Universidad Tecnológica Nacional
Facultad Regional Bahía Blanca

Ingeniería Electrónica

Proyecto Final

Red Inalámbrica de Sensores para Áreas Rurales

Autores
Walter Moreno
Leandro Guanuco

Director
Mg.Ing.Guillermo Friedrich

Codirector
Ing. Adrián Laiuppa

Bahía Blanca | 31 de Octubre de 2023

Resumen

El presente Proyecto Final de la carrera de Ingeniería Electrónica, consistió en el desarrollo de una red inalámbrica de nodos equipados con sensores, para el monitoreo de variables físicas en áreas rurales. Se utilizaron herramientas de software libre y se aplicó un fuerte enfoque hacia Internet de las Cosas.

Todos los nodos están basados en el microcontrolador cc2538 y el sistema operativo ContikiOS. En la red existe un solo Nodo Gateway y varios Nodos Sensores. Cada nodo posee una dirección IPv6, y la transmisión de datos de cada Nodo Sensor va dirigida hacia el Nodo Gateway. El Nodo Gateway hace de router de borde de la red IPv6, y a su vez sube los datos hacia Internet mediante la red de telefonía móvil. La alimentación de los nodos consta de paneles solares y baterías, para que el sistema pueda funcionar en entornos rurales.

Se verificó el correcto funcionamiento del sistema, recolectando datos durante varios días en el Establecimiento Rukalen (ubicado en cercanía de Cabildo, partido de Bahía Blanca). También se verificó la correcta comunicación entre nodos separados por varios kilómetros de distancia.

Palabras Clave: IoT, cc2538, IEEE 802.15.4, Contiki, IPv6, 6lowPAN.

Índice

1 Introducción 1
2 Conceptos Teóricos 2
2.1 Internet de las Cosas (IoT) 2
2.1.1 Definición 2
2.1.2 Características 2
2.1.3 Tendencias de uso 2
2.2 Redes Inalámbricas de Sensores 3
2.2.1 Definición 3
2.2.2 Topologías de Red 4
2.3 Sistema Operativo ContikiOS 5
2.3.1 Principales Características 5
2.3.2 Procesos 5
2.3.3 Protothreads 5
2.3.4 Eventos 6
2.3.5 Timers 6
2.4 Capas de la Red 6
2.4.1 Capa Física 7
2.4.2 Radio Duty Cycle 7
2.4.3 Capa de enlace 10
2.4.4 Capa de adaptación 10
2.4.5 Capa de Red 10
2.4.6 Capa de Transporte 11
2.4.7 Capa de Aplicación 12
2.5 Presupuesto de enlace inalámbrico 13
3 Desarrollo del Proyecto 15
3.1 Introducción 15
3.2 Esquema general de funcionamiento 15
3.2.1 Nodo sensor 18
3.2.2 Nodo Gateway 18
3.2.3 Enrutamiento y formación de la Red de Sensores 19
3.2.4 Thingspeak y Thingview 22
3.3 Hardware 25
3.3.1 Introducción 25
3.3.2 Diseño de Hardware 26
3.4 Alimentación de los dispositivos de red 30
3.4.1 Alimentación de los nodos sensores 30
3.4.2 Alimentación del Nodo Gateway 31
3.5 Software 32
3.5.1 Introducción 32
3.5.2 Descripción funcional de la aplicación 33
3.5.3 Diseño 33
3.5.4 Implementación 35
3.6 Tareas automatizadas en la Estación Base (Raspberry) 41
4 Cálculos, ensayos y Mediciones 42
4.1 Cálculo de paneles solares y baterías 42
4.2 Medición de Voltaje de las Baterías con el cc2538 47
4.3 Cálculo de presupuesto de enlace para los módulos CC2538EM 51
4.4 Cálculo de presupuesto de enlace para los módulos CC2538+CC2592 53

4.5 Ensayos de rendimiento de la red 55
4.5.1 Ensayos de los módulos CC2538EM 57
4.5.2 Ensayos de los módulos CC2538+CC2592 58
4.6 Ensayo final 60
4.6.1 Resultados de ensayo del Nodo Gateway 67
4.6.2 Resultados de ensayo del Nodo 1 69
4.6.3 Resultados de ensayo del Nodo 2 70
4.6.4 Resultados de ensayo del Nodo 3 70
5 Conclusiones 71

1 Introducción

El presente informe aborda el diseño e implementación de una red inalámbrica de sensores para el monitoreo de variables físicas en áreas rurales, las cuales pueden ser de interés para una persona/operario que deba ejecutar una acción en función de esos valores. En la Figura 1.1 se presenta un esquema general de la red. La misma está conformada por Nodos Sensores y un Router de Borde, todos ellos basados en módulos cc2538 con amplificador cc2592 (además de contar con baterías, paneles solares y antenas omnidireccionales). El Router de Borde IPv6 es el encargado de armar la red, y es uno de los componentes del Nodo Gateway que, además dispone de una Raspberry Pi 3 (Estación Base) y un Módem 3G. El Nodo Gateway cumple la función de obtener las mediciones de todos los nodos y subirlas a la nube de forma automática, para poder ser visualizadas desde la plataforma Thingspeak, a través de un ordenador, o desde un Smartphone con la aplicación Thingview.

En el informe también se detallan los cálculos para dimensionar las baterías y paneles solares, la elección de los sensores utilizados, y las principales características de hardware y código fuente correspondiente a cada uno de los nodos.

Por último, para verificar el correcto funcionamiento de la red, el sistema fue desplegado en un campo cercano a la localidad de Cabildo, a fin de realizar pruebas de distancia y recolección de datos. Dichas pruebas fueron exitosas.

A continuación, se presenta el escenario completo del sistema implementado (Figura 1.1):

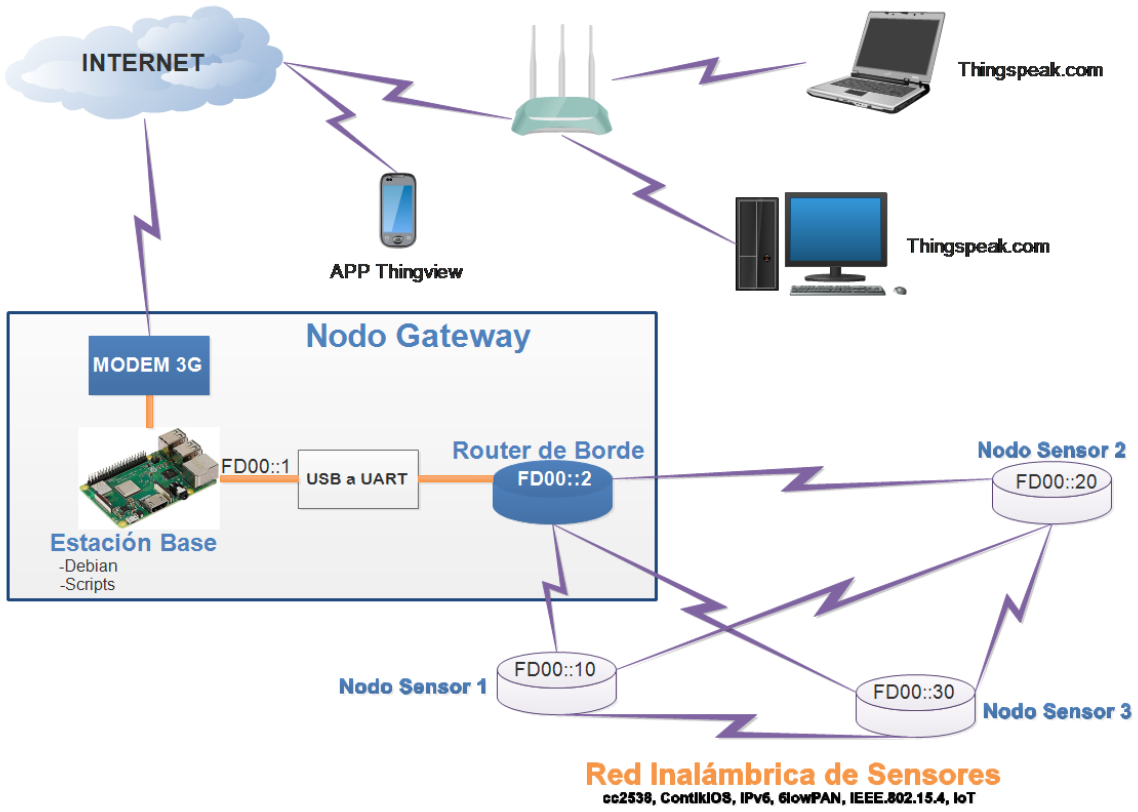


Figura 1.1: Sistema completo de la red inalámbrica de sensores

2 Conceptos Teóricos

2.1 Internet de las Cosas (IoT)

2.1.1 Definición

Internet de las Cosas (IdC o IoT) es un concepto abstracto que implica la conexión de objetos de uso cotidiano a Internet. Este concepto transforma dispositivos como comunicadores, cámaras y sensores, que solían estar limitados a circuitos cerrados, permitiéndoles comunicarse a través de la red mundial.
Una forma de definir el IoT es como una red que une objetos físicos utilizando Internet. Estos dispositivos emplean software integrado que no solo les permite conectarse a la web, sino también ejecutar acciones específicas basadas en instrucciones remotas. En resumen, se trata de la conexión digital entre objetos físicos.

2.1.2 Características

El Internet de las cosas, o IoT, utiliza chips y circuitos especiales para realizar trabajos específicos. Cada objeto conectado a Internet tiene su propia dirección IP, la cual se utiliza para enviar órdenes y recibir información. Estos objetos pueden comunicarse con otros servidores y enviar la información que recopilan. En resumen, las características principales del IoT son:

- **Conexión Total:** Todo puede conectarse a la red global.
- **Servicios para las cosas:** El IoT ofrece servicios para las cosas, sin las limitaciones propias de esos objetos. Esto incluye mantener la relación entre lo físico y lo virtual.
- **Variedad en dispositivos:** Los dispositivos en el IoT son muy variados en hardware, redes y plataformas. Pueden comunicarse con otros dispositivos igualmente heterogéneos.
- **Dinamismo:** Los dispositivos cambian constantemente, adaptándose al usuario, al contexto y a la velocidad requerida. Por ejemplo, de estar en reposo a estar activos, conectados o desconectados, según la situación.

2.1.3 Tendencias de uso

En el ámbito privado, el IoT se está volviendo muy popular. Algunos sectores empresariales que lo están adoptando son:

- **Industria manufacturera:** Utilización de maquinaria, robots, y sensores para controlar procesos de fabricación y supervisar la producción.
- **Control urbano:** Gestión de semáforos, puentes, cámaras y vías de tren para mejorar la infraestructura de las ciudades.
- **Monitoreo ambiental:** Empleo de sensores para medir condiciones atmosféricas, meteorológicas y sísmicas.
- **Salud:** Monitoreo no invasivo de pacientes, permitiendo seguimiento fuera de hospitales.
- **Agricultura:** Uso de sensores para supervisar suelos, cultivos y variables ambientales.

- **Transporte:** Rastreo satelital, control del estado de vehículos y desarrollo de automóviles conectados.
- **Energía:** Implementación de Smart Grids para gestionar la demanda de electricidad, integrar energías renovables y mejorar el servicio al cliente mientras se reduce el consumo energético.

2.2 Redes Inalámbricas de Sensores

2.2.1 Definición

Las redes inalámbricas de sensores (WSN, por sus siglas en inglés, Wireless Sensor Network) se basan en dispositivos de bajo costo y consumo energético, conocidos como nodos. Estos nodos tienen la capacidad de recolectar información de su entorno, procesarla localmente y transmitirla mediante enlaces inalámbricos a un nodo central de coordinación. En este proceso, los nodos desempeñan un papel fundamental al actuar como componentes de la infraestructura de comunicación, al reenviar los mensajes generados por nodos ubicados a mayor distancia hacia el centro de coordinación.
Una red inalámbrica de sensores se compone de numerosos dispositivos distribuidos espacialmente, que utilizan sensores para supervisar diversas condiciones, como, por ejemplo: temperatura, sonido, vibración, presión, movimiento o niveles de contaminantes. Estos sensores pueden ser estacionarios o móviles.
Cada uno de estos dispositivos (Figura 2.1), son unidades autónomas que incluyen un microcontrolador, una fuente de energía (generalmente una batería), un radio-transceptor (RF) y sensores específicos para la tarea de monitoreo.

Figura 2.1: Dispositivo de una WSN (red inalámbrica de sensores)

Debido a las limitaciones de la duración de las baterías, los nodos se diseñan considerando la conservación de la energía, y suelen permanecer en modo de bajo consumo (o durmiente) Las WSN tienen capacidad de autorrepararse; si un nodo se avería, la red buscará nuevas rutas para encaminar los paquetes de datos. De esta forma, la red, sobrevivirá en su conjunto, aunque haya algunos nodos que pierdan potencia o se destruyan. Las capacidades de auto-diagnóstico, auto-configuración, auto-organización, auto-restauración y reparación, son propiedades desarrolladas en este tipo de redes para solucionar problemas que no podrían abordarse con otras tecnologías.
Las redes de sensores se definen por ser desatendidas (sin intervención humana), con una alta probabilidad de fallos (en los nodos o en la topología) y generalmente se construyen ad-hoc para resolver problemas específicos.

2.2.2 Topologías de Red

Los nodos que integran las WSN suelen adoptar una organización o estructura de red particular para transmitir su información a través de toda la red. Algunas de las topologías de red más habituales (ver figura 2.2) son:

- Estrella (Star).
- Red de árboles aglomerados (Cluster Tree Network).
- Redes en malla (Mesh).

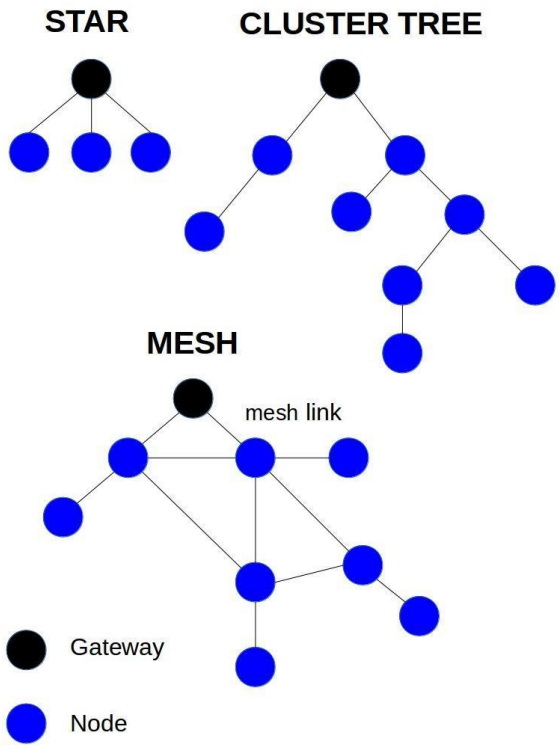


Figura 2.2: Topologías de las WSN

Cuando se emplea una topología de tipo estrella (Star) en una WSN, todos los nodos se conectan directamente a un coordinador único, lo que facilita una comunicación directa y simple. Sin embargo, en redes de gran tamaño, esta configuración puede no ser recomendable, ya que la distancia puede causar pérdida de comunicación y de la información transmitida.
En cuanto a la topología de red de árboles aglomerados (Cluster Tree Network), cada nodo (representado como puntos azules en la Figura 2.2) se enlaza con un nodo de nivel superior en una estructura jerárquica en forma de árbol. La transmisión de información sigue esta jerarquía, permitiendo que los nodos de niveles inferiores envíen datos a los superiores, sin acceso a la información de estos últimos. Sin embargo, esta configuración presenta la problemática de que, si un nodo de nivel superior falla o se daña, los nodos bajo su jerarquía quedan desconectados, resultando en la pérdida de la información que adquieren.

En el caso de una topología de red malla (Mesh), los nodos pueden establecer múltiples conexiones con diversos nodos de la red y ajustarse automáticamente si un nodo falla o su información se corrompe. Esto asegura una mayor confiabilidad gracias a la redundancia en la transmisión de datos y garantiza que se envíen por la ruta más confiable disponible.

2.3 Sistema Operativo ContikiOS

Contiki es un sistema operativo de código abierto diseñado para la Internet de las Cosas. Se caracteriza por ser multi-hilo y está diseñado para facilitar la conexión a Internet de microcontroladores (MCUs) que incorporan radios de bajo costo y consumo energético.

2.3.1 Principales Características

Contiki es un sistema operativo diseñado para sistemas con recursos limitados que posee un pequeño footprint (cantidad memoria utilizada para la ejecución de un programa) de memoria. Lo que lo hace ideal para operar en dispositivos de baja potencia como las Redes Inalámbricas de Sensores. Ofrece mecanismos para controlar y minimizar el consumo de energía, además de admitir estándares como IPv4, IPv6 y otros protocolos específicos para redes de baja potencia y pérdidas (LLN).
Las aplicaciones se desarrollan en lenguaje C y el sistema ha sido adaptado a una amplia gama de dispositivos, con un equipo de desarrollo respaldado por colaboradores de empresas líderes en la industria de microcontroladores. Contiki ejecuta múltiples hilos en paralelo, los cuales se activan según condiciones específicas, procesan eventos en orden cronológico y cada evento tiene un hilo asociado para su manejo.

2.3.2 Procesos

El código en Contiki puede operar en dos contextos: cooperativo o apropiativo (con desalojo; preemptive). En el contexto cooperativo, el código se ejecuta secuencialmente con respecto a otro código similar y debe completarse antes de que otro pueda iniciar. En cambio, el código apropiativo suspende temporalmente el código cooperativo, impidiendo que este retome su ejecución hasta que el apropiativo finalice por completo. En Contiki, los procesos se ejecutan en el contexto cooperativo, mientras que las interrupciones y los temporizadores operan en el contexto apropiativo.
Cada código ejecutable en Contiki contiene procesos que generalmente inician al arrancar el sistema o cuando se carga un módulo con procesos al sistema. Estos procesos se activan cuando, por ejemplo, finaliza un temporizador o se produce un evento externo.
Un proceso en Contiki se compone de dos partes: un bloque de control y un hilo (thread). El bloque de control, almacenado en la RAM, incluye información de tiempo de ejecución del proceso, como su nombre, estado (activo, inactivo, llamado) y un enlace a su hilo correspondiente. Por otro lado, el hilo se guarda en la memoria de programa (normalmente no volátil), es utilizado por el usuario y contiene el código de ejecución del proceso. Este hilo está conformado por un protothread que es invocado por el planificador de procesos, responsable de asignar el tiempo de CPU disponible entre los procesos disponibles para ejecución.

2.3.3 Protothreads

Los protothread son funciones capaces de esperar por eventos. Es una forma de estructurar el código que le permite al sistema ejecutar otras actividades mientras el código está esperando que ocurra un evento.

Cuando un evento inicializa un proceso, se ejecuta su código desde el principio y el proceso queda en estado llamado. Cuando el protothread deba esperar por un evento, le devuelve el control al process scheduler y su proceso pasa al estado activo. Cuando un evento llame al proceso se ejecutará si el evento es el esperado y en caso afirmativo se ejecutará el código del protothread desde donde se había quedado anteriormente.

2.3.4 Eventos

En Contiki, los procesos se ejecutan en respuesta a eventos que reciben. Estos eventos están asociados al proceso que van a despertar y pueden transmitir datos al mismo. Se distinguen dos tipos de eventos: asíncronos y síncronos.
Cuando ocurre un evento asíncrono, éste es encolado en una cola de eventos del Kernel y será entregado al proceso al llegar su turno. Son eventos que pueden ser enviados o posteados a un proceso o a todos.
Cuando un evento síncrono es posteado el evento se entrega inmediatamente al proceso. Estos eventos solo pueden tener como destinatario un único proceso.

2.3.5 Timers

Contiki proporciona librerías de timers que pueden ser empleadas tanto por las aplicaciones desarrolladas por los usuarios como por el sistema operativo en sí. Estas bibliotecas contienen funciones para verificar si un timer ha expirado, despertar al sistema de un estado de bajo consumo en un momento específico y programar tareas en tiempo real. Los distintos tipos de timers implementados en Contiki son:

- Timer: Se utilizan para chequear si transcurrió un cierto período de tiempo. Cuando expira no toma ninguna iniciativa, por lo cual hay que consultarlo. Utiliza los ticks del reloj del sistema como unidad de medida, por lo que sirven para medir períodos cortos de tiempo (tiene gran resolución).
- sTimer: Es similar al Timer, a diferencia que utiliza segundos como medida, por lo que pueden medir grandes cantidades de tiempo a cambio de tener menor resolución.
- eTimer: Permite generar eventos temporizados, utilizando clock time() para obtener la hora del sistema. El eTimer se implementa como un proceso, el cual postea al proceso que lo configuro, el evento PROCESS_EVENT_TIMER cuando vence su tiempo.
- cTimer: Este timer llama a una cierta función, función de callback, cuando expiran, utilizando clock time() para obtener la hora del sistema.
- rTimer: Permite agendar y ejecutar tareas de tiempo real, utilizando su propio módulo de reloj para permitir mayor resolución. Las tareas se ejecutan en modo apropiativo. Generalmente se usa para llamar un process poll(. . .)

2.4 Capas de la Red

Por lo general, las implementaciones de redes TCP/IP necesitan una gran cantidad de memoria y utilizan paquetes de gran tamaño debido a sus extensos encabezados. Esto limita su utilidad en entornos de recursos limitados. En las redes convencionales, como las redes informáticas, los nodos suelen estar conectados a la red eléctrica y los enlaces son estables. No obstante, en redes de recursos limitados, los nodos tienen alimentación limitada y suelen operar en entornos ruidosos, lo que puede generar enlaces inestables. Por lo tanto, los requisitos para una red inalámbrica de sensores son bastante distintos a los de las redes tradicionales.

La pila de protocolos diseñada para redes inalámbricas de sensores se ha desarrollado con el propósito de ser más ligera y adaptarse a los desafíos presentes en entornos con recursos limitados y alta pérdida de datos. Un modelo de capas adecuado para este tipo de redes se muestra en la Tabla 2.1.

Capa de Red	**Protocolo**
Aplicación	CoAP
Transporte	UDP
Red	RPL IPv6
Adaptación	6lowPAN
Enlace	IEEE 802.15.4 MAC
RDC (Radio Duty Cycle)	ContikiMAC
Física	IEEE 802.15.4 PHY

Tabla 2.1: Pila de protocolos

2.4.1 Capa Física

La capa física provee el servicio a la capa de enlace al codificar los datos de la trama de enlace en un patrón de bits para transmitir a través del medio. Esta capa define características como voltaje, frecuencia, tiempos, entre otros.
En entornos de bajos recursos y alta pérdida de datos, existen varios estándares que especifican la capa física. Algunos ejemplos son IEEE 802.15.4 y Power Line Communication (PLC). Para este proyecto en particular, se optó por trabajar con el estándar IEEE 802.15.4 debido a su capacidad para facilitar la comunicación inalámbrica con bajo consumo de energía.
IEEE 802.15.4 es un estándar especificado por el Institute of Electrical and Electronics Enginners (IEEE) para las redes de área personal (Personal Area Networks-PAN) cuya tasa de transferencia varía entre 20 kbps y 250 kbps dependiendo de la frecuencia. En IEEE 802.15.4 se usa el rango de frecuencia 2400-2483,5 Mhz. Esta banda de 2400 Mhz está dividida en 16 canales de 3 Mhz de ancho de banda y pueden solaparse con señales de Wi-Fi, haciendo que aumente la posibilidad de tener interferencia. El estándar utiliza modulación de fase binaria (BPSK) o en cuadratura (QPSK).

2.4.2 Radio Duty Cycle

La capa Radio Duty Cycle (RDC) se encarga de gestionar el ciclo de trabajo de la radio. Dado que la radio consume mucha energía tanto en transmisión como en recepción, es crucial mantenerla apagada durante el mayor tiempo posible para lograr un bajo consumo energético. Contiki dispone de varios mecanismos para administrar el ciclo de la radio, entre los que se incluyen:

- NullRDC
- ContikiMAC (desarrollado por Contiki)
- XMAC
- LPP

Para el proyecto se utilizó ContikiMAC, que fue desarrollado y presentado como un mecanismo eficiente para el manejo del ciclo de trabajo de la radio en redes de sensores que utilizan el Sistema Operativo Contiki. Debido a que, por lo general, el mayor consumo de la comunicación está en el transmisor, los nodos pueden despertar su radio periódicamente para escuchar el canal. El emisor transmite varias veces la misma trama a lo largo de un ciclo completo o hasta recibir un reconocimiento (ACK) del receptor. En este mecanismo la sincronización no es necesaria, pero es de suma utilidad para hacer que el emisor comience a transmitir justo antes del momento en el que el receptor se enciende. En la Figura 2.3 se muestra este funcionamiento, donde transmisiones broadcast son enviadas con paquetes de datos repetidos durante un ciclo de escucha o hasta que se recibe un reconocimiento. Los mensajes Broadcast son paquetes de datos que tienen como dirección destino la dirección MAC FF.FF.FF.FF.FF.FF. De esta forma el mensaje es enviado a todos los dispositivos de la red.

ContikiMAC es el responsable de manejar el ciclo de trabajo del radio (RDC); hace que cada nodo se despierte periódicamente cada t_{cycle}. Al despertarse, el nodo hace dos sensados de canal (CCA clear channel assessment) separados un tiempo t_c (se requieren dos CCA en caso de que el primero ocurra entre dos tramas). Si no detecta señal se vuelve a estado inactivo. Si se detecta señal mantiene la radio encendida hasta que la trama completa haya sido recibida y envía un reconocimiento al emisor. La frecuencia con la que se despierta el nodo se conoce como tasa de sensado de canal (*Channel Check Rate*-CCR) que es $1/t_{cycle}$. En el presente proyecto se modificó el CCR para disminuir el consumo del sistema. Al disminuir el CCR el período de tiempo en el que el MCU permanece en estado inactivo aumenta y por ende el consumo disminuye.

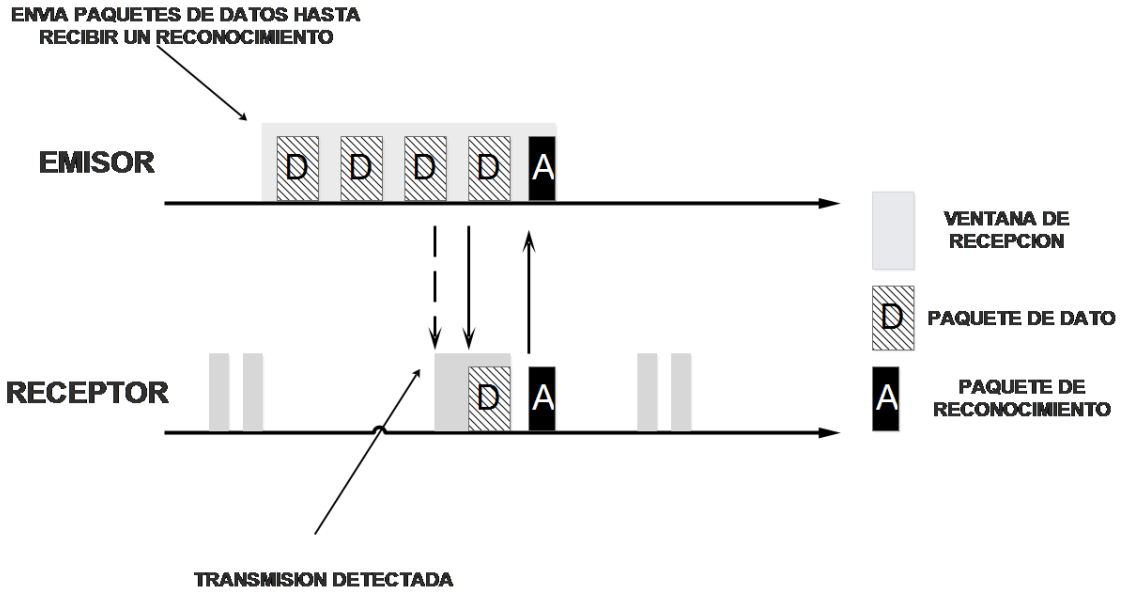


Figura 2.3: Ciclo de ContikiMAC

En la Figura 2.4 se pueden observar los tiempos requeridos, siendo los mismos:

- t_i**:** intervalo entre paquetes transmitidos.
- t_r**:** tiempo requerido para obtener una RSSI (indicador de intensidad de señal recibida) estable.
- CCA**:** sensados del canal.
- t_c**:** intervalo entre CCA.

- t_a**:** tiempo entre que se recibe un paquete y se manda un ACK.
- t_d**:** tiempo requerido para hacer una detección exitosa de un ACK.
- t_s**:** tiempo de envío de un paquete de datos.

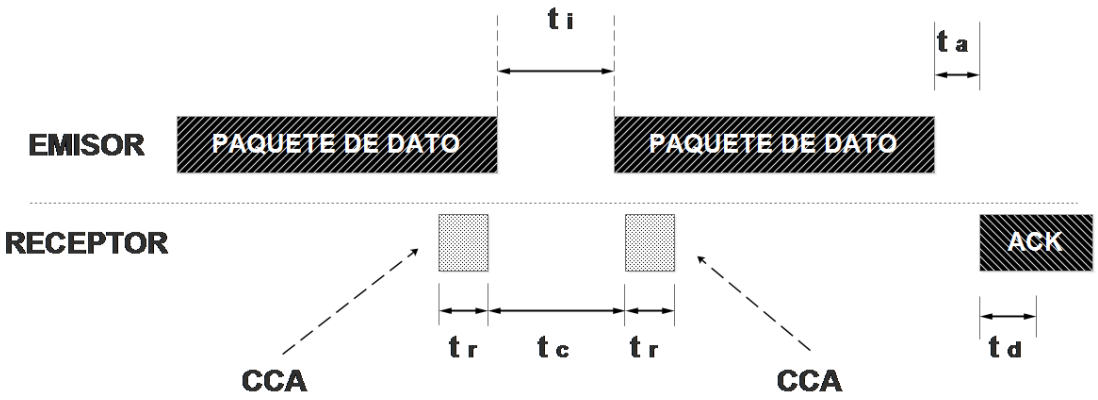


Figura 2.4: Tiempos de ContikiMAC

Como consecuencia de este funcionamiento se debe cumplir la siguiente restricción para los tiempos mencionados:

$$t_a + t_d < t_i < t_c < t_c + 2t_r < t_s$$

Esto significa que el tiempo necesario para detectar un reconocimiento tiene que ser menor que el intervalo entre las tramas. Este intervalo además tiene que ser menor que el intervalo entre los CCA. De lo contrario, dos CCA no garantizarán que se detecte la secuencia de tramas. Los últimos dos términos implican que una trama debe ser mayor que el tiempo que toma hacer dos CCA.
En cuanto a los valores numéricos, según el estándar IEEE 802.15.4, t_a es igual la duración de 12 símbolos, donde cada símbolo dura 0,016 ms, por lo tanto t_a= 0.192 ms. Por otro lado, un receptor IEEE802.15.4 puede detectar de forma fiable un ACK en 0,16 ms, por lo que t_d= 0.16 ms. Entonces, $t_a + t_d$ = 0,352 ms.
t_r depende del hardware (según datasheet del cc2538, es de 0,192 ms), mientras que t_c y t_i pueden ser determinados por el usuario siempre que se cumpla con la desigualdad anterior.

Reemplazando tenemos que:

$$0{,}352 \text{ ms} < t_i < t_c < t_c + 0{,}384\ ms < t_s$$

Los valores por defecto en Contiki son:
t_i= 0.4 ms
t_c= 0.5 ms
t_s= 0.884 ms

Si bien la elección de t_{cycle} no tiene restricciones, por defecto tiene un valor de 125 ms. Para hacer aún más eficiente el mecanismo, ContikiMAC cuenta con phase-lock (mecanismo para establecer el ciclo de trabajo, mantiene lista de vecinos y fases para despertar al receptor), donde cada nodo guarda información sobre la fase de sus vecinos para poder enviar las tramas justo antes de que ellos se despierten.

2.4.3 Capa de enlace

La capa de enlace es clave para garantizar la transferencia de datos sin errores entre dos máquinas conectadas directamente por cable. Esta capa atiende las solicitudes provenientes de la capa de red y hace uso de los servicios ofrecidos por la capa física. Además, se encarga de controlar el flujo de datos y gestionar errores en la comunicación.
En el transcurso del proyecto, se emplea el protocolo MAC (Control de Acceso al Medio), definido en el estándar IEEE 802.15.4. En este contexto, el tamaño máximo de trama es de 128 bytes y se aceptan direcciones MAC tanto más largas de 64 bits como más cortas de 16 bits. La capa MAC utiliza el método de acceso al medio CSMA/CA no persistente, lo que significa que, si el canal de comunicación está libre, se procede con la transmisión; de lo contrario, se espera un tiempo aleatorio antes de intentar la transmisión nuevamente.

2.4.4 Capa de adaptación

La capa de adaptación se encarga de ajustar la transmisión de paquetes IPv6 sobre IEEE 802.15.4. Además, realiza funciones como la compresión de encabezados, la fragmentación de paquetes e informa a las capas superiores sobre las restricciones que requiere la capa MAC. Para este propósito, en nuestro proyecto se empleó el estándar 6lowPAN.
Dentro de Contiki, la implementación de 6lowPAN es conocida como "sicslowpan" y se encuentra en los archivos:

contiki/core/net/ipv6/sicslowpan.h
contiki/core/net/ipv6/sicslowpan.c

2.4.5 Capa de Red

La capa de red posibilita la interacción entre nodos distantes en una red. En la red inalámbrica de sensores se ha optado por IPv6, la cual es una evolución de IPv4. Su rasgo distintivo radica en que sus direcciones IP son de 128 bits, lo que permite una identificación única de dispositivos dentro de redes empleando este protocolo. Además, IPv6 ofrece una mayor capacidad de direcciones, fortaleciendo la seguridad y facilitando la conexión de un mayor número de dispositivos a la red global.

En cuanto al ruteo, en Contiki se implementa el *Routing Protocol for Low-power and Lossy Networks* (RPL) desarrollado por el grupo *Routing over Low-power and Lossy netw*orks (ROLL) perteneciente a la IETF.
RPL conecta los nodos a través de un árbol denominado *Destination Oriented Direct Acyclic Graph* (DODAG). El nodo que se encuentra en la raíz del árbol de comunicación se llama root, y se considera tráfico hacia arriba al que se dirige hacia el root, mientras que el que va desde el root se dice que es hacia abajo. La comunicación en un DODAG se puede dividir en 4 modos de operación (MOP) diferentes y es el root el que define qué MOP se usa:

1. Nodos no mantienen rutas hacia abajo.
2. Nodos no almacenan información de la ruta.
3. Nodos almacenan información de la ruta, pero no soporta multicast.
4. Nodos almacenan información de la ruta y soporta multicast.

En el modo 1 todos los nodos envían información al root, a nodos en el camino hacia el root o a otra red conectada al root. El modo 2 permite también el tráfico hacia abajo, pero requiere

que el ruteo lo haga el root. Esto quiere decir que el root tiene que especificar el camino completo hacia el nodo destino. Los nodos que no son root no almacenan información de la ruta y el tráfico en el DODAG siempre tiene que pasar por el root, ya que es el único capaz de rutear. Los modos 3 y 4 superan las restricciones anteriores, haciendo que todos los nodos almacenen información de las rutas. Esto quiere decir que el tráfico puede ir hacia arriba o hacia abajo. La diferencia entre los modos 3 y 4 es que el segundo soporta multicast.
La red se forma intercambiando diferentes tipos de mensajes entre los nodos:

- DIO (DODAG Information Object): contienen información de la configuración del DODAG y sirven para crear, mantener y descubrir el DODAG.
- DAO (Destination Advertisement Object): se usan para propagar información hacia arriba y completar las tablas de ruteo de los nodos padres. (Nodo seleccionado por un nodo, para enviar los mensajes hacia el nodo raíz con el menor costo posible.)
- DIS (DODAG Information Solicitation Message): sirven para solicitar mensajes DIO a los nodos vecinos.

Además de los mensajes anteriores, para formar las tablas de ruteo se usa el Rank. Este parámetro sirve para evaluar qué tan buenos son los enlaces y evaluar sus potenciales padres, para poder determinar cuál es el mejor camino hacia el root. El rank aumenta hacia abajo, y cuanto más pequeño sea el rank mejor es el camino para llegar al root, siendo 256 el menor valor del rank (es el que tiene el root). Surge así el concepto de padre preferido, que es el nodo vecino con el que se consigue el mejor camino para llegar al root.
En Contiki, lo módulos principales en la implementación de RPL son:

contiki/core/net/rpl/rpl-conf.c
contiki/core/net/rpl/rpl-dag-root.c
contiki/core/net/rpl/rpl-dag.c
contiki/core/net/rpl/rpl-of0.c
contiki/core/net/rpl/rpl-mrhof.c

2.4.6 Capa de Transporte

La capa de transporte es la encargada de la calidad del servicio (fiabilidad, control de flujo y corrección de errores). En TCP/IP se cuenta con dos posibles protocolos, Transmission Control Protocol (TCP) y User Datagram Protocol (UDP). TCP proporciona una forma de conexión confiable, ordenada, con confirmación y chequeo de errores. Por otro lado, UDP es un protocolo basado en la transmisión de datagramas, permitiendo el intercambio de datagramas por la red sin la necesidad de establecer previamente una conexión. Tampoco cuenta con confirmación ni control de flujo como TCP. Los datagramas pueden llegar al receptor desordenados y no se tiene certeza si fueron recibidos ya que tampoco cuenta con confirmación. Al utilizar UDP cualquier tipo de garantías para la transmisión de datos deben ser implementadas en capas superiores. Por otro lado, TCP garantiza que los datos sean entregados al destinatario y en el mismo orden en que fueron enviados. Además, proporciona control de flujo y de congestión de la red, siendo por estos y otros motivos, que TCP da soporte a muchas de las aplicaciones y protocolos de aplicación más populares de Internet.
A nivel de capa de transporte, en la red inalámbrica de sensores implementada se utilizó el protocolo UDP. La principal desventaja de TCP y las LLN, es que interpreta las pérdidas de segmentos como congestión, haciendo una reducción innecesaria de la velocidad de transferencia. Otro punto a tener en cuenta es que el encabezado de TCP es de 20 bytes,

mientras que en UDP es de 8 bytes (sólo envía puertos de origen y destino), generando paquetes UDP más pequeños y con menor riesgo de fragmentación. Finalmente, TCP provee confiabilidad a la transmisión mediante el envío de reconocimientos lo que, en la mayoría de las implementaciones, genera un tráfico extra en sentido opuesto al de los datos, provocando congestión y colisiones en las LLN.
En Contiki la implementación de TCP y UDP se encuentra en:

contiki/core/net/ip/tcpip.h
contiki/core/net/ip/tcpip.c
contiki/core/net/ip/udp-socket.h
contiki/core/net/ip/udp-socket.c
contiki/core/net/ip/tcpip-socket.h
contiki/core/net/ip/tcpip-socket.c

2.4.7 Capa de Aplicación

La capa de aplicación es la encargada de manejar protocolos de alto nivel, los cuales tienen en cuenta la representación, codificación y control de diálogo. El protocolo más utilizado es el protocolo de transferencia de hipertexto (HTTP).
Sin embargo, este protocolo en las redes con recursos limitados presenta, además del problema de grandes encabezados, la falta de soporte para mensajes multicast y mecanismos de suscripción. Para superar estas carencias la IETF desarrolló el *Constriñe Application Protocol* (CoAP).
El protocolo CoAP está pensado para comunicar dispositivos de bajos recursos que necesitan ser controlados o monitoreados a través de Internet. CoAP fue diseñado para poder traducir fácilmente los paquetes a HTTP, para ser integrado con la web.
Como se mencionó anteriormente, se utiliza UDP en capa de transporte ya que es muy simple y sólo agrega un encabezado con puertos de origen y destino, a diferencia de TCP que agrega *handshakes*, control de flujo y manejo de errores. UDP no provee una comunicación libre de errores, pero de ser necesario, CoAP proporciona un mecanismo de retransmisión. Los mensajes de CoAP, contienen el campo confirmable que solicita que se retransmitan los paquetes corruptos, logrando obtener de esa forma, una comunicación confiable usando UDP.
Algunas de las principales características de CoAP son:

- Protocolo RESTful con mapeo sencillo a HTTP.
- Bajo overhead en el encabezado.
- Baja complejidad en el parseo o análisis sintáctico.
- Soporte para descubrir recursos usando un servicio conocido.
- Suscripción simple a recursos.

Como el servidor no tiene memoria, y por ende no recuerda ninguna de las solicitudes previas, utilizando CoAP los recursos son accedidos y manipulados usando protocolos de capa de aplicación basados en diálogos cliente-servidor del tipo solicitud-respuesta. Este tipo de comunicación forma el modelo REST. Dicho modelo no especifica el protocolo a usar, pero solicita que el mismo tenga comandos POST, GET, PUT, DELETE. Las aplicaciones que usan REST se les conoce como RESTful, y en el modelo REST aplicado a IoT o M2M los recursos son identificados por un identificador de recursos uniforme (URI), el cual es una cadena de caracteres que identifica los recursos de una red de forma unívoca.

Como se mencionó, CoAP es un protocolo RESTful, por lo cual utiliza los comandos REST para la comunicación.
A continuación, se detallan los comandos del protocolo:

- **GET:** Comando con el cual se recupera una representación de la información que corresponde al recurso identificado por la URI.
- **POST:** Envía una acción al recurso. Esta acción puede ser crear, modificar o eliminar un recurso.
- **PUT:** solicita que el recurso identificado por la URI se actualice o se cree.
- **DELETE:** Elimina el recurso identificado por la URI.

Los comandos anteriores son una pieza clave para la comunicación de los nodos sensores, realizando la configuración de recursos y solicitud de información con estos comandos.
En Contiki, la implementación de CoAP se llama erbium y fue implementada como una aplicación, por lo tanto, sus archivos fuentes se encuentran en el directorio *apps/er-coap*. Esta aplicación utiliza archivos de una aplicación denominada rest-engine que implementa un motor de REST.

2.5 Presupuesto de enlace inalámbrico

Un presupuesto de enlace son los cálculos que nos permiten determinar si la transmisión inalámbrica de datos va a ser exitosa o no.
En la Figura 2.5 tenemos un enlace inalámbrico entre dos puntos. Dicho enlace está caracterizado (a fines de los cálculos) por la distancia entre nodos d, la altura de las antenas h, y el radio de Fresnel r.

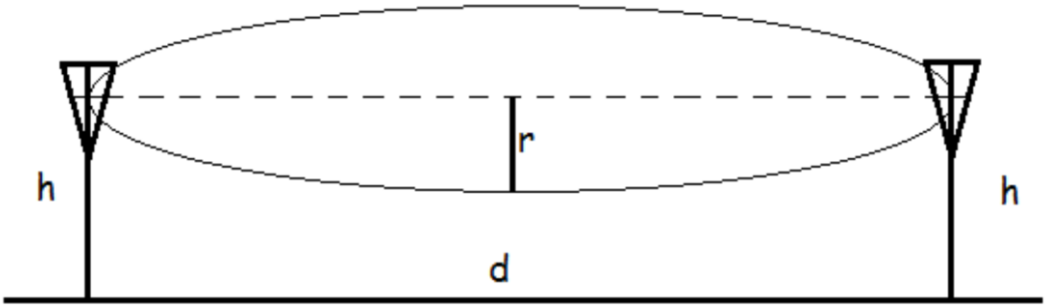


Figura 2.5: Enlace Inalámbrico entre dos puntos

Cálculo de la potencia recibida P_r en dBm:

$$P_r[dBm] = P_t[dBm] + G_t[dBi] + G_r[dBi] - (FSL[dB] + A_t[dB] + A_r[dB])$$

Donde:

P_t: Potencia del transmisor en dBm.
G_t: Ganancia de la antena del transmisor en dBi.
G_r: Ganancia de la antena del receptor en dBi.
FSL: Pérdidas en el aire en dB.
A_t: Otras pérdidas en dB en el transmisor.
A_r: Otras pérdidas en dB en el receptor.

Cálculo del FSL en dB:

$$FSL[dB] = 20log(f[Mhz]) + 20log(d[Km]) + 32{,}44$$

Donde:

f: frecuencia de la portadora en MHz.
d: distancia entre los nodos en km.

Cálculo de los márgenes:

$Margen\ RX = Pr - RSL$; donde el RSL es la sensibilidad del receptor.

$Margen\ RX\ mínimo = 5{,}25dBm + 11log(d[Km])$

Se desea que $Margen\ RX > Margen\ RX\ mínimo$

Cálculo del radio de Fresnel:

$r[m] = 17{,}32 * \sqrt{\frac{(d1[Km]*d2[Km])}{(d[Km]*f[Ghz])}}$ (Un 60% es aceptable).

Donde $d[Km] = d1[Km] + d2[Km]$
A mitad de distancia, el radio de Fresnel será mayor, de manera que:

$\frac{d}{2} = d1[Km] = d2[Km]$

La fórmula queda simplificada a:

$$r[m] = 17{,}32\sqrt{d\frac{[Km]}{(4*f[Ghz])}}$$

Cálculo de la altura de las antenas:

$Altura_{óptima} = r$

$Altura_{mínima} = 0{,}6 * r$

3 Desarrollo del Proyecto

3.1 Introducción

El proyecto consistió en el diseño e implementación de una red inalámbrica de sensores para el monitoreo de variables físicas en áreas rurales. El sistema es capaz de monitorear periódicamente dichas variables, para luego enviar toda la información a Internet y que la misma sea visualizada.
La red inalámbrica de sensores fue pensada para ser utilizada en cualquier tipo de escenario del ámbito rural, por lo tanto, a los nodos se le pueden conectar diferentes tipos de sensores de acuerdo a la problemática (sistema de riego, control de plagas, monitoreo de bebederos, etc). Para nuestro proyecto en particular sólo se incorporaron los siguientes sensores:

- Sensor de temperatura del suelo.
- Sensor de humedad del suelo.
- Sensor de temperatura ambiente.
- Sensor de humedad ambiente.
- Sensor de contacto o fin de carrera.

El sistema monitorea, además, el nivel de carga de la batería y estados de algunos componentes que conforman el nodo sensor.
Otras variables que el sistema podría llegar a monitorear serían:

- Nivel de Agua de un bebedero.
- Estado de una tranquera (abierta/cerrada).
- Estado de una cerca.
- Etc.

Las variables y datos medidos se envían a una plataforma IoT llamada Thingspeak para que puedan ser visualizados de forma remota desde cualquier dispositivo con conexión a Internet, o bien, desde un Smartphone por medio de la aplicación Thingview.

3.2 Esquema general de funcionamiento

En la Figura 3.1 se muestra el sistema completo de la red inalámbrica de sensores que está constituido por los siguientes dispositivos:

- **Nodos Sensores:** Son los nodos 1, 2 y 3 los cuales recogen la información obtenida por los sensores y la transmiten a la Estación Base.
- **Router de Borde:** Es el encargado de establecer la red IPv6 y al mismo tiempo hace de puerta de enlace de todos los nodos sensores.
- **Estación Base:** Elemento que recoge y procesa todos los datos provenientes de la red de sensores y los sube a Internet mediante la red de telefonía móvil.

En la Figura 3.1 se muestran los dispositivos con sus respectivas direcciones IPv6.
Cada uno de estos dispositivos cumplen una determinada tarea dentro de la red, las cuales se explican con mayor detalle en las siguientes subsecciones.

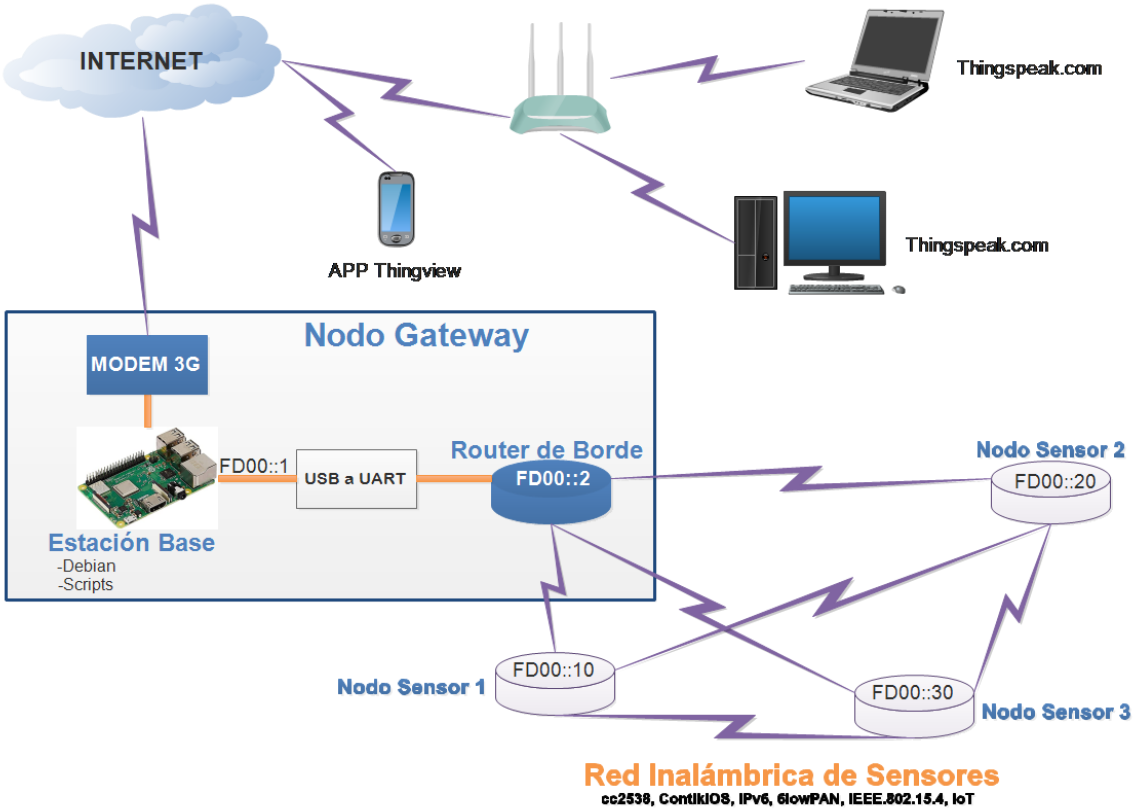


Figura 3.1: Sistema Completo de la Red Inalámbrica de Sensores

Los nodos sensores pueden medir temperatura y humedad del suelo, así como la temperatura y humedad ambiente. Además de las variables antes mencionadas, este nodo puede medir el estado de carga de la batería y conocer el estado de algunos componentes que son parte del nodo (relé, pulsadores, etc).
Cada nodo sensor está equipado con sensores de interfaz analógica o digital, montados en una placa PCB diseñada en base a 2 tipos de módulos: el CC2538EM y el CC2538+CC2592; ambos módulos poseen un microcontrolador cc2538 de Texas Instruments.
El sistema hace uso de la pila de protocolos presentada en la Sección 2.4. Esta pila de protocolos está especialmente diseñada para redes inalámbricas de sensores, siendo ligera y adaptable a las limitaciones y las pérdidas presentes en estas redes. A través de estos protocolos, cada nodo establece rutas hacia otros nodos, buscando el mejor camino para enrutar los paquetes. En la Tabla 2.1 se detalla la pila de protocolos seleccionada.
En los SoC (System on a Chip) cc2538, se ejecuta una aplicación programada utilizando el sistema operativo ContikiOS. Cada sensor conectado al nodo realiza periódicamente lecturas, almacenándolas en la memoria del cc2538. Posteriormente, estas lecturas se envían hacía la Estación Base (Raspberry Pi) cuando esta última lo solicita (ver Figura 3.2).

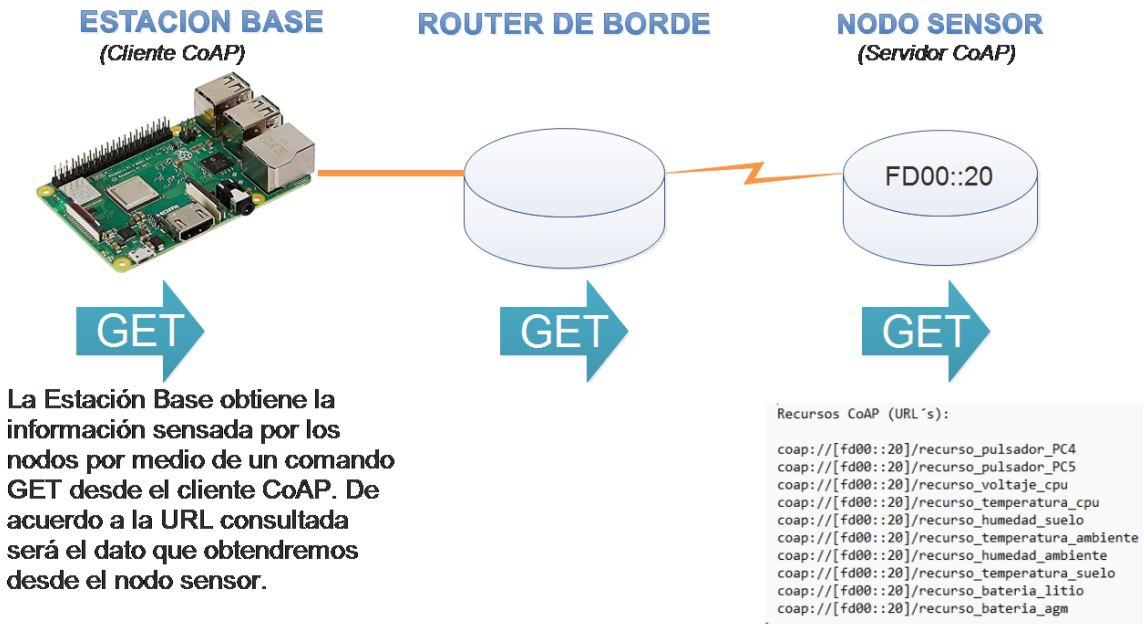


Figura 3.2: Solicitud de la información al nodo sensor

La comunicación entre la Estación Base y el nodo sensor se logra utilizando el protocolo CoAP. La Estación Base ejecuta un cliente CoAP mientras que cada nodo ejecuta un servidor CoAP con recursos que corresponden a las variables medidas. Toda la comunicación pasa por el Router de Borde que cumple la función de Gateway de toda la red inalámbrica de sensores. Una vez que la Estación Base recibe el dato, lo almacena y luego lo envía a Thingspeak (Ver Figura 3.3).

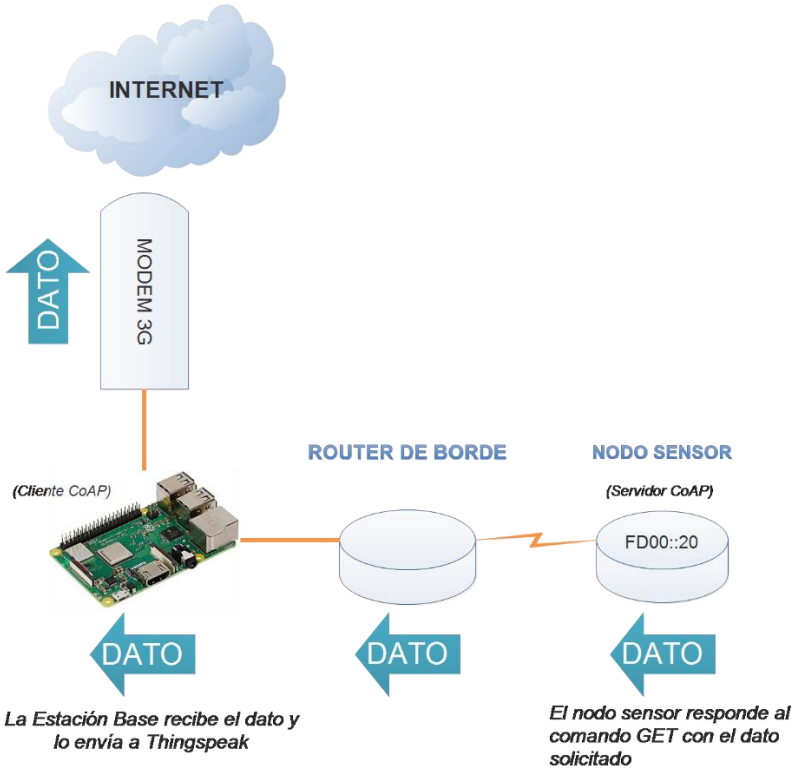


Figura 3.3: Respuesta del nodo sensor

Este sistema es capaz de funcionar de forma automática, o ser manejado por un usuario interactuando directamente con la Estación Base ejecutando comandos CoAP de forma manual. De esta última forma, por ejemplo, podemos encender un relé ejecutando un comando POST o modificar el período de muestreo de los sensores como se indica en la Figura 3.4. Por otro lado, utilizando el comando GET, por ejemplo, es posible conocer el voltaje y la temperatura del MCU (estos últimos dos también los agregamos como recursos del servidor CoAP en cada uno de los nodos sensores).

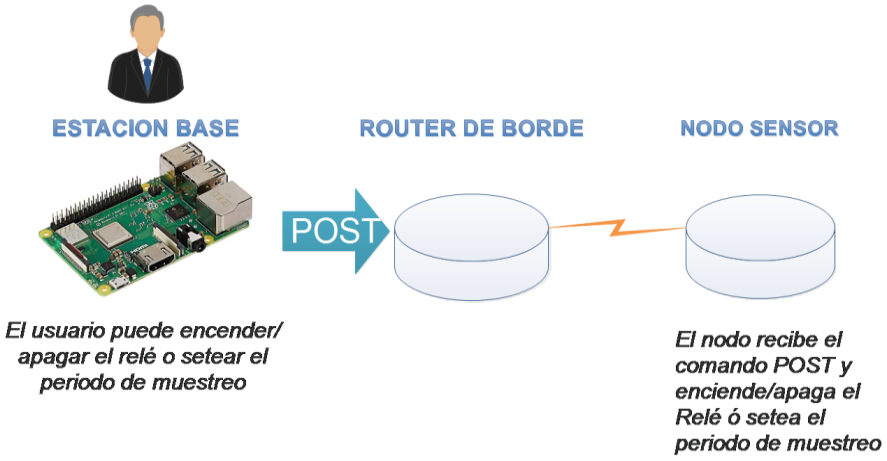


Figura 3.4: Comando ejecutado por un usuario para modificar un parámetro del nodo sensor

3.2.1 Nodo sensor

Está compuesto por un módulo basado en el cc2538, sensores, pulsadores, relé y batería.
El nodo sensor cumple con la función de realizar mediciones (cada 5 minutos) y observaciones, para luego enviarlas al Nodo Gateway cuando éste las solicite.
Las mediciones son las que se obtienen de los sensores y un puerto ADC del MCU (este puerto entrega el nivel de carga de la batería). Mientras que las observaciones corresponden a los estados de los pulsadores y al estado del relé.
Los nodos que conforman la red, intercambian mensajes RPL entre ellos, manteniendo así las tablas de ruta de cada nodo actualizada incluso cuando hay cambios en la topología de la red. Cualquier nodo que esté en la red podrá comunicarse con el Nodo Gateway sin importar la distancia a la que se encuentre del mismo.
Por otro lado, cada uno de los nodos implementa un servidor CoAP con varios recursos.

3.2.2 Nodo Gateway

Este dispositivo está compuesto por varios dispositivos que cumplen distintas funciones.

Router de Borde

El Router de Borde además de actuar como raíz DODAG, es el encargado de proporcionar conectividad encaminando paquetes de datos de una red a otra; en este caso, de la red inalámbrica de sensores (WSN) a Internet.
Este dispositivo se encuentra conectado a la Estación Base por medio de un túnel (Tunslip) mediante una interfaz serie de USB a UART.

Estación Base

Este dispositivo se encarga de enviar la información recopilada por los nodos sensores a Internet. Implementa un cliente CoAP que le permite no sólo solicitar información de los recursos asociados a cada nodo sensor (servidor CoAP) por medio de comandos CoAP, sino también de modificar el tiempo de muestreo de los nodos y encender un relé.
Este dispositivo tiene conectado un modem 3G en su puerto USB que le permitirá conectarse a Internet utilizando la red de datos móviles. Además, como se mencionó en el apartado anterior está conectado al Router de Borde. Para el proyecto la Estación Base es representada por una Raspberry Pi.

Comunicación entre el Router de Borde y la Raspberry Pi

La comunicación entre ambos dispositivos se logra gracias a la herramienta llamada tunslip6 de ContikiOS (*contiki/tools*). Tunslip es una herramienta que se usa para permitir tráfico IP entre un host y otro elemento de red, generalmente un Router de Borde a través de una línea serie.
Tunslip crea una interfaz de red virtual llamada *tun* en el lado del host (Raspberry Pi), y utiliza SLIP (Protocolo de Internet de línea Serie) para hacer un encapsulado de enlace tipo UNSPEC y pasar tráfico IP hacia y desde el otro lado de la línea serie.
El elemento de red que se encuentra en el otro lado de la línea serie hace un trabajo similar con su interfaz de red. La interfaz *tun* puede usarse como cualquier interfaz de red real: enrutamiento, reenvío de tráfico, análisis de Wireshark, etc.

Modem 3G

Este dispositivo le otorga a la Raspberry Pi acceso a Internet mediante la tecnología 3G a través de una de las redes de un operador de telefonía móvil (en este proyecto se usó la red de Movistar).
El Protocolo punto a punto (PPP) es utilizado para establecer la conexión a Internet de la Raspberry Pi con su ISP (Proveedor de servicio de Internet) a través del módem 3G.

3.2.3 Enrutamiento y formación de la Red de Sensores

Para el enrutamiento y formación de la Red Inalámbrica de Sensores utilizamos el protocolo RPL (Routing Protocol for Low-Power and Lossy Networks).

Proceso de formación del DODAG (Destination-Oriented Directed Acyclic Graph):

RPL crea y mantiene una topología similar a un árbol llamada DODAG que se origina en el Router de Borde (Root o raíz del DODAG).

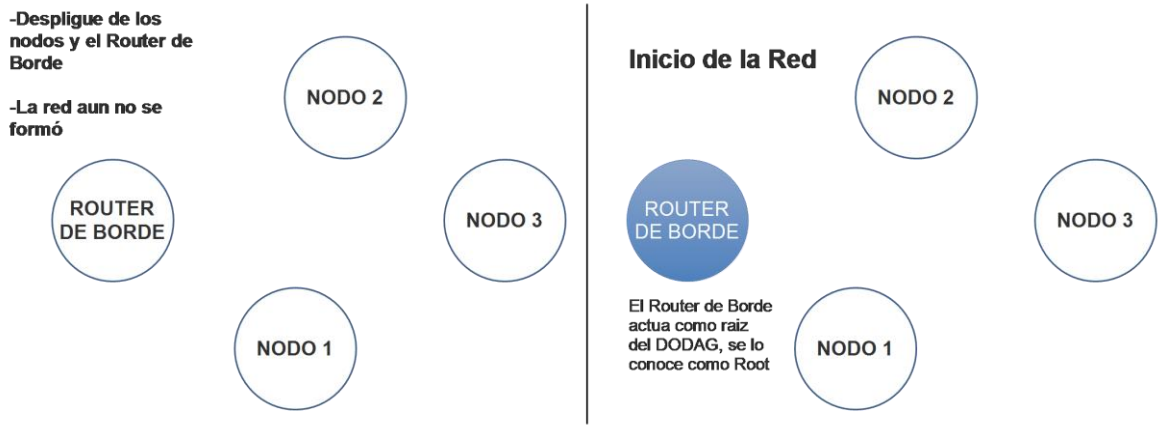


Figura 3.5: Inicio de la red

Los nodos de la red intercambian mensajes RPL (DIO, DAO y DIS) con el fin de descubrir y mantener la información sobre vecinos y su posición en la topología.

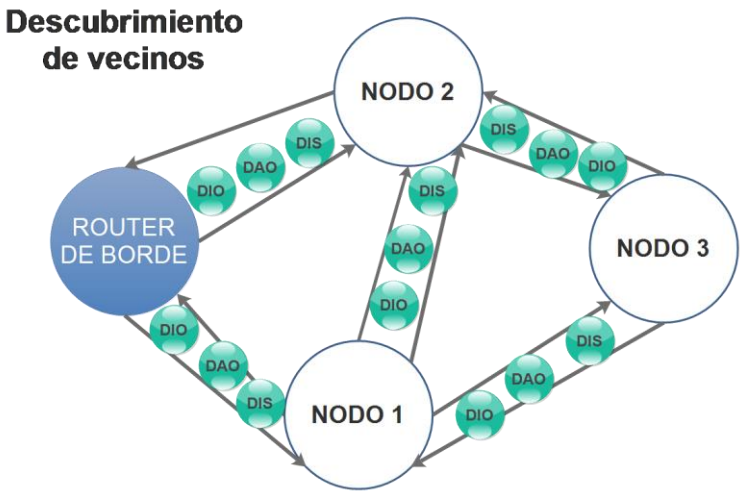


Figura 3.6: Intercambio de mensajes para el descubrimiento de vecinos

Para establecer la ruta más óptima hacia el root se tienen en cuenta métricas respecto al costo de la ruta y la calidad de la conexión. De esta manera todos los nodos quedan enlazados a través de un árbol DODAG (Ver Figura 3.7).

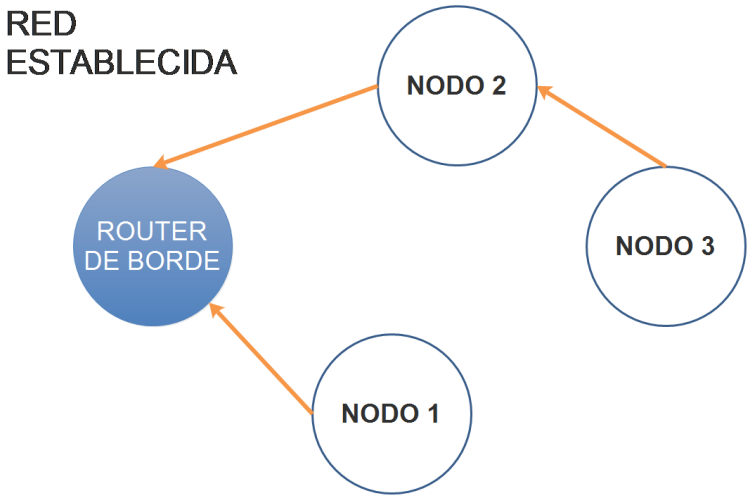


Figura 3.7: Árbol DODAG

Enrutamiento de datos

Los nodos determinan la ruta más óptima hacia el root para reenviar los paquetes de manera eficiente. A continuación, se muestra el flujo de datos entre los nodos, el mismo recorre la ruta establecida por el DODAG.

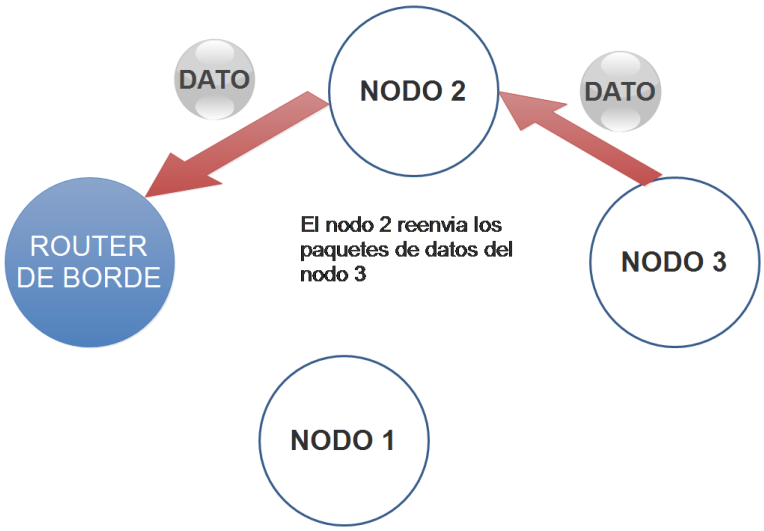


Figura 3.8: Envió de paquete del nodo 3 al Router de Borde

Mantenimiento de la topología

Los nodos intercambian información de manera periódica para mantener actualizada la topología de la red. Esta capacidad les permite adaptarse a cambios en la red, como la incorporación de nuevos nodos o la salida de alguno de ellos. Por ejemplo, si se retira el nodo 2, el nodo 3 buscará una nueva ruta hacia el Router de Borde.

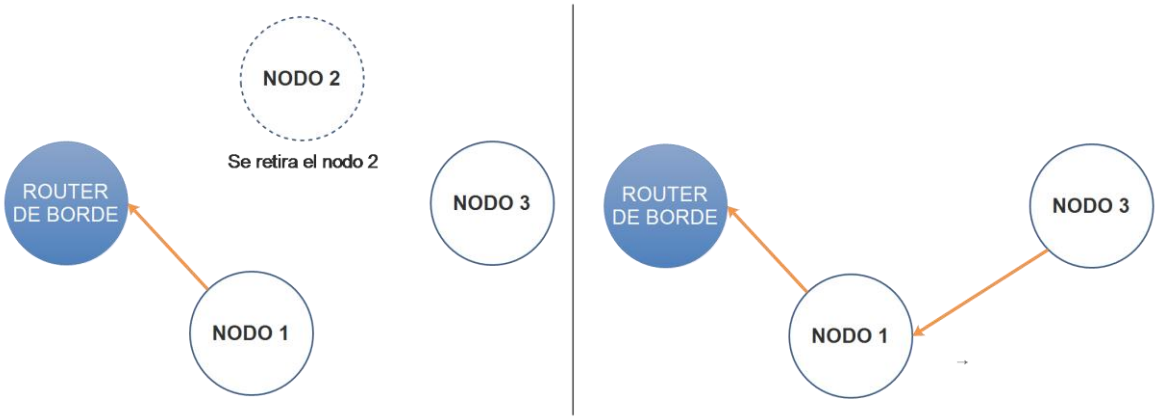


Figura 3.9: Reconfiguración del DODAG

3.2.4 Thingspeak y Thingview

Thingspeak

Thingspeak es una plataforma de Internet de las cosas (IoT) que posibilita la recolección y almacenamiento de datos provenientes de sensores en la nube, además de facilitar el desarrollo de aplicaciones IoT. Esta plataforma también ofrece herramientas para analizar y visualizar datos en MATLAB y actuar sobre los datos, aunque esto último no fue implementado.
Los datos de los sensores pueden ser enviados desde diferentes plataformas, en nuestro proyecto utilizamos una Raspberry Pi (Estación Base).

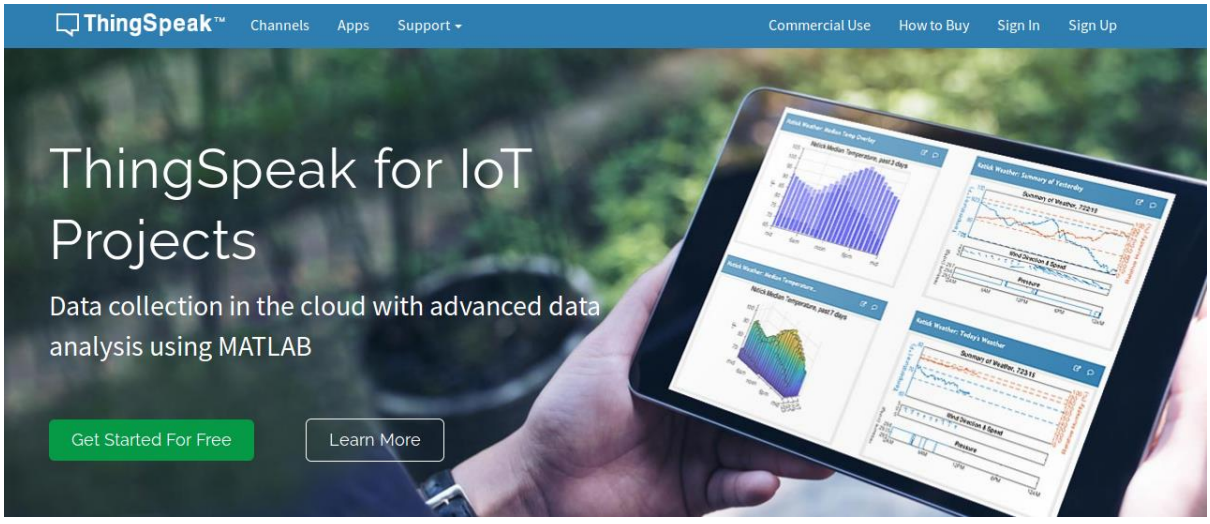


Figura 3.10: www.thingspeak.com

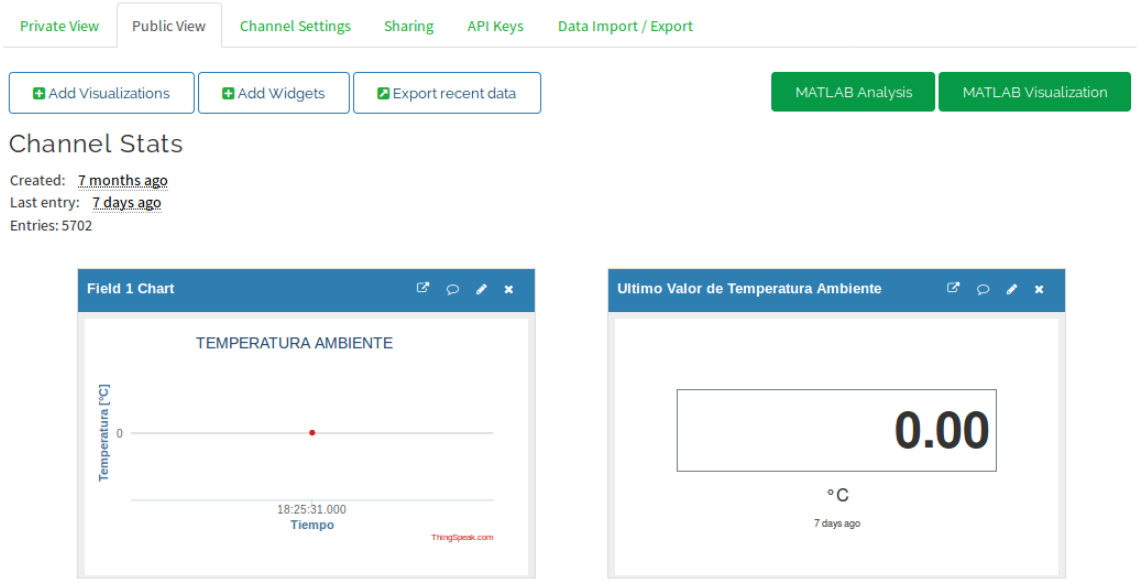


Figura 3.11: Visualización de un canal de Thingspeak

La estructura de Thingspeak es:

- Canales (Channels): los datos que recogemos en los dispositivos se guardan en canales. Cada canal tiene su propio ID (identificador de canal).
- En cada canal se dispone de una serie de campos para guardar datos, así como otra información adicional.
- Los canales pueden ser públicos o privados.
- Dentro de cada canal podemos añadir visualizaciones o Widgets.
- Los datos del canal se pueden importar o exportar.
- En la pestaña de API keys está la información con las contraseñas (API Keys) para usar con las APIs.

Para poder visualizar los datos en Thingspeak creamos una cuenta gratuita (también puede ser de licencia paga) la cual nos facilitó 4 canales con 8 campos de datos cada uno.
A continuación, se muestra una captura de la pantalla principal de nuestros canales.

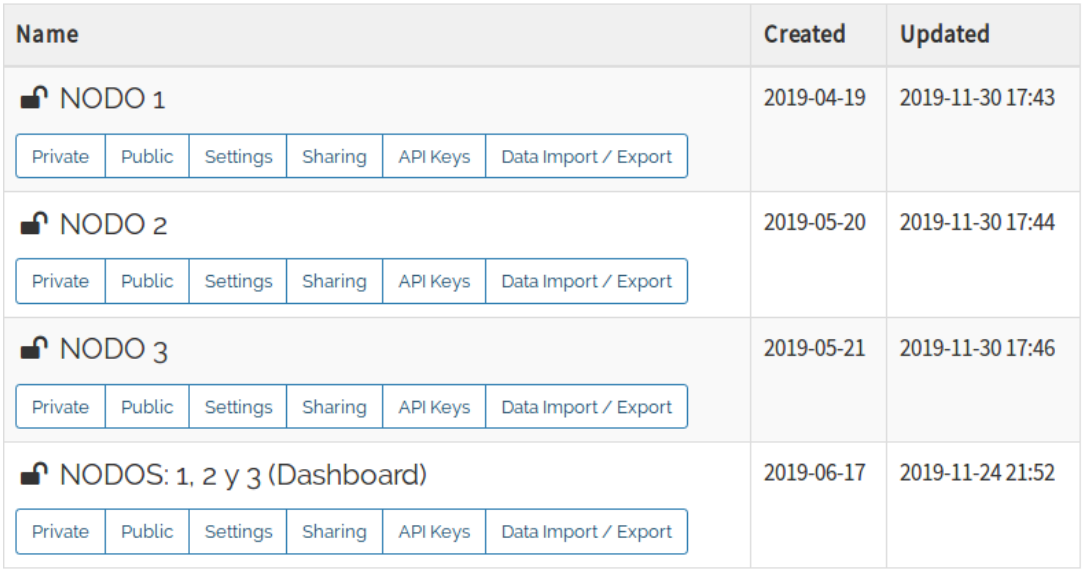


Figura 3.12: Visualización de nuestros canales de Thingspeak

Puede observarse que se utilizaron los 4 canales. En la tabla 3.1 se da una breve descripción de cada uno de los canales utilizados.

Canal	Nombre de Canal	Descripción
1	NODO 1	Muestra la información del nodo 1
2	NODO 2	Muestra la información del nodo 2
3	NODO 3	Muestra la información del nodo 3
4	NODOS: 1, 2 y 3 (Dashboard)	Muestra la información de todos los nodos.

Tabla 3.1: Descripción de nuestros canales de Thingspeak

Thingview

Thingview es una aplicación móvil de Android que permite visualizar canales de Thingspeak con sólo ingresar el ID del canal.
La versión actual admite gráficos de líneas y columnas.

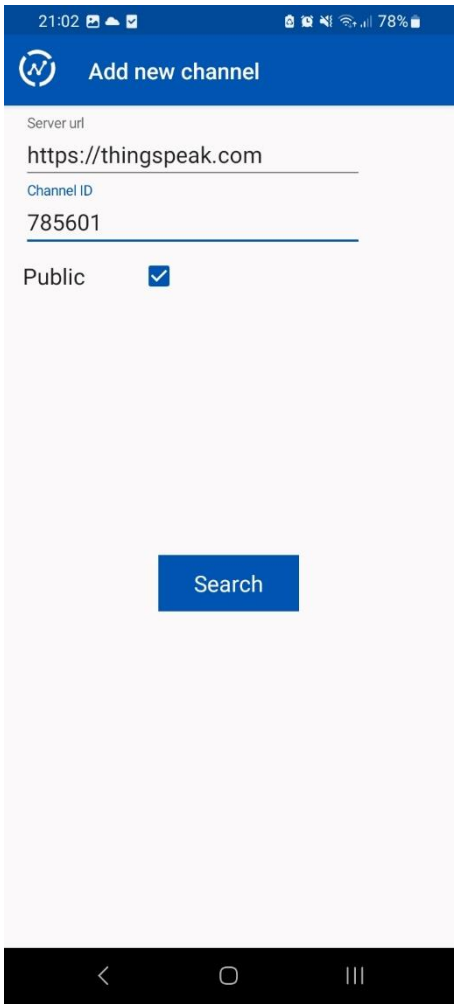


Figura 3.13: Pantalla de ingreso a un canal

Figura 3.14: Visualizaciones de un canal Thingspeak

Esta aplicación fue utilizada de forma opcional para poder visualizar los valores medidos por cada nodo sensor desde un dispositivo Android con conexión a Internet.

3.3 Hardware

3.3.1 Introducción

Los nodos de la red están basados en el microcontrolador cc2538, lo que permite ejecutar una aplicación programada en ContikiOS junto con los protocolos de comunicación definidos para este proyecto. Dada la variedad de variables medidas, se requiere integrar sensores en el MCU para realizar estas mediciones. Cada nodo, para formar parte de la red, debe tener una radio y una antena. Por ende, se diseñaron los nodos basados en módulos que contienen al cc2538 y a su vez tienen resuelto en el PCB la etapa de RF.
Se diseñó un PCB para integrar este módulo, con borneras para los sensores, proveer alimentación, elementos de programación del MCU y componentes para pruebas de funcionamiento. Además, se agregaron elementos adicionales en previsión de un diseño reutilizable.

Durante el diseño, se priorizó el bajo consumo del sistema y la versatilidad para lograr un producto reutilizable. Fue esencial considerar elementos que permitieran la programación y depuración del MCU. En el PCB, se incluyó una etapa de alimentación para el módulo y los sensores, los cuales se conectan mediante borneras, facilitando su reemplazo en caso de daño. Además, se tuvo en cuenta que los nodos estarían expuestos a la intemperie, por lo que se optó por utilizar una caja estanca para su protección.
El proceso de fabricación de los nodos comenzó con la selección de los componentes del circuito y los sensores, seguido por el diseño del PCB. Posteriormente, se creó un prototipo sobre el cual se llevaron a cabo pruebas de funcionamiento del hardware.
Durante el diseño, se tuvo en cuenta el consumo energético reducido y se buscó la versatilidad, lo que resultó en un producto reutilizable. Se consideró esencial incluir elementos que facilitaran la programación y la depuración del MCU.

3.3.2 Diseño de Hardware

En la construcción de los nodos, se priorizó un diseño versátil, reutilizable y enfocado en la eficiencia energética. El proceso de diseño comenzó con la selección del MCU, la radio y los sensores, seguido por el diseño de un PCB que cumpliera con las especificaciones deseadas. Este PCB no solo integró los sensores, sino también los elementos necesarios para la programación.
Cabe aclarar que el diseño del PCB se empleó tanto para los nodos como para el Router de Borde.

Selección del microcontrolador y radio

Para el desarrollo de la aplicación y las pruebas de hardware, se empleó el kit de desarrollo CC2538DK de Texas Instruments. Este kit permite la creación de aplicaciones cuando no se dispone del hardware definitivo. Está compuesto por dos plataformas de desarrollo SmartRF06 y dos módulos de evaluación CC2538EM. Estos módulos integran el SoC con una antena, elementos auxiliares y puertos de entrada y salida, conectándose a la plataforma SmartRF06 para formar un kit completo de desarrollo y evaluación. Esto proporciona herramientas para grabar, ejecutar y depurar programas en el cc2538.
Para el funcionamiento correcto del SoC, se requieren circuitos externos, como un circuito de adaptación de señales de radio, conexión a una antena y un oscilador externo. Debido a que soldar el chip y diseñar su circuito externo excedía el alcance del proyecto, se empleó un módulo que resuelve esta conexión.
En el diseño del PCB se utilizó un módulo genérico (de origen chino) denominado simplemente CC2538+CC2592, el cual integra el SoC seleccionado y un amplificador de radio PA/LNA cc2592 de Texas Instruments. Este amplificador proporciona una ganancia de transmisión de hasta +22 dBm (PA) y una ganancia de recepción (LNA) de 12 dB. Dado que el cc2538 tiene una sensibilidad de recepción de -97 dBm, con el cc2592 se logra una sensibilidad de recepción de -103 dBm.
Otro módulo disponible, por ejemplo, es el CC2538EM de Texas Instruments, pero este módulo pertenece al kit de desarrollo CC2538DK y no se pueden comprar en grandes cantidades ya que su costo es muy elevado. Sin embargo, utilizamos este módulo en conjunto con una placa adaptadora para poder adicionarlo a nuestro PCB y así conformar uno de los nodos sensores.
Las ventajas del módulo genérico CC2538+CC2592 sobre el módulo CC2538EM de Texas Instruments son las siguientes:

- Lleva integrado un amplificador cc2592 que permite extender el alcance de la radio.
- Posibilidad de utilizar una antena externa o utilizar la integrada en la misma placa PCB.
- No incorpora leds ni botones.
- Precio más bajo.
- Dimensiones más reducidas y es más fácil de soldarlo a una placa PCB, no requiere la utilización de conectores especiales.

Por las anteriores ventajas nombradas fue que se eligió el módulo CC2538+CC2592 para el desarrollo de casi todos los nodos de la red.

Elementos para la programación: JTAG y UART

Para realizar la programación del MCU se colocaron dos pineras correspondientes a cada interfaz de programación, JTAG y UART.

Sensores

Cada uno de los nodos sensores cuenta con borneras en el PCB para conectar un total de 3 sensores: uno de temperatura y humedad ambiente, otro de temperatura del suelo y un tercero de humedad del suelo. Se hizo hincapié en minimizar la tensión de alimentación al nodo para reducir el consumo de energía del sistema. En este sentido, se consideró cuidadosamente la tensión de alimentación necesaria para los sensores, buscando que fuera lo más baja posible.
La selección de los sensores no fue muy difícil ya que el sistema está pensado para monitorear variables del ámbito rural, sin importar la problemática en cuestión. Por lo tanto, se emplearon los sensores más económicos y de mayor disponibilidad en el mercado.
Los sensores elegidos fueron suficientes como para poder realizar las pruebas y ensayos necesarios que nos permitieron evaluar la performance del sistema.
Por otro lado, para conocer el estado de una tranquera (por ejemplo) no se utilizó ningún tipo de sensor, simplemente se consulta por el estado de un pin GPIO (*General Purspose Input/Output*) del micro (0 a 1) y de acuerdo al nivel lógico se determina si la tranquera está abierta /cerrada. Para simular los niveles lógicos se colocaron pulsadores.
De todas maneras, si en un futuro se elige una aplicación específica (ej. sistema de riego, control de cultivos, etc) estos pueden ser reemplazados por otros que sean necesarios para la aplicación.
A continuación, se describen los sensores utilizados para este proyecto.

Sensor de Temperatura de Suelo: DS18B20

Este sensor proporciona lecturas de temperatura de 9 a 12 bits (configurables) a través de una interfaz de 1 hilo (1-Wire), de modo que sólo se necesita conectar un cable (y tierra) desde un microcontrolador central. Compatible con sistemas 3.0 – 5.5 V.
Especificaciones:

- Voltaje de funcionamiento: 3.0 ~ 5.5 V
- ± 0,5 °C de Precisión para un rango de -10 °C a +85 °C
- Rango de temperatura útil: -55 a 125 °C
- Resolución seleccionable de 9 a 12 bits
- Utiliza interfaz 1-Wire, requiere solo un pin digital para la comunicación
- ID único de 64 bits grabado en un chip
- Varios sensores pueden compartir un pin
- Sistema de alarma de límite de temperatura

- El tiempo de consulta es inferior a 750 ms

Sensor de humedad del suelo: Higrómetro YL-69
Este sensor se utiliza generalmente para detectar la humedad en la tierra o arena o cualquier sustrato permeable a la humedad y que se pueda enterrar.
El sensor está conformado por dos piezas o placas y un cable que las une: la placa electrónica YL-38 (a la izquierda) y la sonda con dos patas, que detecta el contenido de agua YL-69 (a la derecha).
El módulo YL-69 es un sensor de humedad de suelo que utiliza la conductividad entre dos terminales para determinar ciertos parámetros relacionados a agua, líquidos y humedad.
Tiene la capacidad de medir la humedad del suelo. Aplicando una pequeña tensión entre los terminales del módulo YL-69 hace pasar una corriente que depende básicamente de la resistencia que se genera en el suelo y ésta depende mucho de la humedad. Por lo tanto, al aumentar la humedad la corriente crece y al bajar la corriente disminuye.
Especificaciones:

- Voltaje de entrada: 3.3 ~ 5 V
- Voltaje de salida: 0 ~ 4.2 V
- Corriente: 35 mA
- VCC: Tensión de Alimentación
- GND: Tierra
- AO: Salida analógica que entrega una tensión proporcional a la humedad. Puede ser medida directamente por un puerto analógico en un microcontrolador.
- DO: Salida digital; este módulo permite ajustar cuando el nivel lógico en esta salida pasa de bajo a alto mediante el potenciómetro.

Sensor de Humedad y Temperatura del aire: DHT22
El DHT22 es un sensor digital de temperatura y humedad relativa de buen rendimiento y bajo costo. Integra un sensor capacitivo de humedad y un termistor para medir el aire circundante, y muestra los datos mediante una señal digital en el pin de datos (no posee salida analógica). Utilizado en aplicaciones de control automático de temperatura, aire acondicionado, monitoreo ambiental en agricultura y más.
Especificaciones:

- Voltaje de Operación: 3V – 6 V
- Rango de medición de Temperatura: -40 a 80 °C
- Precisión de Medición de Temperatura: < ± 0.5 °C
- Resolución Temperatura: 0.1 °C
- Rango de medición de Humedad: de 0 a 100 % RH
- Precisión de medición de Humedad: 2 % RH
- Resolución de Humedad: 0.1 %RH
- Tiempo de sensado 2 seg.
- Modelo: AM2302
- Carcasa de plástico blando

Raspberry Pi

Para este proyecto el dispositivo utilizado como estación base es una Raspberry Pi. Esta es una plataforma hardware del tamaño de una tarjeta de crédito que permite ejecutar sistemas operativos mediante una tarjeta SD/Micro SD (según el modelo). Los sistemas operativos

más comunes para esta placa son distribuciones Linux, pero también existe Windows 10 IoT para la Raspberry Pi 2 en adelante.
La Raspberry Pi 3 Model B+ es la revisión final en la gama Raspberry Pi 3 y es la utilizada para actuar de Estación Base.
A continuación, se muestran las especificaciones:

- CPU + GPU: Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4 GHz
- RAM: 1GB LPDDR2 SDRAM
- Wi-Fi + Bluetooth: 2.4 GHz y 5GHz IEEE 802.11.b/g/n/ac, Bluetooth 4.2, BLE
- Ethernet: Gigabit Ethernet sobre USB 2.0 (300 Mbps)
- GPIO de 40 pines
- HDMI
- 4 puertos USB 2.0
- Puerto CSI para conectar una cámara.
- Puerto DSI para conectar una pantalla táctil
- Salida de audio estéreo y vídeo compuesto
- Micro-SD
- Power-over-Ethernet (PoE)

Modem 3G- Huawei E176

El módem utilizado es el HUAWEI E176 HSDPA USB Stick. Este es un módem USB móvil de Banda Ancha que ofrece acceso a Internet. Para ello se le debe insertar una tarjeta SIM y conectar el módem a cualquier PC o dispositivo portátil.
También, cuenta con una ranura para la conexión de una antena externa para ganar mayor señal. Tiene una velocidad de transferencia de datos de 7.2 Mbps de enlace descendente y 5.6 Mbps de enlace ascendente.
Para el proyecto se insertó al módem una tarjeta SIM de Movistar y se conectó el módem a un puerto USB de la Raspberry Pi para brindarle conexión a Internet.

3.4 Alimentación de los dispositivos de red

3.4.1 Alimentación de los nodos sensores

Los nodos sensores se alimentan con un voltaje de entrada de 5 V. Con este nivel de voltaje alimentamos varios componentes que forman parte del circuito del nodo.
La forma en que se genera el voltaje de alimentación se produce por medio del siguiente sistema:

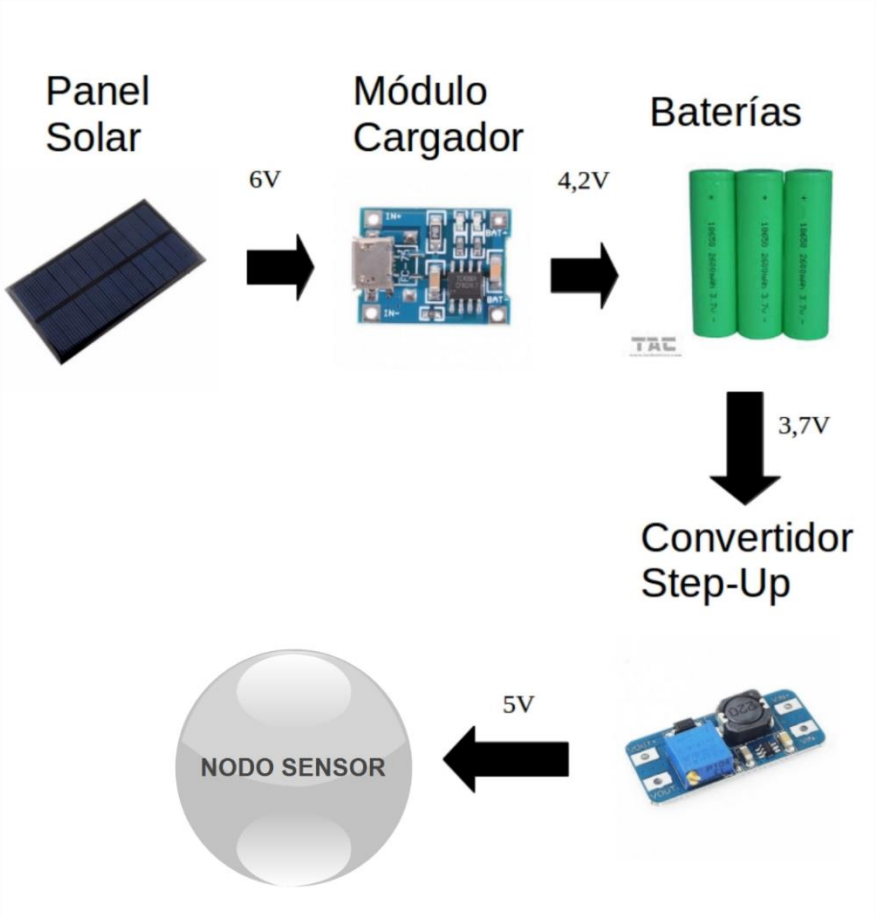


Figura 3.15: Sistema de alimentación para el Nodo Sensor

Durante el día (cuando se recibe luz solar), el panel solar genera una tensión de 6 V que sirve para alimentar al módulo cargador de batería FC-75. Este módulo carga las baterías de Litio hasta alcanzar los 4,2 V (voltaje de corte de carga) y se detiene hasta que la carga de las baterías se reduzca a 2,5 V, ya que el módulo cuenta con un sistema de protección contra sobre-descarga.
Luego, se utiliza un convertidor step-up MT3608 que eleva la tensión que entrega la batería a 5 V. Esté convertidor CC-CC tiene una tensión mínima de entrada de 2 V, por tanto, si la batería esta descargada (se encuentra debajo del voltaje nominal) la salida del convertidor será de 5 V siempre.
En ausencia de sol el sistema seguirá alimentado por las baterías.

3.4.2 Alimentación del Nodo Gateway

El voltaje de alimentación para este nodo es de 12 V. El sistema de alimentación cuenta con un panel solar, un regulador (o controlador de carga solar), un convertidor step-down y una batería AGM de 12 V. Claramente, este voltaje de alimentación mayor al de un nodo sensor se debe a que el Nodo Gateway está compuesto por varios componentes.
El sistema implementado es el siguiente:

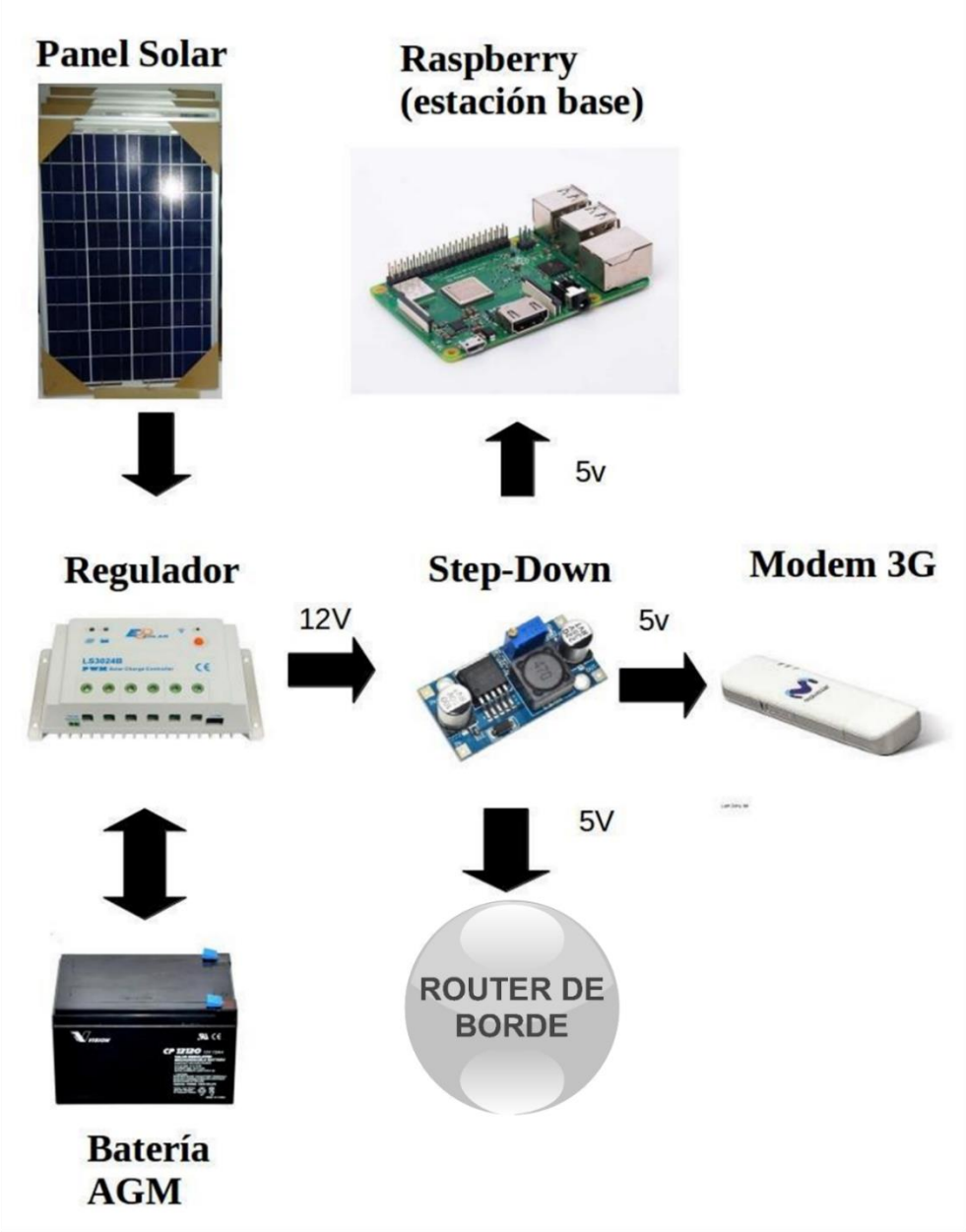


Figura 3.16: Sistema de alimentación del Nodo Gateway

Durante el día el panel solar va a entregar tensión al regulador LS3024B, este es el encargado de estabilizar los niveles de tensión del panel solar y seleccionar si el panel o la batería se conecta a la carga. La salida producida por el regulador es de 12 V y se conecta a la entrada del módulo step-down LM2596S. Este último va a reducir la tensión de 12 a 5 V. El voltaje de 5 V regulado sirve para alimentar al Router de Borde, al módem 3G y a la Raspberry Pi. En ausencia de sol, la batería es la encargada de proveer la energía para alimentar al sistema.

3.5 Software

3.5.1 Introducción

Para cumplir con los objetivos del proyecto, era crucial disponer de un Sistema Operativo y una pila de protocolos que permitieran desarrollar eficientemente las funcionalidades requeridas, optimizando el uso de los recursos del nodo. Para ello, se empleó el Sistema Operativo Contiki y los protocolos descritos en la Sección 2.4.
La aplicación desarrollada consta de un módulo principal (*proyecto_iot*) que contiene un proceso (*proceso_medir*) y 10 recursos de CoAP.

- ***recurso_pulsador_PC4*:** devuelve el estado del pin PC4.
- ***recurso_pulsador_PC5:*** devuelve el estado del pin PC5.
- ***recurso_temperatura_cpu:*** devuelve el valor de temperatura del cpu del MCU.
- ***recurso_voltaje_cpu:*** devuelve el valor de voltaje de la cpu.
- ***recurso_humedad_suelo:*** devuelve el valor de humedad del suelo.
- ***recurso_temperatura_ambiente:*** devuelve el valor de temperatura ambiente.
- ***recurso_humedad_ambiente:*** devuelve el valor de humedad ambiente.
- ***recurso_temperatura_suelo:*** devuelve el valor de temperatura del suelo.
- ***recurso_bateria_litio:*** devuelve el nivel de carga de la batería de litio.
- ***recurso_configuracion:*** devuelve el tiempo de muestreo, permite configurar el tiempo de muestreo, permite encender/apagar un relé y conocer su estado.

Los recursos *recurso_voltaje_cpu* y *recurso_temperatura_cpu* se agregaron para mostrar información adicional sobre el MCU. Estos sólo se pueden acceder de forma manual (ejecutando el comando GET) como se explicará más adelante y no serán subidos a la nube.
El módulo principal se encarga de inicializar el motor de CoAP, activar los recursos, configurar los puertos usados y dar comienzo al proceso *proceso_medir*. Este proceso es el encargado de tomar medidas periódicamente de los 3 sensores, leer el puerto ADC que entrega el nivel de carga de la batería y de almacenar la última medida/lectura en la memoria del MCU. Los recursos le permiten al Nodo Gateway obtener las medidas tomadas y acceder a algunos datos que brindan información sobre el nodo. También, le va a permitir a un usuario (conectado a la consola de la Raspberry Pi) configurar algunos parámetros de los nodos sensores (período de muestreo, encender un relé o ver estado del relé) y conocer el voltaje y temperatura del MCU.
En esta sección se presentará la descripción funcional de la aplicación, se mostrará su diseño e implementación.
Cabe aclarar que para poder programar (compilar y grabar un programa) nuestros nodos se utilizaron diferentes herramientas, tales como Eclipse, compilador GCC, entre otras.

3.5.2 Descripción funcional de la aplicación

La red está formada por 3 nodos sensores y un Nodo Gateway. Cada uno actúa de la siguiente manera:

Tipo de Nodo	Nodos sensores	Nodo Gateway
Participación en la red	Cada nodo sensor actúa como un servidor CoAP.	El Nodo Gateway actúa como cliente CoAP.
Función	Es un nodo que almacena información y la envía al cliente a través de la red.	El cliente le solicita al servidor información que éste almacena.

Tabla 3.2: Tipos de nodos de la red y sus funciones

Como se mencionó anteriormente, uno de los componentes que forma parte del Nodo Gateway es una Raspberry Pi (Estación Base). Esta última, utilizando el protocolo CoAP a través del Router de Borde (nodo enrutador), puede solicitar información de configuración de los parámetros de cada nodo sensor, obtener los valores medidos, reestablecer el período de muestreo y encender un relé.
Los nodos sensores toman periódicamente medidas de cada uno de los 3 sensores y del puerto ADC con el cual se va a leer el nivel de carga de la batería, y almacenan toda la información en memoria. Esta información es enviada al cliente cuando éste lo solicite.
El sistema fue automatizado para que el cliente (Nodo Gateway) le solicite a cada nodo sensor la información medida por los sensores, el nivel de carga de la batería, estados de los pulsadores, etc; para luego subir la información a Thingspeak.

3.5.3 Diseño

Para poder solicitar las medidas y enviar comandos desde el Nodo Gateway, se utilizaron diferentes recursos de CoAP que permiten realizar este tipo de tareas de manera sencilla.
Durante la etapa de diseño de la aplicación, se determinó que sería más eficiente tomar las medidas de los sensores utilizando un proceso de Contiki. Inicialmente, se había considerado tomar la medida del sensor dentro del callback o handler del recurso correspondiente, prescindiendo del uso de procesos. En esta configuración, un recurso leía un registro del sensor, procesaba la información para obtener una medida y luego enviaba dicha medida a través de la radio.
Para lograr eficiencia energética, era crucial alimentar los sensores al recibir la solicitud de medición a través de la radio. Sin embargo, una vez alimentado el sensor, era necesario aguardar un tiempo específico antes de poder realizar la lectura. La manera de esperar este tiempo sin emplear un proceso implicaba mantener ocupado el procesador durante dicho lapso. En la práctica, se notó que esto afectaba la comunicación. La ocupación prolongada del procesador dificultaba el adecuado control de la radio. Por esta razón, se optó por utilizar un proceso para tomar las medidas, empleando dos temporizadores etimer. Uno de estos temporizadores se configuró con el período de muestreo, mientras que el otro se estableció con el tiempo de espera entre el encendido de los sensores y su lectura.
De manera periódica, el proceso lleva a cabo la toma secuencial de medidas de los sensores. Para lograrlo, activa los 3 sensores (estableciendo en 1 el pin PC1 del MCU). Sin embargo, para la medición de la batería, simplemente lee el pin ADC (PIN PA5), por lo que no requiere

activación previa. El proceso continúa tomando medidas en el siguiente orden: primero del sensor DHT22, luego del DS18B20, seguido por el Higrómetro YL-69 y, por último, se procede a medir el nivel de carga de la batería de Litio. Finalmente, desactiva los 3 sensores (colocando en 0 el pin PC1) y reinicia el temporizador correspondiente al tiempo de muestreo. El diagrama de flujo del *proceso_medir* implementado se muestra en la Figura 3.18.

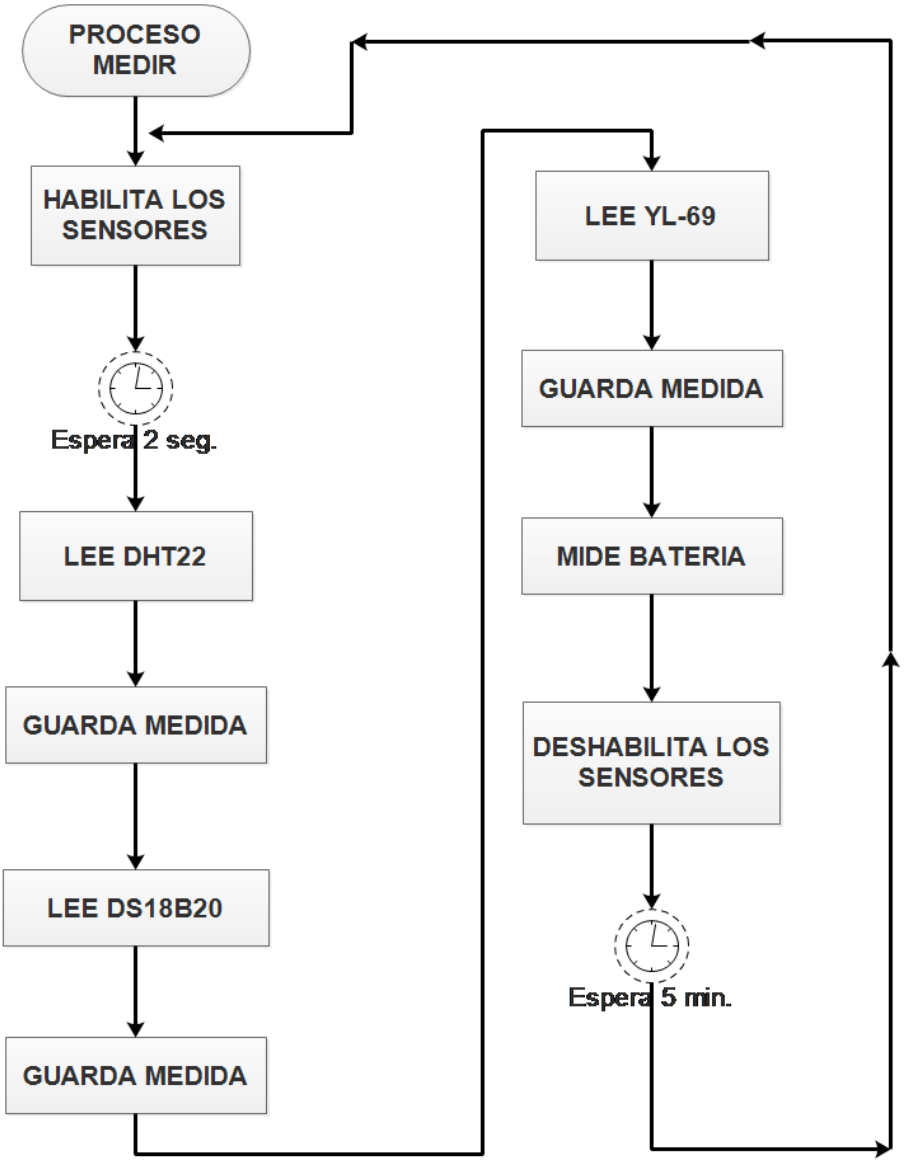


Figura 3.17: Diagrama de flujo del proceso proceso_medir

Aprovechando la capacidad de CoAP para manejar queries (parte de una URL que contiene datos para aplicaciones web), se definió que el campo query del mensaje indique el parámetro a configurar, mientras que el valor a establecer se coloca en el payload. Además, el recurso responde a solicitudes GET proporcionando, en el payload del mensaje, el período de muestreo y el estado del relé.

Los recursos:

- *recurso_pulsador_PC4*
- *recurso_pulsador_PC5*
- *recurso_temperatura_cpu*

- *recurso_voltaje_cpu*
- *recurso_humedad_suelo*
- *recurso_temperatura_ambiente*
- *recurso_humedad_ambiente*
- *recurso_temperatura_suelo*
- *recurso_bateria_litio*

Almacenan las últimas medidas de sensores, el voltaje de la batería y los estados de los pulsadores, así como la temperatura y el voltaje del MCU. Estos recursos responden a solicitudes GET proporcionando la última medida almacenada.
Para comunicarse con los nodos sensores se utilizó la Raspberry Pi (Estación Base) que ejecuta un cliente CoAP, mientras que cada nodo sensor ejecuta un servidor CoAP. El enrutamiento de los paquetes de una red a otra, es decir, de Internet a la red inalámbrica de sensores, se logra por medio del Router de Borde. Este último no es más ni menos que un nodo al que se programa con la app *rpl-border-router* que se encuentra en *contiki/examples/ipv6/rpl-border-router* y permite la comunicación entre la red de sensores y la Raspberry Pi.

3.5.4 Implementación

El código fuente de los nodos se encuentra ubicado en *contiki/examples/proyecto_iot/*. Sus módulos principales están distribuidos según se indica en la Figura 3.19. Además, fue necesario implementar y modificar archivos indicados en la Figura 3.20.
CoAP está implementado en Contiki mediante el motor llamado *erbium.* Sus archivos principales residen en el directorio *contiki/apps/er-coap.* La implementación de CoAP consta de varios módulos, como *er-coap* y *er-coap-engine.* Para crear una aplicación que utilice CoAP es necesario incluir esta aplicación en el *Makefile*. Esto se logra con las siguientes líneas:
APPS += er-coap
APPS += rest-engine

Para la implementación de las aplicaciones que usan CoAP, generalmente se crea un archivo *recurso-nombre.c* para cada recurso. Además, es recomendable para mantener el código ordenado, crear una carpeta dentro de la carpeta del proyecto para guardar todos los recursos. (separados del archivo principal de la aplicación)

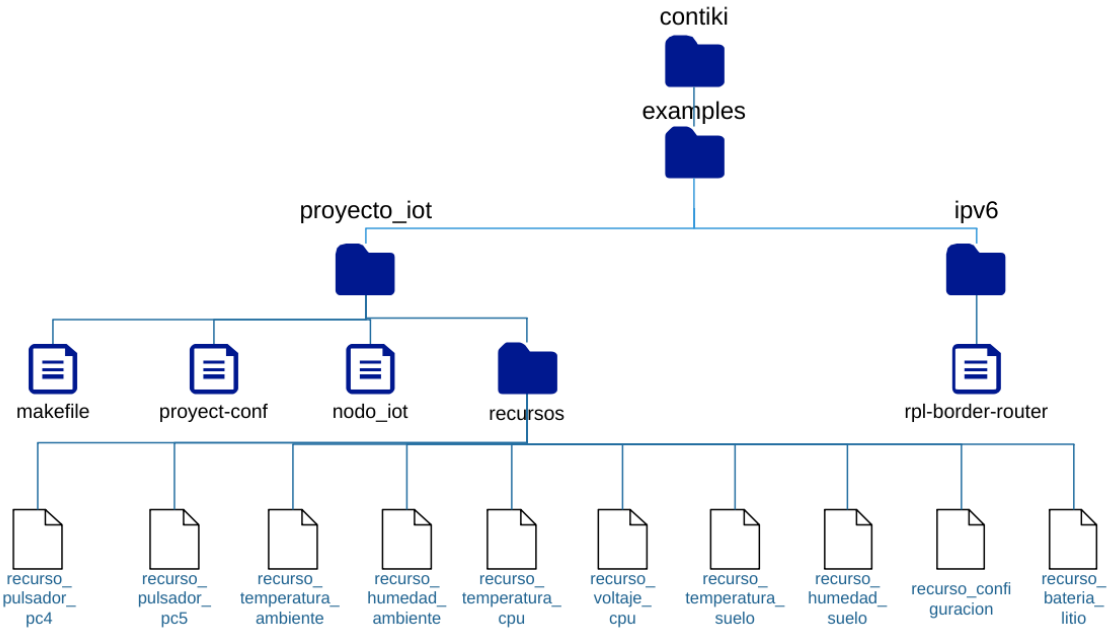


Figura 3.18: Principales archivos de la aplicación

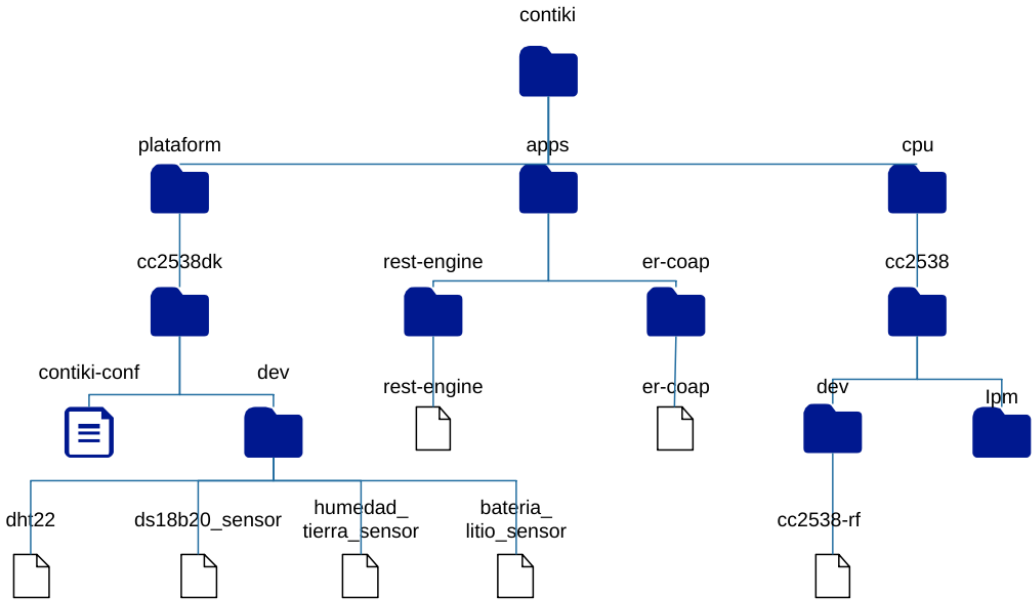


Figura 3.19: Archivos secundarios de la aplicación

El proceso proceso_medir

Como se mencionó anteriormente este proceso es el encargado de tomar las medidas de los sensores y de medir el nivel de carga de la batería de forma periódica. En primer lugar, el proceso habilita los sensores.

Para habilitar los 3 sensores se configura el pin PC1 del MCU como salida y se lo pone a 1. Esto es posible ya que PC1 corresponde a un pin GPIO (General Purpose Input/Output). Luego se activan los sensores con las siguientes funciones:

- *SENSORS_ACTIVATE(bateria_litio_sensor).*
- *SENSORS_ACTIVATE(dht22).*
- *SENSORS_ACTIVATE(ds18b20_sensor).*
- *SENSORS_ACTIVATE(humedad_tierra_sensor).*

Una vez activados se establece un *etimer* de 2 segundos para inicializar los sensores.
Las mediciones de los sensores se realizan de forma secuencial:

1°) SENSOR DHT22.

2°) SENSOR DS18B20.

3°) SENSOR YL-69.

4°) MEDICION DE VOLTAJE DE BATERÍA DE LITIO.

Cabe aclarar que para obtener el voltaje de la batería de Litio y para medir la humedad de la tierra se realizaron conversiones ya que estos sensores están conectados a puertos ADC. El puerto ADC va a entregar un valor digital por lo tanto se requiere un cálculo adicional para obtener los valores porcentuales de dichas medidas. Las conversiones y cálculos se explican en la Sección 4.2.
Luego los sensores se desactivan con las siguientes funciones:

- *SENSORS_DEACTIVATE(bateria_litio_sensor).*
- *SENSORS_DEACTIVATE(bateria_agm_sensor).*
- *SENSORS_DEACTIVATE(dht22).*
- *SENSORS_DEACTIVATE(ds18b20_sensor).*

Una vez tomadas y almacenadas todas las mediciones se pone a 0 el pin PC1 para deshabilitar los sensores.
Por último, se establece un etimer con el valor almacenado en la variable *periodo_de_sensado*. Esta variable representa el tiempo en minutos que transcurre entre medidas, durante el cual se libera el procesador para realizar otras tareas (como por ejemplo manejar la radio). Fue fundamental utilizar un proceso y un temporizador para poder liberar el procesador. El ciclo anterior se repite indefinidamente ya que se encuentra en un lazo While (1) dentro del código fuente del proceso medir.

Recursos de CoAP

Un recurso de CoAP se define utilizando una variable de tipo *resource_t.* Esta estructura está formada principalmente por punteros a funciones. Algunas de estas funciones son *get_handler, post_handler, put_handler, delete_handler* y se deben implementar aquellos métodos soportados por el recurso (GET, PUT, POST y DELETE). La inicialización de esta estructura se realiza utilizando la función *rest_activate_resource (resource t *resource, char*

**path),* donde el primer parámetro es el recurso declarado y el segundo es el *string* que define la ubicación del recurso en el servidor.

Configuración

Para realizar la configuración de parámetros del nodo a través de la Raspberry se crea en los nodos sensores el recurso *recurso_configuracion.* El recurso responde a los métodos GET y POST. Para esto fue necesario implementar los respectivos *handlers.*

El *handler rest_get_handler* se encarga de generar una respuesta con los valores de los parámetros que tiene configurados el nodo. La respuesta contiene el estado del relé y período de sensado. Luego dentro del *handler*, se indica que la respuesta contiene texto plano (compuesto únicamente por texto sin formato, sólo caracteres) y se guarda en el *payload* del mensaje la respuesta creada.

Para poder realizar la configuración de los parámetros se implementó el *handler res_post_put_handler.* El nodo recibe un paquete que en su campo *query* indica que parámetro se quiere configurar y en su *payload* contiene el valor que se desea cargar en ese parámetro. Usando esta información se escribe la variable asociada al parámetro indicado. Por ejemplo, si el campo *query* del mensaje contiene el *string "*parámetro = p" y el *payload* "valor = 5" se va a configurar el período de muestreo con el valor de 5 minutos. Los parámetros que permiten configurar este recurso son:

- Período de muestreo de los sensores (frecuencia de toma de medidas de los sensores y la carga de la batería).
- Estado del relé (abierto o cerrado).
- Encendido o apagado del relé.

Sensores

Como se mencionó anteriormente, para enviar las últimas medidas tomadas por los sensores se implementaron los recursos que se indican en la Sección 3.5.3. Estos recursos responden al comando GET con la última medida almacenada.

El *handler res_get_handler* es el encargado de formar el mensaje de respuesta al mensaje del tipo GET. Este mensaje contiene la última medida almacenada en el MCU.

Drivers de los sensores

Este sistema cuenta con 3 sensores: el DTH22 para medir la humedad y temperatura ambiente, el DS18B20 para medir la temperatura del suelo y el YL-69 para medir la humedad del suelo. El primero y el segundo son sensores de interfaz digital, mientras que el tercero es de interfaz analógica.

Los drivers de los tres sensores fueron implementados aprovechando las estructuras definidas en el módulo *sensors* ubicado en *contiki/core/lib* para tener mayor versatilidad en el código fuente. En este módulo se define la estructura *sensor_sensors* para utilizar de forma ordenada las funciones encargadas de activar y obtener las medidas de los sensores. Al definir los diferentes sensores utilizando esta estructura es necesario implementar las funciones de la estructura para cada uno de ellos.

Los sensores se activan utilizando la macro *SENSORS_ACTIVATE(sensor).* En el caso del DHT22 y el DS18B20 habilita la comunicación 1-Wire (protocolo) mientras que en el YL-69 inicializa los puertos del conversor ADC.

Para medir el nivel de carga de la batería se utilizó un puerto ADC y la implementación fue similar a la del sensor de humedad del suelo (YL-69).

La implementación de la obtención de las medidas de los distintos sensores se detalla a continuación en la sección correspondiente a cada sensor.

Driver del sensor YL-69

Para realizar la lectura de este sensor se crearon los archivos *humedad_tierra_sensor.h* y *humedad_tierra_sensor.c* que están ubicados en *contiki/platform/cc2538dk/dev/*. El sensor YL-69 se conectó a un GPIO del MCU (pin PA6), el cual se configuró como conversor ADC para leer la medida. A este conversor se le puede configurar el pin del cual se lee el voltaje a convertir, la cantidad de bits de la medida convertida a digital y el rango de tensión de referencia, el cual puede ser externo o interno. Para tener mayor resolución se configuró la cantidad de bits de la medida convertida en 12 bits, siendo esta la máxima cantidad de bits a la que el conversor ADC puede convertir. Además, para simplificar el diseño del hardware se decidió utilizar una referencia de tensión interna (3.3V). Las configuraciones anteriores se implementaron dentro de las funciones *value* y *configure* en el archivo *humedad_tierra_sensor.c*.

Driver del Sensor DHT22

Para realizar la lectura de este sensor se crearon los archivos *dht22.h* y *dht22.c* que se encuentran en *contiki/platform/cc2538dk/dev/*. El pin GPIO elegido para conectar la línea de datos del sensor es el PB2 el cual se definió como entrada digital.

El DHT22 utiliza un bus simple de comunicación, es un bus de una sola línea (1-Wire) para la transmisión y recepción de datos.

La transferencia de datos es iniciada por el cc2538 el cual manda el bus a nivel BAJO al menos por 18 ms, transcurrido este tiempo, el cc2538 manda el bus a estado ALTO, en este punto el MCU queda en espera de la respuesta del sensor la cual tiene un tiempo de 20 a 40 μs.

Antes de que el DHT22 envíe algún dato, este nos debe responder que ha recibido la petición de transmitir los datos de temperatura y humedad, esto lo hace mandando el bus a nivel BAJO por 80 μs y luego regresando el bus a nivel ALTO por otros 80 μs, una vez terminada la confirmación empieza la transmisión de datos.

Luego de la inicialización el sensor envía un tren de pulsos (información). Los anchos de los pulsos van a determinar si se trata de un 1 o 0 lógico, por lo que debemos utilizar un timer para contar la duración de los pulsos.

Cada bit es separado por un pulso BAJO que dura 50 μs, después el sensor nos enviará un pulso ALTO y dependiendo de la duración de dicho pulso, sabremos a que valor lógico corresponde.

El sensor manda el bus a nivel BAJO durante 50 μs antes de enviar el bit, esto lo hace con el fin de separar los bits y poder diferenciarlos. Un pulso que dura 70 μs o más, corresponde a un 1 lógico.

Al inicio un pulso BAJO de 50 μs seguido de un pulso ALTO de una duración de 26 a 28 μs lo cual equivale a que el sensor nos está enviando un 0 lógico.

En el transcurso de la transmisión, el sensor enviará un total de 5 Bytes: 2 Bytes de temperatura, 2 Bytes de humedad relativa y 1 Byte para el Checksum. Una vez recibidos los datos no debemos utilizar una fórmula para descifrar qué valor hemos recibido porque en el MSB tenemos sus valores enteros y en su LSB sus decimales.

Driver del sensor DS18B20

Por último, para lograr la comunicación y leer este sensor se crearon los archivos *ds18b20_sensor.c* y *ds18b20_sensor_h* también ubicados en *contiki/platform/cc2538dk/dev/*. Se definió el pin PB1 como entrada digital para recibir las lecturas del DS18B20.

La comunicación entre el cc2538 y el DS18B20 se lleva a cabo mediante el protocolo 1-Wire de Dallas Semiconductor.

La comunicación en el bus se hace por medio de Time-slots (ranuras de tiempo), el maestro (cc2538) inicia un Time-slot generando un pulso durante un tiempo determinado, así un 1 lógico se verá como un time-slot corto, y un 0 lógico se verá como time-slot largo.

El reset es usado para iniciar cualquier transferencia de datos entre el MCU y el DS18B20, este constituye la secuencia de inicialización.

En primer lugar, se realiza un sondeo de canal (bus) para saber si hay un sensor conectado. El maestro transmite (modo TX) un pulso de reset llevando el bus a nivel BAJO durante un mínimo de 480 μs y luego libera el bus pasando a modo de recepción (RX). Cuando se libera el bus, la resistencia de Pull-up (que se conecta al pin de datos) eleva el bus 1-Wire hacia estado ALTO. Cuando el DS18B20 detecta el flanco ascendente, espera de 15 a 60 μs y luego transmite un pulso de presencia bajando el bus durante 60 a 240μs.

Luego de la inicialización el cc2538 detecta que en el bus hay un dispositivo (DS18B20) listo para recibir comandos ROM. Debido a que sólo tenemos un sensor conectado podemos acceder al mismo sin necesidad de especificar la ID (identificador único de dispositivo). Para ello el MCU envía el comando Skip Rom [CCh] seguido del comando Convert T [44h] que inicia la conversión de temperatura única. Después de la conversión, los datos térmicos resultantes se guardan en un registro de temperatura de 2 bytes en la memoria del Scratchpad y el sensor vuelve a su estado inactivo de bajo consumo. Como el DS18B20 está alimentado por una fuente externa, el maestro puede emitir Time-slots de lectura después del comando Convert T y el DS18B20 responderá transmitiendo un 0 mientras la conversión de temperatura está en progreso y un 1 cuando se realiza la conversión.

En este punto el sensor ya ha realizado la medición de temperatura por lo que deberíamos enviarla al cc2538.

Como se dijo anteriormente, cada acceso al sensor requiere de 3 transiciones, la primera corresponde a la inicialización, luego se accede al sensor (enviando un Comando Room) y por último se envía el comando de función.

Para poder leer el valor medido se envía el comando Read Screatchpad [BEh], este comando le permite al cc2538 leer el contenido del Scratchpad. La transferencia de datos comienza con el bit menos significativo del byte 0 y continúa a través del Scratchpad hasta que se lee el noveno byte (byte 8 - CRC).

Driver de la Radio

CC2592-PLUS es el nombre que le hemos puesto al trozo de código necesario para acoplar el cc2592 al cc2538 en los módulos CC2538+CC2592.

El cc2592 posee tres pines de control: PA_EN, LNA_EN y HGM. Los cuales vienen conectados generalmente con los pines PC3, PC2 y PD2 del cc2538 respectivamente.

Como ya se dijo antes, el proyecto entero se encuentra en el directorio “contiki”, incluido este driver.

El trozo de código se encuentra en el archivo: “*contiki/cpu/cc2538/dev/cc2538-rf.c*”.

3.6 Tareas automatizadas en la Estación Base (Raspberry)

Cuando inicia la Raspberry se ejecutan tres scripts declarados en el archivo /etc/rc.local:

- **umtskeeper:** Es el encargado de iniciar el Módem 3G para tener acceso a internet.
- **tunslip6:** Establece el túnel slip entre la Raspberry y el router de borde.
- **sondeo_nodo.sh:** Sondea cada 5 segundos si los nodos sensores son visibles.

Luego, cada 5 minutos se ejecutan dos scrips programados en Crontab:

- **check_ppp0_tun0_rb.sh:** Es el encargado de chequear si el módem 3G, el túnel slip y el router de borde se encuentran activos, en caso contrario se reestablecerán para que vuelvan a funcionar.
- **recoleccion_datos.sh:** Obtiene todos los datos sensados por cada uno de los nodos sensores y los sube a la plataforma ThingSpeak.

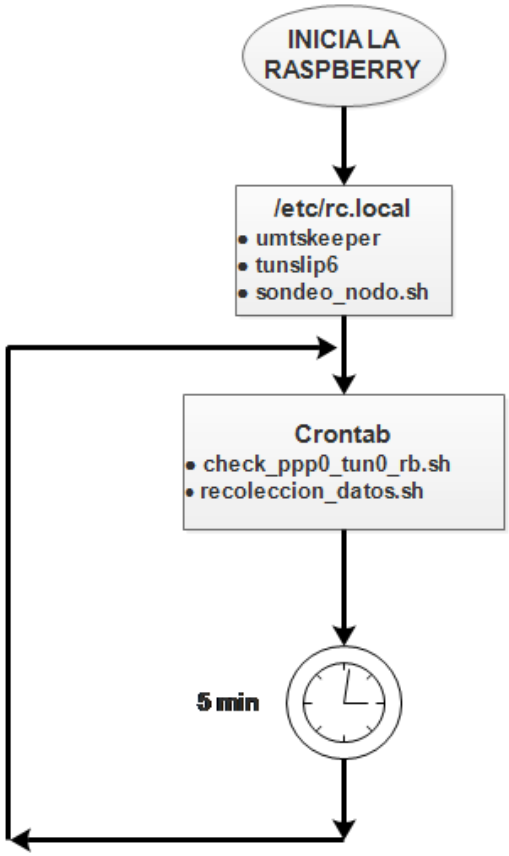


Figura 3.20: Automatizaciones en la Raspberry

.

4 Cálculos, ensayos y Mediciones

4.1 Cálculo de paneles solares y baterías

Los cálculos mostrados a continuación están pensados para sistemas fotovoltaicos autónomos y de bajo consumo de energía, como lo son las redes de sensores.
Normalmente este tipo de sistemas se componen de los siguientes elementos:

- **Panel Solar:** Suministra energía al sistema en los momentos que recibe luz solar.
- **Batería:** Almacena energía eléctrica para ser entregada al sistema en ausencia de luz solar.
- **Regulador:** Estabiliza los niveles de tensión del conjunto panel solar y batería, para entregar siempre una tensión constante a la carga.
- **Carga:** Es el equipamiento eléctrico que consume la energía entregada por el conjunto panel solar y batería.

Vamos a alimentar dos tipos de dispositivos: el primero corresponde al Nodo Gateway y el segundo al Nodo Sensor.

Nodo Gateway
1° Paso →se calcula el consumo estimado para un día de funcionamiento.
El nodo gateway tiene los siguientes consumos de corriente:
0.23 A → cuando el nodo no transmite datos a internet
0.37 A → cuando el nodo transmite datos a internet

Debido a que la tensión de alimentación es de 12 V, las potencias que se consumen en ambos casos son:

$$Potencia(sin\ transmitir\ datos) = 0.23\ A \times 12\ V = 2{,}76\ W$$
$$Potencia(transmitiendo\ datos) = 0.37\ A \times 12\ V = 4{,}44\ W$$

Si bien es claro que el mayor consumo de potencia es durante la transmisión de datos a internet, esto solo ocurre por pocos segundos. Por lo tanto, para los cálculos elegimos un valor de potencia estimado que se encuentre entre medio de los valores calculados anteriormente.

$$Potencia\ consumida = 3\ W$$

La energía consumida se calculó para las horas de funcionamiento. En este caso corresponde a un día completo (24 hs).

$$Energía\ Consumida = Horas\ de\ funcionamiento \times Potencia\ consumida$$
$$= 24\ h \times 3\ W = 72\ W.h$$

Entonces, el consumo total para un día de funcionamiento es:

$$Total\ Consumo\ Estimado(Cde) = 72\ W.\frac{h}{día}$$

Aplicando un rendimiento de la instalación del 75 %, se calcula la energía total necesaria para abastecer la demanda.

$$Total\ Energía\ Necesaria = \frac{Cde}{0.75} = 96\ W.\frac{h}{día}$$

2° Paso → Radiación solar disponible.

Para los cálculos siempre hay que tener en cuenta la época del año con menor radiación solar disponible para la región en la cual va a ser instalado el sistema; y esto se debe a que siempre hay que tener en cuenta la peor condición para los cálculos. En nuestro caso, nuestro objetivo es poder realizar los ensayos el mes de Diciembre en el Establecimiento Rukalen, ubicado en la Localidad de Cabildo, Partido de Bahía Blanca. El dato se obtuvo del Atlas de Energía Solar de la República Argentina (Figura 4.1).

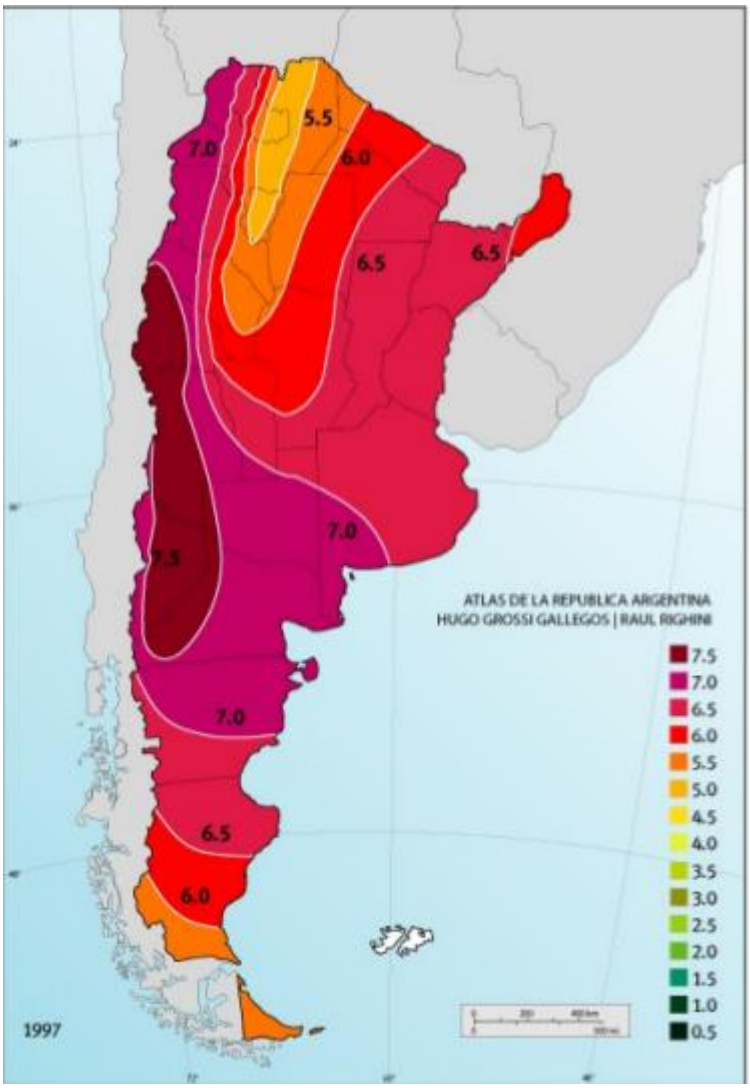


Figura 4.1: Distribución espacial del promedio de la irradiación solar global diaria (kWh/m2) correspondiente al mes de Diciembre

Por lo tanto, la radiación incidente para el mes de Diciembre es 7.0 kW.h /m^2.
Una vez que conocemos la radiación solar incidente, la dividimos por 1 kW/m² y obtenemos la cantidad de horas sol pico (HSP).

$$HSP = \frac{radiación\ solar}{1\ \frac{kW}{m^2}} = 7.0\ HSP$$

3° Paso → Cálculo de paneles solares necesarios

Vamos a realizar los cálculos para determinar la cantidad de módulos de paneles solares necesarios en función de las condiciones anteriormente establecidas. Para realizar este cálculo elegimos paneles solares de 10 W.
Para instalaciones de uso diario utilizaremos la siguiente fórmula:

$$Número\ de\ módulo = \frac{Energía\ Necesaria}{HSP \times Rendimiento\ de\ Trabajo \times potencia\ pico\ del\ módulo}$$

El rendimiento de trabajo tiene en cuenta pérdidas por suciedad y/o deterioro de los paneles fotovoltaicos (normalmente 0.7 – 0.8).

$$Nmd = \frac{96}{7.0 \times 0.8 \times 10} = 1.71$$

Por lo tanto, la cantidad de paneles necesarios serían de dos unidades.
Para nuestro caso, utilizamos un solo panel solar de 20 W (ver Figura 4.2) con las siguientes características:

- Potencia Nominal (PN): 20 W
- Policristalino
- Medidas 530 x 350 x 30 mm
- Corriente a PN: 1,23 A
- Tensión a PN: 17,80 V
- Tensión a Circuito Abierto: 22,52 V
- Peso: 2,50 kg

Figura 4.2: Panel Solar policristalino 20 W

4° paso → Capacidad de los acumuladores (baterías)
Para determinar la capacidad necesaria de las baterías, primero tenemos que establecer la autonomía deseada en caso de tener días desfavorables de insolación por abundante nubosidad.

$$Capacidad\ de\ la\ batería = \frac{energía\ necesaria \times días\ de\ autonomía}{voltaje \times profundidad\ de\ descarga\ de\ la\ batería}$$

La profundidad de descarga depende del tipo de batería elegida. Estos valores oscilan entre 0.5 y 0.8.
En este nodo se usó una batería AGM (Absortion Glass Mat) CP12120 Vision de 12 V cuya profundidad de descarga es de 0.5.

$$Capacidad\ de\ la\ batería = \frac{96 \times 1}{12 \times 0.5} = 16\ A.h$$

El Nodo Gateway (router de borde + raspberry + modem 3G), de acuerdo a los componentes que lo conforman, es el nodo de mayor consumo que posee la red.
Otro elemento importante es la selección del regulador de carga solar (este podría considerarse un 5°paso), que se utilizará para cargar las baterías, ya que el panel entregará una energía en función de la cantidad de energía solar que reciba, la cual no es constante. Por lo que debe existir un elemento que reciba dicha energía y la administre de la mejor forma para cargar las baterías o alimentar el nodo.
Para suplir dicha necesidad se utilizará el Controlador de Carga Solar LS3024B, con las siguientes características:

- Tensión Nominal del Sistema 12/24 VDC
- Máximo Voltaje de Entrada PV 50 V
- Carga Nominal/Corriente de Descarga 30 A

Nodo Sensor
1° paso → se calcula el consumo estimado para un día de funcionamiento.
El nodo sensor es el que menos consumo tiene. La corriente que consume el nodo sensor es de 40 mA.
Debido a que este nodo se alimenta con 5 V la potencia consumida es de:

$$Potencia\ consumida = 40\ mA\ \times\ 5\ V = 0{,}2\ W$$

La energía consumida se calculó para las horas de funcionamiento. En este caso corresponde a un día completo.

$$Energía\ Consumida = Horas\ de\ funcionamiento \times Potencia = 24\ hs \times 0{,}2\ W = 4.8\ W.h$$

Entonces, el consumo total para un día de funcionamiento es:

$$Total\ Consumo\ Estimado(Cde) = 4.8\ W.\frac{h}{dia}$$

Se aplica un rendimiento de la instalación del 75 % para calcular la energía total necesaria para abastecer la demanda.

$$Total\ Energía\ Necesaria = \frac{Cde}{0.75} = 6.4\ W.\frac{h}{dia}$$

2° paso →Radiación solar disponible.
Al igual que en el caso del Nodo Gateway, la radiación solar disponible para el mes de Diciembre es de 7.0 kW.h/m² (Ver Figura 4.1).
El cálculo de la cantidad de horas sol pico es igual al caso anterior:

$$HSP = \frac{radiación\ solar}{1\ k.\frac{W}{m^2}} = 7.0\ HSP$$

3° paso → Cálculo de paneles solares necesarios.
Vamos a realizar los cálculos para establecer el número de módulos (paneles solares) en función de las condiciones de radiación más desfavorables. Para realizar este cálculo elegimos paneles de 1 W.
Para instalaciones de uso diario utilizaremos la siguiente formula:

$$Número\ de\ módulo = \frac{Energía\ Necesaria}{HSP \times Rendimiento\ de\ Trabajo \times potencia\ pico\ del\ módulo}$$

El rendimiento de trabajo tiene en cuenta pérdidas por suciedad y/o deterioro de los paneles fotovoltaicos (normalmente 0.7 – 0.8)

$$Nmd = \frac{6.4}{7.0 \times 0.8 \times 1} = 1.14$$

Esto nos daría, 1 panel solar de 1 W.
Para este caso se utilizó un panel solar monocristalino de 5 V con las siguientes características:

- Voltaje Nominal: 5 V
- Potencia 1 W
- Dimensiones: 135 x 64,7 mm

4° paso → Capacidad de los acumuladores
Para determinar la capacidad necesaria de las baterías, primero tenemos que establecer la autonomía deseada en caso de tener días desfavorables de insolación por abundante nubosidad.

$$Capacidad\ de\ la\ batería = \frac{energía\ necesaria \times días\ de\ autonomía}{voltaje \times profundidad\ de\ descarga\ de\ la\ batería}$$

La profundidad de descarga depende del tipo de batería elegido. Estos valores oscilan entre 0.5 y 0.8. Para este nodo se utilizó una batería de Li-Ion 18650 de 3.7 V, cuya profundidad de descarga es de 0.8. Y se eligió un solo día de autonomía solo con fines de realizar ensayos.

$$CapacidaddeAcumulación = \frac{6.4 \times 1}{3.7 \times 0.8} = 2162.21\ mA.h$$

Para el caso de los nodos sensores se utiliza una batería de Litio 18650 de 2200 mA.

4.2 Medición de Voltaje de las Baterías con el cc2538

Consideramos necesario conocer el estado de carga de las baterías de cada componente del sistema y así evitar que alguno de estos deje de funcionar por falta de alimentación.
Los nodos sensores se alimentan con baterías de Litio 18650. A partir de su hoja de datos se establecieron los umbrales de tensión mínimos y máximos.

Umbrales de voltaje de carga: 2 V →Voltaje mínimo (batería descargada)
4.2 V →Voltaje máximo (batería con carga completa)

Para poder conocer el nivel de carga de la batería se utiliza un puerto ADC del MCU, en este caso se utiliza el puerto (pin) PA3.
En primer lugar, se implementó un divisor de tensión como indica la Figura 4.3. Los valores de las resistencias se obtuvieron teniendo en cuenta de que no le deben representar una carga considerable a la batería, y que la caída de tensión en R2 no debe superar los 3,3 V.

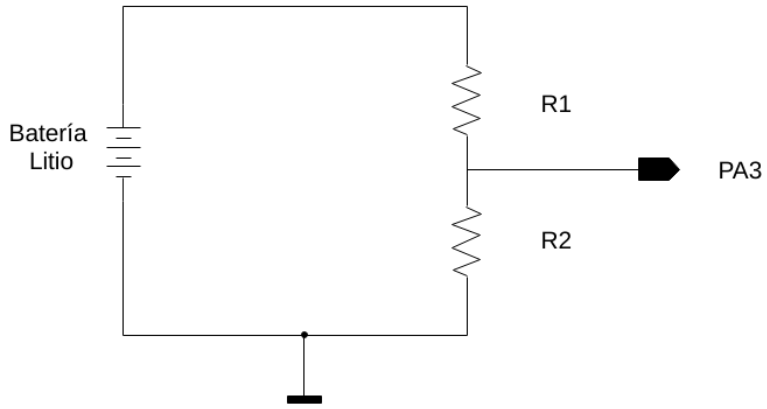


Figura 4.3: Divisor de tensión utilizado para medir el nivel de carga de la batería de Litio

La salida del divisor se conectó al pin PA3 del MCU que corresponde a una de las entradas ADC. Como el voltaje de entrada al pin PA3 no debe superar los 3.3 V se eligieron los siguientes valores de resistencias:

$$R1 = 3.3\ K$$
$$R2 = 6.8\ K$$

Ahora calculamos los voltajes de salida del divisor correspondiente a los umbrales de voltaje de carga de la batería.
Cuando la batería tiene un voltaje de 2 V (umbral mínimo) la salida del divisor de tensión es:

$$Vmin = \frac{(R2\ x\ Vbat)}{(R1 + R2)}$$
$$Vmin = \frac{(6.8\ K\ x\ 2\ V)}{(3.3\ K + 6.8\ K)}$$
$$Vmin = 1.3465\ V$$

Cuando la batería tiene un voltaje de 4.2 V (umbral máximo) la salida del divisor de tensión es:

$$Vmax = \frac{(6.8\ K\ x\ 4.2\ V)}{(3.3\ K + 6.8\ K)}$$
$$Vmax = 2.8277\ V$$

Por lo tanto, los valores de tensión para los umbrales de la batería son:

(Vmin - Vmax) → (1.3465 V – 2.8277 V)

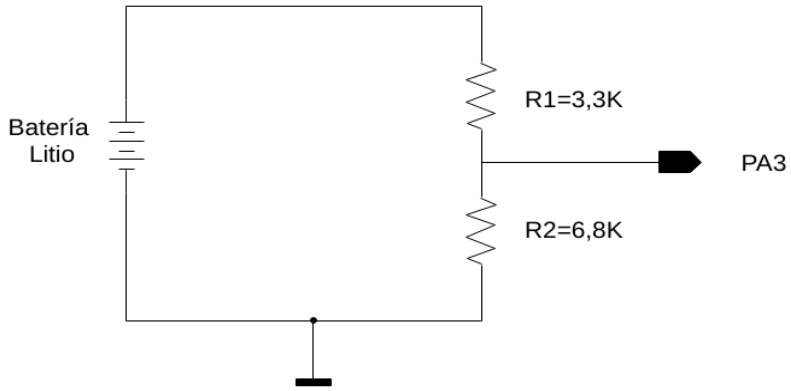


Figura 4.4: Divisor de tensión con sus correspondientes valores de resistencias

Estos valores (Vmin y Vmax) los usaremos más adelante para determinar el porcentaje de carga de la batería.
Por otra parte, se volcaron los valores en la tabla 4.1 y se levantó una curva para determinar el valor del ADC para diferentes tensiones de entrada (Figura 4.5).

TENSION	VALOR_ADC
0.00	-64
0.20	2022
0.38	3880
0.65	6551
0.80	8122
0.97	9863
1.11	11233
1.24	12616
1.39	14134
1.54	15606
1.77	17986
1.93	19596
2.00	20262
2.25	22831

2.37	24040
2.56	25946
2.79	28234
2.96	29856
3.08	30820

Tabla 4.1: Valores de tensión vs Valor del ADC

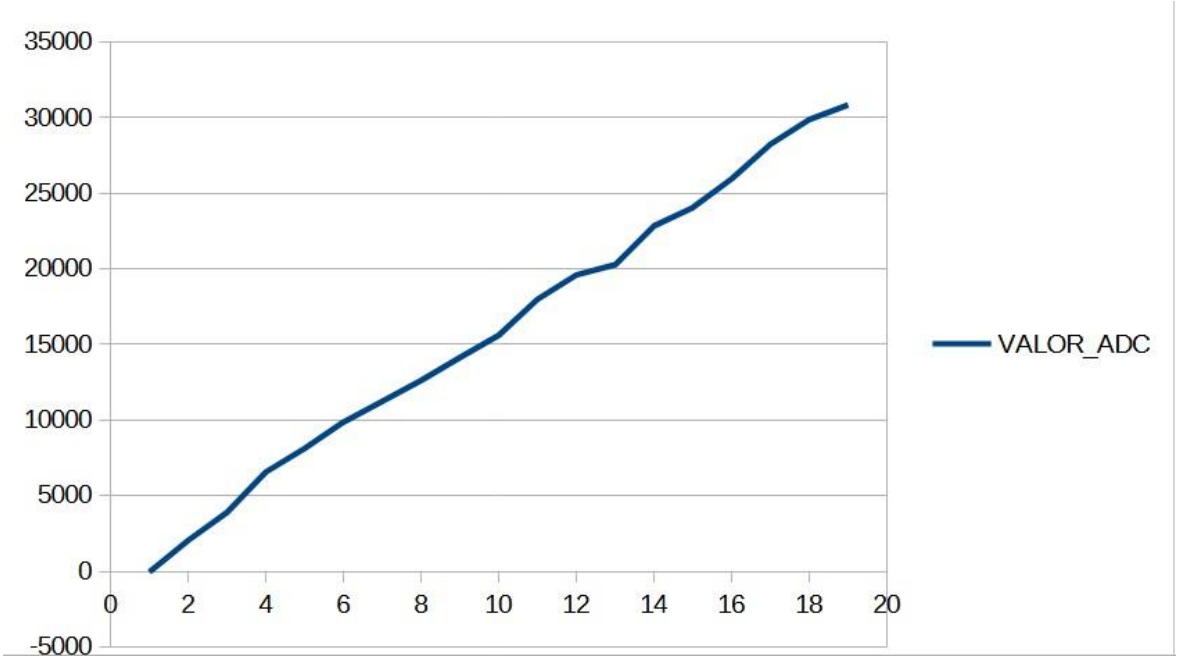


Figura 4.5: Curva correspondiente a la tabla 4.1

Debido a que la curva presenta un comportamiento prácticamente lineal, y teniendo en cuenta los valores de la tabla, se utilizó una ecuación lineal para determinar la tensión de la batería.

$$Vbat = \frac{(valores_{ADC} + 64)}{10027.27}$$

Y para determinar el porcentaje de batería de Litio usamos la siguiente fórmula:

$$Porcentaje = \frac{(Vbat - 1.3465)}{0.0148}$$

Por último, para la batería CP12120 (AGM-Absorbent Glass Mat) se realizó el mismo procedimiento con los mismos criterios anteriores. En este caso se eligió el pin PA4 para medir la tensión del divisor de tensión.

Niveles de tensión de la batería AGM: 10.5 V → mínimo
13.8 V →máximo

Valores de resistencias del divisor de tensión: $R1 = 8.2\ K$

$$R2 = 2.2\ K$$

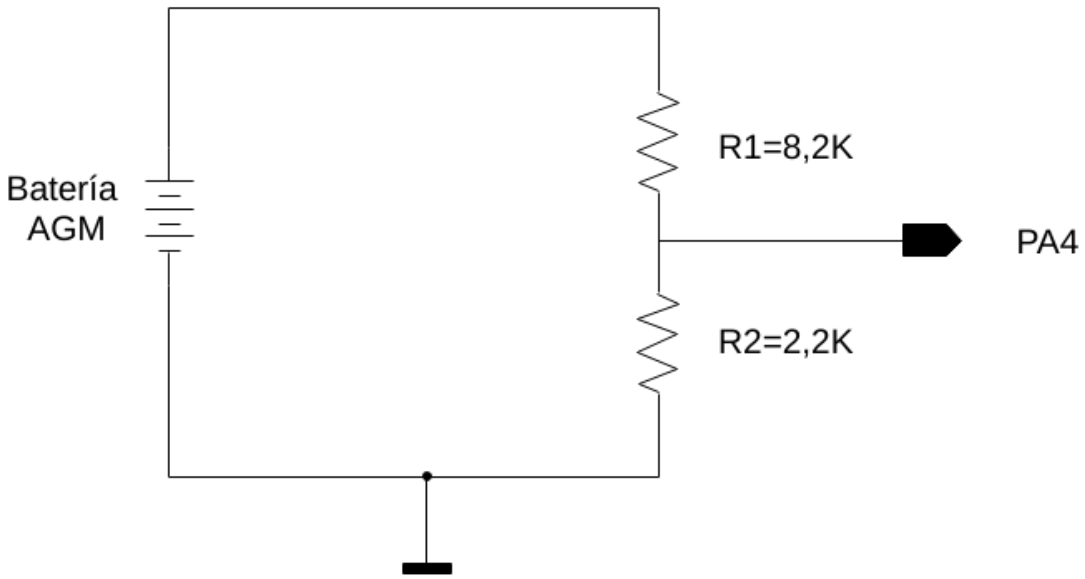


Figura 4.6: Divisor de tensión para medir el nivel de carga de la batería AGM

Cuando la batería tiene un voltaje de 10.5 V (umbral mínimo) la salida del divisor de tensión es:

$$Vmin = \frac{(R2\ x\ Vbat)}{(R1 + R2)}$$
$$Vmin = \frac{(2.2\ K\ x\ 10.5\ V)}{(8.2\ K + 2.2\ K)}$$
$$Vmin = 2.2211\ V$$

Cuando la batería tiene un voltaje de 13.8 V (umbral máximo) la salida del divisor de tensión es:
En este caso el porcentaje es:

$$Vmax = \frac{(2.2\ K\ x\ 13.8\ V)}{(8.2\ K + 2.2\ K)}$$
$$Vmax = 2.9192\ V$$

Por lo tanto, de todos los valores que va leer el ADC nos interesa el siguiente rango:

(Vmin - Vmax) → (2.2211 V – 2.9192 V)

En base a este rango se obtuvo la ecuación para representar el porcentaje:

$$Porcentaje = \frac{(Vbat - 2.2211)}{0.00698}$$

Las ecuaciones de *Vbat* y *Porcentaje* se agregaron al desarrollo de software de cada nodo para realizar las conversiones y mostrar el porcentaje actual de las baterías.

4.3 Cálculo de presupuesto de enlace para los módulos CC2538EM

Los módulos CC2538EM poseen una potencia máxima de transmisión de +7 dBm y una sensibilidad de -97 dBm. Dichos módulos poseen una antena omnidireccional DN007 incorporada en el PCB, la cual posee una ganancia de 1,1 dBi:
Para los cálculos, y por ende para los ensayos, se utilizó una frecuencia de 2480 Mhz (canal 26). Utilizamos esta frecuencia porque no interfiere con las bandas de Wifi en 2,4 Ghz, y nos permitió poder realizar ensayos sin mayores problemas.

Datos para los cálculos:

$Potencia\ máxima\ de\ transmisión = Pmáx = +7\ dBm$
$Sensibilidad\ en\ la\ recepción = RSL = -97\ dBm$
$f = 2480\ Mhz$
$Ganancia\ de\ la\ antena = 1{,}1\ dBi$

Presupuesto de enlace para una distancia de 1 km entre nodos:

$FSL = 20log(2480) + 20log(1) + 32{,}44 = 100{,}32\ dB$
$P_r = 7\ dBm + 1{,}1\ dBi + 1{,}1\ dBi - 100{,}32\ dB = -91{,}12\ dBm$
$Margen\ RX = Pr - RSL = -91{,}12\ dBm - (-97\ dBm) = 5{,}88\ dBm$
$Margen\ RX\ mínimo = 5{,}25\ dBm + 11log(1) = 5{,}25\ dBm$

Como $Margen\ RX > Margen\ RX\ mínimo$ es posible establecer el enlace con línea de vista (LOS).
Observando el resultado anterior, ya nos damos cuenta de que no se puede transmitir a más de 1 km de distancia utilizando los módulos CC2538EM originales sin modificación (coincide con lo que dice el fabricante del módulo).
Para este cálculo es importante conocer el radio de Fresnel. A mitad de distancia, el radio de Fresnel será mayor, de manera de que los calcularemos para esa distancia:

$r = 17{,}32\sqrt{\frac{1}{(4*2{,}48)}} = 5{,}49\ m.$ Esta es la altura óptima para un 100 % de LOS.

Colocar los módulos a una altura de 5,49 m no es algo factible por lo que calcularemos una altura mínima para asegurar la transmisión y recepción de datos.

$Altura\ mínima\ de\ las\ antenas = 5{,}499\ m * 0{,}6 = 3{,}29\ m$

Presupuesto de enlace para una distancia de 800 m entre nodos:

$FSL = 20log(2480) + 20log(0{,}8) + 32{,}44 = 98{,}39\ dB$
$P_r = 7\ dBm + 1{,}1\ dBi + 1{,}1\ dBi - 98{,}39\ dB = -89{,}19\ dBm$
$Margen\ Rx = Pr - RSL = -89{,}19\ dBm - (-97\ dBm) = 7{,}8\ dBm$

$Margen\ RX\ mínimo = 5{,}25\ dBm + 11log(0{,}8) = 4{,}18\ dBm$

Como $Margen\ RX > Margen\ RX\ mínimo$ es posible establecer el enlace con línea de vista (LOS).

$r = 17{,}32\sqrt{\frac{0{,}8}{(4*2{,}48)}} = 4{,}91\ m$. Esta es la altura óptima para un 100 % de LOS

$Altura\ mínima\ de\ las\ antenas = 4{,}91\ m * 0{,}6 = 2{,}95\ m$

Presupuesto de enlace para una distancia de 500 m entre nodos:

$FSL = 20log(2480) + 20log(0{,}5) + 32{,}44 = 94{,}3\ dB$
$P_r = 7\ dBm + 1{,}1\ dBi + 1{,}1\ dBi - 94{,}3\ dB = -85{,}1\ dBm$
$Margen\ Rx = Pr - RSL = -85{,}1\ dBm - (-97\ dBm) = 11{,}89\ dBm$

$Margen\ RX\ mínimo = 5{,}25\ dBm + 11log(0{,}5) = 1{,}93\ dBm$

Como $Margen\ RX > Margen\ RX\ mínimo$ es posible establecer el enlace con línea de vista (LOS).

$r = 17{,}32\sqrt{\frac{0{,}5}{(4*2{,}48)}} = 3{,}88\ m$. Esta es la altura óptima para un 100 % de LOS

$Altura\ mínima\ de\ las\ antenas = 3{,}88\ m * 0{,}6 = 2{,}33\ m$

Presupuesto de enlace para una distancia de 100 m entre nodos:

$FSL = 20log(2480) + 20log(0{,}1) + 32{,}44 = 80{,}32\ dB$
$P_r = 7\ dBm + 1{,}1\ dBi + 1{,}1\ dBi - 80{,}32\ dB = -71{,}12\ dBm$
$Margen\ Rx = Pr - RSL = -71{,}12\ dBm - (-97\ dBm) = 25{,}87\ dBm$

$r = 17{,}32\sqrt{\frac{0{,}1}{(4*2{,}48)}} = 1{,}73\ m$. Esta es la altura óptima para un 100 % de LOS

$Altura\ mínima\ de\ las\ antenas = 1{,}73\ m * 0{,}6 = 1{,}04\ m$

Los módulos CC2538EM tienen la posibilidad de agregar un conector SMA para conectar una antena omnidireccional.

En la Figura 4.7 se puede apreciar que para realizar dicha modificación hay que desoldar la bobina SMD L375 de 3,3 nH y soldarla en la posición L374. Está claro que, al realizar éste cambio, queda inhabilitada la antena del PCB DN007.
Con una antena de 8 dBi es posible superar al menos el doble de distancia, es decir, podríamos llegar a los 2 km (una regla mnemotécnica es que cada 6 dB se duplica aproximadamente la distancia). Igualmente, hay que tener en cuenta que hacer modificaciones en un circuito que trabaja en el rango de los 2,4 Ghz es muy delicado. Cualquier mala manipulación creará desadaptaciones de impedancias, y por ende perdidas de potencia. Nosotros hemos realizado dicha modificación para conectar una antena de 8

dBi, y sólo logramos ganar 2,85 dB en total, dicha ganancia la obtuvimos realizando mediciones en campo.

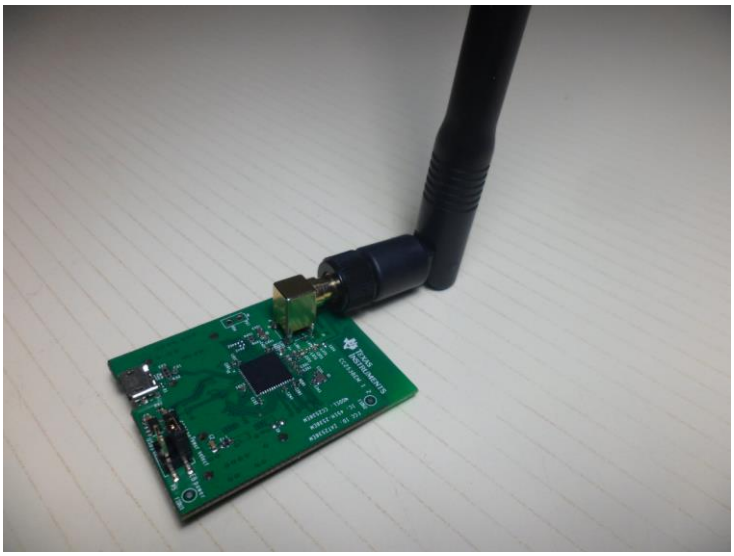


Figura 4.7: Módulo cc2538 + conector SMA + antena

Colocando la antena de 8 dBi deberíamos haber ganado 6,9 dB (ganancia de la antena incorporada 8 dBi restado la ganancia del PCB 1,1 dBi), y sólo logramos ganar 2,85 dB. Los 4,05 dBi faltantes se deben a las pérdidas ocasionadas por la manipulación de la circuitería en 2,4 GHz. Fuimos muy cuidadosos en realizar esta modificación, pero no logramos los resultados esperados, de manera que la opción de modificar los módulos no es factible ni recomendable.

4.4 Cálculo de presupuesto de enlace para los módulos CC2538+CC2592

Los módulos CC2538+CC2592 poseen una potencia máxima de transmisión de +22 dBm y una sensibilidad de -103 dBm. Dichos módulos poseen un conector UFL, en el cual podemos conectar un pigtail UFL A RP-SMA para poder instalar una antena a elección. En nuestro caso utilizamos una antena omnidireccional TP-Link de 8 dBi.
Para los cálculos, y por ende para los ensayos, se utilizó una frecuencia de 2480 Mhz (canal 26). Utilizamos esta frecuencia porque no interfiere con las bandas de Wifi en 2,4 Ghz, y nos permitió poder realizar ensayos sin mayores problemas.

Datos para los cálculos:

$Potencia\ máxima\ de\ transmisión = Pmáx = +22d\ Bm$
$Sensibilidad\ en\ la\ recepción = RSL = -103\ dBm$
$f = 2480\ Mhz$
$Ganancia\ de\ la\ antena = 8\ dBi$

Presupuesto de enlace para una distancia entre nodos de 10 km:

$FSL = 20log(2480) + 20log(10) + 32{,}44 = 120{,}32\ dB$
$P_r = 22\ dBm + 8\ dBi + 8\ dBi - 120{,}32\ dB = -82{,}32\ dBm$
$Margen\ RX = Pr - RSL = -82{,}32\ dBm - (-103\ dBm) = 20{,}68\ dBm$
$Margen\ RX\ mínimo = 5{,}25\ dBm + 11log(10) = 16{,}25\ dBm$

Como $Margen\ RX > Margen\ RX\ mínimo$ es posible establecer el enlace con línea de vista (LOS).
Observando el resultado anterior, ya nos damos cuenta de que no se puede transmitir a más de 10 km de distancia utilizando los módulos CC2538+CC2592 con la antena omnidireccional de 8 dBi.
Para este cálculo es importante conocer el radio de Fresnel. A mitad de distancia, el radio de Fresnel será mayor, de manera de que los calcularemos para esa distancia:

$r = 17{,}32\sqrt{\frac{10}{(4*2{,}48)}} = 17{,}38\ m$. Esta es la altura óptima para un 100 % de LOS.

Colocar los módulos a una altura de 17,38 m no es algo factible. Tampoco es factible utilizar la altura mínima, la cual es:
$Altura\ mínima\ de\ las\ antenas = 17{,}38\ m * 0{,}6 = 10{,}43\ m$

Como se observa en los cálculos anteriores, es poco práctico utilizar las antenas a 10,43 m de altura. De manera de que se descarta realizar pruebas con 10 km de distancia entre nodos, quedando de manifiesto de todas maneras, de que es posible lograr comunicación entre los nodos a tal distancia y hasta casi 15 km.

Presupuesto de enlace para una distancia entre nodos de 5 km:

$FSL = 20log(2480) + 20log(5) + 32{,}44 = 114{,}3\ dB$
$P_r = 22\ dBm + 8\ dBi + 8\ dBi - 114{,}3\ dB = -76{,}3\ dBm$
$Margen\ Rx = Pr - RSL = -76{,}3\ dBm - (-103\ dBm) = 26{,}69\ dBm$

$Margen\ RX\ mínimo = 5{,}25\ dBm + 11log(5) = 12{,}93\ dBm$

Como $Margen\ RX > Margen\ RX\ mínimo$ es posible establecer el enlace con línea de vista (LOS).

$r = 17{,}32\sqrt{\frac{5}{(4*2{,}48)}} = 12{,}29\ m$. Esta es la altura óptima para un 100 % de LOS

$Altura\ mínima\ de\ las\ antenas = 12{,}29\ m * 0{,}6 = 7{,}37\ m$

Presupuesto de enlace para una distancia entre nodos de 3 km:

$FSL = 20log(2480) + 20log(3) + 32{,}44 = 109{,}87\ dB$
$P_r = 22\ dBm + 8\ dBi + 8\ dBi - 109{,}87\ dB = -71{,}87\ dBm$
$Margen\ Rx = Pr - RSL = -71{,}87\ dBm - (-103\ dBm) = 31{,}13\ dBm$

$Margen\ RX\ mínimo = 5{,}25\ dBm + 11log(3) = 10{,}49\ dBm$

Como $Margen\ RX > Margen\ RX\ mínimo$ es posible establecer el enlace con línea de vista (LOS).

$r = 17{,}32\sqrt{\frac{3}{(4*2{,}48)}} = 9{,}52\ m$. Esta es la altura óptima para un 100 % de LOS

$Altura\ mínima\ de\ las\ antenas = 9{,}52\ m * 0{,}6 = 5{,}71\ m$

Presupuesto de enlace para una distancia entre nodos de 1 km:

$FSL = 20log(2480) + 20log(1) + 32{,}44 = 100{,}32\ dB$
$P_r = 22\ dBm + 8\ dBi + 8\ dBi - 100{,}32\ dB = -62{,}32\ dBm$
$MargenRx = Pr - RSL = -62{,}32\ dBm - (-103\ dBm) = 40{,}68\ dBm$

$r = 17{,}32\sqrt{\frac{1}{(4*2{,}48)}} = 5{,}49\ m$Esta es la altura óptima para un 100 % de LOS

$Altura\ mínima\ de\ las\ antenas = 5{,}49\ m * 0{,}6 = 3{,}29\ m$

Como se observa en los cálculos, es posible alcanzar distancias muy largas. La única limitante es tener que elevar demasiado las antenas omnidireccionales para lograr tales distancias.

4.5 Ensayos de rendimiento de la red

Para realizar las pruebas de rendimiento se utilizó la herramienta ping6 de Linux, instalada en la Raspberry Pi. Ping6 es la versión para IPv6 de la herramienta ping para IPv4, y nos permite evaluar el estado de la red entre dos nodos dentro de la LAN.
A continuación, se observan las distintas opciones con las que se puede utilizar el comando ping6 desde la terminal de Linux:

```
user@hostname ~ $ ping6 -h
Usage: ping6 [-aAbBdDfhLnOqrRUvV] [-c count] [-i interval] [-I interface]
        [-l preload] [-m mark] [-M pmtudisc_option]
        [-N nodeinfo_option] [-p pattern] [-Q tclass] [-s packetsize]
        [-S sndbuf] [-t ttl] [-T timestamp_option] [-w deadline]
        [-W timeout] destination
```

Para las pruebas de rendimiento de la red de sensores se utilizaron las siguientes opciones de ping6:

- **D:** Imprime el tiempo de marca (Tiempo de Unix + tiempo transcurrido en segundos) antes de cada línea.
- **c (count):** Cantidad de ping a realizar.
- **i (Interval):** intervalo entre envío de paquetes.
- **I (Interface):** Dirección o nombre de interfaz desde donde se realiza el ping.

- **s (packetsize):** Número de bytes de datos a enviar junto con los 8 bytes de encabezado ICMPv6. Por defecto se envían 56 bytes de datos, que se traduce en 64 bytes en total.

El formato de comando ping6 es el siguiente:

ping6 -D -c cantidad -i intervalo -I interfaz -s tamaño dirección_IPV6

Para las pruebas de rendimiento de la red se utilizaron los siguientes valores:

-c	100 (para obtener un promedio aceptable de packet loss y RTTavg)
-i	Se utilizará el valor por default (1 seg.). No hace falta declararlo en este caso.
-I	Tun0 (interfaz de red virtual que utiliza SLIP)
-s	16 (paquete mínimo de 24 bytes),default (paquete default de 64 bytes) y 84 (paquete máximo de 92 bytes)

Observación 1: Un valor de s=0 genera un paquete de 8 bytes de encabezado ICMPv6 sin ningún byte de dato. Enviando un ping6 con hasta 15 bytes de datos (s=15), no se pueden obtener los tiempos de ida y vuelta (time o rtt), y por ende tampoco la desviación estándar (mdev). De manera que, para obtener los datos de tiempos de ida y vuelta, hay que utilizar ping6 con $s \geq 16$.

Observación 2: los nodos no soportan tamaños de paquetes superior a 92 bytes, por lo que los bytes de datos máximo es de 84 (s=84).

Las IPv6 de los nodos en cuestión son las siguientes:
- Raspberry Pi = fd00::1
- Router de borde = fd00::2
- Nodo sensor1 = fd00::10
- Nodo sensor 2 = fd00::20
- Nodo sensor 3 = fd00::30

Parámetros no modificables del radio del SoC cc2538:
- Espacio entre canales = 5 Mhz
- Ancho de banda del canal = 2 Mhz
- Tasa de transferencia = 250 kbps
- Modulación = O-QPSK

4.5.1 Ensayos de los módulos CC2538EM

Los pings fueron realizados desde la Raspberry Pi (fd00::1) hacia el Nodo 1 (fd00::10), con un salto de por medio en el router de borde (fd00::2).

[Raspberry (fd00::1) → RB (fd00::2)] → [Nodo 1 (fd00::10)]

El comando utilizado para este ensayo fue el siguiente:

ping6 -c100 -s** fd00::10

A continuación, se vuelcan los datos obtenidos en las pruebas de campo:

Tamaño del paquete	TTL	RTT avg	Packet loss
32 (s = 24)	63	26,821 ms	0 %
64 (default)	63	34,781 ms	0 %
92 (s = 84)	63	47,868 ms	0 %
Antenas de 1,1 dBi; Pt=+7 dBm; CH=26; d=100 m; Altura=1 m; Saltos = 1			

Tamaño del paquete	TTL	RTT avg	Packet loss
32 (s = 24)	63	26,862 ms	0 %
64 (default)	63	34,937 ms	1 %
92 (s = 84)	63	47,913 ms	2 %
Antenas de 1,1 dBi; Pt=+7 dBm; CH=26; d=300 m; Altura=1 m; Saltos = 1			

Tamaño del paquete	TTL	RTT avg	Packet loss
32 (s = 24)	63	26,863 ms	10 %
64 (default)	63	34,955 ms	15 %
92 (s = 84)	63	47,934 ms	27 %
Antenas de 1,1 dBi; Pt=+7 dBm; CH=26; d=400 m; Altura=1 m; Saltos = 1			

Tamaño del paquete	TTL	RTT avg	Packet loss
32 (s = 24)	---	---	100 %
64 (default)	---	---	100 %
92 (s = 84)	---	---	100 %
Antenas de 1,1 dBi; Pt=+7 dBm; CH=26; d=500 m; Altura=1 m; Saltos = 1			

Los resultados de las pruebas anteriores fueron acorde a los cálculos de presupuesto de enlace. Recordemos que, para transmitir a una distancia de 100 m, necesitamos una altura mínima de 1,04 m; y para transmitir a una distancia de 1 km, necesitamos una altura mínima

de 3,29 m. Esta es la razón por la que a partir de los 400 m la pérdida de paquetes es notable, y a los 500 m ya no es posible establecer el enlace de comunicación.
Ahora presentamos algunos ensayos más, pero para una altura de antenas de 3 m:

Tamaño del paquete	TTL	RTT avg	Packet loss
32 (s = 24)	63	26,861 ms	0 %
64 (default)	63	34,952 ms	0 %
92 (s = 84)	63	47,932 ms	0 %
Antenas de 1,1 dBi; Pt=+7 dBm; CH=26; d=500 m; Altura=3 m; Saltos = 1			

Tamaño del paquete	TTL	RTT avg	Packet loss
32 (s = 24)	63	26,865 ms	1 %
64 (default)	63	34,841 ms	3 %
92 (s = 84)	63	47,856 ms	6 %
Antenas de 1,1 dBi; Pt=+7 dBm; CH=26; d=800 m; Altura=3 m; Saltos = 1			

|||

Tamaño del paquete	TTL	RTT avg	Packet loss
32 (s = 24)	63	26,872 ms	10 %
64 (default)	63	34,841 ms	13 %
92 (s = 84)	63	47,887 ms	18 %
Antenas de 1,1 dBi; Pt=+7 dBm; CH=26; d=1 km; Altura=3 m; Saltos = 1			

Los resultados de las pruebas anteriores fueron acorde a los cálculos de presupuesto de enlace. También se comprueba de ésta manera de que es posible lograr la comunicación entre dos módulos CC2538EM a una distancia de 1 km, tal cual lo indica el fabricante.

4.5.2 Ensayos de los módulos CC2538+CC2592

Los pings fueron realizados desde la Raspberry Pi (fd00::1) hacia el Nodo 1 (fd00::10), con un salto de por medio en el router de borde (fd00::2).

[Raspberry (fd00::1) → RB (fd00::2)] → [Nodo 1 (fd00::10)]

El comando utilizado para este ensayo fue el siguiente:

```
ping6 -c100 -s** fd00::10
```

A continuación, se vuelcan los datos obtenidos en las pruebas de campo:

Tamaño del paquete	TTL	RTT avg	Packet loss
32 (s = 24)	63	26,829 ms	0 %
64 (default)	63	34,788 ms	0 %
92 (s = 84)	63	47,876 ms	1 %
Antenas de 8 dBi; Pt=+22 dBm; CH=26; d=1 km; Altura=3 m; Saltos = 1			

Tamaño del paquete	TTL	RTT avg	Packet loss
32 (s = 24)	63	26,859 ms	0 %
64 (default)	63	34,934 ms	1 %
92 (s = 84)	63	47,922 ms	3 %
Antenas de 8 dBi; Pt=+22 dBm; CH=26; d=3 km; Altura=3 m; Saltos = 1			

Tamaño del paquete	TTL	RTT avg	Packet loss
32 (s = 24)	63	26,874 ms	11 %
64 (default)	63	34,951 ms	16 %
92 (s = 84)	63	47,936 ms	23 %
Antenas de 8 dBi; Pt=+22 dBm; CH=26; d=4 km; Altura=3 m; Saltos = 1			

Tamaño del paquete	TTL	RTT avg	Packet loss
32 (s = 24)	---	---	100 %
64 (default)	---	---	100 %
92 (s = 84)	---	---	100 %
Antenas de 8 dBi; Pt=+22 dBm; CH=26; d=5 km; Altura=3 m; Saltos = 1			

Los resultados de las pruebas anteriores fueron acorde a los cálculos de presupuesto de enlace. Recordemos que, para transmitir a una distancia de 1 km, necesitamos una altura mínima de 3,29 m; y para transmitir a una distancia de 5 km, necesitamos una altura mínima de 7,37 m. Esta es la razón por la que a partir de los 4 km la pérdida de paquetes es notable, y a los 5 km ya no es posible establecer el enlace de comunicación.
Como se observa hasta aquí, los módulos tienen un buen alcance en distancia. Logramos comunicar dos nodos distanciados 4 km entre sí, lo cual es una referencia muy buena.
En el siguiente ensayo se agregó un salto en el medio (entre el router de borde y el nodo al cuál le realizamos los pings), logrando de esa manera un total de dos saltos. La distancia de separación entre cada nodo es de 3 km, lo cual resulta en una separación de 6 km entre el router de borde y el Nodo 2:

[Raspberry (fd00::1) → RB (fd00::2)] → [Nodo 1 (fd00::10)] → [Nodo 2 (fd00::20)]

Se eligió una distancia de 3 km porque a esa distancia se logró una buena comunicación. Y además, utilizando esa distancia y una altura de 3 m, el router de borde no se puede enlazar directamente con el nodo 2 que se encuentra a 6 km. Y va a preferir enlazarse contra el nodo 1, por lo que la ruta para llegar desde la Raspberry al Nodo 2 será a través del Nodo 1.
El comando utilizado para este ensayo fue el siguiente:

ping6 -c100 -s** fd00::20

Tamaño del paquete	TTL	RTT avg	Packet loss
32 (s = 24)	62	53,721 ms	0 %
64 (default)	62	69,869 ms	2 %
92 (s = 84)	62	95,847 ms	5 %
Antenas de 8 dBi; Pt=+22 dBm; CH=26; d=6 km; Altura=3 m; Saltos = 2			

El ensayo realizado nos da un indicativo de lo factible que es cubrir grandes distancias simplemente con el agregado de un nodo como salto adicional cada 3 km.

4.6 Ensayo final

Con el fin de encontrar puntos de falla o detalles que permitan mejorar el proyecto, hemos realizado varios ensayos. Los ensayos consistieron desde dejarlos en prueba en una habitación, en un patio, en un techo, hasta concluir el ensayo final en un campo.

Figura 4.8: Instalación del sistema en el techo de un domicilio

Pero luego de una serie de varios ensayos de todo tipo (evaluar comportamiento de la red, alcances en distancia, ciclos de carga y descarga de las baterías, ensayos a escala a mínima potencia, etc), pasamos a realizar la prueba final en un campo llamado Establecimiento Rukalen, ubicado en la Localidad de Cabildo, Partido de Bahía Blanca.

Figura 4.9: Entrada al establecimiento Rukalen

En dicho establecimiento crían vacas, y nos permitieron instalar los nodos en un sector del campo, el cual estaba protegido y sin animales (no iban a pasar animales allí hasta dentro de varios días). Ya con todos los elementos preparados de antemano, procedimos a enterrar los postes de 2,25 m de altura para instalarles los nodos.

Figura 4.10: Elementos utilizados para la prueba en campo

En las siguientes Figuras presentamos el prototipo de un nodo completo:

Figura 4.11: Vista interior del nodo

Figura 4.12: Vista exterior del nodo

Para los ensayos solamente dejamos conectado el sensor de temperatura del suelo como se observa a continuación:

Figura 4.13: Nodo con sólo el sensor de temperatura

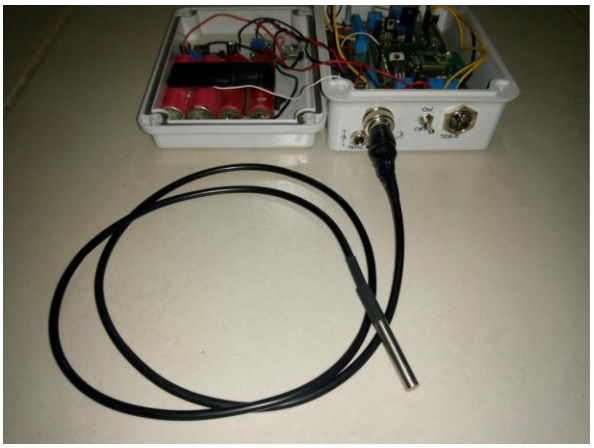


Figura 4.14: Vista cercana al nodo y al sensor de temperatura

En la Figura 4.15 se muestra como quedó instalado el Nodo Gateway (Modem 3G, Raspberry Pi y el Router de Borde de la red IPv6), y el Nodo 1:

En esta oportunidad se instaló el Nodo 1 al lado del Nodo Gateway, porque no disponíamos de más paneles solares. De manera que el panel de 20 W recargó las baterías de ambos nodos durante los días de ensayo.

Figura 4.15: Instalación en campo del Nodo Gateway y el Nodo 1

El Nodo Gateway posee un módulo CC2538+CC2592 con una antena de 8 dBi, y el Nodo 1 posee un módulo CC2538EM con la antena PCB de 1,1 dBi.
Según el pronóstico iba a llover durante el transcurso del ensayo final. De manera que se decidió a último momento cubrir a todos los nodos con elementos que conseguimos en el sitio. Si bien el gabinete de los nodos consta de cajas estancas, queríamos evitar posibles inconvenientes con la entrada de agua por las antenas omnidireccionales.

Figura 4.16: Nodo cubierto con nylon para evitar el traspaso de agua en caso lluvia

Figura 4.17: Vista cercana al nodo cubierto con nylon

La instalación del Nodo 2 quedó a 200 m del conjunto Nodo Gateway + Nodo 1. Como se observa en la siguiente imagen, el Nodo 2 quedó con el panel solar de 10 W:

Figura 4.18: Instalación en campo del Nodo 2

Como estábamos limitados a un automóvil como medio de transporte, no pudimos llevar postes de más de 2,25 m de altura. Por lo que medio metro del poste quedaba enterrado y en la punta del mismo iba el panel solar. La antena omnidireccional de 8 dBi quedaba aproximadamente a 1,5 m de altura. Dicha altura nos limitaba la distancia máxima entre nodos a la cual podíamos alejar a los nodos entre sí.
La instalación del Nodo 3 quedó a 200 m del conjunto Nodo Gateway + Nodo 1 y a 200 m del Nodo 2. Conformando así un triángulo entre todos los nodos. Como se observa en la siguiente imagen, el Nodo 3 quedó con el panel solar de 1W:

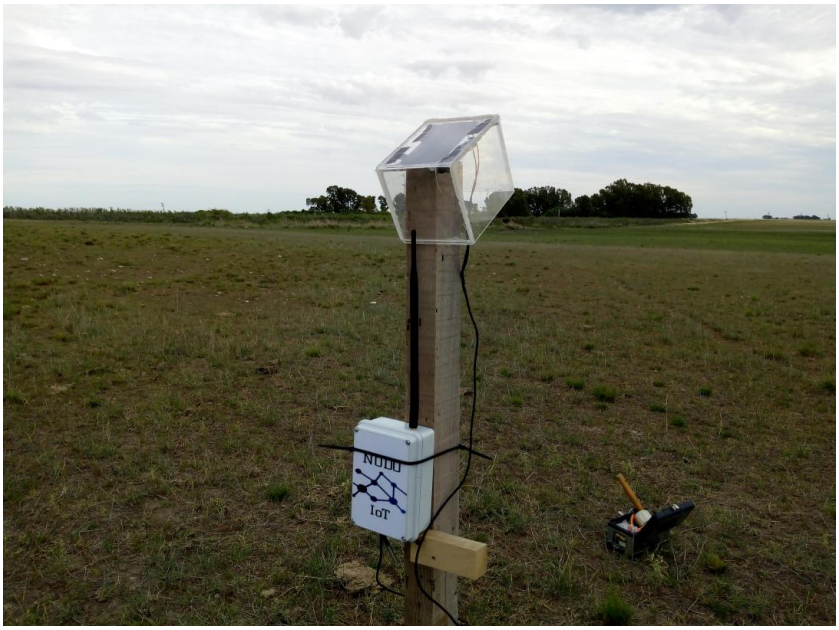
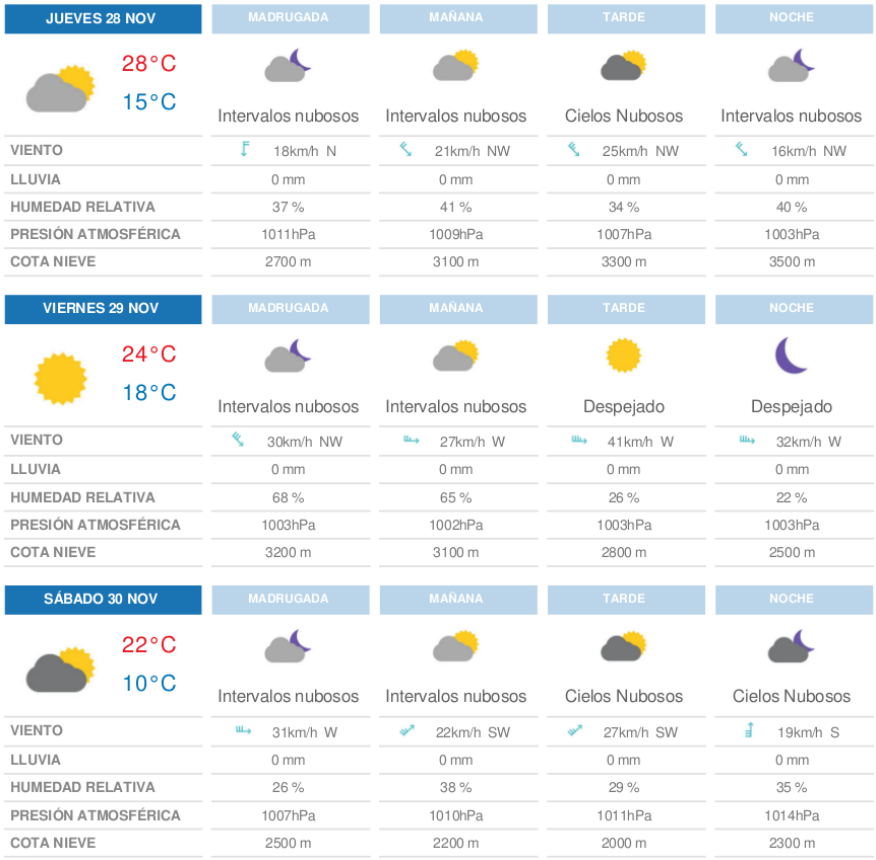


Figura 4.19: Instalación en campo del Nodo 3

El ensayo en campo se inició el día 28 de Noviembre del 2019 a las 16:15 hs y finalizó el 30 de Noviembre a las 21:35 hs.
Como se dijo en un principio, el objetivo del ensayo era evaluar el comportamiento de la red, evaluar la recolección de datos de los distintos nodos y evaluar si los datos eran subidos correctamente a la plataforma de monitoreo Thingspeak. Dicho objetivo fue cumplido, ya que ha funcionado todo perfectamente como estaba previsto. Es decir, pudimos observar desde nuestros hogares en tiempo real, las mediciones que estaban efectuando cada uno de los nodos.
El único inconveniente que tuvimos, fue que, por haber estado todos los días nublados, la batería del Nodo Gateway se agotó al tercer día de ensayo. Sabíamos que la batería se podía llegar a agotar si tocaban todos los días nublados; ya que no estábamos utilizando la batería adecuada en autonomía, sino que era la que disponíamos en ese entonces para realizar los ensayos.
Como decisión de último momento en el día de instalación de los nodos, conectamos el Nodo Gateway y el Nodo 1 directamente a los bornes de la batería (al mismo tiempo, también quedaba directamente conectado a la salida del módulo cargador de batería). Esta decisión fue tomada por los días nublados que iban a tocar durante el ensayo, y queríamos que el Nodo Gateway permaneciera encendido el mayor tiempo posible hasta agotar la batería por completo (lo cual no es bueno para la batería).
El pronóstico para los días de ensayo fue el siguiente:



JUEVES 28 NOV	MADRUGADA	MAÑANA	TARDE	NOCHE
28°C / 15°C	Intervalos nubosos	Intervalos nubosos	Cielos Nubosos	Intervalos nubosos
VIENTO	18km/h N	21km/h NW	25km/h NW	16km/h NW
LLUVIA	0 mm	0 mm	0 mm	0 mm
HUMEDAD RELATIVA	37 %	41 %	34 %	40 %
PRESIÓN ATMOSFÉRICA	1011hPa	1009hPa	1007hPa	1003hPa
COTA NIEVE	2700 m	3100 m	3300 m	3500 m

VIERNES 29 NOV	MADRUGADA	MAÑANA	TARDE	NOCHE
24°C / 18°C	Intervalos nubosos	Intervalos nubosos	Despejado	Despejado
VIENTO	30km/h NW	27km/h W	41km/h W	32km/h W
LLUVIA	0 mm	0 mm	0 mm	0 mm
HUMEDAD RELATIVA	68 %	65 %	26 %	22 %
PRESIÓN ATMOSFÉRICA	1003hPa	1002hPa	1003hPa	1003hPa
COTA NIEVE	3200 m	3100 m	2800 m	2500 m

SÁBADO 30 NOV	MADRUGADA	MAÑANA	TARDE	NOCHE
22°C / 10°C	Intervalos nubosos	Intervalos nubosos	Cielos Nubosos	Cielos Nubosos
VIENTO	31km/h W	22km/h SW	27km/h SW	19km/h S
LLUVIA	0 mm	0 mm	0 mm	0 mm
HUMEDAD RELATIVA	26 %	38 %	29 %	35 %
PRESIÓN ATMOSFÉRICA	1007hPa	1010hPa	1011hPa	1014hPa
COTA NIEVE	2500 m	2200 m	2000 m	2300 m

Figura 4.20: Pronostico del tiempo para los días del ensayo

Ya contextuado como se realizaron los ensayos, a continuación, pasamos a detallar los resultados de todos los nodos:

4.6.1 Resultados de ensayo del Nodo Gateway

Los datos del Nodo Gateway son los siguientes:
- CPU central: Raspberry PI 3 Model B
- Módem 3G: Huawei E176
- Módulo: CC2538+CC2592. Funciona como Router de Borde de toda la red de sensores.
- Alimentación: Posee un regulador Step Down alimentado desde un controlador de carga solar LS3024B. Dicho controlador es alimentado a través de un panel solar de 20 W policristalino (17,8 V), y el mismo panel le brinda carga a una batería AGM Visión CP12120 (12 V - 12 Ah).

Según nuestros cálculos sólo tenemos 18 hs de autonomía, pero debido a que utilizamos un controlador que no carga al 100% la batería, estimamos menos de 18 horas de autonomía.
- Antena: onminidirecional de 8 dBi.
- Consumo: 3 Wh.
-Ubicación: https://maps.google.com/maps?q=-38.501515%2C-61.918303&z=17&hl=es

El Nodo Gateway quedó instalado el día 28 de Noviembre a las 16:15hs. Dicho día estuvo un poco soleado, por lo que la batería se pudo cargar algo. Los días 29 y 30 tocaron todo el día nublado, apagándose de esta manera el Nodo Gateway a las 21:35 hs. Cabe recordar que toda la red de sensores queda incomunicada al apagarse el Nodo Gateway.

El día 31 tocó soleado y ya sabíamos de antemano que el sistema no iba a arrancar nuevamente. Esto se debe a que habíamos conectado el Nodo Gateway directamente a la batería, llegando ésta a niveles muy bajos de carga. Debido a ello, el controlador de carga solar se encontraba inhabilitado e incapaz de volver a cargar la batería debido a su sistema de protección interno de no empezar a cargar una batería con niveles muy bajos de tensión.
Luego de unos días viajamos a Cabildo para retirar la red completa. Comprobamos que el panel solar estaba alimentando al controlador de carga con un voltaje de 20,8 V (Figura 4.21). Al mismo tiempo la batería se encontraba con un voltaje de 2,7 V (Figura 4.22), el cuál es muy bajo y el módulo cargador queda inhibido de cargar la batería. Por tal decisión errónea de último momento (conectar el Nodo Gateway directamente a la batería), la batería se dañó de manera permanente por llegar a ese nivel tan bajo de tensión.

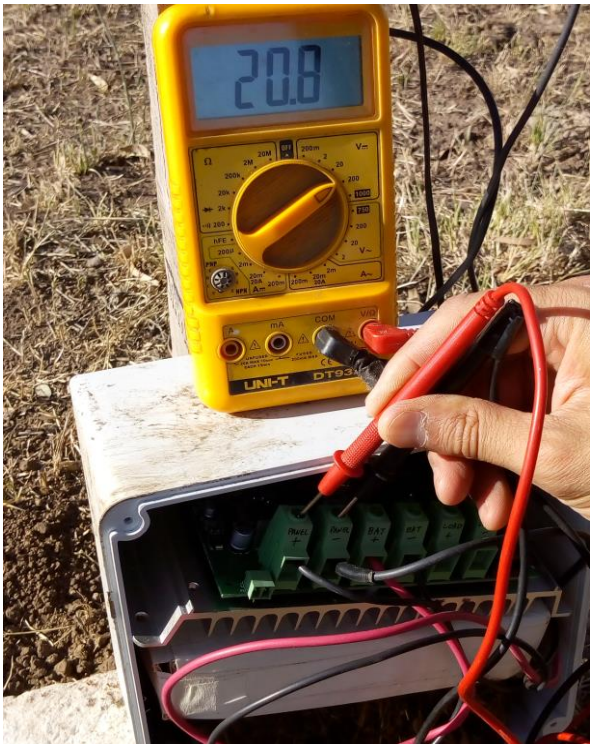


Figura 4.21: Voltaje proporcionado por el panel solar

Figura 4.22: Nivel bajo de la batería AGM

4.6.2 Resultados de ensayo del Nodo 1

Los datos del Nodo 1 son los siguientes:
- Módulo: CC2538EM (de Texas Instruments), conectado por medio de placa adaptadora.
- Alimentación: Posee baterías de Litio para ir cargando con un panel solar de 1 W policristalino (5 V). Para el ensayo no utilizamos el panel de 1 W, ya que el Nodo 1 se conectó a la salida de 5 V del Step Down del Nodo Gateway con fines de utilizar la misma batería.
- Antena: omnidireccional de 1,1 dBi (antena integrada en el PCB del módulo CC2538EM).
- Consumo: 0,2 Wh.
- Ubicación: https://maps.google.com/maps?q=-38.501515%2C-61.918303&z=17&hl=es
- Canal de Thingspeak: https://thingspeak.com/channels/762863

Las variables sensadas fueron las siguientes:
1- Temperatura del suelo
2- Carga de batería
3- Carga de batería AGM del Nodo Gateway

El Nodo 1 quedó instalado el día 28 de Noviembre y se encendió a las 16:15 hs. Este día no hubo mensajes perdidos hacia Thingspeak. Llegaron todos los mensajes con los 8 valores que el nodo transmite.
El día 29 hubo solamente 2 mensajes perdidos hacia Thingspeak, uno fue por pérdida de comunicación entre el Router de Borde y Nodo 1, y el otro por desconexión del módem 3G del Nodo Gateway.
El día 30 no hubo mensajes perdidos hacia Thingspeak, llegaron todos los mensajes con los 8 valores hasta que se apagó el Nodo Gateway a las 21:35 hs.

4.6.3 Resultados de ensayo del Nodo 2

Los datos del Nodo 2 son los siguientes:
- Módulo: CC2538+CC2592
- Alimentación: Posee baterías de Litio para ir cargando con un panel solar de 1 W policristalino (5 V).
- Antena: omnidireccional de 8 dBi.
- Consumo: 0,2 Wh.
-Ubicación: https://maps.google.com/maps?q=-38.5025191%2C-61.9192658&z=17&hl=es
- Canal de Thingspeak: https://thingspeak.com/channels/784758

Las variables sensadas fueron las siguientes:
1- Temperatura del suelo
2- Carga de batería

El Nodo 2 quedó instalado el día 28 de Noviembre y se encendió a las 16:15 hs. Este día no hubo mensajes perdidos hacia Thingspeak. Llegaron todos los mensajes con los 8 valores que el nodo transmite.
El día 29 hubo solamente 1 mensaje perdido hacia Thingspeak, y fue por desconexión del módem 3G.
El día 30 no hubo mensajes perdidos hacia Thingspeak, llegaron todos los mensajes con los 8 valores hasta que se apagó el Nodo Gateway a las 21:35 hs.

4.6.4 Resultados de ensayo del Nodo 3

Los datos del Nodo 3 son los siguientes:
- Módulo: CC2538+CC2592
- Alimentación: Posee baterías de Litio para ir cargando con un panel solar de 10 W monocristalino (17,5 V).
- Antena: omnidireccional de 8 dBi.
- Consumo: 0,2 Wh.
-Ubicación: https://maps.google.com/maps?q=-38.5026033%2C-61.9177967&z=17&hl=es
- Canal de Thingspeak: https://thingspeak.com/channels/785601

Las variables sensadas fueron las siguientes:
1- Temperatura del suelo
2- Carga de batería
El Nodo 3 quedó instalado el día 28 de Noviembre y se encendió a las 16:15hs. Este día no hubo mensajes perdidos hacia Thingspeak. Llegaron todos los mensajes con los 8 valores que el nodo transmite.
El día 29 hubo solamente 1 mensaje perdido hacia Thingspeak, y fue por desconexión del módem 3G.
El día 30 no hubo mensajes perdidos hacia Thingspeak, llegaron todos los mensajes con los 8 valores hasta que se apagó el Nodo Gateway a las 21:35 hs.

5 Conclusiones

En este proyecto hemos podido diseñar e implementar una red inalámbrica de sensores de manera exitosa para aplicarlo a un escenario del ámbito rural, monitoreando variables de interés para una persona/operario que trabaje en un establecimiento rural. Con ello le permitiría al interesado realizar algún tipo de acción correctiva o de prevención en función de estas variables, ya que dicha persona tendrá toda la información al alcance de sus manos mediante un ordenador o un smartphone.

Se logró implementar en cada nodo el desarrollo de un código, el cual corre sobre el sistema operativo ContikiOS. Durante el desarrollo de este código se estudió y se hicieron modificaciones al sistema operativo ContikiOS, modificando tanto parámetros de la pila de protocolos como de los propios del módulo de radio del cc2538. Cabe destacar que tanto ContikiOS como el entorno de desarrollo Eclipse son de libre distribución. Esto fue un punto fundamental a la hora de seleccionar las herramientas para comenzar a desarrollar la aplicación, y no utilizar otras que requieran de licencia paga. Fue de nuestro agrado realizar todo este proyecto utilizando estas herramientas, además de otras como el compilador GCC, Debian, GitHub, entre otras. Si bien fue muy difícil crear nuestro entorno de desarrollo, valió la pena, ya que todo nuestro trabajo quedó documentado como guía para futuros desarrollos. En Eclipse pudimos crear un entorno totalmente funcional, con capacidades de programar, depurar, disponer de puerto serie contra el cc2538, flashear el cc2538, entre otros.

Previamente a realizar el ensayo final, llevamos a cabo pruebas exhaustivas de distancia para garantizar que los nodos fueran capaces de comunicarse de manera efectiva tanto en entornos con obstáculos como en campo abierto. Gracias a que direccionamos a cada nodo con IPv6, pudimos llevar a cabo todas estas pruebas de distancia con ping. Pudimos probar en la práctica de que podemos lograr más alcance en km intercalando nodos en el medio, y de esa manera poder corroborar que el protocolo de ruteo RPL funciona correctamente. Esto validó la robustez y confiabilidad de nuestros nodos.

Un aspecto a destacar es que el sistema de alimentación de cada nodo funciona correctamente y evita que se queden sin energía, lo que le da una buena autonomía al sistema.

Durante las pruebas llevadas a cabo en el Establecimiento Rukalen se pudo verificar el funcionamiento del sistema completo. Una vez instalados los Nodos Sensores y el Nodo Gateway, el sistema inició con la recolección de datos de manera automática durante los días de ensayo.

Debemos mencionar que nuestra red inalámbrica de sensores puede funcionar en cualquier situación o problemática del ámbito rural, ya que el hardware se diseñó de tal manera de poder sustituir con facilidad los sensores, de acuerdo a lo que se requiera sensar. Además, no hay límite en cuanto a la distancia que deba cubrir la red de sensores, basta con agregar más nodos con línea de vista entre ellos para abarcar un área más extensa.

De esta manera, culminamos con este proyecto de manera exitosa, ya que la red inalámbrica de sensores es robusta, y es apta para cubrir muchas de las necesidades del sector rural de hoy en día.

Bibliografía

Wiki de ContikiOS. (s.f) Recuperado el 31, de Octubre, de 2023, de https://www.github.com/contiki-os/contiki/wiki

Agradecimientos

Queremos expresar nuestro agradecimiento a todas las personas y entidades que han contribuido a la realización de este proyecto y a nuestro recorrido en la carrera de Ingeniería Electrónica.
En primer lugar, agradecemos a nuestras familias por su apoyo inquebrantable y su comprensión a lo largo de nuestra educación y este proyecto en particular. Su aliento y paciencia fueron fundamentales para nuestra perseverancia.

A la Universidad Tecnológica Nacional de Bahía Blanca, y en específico al Departamento de Ingeniería Electrónica, le agradecemos por proporcionarnos herramientas y conocimientos necesarios para enfrentar desafíos como este. Nos sentimos afortunados de haber tenido la oportunidad de aprender y crecer en este entorno académico.

No podemos pasar por alto la contribución invaluable de nuestros profesores, en particular, el Mg. Ing. Guillermo Friedrich y el Ing. Adrián Laiupa, cuya experiencia y orientación fueron esenciales en la culminación de este proyecto. Sus enseñanzas han dejado una huella indeleble en nuestro camino.

Además, deseamos agradecer a Ignacio, propietario del Establecimiento Rukalen, por su generosidad al permitirnos llevar a cabo pruebas en su terreno. Su colaboración fue esencial para validar nuestras soluciones en condiciones de campo real.

Finalmente, agradecemos a todas las personas que han confiado en nuestro trabajo y en la visión de un futuro más conectado y eficiente a través de la tecnología IoT. Este logro no hubiera sido posible sin la suma de todos estos esfuerzos.

¡Gracias a todos por ser parte de este camino!