Universidad de Mendoza

Facultad de Ingeniería

Ingeniería en Electrónica

"Laboratorio de robótica móvil en Cloud"

Autor: Biasotti, Victor Hugo

Profesor Asesor: Dr. Ing. Iacono Lucas
Profesor Co-Asesor: Ing. Fontana Daniel

Mendoza, Agosto de 2019

AGRADECIMIENTOS

Quiera agradecer especialmente a mis padres Biasotti, Jose y Fernandez, Elsa por su apoyo incondicional a lo largo de la carrera, y a mi hermano Biasotti, Bruno con quien tuve la fortuna de poder compartir esta carrera con él.

También debo agradecer al GRUM (Grupo de Robótica Universidad de Mendoza), lugar donde encontré “mi sitio” dentro de la universidad, compartiendo valiosas experiencias con todos sus miembros. Y en especial a Iacono, Lucas por el asesoramiento durante este trabajo final y en todas las situaciones académicas.

Finalmente agradecer a todos los docentes y compañeros de la universidad, quienes de alguna u otra forma ayudaron a que hoy esto sea posible.

RESUMEN

En el presente trabajo final se desarrolló un conjunto de robots móviles con capacidad de generación y seguimiento de trayectoria con procesamiento en el Cloud, el cual permitirá la evaluación y experimentación de diferentes algoritmos para dicho fin. El sistema planteado deja abierta la posibilidad de incorporar otros servicios del Cloud como por ejemplo ROS (Robot Operating Sistem).

Todos los componentes utilizados en este trabajo son adquiribles en el mercado local a bajos costos, logrando así que el sistema pueda ser expandido fácilmente.

Este trabajo es la base para el proyecto de investigación “laboratorio de robótica en el Cloud”, el cual ha sido financiado por DIUM (Dirección de Investigaciones de la Universidad de Mendoza), y cuenta con un enfoque más amplio, incluyendo también a robots industriales.

Al finalizar el proyecto, se evaluó al conjunto de robots móviles desarrollados con una generación y seguimiento de trayectoria simple, logrando los resultados esperados.

ÍNDICE

1 INTRODUCCIÓN

Actualmente, los avances en robótica y electrónica han dado lugar al surgimiento de robots móviles e industriales educativos de bajo costo económico. Estos robots pueden utilizarse para llevar a cabo desarrollos e investigaciones en aspectos de la robótica móvil e industrial como: percepción, planeamiento, control, estimación de posición, configuración mecánica, control, simulación de procesos, etc. Además, gracias a Cloud Computing, se puede acceder a recursos computacionales de altas prestaciones para procesar los algoritmos requeridos para controlar dichos robots.

Dichas tecnologías permiten desarrollar laboratorios compuestos por varios robots, los cuales puedan comunicarse y realizar actividades colaborativas (como por ejemplo búsqueda y rescate, simulación de actividades industriales, etc.). Para realizar actividades colaborativas, los robots deben comunicarse entre ellos y definir una estrategia conjunta para solucionar un problema determinado. Dicha estrategia puede resolverse empleando algoritmos de distinto tipo utilizando técnicas de aprendizaje de máquinas y metaheurísticas.

El desarrollo de estos algoritmos representa un verdadero desafío para estudiantes de Ingeniería Industrial, en Computación, Informática y Electrónica. Es por este motivo que se requiere contar con un laboratorio de Robótica, el cual permita desarrollar algoritmos para controlar los robots y explorar nuevas opciones para mejorar las técnicas de control de robots móviles e industriales.

Cloud Computing proporciona a los usuarios un nuevo paradigma para el desarrollo de aplicaciones y la utilización de recursos de cómputo y almacenamiento. Gracias al uso de virtualización y servicios web, los recursos de hardware y las aplicaciones pueden ser provistas dinámicamente al usuario.

En [1], Buyya et al. definen a Cloud Computing como “Un sistema de computación distribuida orientado al mercado que consiste en un conjunto de computadoras interconectadas y virtualizadas. Dicho sistema se provee y presenta en forma dinámica como uno o más recursos de cómputo unificado (s) en base a acuerdos de servicio establecidos mediante negociaciones entre el proveedor y los consumidores de servicios”.
Generalmente, la tecnología Cloud ha sido utilizada para ejecutar aplicaciones que requieren acceso a recursos de hardware de altas prestaciones. Sin embargo, en la última década han surgido una serie de propuestas que plantean la posibilidad de utilizar servicios de Cloud Computing para gestionar, almacenar y procesar la información y algoritmos de control de robots. Dichas propuestas han dado lugar a un nuevo tipo de infraestructura IT denominada Robótica en el Cloud.

El paradigma de Robótica en el Cloud plantea que los robots pueden beneficiarse del poder computacional, almacenamiento, y recursos de comunicación provistos por el Cloud. Esto servicios de Cloud Computing permiten procesar e intercambiar información proveniente de varios robots o agentes (dispositivos, otras máquinas, personas, etc.). Además, las personas pueden asignar tareas a los robots de forma remota a través de redes.

El uso de Cloud Computing permite dotar a los robots con capacidades de procesamiento remoto. Esto permite reducir costos económicos ya que los robots tienen un “cerebro en la nube” y son más ligeros, inteligentes y baratos. Dicho "cerebro" consta de un centro de datos, una base de conocimiento, planificadores de tareas, aprendizaje profundo, procesamiento de información, modelos de entorno, soporte de comunicación, y otros.
Respecto a los laboratorios remotos de robótica, estos son plataformas que permiten el acceso remoto a diferentes tipos de robots a través de Internet, con la finalidad de que los usuarios puedan realizar experimentos, ya sea científicos, educativos, etc., a través de una interfaz remota [15].

Existen diversas arquitecturas empleadas para el montaje de laboratorios remotos de robótica. La arquitectura más simple emplea una computadora (PC de escritorio) a la cual se conecta el robot bajo prueba, y al mismo tiempo ejecuta un servidor que permite el acceso remoto a los usuarios. La ventaja de esta arquitectura es su bajo costo, siendo su principal desventaja la capacidad limitada respecto a cantidad de usuarios, cantidad de información que puede almacenar, robustez, confiabilidad y disponibilidad. Por otro lado, algunos laboratorios remotos se montan empleando servidores compuestos por un número grande de computadoras, lo que les otorga gran capacidad, pero implican mayores costos.

Los principales componentes de Robótica en el Cloud son los siguientes:
• Un repositorio global de imágenes, mapas, datos y sistema expertos.

• Computación paralela, provista bajo demanda mediante modelización estadística basada en muestreo y planificación de movimiento, planificación de tareas, colaboración de robots, planificación y coordinación de sistemas.
• Robots que comparten sus resultados, trayectorias, políticas de control dinámico y soporte de aprendizaje del robot.
• Personas que contribuyen mediante código abierto, datos, experimentos, y hardware abierto.
• Tecnologías de interacción robot-persona como bases de conocimiento semántica.

El paradigma de Robótica en el Cloud tiene distintas ventajas para robótica móvil e industrial, sin embargo, presenta las siguientes limitaciones:
• El control de movimiento de robots basados fuertemente en sensores y retroalimentación para su funcionamiento.
• Las aplicaciones basadas en el Cloud pueden ser lentas debido a respuestas de alta latencia o conexión de red, por lo que no se recomienda su uso en aplicaciones críticas de tiempo real que requieran procesamiento a bordo.

Respecto a los laboratorios remotos, en los últimos años se han propuesto la implementación de laboratorios empleando servicios de Cloud Computing [2,3]. La implementación de un laboratorio remoto mediante Cloud Computing permite aprovechar las ventajas de esta tecnología, que principalmente son: alta disponibilidad, elevada robustez, elevado poder de procesamiento y almacenamiento de datos, elasticidad y escalabilidad. Sin embargo, los laboratorios remotos basados en Cloud construidos hasta la fecha están dirigidos a experimentos relacionados

con computadoras, en la cual el sistema bajo prueba consiste en una máquina virtual o conjunto de máquinas virtuales provistas por el proveedor de servicios de Cloud Computing. En estos laboratorios remotos, los módulos de gestión del laboratorio remoto y el equipo bajo prueba están completamente implementados en la infraestructura de Cloud Computing. Solo la interfaz web se ejecuta fuera del Cloud. Este modelo no es aplicable a laboratorios donde el equipo bajo prueba debe estar fuera del Cloud, o deba estar parcialmente implementado dentro y fuera del Cloud, siendo por lo tanto aplicable a un grupo reducido de laboratorios.

En este trabajo final se desarrolla un laboratorio integral de Robótica en el Cloud, compuesto por robots móviles. Esto permitiría capacitar a los alumnos de la Facultad en el uso de robots de distinto tipo. Gracias al laboratorio, se podrán implementar distintos algoritmos de control de robots, búsqueda y rescate, generación de trayectorias, etc. Este trabajo final es parte de un proyecto que incluye robots adicionales. En el proyecto global también se trabajarán con Clouds públicos como el de Amazon y privados como el que actualmente posee la Facultad de Ingeniería de la Universidad de Mendoza.

Se puede desarrollar un laboratorio de Robótica en el Cloud dedicado a estudio y desarrollo de Aplicaciones Industriales y Móviles, empleando tecnologías de bajo costo y disponibles en el mercado local.

Objetivo General:

Desarrollar un conjunto de robots móviles para un proyecto de laboratorio que permita implementar algoritmos y explorar nuevas alternativas para realizar el control, seguimiento de trayectorias, generación de mapas y tareas colaborativas entre robots móviles.

Objetivos Específicos:

• Estudiar los requerimientos necesarios para desarrollar e implementar una plataforma de robot móvil que pueda utilizarse en el laboratorio.

• Diseñar e implementar al menos 2 robots móviles.

• Realizar la comunicación de los robots mediante algún protocolo especifico para dicha tarea

• Coordinar el funcionamiento de dichos robots mediante tecnologías de Cloud Computing.

• Implementar primero en simulador y luego en los robots algoritmos de seguimiento de trayectorias y generación de mapas.

2 MARCO TEÓRICO

Robótica en el Cloud puede definirse como “cualquier robot cuyo código y/o datos dependen de soporte provistos por Cloud Computing para llevar a cabo las funciones para las cuales está programado” [4]. Esto implica que parte de sus funciones de sensado, cómputo y memoria no están incluidas en el robot. Esta definición está destinada a incluir en un futuro distintos sistemas robóticos que incluyan tele-operación mediante redes o interconexión con otros robots como UAVs (Unmaned Aerial Vehicles) [5], robots de almacenes [6], líneas de ensamblado y plantas de procesamiento. Debido a las latencias de redes, la calidad de servicio variable y el tiempo fuera de funcionamiento, el paradigma de Robótica en el Cloud incluye generalmente algunas capacidades para realizar procesamiento local, ofrecer respuestas de baja latencia y permitir el funcionamiento de los robots en caso de que las redes dejen de funcionar.

Los robots pueden cumplir en distintos grados con la definición planteada de Robótica en el Cloud, es decir no necesariamente tienen que cumplir en un 100% con los contenidos de la definición de Robótica en el Cloud. A modo de ejemplo puede citarse el coche autónomo desarrollado por Google. Este vehículo indexa desde el Cloud mapas e imágenes recolectadas y actualizadas por satélites, Google Streetview, y fuentes de crowdsourcing para facilitar localizaciones precisas. Otro ejemplo es el robot de movimiento de palets de Kiva Systems, el cual es utilizado para logística en almacenes. Estos robots se comunican de forma inalámbrica con un servidor central local para coordinar el enrutamiento y compartir actualizaciones sobre los cambios detectados en el entorno.

En 2010, James Kuffner acuñó el término "Robótica en el Cloud" y describió una serie de beneficios potenciales [7]. Un artículo en IEEE Spectrum siguió rápidamente [8] y Steve Cousins resumió el concepto como "Ningún robot es una isla." A continuación, se detallan algunos trabajos sobre Robótica en el Cloud

En 2009, se anunció el proyecto RoboEarth. Este proyecto preveía "una World Wide Web para robots, es decir una red gigante y un repositorio de bases de datos donde los robots puedan compartir información y aprender unos de otros acerca de su comportamiento y entorno" [9] El equipo de investigación de RoboEarth desarrolló una serie de arquitecturas de sistemas para robótica de servicios [10], desarrollo de redes en el Cloud [11] y recursos informáticos para generar modelos 3D de entornos, reconocimiento de voz, y reconocimiento facial.

Respecto a la Industria, en 2011, surgió en Alemania el término "Industria 4.0", el cual predice una cuarta revolución industrial que utilizará el Cloud y las redes para seguir a la primera (mecanización de la producción utilizando agua y vapor), segunda (producción en masa con energía eléctrica) y tercera (uso de la electrónica para automatizar la producción) revolución industrial [12]. Bekris et al. [13] proponen una arquitectura para planificar trayectorias de manipuladores robots de manera eficiente. En este caso, el cálculo requerido para planificar la trayectoria de los manipuladores se divide entre el robot y la nube.

Otro ejemplo de transporte puede verse en [14]. En este trabajo Hunter et al. presentan una serie de algoritmos para un sistema de transporte basado en la nube denominado Mobile Millennium, el cual utiliza GPS de

teléfonos celulares para recopilar información de tráfico, procesarla y distribuirla. Además, Mobile Millenium permite recopilar y compartir datos sobre niveles de ruido y calidad del aire.

Respecto a aprendizaje colectivo de robots, puede afirmarse que Cloud Computing facilita el intercambio de datos para el aprendizaje de robots. Esto es así debido a que Cloud Computing permite la recopilación de datos de resultados de gran variedad de experimentos e interacciones con distinto tipo de entornos. Por ejemplo, los pueden compartir las condiciones iniciales y esperadas, las políticas de control, las trayectorias, sus datos sobre rendimiento y otros ítems.

En [15], se propone un framework denominado "Lightning", el cual brinda herramientas para aprendizaje colectivo de robots. Lighting permite acceder a trayectorias generadas por gran cantidad de robots aplicados a distintas tareas y mediante Cloud Computing realizar la planificación paralela y el ajuste de trayectoria de robots que desarrollen nuevas tareas. Estos resultados obtenidos mediante Lighting pueden ser expandidos a redes globales para facilitar la planificación de rutas compartidas, incluyendo el ruteo de tráfico.

Otro ejemplo es el proyecto MyRobots [16] de RobotShop. Este proyecto propone una "red social" para robots: "De la misma manera que los seres humanos se benefician de socializar, colaborar y compartir, los robots también pueden beneficiarse de esas interacciones compartiendo la información de sus sensores y brindando la propia perspectiva del robot sobre su estado actual ".

Como se ha visto en la presente sección, actualmente el tema de Robótica en el Cloud es de interés en la comunicad científica. En el estado del arte se han analizado algunas propuestas relacionadas a las principales aplicaciones de Robótica en el Cloud. Estas aplicaciones son de Transporte, Logística, Manufactura y Aprendizaje Colaborativo. Si bien hay gran variedad de aplicaciones, aún se deben profundizar los desarrollos en el área de laboratorios educativos, ya que esto permitirá contar con robots industriales y móviles controlados desde el Cloud para generar nuevas competencias en los distintos alumnos de la Facultad de Ingeniería de la Universidad de Mendoza y dar lugar a nuevas investigaciones en el área que sean de interés a la comunidad científica nacional e internacional.

3 DESARROLLO DE INGENIERÍA

3.1 ESQUEMA GENERAL DEL SISTEMA

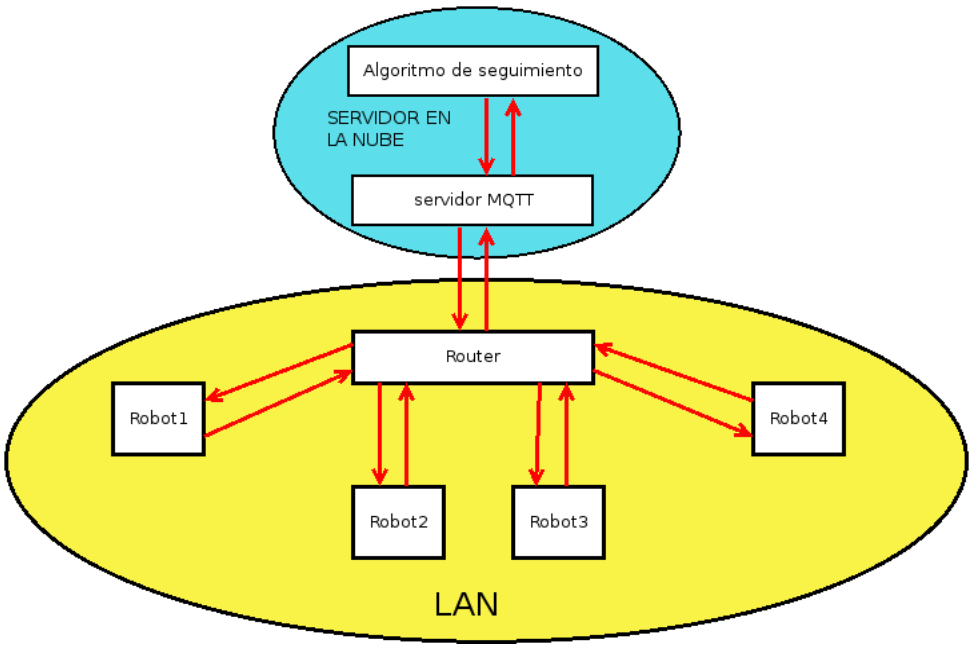


Figura 1: Esquema completo del sistema

El proyecto consiste como podemos observar en la figura 1 en un sistema formado por un servidor GNU/Linux alojado en cloud y una red LAN. En el cloud se montara un servidor de comunicación de un protocolo específico para esta labor, además del algoritmo de control de los robots. La conexión a al cloud y configuración de dichos servicios será por SSH.

La red LAN tiene una topología del tipo estrella que está compuesta por un router WAP (Wireless Access Point) y una N cantidad de robots. Todos estos elementos tienen su IP fija para un debug más sencillo, se recomienda que el ultimo byte de la dirección IPV4 haga referencia al id del robot.

Para las pruebas en campo de este laboratorio de robótica se trabajó todo de forma local. El servidor de comunicación y el algoritmo estaban en la LAN. Más adelante, en este documento, se explicará cuáles son los cambios necesarios a aplicar en los robots para llevar el servidor desde LAN al cloud.

ESQUEMA ESPECIFICO DEL SISTEMA

Figura 2: Esquema de subsistemas

Como se observa de la figura 2 se divide al sistema completo en 3 subsistemas los cuales se detallaran en profundidad en las siguientes secciones:

-Subsistema robot: Incluye todos lo relacionado al robot: chasis, baterías, sensores, motores, elementos de protección, microcontroladores, comunicación y los softwares necesarios el funcionamiento del robot.

-Subsistema servidor: Es el responsable de la comunicación robot-algoritmo, incluye los elementos de router y servidor MQTT(Message Queuing Telemetry Transport) .

-Subsistema algoritmo: Es el encargado del algoritmo de generación de trayectoria y control de los robots. Incluye al algoritmo propiamente dicho, generación de un archivo log de la experiencia y la comunicación del algoritmo con el subsistema servidor.

Para realizar una verificación del sistema más sencilla, el usuario puede entrar al sistema sustituyendo al subsistema robot, comunicándose a través del terminal de Linux con el servidor MQTT; permitiéndole evaluar la respuesta del algoritmo. De igual forma, el usuario puede entrar sustituyendo al subsistema algoritmo, viendo en tiempo real la respuesta de los robots en el log y enviando comandos de acción a los robots a través del terminal de Linux.

3.2 SUBSISTEMA ROBOT

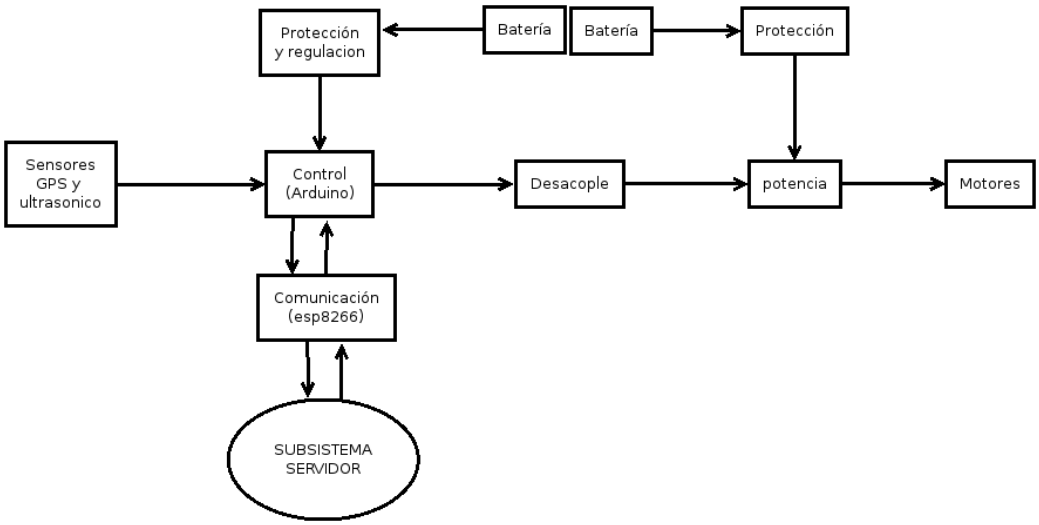


Figura 3: Esquema del subsistemas robot

3.2.1 CHASIS

Para el chasis del robot se escogió una estructura en 2 niveles formadas por 2 plataformas de acrílico unidades por 4 tornillos largos (o varillas roscadas) de 4mm, se añaden arandelas tipo grower para evitar que las tuercas se aflojen por el uso del robot sobre terreno irregular.

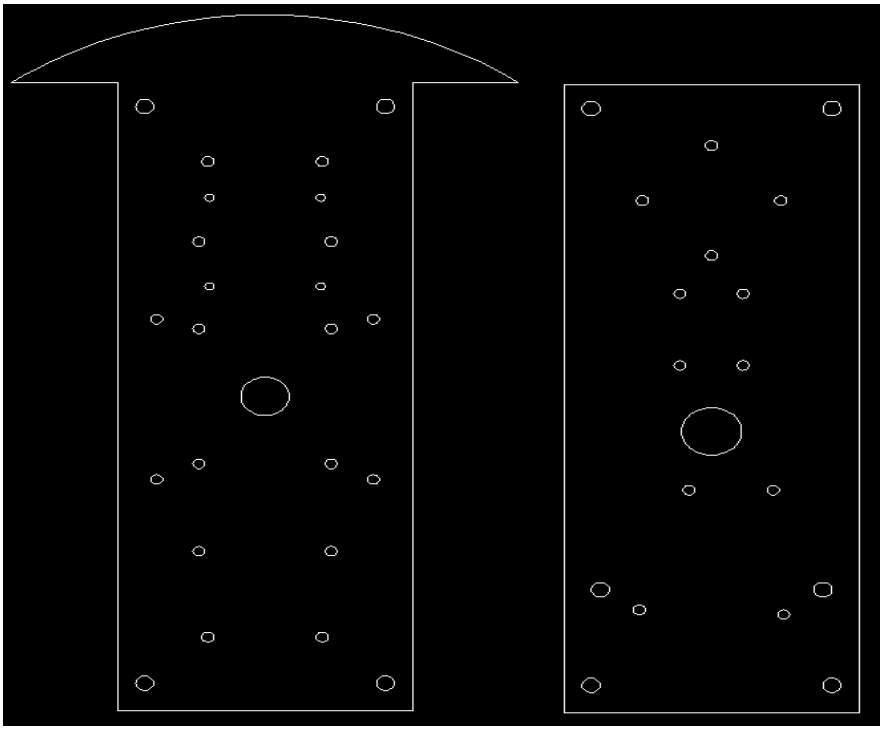


Figura 4: Chasis acrílico

Las dimensiones de dichas piezas las cuales podemos ver en la figura 4 de acrílico son:

Pieza inferior: 290mm x 98mm. Pieza superior: 262,5mm x98mm. Ambas de un grosor de 3mm

También fueron necesarias un total de 7 piezas diseñas de impresión 3D para la sujeción de componentes a las piezas de acrílico. 4 de ellas son soportes individuales para los motores (77cm x 24mm x 25mm), 2 son soportes para el sensor ultrasónico (74mm x 20mm x 10mm y 98mm x 23mm x 24mm) y una última pieza para la antena del GPS (55mm x 37mm x 13mm), dichas piezas las podemos encontrar en la figura 5.

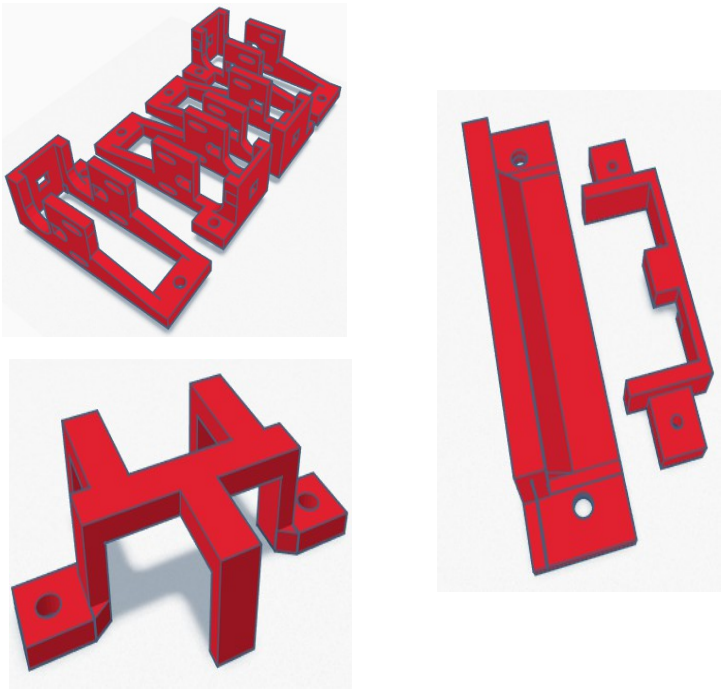


Figura 5: Piezas 3D para sujeción

Para el diseño de las piezas se utilizó el programa Tinkercad en su versión web. Para el diseño de soportes de los motores y del sensor se utilizó como base un diseño del usuario Ashing de la plataforma Thingiverse [17].

La disposición física de los componentes es la siguiente: debajo de la placa inferior acrílica se colocan los 4 motores con sus soportes, entre medio de estos se colocan las baterías, siendo éstas los elementos más pesados que componen al robot, asegurando una buena tracción de las ruedas y un centro de gravedad bajo; logrando así la mejor estabilidad para el robot. Por el lado de arriba de esta plataforma se colocará el puente H y la placa PCB de protección, regulación y desacople.

En la plataforma superior se ubicarán el Arduino (con el puerto USB mirando hacia la parte trasera del robot para facilitar la conexión a la PC) con una placa PCB del tipo Shield, ESP8266 [18], GPS con su antena y el sensor ultrasónico mirando hacia el frente del robot.
El robot tiene su parte de potencia aislada eléctricamente de la parte de control, adquisición de datos y comunicación. De esta forma todos los elementos de potencia y de protección están situadas en la plataforma inferior, y los demás módulos de la otra etapa en la parte superior; haciendo más sencilla la correspondencia de los elementos a cada circuito eléctrico según su disposición física en el robot.

La disposición es tal que permite una fácil accesibilidad desde el exterior a los elementos de manipulación frecuente, como son las llaves de encendido de ambos circuitos eléctricos, ubicados en la parte superior trasera del robot (a la altura del puerto USB del Arduino), y los fusibles de protección, los cuales se encuentran en la parte inferior trasera del robot ubicados debajo de cada llave de encendido.

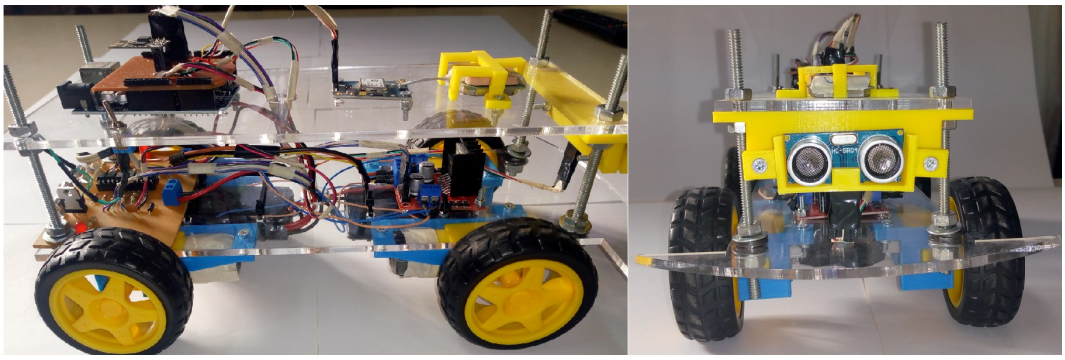


Figura 6: Robot completo

3.2.2 BATERÍA

La alimentación del robot se realiza por medio 4 baterías de ion de litio modelo 18650 de 2400mAh como las de la figura 7. Se utilizarán 2 baterías para la parte de potencia del robot y 2 para la parte de control/comunicación. Las baterías se conectarán en serie para alcanzar una tensión de 7,4V. Para el caso de la etapa de potencia dicha tensión es suministrada directamente a los motores a través del puente H. Para la etapa de control es necesario disminuir la tensión a 5V. La carga de las baterías será por separado mediante un cargador basado en el integrado TP4056 [19].

Figura 7: Baterías ion de litio modelo 18650

Se evaluó como alternativa la utilización de baterías de Li-Po (litio-polímero) del tipo S2 (baterías de 2 celdas con una tensión de 7,4V), estas baterías presentan prestaciones similares a las de ion de litio y son más livianas. La contraprestación es que son más complicadas de conseguir en el mercado local, su costo es mayor y requieren de un cargador especial de carga balanceada el cual tiene un gran costo. Por este motivo se prefirió las baterías de ion de litio.

Ambas baterías son ideales para esta aplicación ya que son recargables, brindan una gran corriente y soportan picos de corrientes bastantes mayores a la corriente nominal de ésta. Ambos tipos son de dimensiones pequeñas y livianas. En cuanto al peso las baterías de litio-polímero tienen la ventaja de ser las mas livianas, por este motivo son las utilizadas en drones donde el peso es el punto clave. Como en la aplicación del robot a desarrollar el peso no es algo determinante no afecta a la elección. Pero el punto de mayor interés se encuentra en las curvas de tensión vs descarga, podemos observar que esta tecnológicas basadas en litio presentan una tensión casi constante entre los valores de carga 85% y 15%, ofreciendo torques y velocidades constantes en los motores durante la mayor parte de las pruebas.

Podemos corroborarlo esto con la siguiente curva de la figura 8 [20] de una batería del tipo ion de litio de características similares a las utilizadas en el proyecto.

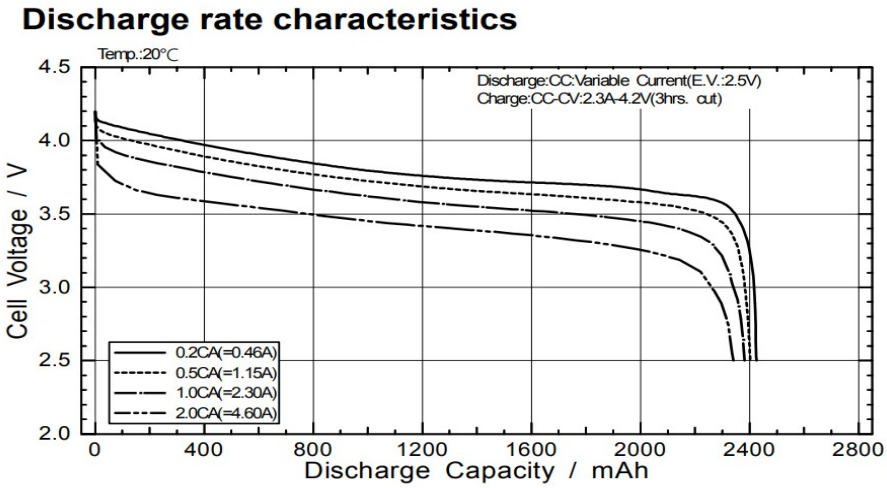


Figura 8: esquema de subsistemas

3.2.3 TOPOLOGIA AISLADA

Para las pruebas iniciales se diseñó un único circuito eléctrico, todo alimentado por única batería y GND común a todos los componentes. Pero durante estas pruebas se descubrió un problema, al dar orden de movimiento a los motores, estos generan un gran consumo y producen ruido en la línea de GND; lo cual produce que el microcontrolador ESP8266 se bloqueara, haciendo imposible la comunicación con el robot hasta apagar el módulo y volverlo a encender. Esto se debe a que dicho microcontrolador trabaja a una tensión de 3,3V (a diferencia del Arduino que funciona con 5V) por lo tanto este es el componente más sensible a al ruido eléctrico y variaciones de tensión.

Por este motivo fue necesario incrementar la robustez del sistema. El método más sencillo para lograrlo es implementar una topología aislada, algo muy común en el área industrial, donde se trabaja con grandes potencias, pero novedoso en robots de este tamaño. Esto causó tener que desdoblar toda la electrónica del robot en 2 partes. La parte de potencia: encargada del funcionamiento de los motores, y la parte de control, sensores y comunicación del robot. Para poder realizar estos cambios fue necesario añadir al robot una etapa de desacople, la cual se encarga de la comunicación de datos desde el microcontrolador Arduino hasta el modulo de potencia a través de optoacopladores PC417. Los datos a comunicar son: la velocidad de los motores y sentido de giro de estos (ENA;ENB;IN1;IN2;IN3;IN4). El orden de conexión para dicha comunicación es la correspondiente a la tabla 1.

Tabla 1: Conexión optoacopaldores - puente H

Pin del Arduino	Numero de optoacoplador	Pin del puente H
10	1	ENA
8	2	IN1
9	3	IN2
12	4	IN3
13	5	IN4
11	6	ENB

El hecho de tener que aislar las 2 partes del robot trajeron consecuencias tanto positivas como negativas. El mayor inconveniente es la necesidad de utilizar 2 baterías, lo cual significa mayor costo, peso y volumen del robot. Pero las mejoras logradas en autonomía y robustez del sistema permiten incluso cambiar completamente las etapas de potencia y motores por unas de dimensiones y potencias mucho mayores sin tener la necesidad de modificar las etapas de sensores, control y comunicación.

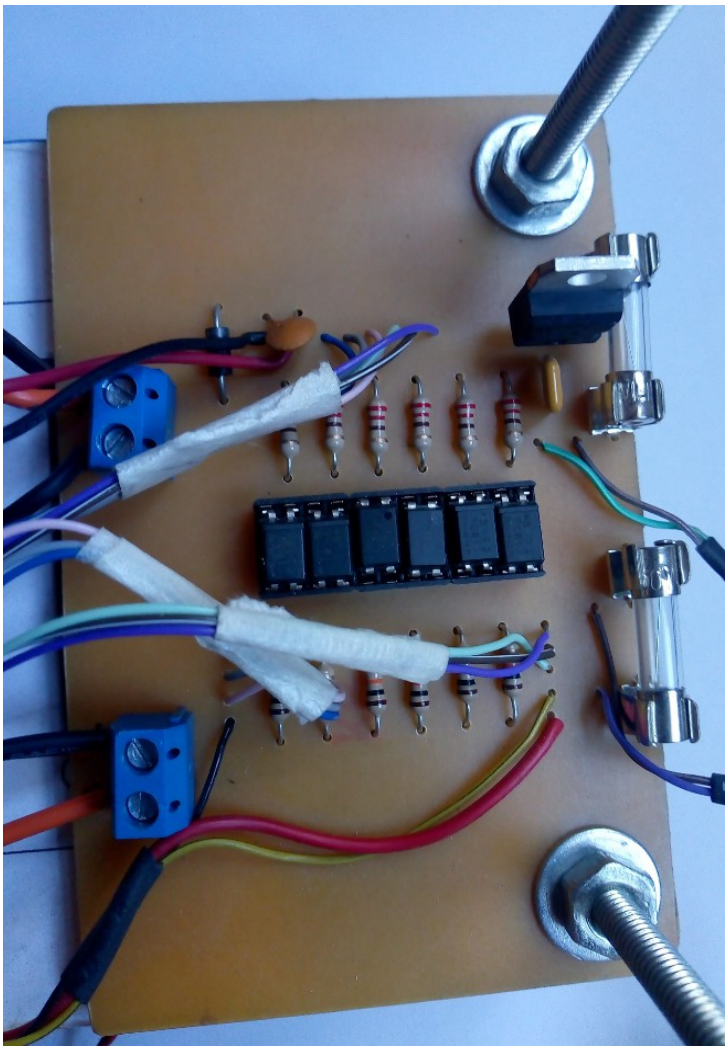


Figura 9: Placa de optoacopaldores y protección.

3.2.4 ETAPA DE POTENCIA

Esta etapa está compuesta por una batería compuesta de 2 celdas de ion de litio, la etapa de optoacopladores, elementos de protección, puente H y motores.

Para la parte motriz del robot se utilizaron 4 motores DC, de 140RPM con una caja de reducción de 1:48. Los consumos de los motores fueron medidos mediante pruebas de laboratorio y se obtuvieron los siguientes resultados mostrados en la tabla 2:

Tabla 2: consumo de motor

Carga del motor	Consumo en mA
Vacío	150
Nominal	600
Rotor bloqueado	1500

Estos motores (figura 10) abundan en el mercado local a muy bajo costo, haciendo que sea sencillo conseguir repuestos. La desventaja es que no presentan una alta performance, por lo cual quedan descartados para usos de altas exigencias. Para esta aplicación no se requiere esfuerzo de los motores, por lo tanto, es la mejor opción.

Las ruedas que se utilizarán son las proporcionadas por el fabricante de los motores, con el encastre específico para el rotor del motor. Estas ruedas tienen un diámetro (66mm) adecuado para las dimensiones del robot y la goma de las cubiertas ofrecen un agarre óptimo para la prestación requerida.

Figura 10: Motor y rueda

Si se requiere mayor agarre por parte de las ruedas, se recomienda reemplazar la goma por cubiertas de silicona como las de la figura 11. Estas cubiertas pueden ser adquiridas en tiendas especializas fuera del país o fabricadas siguiendo un proceso [21] muy sencillo a base de sellador de silicona y almidón de maíz.

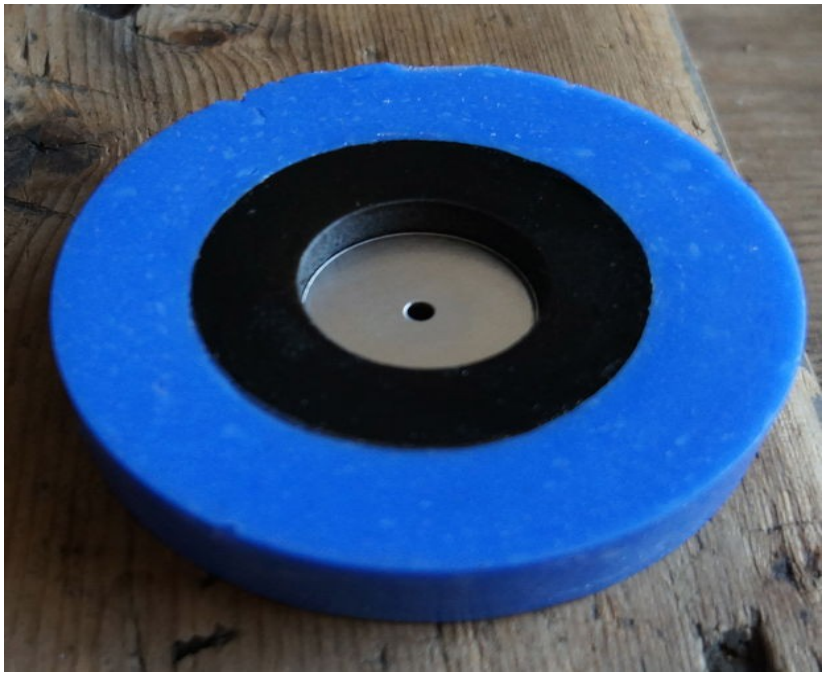


Figura 11: Ruedas de silicona

La disposición de los motores es la siguiente: uno en cada esquina del motor con las cajas de reducción mirando hacia afuera, para dar el mejor punto de apoyo a las ruedas. Sus conexiones serán, mirando el robot de su parte trasera, los 2 motores derechos se conectarán a la bornera del módulo de potencia correspondiente al lado B. Así mismo, los 2 motores izquierdos se conectarán a la bornera correspondiente al lado A.

Para el módulo de potencia se utilizó un Shield (figura 13) muy popular en el mercado, el cual se basa en el integrado L298N. Este integrado cuenta con 2 puentes H que poseen una corriente máxima de 2A cada uno, lo cual lo convierte en el integrado ideal para la alimentación de nuestros motores. Además, el módulo cuenta con diodos volantes (freewheeling) a la salida del puente y un regulador de 5V; el cual será utilizado para la

alimentación de la etapa de desacople. Dicho módulo cuenta con 6 pines para el control de los motores, los cuales se detallan en la tabla 3.

Tabla 3: Modulo de potencia

Pin del módulo de potencia	Función	Corresponde a
ENABLE A (ENA)	Velocidad del motor	Puente H (A), Motores izquierdos
INPUT 1 (IN1)	Sentido de giro	Puente H (A), Motores izquierdos
INPUT 2 (IN2)	Sentido de giro	Puente H (A), Motores izquierdos
INPUT 3 (IN3)	Sentido de giro	Puente H (B), Motores derechos
INPUT 4 (IN4)	Sentido de giro	Puente H (B), Motores derechos
ENANBLE B (ENB)	Velocidad del motor	Puente H (B), Motores derechos

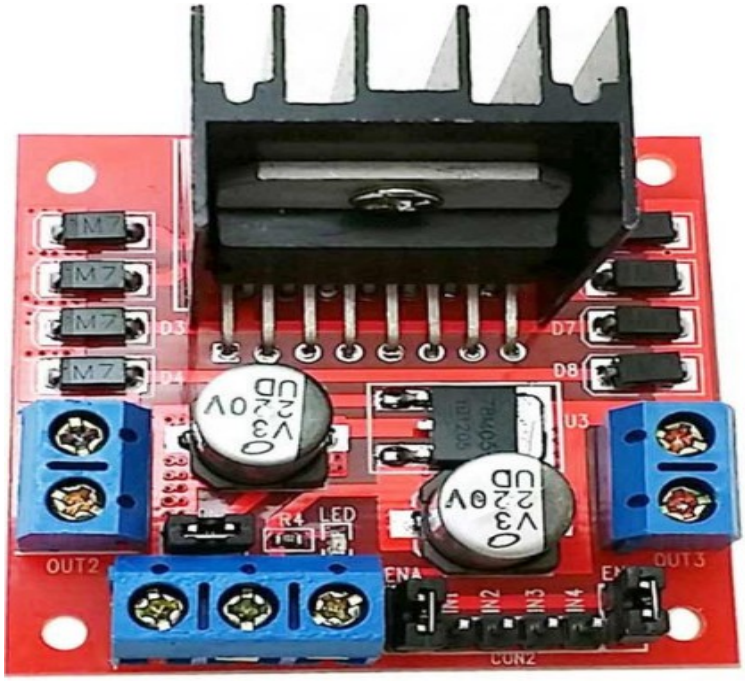


Figura 13: Modulo de potencia basado en L298

Los últimos elementos que restan mencionar de esta etapa son un switch on/off de tipo palanca y un fusible, el switch está conectado en serie con la batería con el fin de realizar el encendido y apagado de la etapa de potencia del robot. Para el fusible se optó por uno de 4A de formato cilíndrico corto. La elección de la corriente de ruptura fue en base a la corriente máxima del módulo de potencia y del consumo de los motores; si tenemos en cuenta que cada motor tiene un consumo nominal de 600mA, el fusible no podría ser menor a 2400mA. Se eligió un fusible sobredimensionado para que soporte los picos de corriente de los motores al momento del arranque.

3.2.5 ETAPA DE CONTROL/COMUNICACIÓN

En esta etapa se cuentan los demás módulos del robot, o sea, los módulos de sensores, procesamiento y comunicación, además cuenta con la batería que alimenta a esta etapa, junto con su elementos de protección y regulación.

3.2.5.1 SENSORES

El robot cuenta con 2 sensores: uno de posición (GPS), ubicado en la parte superior delantera del robot, y el otro de obstáculos (ultrasonido) ubicado en la parte superior frontal mirando hacia el frente del robot.

ULTRASÓNICO

La alternativa a este sensor sería un detector de obstáculo infrarrojo como lo son los sensores FC-51, sharp o vl53l0x. Pero debido a la necesidad de realizar las pruebas en exteriores para garantizar un mejor funcionamiento del GPS, es que optamos por sensores sonicos en vez de ópticos, ya que estos podrían verse afectados por la luz solar.

Usaremos el sensor HC-SR04, este elemento es un detector activo, lo cual significa que posee un emisor y un receptor, ambos sintonizados a 40Khz. Este modelo es muy común en el mercado y se puede adquirir por un costo muy bajo, el alcance y resolución del sensor son modificables desde el código en la librería del sensor, este podría llegar a medir una distancia de hasta 4 metros a costa de una baja resolución. Para esta prueba hemos optado por lo contrario, priorizar la precisión, por lo cual el alcance del sensor tuvo que ser reducido a 50 cm.

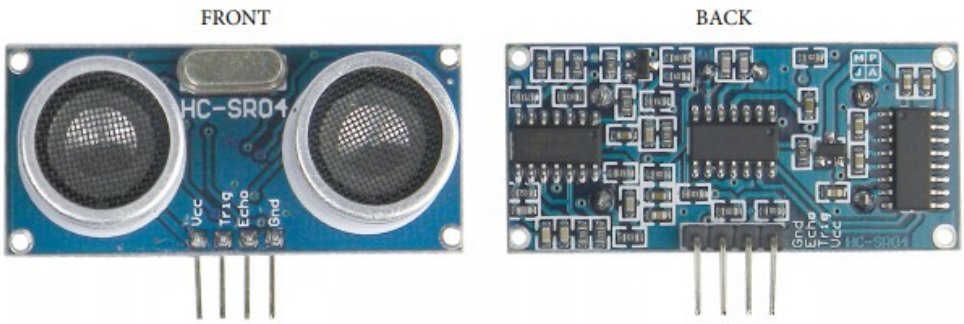


Figura 14: Sensor ultrasonico HC-SR04

Las características obtenidas de la hoja del fabricante[22] de este sensor son:

-Dimensiones físicas: 43mm x 20mm x17mm
-Tensión de alimentación: 5V
-Consumo: 15mA
-Frecuencia de operación: 40Khz

Realizamos algunas pruebas de laboratorio para la observación de las señales emitidas y recibidas por el sensor para corroborar el alcance y correcto funcionamiento de este, los resultados los observamos en la figura 15.

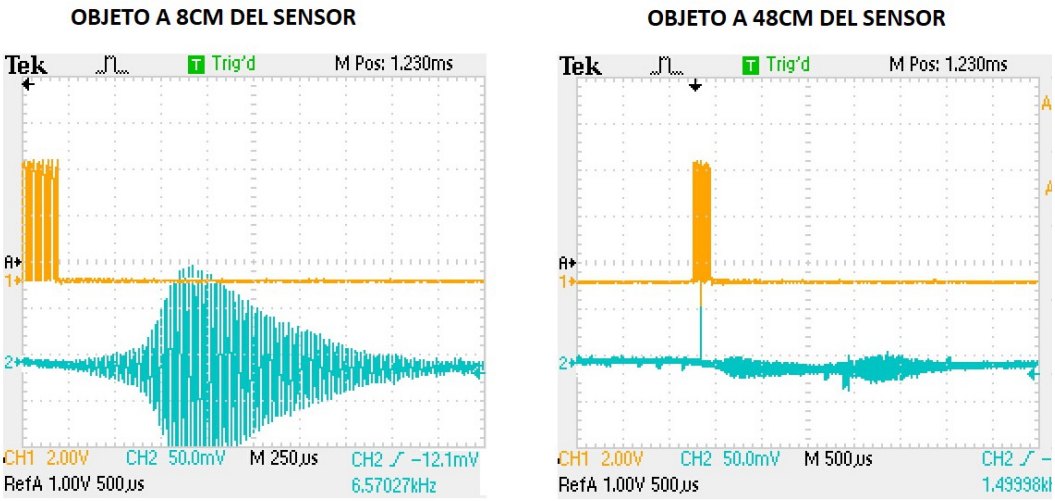


Figura 15: Señales sensor ultrasonico HC-SR04

Con estas prácticas podemos observar la atenuación de la señal de recepción comparando las amplitudes del eco recibido entre las 2 pruebas de objeto cercano y objeto lejano.

GPS

Para el posicionamiento se utilizó un modulo Gps Gy-neo6mv2 basado en el integrado NEO-6M [23] de la empresa u-blox. Este módulo se puede conseguir en tiendas del país, es el mas económico que existe y se puede adquirir por un valor aproximado de $15 USD (incluye antena), este modelo solo posee GPS, no puede tomar señal de los satélites correspondientes a GLONAS (sistemas de posicionamiento ruso) ni beidou (sistemas de posicionamiento chino). De la hoja de datos del NEO-6M podemos extraer los siguientes datos técnicos:

-Tensión de alimentación: 3,3V (se puede alimentar el modulo a 5V ya que
-posee un regulador incorporado)
-Corriente máxima: 67mA
-Precisión de posición horizontal: 2,5m
-Dimensiones: 16 mm x 12.2 mm x 2.4 mm
-Canales máximos: 50
-Comunicación: UART, 9600 baudio

Para este proyecto el dato que mas nos importa es la precisión del GPS, para corroborar el dato obtenido de la hoja de datos [23] (2,5 metros) realizaremos una prueba del módulo, para la cual utilizaremos el software del fabricante, el cual puede descargarse gratuitamente del sitio del fabricante (https://www.u-blox.com/en/product/u-center).
Para la conexión del modulo a la computadora es necesario un conversor USB-FTDI, en caso de no disponer de uno podemos reemplazarlo con un Arduino.

En los graficos se presentan los resultados obtenidos durante la prueba, se tomaron muestras durante 30 minutos, la cantidad máxima de satélites enlazados fueron 7 y los podes individualizar en las figuras 16 y 17.

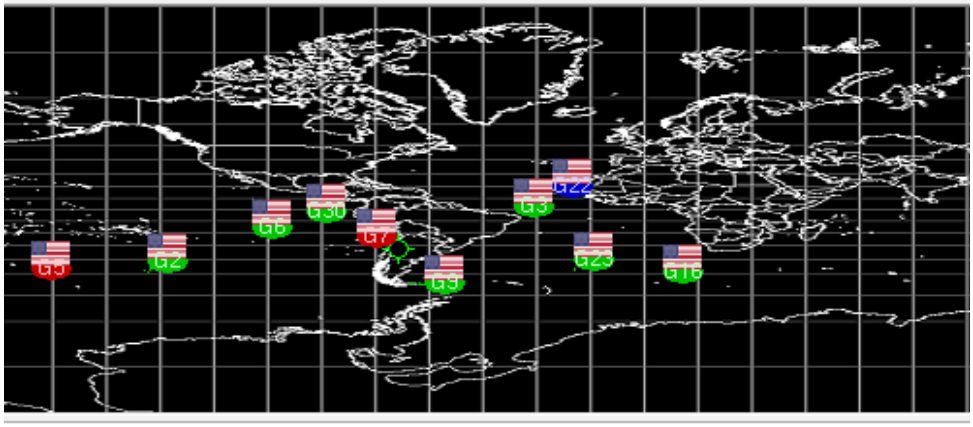


Figura 16: Posición de los satélites

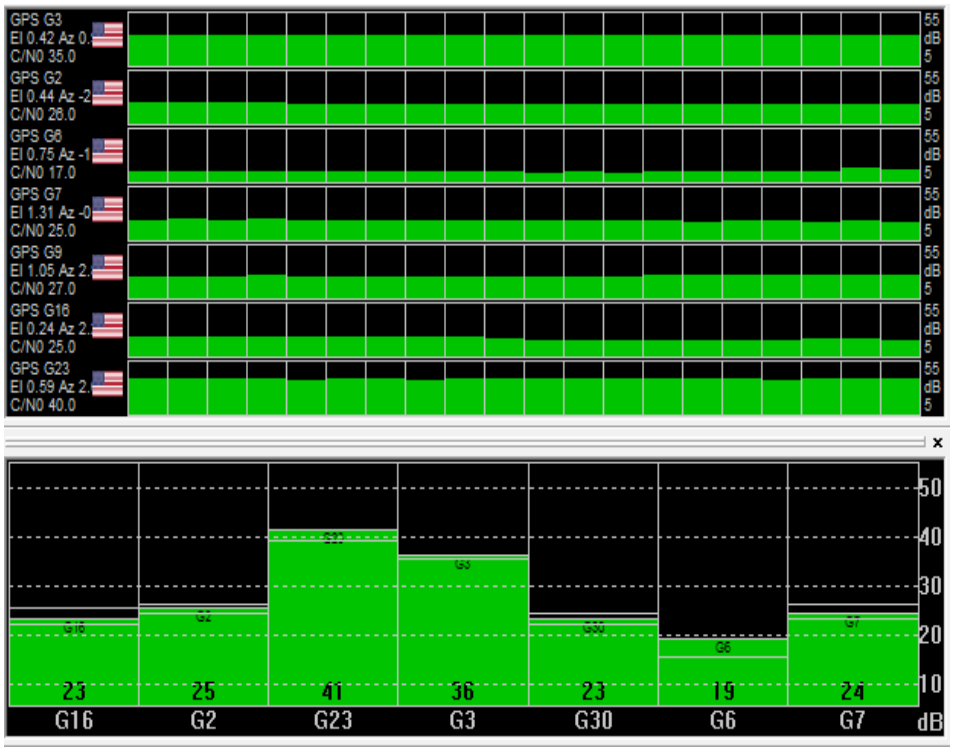


Figura 17: Potencia de las señales de los satélites

Para que el GPS comience a transmitir valores de posición requiere de un mínimo de 4 satélites, cuando alcanza este valor comienza a transmitir un stream por puerto serial. Visualmente podemos saber que ya esta en condiciones de operar porque hace parpadear un led en su placa, el módulo transmite el stream cada 1 segundo.

Con las primeras muestras obtenidas el error máximo fue de **35** metros, con una rápida corrección a un radio de error absoluto de **10** metros. Al finalizar la prueba pudimos comprobar que el radio de error absoluto de **2,5** metros contenía a la mayor cantidad de muestras. Ocasionalmente las muestras salían de esta área; esto ocurría con el cambio de la constelación, es decir, cuando entraba o salia un satélite.

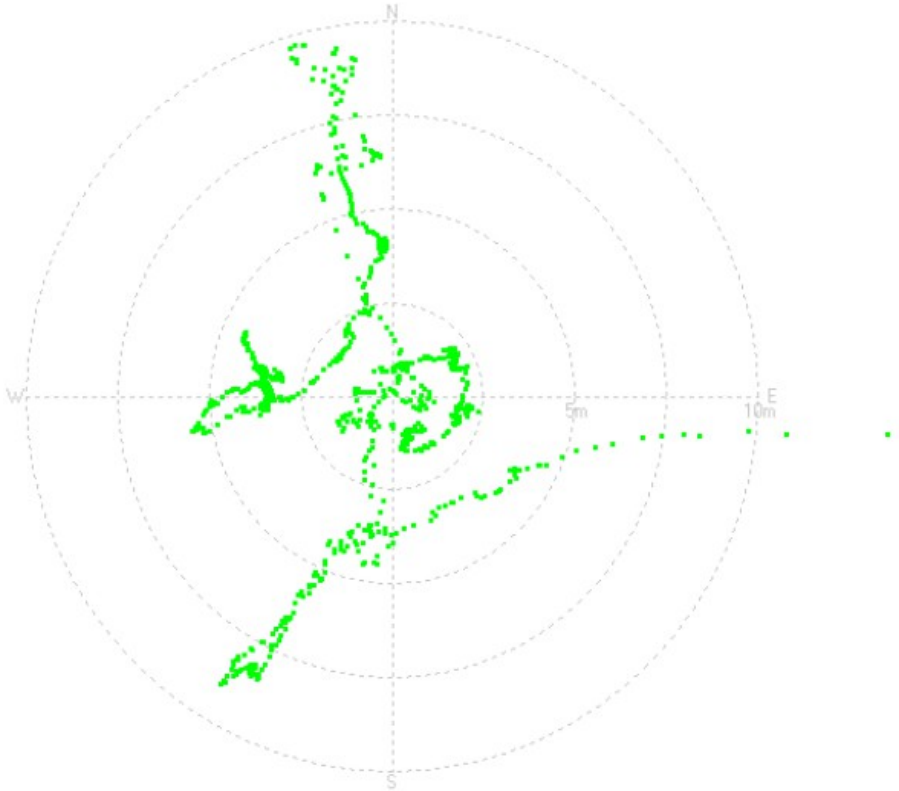


Figura 18: Muestras del GPS con radio 10 metros

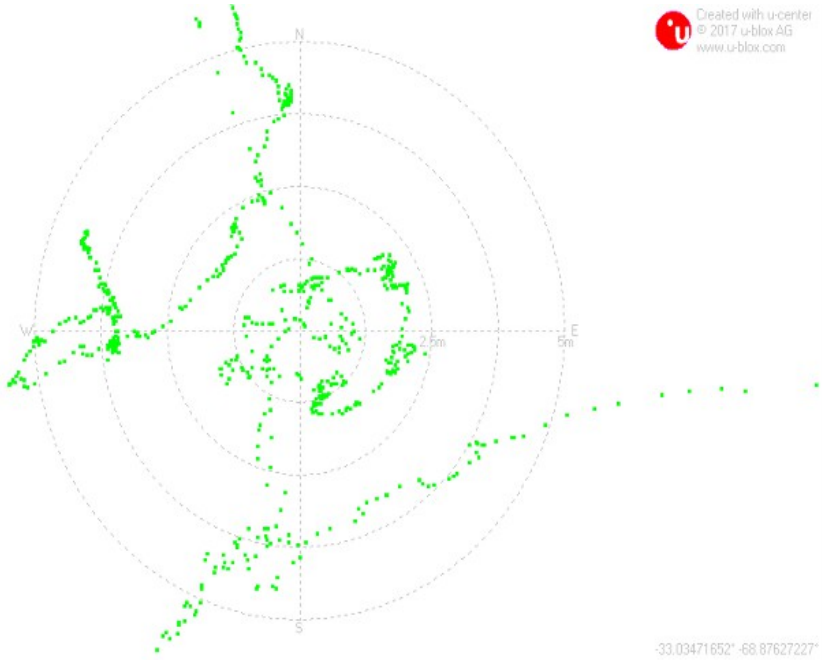


Figura 19: Muestras del GPS con radio 5 metros

También podemos obtener los datos de la altitud a pesar de no ser prioridad para esta etapa del proyecto, pero pueden ser de gran importancia para desarrollos futuros. Como podemos observar, el error de altitud es mucho mayor al error horizontal. Este es un error absoluto que ronda los 10 metros. El presente trabajo no demanda el estudio con mayor profundidad de este parámetro.

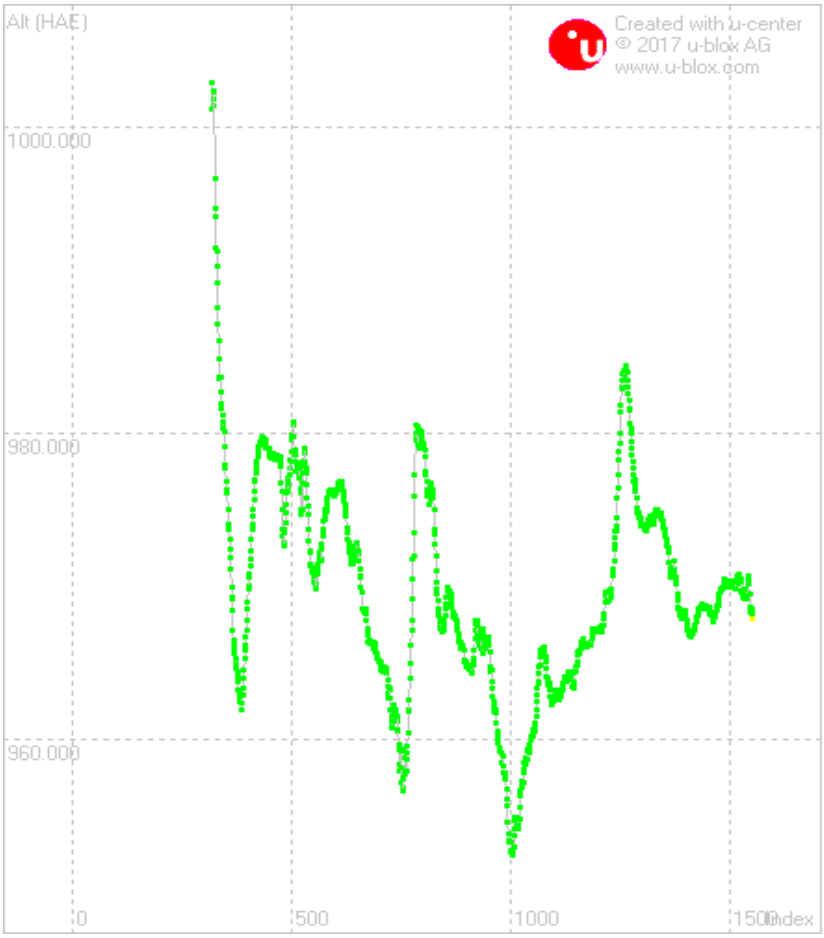


Figura 19: Muestras de altitud

Para utilizar este modulo con nuestro microcontrolador Arduino fue necesario incorporar al IDE la librería TinyGPS++ provista por el usuario mikalhart de la plataforma github[24]

MICROCONTROLADORES

El robot posee 2 módulos de microcontroladores, uno es un Arduino UNO, basado en el integrado atmega328p, y el otro es un ESP01 basado en el integrado esp8266ex. El Arduino se encarga del control de motores y sensores y el ESP01 de la comunicación con el servidor MQTT. Estos 2 microcontroladores están comunicados mediante un canal serie a 9600 baudios. Ninguno de los 2 módulos posee una inteligencia para decidir sobre el accionar futuro del robot; esa labor esta centralizada en el algoritmo alojado en el Cloud; Estos microcontroladores simplemente reciben indicaciones desde el Cloud y las ejecutan.

3.2.5.2 ARDUINO UNO

Este sistema embebido muy popular en el mercado puede ser adquirido en tiendas locales a costos muy accesibles, también debido a sus políticas de Open Source y Open Hardware existen muchos componentes compatibles a este; convirtiendolo el elemento ideal para el control de nuestro robot. La alimentación del robot será por el pin de 5V por medio de un regulador en una placa externa. Esto es debido a que se prefirió usar un regulador externo en vez del regulador que trae incorporado el arduino. Para la alimentación del modulo ESP01 se utilizó el regulador de Arduino de 3,3V.

Figura 20: Arduino uno R3

Veremos algunas características del Arduino:

-Tensión de operación: 5V

-consumo de corriente: 50mA

-GPIO: 14

-Pines PWM: 6

-Pines ADC: 6

-Memoria flash: 32Kb

-Velocidad del Clock: 16Mhz

-Dimensiones: 71 mm x 53 mm x14 mm

Para la conexión de los diferentes módulos al Arudino se optó por el diseño de una placa de formato Shield (figura 21) para hacer de soporte físico de los cables conectores de los módulos. A este Shield se conectan el ESP01, GPS, sensor ultrasónico y placa de desacople/protección.

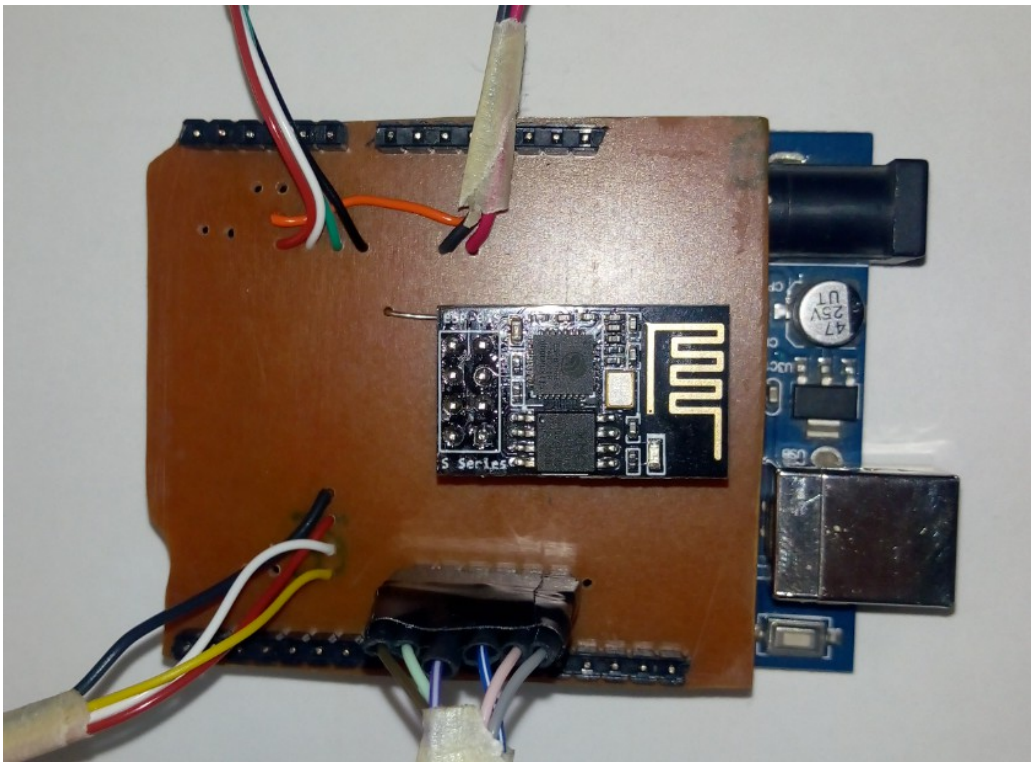


Figura 21: Shield de conexión para Arduino uno

La conexión del Arduino a través de sus pines con los demás elementos del robot se especifica en la tabla 4.

Tabla 4:Conexión del GPIO de Arduino

Pin Arudino	modulo
0	Reservado: comunicación serie a la PC
1	Reservado: comunicación serie a la PC
2	Reservado: futura aplicación de encoders
3	Reservado: futura aplicación de encoders
4	Comunicación serie al GPS (Rx)
5	Comunicación serie al GPS (Tx)
6	Comunicación serie al ESP01 (Tx)
7	Comunicación serie al ESP01 (Rx)
8	Placa de desacople: optoacoplador numero 2 (IN1)
9	Placa de desacople: optoacoplador numero 3 (IN2)
10	Placa de desacople: optoacoplador numero 1 (ENA)
11	Placa de desacople: optoacoplador numero 6 (ENB)
12	Placa de desacople: optoacoplador numero 4 (IN3)
13	Placa de desacople: optoacoplador numero 5 (IN4)
A0	Sensor ultrasonico (ECHO)
A2	Sensor ultrasonico (TRIG)
A3	Libre
A4	Libre
A5	Libre
A5	Libre
GND	GND común de la etapa control/comunicación
5V	Regulador lm7805 en placa de regulación/protección
3,3V	Pin de alimentación del ESP01

El Arduino recibe un comando a ejecutar del ESP01 por puerto serie, si el comando es de movimiento (control de motores), simplemente realiza la acción controlando los pines conectados al puente H por medio de la placa de desacople. Si el comando enviado por el ESP01 llegara a ser de solicitud de datos de sensor, Arduino solicita al sensor el dato requerido y retransmite al ESP01 el dato en crudo.
En lo que respecta al código de Arduino, fue desarrollado en el IDE propio. Este posee 2 puertos de comunicación serial virtuales generados por la librería SoftwareSerial; uno es utiliziado para la comunicación del Arduino-GPS y otro para Arduino-ESP01. El puerto serie original del Arduino (correspondiente a los pines 0 y 1) se reservó para la conexión a la PC en caso que fuese necesario el debug.

Arduino esta constantemente pendiente a una comunicación procedente del ESP01. Al comunicarse, compara el comando recibido con una lista de opciones internas. Si hay coincidencia ejecuta la acción solicitada, sino se descarta el mensaje. Lo único que requiere de una observación adicional es el caso de los datos provenientes del GPS ya que arduino no puede atender 2 puertos seriales virtuales en simultáneo; por lo tanto, cuando recibe una solicitud de datos de GPS, desatiende el puerto correspondiente al ESP01 y se queda esperando un stream proveniente del GPS. Una vez que la recibe vuelve al puerto del ESP01 y envía el dato solicitado. El GPS envía un stream constantemente cada 1 segundo, lo cual significa que el robot queda incomunicado con el servidor MQTT durante un periodo máximo de 1 segundo. Este tiempo no afecta considerablemente al funcionamiento del sistema, pero es importante tener en cuenta de que si solicitamos un dato del GPS tendremos una demora de 1 segundo como máximo para poder enviar el siguiente

comando. Si transcurrido el tiempo de 1 segundo, Arduino no recibe el stream proveniente del GPS, enviará al ESP01 el valor 99, el cual se ha asignado como mensaje de Timeout. Este valor de Timeout fue elegido ya que ningún sensor puede devolver este valor en condiciones de operaciones normales. Este error de Timeout también aplica para el caso del sensor ultrasónico.

3.2.5.3 ESP01

Este módulo es muy popular en el ambiente de IoT, dado su bajo costo y sus excelentes prestaciones para la conectividad Wi-Fi. Una alternativa a este módulo sería el nodeMCU, el cual también está basado en esp8266 y su costo es solo un poco mayor, pero no presenta ninguna ventaja para este proyecto. La única diferencia entre ESP01 y nodeMCU es que este último puede ser grabado conectando el módulo directamente por el puerto micro USB a la PC, en cambio para grabar el ESP01 se requiere de un grabador externo y una pequeña PCB el cual fue facilitado por Carlos Godoy en [25] (se detallará el proceso de grabado de ESP01 en el anexo de este documento).

Esta son las principales características del modulo ESP01:
-Wi-Fi: 2,4Ghz
-Tensión de operación: 3,3V
-Consumo de corriente: 80mA
-Puerto UART: 1
-Memoria Flash: 512Kb
-Velocidad del Clock: 80Mhz
-Dimensiones: 14 mm x 25 mm x 3 mm

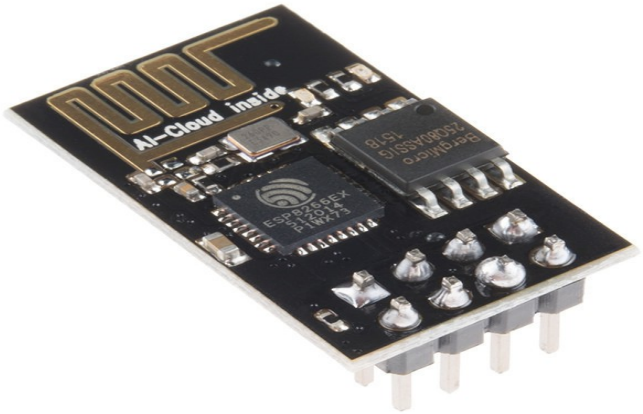


Figura 22: Modulo ESP01

Para la programación del módulo se utilizó el IDE de Arduino, pero fue necesario añadir el complemento para microcontroladores esp8266. Este complemento también incluye la librería ESP8266WiFi, la cual contiene funciones específicas del microcontrolador para Wi-Fi como por ejemplo:

-IPAddress ip(xxx,xxx,xxx,xxx); // IP estática del modulo
-IPAddress gateway(xxx,xxx,xxx,xxx);// dirección IP del gateway
-WiFi.begin(ssid, password); //nombre de la red a conectarse, password

Con estas funciones ya podemos sumar nuestro robot a la red Wi-Fi, pero para la conexión al servidor MQTT debemos añadir la librería específica para este protocolo, la cual fue compartida por el usuario knolleary en la plataforma github[26]. En la librería PubSubClient podemos encontrar las siguientes funciones:

-client.connect(xxxx);// nombre con el que se registra el robot en el servidor MQTT
-client.setServer(xxxx,xxxx); // IP donde se aloja el el servidor MQTT, puerto

-client.publish(xxx, xxx) // topic en el que desea publicar, mensaje a publicar
-client.subscribe(xxxx) // topic al que se subscribe

Se detallará el funcionamiento del protocolo MQTT en la sección de subsistema servidor.

A rasgos generales, la función de ESP01 es la siguiente: en primer momento se conecta a la red Wi-Fi LAN, luego se accede al servidor MQTT, se registra en este y se suscribe a un topic. Aguarda por instrucciones provenientes del algoritmo de control a través del servidor MQTT y cuando recibe un mensaje:
Si el mensaje requiere desplazamiento del robot, responde con la confirmación del comando en formato csv y envía por puerto serie el movimiento correspondiente al robot.
Si el mensaje requiere información de los sensores del robot, envía por puerto serie el comando de solicitud y aguarda a la respuesta del Arduino con el valor en crudo del sensor, forma un mensaje de formato csv con este valor y lo envía al servidor MQTT. Si transcurrido un tiempo el Arduino no llegara a responder, devolverá al servidor en formato csv el valor de sensor 99, el cual, como fue explicado anteriormente, corresponde a error de timeout.

La conexión de los pines del módulo ESP01 se especifica en la tabla 5

Tabla 5: conexión de GPIO de ESP01

Pin ESP01	Se conecta a:
1 (GND)	GND común de la etapa control/comunicación
2 (GPIO1)	Libre
3 (GPIO0)	Libre
4 (Rx)	Pin 7 del Arduino
5 (Tx)	Pin 6 del Arduino
6 (CH_PD)	Libre
7 (RESET)	Pin de 3,3V del Arduino
8 (VCC)	Pin de 3,3V del Arduino

Figura 23: Numeración de los pines de ESP01

3.2.5.4 COMUNICACIÓN M2M (ARDUINO-ESP01)

Para la comunicación entre los módulos se elige una comunicación serial a 9600 baudios. ESP01 inicia la comunicación y Arduino actuá en consecuencia.

Para un correcto funcionamiento de la comunicación fue necesario agregar un caracter de control para indicar el final del comando enviado. Se eligió para control el símbolo &, por lo tanto ampersand no puede ser enviado dentro del stream comando, ya que queda como caracter restringido por el sistema.

Otro punto importante a mencionar es que cuando el stream viaja desde el servidor hacia el robot (en cualquiera de los enlaces de comunicación) el comando es con letras mayúsculas; y cuando el stream viaja desde el robot al servidor es con letras minúsculas. De esta forma solo con ver el stream ya podemos saber el origen y destino.

La lista de comandos posibles con la acción que ejecuta son los determinados en la tabla 6.

Tabla 6: Mensajes Arduino-ESP01

Comando	Acción	Respuesta
ADELANTEW	avanzar a velocidad máxima	ninguna
ADELANTES	avanzar a velocidad media	ninguna
ADELANTEX	avanzar a velocidad lenta	ninguna
ATRAS	retroceder	ninguna
IZQUIERDAQ	Doblar levemente a la izquierda	ninguna
IZQUIERDAA	Doblar a la izquierda	ninguna
IZQUIERDAZ	Doblar a la izquierda de forma pronunciada	ninguna
DERECHAE	Doblar levemente a la derecha	ninguna
DERECHAD	Doblar a la derecha	ninguna
DERECHAC	Doblar a la derecha de forma pronunciada	ninguna
DETENER	Detiene todo movimiento	ninguna
POSICION	Solicita latitud y longitud	[FLOAT]“Dato latitud” /r/n [FLOAT] “dato longitud” /r/n
SATELITE	Solicita cantidad de satélites enlazados	[INT]“número de satélites” /r/n
VELOCIDAD	Solicita velocidad del robot en Km/H	[FLOAT]“velocidad del robot” /r/n
ALTITUD	Solicita altitud	[FLOAT] “dato altitud” /r/n
ULTRA	Solicita dato del sensor ultrasonico en cm	[INT] “dato de distancia” /r/n

3.2.5.5 BATERÍA, PROTECCIÓN Y REGULACIÓN

Esta etapa está alimentada por una batería igual a la de la etapa de potencia. Salvo que en este caso la batería, luego de pasar por el switch on/off, va directo a un regulador LM7805; ya que toda esta etapa trabaja a esa tensión (salvo el ESP01 que se alimenta del regulador de Arduino). El consumo máximo teórico de todos los módulos que componen a esta etapa es de 347mA. Como la corriente máxima que entrega este integrado es de 500mA, no presenta incoveniente. Se seleccionó como medida de protección un fusible de 1A, ya que soporta picos no repetitivos de 2A, igual que el regulador. Dicho fusible está ubicado en el circuito a la entrada de alimentación.

Se considera que los 153mA restantes serán suficientes para alimentar algún sensor extra que se añada al robot. Pero se recomienda ensayar el sensor a colocar para evaluar su consumo; ya que existen sensores, como por ejemplo los sharpy, que son sensores de distancia infrarrojos que su consumo ronda los 400mA.

3.2.6 LISTA DE MATERIALES

Cotización del dólar (26/07/2019): 43,33

LISTA DE COMPONENTES COMPLETA PARA 1 ROBOT			
cantidad	item	precio por unidad	subtotal
1	chasis acrílico (2 piezas)	450	450
1	piezas 3D (6 piezas)	800	800
1	Pertinax 15 x10 cm	80	80
1	arduino uno	480	480
1	puente H L298	180	180
1	esp8266	220	220
1	Gps Ublox Neo6 con antena	600	600
1	ultrasonico HC-SR04	80	80
4	motor amarillo +rueda	200	800
2	switch tipo palanca	38,25	76,5
1	tira de pines hembra	20	20
1	tira de pines macho	20	20
1,5	mts de cable fino multifilar	40,37	60,555
1	mts de termocontraible 2,5mm	31,8	31,8
6	PC817	15	90
3	zocalo dip8	1,7	5,1
1	lm7805	19,1	19,1
6	resistencia 10k 1/4W	0,6	3,6
6	Resistencia 220 1/4W	0,6	3,6
2	Resistencia 680 1/4W	0,6	1,2
2	led	3,4	6,8
1	diodo 1n4007	1,7	1,7
1	Capacitor 104	2,12	2,12
1	Capacitor 334	4,67	4,67
2	porta fusibles para pcb	8,5	17
1	fusible 1A	5,1	5,1
1	fusible 4A	5,1	5,1
2	bornera doble.	12,75	25,5
4	batería ion de litio 18650	160	640
8	Tornillo 5/16” x 1”		0
24	Tornillo 5/16” x 3/4”		0
32	tuerca autofrenante 5/16”		0
1	barilla roscada 3/8”		0
16	Tuerca 3/8”		0
8	arandela plana 3/8”		0
8	arandela grower 3/8”		0
4	precintos chicos.		0
1	costos ferretería	270	270

TOTAL EN PESOS	4999,445
TOTAL EN DOLARES	115,38068313

3.3 SUBSISTEMA SERVIDOR

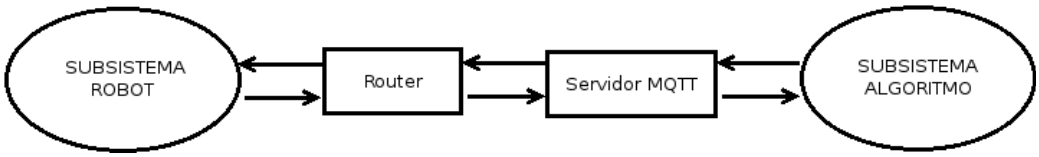


Figura 24: Esquema del subsistema servidor

Como se aprecio de la figura 24, este subsistema es el encargado de la comunicación entre los robots y el algoritmo de control. Está compuesto básicamente por 2 elementos: el router WiFi, que pertenece a la red LAN, donde se encuentran los robots y es el gateway de la red. Y el servidor MQTT que se encontraría alojado en el Cloud.

3.3.1PROTOCOLO MQTT

Como detalla Michael Yuan en [27], MQTT (Message Queue Telemetry Transport) fue desarrollado por IBM en 1999. Su aplicación original era vincular sensores en oleoductos de petróleo a satélites. Tal como sugiere su nombre, se trata de un protocolo de mensajería con soporte para la comunicación asíncrona entre las partes.

Pero la popularidad de este protocolo llegó con los dispositivos IoT, ya que resultó ideal para la conectividad de estos; debido a que es un protocolo liviano que permite la implementación en hardware de dispositivos altamente restringidos y en redes de ancho de banda limitada y de alta latencia; y a que su flexibilidad hace posible el soporte a diversos escenarios de aplicación para dispositivos y servicios de IoT.

El protocolo MQTT funciona mediante un modelo de publicación y suscripción el cual define dos tipos de entidades en la red: un bróker de mensajería y numerosos clientes. Bróker es un servidor que recibe todos los mensajes de los clientes y, en seguida, redirige estos mensajes a los clientes de destino relevantes. Un cliente es cualquier cosa que pueda interactuar con el bróker y recibir mensajes. Un cliente puede ser un sensor de IoT en campo o una aplicación en un centro de datos que procesa datos de IoT.

1.El cliente se conecta al bróker. Puede suscribirse a cualquier “topic” de mensajería del bróker. Esta conexión puede ser una conexión TCP/IP simple o una conexión TLS cifrada para mensajes sensibles.
2.El cliente publica los mensajes en un topic, enviando el mensaje y el topic al bróker.
3.Luego, el bróker remite el mensaje a todos los clientes que se suscriben a este tema.
Como los mensajes de MQTT se organizan por temas, el desarrollador de aplicaciones tiene la flexibilidad de especificar que determinados clientes solo pueden interactuar con determinados mensajes. Por ejemplo, los sensores publicarán sus lecturas en el topic "sensor_data" y se suscribirán al topic "config_change". Las aplicaciones de procesamiento de datos que guardan los datos del sensor en una base de datos de backend se suscribirán al topic "sensor_data". Una aplicación de consola administrativa podría recibir comandos del administrador del sistema para ajustar las configuraciones de los sensores, tales como la sensibilidad y la frecuencia de muestreo, y publicar dichos cambios en el topic "config_change".

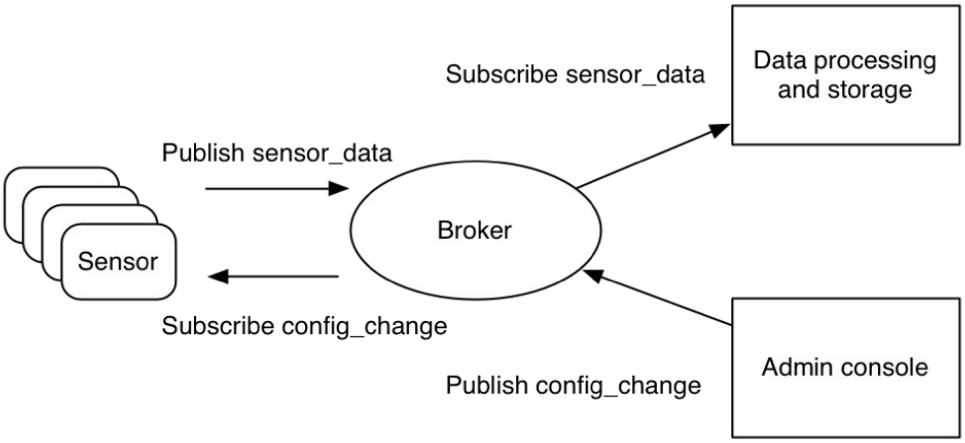


Figura 25: Ejemplo de un sistema MQTT

El puerto asignado para MQTT es el 1883, en caso de requerirlo puede incluirse a MQTT una encriptacion SSL (Secure Sockets Layer) en cuyo caso el puerto cambia al 8883. la utilización de SSL nos brinda mayor seguridad en el protocolo, pero a cambio perdemos la mayor ventaja de MQTT que eran sus paquetes pequeños y rápida respuesta.

Existen sitios webs que ofrecen servicios para MQTT como por ejemplo https://www.cloudmqtt.com/ donde registrándonos en el sitio web y asociando nuestros dispositivos IoT a una cuenta podemos llevar estadísticas de variables de sensores.

Pero en nuestro caso hemos optado por levantar nosotros mismos un servidor MQTT. Para eso utilizaremos Mosquitto, el cual es muy sencillo de instalar siguiendo tutoriales online como explica, por ejemplo, Luis del Valle Hernández en [28].

Mosquitto cuenta con 2 funciones, mosquitto_sub y mosquitto_pub, las cuales detallaremos a continuación:

3.3.1.1 COMANDO mosquitto_sub

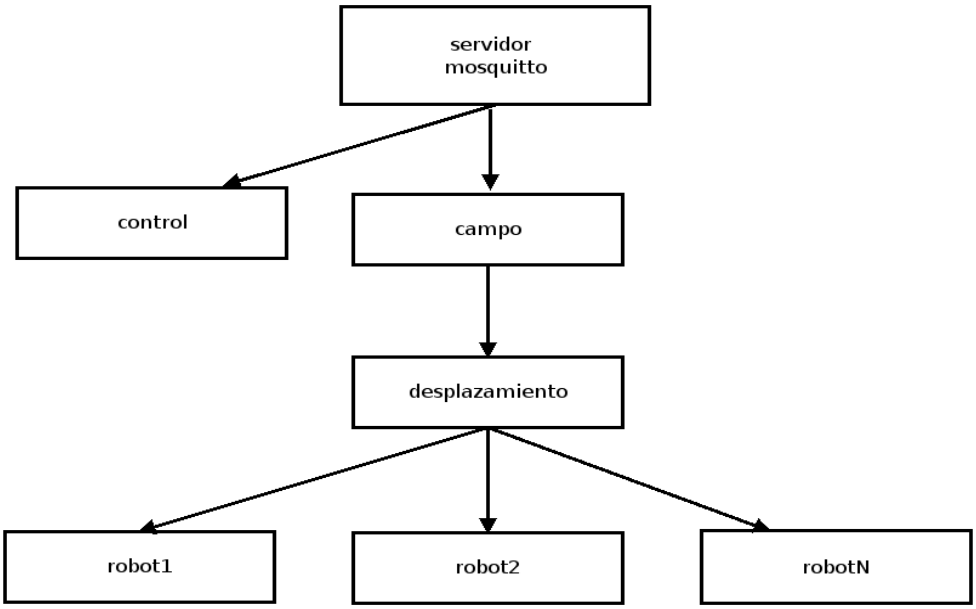
Este comando nos permite suscribirnos a un topic escuchando y mostrando por pantalla los mensajes enviados al mismo. Puede tomar muchos parámetros pero nosotros nos vamos a centrar en 4.

mosquitto_sub -h BROKER -t TOPIC

Donde:

- -h: indica que lo que viene después es el host. El host se refiere a la dirección IP o nombre de la máquina en la red del Broker Mosquitto. Si no pasamos este parámetro tomara por defecto localhost.
- BROKER: dirección IP o nombre de la máquina en la red del Broker Mosquitto.
- -t: indica que lo que viene después es el topic al que queremos suscribirnos.
- TOPIC: nombre del topic al que nos vamos a suscribir.

En nuestro caso usaremos este comando para crear un topic con el nombre control, donde todos los robots publicarán. Entonces, la salida de este comando vendría a ser todas las comunicaciones provenientes de los robots; las cuales guardaremos en un archivo de formato csv del cual leerá el algoritmo de control.



También cada robot crea su topic campo/desplazamiento/robotN siendo N el número del robot, de esta forma la red nos quedaría compuesta de la forma que lo expresa la figura 26.

Figura 26:Organización de los topics utilizados

Otra ventaja de este esquema es que nos permite reemplazar robotN por un asterisco (*), el cual hace que el mensaje deseado sea enviado a todos los robots; permitiéndonos, en caso de una emergencia, enviar un mensaje de parada general en un solo comando.

Por ejemplo, si el algoritmo desea comunicarse con el robot2, debe publicar en el topic campo/desplazamiento/robot2 (mosquitto_pub -h localhost -t campo/desplazamiento/robot2 -m “text”) y el robot2 para contestarle al algoritmo publicara en el topic control (client.publish("control", msg);)

3.3.1.2 COMANDO mosquitto_pub

Con este comando podemos publicar mensajes muy simples. Como ocurre con mosquitto_sub puede tomar muchos parámetros, pero a nosotros nos importan sólo 6.

mosquitto_pub -h BROKER -t TOPIC -m MENSAJE

Donde:

- -h: indica que lo que viene después es el host. El host se refiere a la dirección IP o nombre de la máquina en la red del Broker Mosquitto. Si no pasamos este parámetro tomara por defecto localhost.
- BROKER: dirección IP o nombre de la máquina en la red del Broker Mosquitto.
- -t: indica que lo que viene después es el topic al que queremos enviar el mensaje.
- TOPIC: nombre del topic al que vamos a enviar el mensaje.
- -m: indica que lo que viene después es el mensaje.
- MENSAJE: el mensaje que queremos enviar.

Este comando es el que utiliza el algoritmo de control de los robots para poder enviarle comandos, que son solo un carácter en mayúscula. Si el comando enviado es de movimiento, el ESP01 recibe el comando y lo envía al Arduino y devuelve la confirmación de recepción del comando al servidor. Si el comando es un requerimiento de datos de sensores, el ESP01 aguarda a que Arduino responda con el valor del sensor, forma el mensaje con formato csv y lo envía al servidor. La respuesta de los robots se realiza publicando en el topic /control y es siempre en letras minúsculas.

3.3.2 COMUNICACIÓN M2M (ESP01-mosquitto)

En la tabla 7 podemos observar cuales son los comandos que pueden ser enviados desde el servidor MQTT al ESP01 y cual es la respuesta de este que queda grabada en un archivo log.

Tabla 7: Mensajes ESP01-servidor MQTT

CARACTER QUE SE ENVÍA	ACCIÓN A EJECUTAR POR EL ROBOT	RESPUESTA DEL ROBOT
Q	Doblar levemente a la izquierda	robotN,desplazamiento, doblando,izquierda, abierto
A	Doblar a la izquierda	robotN,desplazamiento, doblando,izquierda,medio
Z	Doblar a la izquierda de forma pronunciada	robotN,desplazamiento, doblando,izquierda, cerrado
W	Avanzar a velocidad máxima	robotN,desplazamiento, avanzando, rapido,
S	Avanzar a velocidad media	robotN,desplazamiento, avanzando, medio,
X	Avanzar a velocidad lenta	robotN,desplazamiento, avanzando, lento,
E	Doblar levemente a la derecha	robotN,desplazamiento, doblando,derecha, abierto

D	Doblar a la derecha	robotN,desplazamiento, doblando,derecha,medio
C	Doblar a la derecha de forma pronunciada	robotN,desplazamiento, doblando,derecha, cerrado
R	Retroceder	robotN,desplazamiento, retrocediendo,,
F	Detiene todo movimiento	robotN,desplazamiento, detenido,,
V	Solicita velocidad	robotN,sensor,velocidad,”valor de velocidad”,
P	Solicita latitud y longitud	robotN,sensor,posicion,” valor latitud”,” valor longitud”
N	Solicita cantidad de satélites enlazados	robotN,sensor,satelite,” numero de satelites”,
H	Solicita altitud	robotN,sensor,altitud,”valor de altitud”,
U	Solicita dato del sensor ultrasónico	robotN,sensor,ultrasonico,”valor de ultrasonico”,

3.4 SUBSISTEMA ALGORITMO

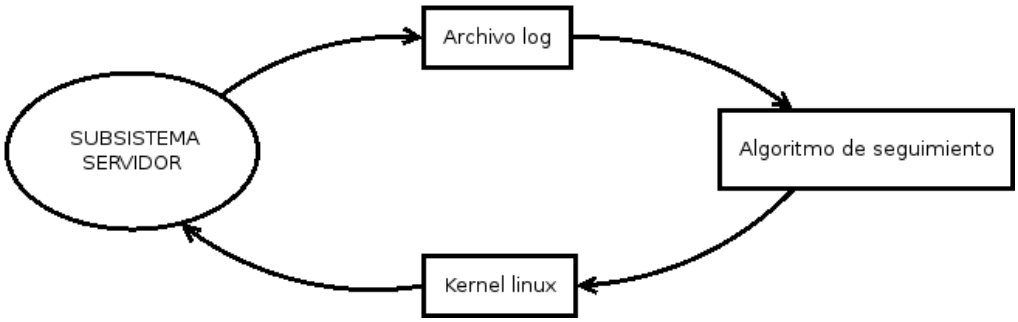


Figura 27: Esquema del subsistema algoritmo

Esta etapa compuesta por los elementos de la figura 27 la cual está comprendida completamente en el servidor en el Cloud. Está compuesta por el archivo log creado con la salida del servidor MQTT, el algoritmo de control de los robots, el cual es el encargado de la generación de la trayectoria y seguimiento de la misma; el cual utiliza la función mosquitto_pub del sistema operativo para acceder al servidor.

El elemento de mayor importancia de esta etapa es el algoritmo de seguimiento. Humberto Alejandro Secchi explica en [29] que la generación de trayectoria se puede dividir en 3 etapas:

-GGT (Generador Global de Trayectoria): Es el nivel jerárquico superior. Este nivel es el encargado de decidir, en base a la tarea asignada, las coordenadas del punto destino, de puntos intermedios en la trayectoria y, en caso que el camino esté obstruido, redefinir la trayectoria elegida. La información que emplea este nivel jerárquico puede ser generada fuera de línea (conocimiento a priori del ambiente de trabajo) o en línea, en base a criterios predefinidos y utilizando la información provista por el sistema

sensorial (desconocimiento parcial o total del ambiente de trabajo) a partir de la elaboración de mapas del entorno.

-GLT (Generador Local de Trayectorias): Es el nivel jerárquico intermedio. Este nivel jerárquico hace las veces de operador (piloto) del robot móvil, evitando los obstáculos del camino, realizando correcciones en la trayectoria y adecuando la velocidad del vehículo de acuerdo a la maniobra que se realiza. Permite un control dinámico del robot móvil. Mantiene informado al GGT sobre los resultados del objetivo asignado, y en caso de no tener un conocimiento a priori del ambiente de trabajo, genera información para ser almacenada en la memoria del GGT. Está directamente comunicado con el sistema sensorial, lo que le permite tomar decisiones en línea y además genera los valores de referencia para el Control Local del Sistema de Tracción y Dirección.

-CLSTD (Control Local del Sistema de Tracción y Dirección): Es el nivel jerárquico inferior. Interpreta las referencias enviadas desde el GLT y genera las acciones de control para que los motores de tracción y dirección trabajen en forma coordinada y de esta manera se alcance el punto destino siguiendo trayectorias suaves, libres de oscilaciones y maniobras violentas para la carga.

Para nuestro caso el algoritmo en el Cloud contiene al GGT en su totalidad y la parte de CLSTD que corresponde a alcanzar el punto deseado. La parte restante del CLSTD que contempla el control directo sobre los motores está alojada en el robot; más específicamente en el programa del Arduino. En este trabajo no se contempla el GLT, ya que esto implica que el robot toma la decisión sobre la trayectoria. Como

nuestro objetivo es realizar el mayor procesamiento posible fuera del robot, no contamos con este seguimiento como tal, aunque si se prevee para trabajos futuros la corrección de trayectoria, la cual será asignada al algoritmo en el Cloud.

Para la generación global de la trayectoria se optó por lo siguiente: solicitarle al operador la posición final (latitud y longitud) al cual desea que el robot arribe, luego el algoritmo solicita la posición al robot, esta posición es tomada como posición inicial. Con estos 2 puntos traza una trayectoria recta la cual se puede observar en la figura 28, y será labor del CLSTD (alojado en el Cloud) el poder seguirla.

Figura 28: Trayectoria ideal

Para el seguimiento de la trayectoria se utilizó como base un algoritmo de K. C. Koh y H. S. Cho[30]. El cual consiste en suponer un triángulo rectángulo donde los catetos son la diferencia de la latitud (latitud final menos latitud actual) y la diferencia de la longitud (longitud final menos longitud actual), y por ende la hipotenusa es la trayectoria a seguir. De donde podemos saber si nos acercamos o alejamos del punto deseado simplemente viendo si el valor de la hipotenusa crece o disminuye. Este dato sera el que nos indique cuando lleguemos al punto final.

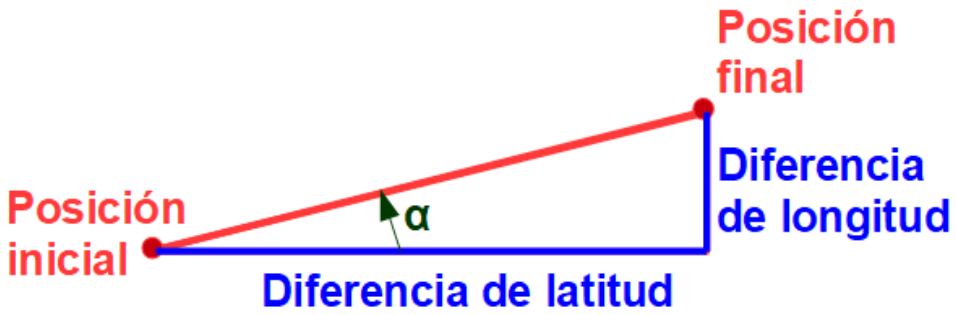


Figura 29: Determinación del angulo α

Ahora sabemos que la forma más rápida para que nuestro robot llegue al objetivo es viajar en línea recta sobre la hipotenusa, pero la incertidumbre ocurre en como saber si lo estamos logrando. Para solucionarlo basta con observar el ángulo (alfa) que forman la hipotenusa con el cateto adyacente. Este ángulo lo podemos obtener mediante la trigonometría como se observa de la figura 29.

El ángulo inicial lo sabemos de la generación global de la trayectoria. Si el angulo actual formado por los puntos posición actual y posición final es igual al ángulo inicial significa que estamos sobre la hipotenusa. Ahora si este difiere, según sea por defecto o por exceso, y cuanto sea la magnitud del desfasaje podemos tomar las acciones correspondiente para volver a nuestra trayectoria ideal.

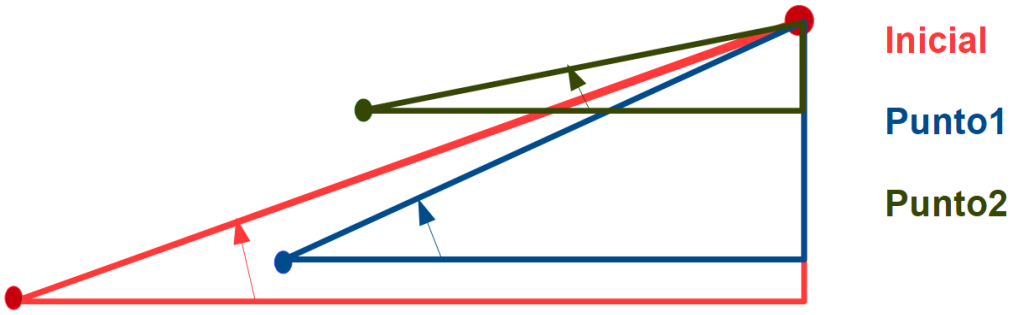


Figura 30: Comparación de ángulos

Como podemos ver en la figura 30, el triángulo rojo está formado por los puntos posición inicial y posición final, y dan como resultado la trayectoria ideal a seguir, la cual se cumple si el robot mantiene el angulo constante. Supondremos ahora que el robot se desvía de la trayectoria ideal y llega al punto 1. Aquí el algoritmo solicitará datos de posición para calcular el valor del ángulo actual, este será más grave que el ángulo inicial, y por lo tanto, tras compararlos, el algoritmo indicará al robot que debe doblar a la izquierda. Supongamos que el robot corrige hasta llegar al punto 2. El algoritmo nuevamente solicitará posición y calculara el ángulo actual, el cual resultará más agudo que el ángulo inicial. Por lo tanto, el algoritmo le indicará al robot que debe doblar a la derecha.

Este seguimiento, así como esta planteado, no sería el más conveniente, ya que nos generaría en el robot un comportamiento del tipo on/off dada la imposibilidad del robot de coincidir siempre con una recta, haciendo que el robot siempre esté entre las decisiones de doblar a la izquierda o doblar a la derecha. Es por eso que la recta se transforma en un angulo, como observamos en la figura 31, donde el usuario es el que decide cuanta libertad darle al robot, de esta forma la recta ideal en la que el robot avanza hacia adelante pasa a transformarse en una zona.

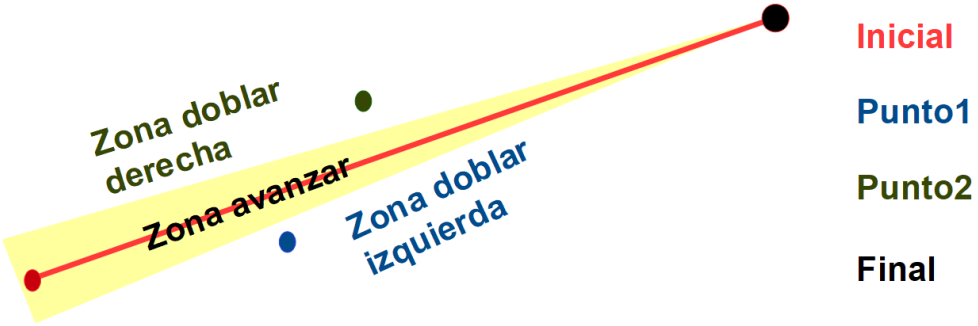


Figura 31: Determinación de la zona avanzar

4 RESULTADOS

Para la prueba del sistema se optó por la cancha de football del campus de la universidad ubicado en la intersección de Carlos Washington Lencinas y Av. Boulogne Sur Mer (ver figura 32). Se estableció como objetivo que un robot inicie en una esquina de la cancha y llegue al punto final, el cual se encontraba dentro del arco opuesto a la esquina elegida. Consideraremos que el sistema es funcional si el robot consigue llegar al área de meta.

Figura 32: Campus deportivo Universidad de Mendoza

El experimento se realizó con el servidor MQTT y el algoritmo montados en una notebook Dell Latitude E6400 con sistema operativo Debian 9.9. Para la comunicación se utilizo un router wireless modelo Cnet CQR-980, el cual se alimenta a 5V por miniUSB, lo que permite encenderlo conectándolo a un puerto USB de la notebook, logrando así que todo el sistema pueda ser móvil.

La red estaba formada por la notebook con una IP estática 192.168.1.200 conectada por medio de un cable de red al router con dirección 192.168.1.1. El robot se comunicaba con el router de forma inalámbrica con la IP estática 192.168.1.101.

La precisión del dato del GPS es al sexto decimal, pero, durante esta prueba, los 2 primeros decimales permanecieron constantes; siendo para latitud -32,88XXXX y para la longitud -68,86XXXX. Por lo tanto tomamos como datos útiles los 4 últimos decimales de ambos.

Para la prueba se decidió darle al robot una libertad del 10% del ángulo inicial, lo cual significaría una tolerancia de 6 grados para definir la zona de avanzar.

Inicialmente tomamos algunos puntos de la cancha para tener como referencia, estos son: las 4 esquinas, los 2 palos de los arcos y el centro de la cancha, los cuales se pueden observar en la figura 33.

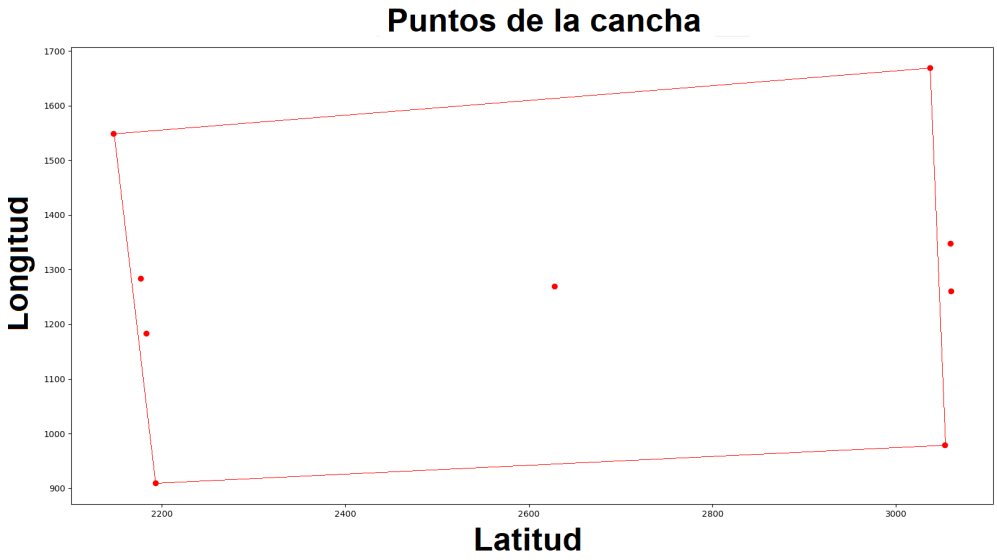


Figura 33: Puntos de importancia de la cancha

El punto de salida del robot será la esquina superior derecha, el cual está ubicado en (-32,883045 ; -68,861680). Esta es la esquina de la chacha más cercana a la entrada del campus. El punto final se encuentra en (-32,882157 ; -68,861237). Por lo tanto, la trayectoria ideal a seguir es la recta que une a estos 2 puntos, la cual esta indicada de color verde en la figura 34.

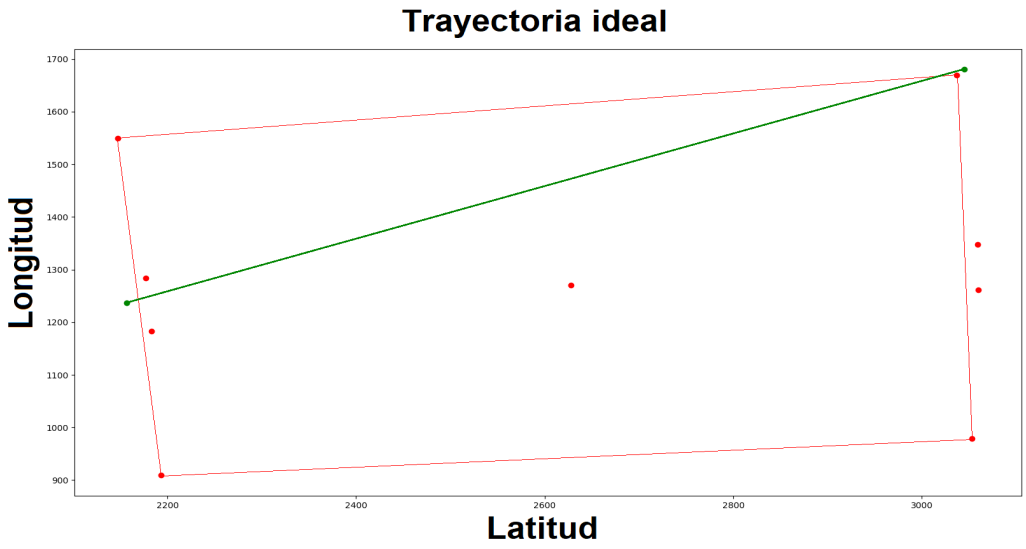


Figura 34: Trayectoria ideal

Antes de iniciar con el algoritmo de seguimiento se enciende el robot y se le da un tiempo de 5 minutos para que enlace la mayor cantidad de satélites posible, para lograr la mejor experiencia. Al momento de iniciada la prueba, la cantidad de satélites eran 11. Durante la experiencia, la cantidad de satélites varió entre 12 y 11 satélites.

A continuación podemos observar en al figura 35 el resultado de la prueba.

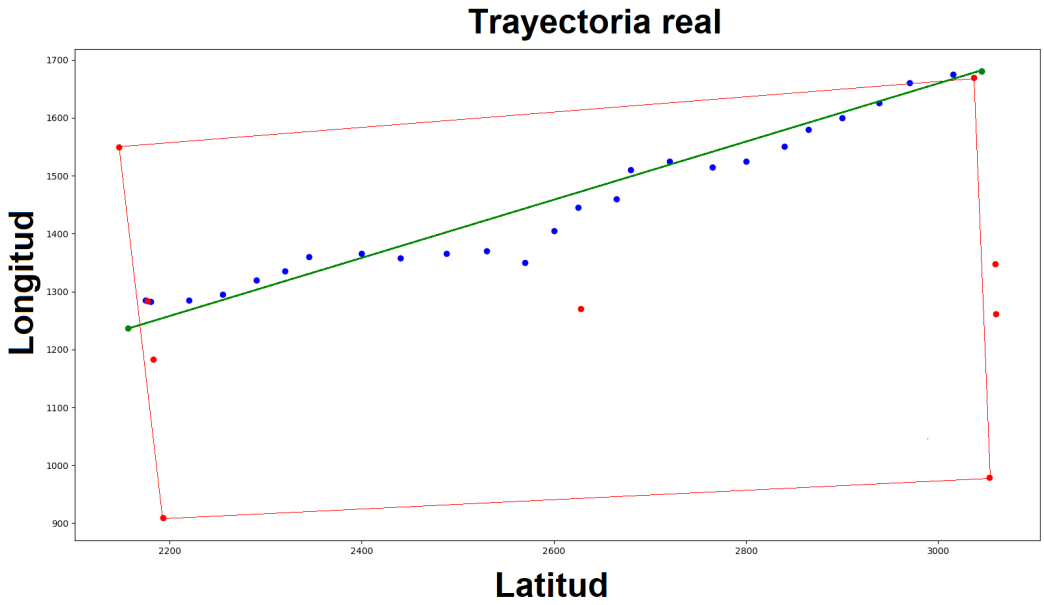


Figura 35: Coordenadas obtenidas durante la prueba

Desde el algoritmo se decidió que se solicitarían muestras de posición cada 5 segundos. A lo largo de la experiencia, la cantidad de muestras tomadas fueron 26. Cada 5 muestras de posición, se solicitaba una muestra de cantidad de satélites.

Es importante tener en cuenta que el gráfico corresponde no a la posición del robot, sino a la posición más el error del GPS. Por lo tanto, lo observado de este gráfico no necesariamente coincide con lo sucedido en la realidad. El punto de mayor error con la trayectoria ideal la podemos observar en la muestra 14. Es importante el conocimiento obtenido en la muestra 10 que se enlazó 12 satélites, y en la muestra 15 hubieron 11 satélites. Como ya vimos en las pruebas de ensayo del GPS, este cambio de constelación nos provoca el mayor error. La posición real observada del robot durante la experiencia no coincide con la del gráfico, el robot no se encontraba tan desfasado, pero este actuó según se lo indicó el algoritmo; por lo tanto, el robot ejecutó el comando recibido de doblar a la derecha, y esto provoco que el mismo se saliera de la trayectoria ideal y se alejara. Luego el robot recuperó el rumbo hacia el punto final pero no fue capaz corregir lo suficiente como para terminar dentro del arco. Finalmente, el robot termina pasando a 30 cm por afuera del palo del arco y enganchándose en la red.

Consideramos que el objetivo fue cumplido, a pesar de no poder haber terminado dentro del arco. Como consecuencia del resultado, consideramos oportuno un estudio posterior sobre el algoritmo, en el cual descubrimos que este posee un error mayor al del GPS; el cual es determinado de la siguiente manera: Primero recordemos que el error del GPS es de un radio de 2,5 metros, para simplificar este análisis consideraremos el error del GPS solamente en una dimensión, o sea, que si estamos parados en un punto de la trayectoria ideal, podemos estar como máximo a 2,5 metros por encima de este o 2,5 metros por debajo. Supongamos un ángulo (alfa) de tolerancia para la zona de avanzar de 10º.

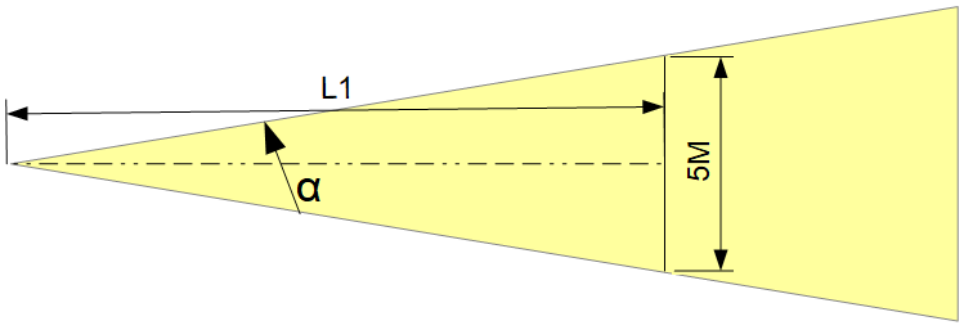


Figura 36: Determinación de distancia L1

Definiremos a la distancia crítica L1 como la distancia aquella para la cual valores mayores pueden contener en su totalidad el radio de 2,5 metros de error del GPS. Esta distancia la podemos obtener del siguiente desarrollo: Tomaremos solo la mitad de la zona, por lo tanto trabajaremos con un ángulo mitad beta, el cual es la mitad de alfa. Con esto obtendremos un triángulo rectángulo donde un cateto será 2,5 metros y el otro será la distancia a obtener.

$$\beta = \alpha / 2$$

$$L1 = 2{,}5\,M / tg(\beta) = 2{,}5\,M / 0{,}0875 = 28{,}57\,M$$

Ecuaciones 1: Determinación de distancia L1

Por lo tanto podemos concluir que con una zona de avanzar definida por un ángulo de 10 grados, el algoritmo funciona sin errores hasta los 28,57 metros

Ahora definiremos una distancia límite L2 a la cual, el algoritmo, deja de ser eficiente; ya que a distancia menores a esta, el error propio del GPS hará que la posición dada caiga en cualquiera de las 3 opciones a tomar, por lo tanto, a esta distancia, la probabilidad de cada decisión (doblar derecha, doblar izquierda y avanzar) es del 33%; y esto ocurre a una distancia de:

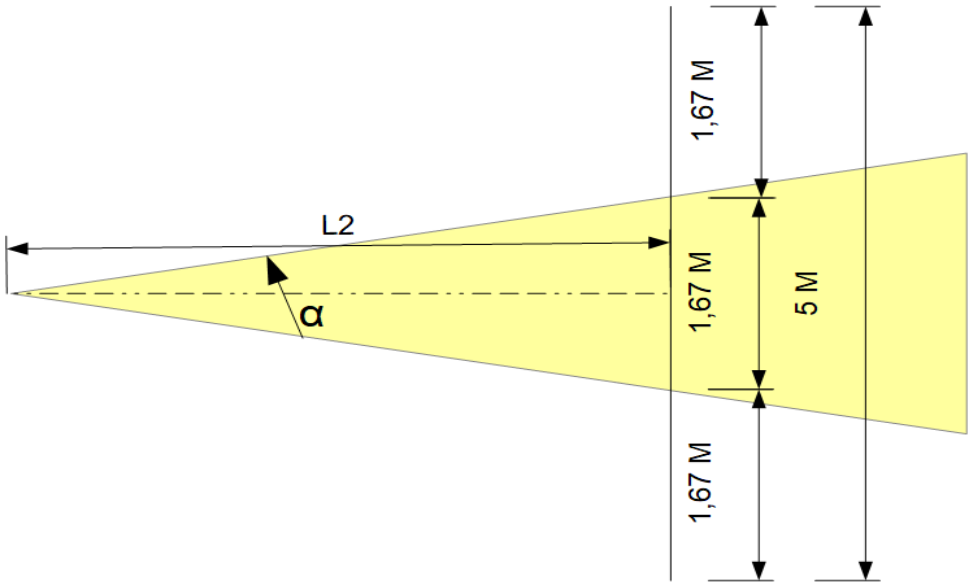


Figura 37:Determinación de distancia L2

$$L2 = 0{,}835\,M \,/\, tg(\beta) = 0{,}835\,M \,/\, 0{,}0875 = 9{,}54\,M$$

Ecuaciones 2:Determinación de distancia L2

Por lo tanto, cuando el robot se encuentra a menos de 9,54 metros ya no nos podemos fiar del algoritmo; porque a medida que nos acercamos al punto final, la probabilidad de avanzar en línea recta disminuye tendiendo a un sistema de control del tipo on/off.

5 CONCLUSIÓNES

Consideramos que el objetivo del trabajo fue alcanzado ya que:

Se desarrollaron 2 robots de bajo costo, los cuales pueden ser construidos con piezas adquiribles en el mercado local. Los robots cuentan con una gran confiabilidad, debido a las etapas aisladas en su circuito eléctrico. Por el hecho de añadir fusibles se protege a los elementos más caros como los módulos de microcontroladores y GPS.

A pesar que el objetivo inicial era el procesamiento en el Cloud, se montó con éxito un servidor MQTT de forma local; en el cual se vincularon los robots mediante una conexión wireless. El protocolo MQTT resultó ideal para la aplicación deseada. La implementación de este en Cloud quedará para futuros trabajos.

Se logró la comunicación del algoritmo de seguimiento por medio de un software y la comunicación con los robots.

La prueba final de generación y seguimiento de trayectoria finalizó con éxito, con un error dentro de lo aceptable para los requisitos de la prueba.

Se analizó un algoritmo de generación y seguimiento de trayectoria a partir de lo observado en el sistema, obteniendo los puntos claves, alcances y limitaciones del mismo.

6 LIMITACIONES

Como ya se detalló en el trabajo, el sistema está limitado en precisión a un radio de 2,5 metros debido al error propio del GPS.

De la forma que fue planteado el software para el algoritmo en el Cloud, el cual genera una cola que se atiende por orden de llegada (FIFO), este será el primer punto donde tengamos el efecto cuello de botella cuando queramos incrementar la cantidad de robots en el campo. Este efecto también se verá agravado con la implementación de algoritmos de seguimiento y trabajo colaborativo que requieran de una mayor carga computacional, haciendo que el tiempo de atención de los robots sea mayor.

Se optó por el protocolo MQTT debido a que era ideal para la transmisión de pequeños datos, pero esto también nos trae el contratiempo de no poder tener una gran trasmisión de datos, dejando afuera así la transmisión de video mediante la utilización de cámaras montadas en el robot.

7 TRABAJOS FUTUROS

Se plantean los siguientes trabajos a futuro para continuar con el desarrollo del sistema:

Implementación del algoritmo en el Cloud mediante el servicio MQTT de AWS (Amazon Webs Services). Incorporar procesamiento multi-hilo y entorno para el usuario.

Prueba, análisis y comparación de diferentes algoritmos de generación y seguimiento de trayectoria contemplando trayectorias de colisión. Desarrollo de trabajo cooperativo entre robots mediante el sistema de posicionamiento.

Añadir la capacidad de grabado de microcontroladores de forma inalámbrica durante una prueba.

Incorporar en el robot un plan de contingencia para pérdida de conexión con el servidor y un sistema de *back to home*

Añadir otros servicios AWS al sistema, como por ejemplo el simulador ROS (Sistema Operativo Robótico), realizando en este la generación y seguimiento de la trayectoria.

8 BIBLIOGRAFIA

[1] R. Buyya, C. Yeo, S. Venugopal, J. Broberg, and I. Brandic. Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility. Future Generation Computer Systems, 25(6):599–616, 2009.

[2] Godoy, Pablo; Plataforma de Desarrollo de Laboratorios Remotos de Redes de Sensores Inalámbricos basados en Cloud Computing; Memoria de tesis doctoral; Universidad de Mendoza; 2016.

[3] K. Salah, M. Hammoud, and S. Zeadally. Teaching cybersecurity using the cloud. IEEE Transactions on Learning Technologies, 8(4):383–392, Oct 2015

[4] Kehoe, B., Patil, S., Abbeel, P., & Goldberg, K. (2015). A survey of research on cloud robotics and automation. *IEEE Transactions on automation science and engineering*, *12*(2), 398-409.

[5] Kumar, V., & Michael, N. (2012). Opportunities and challenges with autonomous micro aerial vehicles. The International Journal of Robotics Research, 31(11), 1279-1291.

[6] D'Andrea, R. (2012). Guest editorial: A revolution in the warehouse: A retrospective on kiva systems and the grand challenges ahead. IEEE Transactions on Automation Science and Engineering, 9(4), 638-639.

[7] Kuffner, J. (2010). Cloud-enabled humanoid robots. In Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on, Nashville TN, United States, Dec.

[8] Guizzo, E. (2011). Cloud robotics: Connected to the cloud, robots get smarter. IEEE Spectrum, 2014-25.

[9] Waibel, M., Beetz, M., Civera, J., d'Andrea, R., Elfring, J., Galvez-Lopez, D. & Schiessle, B. (2011). Roboearth. IEEE Robotics & Automation Magazine, 18(2), 69-82.

[10] Arumugam, R., Enti, V. R., Bingbing, L., Xiaojun, W., Baskaran, K., Kong, F. F., & Kit, G. W. (2010, May). DAvinCi: A cloud computing framework for service robots. In Robotics and Automation (ICRA), 2010 IEEE International Conference on (pp. 3084-3089). IEEE.

[11] Hunziker, D., Gajamohan, M., Waibel, M., & D'Andrea, R. (2013, May). Rapyuta: The roboearth cloud engine. In Robotics and Automation (ICRA), 2013 IEEE International Conference on (pp. 438-444). IEEE.

[12] “Industry 4.0.” [Online]. Available: http://www.bmbf.de/en/19955.php

[13] Bekris, K., Shome, R., Krontiris, A., & Dobson, A. (2015). Cloud automation: Precomputing roadmaps for flexible manipulation. IEEE Robotics & Automation Magazine, 22(2), 41-50.

[14] Hunter, T., Moldovan, T., Zaharia, M., Merzgui, S., Ma, J., Franklin, M. J., ... & Bayen, A. M. (2011, October). Scaling the mobile millennium system in the cloud. In Proceedings of the 2nd ACM Symposium on Cloud Computing (p. 28). ACM.

[15] Berenson, D., Abbeel, P., & Goldberg, K. (2012, May). A robot path planning framework that learns from experience. In Robotics and Automation (ICRA), 2012 IEEE International Conference on (pp. 3671-3678). IEEE.

[16] “MyRobots.com.” [Online]. Available: http://myrobots.com

[17] “BT_SmartCar” [Online]. Available: https://thingiverse.com/

[18] ESP-01 WiFi Module.Shenzhen Anxinke Technology. China. (2015)

[19] TP4056. NanJing Top Power ASIC Corp. China. (2012)

[20] Lithium ion Cell type UR18650F. Sanyo. USA. (2004, aug)

[21]“Las ruedas del robot agarre” [Online]. Available: https://www.askix.com

[22]“HC-SR04 User Guide” [Online]. Available:https://www.mpja.com

[23]NEO-6 GPS Modules Datesheet. U-blox. USA (2011)

[24]“TinyGPS” [Online]. Available: https://github.com
[25]Godoy C. G.[CarlosVolt Tutoriales] (2016,abril 6).IOT: Actualización de firmware módulo wifi esp8266 [Archivo de video]. Available: https://www.youtube.com/watch?v=ahPCISKNeoE
[26]”pubsubclient“ [Online]. Available: https://github.com
[27]Yuan M. (2018, Dic 5) “Conociendo MQTT“ [Online]. Available: https://www.ibm.com/developerworks/ssa/library/iot-mqtt-why-good-for-iot/index.html
[28]del Valle Hernandez L. “Guía de introducción a MQTT con ESP8266 y Raspberry Pi”[Online]. Available: https://programarfacil.com/esp8266/mqtt-esp8266-raspberry-pi/
[29]Bambino, I. (2008). Una Introducción a los Robots Móviles.
[30]Koh, K.C. & Cho, H.S. Journal of Intelligent and Robotic Systems (1999) 24: 367. https://doi.org/10.1023/A:1008045202113

9 ANEXOS

9.1 GRABAR ESP01

Para poder subir el código al ESP01 se requiere que se encuentre en modo programación. En este modo, el microcontrolador no funciona de su manera habitual. Es decir, no ejecuta el código alojado en su memoria. Para poder habilitar este modo se debe colocar el pin GPIO0 a GND, y reiniciar el módulo. También será necesario un adaptador USB-TTL como, por ejemplo, el modelo CP2102, el cual se utilizó en este caso. Para facilitar la labor de grabado del módulo se proporciona una PCB, en la cual se conecta el ESP01 y el CP2102. Esta PCB cuenta con un botón para reiniciar el ESP01 y un jumper para pasarlo de modo normal a modo programación. El proceso de grabado es el siguiente:

1) Conectar la PCB al ESP01 y al CP2102.
2) Conectar el CP2102 a la PC
3) Colocar el jumper (modo programación) y presionar el botón
4) Subir el código deseado por medio del IDE de Arduino
5) Quitar el jumper (modo normal) y presionar el botón

Con esos pasos ya debería estar corriendo nuestro código en el ESP01. Podemos utilizar el monitor serial en caso de ser necesario para el debug.

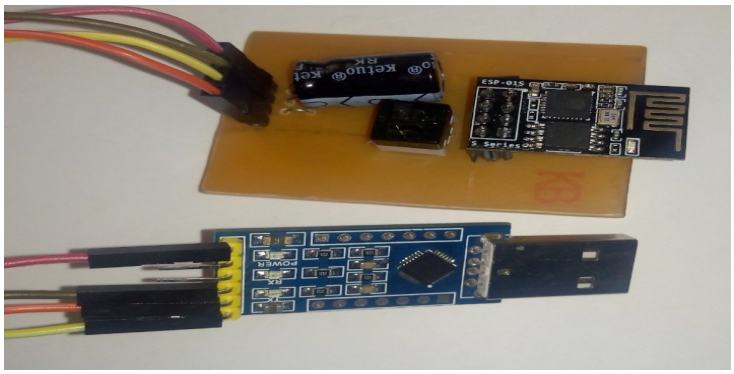


Figura 38: Elementos para el grabado de ESP01

9.2 DATASHEET

A continuación se detalla los datos técnicos del robot

Dimensiones: 15,5 cm x 29 cm x 13,2 cm

Peso: 1 Kg

Etapa de control/comunicación:

- Tensión de alimentación: 6,5V a 15V
- Consumo de corriente: 350 mA
- Autonomía: 410 minutos (6 horas 50 minutos)
- Fusible: 1A

Etapa de potencia:

- Tensión de alimentación: 7,4V a 15V
- Consumo de corriente: 2400mA
- Autonomía: 60minutos (1 hora)
- Fusible: 4A

Velocidad máxima: 0,61 m/s (2,196km/h) *¹

Velocidad normal: 0.483 m/s (1.74km/h) *¹

Alcance de sensor de obstáculo: máximo 4 metros *²

Precisión del GPS: 2,5 metros

Puerto de comunicación: UART (9600 baudios)

Comunicación Wireless: 802.11 b/g/n

Expansiones: 4 ADC, 2 Encoders, 1 puerto I2C *³.

Nota:

*¹ Con tensión de alimentación de motores de 7,4V

*² Reducido a 50cm por software

*³ En caso de usar I2C solo quedarán disponibles 2 ADC

9.4 GITHUB

Tanto el presente documento como todos los archivos necesarios (como lo son los códigos fuente de Arduino y ESP01, diseño de placas PCB, diseño de piezas 3D, diseño de chasis) para replicar este trabajo final se encuentran en el formato digital que acompañan a este documento físico y compartidos en el sitio GitHub bajo licencia GNU.

https://github.com/inggunny/Laboratorio_de_robotica_movil_en_Cloud