la creación de una biblioteca en el software de desarrollo Arduino Ide. Su presentación de un canal en la Figura 8-2 (VNH2SP30, 2018).

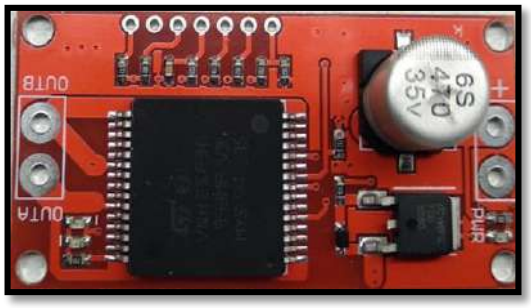


Figura 8-2: Puente H Monster de 1 canal.
Realizado por: Gonzalez, Henry & Carrillo, Maria**,** 2019

CARACTERÍSTICAS PRINCIPALES

- Corriente del PWM: +- 10 ma
- Consumo de Corriente: 200ma
- Terminales: 4
- Temperatura: -40 a 150 °C

Módulo de control para motor IBT-2 7960. – En la Figura 9-2 se visualiza un puente H capaz de soportar una corriente máxima de 43A con un voltaje de alimentación de 5.5V a 28V, cuenta con cinco conectores de trabajo 2 de alimentación, 1 PWM y 2 para la salida del motor, tiene un chip incorporado 74HC244 de 8 bits siendo el microcontrolador para el motor, trabaja a una frecuencia máxima de 25KHz, una altura de 38mm, un largo y ancho de 46mm.

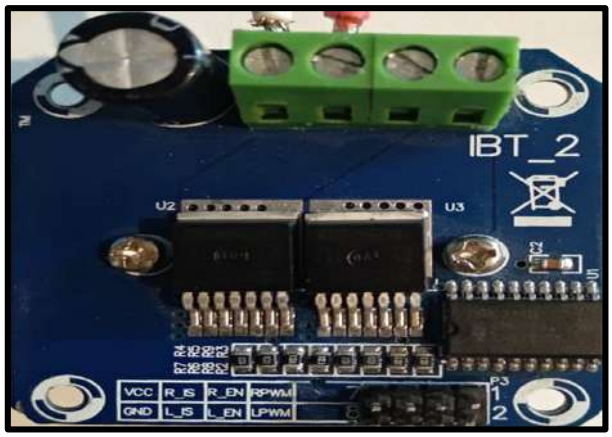


Figura 9-2: Modulo de control para motor IBT-2 7960
Realizado por: Gonzalez, Henry & Carrillo, Maria**,** 2019

CARACTERÍSTICAS PRINCIPALES

- Tiempo de Respuesta: 0.5 a 22.4 us
- Consumo de Corriente: 0.5ma
- Terminales: 4
- Temperatura: -40 a 85 °C

Convertidor DC-DC KIM-055L. – En la Figura 10-2 se muestra un convertidor reductor de voltaje DC-DC siendo un dispositivo capaz de reducir su voltaje de 12V a 5V con una corriente máxima de 8A.

Figura 10-2: Convertidor DC-DC KIM-055L.
Realizado por: Gonzalez, Henry & Carrillo, Maria**,** 2019

CARACTERÍSTICAS PRINCIPALES

- Dimensiones: 22*17*4 mm
- Consumo de Corriente: 0.5ma
- Terminales: 3

- Temperatura: -40 a 85 °C
- Eficiencia: 93%

Batería. – Son celdas unidas en serie empleadas en robots o dispositivos de alto consumo eléctrica en la Figura 11-2 se muestra la batería de alimentación para el prototipo AGSEM siendo de 12V a 5A con dos terminales un positivo y negativo.

Figura 11-2: Batería de 12V a 5A.
Realizado por: Gonzalez, Henry & Carrillo, Maria**,** 2019

2.2 Esquema de conexión electrónica para el prototipo AGSEM.

El proceso de conexión de los dispositivos implementados y programados en los microcontroladores NodeMCU y Arduino uno, para el funcionamiento del para el prototipo AGSEM son detallados a continuación.

Sensor acelerómetro y giroscopio MPU6050.- La conexión entre el módulo MPU con el microprocesador NodeMCU (SP8266 V12) se observa en la Tabla 2-2, el terminal digital 21 es conectado al SDA, el 22 al SCL el voltaje del 3.3V y GND fueron conectados al MPU respectivamente.

Tabla 2-2: Conexión Modulo MPU6050.

Terminal NodeMCU	Terminal Modulo MPU
21	SDA
22	SCL
3.3v Vcc	3.3v Vcc
GND	GND

Fuente: Gonzalez, Henry & Carrillo, Maria**,** 2019**.**
Realizado por: Gonzalez, Henry & Carrillo, Maria**,** 2019

Módulo de control para motor Monster de 1 canal. – en la Tabla 3-2 se muestra la conexión del módulo con el microprocesador Arduino, donde sus terminales A y B son conectados a los terminales 7 y 8 del Arduino respectivamente, el modulo es energizado al voltaje del Arduino 5v Vcc y GND.

Tabla 3-2: Conexión Modulo Monster.

Terminal Arduino	Terminal Puente H Monster
7	A
8	B
5v Vcc	5v Vcc
GND	GND

Fuente: Gonzalez, Henry & Carrillo, Maria**,** 2019**.**
Realizado por: Gonzalez, Henry & Carrillo, Maria**,** 2019

Módulo de control para motor IBT-2 7960. – La conexión del módulo derecho con el Arduino se vizualiza en la Tabla 4-2, donde los terminal PWM 1 y PWM 2, se conectan a los terminales digitales 5 y 6 del Arduino, la alimentación de voltaje es entregada por el Arduino con los terminales 5v Vcc y GND.

Tabla 4-2: Conexión Modulo IBT-2 7960 Derecho.

Terminal Arduino	Terminal Puente H IBT-2
5	PWM 1
6	PWM 2
5v Vcc	5v Vcc
GND	GND

Fuente: Gonzalez, Henry & Carrillo, Maria**,** 2019**.**
Realizado por: Gonzalez, Henry & Carrillo, Maria**,** 2019

La conexión del módulo Izquierdo con el Arduino se vizualiza en la Tabla 5-2, donde los terminal PWM 1 y PWM 2, se conectan a los terminales digitales 10 y 11 del Arduino, la alimentación de voltaje es entregada por el Arduino con los terminales 5v Vcc y GND.

Tabla 5-2: Conexión Modulo IBT-2 7960 Derecho.

Terminal Arduino	Terminal Puente H IBT-2
10	PWM 1
11	PWM 2
5v Vcc	5v Vcc
GND	GND

Fuente: Gonzalez, Henry & Carrillo, Maria**,** 2019**.**
Realizado por: Gonzalez, Henry & Carrillo, Maria**,** 2019

Modulo Encoder B83609.- En la Tabla 6-2 muestra la conexión del módulo del lado derecho con el microprocesador Arduino, el terminal 3 se conecta con el terminal digital 2 del Arduino, su alimentación corresponde + y – del módulo al 5v Vcc y GND del Arduino respectivamente.

Tabla 6-2: Conexión Modulo Encoder Derecho.

Terminal Arduino	Terminal Encoder Derecho
2	3
5v Vcc	+
GND	-

Fuente: Gonzalez, Henry & Carrillo, Maria, 2019.
Realizado por: Gonzalez, Henry & Carrillo, Maria, 2019

En la Tabla 7-2 muestra la conexión del módulo del lado izquierdo con el microprocesador Arduino, el terminal 3 se conecta con el terminal digital 3 del Arduino, su alimentación corresponde + y – del módulo al 5v Vcc y GND del Arduino respectivamente.

Tabla 7-2: Conexión Modulo Encoder Izquierdo.

Terminal Arduino	Terminal Encoder Izquierdo
3	3
5v Vcc	+
GND	-

Fuente: Gonzalez, Henry & Carrillo, Maria, 2019.
Realizado por: Gonzalez, Henry & Carrillo, Maria, 2019

Sensor infrarrojo reflectivo E18-D80NK. – Para la conexión del sensor ubicado en la tolva con el microprocesador Arduino se muestra en la Tabla 8-2, donde el cable color negro es conectado al terminal digital 12 del Arduino, es energizado mediante el voltaje entregado por el microprocesador 5v Vcc y GND conectador al cable color marrón y azul respectivamente.

Tabla 8-2: Conexión Sensor Infrarrojo Tolva.

Terminal Arduino	Cable Infrarrojo Tolva
12	Negro
5v Vcc	Marrón
GND	Azul

Fuente: Gonzalez, Henry & Carrillo, Maria, 2019.
Realizado por: Gonzalez, Henry & Carrillo, Maria, 2019

Para la conexión del sensor ubicado en la punta con el microprocesador Arduino se muestra en la Tabla 9-2, donde el cable color negro es conectado al terminal digital 13 del Arduino, es

energizado mediante el voltaje entregado por el microprocesador 5v Vcc y GND conectador al cable color marrón y azul respectivamente.

Tabla 9-2: Conexión Sensor Infrarrojo Punta

Terminal Arduino	Cable Infrarrojo Punta
13	Negro
5v Vcc	Marrón
GND	Azul

Fuente: Gonzalez, Henry & Carrillo, Maria**,** 2019**.**
Realizado por: Gonzalez, Henry & Carrillo, Maria**,** 2019

Microprocesador NodeMCU (SP8266 V12). – Para la transmisión de señal se conecta el microprocesador NodeMCU con el microprocesador Arduino como se observa en la Tabla 10-2, donde el Rx y Tx son conectados al Rx y Tx respectivamente de cada microprocesador, la NodeMCU es energizada con el Voltaje del arduino a 5v Vcc y GND.

Tabla 10-2: Conexión Microprocesador NodeMCU

Terminal Arduino	Cable Node MCU
Rx	Rx
Tx	Tx
5v Vcc	Marrón
GND	Azul

Fuente: Gonzalez, Henry & Carrillo, Maria**,** 2019**.**
Realizado por: Gonzalez, Henry & Carrillo, Maria**,** 2019

En la Figura 12-2 se muestra el diagrama de conexión con cada uno de sus elementos que conforman el prototipo AGSEM

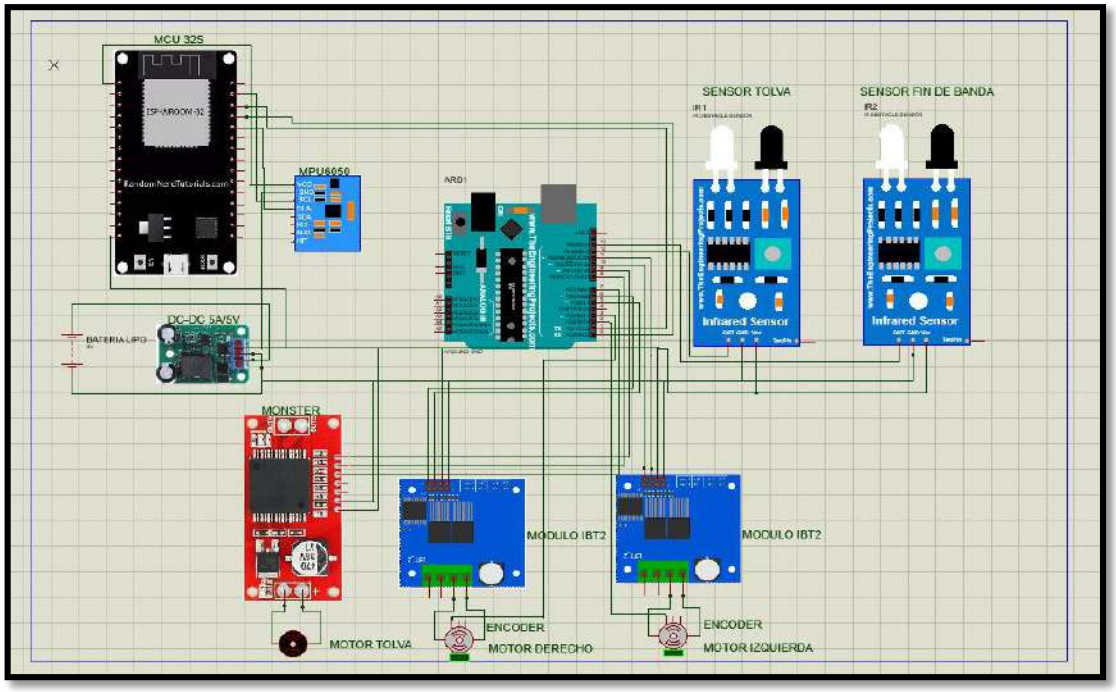


Figura 12-2: Esquema de conexión electrónica.
Realizado por: Gonzalez, Henry & Carrillo, Maria**,** 2019

En la Figura 13-2 se muestra la placa armada con cada uno de los dispositivos que lo conforman y se muestra en funcionamiento para el prototipo AGSEM.

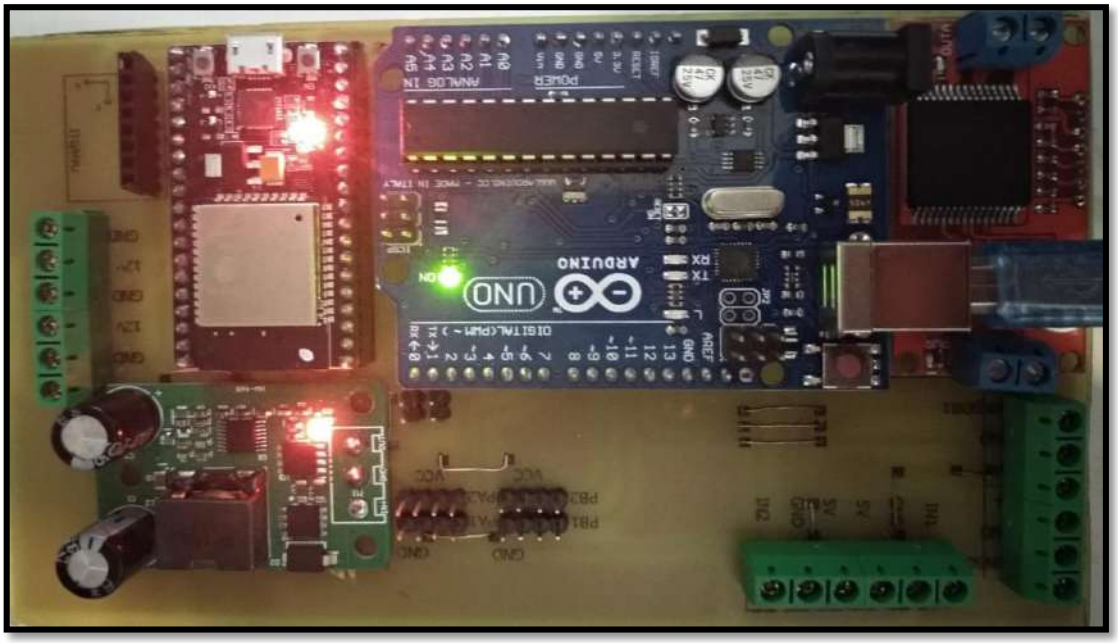


Figura 13-2: Conexión Electrónica de Dispositivos en la Placa
Realizado por: Gonzalez, Henry & Carrillo, Maria**,** 2019

2.3 *Software* para el prototipo AGSEM.

Durante la implementación y programación del prototipo AGSEM se utilizaron diversos *softwares*, detallados a continuación

2.3.1 Software Arduino Ide 1.8.1.

Es un software de desarrollo libre y de código abierto capaz de ser instalado en cualquier sistema operativo, tiene un entorno grafico de programación amigable con el usuario utilizando lenguaje en C y C++, el entorno facilita la programación de los diversos sensores y dispositivos entregados por Arduino, entre las opciones del software nos presenta la compilación, librerías y ejemplos facilitando el manejo de las características de cada dispositivo para Arduino, cuenta una comunicación serial USB 3.0 facilitando la comunicación entre *hardware* y *software* (Paredes, 2017).

2.3.1.1 *Diagrama de Flujo del Prototipo AGSEM.*

Para el control del prototipo su acción principal es la odometria la cual permite al robot moverse de un punto determinado a otro, controlando la velocidad por la Formula 3 de los actuadores trabajando a través del ángulo emitido por el encoder ubicados en cada lado del prototipo utilizando la Formula 1, calculada dicha velocidad el prototipo podrá dirigirse en línea recta del punto inicial al punto final en la Formula 5 deseado el módulo MPU el cual tiene integrado un giroscopio y un acelerómetro, comprueba la velocidad angular de la Formula 4, de cada eje dependiendo su dirección, en este caso se trabaja con el eje x para controlar la velocidad angular, si de existir una variación notable la giroscopio la corregirá para ubicarla en el sentido de la trayectoria predeterminada por la aplicación móvil. La Fórmula 2 permite al robot calcular la distancia recorrida por el prototipo.

- **Fórmulas para Definir la Posición del Prototipo por Odomedria.**

En la Tabla 11-2 se observa las formulas a aplicar para el cálculo de trayectoria en el concepto de odometria

Tabla 11-2: Formula de Odometria eje x

Numero	Formula
1	$\partial' = \partial + \frac{Dd - Di}{L}$
2	$Dc = \frac{Dd + Di}{2}$
3	$Vx = V\cos(\partial)$
4	$w = \frac{Vd - Vi}{L}$
5	$X' = X + (Dc\ x\ \cos(\partial))$

Fuente: Gonzalez, Henry & Carrillo, Maria**,** 2019
Realizado por: Gonzalez, Henry & Carrillo, Maria**,** 2019

- **Donde**

Las variables de las formulas ocupadas para definir la trayectoria se visualiza en la Tabla 12-2.

Tabla 12-2: Variables de Formulas de Odometria

VARIABLE	DEFINICIÓN
V	Velocidad.
Vx	Velocidad eje x.
W	Velocidad Angular.
Vd	Velocidad Derecha.
Vi	Velocidad Izquierda.
Dc	Distancia Central.
Dd	Distancia Derecha.
Di	Distancia Izquierda.
X'	Posición Actual eje x.
X	Posición Anterior eje x.
∂	Angulo.

Fuente: Gonzalez, Henry & Carrillo, Maria**,** 2019
Realizado por: Gonzalez, Henry & Carrillo, Maria**,** 2019

En la Figura 14-2 se observa el diagrama de flujo para el algoritmo de control para el prototipo AGSEM, se declara e inicializa las bibliotecas y variables necesarias para su funcionamiento, la aplicación móvil permite la configuración de los parámetros iniciales como largo, ancho y N° de surcos en el terreno y se dibuja la trayectoria a seguir por el prototipo, se da inicio a la lectura de los sensores implementados en el prototipo.

En las funciones repetitivas utilizadas del algoritmo para el control del prototipo AGSEM nos permite visualizar las advertencias enviadas como inicio, paro, nivel de tolva, nivel de batería. La actualización en tiempo real por medio de la transmisión de datos recolectados por los sensores como, distancia recorrida, tiempo de siembra, numero de tubérculos sembrados serán visualizadas por el usuario en la aplicación móvil, el proceso de siembra por el prototipo termina si la distancia entregada por el encoder cumple con los parámetros de inicio.

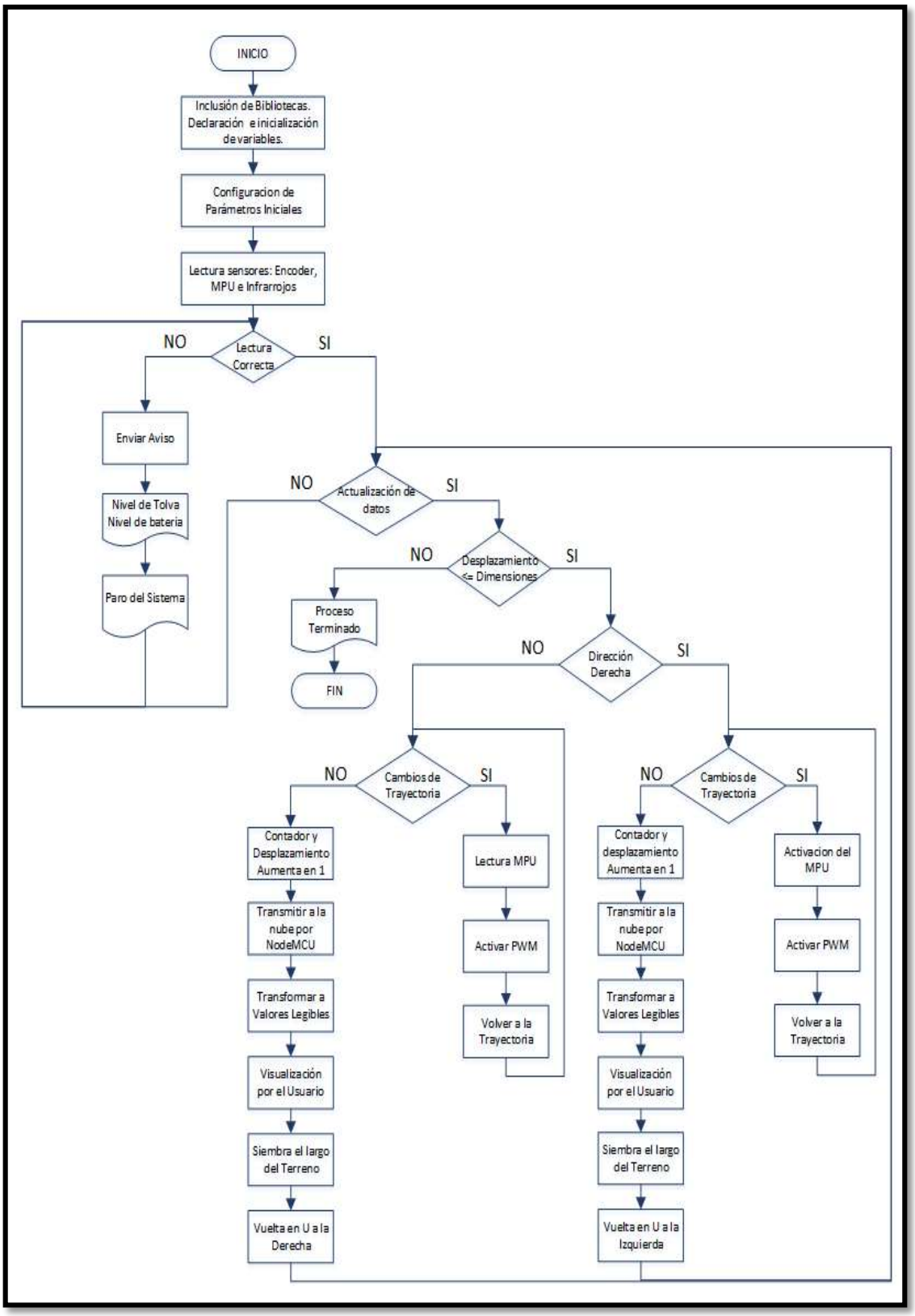


Figura 14-2: Diagrama de Flujo del Prototipo AGSEM.
Realizado por: Gonzalez, Henry & Carrillo, Maria**,** 2019

Para realizar el proceso matemático de transformación según las fórmulas de odometria es importante incluir la biblioteca <math.h> utilizado para el proceso con funciones matemáticas para el desplazamiento y la velocidad del prototipo AGSEM su código se ve en el ANEXO H.

Los encoders calcula los tics obtenidos en un determinado tiempo gracias a las interrupciones que posee el Arduino UNO, el cual nos permite observar las interrupciones tanto para el movimiento del motor izquierdo y derecho en flancos vistos de un flanco de partida negativo hacia un flanco positivo llamado FALLING en el lenguaje de programación el cual utiliza Arduino, evitando al prototipo tener cambios bruscos en su velocidad lineal excepto cuando el robot realiza un giro, para después ser transformados a frecuencias, para ser convertidas en medidas de desplazamiento, mediante funciones matemáticas detalladas a continuación.

Dependiendo del número de pulsos a recibir por el encoder se determinó, a un menor número obtenido de pulsos su ingreso de ruido es mayor, por ende, se decidió realizar un filtro el cual permitirá el paso de los pulsos más exactos realizando la transformación a desplazamiento más exacta. Esto permitirá al control de los motores enviar un PWM capaz de regular la velocidad angular y lineal lo cual permitirá ir hacia delante o hacia atrás y por odometria permitir al prototipo cumplir la ruta planteada.

- **BIBLIOTECAS.**

<I2Cdev.h>.- Activa la comunicación tipo serial i2c con la que el arduino se comunica con algunos dispositivos y sensores.

<MPU6050.h>.- Toma de datos del mpu encargada de la aceleración y ángulos del programa.

<Wire.h>.- Es un complemento para la librería de la comunicación serial.

<math.h>.- Activa todas las funcione matemáticas.

- **FUNCIONES.**

Función Advertencia(). – Esta función enviara las notificaciones a la aplicación móvil para ser evaluadas por el usuario de sus posibles fallos.

Fallo Tolva().- Esta función enviara una notificación si el sensor de la tolva deja de transmitir datos.

Fallo Bateria().- Esta función es encargada de notificar al usuario si el nivel de la batería es más baja al rango permitido.

Void Paro(). Para el sistema si alguna de las anteriores funciones es activada o el usuario desea parar el proceso.

Void loop().- Esta función toma la velocidad angular y lineal para el envió pwm a los motores, y recibe los parámetros iniciales.

Void odometría().- Esta aplica el algoritmo para odometría dado como cálculo de distancia por rueda y posición en el que se encuentra el prototipo según la posición inicial setiada

Void Rencoder().- Es la función para controlar la interrupción de la llanta derecha.

Void Lencoder().- Es la función para controlar la interrupción de la llanta izquierda.

Void Motores().- Es el control de los motores mediante el pwm enviado desde el void loop.

Void Banda().- Es la encargada de transportar el tubérculo hasta el final de la banda para su siembra.

Void Envio.- Envía la condición la actualización en tiempo real del prototipo.

Void contador().- esta función permite el conteo el tubérculo sembrado desde la banda transportdora.

2.3.2 Aplicación Móvil.

La aplicación móvil se realizó en el *software* Android Studio es una plataforma para aplicaciones móviles, compatible con todos los sistemas operativos, orientado para dispositivos con sistema Android, tiene un lenguaje de programación basado en Java para aplicaciones orientadas a objetos como correo electrónico, mensajes, etc. Para programar en Android Studio se debe tener en cuenta con que versión de sistemas Android se va a trabajar, así como su dispositivo, el tipo de Android permite conocer el número de API el cual nos dará a integrar las bibliotecas por ejemplo un Android versión 4.4 denominado KitKat tiene un nivel API 19 es decir contiene 19 bibliotecas, se debe colocar una versión mínima y una máxima de Android el cual permite determinar en que dispositivos son compatibles (Fernandez C, 2018).

Parte del código utilizado para la aplicación del Prototipo AGSEM se visualiza en la Figura 15-2 las bibliotecas definidas por la API gracias a la versión del Android 9 denominada PIE, enviadas y procesadas en el Arduino para continuar una trayectoria predefinida del prototipo AGSEM y la obtención de datos a través de los sensores al Arduino y Transmitidos a la Aplicación móvil de forma inalambrica el código completo se visualizará en el ANEXO I.

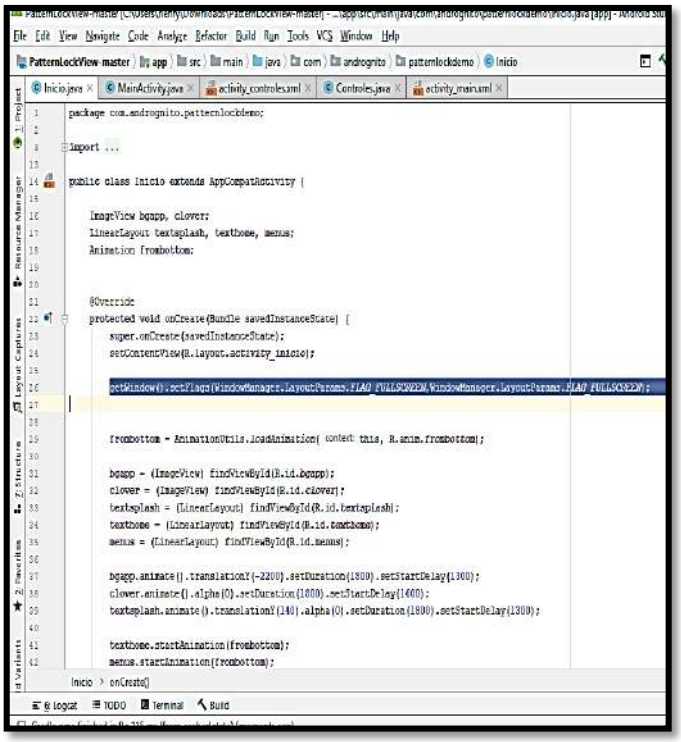


Figura 15-2: Programación en Andorid Studio.
Realizado por: Gonzalez, Henry & Carrillo, Maria**,** 2019

La aplicación móvil está compuesta por la pantalla de inicialización, configuración de parámetros iniciales y supervisión. En la Figura 16-2 se presenta la pantalla de configuración en el cual se coloca los parámetros iniciales como ancho, largo y numero de surcos del terreno. Se especificará la trayectoria a seguir por el prototipo AGSEM para cumplir el proceso de siembra.

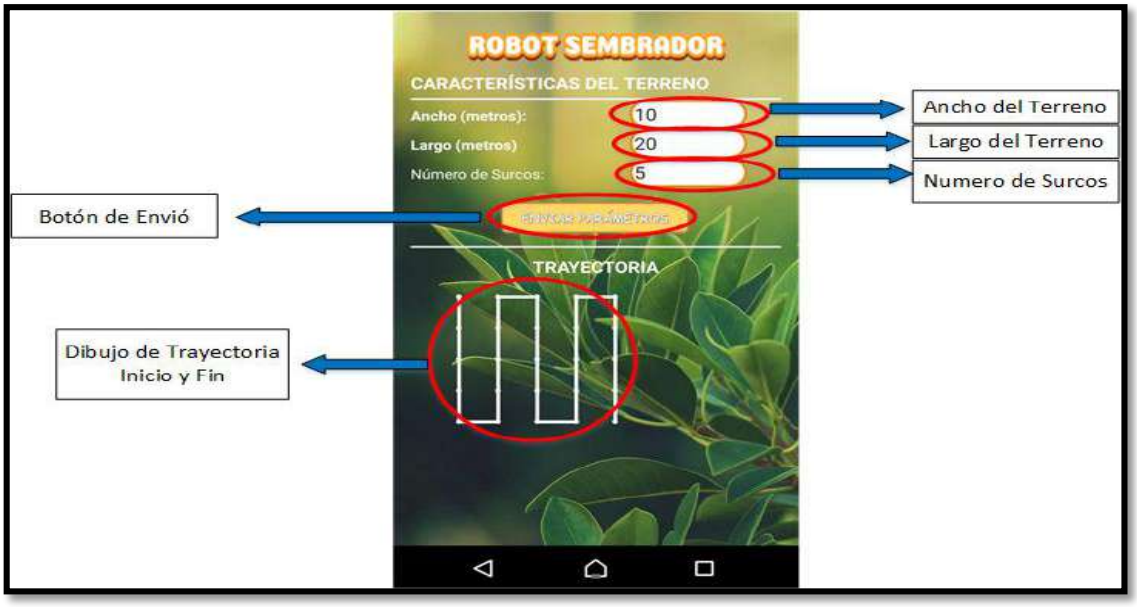


Figura 16-2: Pantalla de Configuración de Parámetros Iniciales.
Realizado por: Gonzalez, Henry & Carrillo, Maria**,** 2019

Para observar el tiempo de siembra, estado de la batería, nivel de la tolva y numero de tubérculos transportados se realizó una pantalla de visualización que se muestra en la Figura 17-2, cuenta con dos botones uno realiza el paro total del sistema por cualquier tipo de emergencia y por paro de procesos anteriormente predefinidos en la concepción, el otro botón permite el reinicio de las actividades hasta la finalización de dicho proceso de siembra.

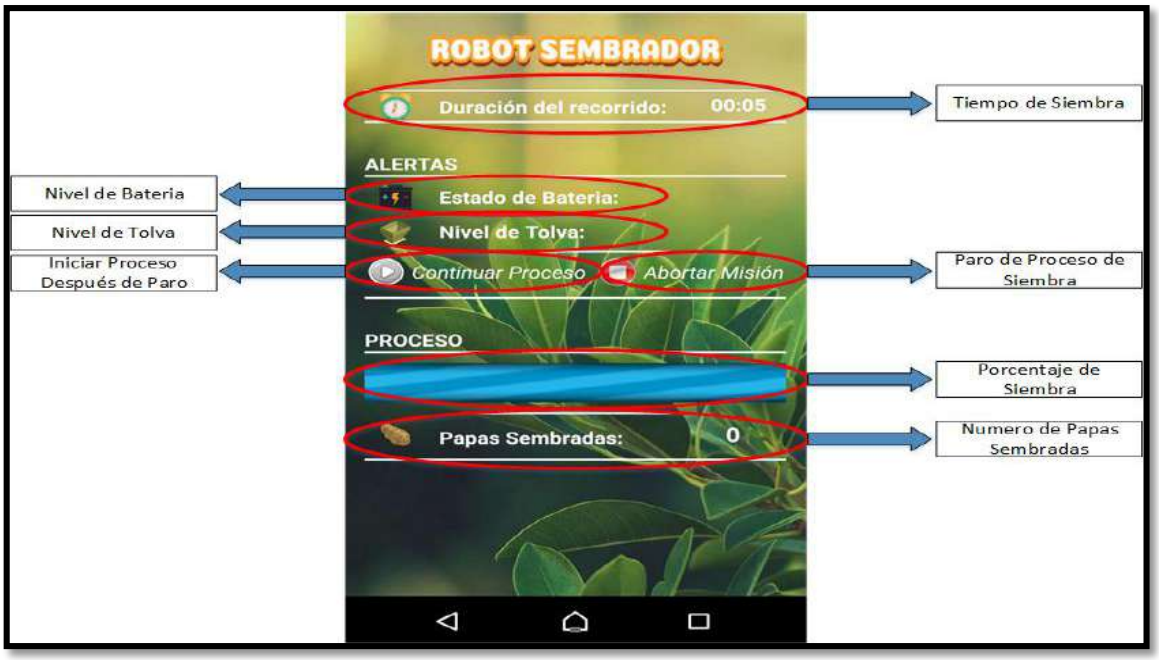


Figura 17-2: Pantalla de Seguimiento y Notificaciones.
Realizado por: Gonzalez, Henry & Carrillo, Maria**,** 2019

2.4 Construcción mecánica del prototipo AGSEM.

El *software* de diseño mecanico Solid Works v18 es la herramienta utilizada para la arquitectura mecánica de prototipo AGSEM. permite al diseñador de manera fácil y simultanea la modelación de piezas y estructuras en 2D y 3D para poder realizar su simulación de ensamblaje como se muestra en la Figura 18-2 (Dassault Systèmes SolidWorks Corporation, 175 Wyman Street, Waltham, 2015).

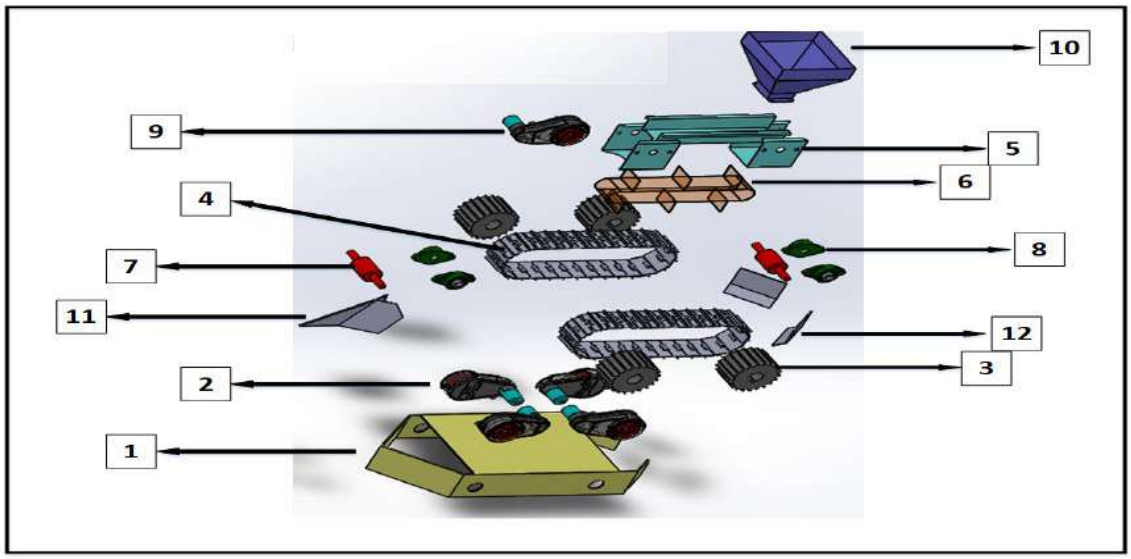


Figura 18-2: Desglose de Piezas en Diseño 3D.
Realizado por: Gonzalez, Henry & Carrillo, Maria**,** 2019

El orden de cada parte del prototipo AGSEM para poder realizar el armado mecánico se detalla en la Tabla 13-2 en orden descendente.

Tabla 13-2: Nombres de las Piezas Mecánicas del Prototipo AGSEM.

PIEZAS MECÁNICAS DEL PROTOTIPO AGSEM	
N° DE PIEZA	NOMBRE
1	Cuerpo del prototipo
2	Motores de tracción para cadena
3	Piñón para motores de tracción
4	Cadena con aletas de agarre
5	Protector de banda transportadora
6	Banda transportadora
7	Eslabón de banda transportadora
8	Tensores de banda transportadora
9	Motor de banda transportadora
10	Tolva de almacenamiento
11	Punta para surco
12	Tapa para surco

Fuente: Gonzalez, Henry & Carrillo, Maria**,** 2019
Realizado por: Gonzalez, Henry & Carrillo, Maria**,** 2019

La Tabla 14-2 muestra las dimensiones del prototipo implementado en largo, ancho y altura, así como el peso sin considerar la carga del tubérculo en la tolva, todas las dimensiones están en la unidad de medida de mm en el Anexo F se muestra los planos de la arquitectura mecánica del prototipo AGSEM.

Tabla 14-2: Dimensiones del Prototipo AGSEM.

DIMENSIONES DEL PROTOTIPO AGSEM	
Altura	533.75 mm
Ancho	492.00 mm
Largo	903.24 mm
Peso	30 Kg

Fuente: Gonzalez, Henry & Carrillo, Maria**,** 2019
Realizado por: Gonzalez, Henry & Carrillo, Maria**,** 2019

2.4.1 Sistema de Tracción.

En la Figura 29-2 a) se muestra el sistema de tracción en forma de oruga, en la parte superior de la Figura 19-2 b) se puede ver lo motores los cuales permiten el movimiento de la cadena, se colocaron 4 soportes móviles las cuales fueron diseñadas para resistir el peso de todo el prototipo, se colocaron aletas en la cadena tener un agarre en la tierra y no se desvié si presentan obstáculos como se ve en la Figura 19-2 d).

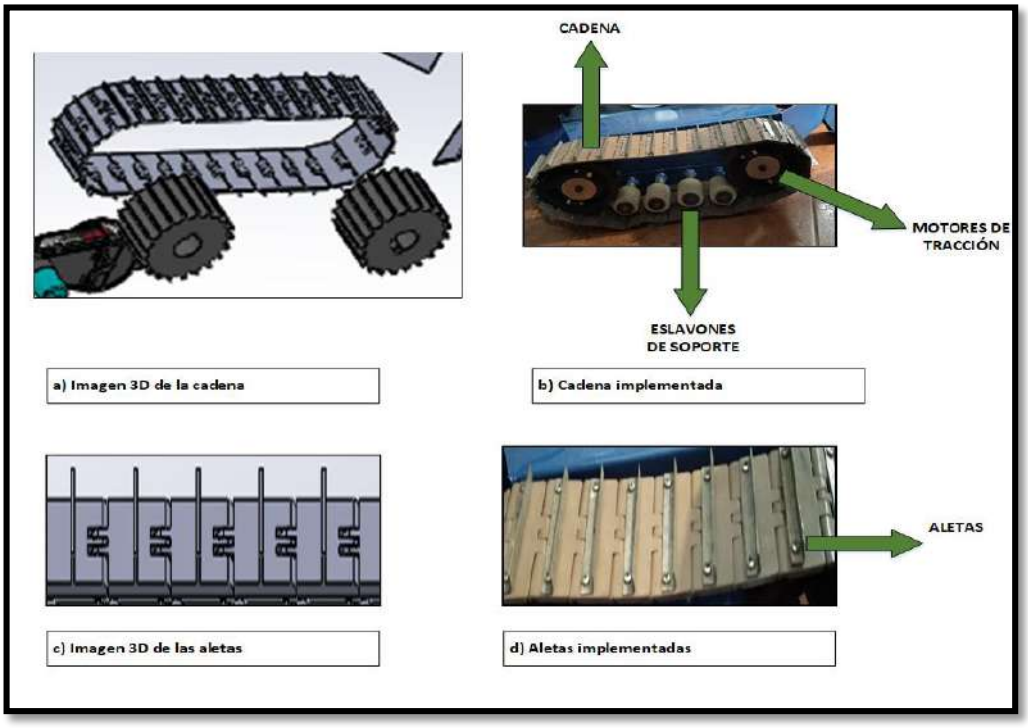


Figura 19-2: Diseño y Construcción de la Cadena del Prototipo.
Realizado por: Gonzalez, Henry & Carrillo, Maria**,** 2019

En la Tabla 15-2 muestra las dimensiones de la cadena en largo ancho, alto, altura de las aletas sobrepuestas y radio necesario de las llantas para la tracción en terreno arenoso sus cotas se observan en el Anexo F.

Tabla 15-2: Dimensión de la Cadena Tipo Oruga para el Prototipo AGSEM

DIMENSIONES DE LA ORUGA DEL PROTOTIPO AGSEM	
Largo	483.58 mm
Ancho	82.00 mm
Alto	183.75 mm
Radio de la llanta de tracción	50.00 mm
Altura de las aletas	20.00 mm

Fuente: Gonzalez, Henry & Carrillo, Maria**,** 2019
Realizado por: Gonzalez, Henry & Carrillo, Maria**,** 2019

2.4.2 Sistema de Almacenamiento.

En la Figura 20-2 muestra tiene la tarea de llevar 3 libras de papas que serán depositadas en la tolva tipo cónica con dimensiones en la boca superior de 20 x 20 cm y de boca inferior se tomó la dimensión de la semilla de papa certificada de 7 x 7 cm.

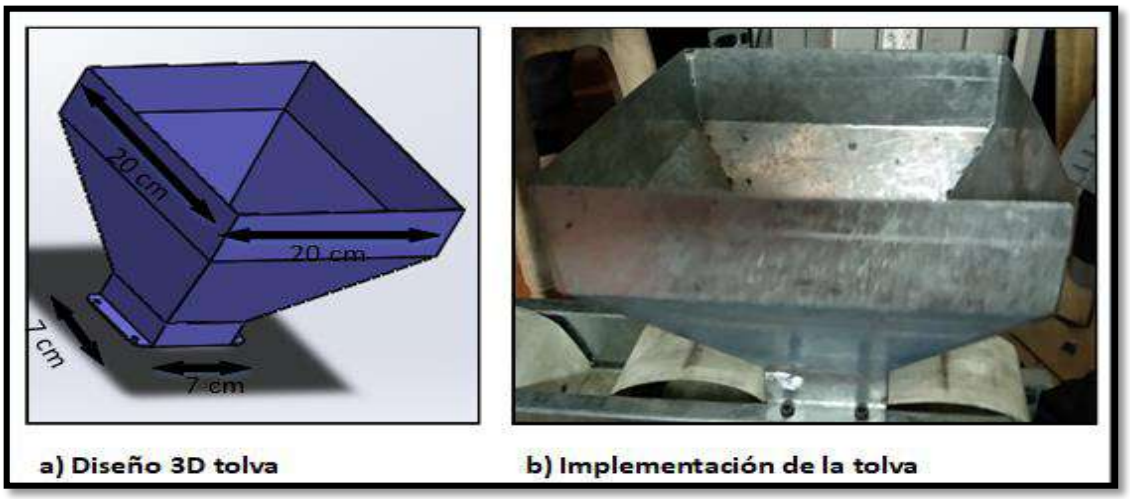


Figura 20-2: Diseño y Construcción de la Tolva de Almacenamiento.
Realizado por: Gonzalez, Henry & Carrillo, Maria**,** 2019

2.4.3 Sistema de Transporte

En la Figura 21-2 se muestra la banda transportadora cuenta con un sistema sobrepuesto, permitiendo la colocación del tubérculo a una medida de 15cm recogidas previamente de la boquilla inferior de la tolva de almacenaje.

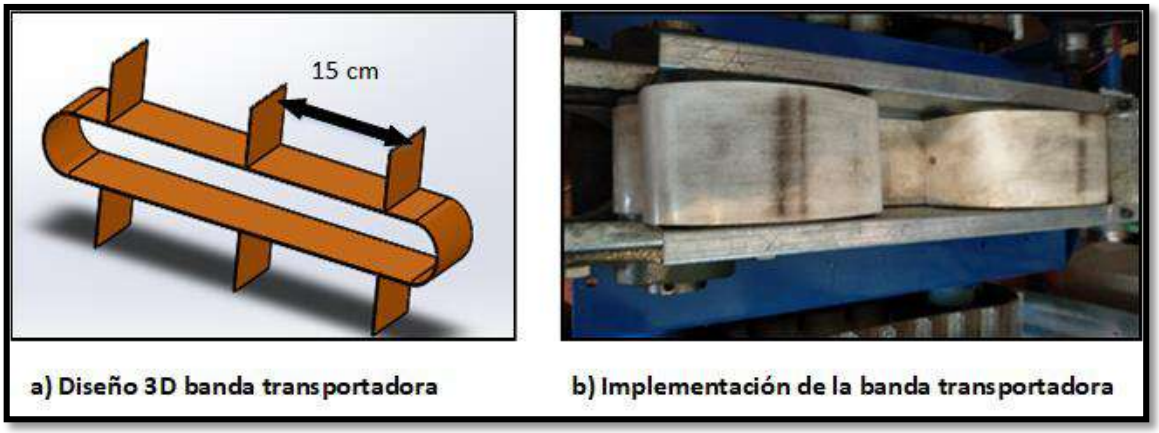


Figura 21-2: Diseño y Construcción de la Banda Transportadora.
Realizado por: Gonzalez, Henry & Carrillo, Maria**,** 2019

La Tabla 16-2 muestra las dimensiones de la banda transportadora para ser implementadas de manera real sus cotas están en el plano del Anexo F.

Tabla 16-2: Dimensiones de la Banda Transportadora Para el Prototipo AGSEM

DIMENSIONES DE LA BANDA TRANSPORTADORA	
Ancho	68.00 mm
Largo	350.00 mm
Alto	50.00 mm
Distancia entre ejes	15.00 mm

Fuente: Gonzalez, Henry & Carrillo, Maria**,** 2019
Realizado por: Gonzalez, Henry & Carrillo, Maria**,** 2019

En la Figura 22-2 muestra el diseño del guarda banda este sistema fue diseñado y construido con el propósito que el tubérculo no se desvié hacia los lados, se dirija hasta el final de la banda y a su punto de siembra.

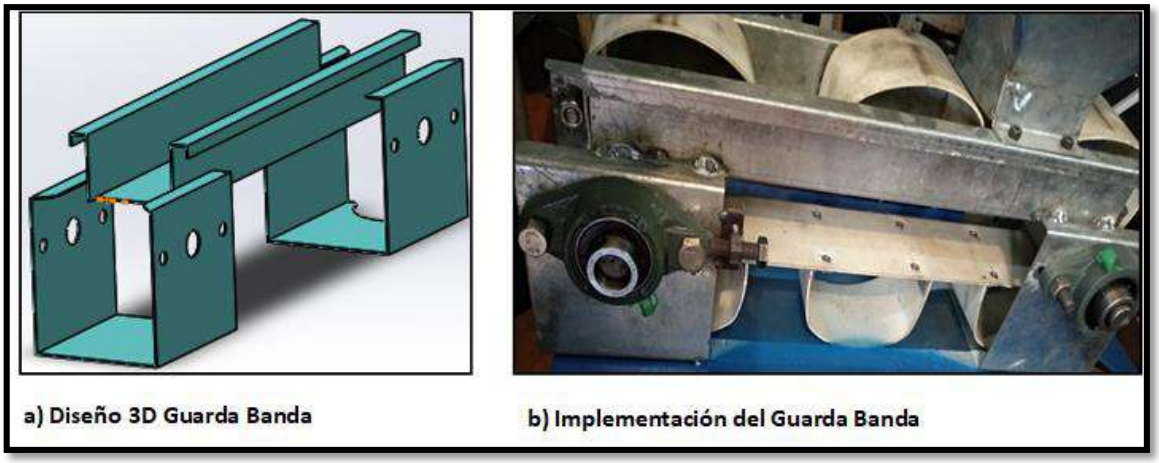


Figura 22-2: Diseño y Construcción del Guarda Banda.
Realizado por: Gonzalez, Henry & Carrillo, Maria**,** 2019

La Tabla 17-2 muestra las dimensiones de construcción del Guarda banda para el prototipo AGSEM sus cotas se encuentra en el Anexo F.

Tabla 17-2: Dimensiones del Guarda banda para el Prototipo AGSEM.

DIMENSIONES DEL GUARDA BANDA	
Alto	120.00 mm
Ancho	70.00 mm
Largo	350.00 mm

Fuente: Gonzalez, Henry & Carrillo, Maria**,** 2019
Realizado por: Gonzalez, Henry & Carrillo, Maria**,** 2019

En la Figura 23-2 se muestra el diseño e implementación de la punta encargada de abrir el surco para el implantar el tubérculo después de haber pasado por los dos procesos anteriores.

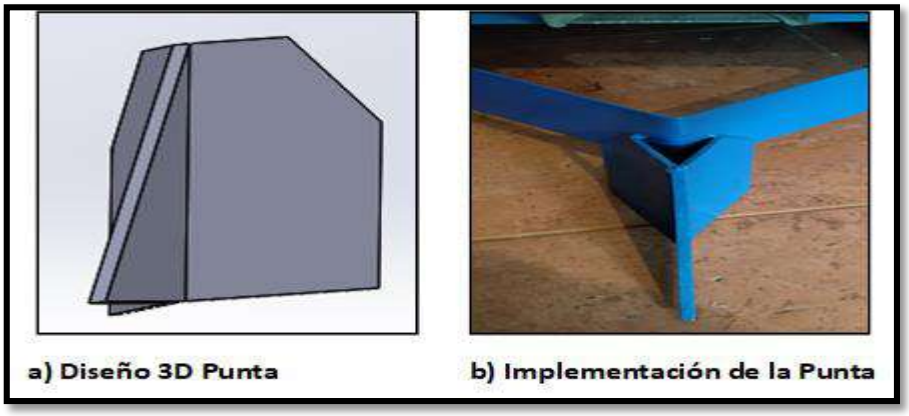


Figura 23-2: Diseño 3D y Construcción de la Punta.
Realizado por: Gonzalez, Henry & Carrillo, Maria**,** 2019

En la Tabla 18-2 muestra las dimensiones de construcción de la punta del prototipo su plano con sus dimensiones se encuentra en el Anexo F.

Tabla 18-2: Dimensiones de la Punta Para el Prototipo AGSEM.

DIMENSIONES DE LA PUNTA DEL PROTOTIPO AGSEM	
Alto	183.75 mm
Ancho	90.00 mm
Largo	190.25 mm

Fuente: Gonzalez, Henry & Carrillo, Maria**,** 2019
Realizado por: Gonzalez, Henry & Carrillo, Maria**,** 2019

CAPITULO III

3 RESULTADOS Y PRUEBAS DEL PROTOTIPO AGSEM.

En este capítulo se detalla las pruebas hardware y software para validar el funcionamiento del prototipo AGSEM, en este capítulo se muestra la calibración de los sensores, tiempo de conexión, tiempo de transmisión y actualización de datos en la aplicación móvil, autonomía del prototipo. Finalmente se realizó una tabla comparativa del costo del prototipo.

3.1 Ubicación del terreno para la realización de pruebas.

El terreno en el cual se realizaron las pruebas se ubica en el Cantón de San Andrés provincia de Chimborazo, en la Figura 1-3 a) se muestra la ubicación por medio de google maps. en la Figura 1-3 b) se muestra el robot y el terreno a ser sembrado.

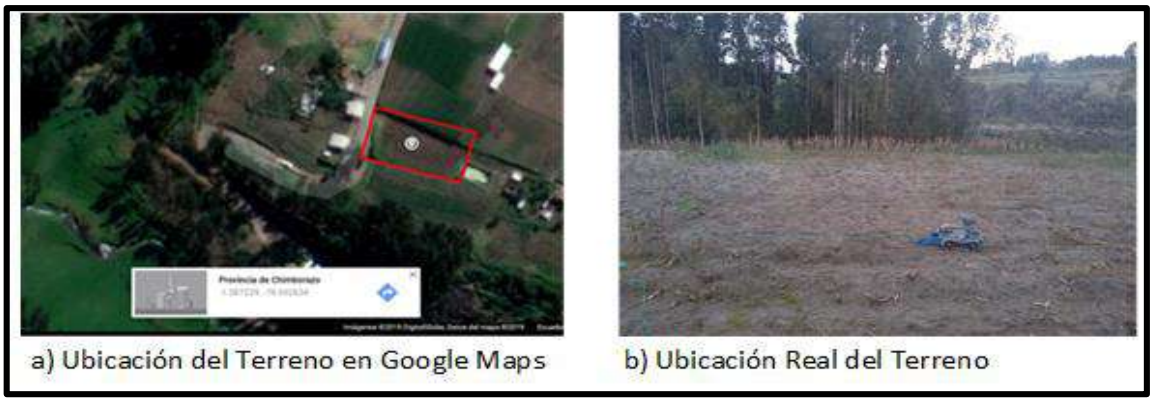


Figura 1-3: Ubicación del Terreno.
Realizado por: Gonzalez, Henry & Carrillo, Maria, 2019.

3.2 Pruebas de estabilidad del prototipo AGSEM.

Para verificar el funcionamiento del prototipo AGSEM se sometieron a diversas pruebas como, trayectoria, transporte y contador del tubérculo, para verificar que cumplan con los objetivos planteados en esta investigación.

3.2.1 Pruebas de estabilidad en Trayectoria.

Para verificar si el prototipo AGSEM sigue trayectoria de siembra se procedió a tomar dos puntos como, distancia recorrida y desvio de trayectoria aplicadas a un terreno de dimensiones de 5x5m.

- **Distancia recorrida.**

Para determinar la distancia recorrida se puso en funcionamiento el robot tomando como referencia desplazándose 15cm basándose con un flexómetro como observa en la Figura 2-3, se recolectaron datos a través de los sensores encoders, los cuales entregan una visualización en centímetro de la distancia que se desplaza el prototipo AGSEM.

Figura 2-3: Prueba de Distancia
Realizado por: Gonzalez, Henry & Carrillo, Maria, 2019.

En la Tabla 1-3 muestra un conjunto de datos recolectados del desplazamiento y el serial del arduino durante la prueba de distancia en centímetros recorridos.

Tabla 1-3: Prueba de Distancia.

N° Muestras	Desplazamiento Prototipo	Desplazamiento del Encoder.	Error
1	15 cm	15.02 cm	0.01
2	15 cm	15.25 cm	0.25
3	15 cm	15.30 cm	0.30
4	15 cm	15.30 cm	0.30
5	15 cm	15.25 cm	0.25
6	15 cm	15.35 cm	0.35

7	15 cm	15.85 cm	0.85
8	15 cm	15.56 cm	0.56
9	15 cm	15.65 cm	0.65
10	15 cm	15.25 cm	0.25
11	15 cm	15.85 cm	0.85
12	15 cm	15.36 cm	0.36
13	15 cm	15.45 cm	0.45
14	15 cm	15.85 cm	0.85
15	15 cm	16.00 cm	1
PROMEDIO		15.49 cm	0.49
DESVIACIÓN ESTÁNDAR		0.046	
COEFICIENTE DE VARIACIÓN		0.297%	

Fuente: Gonzalez, Henry & Carrillo, Maria, 2019
Realizado por: Gonzalez, Henry & Carrillo, Maria, 2019

Mediante los resultados obtenidos se observa que el coeficiente de variación es de 0.297% medida aceptada entre la norma iso para pruebas de laboratorio (NTCISO/IEC 17025) que entrega un máximo de 10%.

- **Desvió de la trayectoria.**

Para determinar si existe un desvió de la trayectoria se puso a prueba el prototipo durante una línea recta si existiera un desvió el sensor MPU procederá a corregir su trayectoria, En la Figura 3-3 se visualiza al prototipo saliendo de la trayectoria predefinida en la aplicación móvil esperando actuar el giroscopio para volver a su trayectoria.

Figura 3-3: Prueba de Desplazamiento.
Realizado por: Gonzalez, Henry & Carrillo, Maria, 2019.

Para demostrar si el robot cumple con el seguimiento en línea recta según la trayectoria predefinida se puso a prueba el robot en el terreno previamente preparado para el análisis se tomó una secuencia de muestras para comparar si cumple dicho proceso mostrada en la Tabla 2-3.

Tabla 2-3: Prueba de Desviacion.

N° Muestras	Distancia	Desvió de la trayectoria	Activación del Giroscopio	Error
1	1 m	0.00 cm	No	0.00
2	1 m	1.00 cm	Si	1.00
3	1 m	0.50 cm	No	0.50
4	1 m	0.50 cm	No	0.50
5	1 m	0.50 cm	No	0.50
6	1 m	0.50 cm	No	0.50
7	1 m	1.50 cm	Si	1.50
8	1 m	0.75 cm	No	0.75
9	1 m	0.75 cm	No	0.75
10	1 m	0.75 cm	No	0.75
11	1 m	0.75 cm	No	0.75
12	1 m	2.00 cm	Si	2.00
13	1 m	1.00 cm	No	1.00
14	1 m	1.00 cm	No	1.00
15	1 m	1.00 cm	No	1.00
PROMEDIO		0.83 cm		0.83
DESVIACION ESTANDAR		0.011		
CV%		0.14%		

Fuente: Gonzalez, Henry & Carrillo, Maria, 2019
Realizado por: Gonzalez, Henry & Carrillo, Maria, 2019

Mediante los resultados de la Tabla 2-3 se observa que el coeficiente de variación es de 0.14% medida aceptada entre la norma iso para pruebas de laboratorio (NTCISO/IEC 17025) que entrega un máximo de 10%.

3.2.2 Pruebas de estabilidad en Transporte.

Esta Prueba se realizó según la dimensión de la tolva utilizando diferentes tamaños de tubérculos los cuales fueron sometidos a pruebas de numero de semillas transportadas y atascamiento en la tolva.

- **SEMILLAS DE 1cm A 4 cm.**

En la Figura 4-3 se visualiza la tolva del prototipo con la semilla de dimensiones de de 1cm a 4cm para realizar las pruebas de transporte y atascamiento.

Figura 4-3: Semillas de 1cm a 4cm en la tolva.
Realizado por: Gonzalez, Henry & Carrillo, Maria, 2019.

En la Tabla 3-3 se muestra los datos obtenidos del transporte de los tubérculos de medidas de 1cm a 4cm enviados desde su tolva hacia su punto de destino en un tiempo determinado.

Tabla 3-3: Transporte de Tubérculos con Semillas de 1cm a 4cm.

N° Muestras	Tiempo	N° Tubérculos Transportados	Atascamiento
1	1 min	1	No
2	2 min	2	No
3	3 min	3	No
4	4 min	2	No
5	5 min	2	No
6	6 min	2	No
7	7 min	3	No
8	8 min	1	No
9	9 min	2	No
10	10 min	2	No
11	11 min	0	Si
12	12 min	0	Si
13	13 min	0	Si
14	14 min	0	Si
15	15 min	0	Si
TOTAL		20	

Fuente: Gonzalez, Henry & Carrillo, Maria, 2019
Realizado por: Gonzalez, Henry & Carrillo, Maria, 2019

Las semillas de 1cm a 4cm no cumplen con las especificaciones del prototipo teniendo la capacidad de acumularse en la tolva y parar el sistema, siendo este tipo de tubérculos no recomendables para el prototipo AGSEM.

- **SEMILLAS DE 5 A 7 CM.**

Para la semilla de medidas de 5cm a 7cm depositada previamente en la tolva del prototipo es visualizada en la Figura 5-3.

Figura 5-3: Semillas de 5cm a 7cm en la tolva.
Realizado por: Gonzalez, Henry & Carrillo, Maria, 2019.

En la Tabla 4-3 se muestra los datos obtenidos del transporte de los tubérculos de medidas de 5cm a 7cm siendo la máxima abertura de la tolva de 7cm enviados desde su tolva hacia su punto de destino en un tiempo determinado.

Tabla 4-3: Transporte de Tubérculos con Semillas de 5cm a 7cm.

N° Muestras	Tiempo	N° Tubérculos Transportados	Atascamiento
1	1 min	1	No
2	2 min	1	No
3	3 min	1	No
4	4 min	1	No
5	5 min	2	No
6	6 min	1	No
7	7 min	1	No
8	8 min	1	No
9	9 min	2	No
10	10 min	1	No
11	11 min	1	No
12	12 min	1	No
13	13 min	2	No
14	14 min	1	No
15	15 min	1	No
TOTAL		18	

Fuente: Gonzalez, Henry & Carrillo, Maria, 2019
Realizado por: Gonzalez, Henry & Carrillo, Maria, 2019

Las semillas de 5cm a 7cm cumplen con las dimensiones máximas de la boca inferior de la tolva cumpliendo con las especificaciones del prototipo teniendo la capacidad de distribuirse en la tolva y continuar su proceso con normalidad, siendo este tipo de tubérculos recomendables para el prototipo AGSEM.

3.2.3 Pruebas de estabilidad en Contador.

Estas pruebas se realizan a través del sensor infrarrojo reflectivo colocado en la punta del prototipo, para el conteo del tubérculo se colocaron 15 semillas de papa para saber si el conteo es real o manifiesta algún error, en la Tabla 5-3 se observa los datos obtenidos por un numero de muestras.

Tabla 5-3: Contador de Tubérculos.

N° Muestras	N° Tubérculos
1	1
2	1
3	1
4	0
5	1
6	1
7	1
8	1
9	0
10	1
11	1
12	1
13	0
14	1
15	1
TOTAL	12
ERROR	20%

Fuente: Gonzalez, Henry & Carrillo, Maria, 2019
Realizado por: Gonzalez, Henry & Carrillo, Maria, 2019

Para buscar el error en el sensor infrarrojo reflectivo se colocaron 15 semillas de papa para ser transportador por la banda transportadora hacia su destino, donde se ubicó un error de 20% según la RAS el prototipo cumple con los parámetros de una agricultura sostenible.

3.3 Pruebas de comunicación del prototipo AGSEM.

Al realizar las pruebas de la aplicación móvil se tomaron en cuenta dos parámetros principales el envío y recepción de datos, los cuales permiten al usuario manipular los parámetros iniciales, observar la cantidad de tubérculos sembrados y las advertencias predefinidas en el Capítulo II.

Figura 6-3: Pantalla de Configuración de la Aplicación Móvil.
Realizado por: Gonzalez, Henry & Carrillo, Maria,2019

3.3.1 Tiempo de actualización de datos.

El tiempo de actualización trata sobre el tiempo en que el microcontrolador demora en la transmisión de datos, se realizó a base de la toma de 10 muestras mostradas en la Tabla 6-3 para su observación y control para el usuario con la aplicación móvil con la máquina.

Tabla 6-3: Tiempo de Actualización de Datos.

Tiempo de Actualización de Datos		
N° MUESTRA	**DISTANCIA**	**TIEMPO**
1	0 m	1.17 s
2	1 m	1.34 s
3	1.5 m	1.02 s
4	2 m	1.58 s
5	2.5 m	1.96 s
6	3 m	1.64 s
7	3.5 m	1.41 s
8	4 m	1.82 s

9	4.5 m	1.91 s
10	5 m	1.35 s
Promedio		1.52 s
Desviación Estándar		0.064
Coeficiente de Variación		0.042

Fuente: Gonzalez, Henry & Carrillo, Maria**,** 2019
Realizado por: Gonzalez, Henry & Carrillo, Maria**,** 2019

Según los resultados obtenidos en el muestreo para la prueba de software en la actualización, muestra el tiempo del prototipo AGSEM, dando como resultado si el robot se encuentra dentro del rango de conexión del microcontrolador NodeMCU, el prototipo enviara los datos con un tiempo promedio de 1.52s.

3.3.2 Tiempo de respuesta a parámetros iniciales.

El tiempo de respuesta, es la demora del prototipo AGSEM de reaccionar a los parámetros iniciales como largo, ancho, numero de surcos y definición de la trayectoria, se realizó un muestreo a base de 10 datos mostrada en la Tabla 7-3 para probar la conexión del usuario en la aplicación móvil con el prototipo AGSEM.

Tabla 7-3: Tiempo de Respuesta de la Aplicación Móvil.

	Tiempo de Respuesta del Prototipo	
N° MUESTRAS	**DISTANCIA**	**TIEMPO**
1	0 m	1.71 s
2	1 m	1.08 s
3	1.5 m	1.34 s
4	2 m	1.14 s
5	2.5 m	1.54 s
6	3 m	1.48 s
7	3.5 m	1.37 s
8	4 m	1.15 s
9	4.5 m	1.48 s
10	5 m	1.81 s
Promedio		1.41 s
Desviación Estándar		0.038
Coeficiente de Variación		0.027

Fuente: Gonzalez, Henry & Carrillo, Maria
Realizado por: Gonzalez, Henry & Carrillo, Maria

Por medio del muestreo de datos, se determinó que el prototipo reacciona a los parámetros iniciales con un promedio de tiempo de 1.41s siempre y cuando el robot se encuentre en el rango operativo del microcontrolador el cual tiene una distancia máxima de conexión de 10m.

3.4 Pruebas de siembra de funcionamiento del prototipo AGSEM.

Para las pruebas de siembra se tomaron dos parámetros como el movimiento del prototipo en terrenos con diferentes dimensiones y la siembra artesanal vs la siembra automatizada.

3.4.1 Prueba del prototipo en Terrenos con Diferentes Dimensiones.

Para terrenos con dimensiones menores al rango de transmisión de la NodeMCU, el cual es de 10m máximo mostrado en la Figura 7-3 el prototipo no tuvo problemas de transmisión y de envió de información con el usuario hacia la aplicación móvil realizada. El prototipo AGSEM efectuó un proceso de sembrado manteniéndose en las normas técnicas detalladas en el Capítulo I.

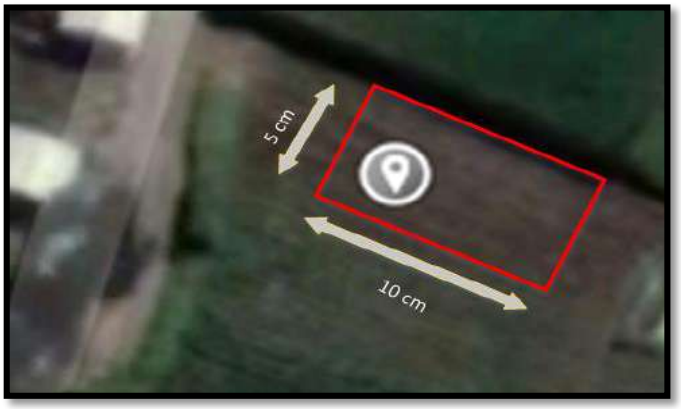


Figura 7-3: Terreno de dimensión menores a 10m.
Realizado por: Gonzalez, Henry & Carrillo, Maria, 2019

Para terrenos mayores a 10m como muestra la Figura 8-3 se procede a dividir la superficie en 2 partes iguales siguiendo el rango de transmisión de la NodeMCU y el prototipo AGSEM cumpla con el objetivo de siembra establecido por las normas técnicas. Tendiendo así dos puntos de inicio y dos puntos de finalización.

Figura 8-3: Terreno de dimensión mayores a 10m.
Realizado por: Gonzalez, Henry & Carrillo, Maria, 2019

Para terrenos con dimensiones irregulares como muestra la Figura 9-3 se presentaron problemas, siendo el robot muy robusto y no poder realizar la curva adecuada para el sembrado de la siguiente hilera, al terreno se realizaron las divisiones posibles tratando de abarcar la mayor parte de su superficie, de manera rectangular o cuadrática, para que el prototipo AGSEM cumpla con el proceso de siembra establecido, sin ningún tipo de obstrucción en la trayectoria dibujada por el usuario en la aplicación móvil.

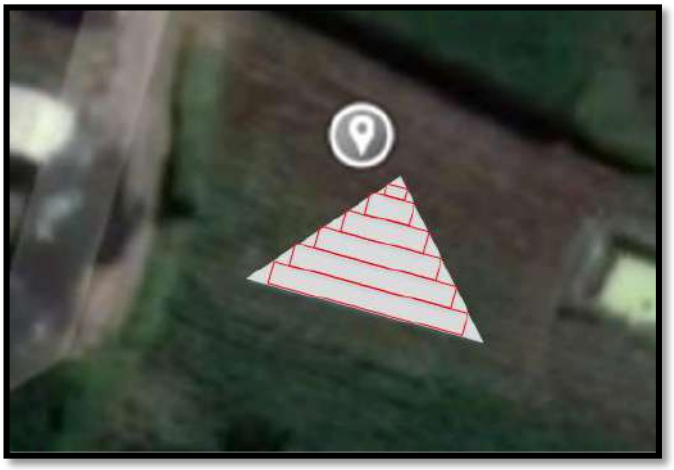


Figura 9-3: Terrenos con Dimensiones Irregulares.
Realizado por: Gonzalez, Henry & Carrillo, Maria**,** 2019

3.4.2 Siembra Artesanal vs Siembra Automatizada.

Para las pruebas en la comparación de sembrado se tomaron diferentes muestras con respecto a técnicas agrónomas definidas en el Capítulo I entre ellas están, tiempo, profundidad, distancia, cantidad del tubérculo sembrado.

Figura 10-3: Distancia y Profundidad del Prototipo AGSEM.
Realizado por: Gonzalez, Henry & Carrillo, Maria. 2019

En la Tabla 8-3 se muestra los datos obtenidos mediante 10 muestras en el proceso de siembra artesanal en un terreno con dimensiones de 5x5m en la parroquia de San Andres.

Tabla 8-3: Siembra Artesanal.

SIEMBRA ARTESANAL				
N° Muestra	**Tiempo**	**N° Tubérculos**	**Profundidad**	**Distancia**
1	01 min	1	15.40 cm	29.40 cm
2	05 min	5	9.10 cm	31.10 cm
3	10 min	3	11.64 cm	19.64 cm
4	15 min	3	12.97 cm	25.97 cm
5	20 min	5	10.93 cm	20.93 cm
6	25 min	4	15.71 cm	22.71 cm
7	30 min	4	9.31 cm	21.31 cm
8	35 min	2	13.28 cm	25.28 cm
9	40 min	3	14.23 cm	31.23 cm
10	45 min	5	11.43 cm	20.43 cm
	PROMEDIO	35	12.4 cm	24.8 cm

Fuente: Gonzalez, Henry & Carrillo, Maria 2019
Realizado por: Gonzalez, Henry & Carrillo, Maria, 2019

Para la siembra automatizada se tomaron un total de 10 muestras mostradas en la Tabla 9-3 obtenidas de un terreno de dimensiones de 5x5m.

Tabla 9-3: Siembra Automatizada.

SIEMBRA AUTOMATIZADA				
N° Muestra	**Tiempo**	**N° Tubérculos**	**Profundidad**	**Distancia**
1	01 min	5	9.24 cm	19.24 cm
2	05 min	3	8.79 cm	17.79 cm
3	10 min	4	9.28 cm	17.28 cm
4	15 min	4	8.24 cm	20.24 cm

5	20 min	3	6.59 cm	15.59 cm
6	25 min	3	5.52 cm	20.52 cm
7	30 min	3	9.94 cm	16.94 cm
8	35 min	5	9.92 cm	15.92 cm
9	40 min	5	8.66 cm	16.66 cm
10	45 min	4	7.97 cm	19.97 cm
	PROMEDIO	39	8.42 cm	18.02 cm

Fuente: Gonzalez, Henry & Carrillo, Maria 2019
Realizado por: Gonzalez, Henry & Carrillo, Maria, 2019

Se puede observar en la comparación entre los datos obtenidos de siembra artesanal en la Tabla 8-3 y siembra automatizada en la Tabla 9-3, en medidas de profundidad el prototipo AGSEM realizo su tarea con al colocarse con un promedio de 8.42cm, encontrado dentro del rango técnico agrónomo el cual es de 5cm a 10cm, es así el promedio de sembrado artesanal de 12.4cm sobrepasa el rango para el cual es aceptada la siembra. Haciendo al prototipo más tangible en la prueba de profundidad.

En la prueba de distancia el rango aceptado se encuentra de 15cm a 25cm para poder cumplir con los requerimientos de siembra, el prototipo AGSEM obtuvo un promedio de 18.02cm entre tubérculo sembrado, artesanalmente se obtuvo un promedio de 24.8cm casi por llegar al límite del rango requerido pero aceptable. El prototipo AGSEM mantuvo un conjunto de muestras casi similares y artesanalmente sus muestras fueron muy variables, haciendo al prototipo más eficaz en el proceso de siembra.

El número de tubérculos sembrados de manera automatizada es mayor a los sembrados de manera artesanal poniendo como resultado 4 tubérculos más, mostrando la eficiencia del prototipo en productividad es de 10.25% mayor.

3.5 Consumo de energía en el prototipo AGSEM.

Para el consumo energético se tomaron en cuenta cada dispositivo electrónico que tenga un consumo de corriente mostrados en el Tabla 10-3, durante el proceso de siembra por el prototipo AGSEM.

Tabla 10-3: Consumo Energético del Prototipo AGSEM.

CONSUMO DE ENERGÍA EN EL PROTOTIPO AGSEM					
Cantidad	**Elemento**	**Corriente c/e**	**Corriente total**	**Voltaje**	**Potencia**

2	Sensores infrarrojos	0.05 A	0.10 A	5 V	0.50 W
1	Puente H monster	0.10 A	0.20 A	5 V	1.00 W
2	Puente H IBT-2	0.05 A	0.10 A	5 V	0.50 W
1	NodeMCU	0.45 A	0.45 A	5 V	2.25 W
1	Arduino UNO	0.20 A	0.20 A	5 V	1.00 W
2	Encoder	0.05 A	0.10 A	5 V	0.50 W
4	Motor de tracción	2.02 A	8.08 A	12 V	96.96 W
1	Motor de banda	0.90 A	0.90 A	12 V	10.80 W
1	Convertidor DC-DC	1.15 A	1.15 A	12 V	13.80 W
TOTAL		4.07 A	10.13 A		

Fuente: Gonzalez, Henry & Carrillo, Maria, 2019
Realizado por: Gonzalez, Henry & Carrillo, Maria**,** 2019

Se tomó en cuanta el consumo de corriente de cada uno de los dispositivos conectados en el prototipo AGSEM para poder saber su tiempo de autonomía, teniendo una tasa de descarga por la batería de 30/40 c con una corriente de 5000mhA, mediante la fórmula a continuación se pudo identificar su tiempo de autonomía para la siembra de 0.49 h equivalente a 49 min.

$$Tiempo = \frac{Corriente\ de\ la\ Bateria}{Corriente\ total}$$

$$Tiempo = \frac{5\ A/h}{10.13\ A}$$

$$Tiempo = 0.49\ h$$

$$Tiempo = 49\ min$$

3.6 Análisis económico del prototipo AGSEM.

Para el análisis económico se colocaron elementos electrónicos y mecánicos utilizados en el ensamblaje del prototipo AGSEM detallados en la Tabla 11-3.

Tabla 11-3: Análisis Económico del Prototipo AGSEM.

ANÁLISIS ECONÓMICO DEL PROTOTIPO				
ELEMENTOS ELECTRÓNICOS	**Cantidad**	**Elemento**	**Precio c/u**	**Precio total**
	2	Sensores infrarrojos	10	20
	1	Puente H monster	25	25
	2	Puente H IBT-2	40	80
	1	NodeMCU	40	40
	1	Arduino UNO	25	25
	2	Encoder	10	10
	4	Motor de tracción	40	40
	1	Motor de banda	10	10

	1	Convertidor DC-DC	15.50	15.50
	1m	Cable N° 12	1.60	1.60
	1	Corte laser	15	15
ELEMENTOS MECÁNICOS	**1m**	Cadena	75	75
	8	Rodamientos	10	80
	1m	Banda Tejida	25	25
	1	Lamina de laton	90	90
	1	Lamina de Acera 4mm	75	75
	1	Barilla	10	10
	2	Tensores	25	50
	1m	Tubo de nailon	48	48
		Piezas en torno	50	50
		Suelda	25	25
		Ensamblaje	200	200
TOTAL				1010.1

Fuente: Gonzalez, Henry & Carrillo, Maria**,** 2019
Realizado por: Gonzalez, Henry & Carrillo, Maria**,** 2019

Según las muestras la parte mecánica tiene un precio de 728$ dólares (USD) y la parte electrónica un precio de 282.1$ dólares (USD) dando un costo total de 1010.1$ (USD), todos los elementos electrónicos montados en el prototipo AGSEM pueden ser removidos.

Al ser AGSEM el primer prototipo sembrador de papas en el Ecuador posee una gran precisión en el proceso de siembra, ofreciendo un amplio campo de investigación en prototipos robóticos sembradores.

El precio total del prototipo de 1010.1$ (USD), mientras el alquiler de maquinaria y mano de obra asciende a los 1500$ (USD), dando una diferencia de 32.66% equivalente a 489.9$ (USD), siendo el prototipo AGSEM de fácil compra para el agricultor.

3.7 Prueba de aceptación del prototipo AGSEM.

Durante el proceso de Siembra se realizó una consulta a los productores de papas cercanos a la zona de sembrado ubicada en San Andres, tomando diferentes puntos mostrados en la Tabla 12-3 sobre el prototipo para saber si cumple con las expectativas y condiciones.

Tabla 12-3: Parámetros de Aceptación de Prototipo AGSEM.

AGRICULTORES PARAMETROS	AG1	AG2	AG3	AG4	AG5
Precio	✓		✓	✓	
Eficiencia	✓	✓	✓	✓	✓
Aplicación Movil	✓		✓		
Capacidad	✓	✓		✓	✓
Tiempo de Siembra	✓	✓	✓	✓	✓
Destreza	✓	✓		✓	✓

Fuente: Gonzalez, Henry & Carrillo, Maria
Realizado por: Gonzalez, Henry & Carrillo, Maria

Según los resultados obtenidos el prototipo cumple con el proceso de siembra con mayor garantía que el proceso artesanal en un menor tiempo, aumentando la producción en un 10.25%, gracias a la encuesta realizada la mayor parte de agricultores de la zona desearían adquirir el prototipo para la siembra en sus terrenos.

Figura 11-3: Capacitación sobre el manejo del Prototipo AGSEM.
Realizado por: Gonzalez, Henry & Carrillo, Maria,2019

CONCLUSIONES.

- Se implementó un prototipo de robot sembrador de papa en terrenos sin inclinación para pequeños productores cuyo funcionamiento está basado en odometria permitiendo al robot dirigirse al punto de finalización siguiendo una trayectoria predefinida con un 95% de confianza.

- El prototipo cuenta con un sistema de envío y recepción de datos, basado en dispositivos móviles con un sistema operativo Android para visualización en tiempo real, dando como resultado un tiempo máximo de demora de 1.51s a 1.42s. Dando como resultado una comunicación sin interrupciones.

- La implementación del prototipo entrego en las pruebas de estabilidad según su desvió y distancia un error 0,14% y 0,29% respectivamente por debajo del margen máximo de 10% según normas (NTCISO/IEC 17025) dando como resultado el sistema estable

- Para los parámetros técnicas de siembra se tomaron en consideración dos variables distancia y profundidad, en el cual el prototipo AGSEM cumplió de manera satisfactoria con los parámetros establecidos con un 33,33% sobre la siembra artesanal.

- Según la RAS la perdida de sembrado debe equivaler al 20% dando como resultado en la prueba de conteo una pérdida de 20% según la norma el prototipo es estable.

- De acuerdo a las dimensiones del terreno se determinó, que el prototipo tiene una mejor eficiencia si está dentro del rango de transmisión de la NodeMCU el cual es una distancia máxima de 10m.

RECOMENDACIONES.

- Realizar un estudio sobre baterías y su consumo energético en el prototipo AGSEM, para tener un mayor tiempo de autonomía en el proceso de siembra.

- Desarrollar un estudio de maquinarias para los procesos agrícolas como, fumigación, recolección, siembra y detección de malezas.

- Realizar un estudio sobre la implementación de un prototipo hibrido en maquinaria pesada para el proceso autónomo de siembra.

- Realizar una investigación sobre el funcionamiento del prototipo en diferentes tipos de suelos.

BIBLIOGRAFÍA.

Armada, E. G. 'ROBOTS'. [En linea] 2015, [Citado el mayo del 2019] , [Disponible en]: https://ebookcentral.proquest.com/lib/espochsp/detail.action?docID=3429422&fbclid=IwAR0IU60uwbwIjBje2KHOJQ_th4FRzTsVU8bL64wYn3cXj5maawztd4gIIzw

Albornoz Guillermo, I. 'PAPA : Cuánta Semil a por Hectárea [NIAP'. [En linea] 1968, [Citado el mayo del 2019] , [Disponible en]: http://repositorio.iniap.gob.ec/handle/41000/162.

Arduino. ' Arduino ' [En linea] 2017, [Citado el mayo del 2019] [Disponible en]: http://arduino.cl.

ARDUINO 'Arduino'. [En linea] 2019, [Citado el mayo del 2019] [Disponible en]: https://www.arduino.cc/en/Guide/Introduction.

Bugallo, D. A. M., Cortés., C. A. P. and Jaimes, C. I. R. 'Solar System for the Operation of a Robot Agrícola', *Revista Colombiana de Tecnologías de Avanzada*, 1(27), pp. 33–39. [En linea] 2016, [Citado el mayo del 2019] [Disponible en]: http://www.unipamplona.edu.co/unipamplona/portalIG/home_40/recursos/05_v25_30/revista_27/16052016/06.pdf.

CORDIS(Community Research and Development Information Service) 'CROPS — Resultado resumido', pp. 1–2. [En linea] 2016, [Citado el mayo del 2019] [Disponible en]: https://cordis.europa.eu/project/rcn/96216/brief/es.

Christesen Bill *HortiBot - Autonomous Weed 'n Feed AgBot: Science Fiction in the News*, *HortiBot - Autonomous Weed 'n Feed AgBot*. [En linea] 2007, [Citado el mayo del 2019] [Disponible en]: http://www.technovelgy.com/ct/Science-Fiction-News.asp?NewsNum=1109 [Citado el]: 15 November 2018).

Dassault Systèmes SolidWorks Corporation, 175 Wyman Street, Waltham, M. 'Introducción a Solidworks', *Solidworks*, pp. 12-4-6–18. doi: 10.1016/0031-9384(94)90368-9. [En linea] 2015, [Citado el mayo del 2019] , [Disponible en]: https://my.solidworks.com/solidworks/guide/SOLIDWORKS_Introduction_ES.pdf

Díaz, F. R. 'La oportunidad de utilizar robots en agricultura'. [En linea] 2018, [Citado el mayo del 2019],[disponible en]: http://agriculturers.com/los-robots-un-nuevo-paradigma-en-la-agricultura/

EINSTRONIC 'INTRODUCTION to NodeMCU ESP8266', *Einstronic*, (July), p. 4. [En linea] 2017, [Citado el mayo del 2019] , [Disponible en]: www.einstronic.com.

FAO (Organización de las Naciones Unidas para la Alimentacion y la Agricultura) 'Enfoques / 2006 Tesoro enterrado : la papa Papa 2008'. [En linea] 2006, [Citado el mayo del 2019], [Disponible en]: http://www.fao.org/ag/esp/revista/0611sp1.htm.

FAO (Organización de las Naciones Unidas para la Alimentacion y la Agricultura) 'TESORO ENTERRADO_ EL CULTIVO'. [En linea] 2008, [Citado el mayo del 2019], [Disponible en]: http://www.fao.org/potato-2008/es/lapapa/cultivo.html.

Fernandez, C "Programación de aplicaciones Android para aprendizaje de Idiomas", [En linea] 2018, [Citado el mayo del 2019], [Disponible en]: http://uvadoc.uva.es/bitstream/handle/10324/31297/TFG-P-843.pdf;jsessionid=37B995B7818236986CA36E26DBA3A7EA?sequence=1

FIALLO JOSE IGNACIO. ' Importancia del Sector Agricola en una Economia Dolarizada**',** [En linea] 2017, [Citado el mayo del 2019] [Disponible en]: http://repositorio.usfq.edu.ec/bitstream/23000/6807/1/134856.pdf.

Físicas, F. D. E. C. *Navegación autónoma de robots en agricultura: un modelo de agentes*. [En linea] 2005, [Citado el mayo del 2019], [Disponible en]: http://webs.ucm.es/BUCM/tesis//fis/ucm-t27198.pdf

Garcia Moreno Emilio *Full-Text, Automatizacion De Procesos*. [En linea] 1999, [Citado el mayo del 2019], [Disponible en]: http://webs.ucm.es/BUCM/tesis//fis/ucm-t27198.pdf

Guerrero, M. 'PANORAMA AGROECONOMICO DEL ECUADOR', *QUITO ECUADOR*. [En linea] 2015, [Citado el mayo del 2019], [Disponible en]: http://fliphtml5.com/ijia/psqo/basic.

Guglielmetti H., P. G. 'Siembra de papas LA DE PAPA DE GUARDA O COSECHA, DESDE FINES DE SEPTIEMBRE A FINES DE OCTUBRE', *IPA La Platina Nº*, 21, pp. 12–15. [En linea] 1984, [Citado el mayo del 2019] [Disponible en]: http://biblioteca.inia.cl/medios/biblioteca/IPA/NR00529.pdf.

INIAP() 'Papa(Solanun Tuberosum)'. [En linea] 2014, [Citado el mayo del 2019] [Disponible en]: http://tecnologia.iniap.gob.ec/index.php/explore-2/mraiz/rpapa.

Martinez Valle Luciano. ' La Agricultura Familiar en el Ecuador Informe del Proyecto Análisis de la Pobreza y de la Desigualdad en América Latina Rural ', [En linea] 2013, [Citado el mayo del 2019] [Disponible en]: http://portalsiget.net/ArchivosSIGET/recursos/Archivos/1682015_AgriculturaFamiliarE.pdf.

MAGAP. 'Ecuador se proyecta a ser exportador de papa'. [En linea] 2014, [Citado el mayo del 2019] [Disponible en]: https://www.agricultura.gob.ec/ecuador-se-proyecta-a-ser-exportador-de-papa/.

MAGAP.(Ministerio de Agricultura, Ganadería, Acuacultura y Pesca) 'REGISTRO DE PRODUCTORES', *ECUADOR*, [En linea] 2014, [Citado el mayo del 2019], [Disponible en]: https://www.agricultura.gob.ec/magap-inicia-registro-de-productores-de-papa-en-cinco-provincias-de-la-sierra/

MAGAP. (Ministerio de Agricultura, Ganadería, Acuacultura y Pesca) 'Informe de rendimientos de papa en el ecuador 2017', pp. 1–15. [En linea] 2018, [Citado el mayo del 2019], [Disponible en]: http://fliphtml5.com/ijia/sfoj/basic.

MAGAP. (Ministerio de Agricultura, Ganadería, Acuacultura y Pesca 'LA POLITICA AGROPECUARIA ECUATORIANA'. [En linea] 2015, [Citado el mayo del 2019], [Disponible en]: http://servicios.agricultura.gob.ec/politicas/La%20Pol%C3%ADtica%20Agropecuaria%20%20al%202025%20II%20parte.pdf

Ministerio de España. 'ROBOTS INDUSTRIALES COLABORATIVOS : Una nueva forma de trabajo', *Seguridad y salud en el trabajo*, 95, p. 84, [En linea] 2018, [Citado el mayo del 2019],[Disponible en]: https://www.insst.es/InshtWeb/Contenidos/Documentacion/MIGRAR%20PUBLICACIONES%20PERIODICAS/Rev_INSHT/2018/SST_95.pdf

Morales García, I. 'Aplicaciones de la robótica en la agricultura, desarrolladas en Holanda', *Vida Rural*, 1, pp. 16–19, [En linea] 2013, [Citado el mayo del 2019]. [Disponible en]: https://www.holanda.es/media/54341/reportaje_vr366.pdf

MPU6050 Arduino, Acelerómetro y Giroscopio. [En linea] 2014, [Citado el mayo del 2019] [Disponible en]: https://hetpro-store.com/TUTORIALES/modulo-acelerometro-y-giroscopio-mpu6050-i2c-twi/.

Muñoz, F. and Cruz, L. 'MANUAL DEL CULTIVO DE PAPA', *Manual del cultivo de papa*, p. 45. [En linea] 1984, [Citado el mayo del 2019], [Disponible en]: http://repositorio.iniap.gob.ec/bitstream/41000/807/4/iniapscm5.pdf.

Museo Informática. 'RASPBERRY PI'. [En linea] 2013, [Citado el mayo del 2019] [Disponible en]: https://histinf.blogs.upv.es/2013/12/18/raspberry-pi/.

Paredes, M. 'Implementación De Un Prototipo De Wsn Con Nodos Inteligentes Para El Sistema De Riego Aplicado a La Agricultura De Precisión Para El Cer – Espoch', p. 22. [En linea] 2017, [Citado el mayo del 2019] [Disponible en]: http://dspace.espoch.edu.ec/bitstream/123456789/7956/1/98T00173.pdf.

PROMUEVEHIDROPONIA. *TRAKÜR, ROBOT QUE ANIQUILA PLAGAS*, *TRAKÜR, ROBOT QUE ANIQUILA PLAGAS*. [En linea] 2014, [Citado el 15 de noviembre] [Disponible en]: http://hidroponia.mx/takur-robot-que-aniquila-plagas/.

Roman Miguel y Hurtado Guillermo. 'Cultivo de la Papa'. [En linea] 2002, [Citado el mayo del 2019], [Disponible en]: http://www.centa.gob.sv/docs/guias/hortalizas/Guia Papa.pdf.

Saha, S. K. 'INTRODUCCION A LA ROBOTICA'. [En linea] 2010, [Citado el mayo del 2019], [Disponible en]: https://ebookcentral.proquest.com/lib/espochsp/detail.action?docID=4585364&fbclid=IwAR1a-dFVtQeo9mCx0Ul6rHllDpHawTrmJtlT4AoOkZtFP2MhayHWizXW8Sk&query=introduccion%20a%20la%20robotica%20

Sensor, E. 'Módulo Encoder Sensor de Velocidad (X2) B83609', pp. 18–20. [En linea] 2019, [Citado el mayo del 2019], [Disponible en]: https://moviltronics.com.co/sensores/115-modulo-encoder-par.html

VELASCO CRUZ JORGE. *El avance de la automatización en la agricultura - Redagrícola, Tendencias tecnológicas en la industria frutícola*. [En linea] 2017,[Disponible en]: http://www.redagricola.com/cl/el-avance-de-la-automatizacion-en-la-agricultura/ [Citado el]: 14 mayo 2019).

VNH2SP30, T. for M. M. S. [En linea] 2018, [Citado el mayo del 2019], [Disponible en]: https://www.instructables.com/id/Monster-Motor-Shield-VNH2SP30/.

ANEXOS

ANEXO A: Características del Arduino UNO

Arduino Uno

Arduino Uno R3 Front

Arduino Uno R3 Back


Arduino Uno R2 Front


Arduino Uno SMD


Arduino Uno Front


Arduino Uno Back

Overview

The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.
The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.
Revision 2 of the Uno board has a resistor pulling the 8U2 HWB line to ground, making it easier to put into DFU mode.
Revision 3 of the board has the following new features:

- 1.0 pinout: added SDA and SCL pins that are near to the AREF pin and two other new pins placed near to the RESET pin, the IOREF that allow the shields to adapt to the voltage provided from the board. In future, shields will be compatible both with the board that use the AVR, which operate with 5V and with the Arduino Due that operate with 3.3V. The second one is a not connected pin, that is reserved for future purposes.
- Stronger RESET circuit.
- Atmega 16U2 replace the 8U2.

"Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the index of Arduino boards.

Summary

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

Schematic & Reference Design

EAGLE files: arduino-uno-Rev3-reference-design.zip (NOTE: works with Eagle 6.0 and newer)
Schematic: arduino-uno-Rev3-schematic.pdf
Note: The Arduino reference design can use an Atmega8, 168, or 328, Current models use an ATmega328, but an Atmega8 is shown in the schematic for reference. The pin configuration is identical on all three processors.

Power

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.
External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.
The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.
The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.** This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We don't advise it.
- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.

Memory

The ATmega328 has 32 KB (with 0.5 KB used for the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
- **External Interrupts: 2 and 3.** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- **PWM: 3, 5, 6, 9, 10, and 11.** Provide 8-bit PWM output with the analogWrite() function.

- **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).** These pins support SPI communication using the SPI library.
- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function. Additionally, some pins have specialized functionality:

- **TWI: A4 or SDA pin and A5 or SCL pin.** Support TWI communication using the Wire library.

There are a couple of other pins on the board:

- **AREF.** Reference voltage for the analog inputs. Used with analogReference().
- **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

See also the mapping between Arduino pins and ATmega328 ports. The mapping for the Atmega8, 168, and 328 is identical.

Communication

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a .inf file is required. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).
A SoftwareSerial library allows for serial communication on any of the Uno's digital pins.
The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation for details. For SPI communication, use the SPI library.

Programming

The Arduino Uno can be programmed with the Arduino software (download). Select "Arduino Uno from the **Tools > Board** menu (according to the microcontroller on your board). For details, see the reference and tutorials.
The ATmega328 on the Arduino Uno comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).
You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.
The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available . The ATmega16U2/8U2 is loaded with a DFU bootloader, which can be activated by:

- On Rev1 boards: connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2.
- On Rev2 or later boards: there is a resistor that pulling the 8U2/16U2 HWB line to ground, making it easier to put into DFU mode.

You can then use Atmel's FLIP software (Windows) or the DFU programmer (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader). See this user-contributed tutorial for more information.

Automatic (Software) Reset

Rather than requiring a physical press of the reset button before an upload, the Arduino Uno is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2/16U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.
This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.
The Uno contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

USB Overcurrent Protection

The Arduino Uno has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Physical Characteristics

The maximum length and width of the Uno PCB are 2.7 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Four screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

ANEXO B: Características del módulo de velocidad encoder.

Double Speed Measuring Module with Photoelectric Encoders

Model: HC-020K

Features:

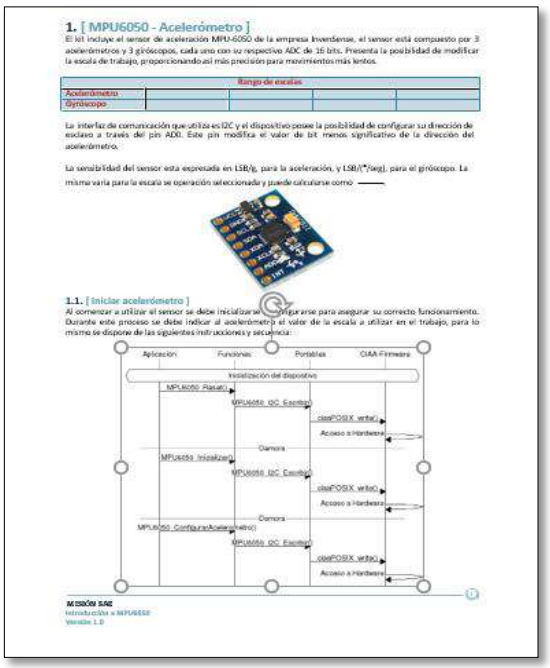
HC-020K speed measuring sensor is a wide voltage, high resolution, short response time, and the switch output speed measurement module. It can test motor's rotational speed with black encoder (measured spec. is related to the encoder, the inner diameter of D type encoder that provided is 4mm, can be used for motor shart w/ 4mm diameter, which is TT motor we matched, yellow shell and white axis).

Specifications:

1. Module Working Voltage: 4.5-5.5V
2. Launch Tube Pressure Drop: VF=1.6V
3. Launch Tube Current: If<20mA
4. Signal output: A, B two lines; TT power level;
5. Resolution: 0.01mm
6. Measurement frequency: 100 KHz
7. Color: Green + black
8. Disc diameter: 24mm
9. Inner Disc Diameter: 4mm
10. Encoder resolution: 20 lines
11. Speed measuring sensor configuration: measure line 1 motor speed
12. Application: For experiment

Made in China

ANEXO C: Características del sensor MPU 6050.



1. [MPU6050 - Acelerómetro]

El kit incluye el sensor de aceleración MPU-6050 de la empresa InvenSense, el sensor está compuesto por 3 acelerómetros y 3 giróscopos, cada uno con su respectivo ADC de 16 bits. Presenta la posibilidad de modificar la escala de trabajo, proporcionando así más precisión para movimientos más lentos.

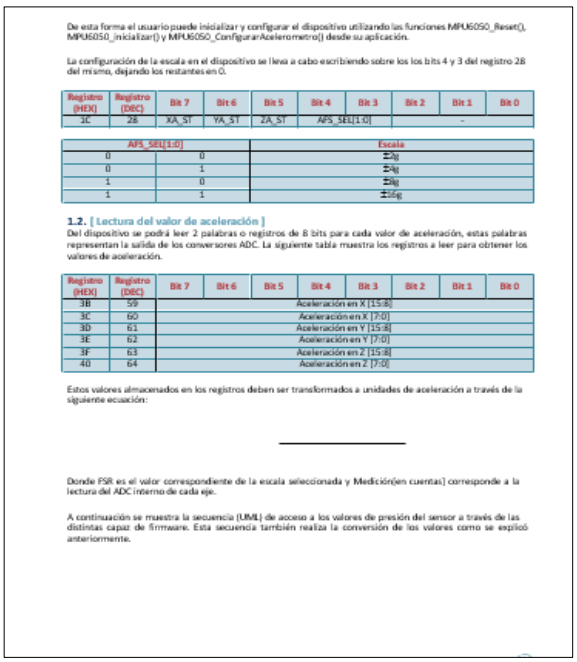
Rango de escalas				
Acelerómetro				
Giroscopo				

La interfaz de comunicación que utiliza es I2C y el dispositivo posee la posibilidad de configurar su dirección de esclavo a través del pin AD0. Este pin modifica el valor de bit menos significativo de la dirección del acelerómetro.

La sensibilidad del sensor esta expresada en LSB/g, para la aceleración, y LSB/(°/seg), para el giróscopo. La misma varía para la escala se operación seleccionada y puede calcularse como: ——

1.1. [Iniciar acelerómetro]

Al comenzar a utilizar el sensor se debe inicializar y configurarse para asegurar su correcto funcionamiento. Durante este proceso se debe indicar al acelerómetro el valor de la escala a utilizar en el trabajo, para lo mismo se dispone de las siguientes instrucciones y secuencia:



De esta forma el usuario puede inicializar y configurar el dispositivo utilizando las funciones MPU6050_Reset(), MPU6050_Inicializar() y MPU6050_ConfigurarAcelerometro() desde su aplicación.

La configuración de la escala en el dispositivo se lleva a cabo escribiendo sobre los los bits 4 y 3 del registro 28 del mismo, dejando los restantes en 0.

Registro (HEX)	Registro (DEC)	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1C	28	XA_ST	YA_ST	ZA_ST	AFS_SEL[1:0]			-	

AFS_SEL[1:0]		Escala
0	0	±2g
0	1	±4g
1	0	±8g
1	1	±16g

1.2. [Lectura del valor de aceleración]

Del dispositivo se podrá leer 2 palabras o registros de 8 bits para cada valor de aceleración, estas palabras representan la salida de los conversores ADC. La siguiente tabla muestra los registros a leer para obtener los valores de aceleración.

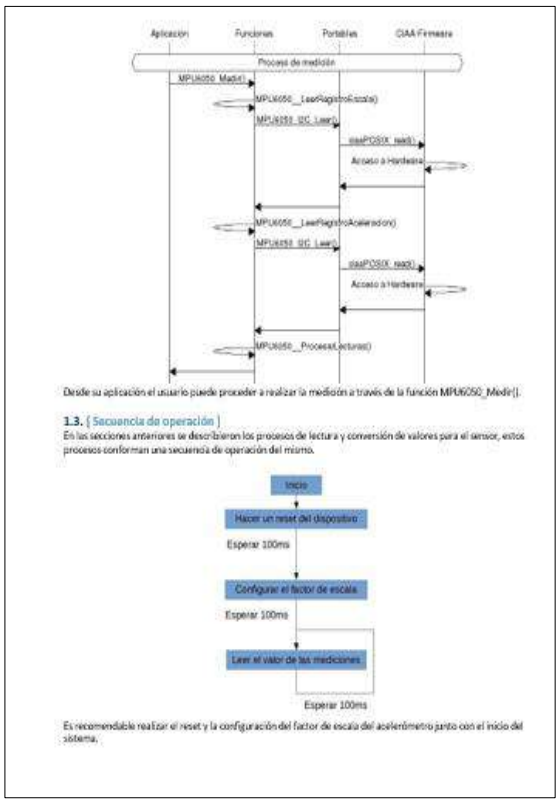
Registro (HEX)	Registro (DEC)	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
3B	59	Aceleración en X [15:8]							
3C	60	Aceleración en X [7:0]							
3D	61	Aceleración en Y [15:8]							
3E	62	Aceleración en Y [7:0]							
3F	63	Aceleración en Z [15:8]							
40	64	Aceleración en Z [7:0]							

Estos valores almacenados en los registros deben ser transformados a unidades de aceleración a través de la siguiente ecuación:

$$\text{Aceleración} = \frac{\text{Medición(en cuentas)}}{\text{FSR}}$$

Donde FSR es el valor correspondiente de la escala seleccionada y Medición(en cuentas) corresponde a la lectura del ADC interno de cada eje.

A continuación se muestra la secuencia (UML) de acceso a los valores de presión del sensor a través de las distintas capaz de firmware. Esta secuencia también realiza la conversión de los valores como se explicó anteriormente.

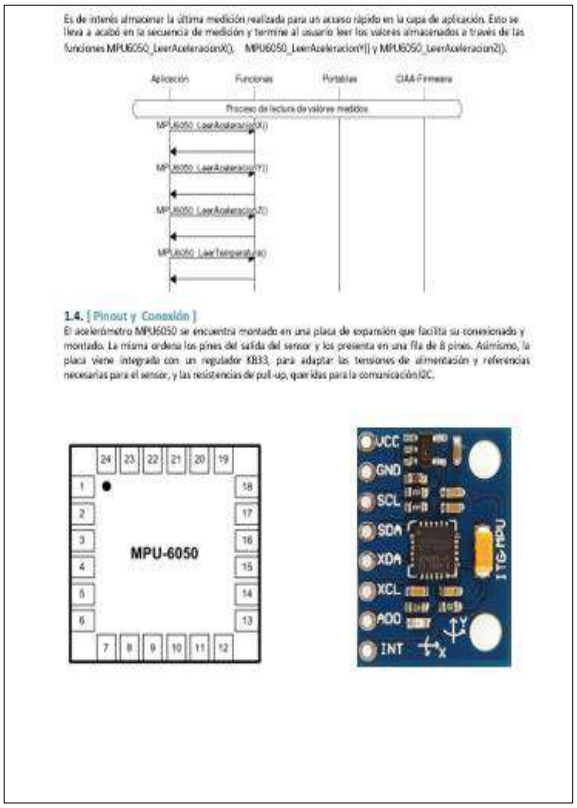


Desde su aplicación el usuario puede proceder a realizar la medición a través de la función MPU6050_Medir().

1.3. [Secuencia de operación]

En las secciones anteriores se describieron los procesos de lectura y conversión de valores para el sensor, estos procesos conforman una secuencia de operación del mismo.

Es recomendable realizar el reset y la configuración del factor de escala del acelerómetro junto con el inicio del sistema.

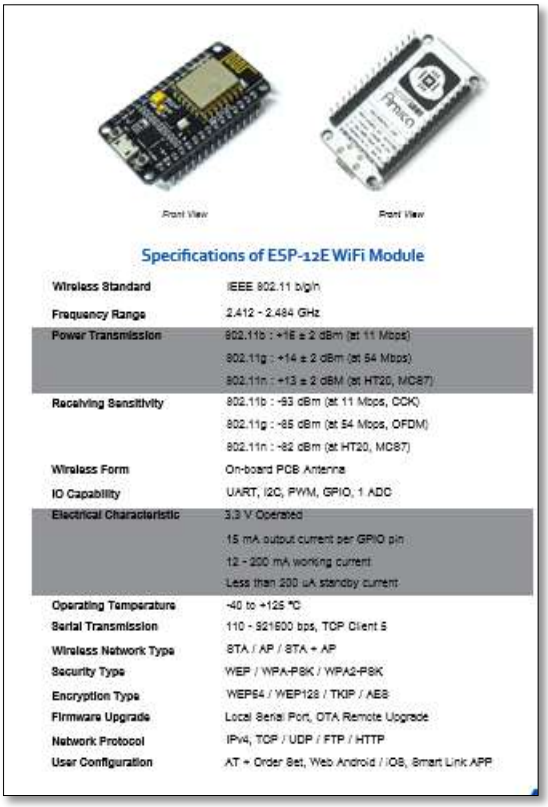


Es de interés almacenar la última medición realizada para un acceso rápido en la capa de aplicación. Esto se lleva a acabo en la secuencia de medición y termina al usuario leer los valores almacenados a través de las funciones MPU6050_LeerAceleracionX(), MPU6050_LeerAceleracionY() y MPU6050_LeerAceleracionZ().

1.4. [Pinout y Conexión]

El acelerómetro MPU6050 se encuentra montado en una placa de expansión que facilita su conexionado y montado. La misma ordena los pines del salida del sensor y los presenta en una fila de 8 pines. Asimismo, la placa viene integrada con un regulador XB33, para adaptar las tensiones de alimentación y referencias necesarias para el sensor, y las resistencias de pull-up, que ldas para la comunicación I2C.

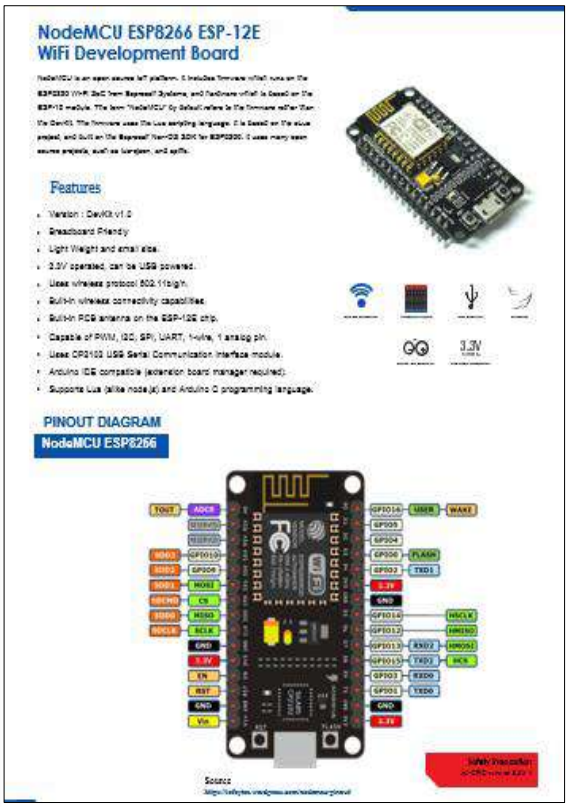
ANEXO D: Características del procesador NodeMCU.



Front View Front View

Specifications of ESP-12E WiFi Module

Wireless Standard	IEEE 802.11 b/g/n
Frequency Range	2.412 - 2.484 GHz
Power Transmission	802.11b : +16 ± 2 dBm (at 11 Mbps) 802.11g : +14 ± 2 dBm (at 54 Mbps) 802.11n : +13 ± 2 dBm (at HT20, MCS7)
Receiving Sensitivity	802.11b : -93 dBm (at 11 Mbps, CCK) 802.11g : -85 dBm (at 54 Mbps, OFDM) 802.11n : -82 dBm (at HT20, MCS7)
Wireless Form	On-board PCB Antenna
IO Capability	UART, I2C, PWM, GPIO, 1 ADC
Electrical Characteristic	3.3 V Operated 15 mA output current per GPIO pin 12 - 200 mA working current Less than 200 uA standby current
Operating Temperature	-40 to +125 °C
Serial Transmission	110 - 921600 bps, TCP Client 5
Wireless Network Type	STA / AP / STA + AP
Security Type	WEP / WPA-PSK / WPA2-PSK
Encryption Type	WEP64 / WEP128 / TKIP / AES
Firmware Upgrade	Local Serial Port, OTA Remote Upgrade
Network Protocol	IPv4, TCP / UDP / FTP / HTTP
User Configuration	AT + Order Set, Web Android / IOS, Smart Link APP



NodeMCU ESP8266 ESP-12E WiFi Development Board

Features

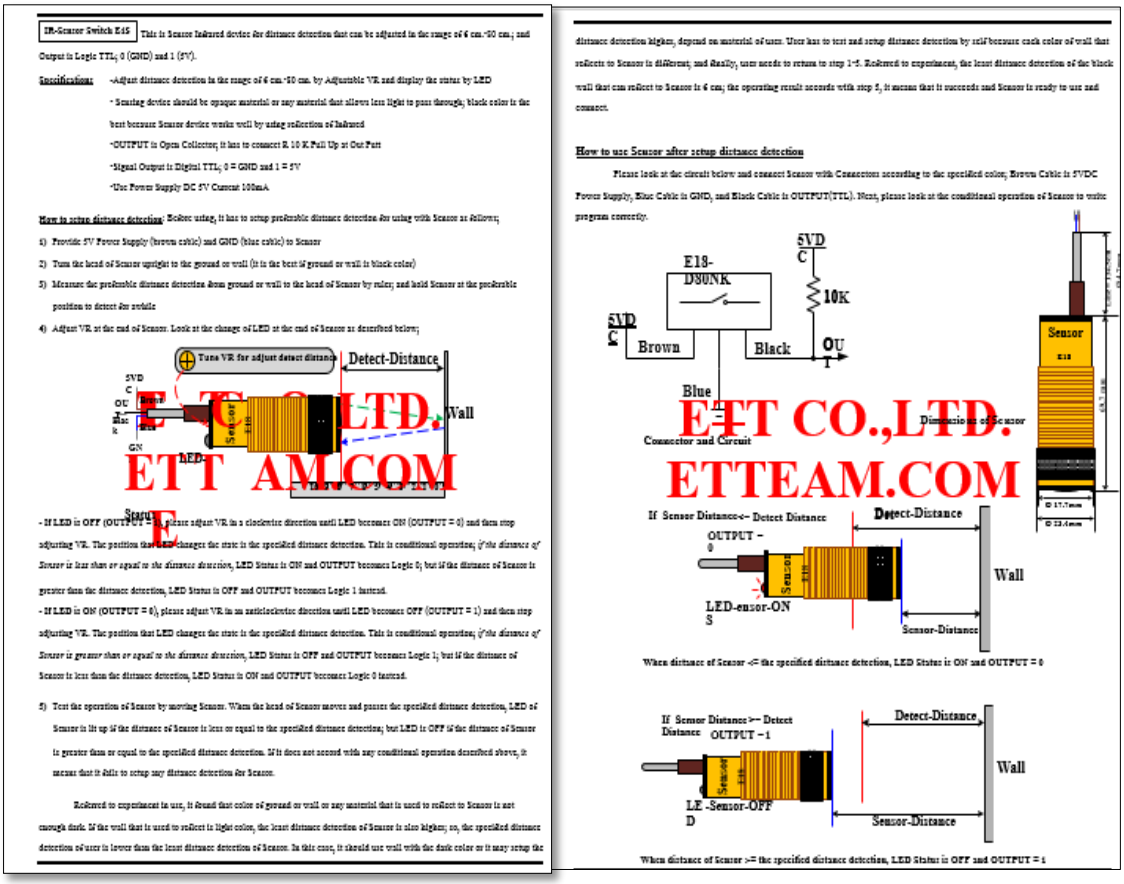
- Version : DevKit v1.0
- Breadboard Friendly
- Light Weight and small size.
- 3.3V operated, can be USB powered.
- Uses wireless protocol 802.11b/g/n.
- Built-in wireless connectivity capabilities.
- Built-in PCB antenna on the ESP-12E chip.
- Capable of PWM, I2C, SPI, UART, 1-wire, 1 analog pin.
- Uses CP2102 USB Serial Communication interface module.
- Arduino IDE compatible (extension board manager required).
- Supports Lua (like node.js) and Arduino C programming language.

PINOUT DIAGRAM

NodeMCU ESP8266

Source

ANEXO E: Sensor fotoeléctrico infrarrojo.



IR-Sensor Switch E18

This is Sensor Infrared device for distance detection that can be adjusted in the range of 6 cm.-80 cm.; and Output is Logic TTL; 0 (GND) and 1 (5V).

Specifications:
-Adjust distance detection in the range of 6 cm.-80 cm. by Adjustable VR and display the status by LED
- Sensing device should be opaque material or any material that allows less light to pass through; black color is the best because Sensor device works well by using reflection of Infrared
-OUTPUT is Open Collector; it has to connect R 10 K Pull Up at Out Put
-Signal Output is Digital TTL; 0 = GND and 1 = 5V
-Use Power Supply DC 5V Current 100mA

How to setup distance detection: Before using; it has to setup preferable distance detection for using with Sensor as follows;

1) Provide 5V Power Supply (brown cable) and GND (blue cable) to Sensor
2) Turn the head of Sensor upright to the ground or wall (it is the best if ground or wall is black color)
3) Measure the preferable distance detection from ground or wall to the head of Sensor by ruler; and hold Sensor at the preferable position to detect for awhile
4) Adjust VR at the end of Sensor. Look at the change of LED at the end of Sensor as described below;

- If LED is OFF (OUTPUT = 1), please adjust VR in a clockwise direction until LED becomes ON (OUTPUT = 0) and then stop adjusting VR. The position that LED changes the state is the specified distance detection. This is conditional operation; *if the distance of Sensor is less than or equal to the distance detection*, LED Status is ON and OUTPUT becomes Logic 0; but if the distance of Sensor is greater than the distance detection, LED Status is OFF and OUTPUT becomes Logic 1 instead.
- If LED is ON (OUTPUT = 0), please adjust VR in an anticlockwise direction until LED becomes OFF (OUTPUT = 1) and then stop adjusting VR. The position that LED changes the state is the specified distance detection. This is conditional operation; *if the distance of Sensor is greater than or equal to the distance detection*, LED Status is OFF and OUTPUT becomes Logic 1; but if the distance of Sensor is less than the distance detection, LED Status is ON and OUTPUT becomes Logic 0 instead.

5) Test the operation of Sensor by moving Sensor. When the head of Sensor moves and passes the specified distance detection, LED of Sensor is lit up if the distance of Sensor is less or equal to the specified distance detection; but LED is OFF if the distance of Sensor is greater than or equal to the specified distance detection. If it does not accord with any conditional operation described above, it means that it fails to setup any distance detection for Sensor.

Referred to experiment in use, it found that color of ground or wall or any material that is used to reflect to Sensor is not enough dark. If the wall that is used to reflect is light color, the least distance detection of Sensor is also higher; so, the specified distance detection of user is lower than the least distance detection of Sensor. In this case, it should use wall with the dark color or it may setup the distance detection higher; depend on material of user. User has to test and setup distance detection by self because each color of wall that reflects to Sensor is different; and finally, user needs to return to step 1~5. Referred to experiment, the least distance detection of the black wall that can reflect to Sensor is 6 cm; the operating result accords with step 5, it means that it succeeds and Sensor is ready to use and connect.

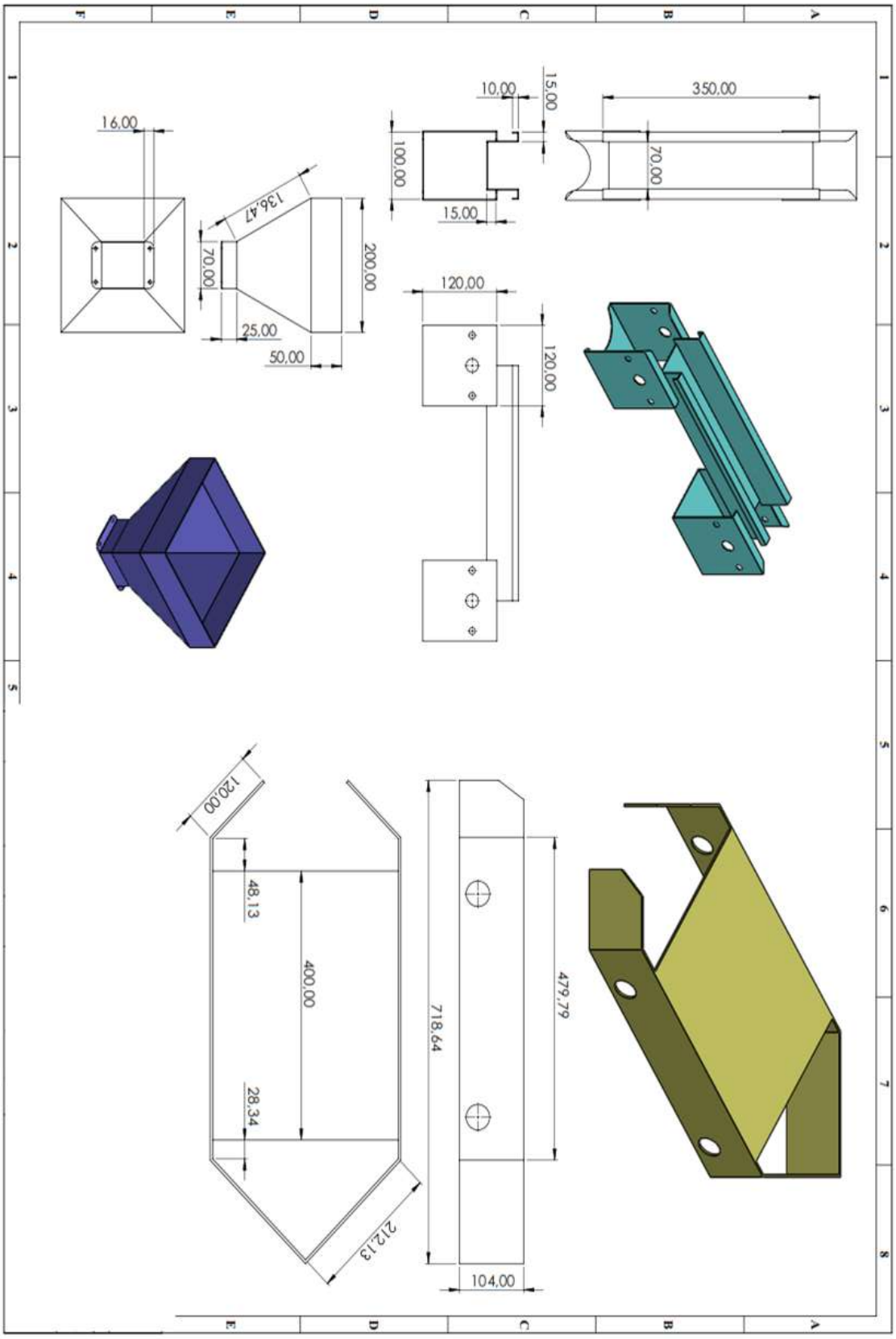
How to use Sensor after setup distance detection

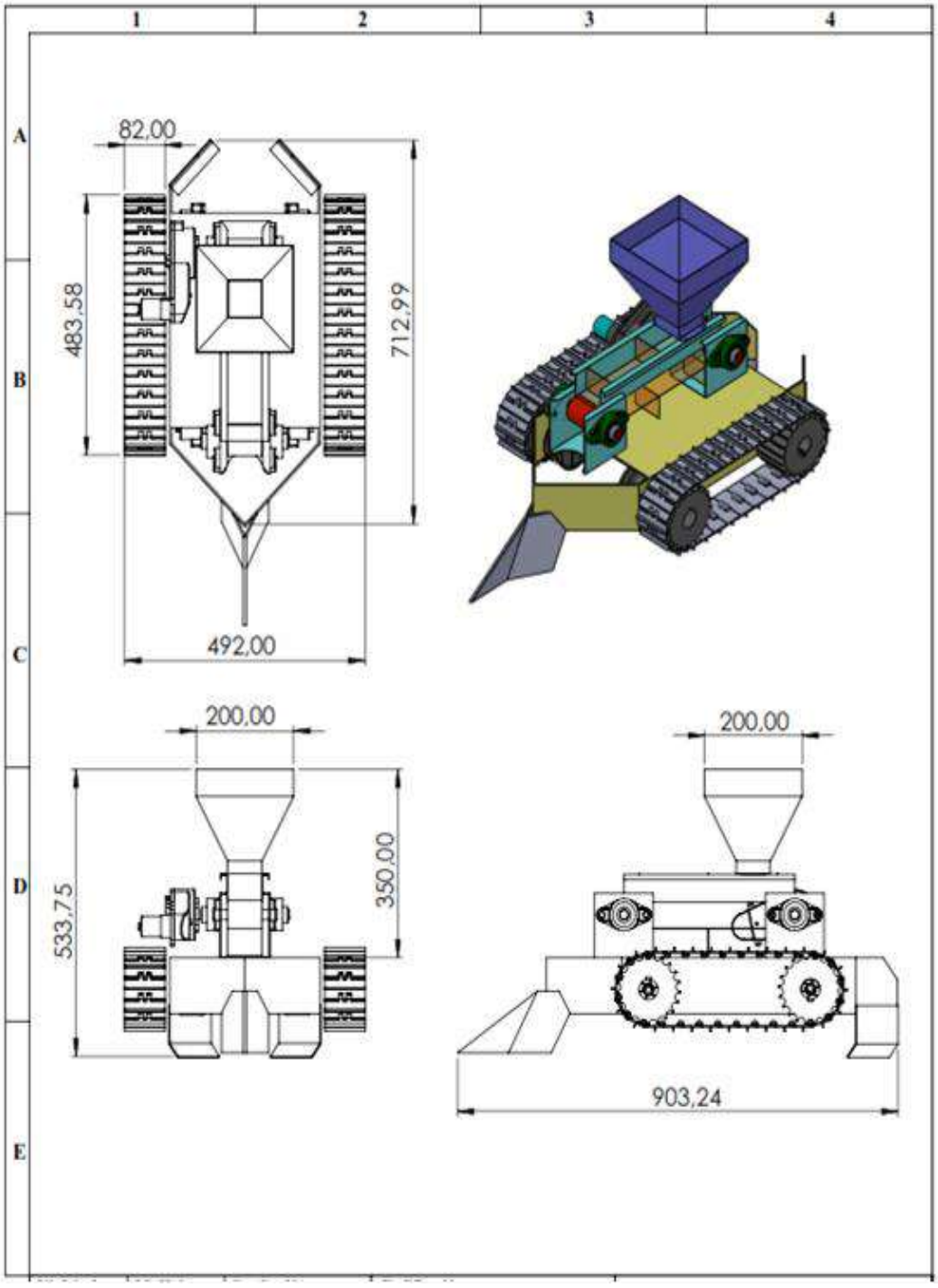
Please look at the circuit below and connect Sensor with Connectors according to the specified color; Brown Cable is 5VDC Power Supply; Blue Cable is GND; and Black Cable is OUTPUT(TTL). Next, please look at the conditional operation of Sensor to write program correctly.

When distance of Sensor <= the specified distance detection, LED Status is ON and OUTPUT = 0

When distance of Sensor >= the specified distance detection, LED Status is OFF and OUTPUT = 1

ANEXO F: Planos Para la Implementación Mecánica




1
2
3
4
A
B
C
D
E
82,00
483,58
712,99
492,00
200,00
533,75
350,00
200,00
903,24

ANEXO G: Formulario de Aceptación.

PROTOTIPO AGSEM

ACEPTACIÓN DEL PROTOTIPO SEMBRADOR DE PAPAS

El prototipo cumple con lo requerimiento de siembra

- ○ si
- ○ no

Su precio es accesible para usted

Tu respuesta

La capacidad de almacenamiento para la siembra es la suficiente para realizar el proceso

- ○ si
- ○ no

El diseño de la aplicación es amigable para el usuario

- ○ si

ANEXO H: Código Arduino

```
#include <math.h>                                          // necesaria para utilizar función atan()
#define PI 3.14159265358979323846264338327795             // definición del número PI

int N = 20;                                                // número de ranuras del encoder
int contadorTicks = 1;                                     // número de ticks para cálculo de velocidad (recordar que entre
menor sea el valor mayor ruido de la medida)
int tam = 10;                                              // tamaño del vector del calculo de promedio (Este valor depende
del tamaño de los vectores de promedio vectorL y vectorR)
int k = 10;                                                // tiempo de muestreo

volatile unsigned muestreoActual = 0;                       // variables para definición del tiempo de muestreo
volatile unsigned muestreoAnterior = 0;
volatile unsigned deltaMuestreo = 0;

float error = 0;                                           // error variables
float Kp = 40;                                             // Contante proporcional control
int PWMr = 0;                                              // PWM de la llanta derecha (señal de control llanta derecha)
int PWMl = 0;                                              // PWM de la llanta izquierda (señal de control llanta
izquierda)

int PWMmax=60;                                             // PWM máximo
int PWMmin=30;                                              // PWM mínimo


///------------------------------ Variables Posición del robot-----------------------------------------
float Cdistancia = 0;                                      // distancia recorrido punto central
float x = 0;                                               // distancia recorrida eje X
float y = 0;                                               // distancia recorrida eje Y
float phi = 0;                                             // posición angular

///------------------------------ Variables Posición deseada -------------------------------------------
float Xd = 100;
float Yd = 100;
float Phid= atan2(Yd-y, Xd-x);

///------------------------------ Variables del robot  -------------------------------------------

float diametro = 6.8;                                       // diametro de la llanta cm
float longitud = 13.4;                                      // longitud del robot entre llantas
float V = 0;                                                // Velocidad lineal del carro
float W = 0;                                                // Velocidad Angular del carro

///------------------------------ Variables de motor derecho-------------------------------------------


  volatile unsigned muestreoActualInterrupcionR = 0;        // variables para definición del tiempo de interrupción y
  calculo de la velocidad motor derecho
  volatile unsigned muestreoAnteriorInterrupcionR = 0;
  double deltaMuestreoInterrupcionR = 0;

  int encoderR = 3;   // pin de conexión del encoder derecho
  int llantaR = 11;      // pin de conexión de llanta derecha   (pin de PWM)

  double frecuenciaR = 0;                                   // frecuencia de interrupción llanta R
  double Wr = 0;                                            // Velocidad angular R
  double Vr = 0;                                            // velocidad Lineal
  int CR = 0;                                               // contador ticks
  float vectorR[] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};    // vector de almacenamiento de datos para promedio del tiempo de
  interrupciones

  float Rdistancia = 0;                                     // distancia recorrida llanta derecha
  int Rtick = 0;                                            // ticks del encoder derecho
  int RtickAnt = 0;                                         // ticks del encoder derecho anteriores
  int deltaRtick = 0;                                       // diferencia del encoder derecho

  //------------------------------  Variables de motor Izquierdo ----------------------------------------------

  volatile unsigned muestreoActualInterrupcionL = 0;        // variables para definición del tiempo de interrupción y
  calculo de la velocidad motor Izquierdo
  volatile unsigned muestreoAnteriorInterrupcionL = 0;
  double deltaMuestreoInterrupcionL = 0;

  int encoderL = 2;   // pin de conexión del encoder Izquierdo
  int llantaL = 10;      // pin de conexión de llanta Izquierda   (pin de PWM)

  double frecuenciaL = 0;                                   // frecuencia de interrupción llanta Izquierda
  double Wl = 0;                                            // Velocidad angular L
  double Vl = 0;                                            // velocidad Lineal
  int CL = 0;                                               // contador Ticks
  float vectorL[] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};    // vector de almacenamiento de datos para promedio del tiempo de
  interrupciones

  float Ldistancia = 0;                                     // distancia recorrida llanta izquierda
  int Ltick = 0;                                            // ticks del encoder izquierdo
  int LtickAnt = 0;                                         // ticks del encoder izquier anteriores
  int deltaLtick = 0;                                       // diferencia del encoder izquierdo
```

```
void setup() {
  attachInterrupt(digitalPinToInterrupt(encoderR),REncoder,FALLING);                // linea para añadir una
interrupción a un PIN
  attachInterrupt(digitalPinToInterrupt(encoderL),LEncoder,FALLING);                // linea para añadir una
interrupción a un PIN
  Serial.begin(9600);                                                               // inicio de la comunicación serial
}

void REncoder() {                                                                                        // función de
interrupción del enconder llanta derecha
      Rtick++;                                                                                          // Número de
ticks llanta derecha
      CR++;                                                                                              // incremento
del contador de ticks
      if (CR == contadorTicks){                                                                          // si el
contador de ticks alcanza el valor de ticks determinado para el cálculo del tiempo
          float media = 0;                                                                               // variable
creada para cálculo del promedio
//------------------------------------- ----------------------------    Filtro promedio    ----------------------------
-----------------------------------------------------//
          for(int i=0;i < tam-1;i++){                                                                    // relleno del
vector para cálculo posterior del promedio
              vectorR[i]=vectorR[i+1];
          }
          vectorR[tam-1]=deltaMuestreoInterrupcionR ;                                                    // último dato
del vector (medida actual)

          for(int i=0;i<tam;i++){                                                                        // Suma de los
valores del vector
              media = vectorR[i]+ media;
          }
          media = media/tam;                                                                             //división por
el total de datos del vector
          deltaMuestreoInterrupcionR = media;                                                            // se reemplaza
por el valor de su medio.
//------------------------------------- ---------------------------- ----------------------------------------------
-----------------------------------------------------//
          frecuenciaR = (1000)/ deltaMuestreoInterrupcionR;                                              // frecuencia
de interrupción
          muestreoAnteriorInterrupcionR = muestreoActualInterrupcionR;                                   // se actualiza
el tiempo de interrupción anterior
          CR = 0;                                                                                        //Reinicio de
contador de ticks



      }
}

void LEncoder() {                                                                                       // función de
interrupción del enconder llanta izquierda
      Ltick++;                                                                                          // Número de
ticks llanta izquierda
      CL++;                                                                                             // incremento
del contador de ticks
      if (CL == contadorTicks){                                                                         // si el
contador de ticks alcanza el valor de ticks determinado para el cálculo del tiempo
          float media = 0;                                                                              // variable
creada para cálculo del promedio
//------------------------------------- ----------------------------    Filtro promedio    ----------------------------
-----------------------------------------------------//
          for(int i=0;i < tam-1;i++){                                                                   // relleno del
vector para calculo posterior del promedio
              vectorL[i]=vectorL[i+1];
          }
          vectorL[tam-1]=deltaMuestreoInterrupcionL;                                                    // último dato
del vector (medida actual)

          for(int i=0;i<tam;i++){                                                                       // Suma de los
valores del vector
              media = vectorL[i]+ media;
          }
          media = media/tam;                                                                            //división por
el total de datos del vector
          deltaMuestreoInterrupcionL = media;                                                           // se reemplaza
por el valor de su medio.
//------------------------------------- ---------------------------- ----------------------------------------------
-----------------------------------------------------//
          frecuenciaL = (1000)/ deltaMuestreoInterrupcionL;                                             // frecuencia
de interrupción
          muestreoAnteriorInterrupcionL = muestreoActualInterrupcionL;                                  // se actualiza
el tiempo de interrupción anterior
          CL = 0;                                                                                       // Reinicio de
contador de ticks
      }
}


void loop() {
```

```
    muestreoActual = millis();                                                          //Tiempo actual
de muestreo
    muestreoActualInterrupcionR = millis();                                             // se asigna el
tiempo de ejecución a el muestreo actual
    muestreoActualInterrupcionL = millis();                                             // se asigna el
tiempo de ejecución a el muestreo actual

    deltaMuestreo =(double) muestreoActual - muestreoAnterior;                          // delta de
muestreo
    if ( deltaMuestreo >= k)                                                            // se asegura
el tiempo de muestreo
    {
        float Phid= atan2(Yd-y, Xd-x);                                                  // Recalcular
el ángulo deseado en cada iteración, dado que el cambia con respecto  a cada movimiento

        deltaMuestreoInterrupcionR = muestreoActualInterrupcionR -  muestreoAnteriorInterrupcionR;      // diferencia
tiempos de interruciones de ticks del motor
        deltaMuestreoInterrupcionL = muestreoActualInterrupcionL -  muestreoAnteriorInterrupcionL;      // diferencia
tiempos de interruciones de ticks del motor

        if(deltaMuestreoInterrupcionR >= 200*contadorTicks){                            // Esta es la
forma de definir cuando el motor se encuentra quieto. Si deltaMuestreoInterrupcionR es mayor a 40 milisegundos por el
preescalado de ticks
          frecuenciaR=0;                                                                // 40 mS es el
tiempo que máximo se tarda un tick a la menor velocidad del motor
        }
        if(deltaMuestreoInterrupcionL >= 200*contadorTicks){                            // Esta es la
forma de definir cuando el motor se encuentra quieto. Si deltaMuestreoInterrupcionR es mayor a 40 milisegundos por el
preescalado de ticks
          frecuenciaL=0;                                                                // 40 mS es el
tiempo que máximo se tarda un tick a la menor velocidad del motor
        }

        Wr = contadorTicks*((2*PI)/N)*frecuenciaR;                                      // frecuencia
angular Rad/s
        Vr= Wr*(diametro/2);                                                            // velocidad
lineal cm/s
        Wl = contadorTicks*((2*PI)/N)*frecuenciaL;                                      // frecuencia
angular Rad/s
        Vl= Wl*(diametro/2);                                                            // velocidad
lineal cm/s



//        V = (Vr+Vl)/2;                                                                 // calculo
de la velocidad del robot
        V = 50;                                                                         // velocidad
constante para alcanzar el ángulo
        error = Phid - phi;                                                             // error
angular Angulo deseado menos el angulo del robot
        W = (Vr-Vl)/longitud + Kp * error;                                              // Cálculo de
la velocidad angular con las variables de control
        PWMr = V + (W*longitud)/2;                                                      // Señal de
control PWM llanta derecha
        PWMl = V -  (W*longitud)/2;                                                     // Señal de
control PWM llanta izquierda

//------------------------------------ condicionales para limites de la señal de PWM --------------------------------
-------------------------------------------------------------------------//

        if(PWMr > PWMmax){
          PWMr = PWMmax;
        }
        if(PWMr < PWMmin){
          PWMr = PWMmin;
        }
        if(PWMl > PWMmax){
          PWMl = PWMmax;
        }
        if(PWMl < PWMmin){
          PWMl = PWMmin;
        }

        if( abs(x-Xd) < 5 && abs(y-Yd) < 5){
          analogWrite(llantaR,0);
          analogWrite(llantaL,0);
        }
        else {
          analogWrite(llantaR,PWMr);
          analogWrite(llantaL,PWMl);
       }

//        analogWrite(llantaR,PWMr);                                                     // PWM de
la llanta derecha
//        analogWrite(llantaL,PWMl);                                                     // PWM de
la llanta izquierda
```

```
//        analogWrite(llantaR,0);                                                    // PWM de la
llanta derecha
//        analogWrite(llantaL,0);                                                    // PWM de la
llanta izquierda

        odometria();                                                                // cálculo de
la odometria

        Serial.print(x);                                                            // se muestra
el tiempo entre TIC y TIC
        Serial.print(",");                                                          // se muestra
el tiempo entre TIC y TIC
        Serial.println(y);                                                          // se muestra
el tiempo entre TIC y TIC

        muestreoAnterior = muestreoActual;                                          //
actualización del muestreo anterior
    }
}

void odometria(){

   deltaRtick = Rtick - RtickAnt;                                                   // comparación
de los ticks recorridos desde el último cálculo llanta derecha
   Rdistancia = PI*diametro*(deltaRtick/(double) 20);                               // distancia
recorrida por la llanta derecha desde el último cálculo

   deltaLtick = Ltick - LtickAnt;                                                   // comparación
de los ticks recorridos desde el último cálculo llanta izquierda
   Ldistancia = PI*diametro*(deltaLtick/(double) 20);                               // distancia
recorrida por la llanta izquierda desde el último cálculo

   Cdistancia = (Rdistancia + Ldistancia)/2;                                        // distancia
recorrida por el punto central desde el último cálculo

   x = x + Cdistancia*cos(phi);                                                     // posición
del punto X actual
   y = y + Cdistancia*sin(phi);                                                     // posición
del punto Y actual

   phi = phi + ((Rdistancia - Ldistancia)/longitud);                                // posición
Angular actual



   phi =
atan2(sin(phi),cos(phi));                                                   //transformación de la
posición angular entre -PI y PI

   RtickAnt = Rtick;                                                                //
actualización de la variable RtickAnt con los valores de Rtick
   LtickAnt = Ltick;                                                                //
actualización de la variable LtickAnt con los valores de Ltick
}
```

ANEXO I: Código de Android Studio.

```
package com.andrognito.patternlockdemo;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.view.WindowManager;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.LinearLayout;

public class Inicio extends AppCompatActivity {

    ImageView bgapp, clover;
    LinearLayout textsplash, texthome, menus;
    Animation frombottom;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_inicio);


getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,WindowManager.LayoutParams.FLAG_FULLSCREEN);


        frombottom = AnimationUtils.loadAnimation(this, R.anim.frombottom);

        bgapp = (ImageView) findViewById(R.id.bgapp);
        clover = (ImageView) findViewById(R.id.clover);
        textsplash = (LinearLayout) findViewById(R.id.textsplash);
        texthome = (LinearLayout) findViewById(R.id.texthome);
        menus = (LinearLayout) findViewById(R.id.menus);

        bgapp.animate().translationY(-2200).setDuration(1800).setStartDelay(1300);
        clover.animate().alpha(0).setDuration(1800).setStartDelay(1600);


        textsplash.animate().translationY(140).alpha(0).setDuration(1800).setStartDelay(1300);

        texthome.startAnimation(frombottom);
        menus.startAnimation(frombottom);

        Button btnIniciar=(Button)findViewById(R.id.btnInicio);


        btnIniciar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(Inicio.this, MainActivity.class));
                finish();

            }
        });
    }
}


package com.andrognito.patternlockdemo;

import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.text.Editable;
import android.text.TextUtils;
import android.text.TextWatcher;
import android.util.Log;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import com.andrognito.patternlockview.PatternLockView;
import com.andrognito.patternlockview.listener.PatternLockViewListener;
import com.andrognito.patternlockview.utils.PatternLockUtils;
```

```
import com.andrognito.patternlockview.utils.ResourceUtils;
import com.andrognito.rxpatternlockview.RxPatternLockView;
import com.andrognito.rxpatternlockview.events.PatternLockCompleteEvent;
import com.andrognito.rxpatternlockview.events.PatternLockCompoundEvent;

import java.util.List;

import io.reactivex.functions.Consumer;

public class MainActivity extends AppCompatActivity {

    private PatternLockView mPatternLockView;
    private EditText editLargo,editAncho,editSurcos;
    private Button btnEnviar;

    private PatternLockViewListener mPatternLockViewListener = new PatternLockViewListener() {
        @Override
        public void onStarted() {
            Log.d(getClass().getName(), "Pattern drawing started");
        }

        @Override
        public void onProgress(List<PatternLockView.Dot> progressPattern) {
            Log.d(getClass().getName(), "Pattern progress: " +
                    PatternLockUtils.patternToString(mPatternLockView, progressPattern));
        }

        @Override
        public void onComplete(List<PatternLockView.Dot> pattern) {
            Log.d(getClass().getName(), "Pattern complete: " +
                    PatternLockUtils.patternToString(mPatternLockView, pattern));
        }

        @Override
        public void onCleared() {
            Log.d(getClass().getName(), "Pattern has been cleared");
        }
    };


     @Override
     protected void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         requestWindowFeature(Window.FEATURE_NO_TITLE);
         getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                 WindowManager.LayoutParams.FLAG_FULLSCREEN);

         setContentView(R.layout.activity_main);

        btnEnviar=(Button)findViewById(R.id.btnenviar);

         editAncho=(EditText)findViewById(R.id.edAncho);
         editLargo=(EditText)findViewById(R.id.edLargo);
         editSurcos=(EditText)findViewById(R.id.edSurcos);

         mPatternLockView = (PatternLockView) findViewById(R.id.patter_lock_view);



         editSurcos.addTextChangedListener(mTextWatcher);


         // run once to disable if empty
         checkFieldsForEmptyValues();


         btnEnviar.setOnClickListener(new View.OnClickListener() {
             @Override
             public void onClick(View v) {
                 startActivity(new Intent(MainActivity.this,Controles.class));
                 finish();
             }
         });

     }
```

```
    private TextWatcher mTextWatcher = new TextWatcher() {
        @Override
        public void beforeTextChanged(CharSequence charSequence, int i, int i2, int i3) {
        }

        @Override
        public void onTextChanged(CharSequence charSequence, int i, int i2, int i3) {
        }

        @Override
        public void afterTextChanged(Editable editable) {
            // check Fields For Empty Values


            checkFieldsForEmptyValues();
         //   Toast.makeText(MainActivity.this,"si llego",Toast.LENGTH_SHORT).show();
        }
    };



    void checkFieldsForEmptyValues(){


        if(TextUtils.isEmpty(editSurcos.getText()) ){
            mPatternLockView.setVisibility(View.INVISIBLE);
        } else {
            mPatternLockView.setVisibility(View.VISIBLE);

            String surcos= editSurcos.getText().toString();
            int numsurcos=Integer.parseInt(surcos);

            mPatternLockView.setDotCount(numsurcos);

            mPatternLockView.setDotNormalSize((int) ResourceUtils.getDimensionInPx(this,
R.dimen.pattern_lock_dot_size));



            mPatternLockView.setDotSelectedSize(30);
            mPatternLockView.setPathWidth(10);
            mPatternLockView.setAspectRatioEnabled(true);
            mPatternLockView.setAspectRatio(PatternLockView.AspectRatio.ASPECT_RATIO_HEIGHT_BIAS);
            mPatternLockView.setViewMode(PatternLockView.PatternViewMode.CORRECT);
            mPatternLockView.setDotAnimationDuration(5);
            mPatternLockView.setPathEndAnimationDuration(10);
            mPatternLockView.setCorrectStateColor(ResourceUtils.getColor(this, R.color.white));
            mPatternLockView.setInStealthMode(false);
            mPatternLockView.setTactileFeedbackEnabled(true);
            mPatternLockView.setInputEnabled(true);
            mPatternLockView.addPatternLockListener(mPatternLockViewListener);

            RxPatternLockView.patternComplete(mPatternLockView)
                    .subscribe(new Consumer<PatternLockCompleteEvent>() {
                        @Override
                        public void accept(PatternLockCompleteEvent patternLockCompleteEvent) throws
Exception {
                            Log.d(getClass().getName(), "Complete: " +
patternLockCompleteEvent.getPattern().toString());

                            btnEnviar.setVisibility(View.VISIBLE);


                        }
                    });

            RxPatternLockView.patternChanges(mPatternLockView)
                    .subscribe(new Consumer<PatternLockCompoundEvent>() {
                        @Override
                        public void accept(PatternLockCompoundEvent event) throws Exception {
                            if (event.getEventType() == PatternLockCompoundEvent.EventType.PATTERN_STARTED)
{
                                Log.d(getClass().getName(), "Pattern drawing started");
                                btnEnviar.setVisibility(View.INVISIBLE);
                            } else if (event.getEventType() ==
PatternLockCompoundEvent.EventType.PATTERN_PROGRESS) {
                                Log.d(getClass().getName(), "Pattern progress: " +
                                        PatternLockUtils.patternToString(mPatternLockView,
```

```
event.getPattern()));
                                btnEnviar.setVisibility(View.INVISIBLE);
                            } else if (event.getEventType() ==
PatternLockCompoundEvent.EventType.PATTERN_COMPLETE) {
                                Log.d(getClass().getName(), "Pattern complete: " +
                                        PatternLockUtils.patternToString(mPatternLockView,
event.getPattern()));
                            } else if (event.getEventType() ==
PatternLockCompoundEvent.EventType.PATTERN_CLEARED) {
                                Log.d(getClass().getName(), "Pattern has been cleared");
                                btnEnviar.setVisibility(View.INVISIBLE);
                            }
                        }
                    });


        }
    }



}
```

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".Controles"
    android:orientation="vertical"
    android:background="@drawable/fondopantalla"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    >


<ImageView
    android:layout_width="240dp"
    android:layout_height="50dp"
    android:background="@drawable/titulo2"
    android:layout_gravity="center_horizontal"
    />


<View
    android:layout_width="match_parent"
    android:layout_height="2dp"
    android:layout_marginTop="5dp"
    android:background="@color/white"/>


<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    >

    <ImageView
        android:layout_width="28dp"
        android:layout_height="28dp"
        android:layout_weight="3"
        android:layout_marginTop="2dp"
        android:background="@drawable/reloj"
        android:layout_marginRight="10dp"
        android:layout_marginLeft="10dp"


        />

    <TextView
        android:id="@+id/TITULO"
        android:layout_width="wrap_content"
```

```
            android:layout_height="wrap_content"
            android:layout_marginTop="2dp"
            android:maxLines="1"
            android:fontFamily="sans-serif"
            android:textStyle="bold"
            android:layout_marginLeft="10dp"
            android:text="Duración del recorrido: "
            android:textColor="@color/white"
            android:textSize="17sp"/>



        <Chronometer
            android:id="@+id/cronometro"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:layout_marginTop="2dp"

            android:fontFamily="sans-serif"
            android:textStyle="bold"

            android:layout_marginLeft="30dp"

            android:textColor="@color/white"
            android:textSize="17sp"/>



    </LinearLayout>

    <View
        android:layout_width="match_parent"
        android:layout_height="2dp"


        android:background="@color/white"/>



</LinearLayout>
```

```
package com.andrognito.patternlockdemo;

import android.os.SystemClock;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.WindowManager;
import android.widget.Chronometer;
import android.widget.TextView;

public class Controles extends AppCompatActivity {


    Chronometer crono;

    long Time=0;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_controles);


getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,WindowManager.LayoutParams.FLAG_FULLSCREEN);


        crono=(Chronometer)findViewById(R.id.cronometro);

        crono.setBase(SystemClock.elapsedRealtime());
        crono.start();
```

```
    }
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/fondopantalla"
    android:orientation="vertical"

    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"

    >

    <ImageView
        android:layout_width="240dp"
        android:layout_height="50dp"
        android:background="@drawable/titulo2"
        android:layout_gravity="center_horizontal"
        />


    <TextView
        android:id="@+id/TITULO"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="2dp"
        android:maxLines="1"
        android:fontFamily="sans-serif"
        android:textStyle="bold"
        android:text="Caracteristicas del Terreno"
        android:textColor="@color/white"
        android:textSize="17sp"/>


   <View
       android:layout_width="match_parent"
       android:layout_height="2dp"
       android:layout_marginTop="5dp"
       android:background="@color/white"/>



       <LinearLayout
           android:layout_width="match_parent"
           android:layout_height="wrap_content"
           android:orientation="horizontal"
           android:layout_gravity="center_horizontal"
           >

           <TextView
               android:layout_width="match_parent"
               android:layout_height="wrap_content"
               android:layout_weight="1"
               android:textColor="@color/white"
               android:text="Ancho(metros):"
               android:textStyle="bold"
               />

           <EditText
               android:id="@+id/edLargo"
               android:layout_width="match_parent"
               android:layout_height="wrap_content"
               android:layout_weight="1"
               android:inputType="number"
               android:background="@drawable/customedittext"
               android:textColor="@color/mine_shaft"
               android:layout_marginLeft="30dp"
               android:layout_marginRight="30dp"


               />
```

```
    </LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_gravity="center_horizontal"
    >

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:textColor="@color/white"
        android:textStyle="bold"
        android:text="Largo (metros)"
        />

    <EditText
        android:id="@+id/edAncho"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:inputType="number"
        android:background="@drawable/customedittext"
        android:textColor="@color/mine_shaft"
        android:layout_marginLeft="30dp"
        android:layout_marginRight="30dp"

        android:hint="Ancho (metros)"


        />



    </LinearLayout>


    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_gravity="center_horizontal"
        >

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:textColor="@color/white"
            android:text="Número de Surcos: "
            />

        <EditText
            android:id="@+id/edSurcos"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:inputType="number"
            android:background="@drawable/customedittext"
            android:textColor="@color/mine_shaft"
            android:layout_marginLeft="30dp"
            android:layout_marginRight="30dp"

            android:hint="Surcos (#)"

            />



    </LinearLayout>
```

```
    <Button
        android:id="@+id/btnenviar"
        android:layout_width="180dp"
        android:layout_height="50dp"
        android:layout_marginTop="10dp"
        android:layout_gravity="center_horizontal"
        android:background="@drawable/img_btnenviar"
        android:visibility="invisible"
        />




    <View
        android:layout_width="match_parent"
        android:layout_height="2dp"
        android:background="@color/white"
        android:layout_marginTop="5dp"
        />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:maxLines="1"
        android:fontFamily="sans-serif"
        android:text="Diseño de la Trayectoria"
        android:textColor="@color/white"
        android:layout_gravity="center_horizontal"
        android:textStyle="bold"
        android:textSize="17sp"/>


    <com.andrognito.patternlockview.PatternLockView
        android:id="@+id/patter_lock_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="5dp"
        app:aspectRatio="height_bias"
        app:aspectRatioEnabled="true"
        android:visibility="invisible"
        app:dotCount="8"/>


</LinearLayout>
```