TÍTULO DEL TRABAJO DE GRADO

Radio enlace mesh con tecnología LoRa para el monitoreo remoto de producciones ganaderas.

AUTORES

Franco Mateo Chinchilla Ballesteros 1095817955

Jorge Andrés Yepes Castillo 1098759761

UNIDADES TECNOLÓGICAS DE SANTANDER

FACULTAD DE CIENCIAS NATURALES E INGENIERIAS

INGENIERIA ELECTRONICA

BUCARAMANGA

FECHA DE PRESENTACIÓN: DD-MM-AAAA

TÍTULO DEL TRABAJO DE GRADO

Radio enlace mesh con tecnología LoRa para el monitoreo remoto de producciones ganaderas.

AUTORES

Franco Mateo Chinchilla Ballesteros 1095817955

Jorge Andrés Yepes Castillo 1098759761

Trabajo de Grado para optar al título de

INGENIERO ELECTRONICO

DIRECTOR

Carlos Lizardo Corzo Ruiz

Codirector Rafael Augusto Núñez Rodríguez

Grupo de Investigación en Control Avanzado-GICAV

UNIDADES TECNOLÓGICAS DE SANTANDER

FACULTAD DE CIENCIAS NATURALES E INGENIERIAS

INGENIERIA ELECTRONICA

BUCARAMANGA

FECHA DE PRESENTACIÓN: DD-MM-AAAA

Nota de Aceptación

_____

_____

_____

_____

_____

Firma del jurado

_____

Firma del Jurado

DEDICATORIA

AGRADECIMIENTOS

A Dios, por permitirme llevar a cabo con éxito este proyecto, por ser mi guía y mi fortaleza en cada decisión.

A mis padres por darme el apoyo incondicional en cada decisión y paso que tome.

Al director del proyecto, el Ph.D CARLOS LIZARDO CORZO RUIZ por su constante exigencia y apoyo para la realización del proyecto.

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

RESUMEN EJECUTIVO

El sector de la ganadería localizado generalmente en zonas remotas del territorio nacional se encuentra aislado y con la obligación de intervención presencial constante en su seguimiento y operación. Para ello, se diseñó una red mesh conformada por varios nodos que determinan una ruta dinámica para que la información llegue a su destino y no se interrumpa del resto de la red cuando alguno de estos falle, de esta manera proporcionar una transmisión de datos permitiendo almacenar y monitorear la información a grandes distancias. Esta recolección de datos y su caracterización se manejó mediante una investigación descriptiva cuantitativa que nos permitió responder a los objetivos del proyecto a desarrollar.

Para la implementación se utilizaron módulos de radio LoRa junto con una plataforma de código abierto especialmente aplicado a entornos de IoT. Se estudió la propagación en el espectro para el diseño de la red mesh y su difusión mediante repetidores distribuidos estratégicamente y con línea de vista entre ellos.

Como resultado final se alcanzó una cobertura amplia de transferencia de información para la monitorización de la productividad ganadera, que a su vez sea base para la implementación de esta nueva tecnología en la región y futuras investigaciones por parte de los estudiantes de la institución que puedan apoyarse en el proceso y análisis contenido en este informe final entregado.

PALABRAS CLAVE. LoRa, mesh, IoT, inalámbrico, red.

INTRODUCCIÓN

Las comunicaciones electrónicas son importantes para el desarrollo social en cualquier ámbito debido a la constante necesidad de enviar y recibir información en nuestros dispositivos, por ello es necesario implementar mecanismos de transferencia en lugares donde no están presentes o de difícil acceso. Menciona la ministra de las TIC Sylvia Constaín, en una de sus columnas que La Cuarta Revolución Industrial no solo enmarca cambios que transforman el modo en que vivimos, trabajamos e incluso la manera de interactuar entre nosotros, sino que también nos presenta grandes oportunidades para lograr mayor crecimiento y desarrollo desde y con nuestras regiones (MinTIC, 2019).

Desde la medición de las condiciones ambientales que influyen en la producción de cultivos hasta el seguimiento de los indicadores de salud del ganado, la tecnología del Internet de las cosas (IoT) para la agricultura permite eficiencias que reducen el impacto ambiental, maximizan el rendimiento y minimizan los gastos. Los casos de uso de agricultura inteligente basados en los dispositivos LoRa de Semtech y las cualidades inalámbricas de largo alcance y baja potencia permiten el uso de sensores de bajo costo para enviar datos desde la granja a la nube, donde pueden analizarse para mejorar las operaciones. (SEMTECH, 2019) .

En este trabajo de grado se propone el desarrollo de una red inteligente de bajo costo, junto con el diseño de una red mesh auto gestionable a partir de estos módulos de radio LoRa, este proyecto analiza la entrega de información con datos actuales de los nodos ubicados con línea de vista entre ellos, con la posibilidad que el usuario pueda acceder a la información a través de una conexión internet desde cualquiera de sus dispositivos y disponer de los recursos de información previamente entregados conexos a este trabajo mediante el empleo de tipo computacional como método utilizado para el diseño y verificación la fiabilidad de los paquetes a transmitir por la red instalada. Como técnica se usó una recolección de datos en tiempo real que se contrastó con los datos de distancia entregados con la plataforma Google Earth, además, sirve como base para proyectos académicos futuros que se realicen dentro del grupo de investigación y en la Universidad en general.

1. DESCRIPCIÓN DEL TRABAJO DE INVESTIGACIÓN

1.1. PLANTEAMIENTO DEL PROBLEMA

La ubicación geográfica y su variedad de ecosistemas hacen de Colombia un territorio amplio de uso del suelo para la ganadería, esta a su vez es la mayor fuente de proteína y otros subproductos del consumo de la población nacional y de la economía que ha logrado imponerse en la forma de vida de los campesinos y empresarios del país, generando ingresos tres veces por encima del valor de producciones más relevantes como el café pero que asimismo se desconoce (FEDEGAN, 2019). Es difícil mantener un buen rendimiento en cualquier actividad de producción sin una adecuada comunicación y monitorización. Si bien en las zonas rurales existen las redes 2G, estas son de carácter privado y se contempla la posibilidad de apagarla definitivamente según la Comisión de Regulación de Comunicaciones (CRC, 2019). Debido a que hablamos de complejos ganaderos que en su mayoría están ubicados en zonas alejadas de las grandes poblaciones, estos no cuentan con acceso a internet, por lo tanto, no existe un medio de comunicación que permita transmitir la información; además de la comprensión de cómo el sector interesado en un proyecto percibe sus propios problemas y las opciones de desarrollo tecnológico que se le proponen debido a la baja eficiencia y elevados precios de los sistemas de comunicaciones convencionales.

Teniendo en cuenta las circunstancias mencionadas y las necesidades quienes determinan los medios de propagación de información adecuados, se puede revelar el impacto del sector ganadero y la cantidad de prácticas enlazadas a sistemas electrónicos que ayuden a su productividad, es por eso que se emplean condiciones especiales de espacio, temperatura, luz y humedad, donde su crianza es a base de cantidades equilibradas y enriquecidas que les permita crecer con mayor rapidez con el fin de obtener un producto de calidad para los consumidores en un corto lapso de tiempo. Para todas estas variables se requieren sistemas de control donde nace la necesidad de monitorizar, enviar y almacenar la información que permita así su control y vigilancia, además de proyecciones estadísticas de forma remota que facilite las labores del operario en su accionar ante los inconvenientes que puedan surgir en el cuidado de los bovinos.

¿Qué injerencia tendrá sobre la disponibilidad de la información pertinente a las actividades ganaderas en el sector rural, la implementación de una red mesh que permita conectar dispositivos IoT a la nube?

1.2. JUSTIFICACIÓN

El sector ganadero y el desarrollo de las IoT(internet de las cosas) es parte fundamental en el crecimiento económico y desarrollo tecnológico del país y es por esto que este proyecto desarrolla una herramienta de comunicación a largas distancias que puede ser la base para que a futuro se pueda favorecer estas actividades en los diferentes aspectos a tratar en el cuidado del ganado y las condiciones para el control de esta, asimismo para cualquier otra actividad de producción que necesite la trasferencia de información y no cuente con acceso a internet proponiendo el uso de la técnica de modulación LoRa que a diferencia de tecnologías conocidas como el Wi-Fi o el bluetooth plantea una comunicación eficiente y segura con un rango de alcance muy superior para aplicaciones de baja potencia. De todas las variables que se pueden transmitir, se orientó en la transmisión de la cantidad de alimento que necesitan los vacunos con datos recolectados y convertidos en un proceso anterior, a información que puedan ser enviados de manera digital a distancias considerables con el diseño de una red mesh auto gestionable a partir de módulos de radio LoRa los cuales son de bajo consumo, alta resistencia a las interferencias y tiempo prolongado de funcionamiento con baterías de reducida capacidad energética, de manera que entre ellos se decida le mejor ruta posible y la información llegue a su destino final. Se busca lograr una mayor interconexión y cobertura en zonas alejadas mediante la emisión de señales con dispositivos que pueden ser integrados en soluciones que permitan detección remota, transmisión de datos y monitoreo mediante tecnología miniaturizada que contribuya en la asistencia de las actividades relacionadas con la ganadería que mejoren su producción y por ende sus ganancias, además de contribuir al desarrollo tecnológico en el entorno de las IoT que pueda servir de asesoramiento y orientación de futuros proyectos en la institución relacionados con la interconexión digital de los dispositivos a través de la red que puedan disminuir la brecha digital en algunas regiones del país, en especial las rurales y sus actividades agropecuarias.

1.3. OBJETIVOS

1.3.1. OBJETIVO GENERAL

Implementar un radio enlace de largo alcance mediante una red mesh con dispositivos de tecnología LoRa y plataformas programables, para el monitoreo y gestión de producciones ganaderas ubicadas en el sector rural.

1.3.2. OBJETIVOS ESPECÍFICOS

- Caracterizar los componentes requeridos en la implementación de un sistema de comunicaciones inalámbrico para el monitoreo de las actividades propias del sector ganadero, mediante el uso de tecnologías IoT y redes mesh.

- Implementar un radio enlace de largo alcance con tecnología LoRa y sistemas programables para la realización de una red mesh que facilite la disposición de información en la nube de los datos pertinentes a la actividad ganadera.

- Validar la funcionalidad de la red implementada, mediante el estudio de casos resultantes de fallas generadas, que modifiquen los porcentajes de disponibilidad de información en la nube.

1.4. ESTADO DEL ARTE / ANTECEDENTES

PROTOTIPO DE UNA RED MESH CON PROTOCOLO DE ENRUTAMIENTO OLSR PARA LA UNIVERSIDAD PONTIFICIA BOLIVARIANA SECCIONAL BUCARAMANGA. (Sanmiguel & Garcia Prada, 2013).

En el cual se plantea un prototipo de una red mesh con protocolo de enrutamiento OLSR para analizar las ventajas de esta tipología de red con el uso de la tecnología Wi-Fi implementada en la universidad Pontificia Bolivariana de la ciudad de Bucaramanga para evaluar el desempeño y la cobertura al implementar 4 nodos, siguiendo cada paquete que transita a través de ella y así se pudo observar el comportamiento de la red mesh en disposición de topología de línea inalámbrica (formando una conexión punto a punto) y de malla completa(todas las cuatro estaciones tienen comunicación directa entre sí) que lograron su objetivo principal intercomunicando cada nodo con todos los demás.

DISPOSITIVO LoRa DE COMUNICACION A LARGO ALCANCE Y BAJO CONSUMO ENERGETICO PARA APLICACIONES DEL AMBITO DEL DESARROLLO. (Rodríguez, 2016).

El cual se basa en el desarrollo de estándares inalámbricos abiertos de bajo consumo energético y costo, además del largo alcance como respuesta a las tendencias y necesidades del mercado del IoT como lo es el estándar LoRaWAN es el adecuado en aplicaciones del ámbito del desarrollo debido a que el uso de la tecnología apropiada puede ser la clave en el fortalecimiento de los servicios básicos que respaldan redes de instalaciones como agua potable, comunicaciones, electricidad y saneamiento para determinar el consumo energético y distancia acorde con condiciones reales como espacios Indoor de obstáculos como infraestructuras, muros y espacio Outdoor de obstáculos como vegetación, junto con características de los dispositivos LoRa plasmadas en este documento con ejemplos de sistemas de monitorización y con sensores remotos. Por consecuencia se requiere utilizar fuentes de alimentación externas extras que aumentan el consumo energético del sistema como una de las desventajas en la ejecución del proyecto**.**

DISEÑO DE UNA RED MESH DE UAVS PARA PROPORCIONAR SERVICIOS DE COMUNICACIONES. (Limón de la Rosa, 2017).

Este consiste en una red mesh, compuesta por (UAVs / Vehículo aéreo no tripulado) que actuarán como nodos de la propia red y a su vez, como puntos de accesos, capaces de proporcionar servicios de comunicaciones vía IP a los clientes que se conecten a ellos. Al estar formada por UAVS que cambiarán de localización física constantemente, esta red tiene la peculiaridad de que poder adoptar distintas topologías de red de forma automática. Se ha logrado armar una infraestructura

robusta capaz de proporcionar servicios de comunicaciones a los clientes que se conecten a ella, cumpliendo con las necesidades que la caracterizan por tratarse de una red móvil sobre UAVs.

FORMULACIÓN DE UNA METODOLOGÍA PARA DISEÑAR E IMPLEMENTAR REDES MESH COMO ALTERNATIVA DE SOLUCIÓN PARA REDES COMUNITARIAS O RURALES; PROYECTO DE APOYO; CONSTRUCCIÓN DE UN ESQUEMA TECNOLÓGICO PARA PROTOCOLOS DE ENRUTAMIENTO EN REDES MESH. (Ruiz Parra & Blanco Garrido , 2014).

El cual consistió en como las tendencias de las redes de computadores hacen que se necesite la información al instante, lo cual está proyectado a que los avances deben dirigirse a respaldar las nuevas tecnologías siendo una alternativa de solución las redes inalámbricas para el acceso a los servicios por parte de los usuarios, es por esto que se implementan las redes mesh para el manejo de las redes Inalámbricas en la frecuencia de 2.4 GHz y 5 GHz. Los procesos requeridos para efectuar estas sofisticadas y complejas redes de comunicación, requiere de continuas innovaciones desde el salto de las redes cableadas a las redes inalámbricas, para facilitar el mejor desempeño de las comunicaciones en las actividades laborales y de entretenimiento. Se concluyó que las redes mesh con respecto a los diferentes tipos de redes de comunicaciones las ubican a futuro como una alternativa optima en la solución a problemas de interconexión y optimización del transporte de la información entre usuarios.

EVALUACIÓN DE LORA/LORAWAN PARA ESCENARIOS DE SMART CITY. (PEREZ, 2017).

En el cual se plantea un estudio sobre los beneficios y limitaciones de los protocolos de comunicaciones inalámbricas para sensores LoRa/LoRaWAN, considerando herramientas analíticas y experimentales que incluyen el proceso de implementación de una red de sensores en un campo real que utilizan esta tecnología y detallando los instrumentos, el software y la configuración utilizados para el funcionamiento de la red. Se mostraron los resultados obtenidos en las distintas pruebas realizadas en laboratorio junto a conclusiones con respecto a su correcto funcionamiento en un posible escenario real. Las pruebas comprenden la verificación de las distintas características que componen cada protocolo, la configuración de equipos y dispositivos para construir una red LoRa/LoRaWAN que permita almacenar los datos recogidos por los diferentes sensores que forman la red brindando la opción de acceder a ellos de forma remota y con cualquier dispositivo autorizado con conexión a Internet.

DESARROLLO DE UNA PASARELA LORA Y EVALUACIÓN DE PRESTACIONES (Carrion, 2017).

El cual consiste en como el internet de las cosas está orientado en la interconexión digital de objetos cotidianos con cualquier otro de su entorno, con el fin de conseguir que estos objetos sean más inteligentes que ofrecen conectividad inalámbrica de largo alcance a un gran número de dispositivos, con un mínimo consumo de energía y más económicos que las redes móviles, pero limitando el ancho de banda. Este proyecto consiste en un transmisor/receptor multicanal de alto rendimiento, capaz de recibir paquetes de diferentes dispositivos finales, enviar con diferentes elementos de propagación en hasta 8 canales en paralelo, además de demodular la señal LoRa sin conocer el factor de propagación utilizado por el nodo emisor. Como conclusión de las pruebas realizadas en este proyecto, este sistema no puede ser aplicado en redes que presenten requerimientos altos de velocidad de transmisión de datos como por ejemplo la video vigilancia o la automatización industrial por sus requisitos de baja latencia, pero si puede ser utilizada en aplicaciones que conlleven un número reducido de mensajes como es la agricultura y smart city.

PROTOTIPO DE SOLUCIÓN IoT CON TECNOLOGÍA "LoRa" EN MONITOREO DE CULTIVOS AGRÍCOLAS. (Triana & Rodríguez, 2018).

Esta monografía indica el proceso teórico-práctico de un prototipo de solución IoT para el monitoreo de cultivos agrícolas, mediante "LoRa", el cual cuenta con un bajo consumo de energía y larga distancia (mayor a 8Km en línea de vista) entre el dispositivo transmisor y el dispositivo receptor, para conocer las variables físicas(temperatura, humedad, radiación y PH) que afectan directamente al proceso de un cultivo a través de un aplicativo web que posee una interface agradable al usuario permitiendo visualizar la información de los sensores de cada nodo, estadísticas en tiempo real y alertas cuando las medidas no están dentro de los parámetros que se pueden establecer mediante el mismo aplicativo. La implementación del prototipo ha optimizado tiempos y redujo pérdidas generadas ya sea por el daño o baja calidad de los productos.

MODELO DE IMPLEMENTACIÓN DE PROTOCOLO OPEN SSL PARA EL MANEJO DE LA SEGURIDAD EN INFRAESTRUCTURA DE REDES MESH (MORENO SANTOS, 2015) .

El cual se basó en buscar la forma de tener la información al instante, esta necesidad hace que la tendencia de las redes de comunicaciones se vuelva totalmente inalámbrica en el futuro, por la facilidad de conexión que esta tiene ya sea mediante un Smartphone, Tablet, Laptop, etc.; aquí aparece las redes inalámbricas mesh, una tecnología sofisticada y compleja, con lo cual facilita un mejor desempeño para la transmisión de datos. Al momento de tener una conexión a estas redes inalámbricas

(sea en el trabajo, hogar y/o comunitaria), la información que se transmite o recibe puede ser interceptada por una o varias personas que estén conectadas a esta red, por lo que se requiere de seguridad para proteger la información, mediante protocolos de enrutamiento para la seguridad. El proyecto realizado contribuye de manera importante de una seguridad mayor en redes mesh comunitarias, mostrando los puntos más importantes a la hora de llevar acabo la implementación del protocolo Open SSL.

DISEÑO DE UN MODELO DE INFRAESTRUCTURA PARA REDES MESH EN ENTORNOS COMUNITARIOS O RURALES DE COLOMBIA: COMO APOYO AL PROYECTO DE INVESTIGACIÓN "FORMULACIÓN DE UNA METODOLOGÍA PARA DISEÑAR E IMPLEMENTAR REDES MESH COMO ALTERNATIVA DE SOLUCIÓN PARA REDES COMUNITARIAS O RURALES". (Cruz Rozo & Collazos Collazos, 2016).

El cual nos indica que la tecnología no discrimina razas, clase social o ubicación geográfica, todos nosotros como usuarios exigimos un buen servicio rápido y eficiente a precios asequibles. Para estos requerimientos la tecnología avanza día a día con el fin de satisfacer a los usuarios de una red desarrollándose una topología de red la cual se conoce como mesh, creada, administrada, gestionada por los propios usuarios y ofreciendo acceso libre, además de gratuito a ella, implementado en diferentes países del mundo arrojando muy buenos resultados y por esto se busca implementar en algunas en zonas comunitarias o rurales de Colombia. Se logró capturar tráfico de la red simulando los datos de una implementación en un entorno real identificando el comportamiento de la red y su posible alcance.

IMPLEMENTACIÓN DE UNA RED DE SENSORES INALÁMBRICOS LPWAN MEDIANTE MÓDULOS LORA PARA EL MONITOREO DE LA CALIDAD DEL AGUA EN 2 RÍOS. (Burbano, 2017).

El cual consiste en la implementación de una red inalámbrica para medir parámetros de calidad del agua en dos ríos, utilizando elementos de bajo coste como lo son los módulos LoRa para la comunicación entre pares, la cual nace de redes de sensores inalámbricos y el IoT (Internet de las Cosas). Los módulos LoRa consumen poca energía a largas distancias a muy bajo precio, razón que los hacen una tecnología adecuada para cumplir con los planes de este proyecto. Construyeron 2 nodos sensores los cuales monitorean parámetros como la Conductividad eléctrica, pH, Temperatura. Por lo que se hace la adquisición, diseño y construcción de los sensores y sus circuitos de acondicionamiento, además se realizara un nodo recibidor que estará conectado a una computadora y presentara los resultados en una interfaz gráfica en Java. Como resultados se obtuvieron problemas con el protocolo escogido

para la ejecución del proyecto, el cual produjo lentitud en el sistema, además de varios ajustes de impedancias, ganancias de las antenas y potencia en la transmisión.

IMPLANTACIÓN DE UN MODELO DE RED ABIERTA TIPO MESH PARA PROPAGAR EL ACCESO LIBRE AL SERVICIO DE BIBLIOTECA Y CONTENIDOS ABIERTOS (Ardila, 2014).

Implantación de un modelo de red abierta tipo mesh para propagar el acceso libre al servicio de biblioteca y contenidos abiertos. Con la implementación de la red abierta tipo mesh en la ciudad de Duitama se abre la puerta a la conectividad libre e independiente para las personas que por motivos económicos o sociales así lo requieran. Ahora que para las personas que ven la innovación tecnológica través de las herramientas libres y son curiosos de las redes, podrán interaccionar y ser parte del proyecto en para sus comunidades locales.

RED MESH LoRa SÍNCRONA PARA EL MONITOREO DE PROCESOS EN INFRAESTRUCTURA SUBTERRÁNEA. (EBI, SCHALTEGGER, RÜST, & BLUMENSAAT, 2019).

El cual consiste en recopilar información precisa en tiempo real sobre el rendimiento del sistema de drenaje urbano para predecir y gestionar situaciones críticas de carga, como inundaciones repentinas urbanas y desbordamientos de alcantarillas.

Aunque las nuevas técnicas de comunicación inalámbrica de baja potencia permiten transferencias de datos eficientes, el rendimiento sobre el suelo, para aplicaciones subterráneas o interiores en un amplio rango de cobertura es difícil lograr debido a limitaciones físicas y topológicas, particularmente en áreas urbanas densas. En este Trabajo se discutió primero las limitaciones de rango del estándar LoRaWAN basado en una evaluación sistemática de una operación a largo plazo de una red de sensores que monitorea la dinámica del proceso en el alcantarillado. Los análisis revelan una pérdida de paquetes de datos cinco veces mayor para los nodos subterráneos, que crece constantemente con el aumento distancia a la puerta de entrada.

Los resultados de la prueba muestran que la sincronización de la red mesh LoRa supera claramente la técnica estándar de LoRaWAN con respecto a la fiabilidad de la entrega de paquetes cuando se transmite desde ubicaciones de rango crítico.

RED DE SENSORES PARA MONITORIZACIÓN INTELIGENTE DE VIVIENDA. (Arcenegui, 2016).

El proyecto está diseñado para medir las principales variables relacionadas con la eficiencia energética, como son la humedad, la temperatura y la luminosidad, mediante una red de sensores. También funciona como una alarma de seguridad que avisa al administrador cuando se detecte una persona no autorizada. Por esta razón los nodos incorporaron un detector de presencia y al menos uno de los nodos tuvo un

sistema de identificación. La red es escalable, para así poder agregar tantos nodos como sean necesarios. Finalmente, Los datos obtenidos por estos sensores fueron mostrados mediante una interfaz sencilla tanto en ordenadores como en dispositivos móviles que permitió la administración de los diferentes elementos de la red mediante una interfaz específica para esta función, que debe estar protegida con usuario y contraseña ya que solo el administrador de la red debe tener acceso a esta.

REDES INALÁMBRICAS MESH: UN ESTUDIO ORIENTADO A LA PERSPECTIVA DE LAS IOT MÁS RELEVANTES. (Cilfone, Davoli, Belli, & Ferrari, 2019).

Este documento proporciona una encuesta exhaustiva de las siguientes tecnologías inalámbricas relevantes: IEEE 802.11, Bluetooth, orientado a IEEE 802.15.4 y LoRa basado en Sub-GHz. Su objetivo es resaltar cómo varias tecnologías de comunicación pueden ser adecuadas para redes mesh, ya sea proporcionando un soporte nativo o siendo adaptado posteriormente.

Por lo tanto, se discutió cómo estas tecnologías inalámbricas, ya sean estándar o propietarias, pueden adaptarse a escenarios de IoT (por ejemplo, ciudades inteligentes y agricultura inteligente) en los que la heterogeneidad de los dispositivos involucrados es una característica clave. Finalmente se proporcionaron casos de uso de referencia que involucran a todos los analizados. Una interesante conclusión es que las tecnologías consideradas, tanto estándar como patentadas, bien adaptado a escenarios en los que la heterogeneidad de los dispositivos que componen la malla es imprescindible en los entornos de IoT. En este caso, una "red de redes" es lo mejor definición que se puede usar y que representa mejor la variedad de comunicaciones, que van desde de corto a largo alcance, que se puede encontrar en escenarios modernos, como en Smart city y escenarios Smart de agricultura.

IMPLEMENTACION DE UN GATEWAY LOW-COST PARA EL PROTOCOLO LoRa. (Chacon & Campos Ramirez, 2018).

Esta monografía propone investigar, diseñar e implementar un Gateway haciendo uso de un sistema embebido que permite reducir considerablemente el costo de producción y venta de tal dispositivo, teniendo en cuenta las bandas de frecuencia libres en Colombia, para el desarrollo del Gateway basado en el protocolo se usó un prototipo funcional y de bajo costo usando los módulos LoRa RN2903 de Microchip disponibles en el grupo de investigación LASER (Laboratorio en Automatización Sistemas Embebidos y Robótica) de la Universidad Distrital y con el módulo de procesamiento embebido RaspberryPi. Se concluyó que se debe considerar que entre más grande sea la distancia entre el nodo y el Gateway el SF y la sensibilidad será mayor, y menor la velocidad de transmisión.

TELEMETRÍA DE CONTENEDORES DE RESIDUOS. (Horovitz & Mayobre, 2018).
El cual explica cómo se construyó un prototipo para realizar telemetría de contenedores de residuos distribuidos sobre una ciudad. Se describe cómo se implementó una solución basada en Arduino, sensores y transmisión de datos mediante LoRa (LPWAN) e Internet,
Se contempló la transmisión de distintas variables, incluyendo las coordenadas (GPS), si el contenedor se está incendiando (sensor de temperatura) y es enviada a un punto central del sistema ("Gateway") y este último lo envía a un servidor de aplicaciones para su análisis y toma de decisiones. Los alcances obtenidos en las pruebas no llegaron a lo esperado (2 km). Sin embargo, por causas como lo pueden ser la antena, ubicación del Gateway, equipos económicos y básicos para prototipos con limitaciones, entre otras.

REDES MESH, UNA ALTERNATIVA A PROBLEMAS DE COBERTURA DE RED: UNA REVISIÓN DE LITERATURA. (Bautista, Sanchez , & Portillo , 2014).
Este artículo aborda una revisión de los proyectos, artículos y trabajos que hacen referencia a las redes mesh. Habla además de los diferentes protocolos de enrutamiento y estándares, los cuales permiten optimizar rutas para el intercambio de información en la red y donde se especifican normas sobre el funcionamiento en una WLAN respectivamente. Se concluye que las redes mesh es una solución a los diferentes problemas presentes en las redes WLAN y diferentes topologías o estándares usados en la actualidad, que abre la posibilidad a una eficiencia mucho mayor si se habla de conectividad tanto en sectores rurales como urbanos.

IMPLANTACIÓN DE UN MODELO DE RED ABIERTA TIPO MESH PARA PROPAGAR EL ACCESO LIBRE AL SERVICIO DE BIBLIOTECA Y CONTENIDOS ABIERTOS. (Martinez, 2014).
El cual consta de una propuesta para implementar un modelo "MESH" de red libre en la ciudad de Duitama y auspiciada por el Colegio Seminario Diocesano, donde se selecciona la arquitectura y el diseño a usar, el protocolo escogido y la implementación de diferentes servicios como el firewall, Wi-Fidog, koha, dando como resultado el acceso a los servicios y recursos de la Red mesh donde Los usuarios podrán acceder libremente a internet únicamente con fines académicos, de investigación, técnicos y de administración propios de la institución.

DISENO DE UNA RED INALAMBRICA MESH EN EL, CAMPUS DE LA UNIVERSIDAD NACIONAL DE CALLAO PARA PROVEER SERVICIOS DE, INTERNET INALÁMBRICO (Palomino, 2014).
El cual tiene como propósito ofrecer una solución tecnológica para ofrecer servicios de Internet inalámbrico a los usuario que se encuentra dentro del campus de la

Universidad Nacional del Callao el cual estará basado en redes mesh y utilizara la tecnología WI-FI para unificar y centralizar las redes inalámbricas de todas las facultades para una mejor administración y calidad de servicio pudiendo así el usuario movilizarse dentro de las instalaciones sin perder la señal donde cada nodo dentro de la red mesh es capaz de tomar decisiones de trazado rutas independiente de los demás nodos, permitiendo un despliegue rentable y seguro dentro del campus en el cual dio como resultado la mejora en la solución ante los posibles problemas presentes en la red de la universidad y la eficiencia de esta en diferentes áreas de cobertura.

ESTUDIO Y DISEÑO PARA LA IMPLEMENTACION DE UNA RED WIRELESS MESH CISCO EN LA ZONA AFORO DE CONTECON. (ARIAS, 2014).
Este estudio consiste en como dimensionar una red mesh con equipos Cisco, detallando diferentes protocolos y entandares de redes inalámbricas existentes, además del diseño y los elementos que hacen parte de este tipo de red, sus alcances y limitaciones para ser implementados finalmente en el área de una empresa, adecuándola según las necesidades de esta, gestionando y monitoreando el tráfico de red de la compañía. Se concluye que se cumplió su objetivo principal en la cobertura y diseño de la red propuesta con el desarrollo de una solución de alta disponibilidad de la información y con una administración centralizada de todos los Access Point instalados.

1.5. MARCOS REFERENCIALES

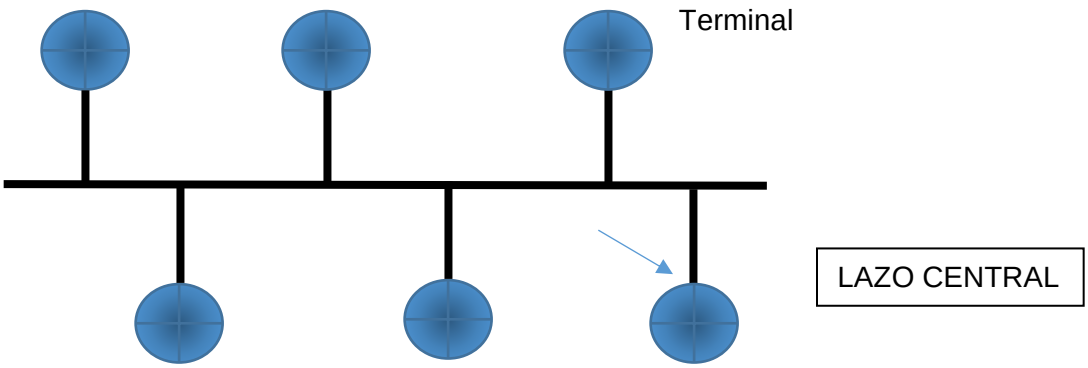
1.5.1. MARCO TEORICO

1.5.1.1 Topologías de red

Con Topología hablamos de la forma en que está diseñada la red que se utilizara según el caso, sea físicamente como las características en su hardware o lógicamente como las características internas de su software, también se puede entender como la representación geométrica de la relación entre todos los enlaces y los elementos que los conectan entre sí. (BARRAGAN, 2012)

Red tipo Bus

Ilustración 1. Red tipo Bus



Fuente: Autor

Cada nodo está conectado a un segmento común de cable mediante un lazo central. Todas las computadoras o dispositivos están unidos a este lazo.

VENTAJAS

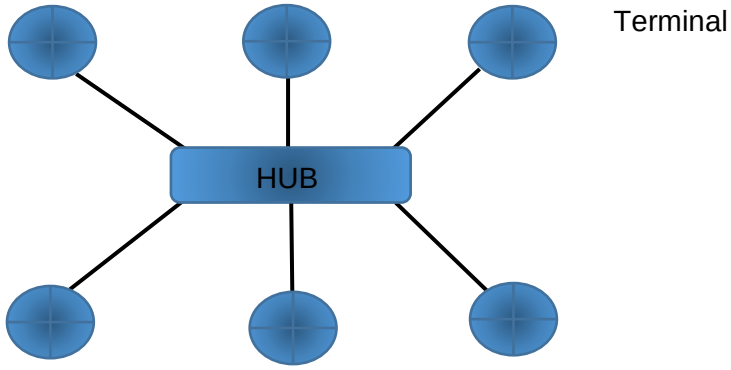
- sencillez en la instalación.
- La topología bus usa menos cable que una malla, una estrella o una topología en árbol.
- Es fácil de conectar nuevas estaciones de trabajo.

DESVENTAJAS

- Hay un límite de equipos dependiendo de la calidad de la señal.
- Complejidad de reconfiguración y aislamiento de fallos.
- Limitación de las longitudes físicas del canal.
- Toda la red se caería si hubiera un fallo en el cable principal.

Red tipo Estrella

Ilustración 2. Red tipo Bus



Fuente: Autor

Todas las computadoras se conectan a un hub o concentrador que controla y realiza todas las funciones de red además de actuar como amplificador de los datos.

VENTAJAS

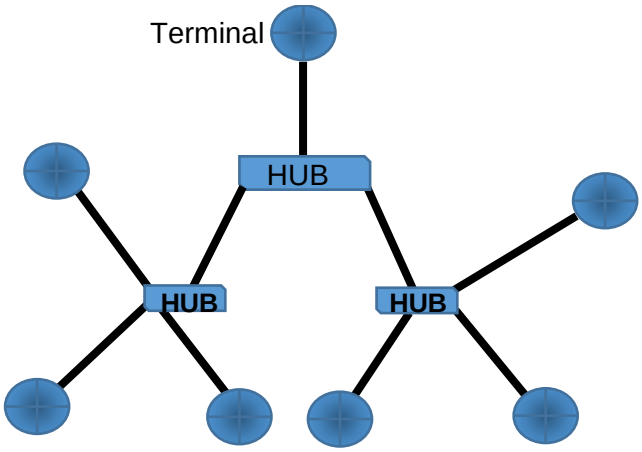
- En una red de estrella, cada dispositivo necesita únicamente un enlace y un puerto de entrada/salida para conectarse a cualquier número de dispositivos.
- Fácil de instalar y reconfigurar.
- Facilidad para la detección de fallos y su reparación.

DESVENTAJAS

- Requiere más cable que la topología de bus.
- Un fallo en el hub provoca el aislamiento de todos los nodos conectados a él.

Red tipo Árbol

Ilustración 3. Red tipo árbol



Fuente: Autor

El controlador central del árbol es un amplificador que a su vez es un repetidor, es decir, un dispositivo que restaura los patrones de bits recibidos antes de retransmitidos.

VENTAJAS

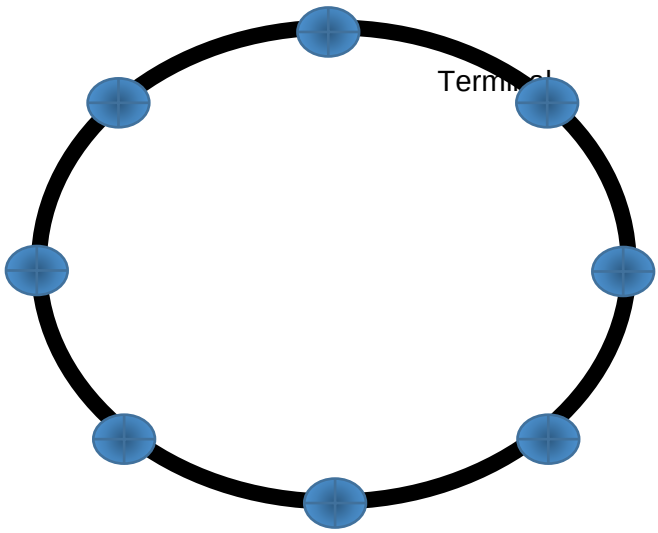
- Si un nodo falla, no se presentan problemas entre los nodos subsiguientes.
- Enlace punto a punto para segmentos individuales.
- Retransmitir las señales de esta forma amplifica su potencia e incrementa la distancia a la que puede viajar la señal.

DESVENTAJAS

- Si falla el concentrador principal todos los demás también se afectan.
- Difícil de configurar.
- Se requiere mucho cable.

Topología tipo anillo

Ilustración 4. Red tipo anillo



Fuente: Autor

Los dispositivos están conectados entre sí en forma de un anillo, es decir, forman un círculo entre ellas.

Ventajas

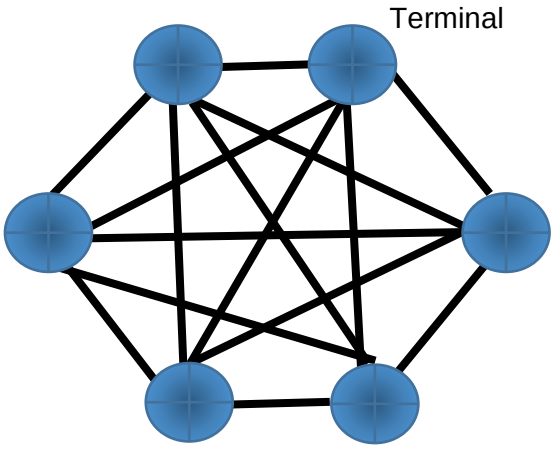
- Permite comprobar si la información fue recibida.
- Sencillez en la arquitectura y fluidez.
- El sistema provee un acceso equitativo para todas las computadoras.

Desventajas

- Lentitud.
- Difícil de diagnosticar y reparar los problemas.
- Si un enlace deja de funcionar se cae la red. (Blogger.com, 2016)

Topología Tipo Malla (Mesh)

Ilustración 5. Red tipo mesh



Fuente: Autor

La topología de malla (mesh), fue la escogida para este proyecto debido a que utiliza varios enlaces entre dispositivos de la red, así como una estrategia o dinamismo de tolerancia a fallas. Cada dispositivo en la red está conectado a todos los demás, o a varios para así en caso de que un enlace falle, este escoja automáticamente otro camino y así la red no se interrumpa. Este tipo de tecnología requiere mucho cable cuando es por medio de este, pero puede ser inalámbrico también. Las redes de malla son más difíciles y costosas para instalar que las otras topologías de red debido al gran número de conexiones requeridas. (Redessobreinformatica, 2012)

Ventajas de la topología mesh

- Gestiona grandes cantidades de tráfico, debido a sus múltiples dispositivos que pueden transmitir datos simultáneamente.
- Una falla de un dispositivo no causa una interrupción en la red o la transmisión de datos.
- Agregar dispositivos adicionales no interrumpe la transmisión de datos entre otros dispositivos.
- Se puede extender tanto como queramos o se necesite.

Desventajas de la topología mesh

- El costo de implementación es más alto que otras topologías de red, por lo que es una opción menos deseable cuando de precios hablamos, debido a que se utilizan más dispositivos en la red.

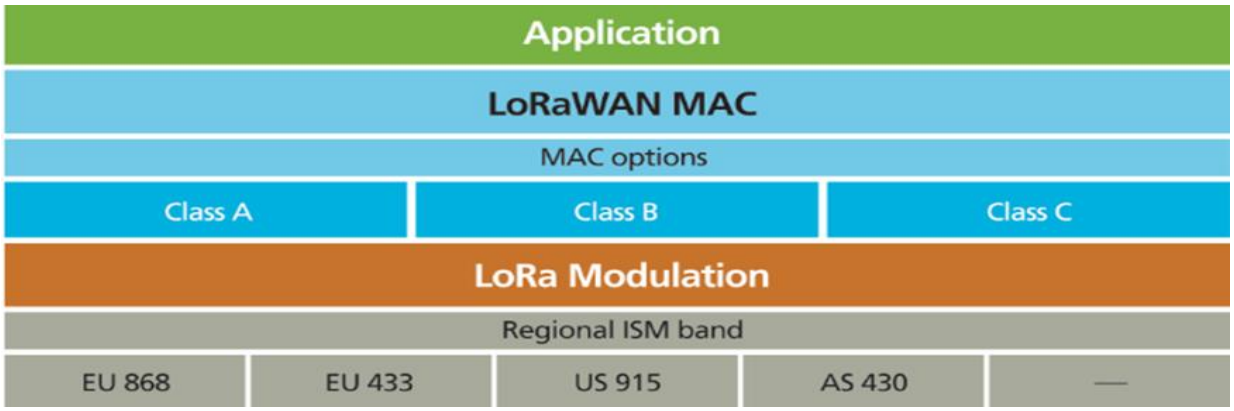
- Construir y mantener la red es difícil y requiere mucho tiempo.

1.5.1.2 LoRa y LoRaWAN

LoRa es una abreviatura de (Long Range) lo que traduce a Largo Alcance en español. Es una técnica de modulación de espectro extendido, también conocida como espectro ensanchado o chirp (CSS). Los dispositivos LoRa fueron desarrollados por la empresa norteamericana Semtech quienes conservan su patente. La tecnología de radiofrecuencia inalámbrica (LoRa Technology) es una plataforma inalámbrica de largo alcance y baja potencia que se ha convertido en pieza importante para las redes de Internet de las cosas (IoT) a nivel mundial. La tecnología LoRa y el protocolo abierto LoRaWAN permiten aplicaciones de IoT inteligentes que resuelven algunos de los mayores desafíos que enfrenta nuestro planeta: gestión de energía, reducción de recursos naturales, control de contaminación, eficiencia de infraestructura, prevención de desastres y más. La tecnología LoRa de Semtech ha acumulado varios cientos de casos de usos conocidos para ciudades inteligentes, hogares y edificios inteligentes, agricultura inteligente, medición inteligente, cadena de suministro inteligente y logística, y más. Con 105 millones de dispositivos conectados a redes en 100 países y creciendo, LoRa Technology es el ADN de IoT, creando un planeta más inteligente.

LoRaWAN es una estructura de capas de red, semejante al modelo OSI de redes el cual es un modelo de estandarización para los protocolos de la red de arquitectura en capas, LoRaWAN es una capa que corresponde a una capa similar a la capa de enlace estando por encima de la capa LoRa, equivalente a una capa física en OSI. La alianza abierta LoRaWAN es un protocolo de red de área amplia (LPWAN) de baja potencia, apoyado en la tecnología LoRa. Diseñado para conectar de manera inalámbrica dispositivos con baterías a Internet en redes regionales, nacionales o globales, el protocolo LoRaWAN aprovecha el espectro de radio sin licencia en la banda Industrial, Científica y Médica (ISM) el cual está por debajo de 1 GHZ. La especificación define el dispositivo a la infraestructura de los parámetros de la capa física LoRa y el protocolo LoRaWAN, y proporciona una interconexión perfecta entre los dispositivos. (SEMTECH, 2019).

Ilustración 6. Protocolo LoRaWAN



Fuente: https://www.semtech.com/lora/what-is-lora

Historia y patente LoRa

LoRa es una tecnología de transferencia inalámbrica, desarrollada entre 2008 y 2013 en Francia y patentada por la compañía Semtech Corporation, quien es un proveedor líder de semiconductores analógicos y de señales mixtas de alto rendimiento y algoritmos avanzados para consumidores finales, informática empresarial y comunicaciones con casi 60 años de experiencia diseñando y fabricando plataformas de innovación, tamaño, eficiencia, y alcance, además de productos semiconductores utilizados por los fabricantes de equipos originales y sus proveedores para automóviles, equipos de transmisión, centros de datos, industria, Internet de las cosas (IoT), televisores LCD, teléfonos inteligentes, tabletas, dispositivos portátiles y Aplicaciones de infraestructura inalámbrica.

Semtech ha evolucionado pasando de ser un fabricante de productos especializados de alta confiabilidad para aplicaciones militares, hasta un proveedor de soluciones de semiconductores integrales con una cartera de productos diversificada. Además, cuenta con un equipo técnico altamente calificado para desarrollar nuevos productos que resuelven los desafíos de diseño más complejos de sus clientes y se enfocan en sectores de rápido crecimiento tecnológico.

Con el paso de los años se creó LoRa Alliance que es una asociación abierta, sin fines de lucro, que ha crecido a más de 500 miembros desde su creación en marzo de 2015, convirtiéndose en la alianza más grande y de mayor crecimiento en el sector de la tecnología. Sus miembros colaboran estrechamente y comparten experiencias para promover e impulsar el éxito de sus diferentes protocolos para la flexibilidad técnica que

abordan una amplia gama de aplicaciones IoT, tanto estáticas como móviles con una amplia expansión continua en 2019 y más.

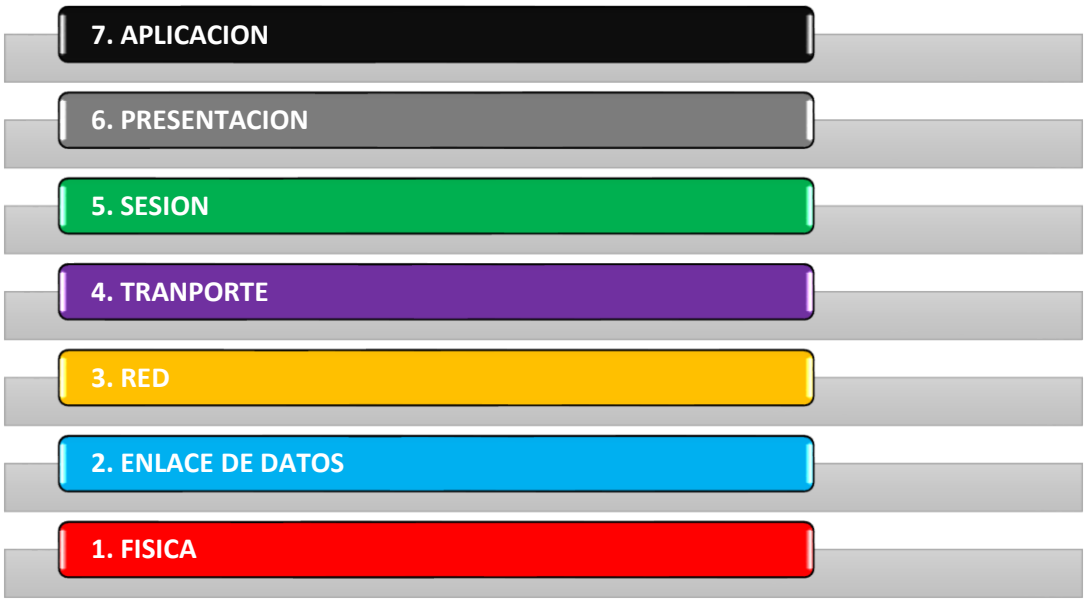
LoRa (Ventajas frente a otras tecnologías similares)

Entre las principales ventajas de la tecnología LoRa frente a otras tecnologías, las que más resaltan son la Geolocalización, que permite aplicaciones de rastreo de baja potencia y sin necesidad de GPS, otra ventaja y que lo hace muy asequible es su bajo costo frente a otras tecnologías de comunicación actuales ya que ofrece una solución económica en infraestructura, gastos operativos y sensores de nodos repetidores y finales. Además de todo esto ofrece interfaces estandarizadas totalmente conocidas globalmente que facilitan la implementación de redes basadas en mesh, LoRaWAN y aplicaciones IoT. La tecnología LoRa también ofrece un bajo consumo ya que nos brinda un protocolo diseñado específicamente para poco gasto energético que extiende la vida útil de la batería hasta 20 años a pesar de su largo alcance en el cual una única estación base suministra una profunda inserción de zonas cerradas y pobladas, además permite la transmisión a distancias muy largas que conectan a las zonas rurales de manera segura con cifrado de datos de extremo a extremo integrado.

1.5.1.3 MODELO OSI

Este modelo nace ante la necesidad de interconectar sistemas de distintas procedencias en el mundo de las telecomunicaciones, para que estos pudieran intercambiar información debido a que cada uno manejaba sus propios protocolos según el fabricante, por lo tanto, se desarrolló este estándar que es utilizado a nivel mundial conformado por 7 capas, cada uno con sus propias funciones para que en conjunto sean capaz de alcanzar su objetivo final el cual es que todos puedan comunicarse y entenderse. (BARBOSA, 2015).

Ilustración 7. Modelo OSI con sus 7 capas



Fuente: Autor

CAPA FISICA

Se refiere al medio físico de transmisión que define la transmisión de bits a través de un canal, es decir el medio por donde viaja la información: el aire (radiofrecuencia), fibra óptica, cable de par trenzado, guías de onda, cable coaxial, etc. Además, también define la topología a usar como la forma en que se transmite los datos.

CAPA DE ENLACE DE DATOS

Esta segunda capa da lugar al direccionamiento físico y control de flujo de la información, a diferencia de la capa física que se encarga de enviar y recibir los datos, la capa de enlace mantiene el control de la información y asimismo comprueba si los datos recibidos son correctos.

CAPA DE RED

A diferencia de la capa de enlace de datos que se encarga del direccionamiento físico, la capa de red se encarga del direccionamiento lógico, identificando el enrutamiento existente entre una o más redes y determinando la ruta hasta el receptor final de los paquetes transmitidos.

CAPA DE TRANSPORTE

Dentro de los paquetes transmitidos que se hablan de la capa anterior, esta capa se encarga del transporte de los datos dentro de estos paquetes, desde su origen hasta su destino final asegurándose que lleguen de manera eficiente y correcta, además define el protocolo a trabajar mediante el control de los errores y servicio de confirmación de la recepción de paquetes (TCP) o sin todos los controles de error y recepción de paquetes (UDP).

CAPA DE SESION

Establece una sesión entre dos o más máquinas, manteniendo y controlando el enlace entre los dispositivos que están transmitiendo, es decir se ocupa de la sincronización entre host mediante protocolos como: SMTP, FTP, SAP, SSH, ZIP, RCP, SCP, NetBIOS, ASP, entre otros.

CAPA DE PRESENTACION

Se objetivo es la representación de la información transmitida, es decir que la información se pueda enviar de manera de que el receptor la pueda entender o reconocer en aspectos como la semántica y la sintaxis de los datos transmitidos a pesar de los diferentes protocolos utilizados trabajando con el contenido útil que nosotros queremos ver.

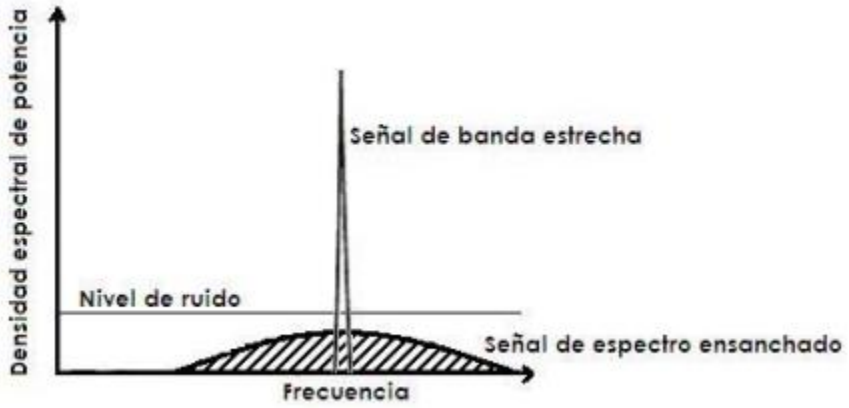
CAPA DE APLICACIÓN

Esta última capa es el más cercano al usuario final. El usuario no interactúa directamente con esta capa, sino, con herramientas que interactúan con el nivel de aplicación haciéndolo más accesible y sirviendo como ventana a los usuarios para acceder a los servicios de red permitiéndole ejecutar acciones y comandos.

MODULACION DE ESPECTRO ENSANCHADO

La modulación SS de “spread spectrum “que su sigla en español significa “espectro ensanchado” es la tecnología de modulación usado por los módulos LoRa, es la modulación más usada de espectro expandido. Esta técnica cambia la fase de la portadora siguiendo una secuencia determinada por un código aleatorio, para lograr esto se hace una multiplicación entre el código y el mensaje que después se montan en la portadora. Para la demodulación solo se multiplica la señal recibida por el mismo código recuperando así el mensaje original.

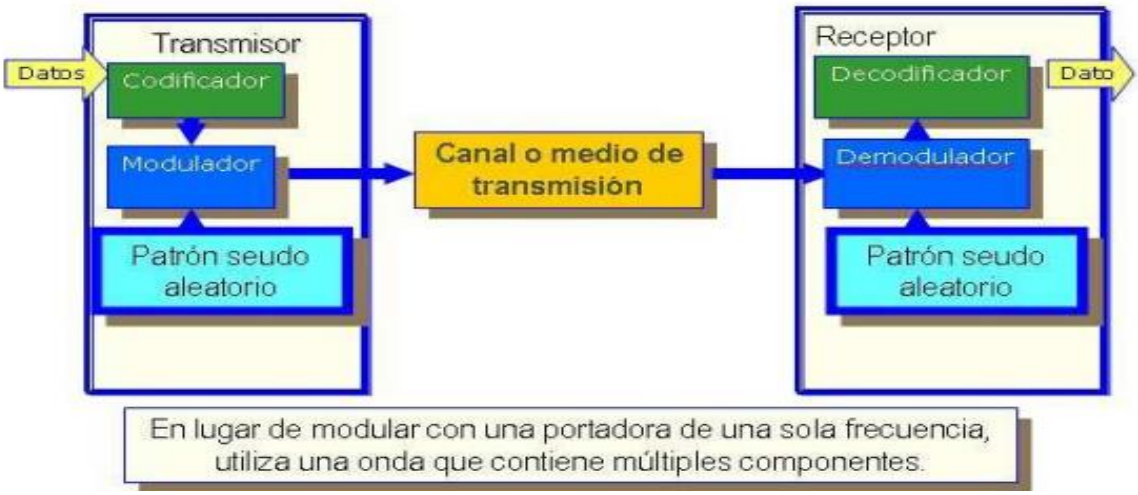
Ilustración 8. Comparación de espectro ocupado por una señal se espectro ensanchado y una señal modulada de banda estrecha.



Fuente: https://www.redalyc.org/pdf/342/34283003.pdf

La señal transmitida es propagada en una banda de frecuencia amplia, mucho más del ancho de banda mínimo requerido para transmitir el mensaje que será enviado. El ensanchamiento se realiza en base a una señal pseudo aleatoria, que se caracteriza por tener una forma de ruido y esta señal enviada tendrá características pseudo aleatorias y solo se podrá demodular si el receptor es capaz de generar la misma señal pseudo aleatoria generada por el trasmisor por esto es importante que tengas los mismos parámetros y configuración. (EPN, 2019)

Ilustración 9. Esquema de comunicación usado por la modulación de espectro ensanchado con sus componentes principales.



Fuente: https://bibdigital.epn.edu.ec/bitstream/15000/83/1/CD-0055.pdf

TIPOS DE ANTENA

Aunque existen varios tipos de antena y configuración para el desarrollo del trabajo de grado se usaron solo las antenas Helicoidales y las Dipolo, expuestas a continuación:

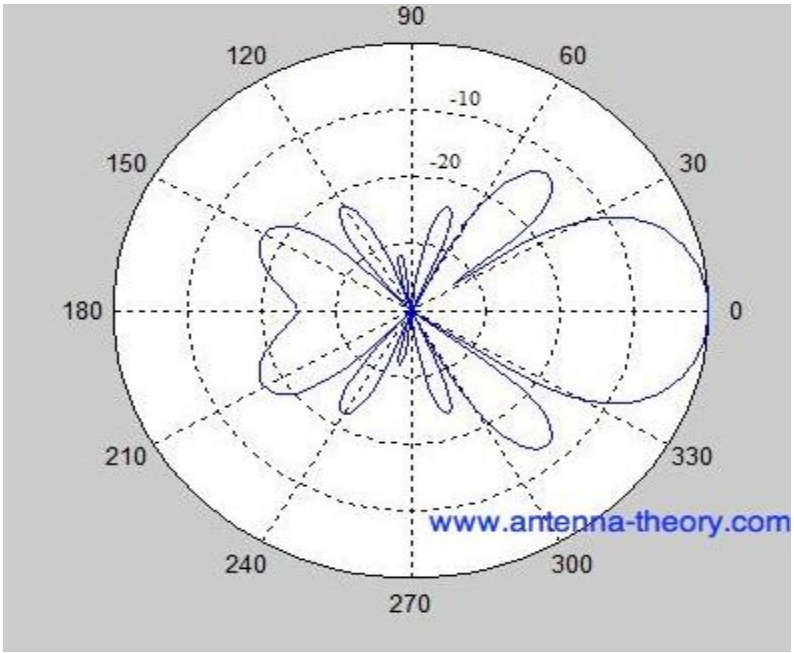
ANTENA HELICOIDAL

Ilustración 10.Antena Helicoidal que viene junto con el módulo Lora Ra-01

Fuente: Autor

La antena helicoidal o antena hélice está formada por un alambre conductor enrollado en forma espiral. Es una evolución del mono polo vertical, en la cual el mono polo ha sido reformado para tomar la forma de un solenoide. Un solenoide es cualquier dispositivo físico capaz de crear una zona de campo magnético uniforme. Al tratarse de un mono polo enrollado en forma espiral, también cuenta con la ventaja de reducir el tamaño de la antena. Este tipo de antenas es ampliamente utilizado, otras son utilizadas en UHF y VHF. (Rincon, 2011).

Ilustración 11. Patrón de radiación antena Helicoidal



Fuente: http://www.antenna-theory.com/spanish/antennas/travelling/helix.php

La aplicación de esta variedad de antena se destina a comunicaciones móviles o lejanas de frecuencia baja. Sus propiedades y parámetros principales son:

Alta directividad.

Ancho de banda amplio.

Polarización: Elíptica o circular.

Ganancia: 4´76 dB.

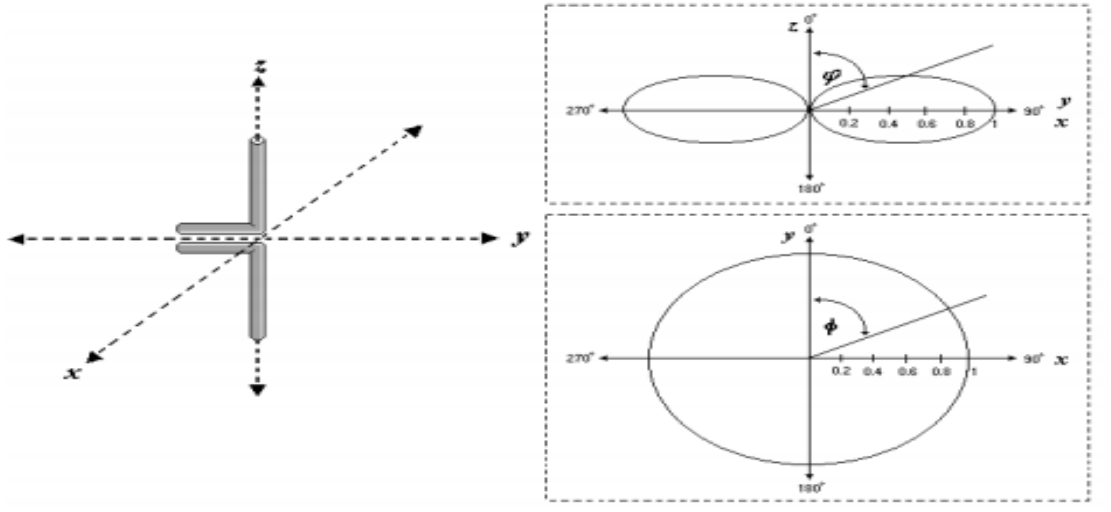
Ancho de banda: 5%

Dimensiones pequeñas.

ANTENA DIPOLO

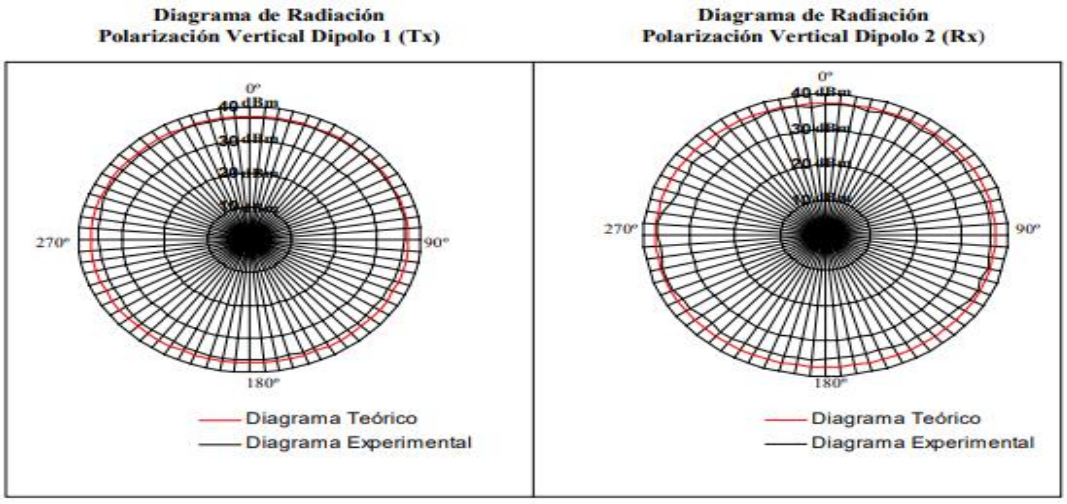
Es una antena con dos polos simétricos mirando en sentidos contrarios, debido a que cada polo se comporta como si fuera un tramo de línea de transmisión. Esta antena puede situarse de manera horizontal o vertical con respecto a la superficie terrestre dependiendo de los requerimientos de los servicios, su patrón de radiación es omnidireccional y es utilizado en su mayoría para aplicaciones como comunicaciones móviles. (TEST AMERICA, 2019)

Ilustración 12. Campo de radiación de radiación del dipolo para polarización vertical y horizontal

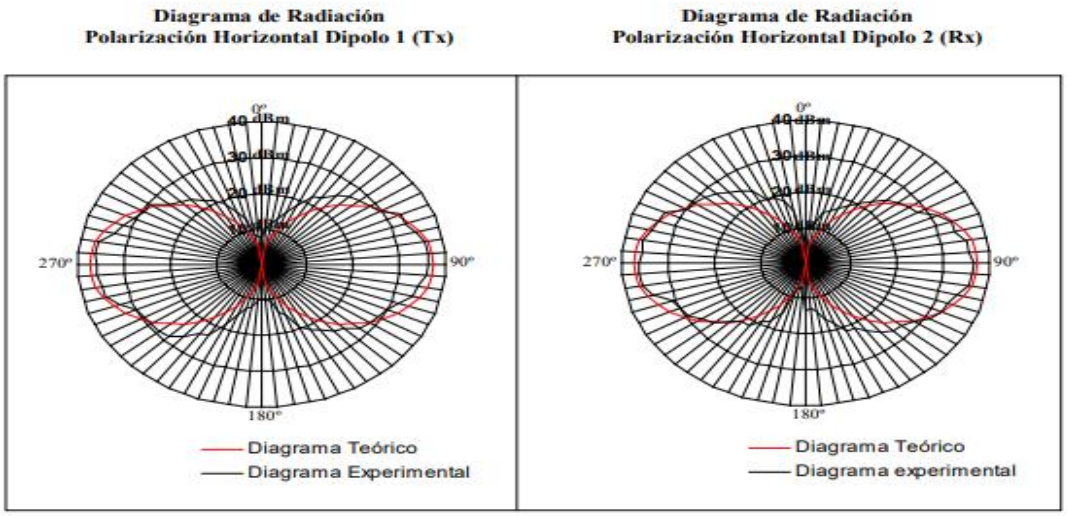


Fuente: http://www2.elo.utfsm.cl/~elo352/biblio/antenas/cap2aflores.pdf

Ilustración 13.Diagramas de radiación según polaridad



Diagramas de Radiación para polarización vertical



Diagramas de radicación para polarización Horizontal

Fuente: http://www2.elo.utfsm.cl/~elo352/biblio/antenas/cap2aflores.pdf

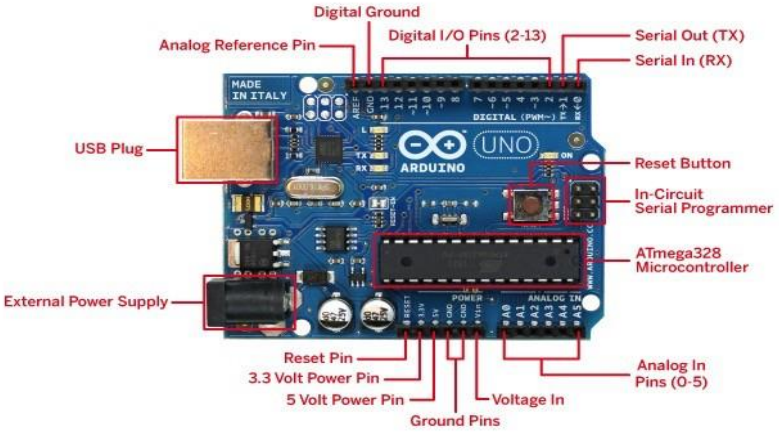
1.5.2. MARCO CONCEPTUAL

1.5.2.1 ARDUINO UNO

Arduino es una plataforma de hardware y software de código abierto, basada en una sencilla placa con entradas y salidas, analógicas y digitales, en un entorno de desarrollo que está basado en el lenguaje de programación Processing. Es decir, una plataforma de código abierto para prototipos electrónicos.

Al ser open source, tanto su diseño como su distribución, puede utilizarse libremente para el desarrollo de cualquier tipo de proyecto sin necesidad de licencia. (electronics, s.f.)

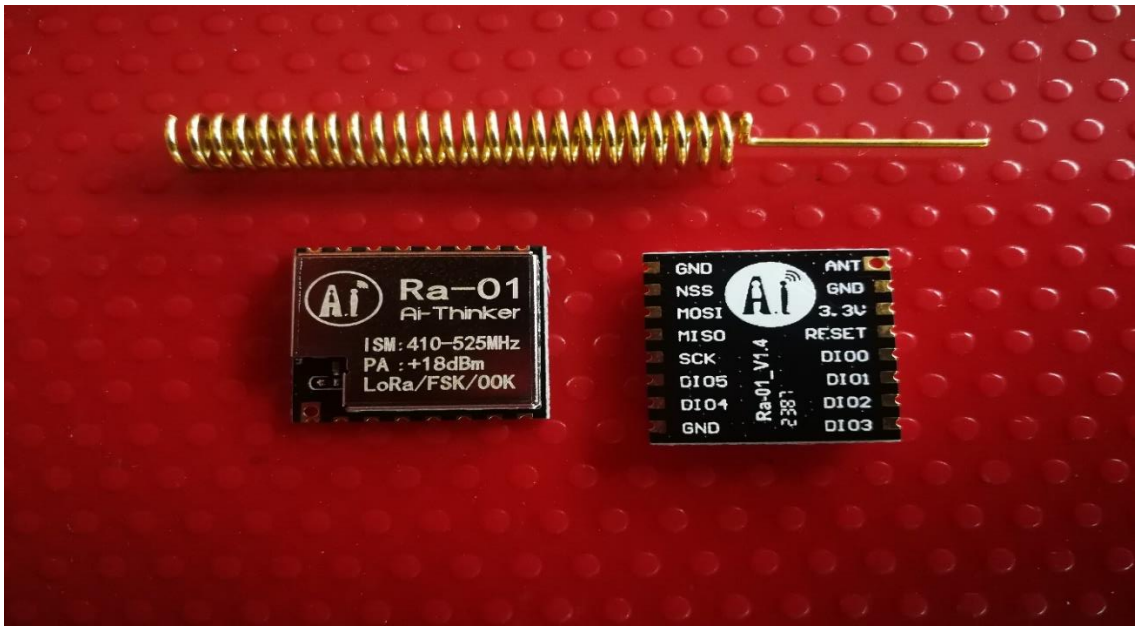
Ilustración 14. Conexiones Arduino UNO



Fuente: https://yorobotics.co/wp-content/uploads/2018/12/ARDUINO-UNO-R3-ATMEGA328-DIP28-3-600x400.jpg

1.5.2.2 LoRa Ra-01

Ilustración 15. Módulo LoRa Ra-01



Fuente: Autores

Referencia MÓDULO LORA RA-01 SX1278
Módulo transceptor RF de 433MHz

Desarrollado por la compañía Ai-Thinker, el módulo Ra-01 de LoRa usa la técnica de modulación de espectro extendido de largo alcance. Esta forma de comunicación inalámbrica, proporciona un mayor ancho de banda aumentando la resistencia a las interferencias y minimizando el consumo de energía.
Este módulo usa el IC SX1278 de Semtech y funciona en una frecuencia de 433MHz. El salto de frecuencia, permite la transmisión de señal de calidad, cubriendo un rango de 420-450MHz. Esta capacidad inalámbrica de largo alcance y unas medidas de tan solo (17 x 16 mm) lo que lo hace particularmente pequeño y normalmente viene con una antena de resorte como se evidencia en la imagen expuesta. (SIGMA ELECTRONICA, 2019)

Características:

- Comunicación de espectro extendido LoRaTM
- + 20dBm – 10mW. Potencia de salida de RF estable cuando el voltaje de entrada cambia.
- Comunicación SPI semidúplex
- La velocidad de bits es programable y puede alcanzar hasta 300 kbps

- Soporta FSK, GFSK, MSK, GMSK, LoRaTM y el modo de modulación OOK
- Rango de onda de 127dB RSSI.
- Detecta automáticamente la señal de RF, el modo CAD y el AFC de alta velocidad
- Con motor de datos CRC de 256 bytes
- Paquete SMD de medio perforado.
- Con carcasa metálica
- Antena de resorte

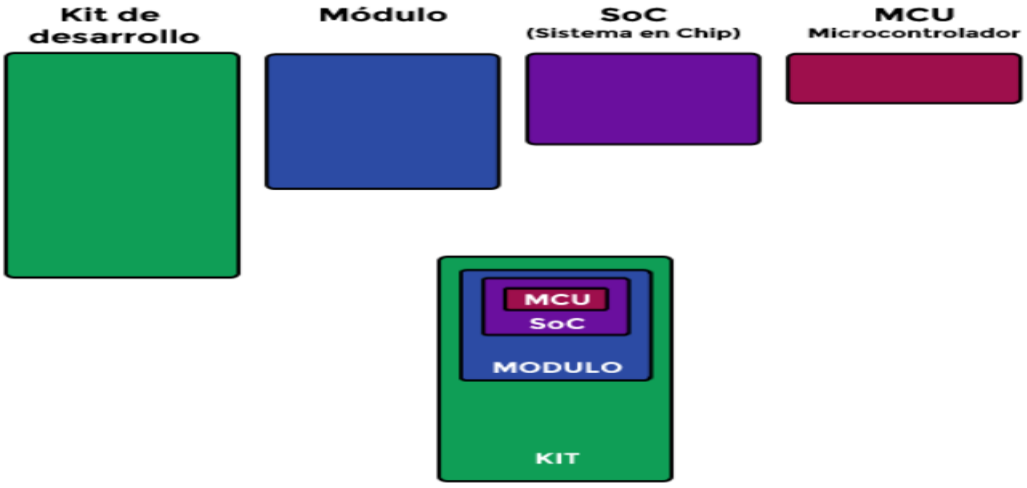
Especificaciones:

- Estándar inalámbrico: 433 MHz
- Rango de frecuencia: 420 – 450 MHz
- Puerto: SPI / GPIO
- Tensión de funcionamiento: 1,8 – 3,7 V, 3,3 V por defecto
- Corriente de trabajo, recibirá: menos de 10.8mA (LnaBoost cerrado, Band 1) de transmisión: menos de 120mA (+20 dBm), en modo reposo: 0.2uA
- Temperatura de trabajo: -40- + 85 grados.

1.5.2.3 NODEMCU LUA WI-FI V1.0 BASADO EN ESP8266

La NodeMCU Lua WI-FI v1.0 es una placa de desarrollo de hardware totalmente abierta que incorpora el chip SoC (System on a Chip) ESP8266 y este dentro tiene un microcontrolador o MCU. Además, tiene el chip CP2102 lo que permite hacer la conexión USB directamente al computador y de esta manera permitir la programación con el IDE de ARDUINO. Por otro lado, para esta tarjeta se ha desarrollado su propio API o entorno de programación que permite programar de una manera sencilla y rápida, de esta manera se ahorra el uso de una tarjeta adicional como Arduino UNO o cualquier otro modelo. En conclusión, el objetivo es programar la MCU o microcontrolador a través del kit o placa de desarrollo. Todo lo demás nos sirve de apoyo para que crear nuestros propios proyectos sea lo más sencillo posible.

Ilustración 16. Esquema general placa NodeMCU



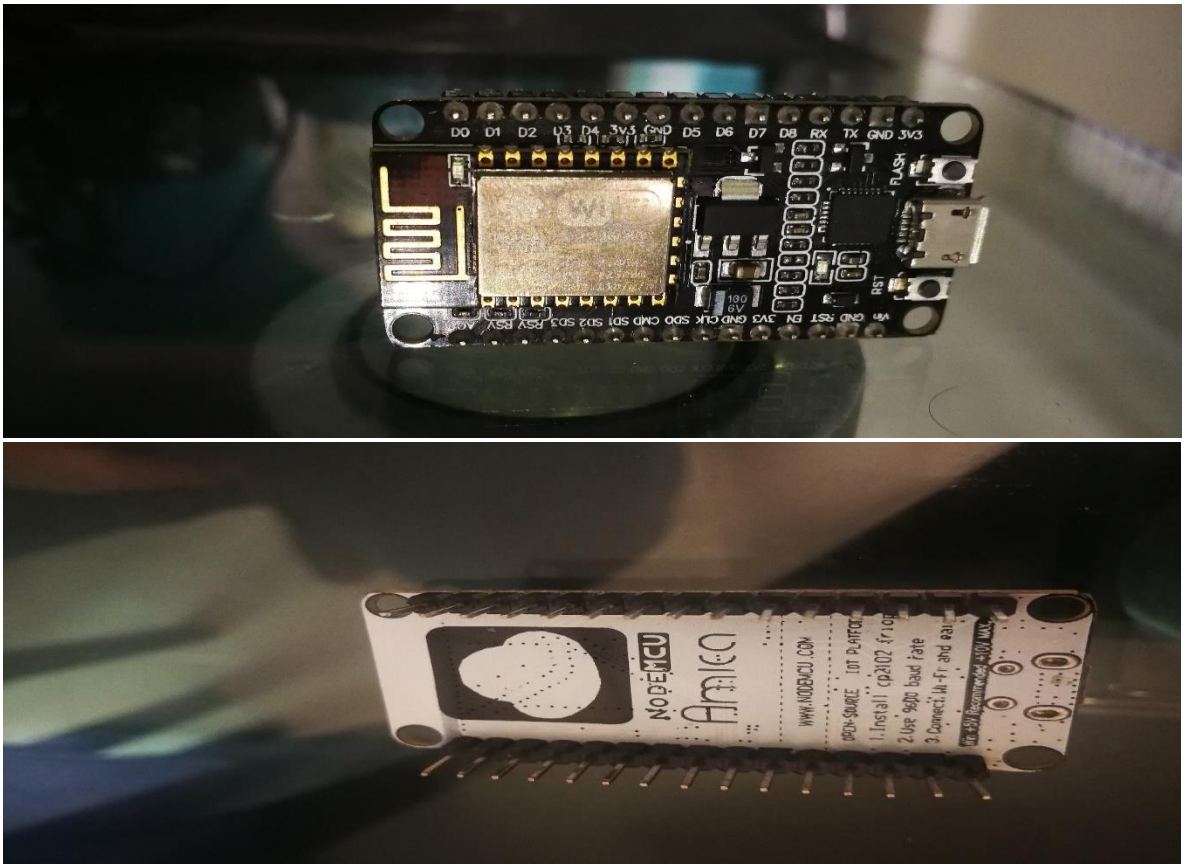
Fuente: https://programarfacil.com/podcast/nodemcu-tutorial-paso-a-paso/

Por su parte el ESP8266, Se trata de un SoC o Sistema en Chip que consiste en un chip que tiene casi todo integrado para que pueda funcionar de forma autónoma como si fuera un ordenador. En el caso del ESP8266 lo único que no tiene es una memoria para almacenar los programas.

Esto supone un inconveniente ya que parte de los pines de entrada y salida, tendrán que ser utilizados para conectarse a una memoria Flash externa. (Hernandez, 2019).
Sus principales características son las siguientes:

- Incorpora una MCU de 32-bit de bajo consumo (microcontrolador (Tensilica Xtensa LX106)
- Módulo Wi-Fi de 2.4 GHz
- RAM de unos 50 kB
- 1 entrada analógica de 10-bit (ADC)
- 17 pines de entrada y salida GPIO (de propósito general)

Ilustración 17. NodeMCU V1.0 Basado en ESP8266



Fuente: Autor

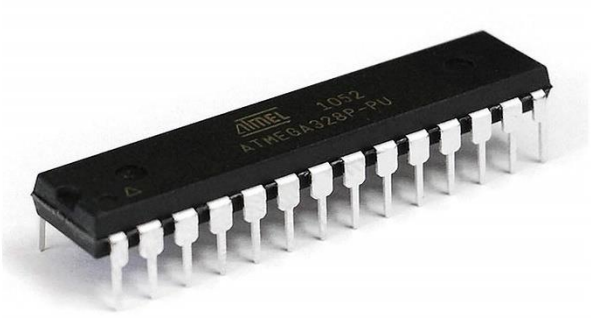
1.5.2.4 ATMEGA328P

Durante los últimos años, la plataforma de desarrollo de Arduino ha crecido de manera exponencial. Esto se debe a su fácil manejo, contribuciones de hardware y software e interés de empresas como Intel, Microsoft, Texas Instruments, entre otras, en adaptar sus productos a esta plataforma. Una de las tarjetas más usadas es la Arduino Uno, cuyo microcontrolador núcleo es el ATmega328P que cuenta con la arquitectura RISC avanzado AVR de Atmel, de alto desempeño, bajo consumo y optimizado para compiladores C, muy usada para cualquier cantidad de proyectos como chip central y eje de muchos proyectos y tarjetas de desarrollo.

Sus principales características son las siguientes:
131 Instrucciones potentes, la mayoría ejecutada en un solo ciclo de reloj.
- Un banco de 32x8 registros de propósito general.

- Hasta 20 MIPS (Millones de instrucciones por segundo) a 20 MHz.
- Un multiplicador hardware on-chip de 2 ciclos.
- Memoria de programa FLASH de 32 KBytes, programable dentro del sistema.
- Memoria SRAM interna de 2 KBytes.
- Memoria EEPROM de 1 KByte.
- 2 Timers/Contadores de 8 bits.
- 1 Timer/Contador de 16 bits.
- 6 Canales PWM.
- 6 Canales analógicos para el ADC.
- 1 Puerto serial USART.
- 1 Interface serial SPI.
- 1 Interface serial 2-Wire, compatible con I2C.
- 1 Timer watchdog.
- 1 comparador analógico on-chip.
- Interrupciones.
- Varios modos de bajo consumo.

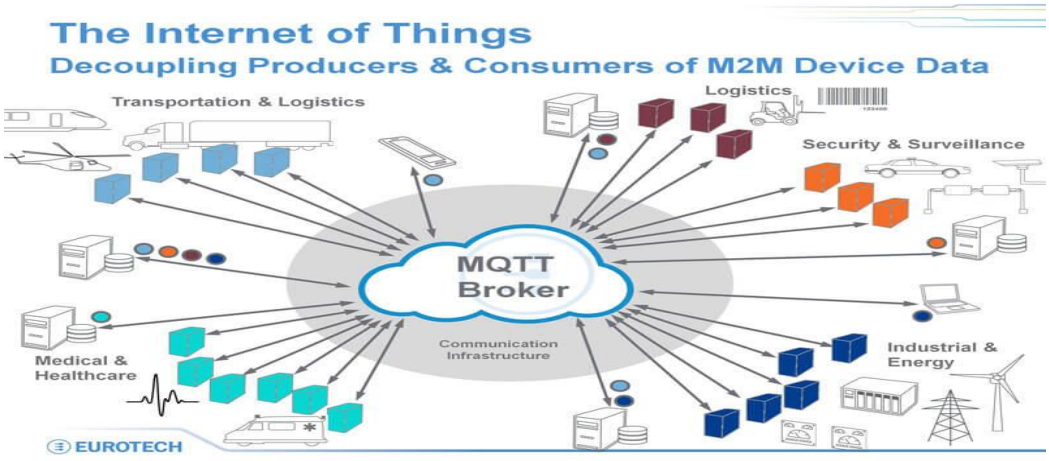
Ilustración 18. Atmega328p de ATMEL
Fuente: https://robu.in/product/atmega328p-pu-pdip-28-microcontroller/



1.5.2.5 SERVIDOR MQTT

Por sus siglas MQTT que es “Message Queue Telemetry Transport” es un protocolo de transportes de mensajes Cliente/Servidor, este último ubicado en la nube y que actualmente es muy usado en el “internet de las cosas” debido a que está orientado más a la comunicación de sensores debido a que consume muy poco ancho de banda y con dispositivos con pocos recursos lo que lo hace muy ligero y simple, permitiéndonos comunicarnos de manera bidireccional acercándonos más a tener un dialogo constate entre dispositivos y la interconexión global de estos.

Ilustración 19. Orientaciones del protocolo MQTT



Fuente: https://aprendiendoarduino.wordpress.com/2018/11/19/mqtt/

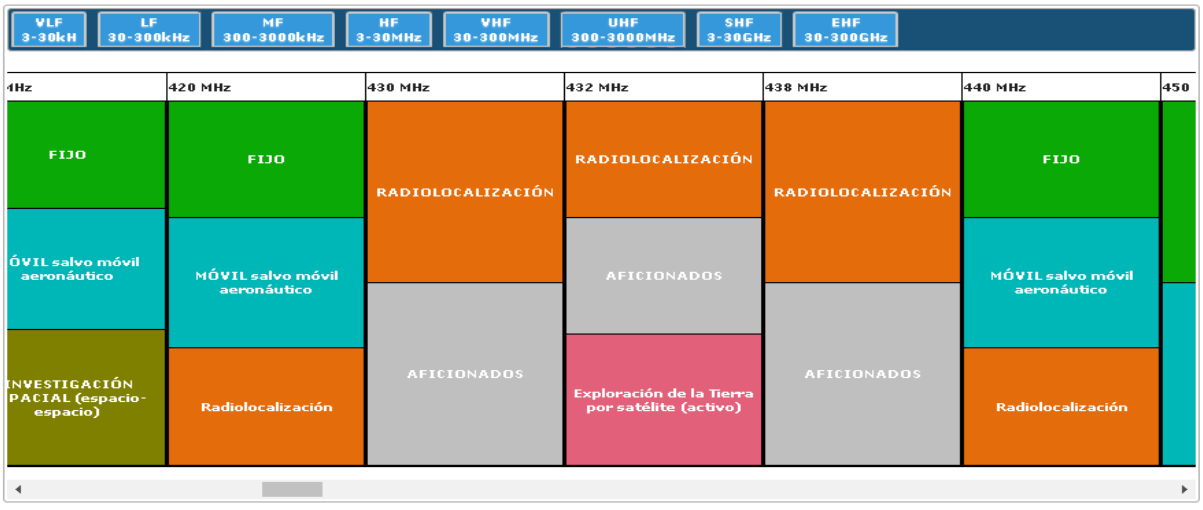
A diferencia de otros protocolos de comunicación más usados como el HTTP que está basado en un protocolo de petición/respuesta, MQTT se basa en publicación/suscripción, es decir, si un suscriptor se conecta al servidor (Este puede suscribirse a cualquier "tema" de mensajería del servidor) el cliente publica los mensajes en un tema, enviando el mensaje y el tema al servidor, después, el servidor 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. (Del Valle, 2019)

1.5.3. MARCO LEGAL

Esta tecnología de radio LoRa utiliza las bandas ISM, Industrial Scientific & Medical. Estas bandas son de uso libre sin licencia, pero limitadas en potencia y tiempo de transmisión. LoRa utiliza tres frecuencias principalmente:

- **433 MHz en Asia y América sur (frecuencia a usar)**
- 915 MHz en América
- 868 MHz en Europa

Ilustración 20. Cuadro nacional de atribución de bandas de frecuencia para Colombia



Fuente:
http://cnabf.ane.gov.co/cnabf/index.php?option=com_k2&view=item&layout=item&id=6&Itemid=140&s=94E1F50794613A73A120774588BC85A5615A1C42

El Decreto 0963 del 20 de marzo del 2009 del Ministerio de las comunicaciones de Colombia indica en sus artículos lo siguiente;

CAPITULO I DISPOSICIONES GENERALES

Artículo 1. SERVICIO DE RADIOAFICIONADO. El servicio de radioaficionado es un servicio de radiocomunicación que tiene por objeto la instrucción individual, la intercomunicación y los estudios técnicos efectuados por aficionados debidamente autorizados que se interesan en la radio experimentación con fines exclusivamente personales y sin ánimo de lucro.

Artículo 2. **PRESTACIÓN DEL SERVICIO.** El servicio de radioaficionado es un servicio especial que será prestado mediante licencia otorgada por el Ministerio de Comunicaciones, de conformidad con lo estipulado en el presente Decreto, la Ley 94 de 1993, la Ley 72 de 1989, el Decreto Ley 1900 de 1990, y las normas que los modifiquen, aclaren o adicionen, siguiendo los principios establecidos en el Código Contencioso Administrativo. El servicio de radioaficionado y radioaficionado por satélite, podrá prestarse en todo el territorio nacional, incluyendo aguas territoriales y espacio aéreo, así como también en los lugares que por convenciones internacionales le reconozcan a Colombia el principio de extraterritorialidad.

Artículo 22°. FOMENTO A LA INVESTIGACIÓN Y DESARROLLO. Es objetivo principal de las asociaciones de radioaficionados, fomentar el estudio, la instrucción, la investigación y la radio experimentación de las comunicaciones a nivel aficionado. Para el despliegue del servicio de radioaficionado, las asociaciones podrán dictar y recibir cursos, talleres, conferencias y seminarios, con el objeto de fomentar la investigación y el desarrollo. La investigación y desarrollo deberá propender, entre otros, por: el establecimiento de estaciones de radioaficionados en zonas rurales y distantes; la formación de técnicos en el diseño, construcción y mantenimiento de sistemas y equipos de radiocomunicaciones; la capacitación en la normatividad de las telecomunicaciones nacionales y las normas internacionales del servicio de aficionado, la promoción para el diseño de sistemas capaces de proporcionar comunicaciones en casos de catástrofe y durante las operaciones de emergencia y la creación de grupos capaces de proporcionar apoyo local y nacional; el desarrollo de conocimientos de los operadores; el intercambio de información técnica y la experimentación con nuevas tecnologías.

CAPITULO VII BANDAS Y PLANES DE FRECUENCIAS

Artículo 29°. ATRIBUCIÓN DE BANDAS DE FRECUENCIAS. El establecimiento y operación de estaciones de aficionado y la utilización de las bandas de frecuencias para la prestación del servicio de radioaficionado, sólo podrán efectuarse de acuerdo con las normas establecidas en la Ley, en el presente Decreto y de conformidad con la atribución internacional del espectro radioeléctrico instituida por la Unión Internacional de Telecomunicaciones UIT para la Región 2 americana.

29.1. Se atribuyen a Titulo Primario y Co-Primario, en todo el territorio nacional, las siguientes bandas de frecuencias del espectro radioeléctrico para el servicio de radioaficionado y radioaficionado por satélite:

Tabla 1. Banda de Frecuencias según servicio y frecuencia.

BANDA	LONGITUD DE ONDA	SERVICIO	ATRIBUCIÓN A TITULO
VHF			
50 - 54 MHz.	6m	Aficionados	Primario
144 - 146 MHz.	2m	Aficionados y aficionados por satélite	Primario
146 - 148 MHz.	2m	Aficionados	Primario
220 - 225MHz.	1.25 m	Aficionados	Co-Primario
UHF			
430 - 440 MHz.	**70 cm**	**Aficionados**	**Co-Primario**
24 -24.05 GHz	1.2 cm	Aficionados y aficionados por satélite	Primario

Fuente: https://www.mintic.gov.co/portal/604/articles-3641_documento.pdf

2. DESARROLLO DEL TRABAJO DE GRADO

2.1. PROCEDIMIENTO

2.1.1. CARACTERIZAR LOS COMPONENTES REQUERIDOS EN LA IMPLEMENTACIÓN DE UN SISTEMA DE COMUNICACIONES INALÁMBRICO.

La metodología desarrollada en este trabajo de grado inicia con la recolección de información referente al Internet de las cosas (IoT) a nivel regional, nacional e internacional, partiendo de que toda la tecnología inalámbrica tiene puntos fuertes y débiles, al igual que

los elementos clasificados para el desarrollo de este proyecto que cuentan con diferentes tipos de características en comparación con otros similares que pueden realizar la misma utilidad o parecidos. Esto se debe a que la intención inicial es desarrollar un sistema de comunicación que ofrezca las ventajas del IoT relativos al bajo consumo energético, la baja tasa en él envió de datos y el largo alcance, además del reducido tamaño y costo.

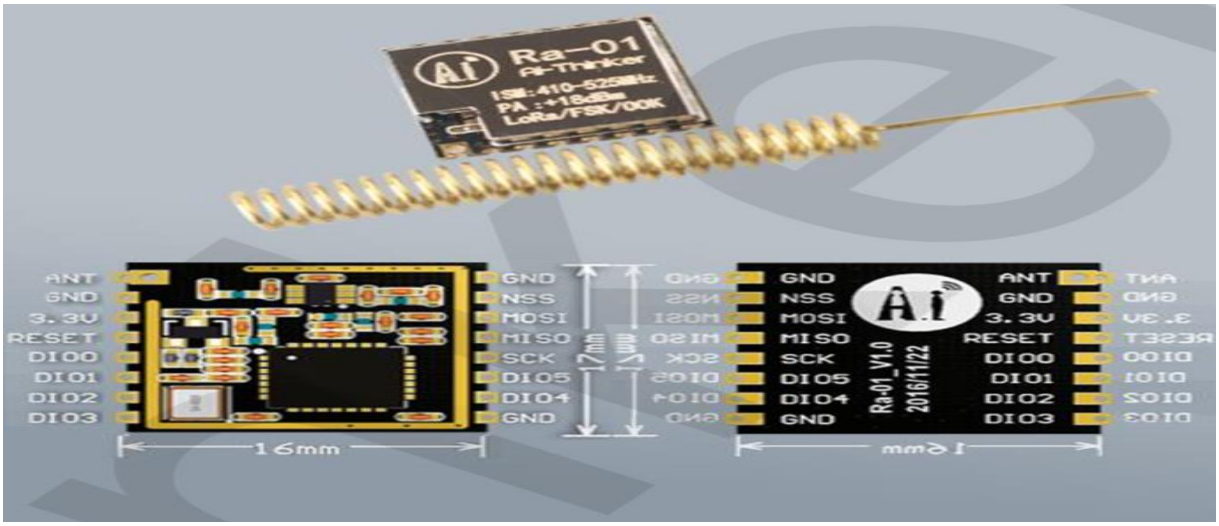
Al identificar la tecnología que cumple las expectativas, se procede a seleccionar cada uno de los elementos a nivel de hardware y de software que permitan cumplir con los objetivos propuestos. Posteriormente se realizará el diseño esquemático a nivel electrónico del prototipo, se construirá y se realizaran las pruebas correspondientes que aseguren el correcto envió de información, en donde se realizara la programación necesaria en torno a la mejor relación de la señal/ruido que permite que los paquetes lleguen de la manera más eficiente a la nube.

Es por esto que en el proceso de selección de los mismos se tuvieron en cuentas las principales factores que hacen de la tecnología LoRa resalte sobre las otras tecnologías inalámbricas como Wi-Fi, Bluetooth, Sigfox o la red celular, teniendo en cuenta que cada una de estas tecnología se adapta a diferentes aplicaciones y siendo los requerimientos más relevantes la comunicación a largas distancias con un tamaño, costo y consumo energético mínimo, que a su vez sea de tipo escalable y acumulativo, es decir, pensado en abrir puertas futuras para el desarrollo en la incorporación de estas nuevas tecnologías tanto en el área de la ganadería como en otras áreas de la región.

2.1.2. SELECCIÓN DE MÓDULOS DE TRABAJO.

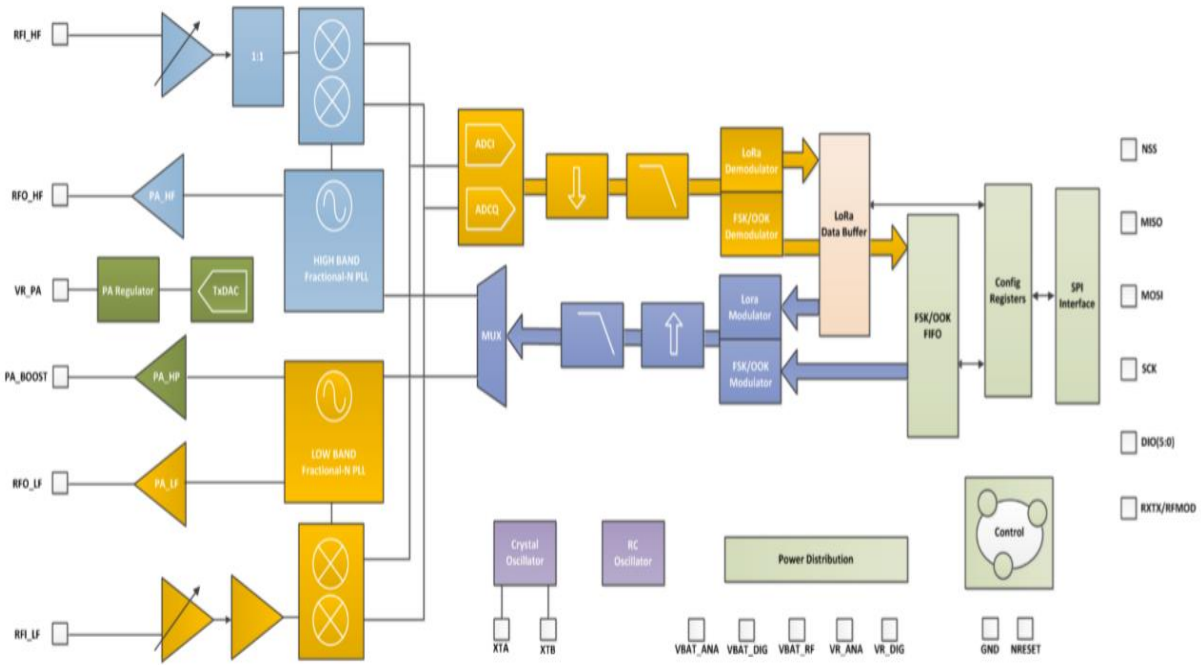
Una vez establecidas las finalidades del proyecto se pre-selecciono el módulo de transmisión inalámbrica LoRa ra-01 basada en SEMTECH SX1278, con la cantidad suficiente de estos dispositivos para el diseño de la red mesh, sin embargo, este módulo de radio requiere de la lógica de programación para ser configurado y es por esto que debe estar acompañado de un microcontrolador. El microcontrolador en una terminal LoRa lee y procesa los datos del sensor o paquetes a enviar, y también se interconecta con el transceptor para transmitir los datos por la red ya que este debe contar con conexión Wi-Fi.

Ilustración 21. Módulo Lora y su distribución de pines



Fuente: *Semtech Corporation*

Ilustración 22. Diagrama esquemático de bloques del SX1278

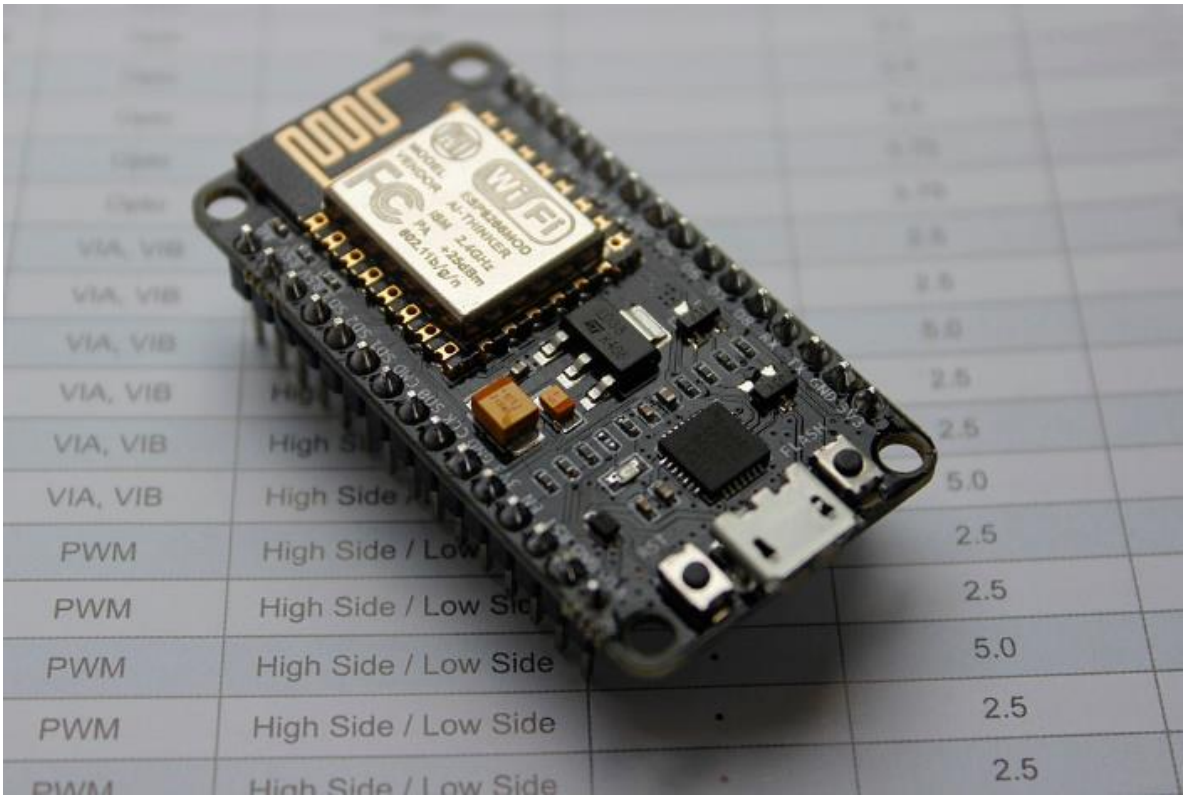


Fuente: *Semtech Corporation*

El ESP8266 es un chip que dispone de conexión Wi-Fi, de bajo costo y se puede programar directamente con el entorno de Arduino con lo que es el chip perfecto para desarrollar nuestras aplicaciones de IoT. Pero tiene sus desventajas al solo usar este chip y es por esto que se usó el kit de desarrollo que incluye este modelo, pero con grandes mejoras entre las que se tiene para el NodeMCU V1.0 basado en ESP8266 las siguientes:

- Conversor Serie-USB para poder programar y alimentar a través del USB
- Fácil acceso a los pines
- Pines de alimentación para sensores y componentes
- LEDs para indicar estado
- Botón de reset

Ilustración 23. Kit de desarrollo NodeMCU V1.0



Fuente: https://github.com/jaimelaborda/Planta-Twittera/wiki/1.-Introducci%C3%B3n-al-ESP8266-y-NodeMCU

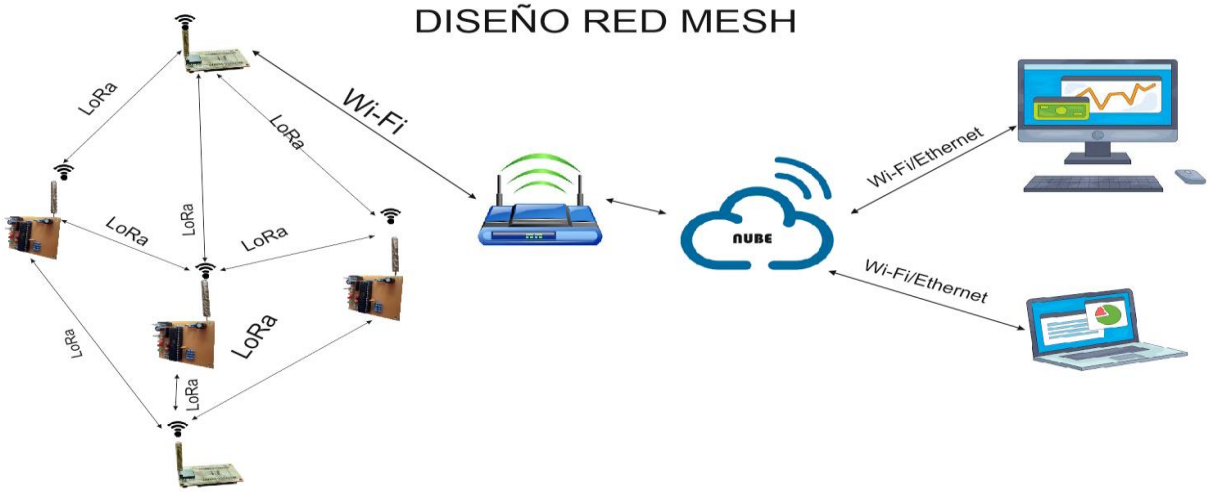
NodeMCU debe tener suficiente memoria para los controladores del SX1278, los controladores de sensores y el código de la aplicación. Esta es una placa muy económica y de hardware libre, esto permite tener a nuestra disposición todos los esquemas para poder construir nuestra propia placa con el LoRa e incluso comercializarla. La gran ventaja de este desarrollo con respecto al resto de módulos básicos de ESP8266, es que su

programación se hace totalmente transparente, al no requerir ningún cambio en sus pines para la programación, y disponer de conexión USB al igual que Arduino.
Siendo suficiente para los alcances del proyecto, en su eficiencia y los bajos costos, se implementaron un par de NodeMCU v1.0, uno ubicado en la posición final e inicial, ya que uno se encargará de recoger los datos y el otro de trasmitirlos por medio de una conexión wifi a la nube.
Para el punto que recogerá los datos y los trasmitirá a los demás nodos, se pensó utilizarlo debido a que este proyecto hace parte de un Macro-proyecto con la idea de poder obtener los paquetes de información de diferentes maneras por medio de sus entradas y salidas análogas o digitales conectados de manera alámbrica o también inalámbrica por medio de su conexión wifi, además los puertos SPI (Serial Peripheral Interface / Interfaz periférica serial) importantes para su comunicación con el LoRa.

Así que para simular los eventos presentes por un sensor se instala un pulsador que hará de este, permitiéndonos enviar un mensaje cada vez que es oprimido para poder llevarlo por medio de la red mesh por la mejor ruta posible hasta su destino final, la nube, que se encargará el NodeMCU final y que no cuenta con el pulsador ya que este solo recogerá los mensajes de los nodos anteriores a él. De igual manera el envío de datos se puede realizar de manera bidireccional, es decir, desde la nube, pasando por la red, hasta el Módulo Inicial.
Como nodos repetidores se utilizaron el microcontrolador nucleó del muy conocido ArduinoUNO, el ATmega328P optado por su alto desempeño, bajo consumo y optimizado para compiladores (Por su menor tiempo de ejecución del programa y reducido espacio que ocupa el mismo), asimismo, por su bajo precio y accesibilidad.

2.1.3. DISEÑO DE LA RED MESH.

Ilustración 24. Diseño red mesh final



Fuente: Autor

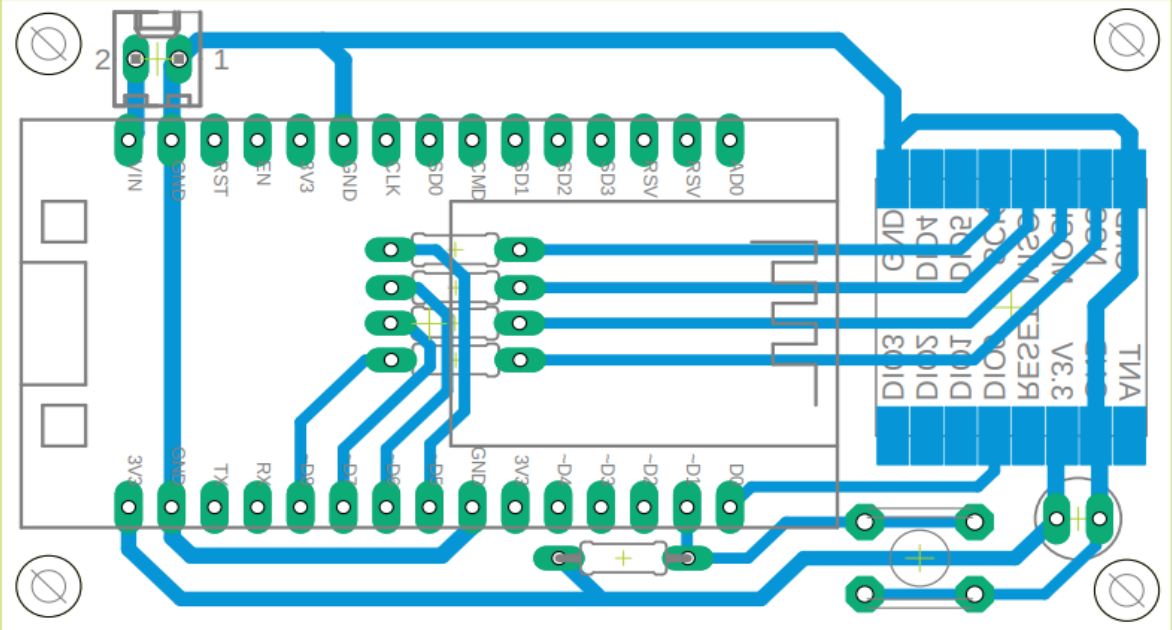
2.1.4. DISEÑO DEL SISTEMA EMBEBIDO EN PCB DE LOS NODEMCU BASADO EN EL ESP8266 CON EL MÓDULO LORA.

Una vez adquiridos los módulos se procede al diseño del PCB que contendrá todo el sistema integrado por medio del software de licencia gratuita EAGLE y que se muestra en las siguientes imágenes:

2.1.5. DISEÑO NODEMCU INICIO (MENSAJE)- LORA RA-01

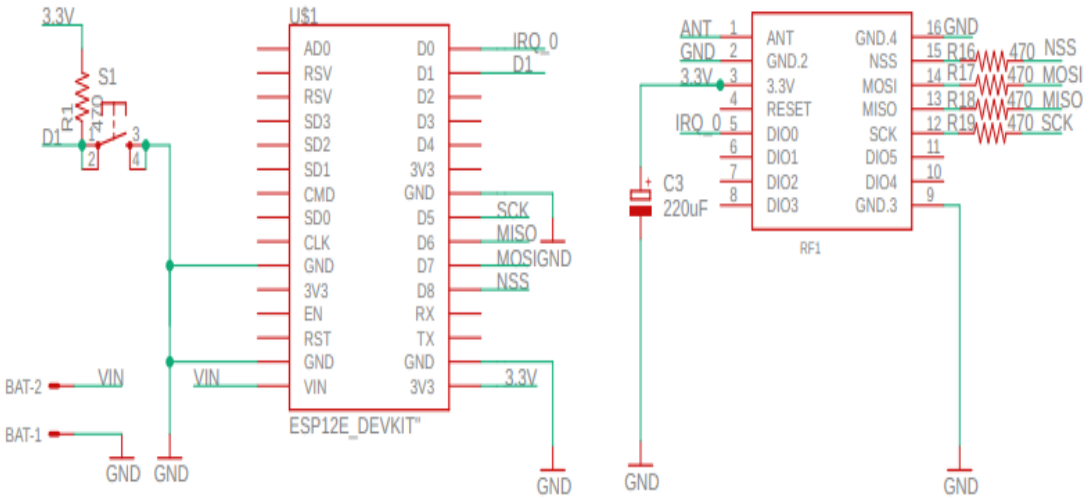
Este módulo a diferencia del otro NodeMCU tiene agregado en el pin D1 un pulsador que envía un mensaje al ser oprimido.

Ilustración 25. Diseño Conexiones NodeMCU con pulsador- LoRa



Fuente: Autor/Software Eagle

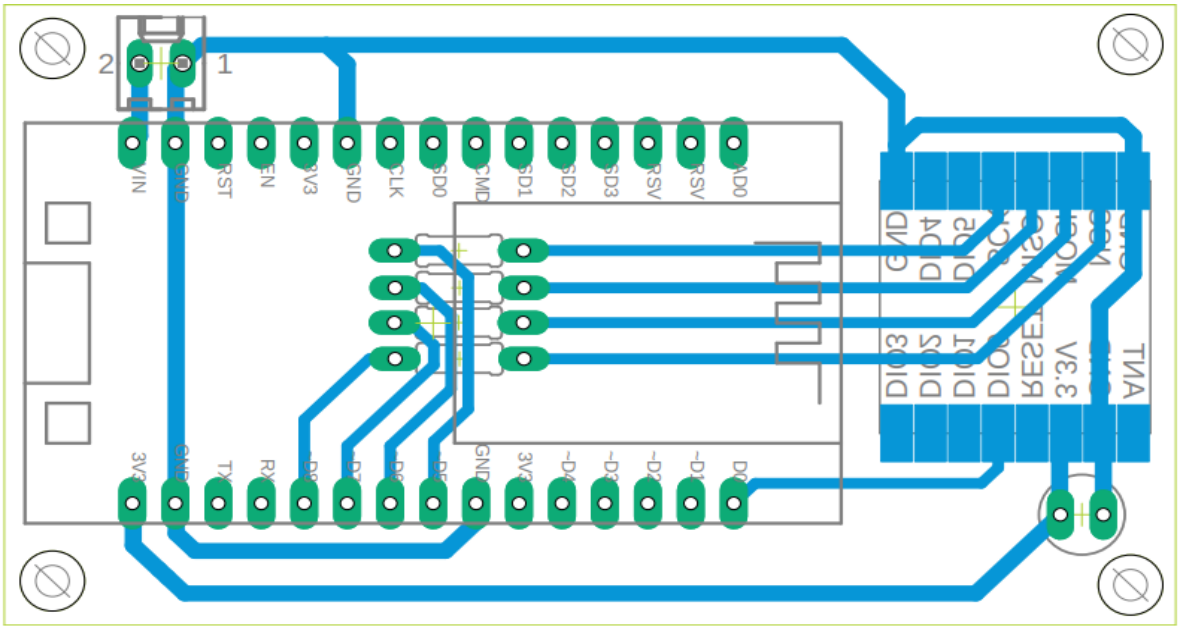
Ilustración 26. Conexiones-pines NodeMCU con pulsador-LoRa



Fuente: Autor/Software Eagle

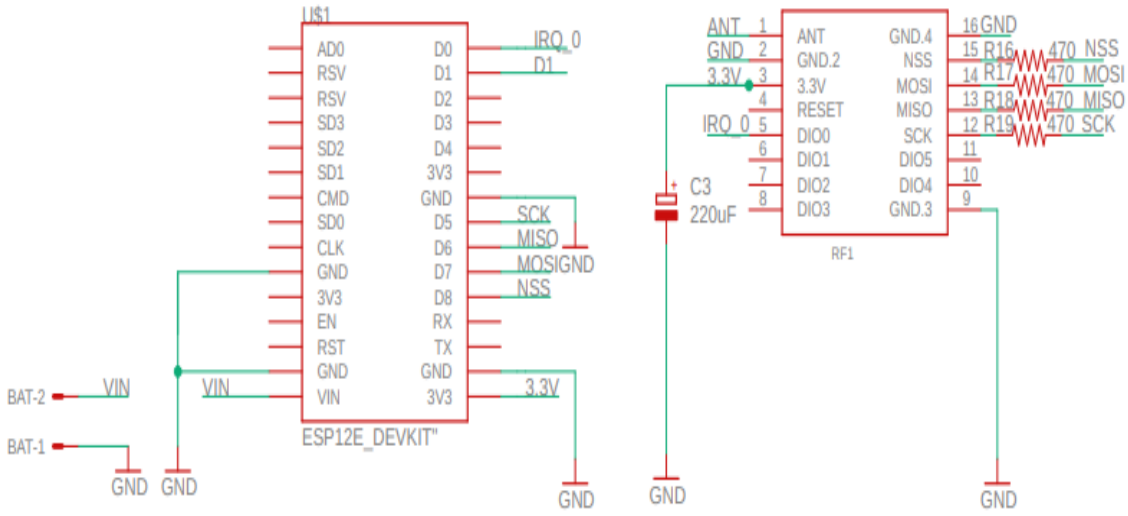
2.1.6. DISEÑO NODEMCU FINAL (NUBE)- LORA RA-01

Ilustración 27. Diseño Conexiones NodeMCU - LoRa



Fuente: Autor/Software Eagle

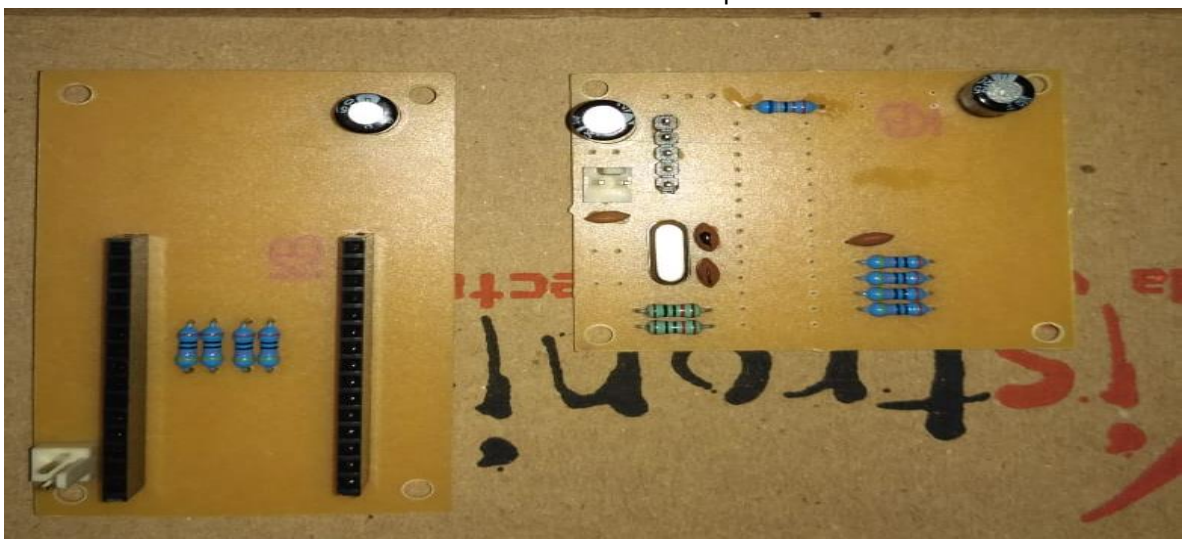
Ilustración 28. Conexiones-pines NodeMCU -LoRa



Fuente: Autor/Software Eagle

2.1.7. CONSTRUCCIÓN DE PLACAS

Ilustración 29. Construcción placas



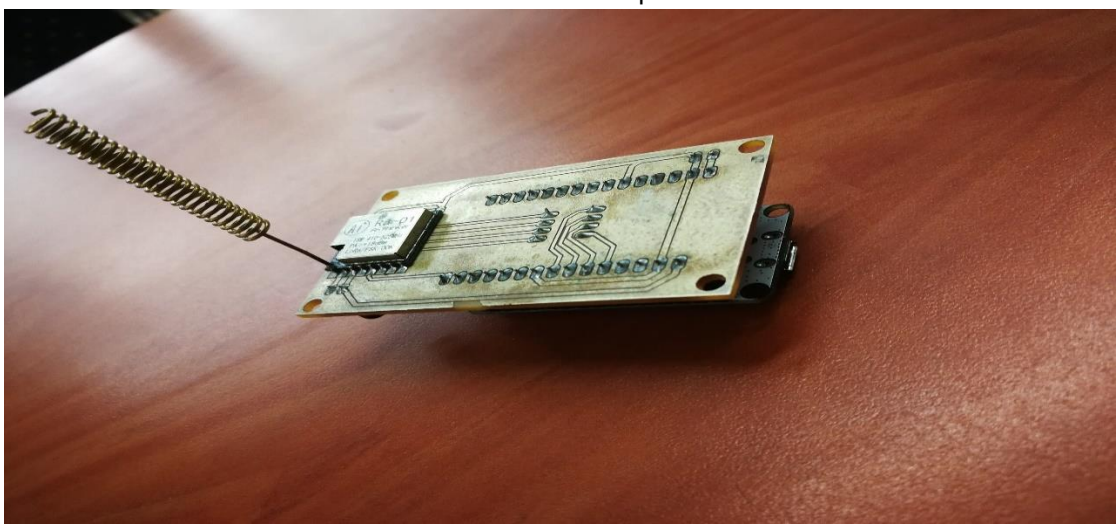
Fuente: Autor

Ilustración 30.Construcción de placas

Fuente: Autor

2.1.8. PLACA FÍSICA TERMINADA NODEMCU BASADO EN ESP8266- LORA RA01

Ilustración 31. Placa vista circuito superior NodeMCU-LoRa



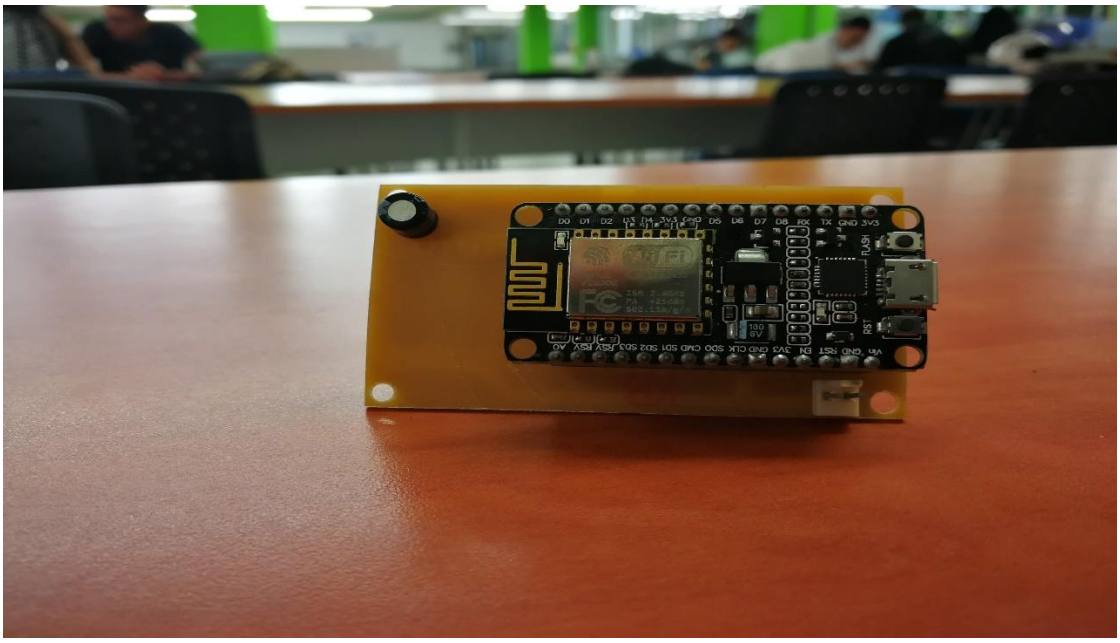
Fuente: Autor

Materiales usados por una placa del NodeMCU-LoRa:

- 1 Módulo Lora Ra-01 de Ai-Thinker
- 1 Antena Helicoidal 433 Mhz
- 1 NodeMCU v1.0 basado en ESP8266
- 1 Condensador electrolítico 220 µF
- 2 regletas hembra-macho de 15 pines cada una
- 1 par de pines para adaptador de alimentación
- Estaño
- Circuito impreso
- 4 resistencias de 470 ohm

Aquí se muestra el circuito ya impreso y soldado. En la imagen se ve a el módulo LoRa ra-01 con la antena helicoidal que viene con este, la antena tiene una longitud de 5.3 cm y va soldada al pin 1 o que marca como ANT, en este caso va soldada a un pequeño orificio que trae el chip donde es recomendable instalarla por su forma.

Ilustración 32. Placa vista inferior- NodeMCU basado en ESP8266



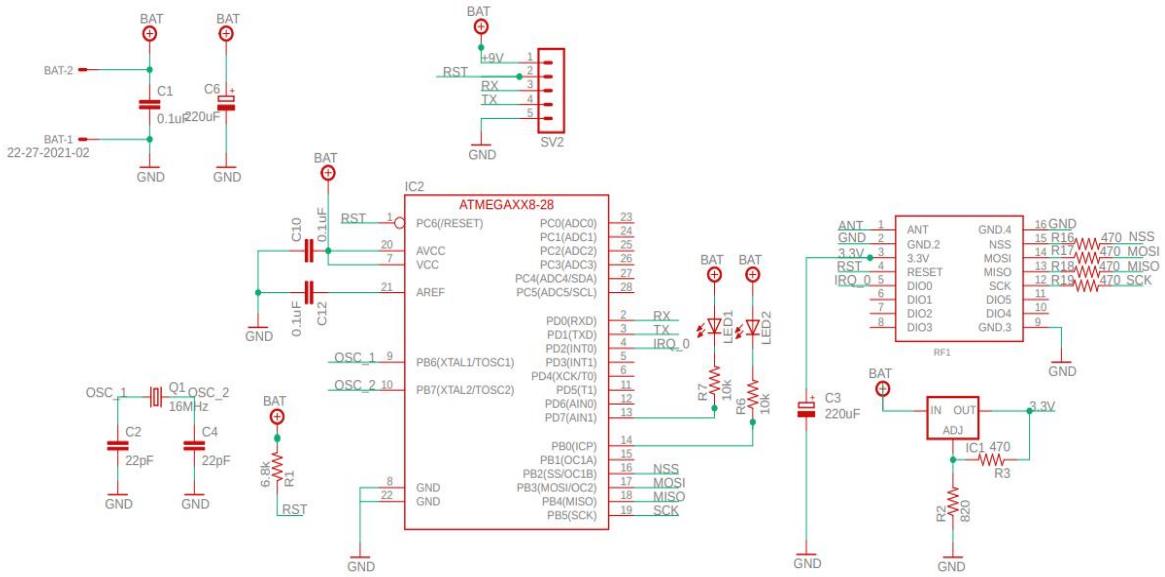
Fuente: Autor

Por otro lado, vemos la parte de la placa donde se observa al NodeMCU y su SoC (sistema en Chip) el ESP8266 que no va soldado a diferencia del módulo LoRa, ya que este si tiene pines que van encajados a dos regletas hembra de 15 pines cada una, las cuales si van soldadas a la placa previamente. El condensador electrolítico que se muestra es de 220µF

para que no tenga picos de corriente y mantenga estable el valor de entrada de la alimentación del LoRa, ya que este se alimenta desde la salida a 3.3V del NodeMCU que estará alimentado externamente con baterías de 9V a el par de pines que se ven en la parte inferior derecha de la placa.

2.1.9. DISEÑO DEL SISTEMA EMBEBIDO EN PCB DE LOS ATMEGA328P CON EL MÓDULO LORA.

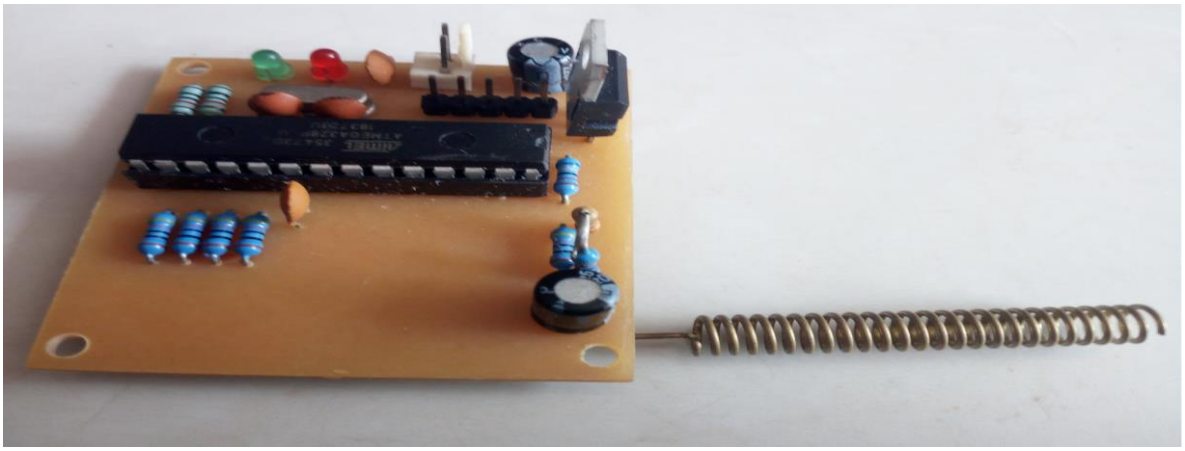
Ilustración 33. Conexiones-pines ATmega328P –LoRa ra-01 con descripción.



Fuente: Autor/Software Eagle

2.1.10. PLACA FÍSICA TERMINADA ATMEGA328P - LORA RA01

Ilustración 34. Placa vista circuito superior ATmega328P-LoRa



Fuente: Autor

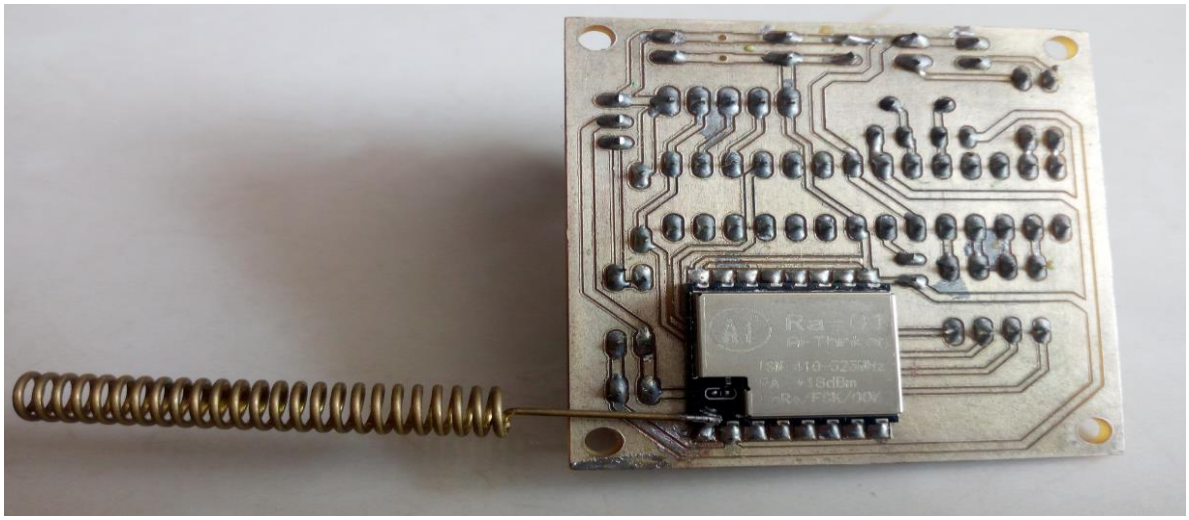
La placa utilizada para el atmega328p está conformada por los implementos nombrados a continuación: el pin 1 es el RESET que lleva al segundo pin del base socket que se utilizará para programar el ATmega328P sin necesidad de montarlo a una placa de Arduino, seguido de esto los pines 2 y 3 que son receptor y transmisor respectivamente irán a los pines 3 y 4 del base socket, los pines 7, 20 y 21 tienen capacitores de 0.1uF para evitar picos de corriente en dichas entradas de alimentación, entre los pines 9 y 10 se conecta un oscilador de 16Mhz ya que inicialmente no podremos usar el reloj interno por no tener bootloader, seguido de 2 capacitores de 22pF en cada extremo del éste para aumentar su estabilidad según indica su datasheet. En los pines 16, 17, 18 y 19 se utilizaron en cada uno resistencias de 470omh para proteger la entrada de los pines SPI del LoRa ya que éste es muy delicado cuando el voltaje sube demasiado.

Pasamos al regulador de voltaje LM317 el cual al tener una resistencia de 470omh de la salida al ajuste, y del ajuste a tierra una resistencia de 820omh según los cálculos realizados y pruebas hechas por medio de un potenciómetro en la R2 en el que se ubica la resistencia de 820omh y nos reduce la entrada de 9 voltios a un voltaje que esté en el rango de entrada de alimentación para el módulo LoRa donde lleva un capacitor de 220uF para estabilizar la corriente de entrada a dicho módulo.

Para programar la placa atmega328p se debe conectar de una placa de Arduino al microcontrolador de la siguiente manera:

- La placa de Arduino tiene una salida de 5V con la cual alimentamos el primer pin de la base socket el cual lleva a la entrada del atmega328p.
- En el segundo pin se debe conectar el RESET de la placa del Arduino.
- Para el tercero se conecta el receptor (RX) el cual se encarga de que reciba el programa que se esté subiendo a dicho micro controlador.
- En el cuarto pin es el trasmisor que se utiliza para observar los cambios que ocurren en el ATmega328P, ya sea sobre el programa en directo o avisos del programa como puede ser que subió el programa o que hubo algún problema.
- Por ultimo estaría la tierra para poder cerrar el circuito de alimentación al microcontrolador.
- El oscilador externo es necesario en todo microcontrolador ya que estos necesitan que le indiquen a qué velocidad trabajar es como el motor, por lo tanto, este pequeño circuito es indispensable.

Ilustración 35. Placa vista inferior- LoRa ra01 – Atmega328P



Fuente: Autor

Materiales usados por una placa del ATmega328P-LoRa:

- 1 Módulo Lora Ra-01 de Ai-Thinker
- 1 Antena Helicoidal 433 Mhz
- 1 ATmega328P
- Circuito Impreso
- 5 resistencias de 470 ohm
- 2 resistencias de 10k ohm
- 1 resistencia de 820 ohm
- 1 resistencia de 6.8k ohm
- 2 condensadores de 22pF cerámico
- 2 condensadores de 220µF electrolítico
- 2 condensadores de 0.1 µF cerámico
- 2 leds
- 1 Base socket de 28 pines
- 1 par de pines para adaptador de alimentación
- 1 Cristal 16Mhz Oscilador
- 1 regulador de voltaje LM317

2.1.11. CONSTRUCCIÓN DE ANTENA DIPOLO

Los materiales necesarios para la construcción de cada uno de los dipolos fueron dos tramos de varillas de cobre de cable coaxial RG-8 de longitud λ / 4 de 2[mm] de diámetro cada.

El cálculo de λ, (longitud de onda), se obtiene de la ecuación:

$$\lambda = \frac{c}{f}$$

Donde λ = Longitud de onda (m)

c = Velocidad de la luz m/s ➡ 299.792.458 m/s

f = frecuencia de trabajo (Hz) ➡ 433.000.000 Hz

$$\lambda = \frac{299.792.458}{433.000.000} = 0{,}69236 \text{ m}$$

Por lo tanto para una antena $\frac{\lambda}{4}$

$$\frac{0{,}69236}{4} = 0{,}173 \text{ m}$$

Como se diseñaron los dipolos para una frecuencia de trabajo igual a 433[Mhz] se obtiene que el valor de la longitud de onda, λ es igual a 0.69236 mts y cada polo debe tener en consecuencia, un largo de 0.17 mts. El procedimiento para construir ambos dipolos consistió quitar cubierta o carcaza del cable semirrígido de un cable coaxial comercial conocido como RG-8 con una impedancia de 50 ohm, quedando armadas las antenas. Para proporcionar mayor rigidez a los dipolos aplicar una capa de silicona sobre la soldadura.

Cada polo estuvo soldado lo más cerca posible el uno del otro entre el pin de la antena y tierra (GND) así la resistencia de radiación es menor logrando una cobertura mucho mayor que la obtenida con la antena helicoidal que viene con el módulo LoRa.

Ilustración 36. Prueba de distancia con antenas dipolo

Fuente: Autor

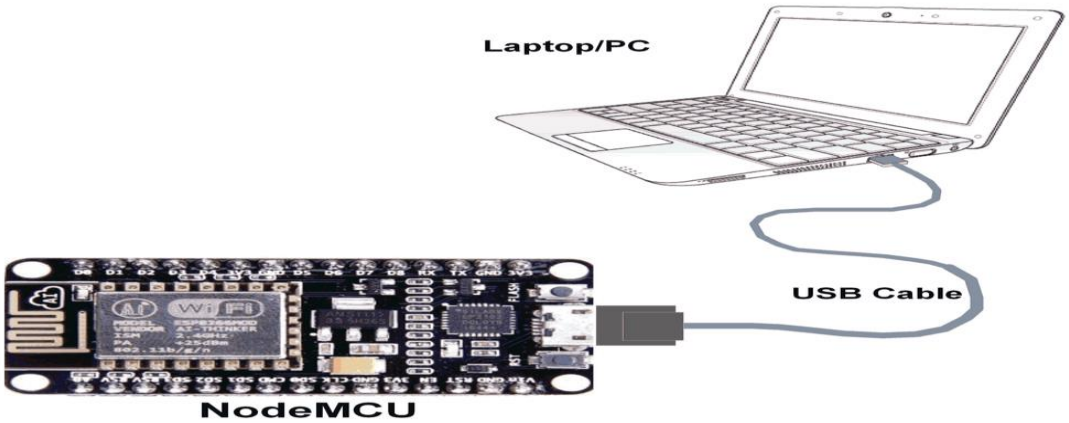
2.1.12. CONEXIÓN BÁSICA DE LA PLACA AL COMPUTADOR

El firmware NodeMCU podía grabarse en un ESP8266, tras lo cual podíamos programarlo con el lenguaje script Lua. La programación el Lua permitía la conexión y programación del ESP8266 de una forma mucho más sencilla, pero en la investigación realizada en varias fuentes afirman esta presenta varias fallas.

2.1.13. PASOS PROGRAMACIÓN NODEMCU V1.0

La placa se conecta a una computadora mediante un conector USB a micro USB, como los usados normalmente para cargar nuestros celulares. Este suministra la alimentación y comunicación entre el computador y el NodeMCU.

Ilustración 37. Conexión NodeMCU a PC



Fuente: https://www.electronicwings.com/nodemcu/getting-started-with-nodemcu-using-esplorer

Paso 1. Instale el IDE Arduino en su versión actual disponible en el sitio web de Arduino, en el siguiente enlace.

Descargar IDE de Arduino Aquí.

El IDE de Arduino es compatible con los sistemas operativos Windows, Mac OS X y Linux.

Paso 2. Una vez instalado el software procedemos a instalar el PLUGGIN ESP8266.
Iniciar Arduino y en Archivo abra la ventana de Preferencias.
Para instalar el pluggin vamos a tener que descargarlo de internet, lo que va a llevar un tiempo dependiendo de nuestra velocidad de internet.

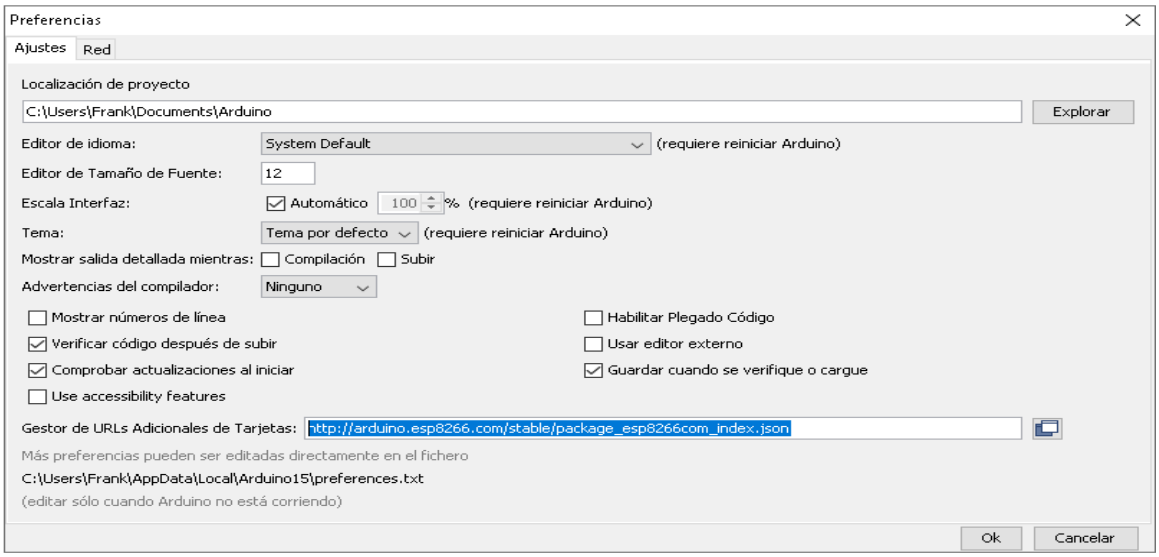
Ilustración 38. Configuración IDE Arduino.

Fuente: Autor

Paso 3. Una vez abierta la ventana de preferencias, en Gestor de URLs Adicionales de tarjetas, copiamos y pegamos el siguiente link del pugglin del ESP8266 y damos en OK

http://arduino.esp8266.com/stable/package_esp8266com_index.json

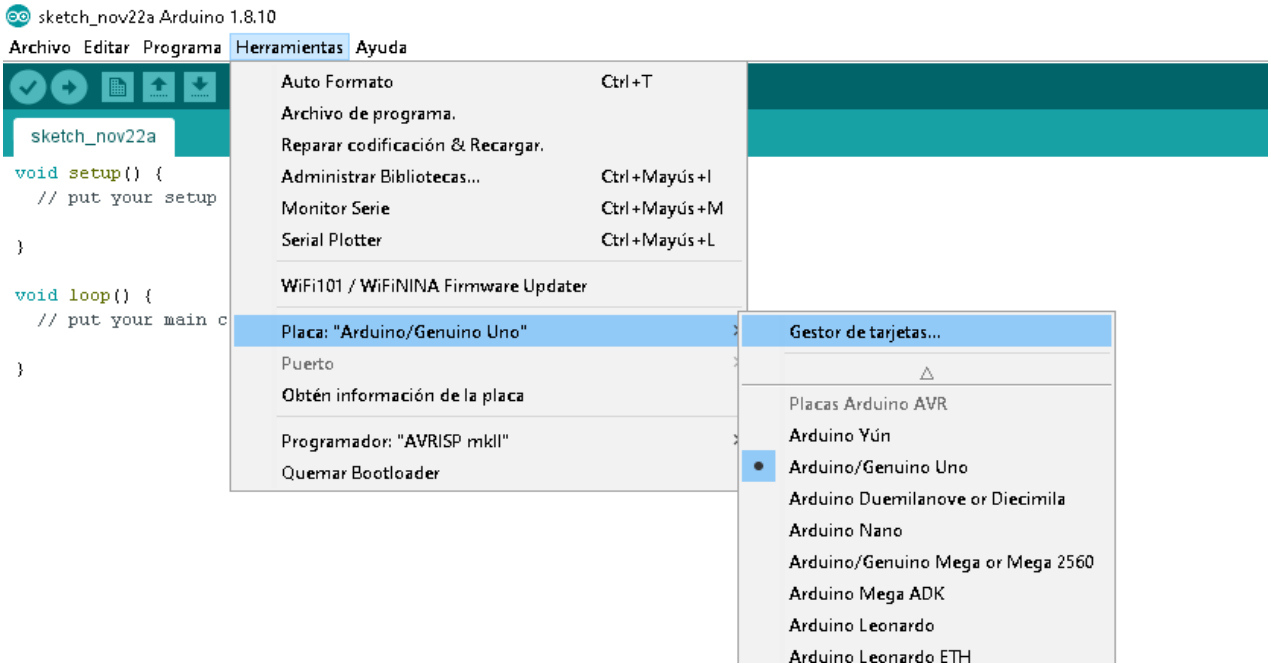
Ilustración 39. Ventana de preferencias del IDE de Arduino



Fuente: Autor

Paso 4. Luego falta elegir el modelo que queremos programar, para esto abrimos el menú herramientas, luego en Placa, y Gestor de tarjetas…

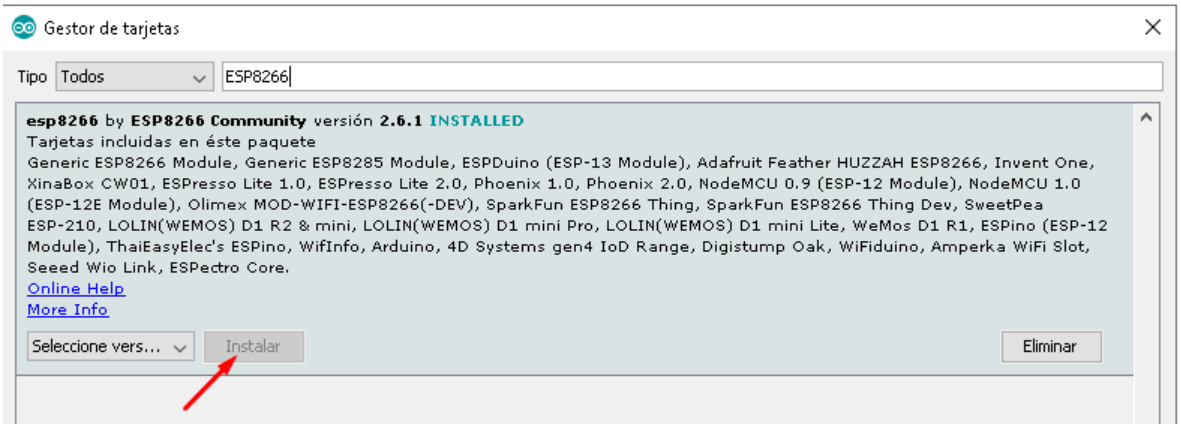
Ilustración 40 . Selección de tarjeta Arduino/Genuino Uno



Fuente: Autor

Allí en el gestor de tarjetas escribimos en la línea en blanco ESP8266 e instalamos la única opción que nos aparece, esp8266 by ESP8266 Community versión 2.6.1 como se muestra en la siguiente imagen y damos en Instalar.

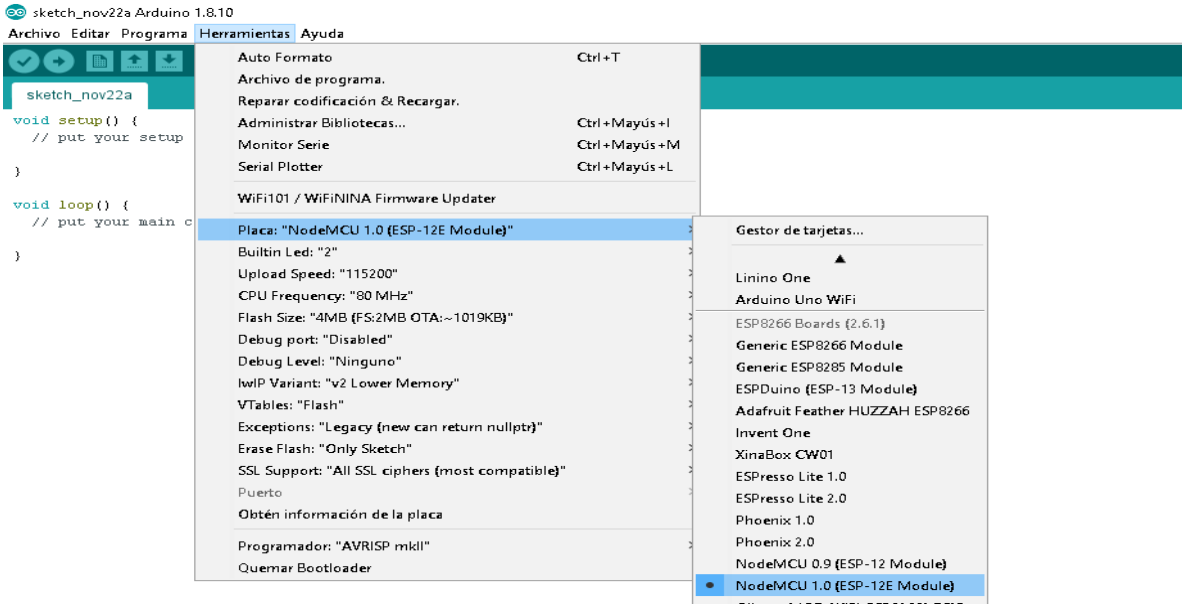
Ilustración 41. Instalación de la tarjeta en el IDE de Arduino



Fuente: Autor

Paso 5. Ya instala la tarjeta podremos seleccionar la placa a trabajar, para este caso trabajaremos con el NodeMCU v1.0 (ESP-12E Module).

Ilustración 42. Selección de la placa NodeMCU 1.0



Fuente: Autor

Paso 6. Instalación de las librerías

Para hacer las primeras pruebas utilizamos unas cuantas librerías de Arduino que se necesitan tener instaladas, para esto vamos al gestor de librerías de Arduino y buscamos las siguientes librerías:

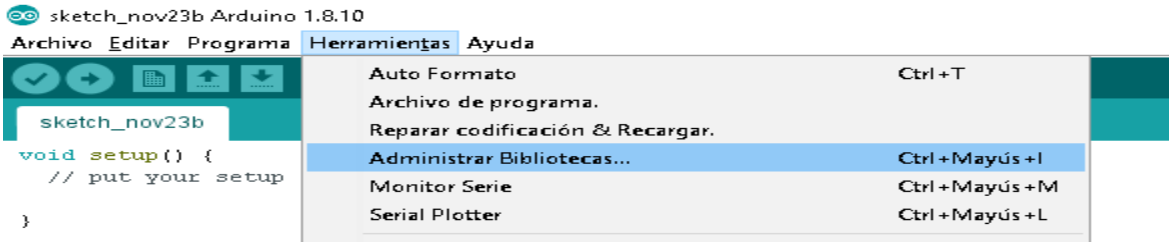
SPI.h

LoRa.h

Arduino soporta de serie el bus SPI, con una librería estándar se llama SPI que está incluida en el IDE de Arduino y que gestiona todas las complicaciones y el arbitraje del protocolo, y que como se indica ya viene por defecto instalada, por lo que no requiere su instalación. Por otra parte, si necesitamos instalar la librería de LoRa.h.

Para esto en el IDE de Arduino vamos a Herramientas, Administrar Bibliotecas…. y allí en el espacio en blanco escribimos LoRa,

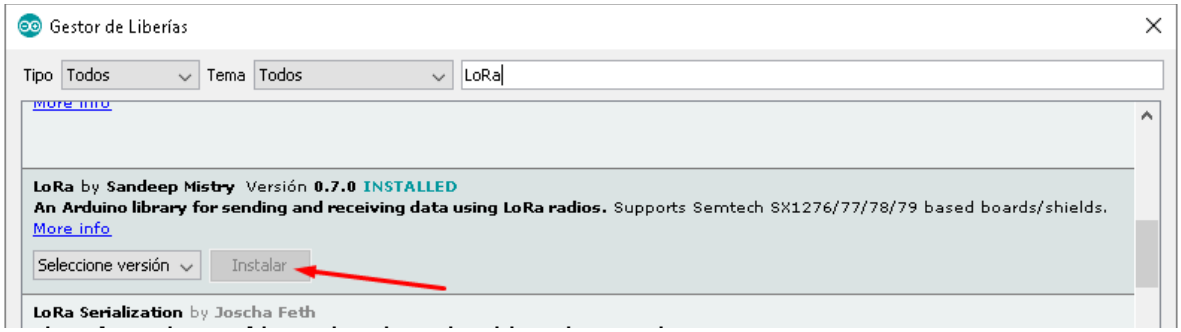
Ilustración 43. Instalación de librerías



Fuente: Autor

Deslizamos hacia abajo con el mouse sobre los resultados hasta encontrar una librería con el nombre LoRa by Sandeep Mistry Version 0.7.0 y damos en Instalar.

Ilustración 44. Selección de la librería utilizada

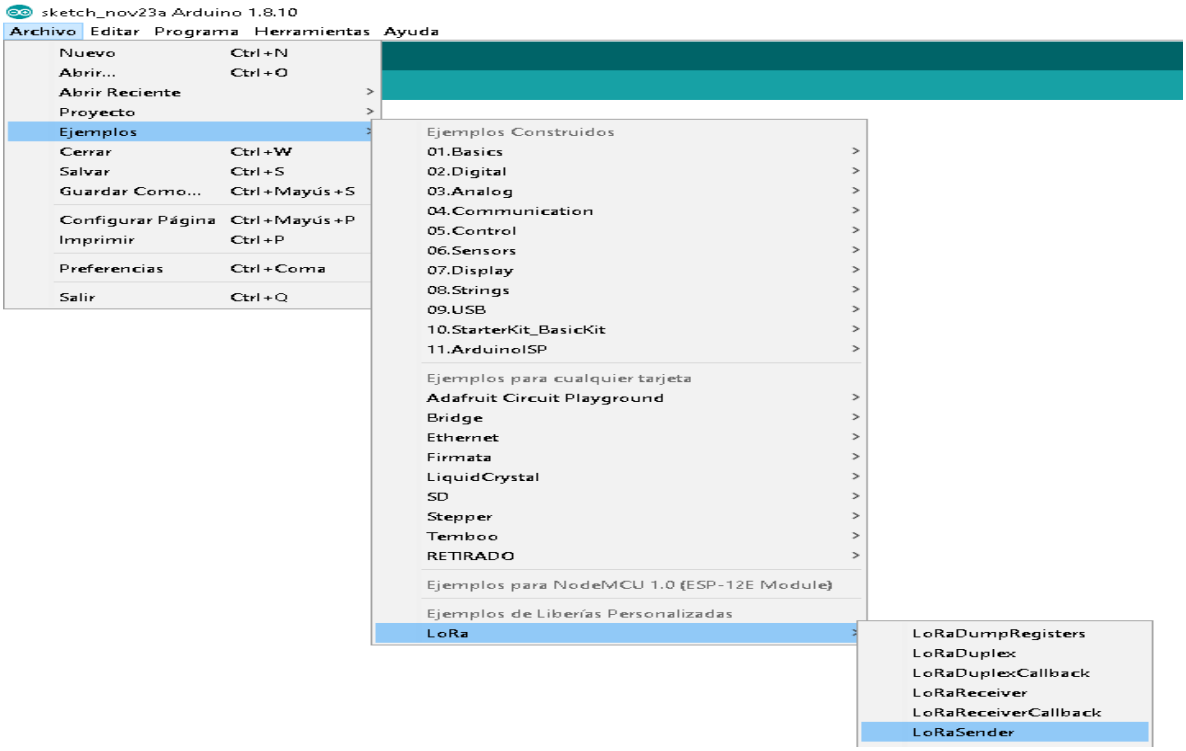


Fuente: Autor

Una vez se instala damos en cerrar.

Paso 7. Esta librería es la que permite enviar y recibir datos con LoRa entre dos módulos que tengan los mismos parámetros de configuración. Para usarla vamos a Archivo, Ejemplos, LoRa, LoRaSender como se muestra a continuación.

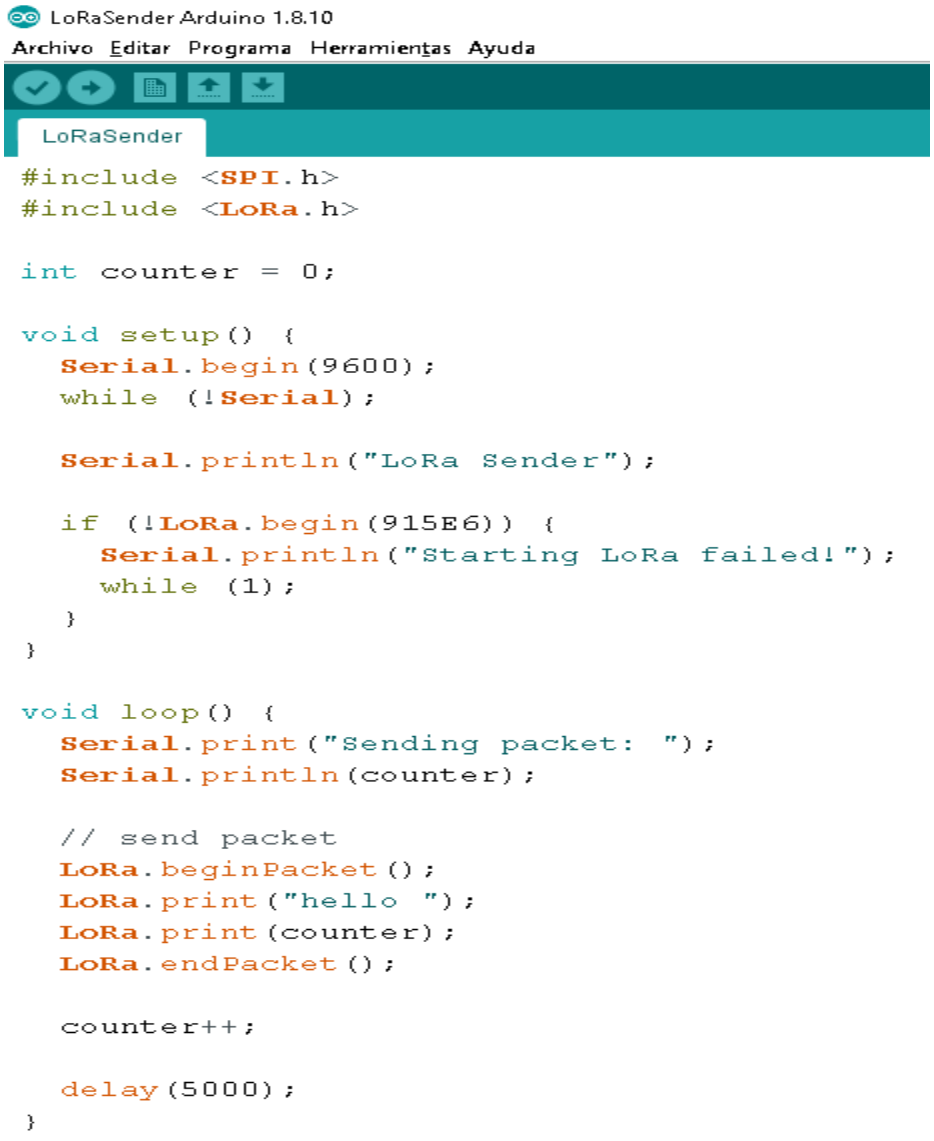
Ilustración 45. Ejemplo LoRaSender



Fuente: Autor

Nos aparece el ejemplo de LoRaSender con el código de este.

Ilustración 46. Programa LoRaSender



Fuente: Autor

Paso 8. Luego, se definen los pines utilizados por el módulo LoRa en el caso que se haya seguido el esquema de conexiones entre este y el NodeMCU de la **ilustración 12**. También se cambia el valor de la frecuencia a usar.

Esta tecnología de radio utiliza bandas ISM (Industrial Scientific & Medical/ Industrial, Científica y Medica). Estas bandas son de uso libre sin licencia, pero limitadas en potencia y tiempo de transmisión.

LoRa utiliza tres frecuencias principalmente:

433 MHz en Asia y América del sur
915 MHz en América
868 MHz en Europa

Esto es muy importante, si vamos a comprar un dispositivo LoRa, este debe contar con la frecuencia adecuada. Si no lo hacemos podemos acabar con un módulo que no pueda hablar con otros dispositivos LoRa y que incluso podemos estar infringiendo las regulaciones del uso del espectro radioeléctrico de donde nos encontremos. Como se expresa en el marco Legal para Colombia, los 433Mhz a usar en este proyecto hace parte de la banda libre disponible para radioaficionados que va desde los 430 Mhz hasta los 440 Mhz, lo que lo hace perfecto para su uso.

También es importante tener en cuenta que al ser una frecuencia muy baja si la comparamos con otras tecnologías inalámbricas como el bluetooth o Wi-Fi, que usan las bandas de los 2.4 GHZ y 5.8GHZ. La longitud de onda para una frecuencia de 433Mhz es de 0,69236 m.

Al usar estos 433MHZ que es una frecuencia menor, obtenemos una longitud de onda más larga permitiendo a las ondas atravesar edificios y diferentes obstáculos, además de la resistencia a las variaciones meteorológicas. Este tipo de propagación les permite alcanzar mayor cobertura, especialmente durante la noche, debido a los cambios que sufre la ionosfera durante ella.

$$\lambda = \frac{c}{f}$$

Donde λ = Longitud de onda (m)

c = Velocidad de la luz m/s ➡ 299.792.458 m/s

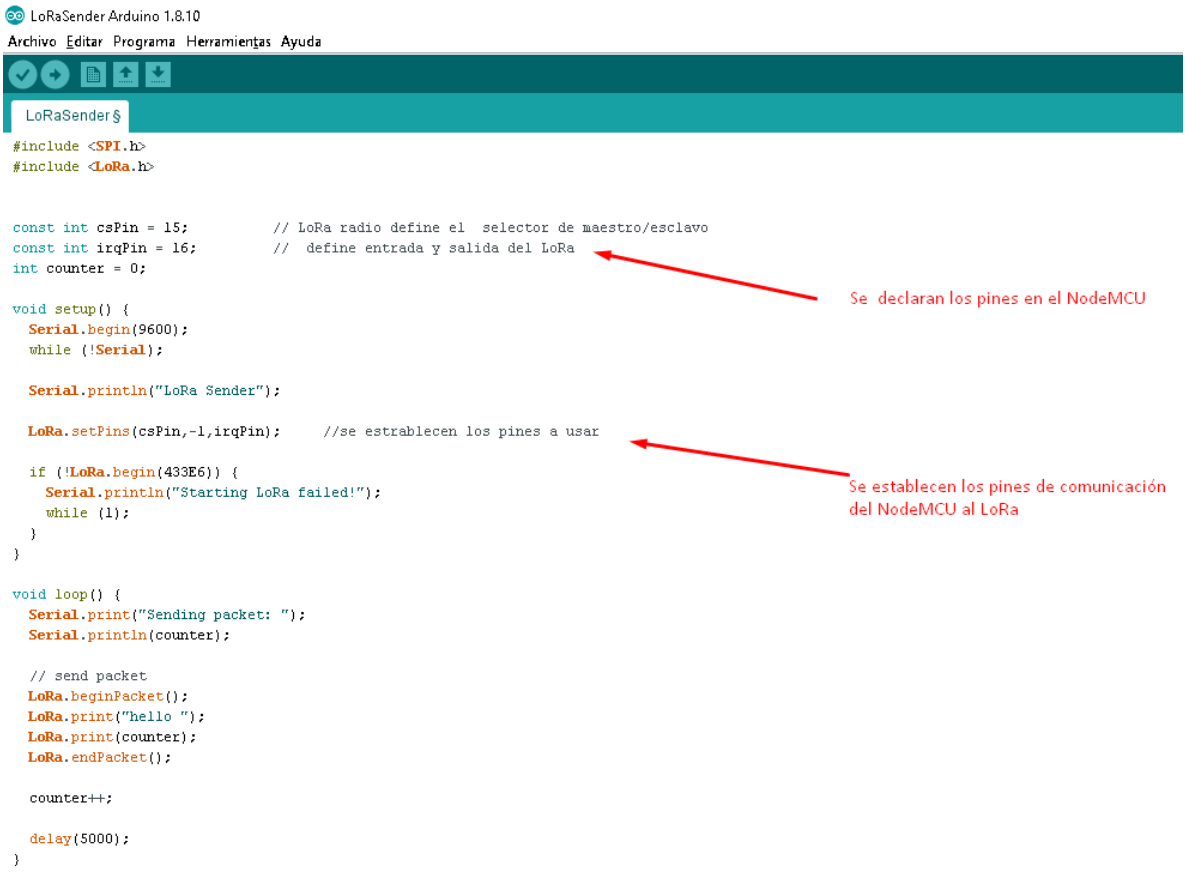
f = frecuencia (Hz) ➡ 433.000.000 Hz

$$\lambda = \frac{299.792.458}{433.000.000} = 0{,}69236 \text{ m}$$

Paso 9. Para definir los pines en el programa del LoRaSender se declaran como constantes enteras el csPin e irqPin a los pines 15 y 16 respectivamente del NodeMCU.

En el setup se escribe el comando que establece los pines de comunicación entre el NodeMCU y el LoRa con el LoRa.setPins (csPin, -1, irqPin).
La librería LoRa.h establece para el comando LoRa.setPins los pines (NSS, RESET, DIO0) en este orden en específico y donde csPin = NSS (Selector maestro/esclavo), -1 = RESET (Esto porque el RESET no va conectado entre los módulos ya que en principio se tenía conectado como se establece, pero no nos permitía el correcto funcionamiento de la placa por razones desconocidas), irqPin = DiO0 (Pin de entrada y salida de datos)

Ilustración 47. Cambios en programa LoRaSender




```
#include <SPI.h>
#include <LoRa.h>

const int csPin = 15;          // LoRa radio define el  selector de maestro/esclavo
const int irqPin = 16;         //  define entrada y salida del LoRa
int counter = 0;

void setup() {
  Serial.begin(9600);
  while (!Serial);

  Serial.println("LoRa Sender");

  LoRa.setPins(csPin,-1,irqPin);     //se estrablecen los pines a usar

  if (!LoRa.begin(433E6)) {
    Serial.println("Starting LoRa failed!");
    while (1);
  }
}

void loop() {
  Serial.print("Sending packet: ");
  Serial.println(counter);

  // send packet
  LoRa.beginPacket();
  LoRa.print("hello ");
  LoRa.print(counter);
  LoRa.endPacket();

  counter++;

  delay(5000);
}
```

Fuente: Autor

Paso 10. Una vez establecidas las configuraciones de los pasos anteriores, compilamos y subimos el programa a la placa según el puerto asignado.

Paso 11. Abrimos el monitor en donde observamos que se inicaliza la conexión del LoRa y el contador de los paquetes enviados.

Ilustración 48. Monitor LoRaSender

Fuente: Autor

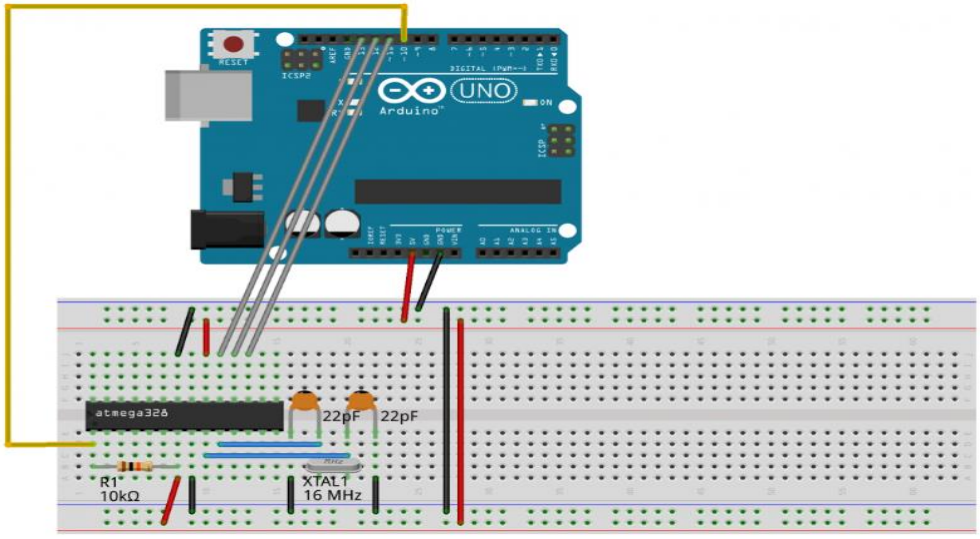
2.1.14. PREPARACIÓN BOOTLOADER ATMEGA328P

Debido a que se utilizará el Atmega328p de manera independiente a la placa de Arduino, debemos cargar el bootloader para proceder la programación de dicho microcontrolador, para ello se requieren los siguientes implementos:

- Arduino UNO.
- Protoboard.
- Atmega328p.
- Oscilador de 16MHz.
- Dos condensadores cerámicos de 22pF.
- Una resistencia de 10K.
- Cables.

Conectamos los pines 7 y 20 del ATmega a los 5V del Arduino mientras que el 8 y 22 a tierra. Procederemos a conectar el oscilador ya que mientras no tenga bootloader no se podrá utilizar el oscilador interno del ATmega, este se conecta entre los pines 9 y 10 con los condensadores de 22pF en cada extremo hacia tierra.
Los pines 11, 12 y 13 del ATmega se conectan a los mismos pines 11, 12 y 13 del Arduino, el pin de reset se conecta al pin 10 del Arduino y con una resistencia a 5V.
De esta manera el circuito quedaría de la siguiente manera:

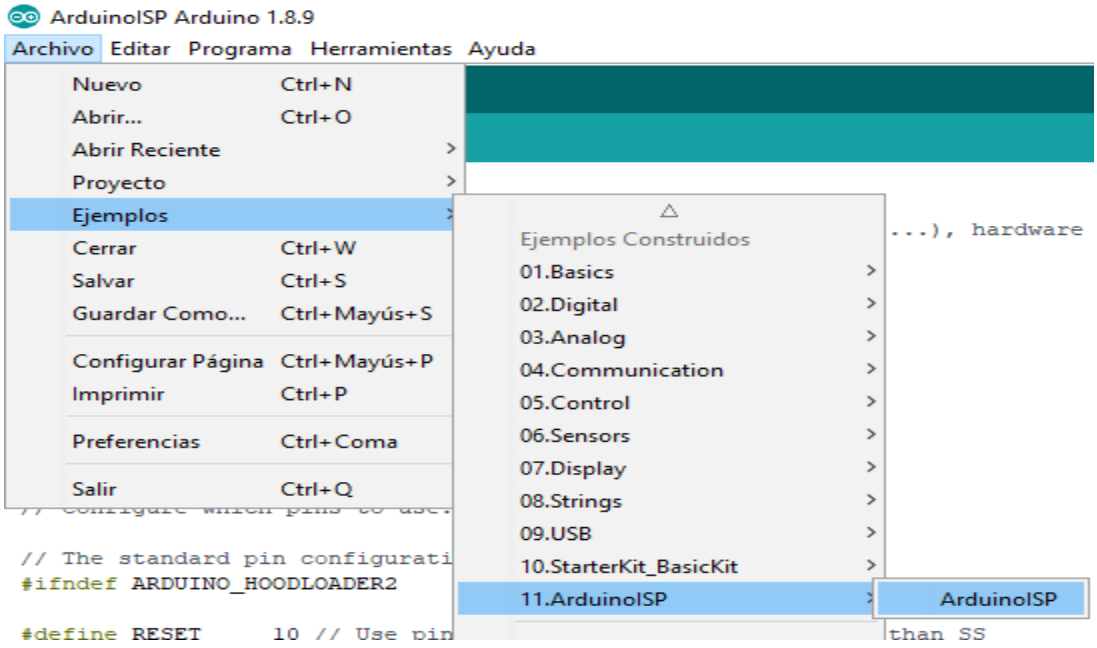
Ilustración 49. Conexiones ArduinoUNO- Protoboard.



Fuente: https://www.digilogic.es/atmega328p-independiente-de-arduino-standalone/

Una vez realizada la conexión conectamos el Arduino al computador, abrimos la IDE de Arduino y seleccionamos: Archivo > Ejemplos > Arduino ISP.

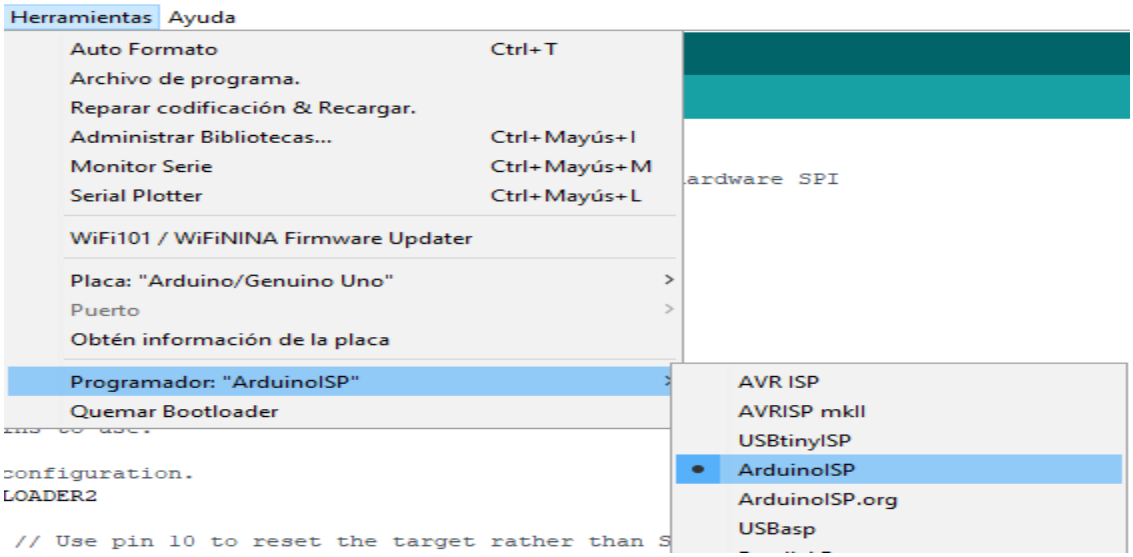
Ilustración 50. Ejemplo ISP- IDE Arduino



Fuente: Autor

Una vez abierto vamos al menú de herramientas: Programador > Arduino as ISP.

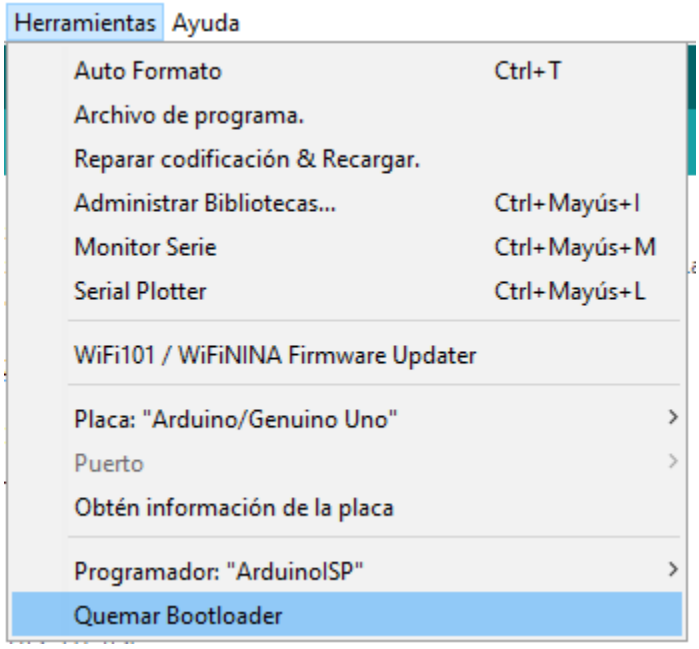
Ilustración 51. Selección Programador ArduinoISP



Fuente: Autor

Por último, de nuevo en herramientas seleccionamos Quemar bootloader.

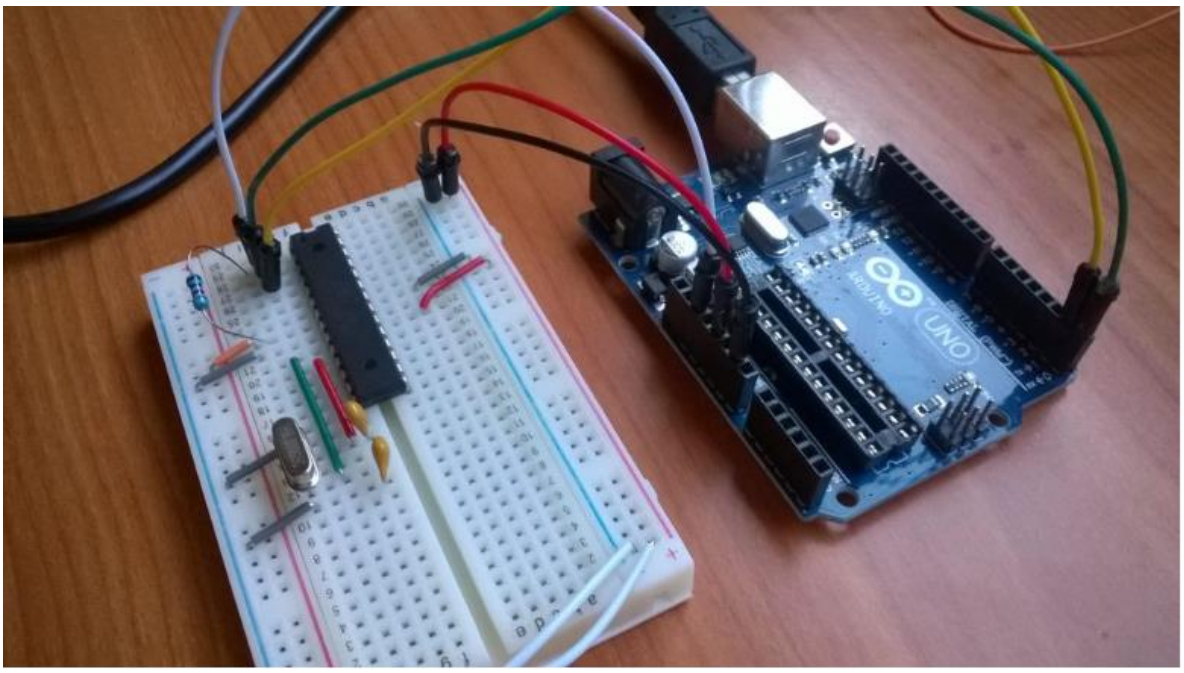
Ilustración 52. Selección Quemador Bootloader.



Fuente: Autor

Subimos el programa y de ésta manera ya podremos programar este microcontrolador. Algo importante que se debe saber es que una vez quemado el bootloader, para programar el ATmega debemos quitar el microcontrolador que tenga la placa de Arduino, de la siguiente manera:

Ilustración 53. Conexiones físicas ArduinUNO-Protoboard

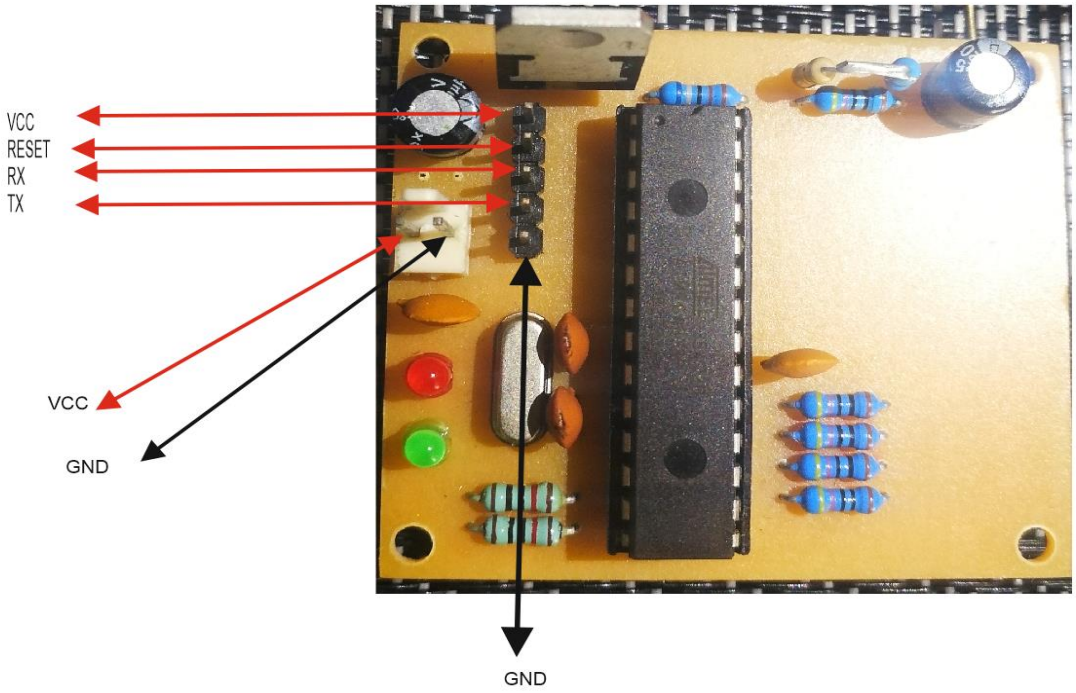


Fuente: Autor

2.1.15. CONEXIÓN DE LA PLACA ATMEGA328P- LORA

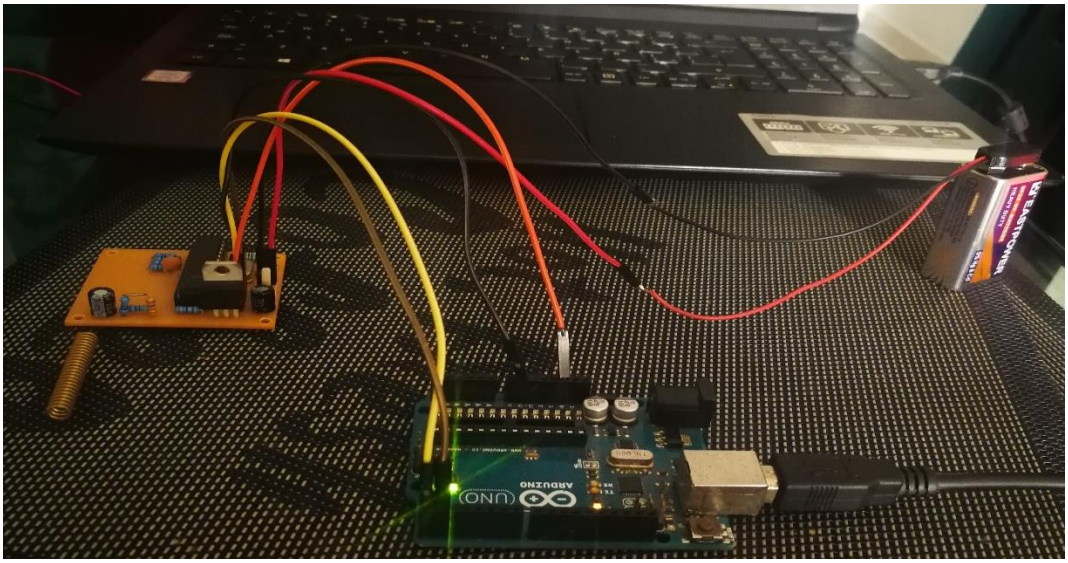
Para recibir los mensajes conectamos a otro computador la placa del ATmega328P, para esto se alimenta la placa con una batería externa de 9V al par de pines de entrada de alimentación de la placa. Para la comunicación de la placa con el computador usaremos la ayuda de un ArduinoUNO.

Ilustración 54. Conexión a la placa del Atmega328p



Fuente: Autor

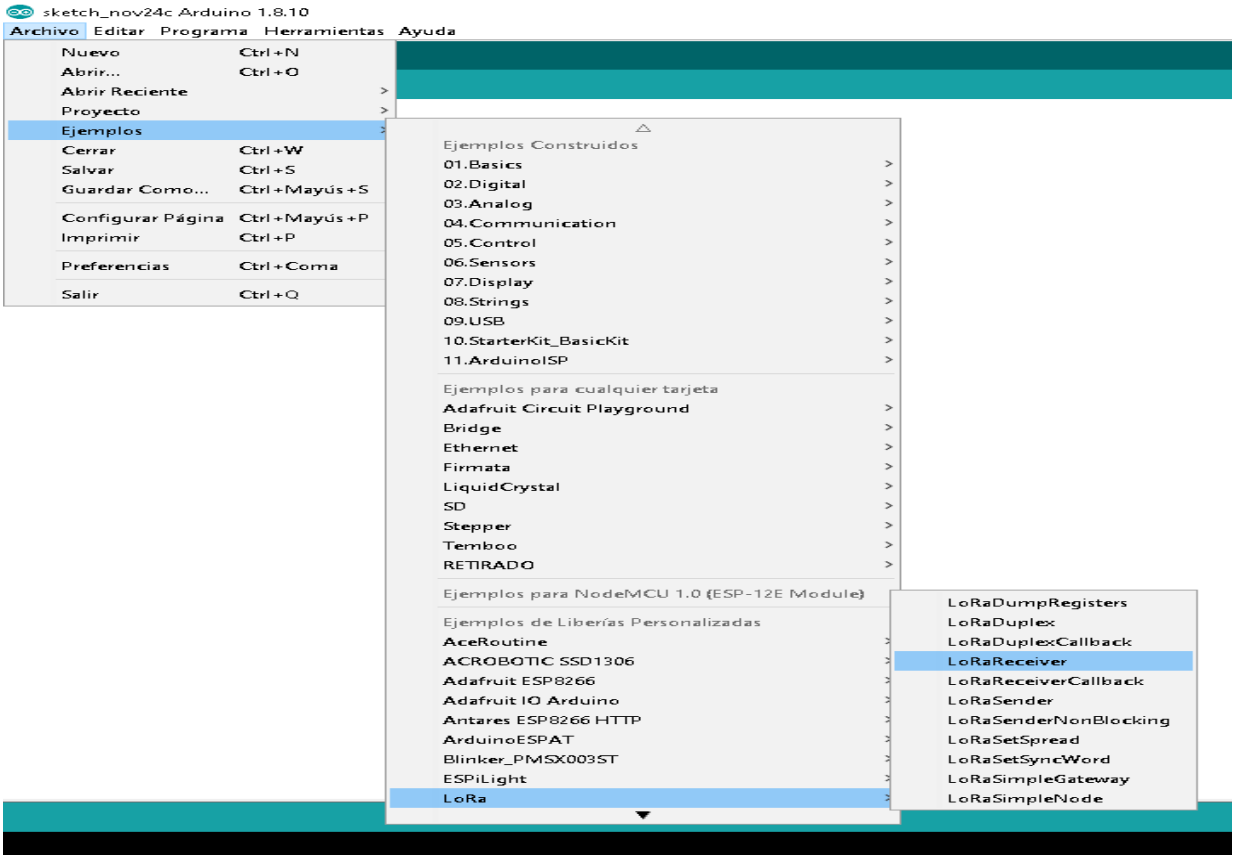
Ilustración 55. Conexión para programar la placa del Atmega328p



Fuente: Autor

Una vez hechas las conexiones se abre el programa para recibir los mensajes desde el IDE de Arduino, en Archivo, LoRa, Ejemplos, LoRaReceiver, como se muestra en la siguiente imagen:

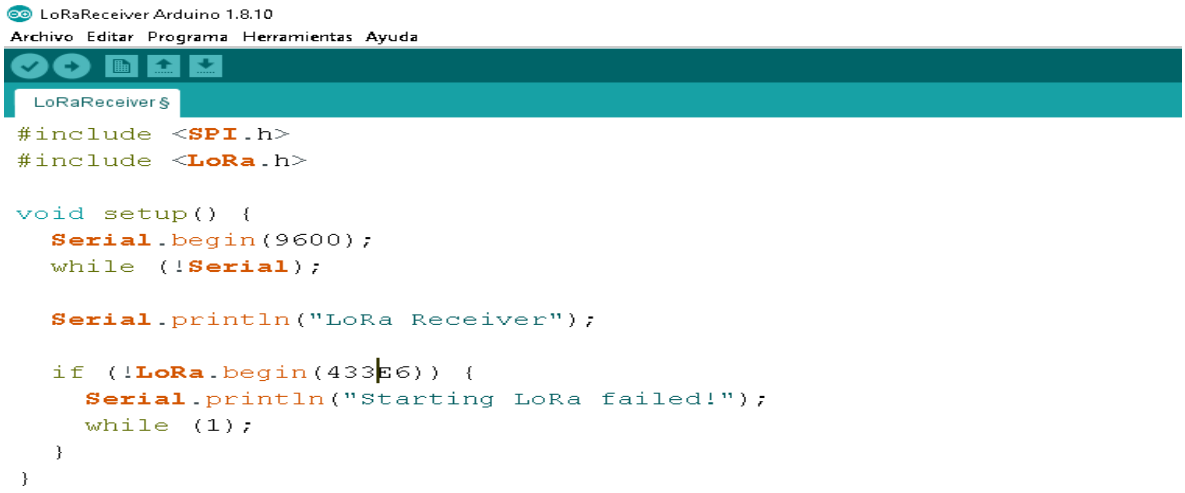
Ilustración 56. Ejemplo LoRaReceiver



Fuente: Autor

Abierto el programa de LoRaReceiver, cambiamos del código únicamente la frecuencia a trabajar de 915E6 a 433E6.

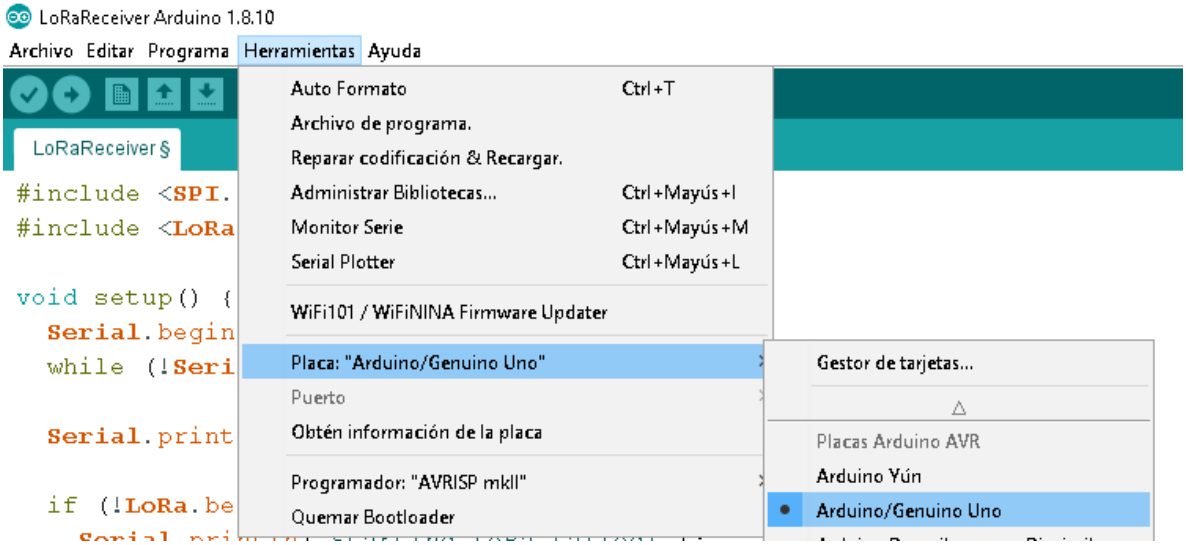
Ilustración 57. Cambio de frecuencia en el programa LoRaReceiver



Fuente: Autor

Antes de compilar el programa cambiamos la placa y seleccionamos la tarjeta Arduino/Genuino Uno.

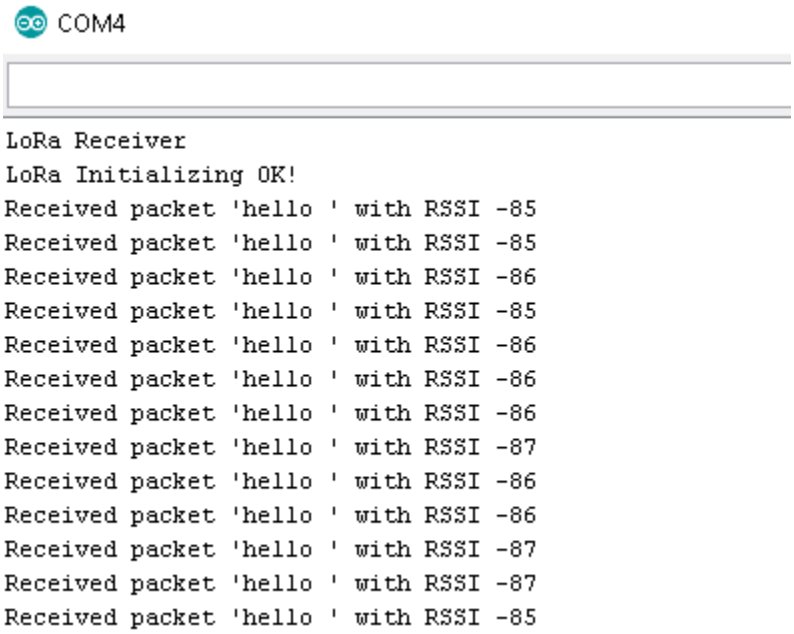
Ilustración 58. Selección de tarjeta para programar



Fuente: Autor

Establecidas las configuraciones de los pasos anteriores, compilamos y subimos el programa a la placa según el puerto asignado. Luego abrimos el monitor en donde podemos visualizar el mensaje recibido del NodeMCU junto a la sensibilidad de la señal.

Ilustración 59. Monitor serial de la recepción del mensaje



Fuente: Autor

2.1.16. PROGRAMACION RED MESH

2.1.17. PROGRAMACIÓN ATMEGA328P

1. Para realizar el programa de la red mesh utilizaremos como base el ejemplo de la librería LoRa.h llamado LoRaDuplex.

Ilustración 60. Librerías Atmega328p

```
#include <SPI.h>
#include <LoRa.h>
```

Fuente: Autor

2. En el programa de LoRaDuplex se declaran la cantidad de destinos necesarios o que conforman la red a diseñar. En este proyecto trabajaremos con 5 placas y descartando el módulo que programaremos como local, usaremos 4 destinos que deben estar nombrados de manera diferente para su identificación y de esta manera facilitar el direccionamiento de los paquetes de información en las posibles rutas de los nodos usados.

Ilustración 61. Direcciones de cada nodo

```
byte localAddress = 0xBB;
byte destination = 0xAA;
byte destination2 = 0xCC;
byte destination3 = 0xDD;
byte destination4 = 0xEE;
```

Fuente: Autor

3. Se declaran variables que se requieren a lo largo del desarrollo del programa.

4. Se cambia la frecuencia a usar de (915E6) a (433E6).

Ilustración 62. Cambio de frecuencia en el programa LoRaDuplex

```
void setup() {
  Serial.begin(9600);
  while (!Serial);

  Serial.println("LoRa Duplex");

  if (!LoRa.begin(433E6)) {
    Serial.println("LoRa init failed. Check your connections.");
    while (true);
  }
  Serial.println("LoRa init succeeded.");
}
```

Fuente: Autor

5. Una vez en el loop, se pretende que este se mantenga en constante percepción de datos provenientes de los demás nodos.

6. Si recibe un mensaje de alguno de los nodos extremos, es decir, de los NodeMCU, desde aquí enviará un dato a cada nodo disponible descartando el origen para guardar cada valor de la sensibilidad en una variable y hacer la respectiva comparación donde se decide la mejor ruta posible.

7. Al realizar las comparaciones dependiendo de la condición que se cumpla sobre las expuestas en el programa, se enviará el mensaje que proviene del origen al respectivo nodo y todas las variables utilizadas toman el valor de “0” a la espera de un nuevo resultado, mientras que la variable “rebote” será “1” para utilizarse más adelante.

8. Cuando el mensaje proviene de algún nodo repetidor, éste será enviado directamente al NodeMCU destino para evitar un posible bucle entre los ATmega.

Ilustración 63. Condiciones en la sección del loop

```
void loop() {
onReceive(LoRa.parsePacket());
onReceive1(LoRa.parsePacket());
if(envio==0x66){
   sendMessage(mensaje);
   Serial.println("enviando " + mensaje);
   envio=0;
   }
   else if(envio==0x11){
   sendMessage2(mensaje);
   Serial.println("enviando2 " + mensaje);
   envio=0;
    }
```

Fuente: Autor

9. Si la variable “rebote” está en “1”, un segundo después pasará a “0” con el fin de evitar que el mensaje enviado regrese y se imprima en el mismo origen.

Ilustración 64. Sección de espera para evitar que el mensaje rebote

```
    if(rebote==1){
    delay(1000);
    rebote=0;}
}
```

Fuente: Autor

10. Se crea la función “Sendmessage” encargada de enviar los mensajes. Esta función requiere la dirección de destino, la dirección local, contador de mensajes, longitud del mensaje y la salida. De esta función se crearán varias, nombradas de manera diferente para todas las posibles rutas.

De todas las posibles rutas de envío se creará funciones para cada una de ellas.

Ilustración 65. Función de enviar mensaje

```
void sendMessage(String outgoing) {
  LoRa.beginPacket();
  LoRa.write(destination);
  LoRa.write(localAddress);
  LoRa.write(msgCount);
  LoRa.write(outgoing.length());
  LoRa.print(outgoing);
  LoRa.endPacket();
  msgCount++;
}
```

Fuente: Autor

11. Ahora se realizan las funciones “onReceive” encargada de recibir los mensajes. De igual manera que la función de enviar mensajes, se requieren varias para cada ruta, pero se diferencian entre los que reciben de los nodos extremos (NodeMCU) y los internos (ATmega).

- En los nodos externos luego del proceso ilustrado que se mantiene del programa original, guardamos los valores del Rssi y Snr para hacer las comparaciones indicadas en el paso 6, se guarda el mensaje que llega en una variable para utilizarla en cualquier parte del programa y por último se incluye una condición para que se identifique si el mensaje es de prueba o el mensaje a enviar, de esta manera envía el dato requerido.

Ilustración 66. Especificación de la dirección del origen del mensaje

```
if (recipient != localAddress && recipient != 0xAA) {
  Serial.println("This message is not for me.");
  return;
}
```

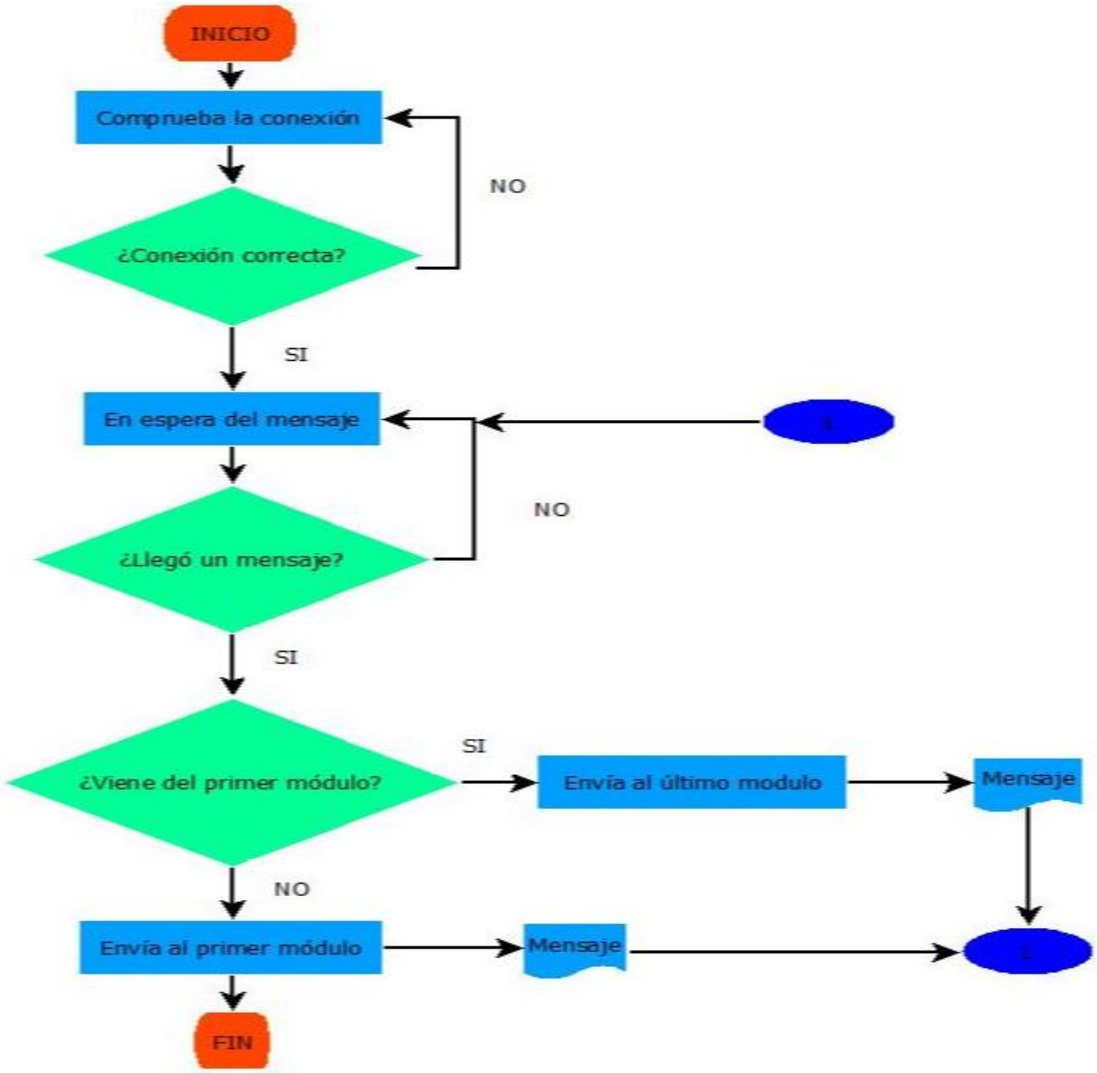
Fuente: Autor

- Los nodos internos luego del proceso indicado previamente y guardar los valores para las respectivas comparaciones, se utilizan “4” condiciones más que en los externos para evitar que el mensaje regrese a su origen si debe desviarse de una ruta directa al otro NodeMCU.

2.1.18. DIAGRAMA DE FLUJO DE LOS NODOS REPETIDORES ATMEGA328P

En resumen, la secuencia de la lógica para cada uno de los programas realizados se muestra en los siguientes diagramas de flujo:

Ilustración 67. Diagrama de flujo del ATmega328P como repetidor.



Fuente: Autor

2.1.19. PROGRAMACIÓN NODEMCU CON PULSADOR

Paso 1. Para del primer NodeMCU utilizaremos como base de nuevo el ejemplo de la librería LoRa.h llamado LoRaDuplex.

Ilustración 68. Librerías utilizadas en el NodeMCU con pulsador

```
#include <SPI.h>
#include <LoRa.h>
```

Fuente: Autor

Paso 2. Se definen los pines en el programa declarando como constantes enteras el csPin e irqPin a los pines 15 y 16 respectivamente del NodeMCU, además otro pin para conectar un pulsador el cual simulará el envío de datos de un sensor decidiendo nosotros cuando se enviará el mensaje.

Ilustración 69. Declaración de pines del NodeMCU

```
const int csPin = 15;
const int irqPin = 16;
const int buttonPin = 5;
int buttonState = 0;
```

Fuente: Autor

Paso 3. En el programa de igual manera que la red mesh se declaran la cantidad de destinos necesarios o que conforman la red a diseñar. Que deben estar nombrados de manera diferente para su identificación y de esta manera facilitar el direccionamiento de los paquetes de información en las posibles rutas de los nodos usados.

Ilustración 70. Direcciones a las que el modulo puede dirigirse

```
byte localAddress = 0xEE;
byte destination = 0xBB;
byte destination2 = 0xCC;
byte destination3 = 0xDD;
```

Fuente: Autor

Paso 4. Se declaran variables que se requieren a lo largo del desarrollo del programa.

Paso 5. Se cambia la frecuencia a usar de (915E6) a (433E6) y el while(true) dentro de este ciclo se sustituye por un delay (1000) por limitantes del NodeMCU.

Paso 6. Cuando se oprime el pulsador se envía un dato a cada nodo disponible a excepción del último, para guardar cada valor de la sensibilidad en una variable y hacer la respectiva comparación donde se decide la mejor ruta posible.

Ilustración 71. Función que envía los mensajes de prueba

```
void loop() {
buttonState = digitalRead(buttonPin);
  if (buttonState==LOW){
  String message = ".";
  sendMessage(message);
  Serial.println("enviando " + message);
  onReceive(LoRa.parsePacket());
  delay(500);
  sendMessage2(message);
  onReceive2(LoRa.parsePacket());
  delay(500);
  sendMessage3(message);
  onReceive3(LoRa.parsePacket());
  delay(500);
```

Fuente: Autor

Paso 7. Al realizar las comparaciones dependiendo de la condición que se cumpla sobre las expuestas en el programa, se enviará el mensaje al respectivo nodo y todas las variables utilizadas toman el valor de “0” a la espera de un nuevo resultado, mientras que la variable “rebote” será “1”.

Ilustración 72. Condición de selección de nodo

```
if ((B < A) && (B < C) && (contenvio == 1)) {
  String message2 = "hola";
  sendMessage(message2);
  contenvio = 0;
  A = 0;
  B = 0;
  C = 0;
  D = 0;
  E = 0;
  F = 0;
  G = 0;
  H = 0;
  I = 0;
  rebote = 1;
}
```

Fuente: Autor

Paso 8. Si la variable “rebote” está en “1” y “contenvio” en “0”, un segundo después “rebote” pasará a “0” con el fin de evitar que el mensaje enviado regrese y se imprima en el mismo origen.

Ilustración 73. Función de espera para evitar que el mensaje rebote

```
if((contenvio==0) && (rebote==1)){
  delay(1000);
  rebote=0;}
```

Fuente: Autor

Paso 9. Cuando las variables “rebote” y “contenvio” están en “0”, se mantiene una constante recepción de mensajes de cualquier nodo y las variables de las señales las mantiene en “0”.

Ilustración 74. En espera de recibir un mensaje

```
if((contenvio==0) && (rebote==0)){
 onReceive(LoRa.parsePacket());
 onReceive2(LoRa.parsePacket());
 onReceive3(LoRa.parsePacket());
 A=0;
 B=0;
 C=0;
 D=0;
 E=0;
 F=0;
 G=0;
 H=0;
 I=0;
   }
 }
```

Fuente: Autor

Paso 10. Se crea la función “Sendmessage” encargada de enviar los mensajes. Esta función requiere la dirección de destino, la dirección local, contador de mensajes, longitud del mensaje y la salida. De esta función se crearán varias, nombradas de manera diferente para todas las posibles rutas.

De todas las posibles rutas de envío se creará funciones para cada una de ellas.

Ilustración 75. Función de enviar mensaje

```
void sendMessage3(String outgoing) {
  LoRa.beginPacket();
  LoRa.write(destination3);
  LoRa.write(localAddress);
  LoRa.write(msgCount);
  LoRa.write(outgoing.length());
  LoRa.print(outgoing);
  LoRa.endPacket();
  msgCount++;
}
```

Fuente: Autor

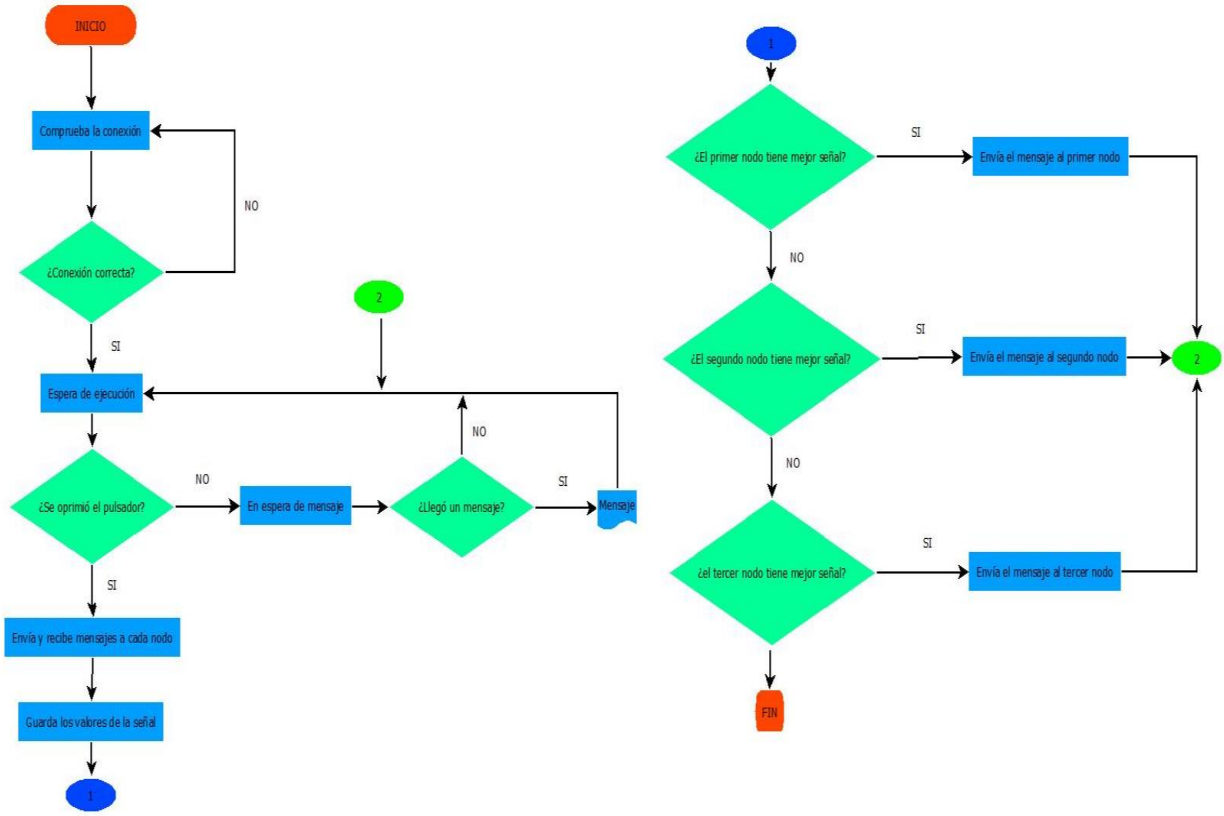
Paso 11. Ahora se realizan las funciones “onReceive” encargada de recibir los mensajes. De igual manera que la función de enviar mensajes, se requieren varias para cada ruta teniendo la condición de que solo se imprimirá el mensaje si es distinto al mensaje de prueba.

Ilustración 76. Especificación de la dirección del origen del mensaje

```
// if the recipient isn't this device or broadcast,
if (recipient != localAddress && recipient != 0xBB) {
  Serial.println("This message is not for me.");
  return;
}
```

Fuente: Autor

Ilustración 77. Diagrama de flujo del NodeMCU con el pulsador para él envió de datos.



Fuente: Autor

2.1.20. PROGRAMA NODEMCU NUBE

- Para este segundo NodeMCU aún se utilizará la librería LoRa.h para tomar el ejemplo de LoRaDuplex, pero se añadirá la librería ESP8266WiFi.h que nos permitirá conectarnos a una red wifi además se necesitará la comunicación con una plataforma de CloudMQTT por medio de la librería PubSubClient.h.

Ilustración 78. Librerías a utilizar en el NodeMCU Para la nube

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <SPI.h>
#include <LoRa.h>
```

Fuente: Autor

- Se escriben los datos necesarios para la conexión a la red que se vaya a utilizar para conectarse a internet.

Ilustración 79. Datos para conectarse a una red Wi-Fi

```
const char *ssid = "David RC";
const char *password = "c0l0mbi@87";
char   SERVER[50]   = "soldier.cloudmqtt.com";
int    SERVERPORT   =  18569;
String USERNAME = "LoRa";
char   PASSWORD[50] = "xxxxxxx";
```

Fuente: Autor

- Se definen los pines en el programa declarando como constantes enteras el csPin e irqPin a los pines 15 y 16 respectivamente del NodeMCU,

```
const int csPin = 15;
const int irqPin = 16;
```

- En el programa de igual manera que los anteriores se declaran la cantidad de destinos necesarios o que conforman la red a diseñar. Que deben estar nombrados de manera diferente para su identificación y de esta manera facilitar el direccionamiento de los paquetes de información en las posibles rutas de los nodos usados.

Ilustración 80. Direcciones a las que puede enviar un mensaje

```
byte localAddress = 0xAA;
byte destination = 0xBB;
byte destination2 = 0xCC;
byte destination3 =  0xDD;
```

Fuente: Autor

- Se declaran variables que se requieren a lo largo del desarrollo del programa.

- Se realiza la función callback, la cual nos permite recibir algún mensaje que sea enviado desde la nube y sube un contador de mensajes a “1”.

Ilustración 81. Función para recibir mensajes desde la nube

```
WiFiClient espClient;
PubSubClient client(espClient);

void callback(char* Topic, byte* payload, unsigned int length) {

  char PAYLOAD[5] = "    ";

  Serial.print("Mensaje Recibido: [");
  Serial.print(Topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++) {
    PAYLOAD[i] = (char)payload[i];
  }
  Serial.println(PAYLOAD);

  if (String(Topic) ==  String(SALIDAMENSAJE)) {
    if (payload[1] == 'N'){
      contmensaje=1;
    }
  }
}
```

Fuente: Autor

- Se ejecuta la conexión a la nube, en caso de se intenta conectar cada 5 segundos.

Ilustración 82. Función para conectarse a CloudMQTT

```
void reconnect() {
  uint8_t retries = 3;
  while (!client.connected()) {
    Serial.print("Intentando conexion MQTT...");
    String clientId = "ESP8266Client-";
    clientId += String(random(0xffff), HEX);
    USERNAME.toCharArray(LORA, 50);
    if (client.connect("", LORA, PASSWORD)) {
      Serial.println("conectado");
      client.subscribe(SALIDAMENSAJE);
    } else {
      Serial.print("fallo, rc=");
      Serial.print(client.state());
      Serial.println(" intenta nuevamente en 5 segundos");
      delay(5000);
    }
    retries--;
    if (retries == 0) {
      while(1);
    }
  }
}
```

Fuente: Autor

- Se cambia la frecuencia a usar de (915E6) a (433E6) y el while(true) dentro de este ciclo se sustituye por un delay (1000) por limitantes del NodeMCU.

- Se asigna una IP al NodeMCU, se determina la puerta de enlace y la máscara de subred de la red wifi.

Ilustración 83. Función que conecta a la red Wi-Fi

```
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED and contconexion <50) {
  ++contconexion;
  delay(500);
  Serial.print(".");
}
if (contconexion <50) {
    IPAddress ip(192,168,1,156);
    IPAddress gateway(192,168,1,1);
    IPAddress subnet(255,255,255,0);
    WiFi.config(ip, gateway, subnet);

    Serial.println("");
    Serial.println("WiFi conectado");
    Serial.println(WiFi.localIP());
}
else {
    Serial.println("");
    Serial.println("Error de conexion");
}
```

Fuente: Autor

- Se llaman las variables del servidor y el puerto del servidor asignados cuando se crea la nube en el sistema CloudMQTT, luego se realiza una conversión de la información que llega de la nube en variables que podamos leer en el programa para poder trabajarlas.

- Si el NodeMCU no conecta a internet vuelve a la función de reconectar, seguido de una condición cuando el contador de mensajes de la función callback está en “1” se envía un dato a cada nodo disponible a excepción del primero, para guardar cada valor de la sensibilidad en una variable y hacer la respectiva comparación donde se decide la mejor ruta posible.

Ilustración 84. Conecta a internet y envía mensajes de prueba

```
void loop() {

  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  if (contmensaje == 1) {
    String message = ".";
    sendMessage(message);
    Serial.println("enviando " + message);
    onReceive(LoRa.parsePacket());
    delay(500);
    sendMessage2(message);
    onReceive2(LoRa.parsePacket());
    delay(500);
    sendMessage3(message);
    onReceive3(LoRa.parsePacket());
    delay(500);
    contenvio = 1;
    contmensaje = 0;;
```

Fuente: Autor

- Al realizar las comparaciones dependiendo de la condición que se cumpla sobre las expuestas en el programa, se enviará el mensaje al respectivo nodo y todas las variables utilizadas toman el valor de “0” a la espera de un nuevo resultado, mientras que la variable “rebote” será “1”.
-

Ilustración 85. Función que envía el mensaje al módulo seleccionado

```
if ((A < B) && (A < C) && (contenvio == 1)) {
  String message2 = "hola";
  sendMessage3(message2);
  contenvio = 0;
  A = 0;
  B = 0;
  C = 0;
  D = 0;
  E = 0;
  F = 0;
  G = 0;
  H = 0;
  I = 0;
  rebote = 1;
}
```

Fuente: Autor

- Si la variable “rebote” está en “1” y “contenvio” en “0”, un segundo después “rebote” pasará a “0” con el fin de evitar que el mensaje enviado regrese y se imprima en el mismo origen.

Ilustración 86. Función de espera para que el mensaje no rebote

```
if((contenvio==0) && (rebote==1)){
  delay(1000);
  rebote=0;}
```

Fuente: Autor

- Cuando las variables “rebote” y “contenvio” están en “0”, se mantiene una constante recepción de mensajes de cualquier nodo y las variables de las señales las mantiene en “0”.

Ilustración 87. En espera de recibir un mensaje

```
if((contenvio==0) && (rebote==0)){
 onReceive(LoRa.parsePacket());
 onReceive2(LoRa.parsePacket());
 onReceive3(LoRa.parsePacket());
 A=0;
 B=0;
 C=0;
 D=0;
 E=0;
 F=0;
 G=0;
 H=0;
 I=0;
   }
```

Fuente: Autor

- Se crea la función “Sendmessage” encargada de enviar los mensajes. Esta función requiere la dirección de destino, la dirección local, contador de mensajes, longitud del mensaje y la salida. De esta función se crearán varias, nombradas de manera diferente para todas las posibles rutas.

De todas las posibles rutas de envío se creará funciones para cada una de ellas.

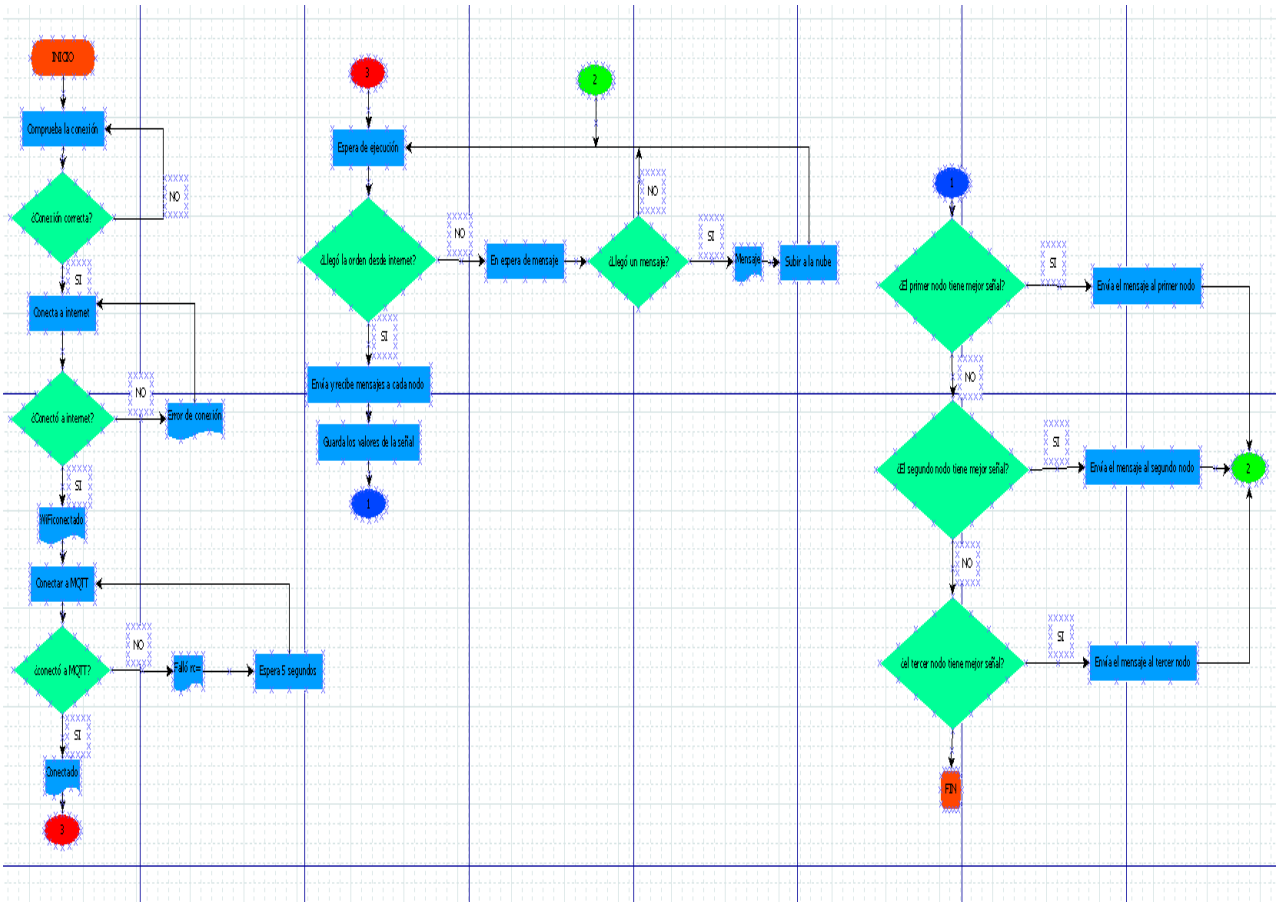
Ilustración 88. Función de enviar mensaje

```
void sendMessage3(String outgoing) {
  LoRa.beginPacket();
  LoRa.write(destination3);
  LoRa.write(localAddress);
  LoRa.write(msgCount);
  LoRa.write(outgoing.length());
  LoRa.print(outgoing);
  LoRa.endPacket();
  msgCount++;
}
```

Fuente: Autor

- Ahora se realizan las funciones “onReceive” encargada de recibir los mensajes. De igual manera que la función de enviar mensajes, se requieren varias para cada ruta teniendo la condición de que solo se subirá a internet si el mensaje si es distinto al mensaje de prueba.

Ilustración 89. Diagrama de flujo del NodeMCU para él envió de datos a la nube.



Fuente: Autor

2.1.21. CONFIGURACION SERVIDOR MQTT

Como se evidencio en los objetivos del proyecto y el diseño de la red, el subir los mensajes la nube y viceversa hace parte importante y una de las principales razones del uso del NodeMCU basado en el ESP8266 con su chip Wi-Fi que hacen posible esta conexión con la ayuda de una herramienta muy fácil de usar y lo mejor, gratuita y suficiente para los pequeños datos a enviar.

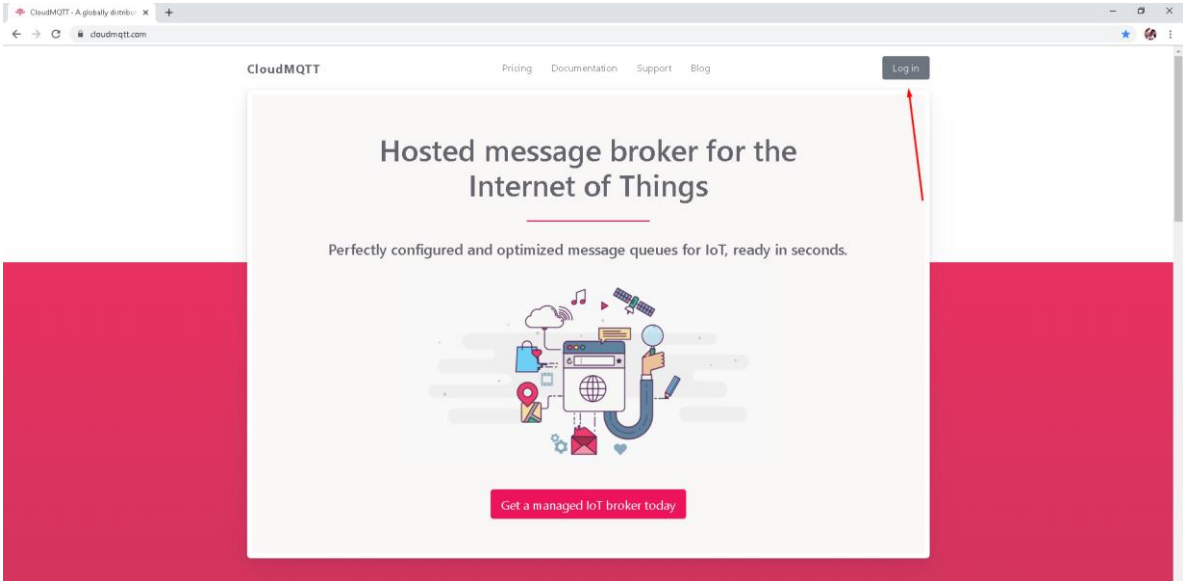
CloudMQTT es una solución perfecta para la mensajería de el "Internet de las cosas" entre sensores de baja potencia o dispositivos móviles, computadoras integradas o plataformas de desarrollo como Arduino.

Los pasos del uso de esta herramienta de muestran a continuación:

Paso 1. Desde cualquier computador con conexión a internet, abrimos un navegador web y allí accedemos a www.cloudmqtt.com.

Paso 2. Luego nos dirigimos a Login in.

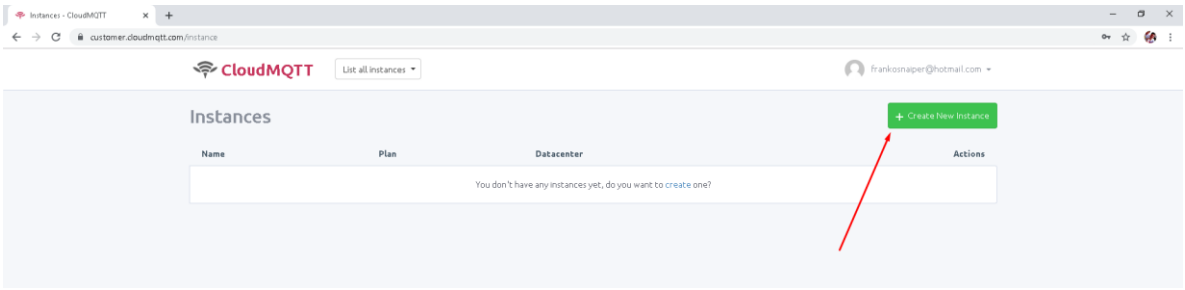
Ilustración 90. Interfaz de entrada a la nube



Fuente: Autor

Paso 3. Una vez allí accedemos de la manera que queramos, con nuestro correo, cuenta Gmail o cuenta GitHub y confirmamos la cuenta en la bandeja de entrada de nuestro correo electrónico. Ya en el panel de control vamos a + Create New Instance.

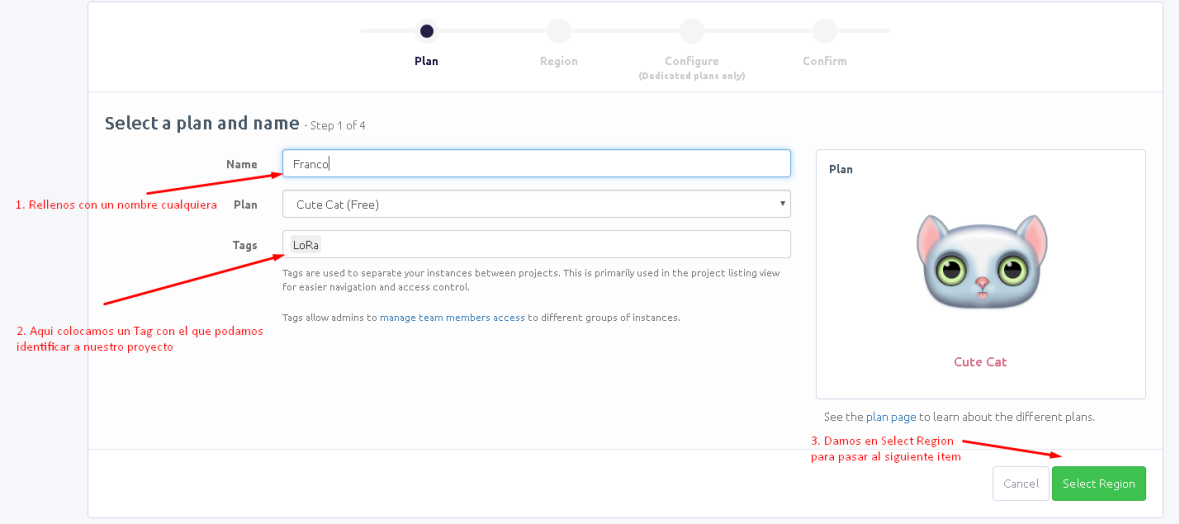
Ilustración 91. Creación de una nueva instancia



Fuente: Autor

Paso 4. rellenamos los campos de Name, dejamos el plan Cute Cat (Free) que nos permite usar los servicios de CloudMQTT de manera gratuita con hasta 10 usuarios, y conexión de hasta 10kbit/s con soporte 24/7. En caso de que se quiera ampliar las conexiones esta plataforma ofrece métodos de pago para hasta 10.000 conexiones con soporte vía correo y teléfono, además sin límite en el ancho de banda a usar, pero para nuestro caso el que nos brinda de manera gratuita es suficiente para el desarrollo del proyecto.

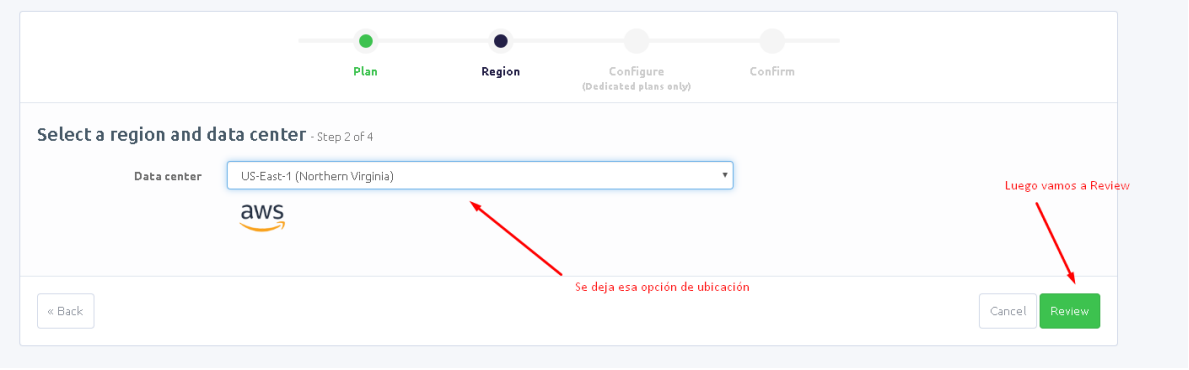
Ilustración 92. Datos y selección de plan



Fuente: Autor

Paso 5. En la siguiente pantalla solo damos en Review para continuar.

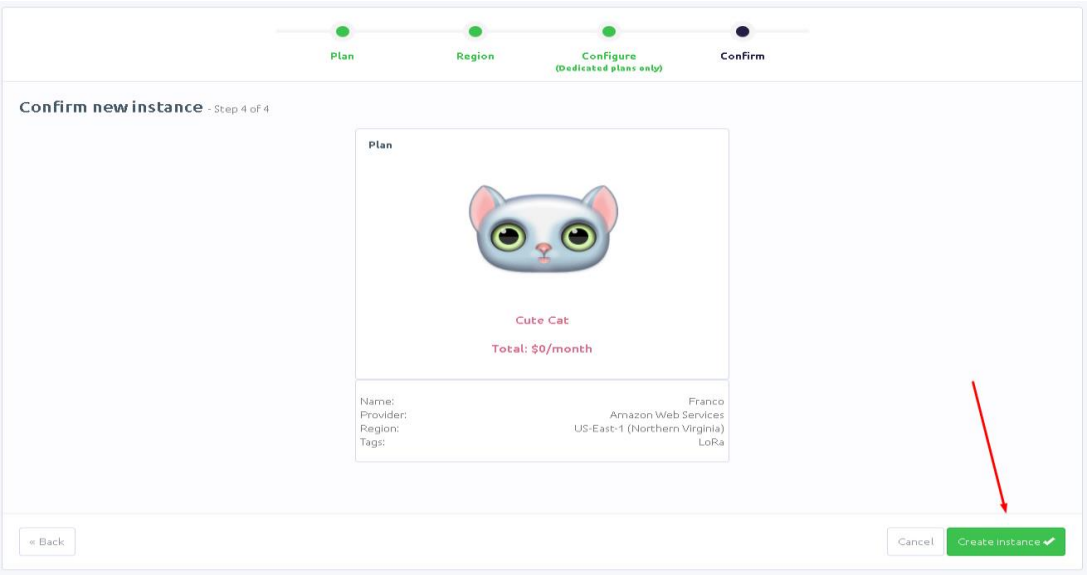
Ilustración 93. Selección del lugar del servidor



Fuente: Autor

Paso 6. Luego damos en Create Instance.

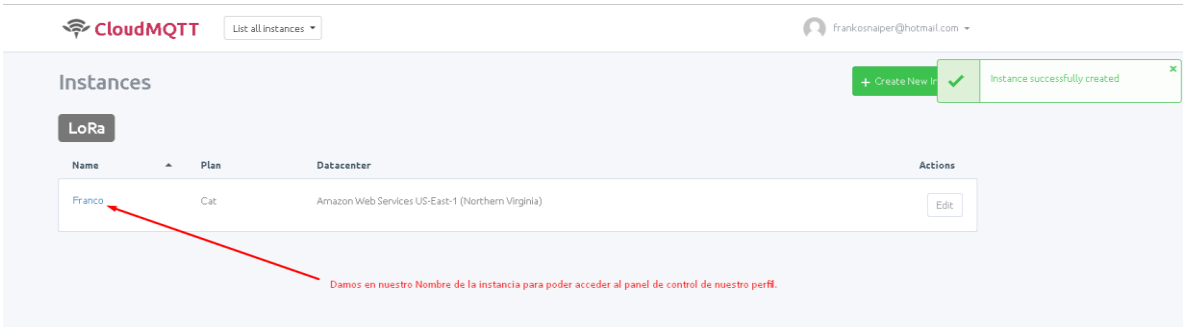
Ilustración 94. Finalización de la creación de instancia



Fuente: Autor

Con esto damos por finalizada la creación de los parámetros de la cuenta y podemos empezar trabajar. Solo vamos en el nombre de la instancia creada como se muestra en la siguiente ilustración.

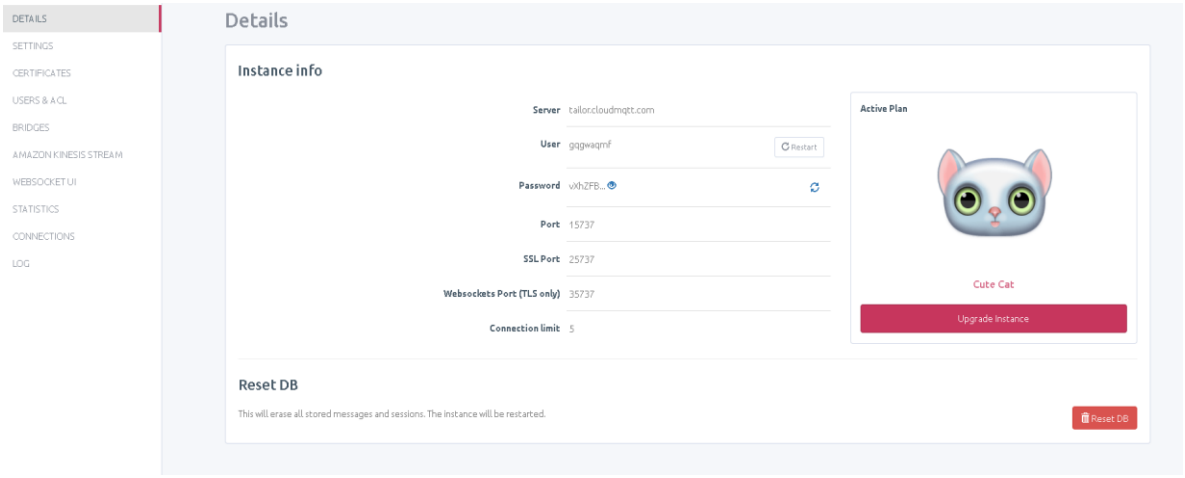
Ilustración 95. Se muestran todas nuestras instancias creadas



Fuente: Autor

Paso 7. Adentro de nuestra estancia creada nos aparecen los siguientes datos.

Ilustración 96. Datos de nuestra instancia

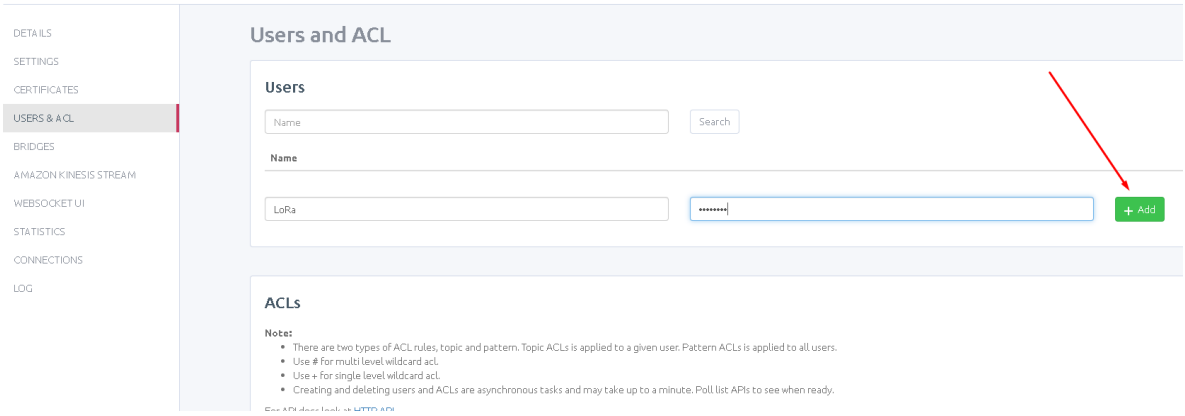


Fuente: Autor

De esos datos solo nos interesa la dirección el Servidor “Server” y el puerto “Port” que irán en el programa del NodeMCU que tendrá la conexión a internet y por consiguiente a la nube.

Paso 8. Luego vamos nos dirigimos a USERS & ACL y creamos un usuario y contraseña que son los que usaremos en el programa para acceder a nuestro perfil de la nube y para finalizar damos en +Add.

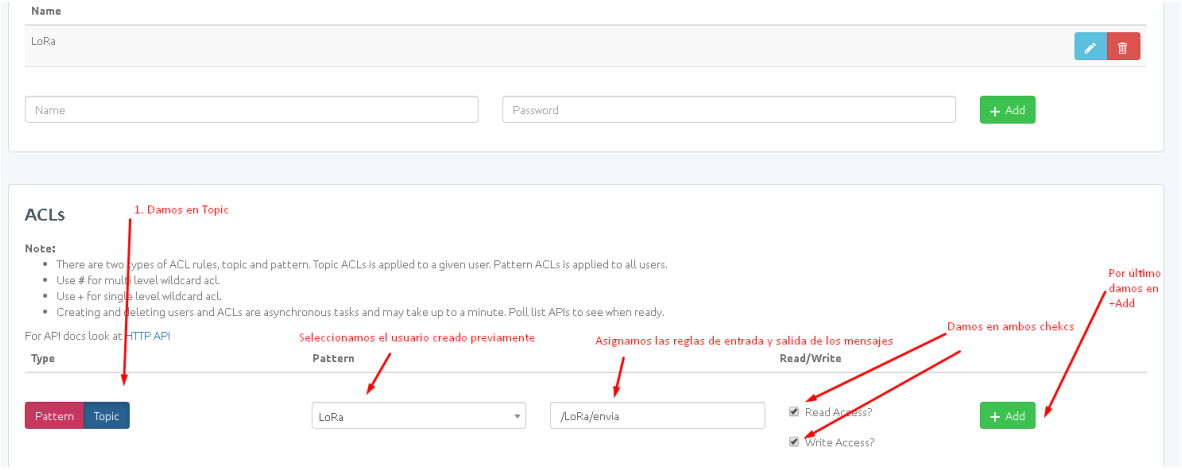
Ilustración 97. Asignación de usuario y contraseña de la instancia



Fuente: Autor

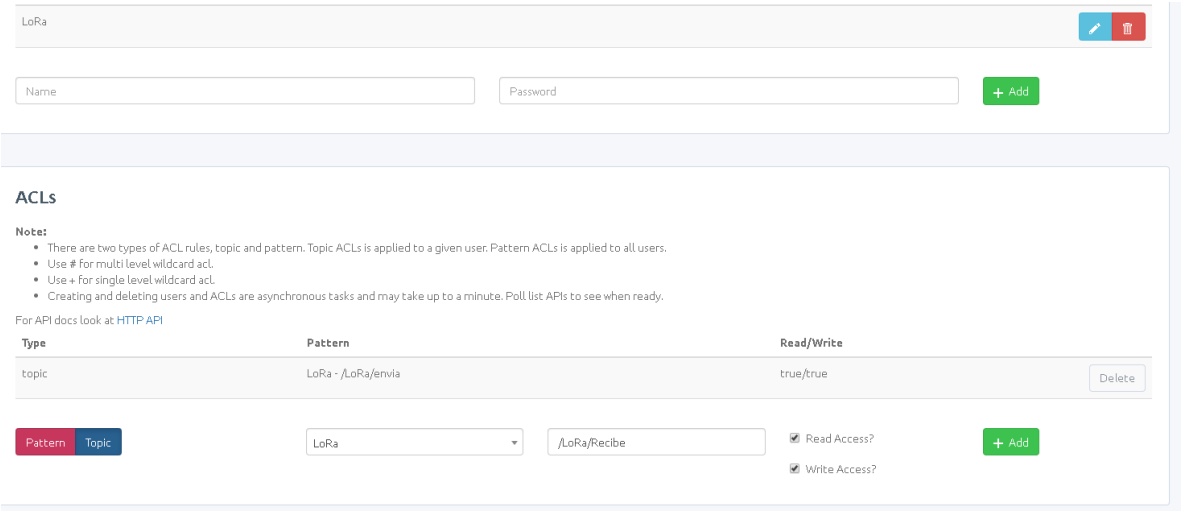
Para finalizar la configuración de la nube sobre la plataforma de CloudMQTT crearemos los Topics, que son las reglas de los datos que vamos a enviar y recibir.

Ilustración 98. Sistema de creación de tópicos



Fuente: Autor

Ilustración 99. Creación de tópicos



Fuente: Autor

Se crean dos reglas, una para los mensajes de llegada hacia la nube y otra para los datos a enviar hacia la red mesh.

En las reglas se puede usar cualquier palabra con la que podamos comprender nuestras pautas es por eso que agregamos por “/” junto con el algo referente a el proyecto y lo que se desea hacer ya sea enviar o recibir el mensaje para de esta forma identificar mejor la ruta de lo que se está ejecutando.

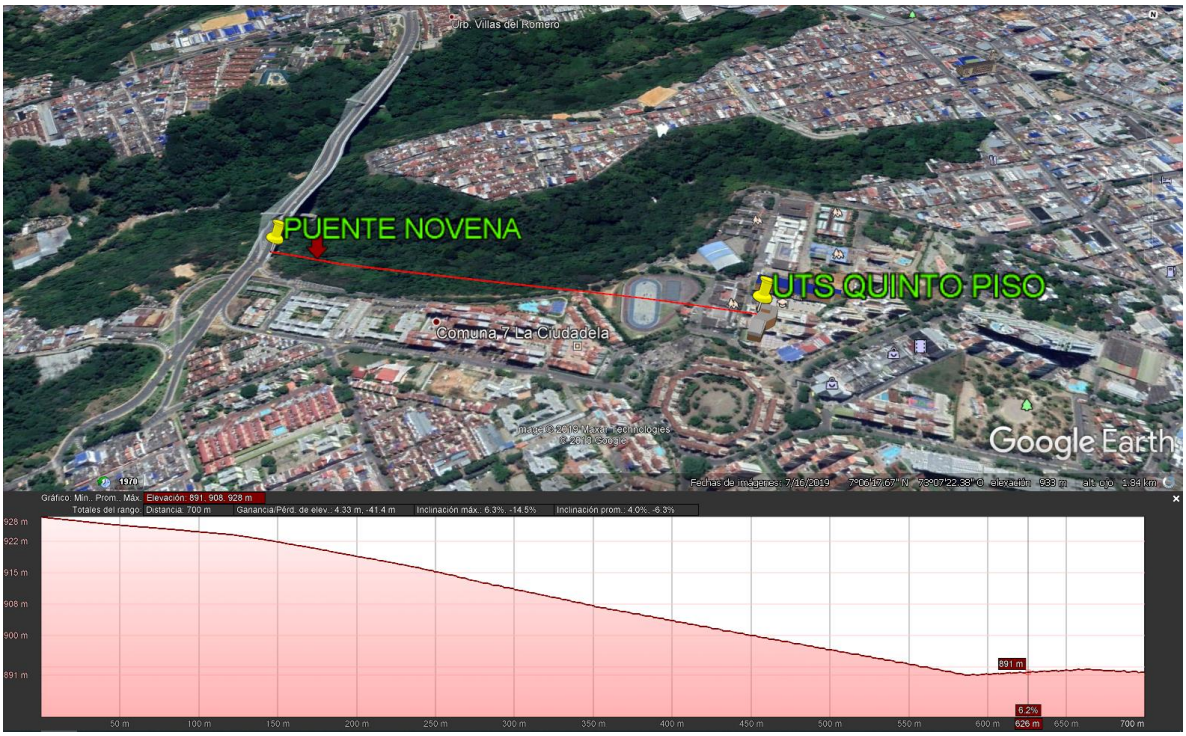
3. RESULTADOS

Se realizaron varias pruebas de cobertura entre las cuales es importante resaltar el uso de dos tipos diferentes de antenas, con sus ventajas ya mencionadas en el desarrollo del marco referencial, por el cual se aprecian resultados notables en la distancia de la comunicación alcanzada en el uso de cada una de estas.

3.1.1. PRUEBAS DE COBERTURA ANTENA HELICOIDAL

La primera prueba de cobertura se realizó entre el edificio B de las unidades tecnológicas de Santander y el puente de la Novena con una distancia de 0.7 Km con el uso de la antena Helicoidal que viene junto con el módulo de radio LoRa.

Ilustración 100. Prueba puente de la novena hasta las UTS con antena helicoidal



Fuente: Autor

Para esta distancia se obtuvieron los siguientes resultados de sensibilidad y ruido.

Tabla 2. Resultados de la calidad de la señal primera prueba de cobertura

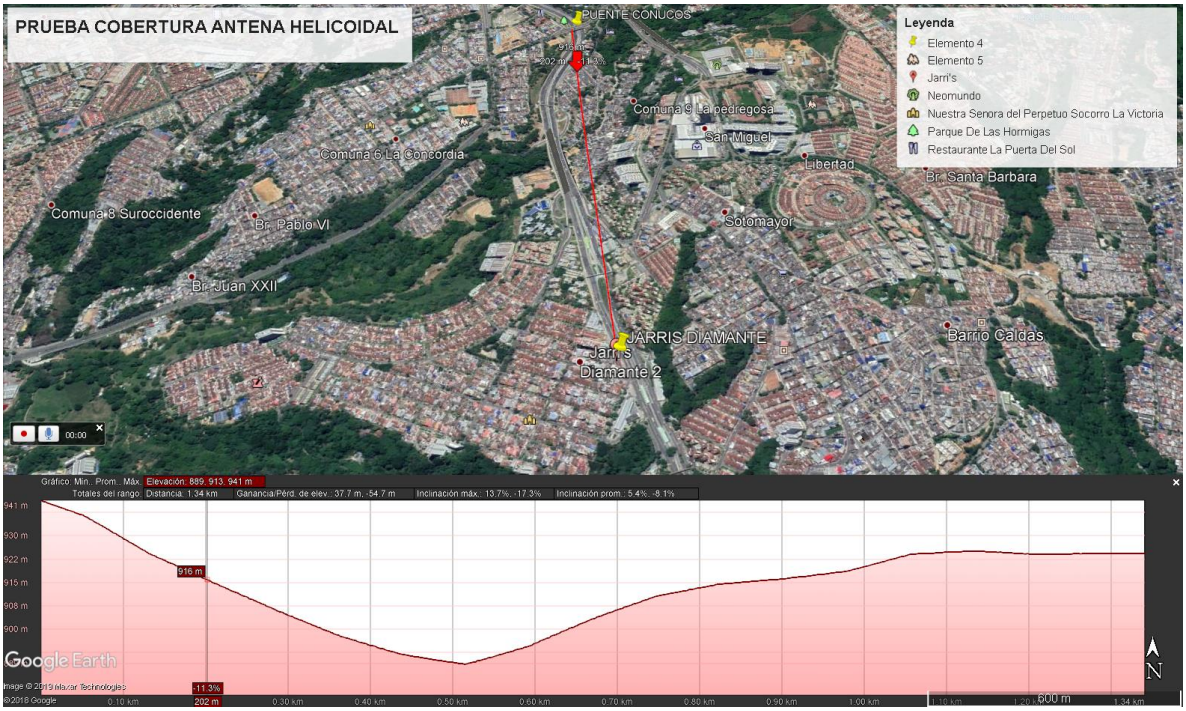
```
This message is not for me.
This message is not for me.
This message is not for me.
Received from: 0x44
Sent to: 0xll
Message ID: ll
Message length: 4
Message: hola
RSSI: -88
Snr: 9.00

Enviando: [/LoRa/envia] hola
```

PRUEBA 1	SI	-88	9
PRUEBA 2	SI	-83	-0,75
PRUEBA 3	SI	-92	8,5
PRUEBA 4	SI	-87	2,75
PRUEBA 5	SI	-93	8
PRUEBA 6	SI	-91	9,5
PRUEBA 7	SI	-83	1,75
PRUEBA 8	SI	-92	1,25
PRUEBA 9	SI	-81	8,5
PRUEBA 10	SI	-90	8,75
Promedio		-88	

Fuente: Autor

La segunda prueba con la antena Helicoidal de cobertura se realizó entre el puente Conucos ubicado sobre la puerta del sol en la ciudad de Bucaramanga, Santander y el restaurante Jarris del barrio Diamante de la misma ciudad, con el uso de la antena Helicoidal con línea de vista.

Ilustración 101. Prueba puente Conucos y el restaurante Jarris del barrio Diamante



Fuente: Autor

Para esta prueba se obtuvieron los siguientes resultados para una distancia de 1.33 Km.

Tabla 3. Resultados de la calidad de la señal segunda prueba de cobertura

```
This message is not for me.
This message is not for me.
This message is not for me.
Received from: 0x44
Sent to: 0x11
Message ID: 7
Message length: 4
Message: hola
RSSI: -89
Snr: 9.25

Enviando: [/LoRa/envia] hola
```

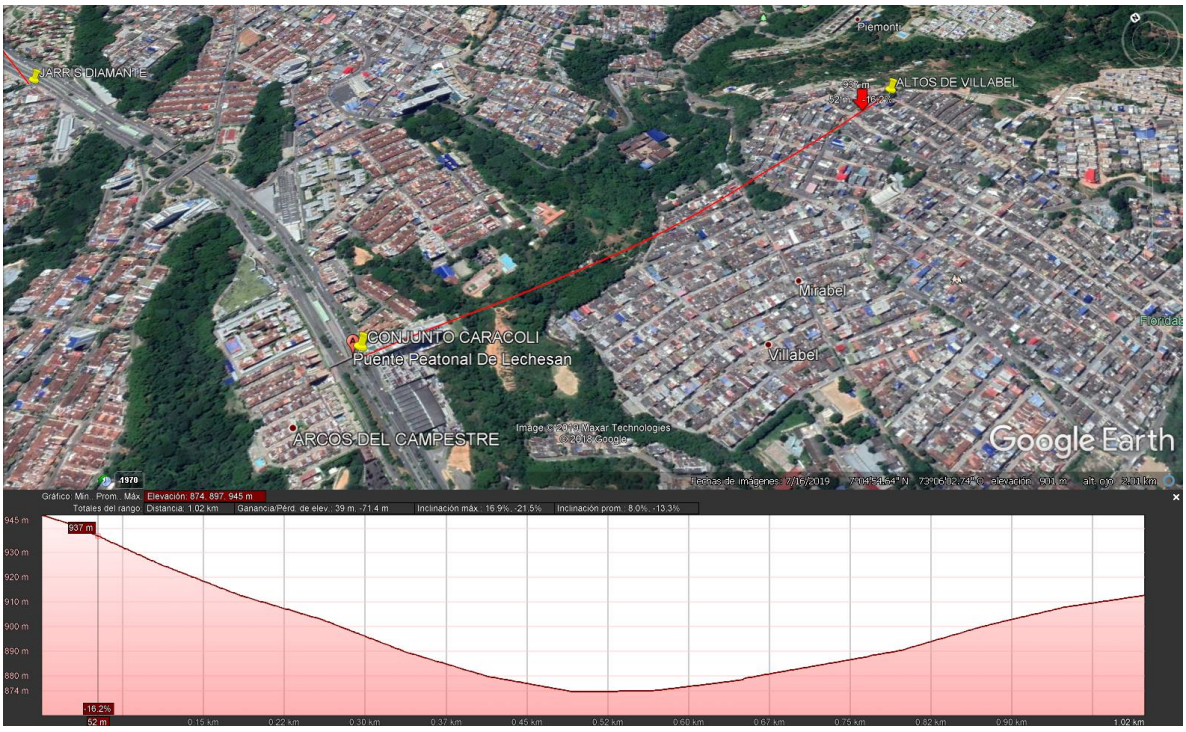
PRUEBA #	MENSAJE RECIBIDO	RSSI	SNR
PRUEBA 1	SI	-89	9,25
PRUEBA 2	SI	-91	8,75
PRUEBA 3	SI	-96	9
PRUEBA 4	SI	-97	-0,75
PRUEBA 5	SI	-93	9
PRUEBA 6	SI	-95	9
PRUEBA 7	SI	-91	9,25
PRUEBA 8	SI	-92	9
PRUEBA 9	SI	-90	9,25
PRUEBA 10	SI	-89	9,25
Promedio		-92,3	

Fuente: Autor

3.1.2. PRUEBAS DE COBERTURA ANTENA DIPOLO

La primera prueba de cobertura con la antena dipolo se realizó entre el barrio Altos de Villabel y el Conjunto que esta sobre la autopista principal llamado Caracolí en Floridablanca, con una distancia de 1.02 Km. Es importante resaltar la línea de vista entre antenas y su polarización vertical.

Ilustración 102. Prueba barrio Altos de Villabel y el conjunto Calacolí con antena dipolo



Fuente: Autor

Tabla 4. Resultados de la calidad de la señal tercera prueba de cobertura

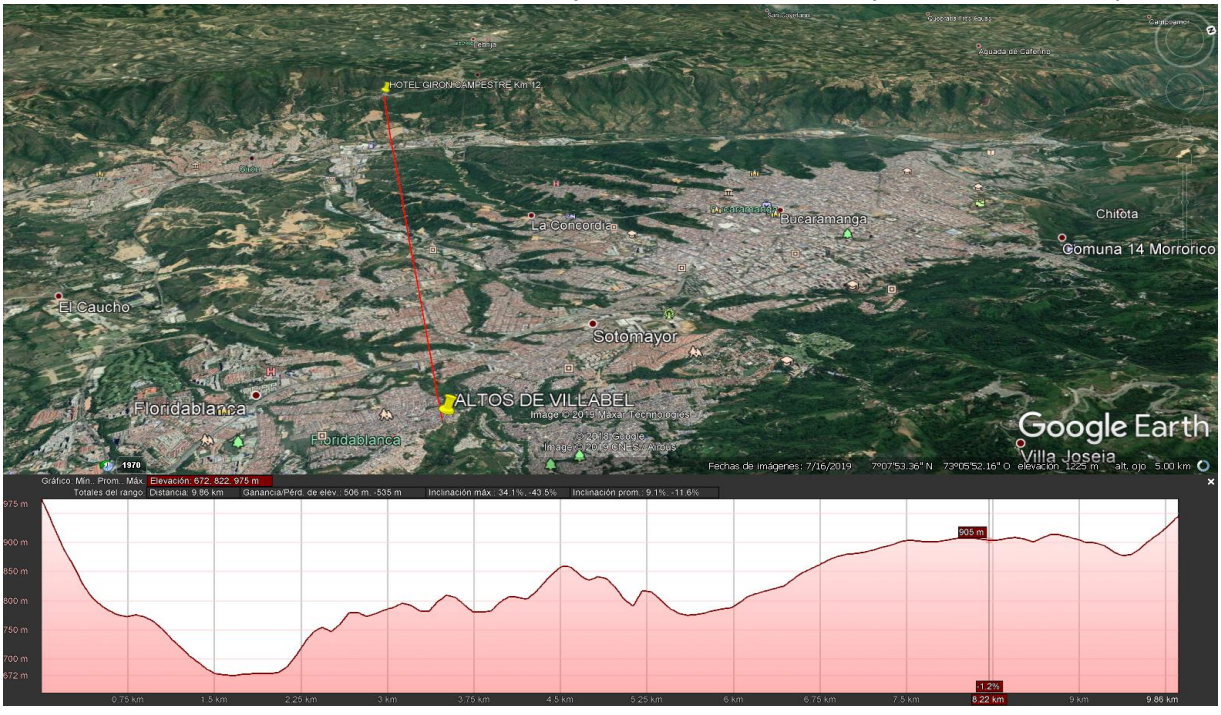
```
This message is not for me.
This message is not for me.
This message is not for me.
Received from: 0x44
Sent to: 0x11
Message ID: 43
Message length: 4
Message: hola
RSSI: -100
Snr: 9.00

Enviando: [/LoRa/envia] hola
```

PRUEBA #	MENSAJE RECIBIDO	RSSI	SNR
PRUEBA 1	SI	-100	9
PRUEBA 2	SI	-103	-9,5
PRUEBA 3	SI	-111	-1
PRUEBA 4	SI	-108	-4,25
PRUEBA 5	SI	-108	-4,25
PRUEBA 6	SI	-106	4,25
PRUEBA 7	SI	-101	-1,5
PRUEBA 8	SI	-98	-0,75
PRUEBA 9	SI	-106	4,25
PRUEBA 10	SI	-99	-0,5
Promedio		-104	

Fuente: Autor

La antena dipolo construida según los cálculos realizados se usó de igual manera para esta prueba entre el barrio alto de Villabel en Floridablanca y el kilómetro 12 de la vía que conduce al aeropuerto de Bucaramanga, donde se logró obtener una distancia de 9.76 Km con línea de vista entre antenas con una polarización vertical, la más larga hecha.

Ilustración 103. Prueba barrio Altos de Villabel y el kilómetro 12 vía aeropuerto con antena dipolo



Fuente: Autor

```
Received from: 0x33
Sent to: 0x11
Message ID: 28
Message length: 4
Message: hola
RSSI: -110
Snr: 8.25

Enviando: [/LoRa/envia] hola
```

Tabla 5. Resultados de la calidad de la señal primera prueba de cobertura

PRUEBA #	MENSAJE RECIBIDO	RSSI	SNR
PRUEBA 1	SI	-110	8,25
PRUEBA 2	SI	-110	-3,75
PRUEBA 3	SI	-105	-6,25
PRUEBA 4	SI	-105	-4,25
PRUEBA 5	SI	-99	-4,5
PRUEBA 6	SI	-101	-4,5
PRUEBA 7	SI	-108	2,5
PRUEBA 8	SI	-99	-3,5
PRUEBA 9	SI	-106	-0,5
PRUEBA 10	SI	-102	-3,75
Promedio		-104,5	

Fuente: Autor

4. CONCLUSIONES

- Es importante destacar que el resultado de este proyecto es un prototipo, pero es un prototipo muy cercano a una red real y hay pocos pasos desde este prototipo hasta una infraestructura de la de red que a su vez es compleja debido a las características de la geografía física de la zona a implementar.

- La eficiencia energética los módulos LoRa comparado con las tecnologías convencionales como las redes celulares 2G que en su mayoría están en zonas rurales con sus inmensas antenas de alta potencia y gran consumo, hacen que esta tecnología sea ideal para la transmisión de datos a largas distancias.

- El monitoreo constante de variables es posible gracias a que el ancho de banda de la red es suficiente para los paquetes enviados a través de esta, teniendo en cuenta que podría usarse en cualquier entorno de las IoT, no solo en zonas apartadas sino en las ciudades donde la interconexión de cualquier tipo de aparato electrónico para su control y automatización es de las tecnologías innovadoras que más se hablan hoy día.

- La autogestión de la red misma hace que la información y el llegar a su destino sea su principal tarea, ya que detecta cualquier nodo que este fallando o que no cuente con la potencia suficiente en su recepción y transmisión.

- La línea de vista entre los módulos es importante para lograr una efectiva comunicación entre ellos para distancias muy largas, se recomienda que estén en el mayor espacio libre posible debido a que en espacios como parqueaderos o sótanos con placas de concreto muy robustas la pérdida de comunicación es muy probable.

- Si se utilizan pausas muy largas en el NodeMCU se generan problemas en el programa debido a las limitaciones en la ejecución del watchdog.

- No es posible un análisis grafico debido a todos los factores externos que pueden afectar las mediciones del RSSI y el SNR del LoRa, debido a que son muy variantes a pesar de que se encuentren estos a distancias muy cercanas o lejanas

- Se comprobó que con la antena dipolo se obtiene una mayor cobertura en la transmisión comparado con la antena helicoidal debido a su mayor trasferencia de potencia logrando hasta 10 veces su distancia y ganancia.

5. RECOMENDACIONES

Con NodeMCU muchas cosas no están claras y las tienes que indagar o descubrir en su desarrollo. Sin embargo, la IDE de Arduino tiene algunas ayudas, pero estás no están del todo completas, siempre se recomienda cuidado de tener la documentación mínima para poder empezar en su conexión y configuración con el módulo LoRa.

En el documento entregado se han detallado claramente los pasos a seguir durante el desarrollo del proyecto junto con los resultados de las pruebas de cobertura y eficiencia de la comunicación del mismo. Es por esto que los trabajos futuros deberán seguir lo propuesto en esta red y en su implementación para su enfoque en las diferentes aplicaciones del internet de las cosas.

Es importante mantener la línea de vista y la polarización correcta entre las antenas de cada nodo para una mejor comunicación entre ellos.

Para un menor consumo energético se puede hacer uso de configuraciones de reposo en el LoRa y los microcontroladores, donde se establezca periodos programados de envío de paquetes.

6. REFERENCIAS BIBLIOGRÁFICAS

Arcenegui, J. (2016). *Bliblioteca de Ingenieria. Universidad de Sevilla*. Obtenido de http://bibing.us.es/proyectos/abreproy/12363/fichero/Red+de+sensores+para+monitorizaci%C3%B3n+inteligente+de+vivienda.pdf

Ardila, J. (11 de Junio de 2014). *Universitat Oberta de Catalunya.* Obtenido de http://openaccess.uoc.edu/webapps/o2/bitstream/10609/34921/6/jardilaTFM0614memoria.pdf

Arduino. (2019). *Arduino.cl*. Obtenido de https://arduino.cl/que-es-arduino/

ARIAS, D. L. (2014). *UNIVERSIDAD CATOLICA DE SANTIAGO DE GUAYAQUIL*. Obtenido de http://repositorio.ucsg.edu.ec/bitstream/3317/1690/1/T-UCSG-PRE-TEC-ITEL-27.pdf

BARBOSA, R. (27 de 11 de 2015). *SEACCNA*. Obtenido de https://www.seaccna.com/modelo-osi-guia-definitiva/

BARRAGAN, J. (07 de 05 de 2012). *Tutoría Virtual de A. Javier Barragán Piña* . Obtenido de https://uhu.es/antonio.barragan/content/5topologias

Bautista, D., Sanchez , L., & Portillo , Y. (2014). *RESEARCHGATE*. Obtenido de https://www.researchgate.net/publication/315475585_Redes_mesh_una_alternativa_a_problemas_de_cobertura_de_red_una_revision_de_literatura

Blogger.com. (19 de 03 de 2016). *Blogger.com* . Obtenido de http://topologias1redes.blogspot.com/2016/03/topologia.html

cE. (s.f.).

Chacon, J. M., & Campos Ramirez, Y. (28 de 06 de 2018). *UNIVERSIDAD DISTRITAL FRANCISCO JOSE DE CALDAS*. Obtenido de http://repository.udistrital.edu.co/bitstream/11349/13877/1/CamposRamirezYeissonAlejandro2018.pdf

Cilfone, A., Davoli, L., Belli, L., & Ferrari, G. (17 de abril de 2019). *MDPI*. Obtenido de https://www.mdpi.com/1999-5903/11/4/99/pdf

CRC. (14 de Marzo de 2019). *CRC.com*. Obtenido de https://www.crcom.gov.co/es/noticia/modernizar-las-redes-m-viles-del-pa-s-la-nueva-apuesta-de-la-crc

Cruz Rozo, V. J., & Collazos Collazos, J. (2016). *UNIVERSIDAD LIBRE DE COLOMBIA*. Obtenido de https://repository.unilibre.edu.co/bitstream/handle/10901/10815/Tesis-Final.pdf?sequence=1&isAllowed=y

Del Valle, L. H. (2019). *programarfacil.com*. Obtenido de https://programarfacil.com/esp8266/mqtt-esp8266-raspberry-pi/

EBI, C., SCHALTEGGER, F., RÜST, A., & BLUMENSAAT, F. (19 de MARZO de 2019). *IEEE XPLORE*. Obtenido de https://ieeexplore.ieee.org/iel7/6287639/8600701/08703036.pdf

electronics, M. (s.f.). *Arduino*. Obtenido de http://arduino.cl/que-es-arduino/

EPN, B. D. (2019). *ESCUELA POLITECNICA NACIONAL*. Obtenido de https://bibdigital.epn.edu.ec/bitstream/15000/83/1/CD-0055.pdf

FEDEGAN. (2019). *Federación Colombiana de Ganaderos.* Obtenido de https://estadisticas.fedegan.org.co/DOC/download.jsp?pRealName=Cifras_Referencia_2019.pdf&iIdFiles=674

Hernandez, L. d. (2019). *PROGRAMARFACIL.COM*. Obtenido de https://programarfacil.com/podcast/nodemcu-tutorial-paso-a-paso/

Horovitz, L., & Mayobre, R. (2018). *UNIVERSIDAD URT URUGUAY*. Obtenido de https://dspace.ort.edu.uy/bitstream/handle/20.500.11968/3878/Material%20completo.pdf?sequence=-1&isAllowed=y

Limón de la Rosa, S. (2017). *BIBLIOTECA DE INGENIERIA, UNIVERSIDAD DE SEVILLLA.* Obtenido de http://bibing.us.es/proyectos/abreproy/91508/fichero/MemoriaTFG.pdf

Martinez, E. (21 de 06 de 2007). *Eveliux*. Obtenido de https://www.eveliux.com/mx/curso/topolog.html

Martinez, J. A. (2014). *UNIVERSIDAD ABIERTA DE CATALUÑA*. Obtenido de http://openaccess.uoc.edu/webapps/o2/bitstream/10609/34921/6/jardilaTFM0614memoria.pdf

MinTIC. (16 de 08 de 2019). *Ministerio de Tecnologías de la Información y las Comunicaciones*. Obtenido de https://www.mintic.gov.co/portal/604/w3-article-102735.html

MORENO SANTOS, D. A. (15 de 09 de 2015). *UNIVERSIDAD LIBRE DE COLOMBIA*. Obtenido de https://repository.unilibre.edu.co/bitstream/handle/10901/8921/Proyecto%20Grado%20David%20Moreno.pdf?sequence=1

Palomino, V. R. (2014). *UNIVERSIDAD NACIONAL DEL CALLAO*. Obtenido de http://repositorio.unac.edu.pe/bitstream/handle/UNAC/332/VictorAlfonso_Tesis_tituloprofesional_2014.pdf?sequence=3&isAllowed=y

Redessobreinformatica. (04 de 2012). *Redessobreinformatica*. Obtenido de https://redessobreinformatica.wordpress.com/topologias/topologia-de-malla-mesh

Rincon, D. A. (21 de 07 de 2011). Obtenido de https://es.slideshare.net/davandrinlop/antenas-helicoidales-8654948

Ruiz Parra , D. M., & Blanco Garrido , F. (12 de 11 de 2014). *ORGANIZACION DE ESTADOS IBEROAMERICANOS.* Obtenido de OEI: https://www.oei.es/historico/congreso2014/memoriactei/941.pdf

Sanmiguel, O., & Garcia Prada, J. E. (MAYO de 2013). *APUNTES DE INVESTIGACION UPB*. Obtenido de http://apuntesdeinvestigacion.bucaramanga.upb.edu.co/wp-content/uploads/2016/03/1.SIRESI-Garcia-Maestre-Redes-Mesh.pdf

SEMTECH. (2019). *SEMTECH*. Obtenido de https://www.semtech.com/lora/lora-applications/smart-agriculture

SIGMA ELECTRONICA. (2019). *Sigma Electrónica*. Obtenido de https://www.sigmaelectronica.net/producto/ra-01/

TEST AMERICA. (2019). *TES AMERICA*. Obtenido de https://tesamerica.com/tipos-antenas-funcionamiento/

7. ANEXOS

PROGRAMA PLACA ATMEGA328P.

```
#include <SPI.h>              // include libraries
#include <LoRa.h>



String outgoing;              // outgoing message

byte msgCount = 0;            // count of outgoing messages
byte localAddress = 0x44;     // address of this device
byte destination = 0x11;      // destination to send to
byte destination2 = 0x66;
int envio = 0;
int A = 0;
int B = 0;
int C = 0;
int D = 0;
String mensaje;

void setup() {
  Serial.begin(9600);                 // initialize serial
  while (!Serial);

  Serial.println("LoRa Duplex");

  // override the default CS, reset, and IRQ pins (optional)


  if (!LoRa.begin(433E6)) {           // initialize ratio at 915 MHz
    Serial.println("LoRa init failed. Check your connections.");
    while (true);                     // if failed, do nothing
  }
  Serial.println("LoRa init succeeded.");
}

void loop() {
onReceive(LoRa.parsePacket());
```

```
onReceive1(LoRa.parsePacket());
if(envio==0x66){
  sendMessage(mensaje);
  Serial.println("enviando " + mensaje);
  envio=0;
  }
  else if(envio==0x11){
  sendMessage2(mensaje);
  Serial.println("enviando2 " + mensaje);
  envio=0;
   }
}

void sendMessage(String outgoing) {
  LoRa.beginPacket();                 // start packet
  LoRa.write(destination);            // add destination address
  LoRa.write(localAddress);             // add sender address
  LoRa.write(msgCount);                 // add message ID
  LoRa.write(outgoing.length());        // add payload length
  LoRa.print(outgoing);                 // add payload
  LoRa.endPacket();                     // finish packet and send it
  msgCount++;                           // increment message ID
}
void sendMessage2(String outgoing) {
  LoRa.beginPacket();                 // start packet
  LoRa.write(destination2);             // add destination address
  LoRa.write(localAddress);             // add sender address
  LoRa.write(msgCount);                 // add message ID
  LoRa.write(outgoing.length());        // add payload length
  LoRa.print(outgoing);                 // add payload
  LoRa.endPacket();                     // finish packet and send it
  msgCount++;                           // increment message ID
}

void onReceive(int packetSize) {
  if (packetSize == 0) return;          // if there's no packet, return

  // read packet header bytes:
  int recipient = LoRa.read();          // recipient address
  byte sender = LoRa.read();            // sender address
```

```
byte incomingMsgId = LoRa.read();     // incoming msg ID
byte incomingLength = LoRa.read();    // incoming msg length

String incoming = "";


while (LoRa.available()) {
  incoming += (char)LoRa.read();
}

if (incomingLength != incoming.length()) {   // check length for error
  Serial.println("error: message length does not match length");
  return;                             // skip rest of function
}

// if the recipient isn't this device or broadcast,
if (recipient != localAddress && recipient != 0x66) {
  Serial.println("This message is not for me.");
  return;                             // skip rest of function
}

// if message is for this device, or broadcast, print details:
Serial.println("Received from: 0x" + String(sender, HEX));
Serial.println("Sent to: 0x" + String(recipient, HEX));
Serial.println("Message ID: " + String(incomingMsgId));
Serial.println("Message length: " + String(incomingLength));
Serial.println("Message: " + incoming);
Serial.println("RSSI: " + String(LoRa.packetRssi()));
Serial.println("Snr: " + String(LoRa.packetSnr()));
Serial.println();
A=LoRa.packetRssi();
 mensaje=incoming;
 envio=sender;
  if(sender==0x33){
  return;}
  if(sender==0x44){
  return;}
     if(sender==0x55){
  return;}
}
```

```
void onReceive1(int packetSize) {
  if (packetSize == 0) return;          // if there's no packet, return

  // read packet header bytes:
  int recipient = LoRa.read();          // recipient address
  byte sender = LoRa.read();            // sender address
  byte incomingMsgId = LoRa.read();     // incoming msg ID
  byte incomingLength = LoRa.read();    // incoming msg length

  String incoming = "";


  while (LoRa.available()) {
    incoming += (char)LoRa.read();
  }

  if (incomingLength != incoming.length()) {   // check length for error
    Serial.println("error: message length does not match length");
    return;                             // skip rest of function
  }

  // if the recipient isn't this device or broadcast,
  if (recipient != localAddress && recipient != 0x11) {
    Serial.println("This message is not for me.");
    return;                             // skip rest of function
  }

  // if message is for this device, or broadcast, print details:
  Serial.println("Received from: 0x" + String(sender, HEX));
  Serial.println("Sent to: 0x" + String(recipient, HEX));
  Serial.println("Message ID: " + String(incomingMsgId));
  Serial.println("Message length: " + String(incomingLength));
  Serial.println("Message: " + incoming);
  Serial.println("RSSI: " + String(LoRa.packetRssi()));
  Serial.println("Snr: " + String(LoRa.packetSnr()));
  Serial.println();
  B=LoRa.packetRssi();
  mensaje=incoming;
  envio=sender;
    if(sender==0x33){
```

```
   return;}
   if(sender==0x44){
   return;}
    if(sender==0x55){
   return;}
}
```

PROGRAMA NODEMCU CON PULSADOR

```
#include <SPI.h>
#include <LoRa.h>

const int csPin = 15;
const int irqPin = 16;
const int buttonPin = 5;
int buttonState = 0;


byte localAddress = 0x66;
byte destination = 0x22;
byte destination2 = 0x33;
byte destination3 = 0x44;
String outgoing;
byte msgCount = 0;
int A=0;
int B=0;
int C=0;
int D=0;
int E=0;
int F=0;
int G=0;
int H=0;
int I=0;
int cont = 0;
int cont1 = 0;
int contenvio = 0;
int rebote = 0;
char valor;
String llegada;
```

```
void setup() {
  pinMode(buttonPin, INPUT);
  Serial.begin(9600);
while (!Serial);

  Serial.println("LoRa Duplex");

  LoRa.setPins(csPin,-1, irqPin);

 while(!LoRa.begin(433E6)) {
    Serial.println("LoRa init failed. Check your connections.");
  delay(1000);
  }
  Serial.println("LoRa init succeeded.");
}
void loop() {
buttonState = digitalRead(buttonPin);
  if (buttonState==LOW){
  String message = ".";
  Serial.println("enviando " + message);
  sendMessage(message);
  onReceive(LoRa.parsePacket());
  delay(500);
  sendMessage2(message);
  onReceive2(LoRa.parsePacket());
  delay(500);
  sendMessage3(message);
  onReceive3(LoRa.parsePacket());
  delay(500);
  contenvio=1;
  if ((A<B) && (A<C) && (contenvio==1)){
  String message2 = "hola";
  sendMessage(message2);
  Serial.println(message2);
  contenvio=0;
  A=0;
  B=0;
  C=0;
  D=0;
  E=0;
```

```
F=0;
G=0;
H=0;
I=0;
rebote=1;
}
if ((B<A) && (B<C) &&(contenvio==1)){
String message2 = "hola";
sendMessage2(message2);
contenvio=0;
A=0;
B=0;
C=0;
D=0;
E=0;
F=0;
G=0;
H=0;
I=0;
rebote=1;
}
  if ((C<A) && (C<B) && (contenvio==1)){
String message2 = "hola";
sendMessage3(message2);
contenvio=0;
A=0;
B=0;
C=0;
D=0;
E=0;
F=0;
G=0;
H=0;
I=0;
rebote=1;
}
}
if((contenvio==0) && (rebote==1)){
  delay(1000);
  rebote=0;}
```

```
 if((contenvio==0) && (rebote==0)){
  onReceive(LoRa.parsePacket());
  onReceive2(LoRa.parsePacket());
  onReceive3(LoRa.parsePacket());
  A=0;
  B=0;
  C=0;
  D=0;
  E=0;
  F=0;
  G=0;
  H=0;
  I=0;
    }
  }
 void sendMessage(String outgoing) {
  LoRa.beginPacket();
  LoRa.write(destination);
  LoRa.write(localAddress);
  LoRa.write(msgCount);
  LoRa.write(outgoing.length());
  LoRa.print(outgoing);
  LoRa.endPacket();
  msgCount++;
 }
 void sendMessage2(String outgoing) {
  LoRa.beginPacket();
  LoRa.write(destination2);
  LoRa.write(localAddress);
  LoRa.write(msgCount);
  LoRa.write(outgoing.length());
  LoRa.print(outgoing);
  LoRa.endPacket();
  msgCount++;
 }
 void sendMessage3(String outgoing) {
  LoRa.beginPacket();
  LoRa.write(destination3);
  LoRa.write(localAddress);
  LoRa.write(msgCount);
```

```
  LoRa.write(outgoing.length());
  LoRa.print(outgoing);
  LoRa.endPacket();
  msgCount++;
}
void onReceive(int packetSize) {
  if (packetSize == 0) return;
  // read packet header bytes:
  int recipient = LoRa.read();
  byte sender = LoRa.read();
  byte incomingMsgId = LoRa.read();
  byte incomingLength = LoRa.read();

  String incoming = "";
  while (LoRa.available()) {
    incoming += (char)LoRa.read();
  }

  if (incomingLength != incoming.length()) {
    Serial.println("error: message length does not match length    ");
    return;
  }

  // if the recipient isn't this device or broadcast,
  if (recipient != localAddress && recipient != 0x22) {
    Serial.println("This message is not for me.");
    return;
  }

  Serial.println("Received from: 0x" + String(sender, HEX));
  Serial.println("Sent to: 0x" + String(recipient, HEX));
  Serial.println("Message ID: " + String(incomingMsgId));
  Serial.println("Message length: " + String(incomingLength));
  Serial.println("Message: " + incoming);
  Serial.println("RSSI: " + String(LoRa.packetRssi()));
  Serial.println("Snr: " + String(LoRa.packetSnr()));
  Serial.println();
  D=LoRa.packetRssi();
  E=LoRa.packetSnr();
  A=D+E;
```

```
  llegada=incoming;
  if(llegada[0]=='.'){
    return;
    }
    else if(sender==0x11){
    return;}
    else{
      Serial.println("mensaje: " + incoming );
      return;
      }
}
void onReceive2(int packetSize) {
  if (packetSize == 0) return;

  // read packet header bytes:
  int recipient = LoRa.read();
  byte sender = LoRa.read();
  byte incomingMsgId = LoRa.read();
  byte incomingLength = LoRa.read();

  String incoming = "";

  while (LoRa.available()) {
    incoming += (char)LoRa.read();
  }

  if (incomingLength != incoming.length()) {
    Serial.println("error: message length does not match length    ");
    return;
  }


  if (recipient != localAddress && recipient != 0x33) {
    Serial.println("This message is not for me.");
    return;
  }

  Serial.println("Received from: 0x" + String(sender, HEX));
  Serial.println("Sent to: 0x" + String(recipient, HEX));
  Serial.println("Message ID: " + String(incomingMsgId));
```

```
  Serial.println("Message length: " + String(incomingLength));
  Serial.println("Message: " + incoming);
  Serial.println("RSSI: " + String(LoRa.packetRssi()));
  Serial.println("Snr: " + String(LoRa.packetSnr()));
  Serial.println();
  F=LoRa.packetRssi();
  G=LoRa.packetSnr();
  B=F+G;
    llegada=incoming;
  if(llegada[0]=='.'){
    return;
    }
    else if(sender==0x11){
    return;}
    else {
      Serial.println("mensaje: " + incoming );
      return;
      }
}
 void onReceive3(int packetSize) {
  if (packetSize == 0) return;

  // read packet header bytes:
  int recipient = LoRa.read();
  byte sender = LoRa.read();
  byte incomingMsgId = LoRa.read();
  byte incomingLength = LoRa.read();

  String incoming = "";

  while (LoRa.available()) {
    incoming += (char)LoRa.read();
  }

  if (incomingLength != incoming.length()) {
    Serial.println("error: message length does not match length    ");
    return;
  }

  if (recipient != localAddress && recipient != 0x44) {
```

```
  Serial.println("This message is not for me.");
  return;
 }

 Serial.println("Received from: 0x" + String(sender, HEX));
 Serial.println("Sent to: 0x" + String(recipient, HEX));
 Serial.println("Message ID: " + String(incomingMsgId));
 Serial.println("Message length: " + String(incomingLength));
 Serial.println("Message: " + incoming);
 Serial.println("RSSI: " + String(LoRa.packetRssi()));
 Serial.println("Snr: " + String(LoRa.packetSnr()));
 Serial.println();
 H=LoRa.packetRssi();
 I=LoRa.packetSnr();
 C=H+I;
  llegada=incoming;
 if(llegada[0]=='.'){
  return;
  }
  else if(sender==0x11){
  return;}
  else{
    Serial.println("mensaje: " + incoming );
    return;
    }
}
```

PROGRAMA NODEMCU PARA SUBIR A LA NUBE

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <SPI.h>
#include <LoRa.h>


//-------------------VARIABLES GLOBALES--------------------------

const char *ssid = "David RC";
const char *password = "c0l0mbi@87";
char   SERVER[50]   = "soldier.cloudmqtt.com";
```

```
int    SERVERPORT   =   18569;
String USERNAME = "LoRa";
char   PASSWORD[50] = "12345678";

const int csPin = 15;
const int irqPin = 16;
byte localAddress = 0x11;
byte destination = 0x22;
byte destination2 = 0x33;
byte destination3 =  0x44;
String outgoing;
byte msgCount = 0;
int contconexion = 0;
int A=0;
int B=0;
int C=0;
int D=0;
int E=0;
int F=0;
int G=0;
int H=0;
int I=0;
int cont = 0;
int cont1 = 0;
int contenvio = 0;
int contllegada = 0;
int rebote = 0;
int conthola = 0;
String mensaje;
int contmensaje=0;
char LORA[50];
char valor[15];
char MENSAJE[50];
char SALIDAMENSAJE[50];


WiFiClient espClient;
PubSubClient client(espClient);

void callback(char* Topic, byte* payload, unsigned int length) {
```

```
  char PAYLOAD[5] = "    ";

  Serial.print("Mensaje Recibido: [");
  Serial.print(Topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++) {
    PAYLOAD[i] = (char)payload[i];
  }
  Serial.println(PAYLOAD);

  if (String(Topic) ==  String(SALIDAMENSAJE)) {
    if (payload[1] == 'N'){
      contmensaje=1;
    }
  }
}

//------------------------RECONNECT-----------------------------
void reconnect() {
  uint8_t retries = 3;
  while (!client.connected()) {
    Serial.print("Intentando conexion MQTT...");
    String clientId = "ESP8266Client-";
    clientId += String(random(0xffff), HEX);
    USERNAME.toCharArray(LORA, 50);
    if (client.connect("", LORA, PASSWORD)) {
      Serial.println("conectado");
      client.subscribe(SALIDAMENSAJE);
    } else {
      Serial.print("fallo, rc=");
      Serial.print(client.state());
      Serial.println(" intenta nuevamente en 5 segundos");
      delay(5000);
    }
    retries--;
    if (retries == 0) {
      while(1);
    }
  }
```

```
}

//------------------------SETUP----------------------------
void setup() {
  Serial.begin(9600);
  Serial.println("");
  LoRa.setFrequency(433E6);
  while (!Serial);
  Serial.println("LoRa Duplex");


  LoRa.setPins(csPin,-1, irqPin);

 while(!LoRa.begin(433E6)) {
   Serial.println("LoRa init failed. Check your connections.");
  delay(1000);
  }
  Serial.println("LoRa init succeeded.");

  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED and contconexion <50) {
    ++contconexion;
    delay(500);
    Serial.print(".");
  }
  if (contconexion <50) {
      IPAddress ip(192,168,1,156);
      IPAddress gateway(192,168,1,1);
      IPAddress subnet(255,255,255,0);
      WiFi.config(ip, gateway, subnet);

      Serial.println("");
      Serial.println("WiFi conectado");
      Serial.println(WiFi.localIP());
  }
  else {
      Serial.println("");
      Serial.println("Error de conexion");
  }
```

```
  client.setServer(SERVER, SERVERPORT);
  client.setCallback(callback);

  String envia = "/" + USERNAME + "/" + "envia";
  envia.toCharArray(MENSAJE, 50);
  String recibe = "/" + USERNAME + "/" + "recibe";
  recibe.toCharArray(SALIDAMENSAJE, 50);

}

//-------------------------LOOP--------------------------------
void loop() {

  if (!client.connected()) {
    reconnect();
  }
  client.loop();

    if(contmensaje==1){
String message = ".";                    //deja un mensaje en blanco
  sendMessage(message);                  //envia mensaje en blanco
  Serial.println("enviando " + message);      //imprime el mensaje enviado
  onReceive(LoRa.parsePacket());             //recibe el mensaje
  delay(500);                            //un segundo para terminar el proceso
  sendMessage2(message);                 //envia el mismo mensaje a la segunda opción
  onReceive2(LoRa.parsePacket());            //recibe el mensaje de la segunda opción
  delay(500);
  sendMessage3(message);
  onReceive3(LoRa.parsePacket());
  delay(500);
  contenvio=1;
  contmensaje=0;
  if ((A<B) && (A<C) && (contenvio==1)){                          //compara si A es mayor a B
  String message2 = "hola";                //el mensaje a enviar
  sendMessage(message2);
  contenvio=0;
  A=0;
  B=0;
  C=0;
  D=0;
```

```
E=0;
F=0;
G=0;
H=0;
I=0;
rebote=1;
}
if ((B<A) && (B<C) &&(contenvio==1)){
String message2 = "hola";
sendMessage2(message2);
contenvio=0;
A=0;
B=0;
C=0;
D=0;
E=0;
F=0;
G=0;
H=0;
I=0;
rebote=1;
}
if ((C<A) && (C<B) && (contenvio==1)){
String message2 = "hola";
  sendMessage3(message2);
Serial.println("aqui3");
contenvio=0;
A=0;
B=0;
C=0;
D=0;
E=0;
F=0;
G=0;
H=0;
I=0;
rebote=1;
}
  }
if((contenvio==0) && (rebote==1)){
```

```
    delay(1000);
    rebote=0;}
  if(contenvio==0){
   contllegada=1;
   onReceive(LoRa.parsePacket());
   onReceive2(LoRa.parsePacket());
   onReceive3(LoRa.parsePacket());
   A=0;
   B=0;
   C=0;
   D=0;
   E=0;
   F=0;
   G=0;
   H=0;
   I=0;
   }
   }
 void sendMessage(String outgoing) {
   LoRa.beginPacket();
   LoRa.write(destination);
   LoRa.write(localAddress);
   LoRa.write(msgCount);
   LoRa.write(outgoing.length());
   LoRa.print(outgoing);
   LoRa.endPacket();
   msgCount++;
 }
 void sendMessage2(String outgoing) {
   LoRa.beginPacket();
   LoRa.write(destination2);
   LoRa.write(localAddress);
   LoRa.write(msgCount);
   LoRa.write(outgoing.length());
   LoRa.print(outgoing);
   LoRa.endPacket();
   msgCount++;
 }
 void sendMessage3(String outgoing) {
   LoRa.beginPacket();
```

```
  LoRa.write(destination3);
  LoRa.write(localAddress);
  LoRa.write(msgCount);
  LoRa.write(outgoing.length());
  LoRa.print(outgoing);
  LoRa.endPacket();
  msgCount++;
}
void onReceive(int packetSize) {
  if (packetSize == 0) return;
  int recipient = LoRa.read();
  byte sender = LoRa.read();
  byte incomingMsgId = LoRa.read();
  byte incomingLength = LoRa.read();

  String incoming = "";
  while (LoRa.available()) {
    incoming += (char)LoRa.read();
  }

  if (incomingLength != incoming.length()) {
    Serial.println("error: message length does not match length    ");
    return;
  }
  if (recipient != localAddress && recipient != 0x22) {
    Serial.println("This message is not for me.");
    return;
  }

  Serial.println("Received from: 0x" + String(sender, HEX));
  Serial.println("Sent to: 0x" + String(recipient, HEX));
  Serial.println("Message ID: " + String(incomingMsgId));
  Serial.println("Message length: " + String(incomingLength));
  Serial.println("Message: " + incoming);
  Serial.println("RSSI: " + String(LoRa.packetRssi()));
  Serial.println("Snr: " + String(LoRa.packetSnr()));
  Serial.println();
  D=LoRa.packetRssi();                          // guarda el RSSI de la primera opción
  E=LoRa.packetSnr();
  A=D+E;
```

```
 mensaje =incoming;
  if (contllegada==1) {
  mensaje.toCharArray(valor, 15);
  if(valor[0] == '.'){
    contllegada=0;
  return;
  }
  else if(sender==0x66){
    return;}
  else{
       Serial.println("Enviando: [" +  String(MENSAJE) + "] " + mensaje);
    client.publish(MENSAJE, valor);
    contllegada=0;
    return;
    }
  }
}
void onReceive2(int packetSize) {
  if (packetSize == 0) return;
  int recipient = LoRa.read();
  byte sender = LoRa.read();
  byte incomingMsgId = LoRa.read();
  byte incomingLength = LoRa.read();

  String incoming = "";

  while (LoRa.available()) {
    incoming += (char)LoRa.read();
  }

  if (incomingLength != incoming.length()) {
    Serial.println("error: message length does not match length    ");
    return;
  }
  if (recipient != localAddress && recipient != 0x33) {
    Serial.println("This message is not for me.");
    return;
  }

  // if message is for this device, or broadcast, print details:
```

```
  Serial.println("Received from: 0x" + String(sender, HEX));
  Serial.println("Sent to: 0x" + String(recipient, HEX));
  Serial.println("Message ID: " + String(incomingMsgId));
  Serial.println("Message length: " + String(incomingLength));
  Serial.println("Message: " + incoming);
  Serial.println("RSSI: " + String(LoRa.packetRssi()));
  Serial.println("Snr: " + String(LoRa.packetSnr()));
  Serial.println();
  F=LoRa.packetRssi();
  G=LoRa.packetSnr();
  B=F+G;
     mensaje =incoming;
  if (contllegada==1) {
  mensaje.toCharArray(valor, 15);
  if(valor[0] == '.'){
    contllegada=0;
  return;
  }
    else if(sender==0x66){
    return;}
  else{
        Serial.println("Enviando: [" +  String(MENSAJE) + "] " + mensaje);
    client.publish(MENSAJE, valor);
    contllegada=0;
    return;
    }
  }
}
 void onReceive3(int packetSize) {
  if (packetSize == 0) return;
  int recipient = LoRa.read();
  byte sender = LoRa.read();
  byte incomingMsgId = LoRa.read();
  byte incomingLength = LoRa.read();

  String incoming = "";

  while (LoRa.available()) {
    incoming += (char)LoRa.read();
  }
```

```
  if (incomingLength != incoming.length()) {
    Serial.println("error: message length does not match length    ");
    return;
  }
  if (recipient != localAddress && recipient != 0x44) {
    Serial.println("This message is not for me.");
    return;
  }

  // if message is for this device, or broadcast, print details:
  Serial.println("Received from: 0x" + String(sender, HEX));
  Serial.println("Sent to: 0x" + String(recipient, HEX));
  Serial.println("Message ID: " + String(incomingMsgId));
  Serial.println("Message length: " + String(incomingLength));
  Serial.println("Message: " + incoming);
  Serial.println("RSSI: " + String(LoRa.packetRssi()));
  Serial.println("Snr: " + String(LoRa.packetSnr()));
  Serial.println();
  H=LoRa.packetRssi();
  I=LoRa.packetSnr();
  C=H+I;
     mensaje =incoming;
  if (contllegada==1) {
    mensaje.toCharArray(valor, 15);
  if(valor[0] == '.'){
    contllegada=0;
  return;
  }
    else if(sender==0x66){
    return;}
  else{
        Serial.println("Enviando: [" +  String(MENSAJE) + "] " + mensaje);
    client.publish(MENSAJE, valor);
    contllegada=0;
    return;
    }
  }
}
```

MENSAJES DE LLEGADA AL ATMEGA328P

```
enviando .
This message is not for me.
Received from: 0x11
Sent to: 0x22
Message ID: 4
Message length: 4
Message: hola
RSSI: -111
Snr: -5.50

enviando hola
```

```
enviando .
This message is not for me.
Received from: 0x66
Sent to: 0x22
Message ID: 76
Message length: 4
Message: hola
RSSI: -104
Snr: 7.25

enviando hola
```

```
enviando .
This message is not for me.
Received from: 0x11
Sent to: 0x22
Message ID: 10
Message length: 4
Message: hola
RSSI: -111
Snr: -1.75

enviando hola
```

```
enviando .
This message is not for me.
Received from: 0x66
Sent to: 0x22
Message ID: 82
Message length: 4
Message: hola
RSSI: -102
Snr: 7.50

enviando hola
```

```
enviando .
This message is not for me.
Received from: 0x11
Sent to: 0x22
Message ID: 41
Message length: 4
Message: hola
RSSI: -112
Snr: -3.00

enviando hola
```

```
enviando .
This message is not for me.
Received from: 0x66
Sent to: 0x22
Message ID: 94
Message length: 4
Message: hola
RSSI: -100
Snr: 8.00

enviando hola
```

```
enviando .
This message is not for me.
Received from: 0x11
Sent to: 0x22
Message ID: 191
Message length: 4
Message: hola
RSSI: -112
Snr: -4.50

enviando hola
```

```
enviando .
This message is not for me.
Received from: 0x66
Sent to: 0x22
Message ID: 10
Message length: 4
Message: hola
RSSI: -102
Snr: 7.75

enviando hola
```

```
enviando .
This message is not for me.
Received from: 0x11
Sent to: 0x22
Message ID: 197
Message length: 4
Message: hola
RSSI: -110
Snr: -6.00

enviando hola
```

MENSAJES DE LLEGADA AL NODEMCU CON PULSADOR

```
This message is not for me.
Received from: 0x33
Sent to: 0x66
Message ID: 85
Message length: 4
Message: hola
RSSI: -103
Snr: 9.00

mensaje: hola
```

```
This message is not for me.
Received from: 0x33
Sent to: 0x66
Message ID: 175
Message length: 4
Message: hola
RSSI: -102
Snr: 8.75

mensaje: hola
```

```
This message is not for me.
Received from: 0x33
Sent to: 0x66
Message ID: 135
Message length: 4
Message: hola
RSSI: -103
Snr: 9.00

mensaje: hola
```

```
This message is not for me.
Received from: 0x33
Sent to: 0x66
Message ID: 139
Message length: 4
Message: hola
RSSI: -104
Snr: 9.00

mensaje: hola
```

```
This message is not for me.
Received from: 0x33
Sent to: 0x66
Message ID: 143
Message length: 4
Message: hola
RSSI: -103
Snr: 9.25

mensaje: hola
```

```
This message is not for me.
Received from: 0x33
Sent to: 0x66
Message ID: 147
Message length: 4
Message: hola
RSSI: -102
Snr: 9.00

mensaje: hola
```

```
This message is not for me.
Received from: 0x33
Sent to: 0x66
Message ID: 151
Message length: 4
Message: hola
RSSI: -102
Snr: 8.75

mensaje: hola
```

```
This message is not for me.
Received from: 0x33
Sent to: 0x66
Message ID: 155
Message length: 4
Message: hola
RSSI: -100
Snr: 9.50

mensaje: hola
```

```
This message is not for me.
Received from: 0x33
Sent to: 0x66
Message ID: 159
Message length: 4
Message: hola
RSSI: -102
Snr: 9.00

mensaje: hola
```

MENSAJE DE LLEGADA AL NODEMCU SUBIENDO A LA NUBE

```
Received from: 0x33
Sent to: 0x11
Message ID: 28
Message length: 4
Message: hola
RSSI: -110
Snr: 8.25

Enviando: [/LoRa/envia] hola
```

```
This message is not for me.
Received from: 0x33
Sent to: 0x11
Message ID: 32
Message length: 4
Message: hola
RSSI: -112
Snr: 6.75

Enviando: [/LoRa/envia] hola
```

```
This message is not for me.
Received from: 0x33
Sent to: 0x11
Message ID: 36
Message length: 4
Message: hola
RSSI: -109
Snr: 8.25

Enviando: [/LoRa/envia] hola
```

```
This message is not for me.
Received from: 0x33
Sent to: 0x11
Message ID: 40
Message length: 4
Message: hola
RSSI: -110
Snr: 8.00

Enviando: [/LoRa/envia] hola
```

```
Received from: 0x33
Sent to: 0x11
Message ID: 44
Message length: 4
Message: hola
RSSI: -110
Snr: 8.25

Enviando: [/LoRa/envia] hola
```

```
This message is not for me.
Received from: 0x33
Sent to: 0x11
Message ID: 48
Message length: 4
Message: hola
RSSI: -111
Snr: 7.75

Enviando: [/LoRa/envia] hola
```

```
This message is not for me.
Received from: 0x33
Sent to: 0x11
Message ID: 52
Message length: 4
Message: hola
RSSI: -112
Snr: 7.00

Enviando: [/LoRa/envia] hola
```

```
This message is not for me.
Received from: 0x33
Sent to: 0x11
Message ID: 56
Message length: 4
Message: hola
RSSI: -110
Snr: 8.25

Enviando: [/LoRa/envia] hola
```

```
This message is not for me.
Received from: 0x33
Sent to: 0x11
Message ID: 64
Message length: 4
Message: hola
RSSI: -111
Snr: 7.50

Enviando: [/LoRa/envia] hola
```