Universidad ORT Uruguay
Facultad de Ingeniería

Telemetría de contenedores de residuos

Entregado como requisito para la obtención del título de Licenciado en Telecomunicaciones

Leonardo Horovitz – 136910

Raúl Mayobre - 138596

Tutor: Álvaro Sánchez

2018

Declaración de autoría

Nosotros, Leonardo Horovitz y Raúl Mayobre, declaramos que el trabajo que se presenta en esa obra es de nuestra propia mano. Podemos asegurar que:

- La obra fue producida en su totalidad mientras realizábamos el Taller Integrador;
- Cuando hemos consultado el trabajo publicado por otros, lo hemos atribuido con claridad;
- Cuando hemos citado obras de otros, hemos indicado las fuentes. Con excepción de estas citas, la obra es enteramente nuestra;
- En la obra, hemos acusado recibo de las ayudas recibidas;
- Cuando la obra se basa en trabajo realizado conjuntamente con otros, hemos explicado claramente qué fue contribuido por otros, y qué fue contribuido por nosotros;
- Ninguna parte de este trabajo ha sido publicada previamente a su entrega, excepto donde se han realizado las aclaraciones correspondientes.

Leonardo Horovitz	Raúl Mayobre
8 de agosto de 2018	8 de agosto de 2018

Agradecimientos

A Universidad ORT por todos estos años.

A nuestras familias por la paciencia.

A Álvaro Sánchez por el apoyo.

A IFSC Uruguay por la ayuda brindada.

Al equipo de proyecto de Licenciatura en Sistemas.

Abstract

El presente documento 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.

En este documento se detallan y explican los puntos que fueron tenidos en cuenta para la toma de decisiones, así como el detalle técnico del *software* y *hardware* utilizado.

Se contempló la transmisión de distintas variables, incluyendo las coordenadas (obtenidas por GPS), si el contenedor se está incendiando (mediante un sensor de temperatura), y el nivel de llenado (mediante un sensor de ultrasonido).

La información generada por cada contenedor ("nodo") 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.

El servidor de aplicaciones junto con el *software* de toma de decisiones fue desarrollado por el equipo de proyecto de Licenciatura en Sistemas como parte de su trabajo final de carrera.

Palabras clave

- LoRa
- LPWAN
- IoT
- Arduino
- Contenedores de residuos
- Detección de incendio

Indice

1. Introducción

Los problemas relacionados con los contenedores de residuos son publicados frecuentemente en los medios de prensa [1] [2]. Los contenedores son incendiados de manera regular tanto por vandalismo como por descuido de los usuarios; la comuna informa que en caso de incendio el ciudadano debe comunicarse con los bomberos, y en caso de daños se debe comunicar con la comuna [3].

El propósito inicial del proyecto se centró en la detección de incendios. Además, se identificó que conocer el nivel de llenado del contenedor podría aportar un gran valor a la solución, debido a que este dato sumado a la ubicación luego permite implementar sistemas de toma de decisiones.

Un segundo propósito apuntó a resolver el desplazamiento de los contenedores, los cuales son movidos de lugar por distintas causas. Esto se logra conociendo la ubicación actual del contenedor y la que debería tener [4].

El objetivo de este proyecto es conocer en tiempo real el estado del contenedor de residuos (ubicación, nivel de llenado y temperatura) de manera de lograr optimizar los recursos.

Si bien el proyecto se enfocó en los contenedores de residuos utilizados en la ciudad de Montevideo, podría ser aplicable a distintos tipos de contenedores de residuos (ej.: papeleras de las veredas, paradas de ómnibus, etc.), o en entornos distintos a una ciudad (ej.: barrios privados, aeropuertos, centros comerciales, zonas francas, entre otros).

En este documento se presenta el proceso de investigación, diseño y construcción de un sistema que permite obtener el estado de un contenedor de residuos, y luego el envío inalámbrico de esta información a un servidor de aplicaciones para su almacenamiento y procesamiento.

Este proyecto se enmarca en lo que se denomina Internet de la Cosas (IoT, de aquí en adelante) debido a la arquitectura, tecnología y componentes utilizados. Se definió como objetivo la construcción de un prototipo escalable para ser presentado como prueba de concepto, utilizando tecnologías y soluciones con el paradigma IoT. Si bien consideramos que otra opción hubiera sido construir un prototipo con Wi-Fi o redes 2G/3G/4G (comunicación entre nodos y *gateway*), se observó que la tendencia va hacia redes de acceso LPWAN, por lo que finalmente se optó por LoRa.

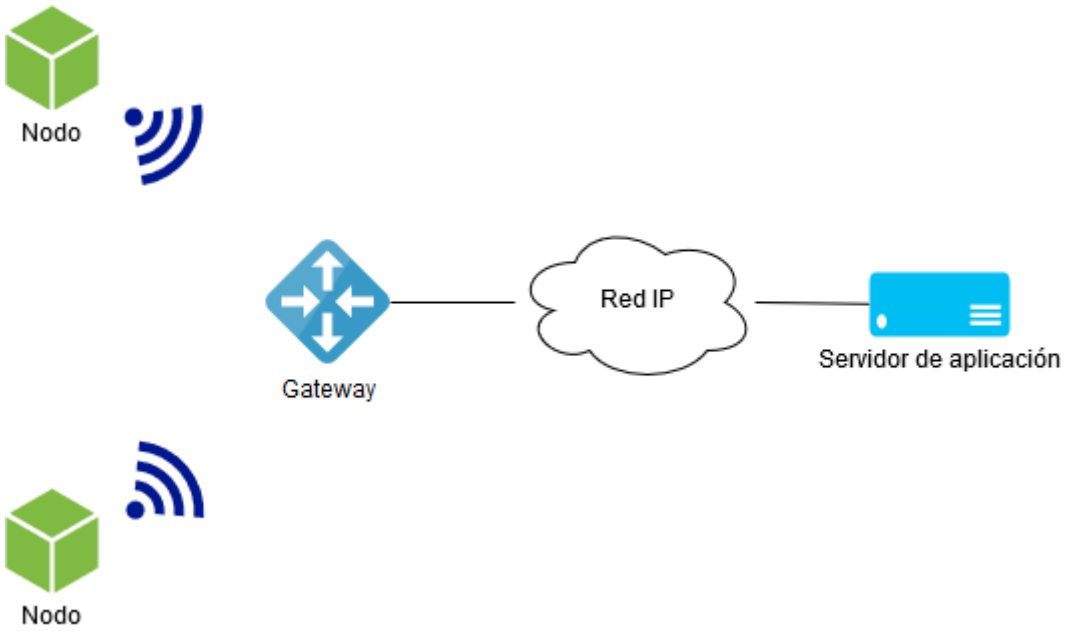


Ilustración 1: diagrama de la solución

El proyecto de Licenciatura en Telecomunicaciones consistió en la elección y ensamblado del *hardware* (sensores, procesador, módulo de radio, y otros periféricos) y desarrollar el *software* del nodo para tomar medidas y luego enviarlas junto con la ubicación a un punto central. Para este punto central (*gateway*) también se desarrolló código para permitir la recepción de información de los nodos para el posterior envío al servidor de aplicaciones en el protocolo y formato definidos.

Se realizó el proyecto en conjunto (con una frontera claramente definida en el ANEXO I y en la sección 9.5.4) con un equipo de la carrera Licenciatura en Sistemas con la finalidad de entregar una solución más completa al problema y además realizar un sistema que sea integrable con otro. Por lo tanto, el proyecto de Licenciatura en Sistemas utiliza la información adquirida y transmitida por los contenedores para, por ejemplo, calcular rutas de recolección, analítica y estadísticas de llenado, etc.

Los integrantes del equipo de proyecto de Licenciatura en Sistemas son Pedro Barrios (182770), Joaquín Muse (175213) y Fabricio Giordano (133968).

Para abordar el problema se realizó una investigación primaria y rápida sobre las distintas tecnologías, protocolos y componentes que podrían ser utilizados. Debido al corto tiempo disponible para este proyecto, no es posible asegurar que las decisiones tomadas sean las

óptimas, dado que no hay un único camino y/o solución para estos nuevos escenarios que requieren de integración de distintas tecnologías y protocolos (muchos de estos en estados iniciales).

Luego del estudio de distintas soluciones comerciales, protocolos, *software* y *hardware* se optó por un conjunto de componentes que resultó accesible económicamente y que de acuerdo con nuestro análisis cumplía los requerimientos planteados. Se intentó crear un prototipo que sea representativo de una solución de mayor escala en la mayor cantidad posible de aspectos.

2. Investigación previa

Se realizó un análisis de cada aspecto que visualizamos como relevante para comenzar a abordar el problema a resolver. A continuación, se presenta un detalle de dicho análisis.

2.1. Contenedores de residuos

El abordaje inicial de este proyecto estuvo basado en la ciudad de Montevideo.

En Montevideo se encuentran distribuidos unos 13.000 contenedores de residuos [5] de 2 tipos básicos (sin incluir los contenedores de las paradas de ómnibus, veredas y otros).

Es de público conocimiento el hecho de que algunos contenedores se llenan y no se recogen, mientras que otros con capacidad ociosa son recogidos. Esto se debe a que no existe información sobre el estado de éstos, antes de la recolección.

Por otra parte, es un hecho habitual que algunos contenedores se incendian (por causas que este análisis no pretende estudiar).

Cuando se incendian sólo acuden los bomberos si algún ciudadano lo informa, pero esto no siempre sucede. Esto implica costos importantes por el reemplazo o reparación del contenedor, y la no disponibilidad hasta su reemplazo o reparación.

En Montevideo se incendian en promedio 250 contenedores por semana. Cada contenedor tiene un costo superior a U$S 1000 [6].

En la ciudad de Montevideo se utilizan 2 tipos de contenedores. Se utiliza un sistema de recolección mediante camiones específicos para cada tipo.

Nuestro análisis preliminar indica que en ambos tipos sería viable instalar los componentes (electrónica, radio y sensores) necesarios para la telemetría de las variables definidas en este proyecto. Para esto sería necesario utilizar gabinetes adecuados (a prueba de agua, humedad, polvo, anti-vandalismo, etc.) y/o desarrollar *hardware* a medida. Serían necesarias modificaciones mínimas a los contenedores (cortes en algunas piezas, instalar tornillos, fijaciones, etc.) para poder adosar estos elementos.

2.1.1. Contenedor tipo A

Ilustración 2: Contenedores tipo A [7]

Este tipo de contenedor, de construcción metálica, posee dos subcategorías de características similares que están destinadas para los residuos orgánicos y para residuos reciclables respectivamente. Su principal diferencia es la abertura a través de la cual se introducen los residuos.

Ambos se apoyan sobre una plataforma que está sujetada mediante fijaciones al suelo, evitando que se pueda volcar o mover. Estos contenedores tienen una capacidad de 2,25 m^3 y 3 m^3 respectivamente [7].

2.1.2. Contenedor tipo B

Ilustración 3: contenedor tipo B [8]

Ilustración 4: etiqueta de contenedor tipo B

Este contenedor es fabricado por la empresa Vetroplast SRL [9] de origen italiano. Posee un volumen de 3200 litros y admite un peso máximo de 1050 Kg. El material de construcción del contenedor es acero, tanto para el recipiente como para la tapa.

2.2.Proyectos similares

En el mercado existen productos comerciales que abarcan problemáticas similares al presente proyecto. La principal diferencia entre este prototipo y otros proyectos radica en que se agrega la variable de detección de incendio.

Algunas de estas soluciones se limitan al *software*, otras al *hardware*, mientras que algunas abarcan ambos aspectos, dependiendo de cada caso.

Productos comerciales similares:

- IBM Intelligent Waste Management Platform [10]
- ECube CleanFLEX [11]
- SmartBin: [12]
- BigBelly: [13]

2.3. Variables a medir

El problema a resolver requiere obtener las siguientes medidas mediante alguna tecnología:

- Fuego/llama: sensor de temperatura y/o llama y/o humo
- Coordenadas: tecnología GPS, GLONASS [14], Galileo [15], triangulación u otra
- Capacidad disponible: sensor de masa y/o sensor de volumen

2.4. Requerimientos de transmisión de datos

Se requiere alguna solución tecnológica de transmisión de datos que permita cumplir con los siguientes requerimientos:

2.4.1. Ubicación y alimentación

Los contenedores de residuos se encuentran distribuidos en la vía pública, tanto en zonas urbanas como suburbanas. El sistema (nodo) a instalar en el contenedor no estará conectado a la red de energía eléctrica. Por lo anterior, se tiene en cuenta como requerimiento un consumo de energía adecuado para ser alimentado por una batería, dado que se pretende construir un prototipo escalable.

2.4.2. Espectro radioeléctrico

Dado que la información de los contenedores se va a enviar inalámbricamente, la tecnología a ser utilizada podrá utilizar bandas libres o bandas licenciadas. El órgano rector para la materia en Uruguay es URSEC (Unidad Reguladora de Servicios de Comunicaciones) [16].

Se definen como “libres” las bandas, sub-bandas, canales y frecuencias que no requieren autorización (decreto 114/2003 [17]).

En particular, las bandas de Uso Libre en Uruguay son las siguientes [18] [19] [20]:

- 40.66-40.70 MHz
- 915 – 928 MHz
- 2400-2500 MHz

2.4.3. Bitrate reducido

Debido a la cantidad y características de la información que se requiere transmitir (coordenadas, temperatura, nivel de llenado y otra información), no se requiere una solución de alta velocidad ni baja latencia. Además, la información del estado del contenedor no requiere ser enviada constantemente, sino cada un determinado lapso, entre otros motivos, porque las variables no evolucionan rápidamente, sino que son procesos físicos que se dan en el tiempo (por ejemplo, un contenedor no se desplaza ni se llena instantáneamente).

2.4.4. Transmisión unidireccional

Debido a que este proyecto apunta a transmitir información de los contenedores, pero no pretende activar elementos dentro de los mismos, no se requiere la transmisión bidireccional, es decir, sólo se requiere transmisión desde cada contenedor hacia un punto central, pero no se requiere en el sentido opuesto, a excepción de las confirmaciones de recepción.

3. Objetivos y alcance

3.1. Objetivos generales

Los objetivos generales del proyecto son:

- Crear una maqueta de un sistema de telemetría para contenedores de residuos.
- Evaluar LoRa como tecnología inalámbrica de comunicación de los nodos.
- Diseñar una solución que permita ser la base de un posible proyecto real.
- Determinar qué consideraciones se deben tener en cuenta al momento de implementar una solución real.
- Lograr la integración a un sistema de un tercero.

3.2. Objetivos del sistema

Los objetivos específicos planteados para esta maqueta son los siguientes:

- Diseñar e implementar una maqueta de escala reducida de un sistema de telemetría de contenedores de residuos capaz de detectar el nivel de llenado, si se está incendiando, y la ubicación.
- Implementar un mecanismo de envío de los datos obtenidos de cada contenedor hacia otro sistema informático.

3.3. Alcance

3.3.1. Alcance general

Se pretende documentar el desarrollo e implementación de un sistema, incluyendo la identificación de requerimientos, la etapa de investigación previa, la definición de objetivos, el ensamblado de los distintos elementos, y las pruebas correspondientes.

3.3.2. Alcance del documento

El alcance del documento incluye la justificación de las decisiones tomadas para la implementación del sistema. Además, se incluye una sección de oportunidades de mejora, las cuales fueron identificadas durante el proceso.

3.3.3. Alcance del prototipo

La maqueta implementada consiste en un *gateway* y al menos un nodo.

Los siguientes puntos no forman parte del alcance del proyecto:

- Telecontrol; accionamiento de elementos en el nodo.
- Alimentación eléctrica del nodo; se asume que será resuelta mediante cambio de baterías.
- Seguridad física y anti-vandalismo de los elementos electrónicos del nodo.
- Adecuación de los elementos al ambiente del contenedor; intemperie, humedad, elementos corrosivos, golpes, etc.
- Diseño y construcción de sensores. Se utilizaron sensores existentes y disponibles comercialmente.
- Diseño industrial que permita la correcta sujeción e integración de los componentes al contenedor.

4. IoT

4.1. Introducción

Con el auge de las TIC en el siglo XXI aparecen nuevas tendencias como Big Data [21] e IoT. Internet no solo ha permitido conectar personas, sino dispositivos, sensores y sistemas complejos. Es así como aparece el término “Internet de las Cosas”, que corresponde a la interconexión de diversos dispositivos a Internet, los cuales generan datos en tiempo real.

Si bien no hay una única definición oficial de IoT, el consenso general es que se denomina IoT a las redes, conjuntos de tecnologías, electrodomésticos, vehículos, y otros dispositivos con electrónica embebida que tienen conectividad a Internet u otras redes y/o entre sí a efectos de permitir el monitoreo y/o control y/o coordinación entre ellos.

Uno de los retos de IoT es la transmisión de datos de sensores a través de cualquier protocolo a cualquier plataforma de forma inalámbrica.

A nuestro entender, desde una perspectiva menos tecnológica y más filosófica y conceptual, el valor de “Internet de las cosas” no radica en la conexión de “objetos” a Internet, sino en la información y conocimiento que estas puedan aportar a la sociedad para mejorar la calidad de vida de las personas.

4.2. Beneficios

El concepto de IoT es de reciente aparición debido a la maduración, proliferación masiva y reducción de costo de tecnologías y componentes como Internet, 3G/4G, Wi-Fi, Arduino, microcontroladores y otros pequeños dispositivos. Esta expansión exponencial es reciente debido a que en los años previos dichas tecnologías y componentes hacían inviables algunos desarrollos por aspectos técnicos y/o económicos.

Las tecnologías IoT resuelven aspectos de medición y/o accionamiento de distintos dispositivos de manera generalmente económica y contemplando un gran número de estos. Este nuevo paradigma impacta a elementos tradicionalmente simples como electrodomésticos (por ejemplo, lavarropas o aspiradoras robotizadas accionables desde el celular), automóviles (por ejemplo, vehículos sin llave que se controlan desde el celular o la nube), y otros segmentos

verticales como ciudades inteligentes (por ejemplo, medidores de energía eléctrica y agua conectados a la nube).

4.3. Things (nodos)

Comenzando a modelar una "cosa" (objeto) en el paradigma IoT, consideramos el siguiente razonamiento: deseamos enviar/recibir información desde un determinado punto geográfico a un servidor en Internet. Se requiere un sensor que pueda obtener la información para luego ser almacenada y/o enviada inalámbricamente. Si consolidamos todos estos elementos en un mismo componente, podría ser considerada como una "cosa" en el concepto de "Internet de las cosas". Esta "cosa" puede ser vista como un nodo dentro de una red de múltiples nodos. Este concepto se denomina "WSN" (Wireless Sensor Network – red de sensores inalámbricos) [22].

4.3.1. Ubicación

En el contexto del nodo en el mundo, su ubicación es el primer aspecto que se debe evaluar. Se deben considerar las decisiones en función de distintos aspectos como, por ejemplo, si se encuentra expuesto a la intemperie o no, a qué distancia está el dispositivo al cual debe enviarle la información, si hay obstáculos físicos, si es posible transmitir de manera inalámbrica, y cómo se resuelve la alimentación eléctrica, entre otros aspectos.

4.3.2. Alimentación

Se debe analizar el problema del suministro de energía, si tiene una fuente de energía infinita o finita. En otras palabras, si está conectado a una red de suministro o utiliza baterías. Si utiliza baterías es necesario planificar durante cuánto tiempo deben suministrar energía, o si se pueden recargar (por ejemplo, con energía solar o eólica). Este punto se debe considerar desde la etapa de arquitectura y diseño.

4.3.3. Características de la información transmitida

Otro factor que se debe considerar es la información que genera este nodo y en una primera instancia los siguientes puntos deben ser evaluados:

- Cantidad de información generada por unidad de tiempo (ej.: bits por segundo).
- Frecuencia de envío de información (ej.: cantidad de mensajes por hora).
- Qué sucede si la información no llega a destino (ej.: considerar reintentos).

En algunos escenarios también se debe evaluar la recepción de información por parte de los nodos para tomar alguna acción.

Las características de la información a transmitir, así como la ubicación afectan directamente el consumo de energía del nodo y definen o descartan que tecnología de comunicación utilizar.

4.4.Redes PAN, LAN, MAN y WAN

Las redes pueden ser clasificadas según su cobertura geográfica. En base a esta clasificación se definen las redes PAN, LAN, MAN y WAN. Cada una hace referencia al área que da conectividad empezando desde pocos metros a miles de kilómetros. En caso de que estas redes sean inalámbricas se agrega la letra W a su denominación.

4.4.1. PAN: Personal Area Network

Una PAN (Personal Area Network - Red de Área Personal) se define como una red de espacio personal del tamaño de área que puede cubrir la voz humana. Estas redes permiten intercambiar información entre computadoras, teléfonos celulares, auriculares y otros dispositivos. Las tecnologías más utilizadas en estas redes son NFC (ISO/IEC 18000-3), ZigBee (IEEE 802.15.4), Bluetooth y Wi-Fi. En el marco de IoT, las redes WPAN (dado que la mayoría de las tecnologías son inalámbricas) se utilizan para la comunicación de las aplicaciones de control y monitorización. La tecnología ZigBee (IEEE 802.15.4) es un protocolo que ha sido diseñado especialmente para la automatización del hogar.

Las velocidades de estas tecnologías típicamente son relativamente bajas (del orden de pocos kbps hasta algunos Mbps) ya que están diseñadas para balancear la performance y el consumo de energía.

4.4.2. LAN: Local Area Network

Una LAN (Local Area Network - Red de Área Local) conecta varios dispositivos de red en un área de corta distancia, en el rango de cientos de metros según el medio de transmisión. Algunos formatos habituales son Ethernet sobre pares trenzados de cobre donde típicamente se alcanzan 90 metros según la versión, fibra óptica hasta algunos cientos de metros y Wi-Fi (IEEE 802.11 en sus múltiples versiones) algunas decenas de metros.

Las velocidades de estas tecnologías son típicamente altas ya que en algunos casos son utilizadas por dispositivos que lo requieren y no tienen grandes restricciones de consumo de energía. Estas velocidades pueden variar desde unos pocos Mbps hasta algunos cientos de Gbps en las tecnologías actualmente disponibles.

4.4.3. MAN: Metropolitan Area Network

Una MAN (Metropolitan Area Network - Red de Area Metropolitana) es una colección de múltiples redes LAN dispersas en una ciudad (decenas de kilómetros). Una MAN utiliza tecnologías como ATM, Frame Relay, xDSL, ISDN, E1/T1, FTTH, MPLS y otras para resolver la conectividad a través de medios de comunicación tales como cobre, fibra óptica y tecnologías inalámbricas (microondas u otras).

Estas redes no tienen restricciones de consumo de energía ya que el equipamiento típicamente es alimentado de la red eléctrica y sus velocidades habitualmente oscilan en la actualidad desde el orden de kbps hasta Gbps.

4.4.4. WAN: Wide Area Network

Una WAN (Wide Area Network - Red de Área Extensa) es una colección de múltiples redes LAN dispersas geográficamente (cientos de kilómetros una de otra).

Un ejemplo típico de una red WAN es Internet, en donde se resuelve la conectividad de redes ubicadas geográficamente en distintos países.

4.5. Análisis preliminar de tecnologías inalámbricas

Al comenzar el proyecto se realizó un análisis preliminar de las tecnologías más difundidas, de las que existen grandes cantidades de implementaciones, documentación y soluciones comerciales que las utilizan.

En este punto se pretende describir brevemente el análisis preliminar de estas tecnologías que fueron evaluadas para ser utilizadas en este proyecto. En una primera instancia se analizaron brevemente las características de las tecnologías Bluetooth, Wi-Fi y 2G/3G.

A continuación, se describen brevemente las características abordadas en el análisis preliminar.

4.5.1. Bluetooth

Bluetooth es una especificación para redes inalámbricas de área personal (WPAN), que permite la transmisión de datos entre diferentes dispositivos mediante un enlace de radiofrecuencia en la banda ISM de 2.4 GHz. Existen múltiples versiones y evoluciones con distintas características de velocidad y consumo de energía, entre otros aspectos.

Los dispositivos que con mayor frecuencia utilizan esta tecnología son los teléfonos móviles, computadoras, y accesorios relacionados a éstos. El rango de alcance de esta tecnología está en el orden de pocos metros. El alcance y *bitrate* varían según la versión.

Clase	Alcance
1	100 m
2	5-10 m
3	1 m

Ilustración 5: Bluetooth - Potencia y alcance [23]

Versión	Bitrate
1.2	1 Mbps
2.0 + EDR	3 Mbps
3.0 + HS	24 Mbps
4.0	32 Mbps

Ilustración 6: Bluetooth – *Bitrate* [23]

4.5.2. Wi-Fi

Wi-Fi [24] es una familia de tecnologías que permiten la interconexión inalámbrica de dispositivos electrónicos. Es una marca propiedad de la Wi-Fi Alliance [25], organización sin fines de lucro que adopta, prueba y certifica que los equipos cumplen los estándares IEEE 802.11 relacionados con redes inalámbricas de área local WLAN. Las distintas versiones de Wi-Fi (802.11 B/G/N/AC, entre otras) permiten distintas velocidades de transmisión (por ejemplo, 11 Mbps, 54 Mbps, 300 Mbps, 1.3 Gbps), entre otras características que las diferencian. Las distintas versiones utilizan bandas ISM de 2.4 GHz y 5 GHz.

Si bien los radios de cobertura teóricos son del orden de unos pocos cientos de metros según la versión, en la práctica (en ambientes urbanos, con interferencia, utilizando canales que están saturados debido a las bandas libres que utiliza, y con obstáculos como ser muros, vehículos u otros elementos de la vía pública) este radio de cobertura habitualmente no supera los 100 metros.

Existe abundante documentación, variedad de dispositivos y oferta de productos comerciales en el mercado ya que es considerada una tecnología robusta y madura.

Característica / Versión	802.11a	802.11b	802.11g	802.11n	802.11ac
Frecuencia de Operación	5 GHZ	2.4 GHZ	2.4 GHZ	2.4 y 5 GHZ	5GHZ
Técnica de Modulación	BPSK, QPSK, 16-QAM, 64-QAM, OFDM	QPSK, DBSPK, DQPSK, CCK, DSSS	BPSK, QPSK, 16-QAM, 64-QAM, OFDM	64-QAM, OFDM, CCK, DSSS	256-QAM
Bitrate máximo	54 Mbps	11 Mbps	54 Mbps	150 Mbps	1300 Mbps
Alcance en exterior	120 m	140 m	140 m	250 m	250 m
Ancho de Banda del Canal	20 MHz	20 MHz y 25 MHz	20 MHz	20 MHz y 40 MHz	80 MHz y 160 MHz
Año de liberación	1999	1999	2003	2009	2014

Ilustración 7: características de versiones de Wi-Fi [26]

4.5.3. Redes celulares (2G/3G/4G)

Las redes celulares son desplegadas y administradas por empresas de telecomunicaciones (TELCOs, en Uruguay: ANTEL, Claro y Movistar). Estas empresas ofrecen la utilización de su infraestructura a cambio de una tarifa.

Estas tecnologías utilizan bandas que son licenciadas en Uruguay, por lo que si se desea implementar un proyecto que las utilice, se debe contratar el servicio. El área de cobertura depende del proveedor seleccionado, y el *bitrate* depende de la versión utilizada, variando desde pocos Kbps hasta pocos cientos de Mbps.

El dispositivo que se conecta a la red celular debe tener la capacidad de implementar el *stack* de protocolos IP. Todo esto se traduce en un aumento de capacidad de *hardware*, elevando el consumo de energía y el costo. Por este motivo, existen desarrollos de estándares [27] que optimizan el consumo de energía de los nodos a ser utilizados en aplicaciones de IoT. Estas alternativas no figuran como productos comercialmente disponibles en los sitios web de los operadores de redes celulares de Uruguay al momento de este proyecto.

4.5.4. Conclusiones

La tecnología Bluetooth presenta un rango de alcance insuficiente, en el orden de pocos metros, lo cual lo hace inadecuado para este proyecto.

Si bien Wi-Fi permite obtener rangos de distancia mayores que Bluetooth, para este proyecto se requeriría de una cantidad muy importante de *Access Points*, del orden de unos pocos en cada manzana, lo cual dificultaría su implementación y gestión, y elevaría considerablemente los costos de la solución.

Utilizar una red celular parece ser la mejor elección debido a su rango de cobertura e infraestructura desplegada. Sin embargo, las versiones actualmente disponibles por parte de los operadores en Uruguay no son las versiones recomendadas por 3GPP [27] [28] para IoT.

Debido a las desventajas presentadas por las tecnologías analizadas, se optó por buscar alternativas. En los últimos años han sido desarrolladas tecnologías y protocolos específicos para el escenario que plantea nuestro proyecto. Estas tecnologías se describen a continuación.

4.6. Redes LPWAN

Debido a que las tecnologías evaluadas en el punto anterior presentan dificultades considerables para la aplicación específica del presente proyecto, se optó por evaluar alternativas. Se plantea desplegar una red para nodos alimentados por baterías. A partir de esto nace el concepto de LPWAN (*Low Power Wide Area Network* - red de área extensa de baja potencia).

Las redes LPWAN son redes inalámbricas diseñadas para permitir comunicaciones de largo alcance a un *bitrate* reducido entre objetos conectados, y alimentados por batería. La energía consumida, el bajo *bitrate* y el uso previsto distinguen LPWAN de WAN, dado que esta última hace referencia a conectar usuarios y empresas, y transportar una mayor cantidad de datos, sin grandes restricciones de consumo de energía. El *bitrate* de las distintas tecnologías LPWAN habitualmente es del orden de unos pocos kbps.

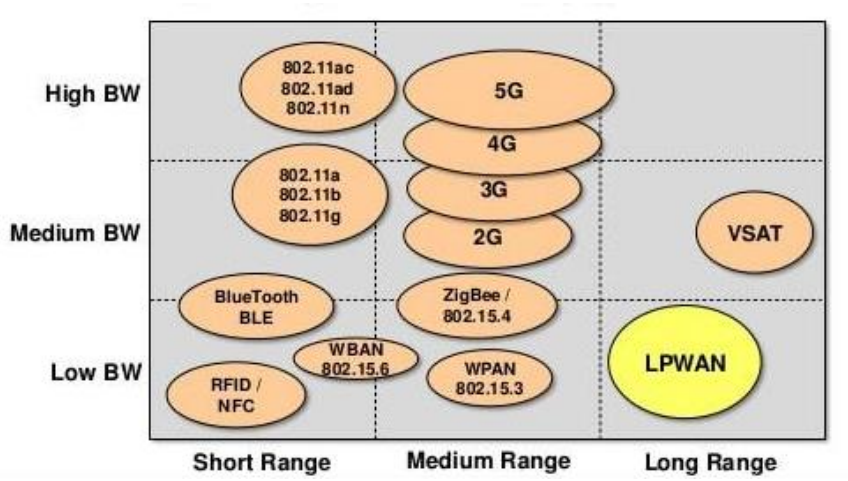


Ilustración 8: tecnologías inalámbricas [29]

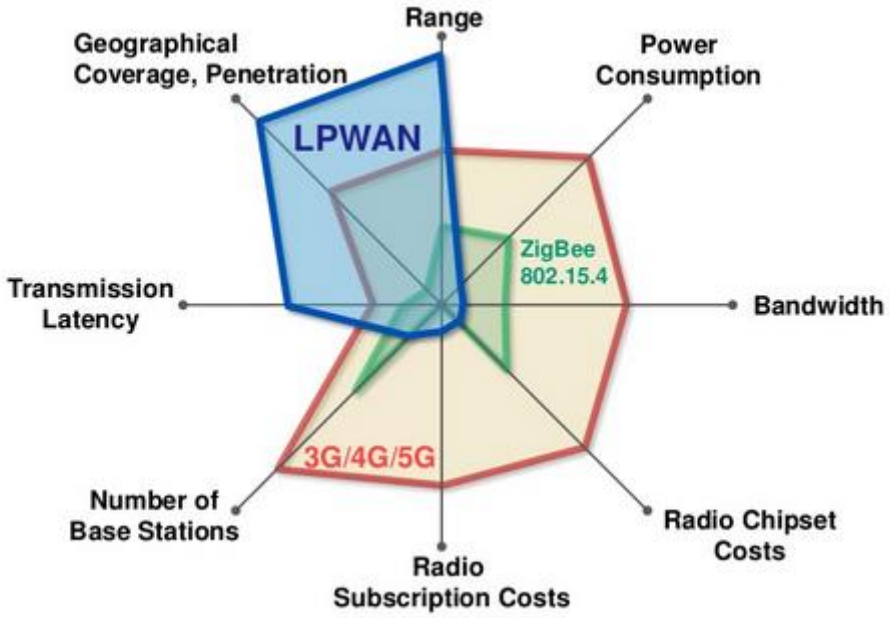


Ilustración 9: LPWAN vs ZigBee vs celular [29]

Una red LPWAN puede ser privada o pública, lo que en este último caso permite a los propietarios de nodos implementar sistemas sin invertir en tecnología ni infraestructura (por ejemplo, el caso de Sigfox, el cual se menciona más adelante).

En la actualidad hay múltiples estándares y proveedores que compiten en el espacio LPWAN para IoT, los cuales se describen en la siguiente figura:

Tecnología	Sigfox	LoRa	NB LTE-M	LTE-M	EC-GSM	5G
Rango	<13 km	<11 km	< 15 km	< 11 km	< 15 km	< 15 km
Banda	No licenciada	No licenciada	Licenciada	Licenciada	Licenciada	Licenciada
Bitrate	<100 bps	<50 kbps	<150 kbps	< 1 Mbps	10 kbps	< 1 Mbps

Ilustración 10: tecnologías de redes para IoT [28]

Si bien algunas de estas tecnologías son consideradas como LPWAN, a nuestro criterio deberían ser catalogadas como WLPMAN (*Wireless Low Power Metropolitan Area Network*) debido a que son inalámbricas y tienen cobertura de área del tamaño una ciudad aproximadamente.

4.6.1. Bandas ISM

Las bandas ISM (*Industrial, Scientific and Medical*) son bandas del espectro radioeléctrico reservadas internacionalmente para uso no comercial en áreas de uso industrial, científico y médico. Estas bandas están definidas por la ITU en las Regulaciones de radio (RR) [18].

El uso de estas bandas de frecuencia está abierto a todo el mundo sin necesidad de licencia, respetando las regulaciones que limitan los niveles de potencia transmitida y ciclos de trabajo [28, p. 328]. Este hecho fuerza a que este tipo de comunicaciones posean cierta tolerancia a errores y que utilicen mecanismos de protección contra interferencias, por ejemplo, mediante técnicas de ensanchado de espectro.

4.6.2. Tecnologías sobre bandas licenciadas

Dentro de las tecnologías que utilizan bandas licenciadas encontramos varias bajo el estándar 3GPP [27]. Desde un principio fueron descartadas debido a que no están disponibles comercialmente en Uruguay. En caso de que alguno de los 3 operadores de telefonía celular de Uruguay (ANTEL, Movistar o Claro) hubiera ofrecido comercialmente estas tecnologías como servicio, hubiese sido una opción para evaluar, pero al momento de la redacción de este documento no figuran entre los servicios promovidos en los sitios web de los operadores.

En el caso de la compañía estatal, de acuerdo con una nota de prensa [30], ANTEL decidió utilizar NB-IoT (que es una de las tecnologías definidas por 3GPP para redes IoT). Sin embargo, a la fecha de redacción de este documento aún no figura entre sus productos disponibles en la web.

4.6.3. Tecnologías sobre bandas libres

En el caso de estas tecnologías se realizó una investigación más profunda debido a que utilizan bandas ISM en las cuales no se requiere licencia y por lo tanto es posible realizar pruebas sin incurrir en costos. Sin embargo, corresponde puntualizar que es necesario homologar el equipamiento ante URSEC.

A continuación, se describen las tecnologías analizadas.

4.6.4. Sigfox

Sigfox [31] es una compañía francesa fundada en 2009. Esta empresa desarrolla una tecnología inalámbrica propietaria en bandas ISM orientada a soluciones IoT.

El modelo de negocio de Sigfox tiene sus particularidades que lo diferencian del resto de tecnologías y operadores. La empresa Sigfox cumple los roles de controlador de tecnología y proveedor de servicios, y realiza la certificación de compatibilidad de los nodos para que estos se puedan suscribir a la red (la cual es administrada de manera exclusiva, también por Sigfox). Esto es análogo al operador de una red celular, porque se abona una tarifa por el uso de la infraestructura, con la diferencia de que toda la red está diseñada exclusivamente para dispositivos IoT.

Esto significa que en el caso de Sigfox las redes ya están creadas, y no es necesario administrar *gateways* o dispositivos de ningún tipo. Sólo es necesario adquirir los nodos homologados por Sigfox para utilizarlos en las redes (desplegadas y operadas por Sigfox). Al igual que en las redes celulares, se debe contratar una suscripción para cada dispositivo (nodo).

Actualmente Sigfox no posee infraestructura ni redes desplegadas en Uruguay, por lo que no es una opción válida para ser utilizada en este proyecto (aunque sí existen despliegues en Brasil y Argentina) [32].

Características principales:

- La topología de red que utiliza es del tipo "*one-hop star*", es decir, desde cada nodo se envía información a un *gateway*, el cual luego la envía a un servidor de aplicaciones.
- Los nodos pueden enviar hasta 140 mensajes por día y cada mensaje puede tener una carga útil de 12 bytes (1680 bytes por día).
- Cada nodo puede recibir un máximo de 4 mensajes por día y estos pueden contener una carga útil de 8 bytes (32 bytes por día).
- Utiliza bandas ISM.
- Utiliza infraestructura instalada, se simplifica el despliegue, pero se depende de un tercero.
- Variedad de fabricantes de circuitos integrados: Texas Instruments, Silicon Labs, ON Semiconductor.
- Tiene un costo de suscripción.

En resumen, la viabilidad de los proyectos que pueden utilizar esta tecnología depende de si existe infraestructura Sigfox en el área a ser desplegados los nodos y también si la cantidad de mensajes enviados o recibidos es suficiente para la aplicación específica.

4.6.5. LoRa

LoRa [33] [34] [35] es una marca patentada de tecnología propietaria de comunicación inalámbrica desarrollada por la empresa francesa Cycleo, que fue adquirida por Semtech en el año 2012. Semtech es el único fabricante de circuitos integrados con tecnología LoRa.

Esta tecnología utiliza bandas ISM. Los circuitos integrados se comercializan según la región [36] [18] de acuerdo con la reglamentación específica de cada país.

La tecnología LoRa refiere a la capa 1 (capa física del modelo de referencia OSI), es decir, la técnica de modulación utilizada en la transmisión y recepción.

Por otra parte, se debe definir (a criterio del operador de la red LoRa) qué utilizar en la capa 2 (control de acceso al medio). La alternativa que actualmente tiene más impulso y apoyo por parte de los distintos fabricantes es LoRaWAN (estándar abierto, de capa 2 definido por LoRa Alliance [37]). Sin embargo, existen alternativas (por ejemplo, Symphony Link [38]).

Adicionalmente existe la posibilidad de hacer un desarrollo propio de acuerdo con las necesidades particulares de cada caso.

Características principales de LoRa:

- Utiliza bandas ISM.
- La arquitectura de red que utiliza es "*one-hop star topology*", es decir, desde los nodos se envía información a un *gateway* y éste la envía a un servidor de aplicaciones. Es posible conseguir arquitecturas *mesh* apoyándose en protocolos que lo implementen en capa 2 y superior.
- El *bitrate* es configurable, pudiendo alcanzar un máximo de 37.5 kbps.
- No hay restricciones de la cantidad de información a enviar y recibir.
- Rango del alcance del orden de pocos km.
- Es posible manipular características de la modulación, las cuales varían el *bitrate*, el consumo de energía, y el rango de alcance obtenido, entre otros parámetros.
- Libertad para construir sistemas a medida.
- Bajo costo (U$S 4,5 – 8,06 según la cantidad, al momento de la redacción de este documento) [39].
- Circuitos integrados fabricados únicamente por Semtech.

4.7.Elección de tecnología de transmisión

En IoT todos los recursos son escasos: memoria reducida, consumos restringidos de energía, menor *bitrate*, menor tamaño físico, y costo. Por lo tanto, el uso de tecnologías LPWAN es necesario para escenarios de nodos que no necesitan enviar ni recibir una gran cantidad de información (en el orden de miles de *bytes* diarios). Además, el tráfico es asimétrico en muchos casos: los nodos envían más información de la que reciben (52:1 en el caso de Sigfox).

Dentro de las tecnologías LPWAN [40] es posible encontrar distintas estrategias de modulación (por ejemplo, banda angosta y espectro ensanchado) para llegar a cubrir largas distancias. También es posible encontrar tecnologías que utilizan bandas licenciadas y no licenciadas.

Dentro de las que utilizan bandas de frecuencias no licenciadas las tecnologías Sigfox, LoRa y DASH7 [41] son 3 ejemplos que utilizan radio de baja frecuencia (denominadas tecnologías sub-GHz, debido a que utilizan frecuencias inferiores a 1 GHz) que permiten la transmisión sin garantía de recepción de poca cantidad de datos en dispositivos IoT, consumiendo poca energía.

Por lo tanto, es posible concluir que la tecnología que se adapta mejor a nuestro escenario para realizar pruebas con un prototipo que utilice LPWAN es LoRa:

- Las limitaciones de envíos y recepciones de mensajes están determinadas por el desarrollador del sistema.
- La administración y despliegue de la infraestructura de la red puede estar a cargo del desarrollador del sistema.
- La única desventaja o riesgo visible es que hay un único fabricante de chips (Semtech), quien además posee las patentes de la tecnología (propietaria).

Es por estos motivos que optamos avanzar en el estudio LoRa, debido a que Sigfox y otras no se encuentran disponibles en Uruguay.

A pesar de que la tecnología de modulación LoRa es propietaria y nueva, los chips que implementan la tecnología LoRa pueden ser incorporados en la construcción de dispositivos debido a que Semtech provee la documentación necesaria. Adicionalmente existen casos de éxito en diversos proyectos reales alrededor del mundo. A modo de ejemplo se mencionan algunos a continuación:

- Suez optimiza la gestión de residuos utilizando LoRaWAN en Holanda [42]
- Recolección de datos de viñas en El Líbano [43]
- Monitoreo de nivel de combustible de calefacción en Bélgica [44]

Adicionalmente, participamos del 4to encuentro de ciudades inteligentes [45] (Montevideo) donde hubo una presentación sobre “Redes de acceso de IoT” en la cual se anunció que la Intendencia Municipal de Montevideo planifica implementar una solución de control de luminarias (entre otras aplicaciones), basada en una red LoRa/LoRaWAN.

4.7.1. Protocolos de intercambio de mensajes

Dado que la mayoría de los sistemas IoT habitualmente se integran con otros sistemas, es necesario contemplar este requerimiento desde el diseño.

En este caso, a partir de la arquitectura definida, se requiere que el *gateway* envíe la información hacia el servidor central (desarrollado por el equipo de proyecto de Licenciatura en Sistemas).

Luego de un breve análisis en conjunto con el equipo de proyecto Licenciatura en Sistemas, se evaluaron posibles protocolos para el intercambio de mensajes, teniendo en cuenta la viabilidad de la implementación en el *gateway*.

Se seleccionó JSON como formato de envío de la información hacia el servidor de aplicación debido a que se trata de un estándar el cual es implementable tanto en el *gateway* como en la plataforma utilizada por el servidor de aplicación. Esta información se envía utilizando HTTP.

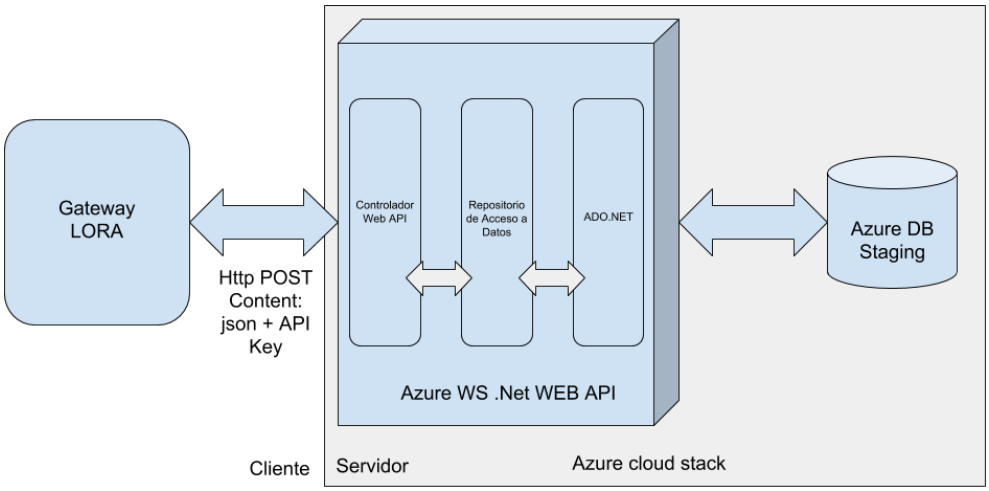


Ilustración 11: arquitectura del *Web Service*

El servidor de aplicación presentará un *Web Service* al cual el *gateway* enviará la información. El contenido detallado del mensaje se describe en el ANEXO I.

Si bien existen alternativas de protocolos especialmente orientados a IoT que ofrecen una mayor eficiencia (por ejemplo, mediante la reducción del *overhead*, como MQTT), estos no fueron utilizados.

5. LoRa y LoRaWAN

LoRa es una tecnología propietaria del fabricante Semtech que define la conectividad a nivel de capa 1 en el modelo OSI. Esta tecnología inalámbrica está diseñada para permitir la interconexión de dispositivos de recursos muy reducidos (energía, memoria y procesador) y a velocidades muy bajas (hasta 37.5 Kbps) [46]. El nombre LoRa deriva de las palabras "*Long Range*" (largo alcance).

Por otro lado, LoRaWAN es un protocolo de capa 2 promovido por LoRa Alliance [37], una asociación sin fines de lucro con la finalidad de desarrollar y promover la tecnología de manera estandarizada e interoperable. La asociación está compuesta por más de 500 empresas que desarrollan activamente soluciones y equipamiento relacionado.

5.1. LoRa

5.1.1. Introducción

LoRa es una tecnología de comunicación inalámbrica, para dispositivos que requieren bajo consumo de energía y admiten un bajo *bitrate*. Está diseñada como solución de infraestructura para IoT. Los nodos se comunican directamente a un *gateway*, el cual tiene un rol de "pasarela" entre los nodos y otros sistemas.

Utiliza un sistema de modulación propietario y típicamente tiene radios de cobertura del orden de 2 km en áreas urbanas y hasta 10 km en áreas rurales (sin obstáculos o interferencia).

Puede implementar topología *peer-to-peer* o *mesh* mediante la utilización de otros protocolos en capas superiores.

LoRa no provee funcionalidades de seguridad, la tecnología sólo resuelve la transmisión, y no garantiza la confidencialidad, autenticidad ni integridad de la información por lo que las capas superiores deben implementar los mecanismos necesarios.

5.1.2. Arquitectura

La arquitectura básica de una solución que utiliza una red LoRa incluye 3 tipos de equipos: nodos, *gateways* (o "pasarelas") y servidor de aplicación. Los nodos se comunican con el *gateway* utilizando LoRa. El *gateway* se comunica con el servidor de aplicación mediante IP.

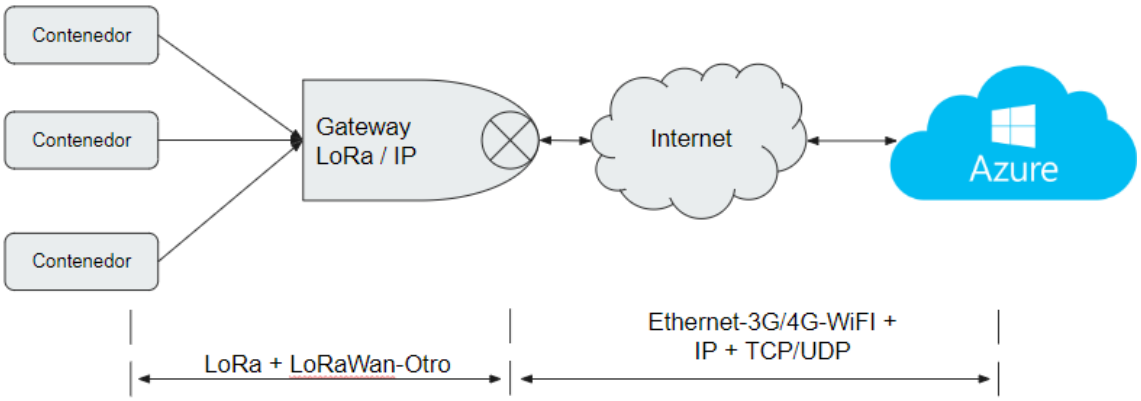


Ilustración 12: diagrama de arquitectura

Es posible implementar redes de tipo *mesh*, en la cual los nodos retransmiten mensajes de otros nodos, pero en este caso los nodos siempre deberían estar recibiendo y transmitiendo mensajes, lo que impacta en un alto consumo de energía de cada nodo. En este caso no estaríamos dentro del marco de IoT y LPWAN. En este caso el nodo podría considerarse un *gateway* de menores prestaciones.

5.1.3. Modulación y espectro

LoRa utiliza modulación de radio basada en *Chirp Spread Spectrum* (CSS [47], [48]). Utiliza las bandas ISM por debajo de 1 GHz (433, 868 y 915 MHz según la versión del chip utilizado). En cada transmisión es posible enviar entre 2 y 255 octetos y la velocidad de transmisión puede llegar hasta los 37.5 Kbps dependiendo del circuito integrado utilizado (SX1276 [46] en nuestro prototipo) y parámetros escogidos.

Debido a la naturaleza propietaria de la técnica de modulación LoRa, hay detalles que no son provistos por el fabricante. Sin embargo, el fabricante provee la información necesaria para dimensionar sistemas y conocer el comportamiento previsto variando los distintos parámetros y configuraciones que están detallados en las hojas de datos.

5.1.4. Parámetros

Hay tres parámetros principales que pueden ser modificados en la modulación LoRa: ancho de banda (BW, *bandwidth*), factor de dispersión (SF, *spreading factor*) y ratio de codificación (CR, *code rate*). Es posible optimizar la modulación LoRa para una aplicación determinada, variando estos tres parámetros. Estos parámetros determinan el *bitrate*, inmunidad a la interferencia y el tiempo que se ocupa el canal en cada transmisión.

5.1.4.1. Factor de dispersión

Se define el *Spreading Factor* (SF) como logaritmo en base 2 de la cantidad de "*chirps*" por símbolo. Cada bit de información (*payload*) es representado por múltiples "*chips*" de información. El ratio en que se envía la información dispersa se refiere al ratio de símbolo (R_s), por lo tanto la relación entre el ratio de símbolo nominal y el ratio de *chips* es el factor de dispersión (*Spreading Factor*).

Este factor incide en el *bitrate* efectivo y la inmunidad a la interferencia por ruido en la detección. Los valores posibles varían de 7 a 12. Un aumento del SF repercute positivamente en la sensibilidad del receptor, pero aumenta el tiempo del símbolo (en consecuencia, se ocupa más tiempo el canal).

El *Spreading Factor* debe ser el mismo tanto para el emisor como por el receptor. Distintos valores de SF son ortogonales entre ellos. Por lo tanto, es posible reutilizar el mismo canal para distintos nodos de manera simultánea variando el SF.

SF	Chips / Symbol
7	128
8	256
9	512
10	1024
11	2048
12	4096

Ilustración 13: características de modulación [49]

5.1.4.2. Ancho de banda

Los anchos de banda típicos utilizados son de 125, 250 y 500 kHz. Este es el ancho de banda que se ocupa en el espectro radioeléctrico durante la transmisión. Es posible utilizar anchos de bandas menores, pero en este caso Semtech recomienda utilizar un reloj externo (TXCO).

Un aumento de uno en el *Spreading Factor* divide la frecuencia del *chirp* en 2. Dado que hay 2^{SF} "*chips*" en un símbolo, un símbolo puede codificar SF *bits* de información.

Por lo tanto, el *symbol rate* (velocidad de símbolo) y el *bitrate* son dados por el SF, los cuales son proporcionales al BW. Queda expresado en la siguiente ecuación, siendo T_s el tiempo de duración de un símbolo para un determinado SF y BW:

$$T_s = \frac{2^{SF}}{BW}$$

Ancho de Banda	SF	CR	Bitrate nominal
125 kHz	12	4/5	293 bps
250 kHz	12	4/5	586 bps
500 kHz	12	4/5	1172 bps

Ilustración 14: características de la modulación [49]

5.1.4.3. Ratio de codificación

LoRa incluye un FEC (*Forward Error Correction*), denominado CR (*Code Rate*) el cual puede tomar los siguientes valores: 4/5, 4/6, 4/7 y 4/8. El primer valor indica cuántos bits son de información (*payload*) y el segundo los bits enviados.

Utilizando la ecuación anterior y sabiendo que SF bits de información son enviados por cada símbolo se puede obtener la ecuación del *bitrate* efectivo de la modulación con los tres parámetros configurables.

$$R_b = SF \times \left(\frac{BW}{2^{SF}}\right) \times CR$$

5.1.4.4. Sensibilidad

Se denomina "sensibilidad" a la mínima señal que un receptor puede decodificar. La modificación de los tres parámetros anteriores influye en la sensibilidad del decodificador, así como en el tiempo de transmisión que se utiliza para enviar la información. Por lo tanto, aumentando el ancho de banda disminuye la sensibilidad del receptor. En cambio, aumentando el *Spreading Factor* la sensibilidad del receptor aumenta. Por otro lado, disminuyendo el *Code Rate* se reduce el ratio de paquetes con error en presencia de pequeñas ráfagas de interferencia, pero el *bitrate* efectivo disminuye.

BW (kHz)	SF	Bitrate (nominal)	Sensibilidad (dBm)
125	7	4557 bps	-123
125	12	244 bps	-137
250	7	9114 bps	-120
250	12	488 bps	-134
500	7	18229 bps	-117
500	12	976 bps	-131

Ilustración 15: sensibilidad y bitrate (con CR = 4/6) [49]

5.1.5. Estructura del paquete LoRa

La estructura del paquete LoRa es la siguiente:

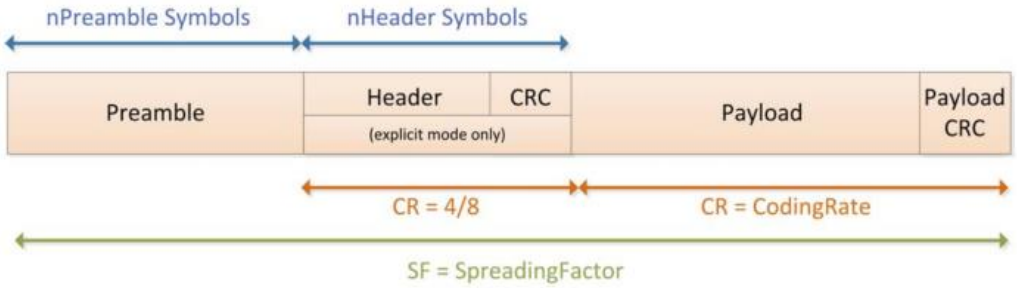


Ilustración 164: Estructura del Paquete LoRa [49, p. 29]

5.1.5.1. Preámbulo

El preámbulo es una serie de *bits* que se envían con la finalidad de sincronizar al receptor con el transmisor. Este valor puede ser parametrizado con valores desde 6 a 65535, quedando un preámbulo de 10 a 65539 respectivamente (debido a que hay 4 símbolos de preámbulo que son fijos). Tanto el emisor como el receptor deben tener configurados el mismo preámbulo para sincronizarse. Como excepción, si el receptor utiliza 65535 como preámbulo puede recibir paquetes con preámbulos menores.

5.1.5.2. Encabezado

Hay dos tipos de encabezados posibles, según el modo que se utilice:

- Implícito (no hay encabezado): el receptor debe conocer de antemano el tamaño del *payload* y del CR (*Code Rate*) utilizado
- Explícito: este es el modo por defecto, en el campo *header* del paquete se envía información sobre el tamaño de la carga, CR (*Code Rate*) y CRC (*Cyclic Redundancy Check* – Verificación de redundancia cíclica).

5.1.5.3. Tiempo en el aire

Luego de seleccionados todos los parámetros de la modulación (SF, BW y CR) y del paquete, es posible calcular el "tiempo en el aire". Esto es el tiempo que se demora en enviar el paquete, durante el que se ocupa el canal de radio.

El fabricante brinda una herramienta simuladora con la finalidad de calcular diversos parámetros, denominada *LoRa Modem Calculator Tool* [46], la cual se encuentra disponible de manera gratuita en el sitio web del fabricante. Esta herramienta permite ingresar los parámetros de la transmisión y en base a estos calcular el tiempo en el aire de cada paquete, así como la energía que consume el circuito integrado.

En nuestro sistema el mensaje del nodo hacia el *gateway* es de 29 bytes, y el de respuesta del gateway hacia el nodo de 14 bytes. Utilizando este software se obtuvo la siguiente tabla (con un mensaje de 29 bytes y CR de 4/6):

Ancho de Banda (kHz)	SF	Bitrate (bps)	Sensibilidad (dBm)	Tiempo en el aire (ms)
62,5	7	2734	-126	134
62,5	10	488	-135	823
62,5	12	146	-140	2965
125	7	5468	-123	67
125	10	976	-132	412
125	12	292	-137	1483
250	7	10937	-120	33
250	10	1953	-129	206
250	12	585	-134	742
500	7	21875	-117	17
500	10	3906	-126	103
500	12	1171	-131	370

Ilustración 17: simulación de valores para una carga de 29 bytes

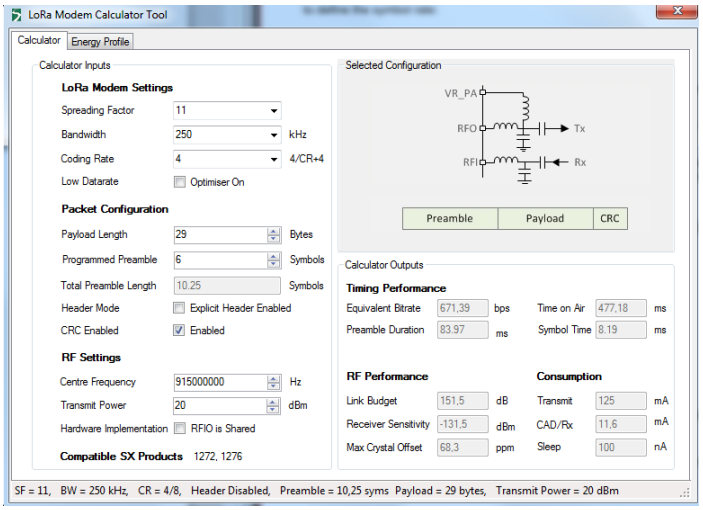


Ilustración 18: LoRa Modem Calculator Tool

5.2.LoRaWAN

5.2.1. Introducción

LoRaWAN es un protocolo de control de acceso al medio (capa 2 del modelo OSI), diseñado para ser implementado sobre redes LoRa. Está basado en topología de varias estrellas utilizando *gateways* como centro de cada estrella. Los *gateways* hacen de "pasarela" entre los nodos y el servidor de aplicación.

Existe un servidor de red cuya función consiste en administrar de manera centralizada los parámetros de los *gateways*.

Es un protocolo de red que esta optimizado para dispositivos alimentados por batería, que pueden estar ubicados en lugares fijos o pueden ser móviles.

Permite variar dinámicamente el *bitrate* de los equipos que están en la red, desde valores de 0,3 Kbps hasta 37,5 Kbps, dependiendo del chip utilizado y parámetros de modulación.

Si bien en esta etapa de construcción de un prototipo no es necesario utilizar este protocolo, resolvería los siguientes puntos en caso de implementar un proyecto real:

- Seguridad: garantiza autenticidad, integridad y confidencialidad de los paquetes
- Tecnología probada y difundida en la industria, garantiza la disponibilidad de equipamiento y documentación.

5.2.2. Arquitectura

La comunicación entre los nodos y los *gateways* se realiza utilizando distintos SF en diferentes canales de frecuencia [50], de distintos anchos de banda. Un *gateway* debe tener al menos 8 módulos de radio en simultaneo, para ser considerado un *gateway* de acuerdo con la especificación técnica de LoRaWAN [51].

Para maximizar la batería de los nodos y la capacidad total de la red, la infraestructura de red puede manejar dinámicamente el *bitrate* y la potencia que utiliza cada nodo individualmente utilizando un esquema adaptativo del manejo del *bitrate*. Esto se denomina ADR (*Adaptative Data Rate*).

Los nodos pueden transmitir en cualquier canal disponible en cualquier momento, utilizando cualquier ancho de banda, siempre y cuando se respeten las siguientes reglas:

- Los nodos deben cambiar aleatoriamente de canal cada vez que transmiten. Esto es para hacer el sistema más robusto e inmune a interferencias.
- Los nodos deben respetar el ciclo máximo de transmisión (% de tiempo que transmite).
- Los nodos deben respetar la duración máxima de transmisión (*dwell time*).
- El *maximum transmit duty-cycle* y el *dwell time per sub-band* son especificados y definidos en capa 1. Esto además se debe ajustar a la normativa vigente, URSEC en el caso de Uruguay.

5.2.3. Clases de nodo

Se definen en LoRaWAN las siguientes clases de dispositivos:

- Clase A: los dispositivos de esta clase permiten realizar comunicaciones bidireccionales, donde el dispositivo luego de enviar información abre dos pequeñas ventanas para recibir información. Los dispositivos envían información cuando ellos lo desean (de manera análoga al protocolo Aloha [52]). Esta clase de operación es la que utilizan los dispositivos con menores recursos de energía. Se utiliza para aplicaciones que envían poca información desde el servidor al dispositivo, y este intercambio se realiza únicamente luego de que el nodo envía un mensaje al servidor. En el caso de que el servidor necesite enviar mensajes al nodo, se debe esperar que éste envíe un mensaje al servidor.

- Clase B: dispositivo bidireccional con slot de recepción agendado; estos dispositivos permiten más cantidad de slots de recepción que los de clase A. Al funcionamiento de los dispositivos de Clase A se añaden ventanas de recepción de información en determinados momentos programados.
- Clase C: siempre están recibiendo información. Solo cierran las ventanas de recepción cuando transmiten. Son los dispositivos que consumen más energía, pero ofrecen la menor latencia de comunicación entre el nodo y el servidor.

Todos los dispositivos LoRa que estén dentro de una red LoRaWAN deben implementar las funciones de la Clase A, y adicionalmente pueden implementar funciones de la Clase B o Clase C según el requerimiento.

5.2.4. Activación de los nodos

LoRaWAN describe 2 posibles modalidades de activación (o registro en la red) de los nodos. Vale aclarar que dentro de la red cada nodo tiene un identificador único (denominado DevAddr) y además cada aplicación tiene una clave de seguridad única (denominada AppSKey), que permite que cada nodo envíe información a un determinado sistema. Esto es debido a que no todos los nodos envían información al mismo sistema, ya que distintos nodos pueden alimentar distintos sistemas, pero formar parte de la misma red LoRaWAN. Adicionalmente, existe una segunda clave a nivel de red para permitir el ingreso del nodo a la red (denominada NwkSKey).

Las modalidades de activación son las siguientes:

- *Activation By Personalization* (ABP): esta modalidad permite la definición de forma manual del *DevAddr* y las claves de seguridad del nodo. Cuando se configura el nodo se ingresan manualmente estos parámetros.
- *Over The Air Activation* (OTAA): es considerada como la modalidad preferida y más segura. En esta modalidad el *DevAddr* y claves de seguridad son negociadas con el dispositivo de manera análoga a la negociación mediante DHCP en una red IP. Esta negociación tiene lugar entre el nodo y el servidor de red.

5.2.5. Adaptive Data Rate (ADR)

El *Adaptive Data Rate* (ADR, control adaptativo de *bitrate*) se puede aplicar sobre cada nodo si este lo tiene habilitado. El servidor de la red puede controlar de manera automática el *bitrate* y la potencia de transmisión de los nodos.

El ADR se utiliza con la finalidad de mejorar la duración de la batería de los nodos y aumentar la capacidad total de la red.

5.2.6. Seguridad

LoRaWAN provee seguridad mediante criptografía y autenticación mutua a nivel de red y a nivel de aplicación. Esto se resuelve a nivel del protocolo con el objetivo de reducir el consumo de energía y evitar utilizar tecnologías como podrían ser VPN IPSec o TLS. Este es un punto fundamental para dispositivos de recursos reducidos (tanto a nivel de *hardware* como de energía).

La clave a nivel de red brinda seguridad entre cada nodo y el servidor de red, mientras que la clave a nivel de aplicación brinda las garantías necesarias para el tráfico entre el nodo y el servidor de aplicación. Las claves se utilizan para cifrar y firmar el tráfico en ambos niveles, estas son configuradas u obtenidas por los nodos según el mecanismo de activación que utilicen.

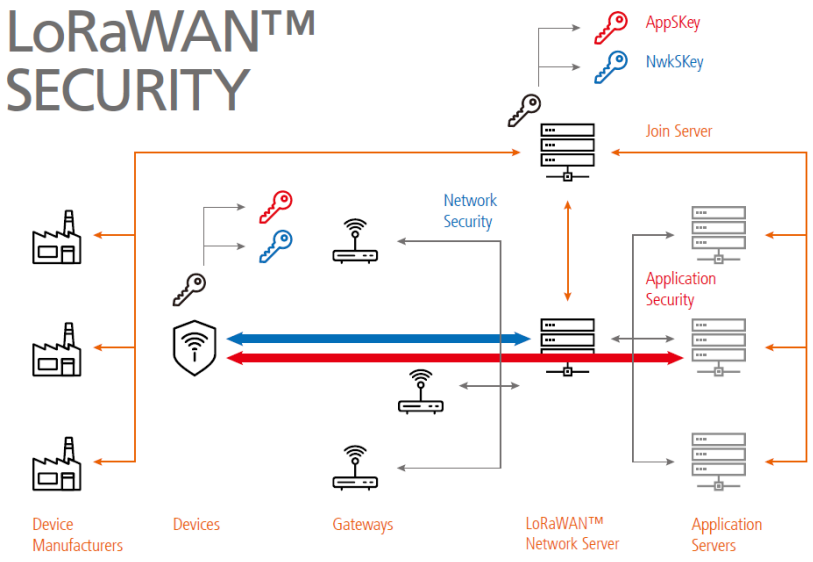


Ilustración 19: Seguridad en LoRaWAN [53]

6. Descripción del sistema

6.1. Introducción

El sistema real (de producción) consistiría de múltiples *gateways* y nodos basados en una arquitectura de "*one hop star topology*", siguiendo la arquitectura habitual de LoRa en este tipo de proyectos.

Se define como "nodo" en nuestro sistema a un conjunto de componentes que tienen la capacidad de medir variables en un contenedor de residuos y transmitirlas de manera inalámbrica.

Se define como "*gateway*" en nuestro sistema al dispositivo que recibe los mensajes de múltiples nodos y los envía al servidor de aplicación a través de una red IP. Este dispositivo tendrá un rol análogo al de una radio base celular, con una ubicación fija y conectado a la red de alimentación.

Se coloca un nodo por cada contenedor de residuos, alimentado con baterías. Cada nodo cuenta con recepción de coordenadas, sensor de nivel de llenado, y sensor de incendio y envía estos datos a un *gateway*. La comunicación entre *gateway* y nodos se realiza mediante LoRa.

Se define un mensaje de envío del nodo hacia el *gateway* y un mensaje de confirmación de recepción desde el *gateway* hacia el nodo.

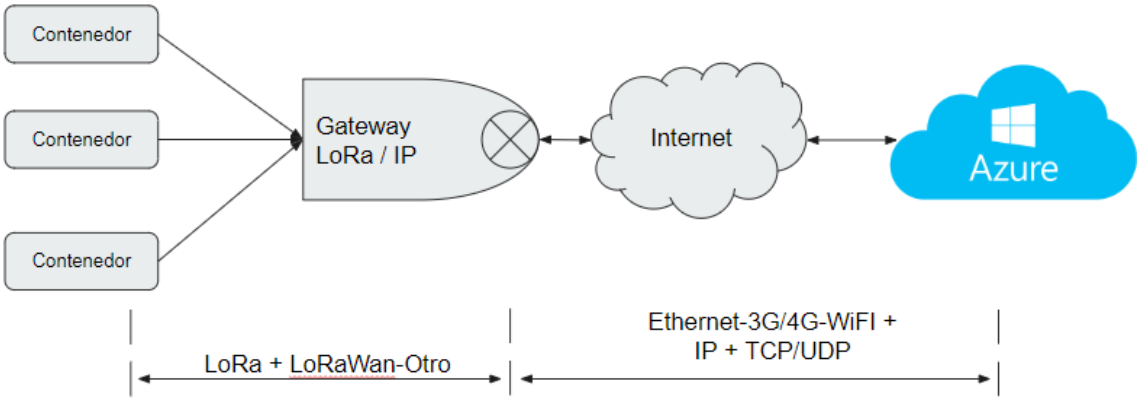


Ilustración 20: Arquitectura del sistema

Se definió que, por cuestiones de plazos, presupuesto y alcance la maqueta a construir consistiría en un *gateway* y uno o dos nodos.

6.2.Descripción del nodo

Se definió que el nodo deberá poder obtener las variables definidas anteriormente (temperatura, nivel de llenado y ubicación).

El nodo deberá contar con un módulo base (CPU, memoria e interfaces de conexión) para poder conectarse a los distintos periféricos (sensores y módulos de radio) y tener la capacidad para ejecutar el *software* necesario para adecuar y hacer un procesamiento básico sobre la información obtenida. Deberá tener la capacidad de enviar la información utilizando LoRa.

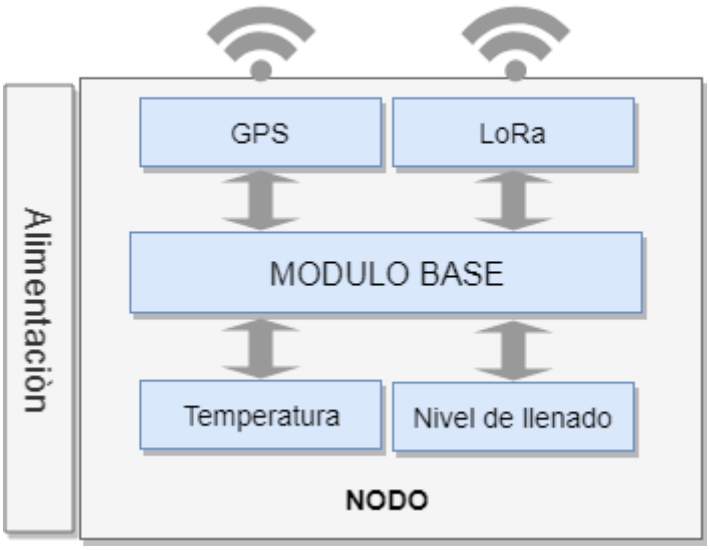


Ilustración 21: diagrama de bloques del nodo

El nodo será un pequeño dispositivo que tendrá un costo reducido debido a que podrá sufrir vandalismo, golpes e incendios. No se requiere que sobreviva a un incendio. Por otra parte, existirán miles de nodos, con lo cual un aumento del costo unitario impacta de manera significativa en el costo total del proyecto.

Cada nodo tendrá un identificador único dentro de la red.

6.3.Descripción del gateway

El *gateway* será un dispositivo que estará mejor protegido de los distintos factores en comparación a los nodos, ya que estará instalado de manera fija y no estará al alcance de los

ciudadanos. Admite un costo superior a los nodos debido a la cantidad que deberán ser desplegados.

Deberá contar con conectividad IP hacia el servidor de aplicación (Internet u otra red IP).

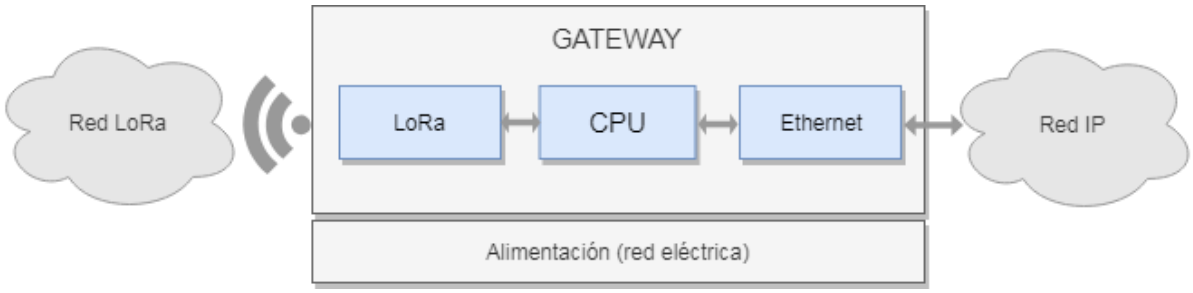


Ilustración 22: Diagrama de bloques del *gateway*

Cada *gateway* tendrá un identificador único dentro de la red.

6.4.Estructura de los mensajes

Se define que el mensaje desde el nodo hacia el *gateway* debe contener la siguiente información:

- Identificador del nodo
- Identificador del *gateway*
- Fecha y hora
- Coordenadas (latitud y longitud)
- Temperatura
- Nivel de llenado
- Otros (a definir, como información de diagnóstico, parámetros de transmisión u otros)

Se define que el mensaje de *acknowledge* (reconocimiento de recepción) desde el *gateway* hacia el nodo debe contener la siguiente información:

- Identificador de *gateway*
- Identificador de nodo
- Otros a definir

Se entiende que no es necesario definir dentro del *acknowledge* un identificador del mensaje específico al que se reconoce debido a que los tiempos entre mensajes (5 minutos en nuestra implementación) impiden que existan 2 mensajes en el aire al mismo tiempo.

6.5. Adquisición de componentes

Una vez definida LoRa como tecnología de transmisión y la arquitectura a implementar, se hizo una breve búsqueda de kits y/o módulos que utilicen esta tecnología. Debido a que LoRa es una tecnología relativamente nueva, no existen muchas opciones en el mercado en lo que a *gateways* y kits de desarrollo se refiere.

Debido a los plazos del proyecto, se contó con poco tiempo para este análisis, en el cual se encontró el *Dragino LoRa IoT Kit* [54]. Este kit está compuesto por un *gateway*, dos placas Arduino UNO, dos placas que incorporan LoRa (una de ellas con GPS), y múltiples sensores. En este breve análisis del kit se estimó que cumpliría los requerimientos planteados para el proyecto, y por lo tanto se definió su compra. Dentro del análisis se estimó una alta viabilidad de la integración con el proyecto de Licenciatura en Sistemas.

Dentro de la compra también se incluyeron sensores y otros accesorios genéricos (*protoboard*, cables, fuentes de alimentación, etc).

Este kit y demás accesorios tuvieron un costo de U$S 315 (trescientos quince dólares americanos).

7. Nodo

Para la definición de la compra (sección 6.5 de este documento), se tuvieron en cuenta los aspectos detallados a continuación.

7.1. Arquitectura

El nodo deberá tener la capacidad de obtener las medidas de los sensores, coordenadas y además deberá poder transmitir la información.

Estos 3 requisitos definen los siguientes bloques:

- Sensores (temperatura y nivel de llenado)
- Ubicación (GPS)
- Transmisión (LoRa)

Esto define la necesidad de un bloque central que vincule todos estos periféricos, a efectos de lograr la obtención, almacenamiento, procesamiento y transmisión. Por lo tanto, es necesario contar con múltiples interfaces de conexión entre el módulo base y los distintos periféricos.

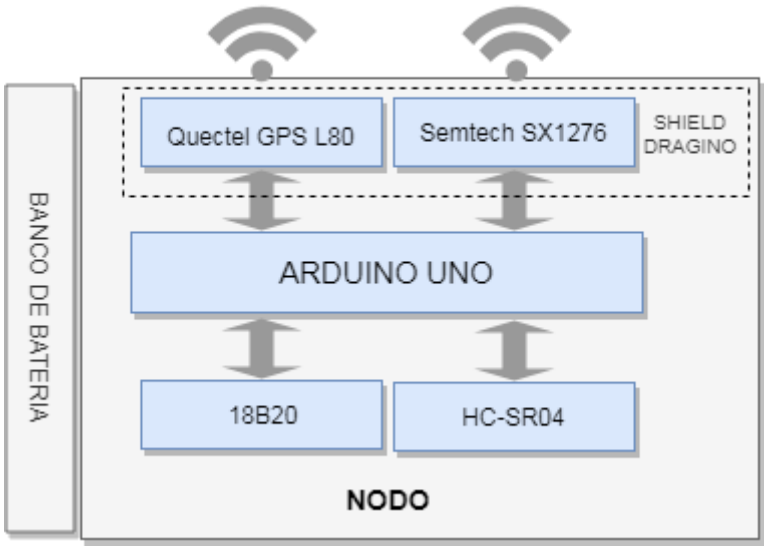


Ilustración 23: diagrama de bloques del nodo

Se describe cada uno de estos bloques a continuación.

7.2.Módulo base del nodo

Se definió como parte de la arquitectura la incorporación de un módulo base, con CPU y memoria en el cual pueda ejecutarse el *software* necesario para recolectar las medidas, adecuarlas y hacer un procesamiento básico, y posteriormente transmitirlas.

Este módulo base debe incluir o permitir incorporar los distintos sensores y tecnologías de transmisión inalámbrica.

Existen en el mercado distintos productos comerciales que calificarían dentro de esta definición: microcontroladores, dispositivos SoC (*System-On-Chip*), placas que incorporan microcontroladores y otros elementos (por ejemplo, transmisores inalámbricos y/o sensores), e incluso placas con sistema operativo.

Dentro del diseño se definió que no es necesario que el módulo base posea un sistema operativo, pero tampoco es un elemento excluyente.

Algunos módulos Arduino cumplen con las características planteadas.

Arduino es un fabricante italiano de placas basadas en microcontroladores de muy bajo costo. Existen múltiples modelos disponibles en el mercado y son fabricadas por múltiples proveedores. El diseño de *hardware* y *software* está basado en la filosofía *open source*, por lo que existe información y documentación ampliamente difundida.

En particular se analizó el modelo UNO [55], que se caracteriza por su simplicidad de uso, su bajo costo, robustez y flexibilidad. Posee una gran variedad y número de entradas/salidas analógicas y digitales. En general toda la familia Arduino se caracteriza por contar con una gran cantidad documentación disponible y la posibilidad de desarrollar el código en lenguaje C/C++.

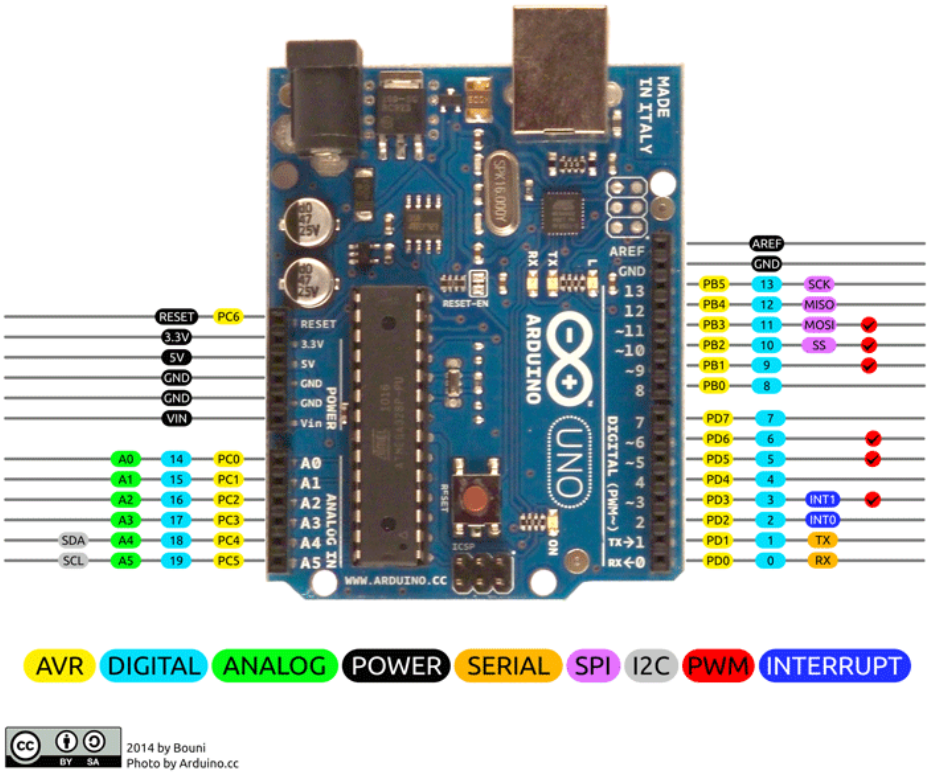


Ilustración 24: Arduino UNO [56]

Características principales:

- 14 puertos digitales
- 6 puertos analógicos/digitales
- Memoria SRAM: 2 KB
- Memoria EEPROM: 1 KB
- Memoria Flash: 32 KB
- Reloj: 16 MHz
- Peso: 25g
- Dimensiones: 68,6 x 53,4 mm
- Microcontrolador: ATmega328P
- Alimentación: 7-12 V

En Internet existen comunidades de proyectos con Arduino basadas en filosofía *open source*. Por lo tanto, es posible encontrar una gran variedad de librerías, código de ejemplo y foros de consultas.

En el mercado existe una variedad de accesorios (sensores, actuadores, pantallas, motores, *shields*, transmisores, gabinetes, etc.) los cuales se encuentran documentados y poseen librerías, permitiendo así rápidamente su implementación.

Es de importancia mencionar el costo reducido de las placas Arduino. En particular, al momento de la redacción de esta documentación, el Arduino UNO (original) tiene un costo de U$S 22.00 en el mercado [55] (y existen de otros fabricantes, de menor costo).

Pudimos concluir que se trata de una plataforma muy flexible y adecuada para construir prototipos de las características buscadas en este proyecto.

7.3.Periféricos

7.3.1. Detección de incendio

Existen múltiples técnicas que permiten detectar un incendio. La técnica más intuitiva que surge consiste en medir un aumento de temperatura. Esto tiene como ventaja la simplicidad, pero en algunos casos este aumento se da cuando el incendio se encuentra en una etapa avanzada. Se definió que, para la etapa de construcción de una maqueta, esta técnica sería adecuada.

La etapa inicial de desarrollo y pruebas se basó en un sensor del tipo DHT11 [57]. Sin embargo, el sensor admite una temperatura máxima de 50 °C, la cual es claramente insuficiente para el propósito de la detección de incendio. Por este motivo se definió la utilización de un sensor del tipo 18B20 [58], el cual admite una temperatura máxima de 125 °C, la cual entendemos como suficiente para validar el concepto en esta etapa de prototipo.

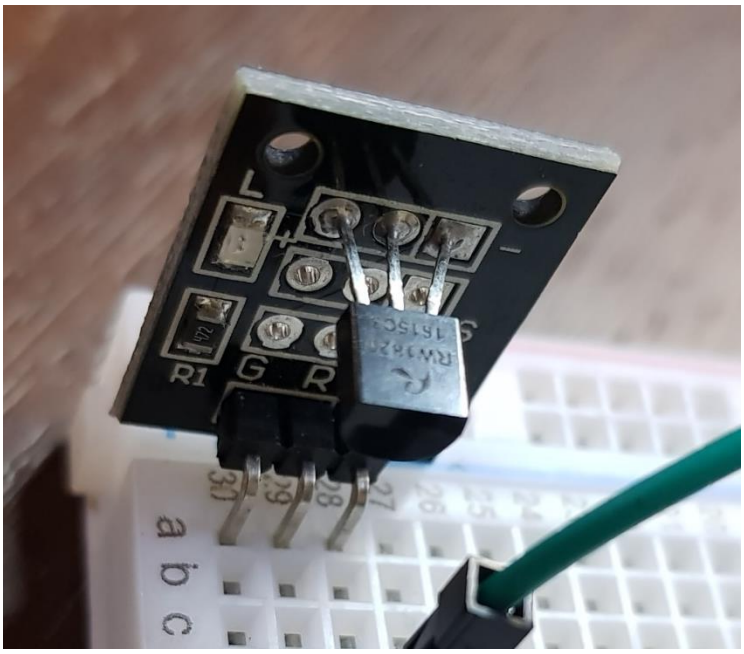


Ilustración 25: sensor 18B20

De acuerdo con nuestro análisis, la ocurrencia de un “incendio” no está definida por la superación del umbral de una temperatura específica, sino en un aumento rápido de temperatura. Es decir, 50 °C podría indicar un incendio, pero también podría ser una temperatura normal en un contenedor expuesto al sol en verano. Sin embargo, de manera intuitiva, un aumento de determinada cantidad de grados por unidad de tiempo podría ser un indicativo de un incendio. Es por este motivo que se optó por definir este esquema para la detección. El nodo envía la medida de temperatura, y el cálculo del aumento es realizado por el servidor de aplicaciones del equipo de proyecto de Licenciatura en Sistemas.

Otra posible técnica es la detección de la llama mediante un sensor sensible a la luz en longitudes de onda infrarrojas. Esto tiene como desventaja la detección de falsos positivos en caso de recibir luz solar de manera directa, y además requiere un incendio en etapa avanzada (con llama) para detectar. Es decir, sólo detecta llamas, no así otros focos (brasas, etc).

Otra posible técnica es la detección de humo. Esto tiene la ventaja de poder detectar el incendio en una fase temprana (aún sin llama). Sin embargo, los distintos sistemas de detección de humo son sensibles al polvo y la contaminación.

El problema de la existencia o no de un incendio es un tema muy amplio y complejo, que este documento no pretende abordar en profundidad por cuestiones de alcance.

El sensor 18B20 utilizado tiene una precisión 0,5 °C hasta 85 °C y de 1 °C por encima de 85 °C, con una resolución (configurable) de entre 9 y 12 bits, y permite la lectura del valor con una frecuencia del orden de 1 segundo, lo cual se entendió que era suficiente para el propósito.

7.3.2. Detección del nivel de llenado

Es posible detectar el nivel de llenado de un contenedor de residuos mediante múltiples técnicas, o incluso una combinación de éstas.

Una posibilidad es medir el volumen ocupado o disponible.

Otra posibilidad es medir la masa (peso) del contenedor. Sin embargo, nuestro análisis preliminar indica que hubiese sido necesario modificar el contenedor para permitir la instalación de elementos, y estas modificaciones son complejas y costosas (instalar y proteger cables que lleguen al piso, adosar sensores en el piso, etc.). Por este motivo se descartó la medición utilizando esta técnica.

Finalmente se decidió implementar la detección del nivel de llenado mediante un sensor de ultrasonido. En particular, en este caso se definió la utilización de un sensor HC-SR04.

Ilustración 26: sensor de ultrasonido [59]

Este tipo de sensor utiliza ultrasonido para medir distancias. Su principio de funcionamiento es la emisión de una onda de ultrasonido, la cual refleja contra el objeto a medir, y la diferencia de tiempos entre la emisión y su eco permite determinar la distancia.

Las características del sensor se mencionan a continuación [60]:

- Rango de medición: 20 – 4000 mm
- Ángulo de medida: 15°
- *Trigger*: pulso de 10 µs
- Resolución: 3 mm

El rango de medición es adecuado para el propósito que persigue el proyecto ya que la distancia a medir en los contenedores no supera los 200 cm, y la aplicación no requiere una gran resolución ni precisión en la medida. Se define como suficiente una precisión de 15 cm.

7.3.3. Ubicación

Se requiere obtener las coordenadas de la ubicación del contenedor. Existen distintas alternativas, la más popular es GPS (*Global Positioning System*) pero existen otras opciones como GLONASS [14] y Galileo [15], así como alternativas basadas en triangulación de la propia red de datos (mediante LoRaWAN [61] u otras tecnologías celulares).

Se optó por GPS debido a que el kit adquirido [54] incorpora un módulo Quectel L80 [62].

Este circuito integrado, de dimensiones reducidas, incorpora la electrónica necesaria e incluye una antena del tipo *patch* en el propio circuito integrado, lo cual lo hace adecuado para la construcción de dispositivos pequeños. Adicionalmente, provee la posibilidad de conectar una antena externa. Sin embargo, esta opción no fue utilizada.

El circuito integrado utiliza el estándar NMEA 0183 [63] para entregar las coordenadas, velocidad y otros parámetros mediante un puerto serie al módulo Arduino a través del *shield*.

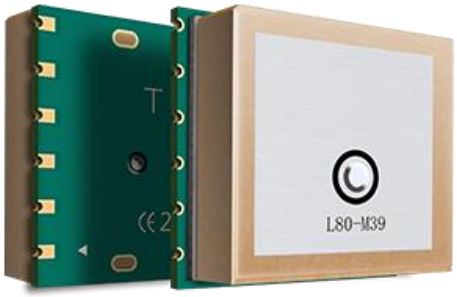


Ilustración 27: Quectel L80 [62]

Ilustración 28: foto del módulo base + shield LoRa/GPS

7.3.4. Módulo LoRa

Se utiliza para la comunicación inalámbrica mediante LoRa un circuito integrado Semtech SX1276 [49]. Este puede ser parametrizado para operar en el rango de frecuencias comprendidas entre 862 MHz y 1020 MHz, y en particular y permite utilizar la banda de 915 MHz, que es la adecuada a la normativa vigente definida por URSEC.

7.4.Placas de expansión

Las placas de expansión (denominados "*shield"* en la jerga de Arduino) se utilizan para añadir funcionalidades mediante conjuntos de componentes, y resuelven la conexión sin requerir de soldaduras, y la sujeción a la placa Arduino. Se adosan a los módulos Arduino UNO en su parte superior.

En nuestro sistema se utilizan dos tipos de *shields*. Ambos utilizan el circuito integrado Semtech SX1276 [64], y uno de ellos adiciona los componentes necesarios para la obtención de coordenadas mediante GPS [65] (basado en el Quectel L80).

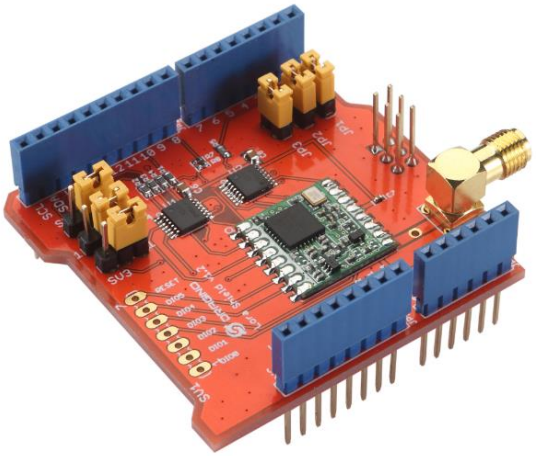


Ilustración 29: shield LoRa [64]

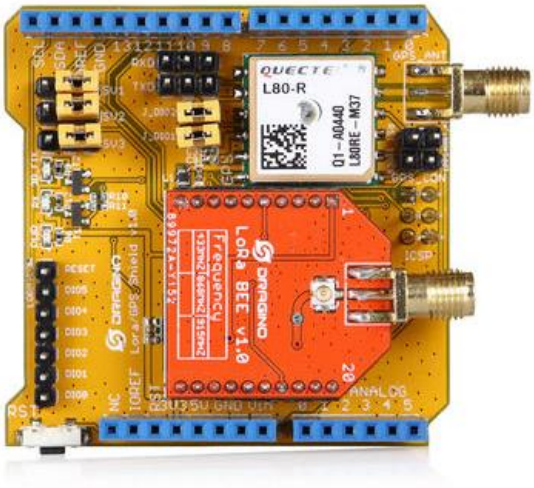


Ilustración 30: shield LoRa GPS [65]

8. Gateway

Para la definición de la compra (sección 6.5 de este documento), se tuvieron en cuenta los aspectos detallados a continuación.

8.1. Arquitectura

En la arquitectura se define la incorporación de un *gateway* cuya función será la de recibir los mensajes de los nodos, adecuarlos y transmitirlos vía IP al servidor de aplicaciones.

El *gateway* utilizado es el Dragino LG01-P [66].

Ilustración 31: Gateway Dragino LG01-P [66]

La arquitectura del *gateway* tiene la siguiente estructura de bloques:

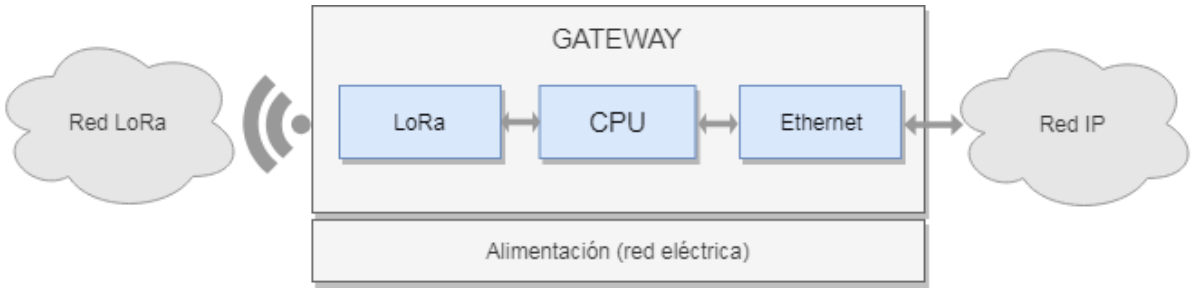


Ilustración 32: diagrama de bloques

La arquitectura del *gateway* Dragino LG01-P se basa internamente en 2 "hemisferios" básicos. Uno está de cara a la red LoRa y el otro hacia la red IP.

Las características del equipo son las siguientes:

- Hemisferio IP (Linux OpenWRT):
  - Procesador: Atheros AR9331
  - Flash: 16 MB
  - RAM: 64 MB
  - 2 interfaces Ethernet
  - 1 interface Wi-Fi 802.11 b/g/n
- Hemisferio LoRa (Arduino)
  - Microcontrolador ATmega328P (el mismo utilizado en el nodo)
  - Flash: 32 KB
  - RAM: 2 KB
  - Semtech SX1276 (el mismo utilizado en el nodo)

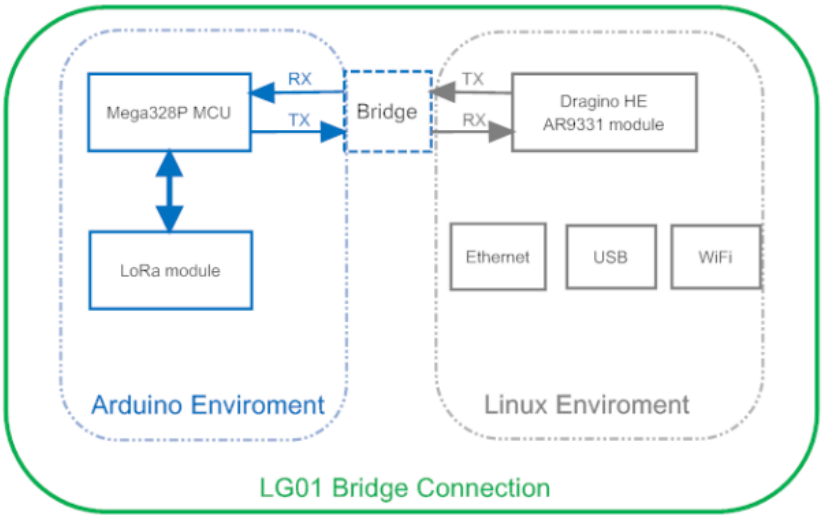


Ilustración 33: Arquitectura del gateway [66, p. 35]

El "hemisferio" de cara a la red LoRa está basado en un ATMega 328P (similar al hardware del Arduino UNO) y el módulo del "hemisferio" de cara a la red IP está basado en un SoC Atheros AR9331. Éste último utiliza OpenWRT, y es el responsable de la gestión de paquetes, *buffers* y otras funcionalidades necesarias para lograr la integración de las dos tecnologías.

8.2. Sistema operativo

El Dragino LG01-P utiliza OpenWRT como sistema operativo, sobre el *hardware* Atheros AR9331.

OpenWRT [67] es una distribución Linux orientada a dispositivos embebidos dedicados al enrutamiento de tráfico de redes. Originalmente nace con el objetivo de reemplazar los firmwares de *routers* hogareños. Esto surge ante la demanda de los usuarios de estos equipos que sufrían de distintos problemas (ej.: equipos "congelados", necesidad de reiniciarlos regularmente, falta de documentación, herramientas y características en el equipo, etc).

OpenWRT surge como respuesta a estas demandas, mejorando distintas características (performance, funcionalidades, estabilidad, entre otras) así como brindando las distintas ventajas de la utilización de código abierto (posibilidad de identificar y resolver problemas, documentación, entre otras).

Debido a que OpenWRT es una distribución de Linux y además es *open source*, es posible contar con documentación a efectos de comprender y modificar parámetros en caso de ser necesario.

Adicionalmente tiene la ventaja de no poseer costos de licenciamiento, debido a su naturaleza *open source*.

Existen múltiples evidencias y casos de éxito documentados en Internet de distintas instalaciones de distinto porte (desde un *router* hogareño hasta instalaciones comerciales para cientos de dispositivos).

9. Implementación

En esta sección se describe el proceso seguido para la implementación del prototipo, así como algunas definiciones, particularidades y cuestiones propias del proyecto. Esto incluye el ensamblado, desarrollo de código y decisiones tomadas en el proceso.

9.1. Introducción

Una vez seleccionado el hardware se comenzó con el proceso de familiarización con el kit, debido a que no contábamos con experiencia en la utilización de dispositivos Arduino.

Esto comenzó con la simple instalación de los códigos de ejemplo disponibles a través del fabricante del *hardware* [68], y la modificación y manipulación para comprender el funcionamiento.

Parte de las pruebas iniciales fueron ejecutadas utilizando la plataforma gratuita ThingSpeak [69], debido a que en una primera etapa el servidor de aplicación (desarrollado por el equipo de proyecto de Licenciatura en Sistemas) aún no estaba operativo. Una vez que estuvo operativo, se continuaron las pruebas sobre este último.

Una vez familiarizados con el *hardware* adquirido y sus librerías, se comenzó con el desarrollo del código basado en los ejemplos y librerías disponibles.

La dificultad principal de la implementación estuvo en el desarrollo y parametrización del *software*, dado que el *hardware* no presentó dificultades de relevancia. Los principales desafíos fueron el estudio del kit, la familiarización con las herramientas propias de Arduino y el desarrollo en el lenguaje, basado en C/C++.

El *hardware* no presentó dificultades de relevancia en el sentido que, una vez instalados los códigos de ejemplo y realizadas las conexiones, en la mayoría de los casos funcionaron casi de inmediato.

Si bien en la mayoría de los casos tanto los códigos de ejemplos como el *hardware* contaban con alguna documentación, la misma generalmente no fue suficiente ni clara, por lo que en algunos casos esto presentó alguna dificultad o requirió investigar en profundidad algún aspecto, lo que insumió tiempo.

Cuando se decidió utilizar LoRa como tecnología de capa 1, también se tuvo en cuenta la utilización de LoRaWAN como protocolo de capa 2, debido a que el control de acceso al medio que realiza lo hace adecuado para nuestro sistema. Sin embargo, no fue posible implementarlo y realizar ensayos debido al plazo para la realización de este proyecto.

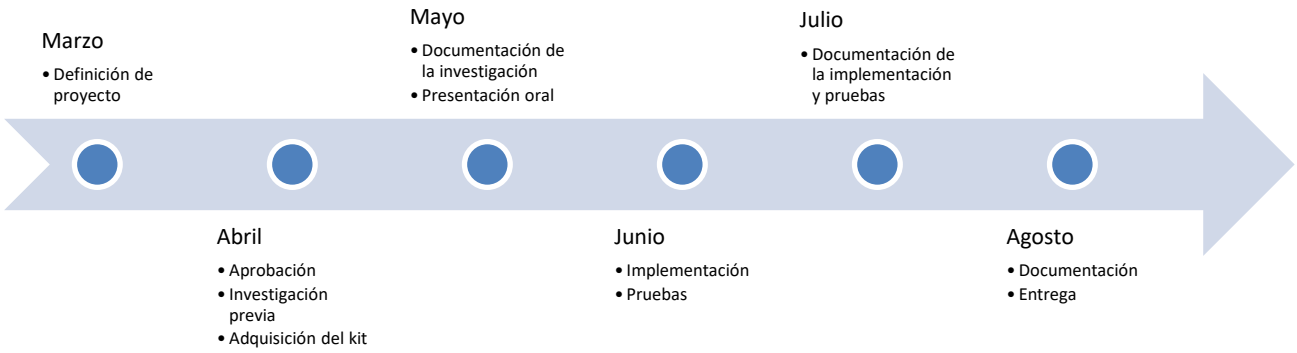
A continuación, se describe la implementación final y sus características.

9.2. Cronograma

Las fechas relevantes fueron las siguientes:

- Hito 1 (inicio): 12 de marzo
- Hito 2 (aprobación del tema): 9 de abril
- Hito 3 (presentación oral): 30 mayo
- Hito 4 (entrega): 8 de agosto
- Hito 5 (defensa): a definir

Se utilizó el tiempo de acuerdo al siguiente cronograma, siendo todos los puntos parte del camino crítico:



Entre el primer y segundo hito se concretaron reuniones con posibles tutores y se evaluaron distintos posibles proyectos a desarrollar. Se concretó el hito 2 cuando nos fue asignado Álvaro Sánchez como tutor, y el tema a desarrollar fue aprobado por el coordinador de la carrera (Guillermo Langwagen).

Entre el segundo y tercer hito se investigó la temática y tecnologías posibles para la implementación de una solución. Se adquirió *hardware* en base a esta investigación para poder

comenzar el desarrollo. Parte del tiempo se utilizó para la preparación de la presentación, y así como también para iniciar la documentación.

La etapa entre el tercer y cuarto hito fue la que insumió más tiempo. En una primera instancia se invirtió tiempo en la documentación de la investigación. A continuación, se realizó la implementación, y finalmente se documentó el resultado obtenido y los detalles de la implementación.

Entre la cuarta y quinta etapa se trabajará en la preparación de la defensa.

Es de relevancia mencionar que el tiempo entre la aprobación (hito 2) y la entrega (hito 4) fue de 4 meses, lo cual limitó el alcance y no permitió implementar todas las oportunidades de mejora identificadas.

9.3.Herramientas de trabajo

Todo el *software* utilizado para el desarrollo de código fuente de este proyecto se apoya en distintas librerías de fabricantes y/o *software* libre disponible públicamente.

A continuación, se detallan las herramientas más relevantes utilizadas en el proyecto.

9.3.1. Arduino IDE

Se trata del entorno oficial de desarrollo de código para Arduino [70], basado en software libre, el cual permite desarrollar programas (denominados *sketch*) en el lenguaje propio de Arduino (basado en C y C++).

Tiene la capacidad de compilar el código, subirlo a la placa, y monitorear lo que sucede dentro de la misma en tiempo real. Esto permite un *debugging* y *troubleshooting* eficaz.

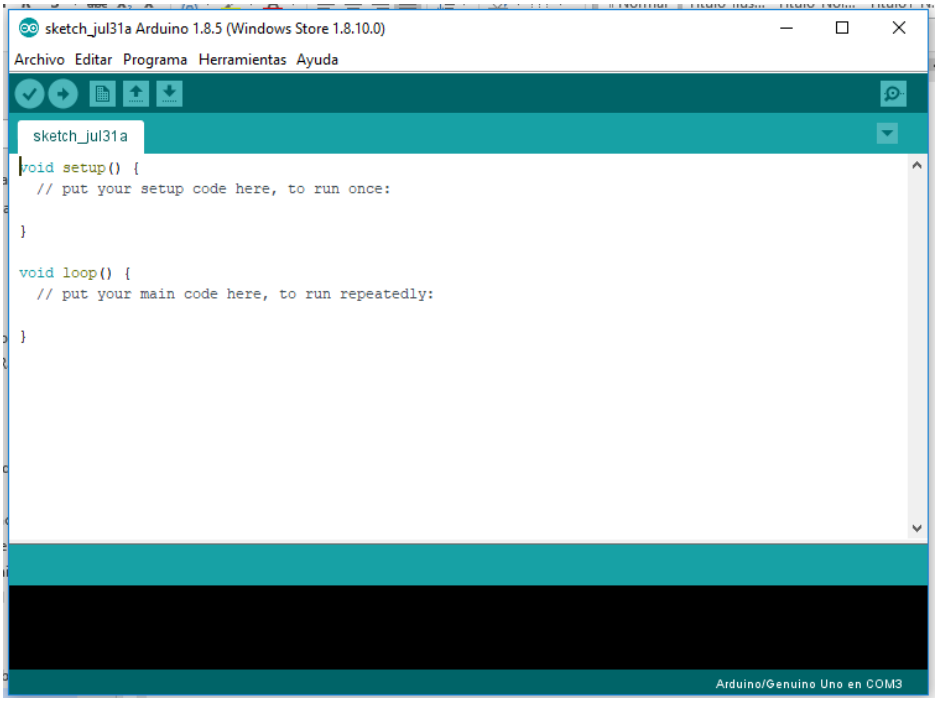


Ilustración 34: Arduino IDE

9.3.2. Google Drive

Se utilizó la plataforma Google Drive para redactar y almacenar un gran porcentaje de este documento y otros de manera colaborativa. Una vez avanzado el documento, se migró a Microsoft Word para poder utilizar el *template* con el formato indicado para este documento.

Adicionalmente se utilizó la plataforma para el almacenamiento de la documentación utilizada, y como plataforma de colaboración, ya que permite el acceso y modificación de los distintos documentos y archivos de manera simultánea por distintos actores, así como el versionado, comentarios y mensajería, entre otras características.

Adicionalmente, la plataforma permite exportar los documentos a distintos formatos (incluidos formatos *open source*, PDF, etc.), y no tiene costo.

9.4.Nodo

El nodo fue ensamblado en base a los elementos ya detallados.

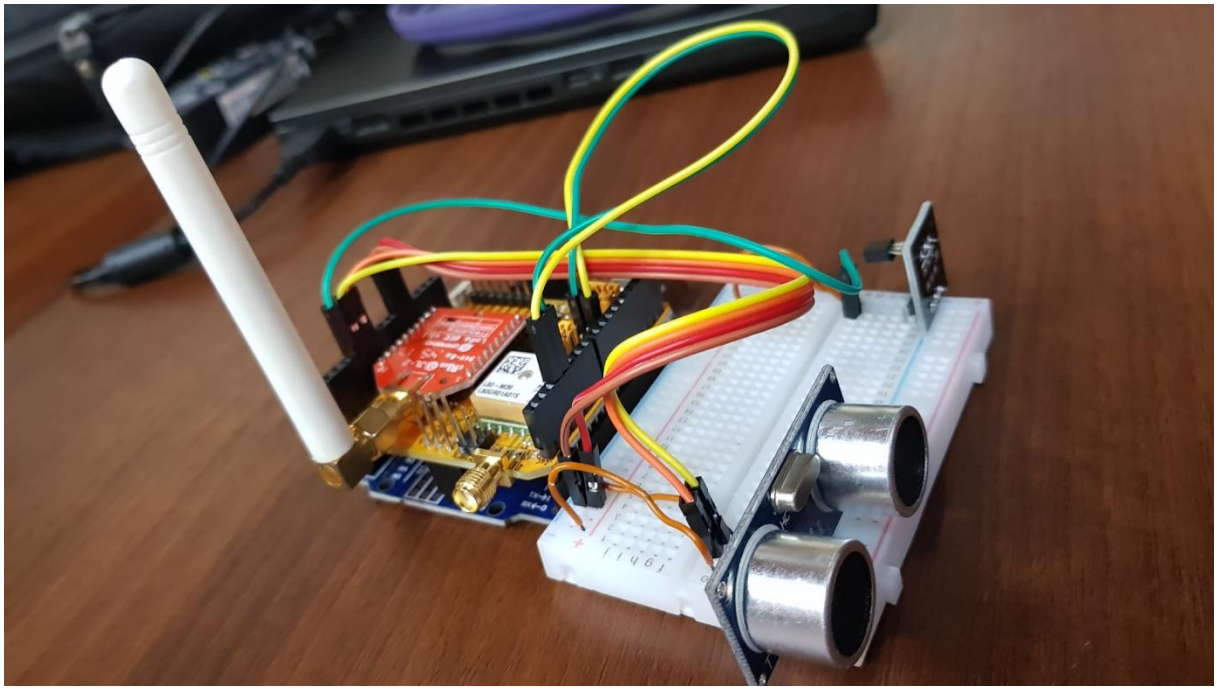


Ilustración 35: Nodo completo

La integración de los distintos periféricos requirió de la incorporación de distintas librerías (las cuales se mencionan en la sección 9.6).

La librería utilizada para el manejo del módulo LoRa (RF95) requirió de un estudio en profundidad de sus funciones y de la arquitectura del SX1276. Entre las funciones de esta librería se incluyen las de envío y recepción de paquetes LoRa. Esta librería permite el envío entre 1 y 50 *bytes* en cada paquete, como *payload*. Esto es una limitación de la librería, dado que el *hardware* soporta hasta 255 *bytes* en cada paquete. Se definió una trama para los mensajes de envío y otra para los de recepción. La estructura de tramas es descrita en la sección 9.4.3.

Siguiendo la filosofía de IoT, se intentó enviar sólo la información mínima y necesaria para minimizar el uso de recursos.

La recepción de GPS presentó algunas dificultades las cuales son descritas en la sección 9.7.1.

El tipo de registro del nodo en la red sigue un concepto análogo al ABP (*Activation By Personalization*) de LoRaWAN, debido a que el identificador del nodo es configurado de manera manual. El servidor de aplicación tomará el identificador de nodo del mensaje enviado.

Se definió un tiempo entre envíos de 5 minutos. Este valor puede ser fácilmente ajustado desde el código desarrollado.

La secuencia ejecutada en el nodo responde al siguiente diagrama de flujo:

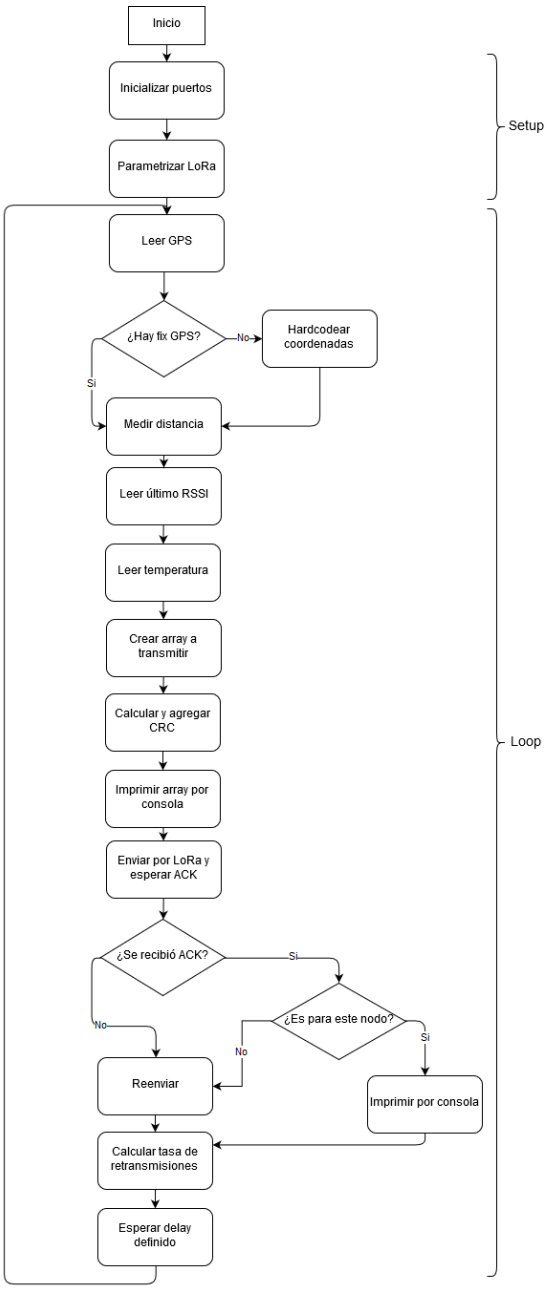


Ilustración 36: diagrama de flujo del nodo

Como se observa en el diagrama, el sistema reenvía el paquete una única vez en caso de no recibir el reconocimiento por parte del *gateway*. Esto es una decisión de diseño.

Tanto el tiempo de espera para el reenvío como el tiempo entre mensajes es configurable.

9.4.1. Funcionamiento del sensor de nivel de llenado

Se definió la utilización de un sensor de distancia HC-SR04 para determinar el nivel de llenado.

El sensor genera ondas de ultrasonido de 40 kHz las cuales se reflejan en el objeto a ser detectado. Una vez reflejadas el sensor detecta el eco y calcula el tiempo que demora en retornar. Mediante el tiempo permite conocer la distancia, debido a que la velocidad del sonido es una constante (340 m/s en el aire).

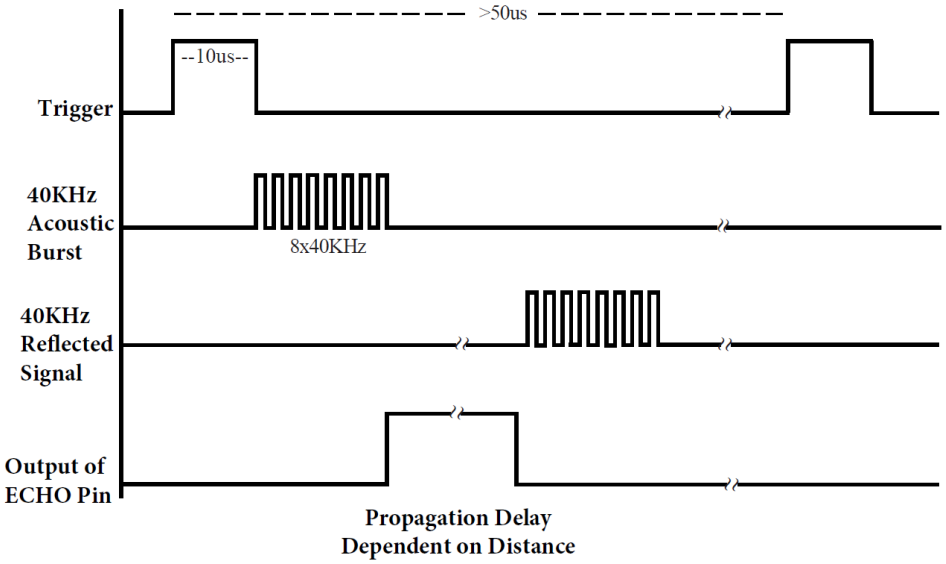


Ilustración 37: secuencia de eventos [60]

La diferencia de tiempo es medida por el Arduino, y el cálculo de distancia también es realizado por éste.

El cálculo para implementar de acuerdo al fabricante [59] es:

$$d = \frac{t}{58}$$

Donde *d* es la distancia en cm, y *t* es la diferencia de tiempo medida en µs.

Este sensor utiliza 4 cables (2 de alimentación, envío del pulso y detección del eco). El envío y detección requieren cada uno 1 pin digital en el módulo Arduino.

9.4.2. Funcionamiento del sensor de temperatura

El sensor 18B20 es un sensor digital que utiliza el protocolo 1-Wire [71], por lo que fue necesario incorporar la librería correspondiente para su comunicación con el módulo Arduino.

Debido a que utiliza un bus 1-Wire, requiere sólo de 3 cables (tierra, señal y alimentación). El pin de señal requiere un puerto digital en el módulo Arduino.

9.4.3. Mensaje LoRa

Existen 2 mensajes LoRa: un envío desde el nodo hacia el *gateway*, y un reconocimiento de recepción desde el *gateway* hacia el nodo.

El paquete LoRa que se envía al *gateway* tiene 29 *bytes* de *payload*, estructurados de acuerdo al siguiente detalle:

Byte	Campo	Referencia	Rango Valores Esperados	Descripción
0	ID GW	ID	0-255	ID del *gateway* al que envía
1	ID Nodo	ID	0-255	ID del nodo
2	Fecha	Dia	0-31	
3		Mes	0-12	
4		Año	00-255	
5	Tiempo	Hora	0-24	
6		Minutos	0-59	
7		Segundos	0-59	
8	Ubicación	Latitud	Decimales	En formato decimal, sin coma, con signo. Ej.: -34.903905, -56.190587 se entrega como: -34903905, -56190587
9				
10				
11				
12		Longitud	Decimales	
13				
14				
15				
16	Temperatura	Celsius	0-125	
17	Nivel de llenado	mm	0-4000	
18				
19	Señal	-dBm	0-150	RSSI de último mensaje recibido por el nodo en -dBm
20	Tasa Error	%	0-100	Porcentaje de mensajes que no recibieron ACK y fueron retransmitidos
21	Tiempo retransmisión	x 15 milisegundos	0-255	Tiempo de espera del ACK antes de retransmitir
22	Frecuencia de envío	x 15 segundos	0-255	Tiempo que se espera entre los envíos
23	Potencia nodo	dBm	0-20	Potencia de transmisión
24	SF Lora	numérico	7-12	*Spreading Factor*
25	CR Lora	numérico	5-8	*Code Rate*
26	BW Lora	Hz x 62500	1-4	*Bandwidth*
27	CRC			
28				

Estructura del mensaje de envío del nodo

Como se puede observar, se define que cada *gateway* soportará un máximo de 255 nodos, y la solución completa soporta un máximo de 255 *gateways*. Esto significa que, en un proyecto dado, pueden coexistir un máximo de 65.025 nodos, lo cual es suficiente para el escenario de la ciudad de Montevideo.

Por otra parte, se define el mensaje de reconocimiento de recepción (ACK o *acknowledge*). Este mensaje de 14 *bytes* es enviado por el *gateway* al nodo cuando recibe el mensaje correctamente. El objetivo de este mensaje es simplemente comunicar al nodo que su mensaje fue recibido correctamente. Para indicar a qué nodo se le envía el reconocimiento, el mensaje debe contar con un ID del *gateway* y nodo. Se describe a continuación su estructura, que consiste en 14 *bytes*.

Byte	Campo	Referencia	Rango Valores Esperados	Descripción
0	ID GW	ID	0-255	ID del *gateway* al que envía
1	ID Nodo	ID	0-255	ID del nodo
2	Reservado para uso futuro			
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				

La estructura de este mensaje podría ser modificada a futuro para ser aprovechada para enviar parámetros hacia el nodo. A modo de ejemplo, podría ser posible modificar el tiempo entre mensajes del nodo, enviando este parámetro. Esto no fue implementado por cuestiones de plazo.

La comunicación entre nodo y *gateway* tiene un funcionamiento similar a un dispositivo clase A en LoRaWAN.

9.4.4. Conexiones

El módulo Arduino utiliza múltiples puertos de entrada y salida para vincularse a los distintos periféricos. Si bien se puede observar que se utilizan algunos puertos analógicos, estos son utilizados en modalidad digital.

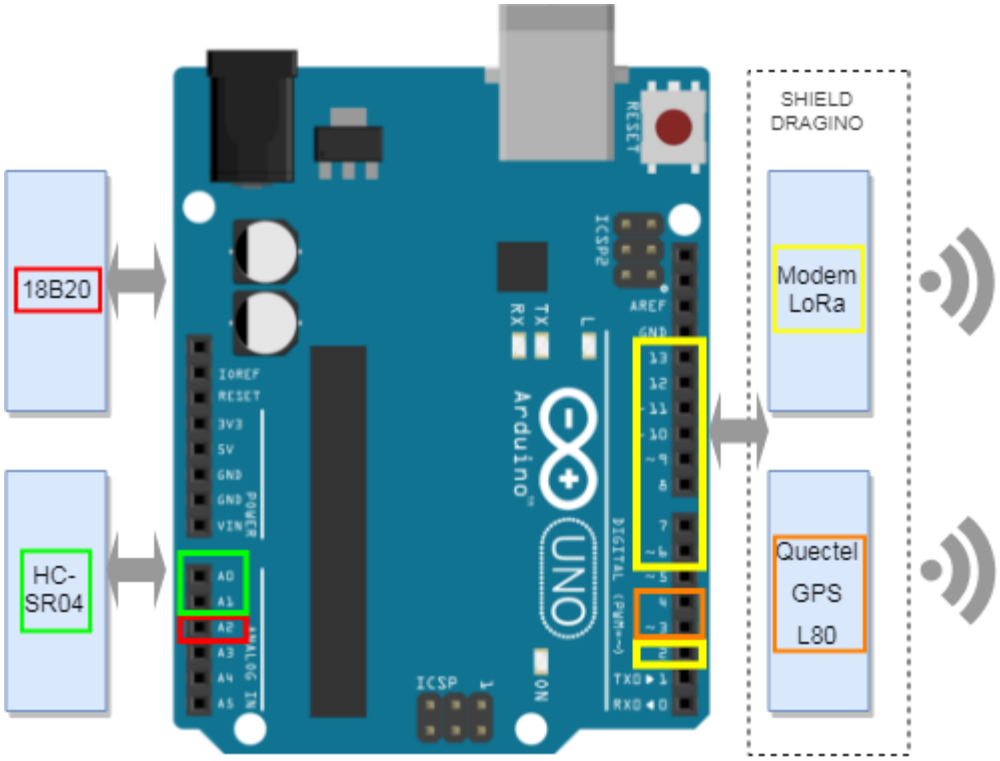


Ilustración 38: Diagrama de conexiones del nodo

Las conexiones responden al siguiente detalle:

ARDUINO	LoRa/GPS shield	18B20	HC-SR04
D13	LORA SCK		
D12	LORA MISO		
D11	LORA MOSI		
D10	LORA NSS		
D9	LORA RESET		
D8	LORA DIO5		
D7	LORA DIO2		
D6	LORA DIO01		
D4	GPS TX		
D3	GPS RX		
D2	LORA DIO0		
D1			
D0			
A2		DATA	
A1			ECHO
A0			TRIGGER
5V	VCC	VCC	VCC
GND	GND	GND	GND

9.5. Gateway

9.5.1. Introducción

El ATmega328P se encuentra vinculado al Atheros AR9331 mediante una conexión UART y una SPI a nivel de hardware, lo cual no es descrito en profundidad por el fabricante. Esto permite el intercambio de información entre ambos. Utilizando la librería *bridge* [72] de Arduino es posible solicitar al OpenWRT la ejecución de comandos de consola Linux, lo cual incluye acceso al sistema de archivos, entre otras posibilidades.

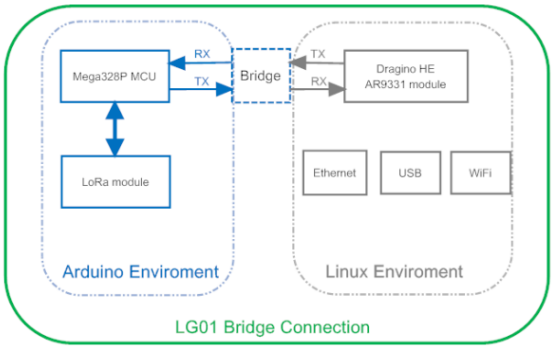


Ilustración 39: Diagrama de bloques del gateway [66, p. 35]

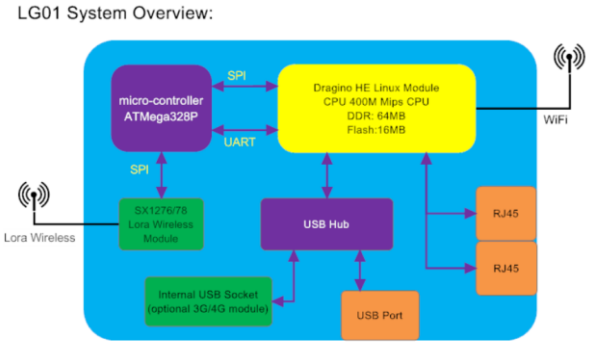


Ilustración 40:Diagrama de bloques del gateway [66, p. 9]

9.5.2. Configuración

Es necesario configurar todos los parámetros habituales en la red IP (dirección IP, máscara, DNS, etc.) para el acceso hacia el servidor de aplicación en Internet, así como los parámetros LoRa (frecuencia, *code rate*, *spreading factor*, etc.).

La configuración de los parámetros IP se realiza mediante la interface web del sistema operativo.

La configuración la interface LoRa se realiza con la IDE Arduino, debido a que tiene un ATmega328P que es visto como un Arduino UNO.

9.5.3. Proceso de recepción y envío de información

El *gateway* recibe paquetes en la red LoRa y los envía hacia el servidor de aplicación en la red IP (*Web Service* que utiliza formato JSON).

El mecanismo para lograr esto se puede dividir en 3 grandes etapas, en el siguiente orden:

1. Recepción en el “hemisferio” LoRa (Arduino)
2. Almacenamiento en el sistema de archivos (OpenWRT)
3. Envío hacia el *Web Service*

A continuación, se detallan los distintos procesos de estas etapas.

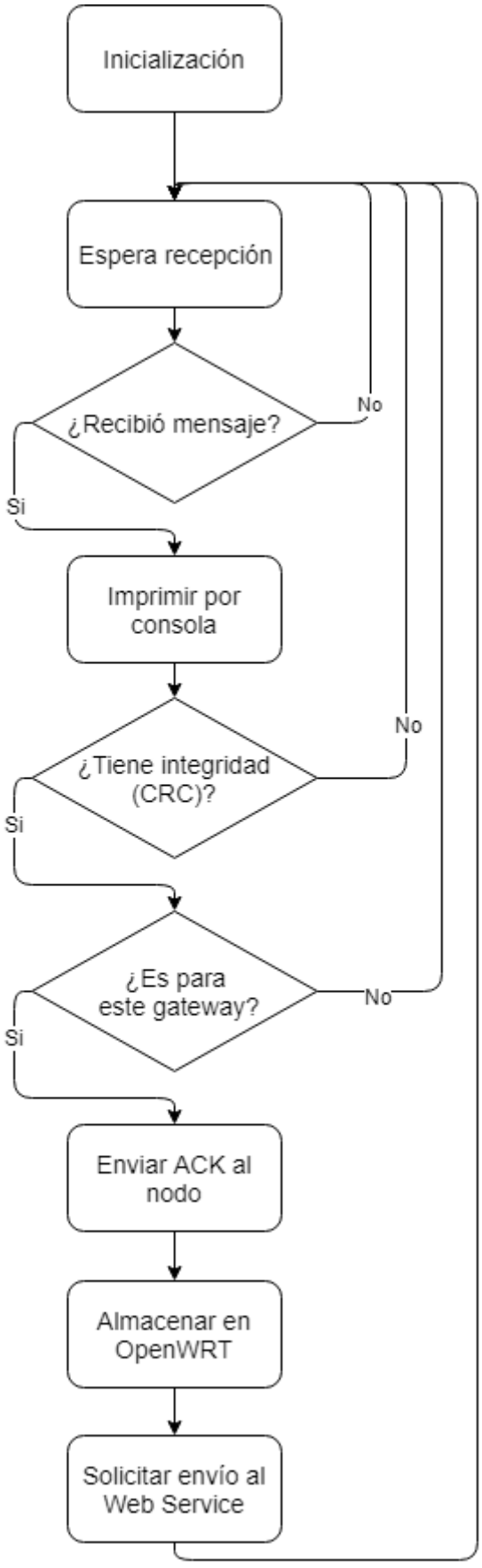


Ilustración 41: Diagrama de flujo de la recepción del *gateway* (Arduino)

Una vez recibido el mensaje en el “hemisferio” LoRa, éste lo almacena en el sistema de archivos del OpenWRT y se le solicita iniciar la secuencia de envío hacia el *Web Service*:

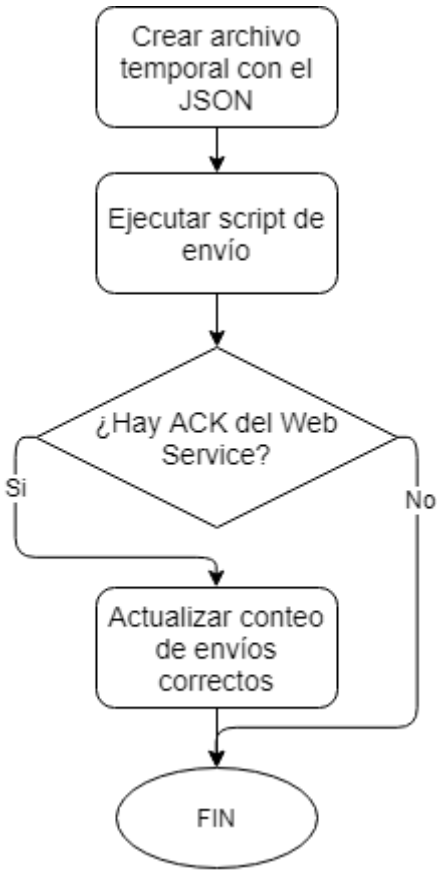


Ilustración 42: diagrama de flujo de la transmisión del *gateway* (OpenWRT)

En caso de fallo en la transmisión al *Web Service*, el archivo es sobre-escrito por el siguiente mensaje, partiendo de la premisa que es deseable tener la última información disponible y no tiene utilidad el mensaje anterior ya que se considera “obsoleto”.

9.5.4. Estructura del mensaje JSON

Se definió en conjunto un mensaje en formato JSON, cuyas características técnicas son descritas en el ANEXO I.

En una etapa inicial se definieron 9 campos en conjunto con el equipo de proyecto de Licenciatura en Sistemas, debido a que no teníamos acordado aún el contenido definitivo. Esto nos dio un margen de flexibilidad.

Una vez avanzado el proyecto, se definió el contenido y formato de la información en estos campos. Los campos del formato JSON son completados de acuerdo al siguiente detalle:

Campo JSON	Datos	Formato	Ejemplo
C1	ID Nodo	ID Nodo; ID gateway	1;236
	ID gateway		
C2	Fecha	Día/Mes/Año	8/8/2018
C3	Hora	HH:MM:SS	23:55:17
C4	Latitud		-349032800
C5	Longitud		-561881600
C6	Temperatura	° C	55
C7	Nivel de llenado	distancia en mm	255
C8	RSSI de última recepción del nodo	RSSI nodo:PTx nodo:RSSI gateway	Recepción del nodo con -45 dBm, transmisión del nodo con 13 dBm, recepción del gateway con -39 dBm: -45:13:39
	RSSI de última recepción del gateway		
	Potencia Tx nodo		
C9	Tasa de errores	tasa de errores:tiempo retransmisión:frecuencia de envío:SF:CR:BW	Tasa de 15%, tiempo de 10x 15 ms, frecuencia de 3x 15s, SF 7, CR 4/8, BW 62500x 2 15-10-3-7-8-2
	Tiempo retransmisión		
	Frecuencia de envío		
	Spreading Factor		
	Code Rate		
	Bandwidth		

9.6.Librerías

Se utilizaron múltiples librerías durante el desarrollo del código. Estas se describen a continuación.

9.6.1. TinyGPS

Esta librería [73] tiene como objetivo el procesamiento del *stream* de datos proveniente del módulo GPS, en formato NMEA 0183 [63]. La librería realiza la interpretación de este formato y permite obtener fácilmente datos como latitud, longitud, velocidad, distancia y altura, entre otros en distintos formatos.

9.6.2. Radiohead

Se utiliza la librería Radiohead [74]. La librería está destinada al manejo de paquetes por radio en dispositivos con microprocesadores embebidos. En particular RF95 provee el *driver* adecuado para el circuito integrado utilizado para permitir la transmisión en redes LoRa.

9.6.3. OneWire

Esta librería [75] se utiliza para poder interpretar correctamente el protocolo 1-Wire [71] utilizado por el sensor de temperatura 18B20.

9.6.4. Dallas Temperature

Esta librería [76] se utiliza para obtener y manipular la información entregada por el sensor de temperatura y permite funcionalidades tales como la conversión entre Celsius y Fahrenheit.

9.6.5. Bridge

La librería Bridge [72] resuelve la conexión entre el ATmega328P y el Atheros AR9331 dentro del *gateway*. Esta librería permite la ejecución de comandos de Linux, los cuales pueden ser iniciados desde el ATmega328P, entre otras funcionalidades.

9.7.Dificultades presentadas

9.7.1. Recepción del GPS

Durante la implementación se detectó que el GPS demoraba en el orden de 90 segundos en obtener "*fix"* (recepción de coordenadas y demás parámetros) a la intemperie. Adicionalmente, esto sucedía estando a cielo abierto y en condiciones óptimas, y en ningún caso se logró *fix* en ambientes interiores, bajo techo. Esto no era compatible con lo descrito en la hoja de datos del Quectel L80. Analizando el tema en mayor profundidad, se halló que el *shield* Dragino no cumple con las características requeridas por el módulo L80 [77] (en particular, la distancia mínima de 10 mm requerida entre el circuito integrado L80 y otros componentes).

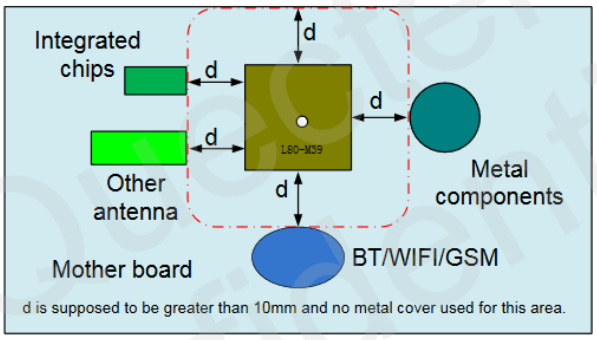


Ilustración 43: instalación de L80 [77, p. 30]

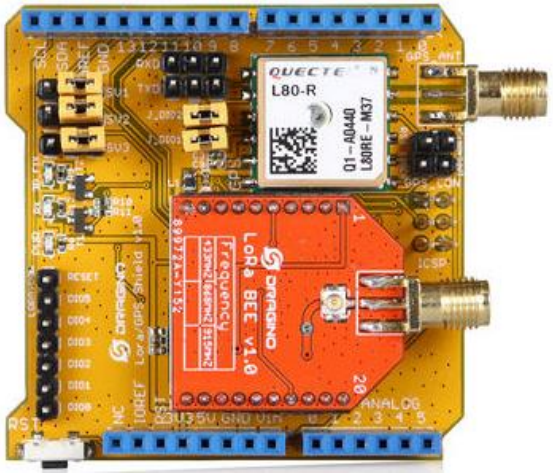


Ilustración 44: shield LoRa/GPS [65]

Adicionalmente, Quectel recomienda que la placa en la cual se instale el módulo tenga dimensiones mínimas de 80 x 40 mm.

Como se puede observar, ninguno de estos dos requerimientos se cumple en el *shield* Dragino [65]. Las dimensiones del *shield* son 60 mm x 53 mm, y la distancia mínima es del orden de 1 mm entre el L80 y otros componentes cercanos.

Si bien no es posible concluir que el mal comportamiento del GPS se deba a estos motivos, tampoco es posible garantizar que se comporte como describe el fabricante ya que no cumple con los requerimientos.

9.7.2. Limitaciones de la librería bridge

El envío de la información desde el hemisferio LoRa hacia el hemisferio IP dentro del *gateway* presentó dificultades en varias oportunidades debido a las limitaciones de memoria del ATmega328P.

Inicialmente se intentó enviar la totalidad de la información ya estructurada en formato JSON junto con los encabezados y demás parámetros necesarios para lograr la transmisión hacia el servidor de aplicación. Sin embargo, debido a la cantidad de memoria que esto ocupa (en el entorno de 300 *bytes*), no funcionó correctamente.

Es por este motivo que se resolvió enviar la información particionada desde el hemisferio LoRa hacia el OpenWRT. Esto se logró almacenando de a segmentos la información en un archivo (dentro del OpenWRT) y luego ejecutando un comando de consola del lado Linux que ejecuta el método ("*method*") POST vía HTTP hacia el *Web Service*.

9.8. Cálculos de dimensionamiento

A modo de ejemplo, se detalla un posible dimensionamiento para un escenario como el de Montevideo (13000 contenedores). Se asumen fijos los parámetros LoRa (SF, BW y CR) detallados en la tabla. Los cálculos a continuación son para un *gateway* con 255 nodos, y son extrapolables a un escenario de múltiples *gateways* debido a que utilizarían distintos canales y/o parámetros LoRa.

Suponiendo el peor escenario posible (255 nodos por cada *gateway*), surgen los siguientes cálculos:

	Dimensionamiento del Sistema	Valor	Unidad	Variable
1	**Configuración Nodo**			
1.1	Tamaño del mensaje Nodo	29	Bytes	M1
1.2	Tamaño del mensaje ACK	14	Bytes	M2
1.3	Frecuencia de envió	1	Cada 5 minutos	F
1.4	*Spreading Factor*	10	-	SF
1.5	*Bandwidth*	125	kHz	BW
1.6	*Code Rate*	4/6	-	CR
2	**Tiempo en el aire los mensajes**			
2.1	Tiempo en el aire de M1 (utilizando SF, BW y CR)	412	milisegundos	Tm1
2.2	Tiempo en el aire de M2 (utilizando SF, BW y CR)	199	milisegundos	Tm2
3	**Mensajes**			
3.1	Cantidad de mensajes por Nodo (1 cada 5 minutos)	288	Mensajes/día	TM
3.2	Total de mensajes por canal de GW: (TM x N)	73440	Mensajes /día	TMS
4	**Uso del canal**			
4.1	Cantidad de Nodos por Canal de Radio en cada GW	255	-	N
4.2	Ocupación del Canal, mensaje + ACK: (Tm1+Tm2)xTMS	44872	Segundos/día	Oc
4.3	Disponibilidad del Radio (GW siempre recibiendo)	86400	Segundos	D
4.4	% Ocupación del canal de radio: (Oc/D)x100	**52%**		
5	**IP: Base de Datos y Enlaces**			
5.1	Cantidad máxima de nodos (13000 contenedores-MVD)	13000	-	Nmax
5.2	Cantidad de mensajes diarios, registros en BD: TM x Nmax	374400	Mensajes/día	TMmax
5.3	Volumen de datos generados por día: TMmax X M1	109	Mbytes/día	
5.4	Volumen de datos generados por año	40	GB	
5.5	Enlace de datos *Web Service* (sin encabezados) 29 bytes	10	Mbps	
5.6	Enlace datos: JSON + Headers (IP, TCP, HTTP) 300 bytes	**102**	Mbps	
5.7	Enlace de datos IP GW, por cada nodo	8	Kbps / nodo	
5.8	Enlace de datos para un GW con 255 Nodos	**2**	Mbps	

A continuación, se detallan los cálculos:

- Los datos en (1) surgen como datos del escenario planteado.
- Los datos en (2) surgen a partir de la calculadora LoRa y dependen de los valores de (1).
- Los datos en (3) surgen de (1) por la frecuencia de envío de mensajes de cada nodo.
- Los datos en (4) surgen:
    - (4.1) es el escenario planteado, donde hay un máximo de 255 nodos por *gateway*.
    - (4.2) es el tiempo que se ocupa el canal con todos los mensajes de todos los nodos y sus ACK correspondientes, en un período de 24 horas. Se asume que no hay retransmisiones.
    - (4.3) es el tiempo de disponibilidad de un *gateway* en un período de 24 horas. Se asume que opera las 24 horas.
    - (4.4) es la relación entre 4.2 y 4.3. Muestra que el canal está ocupado aproximadamente la mitad del tiempo para este escenario.
- (5) muestra los cálculos de la red IP:
    - (5.1) es el escenario de Montevideo
    - (5.2), (5.3) y (5.4) son datos útiles a los efectos de dimensionar una base de datos y el almacenamiento.
    - (5.5) refiere al *bitrate* nominal generado por todo el sistema en la red LoRa, sin sus encabezados IP, TCP ni HTTP.
    - (5.6) agrega al (5.5) los encabezados IP, TCP y HTTP a los efectos de enviar al *Web Service*. Este valor es la capacidad del enlace necesario para el servidor de aplicación.
    - (5.7) representa la capacidad en la red IP que se requiere por cada nodo.
    - (5.8) representa la capacidad del enlace requerido para un *gateway* con 255 nodos.

10. Pruebas y ensayos

10.1. Pruebas de sensores

10.1.1. Sensor de nivel de llenado

Se hicieron pruebas de campo del sensor de distancia sobre varios contenedores en la vía pública, y fue posible concluir que este tipo de sensor es una alternativa válida.

Las pruebas consistieron en medir la distancia desde el sensor hasta el contenido. Estas medidas fueron obtenidas con el propio sensor y con una cinta métrica para poder verificar la medida. El sensor fue ubicado tanto en la parte superior (orientado hacia abajo), como en una esquina (orientado en diagonal).

Estas pruebas fueron realizadas únicamente sobre contenedores de tipo B debido al tipo de abertura que facilita el acceso.

Se hicieron pruebas sobre 4 contenedores, y las diferencias son despreciables (del orden de 5 cm) a los efectos de determinar el nivel de llenado, ya que la precisión requerida es baja (se entiende como suficiente una precisión de 15 cm).

Ilustración 45: sensor en ubicación superior

Ilustración 46: sensor en ubicación diagonal

Ilustración 47: medida con cinta métrica

A partir de las pruebas realizadas, es posible concluir que las medidas obtenidas por el sensor son una medida representativa del nivel de llenado.

Contenedor	Distancia sensor (cm)	Distancia cinta métrica (cm)	Diferencia (cm)
1	74,4	72,5	1,9
2	28,7	30	-1,3
3	104,8	102,5	2,3
4	51,9	46,5	-5,4

A los efectos de comprender estas medidas, es necesario tener en cuenta la distribución del contenido del contenedor y la forma en la que mide el sensor. No es posible saber de manera intuitiva ni certera contra qué objeto refleja la onda, ya que el sensor emite en un cono de 15° de apertura. Por otra parte, la superficie a medir es totalmente irregular.

Sin embargo, es necesario considerar el ambiente de los contenedores (humedad, lluvia, polvo, vandalismo, etc.) y tomar las precauciones necesarias para adecuarlo.

10.1.2. Sensor de temperatura

Se realizaron pruebas sobre el sensor de temperatura utilizando un secador de pelo para simular una fuente de calor.

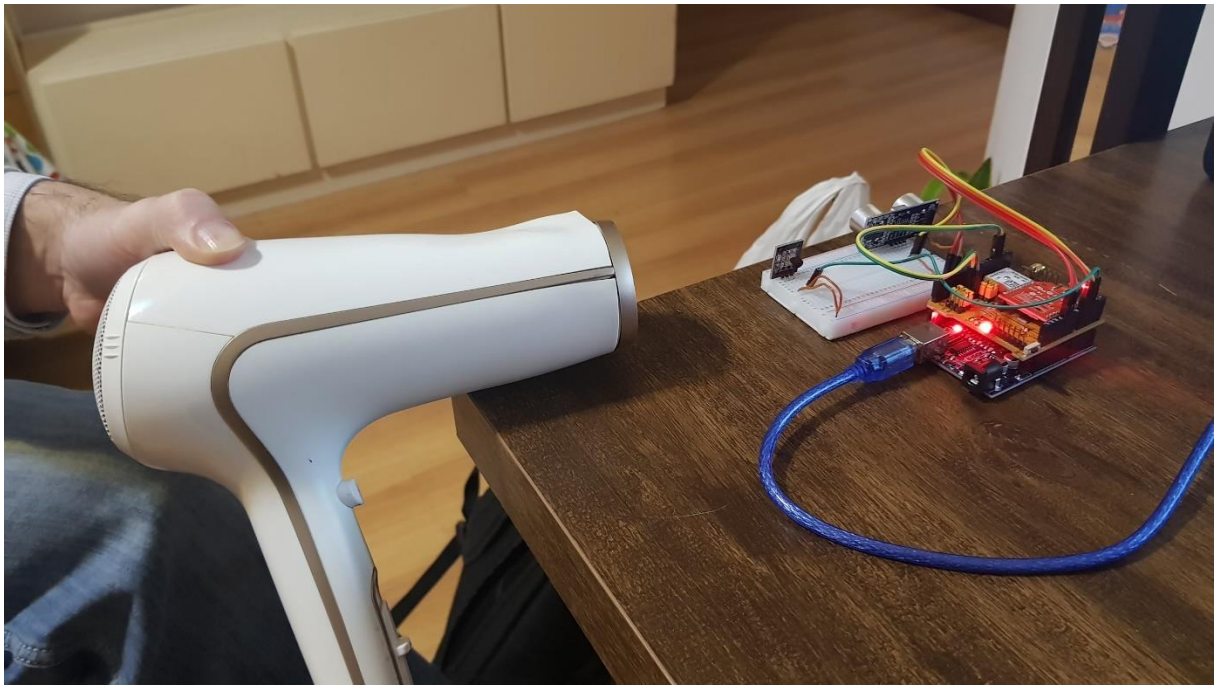


Ilustración 48: prueba de temperatura

Al apuntar el secador de pelo directamente hacia el sensor y tomando medidas cada 3 segundos, éste muestra el aumento rápido de temperatura:

Tiempo (s)	Temperatura (°C)
0	20
3	30
6	48
9	60
12	65
15	67

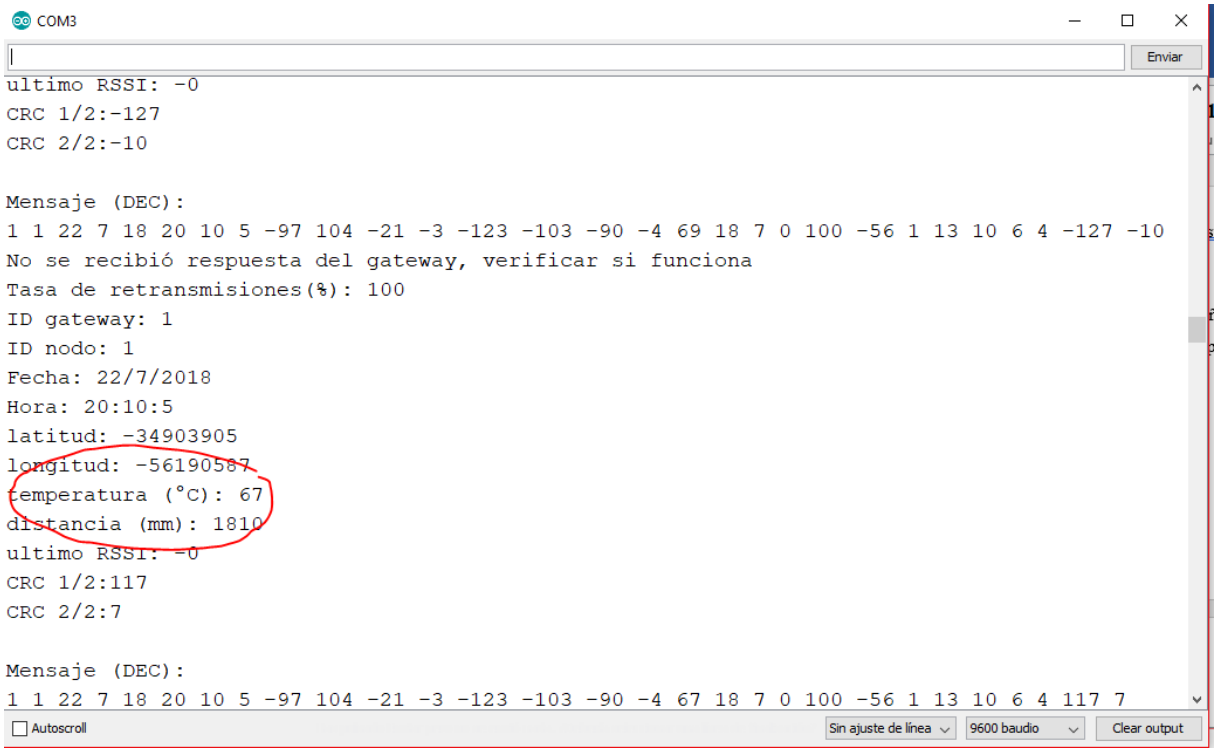


Ilustración 49: prueba de temperatura

No se intentó llegar a temperaturas superiores a 70 °C para evitar dañar los componentes plásticos.

10.2. Pruebas de LoRa

Se realizaron pruebas de radio, las cuales se detallan a continuación.

Se instaló el *gateway* en la azotea de un edificio (piso 11, a 60 metros sobre el nivel del mar) en una zona urbana densamente poblada, y se recorrió la zona realizando distintas pruebas.

Ilustración 50: ubicación del *gateway*

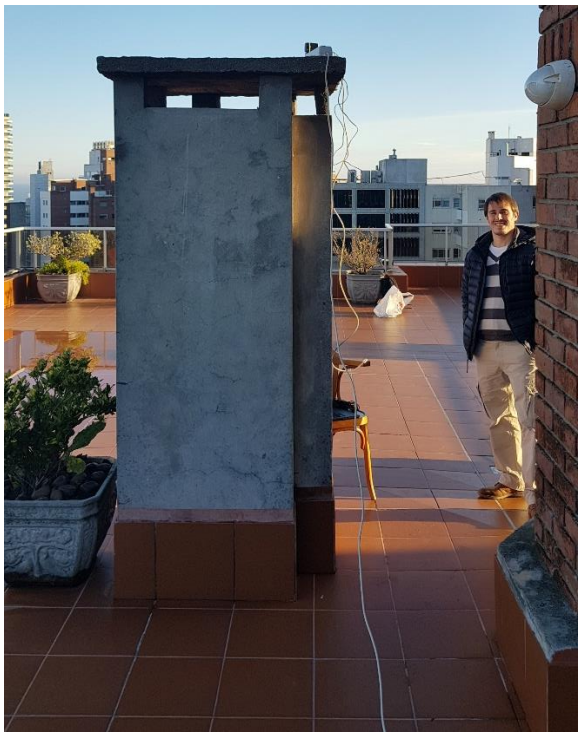


Ilustración 51: ubicación del *gateway*

Ilustración 52: zona del gateway

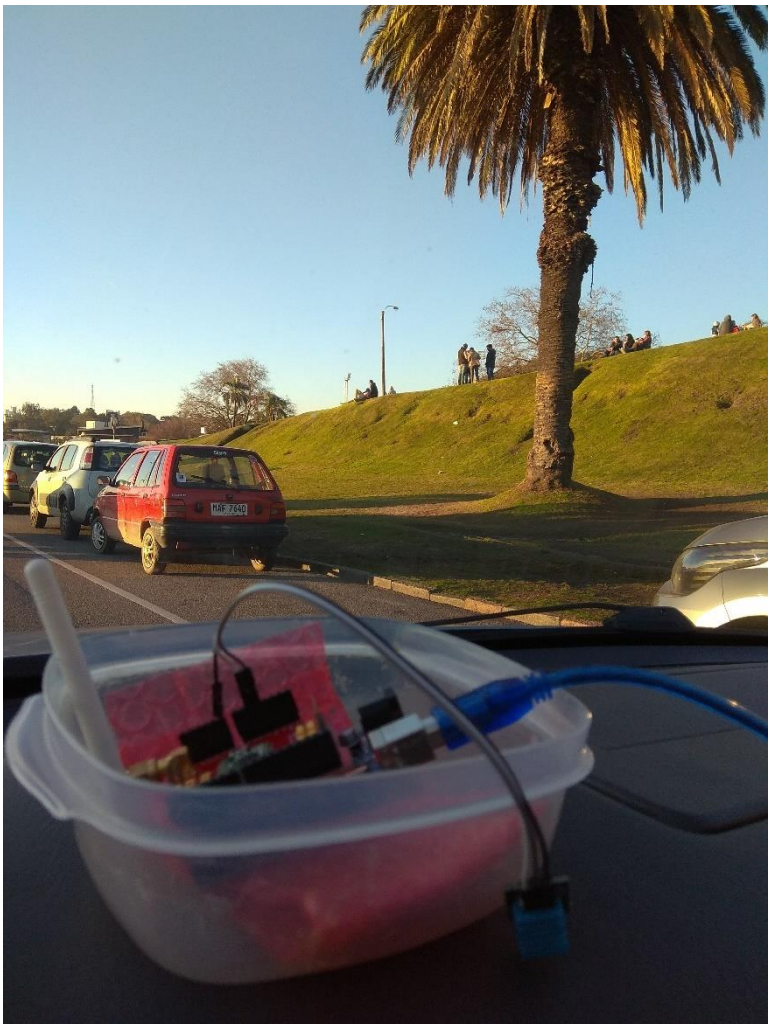


Ilustración 53: nodo en la recorrida de la zona

Se hicieron pruebas en los lugares indicados en el mapa con estos parámetros, tanto en el *gateway* como en el nodo:

- Code Rate: 4/6
- Spreading Factor: 10
- Bandwidth: 125 kHz
- Potencia: 20 dBm

Se seleccionaron estos parámetros ya que se trata de valores típicos e intermedios en redes LoRa. Adicionalmente, su tiempo en el aire es adecuado para el escenario calculado.

Se obtuvo la siguiente tabla:

Lugar	Distancia (m)	RSSI nodo (dBm)
A	781	-101
B	622	-100
C	577	-95
D	450	-100
E	330	-84
F	347	-98
G	170	-88
H	109	-83

Como se puede observar en la siguiente ilustración, se obtuvo un alcance máximo de 781 metros (punto A) en la zona de las pruebas, con un RSSI de -101 dBm en el nodo, sin línea de vista.

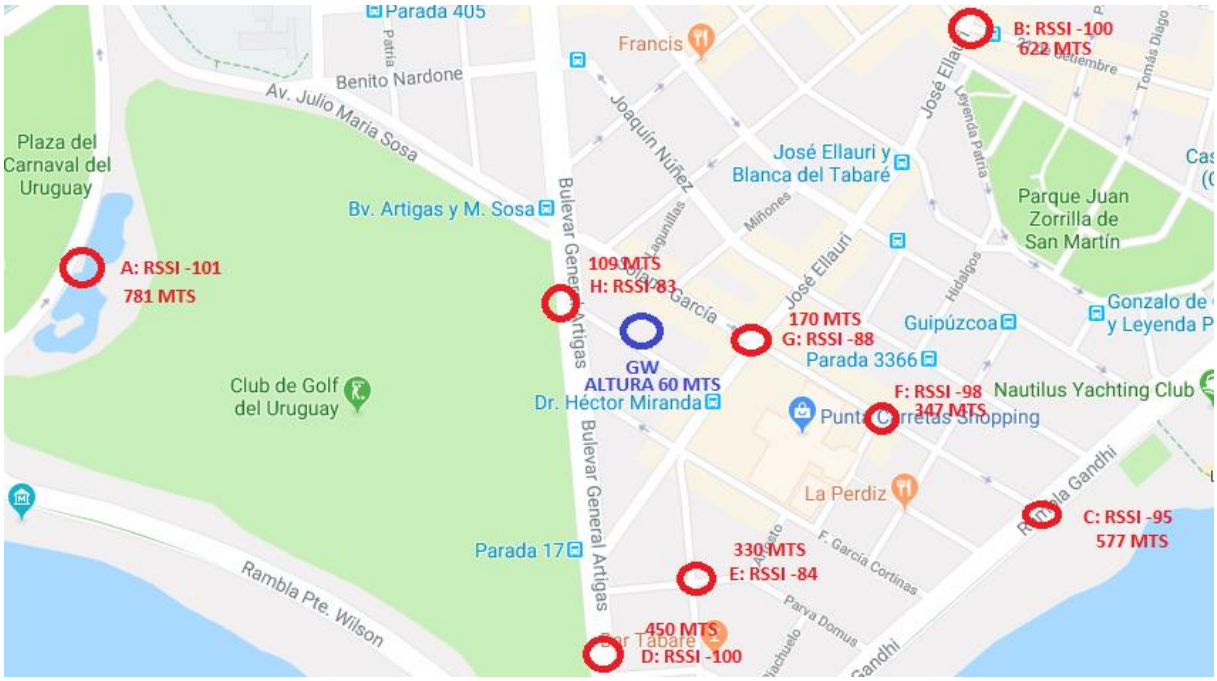


Ilustración 54: mapa de alcance

El radio obtenido (781 metros) cubriría aproximadamente 100 manzanas. Considerando 4 contenedores por manzana, esto sería un total de 400 nodos. En este escenario, sería necesario un *gateway* con 2 canales de radio, lo cual es totalmente viable dado que los *gateways* comerciales cuentan con al menos 8 radios, dado que es lo exigido por LoRaWAN.

Es relevante mencionar que un *gateway* comercial para producción utilizaría antenas de mayor ganancia, con lo cual este escenario sería mejorable.

10.3. Pruebas del sistema completo

A continuación, se presentan capturas de pantalla que muestran todo el proceso de recepción y envío, desde el nodo al *gateway*, y desde el *gateway* a la aplicación desarrollada por el equipo de proyecto de Licenciatura en Sistemas, que recibe correctamente los datos en el formato acordado:

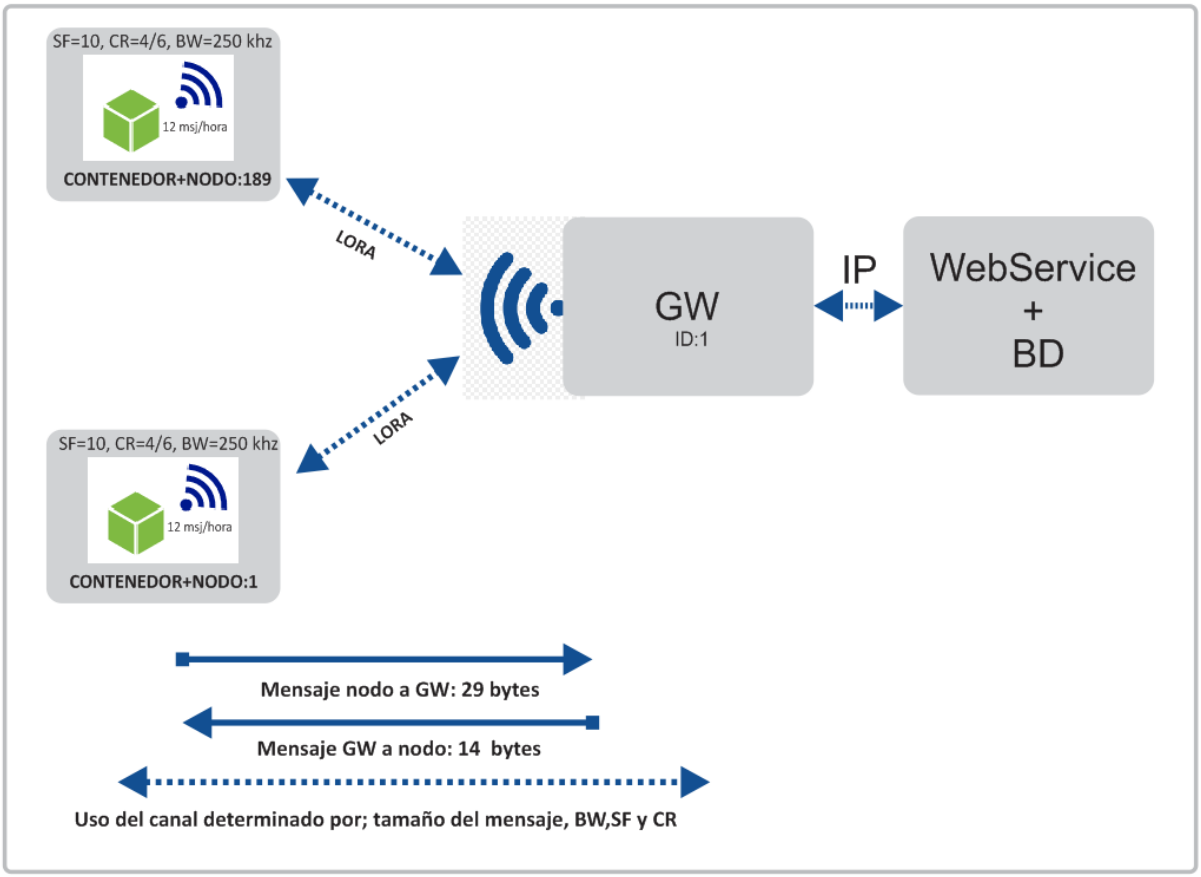


Ilustración 55: detalle de la maqueta

Se reciben los datos del nodo en el hemisferio LoRa del *gateway*:

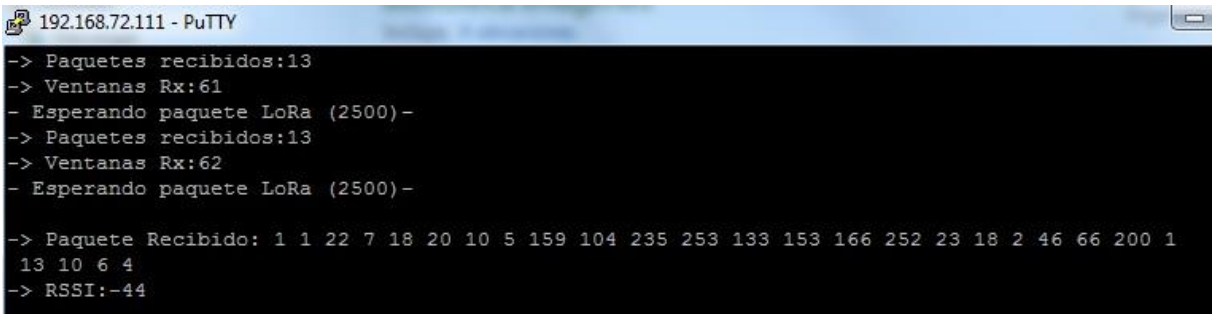


Ilustración 56: recepción del mensaje en el *gateway*

Luego, se envía hacia el hemisferio IP y se almacena en un archivo (gw.json) en formato JSON:

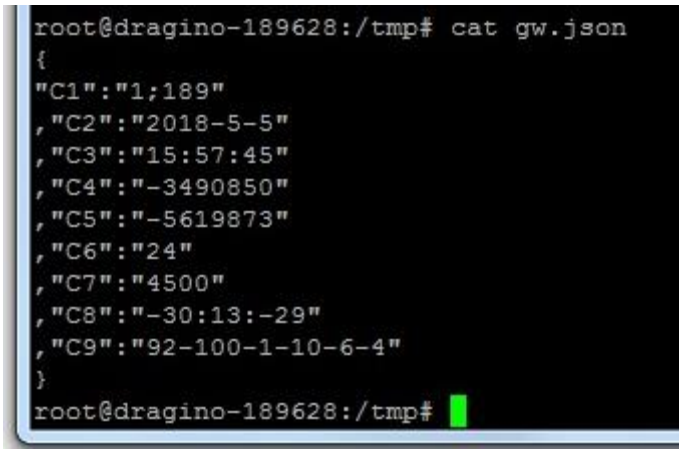


Ilustración 57: generación del mensaje en formato JSON

Luego, desde el Arduino se comanda la ejecución de un *script* que lo envía al *Web Service*. A continuación, se muestra el contenido del *script*.

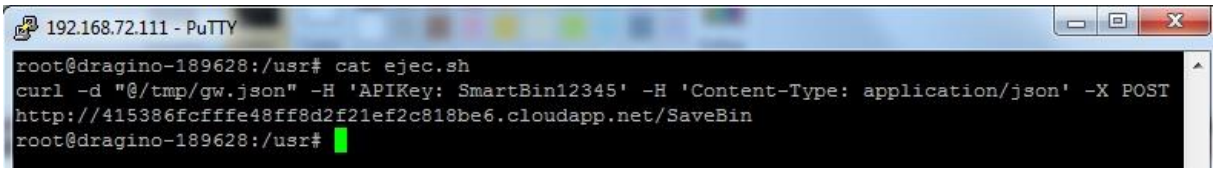


Ilustración 58: detalle del *script* de envío

Detalle de la recepción en el sistema del equipo de proyecto de Licenciatura en Sistemas:

ID	Fecha	Hora	Latitud	Longitud	Temperatura	Distancia	Señal	Info	
1;189	2018-5-5	15:57:45	-3490850	-5619873	24	4500	-30:13:-29	92-100-1-10-6-4	Seleccionar
1;189	2018-5-5	15:57:45	-3490850	-5619873	24	4500	-30:13:-29	92-100-1-10-6-4	Seleccionar
1;189	2018-5-5	15:57:45	-3490850	-5619873	24	4500	-30:13:-29	92-100-1-10-6-4	Seleccionar
1;189	2018-5-5	15:57:45	-3490850	-5619873	24	4500	-30:13:-29	92-100-1-10-6-4	Seleccionar
1;189	2018-5-5	15:57:45	-3490850	-5619873	25	4500	-34:13:-30	88-100-1-10-6-4	Seleccionar
1;189	2018-5-5	15:57:45	-3490850	-5619873	25	4500	-33:13:-31	80-100-1-10-6-4	Seleccionar
1;1	2018-7-22	20:10:5	-34903905	-56190587	23	490	-0:13:-52	0-200-1-10-6-4	Seleccionar
1;1	2018-7-22	20:10:5	-34903905	-56190587	23	500	-0:13:-50	0-200-1-10-6-4	Seleccionar
1;1	2018-7-22	20:10:5	-34903905	-56190587	23	490	-49:13:-50	0-200-1-10-6-4	Seleccionar
1;189	2018-5-5	15:57:45	-3490850	-5619873	24	4500	-45:13:-25	61-100-1-10-6-4	Seleccionar

Ilustración 59: detalle del servidor de aplicación

Los datos son recibidos correctamente y en el formato acordado.

10.4. Conclusiones

Los alcances obtenidos en las pruebas no llegaron a lo esperado (2 km). Sin embargo, están en el orden. Las causas pueden ser múltiples (antena, ubicación del *gateway*, equipos económicos y básicos para prototipos con limitaciones, entre otras).

Los alcances obtenidos con la misma potencia que Wi-Fi son claramente superiores.

Las pruebas de los sensores indican que son viables con algunas adecuaciones y consideraciones.

Se cumplieron todos los objetivos planteados.

Es posible concluir que LoRa es una tecnología viable para la aplicación de este proyecto.

11. Oportunidades de mejora

La maqueta implementada intenta replicar una solución real de producción en la mayor cantidad de aspectos. Sin embargo, por cuestiones de plazos, costo y alcance, las características que se describen a continuación no fueron incorporadas a la maqueta.

11.1. Nodos

11.1.1. Adecuación al entorno

El prototipo construido necesita una adecuación para intemperie (humedad, lluvia, polvo, bajas y altas temperaturas, etc).

Se debería contemplar un gabinete adecuado que no permita el ingreso de agua y otros elementos. Además, es necesario que la electrónica sea adecuada para la temperatura en la que operaría dentro de un contenedor cerrado expuesto al sol, así como el frío del invierno. Es necesario considerar las características individuales de cada componente y sus rangos de funcionamiento.

Se identifica como necesaria la adecuación al ambiente del contenedor el cual puede ser vandalizado o sufrir impactos por el uso normal.

11.1.2. Alimentación

El prototipo desarrollado no tiene dentro de su alcance la alimentación. Sin embargo, en caso de desarrollar un producto final es necesario plantear una solución.

Se plantean algunas posibles soluciones:

- Es posible adecuar la electrónica para que reduzca su consumo y que, mediante una batería de capacidad y características adecuadas, funcione con una duración suficiente del orden de años. Esta batería debería ser reemplazada periódicamente. Esto tiene como desventaja que se requiere contar con la infraestructura y mano de obra necesarias para reemplazar todas las baterías, las cuales no son despreciables debido a la cantidad de nodos.

- Una alternativa para la alimentación es la instalación de paneles solares. Esto hace necesario la incorporación de una batería recargable y un sistema de gestión de alimentación de forma tal que el equipo siga funcionando cuando no hay luz solar.
- El prototipo desarrollado utiliza Arduino UNO como base. Esta placa utiliza el procesador ATmega328P. Si bien el procesador admite la configuración de un modo de consumo reducido, el resto del dispositivo permanece activo y consumiendo energía. Por lo tanto, si bien la placa Arduino UNO es una plataforma valiosa para desarrollar prototipos, no es un dispositivo ideal para el diseño de equipos para producción con requisitos de bajo consumo. Existen algunas posibilidades para reducir el consumo:
    - Desarrollar electrónica a medida
    - Utilizar otro módulo Arduino adecuado (por ejemplo, Arduino MKR WAN 1300)
    - Desarrollar un módulo externo que encienda/apague el Arduino UNO cuando es necesario. Esto podría no ser sencillo ya que el módulo debería disparar una alerta si se detecta un incendio, y si está apagado no es viable. Por otra parte, es necesario considerar el tiempo de *fix* de GPS, el cual es del entorno de 90 segundos.

11.1.3. Funcionalidades y características

Se plantean algunas posibles mejoras y agregado de funcionalidades que se visualizan para el nodo:

- Dado que la recepción del GPS no siempre es ideal cuando hay obstáculos (ej.: árboles, edificios cercanos, etc), una posible mejora sería la utilización de una antena externa, o bien la utilización de otro módulo GPS de mejores características (por ejemplo, que utilice otras redes como GLONASS o Galileo, u otras tecnologías como triangulación LoRaWAN [61].
- El nodo envía la temperatura cada determinada cantidad de minutos (5 minutos en nuestra implementación). Esto podría ser ineficaz al momento de detectar un incendio. Por lo tanto, debería analizarse la posibilidad de enviar inmediatamente (sin esperar el tiempo estipulado) la alerta. Adicionalmente, se debe considerar la ocupación del canal de radio. Sería deseable analizar la posibilidad de contar con un canal de radio dedicado para emergencias.

- Dado que LoRa permite la comunicación en dos sentidos, sería posible controlar actuadores dentro del contenedor. A modo de ejemplo, sería posible bloquear las puertas del contenedor ante determinados eventos. Por ejemplo, si el contenedor contara con medición de la masa que contiene (balanza), sería posible bloquear la puerta para evitar que se le introduzca más contenido, evitando así exceder la capacidad máxima del contenedor o del camión que lo recoge. Esto podría ser resuelto en el mensaje de reconocimiento, ya que cuenta con campos disponible para usos futuros.
- Los sensores de volumen por ultrasonido tienen ciertas limitaciones. A modo de ejemplo, podría suceder que una bolsa sea detectada por el sensor de volumen y el contenedor sea reportado como lleno. Sin embargo, el contenedor podría tener capacidad ociosa debido a la naturaleza mecánica de las bolsas. Una posible solución a esta problemática sería complementar el sensor de volumen con un sensor de masa. De esta manera, se podría considerar el contenedor como lleno sólo si excede los umbrales de masa y/o volumen de acuerdo a los criterios definidos.
- El prototipo construido tiene un costo estimado en U$S 60 incluyendo sus sensores, transmisor LoRa y Arduino (no incluye el *gateway*). Sin embargo, en una escala mayor y mediante la adecuación al proyecto de contenedores de residuos sería deseable hacer un análisis exhaustivo del costo para lograr reducirlo. Los costos de los componentes se dividen en:
    - Placa Arduino UNO (U$S 22)
    - Shield LoRa/GPS (U$S 24)
    - Sensor HC-SR04 (U$S 6)
    - Sensor de temperatura (U$S 3)
    - Cables y otros (U$S 5)
- La maqueta construida carece de algunas capacidades a nivel de red, como pueden ser seguridad e identificación de los nodos, entre otras. Estas características podrían resolverse mediante LoRaWAN. Esto es descrito en mayor detalle en la sección 5.2.6.
- Sería deseable contar con una consola de administración y gestión de la solución. Esta consola debería contemplar aspectos tales como el enrolamiento de nuevos dispositivos, así como la eliminación de éstos, visualización del registro histórico, información técnica para diagnóstico, etc.
- Podría ser deseable consolidar envíos de múltiples nodos desde el *gateway* hacia el servidor de aplicación a efectos de optimizar el uso de los enlaces de la red IP. Sería

posible generar menos tráfico si se consolidan múltiples envíos debido al *overhead* asociado a los protocolos, debido a que el *payload* es un porcentaje pequeño de cada envío al servidor de aplicación. Es necesario considerar el factor de demora que esto pueda inducir, debido a que se debe esperar a recibir múltiples mensajes para consolidarlos y enviarlos.

- En la implementación del presente proyecto se definió que el envío de mensajes desde los nodos hacia el *gateway* se realice cada 5 minutos. Sin embargo, podría ser deseable ajustar este valor de acuerdo a decisiones de negocio en una implementación de producción de mayor porte. En caso de contar con un número importante de nodos, sería deseable contar con la posibilidad de configurar este parámetro de manera centralizada y dinámica, impactando así a todos los nodos, sin necesidad de configurarlos uno por uno de manera presencial.
- Sería deseable contar desde un sistema central con el estado e información de errores y problemas de los nodos. A modo de ejemplo, sería posible transmitir el nivel de batería, errores de los sensores, parámetros de radio y otros parámetros que puedan aportar al diagnóstico de problemas. Esto fue implementado de manera parcial.
- El presente prototipo utiliza el formato JSON sobre HTTP para el intercambio de mensajes entre el *gateway* y el servidor de aplicaciones. Durante nuestra investigación surgieron alternativas, como MQTT, que son más eficientes en el envío de poca cantidad de información sobre redes IP. A efectos de ilustrar la problemática, se sugiere ver la tabla de cálculos en la sección 9.8.
- Sería deseable implementar seguridad en el intercambio de información, garantizando la confidencialidad e integridad, tanto entre el nodo y *gateway*, como entre el *gateway* y el servidor de aplicaciones. El primero de éstos podría resolverse utilizando LoRaWAN, que ofrece los mecanismos necesarios. Por otra parte, sería deseable utilizar HTTPS y/o VPN (por ejemplo, mediante IPSec) entre el *gateway* y el servidor de aplicación.

12. Referencias bibliográficas

[1] Diario El Pais, «Bomberos tuvo que sofocar fuego en 120 contenedores,» 02 01 2018. [En línea]. Available: https://www.elpais.com.uy/informacion/bomberos-tuvo-sofocar-fuego-contenedores.html. [Último acceso: 26 07 2018].

[2] Diario El Observador, «116 contenedores incendiados profundiza situación crítica de la basura,» 25 12 2017. [En línea]. Available: https://www.elobservador.com.uy/116-contenedores-incendiados-profundiza-situacion-critica-la-basura-n1154055. [Último acceso: 7 26 2018].

[3] Intendencia Municipal de Montevideo, «¿Dónde me comunico para denunciar un contenedor que está prendido fuego?,» 9 10 2014. [En línea]. Available: http://www.montevideo.gub.uy/institucional/preguntas-frecuentes/1donde-me-comunico-para-denunciar-un-contenedor-que-esta-prendido. [Último acceso: 7 27 2018].

[4] Desarrollo Ambiental - IMM, «Twitter,» 8 7 2018. [En línea]. Available: https://twitter.com/IMambiente/status/1015959871882178560. [Último acceso: 8 7 2018].

[5] Intendencia Municipal de Montevideo, «Gestión de Residuos IMM,» [En línea]. Available: http://www.montevideo.gub.uy/gestion-de-residuos. [Último acceso: 27 7 2018].

[6] J. Sanchez, «Cada semana se incendian más de 250 contenedores,» 18 5 2017. [En línea]. Available: http://www.montevideo.gub.uy/institucional/noticias/cada-semana-se-incendian-mas-de-250-contenedores. [Último acceso: 26 7 2018].

[7] Intendencia Municipal de Montevideo, «Instalan nuevos contenedores en municipio B,» 25 2 2016. [En línea]. Available: http://www.montevideo.gub.uy/institucional/noticias/instalan-nuevos-contenedores-en-municipio-b. [Último acceso: 26 7 2018].

[8] Intendencia Municipal de Montevideo, «Intendencia instaló 350 contenedores en municipio G,» 4 5 2015. [En línea]. Available: http://www.montevideo.gub.uy/institucional/noticias/intendencia-instalo-350-contenedores-en-municipio-g. [Último acceso: 26 7 2018].

[9] Vetroplast SRL, «Vetroplast,» [En línea]. Available: https://www.vetroplast.com/es. [Último acceso: 27 7 2018].

[10] IBM, «IBM Intelligent Waste Management Platform,» 12 2015. [En línea]. Available: https://www-01.ibm.com/common/ssi/cgi-bin/ssialias?htmlfid=GVW03059USEN. [Último acceso: 27 7 2018].

[11] ECube Labs, «CleanFLEX,» [En línea]. Available: https://www.ecubelabs.com/ultrasonic-fill-level-sensor/. [Último acceso: 27 7 2018].

[12] SmartBin, «SmartBin,» [En línea]. Available: https://www.smartbin.com/solutions/iot-level-sensors/. [Último acceso: 27 7 2018].

[13] BigBelly INC, «Smart Waste & Recycling System,» [En línea]. Available: http://bigbelly.com/platform/#smart_waste_system. [Último acceso: 27 7 2018].

[14] Wikipedia, «Wikipedia - GLONASS,» 27 7 2018. [En línea]. Available: https://es.wikipedia.org/wiki/GLONASS. [Último acceso: 27 7 2018].

[15] Wikipedia, «Wikipedia - Galileo (navegación por satelite),» 27 7 2018. [En línea]. Available: https://es.wikipedia.org/wiki/Galileo_(navegaci%C3%B3n_por_sat%C3%A9lite). [Último acceso: 27 7 2018].

[16] URSEC, «URSEC,» [En línea]. Available: https://www.ursec.gub.uy. [Último acceso: 27 7 2018].

[17] IMPO, «Decreto 114/2003,» 2003. [En línea]. Available: https://www.impo.com.uy/bases/decretos/114-2003. [Último acceso: 30 7 2018].

[18] UIT-R, «Reglamento de Radiocomunicaciones,» 2016. [En línea]. Available: http://handle.itu.int/11.1002/pub/80da2b36-en. [Último acceso: 19 7 2018].

[19] G. Carro y P. Hernandez, «Uso del Espectro Radioeléctrico en Uruguay y Oportunidades para el Uso de Radio Cognitiva,» Tesis de Ingeniería Eléctrica, Facultad de Ingeniería, Universidad de la República, Montevideo, 2016.

[20] A. Valin, F. Hermida y M. G. Czarnievicz , «Tesis: Sistema inalámbrico de comunicación para equipos de Telemetría,» 2015. [En línea]. Available: https://bibliotecas.ort.edu.uy/bibid/81145. [Último acceso: 30 7 2018].

[21] Wikipedia, «Wikipedia: Big data,» 2018. [En línea]. Available: https://en.wikipedia.org/wiki/Big_data. [Último acceso: 30 7 2018].

[22] Wikipedia, «Wikipedia: Wireless sensor network,» 2018. [En línea]. Available: https://en.wikipedia.org/wiki/Wireless_sensor_network. [Último acceso: 30 7 2018].

[23] Wikipedia, «Wikipedia: bluetooth,» 2018. [En línea]. Available: https://es.wikipedia.org/wiki/Bluetooth. [Último acceso: 10 6 2018].

[24] Wikipedia, «Wikipedia: Wifi,» 2018. [En línea]. Available: https://es.wikipedia.org/wiki/Wifi. [Último acceso: 30 6 2018].

[25] Wi-Fi Alliance, «Wi-Fi Alliance,» [En línea]. Available: https://www.wi-fi.org/. [Último acceso: 27 7 2018].

[26] A. Qutaiba y K. J. Jalal, «Investigating the Power Consumption of an IEEE 802.11x Access Point,» *International Journal of Engineering Issues*, nº 1, pp. 11-24, 2016.

[27] 3GPP, «3GPP Standards for the internet of things,» 11 2016. [En línea]. Available: http://www.3gpp.org/news-events/3gpp-news/1805-iot_r14. [Último acceso: 28 7 2018].

[28] O. Liberg, M. Sundberg, E. Wang, J. Bergman y J. Sachs, Cellular Internet of Things, Elsevier Ltd, 2018, p. 390.

[29] P. R. Egli, «Overview of emerging technologies for Low Power Wide Area Networks in Internet of Things and M2M scenarios,» 15 3 2015. [En línea]. Available: https://www.slideshare.net/PeterREgli/lpwan. [Último acceso: 10 5 2018].

[30] Convergencia Latina, «Antel desarrolla NB-IoT y espera que el 5% de sus ingresos provenga de IoT en 2021,» 16 1 2018. [En línea]. Available: http://www.convergencialatina.com/Nota-Desarrollo/243801-3-9-Antel_desarrolla_NBIoT_y_espera_que_el_5_de_sus_ingresos_provenga_de_IoT_en_2021__. [Último acceso: 27 7 2018].

[31] Sigfox, «Sigfox - The Global Service Provider for the IoT,» [En línea]. Available: https://www.sigfox.com/. [Último acceso: 12 6 2018].

[32] Sigfox, «Sigfox coverage map,» [En línea]. Available: https://www.sigfox.com/en/coverage. [Último acceso: 20 4 2018].

[33] Semtech, «LoRa transceivers,» [En línea]. Available: https://www.semtech.com/products/wireless-rf/lora-transceivers. [Último acceso: 14 7 2018].

[34] Wikipedia, «Wikipedia: LoRa,» 2018. [En línea]. Available: https://en.wikipedia.org/wiki/LoRa. [Último acceso: 12 4 2018].

[35] J. Depuydt, «Jensd's I/O buffer,» 27 12 2016. [En línea]. Available: http://jensd.be/755/network/lorawan-simply-explained. [Último acceso: 24 5 2018].

[36] ITU-R, «Regiones de ITU-R,» 17 06 2010. [En línea]. Available: https://www.itu.int/net/ITU-R/index.asp?category=information&rlink=emergency-bands&lang=es. [Último acceso: 5 2018].

[37] LoRa Alliance, «LoRa Alliance,» [En línea]. Available: https://lora-alliance.org/. [Último acceso: 15 5 2018].

[38] Link Labs, «Symphony Link,» [En línea]. Available: https://www.link-labs.com/symphony. [Último acceso: 20 6 2018].

[39] Digikey, «Digikey SX1276IMLTRT,» 30 7 2018. [En línea]. Available: https://www.digikey.com/product-detail/en/semtech-corporation/SX1276IMLTRT/SX1276IMLTRTCT-ND/4259551. [Último acceso: 30 7 2018].

[40] A. Augustin, J. Yi, T. Clausen y W. M. Townsley, «A Study of LoRa: Long Range & Low Power Networks for the Internet of Things,» 9 9 2016. [En línea]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5038744/#B16-sensors-16-01466). [Último acceso: 7 7 2018].

[41] Wikipedia, «Wikipedia: DASH7,» 2018. [En línea]. Available: https://en.wikipedia.org/wiki/DASH7. [Último acceso: 4 7 2018].

[42] Actility, «Suez optimizes waste management in the Netherlands,» 2016. [En línea]. Available: https://www.actility.com/customer-stories/suez-implements-connected-bins-netherlands/. [Último acceso: 30 7 2018].

[43] Actility, «In Lebanese Vineyard, LoRaWAN Is Making Wine Better,» 12 2017. [En línea]. Available: https://www.actility.com/wp-content/uploads/2017/12/Customer-stories_Lebanese-vineyard_01.pdf. [Último acceso: 30 7 2018].

[44] Actility, «Fuel tank monitoring in Belgium made easy by FULLUP,» [En línea]. Available: https://www.actility.com/customer-stories/fuel-tank-monitoring-belgium-by-fullup/. [Último acceso: 3 8 2018].

[45] Intendencia Municipal de Montevideo, «Encuentro de Ciudades Inteligentes,» [En línea]. Available: http://www.montevideo.gub.uy/encuentro-de-ciudades-inteligentes. [Último acceso: 4 8 2018].

[46] Semtech, «SX1276 datasheets,» [En línea]. Available: https://www.semtech.com/products/wireless-rf/lora-transceivers/SX1276. [Último acceso: 30 4 2018].

[47] Wikipedia, «Wikipedia: Chirp spread spectrum,» 2018. [En línea]. Available: https://en.wikipedia.org/wiki/Chirp_spread_spectrum. [Último acceso: 30 7 2018].

[48] Semtech, «LoRa modulation basics,» 5 2015. [En línea]. Available: https://www.semtech.com/uploads/documents/an1200.22.pdf. [Último acceso: 18 5 2018].

[49] Semtech, «Semtech SX127X datasheet,» 8 2016. [En línea]. Available: https://www.semtech.com/uploads/documents/DS_SX1276-7-8-9_W_APP_V5.pdf. [Último acceso: 25 4 2018].

[50] The Things Network, «LoRaWAN Frequencies Overview,» [En línea]. Available: https://www.thethingsnetwork.org/docs/lorawan/frequency-plans.html. [Último acceso: 23 6 2018].

[51] LoRa Alliance, «LoRaWAN Specification v1.0.3,» 7 2018. [En línea]. Available: https://www.lora-alliance.org/resource-hub/lorawantm-specification-v103. [Último acceso: 30 7 2018].

[52] Wikipedia, «Wikipedia: ALOHAnet,» 2018. [En línea]. Available: https://en.wikipedia.org/wiki/ALOHAnet#ALOHA_protocol. [Último acceso: 29 7 2018].

[53] LoRa Alliance, «LoRa Alliance security whitepaper,» 4 2018. [En línea]. Available: https://www.lora-alliance.org/resource-hub/lora-alliance-security-whitepaper. [Último acceso: 4 2018].

[54] Dragino, «Dragino LoRa IoT kit,» [En línea]. Available: http://www.dragino.com/products/lora/item/120-lora-iot-kit.html. [Último acceso: 15 3 2018].

[55] Arduino, «Arduino UNO rev3,» [En línea]. Available: https://store.arduino.cc/usa/arduino-uno-rev3. [Último acceso: 21 6 2018].

[56] Bouni, «Arduino UNO,» 2014. [En línea]. Available: https://components101.com/microcontrollers/arduino-uno. [Último acceso: 6 8 2018].

[57] Adafruit, «Overview: DHT11, DHT22 and AM2302 sensors,» [En línea]. Available: https://learn.adafruit.com/dht?view=all. [Último acceso: 1 8 2018].

[58] Maxim Integrated, «DS18B20,» 1 2015. [En línea]. Available: https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf. [Último acceso: 15 6 2018].

[59] Makerfabs, «HC-SR04,» [En línea]. Available: https://www.makerfabs.com/index.php?route=product/product&product_id=72. [Último acceso: 1 8 2018].

[60] Marlin P Jones & Assoc. Inc., «HC-SR04 User Guide,» [En línea]. Available: https://www.mpja.com/download/hc-sr04_ultrasonic_module_user_guidejohn.pdf. [Último acceso: 1 6 2018].

[61] LoRa Alliance, «LoRa Alliance Geolocation Whitepaper,» 1 2018. [En línea]. Available: https://www.lora-alliance.org/resource-hub/lora-alliance-geolocation-whitepaper. [Último acceso: 28 5 2018].

[62] Quectel, «Quectel L80,» [En línea]. Available: http://www.quectel.com/product/l80.htm. [Último acceso: 4 6 2018].

[63] Wikipedia, «Wikipeda: NMEA 0183,» [En línea]. Available: https://en.wikipedia.org/wiki/NMEA_0183. [Último acceso: 5 5 2018].

[64] Dragino, «Dragino LoRa shield,» [En línea]. Available: http://www.dragino.com/products/module/item/102-lora-shield.html. [Último acceso: 20 5 2018].

[65] Dragino, «Dragino LoRa GPS shield,» [En línea]. Available: http://wiki.dragino.com/index.php?title=Lora/GPS_Shield. [Último acceso: 20 5 2018].

[66] Dragino, «LG01 LoRa gateway user manual,» 3 4 2018. [En línea]. Available: http://www.dragino.com/downloads/downloads/UserManual/LG01_LoRa_Gateway_User_Manual.pdf. [Último acceso: 20 4 2018].

[67] OpenWRT, «OpenWRT,» [En línea]. Available: https://openwrt.org/. [Último acceso: 18 6 2018].

[68] Dragino, «Dragino examples catalog,» 6 5 2018. [En línea]. Available: http://wiki.dragino.com/index.php?title=Examples_Catalog. [Último acceso: 10 5 2018].

[69] ThingSpeak, «ThingSpeak,» [En línea]. Available: https://thingspeak.com/. [Último acceso: 3 8 2018].

[70] Arduino, «Arduino IDE,» [En línea]. Available: https://www.arduino.cc/en/Main/Software. [Último acceso: 31 7 2018].

[71] Wikipedia, «Wikipedia: 1-Wire,» 2018. [En línea]. Available: https://es.wikipedia.org/wiki/1-Wire. [Último acceso: 1 7 2018].

[72] Arduino, «Bridge Library for Yún devices,» [En línea]. Available: https://www.arduino.cc/en/Reference/YunBridgeLibrary. [Último acceso: 3 8 2018].

[73] M. Hart, «TinyGPS,» 31 8 2013. [En línea]. Available: http://arduiniana.org/libraries/tinygps/. [Último acceso: 12 5 2018].

[74] M. McCauley, «RadioHead Packet Radio library for embedded microprocessors,» 14 4 2014. [En línea]. Available: http://www.airspayce.com/mikem/arduino/RadioHead/. [Último acceso: 7 5 2018].

[75] P. Stoffregen, «OneWire,» [En línea]. Available: https://github.com/PaulStoffregen/OneWire.

[76] M. Burton, «Github,» [En línea]. Available: https://github.com/milesburton/Arduino-Temperature-Control-Library. [Último acceso: 17 6 2018].

[77] Quectel, «L80 Hardware Design,» 10 8 2013. [En línea]. Available: https://www.quectel.com/UploadImage/Downlad/L80_Hardware_Design_V1.1.pdf. [Último acceso: 5 7 2018].

[78] DaveO, «DS18B20 returns Zero - sensor or code - or normal ?,» 27 7 2011. [En línea]. Available: https://forum.arduino.cc/index.php?topic=67747.0. [Último acceso: 1 7 2018].

ANEXO 1 - Arquitectura del Software – Acceso y consumo del Web Service

El contenido de este anexo fue redactado y entregado a nosotros por el equipo de proyecto de Licenciatura en Sistemas a los efectos de lograr el envío en el formato acordado.

El detalle de cómo son utilizados los campos se encuentra en la sección 9.5.4 de este documento.

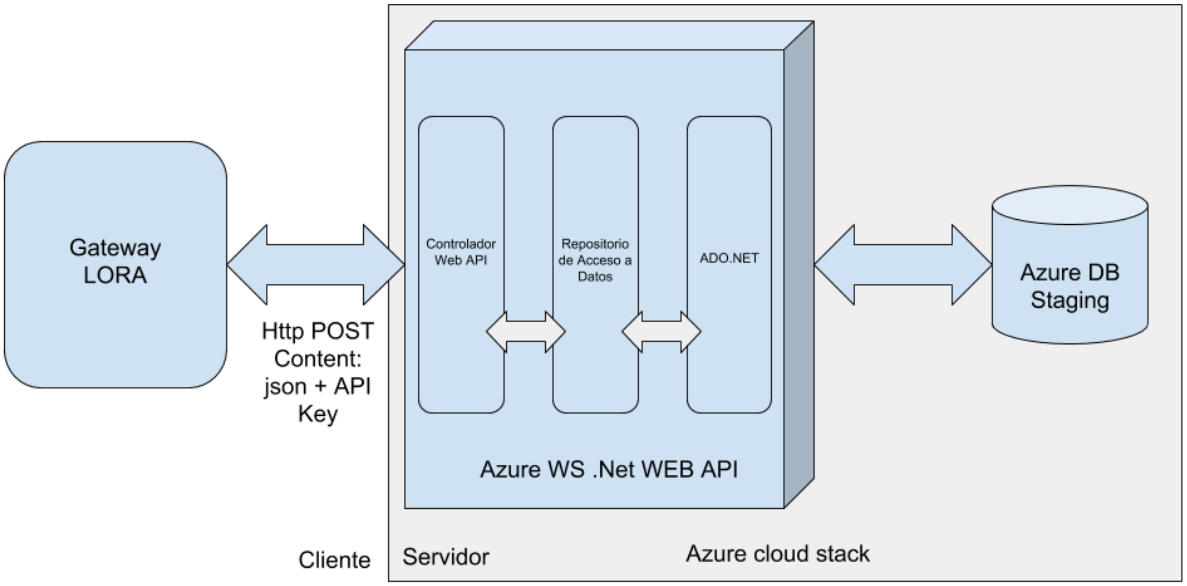


Ilustración 60: Arquitectura de alto nivel del Web Service.

A los efectos de su consumo, el *Web Service* alojado en Azur escucha únicamente los mensajes POST HTTP en la URL:

http://415386fcfffe48ff8d2f21ef2c818be6.cloudapp.net/SaveBin

El formato de los *request* JSON se indica a continuación:

```
{

  "C1": "sample string 1",

  "C2": "sample string 2",

  "C3": "sample string 3",

  "C4": "sample string 4",

  "C5": "sample string 5",

  "C6": "sample string 6"

  "C7": "sample string 7"

  "C8": "sample string 8"

  "C9": "sample string 9"

}
```

Debe tenerse presente que el servicio utiliza seguridad para el acceso al controlador a través de una clave de acceso que denominamos API Key. Esta clave se encuentra almacenada en el servicio y únicamente será conocida por los integrantes del equipo de sistemas y de telecomunicaciones, estos últimos serán quienes consuman el servicio para la carga de los datos de telemetría enviados por el *Gateway*.

La API Key se integrará como un encabezado (APIKey) en el request HTTP POST. En el ejemplo de sintaxis para el comando cURL, la API Key es “SmartBin12345”.