.

UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA

La Universidad Católica de Loja

ÁREA TÉCNICA

TÍTULO DE INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

Diseño e implementación de una plataforma hardware para aplicaciones de redes inalámbricas de sensores

TRABAJO DE TITULACIÓN

AUTORES: Bosmediano Carrión, María Soledad
Ochoa Quezada, Pablo David

DIRECTOR: Quiñones Cuenca, Manuel Fernando

LOJA-ECUADOR

2017

Septiembre, 2017

.

APROBACIÓN DEL DIRECTOR DEL TRABAJO DE TITULACIÓN

Ingeniero

Manuel Fernando Quiñones Cuenca

DOCENTE DE LA TITULACIÓN

De mi consideración:

El presente trabajo de titulación: Diseño e implementación de una plataforma hardware para aplicaciones de redes inalámbricas de sensores realizado por Bosmediano Carrión María Soledad y Ochoa Quezada Pablo David ha sido orientado y revisado durante su ejecución, por cuanto se aprueba la presentación del mismo.

Loja, marzo de 2017

f)……..

.

DECLARACIÓN DE AUTORÍA Y CESIÓN DE DERECHOS

Nosotros, Bosmediano Carrión María Soledad y Ochoa Quezada Pablo David, declaramos ser autores del presente trabajo de titulación: Diseño e implementación de una plataforma hardware para aplicaciones de redes inalámbricas de sensores, de la titulación de Electrónica y Telecomunicaciones, siendo el Ing. Manuel Fernando Quiñones Cuenca director del presente trabajo; y eximimos expresamente a la Universidad Técnica Particular de Loja y a sus representantes legales de posibles reclamos o acciones legales. Además, certificamos que las ideas, conceptos, procedimientos y resultados vertidos en el presente trabajo investigativo son de nuestra exclusiva responsabilidad.

Adicionalmente declaramos conocer y aceptar la disposición del Art. 88 del Estatuto Orgánico de la Universidad Técnica Particular de Loja que en su parte pertinente textualmente dice: “Forman parte del patrimonio de la Universidad la propiedad intelectual de investigaciones, trabajos científicos o técnicos y tesis de grado que se realicen a través, o con el apoyo financiero, académico o institucional (operativo) de la Universidad.”

f):.....

Autor: María Soledad Bosmediano Carrión

Cédula: 1714503263

f).....

Autor: Ochoa Quezada Pablo David

Cédula: 1103860043

.

DEDICATORIA

Este trabajo va dedicado en primer lugar a la mujer que más admiro en mi vida, mi madre Lorena, por ser el motor de mi existencia, mi mejor cómplice y mi mejor amiga; a mi padre Vinicio, que aunque no esté físicamente conmigo siempre está presente cuidándome desde el cielo; a mi pequeña hermana Evan, que es la razón por la que día a día me esfuerzo por superarme personal y académicamente y finalmente y sin desmerecer, a mi gran amigo y eterno compañero de proyectos Pablo, sin su sincera e incondicional amistad todo el esfuerzo sembrado no tuviera frutos.

María Soledad

Dedico este trabajo principalmente a Dios, por haberme dado la vida y permitirme lograr una de mis metas más anheladas. A mi madre, que lamentablemente falleció, sin embargo, ella siempre está cuidándome desde el cielo. A mi abuelita, una mujer muy trabajadora a quien sigo como ejemplo desde que era un niño y por ser el pilar más importante de mi vida ya que supo demostrarme su cariño y apoyo incondicional en los momentos más duros y felices de mi vida. A mi familia en general en especial a mi padre y tíos quienes me brindaron su apoyo moral y me forjaron de valores; a mi pareja por entenderme en todo momento, siendo mi apoyo incondicional, inspiración, motivación y el ingrediente perfecto para poder lograr alcanzar esta muy merecida victoria. Y no podía faltar mi querida amiga y compañera de trabajo Soledad, por brindarme su sincera amistad, confianza, cariño, apoyo y permitirme conformar el mejor equipo de trabajo para batallar día a día y finalmente lograr esta importante meta de mi vida.

Pablo David

.

AGRADECIMIENTO

A Dios, por ser el sustento de nuestras vidas.

A nuestras madres Lorena y Leonila, por su apoyo incondicional, paciencia, generosidad y confianza brindada durante todo este proceso de formación profesional.

A nuestro director de tesis y amigo: Ing. Manuel Quiñones, quien supo brindar su confianza y tiempo para guiarnos en la realización de este trabajo.

A nuestra familia, amigos y compañeros.

María Soledad y Pablo.

.

ÍNDICE DE CONTENIDOS

.

.

.

LISTA DE FIGURAS

.

.

.

LISTA DE TABLAS

.

RESUMEN

En el presente trabajo de titulación se ha desarrollado el diseño e implementación de una plataforma hardware para aplicaciones de Redes de Sensores Inalámbricos (WSN) e Internet de las cosas (IoT). El sistema es aplicable en el monitoreo de variables medioambientales, redes de distribución de agua, entornos inteligentes, control de consumo energético, estudio de invernaderos y estacionamiento vehicular.

Para cumplir con este propósito se realiza una investigación de las diferentes plataformas de hardware existentes en el mercado, buscando así desarrollar una arquitectura apropiada que integre en el diseño del circuito componentes de alto desempeño.

PALABRAS CLAVE: Redes de Sensores Inalámbricos, Internet de las cosas, plataforma, hardware abierto, sensores.

.

ABSTRACT

In the present university work was developed the design and implementation of a hardware platform for applications of wireless sensor networks and the Internet of Things. The system is applicable in the monitoring of environmental variables, water distribution networks, intelligent environments, control of energy consumption, study of greenhouses, parking vehicular, among others.

In order to fulfill this purpose is carried out an investigation of the different existing hardware platforms in the market, seeking to develop an appropriate architecture that integrates in the circuit of the platform the more robust hardware components.

KEYWORDS: WSN, IoT, plataform, open hardware, sensors.

.

ACRÓNIMOS

HW	*Hardware*
SW	*Software*
WSN	*Wireless Sensor Network*
IoT	*Internet of Things*
RAM	*Random Access Memory*
RTC	*Real Time Clock*
FPGA	*Field Programmable Gate Array*
FTDI	*Future Technology Devices International*
I^2C	*Inter-Integrated Circuit*
GPRS	*General Packet Radio Service*
GPS	*Global Positioning System*
GPIO	*General Purpose Input Output*
SPI	*Serial Peripheral Interface*
USB	*Universal Serial Bus*
Wi-Fi	*Wireless Fidelity*
UART	*Universal Asynchronous Receiver/Transmitter*
PCB	*Printed Circuit Board*
GPL	*General Public License*
SAP	*Sistemas, Aplicaciones y Productos en Procesamiento de Datos*
RPL	*Routing Protocol for Low Power*
TCP	*Transmission Control Protocol*
API	*Application Programming Interface*
SERCOM	*Serial Communication Module*
SGBD	*Sistema Gestor de Base de Datos*
EIC	*External Interrupt Controler*
ADC	*Analog Digital Converter*
DAC	*Digital Analog Converter*
AC	*Analog Comparators*
MISO	*Master Input Slave Output*
MOSI	*Master Output Slave Input*
SS	*Slave Select*
CI	*Circuito Integrado*
SD	*Secure Digital*

.

INTRODUCCIÓN

El campo de aplicación de las Redes de Sensores Inalámbricos en la actualidad, abarca múltiples áreas como: energía, salud, seguridad, agricultura, transporte, industria o cualquier otro entorno dónde se requiera el uso factible de la información por el usuario. Con el constante desarrollo de la tecnología y la inmersión de la misma en el mundo del IoT, ha surgido la necesidad de contar con diseños de plataformas capaces de proporcionar seguridad, fiabilidad y robustez en su arquitectura hardware y software. El presente trabajo diseña e implementa una plataforma hardware modular que adapta estas características y optimiza recursos limitados como: memoria Flash, memoria RAM, precisión con el RTC, velocidad de procesamiento, facilidad de montaje y bajo consumo energético.

El proyecto tiene la siguiente estructura: el primer capítulo plantea las razones por las que este prototipo es factible, los objetivos que se busca alcanzar y la metodología que se procura utilizar. En el segundo capítulo se hace una pequeña conceptualización sobre las temáticas más relevantes de las WSNs e IoT, además de una valoración técnica de las plataformas hardware comerciales y de código abierto. En el tercer capítulo se especifica la arquitectura y diseño del prototipo, detallando los componentes del bloque de control de energía, adquisición, comunicación y de protección de la mota o nodo. Además, se realiza una descripción técnica de toda la plataforma desarrollada. Finalmente, en el cuarto capítulo, se trata la implementación, pruebas de desempeño de la placa y presupuesto referencial del proyecto, a más de especificar recomendaciones, tomando en cuenta las dificultades y problemas encontrados en el diseño e implementación del mismo.

La importancia de esta investigación está en promover al desarrollo de hardware propio, capaz de ser implementado en ambientes de condiciones climáticas adversas. La plataforma diseñada fue puesta a prueba en escenarios de laboratorio para realizar su evaluación y validación, para así asegurar que la plataforma puede ser usada en cualquier entorno inteligente.
La metodología usada para la realización del proyecto consiste de cuatro fases: fase de investigación, en la que se tomó en cuenta las especificaciones técnicas requeridas en el sistema; una fase de desarrollo, en la que se obtuvo el diseño y visión general de la plataforma; una fase de implementación; donde se materializó el prototipo y por último una fase de pruebas y validación, en la que se realizó evaluaciones a detalle por cada bloque que conforma la plataforma.

.

.

CAPÍTULO 1

1. ALCANCE DE LA INVESTIGACIÓN

.

1.1 Objetivos

1.1.1 Objetivo general

Diseñar e implementar una plataforma hardware genérico para aplicaciones de Redes de Sensores Inalámbricos (WSN) e Internet de las Cosas (IoT) mediante el uso de dispositivos programables y de comunicación.

1.1.2 Objetivos específicos

- Realizar una revisión del estado actual de las plataformas de hardware abierto y comerciales para aplicaciones en WSNs e IoT.
- Desarrollar la arquitectura de la plataforma y determinar sus componentes.
- Diseñar el esquema y la placa de circuito impreso para la implementación.
- Implementar los componentes hardware en la placa de circuito impreso.
- Validar el funcionamiento de la plataforma diseñada mediante software en un ambiente de laboratorio.
- Analizar los resultados obtenidos y puntualizar las diferentes aplicaciones de la plataforma.

1.2 Justificación

La emergente digitalización del mundo en la actualidad ha llevado al hombre a buscar herramientas de alta calidad y precisión, sobre todo en países desarrollados donde es necesario el uso de redes de computadoras para la recolección y procesamiento de grandes cantidades de datos. En Ecuador, también se ha visto la necesidad de desplegar redes de sensores inalámbricos para monitorear diferentes magnitudes físicas del entorno. En la ciudad de Loja, por ejemplo, se han implementado estaciones meteorológicas equipadas con plataformas hardware (HW) de recursos abiertos y comerciales como Arduino, Rasperry, Waspmote o SparkFun. No obstante, se han encontrado algunas falencias en cuanto a capacidad de almacenamiento, consumo y abastecimiento de energía, grado de protección, compatibilidad y facilidad de ensamblaje. Otra de las limitaciones que implica la selección de una plataforma, es encontrar diseños desarrollados con software (SW) privado, que dificultan a los investigadores la implementación de una idea por el escaso soporte y recursos, e inclusive en algunos casos imposibilita la compatibilidad en hardware.

.

La mayoría de las plataformas de desarrollo usan TinyOS; un sistema operativo de código abierto diseñado para aplicaciones de redes de sensores que posee una interfaz simple. Sin embargo, la adaptación de nuevos sensores a las tarjetas se torna un trabajo tedioso, debido a que la documentación es limitada y aun siendo plataformas con soporte no son funcionales en su totalidad [1].

La motivación principal del trabajo es, por lo tanto, dar una solución técnica y económicamente viable a las necesidades previstas mediante el desarrollo de una plataforma HW modular basada en tecnología inalámbrica. La plataforma integra un microcontrolador *Atmel* SAMD21 de bajo consumo energético con capacidad de almacenamiento dos veces mayor a la plataforma comercial de Libelium y en memoria RAM tres veces superior. Para garantizar su interoperabilidad cuenta con un sistema híbrido de dos ranuras XBee para módulos de comunicación inalámbrica, lo que lo hace disponible para las siguientes tecnologías: Zigbee, Wi-Fi, GPRS, GPS y Bluetooth. En cuanto al grado de protección de la mota se propuso el uso de conectores IP67 e IP68 de fácil ensamblaje que garantizan el funcionamiento del sistema en ambientes climáticos adversos. Además, se procura que el diseño de la plataforma cuente con interfaces de hardware estándar y software de código abierto para facilitar la adaptación de sensores.

Uno de los proyectos que sirvió de preámbulo para nuestro trabajo es el repositorio web OpenWSN [2], el cual cuenta con gran cantidad de información para implementaciones de código abierto utilizando plataformas de hardware y software. La creación del prototipo basado en tecnología inalámbrica reúne requisitos propios de aplicaciones de IoT; un ejemplo claro es el proyecto OpenMote que proporciona las herramientas necesarias (OpenBase, OpenBatery, OpenMote-Cc2538) para poner en funcionamiento aplicaciones de detección y control habilitadas por internet [3].

1.3 Metodología

En este apartado se describe la forma como se dará cumplimiento a los objetivos plateados para el presente trabajo de fin de titulación. Se utilizó la metodología en V [4], la cual define algunas etapas de desarrollo como indica la figura 1.1:

.

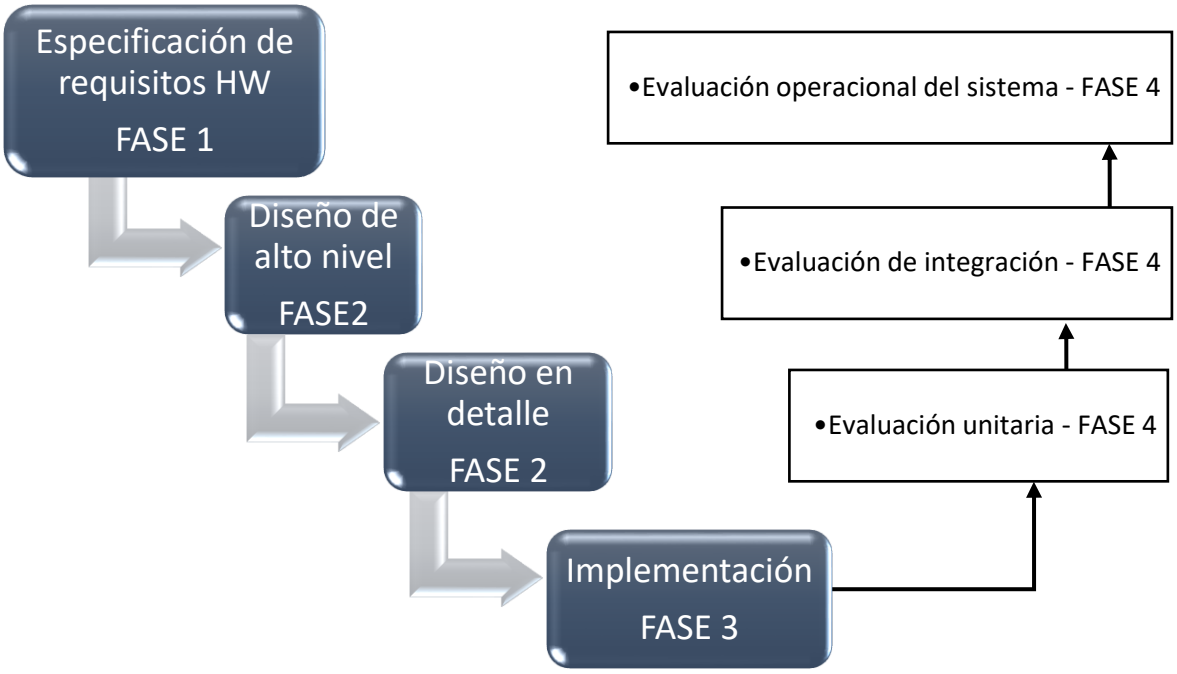


Figura. 1.1 Fases de desarrollo del proyecto.
Fuente: [4]
Elaboración: Autores

Especificaciones o Fase 1: Se define y documenta detalladamente los diferentes requisitos del sistema a desplegar, identificando con valores numéricos a cada módulo de la plataforma [4], por ejemplo capacidad de memoria RAM y Flash, número de puertos e interfaces del microcontrolador a seleccionar. Además, se enfatizará en las limitaciones que se presenten en el desarrollo del prototipo y principalmente en dar cumplimiento a los objetivos.

Diseño o fase 2: se divide en dos subprocesos; el diseño global o también llamado diseño de alto nivel. El objetivo es obtener un diseño y visión general del sistema tanto en software como en hardware. Para ello se estable la arquitectura general del sistema y las herramientas que sirvan de apoyo para efectuar el proyecto. El diseño en detalle, en cambio, consiste en definir cada bloque de la fase anterior, distinguiendo las diferentes partes que conforman la plataforma como son: módulos de procesamiento, energía, sensores y comunicación [4].

Implementación o Fase 3: en esta fase se materializa el diseño en software y hardware por cada unidad que conforme el sistema [4].

.

Evaluación o Fase 4: se subdivide en tres subprocesos: test unitario donde se verifica cada módulo HW y SW de forma separada, comprobando que funcione adecuadamente; test de integración, en esta fase se integran los distintos módulos que forman el sistema. Como en el caso anterior, ha de generarse un documento de pruebas. Por una parte, se debe comprobar todo el funcionamiento correcto del sistema, y por otra, en caso de tratarse con un sistema tolerante a fallos, debe verificarse que ante la presencia de un fallo, persiste el funcionamiento correcto. Finalmente, el test operacional del sistema, en el cual se realizan las últimas pruebas sobre un escenario de laboratorio, anotando una vez más las pruebas realizadas y los resultados obtenidos. De ser necesario se añadirán nuevos requisitos con el fin de mejorar el alcance del proyecto [4].

.

CAPÍTULO II

2. ESTADO DEL ARTE

.

En este capítulo se pretende dar una visión general del desarrollo tecnológico y estado actual de las redes sensores inalámbricos, así como también de sus principales características. Antes de hablar de redes de sensores inalámbricos es necesario sin embargo entender su conceptualización.

2.1 WSN (Wireless Sensor Network)

Una red de sensores inalámbricos consiste en un grupo de dispositivos multifuncionales (nodos) de bajo costo y consumo, distribuidos y espaciados autónomamente en una región de interés, como lo representa la figura 2.1. Estos nodos son capaces de obtener información de su entorno ya sean condiciones físicas o ambientales, procesarla localmente, y comunicarla a través de enlaces inalámbricos hasta un nodo central de coordinación [5].

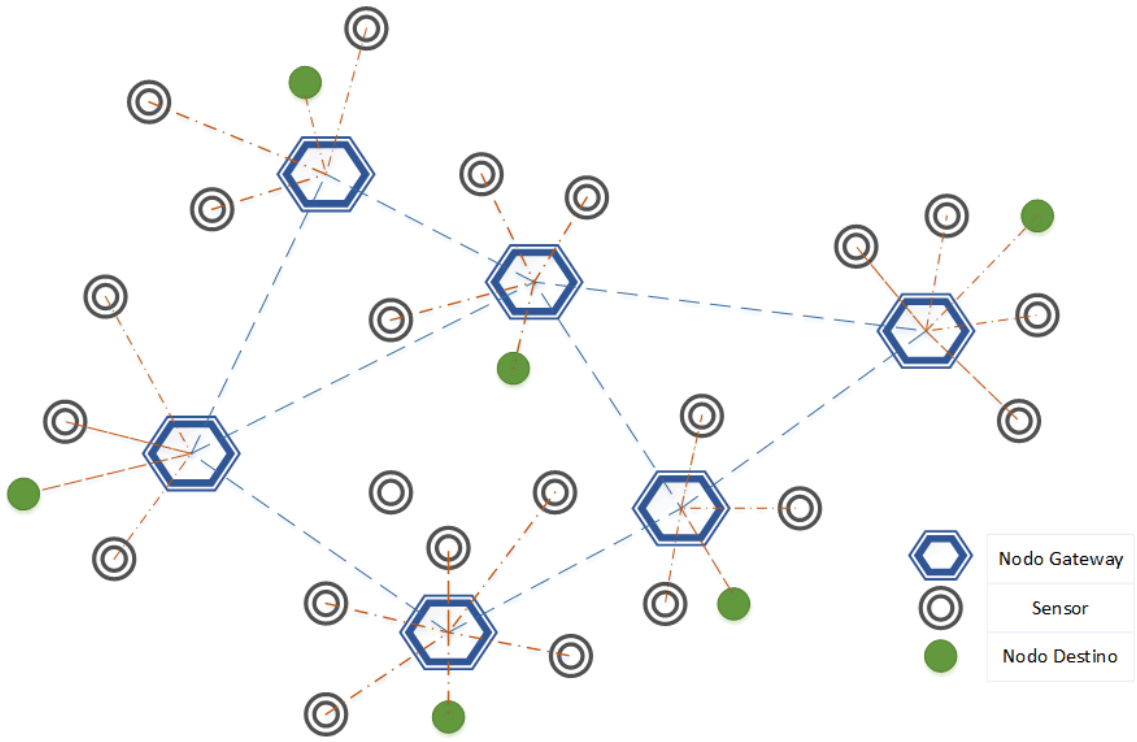


Figura. 2.1 Arquitectura de una WSN
Fuente: [6]
Elaboración: [6]

2.1.1 Características

- Son escalables, es decir pueden llegar a tener gran cantidad de nodos [5].
- Poseen capacidades en desarrollo de autodiagnóstico, autonomía energética, autoconfiguración, auto organización, auto restauración y reparación [5].
- No es necesaria una infraestructura de red para poder desplegar una WSN [7].

.

- Permite la integración de diversas ramas: biología, agricultura, minería, medicina, ingeniería civil, medioambiente, entre otras [7].
- Habitualmente construidas en modo Ad-Hoc para resolver un problema en concreto [5].
- Bajo costo de aplicación y facilidad de instalación en redes de gran tamaño [5].

2.1.2 Limitaciones.

- Alta probabilidad de fallo por ser redes desatendidas [5].
- Hardware sencillo que limita capacidades computacionales y de almacenamiento [7].
- Altas limitaciones en cuanto a rango de comunicaciones y capacidades energéticas de los nodos [7].
- Los nodos son propensos a fallas o daños físicos, por usarse generalmente en ambientes hostiles [7].
- Usualmente no es posible obtener un esquema de direccionamiento global en una red de sensores, debido al gran número que la conforman [7].

2.1.3 Componentes

Los elementos básicos que conforman una red inalámbrica de sensores se detallan a continuación:

2.1.3.1 Sistema de adquisición de datos.

Los sensores pueden ser de distinta naturaleza y tecnología. Toman del medio la información y la convierten en señales eléctricas. En el mercado existen placas con sensores de medida de diversos parámetros, como presión barométrica, GPS, luz, temperatura, humedad, sonido, velocidad del viento, entre otros [5]. Como ejemplos podemos citar: *MTS300/310*, una placa sensor de MEMSIC, capaz de detectar aceleración, luminosidad, sonido, magnetómetro y temperatura [8]; y *Agriculture 2.0 Sensor Board* de Libelium, que es una placa que consta de 15 tipos de sensores capaces de detectar variables de presión atmosférica, temperatura, humedad, luminosidad, radiación solar, velocidad, dirección y densidad del viento, además de poseer un dendrómetro [9].

2.1.3.2 Motas.

Las motas o nodos son unidades autónomas con funciones de procesamiento y de comunicación. Principalmente están conformadas por un microcontrolador (CPU), una

.

fuente de energía (batería), un elemento radio-transceptor (RF), memoria flash y elementos para la adquisición de datos mediante sensores [7]. Véase la figura 2.2.

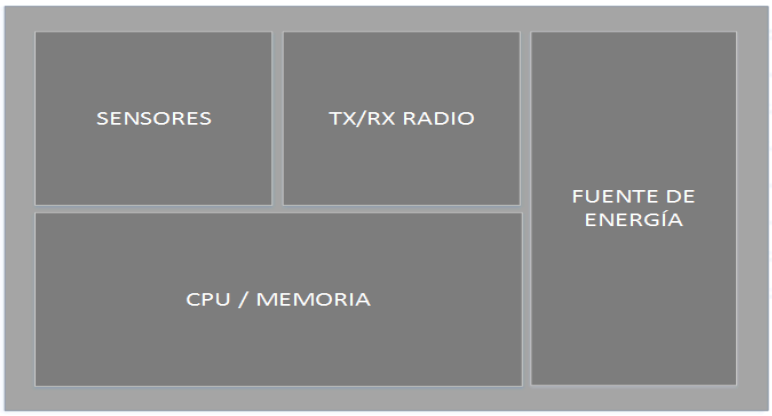


Figura. 2.2 Estructura básica de una mota
Fuente: [7]
Elaboración: Autores

Algunos modelos de plataforma para nodos incluyen sensores integrados en su diseño. Las variables medidas más comunes de encontrar en estos modelos son: temperatura, humedad, luminosidad y aceleración. Más adelante, en los apartados 2.6 y 2.7 se hace una recopilación de las características de plataformas libres y comerciales.

2.1.3.3 Nodo Gateway.

Es el dispositivo intermedio. Los más habituales son el *switch*, el router o el punto de acceso inalámbrico. Estos dispositivos se encargan de redirigir los datos a otros nodos de la red. Nunca, salvo alguna excepción, generan información o son el destino final de la información.

2.1.3.4 Gateway o pasarela de medio.

Gateway es un elemento que actúa como puente entre una WSN y otra red, es decir, permite la interconexión entre distintas redes [5]. El protocolo inalámbrico de la pasarela de medio depende de los requerimientos de la aplicación. Algunos de los estándares disponibles incluyen radios de 2.4GHz basados en los estándar IEEE 802.11.n (Wi-Fi) o radios propietarios [10].

Ejemplo: A manera de ejemplo, en la figura 2.3 se puede apreciar a *Dragino IoT Gateway*, un dispositivo incorporado con Linux que posee un puerto USB y capacidades Ethernet y 802.11 b/g/n Wi-Fi [11].

.

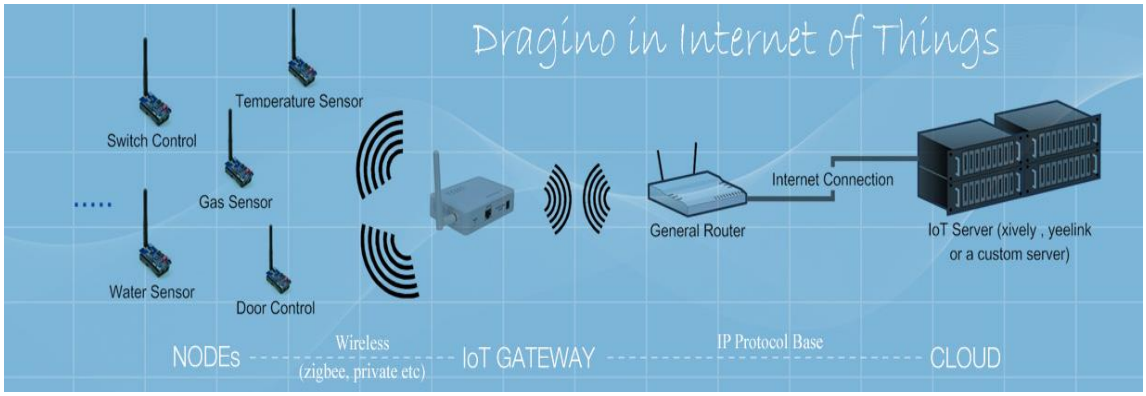


Figura. 2.3 Pasarela de medio Dragino en IoT
Fuente: [11]
Elaboración: [11]

2.1.3.5 Estación base.

Es el dispositivo de red de mayor capacidad para el almacenamiento, análisis y procesamiento de datos procedentes de los nodos o motas [12]. Un ejemplo de estación base es un servidor en un centro de control. Véase la figura 2.4.

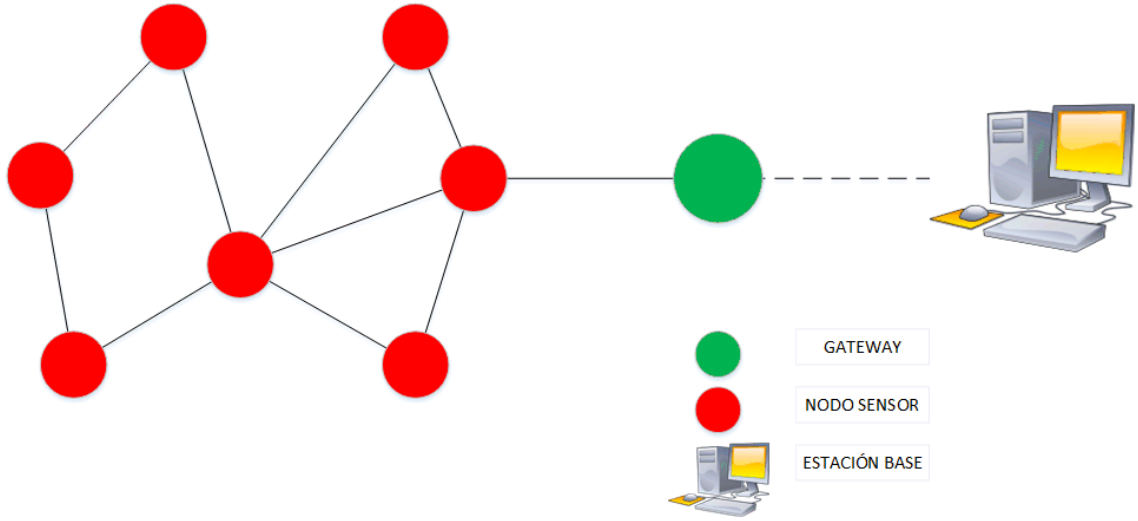


Figura. 2.4 Esquema de una red inalámbrica de sensores.
Fuente: Autores
Elaboración: Autores

2.2 Estándares inalámbricos en WSN

Uno de los componentes claves de una red inalámbrica de sensores es la tecnología inalámbrica que se usa, la cual define la cobertura, alcance y precisión del sistema. Actualmente existe un número de tecnologías inalámbricas que poseen múltiples ventajas que se adaptan a la aplicación concreta que se quiera desarrollar. Entre las más utilizadas en WSN tenemos: Wi-Fi, Bluettoth y Zigbee. A continuación, en la tabla 2.1, se detalla las características más relevantes de cada una de ellas.

.

Tabla 2.1. Cuadro comparativo de tecnologías inalámbricas.

Característica	Wi-Fi	Bluetooth	Zigbee
Estándar	IEEE 802.11	IEEE 802.15	IEEE 802.15.4
Velocidad	< 50 Mbps	1Mbps	< 250 kbps
Número de nodos	32	8	255/65535
Duración de batería	Horas	Días	Años
Consumo en TX	400mA	40mA	30mA
Consumo en Reposo	20mA	0,2mA	3µA
Precio	Alto	Medio	Bajo
Configuración	Compleja	Compleja	Simple
Frecuencia de trabajo	0.868 GHz 0.915 GHz 2.4 GHz	2.4 GHz	2.4 GHz
Rango de trabajo	30m-100m	1m-100m	1m-100m
Aplicaciones	Internet en edificios	Informática y móviles	Domótica y monitorización

Fuente [13][14]
Elaboración: Autores

2.3 Hardware para WSN

Existe una variedad de plataformas de Open Hardware y comerciales disponibles para diferentes aplicaciones de WSN. El propósito es hacer un análisis de las arquitecturas utilizadas, clasificando la estructura hardware de acuerdo a la naturaleza de los elementos de procesamiento como lo hace J. Portilla en su artículo sobre seguridad en redes inalámbricas, en las siguientes categorías [15]:

- Basados en un microcontrolador.
- Basados en FPGA.
- Basados en HW configurable.

2.3.1 Basado en un solo microcontrolador.

.

Tradicionalmente, las plataformas para WSN han sido basadas en microcontroladores de bajo costo. El enfoque está sustentado sobre la idea de que los nodos no pueden realizar tareas de alta complejidad y los sistemas deben pasar en modo *sleep* la mayor parte del tiempo para ahorrar energía [15]. Como ejemplo de este tipo de plataformas podemos citar: *Waspmote* de *Libelium* (figura 2.5) que trabaja con un ATmega1281 y memoria flash extendida.

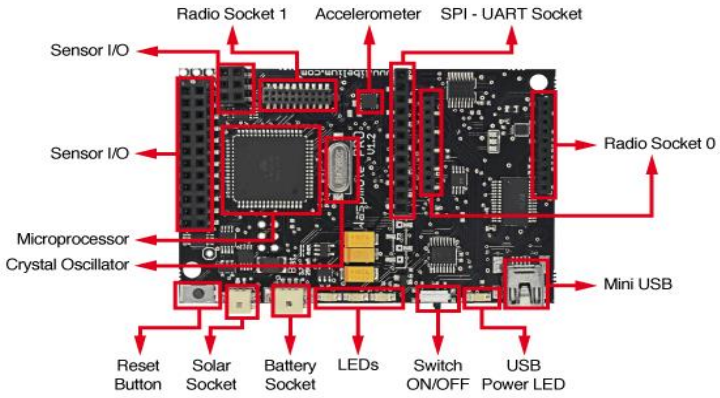


Figura. 2.5 Componentes principales de Waspmote.
Fuente: [16]
Elaboración: [16]

2.3.2 Basado en FPGA

La solución de este tipo de plataformas está enfocada en el uso de procesadores embebidos en sistemas programables como FPGA (*Field Programmable Gate Array)*, proponiendo entornos de bajo costo debido a la reutilización de hardware, además de un bajo consumo de energía. Las FPGAs alcanzan un equilibrio óptimo entre la potencia de procesamiento, los requerimientos de energía y flexibilidad [15]. Por ejemplo, la plataforma de desarrollo Mojo FPGA (figura 2.6).

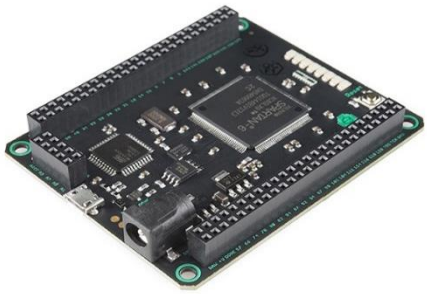


Figura. 2.6 Plataforma de desarrollo Mojo V3.
Fuente: [18]
Elaboración: [18]

.

2.3.3 Basado en HW configurable y un microprocesador.

Algunos investigadores trabajan en soluciones donde SW y HW coexisten a partir de la formulación conceptual del circuito, de modo que hay un elemento reconfigurable unido al procesador que puede ser utilizado como el diseñador prefiera. Este enfoque permite crear plataformas del todo flexibles, supliendo la necesidad de cambiar HW de acuerdo a la aplicación o cambios del entorno, es decir, un nodo flexible que puede estar configurado para ser utilizado en varias aplicaciones y podría ser más eficiente que los específicos al momento de ser desplegados en la red. Esto es posible, al incorporar una RFU (unidad funcional reconfigurable) en paralelo con la ALU (unidad lógica aritmética) a un procesador de arquitectura RISC (*Reduced Instruction Set Computer*) [15].

2.4 Software para WSN

En cuanto a software para nodos WSN, se tiene una variedad de sistemas operativos para el control programable con el que cuente la plataforma. Este estudio se enfoca en las herramientas de programación para microcontroladores más usadas por las grandes empresas desarrolladoras de hardware para redes inalámbricas de sensores. A continuación, se describen las más comunes.

2.4.1 TinyOS

Es un sistema operativo de código abierto que la Universidad de Berkeley desarrolló exclusivamente para responder a las características y necesidades de una WSN, tales como tamaño reducido de memoria, bajo consumo de energía y simultaneidad en operaciones de concurrencia masiva. En comparación con otros sistemas operativos, TinyOS se basa en un modelo de programación controlado por eventos, en vez de multiprocesos. Tanto TinyOS como sus programas, son escritos en NesC, una extensión del lenguaje de programación C [19].

2.4.2 ConTiKi

Es otra propuesta de código abierto creado para sistemas embebidos como microcontroladores o nodos WSN (específicamente dispositivos de 8 bits). Ha sido portado a varias plataformas de redes de sensores inalámbricas y está muy orientado a la interconexión de este tipo de dispositivos a Internet. Se programa en C y aunque incluye gestor de tareas y varias pilas de protocolos de comunicaciones (TCP/IP, por ejemplo) es un sistema ligero, tanto en el tamaño de los programas como en el uso de la memoria RAM. Esto se consigue gracias a que soporta programación orientada a eventos y basada en hilos, que carga y descarga dinámicamente los programas según se va ejecutando la aplicación [20].

.

2.4.3 Arduino.

Es un proyecto GPLv2 de HW y SW para sistemas embebidos. El Arduino original se programa en C/C++ usando AVR-GCC para la compilación. El IDE de Arduino es una aplicación multiplataforma escrita en Java, que consiste en un compilador de lenguaje de programación estándar y un cargador de arranque que se ejecuta en la placa. El software de Arduino es de código abierto y por ello se encuentra en la base de algunas plataformas para WSN, como es el caso de la empresa Libelium [21].

2.4.4 eCos:

Es un sistema operativo de código abierto, gratuito y de tiempo real desarrollado para sistemas empotrados y para aplicaciones que necesitan un procesador multisesión. Desarrollado en C, con capas y APIs compatibles con POSIX y ulTRON, es personalizable según la aplicación. Aunque es indicado especialmente para aquellos dispositivos que cuenten con muy poca memoria RAM. Además eCos tiene una estrecha relación con los sistemas UNIX, que permiten soportar sistemas Linux empotrados [22].

2.4.5 RIOT:

Fue creado como parte del proyecto FeuerWare, un sistema operativo de uso exclusivo para WSN. Con el fin de incrementar flexibilidad en el software e incluir nuevos protocolos IETF, RIOT (llamado anteriormente µkleos) fue bifurcado desde el repositorio original de FeuerWare. Tiene soporte para 6LoWPAN, RPL y TCP. Programación estándar en C o C ++ basada en prioridades, disponible para plataformas de 8, 16 y 32 bits [23].

2.5 Internet de las Cosas

El término IoT (*Internet of Things* por sus siglas en inglés), está definido por el centro de investigación SAP (Sistemas, Aplicaciones y Productos; en Procesamiento de Datos) como: “Un mundo en el que los objetos físicos están perfectamente integrados en la red de información y estos pueden convertirse en participantes activos en procesos de negocios. [24] ” Es decir, se define como la capacidad de un objeto o “cosa” para conectarse a la web e interactuar con cualquier otro dispositivo o persona. La empresa tecnológica Cisco estima que para el año 2020 se tenga 4 mil millones de conexiones a Internet, lo que indica el gran impacto a futuro que tendrá IoT en el mundo [25].

.

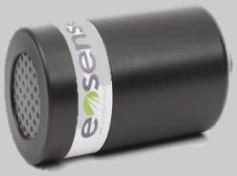
2.5.1 Sensores

Para que un nodo de una WSN sea capaz de recolectar información de su entorno es necesario que cuente con sensores para medir parámetros específicos. Un sensor es un dispositivo capaz de detectar magnitudes físicas o químicas (llamadas variables de instrumentación), y transformarlas en variables eléctricas. Las variables de instrumentación pueden ser de distintos tipos como: temperatura, intensidad luminosa, distancia, presión, humedad, movimiento, pH, etc. Una variable eléctrica puede ser la resistencia , tensión y corriente eléctrica [26]. En la tabla 2.2 se realiza una recopilación de los sensores más utilizados en aplicaciones de WSN:

Tabla 2.2 Ejemplos de tipos de sensores para la plataforma.

Sensor	Fabricante	Interfaz	Rango de medición
Sensor de velocidad y dirección del viento (Anemómetro) 	Davis Instruments	Digital	1-320 km/h
Sensor de temperatura y humedad (Termo-higrómetro) 	Davis Instruments	Digital	-40ºC a 65ºC 1 -100%
Sensor de precipitación (Pluviómetro) 	Davis Instruments	Digital	0.0-999.8 mm

.

Sensor de radiación solar y UV  	Davis Instruments	Analógico	0-1800 W/Mm^2 0-16 índice UV
Sensor de humedad y temperatura SHT11 para suelo 	SENSIRION	Digital	-40ºC a 124ºC
Sensor de presión barométrica MS5611 	Embedded Adventures	I2C	0-10cm de altura
Sensor de CO2 eosGP 	Eosense	Analógico Digital (RS232)	0-5000 ppm 0-20000 ppm
Sensor de luminosidad HFA/A 	Fuehler Systeme	Analógico	0-50klux

.

Sonda de conductividad iónica CON-BTA 	Vernier	Analógico	0-100 mg/L 0-100 mg/L 0-10000 mg/L
Sonda de oxígeno ODO-BTA 	Vernier	Analógico	0-20 mg/L
Sonda PH-BTA 	Vernier	Analógico	0-14 PH
Sonda de temperatura TPL-BTA: 	Vernier	Analógico	-50 a 150ºC
Sensor de distancia ultrasónico TA0050KA 	HURICANE	Digital	0.3-8m

Fuente: [27] [28] [29] [30] [31][32] [33] [34] [35] [36] [37] [38] [39].
Elaboración: Autores

2.5.2 Bases de datos

La cantidad de información que generan las WSNs se puede gestionar usando servicios en la nube, marcos de hardware y software flexibles capaces de entregar la informática como un servicio [40]. Se puede definir un SGBD (Sistema Gestor de Bases de Datos) como una colección de datos relacionados entre sí, estructurados y organizados, y un

.

conjunto de programas que acceden y gestionan esos datos. Los SGBD mayormente conocidos son: Oracle, DB2, PostgreSQL, MySQL [41].

La información recolectada por los nodos se debe almacenar en una base de datos y de ahí la aplicación web sustraerá los datos para presentarlos a quien tenga el acceso correspondiente. En la figura 2.7 se representa este proceso.

1) Nodos de borde
2) Visualización y análisis de datos por el usuario
3) Almacenamiento en la base de datos

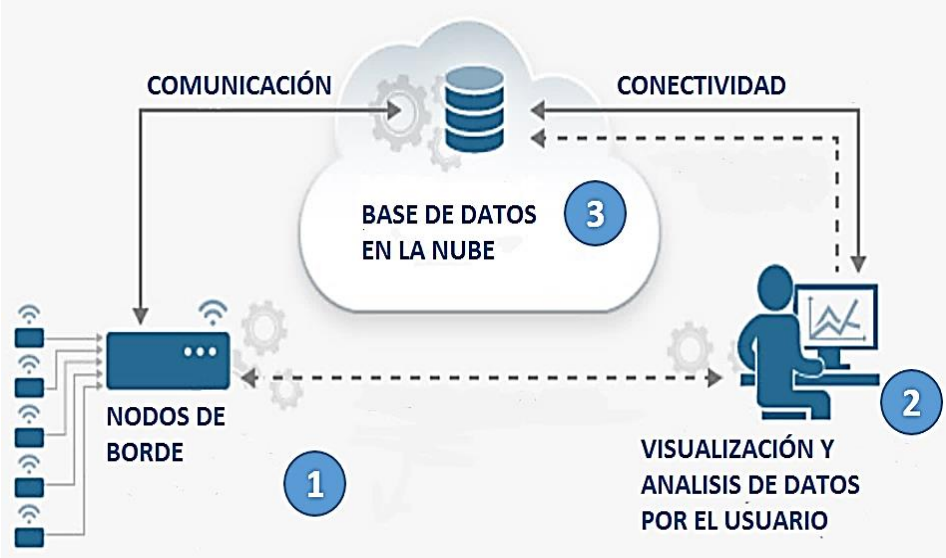


Figura. 2.7 Descripción del internet de las cosas en WSN.
Fuente: [42]
Elaboración: [42]

2.5.3 Aplicaciones web para IoT

Según Link-labs, una plataforma web integrada al IoT es el software que conecta al hardware, puntos de acceso y redes de datos con las aplicaciones que permiten al usuario final automatizar su entorno de trabajo. Esto demanda de un esfuerzo para los desarrolladores porque deben crear aplicaciones robustas y ligeras para que puedan ser accedidas por medio de un servidor Web [41][43].

Actualmente, existen algunas plataformas web para IoT, que acorde a la aplicación se adaptan a las finalidades que busca el usuario. A continuación, se citan dos de las más conocidas.

.

2.5.3.1 Ubidots

Ubidots ofrece una plataforma para los desarrolladores que permite capturar fácilmente los datos del sensor y convertirla en información útil. Los datos se pueden enviar a la nube desde cualquier dispositivo conectado para Internet. Permite configurar acciones y alertas en base a datos en tiempo real y mostrar el valor de los mismo a través de herramientas visuales, como se observa en la figura 2.8. [44].

Figura. 2.8 Interfaz Ubidots.
Fuente: [45]
Elaboración: [45]

2.5.3.2 Thingspeak

Es una plataforma que permite recopilar y almacenar datos en la nube y desarrollar aplicaciones de IoT. La plataforma ThingSpeak ofrece herramientas que permiten analizar y visualizar los datos en MATLAB y posteriormente actuar en base a ellos [46]. En la figura 2.9 se puede apreciar el monitoreo de un canal.

Figura. 2.9 Interfaz de Thingspeak.
Fuente: [47]
Elaboración: Autores

.

2.6 Plataformas de hardware abierto

En la actualidad, dentro del mercado tecnológico, gran parte de las empresas despliegan plataformas enfocadas al IoT y WSN, sobre todo en lo que se refiere al campo de desarrollo de plataformas de recursos abiertos. Afortunadamente, los diseños son cada vez más pequeños y con grandes capacidades de procesamiento. A continuación, en la tabla 2.3, se indica las diferentes opciones de plataformas basadas en de microcontroladores ATmega.

Tabla 2.3. Plataformas de hardware abierto.

Fabricante	Plataforma	MCU	SO	ROM	SRAM	Interfaz
Libelium	Waspmote	ATmega 1281	Programable	128 KB	4 KB	UART, I2C, SPI, WB, PWM, SD
Chip45	iDwaRF - BOX V.1.2	Atmega128	Cypress' WirelessUSB N:1 DVK (CY3635)	128 KB	SRAM externa 32 KB	UART, Lantronix export ethernet, SD
Arduino	Arduino UNO	Atmega328p	Programable	32 KB	2 KB	UART, I2C, SPI
Seed Studio	Seeduino Stalker V3	Atmega328p	Programable	32 KB	2 KB	UAR, I2C, SPI

Fuente: [48][49][21][50]
Elaboración: Autores

2.6.1 Ventajas

- Implementación de bajo costo.
- No tienen limitaciones en cuanto a compatibilidad con sensores y módulos de comunicación.
- Ofrece la posibilidad de proporcionar acceso a una gran comunidad de desarrolladores de softwarey facilita el uso del hardware.

2.6.2 Desventajas

- Muchas de las plataformas no son aptas para entonos industriales.
- Baja capacidad de almacenamiento.
- En el caso de reproducir hardware el diseño del mismo implica conocimientos en infraestructura, simulación e implementación.

.

2.7 Plataformas comerciales

Al igual que las plataformas de hardware abierto, en el mercado existen algunos modelos de plataformas comerciales (ver tabla 2.4).

Tabla 2.4. Plataformas comerciales

Fabricante	Plataforma	MCU	SO	ROM	SRAM	Interfaz
Crossbrow	Mica - MPR300CB	ATmega 128L	TinyOS	128 KB	4 KB	UART
Crossbrow	Mica2 - MPR400CB	ATmega 128L	TinyOS	128 KB	4 KB	UART, I2C, SPI
Crossbrow	Micaz - MPR2400	ATmega 128L	TinyOS	128 KB	4 KB	UART, I2C, SPI
Crossbrow	Mica2dot - MPR500CA	ATmega 128L	TinyOS	128 KB	4 KB	UART, DIO
Moteiv	Telos - MSP430	MPS430 F1611	TinyOS	48 KB	10 KB	UART, I2C
Moteiv	TelosB TPR2420	TI MSP430	TinyOS	48 KB	10 KB	UART, I2C, SPI
Moteiv	TmoteSky - XM1000	TI MPS430 F2618	TinyOS, ContikiOS	116 KB	8 KB	UART, I2C, SPI, USB, USCI
MEMSIC	LOTUS - LPR2400	Cortex-M3 32 bits		512 KB	64 KB	UART, I2C, SPI, GPIO, ADC

Fuente: [51][52][17][53]
Elaboración: Autores

2.7.1 Ventajas

- Provee de asistencia técnica tanto en hardware como software.
- Desarrolla módulos específicos (para sensores o de comunicación)

2.7.2 Desventajas

- Pueden llegar a ser costos.
- No compatibles con otras plataformas comerciales.
- Algunos constan de sistemas operativos que se limitan a funciones específicas.

.

CAPITULO III

3. DISEÑO Y COMPONENTES DE LA PLATAFORMA

.

3.1 Esquema general del diseño

El esquema propuesto para el diseño de la plataforma se desarrolla principalmente sobre un bloque computacional del cual se distinguen tres bloques secundarios: bloque de sensores, de comunicación y de control de energía.

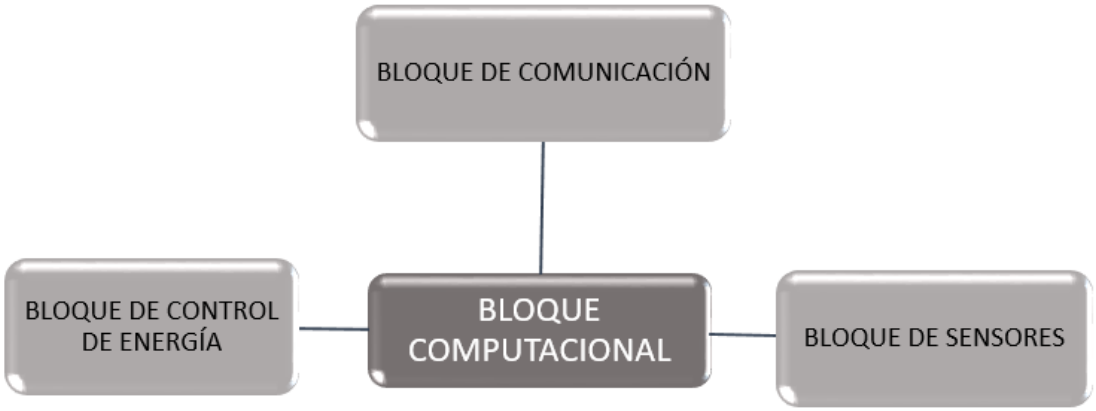


Figura. 3.1 Diagrama general de diseño de la plataforma.
Fuente: Autores
Elaboración: Autores

En los apartados siguientes se detallan los componentes que conforman cada bloque, y los requerimientos que se cubre con el hardware seleccionado.

3.2 Bloque computacional: SODAQ Autonomo

El principal reto en el diseño de hardware es la capacidad computacional, lo que plantea el desafío de desarrollar plataformas que permitan incorporar un sistema operacional que minimice retrasos en el procesamiento de datos y optimice recursos en un nodo.

Para la elección del elemento computacional, se tomó en cuenta tres características principales:

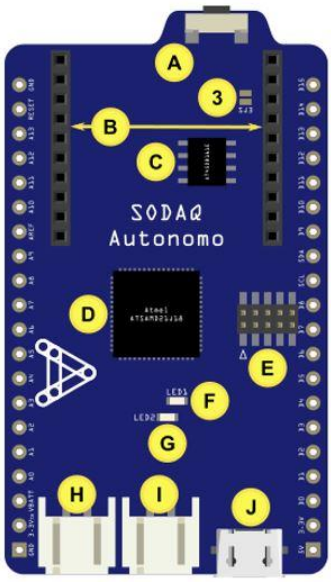
- Capacidad para de almacenamiento.
- Disponibilidad periférica.
- Eficiencia energética.

.

	Autonomo	Arduino/ Genuino Zero	Neutrino	FemtoUSB
Microcontroller	ATSAMD21J18	ATSAMD21G18	ATSAMD21G18	ATSAMD21E18
Arduino-compatible	✓	✓	✓	✗
Digital I/O	18	14	18	11
Analog I/O	14	6	6	6
Battery/Solar Support	✓	✗	✗	✗
Flash Storage	✓	✗	✗	✗
Bee Socket	✓	✗	✗	✗
SD Card	✓	✗	✗	✗

Figura. 3.2 Comparativa de plataformas computacionales.
Fuente: [54]
Elaboración: [54]

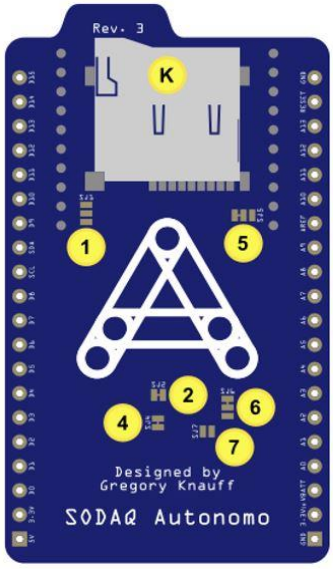
Como podemos apreciar en la figura 3.2, existen placas que trabajan con microcontroladores *Atmel* de la familia SAM D que cumplen con algunos de los requerimientos, pero la tarjeta más destacada es SODAQ Autonomo; un dispositivo idóneo para aplicaciones de WSN por su bajo consumo y autonomía energética. En las figuras 3.3 y 3.4 se describen sus componentes principales.



Ítem	Descripción
A	Botón para resetear
B	Enchufe para Bee
C	AT45DB161E 2Mb Flash Memory
D	Atmel SAM D21J18
E	Enchufe para el cable depurador serial
F	LED1 de encendido
G	LED2 indicador de carga Lipo
H	Enchufe para batería Lipo
I	Enchufe para panel solar
J	Enchufe Micro-USB

Figura. 3.3 Vista frontal del SODAQ Autonomo.
Fuente: [55]
Elaboración: Autores

.



Ítem	Descripción
K	Enchufe para la tarjeta Micro-SD
1	SJ1 conecta al modo RI o AS
2	SJ2 desconecta el LED1
3	SJ3 conecta el DIO3 al reset
4	SJ4 desconecta el LED2
5	SJ5 cambia de BEE_VCC a siempre ON
6	SJ6 cambia de VCC_SW a siempre ON
7	SJ7 conecta el pinA1(estado de carga)

Figura. 3.4 Vista posterior del SODAQ Autonomo.
Fuente: [47]
Elaboración: Autores

3.2.1 Características

La información sobre el entorno de programación y particularidades en hardware de la tarjeta SODAQ Autonomo se encuentran disponibles en su página web http://support.sodaq.com/sodaq-one/autonomo/getting-started-autonomo/.

A continuación, en la tabla 3.1 se detallan las características más importantes.

Tabla 3.1. Especificaciones de la tarjeta SODAQ Autonomo

Microcontrolador	ATSAMD21J18, 32-Bit ARM Cortex M0+
Compatibilidad	Arduino M0 compatible
Tensión de funcionamiento	3.3V
Dimensiones	58.5 x 33.5 mm
Puertos I/O Digitales	16, con 2 UART, 2 SPI y 4 TWI (I2C)
Puertos de entrada análoga	6, canales ADC de 12 bits
Puertos de salida analógica	DAC de 10 bits
Corriente en los puertos	7mA (DC)
Memoria flash	256KB
SRAM	32KB
EEPROM	Hasta 16KB por emulación
Velocidad de reloj	48MHz
Almacenamiento	Módulo Data flash de 2MB y ranura para Micro SD
Voltaje de alimentación	5V (USB) y 3,7V (batería LiPo)
Comunicaciones	Ranura para módulos Bee

Fuente: [47]
Elaboración: Autores

SODAQ Autonomo puede ser alimentado mediante el puerto Micro USB o con una fuente de alimentación externa, como una batería LiPo. Asimismo, cuenta con dos indicadores led: un amarillo que informa el estado de carga y un led verde que puede ser usado como indicador de estado en el desarrollo del algoritmo.

.

3.2.2 Puertos y esquema.

La figura 3.5 muestra la correspondencia de los *pins* entre la MCU, SODAQ Autonomo y la plataforma diseñada. Se indica en color rojo la función que tiene esta tarjeta y en color verde se especifica el empleo que se da en el prototipo diseñado.

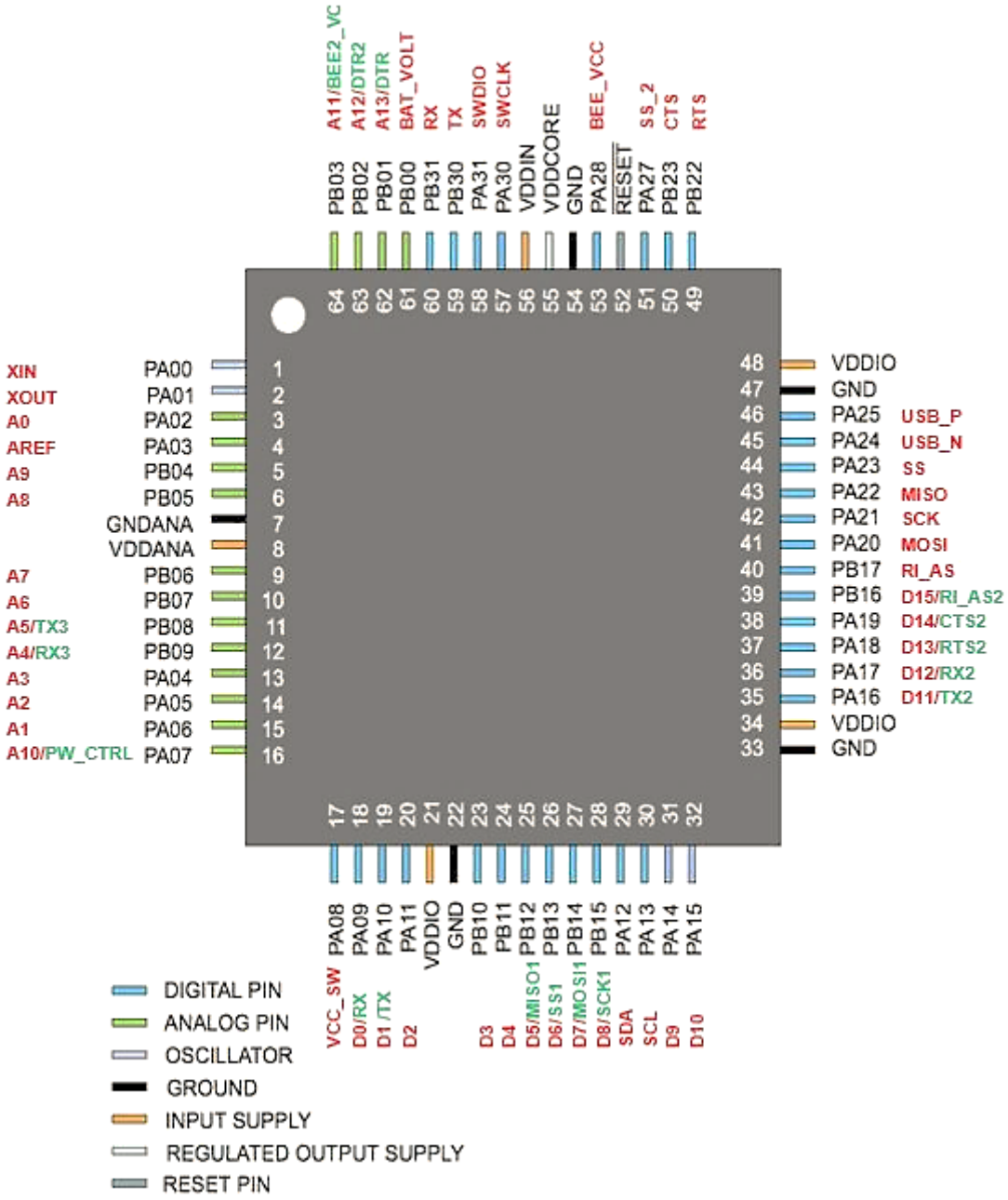


Figura. 3.5 *Pins* del ATSAM D21J18 QFN64 y su funcionalidad.
Fuente: [56]
Elaboración: Autores

La característica de *pin mapping* o *multiplexed signals* como se menciona en la hoja técnica del microcontrolador ATSAM D21, permite asignar a los puertos una de las funciones periféricas (A, B, C, D, E, F, G, H) puntualizadas en el Anexo 1.

.

Descripción de las funciones periféricas:

- *EIC*: o controlador de interrupción externa, permite que los puertos externos se configuren como líneas de interrupción. Se cuenta con 16 interrupciones. Cada puerto externo puedo ser configurado como asíncrono con el fin de despertar a un dispositivo cuando está en modo *sleep* [57].
- *ADC:* contiene un conversor análogo digital de 12 bits que permite hasta un máximo de 20 canales [57]
- *SERCOM:* o módulo de comunicación serial. La MCU cuenta con seis módulos que pueden ser configurados para actuar como un USART, UART, I2C, SPI de hasta 3.4MHz, SMBus, PMBus, y LIN slave [57].
- *SERCOM_ALT:* es un módulo de comunicación serial alternativo, que se usa para que un puerto pueda funcionar con dos interfaces de comunicación [57].
- *TC:* consiste en un contador, un pre-escalador, canales de comparación/captura y un control lógico. El contador puede configurarse para contar eventos, o para contar los pulsos de reloj. El contador, junto con los canales de comparación/ captura, puede configurarse para marcar el tiempo de eventos de entrada, permitiendo la captura de frecuencia y ancho de pulso. También puede realizar la generación de una forma de onda, como la generación de frecuencia y la modulación de ancho de pulso (PWM) [57].
- *TCC:* es un temporizador/contador de 24 bits para control que permite configurar hasta cuatro canales de comparación con salida complementaria opcional, también se puede generar un patrón de modulación de ancho de pulso sincronizado (PWM) a través de los puertos [57].
- *GCLK: o* controlador de reloj genérico que controla el sistema de distribución de reloj, formado por generadores de reloj genérico que puede utilizar cualquiera de las fuentes de reloj del sistema como su reloj de origen. Y el Generic Clock Generator 0, también llamado GCLK_MAIN, es el reloj que alimenta el Power Manager utilizado para generar relojes síncronos [57].
- *SYSCTRL:* el controlador de sistema proporciona una interfaz de usuario a las fuentes de reloj, detectores de apagado, regulador de voltaje y referencia de voltaje del dispositivo. A través de la interfaz de registros, es posible habilitar, deshabilitar, calibrar y monitorear sub-periféricos de SYSCTRL [57].
- *USB:* bus universal serial, cumple con la especificación USB 2.1 que soporta dos modos: *device* y *embedded host*. El modo *device* soporta hasta ocho direcciones de dispositivos finales. Además permite velocidades de 1.5 hasta 12 Mbps [57].

.

3.2.3 Entorno de trabajo.

Para la programación en el SODAQ Autonomo se usa el compilador Arduino IDE. Su entorno de trabajo está escrito en java y su lenguaje está basado en C/C++. Arduino IDE, al ser un software de código abierto, cuenta con soporte de diferentes desarrolladores de todo el mundo. *Github* es uno de los principales repositorios libres que cuenta con diferentes ejemplos como librerías para la programación de la mayoría de tarjetas y módulos [21].

La tarjeta SODAQ Autonomo se encuentra disponible en el gestor de tarjetas de Arduino. Para proceder a instalar el paquete SODAQ SAM Boards, se debe agregar la siguiente dirección http://downloads.SODAQ.net/package_SODAQ_index.json en el en el Gestor de URLs Adicionales de Tarjetas. Véase la figura 3.6.

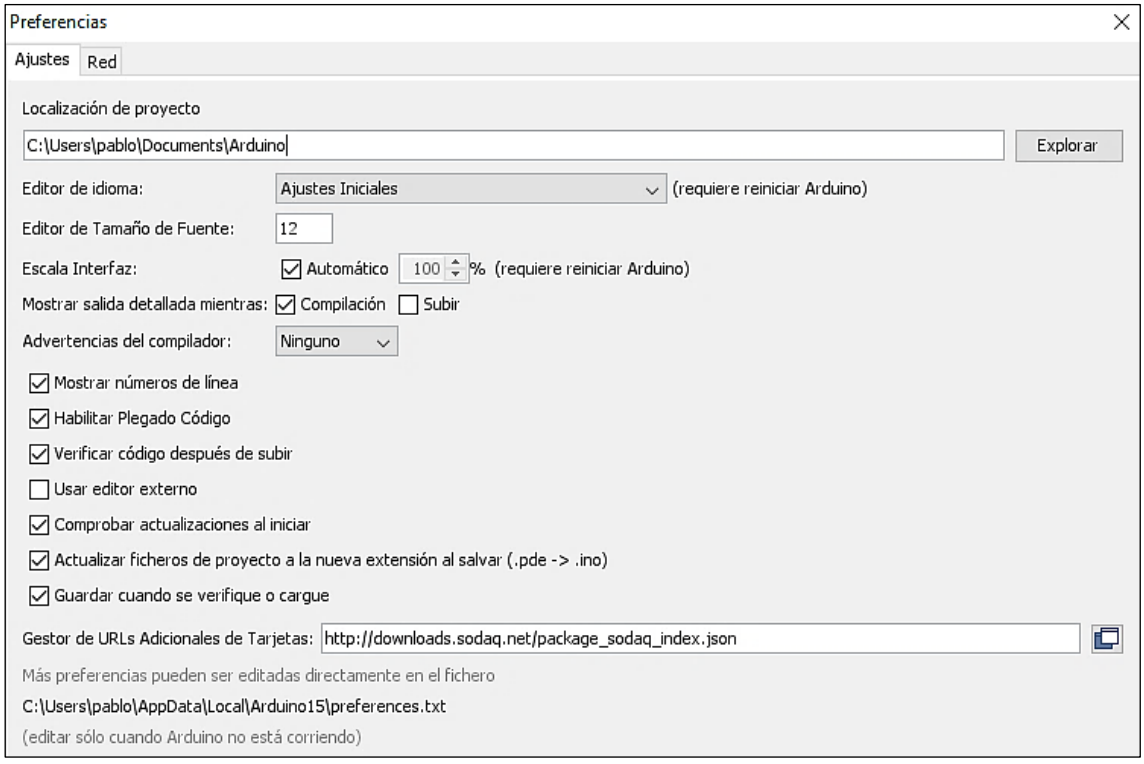


Figura. 3.6 Instalación de los archivos de la tarjeta SODAQ Autonomo.
Fuente: Autores
Elaboración: Autores

Luego, como se puede observar en la figura 3.7, en la pestaña gestor de tarjetas se procede a la descarga e instalación del paquete. Finalmente se escoge la placa a programar (SODAQ Autonomo) con su respectivo programador Atmel-SAM-ICE.

.

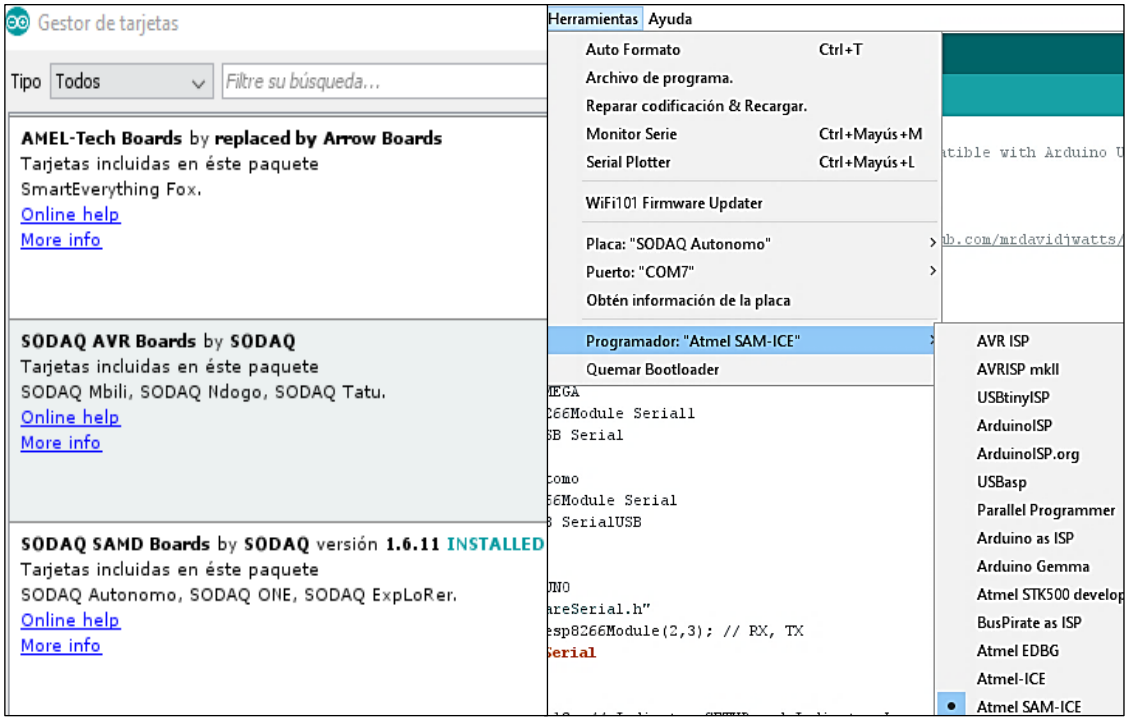


Figura. 3.7 Selección del programador.
Fuente: Autores
Elaboración: Autores

3.3 Bloque de control de energía

Los componentes más relevantes del bloque de control de energía se detallan en las siguientes secciones.

3.3.1 LDO XC6226B.

Es un regulador de voltaje LDO de alta precisión y bajo ruido con función *Green Operation*. El rendimiento en caída de tensión ultra bajo con salida de corriente de hasta 1 Amperio, prolonga la vida útil de la batería, al igual que la función *Green Operation* que puede conmutar entre modos de alta velocidad y ahorro de energía automáticamente. En la figura 3.8, se representa la función del puerto CE, el cual permite que el voltaje de salida se apague y el chip pase a modo espera [58].

.

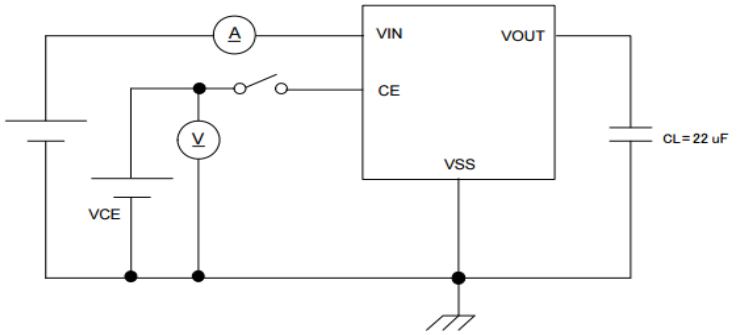


Figura. 3.8 Circuito de aplicación típica del regulador de voltaje.
Fuente: [58]
Elaboración: [58]

3.3.2 TCA9554.

El circuito integrado TCA9554 es un dispositivo I2C de 16 *pins*, que proporciona una expansión de puertos I/O de 8 bits. Está diseñado para operar con tensiones desde 1.65V a 5.5V. El objetivo de usar este dispositivo dentro del bloque de control, fue para obtener puertos auxiliares de salida digital, y no utilizar los de la tarjeta SODAQ Autonomo. En la figura 3.9 se observar los diferentes usos que se puede asignar a los puertos. Y los pins A0, A1 y A2 que son usados para programar y variar la dirección física del esclavo I2C.

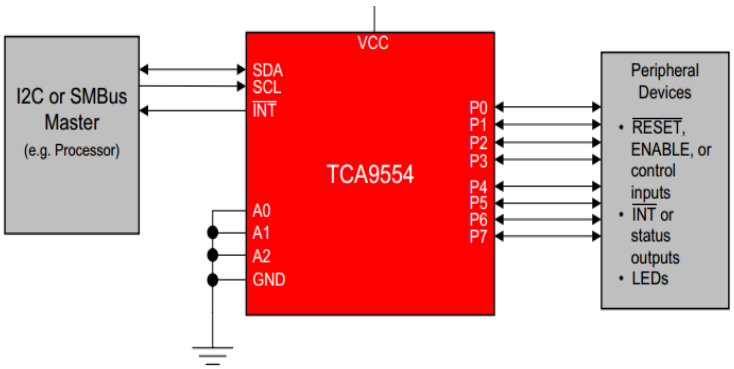


Figura. 3.9 Diagrama de bloques del TCA9554.
Fuente: [59]
Elaboración: [59]

3.3.3 Funcionamiento y descripción del circuito de control de energía.

El bloque de control de energía descrito en el diagrama de la figura 3.10, indica que los puertos P0 a P3 del TCA9554 configurados como salidas, están conectados al puerto CE de cada regulador de voltaje XC6220B. Este puerto se encarga del control ON/OFF en los suministros de energía para las diferentes interfaces de los sensores (digitales, analógicos y de comunicación I2C), sensores internos y la ranura XBee2. Esta última, es controlada desde la tarjeta SODAQ Autonomo mediante el *pin* A11. Los puertos restantes, P4 a P7 son usados como salidas digitales auxiliares.

.

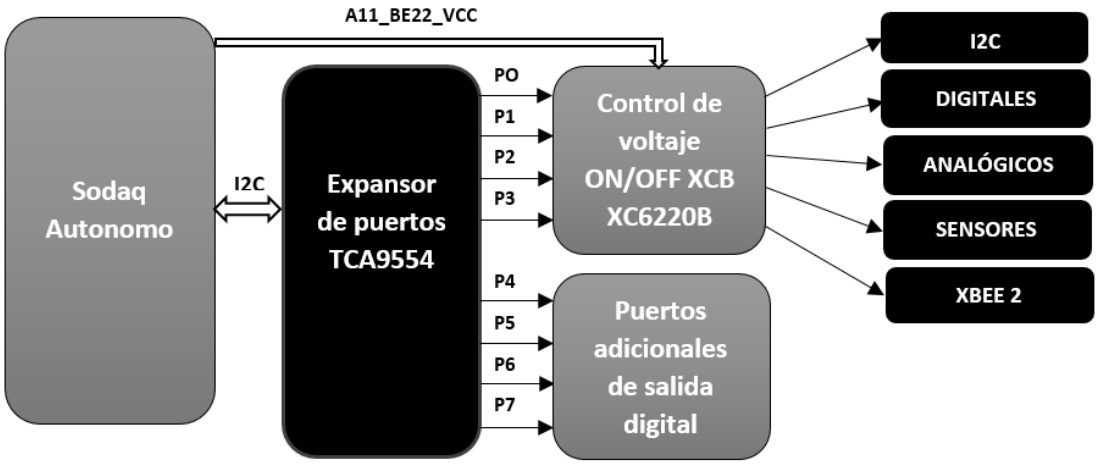


Figura. 3.10 Diagrama de bloques de control de energía y puertos adicionales.
Fuente: Autores
Elaboración: Autores

En la figura 3.11 se indica el circuito de control para sensores que usan interfaz I2C. Una de las observaciones importantes es la utilización de un transistor MOSFET tipo P, que actúa como interruptor. Cuando la tarjeta SODAQ Autonomo está alimentada directamente por USB (5V) y al mismo tiempo por la batería, el transistor procede a impedir el paso de voltaje (V_BAT).

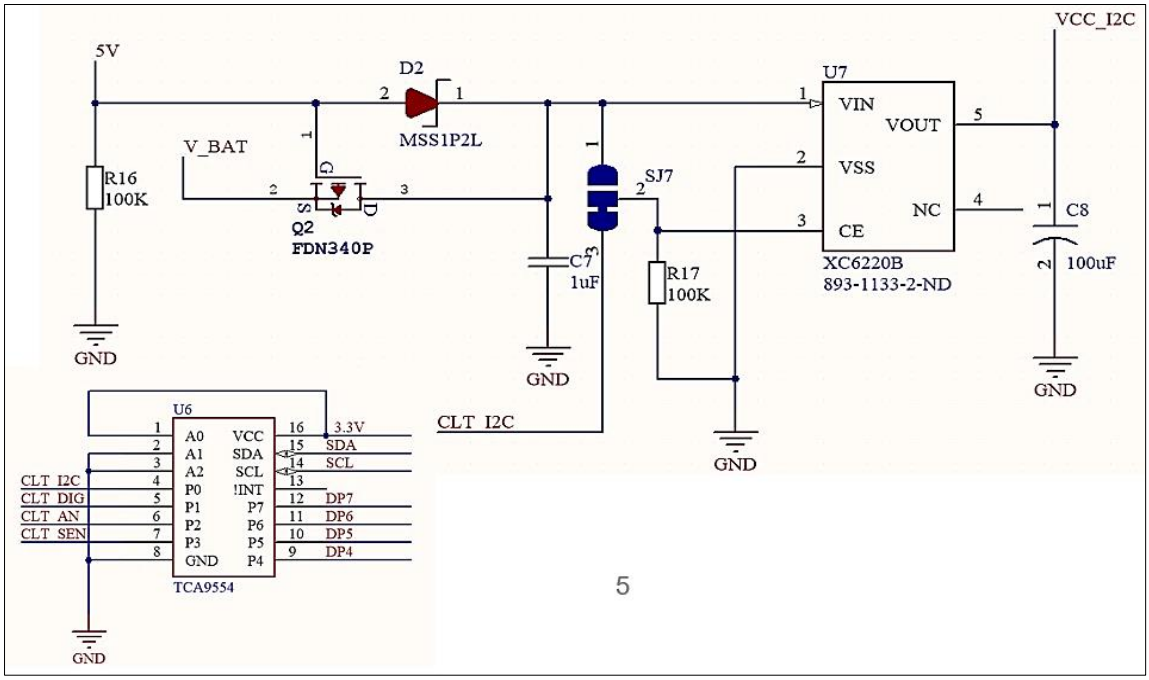


Figura. 3.11 Esquema del circuito de control de energía para conectores I2C.
Fuente: Autores
Elaboración: Autores

.

3.4 Bloque de sensores

En este bloque se especifica los sensores incorporados en la placa y conectores para la integración de sensores externos.

3.4.1 Sensores internos.

La plataforma incorpora 2 tipos de sensores. Un acelerómetro (ADXL345) de *Analog Devices* [60] que informa a la mota sobre las variaciones de aceleración que experimenta en cada uno de los 3 ejes (x, y, z); esto con la finalidad de manipular datos de posicionamiento de la mota por seguridad a posibles caídas o cambios de dirección de la misma. También adapta un sensor de humedad y temperatura (HIH9130) fabricado por *Honeywell Humidlcon* [61], un dispositivo de alta precisión(±1.7% RH) y bajo consumo pensado para hacer el trabajo de dos sensores en uno solo.

3.4.1.1 Temperatura y humedad HIH9130

HIH9130 es un sensor que puede operar desde los 2.3V; que es una tensión baja que permite el ahorro de energía y prolonga la vida útil de la batería. En modo *sleep* consume solo 1µA y en funcionamiento hasta 650µA. Tiene una resolución de 14 bits tanto para el sensor de humedad como para el de temperatura, permitiendo al usuario detectar cambios muy pequeños. Posee un rango de medición de temperatura desde -40°C a 125°C y en humedad relativa desde 10% a 90%.

La configuración típica del sensor para comunicación I2C se puede observar en la figura 3.12.

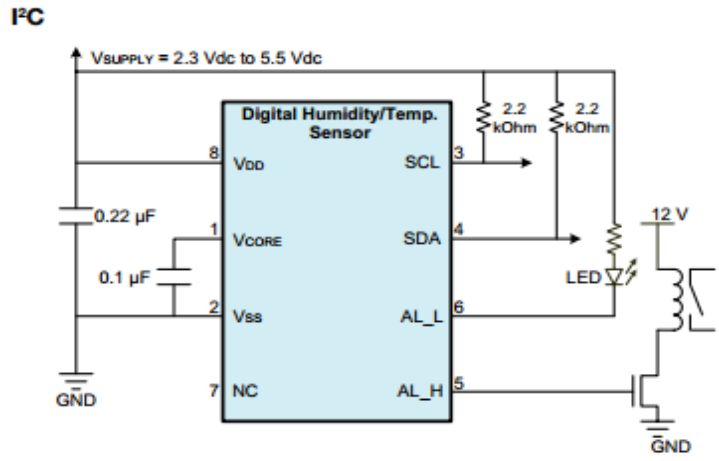


Figura. 3.12 Configuración típica del HIH9130 para comunicación I2C.
Fuente: [62]
Elaboración: Autores

.

3.4.1.2 Acelerómetro ADXL345

El ADXL345 es dispositivo pequeño de montaje superficial, bajo consumo y alta resolución (13 bits) en un máximo de ±16g. El dato de salida digital tiene un formato de 16 bits y es accesible a través de interfaz SPI (de 3 o 4 hilos) o I2C digital. Las variaciones de movimiento medidas pueden ser utilizadas en funciones específicas como detectar cambios de dirección, golpes leves, caída libre y la posición de la mota.

Además, cuenta con tecnología para un sistema de gestión de memoria integrado con FIFO de 32 niveles, es decir, se puede acumular y almacenar hasta 32 muestras de los tres ejes (x, y, z). Esto minimiza en carga al procesador central. En la figura 3.13, se puede observar el diagrama de bloques funcional y su configuración típica para comunicación I2C.

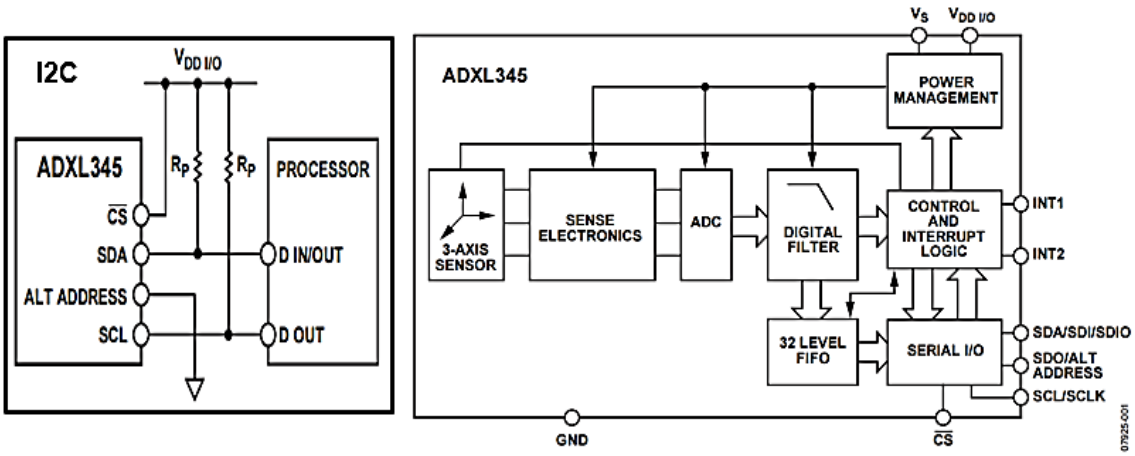


Figura. 3.13 Configuración I2C y diagrama de bloques funcional del ADXL345.
Fuente: [63]
Elaboración: Autores

3.4.2 Descripción del circuito de sensores internos

A continuación, en la figura 3.14, se muestra el diseño esquemático para ambos sensores utilizando las configuraciones típicas para la conexión mediante interfaz I2C. El voltaje de alimentación de ambos circuitos es proporcionado por VCC_SEN y gestionado por el bloque de control de energía descrito en el apartado 3.3.

.

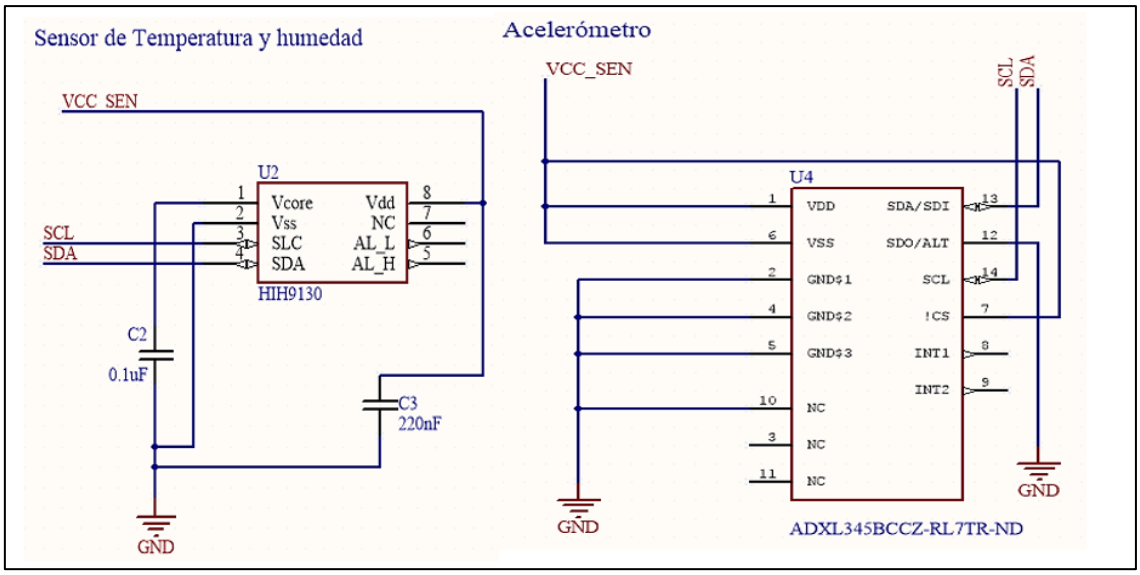


Figura. 3.14 Esquemático del sensor de humedad, temperatura y acelerómetro.
Fuente: Autores
Elaboración: Autores

3.4.3 Sensores externos.

Para la adaptación de sensores a la plataforma se propone dos tipos de conectores.

3.4.3.1 Conectores 15EDGK

El prototipo cuenta con conectores tipo 15EDGK compuestos por dos partes; una base que va soldada a la placa de circuitos y un componente que va enchufado a la base, indicados en la figura 3.15. La integración de este tipo de conectores facilita el acople de los sensores, brindando conexiones estables y protegidas ante posibles oxidaciones.

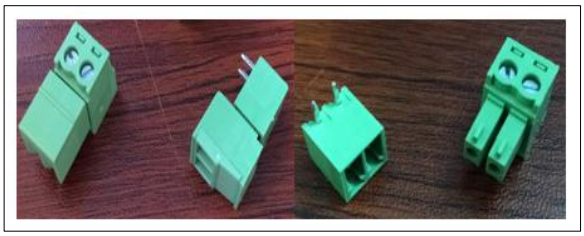


Figura. 3.15 Conectores 15EDGK.
Fuente: Autores
Elaboración: Autores

3.4.3.2 Conectores JST

Los conectores JST tipo hembra de cuatro *pins* son ideales para conectar sensores digitales, analógicos, y de interfaz I2C, así también como actuadores o algún tipo de elemento de comunicación que requiera de dos puertos para alimentación y otros dos para comunicación.

.

3.5 Bloque de comunicación

La mayoría de las plataformas hardware para WSN se limitan a usar solo una tecnología es sus diseños. Esto se ha convertido en una problemática cuando el usuario requiere aplicarlo dentro del campo del IoT. Por ello, debido a que el objetivo principal es diseñar una plataforma genérica que se adapte en ambas aplicaciones, se vio la necesidad de diseñar una plataforma que integre dos módulos de tecnologías inalámbrica en formato XBee, una de las soluciones integradas más conocidas en aplicaciones de redes inalámbricas de sensores [64].

3.5.1 XBee

Existen diferentes tipos de módulos XBee, teniendo una amplia gama de dispositivos inalámbricos para trabajar, con la ventaja que los módulos de comunicación tienen la misma distribución de puertos, como se indica en la figura 3.16.

Figura. 3.16 Distribución de puertos en XBee.
Fuente: [65]
Elaboración: [65]

Asimismo, en el mercado de productos para comunicaciones inalámbricas se encuentran disponibles módulos desarrollados en formato XBee que incluyen tecnologías como: Wi-Fi, GPRS / 2G, 3G, Lora, SigFox y Bluetooth (figura 3.17).

Figura. 3.17 Módulos de comunicación en formato XBee.
Fuente: [54]
Elaboración: [54]

.

3.5.1.1 GPRSBee.

El núcleo de esta placa es el módulo SIM800H. Este módulo, al igual que la mayoría de los otros módulos GPRS/GSM, tiene una tensión de funcionamiento de 3.5V a 4.5V y puede consumir hasta 2A de corriente durante los tiempos de emisión. Esto hace que la tensión de 3.3V que se tiene en la ranura Bee pueda ser inadecuada, pero esto se resuelve mediante la alimentación de la GPRSBee a una batería Lipo de 3.7V [50].

Figura. 3.18 Módulo GPRSBee.
Fuente: [50]
Elaboración: [50]

3.5.1.2 Wi-Fi Bee.

Los módulos XBee Wi-Fi proporcionan una conexión simple en serie IEEE 802.11 para crear nuevas opciones inalámbricas para las WSNs. Son dispositivos de bajo costo y bajo consumo de energía con una corriente de desconexión menor a 6µA. Su conectividad nativa a *Digi Device CloudSM* proporciona una capacidad más rápida de IP a dispositivo y de dispositivo a nube. Tiene interfaces SPI y UART flexibles y una velocidad de transmisión de datos de hasta 72Mbps [66].

Figura. 3.19 Módulo Wi-Fi Bee.
Fuente: [66]
Elaboración: [66]

.

3.5.2 Descripción del circuito XBee

El segundo zócalo de la plataforma permite adaptar cualquiera de los módulos descritos en la sección 3.5.1. Es decir, el prototipo admite dos tipos de módulos de comunicación tipo XBee. Un ejemplo a usar puede ser un XBee-PRO 900/DigiMesh y un GPRS Bee.

El circuito cumple un esquema de conexión tipo XBee para que la integración de estos productos sea fácil. El primer zócalo para XBee está directamente conectado a la MCU de la tarjeta SODAQ Autonomo, mientras que la segunda ranura para módulos XBee está conectada a los puertos de la tarjeta, configurados como interfaz serial 2. En la figura 3.20 se indica el esquema de conexión completo.

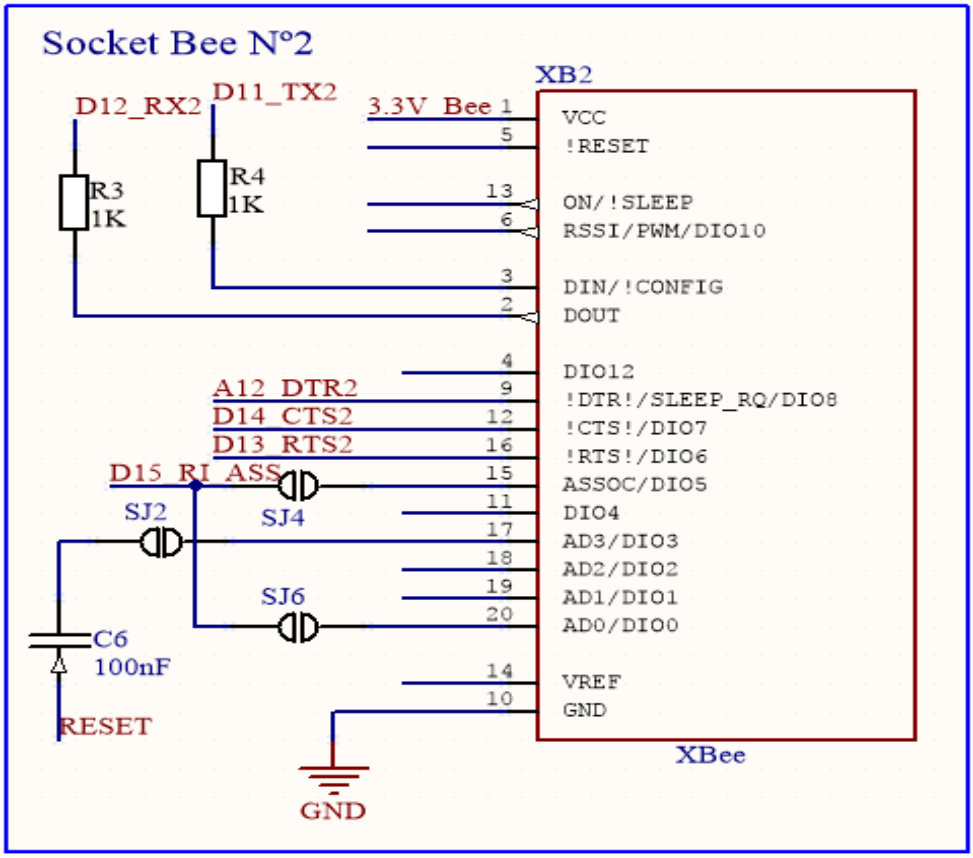


Figura. 3.20 Circuito de conexión para el zócalo XBee2.
Fuente: Autores
Elaboración: Autores

3.5.3GPS

Otro módulo de comunicación que puede ser adaptado a la plataforma es el GPS GP-635T. El diseño del módulo es ideal para ser integrado en espacios reducidos debido a sus dimensiones (35mm x 8mm). Tiene una sensibilidad de seguimiento de -162dBm y tarda sólo 29 segundos para inicializar su funcionamiento en frío. Este módulo de 56 canales, basado en el chip uBlox de séptima generación, tiene un voltaje de funcionamiento entre 3.3 y 5.5V, haciéndolo una opción adecuada para el prototipo. La

.

tasa de actualización de 1s es lo suficientemente rápida para la mayoría de las aplicaciones, pero puede ser reducida hasta 0.2 segundos.

Existe también la opción de integrar un módulo de GPS más robusto vía socket XBee o conector JST-SH de 6 *pins*, si el objetivo es tener mayor sensibilidad, precisión y la adaptabilidad con una antena externa de ganancia óptima para aplicaciones de motas móviles donde se necesita rastrearla continuamente.

Figura. 3.21 Módulo GPS GP-635T.
Fuente: [67]
Elaboración: [67]

3.5.4 Descripción del circuito GPS

En la figura 3.22 se indica el circuito que se usa para la conexión del módulo GPS. Usando un switch DPDT (*Double Pole Double Throw*) para cambiar la conexión serial UART de cruzada a directa. Esto con la finalidad de configurar mediante comandos el módulo. Para el control de suministro de energía se utilizó el *pin* A10 de la tarjeta SODAQ Autonomo.

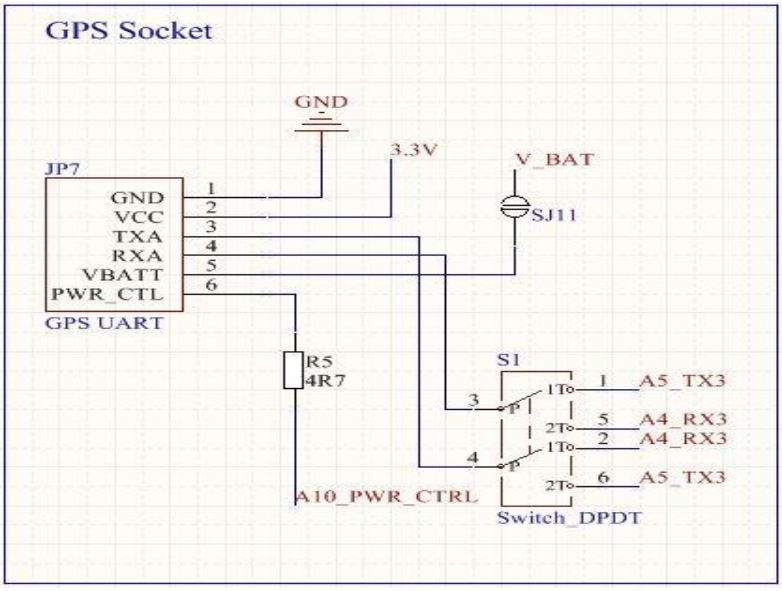


Figura. 3.22 Circuito de conexión para módulo GPS.
Fuente: Autores
Elaboración: Autores

.

3.6 Fabricación del circuito impreso

Para desarrollar el circuito impreso se utilizó la versión gratuita del programa Altium Designer 16, que es una herramienta de software profesional para el diseño de circuitos PCB, fácil de usar, robusta y adaptada a las necesidades de los diseñadores profesionales de circuitos electrónicos.

El primer paso es elaborar el archivo esquemático del prototipo. Previamente a esto, se crea la librería de cada componente en la que se añade el archivo tipo librería integrada a la ventana de proyectos dirigiéndose a la barra de herramientas en **File > New > Project > Integrated Library.** Una vez creado el proyecto con el nombre del componente a diseñar, se adjunta dos tipos de documentos: *Schematic Library* (.SchDoc) y *PCB Library* (.PcbDoc). En SchDoc se indica la forma de conexión del CI, tal y como especifica su hoja de datos técnicos o *Datasheet*. Seguido, en el documento PcbDoc se realiza el diseño en 2D del componente tomando en cuenta las dimensiones para circuitos impresos detalladas por el fabricante. Para la visualización 3D, se agrega el archivo en formato *step* en **Place > 3D Body > Generic 3D Model**. Otra forma de diseño para PcbDoc es utilizar los modelos prediseñados por Altium desde la opción **Tools > IPC Compliant Footprint Wizard**. En la figura 3.23, se indica el ejemplo de una librería creada por los autores.

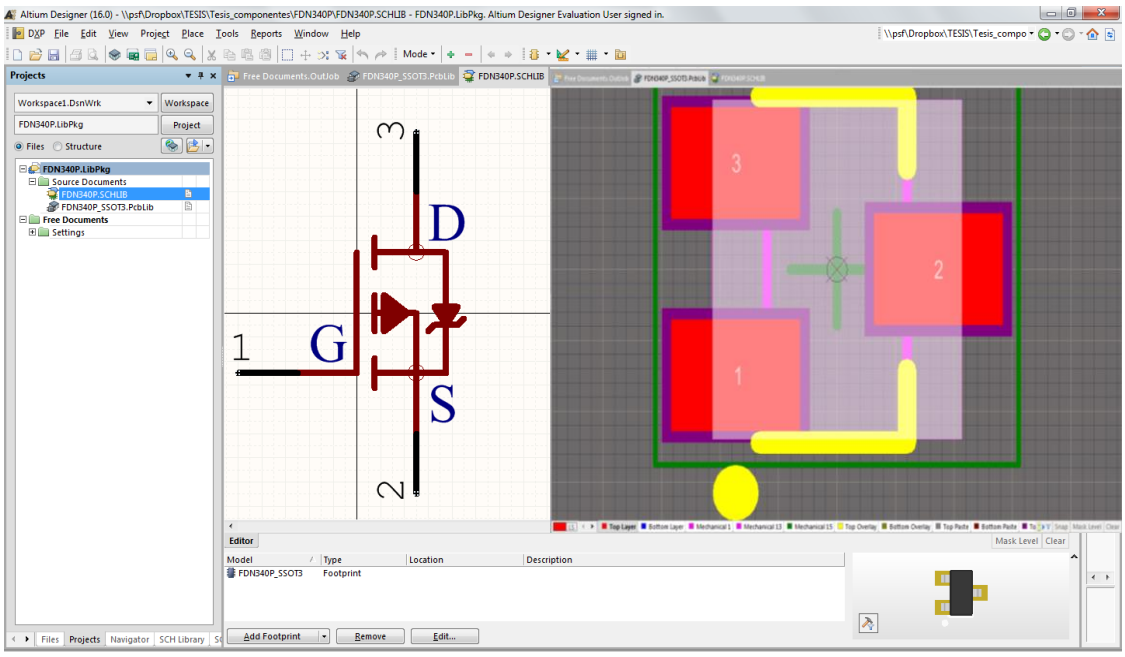


Figura. 3.23 Librería Integrada de FDN340P.
Fuente: Autores
Elaboración: Autores

.

Luego de compilar y validar los cambios del archivo esquemático a PCB, se organizó los componentes teniendo en cuenta los objetivos de una arquitectura modular tanto para shields de Arduino, como para la fácil integración de sensores e implementación de la mota. El diseño final ruteado se puede apreciar en la figura 3.24.

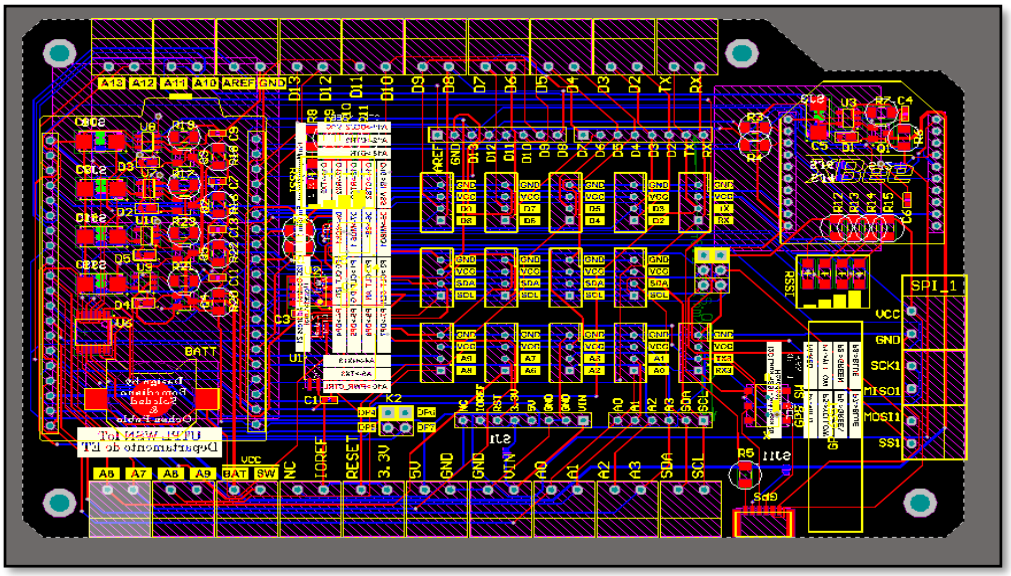


Figura. 3.24 Plataforma WSN e IoT ruteada.
Fuente: Autores
Elaboración: Autores

En las figuras 3.25 y 3.26 se puede visualizar los polígonos, planos sólidos de cobre paralelos entre sí conectados directamente al plano de tierra. Tanto la capa superior como la inferior cuentan con polígonos con la finalidad de reducir y eliminar el ruido eléctrico.

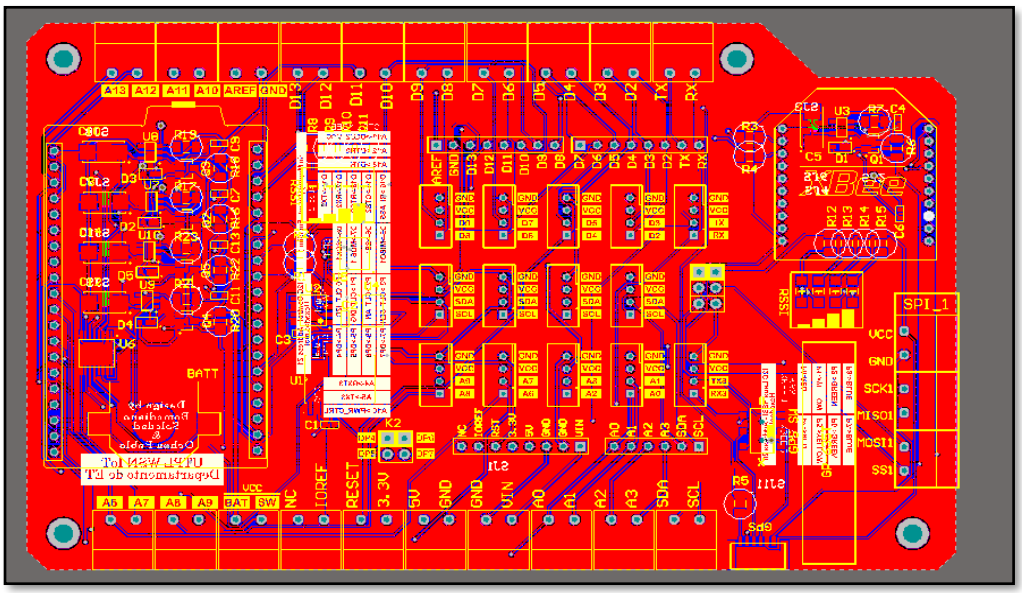


Figura. 3.25 Vista frontal del polígono.
Fuente: Autores
Elaboración: Autores

.

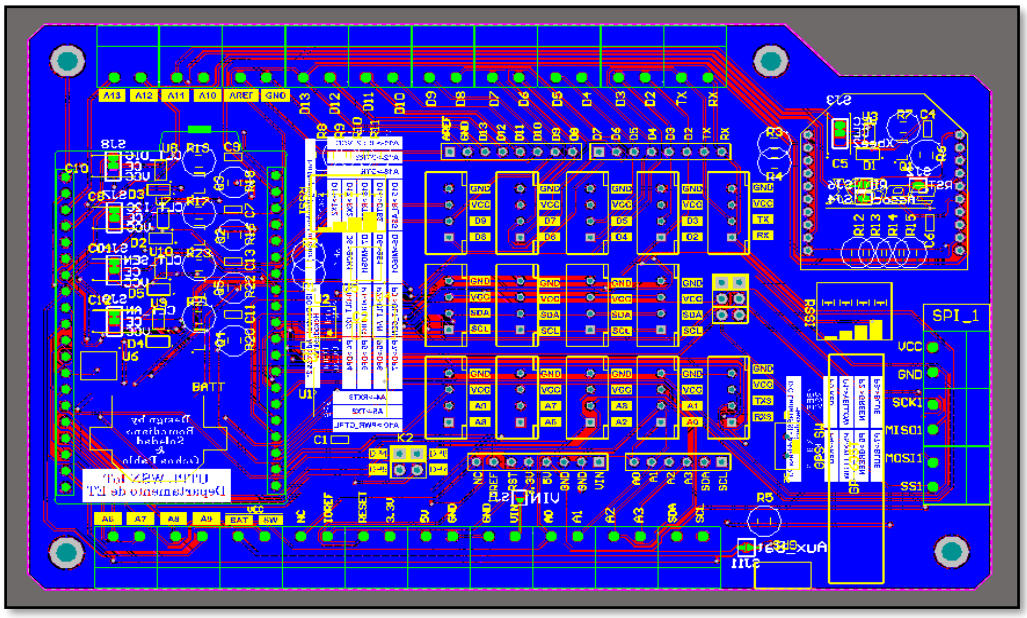


Figura. 3.26 Vista posterior del polígono.
Fuente: Autores
Elaboración: Autores

Altium Designer admite visualizar el diseño en 3D, esta herramienta permite validar la posición adecuada de los elementos y estética de la plataforma. Véase las figuras 3.27 y 3.28.

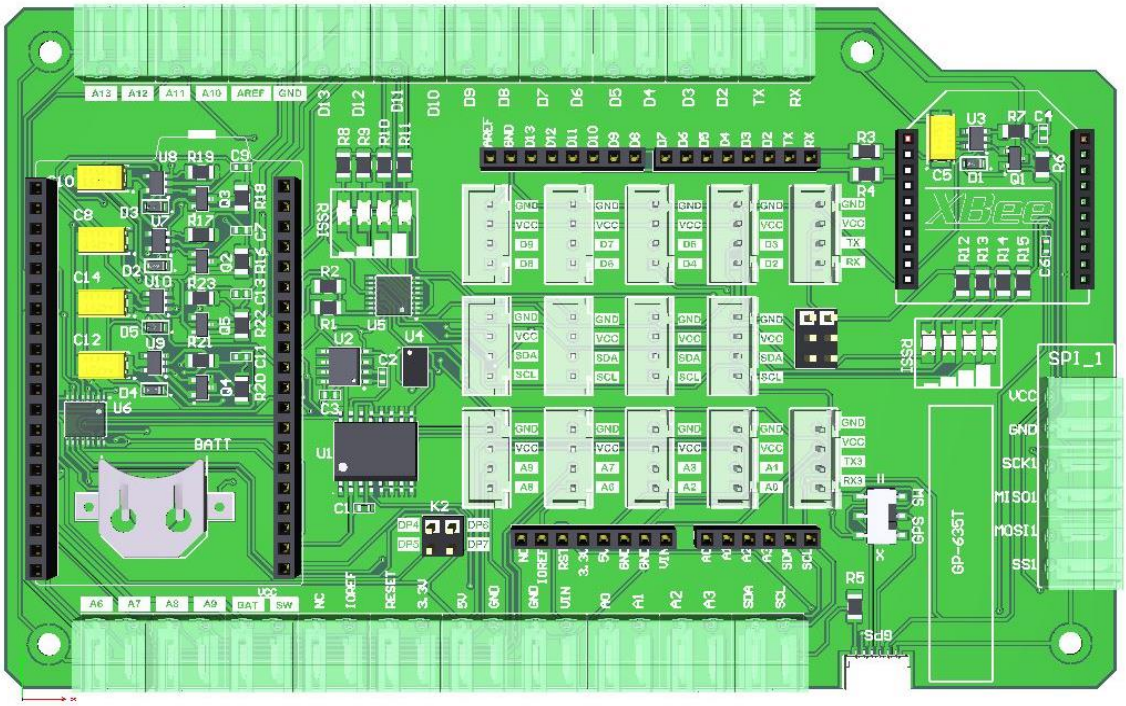


Figura. 3.27 Vista frontal 3D de la plataforma.
Fuente: Autores
Elaboración: Autores

.

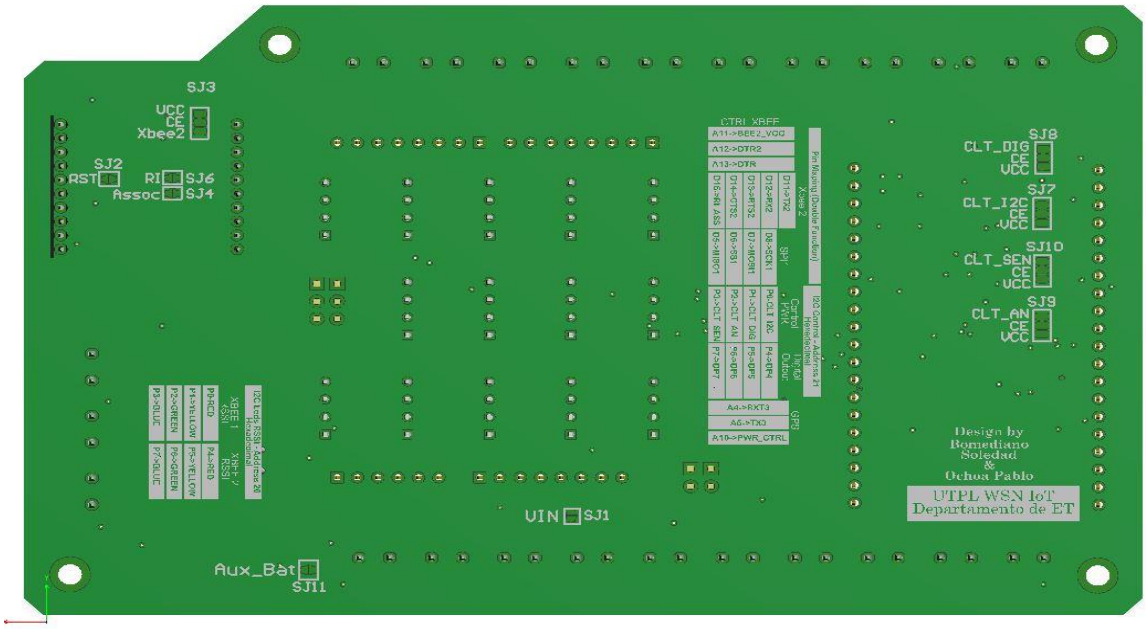


Figura. 3.28 Vista posterior 3D de la plataforma.
Fuente: Autores
Elaboración: Autores

En la figura 3.29 se hace un esbozo del prototipo con su módulo central de procesamiento SODAQ Autonomo y los módulos de comunicación XBee Series 2.

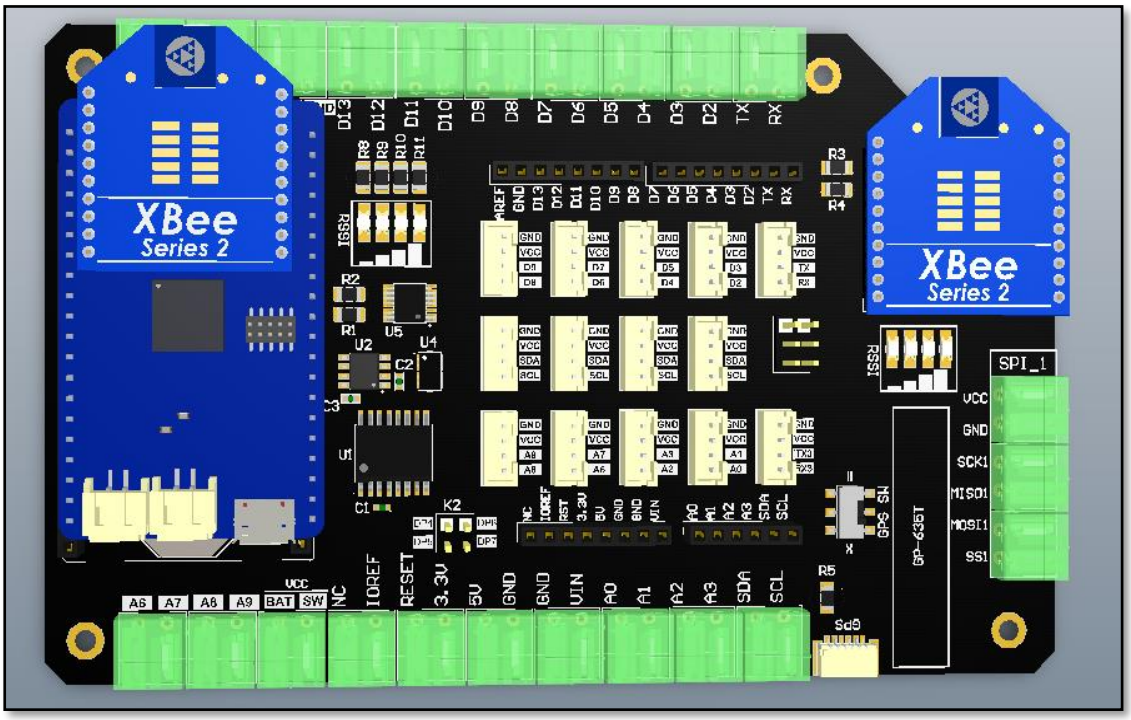


Figura. 3.29 Plataforma WSN + SODAQ Autonomo + XBee.
Elaboración: Autores
Fuente: Autores

.

3.7 Componentes de protección de la mota

Para aplicaciones donde la mota esté expuesta a condiciones ambientales adversas se requiere contar con equipamiento que asegure la funcionalidad de la mota y lo proteja frente a la presencia de objetos o factores externos como polvo y humedad.

Es así que surge la necesidad de medir el grado de exposición que tiene el equipo frente a condiciones hostiles, a través de una medida de protección conocida como IP (*Impermable Protecction* por sus siglas en inglés). El código IP está compuesto por dos números: el primero expresa el índice de protección que tiene contra cuerpos sólidos, mientras que el segundo dígito muestra el grado de protección contra el agua. En la tabla 3.2 se explica cada índice de acuerdo a la normativa DIN EN IEC 60529, tomado del apéndice 7.2 [68].

Por consiguiente, además de seleccionar hardware óptimo para la plataforma, también se seleccionó el adecuado equipo de protección para el nodo, conformado por una caja y conectores IP68.

Tabla 3.2. Grado de protección IP.

Grado de protección contra la introducción de cuerpos sólidos		**Grado de protección contra el agua**	
Primer índice	Descripción	**Segundo índice**	Descripción
0	Sin protección	**0**	Sin protección
1	Protección contra cuerpos sólidos grandes	**1**	Protección contra el goteo de agua vertical (condensación)
2	Protección contra cuerpos sólidos medianos	**2**	Protección contra el goteo de agua inclinada verticalmente
3	Protección contra cuerpos sólidos pequeños	**3**	Protección contra agua en spray
4	Protección contra cuerpos sólidos muy pequeños (granulados)	**4**	Protección contra las salpicaduras de agua
5	Protección contra residuos de polvo	**5**	Protección contra chorros de agua de cualquier dirección con manguera

.

6	Protección total contra la penetración de cualquier cuerpo sólido (estanqueidad)	**6**	Protección contra inundaciones
-	-	**7**	Protección contra inmersión temporal
-	-	**8**	Protección durante inmersión continua
-	-	**9k**	Protección contra introducción de agua pistolas de limpieza de alta presión

Fuente: [69]
Elaboración: Autores

3.7.1 Conectores y caja.

El diseño propuesto se compone de una caja IP65 y conectores IP68, para asegurar completa impermeabilidad al agua y defensa contra objetos extraños. Los conectores impermeables acoplados son de 2, 3 y 4 puertos, usados para conectar el panel solar, sensores de señal digital o analógica, y sensores de conectividad I2C, respectivamente. Adicional a estos conectores, se proporciona un conector de interfaz USB para facilitar al usuario el mantenimiento del nodo. Véase la figura 3.30.

Figura. 3.30 Equipamiento de protección: caja IP65 y conectores IP68.
Fuente: Autores
Elaboración: Autores

.

3.8 Descripción técnica de la plataforma desarrollada

En esta sección se detalla todos los datos técnicos de la plataforma diseñada. La finalidad de este apartado es realizar una guía del hardware y software del prototipo.

3.8.1 Arquitectura modular

La plataforma sigue un esquema de arquitectura modular porque integra bloques al sistema principal. En la figura 3.31 se pueden reconocer fácilmente cuatro subsistemas básicos:

1. Bloque computacional y de almacenamiento: SODAQ Autonomo.
2. Bloque de comunicaciones que puede integrar módulos de la categoría de: ZigBee/802.15.4, GPS y GSM-3G/GPRS.
3. Bloque para shields basadas en Arduino, como pueden ser: Wheater Shield, Vernier Shield, Ethernet Shiel, entre otras.
4. Bloque de sensores con acople eléctrico.

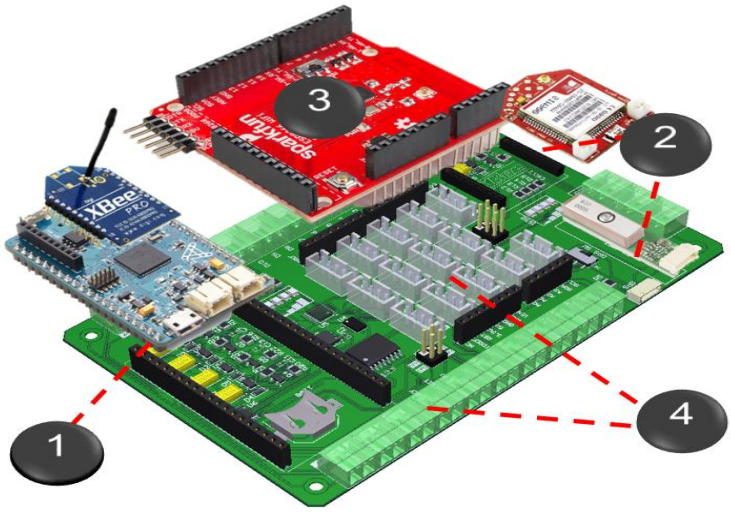


Figura. 3.31 Arquitectura modular.
Fuente: Autores
Elaboración: Autores

3.8.2 Especificaciones

A continuación, se especifican las características técnicas de la plataforma y en la figura 3.32 se señalan sus partes:

- Microcontrolador: ATSAM D21J18
- Frecuencia: 48MHz
- SRAM: 32Kb
- EEPROM: 16Kb
- FLASH: 256Kb

.

- SD Card: 4Gb
- Dimensiones: 139.2x87.6x7.5 mm
- Rango de temperatura: [-40ºC, +85C]

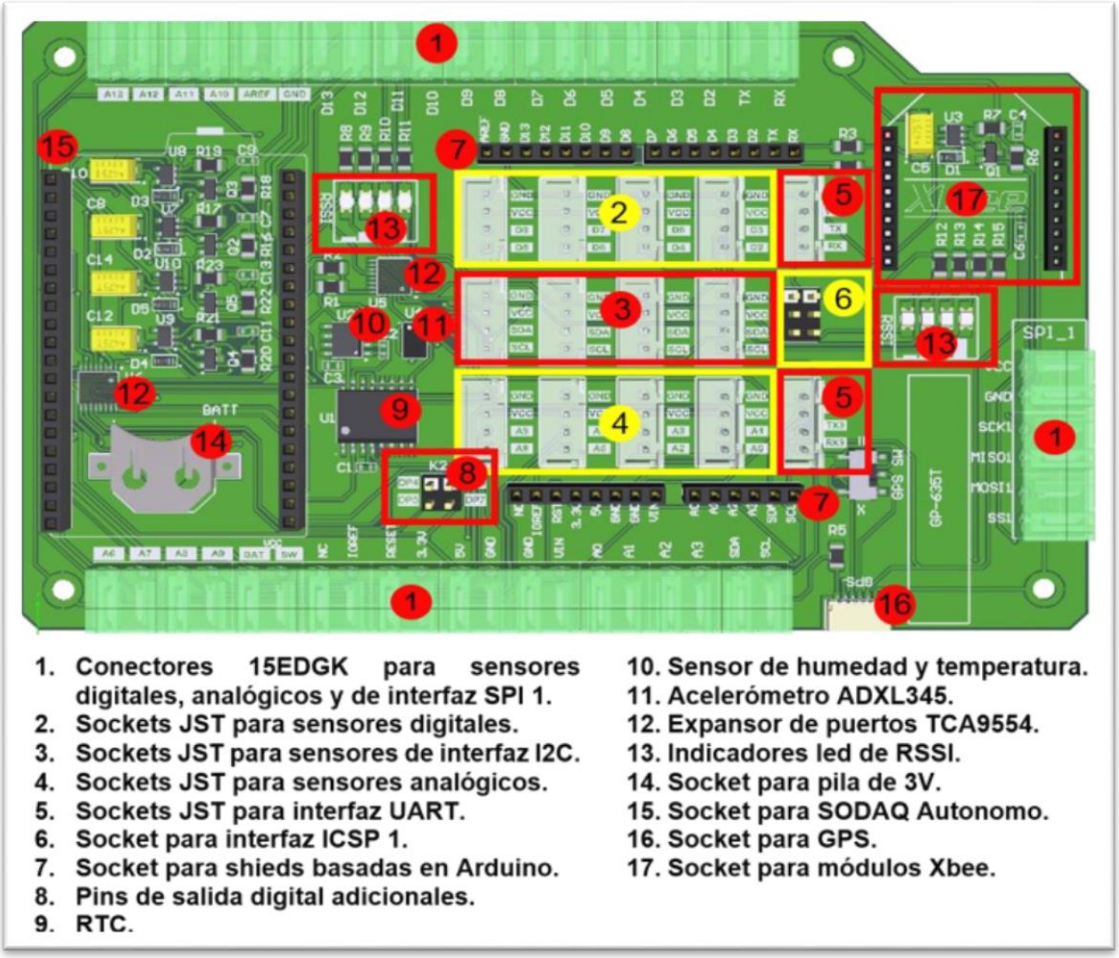


Figura. 3.32 Principales componentes de la vista frontal.
Fuente: Autores
Elaboración: Autores

3.8.3 Diagrama de bloques

El diagrama de bloques representa la estructura de conexión del prototipo. En la figura 3.33 se identifican las conexiones de los subsistemas a la unidad central de procesamiento, SODAQ Autonomo. El microcontrolador cuenta con seis módulos seriales de comunicación que permite configurar los puertos para interfaces I2C, SPI y UART. Para la conexión de dispositivos mediante interfaz I2C, se usó los periféricos definidos SCL (*pin* 32) y SDA (*pin* 33) de la tarjeta, resultando necesaria la utilización de un bus I2C. Cada dispositivo I2C usa un espacio físico de direccionamiento de 7 bits, lo que indica que se puede conectar hasta 128 nodos pero en la práctica es posible conectar hasta 112 a un bus, ya que 16 direcciones están reservadas para fines especiales [70].

.

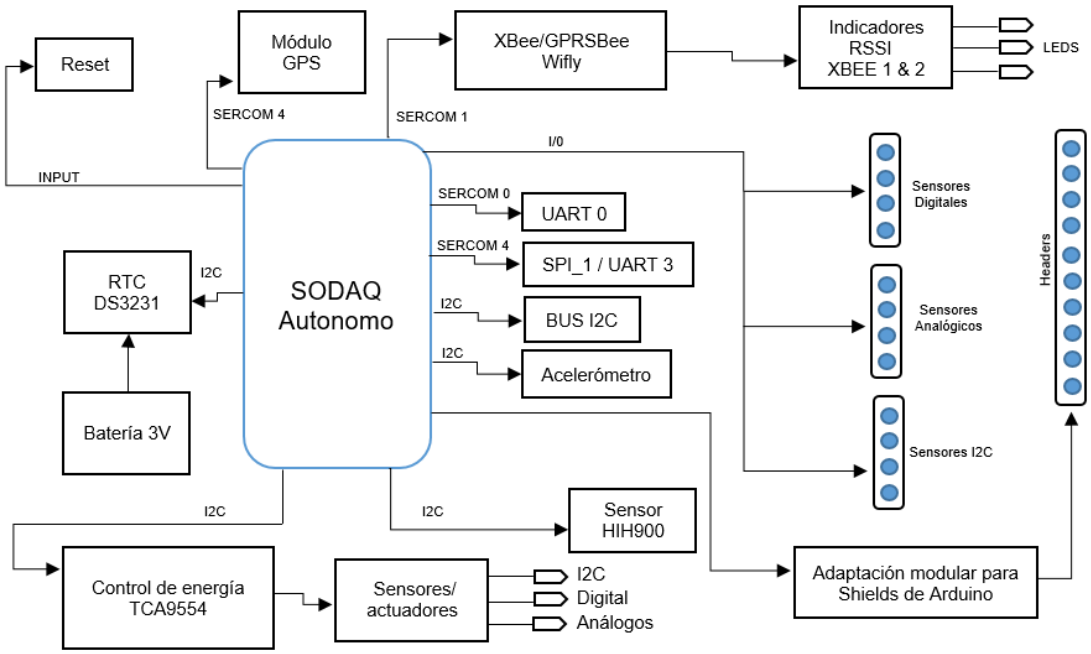


Figura. 3.33 Diagrama de bloques de la plataforma
Fuente: Autores
Elaboración: Autores

El bloque de comunicación inalámbrica es de tipo serial (UART) en los módulos SERCOM 1, SERCOM 4 y SERCOM 5. Como característica adicional al bloque de comunicaciones, se cuenta con indicadores led para control del nivel de intensidad señal RSSI en los dos módulos de formato XBee.

3.8.4 Datos eléctricos

Valores operacionales:

- Voltaje mínimo operacional de la batería 3.3V
- Voltaje de batería máximo operacional 4.2V
- Voltaje de carga USB 5V
- Voltaje de carga de panel solar 4.5V-6V
- Corriente de carga por la USB 200mA
- Corriente de carga por el panel solar 500mA (máx)

Valores máximos absolutos:

- Voltaje en cualquier puerto 3.3V
- Corriente para cualquier puerto I/O digital 7mA
- Voltaje de alimentación de USB 5V
- Voltaje de alimentación de panel solar 6V
- Voltaje de carga de la batería 4.2V

.

3.8.5 Puertos I/O

La plataforma diseñada puede conectarse con otros dispositivos externos a través de diferentes puertos de entrada o salida. En esta sección se explica a detalle cada interfaz periférica disponible en la placa.

3.8.5.1 Análogos

La plataforma dispone de ocho entradas analógicas distribuidas en tres tipos de conectores: regletas, JST de 4 *pins* y 15EDGRC como se indica en la figura 3.26. Cada entrada está directamente conectada a los *pins* del SODAQ Autonomo. El microcontrolador usa un convertidor análogo digital (ADC) de aproximación de 12 bits. El valor de referencia para la entrada es de 0V (GND) y el máximo valor de voltaje de entrada es 3.3V, que corresponde al voltaje de operación del microcontrolador. Un puerto de entrada analógica también puede ser utilizado como puerto de salida digital y algunos de ellos para comunicación serial. Por consiguiente, de las trece entradas analógicas que tiene habilitadas la tarjeta SODAQ Autonomo, se utilizó seis puertos como se indica en la tabla 3.3:

Tabla 3.3. Uso de puertos analógicos

Puerto analógico		**Puerto digital**
ATSAM D21J18	**SODAQ Autonomo**	**Función en la Plataforma**
PB09	A4	RX3 (GPS/UART3)
PB08	A5	TX3 (GPS/UART3)
PA07	A10	PWR_CTRL (Para control de alimentación del GPS)
PB03	A11	BEE2_VCC (Para control de alimentación del Xbee 2)
PB02	A12	DTR2 (Para control de flujo Xbee 2)
PB1	A13	DTR (Para control de flujo Xbee 1)

Fuente: Autores
Elaboración: Autores

3.8.5.2 Digitales

El prototipo tiene puertos digitales los cuales pueden ser configurados como entradas o salidas dependiendo de la necesidad de la aplicación. Los valores de voltaje correspondientes a estados digitales son:

- 0V para 0 lógico
- 3.3V para 1 lógico

.

Se han ocupado 9 puertos digitales de los 15 disponibles que tiene el SODAQ Autonomo. Los puertos son utilizados para funciones específicas que se detalla en la tabla 3.4:

Tabla 3.4. Uso de puertos digitales para comunicación.

Tipo de comunicación	Puerto en el SODAQ Autonomo	Función	Tipo de Comunicación	Puerto en el SODAQ Autonomo	Función
UART Serial 2 para el XBee 2	D11	TX2	SPI1	D5	MISO1
	D12	RX2		D6	SS1
	D13	RTS2		D7	MOSI1
	D14	CTS2		D8	SCK1
	D15	RI_AS_2			

Fuente: Autores
Elaboración: Autores

3.8.5.3 PWM (Pulse-Width Modulation)

Para generar una forma de onda periódica se utiliza los puertos con la función *Timer/Control*. El TC mediante canales de comparación puede generar una forma de onda en dos puertos de salida. Existen cuatro configuraciones que se puede elegir con el registro de control de generación de onda (CTRLA.WAVEGEN) del microcontrolador, estas son:

- Frecuencia normal (NFRQ)
- Frecuencia de coincidencia (MFRQ)
- Normal PWM (NPWM)
- Match PWM (MPWM)

Los puertos digitales disponibles para generar PWM se indican en la tabla 3.5:

Tabla 3.5. Puertos para PWM.

Timer/Control	ATSAM D21J18	SODAQ Autonomo	Timer/Control	ATSAM D21J18	SODAQ Autonomo
TC5	PB10	D3	TC4	PB12	D5
	PB11	D4		PB13	D6
	PB14	D7	TC3	PA14	D9
	PB15	D8		PA15	D10

Fuente: Autores
Elaboración: Autores

.

3.8.5.4 UART

Se dispone de cuatro interfaces seriales (UART) en la plataforma: una de ellas usa el SERCOM5 para conectar el módulo XBee1. El segundo serial corresponde al SERCOM1 para la conexión del módulo XBee2. Un tercer serial fue configurado mediante SERCOM4 para ser usado como interfaz serial auxiliar (UART3) o para la conexión al módulo GPS. Mientras que el cuarto serial corresponde al SERCOM0 y es usado como interfaz serial para *shields* basadas en Arduino.

Tabla 3.6. Correspondencia de puertos para Seriales.

Módulo de comunicación serial	UART	Puertos	Usos
SERCOM0	Serial	RX, TX	D0, D1 / UART0
SERCOM5	Serial1	RX1, TX1	XBee1
SERCOM1	Serial2	RX2, TX2	XBee2
SERCOM4	Serial3	RX3, TX3	GPS / UART3

Fuente: Autores
Elaboración: Autores

3.8.5.5 I2C

Se utiliza un bus I2C vía SERCOM 2 para comunicar dispositivos en paralelo; el acelerómetro, el RTC, dos expansores de puertos I/O, un sensor de temperatura y humedad y cuatro conectores para sensores auxiliares. En todos los casos el microcontrolador actúa como maestro mientras que el resto de dispositivos son esclavos. El valor de Rp (resistor *pull-up*) usado en el bus I2C es de 2.2 kΩ. Véase figura 3.34 [70].

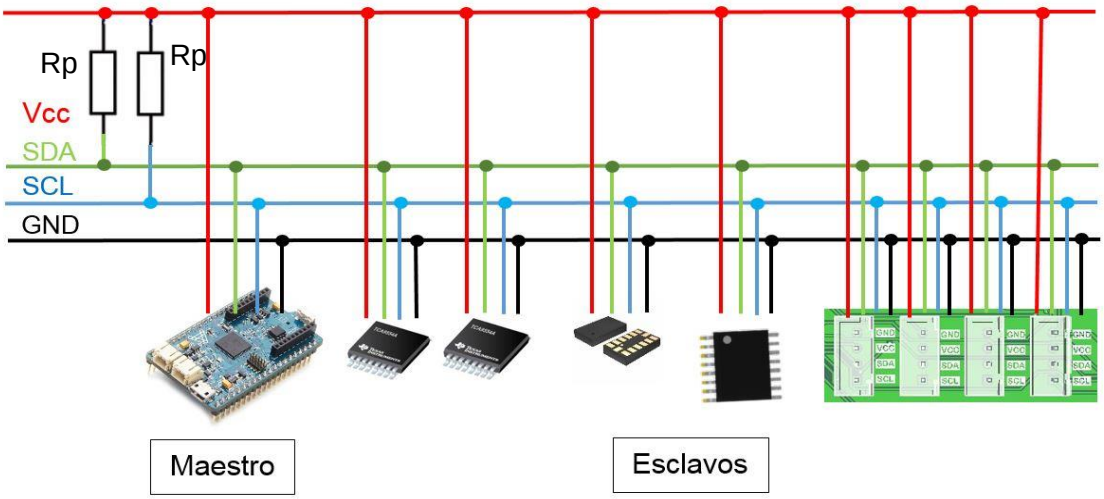


Figura. 3.34 Conexión bus I2C.
Fuente: Autores
Elaboración: Autores

.

3.8.5.6 SPI

SPI es un estándar de comunicación usado para transferencia de información entre dispositivos electrónicos. Dentro del bloque computacional SODAQ Autonomo, se usa la interfaz SPI del microcontrolador para la comunicación con la tarjeta Micro SD y la memoria flash externa mediante SERCOM3. Además, con modificación de la librería, se configuró el SERCOM4 como SPI1 alternativo para ser usado como interfaz auxiliar de acuerdo a la aplicación que necesite el usuario.

3.8.6 RTC

El prototipo fue diseñado con un RTC para adquirir la hora actual y sincronizar tareas. Incluye una fuente de poder externa de 3V. La información que se adquiere del RTC DS3231NS permite que la plataforma sea programada para realizar acciones específicas como dormir o hibernar.

3.8.7 Indicadores de intensidad de señal

Se cuenta con 2 indicadores led de nivel de señal recibida (RSSI) para cada *socket* XBee de la plataforma. Los leds están programados para ser activados con niveles lógicos en bajo de acuerdo a cuatro rangos de valor de RSSI. Esta información es directamente obtenida mediante comandos AT en modo API.

Por ejemplo, el módulo (XBee-PRO 900/DigiMesh) tiene una potencia de 50mW y sensibilidad de -100dBm, resultando una potencia de salida de -83dBm. Tomando en cuenta estos parámetros, se estableció una escala cada -5dBm que representan a los cuatro niveles de señal indicados en la figura 3.35.

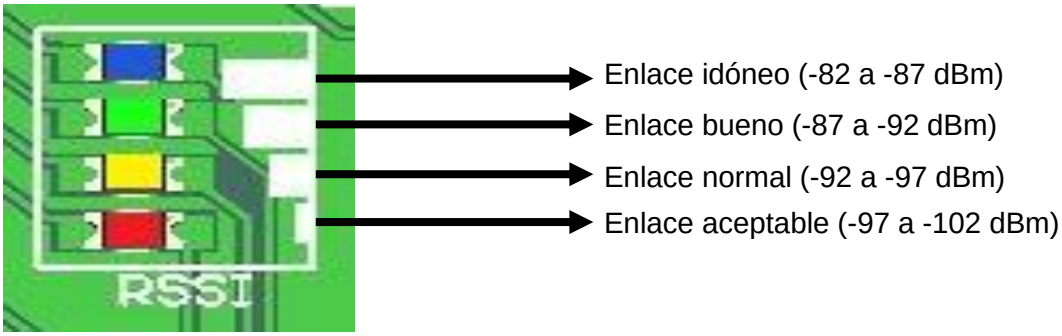


Figura. 3.35 Indicadores RSSI
Fuente: Autores
Elaboración: Autores

.

3.8.8 Jumpers

Hay diez *jumpers* en la plataforma que sirven para activar o desactivar ciertas funciones. En la figura 3.36 se indica la localización de estos y su respectiva función en la tabla 3.7.

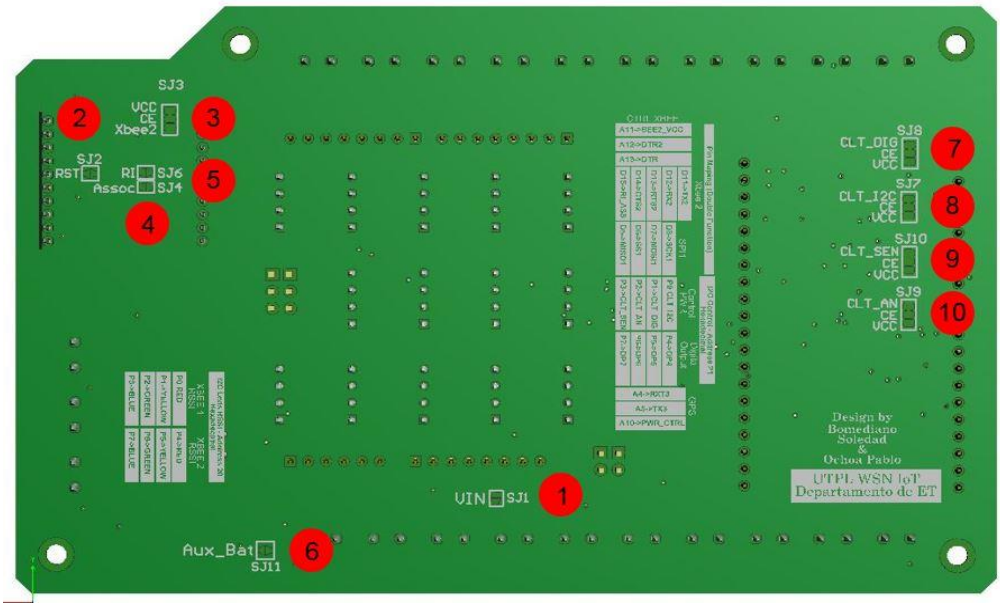


Figura. 3.36 Vista posterior de la plataforma con sus respectivos jumpers.
Fuente: Autores
Elaboración: Autores

Tabla 3.7. Funcionalidad de *jumpers*.

JUMPER	FUNCIÓN
SJ1	Puede usarse para conectar 3.3V a VIN.
SJ2	Puede usarse para conectar DIO3 al reset.
SJ3	Puede usarse para mantener siempre el activo los 3.3V del XBee2.
SJ4	Puede usarse para conectar al *pin* Network Association" (DIO5).
SJ6	Puede usarse para conectar al *pin* "Ring Indicator" (DI00).
SJ11	Puede usarse para energizar el *pin* VBAT del GPS.
SJ8	Puede usarse para mantener siempre activo 3.3V en los conectores para sensores digitales.
SJ7	Puede usarse para mantener siempre activo los 3.3 V en conectores para sensores I2C.
SJ10	Puede usarse para mantener siempre activo los 3.3V de alimentación de los sensores internos.
SJ9	Puede usarse para mantener siempre activo los 3.3V en los conectores para sensores analógicos.

Fuente: Autores
Elaboración: Autores

.

3.8.9 Tarjeta SD Card

SODAQ Autonomo tiene un módulo externo para almacenamiento de datos denominada SD *Card* con una capacidad máxima de almacenamiento de 4Gb. La conexión de la tarjeta SD se puede observar en la figura 3.37.

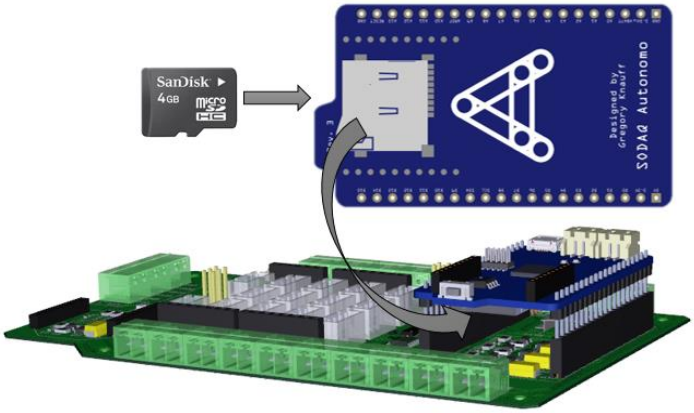


Figura. 3.37 Conexión de la tarjeta SD a la plataforma.
Fuente: Autores
Elaboración: Autores

3.8.10 Fuentes de alimentación

3.8.10.1 Batería

La tarjeta permite integrar una batería Lipo con voltaje nominal de 3.7V y corriente de 6600mA mediante un socket JST de dos *pins*. En la figura 3.38 se indica la conexión de la batería a la plataforma.

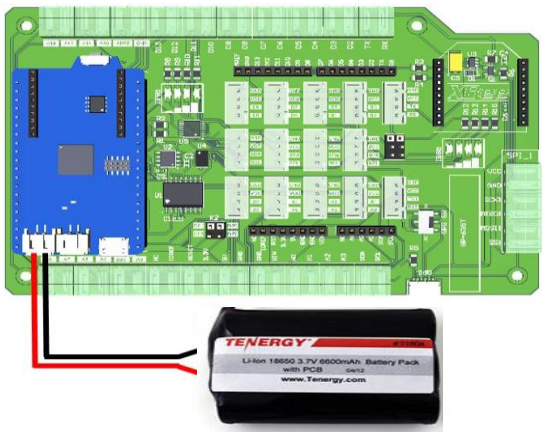


Figura. 3.38 Conexión de la batería de Lipo.
Elaboración: Autores
Fuente: Autores

3.8.10.2 Panel Solar

La plataforma cuenta con autonomía energética, gracias a la disponibilidad de un enchufe para panel solar. El voltaje funcional es de 5V y va directamente conectado a la caja de la mota mediante un conector impermeable de dos puertos.

.

Figura. 3.39 Conexión del panel solar a la plataforma.
Elaboración: Autores
Fuente: Autores

3.8.10.3 USB

La plataforma puede ser alimentada por medio de interfaz micro USB a USB desde tres fuentes:

- USB del computador.
- Cargador USB
- Adaptador de carga USB para vehículo.

El voltaje de carga de la USB debe ser de 5V y un máximo de corriente de 100mA.

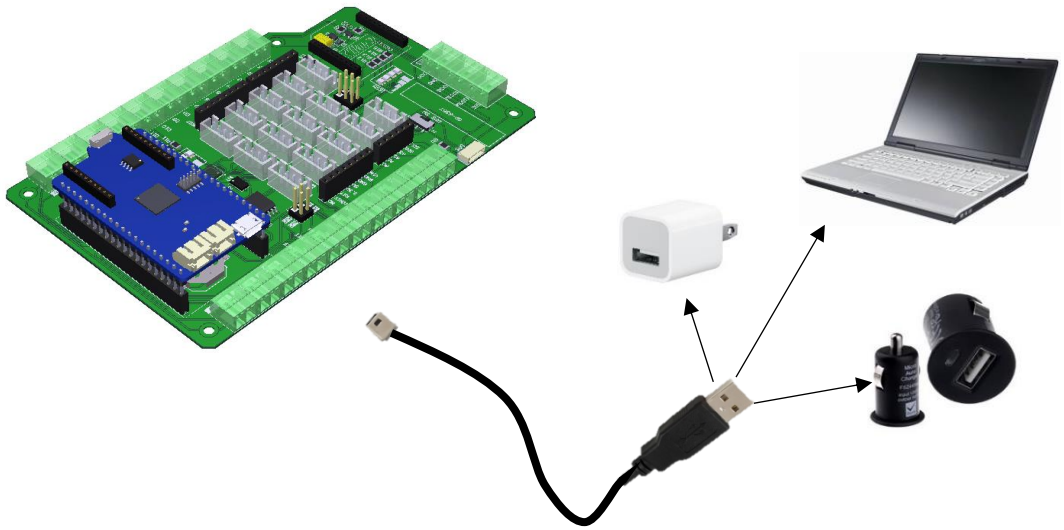


Figura. 3.40 Posibles conexiones USB.
Elaboración: Autores
Fuente: Autores

.

3.8.11 Uso de librerías

Para la validación de cada componente de la plataforma, módulos de comunicación y las diferentes interfaces, se utilizó librerías estándar y de terceros especificadas en la sección siguiente. El entorno de Arduino IDE viene integrado con una serie de librerías estándar que facilitan al usuario la programación. Las librerías se encuentran en Archivos > Ejemplos como se muestra en la figura 3.41.

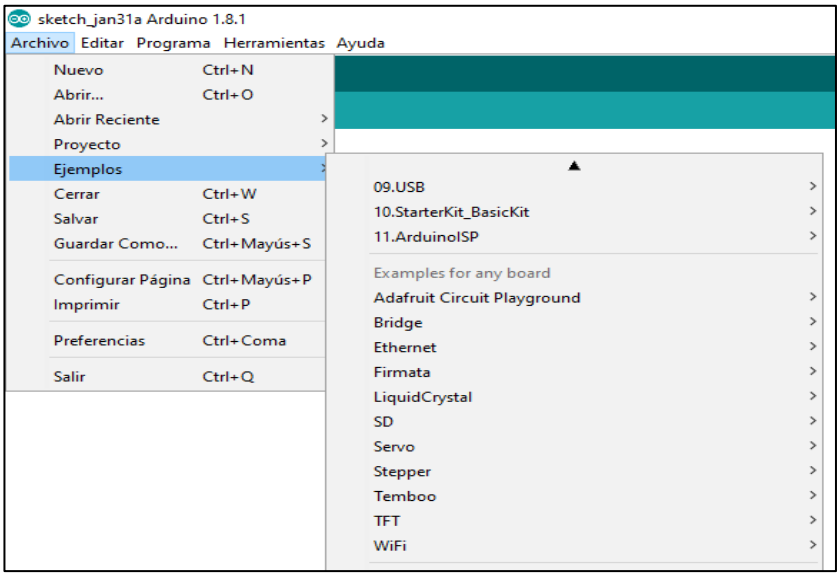


Figura. 3.41 Librerías estándar.
Elaboración: Autores
Fuente: Autores

Además, se realizó modificaciones en las librerías "*variant.h*" y "variant.cpp" correspondiente al paquete de software de la tarjeta SODAQ Autonomo, como se puede observar en la figura 3.42. Esta modificación permite el uso de las interfaces de comunicación (Serial2 y Serial3) adaptados al hardware de la plataforma en los *sockets* XBee y GPS (ver tabla 3.6), así como también el uso de registros en los puertos configurados con funciones específicas de *pin mapping* (ver tablas 3.3 y 3.4).

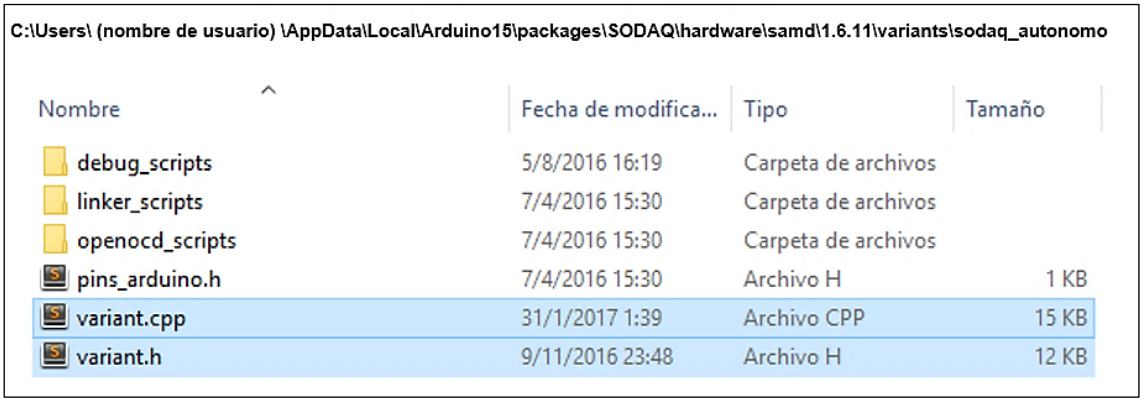


Figura. 3.42 Ubicación de las librerías *variant.h* y *variant.cpp.*
Elaboración: Autores
Fuente: Autores

.

Las librerías tanto estándar como de terceros usadas para la validación de la plataforma, son las que se describe a continuación.

3.8.11.1 Librerías estándar

SD: biblioteca SD que permite leer y escribir en tarjetas SD.

- Información de la tarjeta: obtiene información sobre la tarjeta SD.
- *Datalogger*: Registra los datos de tres sensores analógicos a una tarjeta SD.
- Archivo de volcado: lee un archivo de la tarjeta SD.
- Archivos: Crea y destruye un archivo de tarjeta SD.
- Lista de archivos: Imprime los archivos en un directorio de una tarjeta SD.
- Leer Escribir: Lee y escribe datos desde y hacia una tarjeta SD.

Wire: permite la comunicación entre dispositivos o sensores conectados a través del bus de interfaz de dos cables (I2C).

SPI: habilita la comunicación con dispositivos que usan el Bus de Interfaz Periférica Serie (SPI) [21].

3.8.11.2 Librerías de terceros

TinyGPSPlus: librería usada para la adquisición de la posición geográfica por satélite con el módulo GPS-635T. TinyGPSPlus es una pequeña biblioteca de GPS para Arduino que proporciona análisis NMEA (*National Marine Electronics Association*) universal basado en el trabajo de "distanceBetween" y "courseTo" cortesía de Maarten Lamers. La opción para agregar satélites courseTo (.), y cardinal (.) es de Matt Monson. Las mejoras de precisión de ubicación fueron sugeridas por Wayne Holder. Librería creada por Mikal Hart [71].

SparkFun_ADXL345: utilizada para el acelerómetro digital ADXL345 configurado para comunicación I2C. La librería incluye diferentes ejemplos creados por la autora Elizabeth Robert [72].

Adafruit_ADXL345: otra de las librerías para el acelerómetro ADXL345 en el cual se encuentra un ejemplo para probar el sensor. Creado por Kevin Townsend [73].

Adafruit_Sensor: librería complementaria de Adafruit_ADXL345, la misma que contiene los drivers de algunos sensores soportados por Adafruit, entre ellos el ADXL345. Creado por Kevin Townsend [74].

xbee-arduino-master: librería Arduino para comunicarse con XBee en modo API, con soporte para Series 1 (802.15.4) y Series 2 (ZB Pro / ZNet). Esta biblioteca incluye soporte para la mayoría de los tipos de paquetes, incluyendo: TX / RX, comandos AT, AT remoto, muestras de I/O y estado del módem. Creado por Andrew Wrapp [75].

.

CAPITULO IV

4. IMPLEMENTACIÓN Y EVALUACIÓN DE LA PLATAFORMA

.

4.1 Pruebas y simulación en *protoboard*

Durante las pruebas preliminares se utilizó tres *breakout board* de sensores de Sparkfun. Un sensor de luminosidad (TCL2561), un giroscopio (ITG-3200) y un sensor de temperatura y humedad (HIH-6130). Todos los sensores están conectados al bus I2C de la tarjeta SODAQ Autonomo.

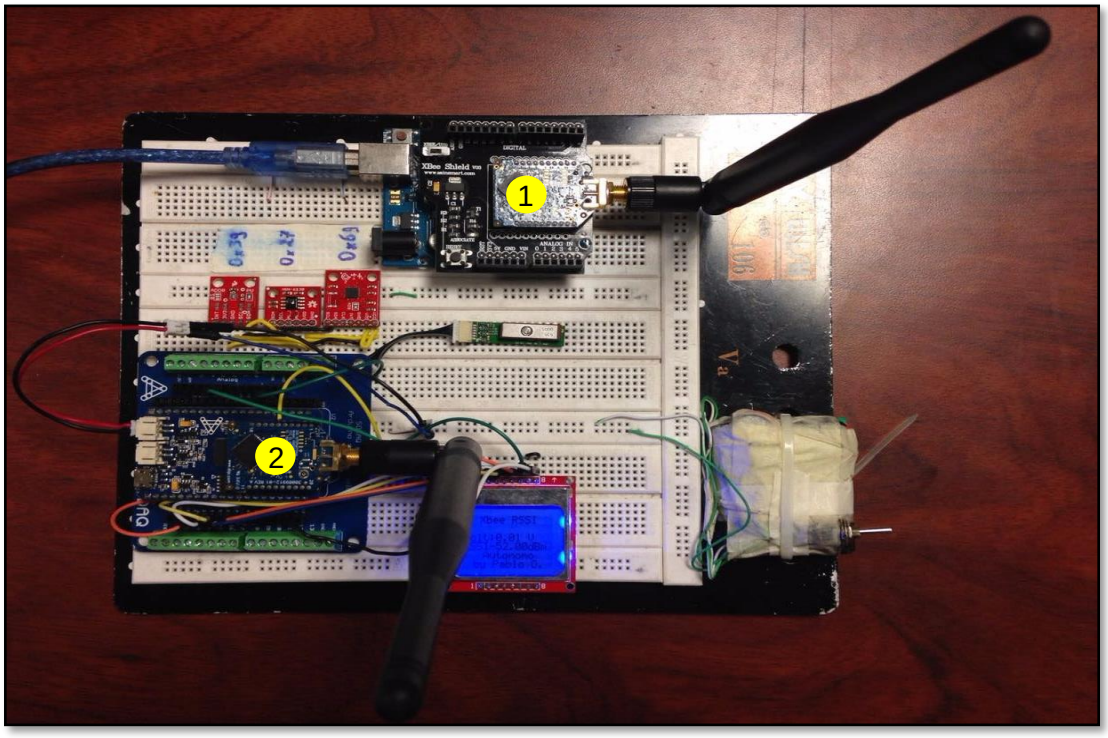


Figura. 4.1 Simulación SODAQ Autonomo con sensores I2C, GPS. XBee Pro 900.
Fuente: Autores
Elaboración: Autores

Para las pruebas de transmisión y recepción se utilizó dos módulos XBee Pro 900 configurados en modo API 2 con escapes. El módulo 1, como se indica en la figura 4.1, se establece como coordinador encargado de enviar el estado de los sensores; mientras que el módulo 2, se configura como dispositivo final que recibe la información. Al enviar en modo API se puede acceder a la información almacenada en cada byte de la trama; por ende, se estima el nivel de señal recibida al enviar un comando ATDB.

Se utilizó también, el ancho de pulso (PWM) del *pin* 6 del XBee. Este *pin* emite variaciones de voltaje que representan la intensidad de señal recibida. En la imagen 4.2 se indica el resultado obtenido. Para esta prueba se utilizó adicionalmente un Arduino Uno, una XBee Shield y una Autonomo Screw Shield.

.

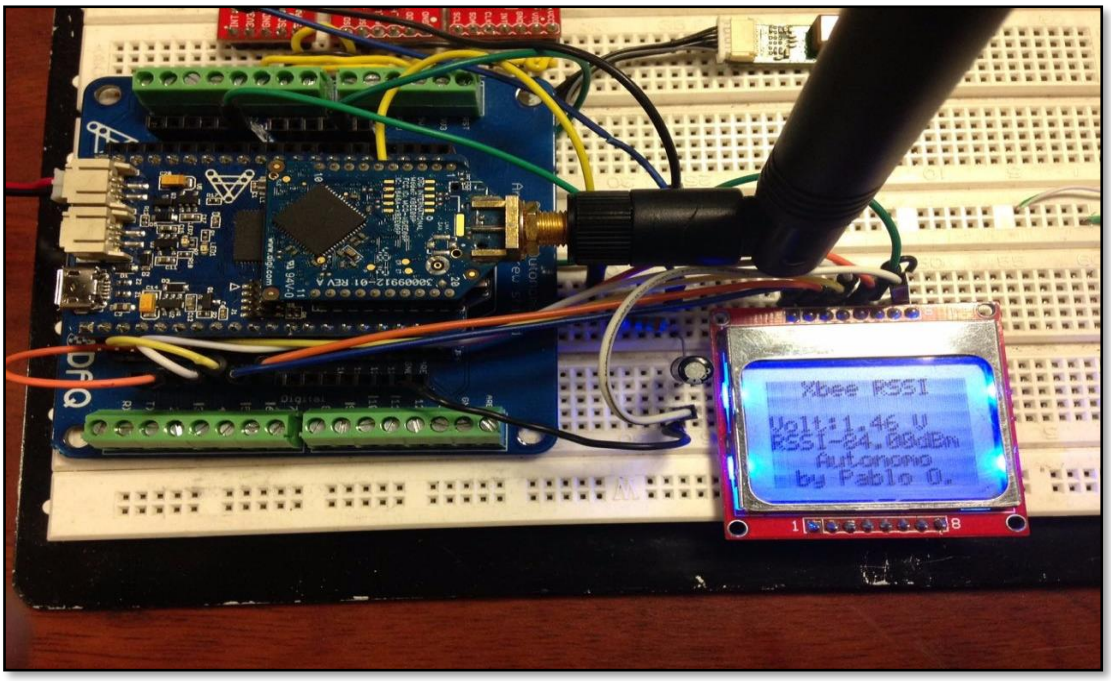


Figura. 4.2 Indicador RSSI, con envío de paquetes.
Fuente: Autores
Elaboración: Autores

4.2 Pruebas de consumo de energía

Para las pruebas de consumo de energía se utilizó una fuente de poder de 5V, simulando alimentación por panel solar y una fuente de 3.7V en función de una batería de Lipo. Véase figuras 4.3 y 4.4.

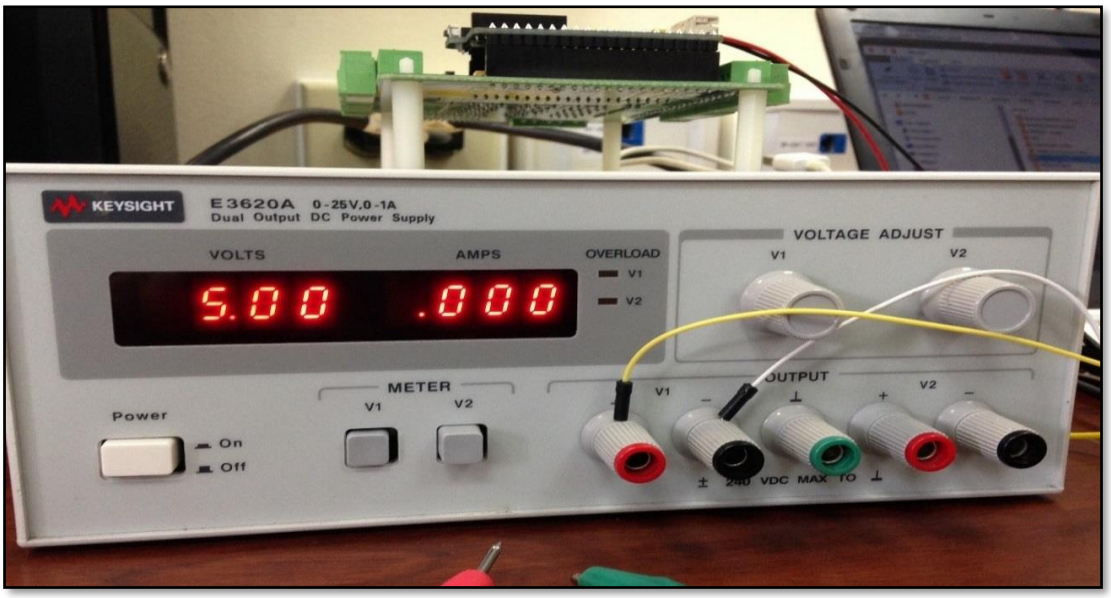


Figura. 4.3 Prueba de energía con fuente de 5V.
Fuente: Autores
Elaboración: Autores

.

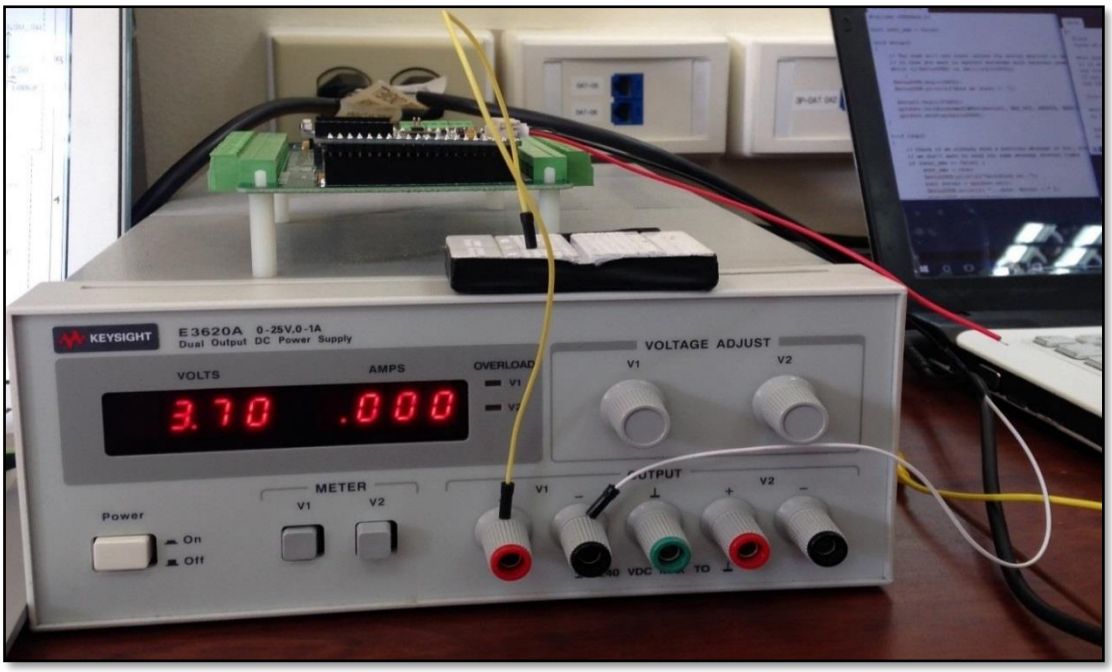


Figura. 4.4 Prueba de energía con fuente de 3.7 V.
Fuente: Autores
Elaboración: Autores

Para la medición de consumo de corriente de la plataforma y sus componentes fueron tomadas muestras en un computador con el software BenchVue que incluye el multímetro digital de Keysight. En la figura 4.5 se muestra la gráfica de la señal de corriente en función del tiempo, y en la figura 4.6 se valida la medición realizada.

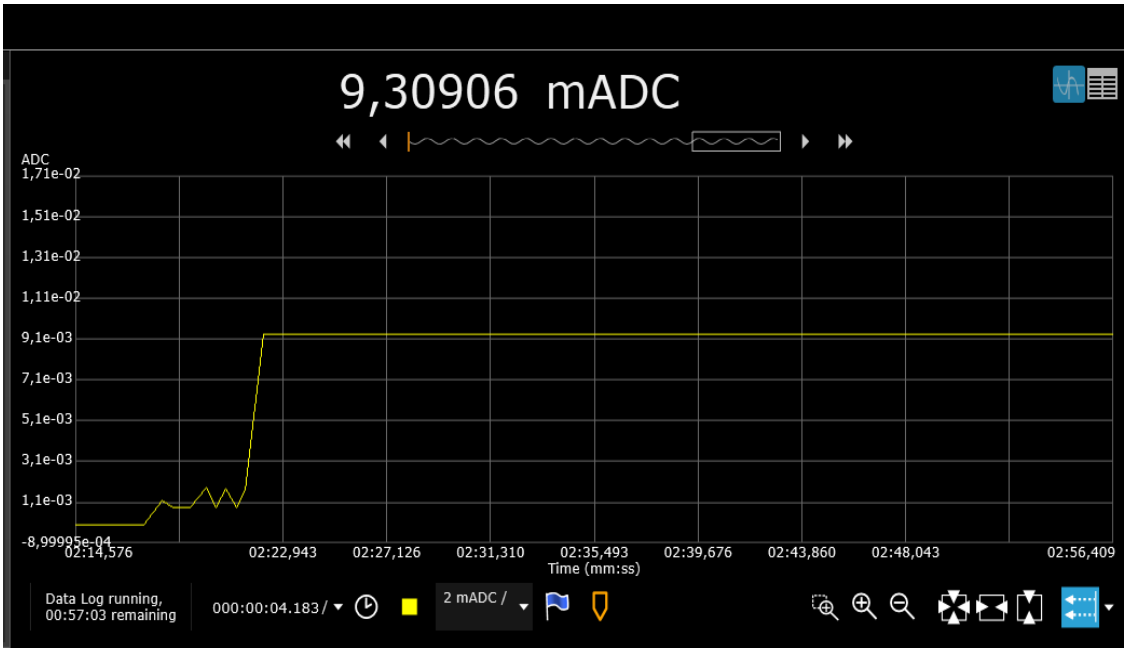


Figura. 4.5 Consumo del prototipo sin módulos.
Fuente: Autores
Elaboración: Autores

.

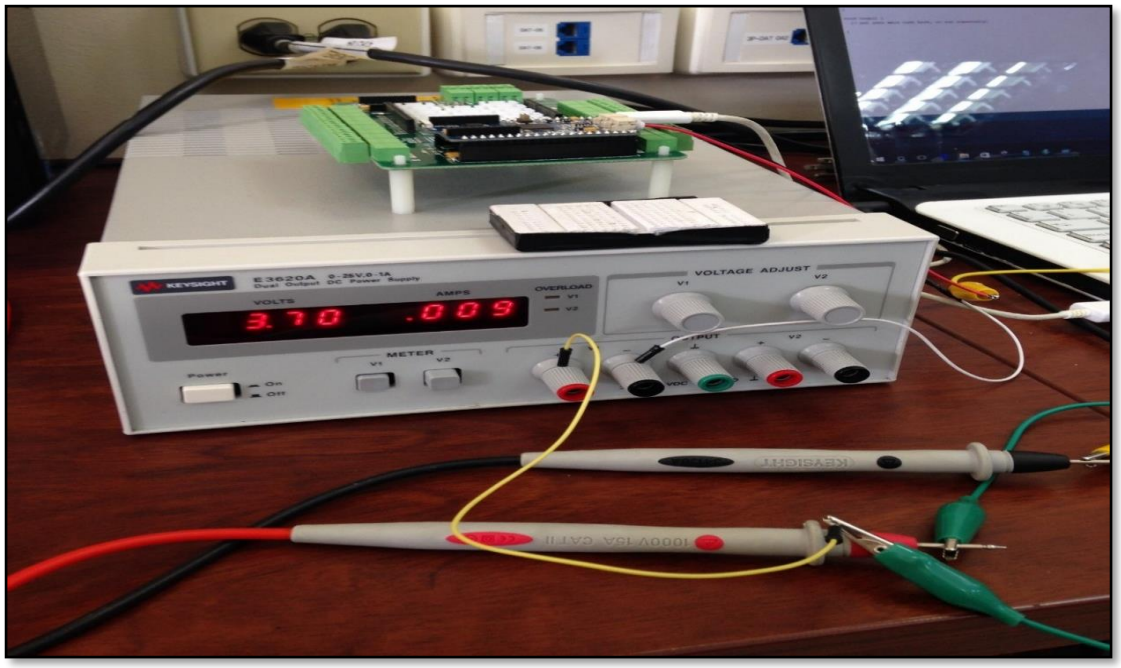


Figura. 4.6 Indicador de consumo de la plataforma en la fuente E3620A Keysight.
Elaboración: Autores
Fuente: Autores

El consumo de la plataforma, al tener activos los indicadores led, es un factor importante dentro del análisis energético del prototipo. En consecuencia, se realizó mediciones del gasto de corriente de los mismos, programando la tarjeta para enviar pulsos en alto y bajo a los leds. La señal obtenida se indica en la figura 4.7.

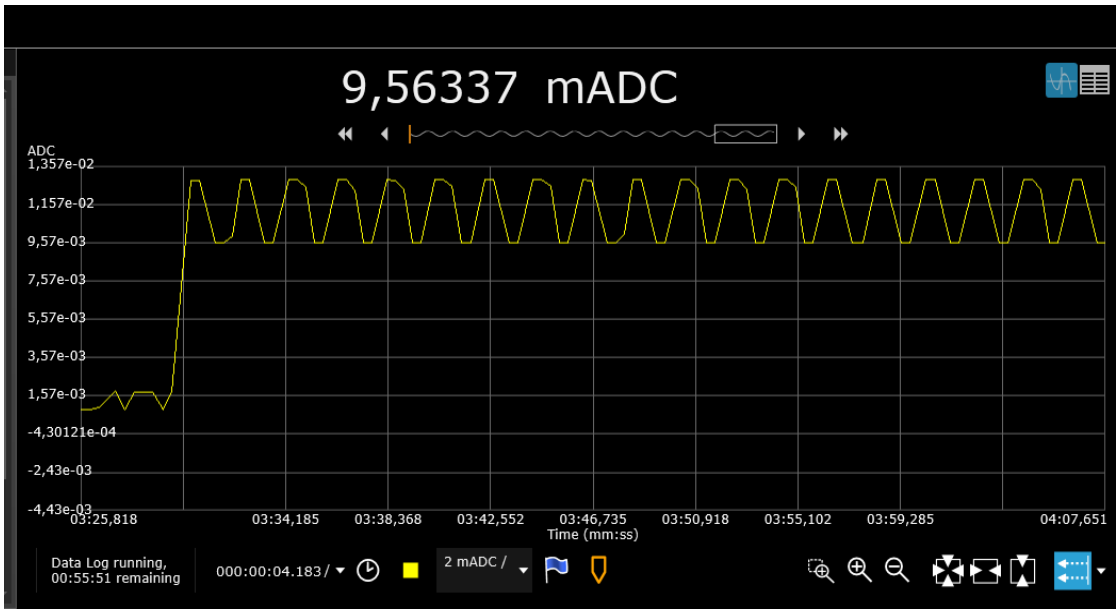


Figura. 4.7 Consumo de leds.
Fuente: Autores
Elaboración: Autores

Los resultados obtenidos al activar los sensores internos, leds, GPS y RTC se muestran en la tabla 4.1:

.

Tabla 4.1 Consumo de corriente

Descripción	ON	OFF	SLEEP
Solo la plataforma	9 mA	0 µA	62 µA
Leds RSSI	0.5 mA	0 µA	-
GPS + plataforma	69 mA	0 µA	-
Acelerómetro	140 µA	0 µA	0,08 mA
RTC	200 µA	0 µA	110 µA
HIH9130	1 mA	0 µA	1 µA

Fuente: Autores
Elaboración: Autores

Asimismo, en las figuras 4.8 a 4.11 se grafican las señales obtenidas al integrar los módulos GPRS Bee, GPS GP-635T y XBee PRO 900, respectivamente.

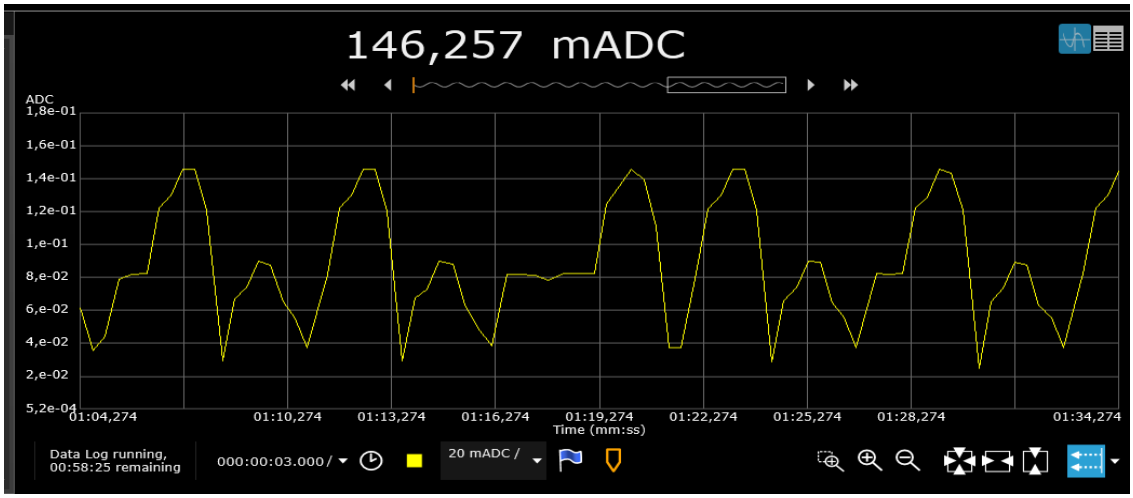


Figura. 4.8 Consumo de la plataforma con el módulo GPRS Bee.
Fuente: Autores
Elaboración: Autores

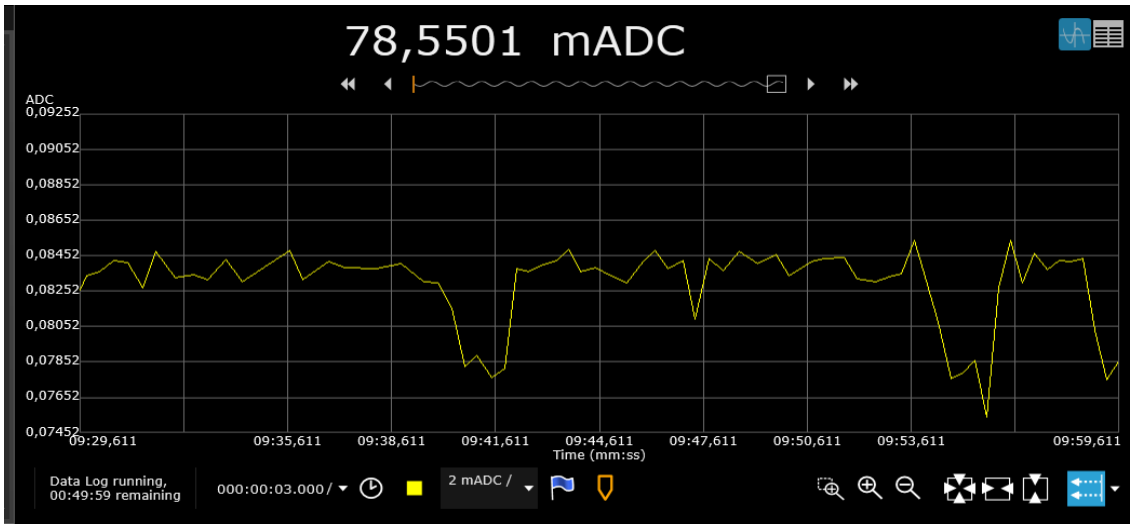


Figura. 4.9 Consumo del módulo GPS.
Fuente: Autores
Elaboración: Autores

.

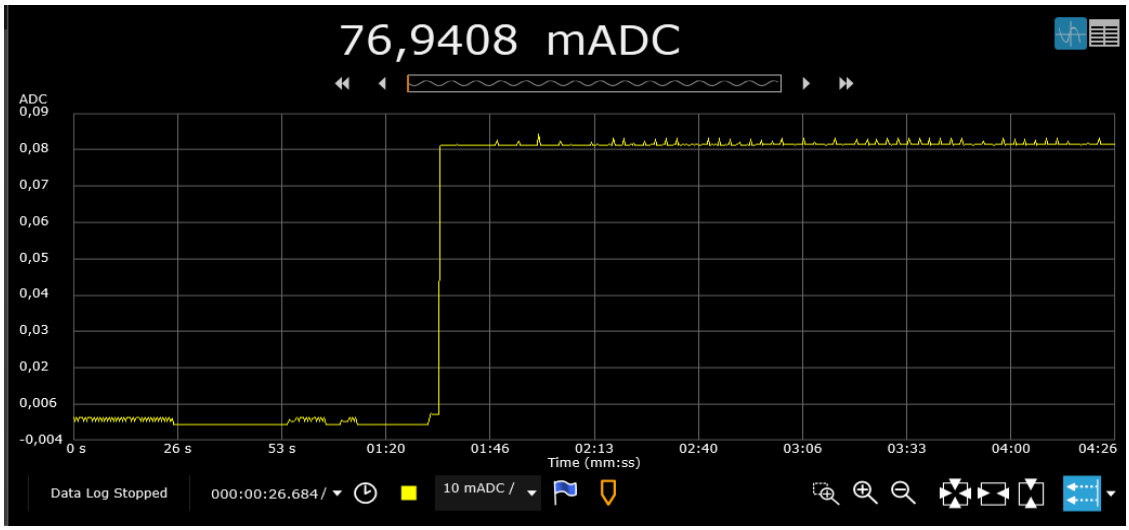


Figura. 4.10 Consumo del módulo transmisor XBee PRO 900.
Fuente: Autores
Elaboración: Autores

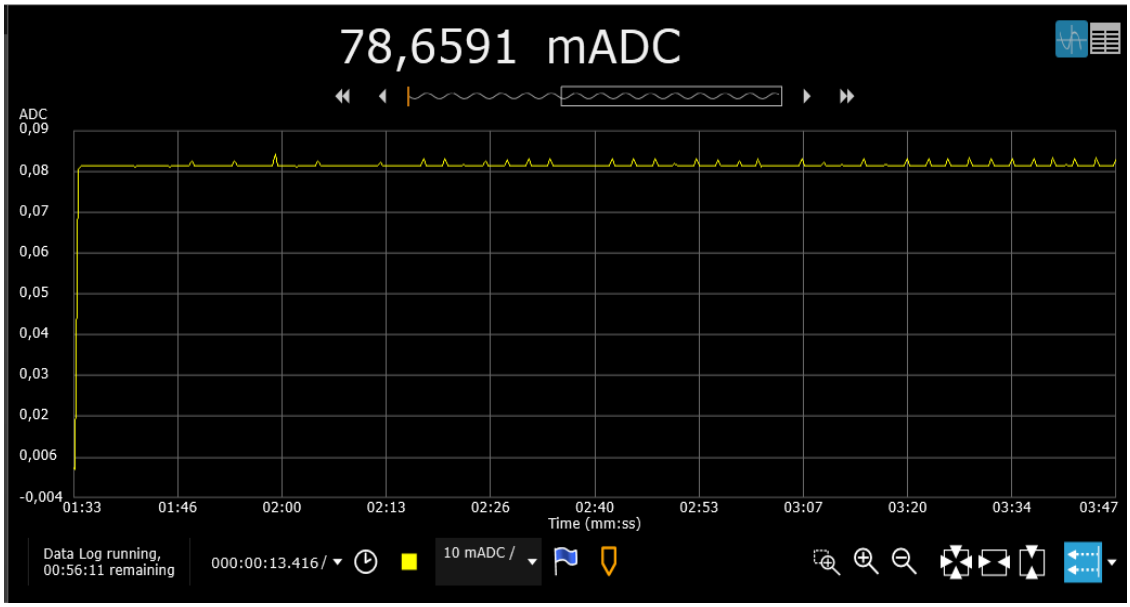


Figura. 4.11 Consumo del módulo receptor XBee PRO 900.
Fuente: Autores
Elaboración: Autores

Tabla 4.2 Consumo de corriente de módulos de comunicación.

ESTADO		ACCIÓN		MÓDULO
ON	56,68 mA	TRANSMITIENDO	76,94 mA	XBee PRO 900
OFF	0 µA	RECIBIENDO + ATDB	78,65 mA	
ON	78,55 mA	CONECTANDO	121,10 mA	GPRS Bee
OFF	0 µA	ENVIANDO	146,25 mA	

Fuente: Autores
Elaboración: Autores

.

En la tabla 4.2 se detalla el consumo energético de los módulos de comunicación, para estas mediciones se tomó en cuenta los módulos inalámbricos más utilizados en las aplicaciones locales.

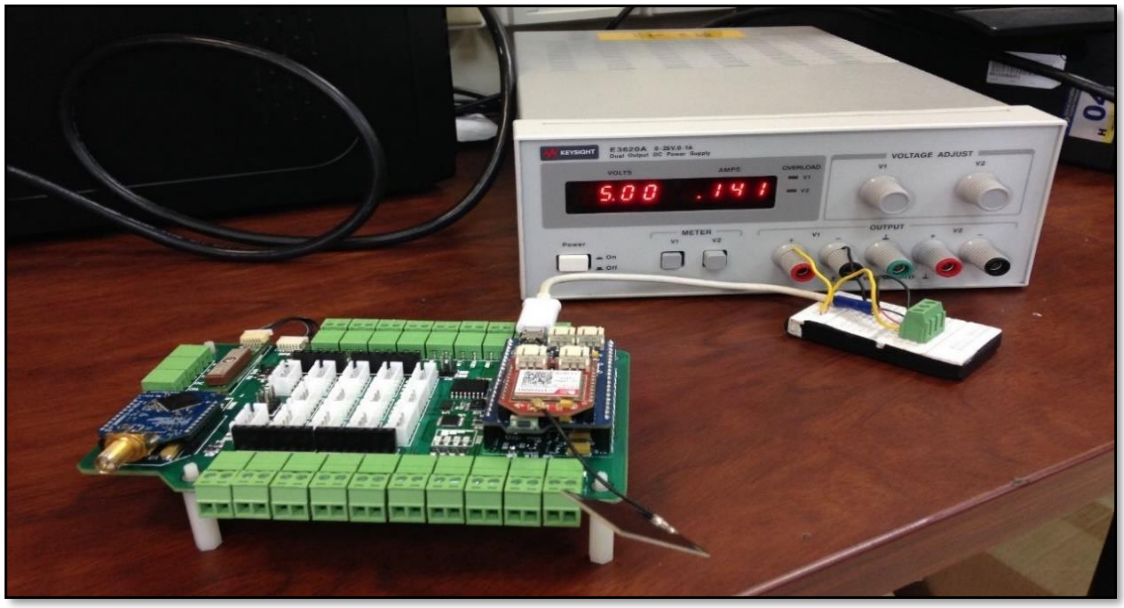


Figura. 4.12 Prueba de consumo con módulo GPRS Bee.
Fuente: Autores
Elaboración: Autores

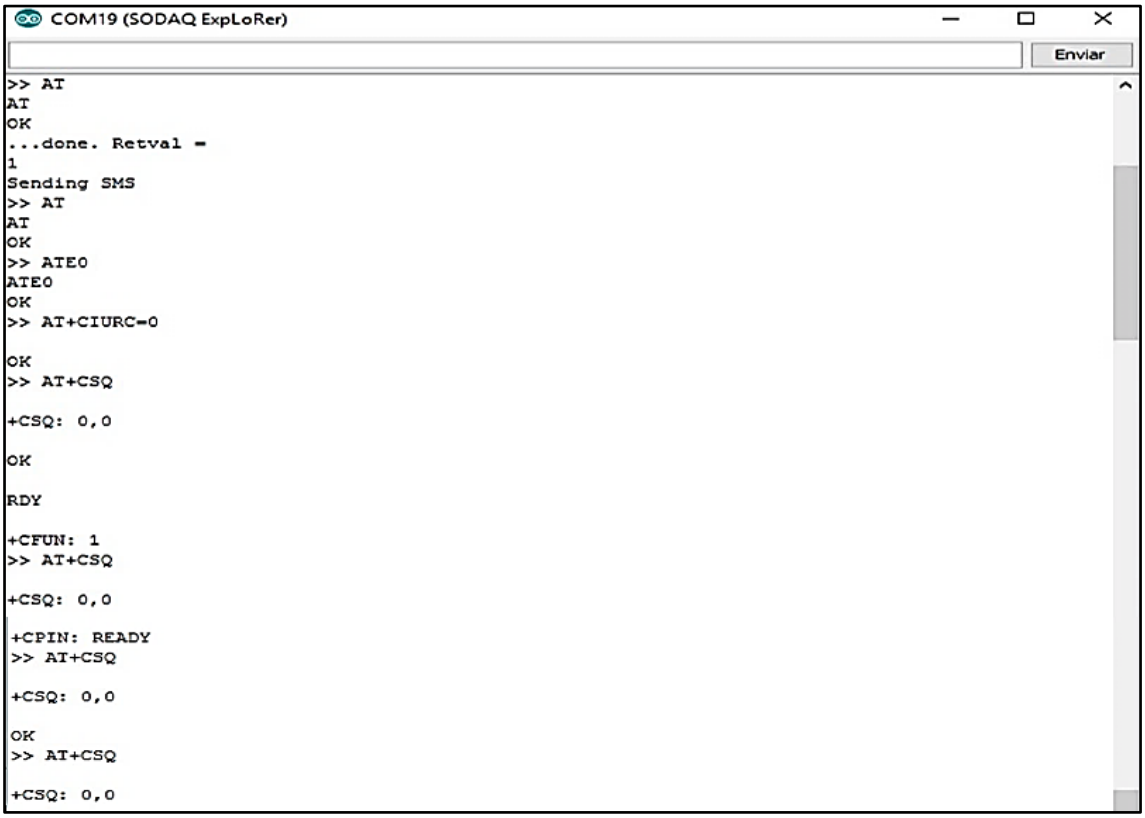


Figura. 4.13 Monitor serial de Arduino - Comandos de conexión a la red GSM/GPRS.
Fuente: Autores
Elaboración: Autores

.

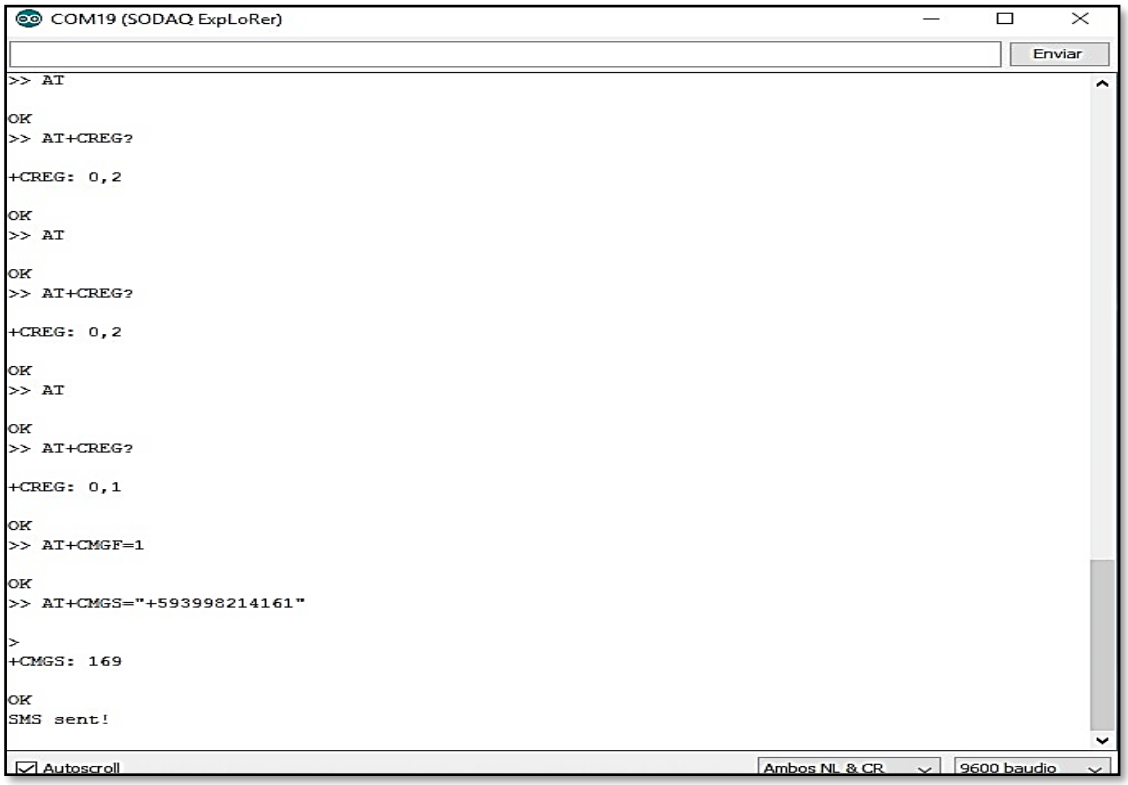


Figura. 4.14 Monitor serial Arduino - Mensaje enviado con éxito
Fuente: Autores
Elaboración: Autores

En las figuras 4.13 y 4.14 se observa el resultado del proceso de asociación y transmisión de un mensaje de texto con el módulo GPRS Bee. El objetivo es realizar mediciones de consumo de corriente en tiempo real en las distintas fases de la comunicación (envío- recepción).

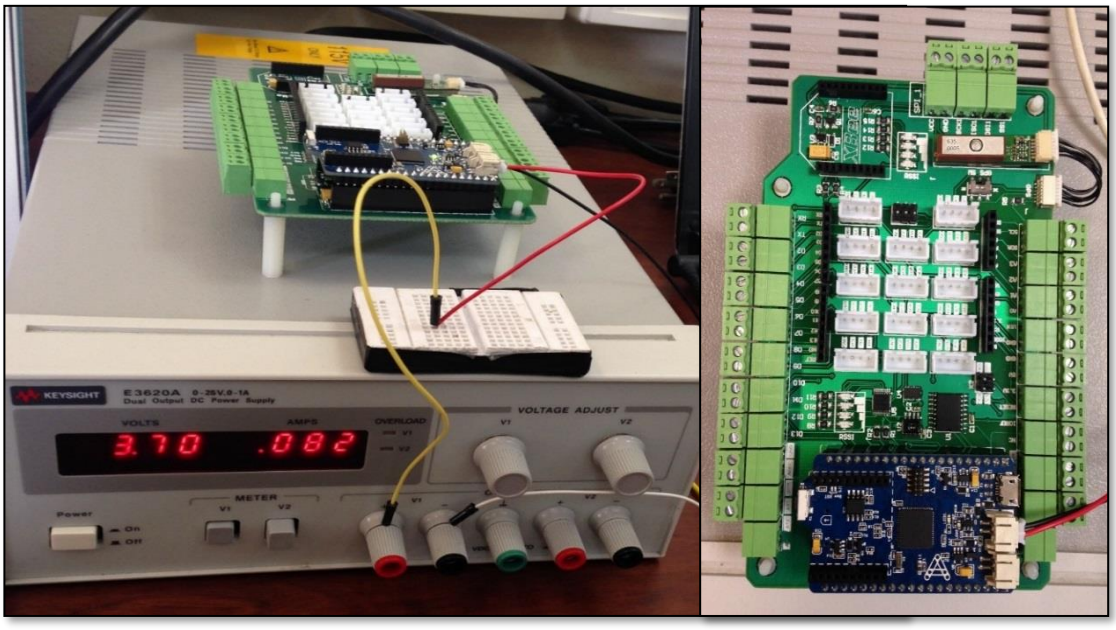


Figura. 4.15 Indicador de consumo con módulo GPS.
Fuente: Autores
Elaboración: Autores

.

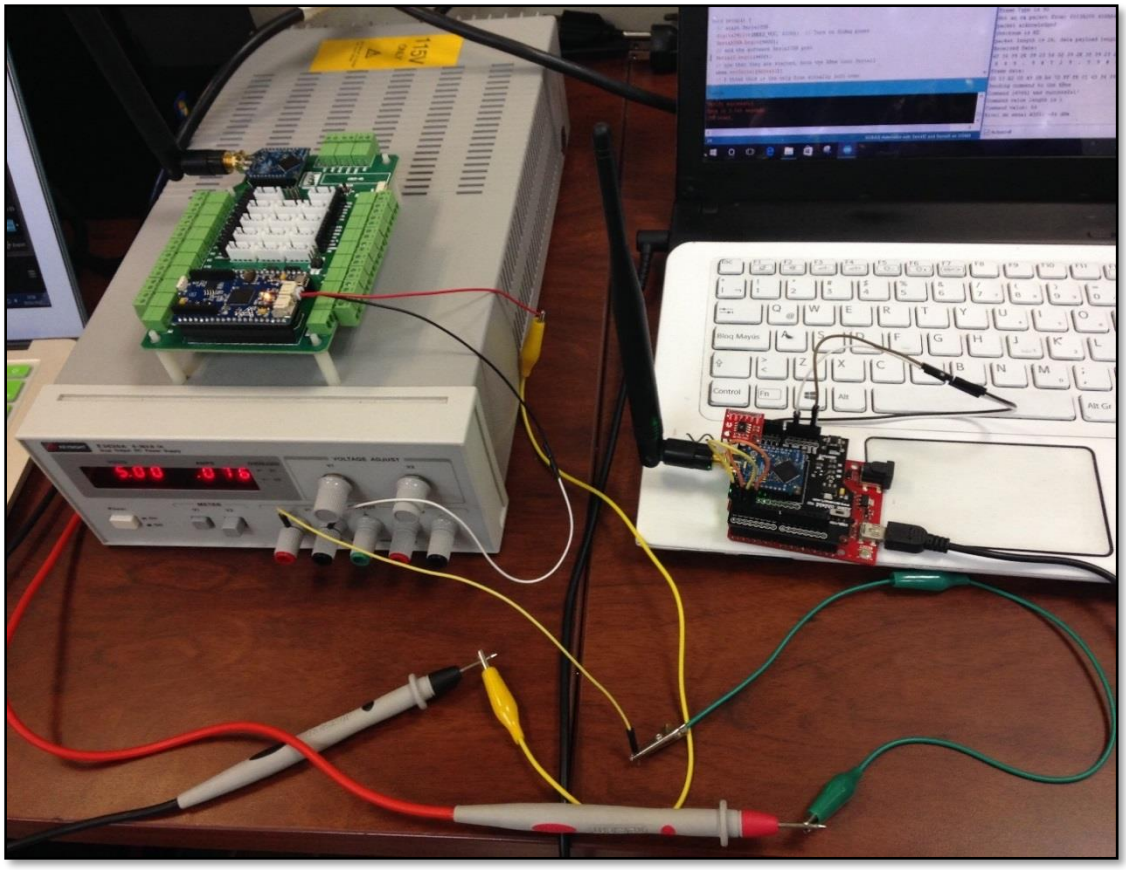


Figura. 4.16 Prueba de consumo con módulos XBee PRO 900.
Fuente: Autores
Elaboración: Autores

4.3 Prueba de sensores incorporados, GPS y RTC

En la figura 4.17, se muestra a detalle el resultado de escanear dispositivos conectados al bus I2C, se encontró cinco dispositivos esclavos.

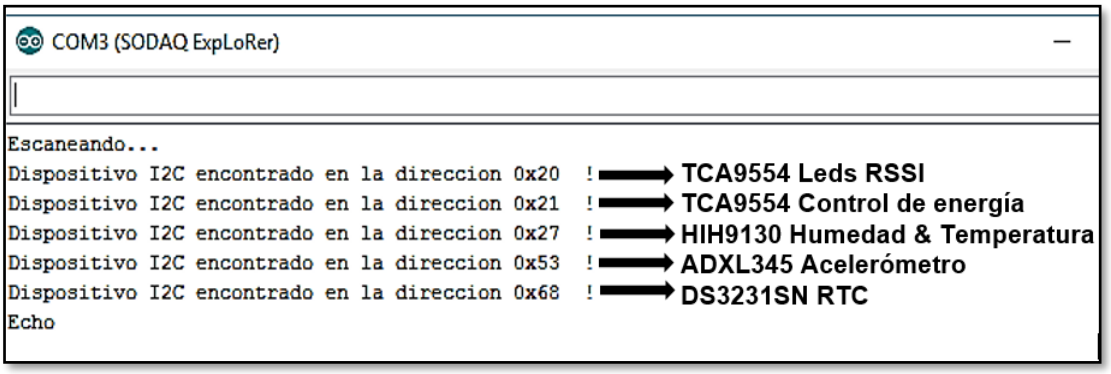


Figura. 4.17 Escáner de dispositivos I2C.
Fuente: Autores
Elaboración: Autores

4.3.1 Acelerómetro ADXL345

El sensor ADXL345 provee de algunas funciones de detección especial, que informan el estado de la mota. Estas funciones sirven para la presencia o ausencia de

.

movimiento. Adicionalmente, es capaz de percibir golpes leves y detectar si la mota experimenta una caída. Todas estas funciones fueron testeadas tal y como se puede observar en la figura 4.18. Véase el código en el Anexo 2.

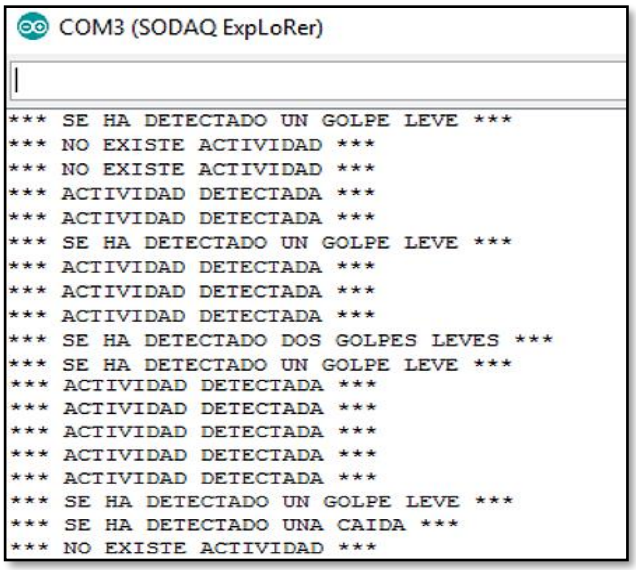


Figura. 4.18 Funciones especializadas del acelerómetro.
Fuente: Autores
Elaboración: Autores

4.3.2 GPS 635T

El GPS conectado al socket JST-SH, a través del interfaz Serial3, se configuró para recolección de datos a cada segundo, con permanencia de 60 segundos encendido y 30 segundos apagado, usando el *pin* PWR_CTRL de control de energía. El código funcional se encuentra en el Anexo 3.

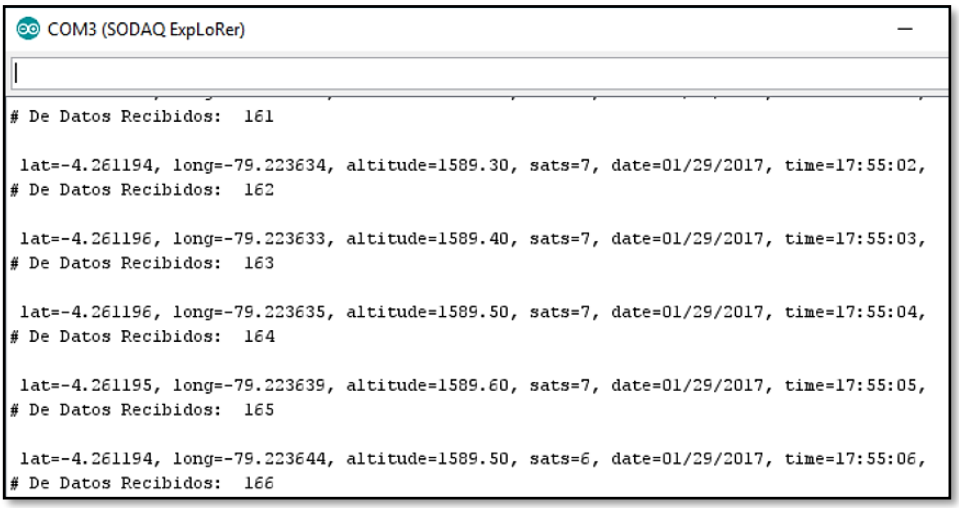


Figura. 4.19 Datos geográficos adquiridos.
Fuente: Autores
Elaboración: Autores

.

4.3.3 RTC DS321SN

Los datos de tiempo proporcionados por el RTC son información de vital importancia cuando la red de sensores pierde la conexión a internet. Las variables medidas pueden ser recolectadas y respaldadas en la tarjeta externa SD con la fecha y hora de captura para posteriormente ser posteadas al servidor cuando la conexión a internet se restablezca. Otro de los usos que tiene se da al RTC son funciones programadas específicas como dormir y despertar la mota para ahorro de energía. El código utilizado para obtener los resultados de la imagen 4.20 se encuentra en el Anexo 4.

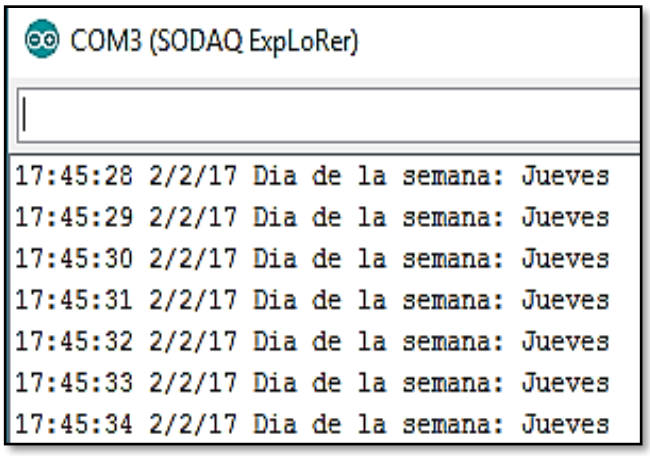


Figura. 4.20 Datos de tiempo obtenidos por el RTC.
Fuente: Autores
Elaboración: Autores

4.4 Pruebas de transmisión-recepción de datos

Se realizó pruebas con los indicadores de RSSI de la plataforma. Los leds están programados para encenderse de acuedo al valor en dBm del último paquete recibido. La escala valorativa se la indica a detalle en el apartado 3.8.7. Durante el análisis de comunicación se utilizó dos módulos XBee PRO 900 (1. Router - 2. Dispositivo final) ubicados a diferentes distancias, para estimar las variaciones de potencia de señal recibida.

Los valores obtenidos son observados mediante el puerto serial del IDE Arduino como se puede indica en la figura 4.21. El código utilizado para la recepción se encuentra en el Anexo 5.

.

```
Tipo de trama (Api ID):  90
Se ha recibido un paquete en el Xbee: 0013A200 408BA47D (FFFE)
Paquete reconocido ACK
Checksum es 5C
La longitud del paquete es 29, longitud de carga util (payload) es 17
Datos recibidos:
50 52 55 45 42 20 52 53 53 49 20 58 42 45 45 20 31
 P  R  U  E  B     R  S  S  I     X  B  E  E     1
Datos de la trama:
00 13 A2 00 40 8B A4 7D FF FE 01 50 52 55 45 42 20 52 53 53 49 20 58 42 45 45 20 31
Enviando comando AT al XBee
Comando [6866] Respuesta Exitosa!
La longitud del valor del comando es: 1
Valor del comando: 84
Nivel de senal RSSI: -84 dBm
Nivel de RSSI XBEE1 entre: -84 a -89
NO SE HA RECIBIDO PAQUETES
NO SE HA RECIBIDO PAQUETES
NO SE HA RECIBIDO PAQUETES
NO SE HA RECIBIDO PAQUETES
NO SE HA RECIBIDO PAQUETES
NO SE HA RECIBIDO PAQUETES
NO SE HA RECIBIDO PAQUETES
NO SE HA RECIBIDO PAQUETES
NO SE HA RECIBIDO PAQUETES
NO SE HA RECIBIDO PAQUETES
NO SE HA RECIBIDO PAQUETES
SE HA PERDIDO LA CONEXION
```

Figura. 4.21 Visualización de los valores de RSSI mediante IDE Arduino.
Fuente: Autores
Elaboración: Autores

En la figura 4.22, los indicadores led se encuentran apagados por la ausencia de paquetes en la recepción.

Figura. 4.22 Inicialización de envío de paquetes.
Fuente: Autores
Elaboración: Autores

Las figuras 4.23 a 4.26 indican la variación de los 4 niveles de RSSI obtenidos durante el análisis de comunicación.

.

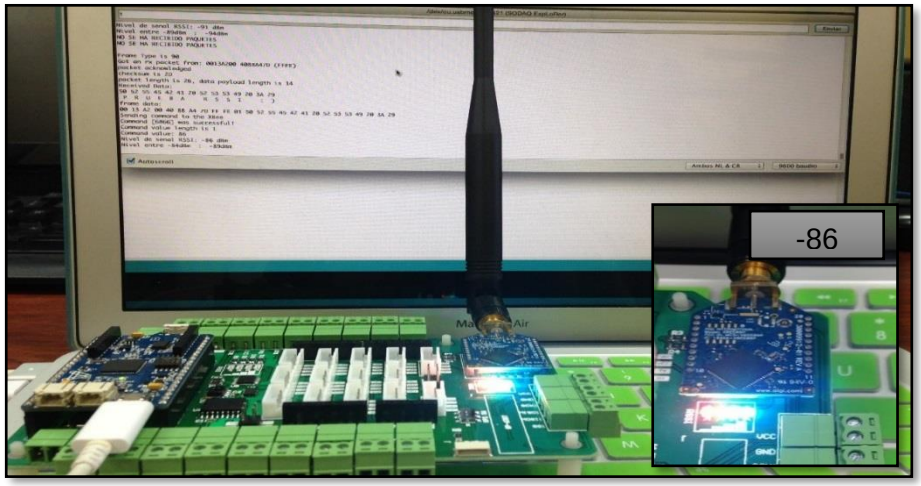


Figura. 4.23 Recepción de paquetes a -86 dBm.
Fuente: Autores.
Elaboración: Autores.

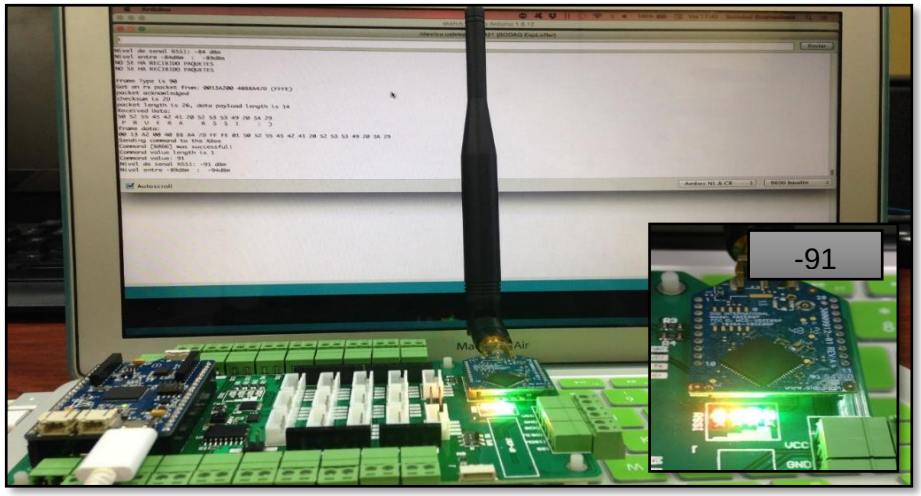


Figura. 4.24 Recepción de paquetes a -91 dBm.
Fuente: Autores.
Elaboración: Autores.

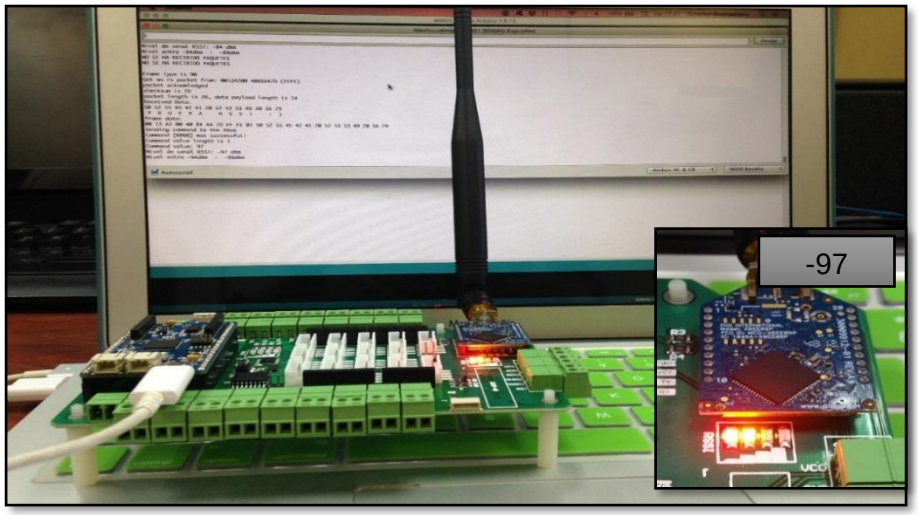


Figura. 4.25 Recepción de paquetes a -97 dBm.
Fuente: Autores.
Elaboración: Autores.

.

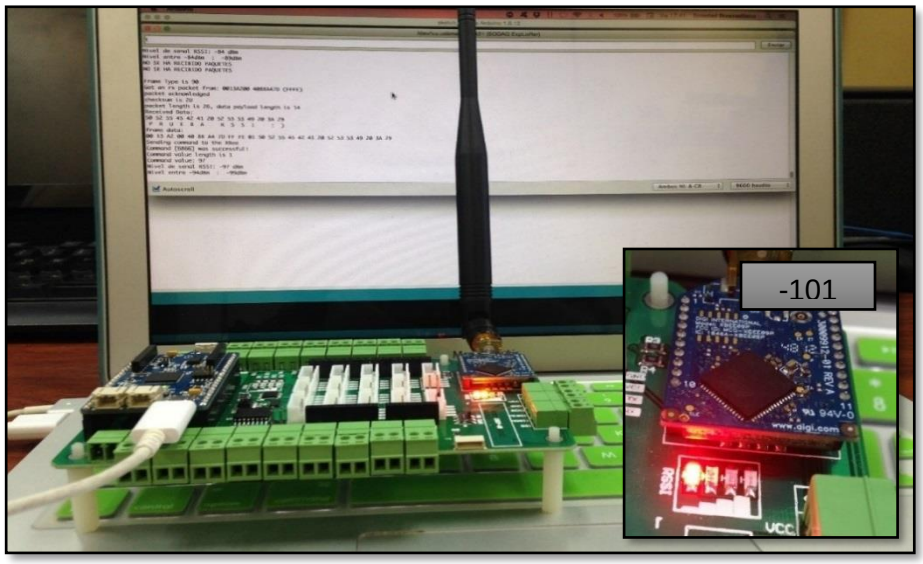


Figura. 4.26 Recepción de paquetes a -101 dBm.
Fuente: Autores
Elaboración: Autores

4.5 Prueba de control de energía

Para verificar el funcionamiento del bloque de control de energía de la plataforma se utilizó una placa auxiliar con tres indicadores leds, que identifican a los tres tipos sensores que pueden ser conectados a la placa como son: digitales, I2C y analógicos. Los conectores disponibles para los tres tipos de sensores están conectados a tres líneas de control del TCA9554 respectivamente. Estas líneas de control son activadas mediante codificación, ya sea para mantenerlas activas o apagadas, según requerimientos de usuario. En la figura 4.27 se observa todas las salidas en bajo (apagadas), indicando consumo nulo en los puertos. El código para la función de control de energía se encuentra en el Anexo 6.

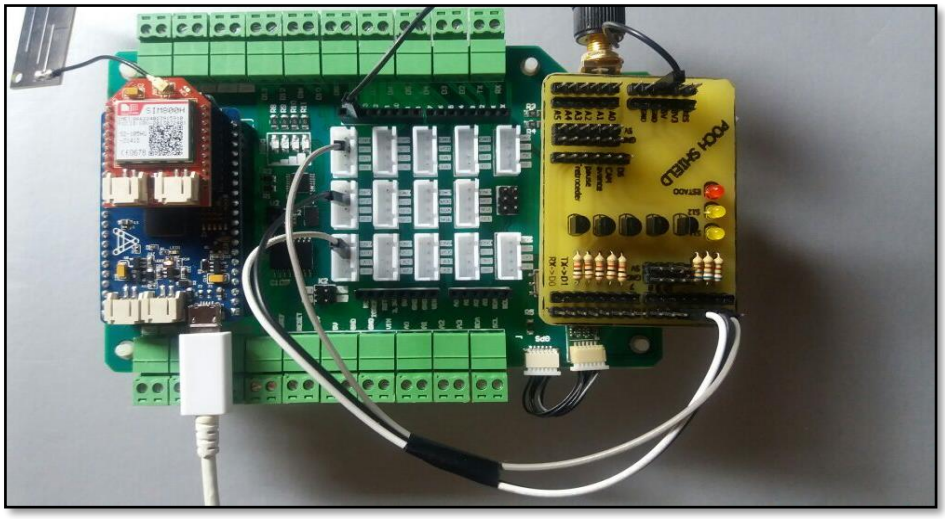


Figura. 4.27 Líneas de control de energía de sensores externos inactivas.
Fuente: Autores
Elaboración: Autores

.

Asimismo, en las figuras 4.28, 4.29 y 4.30 se indica la activación de las líneas de control de energía simultáneamente.

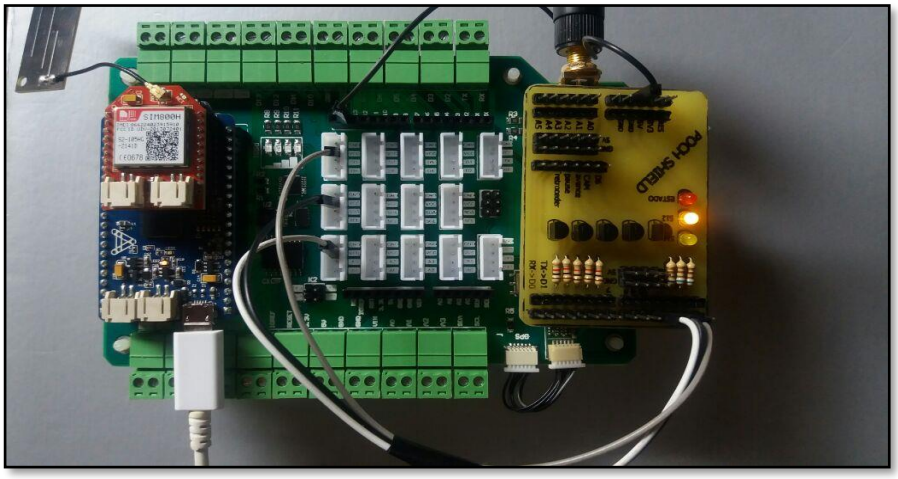


Figura. 4.28 Línea de control de alimentación para sensores I2C activa.
Fuente: Autores
Elaboración: Autores

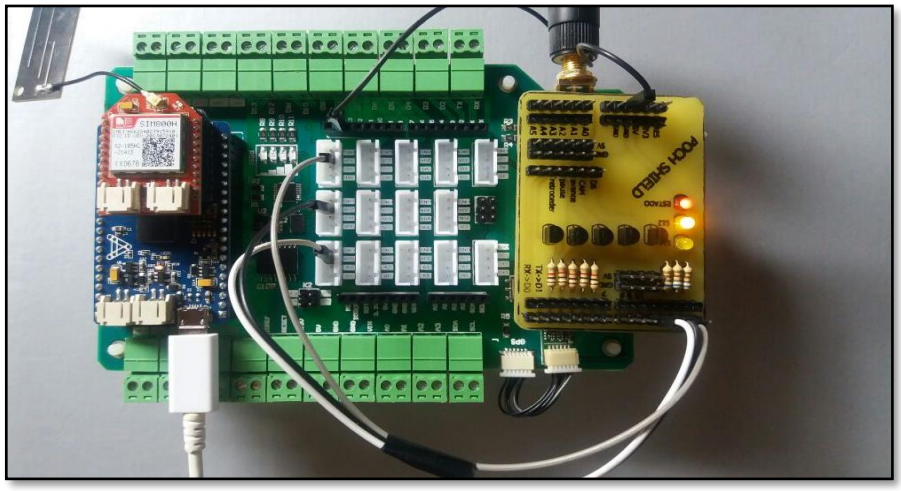


Figura. 4.29 Líneas de control para sensores digitales e I2C, activas.
Fuente: Autores
Elaboración: Autores

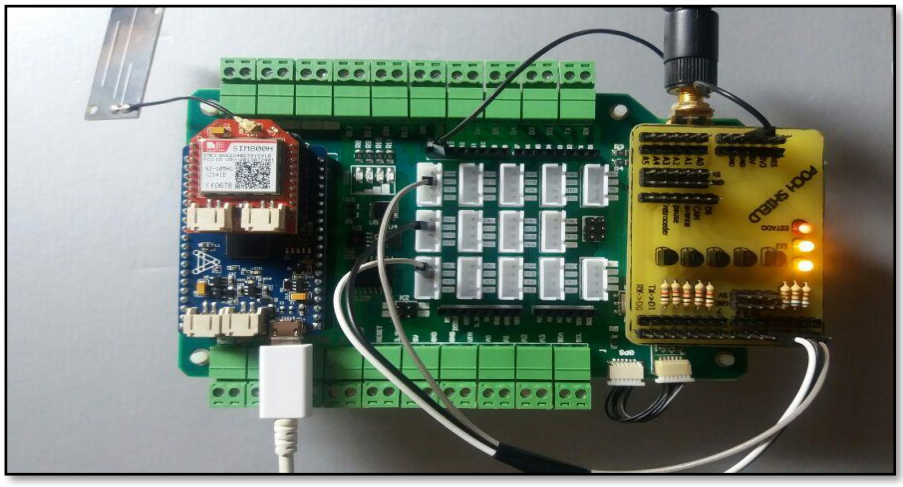


Figura. 4.30 Líneas de control activas para todos los sensores.
Fuente: Autores
Elaboración: Autores

.

4.6 Montaje de la plataforma

En esta sección se indica cómo se debe montar la plataforma y la disponibilidad de los conectores para adaptar sensores a la mota. En la figura 4.31, se puede apreciar la adaptación de conectores IP68 y conectores RPSMA a la caja, mientras que en las figuras 4.32 y 4.33, se observar el resultado obtenido.

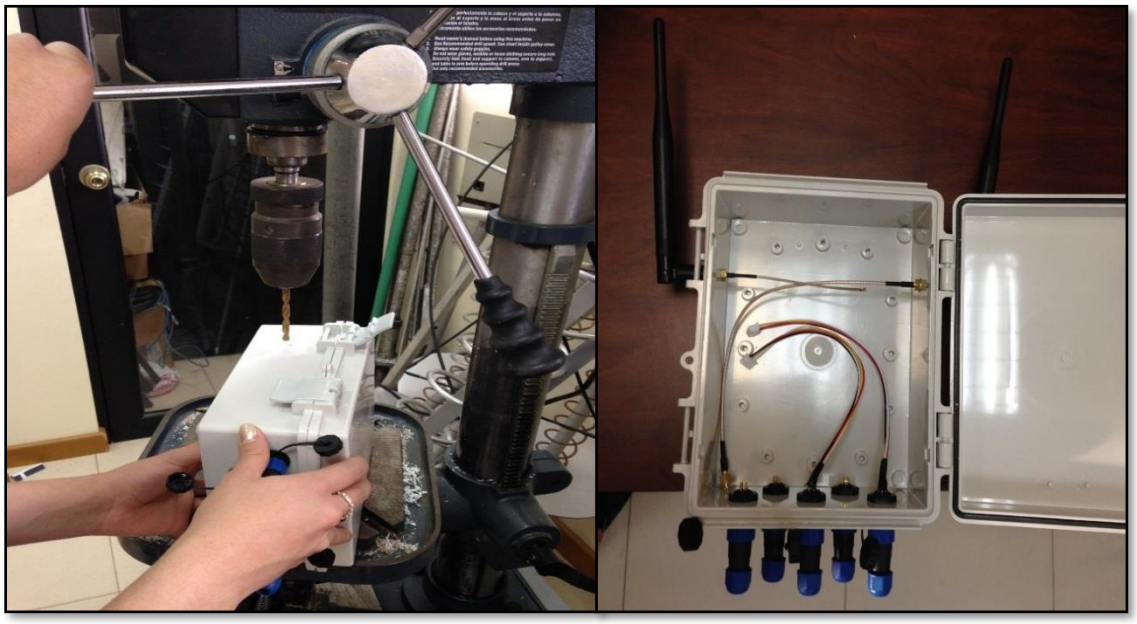


Figura. 4.31 Adaptación de conectores IP68 y RPSMA.
Fuente: Autores
Elaboración: Autores

Figura. 4.32 Caja de protección de la plataforma.
Fuente: Autores
Elaboración: Autores

.

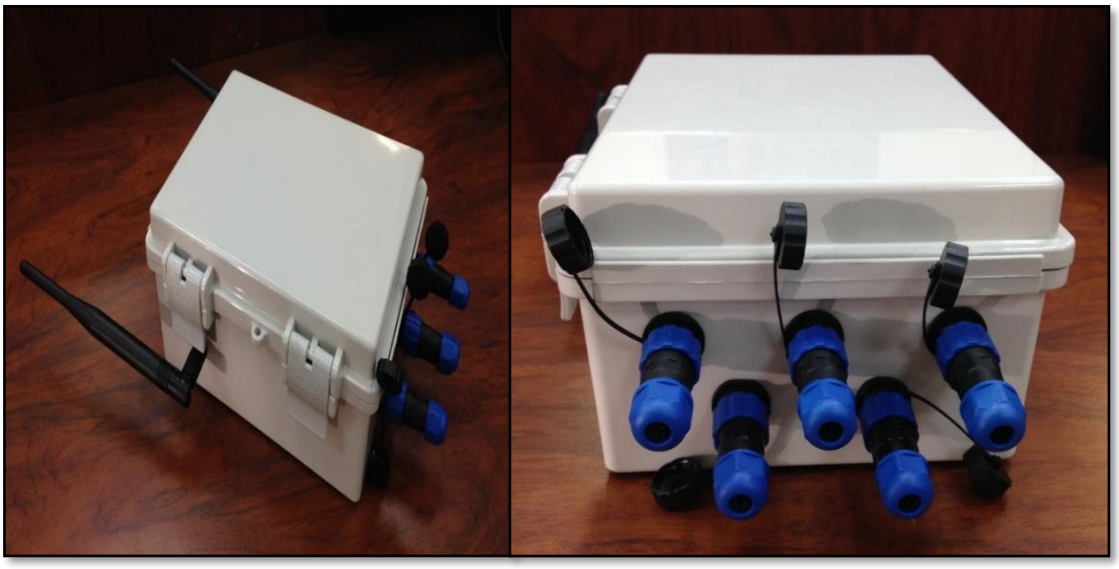


Figura. 4.33 Vista exterior de la mota.
Fuente: Autores
Elaboración: Autores

En los conectores de 3 y 4 puertos instalados exclusivamente para sensores, se acopló una sonda de humedad y temperatura para suelo con salida digital en una configuración de 4 puertos como se puede apreciar en la figura 4.34.

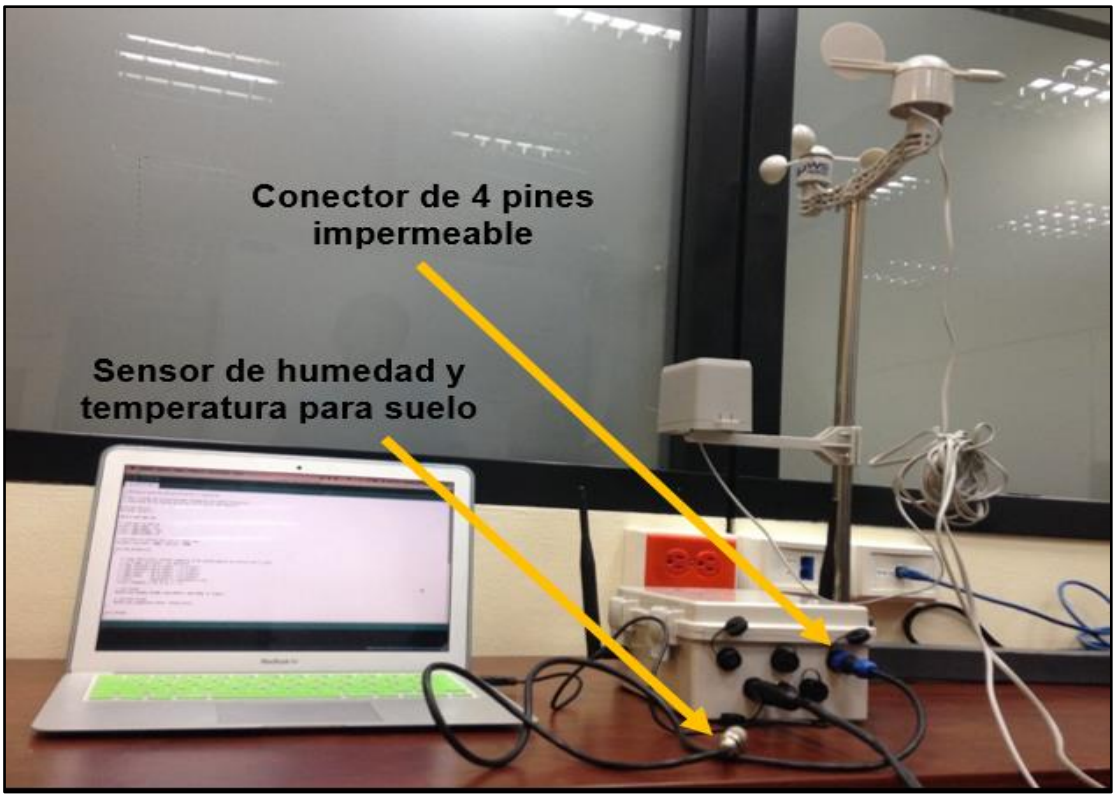


Figura. 4.34 Conexión mediante puerto USB a conector IP68 micro USB de la mota.
Fuente: Autores
Elaboración: Autores

.

Para facilitar el acceso al mantenimiento o cambio en las rutinas de programación de la mota, se adecuó un conector nano USB impermeable a la caja; la figura 4.35 indica la ubicación del conector.

Figura. 4.35 Vista del conector nano USB.
Fuente: Autores
Elaboración: Autores

Posteriormente se representó el montaje de una estación meteorológica básica la cual consta de sensores de temperatura, humedad, velocidad y dirección de viento. En cuanto a la alimentación consta de un panel solar acoplado a un conector IP68 de 2 puertos que se conecta al SODAQ Autonomo. Además, posee dos tecnologías de comunicación inalámbrica, un módulo RF para la transferencia de datos entre motas y un módulo GRPS como auxiliar para subir datos a la nube cuando la estación base pierde conexión a internet. En las figura 4.36 y 4.3 se detalla los elementos utilizados en la representación de la estación.

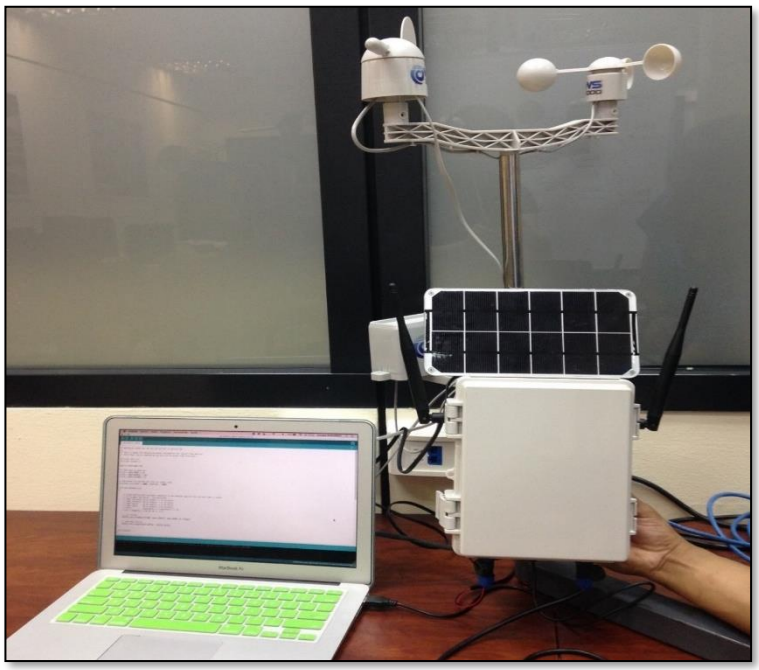


Figura. 4.36 Vista de la conexión interior de la mota.
Fuente: Autores
Elaboración: Autores

.

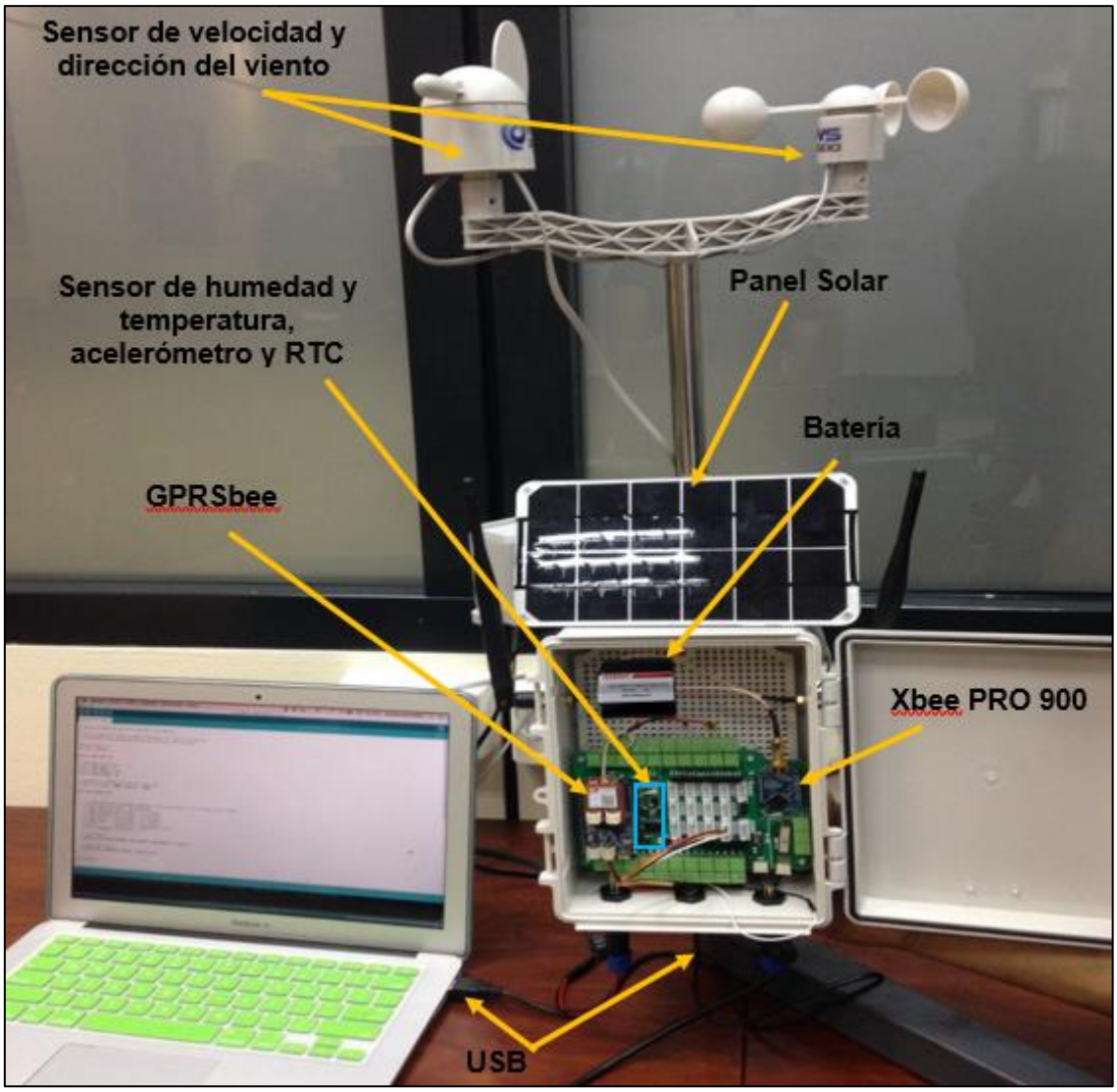


Figura. 4.37 Vista de la conexión exterior de la mota.
Fuente: Autores
Elaboración: Autores

4.7 Presupuesto referencial

En la Tabla 4.3 se indica un presupuesto referencial de los componentes empleados en el desarrollo de la plataforma para WSNs e IoT. Todos los dispositivos detallados en la tabla siguiente son elementos de hardware utilizados dentro del bloque de control de energía, comunicaciones y bloque de sensores. Puesto que la placa diseñada es de arquitectura modular, se tomó en cuenta solo el precio de los componentes de acople; más no el costo de los módulos de integración, como la tarjeta computacional y los módulos de comunicación inalámbrica. Es importante señalar que el costo de manufactura es el más alto, pero en el caso de reproducir el prototipo en grandes cantidades, el costo de venta disminuiría a 86,61 USD.

.

Tabla 4.3. Tabla referencial de precios de componentes y manufactura.

Ítem	Descripción	Cantidad	Precio Unitario	Precio Final
XC6220B331MR-G	Regulador de voltaje LDO 1A - Green Operation	5	$ 0,59	$ 2,95
TCA9554PWR	Expansor I2C de 16 pins	2	$ 2,13	$ 4,26
DPDT	Interruptor para montaje en PCB de 6 pins SMD	1	$ 0,12	$ 0,12
PRT-10210	Conector SMD JST SH Horizontal de 6 pins	1	$ 0,95	$ 0,95
MSS1P2L-M3/89A	Diodo y Rectificador Schottky 1.0 A-20 V	5	$ 0,26	$ 1,30
FDN340P	MOSFET SSOT-3 P-CH -20V	5	$ 0,21	$ 1,05
BHX1-1225-PC-ND	Socktet para batería 1225 CELL PCB	1	$ 0,34	$ 0,34
DS3231SN	Circuito integrado tipo reel RTC	1	$ 1,55	$ 1,55
ADXL345BCCZ-RL7	Sensor de aceleración de movimiento	1	$ 1,99	$ 1,99
HIH9130-021-001	Sensor de temperatura y humedad	1	$ 11,28	$ 11,28
15EDGK	conector enchufable de 3.81mm Pitch 2 pins	23	$ 0,15	$ 3,45
0603 SMD	Capacitores cerámicos	8	$ 0,01	$ 0,08
6032 Tipo C 100 UF	Capacitores de tantalio	5	$ 0,18	$ 0,90
1206 SMD	Resistores	23	$ 0,01	$ 0,23
0603 (1608) SMT SMD	Chips leds	6	$ 0,03	$ 0,18
JST-4 pins	Conector JST de 4 pins	14	$ 0,44	$ 6,16
Headers pitch 2.0	Regletas para socket Bee	1	$ 0,15	$ 0,15
Headers pitch 2.4	Regletas para pins I/O	1	$ 0,15	$ 0,15
Manufactura	Panel FE1 DS+SM	1	$ 26,44	$ 26,44
Manufactura	Tooling plotter DS+SM	1	$ 14,00	$ 14,00
Manufactura	Soldado	1	$ 13,18	$ 13,18
Manufactura	CM fresado	2	$ 0,04	$ 0,08
Manufactura	Screen	1	$ 7,00	$ 7,00
			TOTAL	$ 97,79

Elaboración: Autores
Fuente: Autores

.

Tabla 4.4. Tabla de precios de equipamiento externo de la mota.

Cantidad	Descripción	Precio Unitario	Precio Final
2	Conector impermeable IP68 para cable de 5 puertos - 250 V	$ 4,29	$ 8,58
2	Conector impermeable IP68 para cable de 4 puertos - 250 V	$ 4,01	$ 8,02
2	Conector IP68 impermeable 3-puertos (hembra) enchufe (macho) - 5A	$ 3,26	$ 6,52
2	Conector impermeable IP68 de 2 puertos conector a prueba de agua y polvo	$ 3,84	$ 7,68
1	Conector mini USB impermeable IP67	$ 13,16	$ 13,16
1	Caja IP65	$ 20,00	$ 20,00
TOTAL			$ 63,96

Elaboración: Autores
Fuente: Autores

El presupuesto detallado en la tabla 4.4 referencia el costo de implementación básico de la plataforma de la mota. El precio total especificado en la tabla anterior, varía de acuerdo a la aplicación del usuario. El precio total de implementación es de 161.75 USD.

.

CONCLUSIONES

- Al finalizar el presente trabajo, se cuenta con un prototipo de mayor procesamiento y menor consumo energético, en comparación con la plataforma de Libelium Waspmote. El prototipo puede ser implementado en aplicaciones industriales de WSNs e IoT, gracias a que posee dos sockets XBee que permiten adecuar modelos híbridos de tecnología inalámbrica. Además, la plataforma puede ser adaptada en escenarios de ambientes climáticos adversos, ya que cuenta con equipamiento de protección IP.

- Los resultados obtenidos tras la validación de consumo de corriente son satisfactorios, puesto que el gasto operacional del prototipo de forma individual es de ~9mA, mientras que al integrar módulos de comunicación el rango varía desde los 77mA a 146mA. Estos rangos de consumo de corriente son menores, incluso en comparación con plataformas de hardware reducido.

- El control de abastecimiento de energía para sensores acoplados de interfaz digital, analógica e I2C aportan al ahorro de energía gracias a que el usuario, mediante programación de la mota, puede determinar los intervalos de tiempo necesarios para la recolección de datos. Logrando así, maximizar la utilización de la energía por medio de la selección de un núcleo de procesamiento de última generación y la gestión de control de energía en el bloque de comunicación.

- El diseño de la plataforma permite combinar tecnologías inalámbricas debido a la variedad de módulos en formato XBee existentes. El usuario fácilmente puede adaptar dos módulos inalámbricos de acuerdo a las necesidades de su aplicación. De igual forma, al implementar protocolos de comunicación que han sido ampliamente utilizados en WSNs, se proporciona confiabilidad y portabilidad en la tecnología utilizada para la transferencia de la información.

- La integración de los indicadores de nivel de señal en cada socket XBee, favorecen al despliegue de la red, ya que informan sobre el nivel de intensidad de la señal recibida.

- La plataforma puede desempeñar funciones de nodo coordinador, nodo gateway o nodo sensor (captura de datos). Y también puede ser usada como estación

.

base por su característica modular que permite integrar una *Ethernet Shield* para acceder físicamente a la red y subir los datos almacenados a la nube que están disponibles en la web.

- El prototipo incorpora tres tipos de sensores que adquieren variables de temperatura, humedad y movimiento, dando al usuario la facultad de obtener información del estado del sistema de la mota para optimizar su funcionamiento.

- Para la implementación del prototipo en ambientes externos u hostiles, se protege el hardware con una caja de grado de protección IP65 y conectores impermeables IP68 con diferente número de puertos que posibilita el montaje y desmontaje sensores a la mota con facilidad. Además cuenta con conectores industriales *plug & play* que aseguran la conexión de sensores a la placa.

- Finalmente, al desarrollar una plataforma de hardware abierto y software libre, se promueve la investigación y desarrollo de prototipos de innovación que requieran viabilidad en costo y versatilidad en aplicaciones WSN e IoT.

.

RECOMENDACIONES

- Para trabajos futuros se recomienda integrar la unidad de procesamiento directamente en la placa, con la finalidad de producir prototipos de hardware reducido. Conforme a la investigación realizada se cree factible el uso de microcontroladores de la familia ATSAM-D (Atmel), por sus altas características de procesamiento y ultra bajo consumo energético.

- Se ha identificado un *bug de hardware* en la conexión del bus I2C, por ende, la solución a este problema es mantener la conexión CE-VCC del *jumper* SJ10. Su desconexión causa que el circuito integrado TCA9554 encargado del control de energía y salida de puertos digitales adicionales no funcione adecuadamente.

- Se recomienda usar un panel solar con voltaje de salida de 4.5 a 6V para que el circuito de carga de batería del SODAQ Autonomo entre en operación.

.

REFERENCIAS

[1] M. T. Lazarescu, “Design of a WSN platform for long-term environmental monitoring for IoT applications,” *IEEE J. Emerg. Sel. Top. Circuits Syst.*, vol. 3, no. 1, pp. 45–54, 2013.

[2] T. Watteyne, “OpenWSN,” *Wireless Sensor Network*, 2016. [Online]. Disponible en: https://openwsn.atlassian.net/wiki/.

[3] T. Chang, P. Tuset-Peiro, X. Vilajosana, and T. Watteyne, “OpenWSN & OpenMote: Demo’ing A Complete Ecosystem for the Industrial Internet of Things,” pp. 0–2, 2016.

[4] I. Coop, “Una Metodología Para El Desarrollo De Hardware Y Software Embebidos En Sistemas Críticos De Seguridad,” *Iiisci.Org*, pp. 70–75, 2006.

[5] U. de C. Fernandez Barcell, Manuel, “Introducción a las redes de sensores inalámbricas,” *Wirel. Sens. Netw.*, pp. 1–20, 2008.

[6] A. Tufail, A. Qamar, A. M. Khan, W. A. Baig, and K. H. Kim, “WEAMR - A weighted energy aware multipath reliable routing mechanism for hotline-based WSNs,” *Sensors (Switzerland)*, vol. 13, no. 5, pp. 6295–6318, 2013.

[7] A. Jamalipour and J. Zheng, *Wireless Sensor Networks: A Networking Perspective*. 2009.

[8] MEMSIC, “MTS/MDA,” 6020–0047–04 Rev A.

[9] Libelium, “Agriculture 2.0.”

[10] A. Potenciales and R. Wsn, “¿ Qué es una Red de Sensores Inalámbricos ?,” *Natl. Instruments*, pp. 3–4, 2009.

[11] Ubidots, “Dragino IoT Gateway.” [Online]. Disponible en: https://ubidots.com/docs/devices/Dragino.html#requirements. [Accedido: 10-Sep-2016].

[12] S. E. Díaz, “Wireless sensor networks: fundamentos, estado del arte y desafíos,” 2011.

[13] I. Autor, G. Fortu, and V. Villarino, “Desarrollo e implementación de una red de sensores Zigbee mediante el dispositivo Xbee de Digi,” 2012.

[14] F. P. Roque, E. V. Zaldívar, and O. A. De Fuentes, “Sistema de adquisición de datos con comunicación inalámbrica,” *Rev. Ingeriría Electrónica, Automática y Comun.*, pp. 63–73, 2013.

[15] J. Portilla *et al.*, “Adaptable security in wireless sensor networks by using reconfigurable ECC hardware coprocessors,” *Int. J. Distrib. Sens. Networks*, vol. 2010, 2010.

[16] C. Hacks, “Waspmote - Wireless Sensor Networks Open Source Platform.”

.

[Online]. Disponible en: https://www.cooking-hacks.com/documentation/tutorials/waspmote/#hardware. [Accedido: 11-Sep-2016].

[17] Moteiv, “Telos Manual,” 2004.

[18] E. Micro, “MOJO IDE.” [Online]. Disponible en: https://embeddedmicro.com/blog/introducing-wave-capture. [Accedido: 11-Sep-2016].

[19] P. Levis *et al.*, “TinyOS: An Operating System for Wireless Sensor Networks,” *Ambient Intell.*, pp. 115–148, 2005.

[20] Thingsquare, “Contiki: The Open Source OS for the Internet of Things.” [Online]. Disponible en: http://www.contiki-os.org. [Accedido: 11-Sep-2016].

[21] ARDUINO, “Arduino Software.” [Online]. Disponible en: https://www.arduino.cc/en/main/software. [Accedido: 11-Oct-2016].

[22] eCos, “eCos.” [Online]. Disponible en: http://ecos.sourceware.org. [Accedido: 11-Sep-2016].

[23] RIOT, “RIOT,” 2017. [Online]. Disponible en: http://riot-os.org/#home. [Accedido: 04-Jan-2017].

[24] J. Ranz Abad, “Things City,” 2013. [Online]. Disponible en: http://thingscity.com/definicion-de-internet-de-las-cosas/. [Accedido: 04-Jan-2017].

[25] Felipe Garrido, “El 2020 habrá 4 mil millones de conexiones a internet,” *El 2020 habrá 4 mil millones de conexiones a internet*, 2016. [Online]. Disponible en: https://www.fayerwayer.com/2016/06/el-2020-habra-4-mil-millones-de-conexiones-a-internet/.

[26] L. S. Emilio, “Diseño de un sistema de control domótico basado en la plataforma Arduino,” *Esc. Tècnica Eng. Inform. Univ. Politècnica València*, pp. 1–44, 2012.

[27] Meteoalforins, “Estación Meteorológica Davis Vantage Pro2 Plus Wireless.” [Online]. Disponible: http://www.meteolalfori.es/estacion/. [Accedido: 20-Oct-2016].

[28] Cetronic, “SENSOR DE HUMEDAD Y TEMPERATURA DE SUELO SHT11.” [Online]. Disponible en: http://www.cetronic.es/sqlcommerce/disenos/plantilla1/seccion/producto/DetalleProducto.jsp?idIdioma=&idTienda=93&codProducto=151346015&cPath=1343. [Accedido: 20-Oct-2016].

[29] Cetronic, “SENSOR DE HUMEDAD Y TEMPERATURA DE SUELO SHT11.” .

[30] Mbeddedadventures, “MS5611 24 bit Barometric Pressure Sensor.” [Online]. Disponible: https://www.embeddedadventures.com/ms5611.html. [Accedido: 16-

.

Nov-2016].

[31] Mbeddedadventures, “MS5611 24 bit Barometric Pressure Sensor.” .

[32] Edaphic, “Soil CO2 concentration.” [Online]. Disponible: http://www.edaphic.com.au/products/soils/forerunner-gp-sensor/. [Accedido: 10-Nov-2016].

[33] Edaphic, “Soil CO2 concentration.” .

[34] Directindustry, “Sensor de luz ambiental / fotodiodo.” [Online]. Disponible: http://www.directindustry.es/prod/fuehlersysteme-enet-international-gmbh/product-59132-778869.html. [Accedido: 17-Nov-2016].

[35] Directindustry, “Sensor de luz ambiental / fotodiodo.” .

[36] Vernier, “Vernier Arduino Interface Shield.” [Online]. Disponible: https://www.vernier.com/products/interfaces/bt-ard/. [Accedido: 05-Dec-2016].

[37] Vernier, “Vernier Arduino Interface Shield.” .

[38] Chinaultrasound, “50 KHz Ultrasonic Transducer – Long Distance Measurement 30cm-8meters – TA0050KA.” [Online]. Disponible: http://chinaultrasound.com/product/50-khz-ultrasonic-transducer-long-distance-measurement-30cm-8meters-ta0050ka/. [Accedido: 01-Dec-2016].

[39] Chinaultrasound, “50 KHz Ultrasonic Transducer – Long Distance Measurement 30cm-8meters – TA0050KA.” .

[40] A. Flammini and E. Sisinni, “Wireless sensor networking in the internet of things and cloud computing era,” *Procedia Eng.*, vol. 87, pp. 672–679, 2014.

[41] M. F. Vásconez Barrera, “Diseño de un sistema de control gerencial de plantas avícolas utilizando Redes de Sensores Inalámbricos con tecnología Open Hardware Departamento de Investigación y Postgrados,” Pontificia Universidad Católica del Ecuador, sede Ambato, 2015.

[42] makerzone mathworks, “Counting Cars and Analyzing Traffic with a Raspberry Pi, a Webcam and ThingSpeak.” [Online]. Disponible: http://makerzone.mathworks.com/blog/counting-cars-and-analyzing-traffic-raspberry-pi-thingspeak/. [Accedido: 10-Dec-2016].

[43] LinkLabs, “IoT Plataforms.” [Online]. Disponible en: https://www.link-labs.com/blog/what-is-an-iot-platform. [Accedido: 10-Oct-2016].

[44] programmableweb, “Ubidots API.” [Online]. Disponible en: https://www.programmableweb.com/api/ubidots. [Accedido: 13-Nov-2016].

[45] Ubidots, “Ubidots.” [Online]. Disponible en: https://ubidots.com/home2?utm_expid=45052721-8.B-TgEH7JQlCPW__l9crDnw.1. [Accedido: 24-Oct-2016].

[46] Mathworks, “ThingSpeak.” [Online]. Disponible en:

.

https://www.mathworks.com/help/thingspeak/. [Accedido: 15-Nov-2017].

[47] SODAQ, “Sodaq Support.” .

[48] Libelium, “Waspmote.” [Online]. Disponible en: http://www.libelium.com/products/waspmote/. [Accedido: 01-Jan-2017].

[49] Chip45, “iDwaRF-BOX V1.2 Radio Base Station.” [Online]. Disponible en: https://www.chip45.com/products/idwarf-box-1.2_wireless_sensor_network_base_station.php. [Accedido: 02-Jan-2017].

[50] Seeedstudio, “Seeed.” [Online]. Disponible en: https://www.seeedstudio.com. [Accedido: 20-Dec-2016].

[51] E. O. Laboratory, “Crossbow Datasheet.” [Online]. Disponible en: https://www.eol.ucar.edu/isf/facilities/isa/internal/CrossBow/DataSheets/. [Accedido: 10-Oct-2016].

[52] MEMSIC, “MEMSIC.” [Online]. Disponible en: http://www.memsic.com/wireless-sensor-networks/. [Accedido: 10-Oct-2016].

[53] Moteiv Corporation, “Moteiv: tmote sky low power wireless sensor module,” *Prod. Data Sheet*, pp. 1–28, 2006.

[54] Kickstarter, “Autonomo: The Solar-Powered Thing.” [Online]. Disponible en: https://www.kickstarter.com/projects/sodaq/autonomo-the-solar-powered-thing?lang=es. [Accedido: 02-Dec-2016].

[55] SODAQ, “Sodaq Support.” [Online]. Disponible en: http://support.sodaq.com/sodaq-one/autonomo/getting-started-autonomo/. [Accedido: 20-Oct-2016].

[56] Atmel, “Atmel SAM D21E / SAM D21G / SAM D21J.” Atmel Corporation, San Jose, USA, p. 1112, 2015.

[57] Atmel, “Atmel SAM D21E / SAM D21G / SAM D21J.” Atmel Corporation, San Jose, USA, p. 1112, 2015.

[58] T. Semicoductors, “XC6220 Series 1A LDO Voltage Regulator with ‘GreenOperation.’” Torex Semiconductor, pp. 1–32.

[59] T. Instrument, “TCA9554 Low Voltage 8-Bit I2C and SMBus Low Power I/O Expander With Interrupt Output adn Configuration Registers.” Texas Instrument, Dalas, Texas, p. 42, 2015.

[60] A. Devices, “Digital Accelerometer ADXL345.” USA, p. 16, 2015.

[61] Honeywell, “Honeywell HumidIcon ™ Digital Humidity / Temperature Sensors : HIH-6130 / 6131 Series.” pp. 1–13.

[62] Honeywell, “Honeywell HumidIcon ™ Digital Humidity / Temperature Sensors : HIH-6130 / 6131 Series.” pp. 1–13.

[63] A. Devices, “Digital Accelerometer ADXL345.” USA, p. 16, 2015.

.

[64] J. Carvajal-Godínez *et al.*, “Diseño de un nodo con arquitectura abierta para aplicaciones con redes inalámbricas de sensores (CRTECMOTE),” *Tecnol. en Marcha*, vol. 24, no. 2, pp. 27–33, 2011.

[65] I. Aplicada, “Módulos XBee,” 2012. [Online]. Disponible en: http://alvarounal.blogspot.com/2011/10/modulos-xbee-parte1.html.

[66] Digi, “Digi XBee.” [Online]. Disponible en: https://www.digi.com/products/xbee-rf-solutions/embedded-rf-modules-modems/xbee-wi-fi#overview. [Accedido: 05-Nov-2017].

[67] ADH Technology, “GP-635T Easy to Use.” .

[68] Hellermanntyton, “Catalogo Hellermanntyton,” in *Definición y Tabla del Grado de Protección (IP), acorde a DIN EN IEC 60529*, vol. XXXIII, no. 2, 2012, pp. 81–87.

[69] Hellermanntyton, “Catalogo Hellermanntyton,” in *Definición y Tabla del Grado de Protección (IP), acorde a DIN EN IEC 60529*, vol. XXXIII, no. 2, 2012, pp. 81–87.

[70] R. Arora, “I2C Bus Pullup Resistor Calculation,” no. February, pp. 1–5, 2015.

[71] M. Hart, “TinyGPSPlus.” [Online]. Disponible en: https://github.com/mikalhart/TinyGPSPlus. [Accedido: 02-Jan-2016].

[72] E. Robert, “SparkFun ADXL345 Arduino Library.” [Online]. Disponible en: https://github.com/sparkfun/SparkFun_ADXL345_Arduino_Library. [Accedido: 02-Jan-2016].

[73] K. Townsend, “Adafruit ADXL345.” [Online]. Disponible en: https://github.com/adafruit/Adafruit_ADXL345. [Accedido: 02-Jan-2016].

[74] K. Townsend, “Adafruit Sensor.” [Online]. Disponible en: https://github.com/adafruit/Adafruit_Sensor. [Accedido: 02-Jan-2016].

[75] A. Wrapp, “Xbee Arduino.” [Online]. Disponible en: https://github.com/andrewrapp/xbee-arduino. [Accedido: 03-Jan-2016].

.

ANEXOS

.

ANEXO 1. FUNCIONES DE LOS PUERTOS

x	I/O Puerto	Alimentación	Tipo	A	B(2)(3)					C	D	E	F	G	H
SAMD21J				**EIC**	**REF**	**ADC**	**AC**	**PTC**	**DAC**	**SERCOM(2)(3)**	**SERCOM-ALT**	**TC(4)/TCC**	**TCC**	**COM**	**AC/GCLK**
1	PA00	VDDANA		EXTINT[0]							SERCOM1/ PAD[0]	TCC2/WO[0]			
2	PA01	VDDANA		EXTINT[1]							SERCOM1/ PAD[1]	TCC2/WO[1]			
3	PA02	VDDANA		EXTINT[2]		AIN[0]		Y[0]	VOUT						
4	PA03	VDDANA		EXTINT[3]	ADC/VREFA DAC/VREFA	AIN[1]		Y[1]							
5	PB04	VDDANA		EXTINT[4]		AIN[12]		Y[10]							
6	PB05	VDDANA		EXTINT[5]		AIN[13]		Y[11]							
9	PB06	VDDANA		EXTINT[6]		AIN[14]		Y[12]							
10	PB07	VDDANA		EXTINT[7]		AIN[15]		Y[13]							
11	PB08	VDDANA		EXTINT[8]		AIN[2]		Y[14]			SERCOM4/ PAD[0]	TC4/WO[0]			
12	PB09	VDDANA		EXTINT[9]		AIN[3]		Y[15]			SERCOM4/ PAD[1]	TC4/WO[1]			
13	PA04	VDDANA		EXTINT[4]	ADC/VREFB	AIN[4]	AIN[0]	Y[2]			SERCOM0/ PAD[0]	TCC0/WO[0]			
14	PA05	VDDANA		EXTINT[5]		AIN[5]	AIN[1]	Y[3]			SERCOM0/ PAD[1]	TCC0/WO[1]			
15	PA06	VDDANA		EXTINT[6]		AIN[6]	AIN[2]	Y[4]			SERCOM0/ PAD[2]	TCC1/WO[0]			
16	PA07	VDDANA		EXTINT[7]		AIN[7]	AIN[3]	Y[5]			SERCOM0/ PAD[3]	TCC1/WO[1]		I2S/SD[0]	
17	PA08	VDDIO	I2C	NMI		AIN[16]		X[0]		SERCOM0/ PAD[0]	SERCOM2/ PAD[0]	TCC0/WO[0]	TCC1/ WO[2]	I2S/SD[1]	
18	PA09	VDDIO	I2C	EXTINT[9]		AIN[17]		X[1]		SERCOM0/ PAD[1]	SERCOM2/ PAD[1]	TCC0/WO[1]	TCC1/ WO[3]	I2S/ MCK[0]	
19	PA10	VDDIO		EXTINT[10]		AIN[18]		X[2]		SERCOM0/ PAD[2]	SERCOM2/ PAD[2]	TCC1/WO[0]	TCC0/ WO[2]	I2S/ SCK[0]	GCLK_IO[4]
20	PA11	VDDIO		EXTINT[11]		AIN[19]		X[3]		SERCOM0/ PAD[3]	SERCOM2/ PAD[3]	TCC1/WO[1]	TCC0/ WO[3]	I2S/FS[0]	GCLK_IO[5]
Puerto(1)		**Supply**	**Type**	**A**	**B(2)(3)**					**C**	**D**	**E**	**F**	**G**	**H**

.

SAMD21J	I/O Puerto			EIC	REF	ADC	AC	PTC	DAC	SERCOM(2)(3)	SERCOM-ALT	TC(4)/TCC	TCC	COM	AC/GCLK
23	PB10	VDDIO		EXTINT[10]							SERCOM4/ PAD[2]	TC5/WO[0]	TCC0/ WO[4]	I2S/ MCK[1]	GCLK_IO[4]
24	PB11	VDDIO		EXTINT[11]							SERCOM4/ PAD[3]	TC5/WO[1]	TCC0/ WO[5]	I2S/ SCK[1]	GCLK_IO[5]
25	PB12	VDDIO	I2C	EXTINT[12]				X[12]		SERCOM4/ PAD[0]		TC4/WO[0]	TCC0/ WO[6]	I2S/FS[1]	GCLK_IO[6]
26	PB13	VDDIO	I2C	EXTINT[13]				X[13]		SERCOM4/ PAD[1]		TC4/WO[1]	TCC0/ WO[7]		GCLK_IO[7]
27	PB14	VDDIO		EXTINT[14]				X[14]		SERCOM4/ PAD[2]		TC5/WO[0]			GCLK_IO[0]
28	PB15	VDDIO		EXTINT[15]				X[15]		SERCOM4/ PAD[3]		TC5/WO[1]			GCLK_IO[1]
29	PA12	VDDIO	I2C	EXTINT[12]						SERCOM2/ PAD[0]	SERCOM4/ PAD[0]	TCC2/WO[0]	TCC0/ WO[6]		AC/CMP[0]
30	PA13	VDDIO	I2C	EXTINT[13]						SERCOM2/ PAD[1]	SERCOM4/ PAD[1]	TCC2/WO[1]	TCC0/ WO[7]		AC/CMP[1]
31	PA14	VDDIO		EXTINT[14]						SERCOM2/ PAD[2]	SERCOM4/ PAD[2]	TC3/WO[0]	TCC0/ WO[4]		GCLK_IO[0]
32	PA15	VDDIO		EXTINT[15]						SERCOM2/ PAD[3]	SERCOM4/ PAD[3]	TC3/WO[1]	TCC0/ WO[5]		GCLK_IO[1]
35	PA16	VDDIO	I2C	EXTINT[0]				X[4]		SERCOM1/ PAD[0]	SERCOM3/ PAD[0]	TCC2/WO[0]	TCC0/WO[6]		GCLK_IO[2]
36	PA17	VDDIO	I2C	EXTINT[1]				X[5]		SERCOM1/ PAD[1]	SERCOM3/ PAD[1]	TCC2/WO[1]	TCC0/WO[7]		GCLK_IO[3]
37	PA18	VDDIO		EXTINT[2]				X[6]		SERCOM1/ PAD[2]	SERCOM3/ PAD[2]	TC3/WO[0]	TCC0/ WO[2]		AC/CMP[0]
38	PA19	VDDIO		EXTINT[3]				X[7]		SERCOM1/ PAD[3]	SERCOM3/ PAD[3]	TC3/WO[1]	TCC0/ WO[3]	I2S/SD[0]	AC/CMP[1]
39	PB16	VDDIO	I2C	EXTINT[0]						SERCOM5/ PAD[0]		TC6/WO[0]	TCC0/ WO[4]	I2S/SD[1]	GCLK_IO[2]
40	PB17	VDDIO	I2C	EXTINT[1]						SERCOM5/ PAD[1]		TC6/WO[1]	TCC0/ WO[5]	I2S/ MCK[0]	GCLK_IO[3]
41	PA20	VDDIO		EXTINT[4]				X[8]		SERCOM5/ PAD[2]	SERCOM3/ PAD[2]	TC7/WO[0]	TCC0/ WO[6]	I2S/ SCK[0]	GCLK_IO[4]
42	PA21	VDDIO		EXTINT[5]				X[9]		SERCOM5/ PAD[3]	SERCOM3/ PAD[3]	TC7/WO[1]	TCC0/ WO[7]	I2S/FS[0]	GCLK_IO[5]

.

43	PA22	VDDIO	I2C	EXTINT[6]				X[10]		SERCOM3/ PAD[0]	SERCOM5/ PAD[0]	TC4/WO[0]	TCC0/ WO[4]		GCLK_IO[6]
44	PA23	VDDIO	I2C	EXTINT[7]				X[11]		SERCOM3/ PAD[1]	SERCOM5/ PAD[1]	TC4/WO[1]	TCC0/ WO[5]	USB/SOF 1kHz	GCLK_IO[7]
45	PA24	VDDIO		EXTINT[12]						SERCOM3/ PAD[2]	SERCOM5/ PAD[2]	TC5/WO[0]	TCC1/ WO[2]	USB/DM	
46	PA25	VDDIO		EXTINT[13]						SERCOM3/ PAD[3]	SERCOM5/ PAD[3]	TC5/WO[1]	TCC1/ WO[3]	USB/DP	
Puerto(1)	**I/O Puerto**	**Alimentación**	**Tipo**	**A**	**B(2)(3)**					**C**	**D**	**E**	**F**	**G**	**H**
SAMD21J				**EIC**	**REF**	**ADC**	**AC**	**PTC**	**DAC**	**SERCOM(2)(3)**	**SERCOM-ALT**	**TC(4)/TCC**	**TCC**	**COM**	**AC/GCLK**
49	PB22	VDDIO		EXTINT[6]							SERCOM5/ PAD[2]	TC7/WO[0]			GCLK_IO[0]
50	PB23	VDDIO		EXTINT[7]							SERCOM5/ PAD[3]	TC7/WO[1]			GCLK_IO[1]
51	PA27	VDDIO		EXTINT[15]											GCLK_IO[0]
53	PA28	VDDIO		EXTINT[8]											GCLK_IO[0]
57	PA30	VDDIO		EXTINT[10]							SERCOM1/ PAD[2]	TCC1/WO[0]		SWCLK	GCLK_IO[0]
58	PA31	VDDIO		EXTINT[11]							SERCOM1/ PAD[3]	TCC1/WO[1]		SWDIO(5)	
59	PB30	VDDIO	I2C	EXTINT[14]							SERCOM5/ PAD[0]	TCC0/WO[0]	TCC1/ WO[2]		
60	PB31	VDDIO	I2C	EXTINT[15]							SERCOM5/ PAD[1]	TCC0/WO[1]	TCC1/ WO[3]		
61	PB00	VDDANA		EXTINT[0]		AIN[8]		Y[6]			SERCOM5/ PAD[2]	TC7/WO[0]			
62	PB01	VDDANA		EXTINT[1]		AIN[9]		Y[7]			SERCOM5/ PAD[3]	TC7/WO[1]			
63	PB02	VDDANA		EXTINT[2]		AIN[10]		Y[8]			SERCOM5/ PAD[0]	TC6/WO[0]			
64	PB03	VDDANA		EXTINT[3]		AIN[11]		Y[9]			SERCOM5/ PAD[1]	TC6/WO[1]			

.

ANEXO 2. CÓDIGO DEL ACELERÓMETRO ADXL345

```
/*  ******************************************
 *  Ejemplo de sensor módulo acelerómetro ADXL345
 *  de Sparkfun.
 *
 *  Se hace uso de la librería ADXL345 Library
 *  E.Robert @ SparkFun Electronics
 *  Creado: Jul 13, 2016
 *  Actualizado: Sep 06, 2016
 *
 *  Modificado por: Bosmediano Soledad y Pablo Ochoa
 *  Fecha: 22 de enero de 2017 Loja- Ecuador
 *
 *  El uso de este ejemplo hace uso de las funciones
 *  especiales del sensor ADXL345 que se adaptan a la
 *  plataforma para la detección de diferentes sucesos
 *  como:
 *  UN GOLPE PEQUEÑO
 *  DOS GOLPES PEQUEÑOS
 *  CAIDE LIBRE
 *  ACTIVIDAD
 *  INACTIVIDAD
 *
 *  Adaptada para la tarjeta de SODAQ AUTONOMO
 *  ******************************************/

#include <SparkFun_ADXL345.h>         // Librería de SparkFun ADXL345

/*********** COMUNICACIÓN I2C ***********/
ADXL345 adxl = ADXL345();             // Uso para comunicación I2C

/******************** SETUP ******************/
/*          Configuración del ADXL345        */
void setup(){

  SerialUSB.begin(9600);            // Inicia la comunicación SerialUSB
  SerialUSB.println("Funciones de deteccion especial del acelerometro ADXL345");
  SerialUSB.println();

  adxl.powerOn();                     // Enciende el ADXL345
```

.

```
  adxl.setRangeSetting(16);           // Especificar el rango de configuración
                                      // Valores aceptados son 2g, 4g, 8g ó 16g
                                      // Valores altos = Rango de medición más alto
                                      // Valores bajos = Mayor sensibilidad

  adxl.setActivityXYZ(1, 1, 1);       // Establecer para activar la detección de movimiento en los
                                      // ejes "adxl.setActivityXYZ (X, Y, Z);" (1 == ACTIVADO, 0 == APAGADO)

  adxl.setActivityThreshold(45);      // 62.5mg por incremento // Umbral de inactividad (0-255) //

  adxl.setInactivityXYZ(1, 1, 0);     // Establecer para detectar la inactividad en los
                                      // ejes "adxl.data Inactividad XYZ (X, Y, Z);" (1 == ACTIVADO, 0 == APAGADO)

  adxl.setInactivityThreshold(45);    // 62.5mg por incremento // Umbral de inactividad (0-255)
  adxl.setTimeInactivity(10);         // ¿Cuántos segundos de no actividad ha transcurrido?

  adxl.setTapDetectionOnXYZ(1, 1, 1); // Detectar golpes en las direcciones activadas
                                      // "adxl.setTapDetectionOnX (X, Y, Z);" (1 == ACTIVADO, 0 == APAGADO)

  // Establecer valores para lo que se considera uno y dos GOLPES PEQUEÑO (0-255)
  adxl.setTapThreshold(40);           // 62.5 mg por incremento
  adxl.setTapDuration(15);            // 625  µs por incremento
  adxl.setDoubleTapLatency(80);       // 1.25 ms por incremento
  adxl.setDoubleTapWindow(200);       // 1.25 ms por incremento

  // Establecer valores para los que se considera caída libre (0-255)
  adxl.setFreeFallThreshold(7);       // (5 - 9) recomendado - 62.5mg por incremento
  adxl.setFreeFallDuration(30);       // (20 - 70) recomendado - 5ms por incremento

  adxl.InactivityINT(1);
  adxl.ActivityINT(1);
  adxl.FreeFallINT(1);
  adxl.doubleTapINT(1);
  adxl.singleTapINT(1);
}

/****************** LOOP ******************/

void loop(){
```

.

```
  // Lecturas del acelerómetro
  int x,y,z;
  adxl.readAccel(&x, &y, &z);         //Leer los valores del acelerómetro y almacenarlos
                                      //en las variables declaradas arriba x, y, z

  /*
   * NO COMENTAR SI DESEA VER LOS VALORES DE X Y Z DEL ACELERÓMETRO
   */

  //SerialUSB.print(x);
  //SerialUSB.print(", ");
  //SerialUSB.print(y);
  //SerialUSB.print(", ");
  //SerialUSB.println(z);

  ADXL_ISR();

}

/********************* ISR *********************/
/* Busca interrupciones y acciones disparadas   */
void ADXL_ISR() {

  /*
   * getInterruptSource borra todas las acciones activadas después de devolver el valor
   * No vuelva a llamar hasta que necesite volver a comprobar si hay acciones activadas
   */

  byte interrupts = adxl.getInterruptSource();

  // Detección de caída libre
  if(adxl.triggered(interrupts, ADXL345_FREE_FALL)){
    SerialUSB.println("*** SE HA DETECTADO UNA CAIDA ***");
    //Agregar código aquí para realizar acciones cuando se detecta una caída libre
  }

  // Inactividad
  if(adxl.triggered(interrupts, ADXL345_INACTIVITY)){
    SerialUSB.println("*** NO EXISTE ACTIVIDAD ***");
    //Agregar código aquí para realizar acciones cuando no existe actividad
  }
```

.

```
  // Actividad
  if(adxl.triggered(interrupts, ADXL345_ACTIVITY)){
    SerialUSB.println("*** ACTIVIDAD DETECTADA ***");
    //Agregar código aquí para realizar acciones cuando se detecta actividad
  }

  // Detección de DOS GOLPES PEQUEÑOS
  if(adxl.triggered(interrupts, ADXL345_DOUBLE_TAP)){
    SerialUSB.println("*** SE HA DETECTADO DOS GOLPES LEVES ***");
    //Agregar código aquí para realizar acciones cuando se detecta dos golpes pequeños
  }

  // Detección de UN GOLPE PEQUEÑO
  if(adxl.triggered(interrupts, ADXL345_SINGLE_TAP)){
    SerialUSB.println("*** SE HA DETECTADO UN GOLPE LEVE ***");
    //Agregar código aquí para realizar acciones cuando se detecta un golpe pequeño
  }
}
```

.

ANEXO 3. CÓDIGO DEL GPS-635T

```
/*  ******************************************
 *  CÓDIGO PARA DETERMINAR LA POSICIÓN GEOGRÁFICA DE LA MOTA, ASÍ COMO TAMBIÉN LA FECHA Y HORA.
*   Autor: Matt Moson
 *  Referencia: http://arduiniana.org/libraries/tinygpsplus/
 *  Modificado por: Bosmediano Soledad y Pablo Ochoa
 *  Fecha: 23 de enero de 2017 Loja- Ecuador
 *
 *  Adaptada para la tarjeta de SODAQ AUTONOMO
 *  ******************************************/

#include <TinyGPS++.h> //Librería GPS
TinyGPSPlus gps;
// A5 == Tx GPS
// A4 == Rx GPS

// Contador
int count=1;
int estado=0;
int time_on=300;  // 60 segundos
int time_off=30; // 30 segundos
void setup()
{
  SerialUSB.begin(9600);
  Serial3.begin(9600);
  delay(1000);
  SerialUSB.println("Sistema Listo");
  pinMode(PWR_CONTROL, OUTPUT);
  digitalWrite(PWR_CONTROL, HIGH); // Cuando está en bajo entran en modo Sleep manteniendo el RTC y la RAM
  estado=1;
}

void loop()
{

  if(estado==1){
    smartdelay(1000); // Espera 1 segundo, y recopila datos.
    SerialUSB.print(" lat=");
    SerialUSB.print(gps.location.lat(), 6);
    SerialUSB.print(", long=");
```

.

```
    SerialUSB.print(gps.location.lng(), 6);
    SerialUSB.print(", altitud=");
    SerialUSB.print(gps.altitude.meters());
    SerialUSB.print(", sats=");
    SerialUSB.print(gps.satellites.value());

    char sz[32];
    SerialUSB.print(", fecha=");
    sprintf(sz, "%02d/%02d/%02d", gps.date.month(), gps.date.day(), gps.date.year());
    SerialUSB.print(sz);

    SerialUSB.print(", hora=");
    sprintf(sz, "%02d:%02d:%02d", gps.time.hour(), gps.time.minute(), gps.time.second());
    SerialUSB.print(sz);

    SerialUSB.print(",");
    SerialUSB.println("");

    count= count + 1;
    SerialUSB.print("# De datos recibidos:  ");
    SerialUSB.println(count);
    SerialUSB.println();

    if (count==time_on){
      estado=2;
      count=0;
      digitalWrite(PWR_CONTROL, LOW); // OFF GPS
      SerialUSB.println("GPS en Modo ESPERA");
    }

  } else if(estado==2){

    count= count + 1;
    delay(1000); // cada segundo cuenta
    SerialUSB.print("Tiempo de espera  " +String(time_off)+  "  segundos, tiempo transcurrido:  ");
    SerialUSB.print(count);
    SerialUSB.println(" seg");
    if (count==time_off){
      estado=1;
      count=0;
      digitalWrite(PWR_CONTROL, HIGH); // ON GPS
```

.

```
      SerialUSB.println("GPS ENCENDIDO");
    }
  }
}


static void smartdelay(unsigned long ms){
  unsigned long start = millis();
  do {
    while (Serial3.Disponible en())
      gps.encode(Serial3.read());
  } while (millis() - start < ms);
}
```

.

ANEXO 4. CÓDIGO DEL RTC DS3231SN

```
/*  ******************************************
 *  CÓDIGO PARA DETERMINAR EL TIEMPO ACTUAL DEL RTC
 *  Autor: Matt Moson
 *  Referencia: http://arduiniana.org/libraries/tinygpsplus/
 *  Modificado por: Bosmediano Soledad y Pablo Ochoa
 *  Fecha: 23 de enero de 2017 Loja- Ecuador
 *  Adaptada para la tarjeta de SODAQ AUTONOMO
 *  ******************************************/
//Se incluye la librería I2C
#include "Wire.h"
//Se define la dirección del dispositivo
#define DS3231_I2C_ADDRESS 0x68

// Convierte números decimales a números binarios
byte decToBcd(byte val) {
  return( (val/10*16) + (val%10) );
}

// Convierte binarios codificados a números decimales
byte bcdToDec(byte val){
  return( (val/16*10) + (val%16) );
}

void setup(){
  //Se inicializa la conexión I2C
  Wire.begin();
  SerialUSB.begin(9600);
  // Primero se ajusta el tiempo
  // DS3231 segundos, minutos, horas, día, fecha, mes, año
  setDS3231time(50,44,17,5,02,02,17);
}

void setDS3231time(byte second, byte minute, byte hour, byte dayOfWeek, byte dayOfMonth, byte month, byte year) {

  Wire.beginTransmission(DS3231_I2C_ADDRESS);
  Wire.write(byte(0)); // Se ajusta la siguiente entrada para inicializar en el registro de segundos
  Wire.write(decToBcd(second)); // Para segundos
  Wire.write(decToBcd(minute)); // Para minutos
  Wire.write(decToBcd(hour)); // Para horas
```

.

```
  Wire.write(decToBcd(dayOfWeek)); // Para el día de la semana (1=Domingo, 7=Sábado)
  Wire.write(decToBcd(dayOfMonth)); // Para la fecha (1 a 31)
  Wire.write(decToBcd(month)); // Para el mes
  Wire.write(decToBcd(year)); // Para el año (0 a 99)
  Wire.endTransmission();
}

void readDS3231time(byte *second, byte *minute, byte *hour, byte *dayOfWeek, byte *dayOfMonth, byte *month, byte *year) {

  Wire.beginTransmission(DS3231_I2C_ADDRESS);
  Wire.write(byte(0)); // Ajusta el registro de puntero DS3231 a 00h
  Wire.endTransmission();
  Wire.requestFrom(DS3231_I2C_ADDRESS, 7);
  // Se requiere de siete bytes para los datos del DS3231 inicializados desde el registro  00h
  *second = bcdToDec(Wire.read() & 0x7f);
  *minute = bcdToDec(Wire.read());
  *hour = bcdToDec(Wire.read() & 0x3f);
  *dayOfWeek = bcdToDec(Wire.read());
  *dayOfMonth = bcdToDec(Wire.read());
  *month = bcdToDec(Wire.read());
  *year = bcdToDec(Wire.read());
}

void displayTime() {

  byte second, minute, hour, dayOfWeek, dayOfMonth, month, year;
  // Lee los datos desde el DS3231
  readDS3231time(&second, &minute, &hour, &dayOfWeek, &dayOfMonth, &month,
  &year);
  // envía la información al monitor serial
  SerialUSB.print(hour, DEC); // convierte el byte a numero decimal
  SerialUSB.print(":");
  if (minute<10){
    SerialUSB.print("0");
  }
  SerialUSB.print(minute, DEC);
  SerialUSB.print(":");
  if (second<10){
    SerialUSB.print("0");
  }
  SerialUSB.print(second, DEC);
```

.

```
  SerialUSB.print(" ");
  SerialUSB.print(dayOfMonth, DEC);
  SerialUSB.print("/");
  SerialUSB.print(month, DEC);
  SerialUSB.print("/");
  SerialUSB.print(year, DEC);
  SerialUSB.print(" Dia de la semana: ");

  switch(dayOfWeek){
  case 1:
    SerialUSB.println("Domingo");
    break;
  case 2:
    SerialUSB.println("Lunes");
    break;
  case 3:
    SerialUSB.println("Martes");
    break;
  case 4:
    SerialUSB.println("Miercoles");
    break;
  case 5:
    SerialUSB.println("Jueves");
    break;
  case 6:
    SerialUSB.println("Viernes");
    break;
  case 7:
    SerialUSB.println("Sabado");
    break;
  }
}


void loop() {

  displayTime(); // Muestra los datos en tiempo real en el monitor   SerialUSB
  delay(1000);

}
```

.

ANEXO 5. CÓDIGO PARA RSSI

```
/*  ******************************************
 *  INDICADOR DE RSSI PARA LAS DOS RANURAS XBEEs DE LA PLATAFORMAS MEDIANTE
 *  EL USO DEL TCA9554 PARA EL ENCENDIDO DE LOS INDICADORES LED
 *
 *  Creado por: Andrew Rapp (Librería y Ejemplos)
 *  Primera Modificación: http://www.desert-home.com/2012/10/using-xbee-library.html
 *  Segunda Modificado: Bosmediano Soledad y Pablo Ochoa
 *  Fecha: 22 de enero de 2017 Loja- Ecuador
 *
 *  DESCRIPCIÓN:
 *
 *  En este ejemplo se hace la recepción de paquetes en el módulo de
 *  comunicación XBEE PRO 900 configurado en modo API (With Scapes)
 *  configurado en DIGI-MESH.
 *
 *  El objetivo de este código es el de mostrar el nivel de señal RSSI
 *  tanto en el monitor serial USB como en los 4 niveles de leds de las
 *  ranuras XBee de la plataforma. Para lograr obtener el nivel de
 *  señal se envía el comando ATDB para adquirir el valor de RSSI,
 *  posteriormente se hace uso del TCA9554 para el encendido de
 *  leds correspondientes.
 *
 *  Adaptada para la tarjeta de SODAQ AUTONOMO
 *  ******************************************/

#include <XBee.h>  // Libreria del Xbee
#include <Wire.h>

unsigned char data_rssi; // Variable para almacenar el valor de RSSI recibido
int rssi;                // Convierte el valor de RSSI a entero
int estado=0;            // Estado de recepción, SI se recibe paquetes => estado = 1
                         // si NO => estado = 0
int count=0;             // Contador: Cuenta el número de veces en el que no se recibe paquetes para
                         // determinar que se ha perdido la conexión
int tiempo_espera=10;    // Tiempo de espera de inactividad para el contador (count)

XBee xbee = XBee();
XBeeResponse response = XBeeResponse();
```

.

```
// Crea objetos de respuesta reutilizables para las respuestas que esperamos manejar
ZBRxResponse rx = ZBRxResponse();

// Para RSSI con comandos AT
uint8_t DBCmd[] = {'D','B'}; // Con este comando se obtiene el RSSI del último paquete recibido
AtCommandRequest atRequest = AtCommandRequest(DBCmd); // Envio del comando AT
AtCommandResponse atResponse = AtCommandResponse();   // Obtener respuesta


// I2C Dirección del dispositivo es 0 1 0 0   A2 A1 A0  (ver datasheet)
// A2 A1 A0 = 0x20 = 000
// TCA9554          0100       0000
#define TCA_LED_ADDRESS (0x4 << 3 | 0x0)
#define REGISTER_CONFIG (0x03)  // Para entrar en modo configuración
#define REGISTER_OUTPUT (0x01) // Configura los puertos como salida



void setup() {

  // Inicia comunicación SerialUSB
  SerialUSB.begin(9600);
  // Inicia el puerto serial para cualquier de la ranura Xbee
  Serial1.begin(9600); // Serial1 para Xbee 1 y Serial2 para Xbee2

  // Una vez iniciado el puerto, se establece el (xbee) al Serial1 o Serial2
  xbee.setSerial(Serial1);
  SerialUSB.println("Se ha establecido la comunicación");

  // Control de energía
  // BEE_VCC para Xbee 1
  // BEE2_VCC para Xbee 2
  digitalWrite(BEE_VCC, HIGH);  // Enciende la ranura Xbee

  // TCA 94554
  Wire.begin();                       // Inicio del Bus I2C
  gpio_dir(TCA_LED_ADDRESS, 0x00);    // Envía la dirección de registro de configuración
  gpio_write(TCA_LED_ADDRESS, 0xff);  // Envía la dirección de registro de salida(pines)

  // APAGA LOS LEDS CON NIVEL EN ALTO PARA AMBAS RANURAS XBEE
  Wire.beginTransmission(TCA_LED_ADDRESS);   // Se inicia la transmisión para la dirección del TCA9554
```

.

```
  Wire.write(REGISTER_OUTPUT );              // Se envía el registro 0x01 para usar los puertos como salida
  Wire.write(B11111111);                     // Se envía el byte(8 bits) que contiene el nivel lógico
                                             // de los 8 puertos, en este caso se apagan todos los leds.
  Wire.endTransmission();                    // Se termina la transmisión
}


void loop() {

    xbee.readPacket();  // Lee los paquetes

    // Chequea si existe una respuesta en el Xbee
    if (xbee.getResponse().isDisponible en()){
      SerialUSB.println();
      SerialUSB.print("Tipo de trama (Api ID):  ");
      // Api ID, el primer byte de la trama. (Datos específicos del paquete)
      SerialUSB.println(xbee.getResponse().getApiId(), HEX);

      if (xbee.getResponse().getApiId() == ZB_RX_RESPONSE) {
        xbee.getResponse().getZBRxResponse(rx);

        /* Así es como se obtiene la dirección de 64 bits del
         * paquete entrante para saber de qué dispositivo proviene
        */
        SerialUSB.print("Se ha recibido un paquete en el Xbee: ");
        XBeeAddress64 senderLongAddress = rx.getRemoteAddress64();
        print32Bits(senderLongAddress.getMsb());
        SerialUSB.print(" ");
        print32Bits(senderLongAddress.getLsb());

        // Así es como se obtiene la dirección de 16 bits
        // del remitente
        uint16_t senderShortAddress = rx.getRemoteAddress16();
        SerialUSB.print(" (");
        print16Bits(senderShortAddress);
        SerialUSB.println(")");


        if (rx.getOption() & ZB_PACKET_ACKNOWLEDGED){
            // El remitente consiguió un ACK
          SerialUSB.println("Paquete reconocido ACK");
```

.

```
}
if (rx.getOption() & ZB_BROADCAST_PACKET){
  // Indica que es un paquete broadcast
  SerialUSB.println("Paquete broadcast");
}

SerialUSB.print("Checksum es ");
SerialUSB.println(rx.getChecksum(), HEX);

// Longitud del paquete
SerialUSB.print("La longitud del paquete es ");
SerialUSB.print(rx.getPacketLength(), DEC);

// Longitud de la carga útil
SerialUSB.print(", longitud de carga util (payload) es ");
SerialUSB.println(rx.getDataLength(),DEC);

// Datos actuales que se enviaron
SerialUSB.println("Datos recibidos: ");
for (int i = 0; i < rx.getDataLength(); i++) {
  print8Bits(rx.getData()[i]);
  SerialUSB.print(' ');
}

// Representación en ASCII para visualizar el texto enviado
SerialUSB.println();
for (int i= 0; i < rx.getDataLength(); i++){
  SerialUSB.write(' ');
  if (iscntrl(rx.getData()[i])){
    SerialUSB.write(' ');
  }else{
    SerialUSB.write(rx.getData()[i]);
  }
  SerialUSB.write(' ');
}
SerialUSB.println();

// Así, por ejemplo, se puede hacer algo como esto:
// handleXbeeRxMessage(rx.getData(), rx.getDataLength());
//SerialUSB.println();
```

.

```
  SerialUSB.println("Datos de la trama:");
  for (int i = 0; i < xbee.getResponse().getFrameDataLength(); i++) {
    print8Bits(xbee.getResponse().getFrameData()[i]);
    SerialUSB.print(' ');
  }
  SerialUSB.println();
}

/*
 * ENVÍO DE COMANDO ATDB E INDICADORES LED (RSSI)
*/

// Una vez que se recibe el paquete, se envía el comando AT DB
// para obtener el nivel de señal RSSI del último paquete recibido
sendAtCommand();          // Envido del comando AT DB

/* RSSI_VIEW (intervalo, minimo_rssi, dispositivo_xbee)
 *  intervalo: Debe ser un valor negativo, indica el intervalo entre cada led indicador.
 *  Para este ejemplo esta cada -5dB.
 *
 *  minimo_rssi: el nivel de señal mínimo ( Por ejemplo, puede usarse la sensibilidad del receptor)
 *
 *  dispositivo_xbee= Selecciona de que ranura Xbee se desea indica el nivel de señal.  Si es 1 (Xbee1), 2 (Xbee2)
 *
 *  Por ejemplo para el Xbee PRO900 tiene una potencia de transmisión de 50 mW (+17dBm)
 *  con una sensibilidad en el receptor de -100dBm
 *  el rango operación es:
 *  Máximo= Sensibilidad + potencia de transmisión = -100dBm + 17dBm =-83dBm
 *  Mínimo= Sensibilidad = -100dBm
 *  Rango de medición = -83 dBm a -100 dBm
 *
 *  En este ejemplo se establece el valor mínimo de rssi como -104dBm
 *  en un intervalo de -5dBm el valor máximo se genera en el código
 *  de la siguiente manera (sensibilidad - intervalo*4(niveles)) = -104 - (4*(-5)) =-84dBm
 *
*/
RSSI_VIEW (-5,-104,1); // Indicador de nivel de señal para Xbee1

count=0; // Resetea contador a cero
```

.

```
    } else if (xbee.getResponse().isError()) {
       // Algun tipo de error ha pasado
      SerialUSB.print("********************************** error code:");
      SerialUSB.println(xbee.getResponse().getErrorCode(),DEC);

    } else {

      count +=1;

      SerialUSB.println("NO SE HA RECIBIDO PAQUETES");

      // Si NO se recibe paquetes hasta cierto tiempo, entonces se ha perdido la conexión.
      if(count> tiempo_espera) {
        SerialUSB.println("SE HA PERDIDO LA CONEXION");
      }

      delay(1000); // Cuenta cada segundo

    }

}



// Funciones para recibir

void handleXbeeRxMessage(uint8_t *data, uint8_t length){
  // Muestra cómo obtener los datos, aquí se debe colocar
  // el código para poder realizar algo que se desee hacer
  for (int i = 0; i < length; i++){
  SerialUSB.print(data[i]);
  data_rssi=data[i];
  }
  rssi=int(data_rssi);
}


/* estas rutinas son sólo para imprimir los datos con
 *  los ceros a la izquierda y permitir que el formato
 *  sea fácil de leer. */
```

.

```
void print32Bits(uint32_t dw){
  print16Bits(dw >> 16);
  print16Bits(dw & 0xFFFF);
}

void print16Bits(uint16_t w){
  print8Bits(w >> 8);
  print8Bits(w & 0x00FF);
}

void print8Bits(byte c){
  uint8_t nibble = (c >> 4);
  if (nibble <= 9){
    SerialUSB.write(nibble + 0x30);
  }else{
    SerialUSB.write(nibble + 0x37);
  }

  nibble = (uint8_t) (c & 0x0F);
  if (nibble <= 9){
    SerialUSB.write(nibble + 0x30);
  }else{
    SerialUSB.write(nibble + 0x37);
  }
}

// Función para enviar comandos AT

void sendAtCommand() {
  SerialUSB.println("Enviando comando AT al XBee");

  // Envía el comando AT
  xbee.send(atRequest);

  // Espera 5 segundos para obtener respuesta del estado
  if (xbee.readPacket(2000)) {
    // Si hubo respuesta
   if (xbee.getResponse().getApiId() == AT_COMMAND_RESPONSE) {
      xbee.getResponse().getAtCommandResponse(atResponse);
```

.

```
      if (atResponse.isOk()) {
        SerialUSB.print("Comando [");
        SerialUSB.print(atResponse.getCommand()[0]);
        SerialUSB.print(atResponse.getCommand()[1]);
        SerialUSB.println("] Respuesta Exitosa!");

        if (atResponse.getValueLength() > 0) {
          SerialUSB.print("La longitud del valor del comando es: ");
          SerialUSB.println(atResponse.getValueLength(), DEC);

          SerialUSB.print("Valor del comando: ");

          for (int i = 0; i < atResponse.getValueLength(); i++) {
            SerialUSB.print(atResponse.getValue()[i], DEC);
            SerialUSB.print(" ");

          }

          SerialUSB.println("");
          SerialUSB.print("Nivel de senal RSSI: -");
          handleXbeeRxMessage(atResponse.getValue(), atResponse.getValueLength()); //
          SerialUSB.println(" dBm");
        }
      }
         else {
        SerialUSB.print("El comando regresa un condigo de error: ");
        SerialUSB.println(atResponse.getStatus(), HEX);
      }
    } else {
      SerialUSB.print(" No se consiguio respuesta al comando AT");
      SerialUSB.print(xbee.getResponse().getApiId(), HEX);
    }
  } else {
    // comando AT fallido
    if (xbee.getResponse().isError()) {
      SerialUSB.print("Error al leer el paquete.  Codigo de error: ");
      SerialUSB.println(xbee.getResponse().getErrorCode());
    }
    else {
      SerialUSB.print("No hay respuesta del radio Xbee");
    }
```

.

```
  }
}


// Función de dirección de registro de configuración.
void gpio_dir(int address, int dir) {
  // Envía la dirección de configuración de registro
  Wire.beginTransmission(address);
  Wire.write(REGISTER_CONFIG);

  // Conecta  al dispositivo y envía dos bytes
  Wire.write(0xff & dir);  //  byte en bajo
  Wire.write(dir >> 8);    //  byte en alto

  Wire.endTransmission();
}


// Función dirección de registro de los puertos como salida del TCA9554
void gpio_write(int address, int data) {
  // Envía la dirección de registro de salida
  Wire.beginTransmission(address);
  Wire.write(REGISTER_OUTPUT);

  // Conecta al dispositivo y envía dos bytes
  Wire.write(0xff & data);  //  byte en bajo
  Wire.write(data >> 8);    //  byte en alto
  Wire.endTransmission();

}

// Función RSSI TCA94554
void RSSI_VIEW (int intervalo, int minimo_rssi, int dispositivo_xbee) {
  int inter0= minimo_rssi;
  int inter1= minimo_rssi - intervalo;
  int inter2= minimo_rssi - 2*intervalo;
  int inter3= minimo_rssi - 3*intervalo;
  int inter4= minimo_rssi - 4*intervalo;

  rssi= rssi*(-1);
```

.

```
byte rssi_led;
// Comparación entre intervalos
// EJEMPLO
// -50 a -60 Nivel de señal idóneo
// -60 a -70 Nivel de señal bueno
// -70 a -80 Nivel de señal normal
// -80 a -90 Nivel de señal aceptable

if (dispositivo_xbee==1){

  if(rssi>inter3 && rssi<(inter4+1)){
    rssi_led=B11110000;
    SerialUSB.print("Nivel de RSSI XBEE1 entre: ");
    SerialUSB.print(inter4);
    SerialUSB.print(" a ");
    SerialUSB.println(inter3);
  } else if(rssi>inter2 && rssi<(inter3)){
    rssi_led=B11111000;
    SerialUSB.print("Nivel de RSSI XBEE1 entre: ");
    SerialUSB.print(inter3);
    SerialUSB.print(" a ");
    SerialUSB.println(inter2);
  } else if(rssi>inter1 && rssi<(inter2)){
    rssi_led=B11111100;
    SerialUSB.print("Nivel de RSSI XBEE1 entre: ");
    SerialUSB.print(inter2);
    SerialUSB.print(" a ");
    SerialUSB.println(inter1);
  } else if(rssi>inter0 && rssi<(inter1)){
    rssi_led=B11111110;
    SerialUSB.print("Nivel de RSSI XBEE1 entre: ");
    SerialUSB.print(inter1);
    SerialUSB.print(" a ");
    SerialUSB.println(inter0);
  }
} else if(dispositivo_xbee==2){

  if(rssi>inter3 && rssi<(inter4+1)){
    rssi_led=B00001111;
    SerialUSB.print("Nivel de RSSI XBEE2 entre: ");
    SerialUSB.print(inter4);
```

.

```
      SerialUSB.print(" a ");
      SerialUSB.println(inter3);
    } else if(rssi>inter2 && rssi<(inter3)){
      rssi_led=B10001111;
      SerialUSB.print("Nivel de RSSI XBEE2 entre: ");
      SerialUSB.print(inter3);
      SerialUSB.print(" a ");
      SerialUSB.println(inter2);
    } else if(rssi>inter1 && rssi<(inter2)){
      rssi_led=B11001111;
      SerialUSB.print("Nivel de RSSI XBEE2 entre: ");
      SerialUSB.print(inter2);
      SerialUSB.print(" a ");
      SerialUSB.println(inter1);
    } else if(rssi>inter0 && rssi<(inter1)){
      rssi_led=B11101111;
      SerialUSB.print("Nivel de RSSI XBEE2 entre: ");
      SerialUSB.print(inter1);
      SerialUSB.print(" a ");
      SerialUSB.println(inter0);
    }

  }
     // Encendido de leds RSSI
     Wire.beginTransmission(TCA_LED_ADDRESS);
     Wire.write(REGISTER_OUTPUT);
     Wire.write(rssi_led);
     Wire.endTransmission();
     delay(500);  // Se añade un tiempo de espera para mantener encendidos los leds

     // Apagado de todos los leds
     Wire.beginTransmission(TCA_LED_ADDRESS);
     Wire.write(REGISTER_OUTPUT);
     Wire.write(B11111111);
     Wire.endTransmission();

}
```

.

.

ANEXO 6. CÓDIGO CONTROL DE ENERGÍA

```
/*  ******************************************
 *  CÓDIGO PARA CONTROL DE VOLTAJE DE (I2C, DIGITAL Y ANALÓGICO) Y PINES DIGITALES
 *  DE SALIDA (P4, P5, P6, P7)
 *
 *  Referencia: http://tronixstuff.com/2014/12/01/tutorial-using-ds1307-and-ds3231-real-time-clock-modules-with- arduino/

 *  Modificado por: Bosmediano Soledad y Pablo Ochoa
 *  Fecha: 23 de enero de 2017 Loja- Ecuador
 *
 *  En este código se hace la comunicación I2C con el TCA9554 para lograr controlar el voltaje de
 *  los suministros de energía diseñados, además la salida de 4 pines digitales adicionales.
 *
 *  Adaptada para la tarjeta de SODAQ AUTONOMO
 *  ******************************************/

#include <Wire.h>
byte control_registro=B00000000;
//  I2C Dirección del dispositivo es 0 1 0 0   A2 A1 A0  (ver datasheet)
//
//  TCA9554                   0100       0001
#define TCA_CONTROL_ADDRESS (0x4 << 3 | 0x01)
#define REGISTER_CONFIG (0x03) // Para entrar a configuración
#define REGISTER_OUTPUT (0x01) // Configura los pines como salida


void setup() {

  Wire.begin();
  gpio_dir(TCA_CONTROL_ADDRESS, 0x00);    // Envía la dirección de registro de configuración
  gpio_write(TCA_CONTROL_ADDRESS, 0xff);  // Envía la dirección de registro de salida(pines)
}

void loop() {

// bit = 1 ==> Encender ; bit=0 ==> Apagar
// Byte= [  bit,  bit,  bit,  bit,     bit,        bit,       bit,     bit  ]
// Byte= [ {P7}, {P6}, {P5}, {P4},{NO USADO}, {ANAL0GICO}, {DIGITAL}, {I2C} ]

  ON_OFF_TCA94554("I2C",1); // ENCENDER I2C
```

.

```
  delay(3000);
  ON_OFF_TCA94554("DIGITAL",1); // ENCENDER DIGITAL
  delay(3000);
  ON_OFF_TCA94554("ANALOGICO",1); // ENCENDER ANALÓGICO
  delay(3000);
  ON_OFF_TCA94554("ANALOGICO",0); // APAGAR ANALÓGICO
  delay(3000);
  ON_OFF_TCA94554("DIGITAL",0); // APAGAR DIGITAL
  delay(3000);
  ON_OFF_TCA94554("I2C",0); // APAGAR I2C
  delay(3000);

}

// Función de dirección de registro de configuración.
void gpio_dir(int address, int dir) {
  //  Envía la dirección de configuración de registro
  Wire.beginTransmission(address);
  Wire.write(REGISTER_CONFIG);

  //  Coneceta al dispositivo y envía dos bytes
  Wire.write(0xff & dir);  //  byte en bajo
  Wire.write(dir >> 8);    //  byte en alto

  Wire.endTransmission();
}


// Funcion dirección de registro de pines como salida del TCA9554
void gpio_write(int address, int data) {
  //  Envía la dirección de registro de salida
  Wire.beginTransmission(address);
  Wire.write(REGISTER_OUTPUT);

  //  Conecta al dispositivo y envía dos bytes
  Wire.write(0xff & data);  //  byte en bajo
  Wire.write(data >> 8);    //  byte en alto
  Wire.endTransmission();

}
```

.

```
// ***************FUNCIÓN DE CONTROL******************
// Se envía un byte, cada bit pertenece a cada una de las salidas del TCA9554
// Si el byte enviado es B0000000 todo el suministro de energía y pines adicionales
// se apaga, por el contrario, al enviar B11111111 todos se encienden.
// La correspondencia seria de la siguiente manera
// Byte= [  bit,  bit,  bit,  bit, bit,  bit,  bit,  bit  ]
// Byte= [ {P7}, {P6}, {P5}, {P4}, {P3}, {P2}, {P1}, {P0} ]
// P7=P7
// P6=P6
// P5=P5
// P4=P4
// P3=Sin uso
// P2=ANALÓGICO
// P1=DIGITAL
// P0=I2C

void ON_OFF_TCA94554(String pin_out, int estado_on_off){
  byte I2C;
  byte DIGITAL;
  byte ANALOGICO;
  byte P4;
  byte P5;
  byte P6;
  byte P7;

  if (estado_on_off==1){
    if(pin_out=="I2C"){
      I2C=B00000001;
      control_registro=control_registro | I2C;
    } else if (pin_out=="DIGITAL"){
      DIGITAL=B00000010;
      control_registro=control_registro | DIGITAL;
    } else if (pin_out=="ANALOGICO"){
      ANALOGICO=B00000100;
      control_registro=control_registro | ANALOGICO;
    } else if (pin_out=="P4"){
      P4=B00010000;
      control_registro=control_registro | P4;
    } else if (pin_out=="P5"){
      P5=B00100000;
      control_registro=control_registro | P5;
```

.

```
    } else if (pin_out=="P6"){
      P6=B01000000;
      control_registro=control_registro | P6;
    } else if (pin_out=="P7"){
      P7=B10000000;
      control_registro=control_registro | P7;
    }
  } else if (estado_on_off==0){
    if(pin_out=="I2C"){
      I2C=B11111110;
      control_registro=control_registro & I2C;
    } else if (pin_out=="DIGITAL"){
      DIGITAL=B11111101;
      control_registro=control_registro & DIGITAL;
    } else if (pin_out=="ANALOGICO"){
      ANALOGICO=B11111011;
      control_registro=control_registro & ANALOGICO;
    } else if (pin_out=="P4"){
      P4=B11101111;
      control_registro=control_registro & P4;
    } else if (pin_out=="P5"){
      P5=B11011111;
      control_registro=control_registro & P5;
    } else if (pin_out=="P6"){
      P6=B10111111;
      control_registro=control_registro & P6;
    } else if (pin_out=="P7"){
      P7=B01111111;
      control_registro=control_registro & P7;
    }
  }
  // La decisión tomada se ejecuta
  Wire.beginTransmission(TCA_CONTROL_ADDRESS);
  Wire.write(REGISTER_OUTPUT ); // GPIO
  Wire.write(control_registro); // port
  Wire.endTransmission();
}
```