UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO

FACULTAD DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA, INFORMÁTICA Y MECÁNICA

ESCUELA PROFESIONAL DE INGENIERÍA ELECTRÓNICA

TESIS:

Para optar el título profesional de **Ingeniero Electrónico**

"DISEÑO DE UNA RED INALÁMBRICA CON DISPOSITIVOS BLUETOOTH Y ZIGBEE PARA EL MONITOREO DE TEMPERATURA EN LA CRIANZA DE TRUCHAS DE LAS PISCIFACTORÍAS DEL DISTRITO DE LUCRE "

RESPONSABLE:

Bach. Edwin Oraica Uscamayta

ASESOR:

Ing. Mgt. Milton Jhon Velasquez Curo

Cusco - Perú
2019

Dedicatoria

Dedico este trabajo a mi amada esposa, por su apoyo y animo que me brinda día con día para alcanzar nuevas metas tanto profesionales como personales.

A mis queridos padres que desde el cielo me guían a ser mejor persona, tanto a mi hermano Edison por ser mi mayor aliento en mi formación universitaria y siempre alentarme a cumplir mis metas.

Cada uno de ellos son mi motivación y mi fuerza para seguir adelante los quiero mucho.

Agradecimientos

Agradezco de manera especial a mi esposa Yeny, por su comprensión y apoyo y como no mencionar a mi mayor inspiración, mis hijos Melissa y Salvador.

Finalmente, a los ingenieros aquellos que marcaron cada etapa de mi camino universitario, y que me ayudaron en asesorías y dudas presentadas en la elaboración de la tesis.

RESUMEN

El presente proyecto de innovación tecnológica denominado: DISEÑO DE UNA RED INALÁMBRICA CON DISPOSITIVOS BLUETOOTH Y ZIGBEE PARA EL MONITOREO DE TEMPERATURA EN LA CRIANZA DE TRUCHAS DE LAS PISCIFACTORÍAS DEL DISTRITO DE LUCRE, tiene como objetivo principal, el diseñar y elaborar un prototipo de sistema de monitoreo basado en las tecnologías de la información, que permita mejorar la crianza y el manejo de truchas en las piscifactorías de la cuenca hidrográfica del rio Lucre (en cautiverio), con ello se puede identificar el parámetro como temperatura, y que también este sistema brinda soporte para adicionar mediciones de parámetros como oxígeno, pH, conductividad eléctrica (entre otros), elementos fundamentales en el proceso de producción del trucha; asimismo, la implementación de este sistema permite a los usuarios (dueños de las piscifactorías) tener un mejor control, dado que se tendrá un monitoreo en tiempo real de los parámetros mencionados. Este proyecto que presento tiene la intención de contribuir a los resultados en cuanto a producción y en consecuencia que la evolución económica de la zona sean favorables, estimulante y viables para poner en práctica el proyecto a mayor escala en toda la región del Cusco, y con la finalidad de desarrollar esta idea de negocio teniendo en cuenta el carácter empresarial y de emprendimiento, aplicando óptimamente la tecnología que podemos desarrollar dentro de la universidad; y el factor económico para contribuir a la mejora de la crianza de trucha como actividad productiva para los pobladores de localidades adyacentes a fuentes similares como la cuenca hidrográfica del rio Lucre, de esta manera la interacción Universidad-Empresa-Sociedad, permite generar posibilidades de desarrollo económico a nuestra región y que a largo plazo con la conformación de cadenas productivas a nivel de la región Cusco, permita contribuir al PBI regional y nacional.

Como resultados de la presente tesis es que se ha obtenido el diseño de una red de monitoreo de parámetros del agua con componentes para la comunicación y sensado, y el diseño de los sistemas terminales de monitorización para cada usuario de las piscifactorías, también el desarrollo de un prototipo del sistema indicado, en el cual se hicieron las pruebas de validación con el multímetro Fluke 287 en el módulo prototipo en laboratorio, también se realizó experiencias de prueba del sistema en el área de estudio,

como es una de las piscifactorías en Lucre se pueden observar estas evidencias en las figuras del Capítulo 3.

La conclusión principal de este proyecto es que se ha conseguido el diseño y desarrollo de un sistema prototipo de monitoreo de parámetros del agua en base a dispositivos Zigbee y Bluetooth para la producción de trucha en la ribera del rio Lucre, dentro de este módulo de referencia también se ha conseguido definir los distintos módulos de transmisión y recepción; así, como la elección de los sensores adecuados para la medición de los parámetros indicados. Y culminando en la operación y pruebas de validación satisfactorias del sistema.

SUMMARY

This technological innovation project called: DESIGN OF A WIRELESS NETWORK WITH BLUETOOTH AND ZIGBEE DEVICES FOR TEMPERATURE MONITORING IN THE CROSSING OF TRUCHAS OF THE LISTER DISTRICT PISCIFACTORIES, has as its main objective, to design and develop a prototype system monitoring based on information technologies, which allows to improve the breeding and management of trout in the fish farms of the Lucre river basin (in captivity), with this the parameter can be identified as temperature, and that this system also provides support for adding parameter measurements such as oxygen, pH, electrical conductivity (among others), fundamental elements in the trout production process; Likewise, the implementation of this system allows users (owners of the farms) to have better control, given that there will be real-time monitoring of the mentioned parameters. This project that I present intends to contribute to the results in terms of production and consequently that the economic evolution of the area is favorable, stimulating and viable to implement the project on a larger scale throughout the Cusco region, and with the purpose of developing this business idea taking into account the entrepreneurial and entrepreneurial nature, optimally applying the technology that we can develop within the university; and the economic factor to contribute to the improvement of trout breeding as a productive activity for residents of towns adjacent to similar sources such as the river basin of the Lucre river, in this way the University-Company-Society interaction, allows to generate development possibilities economic to our region and that in the long term with the conformation of productive chains at the level of the Cusco region, it allows to contribute to the regional and national GDP.

As a result of this thesis, the design of a water parameter monitoring network with components for communication and sensing has been obtained, and the design of the terminal monitoring systems for each user of the fish farms, also the development of A prototype of the indicated system, in which the validation tests were carried out with the Fluke 287 multimeter in the prototype module in the laboratory, also tested the system in the study area, as it is one of the fish farms in Lucre. You can see this evidence in the figures in Chapter 3.

The main conclusion of this project is that the design and development of a prototype water parameter monitoring system based on Zigbee and Bluetooth devices for the production of trout on the banks of the Lucre river, within this reference module has been achieved It has also been possible to define the different transmission and reception modules; thus, as the choice of the appropriate sensors for the measurement of the indicated parameters. And culminating in the operation and satisfactory validation tests of the system.

Tabla de contenido

Tabla de figuras

Contenido de Tablas

Introducción

El potencial de crianza de trucha dentro de la región del Cusco y específicamente en el lugar de estudio que es la cuenca hidrográfica del rio Lucre, se debe al recurso hídrico y al clima adecuado en las diferentes fases de crianza. Este proyecto pretende aportar en que este potencial se haga realidad haciendo que los dueños de las piscifactorías tengan un mayor control en la crianza y conviertan más adelante la producción artesanal de trucha en una más elaborada y de producción industrial.

Se ha observado que existen muchos parámetros en el agua que intervienen en la crianza de truchas, entre estos que involucran este proceso están la temperatura, oxigeno, pH, etc. como se explica en el capítulo 1 se realiza un monitoreo nulo y en algunos casos de forma manual el monitoreo de estos parámetros. Esto produce errores que son reflejados en el deterioro de la calidad del agua de los estanques

Ante la ineficiencia del monitoreo de los parámetros antes mencionados, este proyecto busca elaborar un diseño de una red de sensores que permita al usuario obtener la información de los distintos parámetros que intervienen en la crianza de truchas. La tesis muestra en los capítulos detalla los aspectos principales a considerar para el diseño del sistema, la elección de sensores y los dispositivos de comunicación necesarios y óptimos. Luego se describe la configuración de todo el sistema y se elaboró un pequeño prototipo que se utilizó en las pruebas.

Capítulo 1 : MARCO REFERENCIAL

1.1. Título de trabajo de tesis

“DISEÑO DE UNA RED INALÁMBRICA CON DISPOSITIVOS BLUETOOTH Y ZIGBEE PARA EL MONITOREO DE TEMPERATURA EN LA CRIANZA DE TRUCHAS DE LAS PISCIFACTORÍAS DEL DISTRITO DE LUCRE"

1.1.1. Responsable

Br. Edwin Oraica Uscamayta.

1.1.2. Asesor

Ing. Milton Jhon Velasquez Curo.

1.1.3. Ámbito geográfico

El presente trabajo tiene como escenario el diseño de una red inalámbrica utilizando dispositivos bluetooth y zigbee de bajo consumo de energía que permita monitorear la temperatura de las piscifactorías que se encuentran a la ribera del rio Lucre. El desarrollo del presente sistema permitirá adquirir valores de temperatura que serán almacenados y presentados en tiempo real, las cuales servirán como referencia para que los usuarios puedan tomar acciones necesarias.

Rio Lucre datos:

Departamento: Cusco

Provincia: Quispicanchis

Distrito: Lucre

Altitud: 3213 msnm

Ubicación: UTM WGS 84 Zona 19S, 201772 E, 8489282 N

Caudal: 5.20 l/s

Figura 1: ***Localización Geográfica***

Fuente: *Instituto Nacional de Estadística e Informática (INEI)*

1.2. Generalidades

1.2.1. Planteamiento del problema

En un estudio realizado por el Ministerio de la Producción el año 2010 denominado "Elaboración de estudio de mercado de la trucha en Arequipa, Cusco, Lima Huancayo y Puno" cuyo objetivo era el de determinar el equilibrio entre oferta y demanda de trucha a nivel nacional, ha concluido que la velocidad a la que crece la demanda de trucha estaría sobrepasando a la oferta a partir del año 2012 generando una demanda insatisfecha, por lo cual también hace una proyección que debería tener la oferta para mantener el equilibrio. [3]

	Real		Estimado					
	2008	2009	2010	2011	2012	2013	2014	2015
Producción	12.497	12.817	15.786	19.443	23.947	29.495	36.327	44.743
Exportación	591	787	879	983	1.099	1.228	1.372	1.534
Venta Internas	9.174	11.839	15.067	19.176	24.405	31.061	39.531	50.311

Tabla 1: ***Oferta y demanda de trucha***

Fuente: Ministerio de la producción.

Para cubrir la demanda insatisfecha los sistemas de producción de trucha deberían contar con sistemas de monitoreo y control de parámetros para la crianza de truchas como son el nivel de concentración de oxígeno en el agua y temperatura entre otros, este viene a ser un problema en las localidades del departamento del Cusco donde se hace la crianza, ya que estas no cuentan con dichos sistemas.

Por otro lado, en el mes de marzo del año 2017 se desarrolló un "**Sistema Automatizado para el Control y Monitoreo del Comportamiento de Alevinos de Paiche en Cautiverio**" por parte de un proyecto de investigación e innovación tecnológica promovido por el vicerrectorado de investigación (VRIN) de la Universidad Privada de Pucallpa; donde se desarrolló un modelo de sistema basado en las tecnologías de la información, para la crianza y el manejo paiche en cautiverio. El proyecto contempló la implementación de sistemas de automatización y comunicación (red de sensores) y las pruebas del sistema integral. Los resultados y la evaluación económica indican que son favorables, alentadores y viables poner en marcha el proyecto a extensiones mayores, y en consecuencia el desarrollo industrial y/o económico en la zona. [4]

1.2.2. Problemática

El equilibrio entre la oferta y demanda de la trucha en los últimos años ha sufrido un desbalance debido al crecimiento de la velocidad de la demanda, y no se ha mejorado el proceso de producción para sostener este equilibrio; también se tiene que el 77.3% de la población del Cusco no reemplazaría la compra de la trucha por otra alternativa. Es debido a la gran demanda de trucha que se pronostica para los próximos años es que se requiere que la producción dentro del departamento del Cusco sea más eficiente y a mayor

escala, por lo que sería necesario monitorear parámetros dentro de su producción de trucha.

	TOTAL	LIMA METROLITANA	JUNIN	PUNO	CUSCO	AREQUIPA
	%	%	%	%	%	%
-No reemplazaria la compra de trucha	77,2	88,9	75,0	92,0	73,3	46,7
-Si reemplazaria la compra de trucha	22,8	11,1	25,0	8,0	26,7	53,3
-No consideran que la trucha tiene reemplazo	6,9	0,0	3,6	0,0	0,0	40,0
-Si consideran que la trucha tiene reemplazo por:	15,8	11,1	21,4	8,0	26,7	13,3

Tabla 2: Alternativa al consumo de trucha

Fuente: Ministerio de la producción.

Un parámetro fundamental como la temperatura del agua es muy importante porque regula el crecimiento y reproducción, debido a que la trucha no tiene la capacidad de regular su propia temperatura [6]. Entonces entre los factores que no permiten la mejora en el proceso de producción de trucha se puede indicar el deficiente monitoreo y en algunos casos el monitoreo de forma manual de la temperatura en cada deposito, estos pueden llegar a generar errores que afecten la calidad del agua en las piscifactorías y en consecuencia la producción de trucha.

El problema que se puede identificar por lo tanto en las piscifactorías de la cuenca del rio Lucre es el deficiente monitoreo del parámetro de temperatura que se realiza en algunas piscifactorías, y en otros casos no existe ninguna forma de monitoreo de temperatura, por lo que sería necesario que cuente con un sistema de monitoreo integral.

Por lo que para solucionar este problema se busca diseñar una red utilizando tecnologías inalámbricas que permita monitorear la temperatura de los estanques de forma simultánea, fiable y eficiente y que esto permita la mejora de la producción en lo sucesivo.

1.2.3. Justificación

Las piscifactorías dentro del distrito de Lucre ofrecen un gran potencial en la actividad de acuicultura dentro de la región del Cusco debido al recurso hídrico y el clima

que se tiene para la crianza de trucha en la zona. No obstante, la temperatura y la concentración de oxigeno son parámetros importantes en la reproducción y el crecimiento de la trucha como se ve en los antecedentes mencionados; y el deficiente control de estas variables que influyen en el proceso de producción, son debido a gran manera a que las mediciones de estos parámetros se realizan de forma manual y en algunos casos ni se realizan, estos procedimientos hacen que la calidad del agua no sea óptima en la producción necesaria para cubrir la demanda creciente. Por lo cual este trabajo pretende desarrollar un aporte a la solución mediante el diseño de una red de monitoreo inalámbrico haciendo uso de los dispositivos bluetooth y zigbee de bajo consumo de energía.

Este trabajo dentro de su desarrollo también contempla la validación del mismo, para la adquisición de valores adecuados de temperatura en los estanques de las distintas piscifactorías en la ribera del rio Lucre.

1.2.4. Objetivos

1.2.4.1. Objetivo general

Diseñar una red inalámbrica con dispositivos bluetooth y zigbee para el monitoreo de temperatura en la crianza de truchas de las piscifactorías del distrito de Lucre

1.2.4.2. Objetivos específicos

- Definir los sensores que formaran parte de la red para la monitorización de temperatura.
- Definir los diferentes módulos para la transmisión y recepción, explicando la elección de tecnologías inalámbricas como bluetooth y zigbee de bajo consumo de energía.
- Diseñar el sistema de red de comunicación inalámbrica.
- Diseñar el módulo terminal de sensado que se instalara en el estanque para la monitorización de temperatura.
- Diseñar, implementar y validar un prototipo a escala del sistema integral.

1.2.5. Alcances

- El proyecto comprenderá el análisis del parámetro de temperatura relacionado a la crianza de trucha.
- Se tendrá en cuenta la estricta compatibilidad entre las tecnologías inalámbricas utilizadas en el sistema.
- El diseño considera la interconexión de 10 nodos (piscifactorias) a un sistema central de monitoreo (municipio) en un área geográfica promedio de 3Km utilizando Zigbee. Y dentro de cada nodo se considera una distancia máxima de 100m utilizando Bluetooth 4.0 ULP.
- Se realizarán simulaciones de las mediciones de temperatura dentro de un sistema que se asemeje al de las piscifactorías, para sustentación de tesis y con fines de capacitación.
- Se realizará el diseño de los módulos terminales para la monitorización de temperatura, visualizando la temperatura en una pantalla LCD.

1.2.6. Limitaciones

- El presente proyecto está enfocado en el diseño de una red inalámbrica que permita monitorear la variable de temperatura dentro del sistema de piscifactorías que se encuentran únicamente en la ribera del rio Lucre. No así, para el humedal y/o laguna de Lucre.
- El módulo de sensado solo contempla el parámetro de temperatura debido al factor económico.
- El proyecto no contemplará el diseño de un sistema fotovoltaico de energía para la comunicación de los nodos hacia el centro de monitoreo.

Capítulo 2 : MARCO TEÓRICO

2.1. Antecedentes del problema

En proyecto de investigación e innovación tecnológica denominado: **SISTEMA AUTOMATIZADO PARA EL CONTROL Y MONITOREO DEL COMPORTAMIENTO DE ALEVINOS DE PAICHE EN CAUTIVERIO**, tuvo como objetivo principal, definir y ejecutar un modelo de sistema constructivo basado en las tecnologías de la información, que permita la crianza y el manejo de Arapaima gigas (paiche) en cautiverio, con ello se puede identificar parámetros como temperatura, oxígeno, pH, conductividad eléctrica (ce), elementos fundamentales en el proceso de producción del paiche, asimismo la implementación del sistema de cambio de agua y alimentación en forma automatizada. Los resultados y la evaluación económica indican que son favorables, alentadores y viables poner en marcha el proyecto a extensiones mayores, con la finalidad de desarrollar esta idea de negocio bajo el estricto sentido empresarial y de emprendimiento, empleando al máximo los instrumentos técnicos y económicos para contribuir a la consolidación de la paichicultura como actividad productiva, insertando de esta manera la trilogía Universidad-Empresa-Sociedad, mejorando de esta manera las posibilidades de desarrollo económico de nuestra región y posteriormente con la conformación de cadenas productivas a nivel de la cuenca del Ucayali, contribuir al PBI regional y nacional.[1] [4]

En el informe de estudio de mercado de la trucha en Arequipa, Cusco, lima, Huancayo y Puno (informe) ha sido elaborado por Maximixe Consult S.A para el uso exclusivo de la producción del ministerio de la producción con el objeto de conocer el balance entre la oferta y la demanda de la trucha en los próximos años en este estudio La producción de truchas en los últimos años es insuficiente debido a que la demanda ha ido incrementando. [5]

[1] Cultura viva amazónica-revista de investigación científica- Pucallpa,peru.2(1)2017

Dentro de estudios realizados también podemos mencionar el "**Estudio de Ingeniería para la Implementación de una Red Agroclimática Utilizando Tecnología WI-FI para el Distrito De Huayllabamba**" trabajo de tesis realizado el año 2012 por los Sres. Alexander Palomino López y Harry Martínez Sueros dentro de la Escuela Profesional de Ingeniería Electrónica, donde se hace uso de tecnologías inalámbricas robustas como WI-FI para la comunicación de las áreas agrícolas monitoreadas hacia una base de datos central y luego la información enviada hacia un Portal Web informativo [7]; y el trabajo "**Diseño de un Sistema de Monitoreo a Distancia de Signos Vitales de Pacientes en la Clínica Internacional SOS GROUP**" desarrollado el año 2010 por los Sres. Víctor Andrés Ayma Quirita y Darwin Auccapuri Quispetupa dentro de la Escuela Profesional de Ingeniería Electrónica, desarrollando un sistema de telemetría para monitoreo de Signos Vitales en pacientes de una Clínica. [8]

2.2. Bases teóricas.

En la crianza de truchas los sistemas de adquisición de datos es un elemento crucial para la automatización y mejoramiento de procesos.

El diseño de estos sistemas de medición de temperatura es importante debido a los avances de las comunicaciones en los últimos años, se ha logrado incrementar la atención en el desarrollo de redes sensores inalámbricas de bajo costo y bajo consumo de energía.

2.2.1. Red de sensores inalámbricos

Una gran cantidad de pequeños dispositivos, autónomos, distribuidos físicamente, llamados nodos de sensores, instalados alrededor de un fenómeno para ser monitoreado, con la capacidad de almacenar y comunicar datos en una red en forma inalámbrica.[2]

[2] academica-e.unavarra.es/bitstream/handle/.../TFG_IraceburuGonzalezJulen2014.pdf?...1

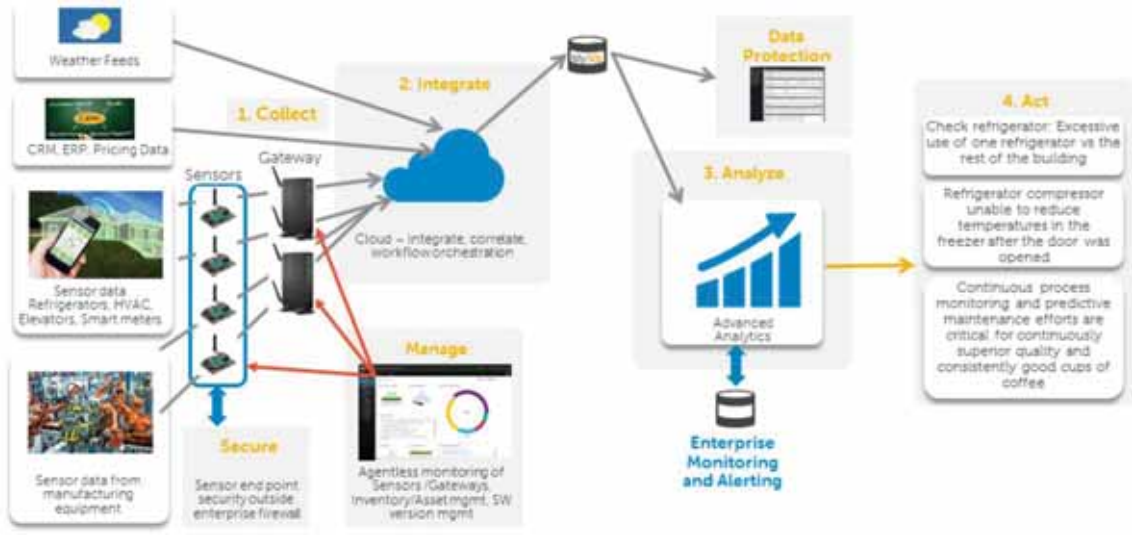


Figura 2: ***Red de sensores para procesos de monitoreo***

Fuente: en.community.dell.com

2.2.2. Bluetooth ULP (Ultra Low Power).

Es un estándar de tecnología inalámbrica para el intercambio de datos a cortas distancias (empleando un enlace por radiofrecuencia en la banda ISM de 2,4 GHz) desde dispositivos fijos y móviles construyendo redes de área personal (PAN) [12]. Utiliza canales RF (79 canales) de f = 2402 + k MHz siendo k = 0.78. El espacio entre los mismos es de 1 MHz más unas bandas de guarda. La distancia nominal del enlace está comprendida entre 10cm y 10m, pero aumentando la potencia de transmisión se puede llegar a 100m. La modulación empleada por Bluetooth es GFSK (Gaussian Frequency Shift Keying) con un producto ancho de banda por tiempo BT = 0.5 y un índice de modulación entre 0.28 y 0.35. Estos dispositivos se clasifican en clases: clase 1, clase 2 y clase 3, en referencia a su potencia de transmisión.[3] La versión utilizada para el presente trabajo será Bluetooth 4.0 ULP (Ultra Low Power) que tiene mejoras en consumo de energía con respecto a otras versiones, y es adecuado para operaciones bajo las condiciones requeridas.

[3] http://academica-e.unavarra.es/bitstream/handle/2454/11846/TFG_IraceburuGonzalezJulen2014.pdf?sequence=1

Figura 3: ***Modulo de transmisión Bluetooth 4.0 ULP***

2.2.3. ZigBee

Es un conjunto de protocolos de alto nivel de comunicación y que trabaja en topologías malla. Y, se utiliza para la radiodifusión digital de datos buscando ahorrar lo máximo posible en energía. Una tecnología basada en el estándar de la IEEE como es la IEEE 802.15.4.

La tecnología de comunicación inalámbrica ZigBee utiliza la banda ISM y por lo general, adopta la banda 2.4GHz para comunicarse con el resto de dispositivos ya que esta se adopta en todo el mundo.[4]

[4] https://elandroidelibre.elespanol.com/2015/08/todo-sobre-zigbee-la-tecnologia-ultrabarata-para-comunicacion-inalambrica.html

Figura 4: ***Modulo de transmisión Zigbee***

2.2.4. Los sensores de temperatura

Se utilizan en diversas aplicaciones tales como aplicaciones para la elaboración de alimentos, climatización para control ambiental, dispositivos médicos, manipulación de productos químicos y control de dispositivos en el sector automotriz (p. ej., refrigerantes, ingreso de aire, temperaturas del cabezal de cilindro, etc.). Los sensores de temperatura se utilizan para medir el calor para asegurar que el proceso se encuentre, o bien dentro de un cierto rango, lo que proporciona seguridad en el uso de la aplicación, o bien en cumplimiento de una condición obligatoria cuando se trata de calor extremo, riesgos, o puntos de medición inaccesibles. [5]

Hay dos variedades principales: sensores de temperatura con contacto y sin contacto. Los sensores de contacto incluyen termopares y termistores que hacen contacto con el objeto a medir, y los sensores sin contacto se encargan de medir la radiación térmica emitida por una fuente de calor para determinar su temperatura. Este último grupo mide la temperatura a distancia y a menudo se utilizan en entornos peligrosos.

[5] https://www.digikey.com/es/articles/techzone/2011/oct/temperature-sensors-the-basics

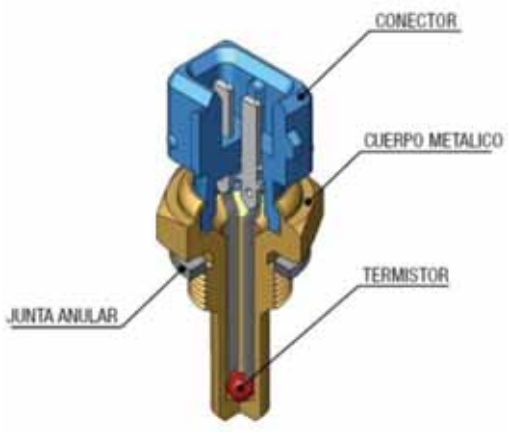


Figura 5: ***Modulo de sensado de temperatura***

La elección del sensor de temperatura se realizará contemplando las condiciones como son el de estar sumergido dentro de la poza de cultivo y además el rango de operación.

2.2.5. El microcontrolador (µC, UC o MCU).

Es un circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria. Está compuesto de varios bloques funcionales, las cuales cumplen una tarea específica programada. Un microcontrolador incluye en su interior las tres principales unidades funcionales de una computadora: Unidad central de procesamiento, memoria y periféricos de entrada/salida.

Figura 6: ***Microcontrolador (µC)***

2.3. Método

El método utilizado en el presente trabajo es el ***método descriptivo***, ya que el objetivo es diseñar un sistema que permita monitorear la temperatura dentro del ámbito geográfico del distrito de Lucre.

2.4. Tipo de trabajo

El presente trabajo se encuentra clasificado dentro del uso ***de la tecnológica aplicada,*** ya que busca aportar en la solución a una problemática que es el de la producción de trucha en la región, mediante el desarrollo de un sistema de monitorización de temperatura haciendo uso de las tecnologías de telecomunicaciones inalámbricas de bajo consumo de energía.

2.5. Proceso de desarrollo

- Recopilación de información necesaria para el diseño del sistema de comunicación y sensado.
- Comparación de las diferentes tecnologías inalámbricas de bajo consumo como Bluetooth y Zigbee frente a tecnologías como Wifi y Wimax para la aplicación propuesta.
- Diseño del módulo terminal de medición del parámetro de temperatura (sensores, microcontrolador y tarjeta de comunicación).
- Diseño de la red inalámbrica de comunicación y centralización.
- Validación del módulo terminal de monitoreo.

Diseño de una red inalámbrica con dispositivos bluetooth y zigbee para el monitoreo de temperatura en la crianza de truchas de las piscifactorías del distrito de lucre.

Capítulo 3 : CARACTERÍSTICA DE LA CUENCA HIDROGRÁFICA DEL RIO LUCRE Y LA PRODUCCIÓN DE TRUCHA

3.1. Cuenca del rio Lucre

Se encuentra localizada al SE. de la cuenca hidrográfica del río Watanay, abarcando los distritos de Lucre y Oropeza de la Provincia de Quispincanchis del Departamento del Cusco. Sus coordenadas geográficas son: I3°37'09"S y 71°41 '40"W. En términos altitudinales, cubre un territorio desde los 4489 m (Señal Pantapuncu) en el cerro Condorsayaña hasta los 3072 (Huacarpay); por tanto, cubre los pisos ecológicos de Queswa, Transición, y Puna de Olarte y Dollfus; Queswa, Suni y Puna de Pulgar Vidal. [11]

Se llama Cuenca del Río Lucre, por ser el Río Lucre, el de mayor jerarquía, el colector común de los torrentes de este sistema de drenaje que pasa por la localidad de Lucre, capital del distrito del mismo nombre, a 3086 m de altitud. La superficie total de la cuenca asciende a 103 km2, que representan el 20.5 % del área total de la cuenca del río Watanay.

Tiene forma triangular invertida, con la base hacia el norte, y los lados hacia SE y SW y con vértice en el lugar denominado Condorsayaña, donde se encuentra la señal de Pantapuncu que es la elevación más alta de la cuenca .Sus límites están dados por el Norte con la cuenca del río W atanay, del que la separa la cadena de montañas denominadas genéricamente cerros de Sayhua, Torioc, Sinchijonernioc, Joricalla y Corihuayrachina; por el Sur con la cuenca del río Paruro, separada por las elevaciones de Condorsayaña, Luichos y Campanayoc; por el Este con la cuenca del Río Marijó (Andahuaylillas), afluente del río Urubamba, separada por las elevaciones de Pucacasa, Comba y oc y Sayhua. Finalmente, por el Oeste con la cuenca del Río Choco que forma el río Watanay, separada por las elevaciones de Patococha, Chaquicocha y Toctobamba.

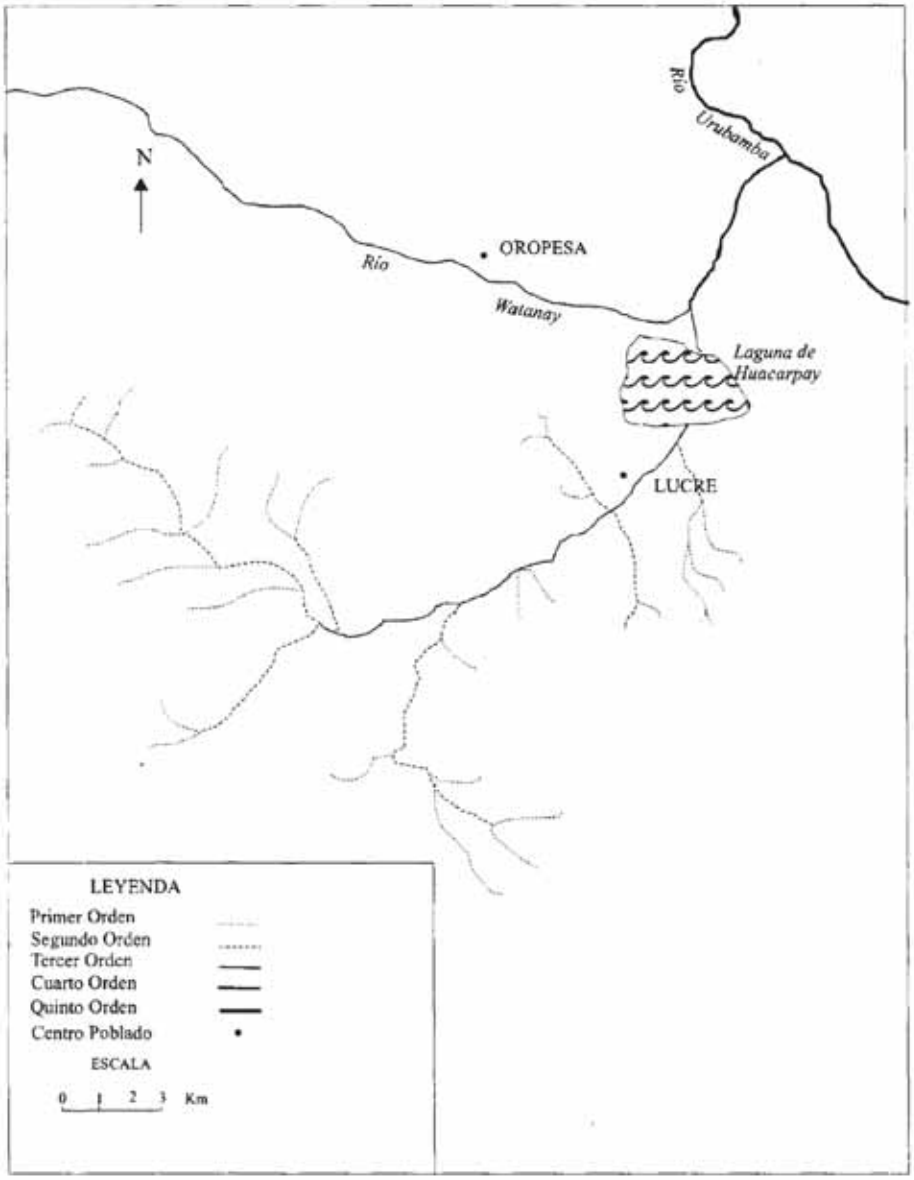


Figura 7: ***Cuenca hidrográfica del rio Lucre***

3.2. Rio Lucre

Se origina en los riachuelos de Pacramayo al pie del cerro Toctohuampa a 4,050 m de altitud, a 13°37'58" S y 71 °49'39" W. Su recorrido total es de 16 km y se pueden identificar tres sectores:

a) Curso Superior, que corresponde a la parte alta del río, desde su origen hasta la afluencia del río Cullumayo. Recorre por encima de 3500 m de altitud paralelo al río Watanay de WNW a ESE. Tiene 6 km de longitud, formando una estrecha quebrada.

b) Curso Medio; que comprende desde la confluencia del río Cullumayo hasta la localidad de Lucre. Recorre 7 kms de SW a NE formando un estrecho valle en forma de V.

c) Curso Bajo; que recorre desde la localidad de Lucre hasta su desembocadura en el río Watanay, formando parte de un valle que gradualmente se ensancha en Huacarpay, confundiéndose con la Laguna del mismo nombre. Tiene aproximadamente 3 km de longitud. En un recorrido con dirección S a N.

Este río es alimentado por 10 afluentes por la margen izquierda y 12 por la margen derecha de primero y segundo orden. Los más importantes son: Por la margen izquierda: los ríos: Cantarán y Perajarán. Por la margen derecha: los ríos: Sinhuarán, Yutujasamayo, Cullumayo y Ullpo.

3.3. Longitud del Río principal

El río Lucre tiene 16 km de longitud y es el de mayor jerarquía y colector común de las aguas de la cuenca. Su longitud principal es distinta de la longitud de la cuenca. Tiene importancia para determinar la gradiente del canal y por consiguiente la velocidad de desplazamiento de las aguas en la cuenca.

Longitud total de los ríos. (Lt). Está dada por la suma de longitudes de los cursos de aguas de distintas órdenes. Este parámetro asciende a 96.0 km. Para mayor objetividad se muestra el cuadro:

Orden	Número	Longitud (km)
1°	36	65.1
2°	07	20.9
3°	01	09.0
Total	44	96.0

Tabla 3: ***Longitud de los torrentes del rio Lucre***

Las longitudes totales de los ríos por márgenes resultan 49.1 km para la margen derecha, 31.2 km para la margen izquierda y 16.0 al canal principal.

3.4. Producción de Trucha en la cuenca del rio Lucre

El río Lucre tiene 16 km de longitud y es a las orillas y en el tramo que se muestra en la figura donde se ubican la mayor cantidad de piscifactorías entre el centro poblado de Lucre y primeros tramos de su origen. Cada una de las piscifactorías artesanal (contiene en promedio 3 piscigranjas) provee al dueño para atender el servicio de restaurantes campestres donde se expende platos elaborados con el elemento principal producido que es la trucha. Aproximadamente existen más de 30 de las cuales para este diseño de consideran las siguientes piscifactorías artesanales en el rio Lucre como se observa en la siguiente tabla:

Nombre de la Piscigranja	Coordenadas
LA RINCONADA II	13°39'4.48"S 71°45'30.59"O
LAS ORQUIDEAS	13°39'3.86"S 71°45'28.15"O
SURI	13°39'2.31"S 71°45'27.00"O
LOS MANANTES	13°39'2.47"S 71°45'24.71"O
VIRGEN DEL CARMEN	13°39'1.78"S 71°45'23.75"O
LA ESCONDIDA DE ADAN	13°38'58.67"S 71°45'23.14"O
LA VICTORIA	13°38'59.61"S 71°45'22.05"O
RINCONADA I	13°38'59.10"S 71°45'21.63"O
FLOR DE CAPULI	13°38'58.21"S 71°45'21.27"O
ROSAS KANCHA	13°38'57.65"S 71°45'22.20"O
SR. QOYLLORITI	13°38'56.15"S 71°45'20.81"O
EL CALLEJON	13°38'55.21"S 71°45'19.97"O
ALTO BATAN	13°38'53.44"S 71°45'18.13"O
SAN JOSE	13°38'51.74"S 71°45'16.45"O
FRUTALES	13°38'49.89"S 71°45'13.14"O
LA ESPERANZA	13°38'50.19"S 71°45'8.24"O
FORTALEZA	13°38'52.17"S 71°45'4.99"O
CRISTO REY	13°38'52.26"S 71°45'13.59"O

Tabla 4: ***Piscigranjas a la Ribera del Rio Lucre (en dirección a la laguna)***

Figura 8: *Distribución aproximada de piscifactorías en la cuenca del rio Lucre*

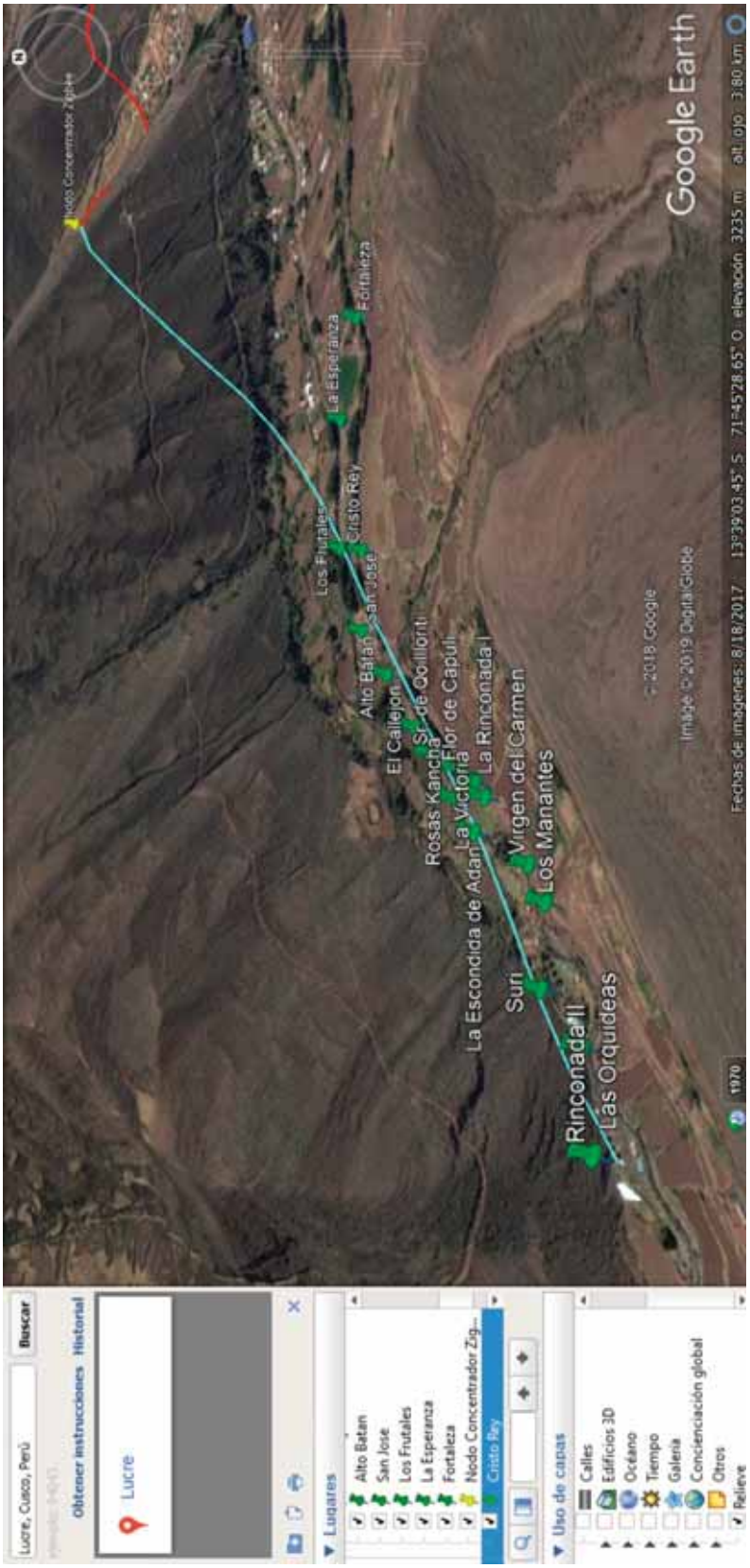


Figura 9: *Nodos Monitoreo – Nodo Concentrador Zigbee (Nodo + lejano – Concentrador 1.5 Km)*

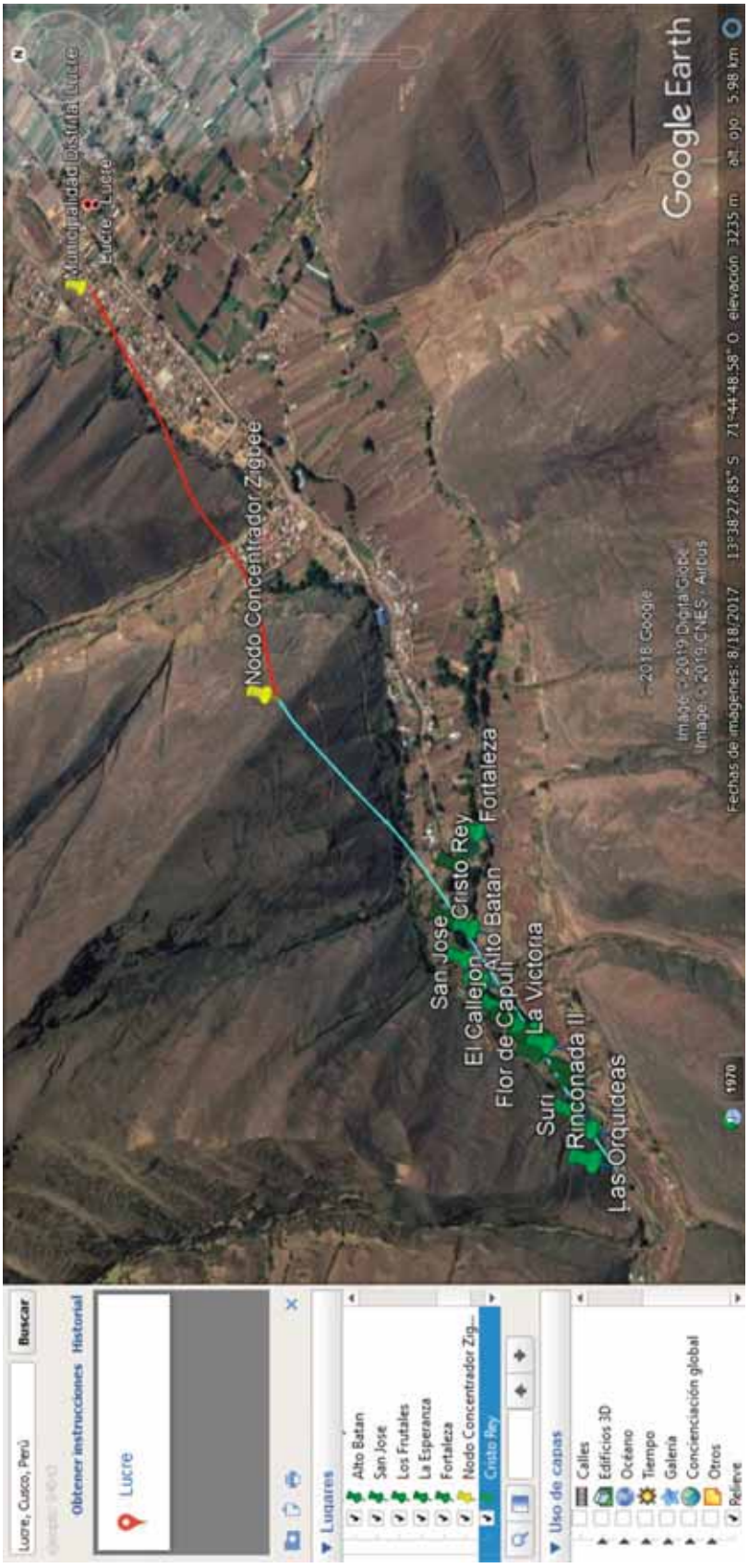


Figura 10: ***Interconexión Nodo Concentrador – Nodo Municipio (1.58Km)***

Figura 11: ***Piscifactoría artesanal al borde del rio Lucre***

Figura 12: ***Pozas o estanques de crianza de la trucha***

Figura 13: ***Funcionamiento de Restaurantes dentro de la cuenca del rio Lucre***

El objetivo de esta tesis consiste en diseñar un sistema inalámbrico de sensores que permitirá monitorear la temperatura en los estanques para el proceso de crianza de truchas. Para lograrlo, es importante poseer un dispositivo bluetooth por cada estanque. Además, se contará con un sensor de temperatura que irá conectado a cada dispositivo bluetooth. Cada una de las pozas o estanques (3 que conforman la piscigranja) se comunican con un dispositivo concentrador bluetooth que permitirá mostrar la información al usuario (dueño de la piscigranja). Esto permite un monitoreo en un rango de 30m de radio y sirve únicamente al usuario para las acciones que prevea realizar. Luego cada módulo concentrador Bluetooth (cada piscigranja) contiene un dispositivo zigbee que permitirá conectar a una central que se encuentra en el centro poblado de Lucre – municipio. Es de esa manera que se concentrara toda la información de todas las piscifactorías artesanales en un solo lugar y es ahí donde se pretende implementar un servidor para acceder la información vía Web.

En la tabla siguiente se realiza la Comparación de las diferentes tecnologías inalámbricas de bajo consumo como Bluetooth y Zigbee frente a tecnologías como Wifi y Wimax para la aplicación propuesta.

Standard	Bluetooth (BR/EDR)	Bluetooth Low Energy	UWB	Zigbee	Wi-Fi	WiMAX
IEEE specification	802.15.1	802.15.1	802.15.3	802.15.4	802.11	802.16
Versions	V 1.2 V 1.2 V 2.0 + EDR V 2.1 + EDR V 3.0 + HS	V4.0	MB-OFDM-UWB SD-UWB TH-UWB FM-UWB UWB-MIMO	0x01 @2004 0x02 @2006	802.11 @WLAN 802.11a @WLAN 802.11b @WLAN 802.11g @WLAN 802.11p @ Vehicular 802.11e @ QoS 802.11f @IAPP 802.11h @5GHz 802.11i @Encryption 802.11nUn@MIMO *1	802.16b-2004 802.16e (4G, mobile phones)
Frequency band	2.4 GHz	2.4 GHz	3.1-10.6 GHz (U.S) 7.5 GHz Unlicensed 6-8.5 GHz (Europe) 3.4-4.8 GHz(Japan)	868/915 MHz, 2.4 GHz	2.4 GHz @11g 5 GHz @11a	2-11 GHz 3.5 GHz (license) 5.8 GHz (unlicensed) 2.3-2.5 GHz (802.16e)
Data Rate	721.2 Kb/s V1.2 2.1Mb/sV2.0+EDR 24Mb/s V3.0+ HS*1	1 Mb/s	500Mb/s @ MB Up to 1320 Mb/s @ SD	250Kb/s @2.4GHz 40Kb/s @915MHz 20Kb/s @868MHz	1-2Mb/s@11-2.4GHz 54Mb/s@11a-5GHz 11Mb/s@11b-2.4GHz 54Mb/s@11g-2.4GHz	75 Mb/s @ 2.3GHz – 3.5 GHz 6Mb/s mobile devices
Covered range	Class 1 - 100m Class 2 - 10m Class 3 - 1m	1m	1-10 m	10-100m	30m @11a-5GHz 100m @11b-2.4GHz 100m @11g-2.4GHz	8-32Km 2.3GHz – 3.5 GHz Up to 50 Km 8-10K mobile devices. Speed of up to 120Km/h
Transmission power	Class 1 - 20dBm Class 2 - 4dBm Class 3 - 0dBm	-20 dBm + 10 dBm	< 0 dB	0dBm	15-20dBm	

Tabla 5: Comparación de Tecnologías Inalámbricas

[6]http://bibing.us.es/proyectos/abreproy/70218/fichero/2.Tecnolog%C3%ADas+Inal%C3%A1mbricas.pdf

Capítulo 4 : DISEÑO E IMPLEMENTACION DEL PROTOTIPO DE LA RED INALÁMBRICA

En este capítulo se describe todo el procedimiento realizado en el diseño y la implementación del prototipo de la red de medida de temperatura basada en BLE y Zigbee, para ello se define la topología a implementarse como se muestra:

4.1. Topología general de la red

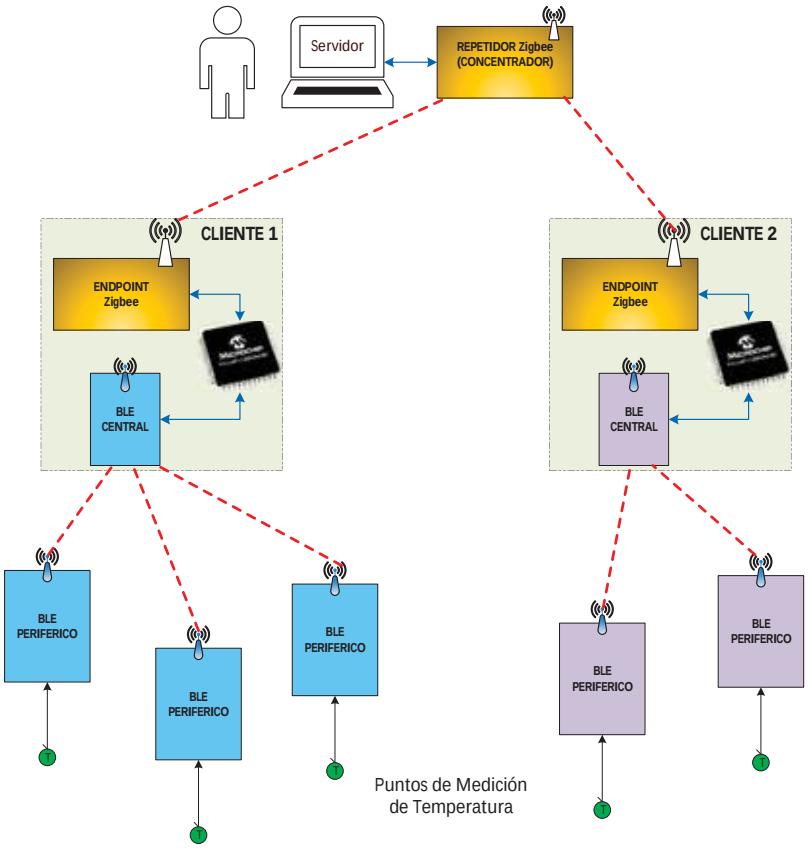


Figura 14: ***Topología general de la red a implantar considerando 2 nodos clientes.***

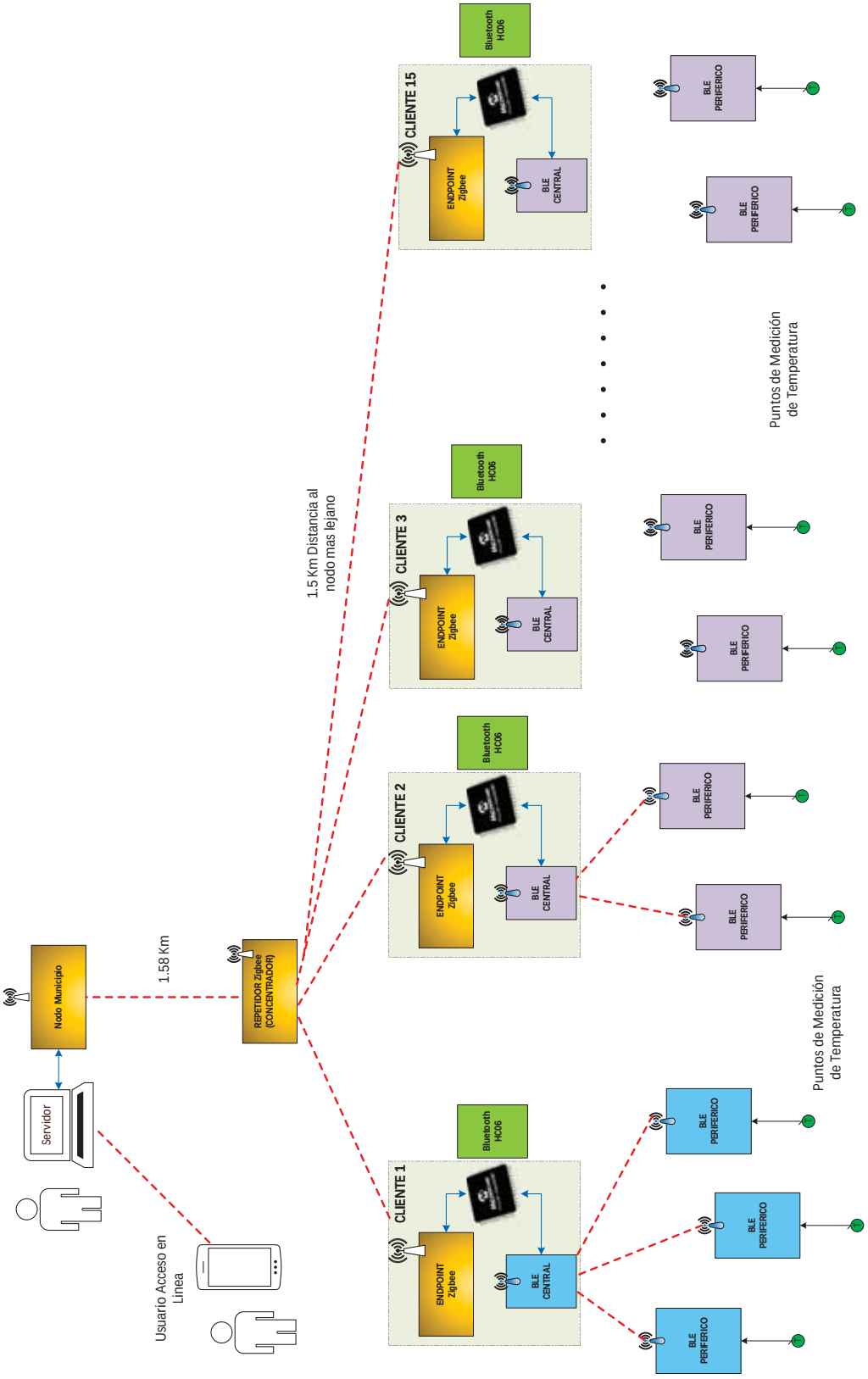


Figura 15: ***Topología de la Red Completa***

La figura 14 muestra la red Zigbee y la red Bluetooth implementada, en ella se observa que el transporte de toda la información de temperatura adquirida en los nodos cliente esta implementada sobre dispositivos Zigbee, la misma que consiste de un repetidor y dos terminales finales; los terminales finales permiten trasladar la información recolectada hacia el repetidor que a la vez se conecta al nodo Municipio que sirve de servidor de la red conjuntamente con el software implementado en LabVIEW. Por otra parte, la red de sensores implementada sobre Bluetooth permite adquirir las medidas de temperatura de las pozas de trucha, Esta red se ha configurado con una central y tres periféricos para este trabajo de tesis. La central BLE permite emparejarse a cada periférico BLE, estos últimos están configurados mediante un script que obtiene y almacena la temperatura en forma autónoma en el registro 0x0E de la memoria del mismo periférico BLE.

4.1.1. Conexión Punto a Punto (Nodo Concentrador Zigbee – Nodo Municipio)

Como se observa en la topología, existe un nodo concentrador intermedio entre los clientes y el nodo servidor Municipio esto se debe a las características geográficas del lugar que impiden tener una conectividad directa hacia el nodo servidor por no tener una línea de vista; se considera pues así que este nodo concentrador (AP – Punto de Acceso) se ubique en las coordenadas: Lat. 13°38'32.95"S y Long. 71°44'54.28"O y las coordenadas del nodo municipio es Lat. 13°38'11"S y Long. 71°44'09"O con una elevación de 3307 msnm como se observa en la figura 16, teniendo una potencia de transmisión desde el nodo concentrador zigbee de 19dBm. Se utilizó, para este análisis de radioenlace el aplicativo del fabricante RADIO MOBILE que nos permite observar las características del enlace punto a punto entre los nodos en estudio.

ANALISIS DE COBERTURA nodo Concentrador al Municipio de Lucre

- La distancia entre nodo concentrador y municipio es 1.5 km (0.9 miles)
- Azimut norte verdadero = 63.48°, Azimut Norte Magnético = 69.07°, Angulo de elevación = -9.4604°
- Variación de altitud de 246.2 m
- El modo de propagación es línea de vista, mínimo despeje 2.9F1 a 1.5km

- La frecuencia promedio es 2450.000 MHz
- Espacio Libre = 103.9 dB, Obstrucción = -0.3 dB, Urbano = 0.0 dB, Bosque = 0.0 dB, Estadísticas = 6.7 dB
- La pérdida de propagación total es 110.3 dB
- Ganancia del sistema de nodo concentrador a municipio es de 127.6 dB (dipole.ant a 63.5° ganancia = 4.8 dB)
- Ganancia del sistema de municipio a nodo concentrador es de 127.6 dB (dipole.ant a 243.5° ganancia = 4.8 dB)
- Peor recepción es 17.3 dB sobre el señal requerida a encontrar

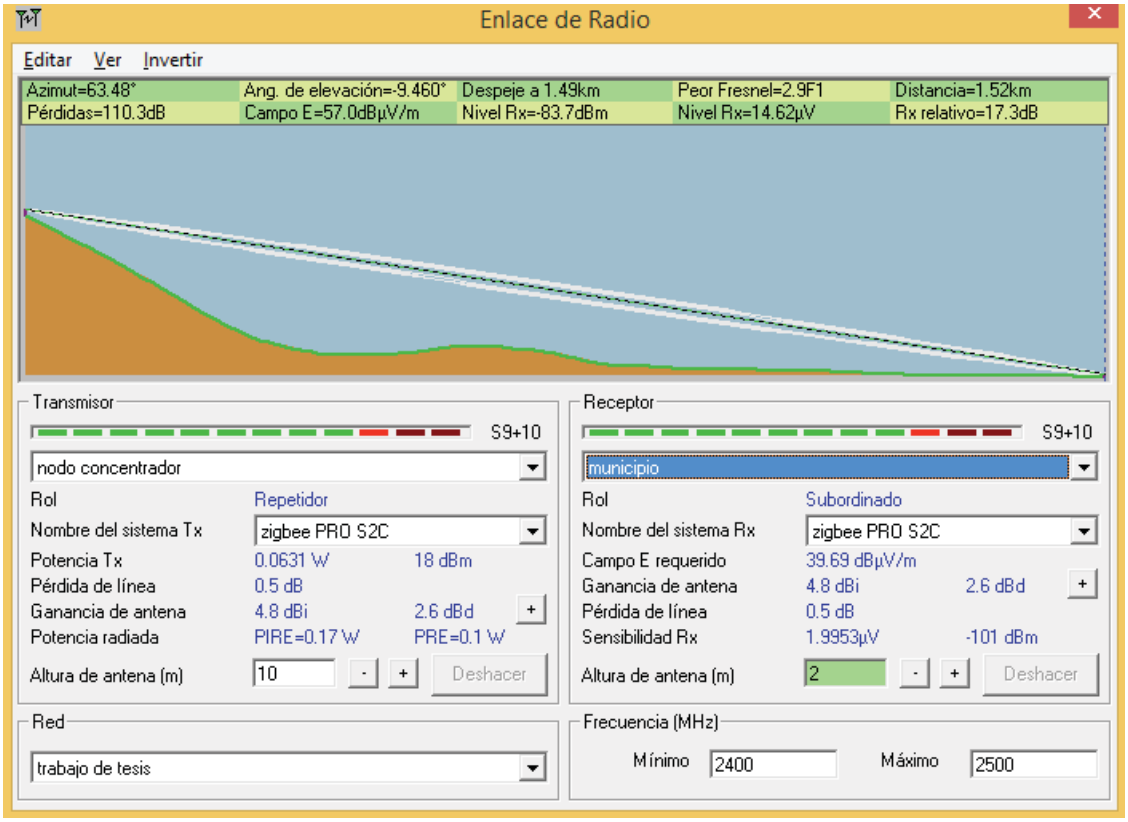


Figura 16: *Esquema del radioenlace nodo concentrador-municipalidad*

Figura 17: ***AP (Nodo Concentrador Zigbee) – CPE (Municipio)***

4.1.2. Conexión Punto a Multipunto (Nodo Concentrador Zigbee – Nodos Cliente)

Para este análisis se considera el nodo cliente más alejado y que presenta un obstáculo por las características del terreno, también la potencia de los transmisores zigbee se mantienen en 19dBm; el nodo más alejado denominamos Nodo Cliente Rinconada II (CPE – Equipo local cliente) como se muestra en la figura:

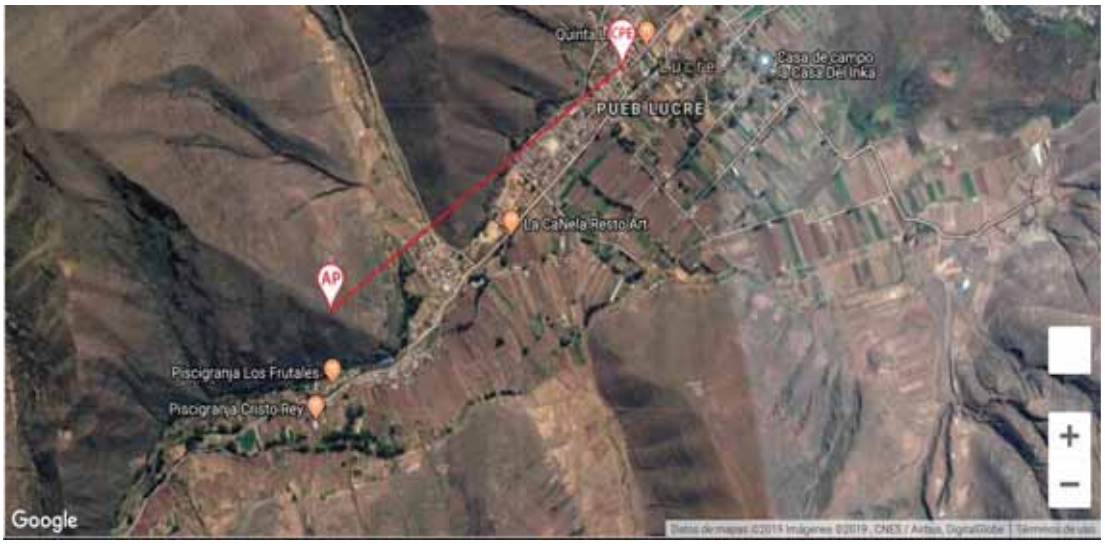


Figura 18: ***AP (Nodo Concentrador Zigbee) – CPE (Rinconada II)***

ANALISIS DE COBERTURA La rinconada II a nodo Concentrador

- La distancia entre la rincona y nodo concentrador es 1.5 km (0.9 miles)
- Azimut norte verdadero = 48.22°, Azimut Norte Magnético = 53.80°, Angulo de elevación = 4.7844°
- Variación de altitud de 170.9 m
- El modo de propagación es línea de vista, mínimo despeje 2.9F1 a 0.0km
- La frecuencia promedio es 2450.000 MHz
- Espacio Libre = 103.5 dB, Obstrucción = 0.4 dB, Urbano = 0.0 dB, Bosque = 0.0 dB, Estadísticas = 6.7 dB
- La pérdida de propagación total es 110.6 dB
- Ganancia del sistema de la rincona a nodo concentrador es de 127.8 dB (dipole.ant a 48.2° ganancia = 4.9 dB)
- Ganancia del sistema de nodo concentrador a la rincona es de 127.8 dB (dipole.ant a 228.2° ganancia = 4.9 dB)
- Peor recepción es 17.3 dB sobre el señal requerida a encontrar

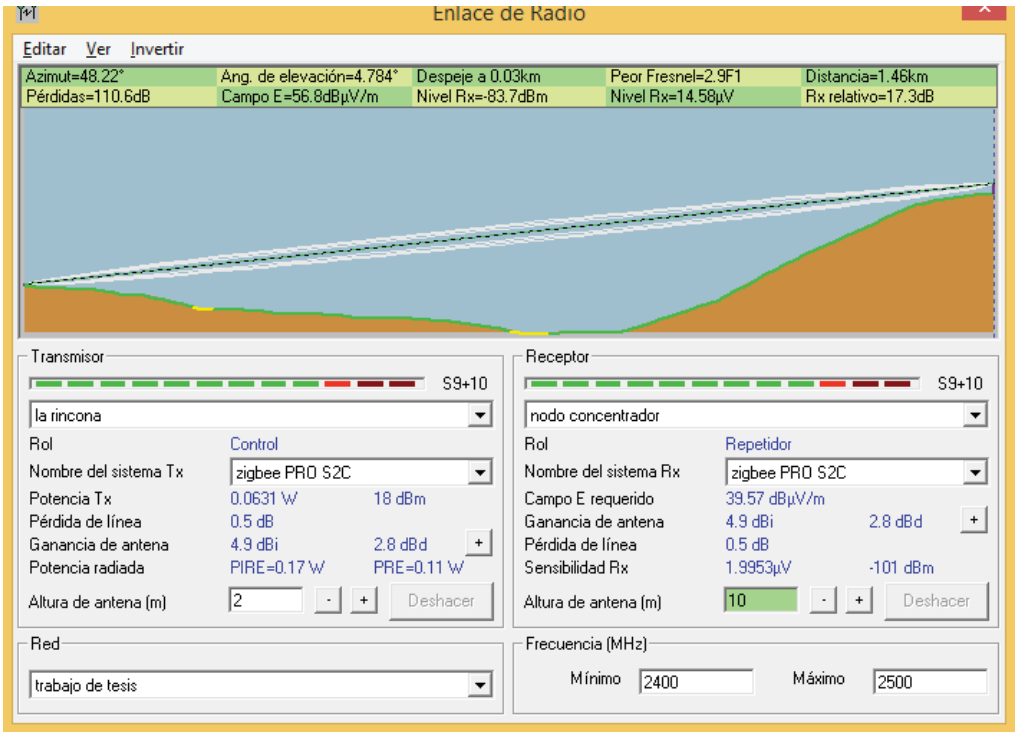


Figura 19: ***Esquema del radioenlace nodo rinconada II-concentrador***

ANALISIS DE COBERTURA la Rinconada de Adan a nodo Concentrador

- La distancia entre Escondida de Adan y nodo concentrador es 1.2 km (0.7 miles)
- Azimut norte verdadero = 47.51°, Azimut Norte Magnético = 53.09°, Angulo de elevación = 7.1722°
- Variación de altitud de 171.5 m
- El modo de propagación es línea de vista, mínimo despeje 2.6F1 a 1.1km
- La frecuencia promedio es 2450.000 MHz
- Espacio Libre = 101.7 dB, Obstrucción = -0.8 dB, Urbano = 0.0 dB, Bosque = 0.0 dB, Estadísticas = 6.7 dB
- La pérdida de propagación total es 107.6 dB
- Ganancia del sistema de Escondida de Adan a nodo concentrador es de 127.8 dB (dipole.ant a 47.5° ganancia = 4.9 dB)
- Ganancia del sistema de nodo concentrador a Escondida de Adan es de 127.8 dB (dipole.ant a 227.5° ganancia = 4.9 dB)
- Peor recepción es 20.2 dB sobre el señal requerida a encontrar

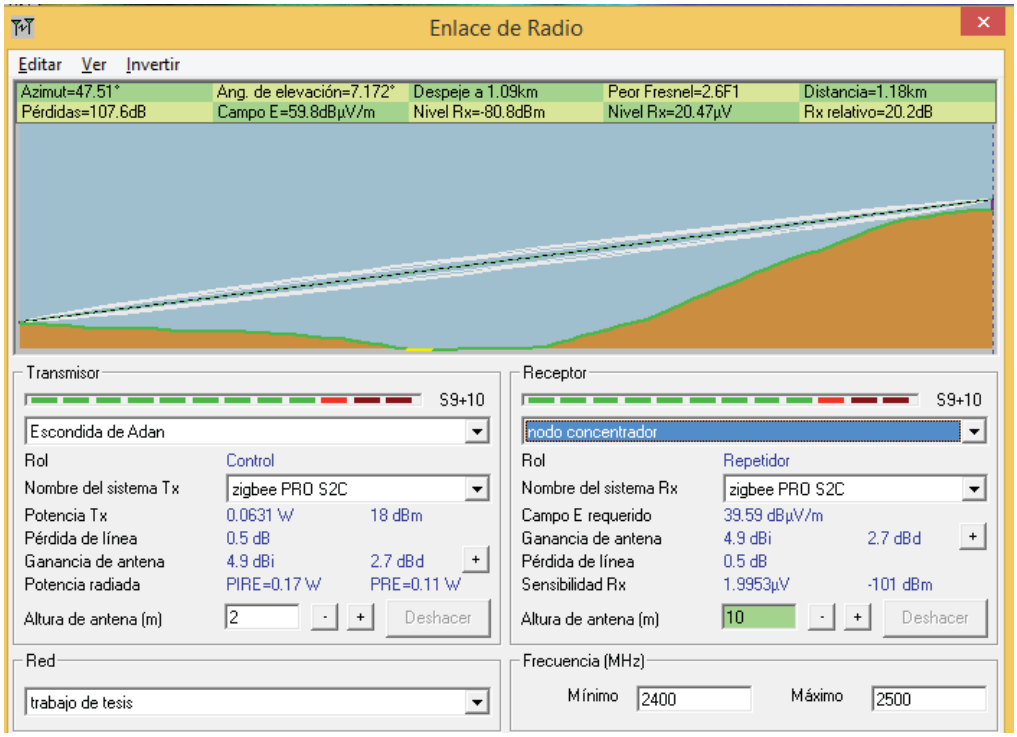


Figura 20: ***Esquema del radioenlace nodo escondida de Adan-concentrador***

4.2. Selección de dispositivos

Siendo el escenario el distrito de lucre, debemos ubicarnos primeramente en los siguientes escenarios:

- Se tienen pozas de distanciadas entre 5 y 8 metros entre sí, además el controlador del cliente requiere energía eléctrica por lo que dicho controlador estará ubicado a más de 10 metros generalmente. Y como se desea realizar una red Bluetooth entre las pozas de truchas, se requiere Bluetooth Low Energy v.4 o superior para poder transmitir sin problemas debido a que estos dispositivos pueden transmitir hasta 100 metros en comparación a las otras versiones Bluetooth que solo alcanzan los 9 metros, motivo por el cual se eligió el BLE RN4020 de la empresa Microchip, cuyas características se adjuntan en el Anexo en su respectiva hoja de datos del fabricante.

Figura 21: ***Modulo Bluetooth Low Energy RN4020***

- La data recolectada de cada cliente debe ser trasportadas hacia el servidor y muy probablemente a futuro administradas por la municipalidad, además no es dable la posibilidad de instalar una torre para tener línea de vista entre los dispositivos ZigBee, por lo que se ha utilizado la versión S2C de los dispositivos Xbee, los mismos que tienen un alcance máximo de 3.2 Km, compatible con antenas tipo cable, y antenas compatibles con conectores de radiofrecuencia U.FL y RPSMA.

Figura 22:16 ***Modulo Zigbee (XBee PRO S2C)***

- Respecto al controlador del cliente, se requieren dos puertos UART y un puerto adicional para futuras mejoras al módulo cliente. Realizando la búsqueda de un microcontrolador que tenga tres o más puertos UART, se eligió el PIC24FJ128gb204 de 16 bits de la empresa Microchip por disponer de 4 puertos UART, oscilador interno de 8 Mhz que permite realizar operaciones de hasta 16 MIPS, cuenta con 5 Temporizadores de 16 bits, y sobre todo que soporte la herramienta MPLAB Code Configurator, el mismo que permite realizar la configuración de los registros internos del microcontrolador de manera amigable y sencilla.

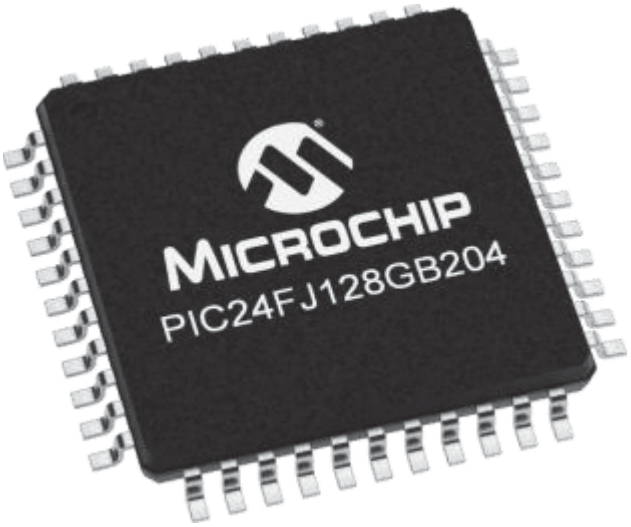


Figura 23: ***Microcontrolador de 16 bits PIC24FJ128gb204***

- En relación al software se eligió el software LabVIEW por la modularidad que posee para desarrollar aplicaciones y herramientas software en forma rápida y además porque actualmente la escuela profesional de ingeniería electrónica posee licencia académica.

Figura 24:17 ***Software Labview 2017***

4.3. Arquitectura de la red implementada

El diagrama de Bloques del prototipo de la red Implementada se muestra en la figura 28, en ella se observa claramente las direcciones físicas tanto de la red Zigbee como de la red BLE. Los cuadros de amarillo representan a los dispositivos Zigbee debidamente configurados mediante el software del mismo fabricante denominado DIGI XCTU. El Concentrador recoge toda la data de los dispositivos terminales para que el servidor almacene y grafique la información de las temperaturas obtenidas.

Los cuadros en azul representan la red Bluetooth implementada mediante los dispositivos RN4020, en la cual se requiere una central para emparejarse con los diferentes periféricos de la red y solicitar la temperatura adquirida de las pozas de truchas. El Procedimiento de Configuración de ambas redes, lo detallaremos a continuación.

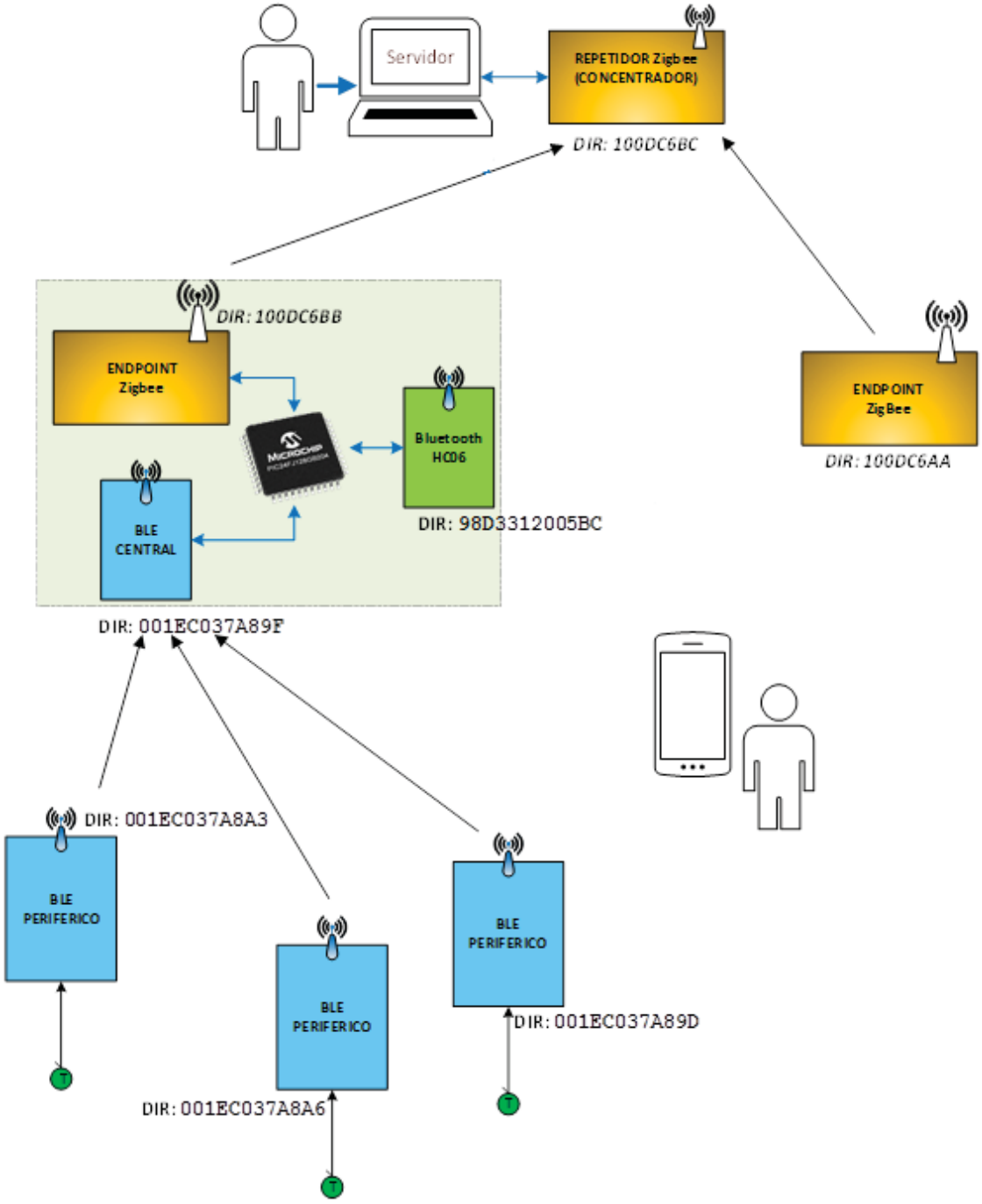


Figura 185: ***Diagrama general de la red implementada***

Figura 26: ***Software DIGI XCTU para configuración de Modulos Zigbee***

4.4. Proceso de configuración de los dispositivos inalámbricos

4.4.1. Configuración de los Bluetooth Low Energy (BLE).

Para Configurar este dispositivo, es necesario primeramente colocar en modo comando. Según la hoja de datos del fabricante del dispositivo bluetooth, se requiere conectar mediante un puerto serie al dispositivo RN4020 (UART_TX, UART_RX), así como poner a nivel lógico "1" el pin 7 (WAKE_SW) para colocar en modo comando al módulo RN4020, en la figura 30 podemos ver el pinout respectivo y la descripción de cada pin se detalla en la página cuatro de la respectiva hoja de datos del fabricante que acompaña este volumen en la parte de los anexos.

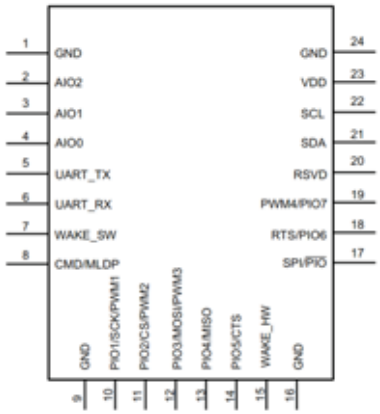


Figura 27: ***Pinout del RN4020***

Asimismo, tal como se muestra en su diagrama de bloques interno del RN4020, en la figura 31, este módulo posee un Convertidor analógico a digital (ADC) así como

soporta Scripts (**Archivo de órdenes)**. Se ha de utilizar estas características para implementar sobre el mismo RN4020, la adquisición de temperatura de los criaderos de truchas.

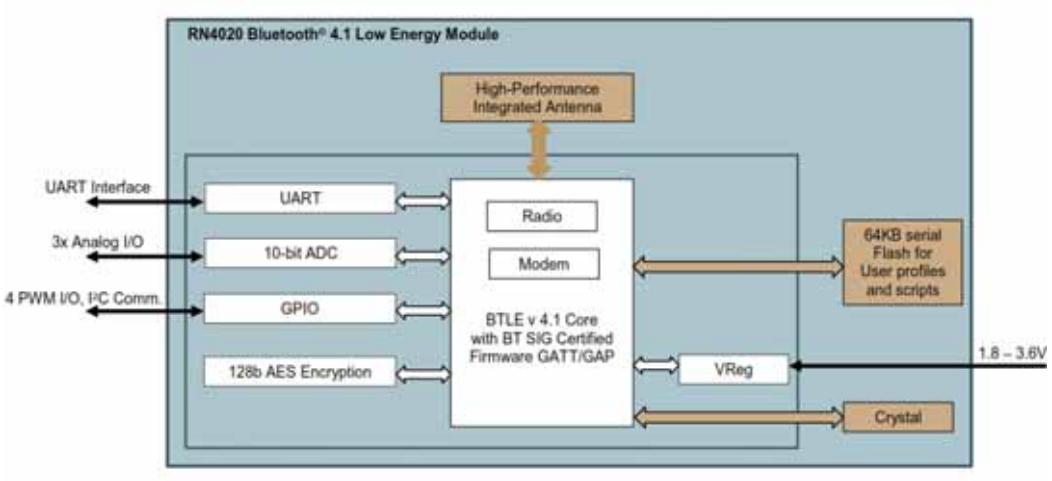


Figura 28: ***Diagrama interno del RN4020***

Según el diagrama de la figura 25, la red BLE está compuesta por una central y tres periféricos, para poder configurar estos dispositivos utilizaremos el software Hércules y lo configuramos a 115200bps según el fabricante, tal como muestra la figura 29.

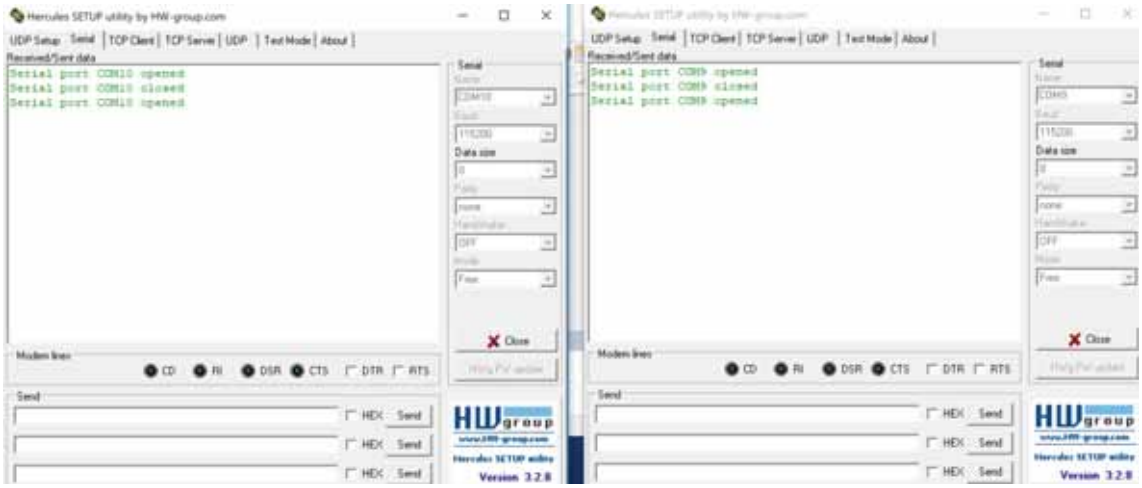


Figura 29: ***Soft Hercules configurado a 115200bps***

4.4.2. Configuración de la central BLE

La central BLE es la encargada de establecer conexión con cada uno de los periféricos hasta obtener una lectura de datos de la temperatura.

Según Hoja del fabricante este módulo se ha configurado de la siguiente manera:

CONFIG_CENTRAL:

SF,1 AOK // Resetea los parámetros de configuración
SS,C0000000 AOK //Habilita el soporte de servicios Device Information

SR,92000000 AOK // Se configura como Central o server

R,1 Reboot // Reinicia el módulo para que las configuraciones realizadas tomen efecto

4.4.3. Configuración de periféricos BLE

Para la configuracion de los perifericos los comandos a ingresar son los siguientes:

//CONFIG SERVICIOS:

+ Echo On // Activa el Eco

SF,1 AOK //

SS,00000001 AOK //

SR,00000000 AOK //

PZ AOK //

PS,12345678901234567890123456789OFF AOK

PC,123456789012345678901234567890 11,12,02 AOK

PC,123456789012345678901234567890 22,02,02 AOK

R,1 Reboot

//---

```
//Ingresar SCRIPT:
WC              AOK
WW              AOK
@PW_ON
A                       //Inicia Scritp
%000E =@I,2             //Asocia canal 2 del ADC con el registro 0x000E
@CONN
SM,1,00100000           //reinicia el Timer1 cada segundo
@TMR1
$VAR1 = @I,2            // asocia la variable 1 con el Canal 2 del ADC
SHW,000E,$VAR1  //Asigna la variable 1 con la dirección 0x000E
SM,1,00100000           //Reinicia el Timer 1
                        //[ESC] para salir del script
END
//-----------------------------------------------------------------------------------------
//INICIA SCRIPT:
SR,21000000             //Inicio Automático de las órdenes del Script
PWR_ON      AOK  //Inicia el SCRIPT después de este comando
R,1     Reboot          //Reset para que se almacenen los cambios
---------------------------------------------------------------
```

Listo, el módulo RN4020 ha sido configurado como periférico con servicios privados y los comandos SCRIPT asignados al canal 2 del ADC. Ahora una vez energizado el

módulo RN4020, en forma autónoma adquiere cada segundo el voltaje que tiene en el canal 2 del ADC.

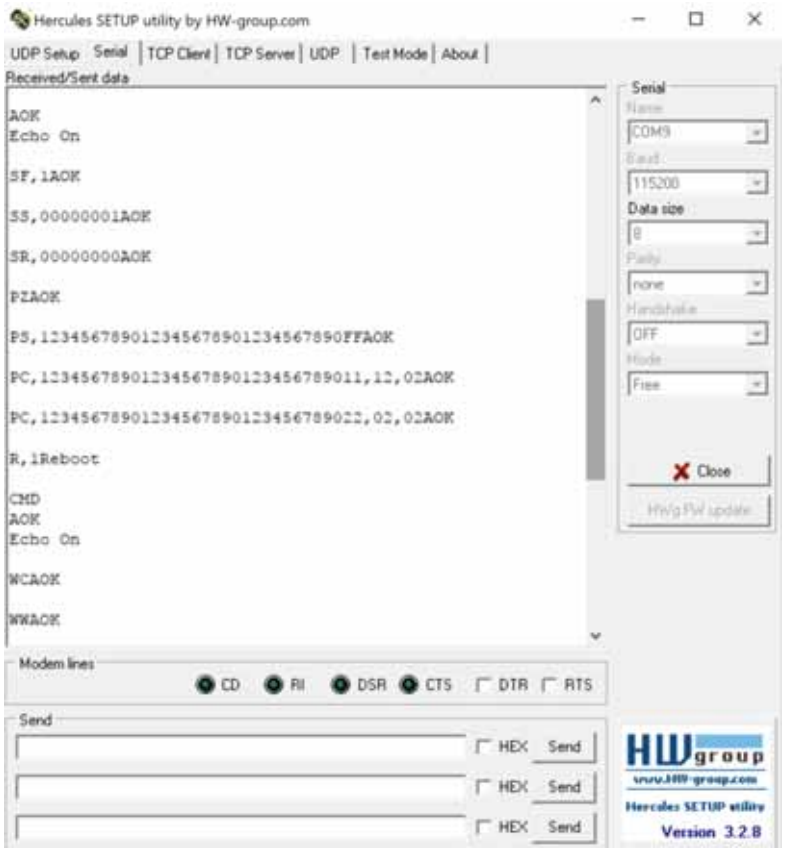


Figura 30: ***Configuración de un Periférico BLE***

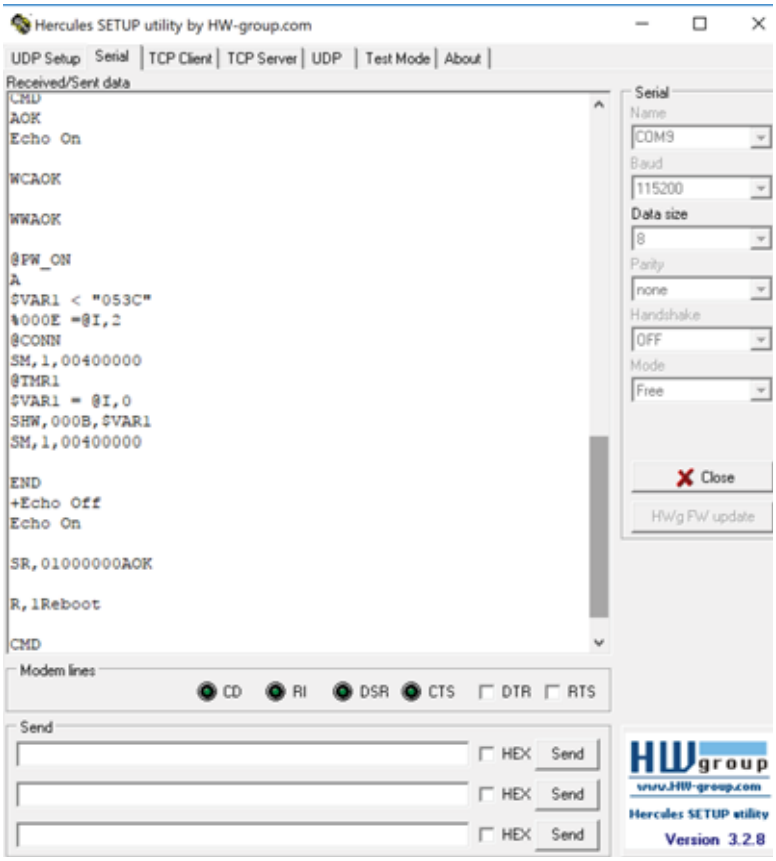


Figura 31: ***Implementación del SCRIPT***

Una vez configurado la central y el periférico BLE, la Central procede a conectarse al periférico mediante los comandos siguientes:

F AOK //Buscar Dispositivos cercanos

001EC037A8A3,0,,12345678901234567890123456789OFF,-49

E47DBDD96DC3,0,,,-52

40163B237102,0,,,-41

//Cuando se envía el comando F (Escanear dispositivos BLE cercanos) se devuelve AOK como respuesta a la ejecución de dicho comando, seguidamente si existiese Periféricos activos se devolverá la dirección MAC (001EC037A8A3) y sus características privadas en caso estén habilitadas, asi como la potencia en dB.

X AOK //El comando X detiene la Búsqueda para que la Central intente una conexión con algún periférico

E,0,001EC037A8A3 AOK // Se envía el Comando E con la Dirección MAC

Connected //Conexión establecida

A partir de este instante la Central con el Periférico pueden realizar transferencia de datos a una tasa de 115200 bps.

Para verificar los servicios privados activos configurados en el periférico BLE, se envía el comando “LS” en la Central, tal como se observa en la figura 32.

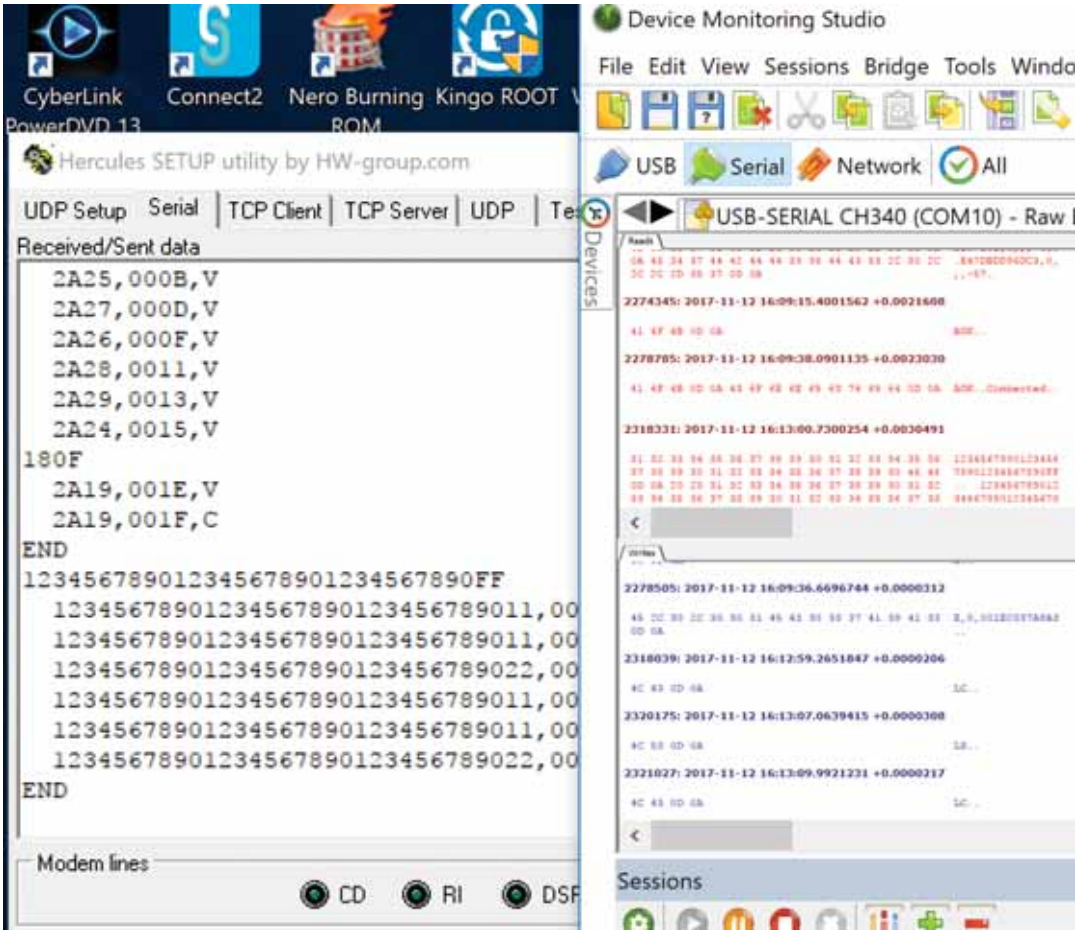


Figura 32: ***Verificación de servicios configurados en el periférico BLE***

Finalmente, para Leer la Temperatura del Periférico, se envía el comando CHR con la dirección de memoria del servicio configurado y el canal ADC asignado en el SCRIPT instalado en el periférico.

Solicitud de la central BLE:

CHR,000E

Respuesta del Periférico BLE:

R,049A.

//El periferico devuelve el voltaje en mV del canal analógico 2

Finalmente se envía el comando K para finalizar una conexión.

K Connection End

Una vez que se han emparejado la central con el periférico BLE y se realiza la solicitud de lectura del valor adquirido por el ADC del periférico; este valor corresponde a un valor de voltaje Hexadecimal, por lo que es necesario realizar la conversión a decimal. Para este caso en particular se tiene:

CHR,000E //Solicitud de lectura de Dato de la Central BLE

R, 049A //Respuesta de Dato adquirido por el Periférico BLE

Entonces:

El valor leído en mV será:

$0x16^3 + 4*16^2 + 9*16^1 + A*16^0 = 0 + 1024 + 144 + 10 = 1178mV = 1.178V$

Del mismo modo se procede a realizar la conexión con los siguientes Periféricos BLE, con la única diferencia que varían sus direcciones MAC de cada módulo.

4.5. Configuración de los dispositivos Zigbee

Los módulos Zigbee se configuran utilizando el software DIGI XCTU como lo habíamos mencionado anteriormente. Según la versión del módulo Xbee, la configuración de estos módulos cada vez es más sencilla; para este caso particular se requiere un Coordinador que debido a la serie S2C se logra configurar como Repetidor. En el caso de los modem /routers que soportaban las versiones anteriores, en esta versión S2C está representado por peer to peer o end point.

Cada módulo Xbee trae la dirección física en la parte trasera del PCB del módulo, pero también es posible leer su dirección mediante el software DIGI XCTU o en todo caso utilizando un terminal y el comando ATMY.

El procedimiento para conectar el dispositivo Xbee S2C es utilizando un módulo serial conectando los pines 2 (Tx) y 3(Rx) del Xbee en forma cruzada a los pines de Rx y Tx del módulo serial respectivamente tal como muestra la figura 33.

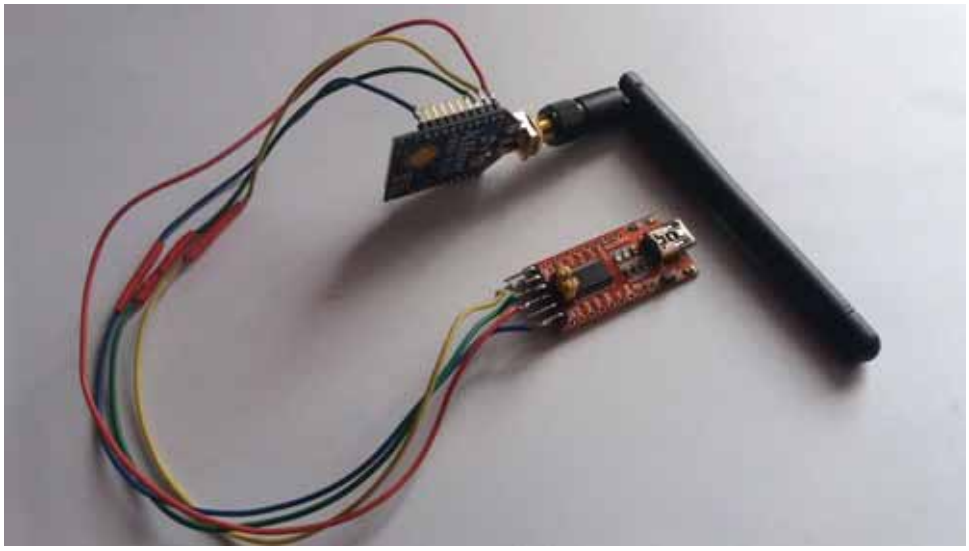


Figura 33: ***Conexión del Xbee con el módulo serie***

Seguidamente ejecutamos el software DIGI XCTU y configuramos el puerto COM detectado con 19200bps que es la configuración por defecto de esta versión de Xbee. Requerimos 3 módulos Xbee y 3 módulos serie para la red Zigbee planteada, por lo que también ejecutaremos 3 veces el software XCTU como muestra la figura 34.

Figura 34:19 ***Diagrama con las conexiones de los tres módulos Xbee***

A continuación, procedemos a Configurar los módulos XBEE tal como indica las figuras siguientes:

Figura 205: ***Configuración de módulos Xbee***

Figura 36: ***Configuración de módulos Xbee***

Figura 37: ***Pruebas de comunicación de la red Xbee***

4.6. Diseño del hardware

Se ha desarrollado dos PCBs uno para el controlador del cliente y el otro para el dispositivo BLE que adquiere la temperatura de las pozas de trucha. La figura 38 muestra el diagrama circuital del controlador del cliente y está conformado por cuatro circuitos:

- La fuente de alimentación conformada por el LM1117 identificado por IC2, el mismo que se encarga de regular el voltaje de entrada a 3.3v; el diodo D3 1N4007 sirve de protección contra inversión de polaridad y los condensadores C11, C8 y C9 logran obtener una señal continua de 3.3v limpia sin ruidos.
- El control del cliente está basado en un PIC24FJ128GB204 el mismo que realiza el control de los dispositivos BLE y Xbee mediante dos puertos UART, siendo a través de los pines RB11(Tx) y RB13(Rx) y el pin RB10(WAKE_SW) los encargados de controlar el Bluetooth LE; en este caso el módulo RN4020 está configurado como Central bluetooth, por lo que el pin RB10 deberá estar siempre nivel alto para que este módulo funcione en modo comando y logre realizar las conexiones con los otros dispositivos BLE que fueron configurados como periféricos. Una vez que el BLE central este en modo comando el LED12 (Azul) se iluminara para indicar dicho estado. Asimismo, cuando la central BLE se logre conectar con un periférico BLE el LED14 (Verde) se encenderá indicando que se encuentra emparejado. Similarmente cuando exista comunicación entre el PIC24 y la central BLE el LED3 (Tx_BLE) parpadeara.
- El condensador C7 de 10uF tiene la función de estabilizar la alimentación del microcontrolador y esta referenciado en la hoja del fabricante como un componente imprescindible que no debe faltar en el ensamblado del PCB.
- Al otro extremo del circuito se encuentra el módulo Xbee S2C, el mismo que es controlado mediante los RC2(Tx) y RC1(Rx), junto al RCO(Reset) que debe estar a nivel lógico 1 para que el módulo Xbee entre en funcionamiento; el LED2 (Xbee_on) se encenderá indicando que el Módulo Xbee está operativo. Asimismo, cada vez que exista comunicación entre el PIC24 y el Xbee el LED5 (Tx_Xbee) parpadeara para indicar dicho estado.

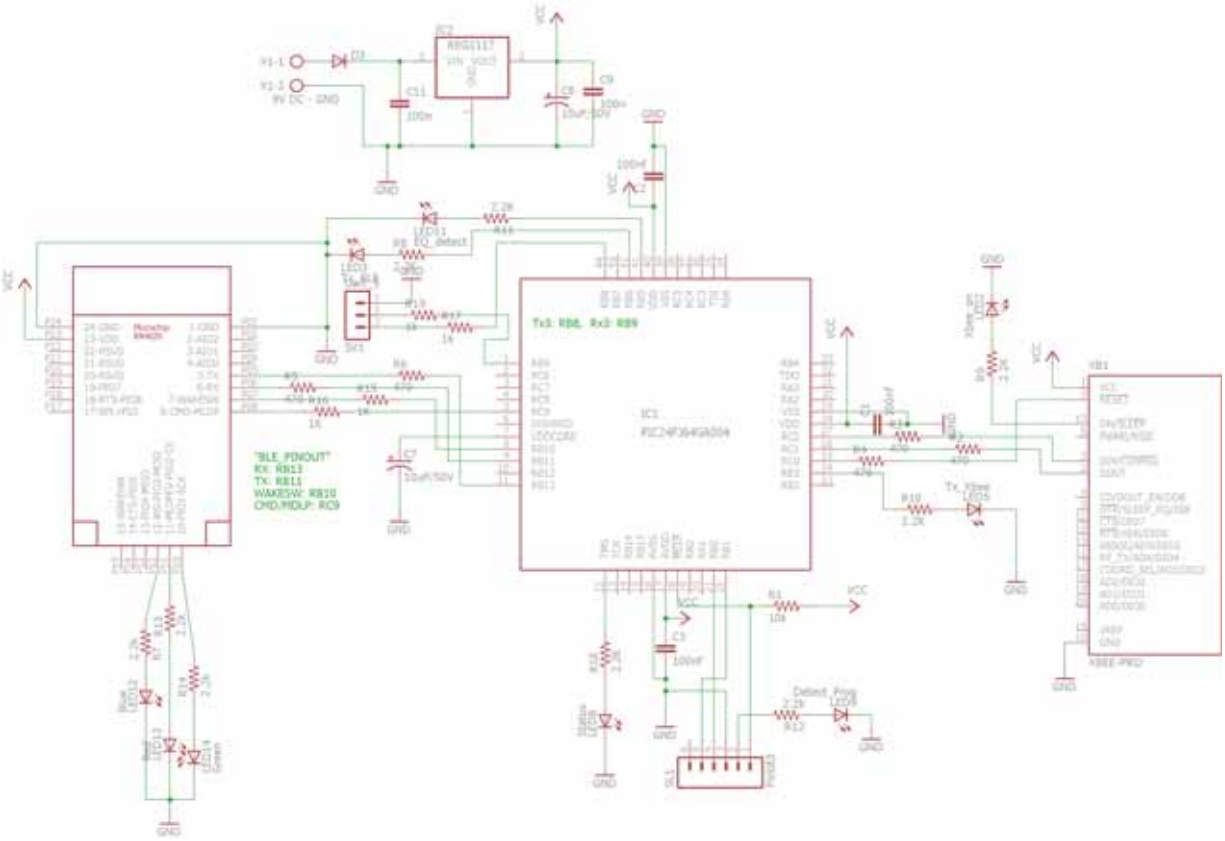


Figura 218: ***Diagrama circuital del controlador Cliente***

- El compilador XC16 permite convertir el código escrito en lenguaje C a un formato denominado *.hex de manera que el microcontrolador PIC24 pueda interpretarlo; pero para trasladar a la memoria de programa, toda esa información generada se requiere de un programador como el Pickit3, el mismo que en forma serial se encarga de trasferir y almacenar la información en forma serial mediante los pines RB0 y RB1. Cada vez que se conecte el pickit3, el LED9 (Detect_Prog) se encenderá indicando la presencia del mencionado hardware.
- El LED8(status) parpadeara a razón de 0.5 segundos indicando el correcto funcionamiento del programa.

En la figura 39 se muestra un dispositivo BLE configurado como periférico, como ya mencioné anteriormente se ha configurado un SCRIPT dentro del dispositivo BLE, el cual permite adquirir el voltaje que se encuentra en el pin 2(AIO2) sin la necesidad de un microcontrolador adicional, se adquiere la señal de medida de temperatura que el termistor NTC de 10KΩ genera del divisor de tensión con la resistencia R2.

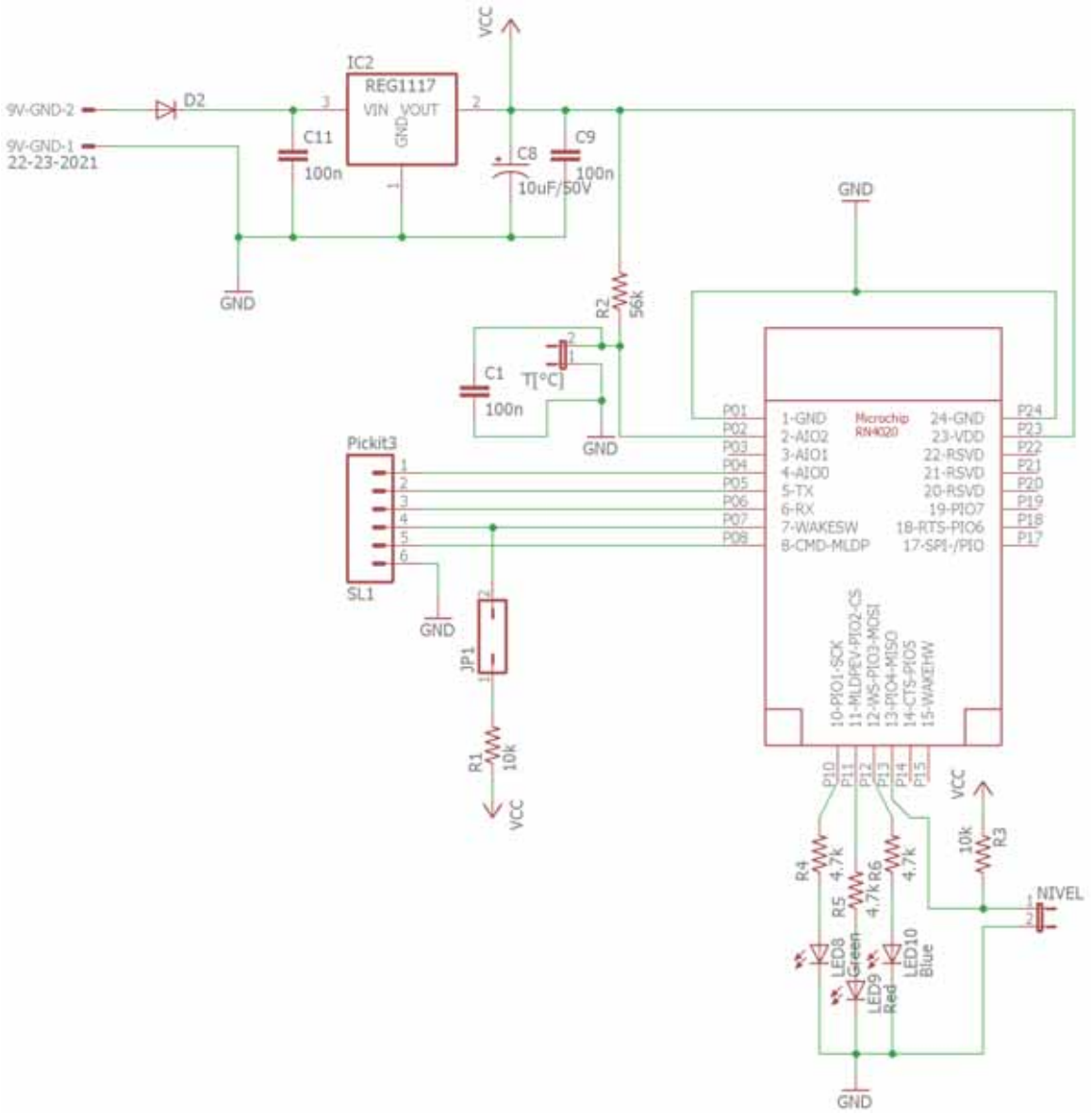


Figura 39: *Diagrama circuital del periférico BLE*

El Conector SL1 permite configurar el módulo RN4020 en caso sea necesario, para lo cual el jumper JP1, debe colocarse de manera que el pin 7 (WAKESW) este a nivel lógico 1 y logre entrar en modo configuración, en ese instante, el LED10(azul) se encenderá indicando el estado de modo comando.

El LED8 (verde) se encenderá cada vez que exista una conexión con la Central BLE; y en caso suceda un error en el RN4020, el LED9 (rojo) se encenderá.

Este Dispositivo requiere una tensión de 3.3V, por lo que se tiene un regulador de Voltaje de 9 a 3.3V representado por IC2.

Para Obtener el valor de R2, se procede del manual de usuario del fabricante del módulo RN4020, la misma que indica en la figura 40, que el rango de medida de cualquier canal analógico es de 0v a 1.3V.

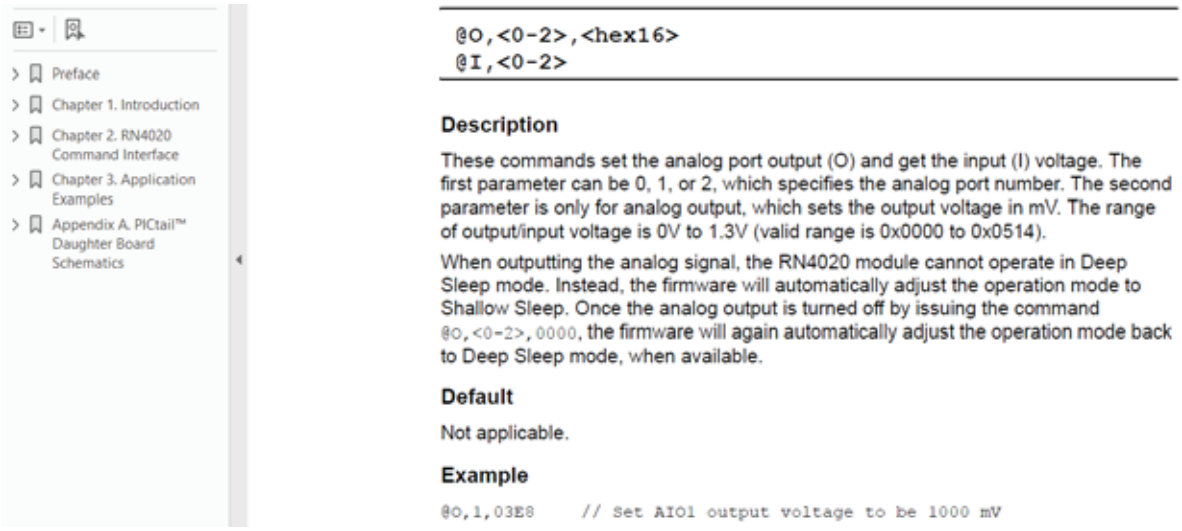


Figura 40: Manual de usuario del Módulo RN4020

Por lo tanto, se debe de elegir una resistencia apropiada para realizar las medidas de temperatura en el rango de voltajes permitidos por el ADC del RN4020.

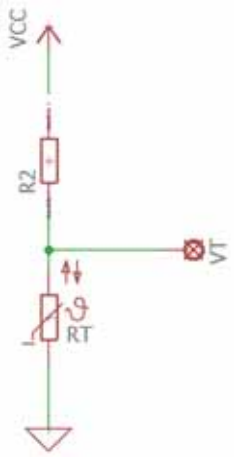


Figura 41.- Circuito de divisor de tensión del NTC

Para obtener R2 se parte del divisor de tensión conformado por la resistencia R2 y el termistor NTC de la figura 41, se define la formula (a).

$$V_T = \frac{R_T * V_{CC}}{R_T + R_2} \ldots\ldots Ec(a)$$

Además, como el canal analógico del ADC lectura valores entre 0 y 1.3V, entonces podemos definir la siguiente expresión:

$$0 < V_T < 1.3V \ldots\ldots Ec(b)$$

Entonces:

Para $V_T > 0v$:

$$R_2 + R_T > R_T * V_{CC}$$

Reemplazando valores:

$$R_2 > 2.3\ R_T \ldots..Ec(c)$$

Similarmente;

Para $V_T < 1.3\ V$:

$$1.3 > R_T * V_{CC}/ (R_2 + R_T)$$

$$R_2/R_T > (3.3/1.3)$$

$$R_2 > 1.5384 * R_T \ldots.. Ec(d)$$

Seguidamente, utilizando el código Matlab, calculamos el valor de la resistencia del termistor para el rango de medida de 3 a 30°C; se elige este rango de medida debido a que es posible llegar a dichas temperaturas durante épocas de invierno y épocas de verano. Obteniendo los valores de resistencias de:

Para 3 °C:

$R_{TH-3} = 24.4341\ K\Omega$

Para 30 °C:

$R_{TH-30} = 8.3114\ K\Omega$

Luego, reemplazando en la ecuación (c) y (d), se obtiene que:

$R_2 >= 56.189\ K\Omega$ y $R_2 > 12.78\ K\Omega$

Eligiendo el valor de R_2 = 56 KΩ por ser un valor comercial y utilizando la ecuación (a); Verificamos que:

Para 3 °C:

R_{TH-3} = 24.4341 KΩ

V_T = 1.0025V < 1.3V

Para 30 °C:

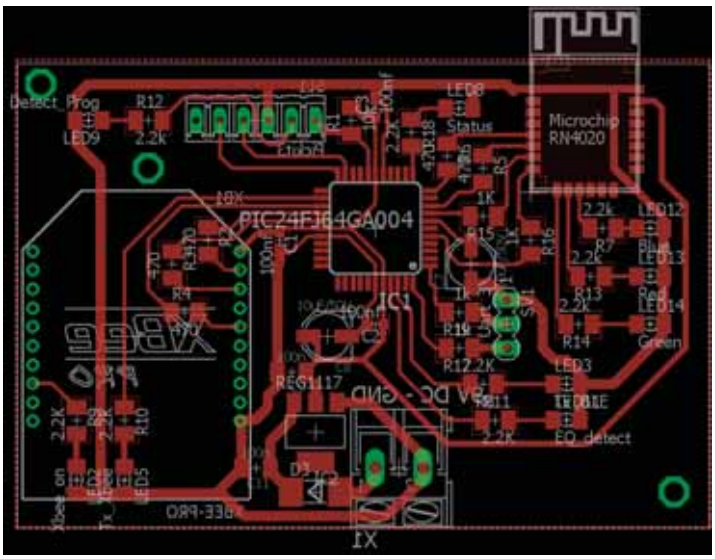
R_{TH-30} = 8.3114 KΩ

V_T = 0.4265V > 0V

Por lo que son valores permitidos dentro del rango de medida del ADC de módulo RN4020.

4.7. Diseño del circuito impreso PCB

A continuación, se presentan los PCBs del cliente, desarrollados en el software EAGLE,



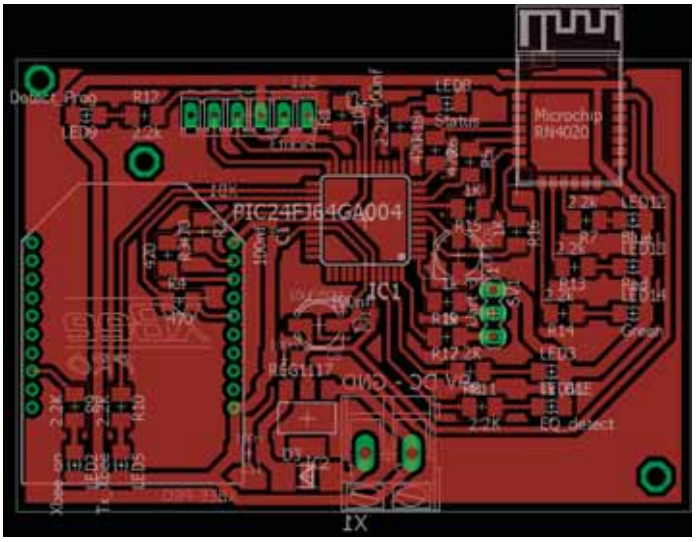


Figura 42: ***PCB del controlador Cliente***

4.8. Software

Como se ha mencionado líneas arriba, el microcontrolador PIC24 es el encargado de controlar tanto al Xbee PRO S2C como al módulo RN4020. La empresa Microchip posee sus propias herramientas de desarrollo para programar a sus microcontroladores, entre ellas citaremos al software MPLAB_X y su respectivo compilador XC16, adicionalmente se debe instalar el MPLAB CODE CONFIGURATOR (MCC) para para la configuración del microcontrolador.

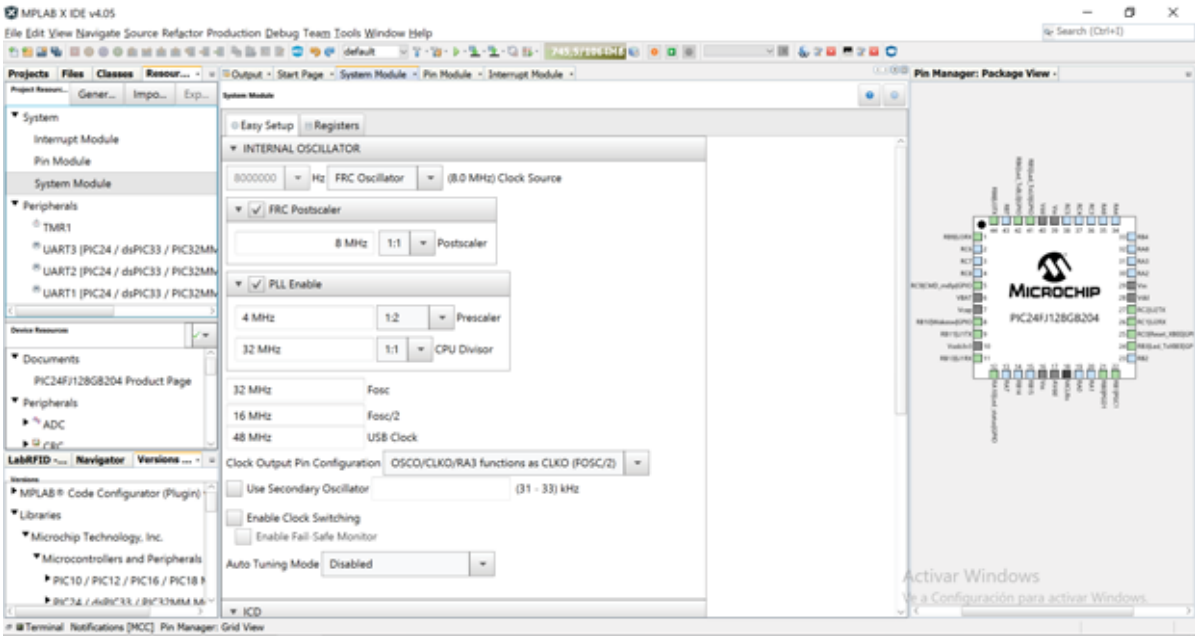


Figura 43: ***Configuración del clock del PIC24***

La figura 43 muestra una pantalla de la configuración del PIC24 utilizando el MCC. Asimismo, la figura 44 detalla la asignación de los pines de entrada y salida del microcontrolador PIC24FJ128GB204. Similarmente se realizaron las configuraciones respectivas de los módulos Timer 1, UART1, UART2 y UART3, habilitando las interrupciones de estos módulos; en el caso del Timer1, esta configurado para provocar una interrupción cada vez 500ms. El UART1 se ha configurado con115200bps y está habilitado para provocar una interrupción cada vez que reciba un byte de la Central Bluetooth Low Energy. El UART2 está configurado a 9600bps y también está habilitado para provocar interrupción por recepción de datos cada vez que el módulo Xbee Cliente, haya recibido datos del Repetidor.

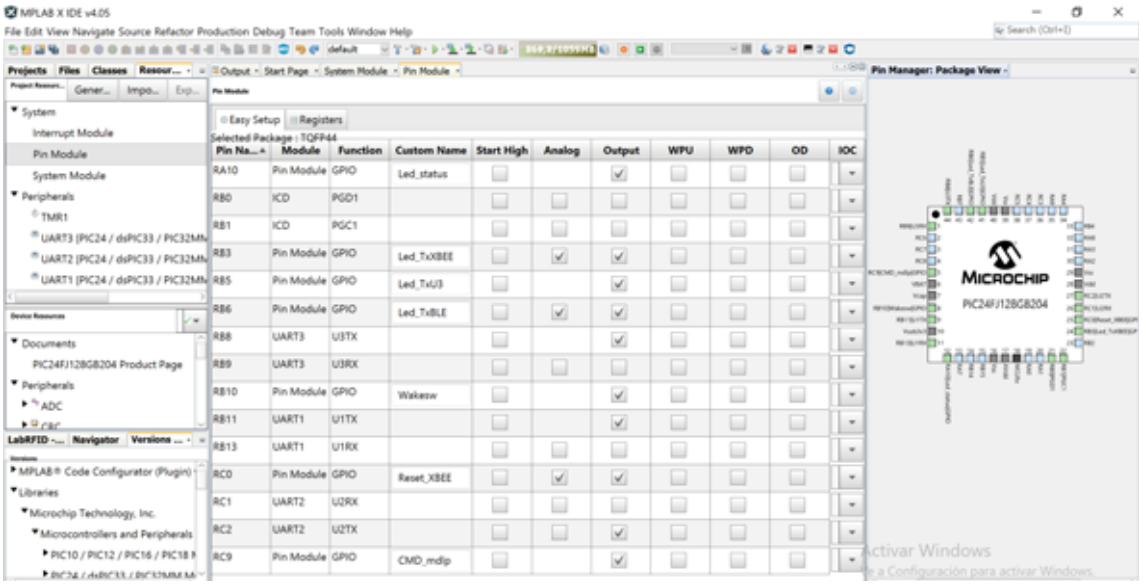


Figura 44: ***Configuración de puertos y periféricos***

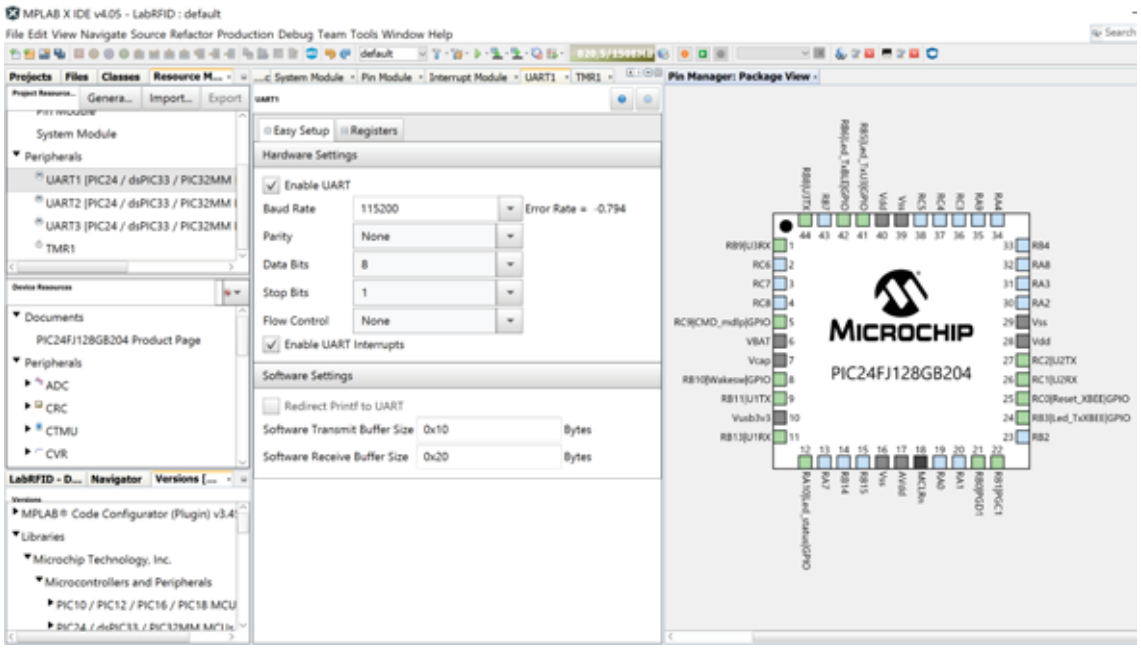


Figura 45: ***Configuración del módulo UART del PIC24***

Una vez realizada la configuración con el MPLAB CODE CONFIGURATOR, se procede a generar el código en C y las librerías respectivas tal como indica la figura 46; en dicha figura podemos observar que se han autogenerado los archivos *.h y *.c; así como el archivo principal donde se escribe el código para el microcontrolador PIC24.

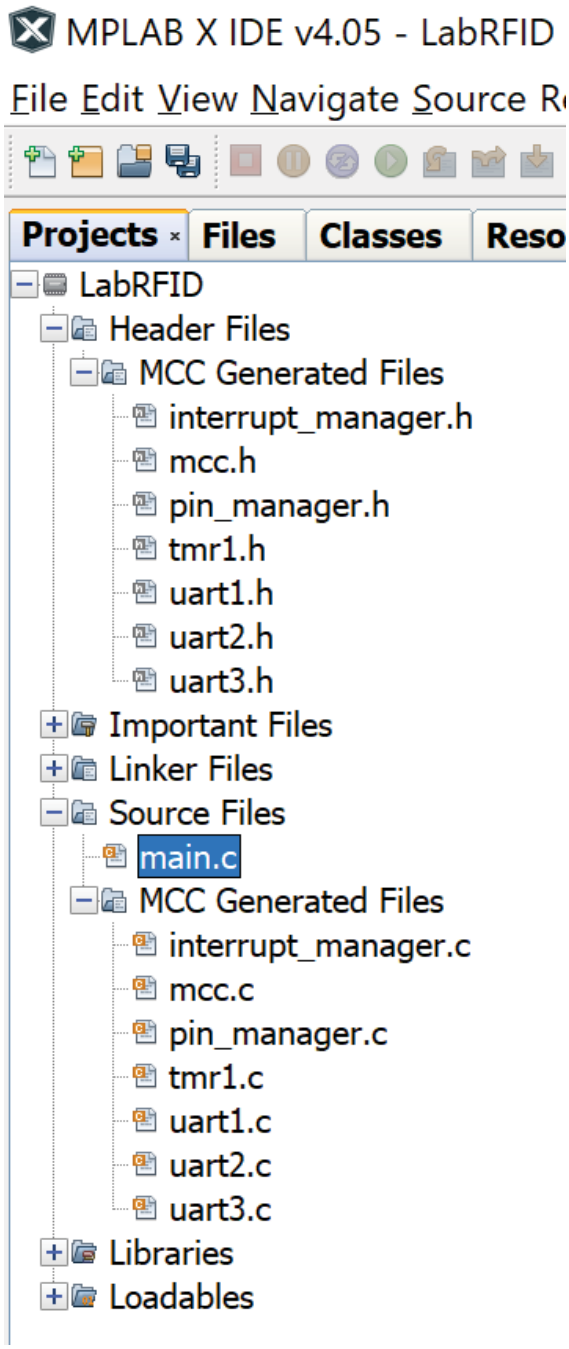


Figura 46: ***Archivos generados con el MCC***

Lo que sigue es abrir el Archivo main.c para empezar a programar.

Los diagramas de flujo del programa cargado en el controlador del cliente (PIC24FJ128GB204) son los que se muestran en las figuras siguientes.

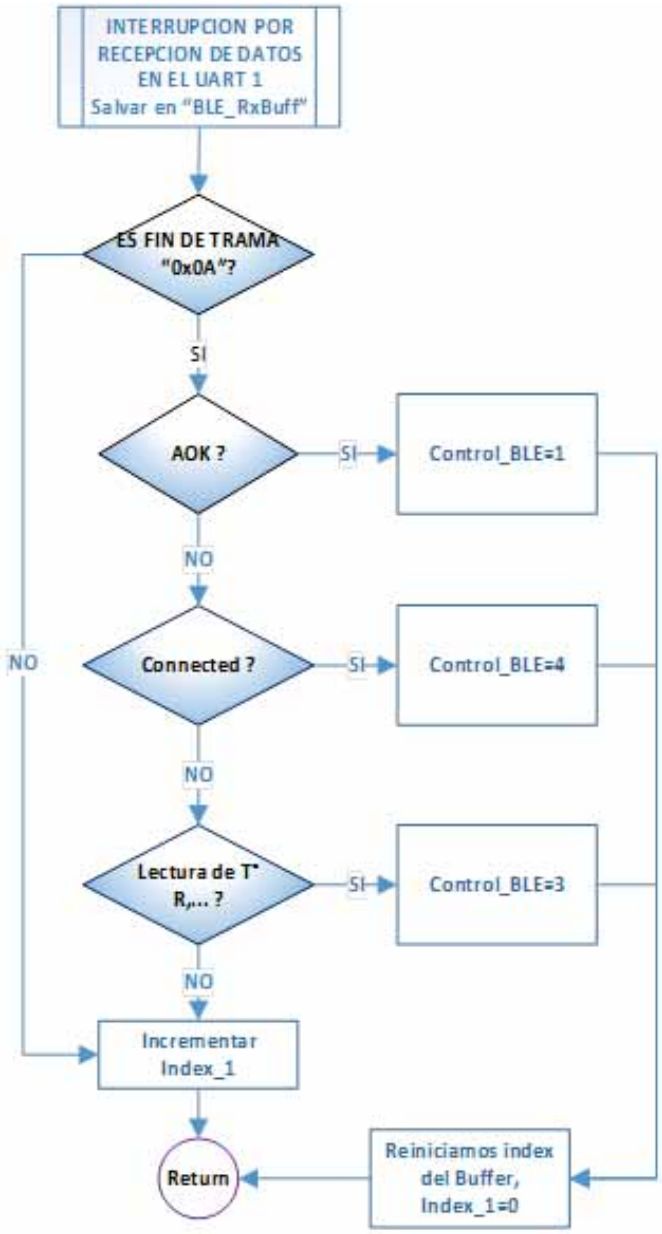


Figura 47: ***Diagrama de flujo de la interrupción por recepción de datos del módulo UART_1***

Cada vez que el BLE Central recibe una respuesta de AOK se carga el registro Control_BLE con el valor 1, similarmente se carga con 4 y 3 cuando se ha recibido "Connected" y "R,0435." Respectivamente. "Connected" es la respuesta que devuelve la Central por el puerto serie cuando se ha establecido una conexión con un periférico y "R,…" es la respuesta a la solicitud de la central para pedir la medida de la temperatura de un periférico.

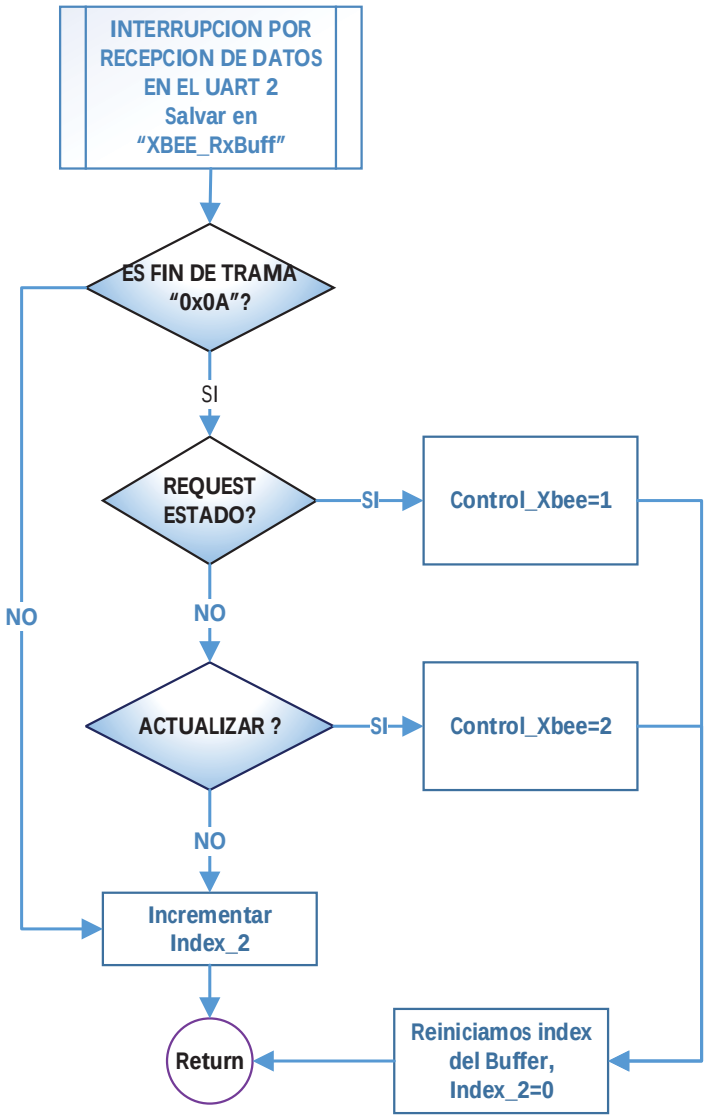


Figura 22 ***Diagrama de flujo de la interrupción por recepción de datos del módulo UART_2***

Cuando el Server realiza solicitudes de estado o actualización, el cliente devuelve dicha solicitud, cargándose el registro Control_Xbee con el valor 1 y 2 respectivamente.

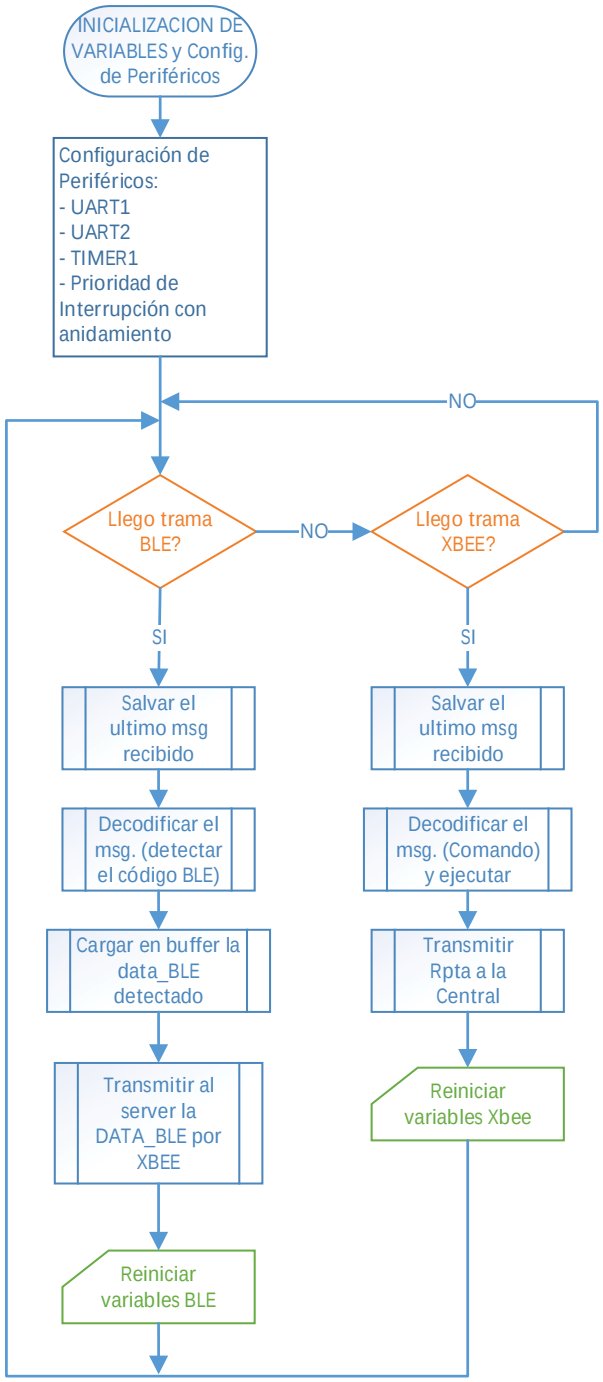


Figura 49: ***Diagrama de flujo del programa principal del cliente***

El UART1 se encarga del control del módulo BLE, el mismo que se inicializa con 115200bps, 8 bits, sin paridad. El UART2 se encarga de comunicar al módulo XBEE, el cual está configurado a 9600bps, 8 bits, sin paridad. Para ambos módulos UART se han habilitado las interrupciones por recepción de datos. Cada vez que se ha detectado una

trama nueva o se ha detectado el carácter 0x0A, se lee el registro Control_BLE o Control_XBEE para realizar la acción correspondiente.

Para el caso del módulo BLE, la Central BLE realiza en forma secuencial la conexión con cada periférico BLE, una vez conectado se solicita la información de temperatura y finalmente se procede a desconectar tal como se detalló el procedimiento en el ítem 4.4.2 (CONFIGURACIÓN DE LA CENTRAL BLE).

Para el caso del Timer1, este es un temporizador de 16 bits, por lo tanto, se ha configurado con un pre-escalamiento de 1:256 y Frecuencia de oscilación de 16Mhz. PR1 es el registro en el cual se actualiza o se escribe el Timer1. Con los datos mencionados y sabiendo que ya tenemos habilitado la interrupción por desbordamiento del Timer1 se realiza el siguiente cálculo:

Periodo del TIMER1 = 256*(1/16000000)* (31250) = 500000 us

Luego para que este Timer1 provoque una interrupción cada 500ms es necesario cargar, en cada desbordamiento, al registro PR1 con el valor:

34245 (65535 – 31250) = 34285 =0x85ED

Recordando que todo este código ha sido autogenerado por el MCC (MPLAB CODE CONFIGURATOR), facilitando la programación.

Finalmente, el programa del Servidor fue implementado en el software LabVIEW, el mismo que consiste en Esperar el carácter de fin de trama 0x0A, tal como indica la figura 50. Cuando le llega una trama completa que el cliente haya enviado, el server decodifica la trama, separa la información para identificar las temperaturas leídas por los Periféricos BLE 1, 2 y 3. O en forma más general, obtener y visualizar las temperaturas T1, T2 y T3 que corresponden a las Pozas 1, 2 y 3 respectivamente.

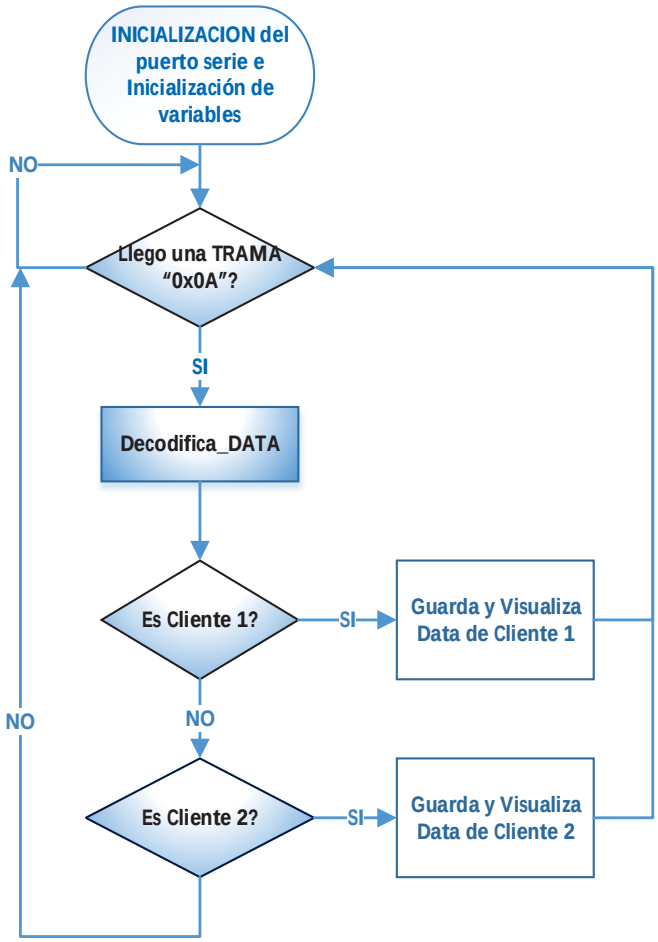


Figura 50: ***Diagrama de flujo del programa del Server***

A continuación, se muestra en la figura 51, el panel de control del server con los datos recibidos de las pozas de trucha 1, 2 y 3.

Similarmente se muestra el código de la configuración de puerto serie de la PC a 9600bps, 8 bits y sin paridad, debido a que los módulos XBEE están configurados del mismo modo; asimismo se ha configurado para que el módulo serie de la PC pueda leer los datos almacenados en su buffer cada vez que llegue el carácter 0x0A.

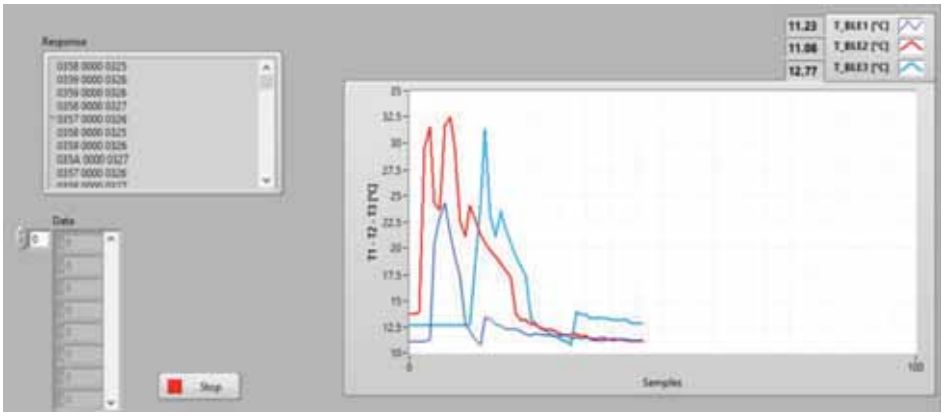


Figura 51: ***Panel de control del servidor***

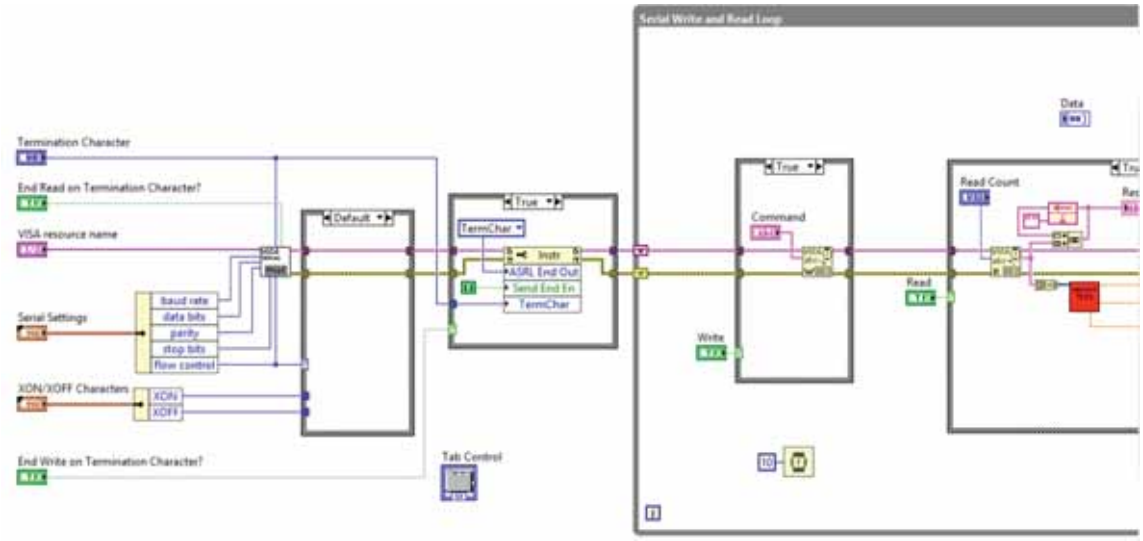


Figura 52: ***Configuración del Puerto serie de la PC del server***

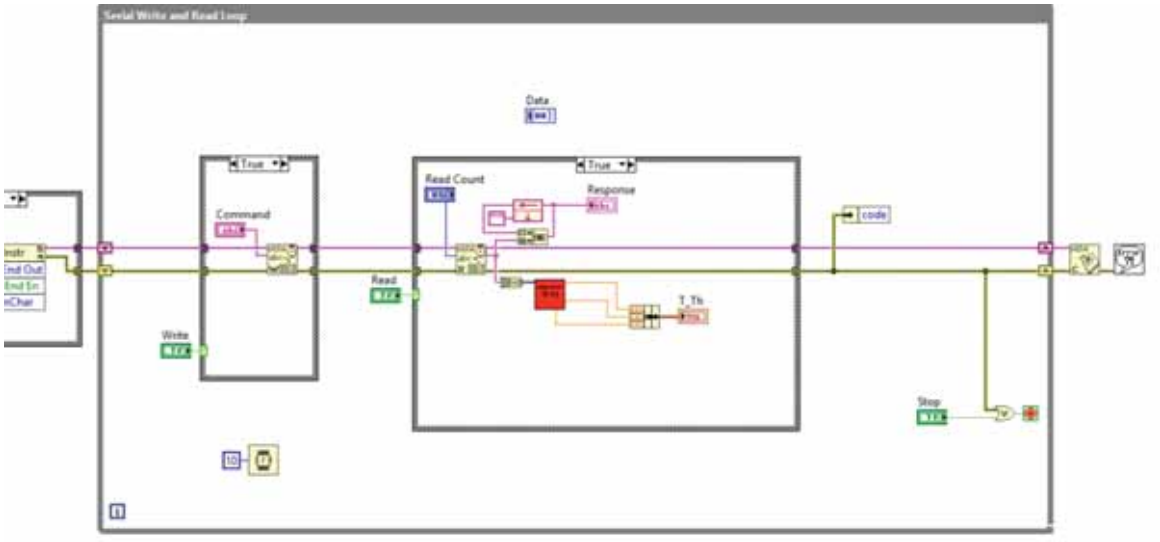


Figura 53: ***Lectura de la trama recibida*****.**

La figura 53 indica cómo se procede a leer la trama de datos recibida; y el bloque Convert_T[°C] permite decodificar y convertir las temperaturas adquiridas por los Dispositivos BLE; en la figura 54 se muestra el procedimiento mencionado. Los periféricos BLE adquieren el voltaje medido en el Canal AN2 del periférico RN4020, dicho voltaje es expresado en 3 dígitos de tipo hexadecimal y trasmitida en formato ASCCI, por lo que es necesario primeramente convertir el formato Ascci a Hexadecimal y luego a decimal, para luego dividirlo entre y obtener el valor en voltios. Seguidamente se utiliza la fórmula que caracteriza al Termistor NTC de 10Kohm definida por:

CALCULO de β:

$$R_{TH} = R_{25} * e^{\beta(1/T - 1T/_{25})}$$

$$LnR_{TH} = Ln\ (R_{25} * e^{\beta\left(\frac{1}{T} - \frac{1}{T_{25}}\right)})$$

$$Ln(R_{TH}/R_{25}) = \beta * (\frac{1}{T} - \frac{1}{T_{25}})$$

$$\beta = Ln(\frac{(R_{TH}/R_{25})}{\frac{1}{T_H} - \frac{1}{T_{25}}})$$

T(°C)	R_{TH} (KΩ)	β
5	221.65	3296.9
10	180.00	3304.69
20	147.20	3318
25	120.99	3327.32
30	83.09	3345.32
40	58.25	3360.51
50	41.60	3376.86
60	30.24	3390
	3339.95	3340

Tabla 5: Calculo de β

De los datos obtenidos mediante el multimetro Fluke 287, se obtiene el valor **β** promedio de la constante de temperatura del material del thermistor.

β =3340

$$R_T = \frac{V_T * R_2}{V_{BAT} - V_T} = \frac{10000 \cdot V_T}{3.3 - V_T}$$

$$T_T = \frac{1}{\frac{1}{\beta}\ln\left(\frac{R_T}{R_{T25}}\right) + \frac{1}{T_{25}}} - 273.15$$

Donde:

R_T: Resistencia del termistor

V_T: Voltaje del termistor medido como divisor de tensión.

V_{BAT}: Voltaje de alimentación 3.3v

R_{T25}: Resistencia del termistor a 25 °C

T_{25}: La temperatura a 25 °C expresado en Kelvin

B: Temperatura del Material del Termistor expresado en Kelvin

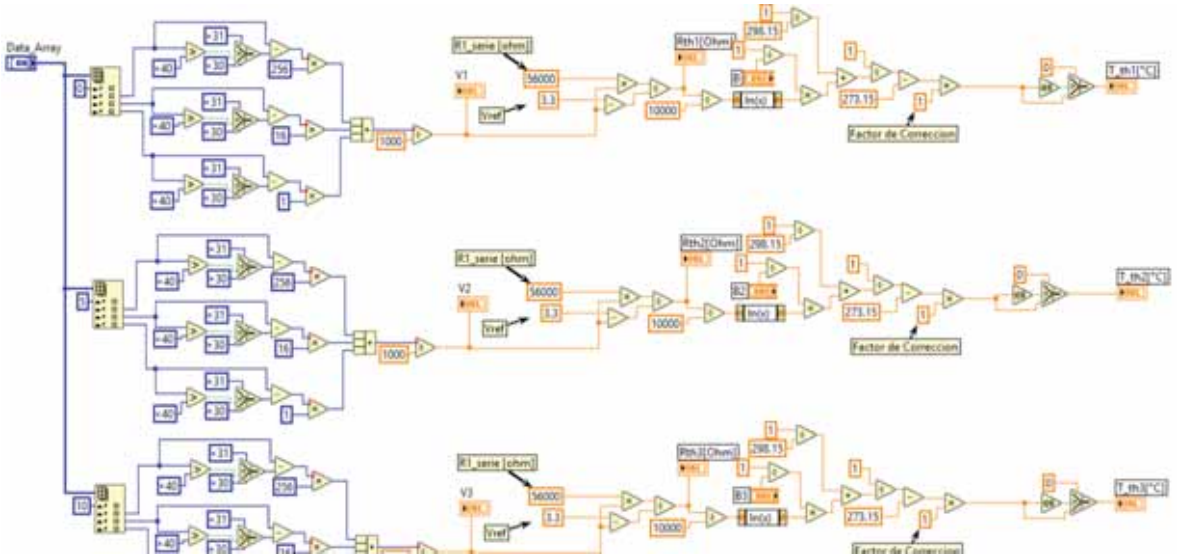


Figura 54: ***Decodificación de la trama recibida***

Es necesario que el Usuario final (Dueño de las factorías de Truchas) visualice las temperaturas de sus pozas, por lo que fue necesario implementar una Aplicación Android básica en APP Inventor para tal fin, como se indica en las figuras siguientes:

Figura 55: ***Entorno de desarrollo de App inventor***

Figura 56: ***Pantalla principal de la aplicación Android***

En la figura siguiente se muestra la inicialización de las etiquetas; una vez que se presiona el botón Conectar definido por ListPicker1, se muestran todos los dispositivos Bluetooth disponibles, luego se debe elegir el Bluetooth HC06 para poder establecer una conexión, una vez establecida la conexión, la etiqueta Label2 se cargara con el Estado: Conectado.

Finalmente, una vez conectado, se decodifica la trama transmitida por el microcontrolador PIC24, el mismo que consiste en el valor de la Temperatura 1, 2 y 3 y un separador entre cada valor como se muestra en el ejemplo:

15,64 | 20,34 | 19,34

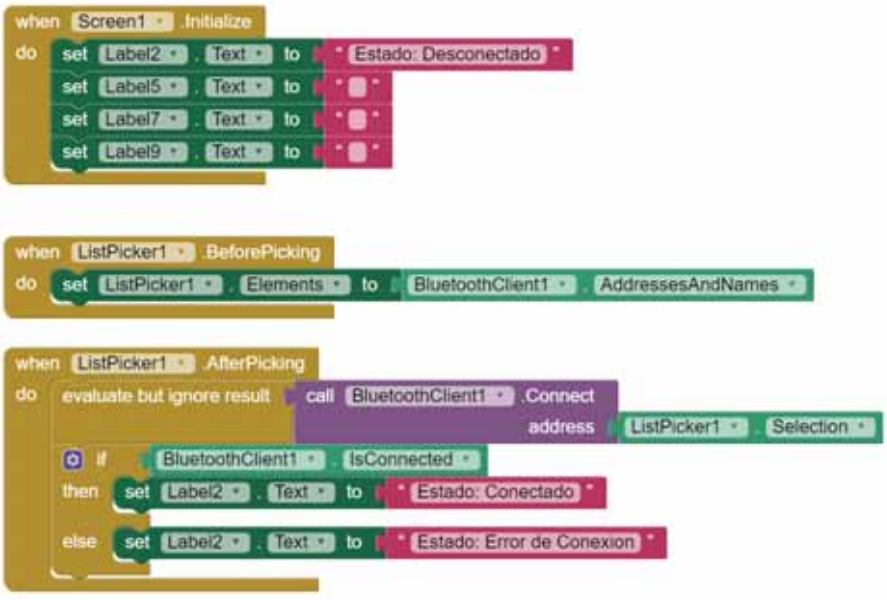


Figura 57: ***Inicialización y Conexión del Bluetooth HC06***

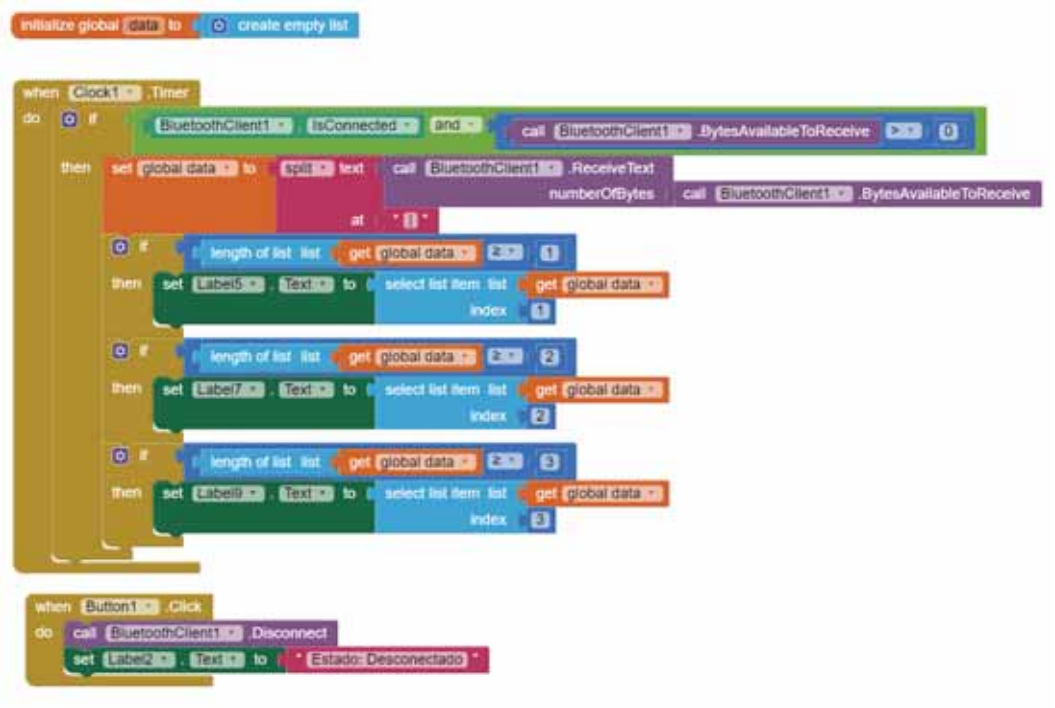


Figura 58: ***Decodificación y visualización de datos de temperatura adquiridos***

La temperatura T1 está asignada al Label5, la temperatura T2 está asignada al Label7 y la temperatura 3 al Label9. La figura 58, muestra que cada vez que llega una trama de datos, este bloque procede a separar las temperaturas de acuerdo al carácter separador " | " (barra vertical).

Capítulo 5 : VALIDACIÓN DEL SISTEMA DE MONITOREO DE TEMPERATURA

5.1. Pruebas de Validación

El proceso de desarrollo de diseño del sistema de monitoreo contempla los pasos que se muestra en el diagrama a continuación, estos procedimientos han permitido como resultado acotar las deficiencias en la medición del parámetro de temperatura a la entrada del diseño y las discrepancias entre el diseño y el sistema propuesto, reduciendo así los errores y corrigiendo en cada etapa del proceso. La validación lo que busca es aumentar la probabilidad de que este sistema aplicado en campo se refleje como un dispositivo que es adecuado para el uso previsto.

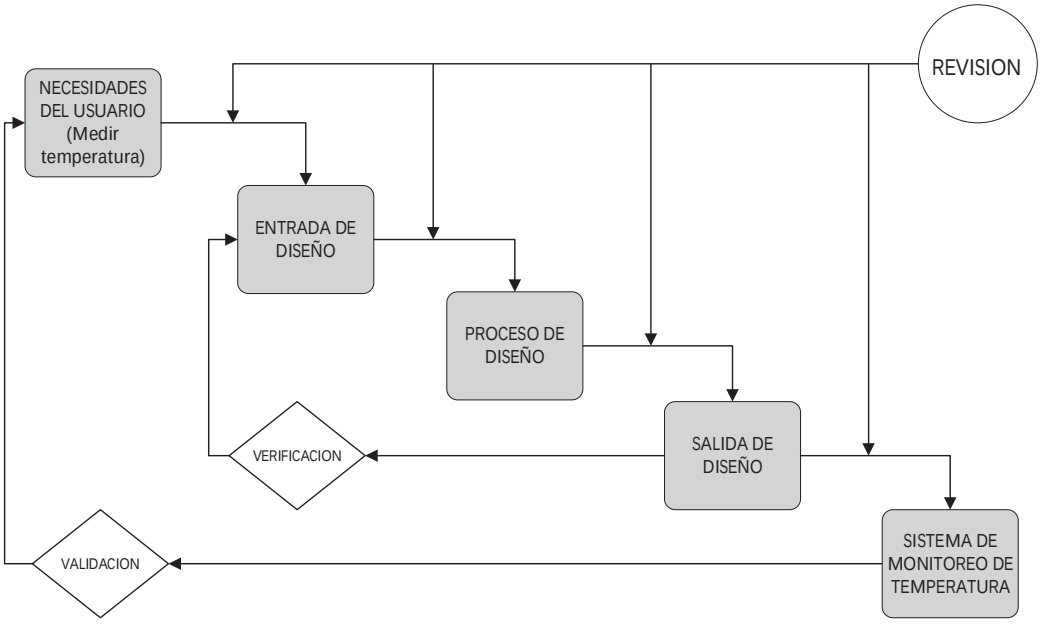


Figura 239: ***Validación del sistema de monitoreo (Adaptado de GHTF-SG3)***

El diagrama anterior es un modelo en cascada y permite la verificación que viene a ser un examen de cada aspecto y en cada etapa, permitiendo encontrar los elementos

(dispositivos) más adecuados para el sistema y la obtención de resultados se ajusta a la medición del parámetro requerido. Por otro lado, la validación incluye el proceso anterior, y evalúa si el diseño se ajusta o satisface a las necesidades del usuario.

5.2. Diseño de prueba experimental

El validar un equipo para uso comercial y certificado requiere de muchas pruebas experimentales como son, por ejemplo:

- Diseño de pruebas de seguridad eléctrica.
- Diseño de pruebas de compatibilidad electromagnética (EMC)
- Diseño de pruebas de flujo, volumen, presión y resistencia ambiental.

La validación también contempla lo siguiente:

A. Calificación del equipo o calificación de instalación.
B. Demostración que el equipo o sistema generara los resultados aceptables.
C. Establecimiento de la estabilidad a largo plazo o calificación de desempeño.

Se puede indicar que el punto A. comprende aspectos como condiciones de instalación, calibración, mantenimiento preventivo, características de seguridad, condiciones ambientales, documentación y manuales; el punto C. que busca garantizar que el sistema operara en condiciones normales y a largo plazo de forma estable. Estos aspectos requieren un estudio y pruebas que demandan más tiempo y pruebas; por lo que sugerimos como parte de estudios posteriores que puedan realizar compañeros de la escuela profesional de ingeniería electrónica.

El punto B. considera límites de control del proceso como pueden ser tiempo, temperatura de trabajo, presión. También se considera parámetros de entrada del sistema y comparación con un patrón de medición para la verificación de la exactitud de las mediciones obtenidas por el sistema.

5.3. Comparativa con FLUKE 287

El multímetro Fluke 287 es un dispositivo multipropósito con una interfaz fácil de usar, y pantallas de ayuda. Permite entre algunas funciones que presenta el medir la temperatura utilizando para ello una termocupla propio del multimetro; permite también

que los datos se pueden guardar y analizar en documentos a través de FlukeView. Sin embargo, es necesario adquirir el software y el adaptador para transferir los datos.

Característica de medición de temperatura: -200.0 °C to 1350.0 °C (-328.0 °F to 2462.0 °F). Con precisión base de 1.0%.

Se realizó las mediciones siguientes de 3 sensores de temperatura del prototipo y se comparan con las medidas del multímetro Fluke 287.

Multimetro			
[°C]	T1S1	T2S2	T3S3
13.2	13.3362	13.1034	13.6164
13.2	13.3362	13.1034	13.6164
13.2	13.2706	13.1034	13.6864
13.2	13.3362	13.1735	13.5465
13.3	13.3034	13.2087	13.7215
13.3	13.2706	13.1385	13.7215
13.4	13.3034	13.2087	13.6864
13.7	13.8658	13.5464	13.181
13.7	13.8992	13.8119	14.4312
13.8	13.8992	10.4495	14.3237
13.8	13.8992	13.8119	14.4312
14.1	14.3598	14.12	14.1857
14.1	14.3981	14.12	14.1483
14.1	14.2834	14.0445	14.1857
14.2	14.2616	14.0445	14.1483
14.2	14.2616	14.0445	14.1483
14.2	14.2616	14.1577	14.1483
14.2	14.2999	14.0822	14.0737
14.2	14.3381	14.0822	14.111
14.2	14.3381	14.1577	14.0737
14.2	14.3381	14.1577	14.2231
14.3	14.4148	14.1577	14.2231
14.3	14.4148	14.1577	14.2231

Tabla 6: ***Mediciones simultaneas de temperatura entre dispositivo patrón y sensores de temperatura del prototipo.***

Las mediciones anteriores se considera que las características técnicas de cada uno de los sensores del prototipo son similares, también se puede indicar que la precisión de cada uno de ellos está sujeto a un rango de tolerancia de medición, por lo que se tendrán

medidas diferentes pero aproximadas. De lo anterior se puede obtener el error de medición con respecto al patrón que es el Fluke 287.

Para esto se tiene que el error (%) se puede obtener a partir de la formula siguiente:

$$Error(\%) = \frac{Valor\ Experimental\ de\ sensores - Valor\ de\ patron}{Valor\ de\ patron}\ x\ 100$$

Multimetro	T1S1	Error (%)	T2S2	Error (%)	T3S3	Error (%)
13.2	13.3362	1.03	13.1034	0.73	13.6164	3.15
13.2	13.3362	1.03	13.1034	0.73	13.6164	3.15
13.2	13.2706	0.53	13.1034	0.73	13.6864	3.68
13.2	13.3362	1.03	13.1735	0.20	13.5465	2.63
13.3	13.3034	0.03	13.2087	0.69	13.7215	3.17
13.3	13.2706	0.22	13.1385	1.21	13.7215	3.17
13.4	13.3034	0.72	13.2087	1.43	13.6864	2.14
13.7	13.8658	1.21	13.5464	1.12	13.181	3.79
13.7	13.8992	1.45	13.8119	0.82	14.4312	5.34
13.8	13.8992	0.72	10.4495	24.28	14.3237	3.79
13.8	13.8992	0.72	13.8119	0.09	14.4312	4.57
14.1	14.3598	1.84	14.12	0.14	14.1857	0.61
14.1	14.3981	2.11	14.12	0.14	14.1483	0.34
14.1	14.2834	1.30	14.0445	0.39	14.1857	0.61
14.2	14.2616	0.43	14.0445	1.10	14.1483	0.36
14.2	14.2616	0.43	14.0445	1.10	14.1483	0.36
14.2	14.2616	0.43	14.1577	0.30	14.1483	0.36
14.2	14.2999	0.70	14.0822	0.83	14.0737	0.89
14.2	14.3381	0.97	14.0822	0.83	14.111	0.63
14.2	14.3381	0.97	14.1577	0.30	14.0737	0.89
14.2	14.3381	0.97	14.1577	0.30	14.2231	0.16
14.3	14.4148	0.80	14.1577	1.00	14.2231	0.54
14.3	14.4148	0.80	14.1577	1.00	14.2231	0.54

Tabla 7: ***Determinación de error porcentual.***

De la determinación del error porcentual que se obtiene en la tabla anterior se puede apreciar que el error en el sensor de temperatura T3S3 contiene valores que alcanzan valores altos de error entre 5% y 3% en las primeras 10 mediciones, este error luego se puede ver que alcanza valores aproximados al resto de sensores. También se encuentra un error de 24.28% en el sensor T2S2, no puedo dar una explicación exacta de

lo que sucedió, pero se puede suponer que se debió a errores de pruebas experimentales al manipular o instalar los dispositivos.

A continuación, se muestra tomas de las pruebas de medición y comparación:

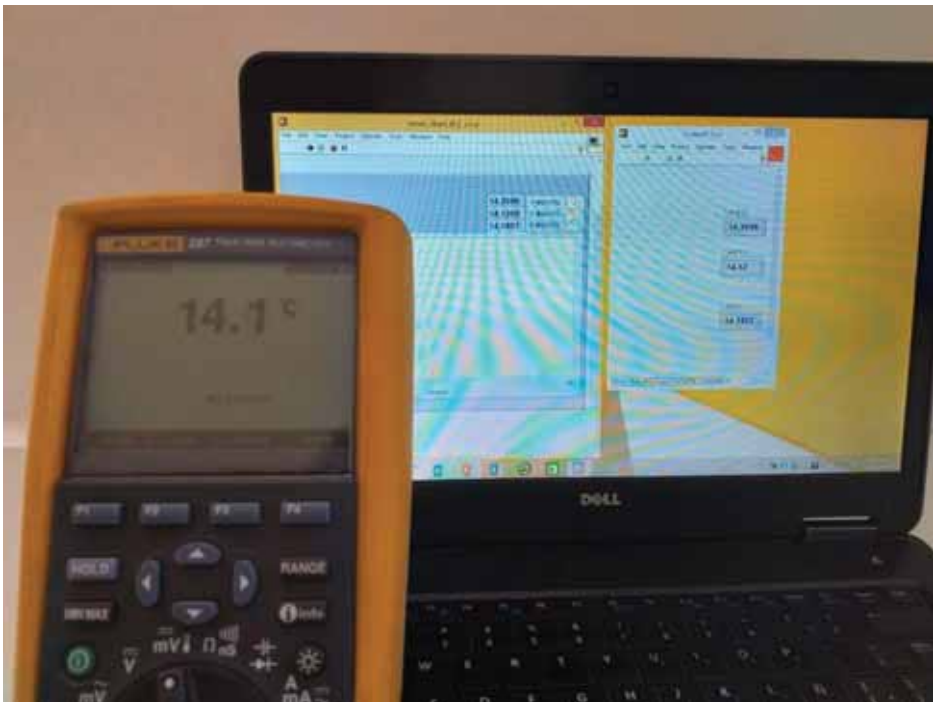


Figura 60: ***Mediciones de prueba con T patrón 14.1***

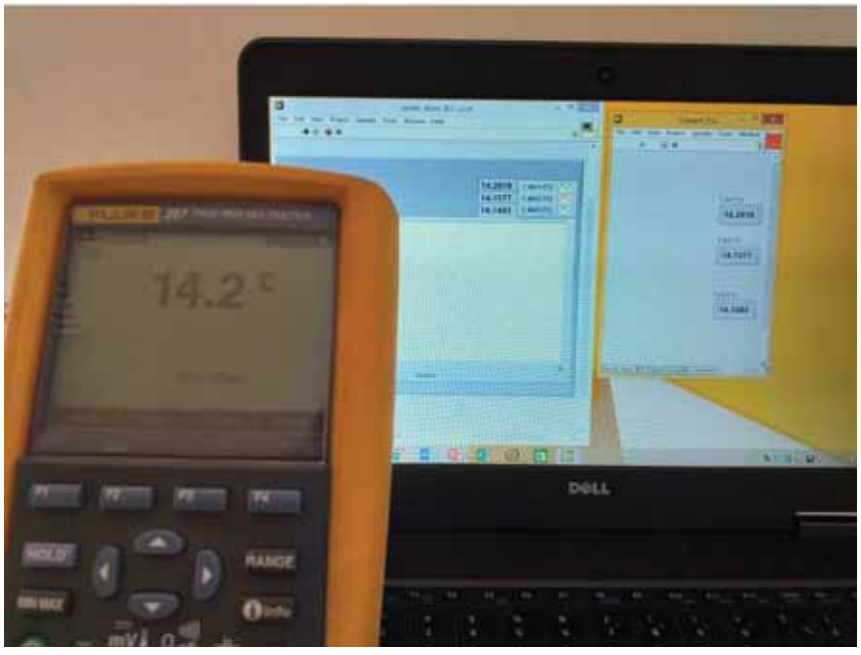


Figura 61: ***Mediciones de prueba con T patrón 14.2***

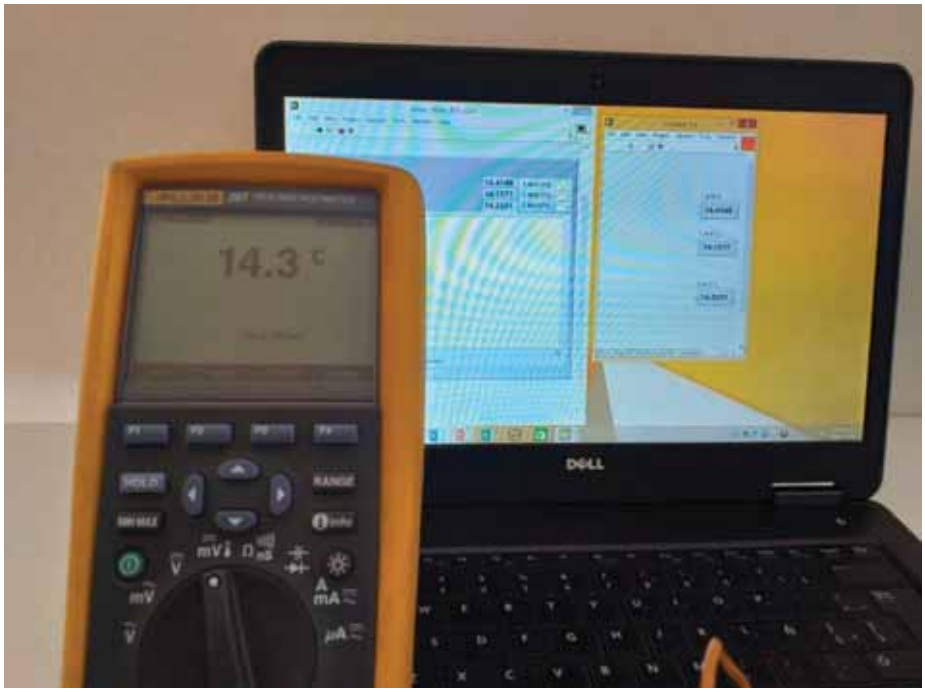


Figura 62: ***Mediciones de prueba con T patrón 14.3***

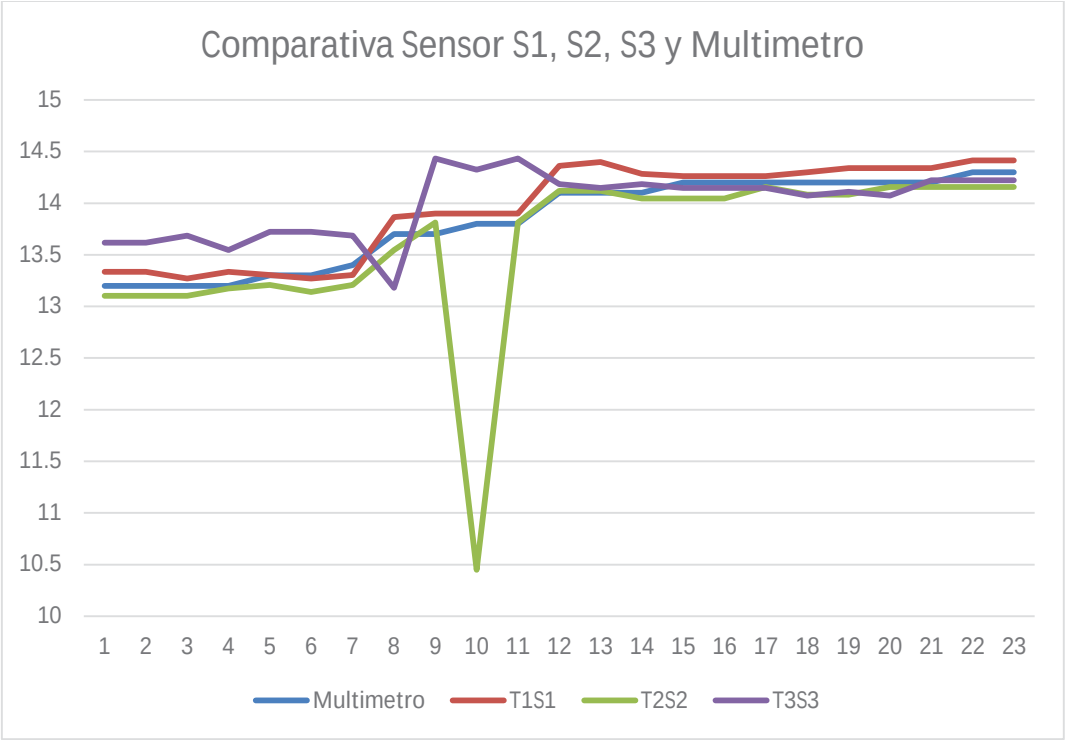


Figura 63: ***Grafica del comportamiento de los sensores y multímetro para 23 muestras***

5.4. PRECIOS DE MATERIALES

Se consideraron los siguientes precios referenciales de los componentes del sistema:

MATERIAL	CANTIDAD	P/U SOLES	COSTO SOLES
Pickit3	1	284.00	284.00
Pic248gb204	1	30.00	30.00
Bluetooth RN4020	4	35.00	140.00
Zigbee	2	350.00	700.00
Resistencias	30	0.50	15.00
Diodos led	10	1.00	10.00
Condensador	10	2.00	20.00
Regulador de Voltaje	5	3.00	15.00
Protoboard	1	45.00	45.00
Computadora	1	2800.00	2800.00
Termistor NTC	9	3.00	27.00
Convertidor UAR a USB	1	50.00	50.00
otros	1	200.00	200.00
total			4336.00

Tabla 8: ***costo de materiales.***

CONCLUSIONES

- Se diseñó una red inalámbrica con dispositivos bluetooth y zigbee para la transmisión y recepción del sistema de monitoreo de temperatura en la crianza de truchas de las piscifactorías del distrito de Lucre, la topología utilizada para la interconexión de los nodos de sensado de temperatura siendo en general el esquema una topología arbol o estrella jerárquica.
- Se definió los sensores que formaran parte de la red para la monitorización de temperatura, siendo utilizados los sensores de temperatura: Termistores NTC de 10k Ohm, determinándose los cálculos para la caracterización del termistor en la aplicación de sensado de temperatura.
- Se definió los diferentes módulos para la transmisión y recepción, explicando la elección de tecnologías inalámbricas como bluetooth y zigbee de bajo consumo de energía, realizando las pruebas del sistema de monitoreo en la Piscifactoría "La Escondida de Adán" a 2.84 Km del Centro Poblado de Lucre.
- Se diseñó el sistema de red de comunicación inalámbrica, la tecnología de comunicación inalámbrica ZigBee utiliza la banda ISM (al igual que los dispositivos Bluetooth RN4020) adoptando la banda de frecuencia en 2.4GHz basada en el estándar de la IEEE como es la IEEE 802.15.4. Para los dispositivos Bluetooth la distancia nominal del enlace está comprendida entre 10cm y 10m, pero aumentando la potencia de transmisión se puede llegar a 100m, con una modulación GFSK (Gaussian Frequency Shift Keying) con un producto ancho de banda por tiempo BT = 0.5 y un índice de modulación entre 0.28 y 0.35.
- Se diseñó el módulo terminal de sensado que se instalara en el estanque para la monitorización de temperatura, para el desarrollo del prototipo se realizó 3 módulos de sensado y se instaló en cada poza de cultivo de Trucha, aproximadamente a 20 cm de profundidad.
- Se diseñó, implemento y valido un prototipo a escala del sistema integral, las validaciones se hicieron en referencia al Multímetro Digital Fluke 287. Las características de este equipo se pueden observar en los Anexos.

RECOMENDACIONES Y SUGERENCIAS.

- Se recomienda en las pruebas de validación que se realizan para uso comercial, puedan ser realizados en estudios posteriores por alumnos de la E.P. de Ingeniería Electrónica; dado que son importantes pruebas de calidad como son de seguridad eléctrica, ambiental, electromagnética entre otros.
- Las pruebas del prototipo se realizaron utilizando energía suministrada por la red eléctrica y también haciendo uso de un panel solar de dimensiones de 15cm x 20 cm, por lo que se recomienda para zonas sin acceso a energía eléctrica se pueda dimensionar de mejor manera sistemas de captura de energía solar a eléctrica con paneles solares y banco de baterías.
- En la temporada de lluvias en el mes de marzo donde se realizo estudios de campo, se pudo observar que hubo aumento de caudal del rio el cual rebaso y afecto a los distintos establecimientos piscifactorías de Lucre, siendo afectados también nuestros sistemas de pruebas, por lo que se recomienda a los productores considerar poner defensas ribereñas para no afectar a sus estanques.
- Los dispositivos utilizados para el prototipo fueron adquiridos via internet y al por menor, por lo cual los costos mostrados en la tabla 8. pueden ser mejorados si se contempla el implementar toda la red del sistema de monitoreo.

BIBLIOGRAFIA

[1] Humberto Zúñiga Valverde - Implementación del sistema de monitoreo de temperatura del cuarto de servidores 3A.

[2] Iván Jesús Romero - Diseño de un sistema inalámbrico usando dispositivos zigbee para el monitoreo de temperatura en la crianza de ovas y alevines en un centro de crianza.

[3] Ministerio de Agricultura, Autoridad Nacional del Agua. RESOLUCIÓN DIRECTORAL Nro. 201-2017-ANA/AAA XII.UV

[4] Revista de Investigación Científica - Pucallpa, Perú. 2(1)2017, SISTEMA AUTOMATIZADO PARA EL CONTROL Y MONITOREO DEL COMPORTAMIENTO DE ALEVINOS DE PAICHE EN CAUTIVERIO.

[5] Maximixe - Elaboración de estudio de mercado de la trucha en Arequipa, Cusco, Lima, Huancayo y Puno.

[6] Organización de las Naciones Unidas para la Alimentación y la Agricultura, Guatemala 2014 - Manual práctico para el cultivo de la Trucha Arcoiris.

[7] Alexander Palomino Lopez y Harry Martinez Sueros, Cusco 2012 trabajo de tesis - Estudio de Ingeniería para la Implementación de una Red Agroclimática Utilizando Tecnología WI-FI para el Distrito De Huayllabamba,

[8] Victor Andres Ayma Quirita y Darwin Auccapuri Quispitupa, Cusco 2010 trabajo de tesis - Diseño de un Sistema de Monitoreo a Distancia de Signos Vitales de Pacientes en la Clínica Internacional SOS GROUP.

[9] https://es.wikipedia.org/wiki/Microcontrolador

[10] https://www.digikey.com/es/articles/techzone/2011/.../temperature-sensors-the-basics

[11] Andrés Choquehuanca Huanca, Cusco 1999, ANÁLISIS MORFOMÉTRICO DE LA CUENCA HIDROGRÁFICA DEL RÍO LUCRE.

[12] Coit, *La situación de las Tecnologías WLAN basadas en el estándar IEEE 802.11 y sus variantes (" Wi-Fi ")*. 2004.

ANEXOS

Contenido de los anexos:

1. Código de Programa PIC24FJ128GB204
2. Código de Programa de la aplicación Android realizada en APP Inventor
3. Código de Programa del Concentrador
4. Pruebas de mediciones de temperatura del prototipo.
5. Hojas de Datos del Fabricante
    - PIC24FJ128GB204 (Microcontrolador)
    - RN4020 (Dispositivo Bluetooth)

ANEXO 1: Código de Programa PIC24FJ128GB204

```
#define FCY 16000000UL
#include "mcc_generated_files/mcc.h"
#include "libpic30.h"
#include "stdio.h"
#include "stdlib.h"
#include "string.h"
#include "float.h"
#include "math.h"


//Variables RFID
unsigned char index1;
unsigned char x;
unsigned char z;
unsigned char Data_request;
unsigned char control_BLE;
unsigned char tam_BLE_data;
unsigned char flag_BLE;
unsigned char flag_wait_rfid;
unsigned char flag_Trama_BLE;
unsigned char cont_error;
unsigned char BLE_RxBuff[100];    //Buffer de Recepcion de datos BLE
unsigned char BLE_TxBuff[100];
unsigned char msg_BLE[80];      // Buffer para transmision de CMD BLE

//Variables XBEE
unsigned char contador,i;
unsigned char Tam_Buff_Xbee;
unsigned char Length_Data;
unsigned char index2;
unsigned char tam_data;            //Almacena tamaño de datos que debe esperar del Xbee
unsigned char flag_xbee;           // Se activa cada vez que se detecta un inicio de trama Xbee
unsigned char flag_wait;           //Espera el byte de tamaño de DATOS del Xbee
unsigned char flag_Trama_xbee;     //Se pone a 1 cuando se ha recibido una trama completa
unsigned char control_xbee;


unsigned char Xbee_TxBuff[100];
unsigned char Xbee_RxBuff[200];
unsigned char msgXbee[100];


unsigned char conex_BLE1[30]={"E,0,001EC037A8A3\r\n|"};
unsigned char conex_BLE2[30]={"E,0,001EC037A8A6\r\n|"};
unsigned char conex_BLE3[30]={"E,0,001EC037A89D\r\n|"};
unsigned char request_T[14]={"CHR,000E\r\n|"};
unsigned char Reset[6]={"R,1\r\n|"};

//Mensajes de verificacion
unsigned char msg_AOK[4]={"AOK|"};
unsigned char msg_Connected[10]={"Connected|"};
unsigned char msg_END[16]={"Connection End|"};


//Data Lecturada de todos los BLE:
unsigned char Data_T[40];
unsigned char Buff_TxU3[40];


/*
                Main application
 */
//************************************ INTERRUPCIONES ****************************

//----------- INTERRUPCION POR Rx UART1 – BLE  ----------------------------

void __attribute__ ( ( interrupt, no_auto_psv ) ) _U1RXInterrupt( void ){
```

```
   if(U1STAbits.URXDA == 1){
     BLE_RxBuff[index1] = U1RXREG;          //ALMACENA EL DATO EN EL BUFFER rfid
     if(BLE_RxBuff[index1]==0x0A){          //Fin de Trama?
       flag_Trama_BLE=1;                    //Trama completa

       if((BLE_RxBuff[index1-4]==0x41)&(BLE_RxBuff[index1-3]==0x4F)&(BLE_RxBuff[index1-2]==0x4B)){
         control_BLE=1;     //AOK?
       }
       if((BLE_RxBuff[index1-10]==0x43)&(BLE_RxBuff[index1-9]==0x6F)&(BLE_RxBuff[index1-
8]==0x6E)&(BLE_RxBuff[index1-3]==0x65)&(BLE_RxBuff[index1-2]==0x64)){
         control_BLE=4;  // Connected?  (Con...ed)
       }

       //R,049A. (52 2C 30 34 39 41 2E 0D 0A
       if((BLE_RxBuff[index1-8]==0x52)&(BLE_RxBuff[index1-7]==0x2C)&(BLE_RxBuff[index1-2]==0x2E)){
          Led_TxU3_SetLow();
          Data_T[z]=BLE_RxBuff[index1-6];
          Data_T[z+1]=BLE_RxBuff[index1-5];
          Data_T[z+2]=BLE_RxBuff[index1-4];
          Data_T[z+3]=BLE_RxBuff[index1-3];
          control_BLE=3;     //R,0345.?
       }

       index1=0;               // Reset Index BLE_BUFF
     }
     index1++;
   }
IFS0bits.U1RXIF = false;
}



//------------ INTERRUPCION POR Rx UART2 - XBEE ---------------------------
void __attribute__ ( ( interrupt, no_auto_psv ) ) _U2RXInterrupt( void ){
//    Led_Xbee=1;
   if(U2STAbits.URXDA == 1){
     Xbee_RxBuff[index2] = U2RXREG;
     if(Xbee_RxBuff[index2]==0x0A){          //FIN de Trama?
       index2=0;
       flag_xbee=1;

     if(Xbee_RxBuff[index2]==0x56){         //request estado?
        control_xbee=1;
     }
     if(((Xbee_RxBuff[index2]==0x41))){      //Actualizar?
        control_xbee=2;
     }

     }
     index2++;
   }
IFS1bits.U2RXIF = false;  // bajamos bandera de Rx de datos para seguir recibiendo datos
//    Led_Xbee=0;
}



//************************************ FUNCIONES ********************************
// funcion para obtener el tamaño del mensaje
void Obtener_Tam_msg(unsigned char *msg){
unsigned char k=0;
   while(msg[k]!=0x7C){
     k++;
   }
Length_Data=k;         //Salva el tamaño de la longitud del msg-DATA
}

//---------------- FUNCIONES PARA EL BLE -----------------
void send_comando_BLE(unsigned char comando){
 Led_TxBLE_SetHigh();
```

```
 UART1_Write(comando);   //Inicia escaneo
 __delay_ms(1);
 UART1_Write(0x0D);
 __delay_ms(1);
 UART1_Write(0x0A);
 Led_TxBLE_SetLow();
}

void send_msg_BLE(unsigned char *msg){
unsigned char m;
Obtener_Tam_msg(msg);
 Led_TxBLE_SetHigh();
    for(m=0;m<Length_Data;m++){
       UART1_Write(msg[m]);    //Escribe en el Uart1
       __delay_ms(1);
    }
 Led_TxBLE_SetLow();
}

void send_msg_Xbee(unsigned char *msg){
unsigned char y;
//Obtener_Tam_msg(msg);
 Led_TxXBEE_SetHigh();
    for(y=0;y<15;y++){
       UART2_Write(msg[y]);    //Escribe en el Uart2
       __delay_ms(1);
    }
    UART2_Write(0x0A);
 Led_TxXBEE_SetLow();
}

//---------------------------------------------------------------------------

void Tx_TramaAPP_UART3(){
unsigned char n;
float V1,V2,V3;
float Rth1,Rth2,Rth3;
float T1,T2,T3;

float f1,f2;
int i1,i2;
char salida[10];
char T[10];


for(n=0;n<14;n++){
    if(Data_T[n]>0x40){
       T[n]=Data_T[n] - 0x31;
    }
    else{
       T[n]=Data_T[n] - 0x30;
    }
}

V1 = (256*T[1] + 16*T[2] + T[3]);
V1  = V1/1000;
V2 = (256*T[6] + 16*T[7] + T[8]);
V2  = V2/1000;
V3 = (256*T[11] + 16*T[12] + T[13]);
V3  = V3/1000;

Rth1 = (5.6*V1)/(3.3-V1);
Rth2 = (5.6*V2)/(3.3-V2);
Rth3 = (5.6*V3)/(3.3-V3);

Rth1=logf(Rth1);
Rth2=logf(Rth2);
Rth3=logf(Rth3);

T1 = 1/((Rth1/4100)+(1/298.15));
T1 = T1 - 273.15;
T2 = 1/((Rth2/4100)+(1/298.15));
```

```
T2 = T2 - 273.15;
T3 = 1/((Rth3/4100)+(1/298.15));
T3 = T3 - 273.15;

if(T1<=0){
  T1=0.00;
}
if(T2<=0){
  T2=0.00;
}
if(T3<=0){
  T3=0.00;
}

   f1 = floor(T1);
   f2 = T1 - f1;
   i1 = (int)f1;
   i2 = (int)100*f2;
   sprintf(salida,"%d,%d", i1,i2);    //Convert Float to Ascci

Buff_TxU3[0]=salida[0];
Buff_TxU3[1]=salida[1];
Buff_TxU3[2]=salida[2];
Buff_TxU3[3]=salida[3];
Buff_TxU3[4]=salida[4];
Buff_TxU3[5]=0x7C;  //|

   f1 = floor(T2);
   f2 = T2 - f1;
   i1 = (int)f1;
   i2 = (int)100*f2;
   sprintf(salida,"%d,%d", i1,i2);    //Convert Float to Ascci

Buff_TxU3[6]=salida[0];
Buff_TxU3[7]=salida[1];
Buff_TxU3[8]=salida[2];
Buff_TxU3[9]=salida[3];
Buff_TxU3[10]=salida[4];
Buff_TxU3[11]=0x7C;  //|

   f1 = floor(T3);
   f2 = T3 - f1;
   i1 = (int)f1;
   i2 = (int)100*f2;
   sprintf(salida,"%d,%d", i1,i2);    //Convert Float to Ascci

Buff_TxU3[12]=salida[0];
Buff_TxU3[13]=salida[1];
Buff_TxU3[14]=salida[2];
Buff_TxU3[15]=salida[3];
Buff_TxU3[16]=salida[4];
//Buff_TxU3[17]=0x7C;  //|

   for(n=0;n<18;n++){
     UART3_Write(Buff_TxU3[n]);       //Escribe en el Uart2
//     Led_TxU3=1;
     __delay_ms(1);
//     Led_Xbee=0;
   }
}
//---------------------------------------------------------------------
void Data_zero(){
   Data_T[z]=0x30;
   Data_T[z+1]=0x30;
   Data_T[z+2]=0x30;
   Data_T[z+3]=0x30;
}
//_____________________________________________________________________


//---------------------------------------------------------------------
//-------------------------- PROGRAMA PRINCIPAL ---------------------------
```

```
int main(void){
    // initialize the device
    SYSTEM_Initialize();
    Led_TxU3_SetHigh();
    contador=0;
    cont_error=0;
    index1=0;
    index2=0;
    __delay_ms(1000);

//*** Inicializa Xbee ***
    Reset_XBEE_SetHigh();

//*** Inicializa BLE ***
    CMD_mdlp_SetHigh();   // MDLP=1 (data), CMD=0
    Wakesw_SetHigh();     // Modo: Dormido=0, Despierta=1

//------------------------------ BUCLE PRINCIPAL -----------------------------
    while(1){

    if(cont_error>=10){
        CMD_mdlp_SetLow();   // MDLP=1 (data), CMD=0
        Wakesw_SetLow();     // Modo: Dormido=0, Despierta=1
        __delay_ms(1500);
        CMD_mdlp_SetHigh();   // MDLP=1 (data), CMD=0
        Wakesw_SetHigh();     // Modo: Dormido=0, Despierta=1
        __delay_ms(1500);
        cont_error=0;
        contador=0;
        control_BLE=0;
//        send_msg_BLE(Reset);    //R,1   (reset Central

    }

//-------------------------------------------------------------------------
//--------------------------- Conexión con BLEs  ----------------------------

    if(contador==10){
        Led_TxU3_SetHigh();
        z=0;
        control_BLE=0;
        send_msg_BLE(conex_BLE1);//Establecer conexion con BLE1(E,0,001EC037A8A3)
        __delay_ms(1000);        //Conexion establecida
        if(control_BLE==4){
            send_msg_BLE(request_T); //Lee el valor de la Temperatura (CHR,000E)
            __delay_ms(500);
            send_comando_BLE(0x4B);  //"K": Finaliza la Conexion
            __delay_ms(50);
        }
        else{
            Data_zero();            //Coloca a cero todos los registros
            send_comando_BLE(0x5A);  //"Z": Detiene Conexion del comando E
            cont_error++;
        }

        Led_TxU3_SetHigh();
        z=5;
        control_BLE=0;
        send_msg_BLE(conex_BLE2);   //Establecer conexion con BLE2(E,0,001EC037A8A3)
        __delay_ms(1000);         //Conexion establecida
        if(control_BLE==4){
            send_msg_BLE(request_T); //Lee el valor de la Temperatura (CHR,000E)
            __delay_ms(500);
            send_comando_BLE(0x4B);  //"K": Finaliza la Conexion
            __delay_ms(50);
        }
        else{
            Data_zero();            //Coloca a cero todos los registros
            send_comando_BLE(0x5A);  //"Z": Detiene Conexion del comando E
            cont_error++;
        }
```

```
        Led_TxU3_SetHigh();
        z=10;
        control_BLE=0;
        send_msg_BLE(conex_BLE3);//Establecer conexion con BLE3(E,0,001EC037A8A3)
        __delay_ms(1000);         //Conexion establecida
        if(control_BLE==4){
           send_msg_BLE(request_T); //Lee el valor de la Temperatura (CHR,000E)
           __delay_ms(500);
           send_comando_BLE(0x4B);  //"K": Finaliza la Conexion
           __delay_ms(50);
        }
        else{
           Data_zero();            //Coloca a cero todos los registros
           send_comando_BLE(0x5A);  //"Z": Detiene Conexion del comando E
           cont_error++;
        }

        send_msg_Xbee(Data_T);   //Tx "RF-Data"(msgx) al coordinador
        Tx_TramaAPP_UART3();     //Envia Data al APP by Bluetooth

     }


     if(contador>=20){
       contador=0;
     }

     contador++;
     __delay_ms(100);

     }
}
/**
 End of File
*/
```

```
//INTERRUP_MANAGER.C
/**
   Section: Includes
*/
#include <xc.h>

/**
   void INTERRUPT_Initialize (void)
*/
void INTERRUPT_Initialize (void)
{
    //    UERI: U2E - UART2 Error
    //    Priority: 1
        IPC16bits.U2ERIP = 1;
    //    UTXI: U2TX - UART2 Transmitter
    //    Priority: 1
        IPC7bits.U2TXIP = 1;
    //    URXI: U2RX - UART2 Receiver
    //    Priority: 3
        IPC7bits.U2RXIP = 3;
    //    UERI: U1E - UART1 Error
    //    Priority: 1
        IPC16bits.U1ERIP = 1;
    //    UTXI: U1TX - UART1 Transmitter
    //    Priority: 1
        IPC3bits.U1TXIP = 1;
    //    URXI: U1RX - UART1 Receiver
```

```
    //    Priority: 5
        IPC2bits.U1RXIP = 5;
    //    TI: T1 - Timer1
    //    Priority: 2
        IPC0bits.T1IP = 2;

}
```

//mcc.c

```
// CONFIG4
#pragma config DSWDTPS = DSWDTPS1F    // Deep Sleep Watchdog Timer Postscale Select bits->1:68719476736 (25.7 Days)
#pragma config DSWDTOSC = LPRC    // DSWDT Reference Clock Select->DSWDT uses LPRC as reference clock
#pragma config DSBOREN = ON    // Deep Sleep BOR Enable bit->DSBOR Enabled
#pragma config DSWDTEN = ON    // Deep Sleep Watchdog Timer Enable->DSWDT Enabled
#pragma config DSSWEN = ON    // DSEN Bit Enable->Deep Sleep is controlled by the register bit DSEN
#pragma config PLLDIV = DIVIDE2    // USB 96 MHz PLL Prescaler Select bits->Oscillator input divided by 2 (8 MHz input)
#pragma config I2C1SEL = DISABLE    // Alternate I2C1 enable bit->I2C1 uses SCL1 and SDA1 pins
#pragma config IOL1WAY = ON    // PPS IOLOCK Set Only Once Enable bit->Once set, the IOLOCK bit cannot be cleared

// CONFIG3
#pragma config WPFP = WPFP127    // Write Protection Flash Page Segment Boundary->Page 127 (0x1FC00)
#pragma config SOSCSEL = OFF    // SOSC Selection bits->Digital (SCLKI) mode
#pragma config WDTWIN = PS25_0    // Window Mode Watchdog Timer Window Width Select->Watch Dog Timer Window Width is 25 percent
#pragma config PLLSS = PLL_FRC    // PLL Secondary Selection Configuration bit->PLL is fed by the on-chip Fast RC (FRC) oscillator
#pragma config BOREN = ON    // Brown-out Reset Enable->Brown-out Reset Enable
#pragma config WPDIS = WPDIS    // Segment Write Protection Disable->Disabled
#pragma config WPCFG = WPCFGDIS    // Write Protect Configuration Page Select->Disabled
#pragma config WPEND = WPENDMEM    // Segment Write Protection End Page Select->Write Protect from WPFP to the last page of memory

// CONFIG2
#pragma config POSCMD = NONE    // Primary Oscillator Select->Primary Oscillator Disabled
#pragma config WDTCLK = LPRC    // WDT Clock Source Select bits->WDT uses LPRC
#pragma config OSCIOFCN = OFF    // OSCO Pin Configuration->OSCO/CLKO/RA3 functions as CLKO (FOSC/2)
#pragma config FCKSM = CSDCMD    // Clock Switching and Fail-Safe Clock Monitor Configuration bits->Clock switching and Fail-Safe Clock Monitor are disabled
#pragma config FNOSC = FRCPLL    // Initial Oscillator Select->Fast RC Oscillator with PLL module (FRCPLL)
#pragma config ALTRB6 = APPEND    // Alternate RB6 pin function enable bit->Append the RP6/ASCL1/PMPD6 functions of RB6 to RA1 pin functions
#pragma config ALTCMPI = CxINC_RB    // Alternate Comparator Input bit->C1INC is on RB13, C2INC is on RB9 and C3INC is on RA0
#pragma config WDTCMX = WDTCLK    // WDT Clock Source Select bits->WDT clock source is determined by the WDTCLK Configuration bits
#pragma config IESO = ON    // Internal External Switchover->Enabled
```

```
// CONFIG1
#pragma config WDTPS = PS32768    // Watchdog Timer Postscaler Select->1:32768
#pragma config FWPSA = PR128    // WDT Prescaler Ratio Select->1:128
#pragma config WINDIS = OFF    // Windowed WDT Disable->Standard Watchdog Timer
#pragma config FWDTEN = OFF    // Watchdog Timer Enable->WDT disabled in hardware; SWDTEN
bit disabled
#pragma config ICS = PGx1    // Emulator Pin Placement Select bits->Emulator functions are shared with
PGEC1/PGED1
#pragma config LPCFG = OFF    // Low power regulator control->Disabled - regardless of RETEN
#pragma config GWRP = OFF    // General Segment Write Protect->Write to program memory allowed
#pragma config GCP = OFF    // General Segment Code Protect->Code protection is disabled
#pragma config JTAGEN = OFF    // JTAG Port Enable->Disabled

#include "mcc.h"

void SYSTEM_Initialize(void)
{
    PIN_MANAGER_Initialize();
    INTERRUPT_Initialize();
    OSCILLATOR_Initialize();
    UART2_Initialize();
    UART1_Initialize();
    UART3_Initialize();
    TMR1_Initialize();
}

void OSCILLATOR_Initialize(void)
{
    // CF no clock failure; NOSC FRCPLL; SOSCEN disabled; POSCEN disabled; CLKLOCK unlocked;
OSWEN Switch is Complete; IOLOCK not-active;
    __builtin_write_OSCCONL((uint8_t) (0x0100 & 0x00FF));
    // CPDIV 1:1; PLLEN enabled; RCDIV FRC/1; DOZE 1:8; DOZEN disabled; ROI disabled;
    CLKDIV = 0x3020;
    // STOR disabled; STORPOL Interrupt when STOR is 1; STSIDL disabled; STLPOL Interrupt when
STLOCK is 1; STLOCK disabled; STSRC SOSC; STEN disabled; TUN Center frequency;
    OSCTUN = 0x0000;
    // ROEN disabled; ROSEL FOSC; ROSIDL disabled; ROSWEN disabled; ROOUT disabled; ROSLP
disabled;
    REFOCONL = 0x0000;
    // RODIV 0;
    REFOCONH = 0x0000;
    // ROTRIM 0;
    REFOTRIML = 0x0000;
}


//Pin_manager.c


/**
    Section: Includes
*/
#include <xc.h>
#include "pin_manager.h"

/**
```

```
    void PIN_MANAGER_Initialize(void)
*/
void PIN_MANAGER_Initialize(void)
{
    /****************************************************************************
     * Setting the Output Latch SFR(s)
     ***************************************************************************/
    LATA = 0x0000;
    LATB = 0x0000;
    LATC = 0x0000;

    /****************************************************************************
     * Setting the GPIO Direction SFR(s)
     ***************************************************************************/
    TRISA = 0x038F;
    TRISB = 0xE287;
    TRISC = 0x01FA;

    /****************************************************************************
     * Setting the Weak Pull Up and Weak Pull Down SFR(s)
     ***************************************************************************/
    CNPD1 = 0x0000;
    CNPD2 = 0x0000;
    CNPD3 = 0x0000;
    CNPU1 = 0x0000;
    CNPU2 = 0x0000;
    CNPU3 = 0x0000;

    /****************************************************************************
     * Setting the Open Drain SFR(s)
     ***************************************************************************/
    ODCA = 0x0000;
    ODCB = 0x0000;
    ODCC = 0x0000;

    /****************************************************************************
     * Setting the Analog/Digital Configuration SFR(s)
     ***************************************************************************/
    ANSA = 0x000F;
    ANSB = 0xC04C;
    ANSC = 0x0001;

    /****************************************************************************
     * Set the PPS
     ***************************************************************************/
    __builtin_write_OSCCONL(OSCCON & 0xbf); // unlock PPS

    RPOR5bits.RP11R = 0x0003;   //RB11->UART1:U1TX;
    RPINR18bits.U1RXR = 0x000D;   //RB13->UART1:U1RX;
    RPOR9bits.RP18R = 0x0005;   //RC2->UART2:U2TX;
    RPOR4bits.RP8R = 0x0013;   //RB8->UART3:U3TX;
    RPINR17bits.U3RXR = 0x0009;   //RB9->UART3:U3RX;
    RPINR19bits.U2RXR = 0x0011;   //RC1->UART2:U2RX;

    __builtin_write_OSCCONL(OSCCON | 0x40); // lock   PPS
}
```

//Tmr1.c

```
#include <xc.h>
#include "tmr1.h"

/**
  Section: Data Type Definitions
*/

/** TMR Driver Hardware Instance Object

  @Summary
    Defines the object required for the maintainence of the hardware instance.

  @Description
    This defines the object required for the maintainence of the hardware
    instance. This object exists once per hardware instance of the peripheral.

  Remarks:
    None.
*/

typedef struct _TMR_OBJ_STRUCT
{
    /* Timer Elapsed */
    bool                                timerElapsed;
    /*Software Counter value*/
    uint8_t                              count;

} TMR_OBJ;

static TMR_OBJ tmr1_obj;

/**
  Section: Driver Interface
*/


void TMR1_Initialize (void)
{
    //TMR1 16;
    TMR1 = 0x0010;
    //Period = 0.25 s; Frequency = 16000000 Hz; PR1 15625;
    PR1 = 0x3D09;
    //TCKPS 1:256; TON enabled; TSIDL disabled; TCS FOSC/2; TECS SOSC; TSYNC disabled;
TGATE disabled;
    T1CON = 0x8030;


    IFS0bits.T1IF = false;
    IEC0bits.T1IE = true;

    tmr1_obj.timerElapsed = false;

}
```

//Uart1.c

```
/**
  Section: Included Files
*/

#include "uart1.h"

typedef union
{
   struct
   {
         uint8_t full:1;
         uint8_t empty:1;
         uint8_t reserved:6;
   }s;
   uint8_t status;
}

UART_BYTEQ_STATUS;

/** UART Driver Hardware Instance Object

  @Summary
    Defines the object required for the maintenance of the hardware instance.

*/
typedef struct
{
   /* RX Byte Q */
   uint8_t                         *rxTail ;

   uint8_t                         *rxHead ;

   /* TX Byte Q */
   uint8_t                         *txTail ;

   uint8_t                         *txHead ;

   UART_BYTEQ_STATUS                    rxStatus ;

   UART_BYTEQ_STATUS                    txStatus ;

} UART_OBJECT ;

static UART_OBJECT uart1_obj ;

/** UART Driver Queue Length

  @Summary
    Defines the length of the Transmit and Receive Buffers

*/
```

```
#define UART1_CONFIG_TX_BYTEQ_LENGTH 16
#define UART1_CONFIG_RX_BYTEQ_LENGTH 32


/** UART Driver Queue

  @Summary
    Defines the Transmit and Receive Buffers

*/

static uint8_t uart1_txByteQ[UART1_CONFIG_TX_BYTEQ_LENGTH] ;
static uint8_t uart1_rxByteQ[UART1_CONFIG_RX_BYTEQ_LENGTH] ;


/**
  Section: Driver Interface
*/


void UART1_Initialize (void)
{
   // STSEL 1; IREN disabled; PDSEL 8N; UARTEN enabled; RTSMD disabled; USIDL disabled;
WAKE disabled; ABAUD disabled; LPBACK disabled; BRGH enabled; URXINV disabled; UEN
TX_RX;
   U1MODE = (0x8008 & ~(1<<15));  // disabling UARTEN bit
   // UTXISEL0 TX_ONE_CHAR; UTXINV disabled; URXEN disabled; OERR NO_ERROR_cleared;
URXISEL RX_ONE_CHAR; UTXBRK COMPLETED; UTXEN disabled; ADDEN disabled;
   U1STA = 0x0000;
   // BaudRate = 115200; Frequency = 16000000 Hz; U1BRG 34;
   U1BRG = 0x0022;
   // ADMADDR 0; ADMMASK 0;
   U1ADMD = 0x0000;
   // T0PD 1 ETU; PTRCL T0; TXRPT Retransmits the error byte once; CONV Direct; SCEN disabled;
   U1SCCON = 0x0000;
   // TXRPTIF disabled; TXRPTIE disabled; WTCIF disabled; WTCIE disabled; PARIE disabled; GTCIF
disabled; GTCIE disabled; RXRPTIE disabled; RXRPTIF disabled;
   U1SCINT = 0x0000;
   // GTC 0;
   U1GTC = 0x0000;
   // WTCL 0;
   U1WTCL = 0x0000;
   // WTCH 0;
   U1WTCH = 0x0000;

   IEC0bits.U1RXIE = 1;

    //Make sure to set LAT bit corresponding to TxPin as high before UART initialization
   U1MODEbits.UARTEN = 1;  // enabling UART ON bit
   U1STAbits.UTXEN = 1;



   uart1_obj.txHead = uart1_txByteQ;
   uart1_obj.txTail = uart1_txByteQ;
   uart1_obj.rxHead = uart1_rxByteQ;
   uart1_obj.rxTail = uart1_rxByteQ;
```

```
  uart1_obj.rxStatus.s.empty = true;
  uart1_obj.txStatus.s.empty = true;
  uart1_obj.txStatus.s.full = false;
  uart1_obj.rxStatus.s.full = false;
}



void UART1_Write(const uint8_t byte)
{
   while(U1STAbits.UTXBF == 1)
   {

   }

   U1TXREG = byte;    // Write the data byte to the USART.
}
```

Uart2.c

```
/**
  Section: Included Files
*/

#include "uart2.h"

/**
  Section: Data Type Definitions
*/

/** UART Driver Queue Status

  @Summary
    Defines the object required for the status of the queue.
*/

typedef union
{
   struct
   {
          uint8_t full:1;
          uint8_t empty:1;
          uint8_t reserved:6;
   }s;
   uint8_t status;
}

UART_BYTEQ_STATUS;

/** UART Driver Hardware Instance Object

  @Summary
    Defines the object required for the maintenance of the hardware instance.
```

```
*/
typedef struct
{
    /* RX Byte Q */
    uint8_t                 *rxTail ;

    uint8_t                 *rxHead ;

    /* TX Byte Q */
    uint8_t                 *txTail ;

    uint8_t                 *txHead ;

    UART_BYTEQ_STATUS               rxStatus ;

    UART_BYTEQ_STATUS               txStatus ;

} UART_OBJECT ;

static UART_OBJECT uart2_obj ;

/** UART Driver Queue Length

  @Summary
    Defines the length of the Transmit and Receive Buffers

*/

#define UART2_CONFIG_TX_BYTEQ_LENGTH 8
#define UART2_CONFIG_RX_BYTEQ_LENGTH 16


/** UART Driver Queue

  @Summary
    Defines the Transmit and Receive Buffers

*/

static uint8_t uart2_txByteQ[UART2_CONFIG_TX_BYTEQ_LENGTH] ;
static uint8_t uart2_rxByteQ[UART2_CONFIG_RX_BYTEQ_LENGTH] ;

/** UART Hardware FIFO Buffer Length

  @Summary
    Defines the length of the Transmit and Receive FIFOs

*/

#define UART2_TX_FIFO_LENGTH 1
#define UART2_RX_FIFO_LENGTH 1

/**
  Section: Driver Interface
*/
```

```
void UART2_Initialize (void)
{
    // STSEL 1; IREN disabled; PDSEL 8N; UARTEN enabled; RTSMD disabled; USIDL disabled;
WAKE disabled; ABAUD disabled; LPBACK disabled; BRGH enabled; URXINV disabled; UEN
TX_RX;
    U2MODE = 0x8008;
    // OERR NO_ERROR_cleared; URXISEL RX_ONE_CHAR; UTXBRK COMPLETED; UTXEN
enabled; ADDEN disabled; UTXISEL0 TX_ONE_CHAR; UTXINV disabled;
    U2STA = 0x0400;
    // LAST disabled; U2TXREG 0;
    U2TXREG = 0x0000;
    // BaudRate = 9600; Frequency = 16000000 Hz; U2BRG 416;
    U2BRG = 0x01A0;
    // ADMADDR 0; ADMMASK 0;
    U2ADMD = 0x0000;
    // T0PD 1 ETU; PTRCL T0; TXRPT Retransmits the error byte once; CONV Direct; SCEN disabled;
    U2SCCON = 0x0000;
    // TXRPTIF disabled; TXRPTIE disabled; WTCIF disabled; WTCIE disabled; PARIE disabled; GTCIF
disabled; GTCIE disabled; RXRPTIE disabled; RXRPTIF disabled;
    U2SCINT = 0x0000;
    // GTC 0;
    U2GTC = 0x0000;
    // WTCL 0;
    U2WTCL = 0x0000;
    // WTCH 0;
    U2WTCH = 0x0000;

    IEC1bits.U2RXIE = 1;

    U2STAbits.UTXEN = 1;

    uart2_obj.txHead = uart2_txByteQ;
    uart2_obj.txTail = uart2_txByteQ;
    uart2_obj.rxHead = uart2_rxByteQ;
    uart2_obj.rxTail = uart2_rxByteQ;
    uart2_obj.rxStatus.s.empty = true;
    uart2_obj.txStatus.s.empty = true;
    uart2_obj.txStatus.s.full = false;
    uart2_obj.rxStatus.s.full = false;
}



/**
    Maintains the driver's transmitter state machine and implements its ISR
*/
void __attribute__ ( ( interrupt, no_auto_psv ) ) _U2TXInterrupt ( void )
{
    if(uart2_obj.txStatus.s.empty)
    {
        IEC1bits.U2TXIE = false;
        return;
    }

    IFS1bits.U2TXIF = false;
```

```
    int count = 0;
    while((count < UART2_TX_FIFO_LENGTH)&& !(U2STAbits.UTXBF == 1))
    {
        count++;

        U2TXREG = *uart2_obj.txHead;

        uart2_obj.txHead++;

        if(uart2_obj.txHead == (uart2_txByteQ + UART2_CONFIG_TX_BYTEQ_LENGTH))
        {
            uart2_obj.txHead = uart2_txByteQ;
        }

        uart2_obj.txStatus.s.full = false;

        if(uart2_obj.txHead == uart2_obj.txTail)
        {
            uart2_obj.txStatus.s.empty = true;
            break;
        }
    }
}




void __attribute__ ( ( interrupt, no_auto_psv ) ) _U2ErrInterrupt ( void )
{
    if ((U2STAbits.OERR == 1))
    {
        U2STAbits.OERR = 0;
    }

    IFS4bits.U2ERIF = false;
}

/**
  Section: UART Driver Client Routines
*/

uint8_t UART2_Read( void)
{
    uint8_t data = 0;

    data = *uart2_obj.rxHead;

    uart2_obj.rxHead++;

    if (uart2_obj.rxHead == (uart2_rxByteQ + UART2_CONFIG_RX_BYTEQ_LENGTH))
    {
        uart2_obj.rxHead = uart2_rxByteQ;
    }

    if (uart2_obj.rxHead == uart2_obj.rxTail)
    {
```

```
        uart2_obj.rxStatus.s.empty = true;
    }

    uart2_obj.rxStatus.s.full = false;

    return data;
}


unsigned int UART2_ReadBuffer( uint8_t *buffer, const unsigned int bufLen)
{
    unsigned int numBytesRead = 0 ;
    while ( numBytesRead < ( bufLen ))
    {
        if( uart2_obj.rxStatus.s.empty)
        {
            break;
        }
        else
        {
            buffer[numBytesRead++] = UART2_Read () ;
        }
    }

    return numBytesRead ;
}



void UART2_Write( const uint8_t byte)
{
    IEC1bits.U2TXIE = false;

    *uart2_obj.txTail = byte;

    uart2_obj.txTail++;

    if (uart2_obj.txTail == (uart2_txByteQ + UART2_CONFIG_TX_BYTEQ_LENGTH))
    {
        uart2_obj.txTail = uart2_txByteQ;
    }

    uart2_obj.txStatus.s.empty = false;

    if (uart2_obj.txHead == uart2_obj.txTail)
    {
        uart2_obj.txStatus.s.full = true;
    }

    IEC1bits.U2TXIE = true ;

}


unsigned int UART2_WriteBuffer( const uint8_t *buffer , const unsigned int bufLen )
{
    unsigned int numBytesWritten = 0 ;
```

```
    while ( numBytesWritten < ( bufLen ))
    {
        if((uart2_obj.txStatus.s.full))
        {
            break;
        }
        else
        {
            UART2_Write (buffer[numBytesWritten++] ) ;
        }
    }

    return numBytesWritten ;

}


UART2_TRANSFER_STATUS UART2_TransferStatusGet (void )
{
    UART2_TRANSFER_STATUS status = 0;

    if(uart2_obj.txStatus.s.full)
    {
        status |= UART2_TRANSFER_STATUS_TX_FULL;
    }

    if(uart2_obj.txStatus.s.empty)
    {
        status |= UART2_TRANSFER_STATUS_TX_EMPTY;
    }

    if(uart2_obj.rxStatus.s.full)
    {
        status |= UART2_TRANSFER_STATUS_RX_FULL;
    }

    if(uart2_obj.rxStatus.s.empty)
    {
        status |= UART2_TRANSFER_STATUS_RX_EMPTY;
    }
    else
    {
        status |= UART2_TRANSFER_STATUS_RX_DATA_PRESENT;
    }
    return status;
}


uint8_t UART2_Peek(uint16_t offset)
{
    if( (uart2_obj.rxHead + offset) > (uart2_rxByteQ + UART2_CONFIG_RX_BYTEQ_LENGTH))
    {
      return uart2_rxByteQ[offset - (uart2_rxByteQ + UART2_CONFIG_RX_BYTEQ_LENGTH -
uart2_obj.rxHead)];
    }
    else
```

```
    {
     return *(uart2_obj.rxHead + offset);
    }
}


unsigned int UART2_ReceiveBufferSizeGet(void)
{
    if(!uart2_obj.rxStatus.s.full)
    {
        if(uart2_obj.rxHead > uart2_obj.rxTail)
        {
            return(uart2_obj.rxHead - uart2_obj.rxTail);
        }
        else
        {
            return(UART2_CONFIG_RX_BYTEQ_LENGTH - (uart2_obj.rxTail - uart2_obj.rxHead));
        }
    }
    return 0;
}


unsigned int UART2_TransmitBufferSizeGet(void)
{
    if(!uart2_obj.txStatus.s.full)
    {
        if(uart2_obj.txHead > uart2_obj.txTail)
        {
            return(uart2_obj.txHead - uart2_obj.txTail);
        }
        else
        {
            return(UART2_CONFIG_TX_BYTEQ_LENGTH - (uart2_obj.txTail - uart2_obj.txHead));
        }
    }
    return 0;
}


bool UART2_ReceiveBufferIsEmpty (void)
{
    return(uart2_obj.rxStatus.s.empty);
}


bool UART2_TransmitBufferIsFull(void)
{
    return(uart2_obj.txStatus.s.full);
}


UART2_STATUS UART2_StatusGet (void)
{
    return U2STA;
}
```

```
/**
  Section: Included Files
*/
#include "uart3.h"

/**
  Section: UART3 APIs
*/

void UART3_Initialize(void)
{
/**
    Set the UART3 module to the options selected in the user interface.
    Make sure to set LAT bit corresponding to TxPin as high before UART initialization
*/
   // STSEL 1; IREN disabled; PDSEL 8N; UARTEN enabled; RTSMD disabled; USIDL disabled;
WAKE disabled; ABAUD disabled; LPBACK disabled; BRGH enabled; URXINV disabled; UEN
TX_RX;
   U3MODE = (0x8008 & ~(1<<15));  // disabling UARTEN bit
   // UTXISEL0 TX_ONE_CHAR; UTXINV disabled; URXEN disabled; OERR NO_ERROR_cleared;
URXISEL RX_ONE_CHAR; UTXBRK COMPLETED; UTXEN disabled; ADDEN disabled;
   U3STA = 0x0000;
   // BaudRate = 9600; Frequency = 16000000 Hz; U3BRG 416;
   U3BRG = 0x01A0;
   // ADMADDR 0; ADMMASK 0;
   U3ADMD = 0x0000;

   U3MODEbits.UARTEN = 1;  // enabling UARTEN bit
   U3STAbits.UTXEN = 1;

}


uint8_t UART3_Read(void)
{
   while(!(U3STAbits.URXDA == 1))
   {

   }

   if ((U3STAbits.OERR == 1))
   {
      U3STAbits.OERR = 0;
   }



   return U3RXREG;
}

void UART3_Write(uint8_t txData)
{
   while(U3STAbits.UTXBF == 1)
   {
```

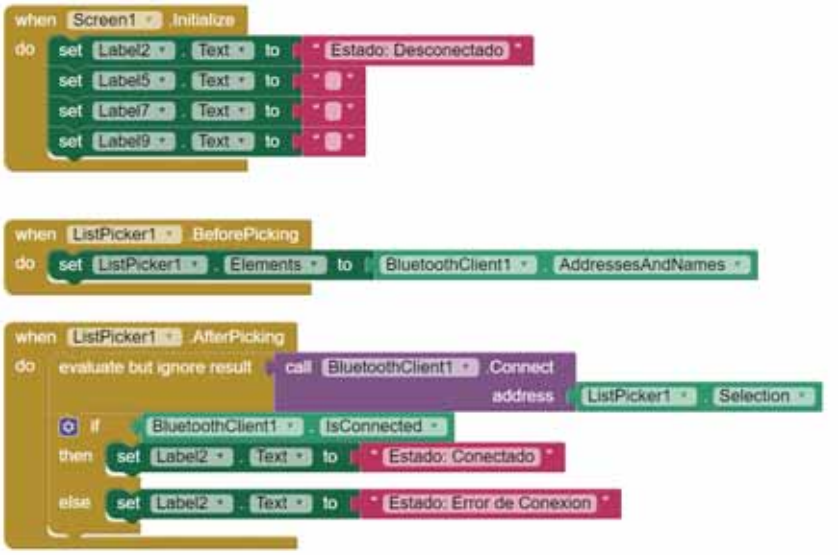
```
    }

    U3TXREG = txData;    // Write the data byte to the USART.
}

UART3_STATUS UART3_StatusGet (void)
{
    return U3STA;
}


/**
  End of File
*/
```

ANEXO 2: Código de Programa de la aplicación Android realizada en APP Inventor



ANEXO 4: Pruebas de mediciones de temperatura del prototipo

FECHA / HORA	T_1	T_2	T_3
10/02/2019 07:45	8.00081	8	8
10/02/2019 07:46	8.0042	8.17096	8.00395
10/02/2019 07:47	8.02737	8.19475	8.03264
10/02/2019 07:48	8.05635	8.22361	8.06165
10/02/2019 07:49	8.08286	8.24979	8.08803
10/02/2019 07:50	8.11222	8.27918	8.11477
10/02/2019 07:51	8.13841	8.30578	8.14208
10/02/2019 07:52	8.16455	8.33164	8.16804
10/02/2019 07:53	8.19278	8.35964	8.19249
10/02/2019 07:54	8.221	8.38763	8.21707
10/02/2019 07:55	8.25015	8.41716	8.24284
10/02/2019 07:56	8.27643	8.44323	8.27263
10/02/2019 07:57	8.30326	8.47057	8.29897
10/02/2019 07:58	8.33085	8.49781	8.32578
10/02/2019 07:59	8.35669	8.52393	8.35454
10/02/2019 08:00	8.38135	8.54826	8.38164
10/02/2019 08:01	8.40557	8.57272	8.41011
10/02/2019 08:02	8.43133	8.5982	8.44145
10/02/2019 08:02	8.43193	8.59883	8.4415
10/02/2019 08:03	8.46123	8.62794	8.46684
10/02/2019 08:04	8.48781	8.6548	8.49156
10/02/2019 08:05	8.51398	8.68139	8.51939
10/02/2019 08:06	8.54353	8.71022	8.54617
10/02/2019 08:07	8.57049	8.73716	8.57667
10/02/2019 08:08	8.59873	8.76617	8.60504
10/02/2019 08:09	8.62953	8.79636	8.6315
10/02/2019 08:10	8.65533	8.82225	8.65813
10/02/2019 08:11	8.68005	8.84708	8.68288
10/02/2019 08:11	8.68029	8.84755	8.68309

MICROCHIP

PIC24FJ128GB204 FAMILY

28/44-Pin, General Purpose, 16-Bit Flash Microcontrollers with Cryptographic Engine, ISO 7816, USB On-The-Go and XLP Technology

Cryptographic Engine

- AES Engine with 128,192 or 256-Bit Key
- Supports ECB, CBC, OFB, CTR and CFB128 modes
- DES/Triple DES (TDES) Engine: Supports 2-Key and 3-Key EDE or DED TDES
- Supports up to Three Unique Keys for TDES
- Programmatically Secure
- True Random Number Generator
- Pseudorandom Number Generator
- Non-Readable, On-Chip, OTP Key Storages

Universal Serial Bus Features

- USB v2.0 On-The-Go (OTG) Compliant
- Dual Role Capable; can Act as Either Host or Peripheral
- Low-Speed (1.5 Mb/s) and Full-Speed (12 Mb/s) USB Operation in Host mode
- Full-Speed USB Operation in Device mode
- High-Precision PLL for USB
- USB Device mode Operation from FRC Oscillator:
  - No crystal oscillator required
- Supports up to 32 Endpoints (16 bidirectional):
  - USB module can use any RAM locations on the device as USB endpoint buffers
- On-Chip USB Transceiver
- Supports Control, Interrupt, Isochronous and Bulk Transfers
- On-Chip Pull-up and Pull-Down Resistors

Extreme Low-Power Features

- Multiple Power Management Options for Extreme Power Reduction:
  - V_{BAT} allows the device to transition to a backup battery for the lowest power consumption with RTCC
  - Deep Sleep allows near total power-down, with the ability to wake-up on internal or external triggers
  - Sleep and Idle modes selectively shut down peripherals and/or core for substantial power reduction and fast wake-up
  - Doze mode allows CPU to run at a lower clock speed than peripherals
- Alternate Clock modes allow On-the-Fly Switching to a Lower Clock Speed for Selective Power Reduction
- Extreme Low-Power Current Consumption for Deep Sleep:
  - WDT: 270 nA @ 3.3V typical
  - RTCC: 400 nA @ 32 kHz, 3.3V typical
  - Deep Sleep current: 40 nA, 3.3V typical

Device	Memory		Pins	Analog Peripherals			Digital Peripherals							USB OTG	Deep Sleep w/V_{BAT}	AES/DES Cryptographic
	Program Flash (bytes)	Data RAM (bytes)		10/12-Bit A/D (ch)	Comparators	CTMU (ch)	Input Capture	Output Compare/PWM	I²C™	SPI	UART w/IrDA®, 7816	EPMP/PSP	16-Bit Timers			
PIC24FJ128GB204	128K	8K	44	12	3	12	6	6	2	3	4	Y	5	Y	Y	Y
PIC24FJ128GB202	128K	8K	28	9	3	9	6	6	2	3	4	N	5	Y	Y	Y
PIC24FJ64GB204	64K	8K	44	12	3	12	6	6	2	3	4	Y	5	Y	Y	Y
PIC24FJ64GB202	64K	8K	28	9	3	9	6	6	2	3	4	N	5	Y	Y	Y

PIC24FJ128GB204 FAMILY

Analog Features

- 10/12-Bit, 12-Channel Analog-to-Digital (A/D) Converter:
  - Conversion rate of 500 ksps (10-bit), 200 ksps (12-bit)
  - Conversion available during Sleep and Idle
- Three Rail-to-Rail, Enhanced Analog Comparators with Programmable Input/Output Configuration
- Three On-Chip Programmable Voltage References
- Charge Time Measurement Unit (CTMU):
  - Used for capacitive touch sensing, up to 12 channels
  - Time measurement down to 100 ps resolution
  - Operation in Sleep mode

Peripheral Features

- Up to Five External Interrupt Sources
- Peripheral Pin Select (PPS); Allows Independent I/O Mapping of many Peripherals
- Five 16-Bit Timers/Counters with Prescaler:
  - Can be paired as 32-bit timers/counters
- Six-Channel DMA supports All Peripheral modules:
  - Minimizes CPU overhead and increases data throughput
- Six Input Capture modules, each with a Dedicated 16-Bit Timer
- Six Output Compare/PWM modules, each with a Dedicated 16-Bit Timer
- Enhanced Parallel Master/Slave Port (EPMP/EPSP)
- Hardware Real-Time Clock/Calendar (RTCC):
  - Runs in Sleep, Deep Sleep and VBAT modes
- Three 3-Wire/4-Wire SPI modules:
  - Support four Frame modes
  - Variable FIFO buffer
  - I^2S mode
  - Variable width from 2-bit to 32-bit
- Two I^2C™ modules Support Multi-Master/ Slave mode and 7-Bit/10-Bit Addressing
- Four UART modules:
  - Support RS-485, RS-232 and LIN/J2602
  - On-chip hardware encoder/decoder for IrDA®
  - Smart Card ISO 7816 support on UART1 and UART2 only:
    - T = 0 protocol with automatic error handling
    - T = 1 protocol
    - Dedicated Guard Time Counter (GTC)
    - Dedicated Waiting Time Counter (WTC)
  - Auto-wake-up on Auto-Baud Detect (ABD)
  - 4-level deep FIFO buffer
- Programmable 32-Bit Cyclic Redundancy Check (CRC) Generator
- Digital Signal Modulator provides On-Chip FSK and PSK Modulation for a Digital Signal Stream
- High-Current Sink/Source (18 mA/18 mA) on All I/O Pins
- Configurable Open-Drain Outputs on Digital I/O Pins
- 5.5V Tolerant Inputs on Most Pins

High-Performance CPU

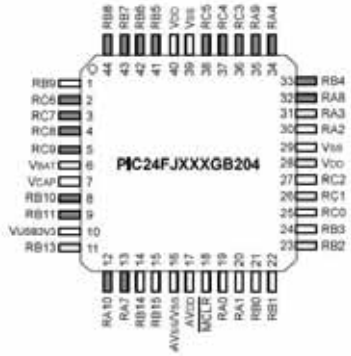
- Modified Harvard Architecture
- Up to 16 MIPS Operation @ 32 MHz
- 8 MHz Internal Oscillator:
  - 96 MHz PLL option
  - Multiple clock divide options
  - Run-time self-calibration capability for maintaining better than ±0.20% accuracy
  - Fast start-up
- 17-Bit x 17-Bit Single-Cycle Hardware Fractional/Integer Multiplier
- 32-Bit by 16-Bit Hardware Divider
- 16 x 16-Bit Working Register Array
- C Compiler Optimized Instruction Set Architecture (ISA)
- Two Address Generation Units (AGUs) for Separate Read and Write Addressing of Data Memory

Special Microcontroller Features

- Supply Voltage Range of 2.0V to 3.6V
- Two On-Chip Voltage Regulators (1.8V and 1.2V) for Regular and Extreme Low-Power Operation
- 20,000 Erase/Write Cycle Endurance Flash Program Memory, Typical
- Flash Data Retention: 20 Years Minimum
- Self-Programmable under Software Control
- Programmable Reference Clock Output
- In-Circuit Serial Programming™ (ICSP™) and In-Circuit Emulation (ICE) via 2 Pins
- JTAG Programming and Boundary Scan Support
- Fail-Safe Clock Monitor (FSCM) Operation:
  - Detects clock failure and switches to on-chip, Low-Power RC Oscillator
- Power-on Reset (POR), Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Separate Brown-out Reset (BOR) and Deep Sleep Brown-out Reset (DSBOR) Circuits
- Programmable High/Low-Voltage Detect (HLVD)
- Flexible Watchdog Timer (WDT) with its Own RC Oscillator for Reliable Operation
- Standard and Ultra Low-Power Watchdog Timers (ULPW) for Reliable Operation in Standard and Deep Sleep modes

Pin Diagrams (Continued)

44-Pin TQFP,
44-Pin QFN(1,2,3)



Note 1: Gray shading indicates 5.5V tolerant input pins.
2: The back pad on QFN devices should be connected to VSS.
3: See Table 1 for complete pinout descriptions.

TABLE 1: PIC24FJXXXGB204 PIN FUNCTION DESCRIPTIONS

Pin	Function	Pin	Function
1	C1INC/C2INC/C3INC/**RP9**/SDA1/T1CK/CTED4/PMD3/CN21/RB9	23	AN4/C1INB/**RP2**/SDA2/T5CK/T4CK/CTED13/CN6/PMD2/RB2
2	**RP22**/PMA1/PMALH/CN18/RC6	24	AN5/C1INA/**RP3**/SCL2/CTED8/CN7/PMWR/RB3
3	**RP23**/PMA0/PMALL/CN17/RC7	25	AN10/**RP16**/PMBE1/CN8/RC0
4	**RP24**/PMA5/CN20/RC8	26	AN11/**RP17**/CN9/RC1
5	**RP25**/CTED7/PMA6/CN19/RC9	27	AN12/**RP18**/PMACK1/CN10/RC2
6	VBAT	28	VDD
7	VCAP	29	VSS
8	**RP10**/CTED11/CN16/PGD2/D+/RB10	30	OSCI/C1IND/CLKI/PMCS1/CN30/RA2
9	REFI/**RP11**/CTED9/CN15/PGC2/D-/RB11	31	OSCO/C2IND/CLKO/CN29/RA3
10	VUSB3V3	32	TDO/PMA8/CN34/RA8
11	AN7/C1INC/REFO/**RP13**/CTPLS/PMRD/CN13/RB13	33	SOSCI/CN1/**RPI4**/RB4
12	TMS/PMA2/PMALU/CN36/RA10	34	SOSCO/SCLKI/CN0/RA4
13	TCK/PMA7/CN33/RA7	35	TDI/PMA9/CN35/RA9
14	CVREF/AN6/C3INB/**RP14**/RTCC/CTED5/CN12/RB14	36	**RP19**/PMBE0/CN26/RC3
15	AN9/C3INA/**RP15**/T3CK/T2CK/CTED6/PMA14/CS1/CN11/PMCS/PMCS1/RB15	37	**RP20**/PMA4/CN25/RC4
16	AVSS/VSS	38	**RP21**/PMA3/CN26/RC5
17	AVDD	39	VSS
18	MCLR	40	VDD
19	CVREF+/VREF+/AN0/C3INC/**RP5**/ASDA1(1)/CTED1/CN2/PMD7/PGD3/RA0	41	CN27/USBID/RB5
20	CVREF-/VREF-/AN1/C3IND/**RP6**/ASCL1(1)/CTED2/CN3/PGC3/RA1	42	PMD6/CN24/VBUS/RB6
21	AN2/CTCMP/C2INB/**RP0**/CN4/PGD1/HLVDIN/PMD0/RB0	43	**RP7**/CTED3/INT0/CN23/PMD5/RB7
22	AN3/C2INA/**RP1**/CTED12/CN5/PMD1/PGC1/RB1	44	**RP8**/SCL1/CTED10/PMD4/CN22/USBOEN/RB8

Legend: **RPn** represents remappable peripheral pins.
Note 1: Alternative multiplexing for SDA1 and SCL1 when the I2C1SEL Configuration bit is set.

1.0 DEVICE OVERVIEW

This document contains device-specific information for the following devices:

- PIC24FJ64GB202
- PIC24FJ128GB202
- PIC24FJ64GB204
- PIC24FJ128GB204

The PIC24FJ128GB204 family expands the capabilities of the PIC24F family by adding a complete selection of Cryptographic Engines, ISO 7816 support and I^2S support to its existing features. This combination, along with its ultra low-power features, Direct Memory Access (DMA) for peripherals and USB On-The-Go, make this family the new standard for mixed-signal PIC® microcontrollers in one economical and power-saving package.

1.1 Core Features

1.1.1 16-BIT ARCHITECTURE

Central to all PIC24F devices is the 16-bit modified Harvard architecture, first introduced with Microchip's dsPIC® Digital Signal Controllers (DSCs). The PIC24F CPU core offers a wide range of enhancements, such as:

- 16-bit data and 24-bit address paths with the ability to move information between data and memory spaces
- Linear addressing of up to 12 Mbytes (program space) and 32 Kbytes (data)
- A 16-element Working register array with built-in software stack support
- A 17 x 17 hardware multiplier with support for integer math
- Hardware support for 32 by 16-bit division
- An instruction set that supports multiple addressing modes and is optimized for high-level languages, such as 'C'
- Operational performance up to 16 MIPS

1.1.2 XLP POWER-SAVING TECHNOLOGY

The PIC24FJ128GB204 family of devices introduces a greatly expanded range of power-saving operating modes for the ultimate in power conservation. The new modes include:

- Retention Sleep with essential circuits being powered from a separate low-voltage regulator
- Deep Sleep without RTCC for the lowest possible power consumption under software control
- VBAT mode (with or without RTCC) to continue limited operation from a backup battery when VDD is removed

Many of these new low-power modes also support the continuous operation of the low-power, on-chip Real-Time Clock/Calendar (RTCC), making it possible for an application to keep time while the device is otherwise asleep.

Aside from these new features, PIC24FJ128GB204 family devices also include all of the legacy power-saving features of previous PIC24F microcontrollers, such as:

- On-the-Fly Clock Switching, allowing the selection of a lower power clock during run time
- Doze Mode Operation, for maintaining peripheral clock speed while slowing the CPU clock
- Instruction-Based Power-Saving Modes, for quick invocation of Idle and the many Sleep modes

1.1.3 OSCILLATOR OPTIONS AND FEATURES

All of the devices in the PIC24FJ128GB204 family offer five different oscillator options, allowing users a range of choices in developing application hardware. These include:

- Two Crystal modes
- Two External Clock (EC) modes
- A Phase-Locked Loop (PLL) frequency multiplier, which allows clock speeds of up to 32 MHz
- A Fast Internal Oscillator (FRC) – Nominal 8 MHz output with multiple frequency divider options and automatic frequency self-calibration during run time
- A separate Low-Power Internal RC Oscillator (LPRC) – 31 kHz nominal for low-power, timing-insensitive applications.

The internal oscillator block also provides a stable reference source for the Fail-Safe Clock Monitor (FSCM). This option constantly monitors the main clock source against a reference signal provided by the internal oscillator and enables the controller to switch to the internal oscillator, allowing for continued low-speed operation or a safe application shutdown.

1.1.4 EASY MIGRATION

Regardless of the memory size, all devices share the same rich set of peripherals, allowing for a smooth migration path as applications grow and evolve. This extends the ability of applications to grow from the relatively simple, to the powerful and complex, yet still selecting a Microchip device.

1.2 DMA Controller

PIC24FJ128GB204 family devices also add a Direct Memory Access (DMA) Controller to the existing PIC24F architecture. The DMA acts in concert with the CPU, allowing data to move between data memory and peripherals without the intervention of the CPU, increasing data throughput and decreasing execution time overhead. Six independently programmable channels make it possible to service multiple peripherals at virtually the same time, with each channel peripheral performing a different operation. Many types of data transfer operations are supported.

1.3 USB On-The-Go (OTG)

USB On-The-Go provides on-chip functionality as a target device compatible with the USB 2.0 standard, as well as limited stand-alone functionality as a USB embedded host. By implementing USB Host Negotiation Protocol (HNP), the module can also dynamically switch between device and host operation, allowing for a much wider range of versatile USB-enabled applications on a microcontroller platform.

PIC24FJ128GB204 family devices also incorporate an integrated USB transceiver and precision oscillator, minimizing the required complexity of implementing a complete USB device, embedded host, dual role or On-The-Go application.

1.4 Cryptographic Engine

The Cryptographic Engine provides a new set of data security options. Using its own free-standing state machines, the engine can independently perform NIST standard encryption and decryption of data, independently of the CPU.

Support for True Random Number Generation (TRNG) and Pseudorandom Number Generation (PRNG); NIST SP800-90 compliant.

1.5 Other Special Features

- **Peripheral Pin Select (PPS):** The Peripheral Pin Select feature allows most digital peripherals to be mapped over a fixed set of digital I/O pins. Users may independently map the input and/or output of any one of the many digital peripherals to any one of the I/O pins.
- **Communications:** The PIC24FJ128GB204 family incorporates a range of serial communication peripherals to handle a range of application requirements. There are two independent I^2C™ modules that support both Master and Slave modes of operation. Devices also have, through the PPS feature, four independent UARTs with built-in IrDA® encoders/decoders, ISO 7816 Smart Card support (UART1 and UART2 only) and three SPI modules with I^2S and variable data width support.
- **Analog Features:** All members of the PIC24FJ128GB204 family include a 12-bit A/D Converter module and a triple comparator module. The A/D module incorporates a range of new features that allows the converter to assess and make decisions on incoming data, reducing CPU overhead for routine A/D conversions. The comparator module includes three analog comparators that are configurable for a wide range of operations.
- **CTMU Interface:** In addition to their other analog features, members of the PIC24FJ128GB204 family include the CTMU interface module. This provides a convenient method for precision time measurement and pulse generation, and can serve as an interface for capacitive sensors.
- **Enhanced Parallel Master/Parallel Slave Port:** This module allows rapid and transparent access to the microcontroller data bus, and enables the CPU to directly address external data memory. The parallel port can function in Master or Slave mode, accommodating data widths of 4, 8 or 16 bits, and address widths of up to 23 bits in Master modes.
- **Real-Time Clock and Calendar (RTCC):** This module implements a full-featured clock and calendar with alarm functions in hardware, freeing up timer resources and program memory space for use by the core application.
- **Data Signal Modulator (DSM):** The Data Signal Modulator (DSM) allows the user to mix a digital data stream (the "modulator signal") with a carrier signal to produce a modulated output.

1.6 Details on Individual Family Members

Devices in the PIC24FJ128GB204 family are available in 28-pin and 44-pin packages. The general block diagram for all devices is shown in Figure 1-1.

The devices are differentiated from each other in six ways:

1. Flash program memory (64 Kbytes for PIC24FJ64GB2XX devices and 128 Kbytes for PIC24FJ128GB2XX devices).
2. Available I/O pins and ports (21 pins on two ports for 28-pin devices, 35 pins on three ports for 44-pin devices).
3. Available Input Change Notification (ICN) inputs (20 on 28-pin devices and 34 on 44-pin devices).
4. Available remappable pins (14 pins on 28-pin devices and 24 pins on 44-pin devices).
5. Analog input channels for the A/D Converter (12 channels for 44-pin devices and 9 channels for 28-pin devices).

All other features for devices in this family are identical. These are summarized in Table 1-1 and Table 1-2.

A list of the pin features available on the PIC24FJ128GB204 family devices, sorted by function, is shown in Table 1-3. Note that this table shows the pin location of individual peripheral features and not how they are multiplexed on the same pin. This information is provided in the pinout diagrams in the beginning of the data sheet. Multiplexed features are sorted by the priority given to a feature, with the highest priority peripheral being listed first.

MICROCHIP

RN4020

Bluetooth® Low Energy Module

Features

- Fully certified Bluetooth® version 4.1 module
- On-board Bluetooth Low Energy 4.1 stack
- ASCII command interface API over UART
- Device Firmware Upgrade (DFU) over UART or Over the Air (OTA)
- Microchip Low-energy Data Profile (MLDP) for serial data applications
- Remote commands over-the-air
- 64 KB internal flash
- Compact form factor: 11.5 x 19.5 x 2.5 mm
- Castellated SMT pads for easy and reliable PCB mounting
- Environmentally friendly, RoHS compliant
- Certifications: FCC, IC, CE, QDID, VCCI, KCC, and NCC

Operational

- Single operating voltage: 1.8V to 3.6V (3.3V typical)
- Temperature range: -30°C to 85°C
- Low-power consumption
- Simple, UART interface
- Integrated Crystal, I^2C Interface, Internal Voltage Regulator, Matching Circuitry, and PCB Antenna
- Multiple IOs for control and status
- GPIO, ADC
- Three Pulse Width Modulation (PWM) outputs

RF/Analog Features

- ISM Band 2.402 to 2.480 GHz operation
- Channels 0-39
- RX Sensitivity: -92.5 dBm at 0.1% BER
- TX Power: -19.0 dBm to +7.5 dBm
- RSSI Monitor

MAC/Baseband/Higher Layer Features

- Secure AES128 encryption
- GAP, GATT, SM, L2CAP, and integrated public profiles
- Create custom services using command API
- Keyboard I/O Authentication
- Software configurable role as peripheral or central and client or server
- Built-in scripting capabilities for hostless operation

Applications

- Health/Medical Devices
  - Glucose meters
  - Heart rate
  - Scale
- Sports Activity and Fitness
  - Pedometer
  - Cycling computer
  - Heart rate
- Retail
  - Point of Sale (POS)
  - Asset tagging and tracking
  - Proximity advertising
- Beacon Applications
- Internet of Things (IoT) Sensor tag
- Remote Control
  - Embedded Device Control
  - AV consoles and game controllers
- Wearable Smart Devices and Accessories
- Industrial Control
  - Private (custom) services
  - Low bandwidth cable replacement
- Smart Energy/Smart Home

Description

Microchip's RN4020 Bluetooth Low Energy Module provides a highly integrated solution for delivering low-power Bluetooth 4.1 solutions. The advanced command interface offers rapid time to market. The RN4020 module complies with Bluetooth specification version 4.1. The module integrates RF, a baseband controller, and a command API processor, making it a complete Bluetooth Low Energy solution. The RN4020 can be used with ultra-low cost microcontroller for intelligent Bluetooth Low Energy applications. For simple sensor applications, the RN4020 internal scripting capabilities enable basic functions to be implemented without the need for external host MCU or software development tools.

1.0 DEVICE OVERVIEW

The RN4020 Bluetooth Low Energy RF module integrates Bluetooth 4.1 radio baseband, MCU, digital analog I/O, on-board stack, and ASCII command API. Figure 1-1 shows the top view of the module. The pinout of the module is shown in Figure 1-2, and the description is presented in Table 1-1. Figure 1-3 lists all the key components of the module.

FIGURE 1-1: RN4020 TOP VIEW

FIGURE 1-2: RN4020 PIN DIAGRAM

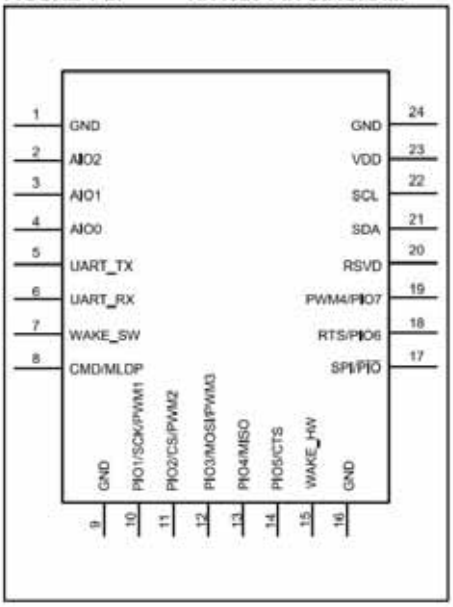


FIGURE 1-3: RN4020 BLOCK DIAGRAM

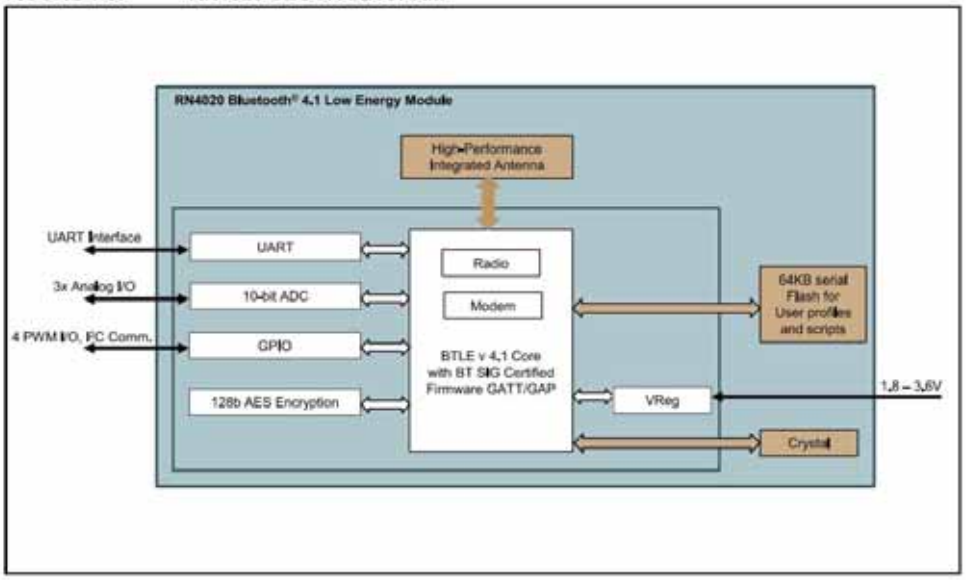


TABLE 1-1: PIN DESCRIPTION

Pin	Name	Description	Function
1	GND	Ground	Ground
2	AIO2	Bi-directional with programmable analog I/O	1.35V and 30 mA max out
3	AIO1	Bi-directional with programmable analog I/O	1.35V and 30 mA max out
4	AIO0	Bi-directional with programmable analog I/O	1.35V and 30 mA max out
5	UART TX	UART Transmit (TX)	Output
6	UART RX	UART Receive (RX)	Input
7	WAKE_SW	Deep Sleep Wake; active-high to wake module from Deep Sleep. If the module runs without a host micro-controller, connect the UART_RX pin to VDD via a 10K resistor to conserve power in Deep Sleep.	Input; weak pull down
8	CMD/MLDP	Command or MLDP mode – In Command mode, UART traffic is sent to the command interpreter. In MLDP mode, UART traffic is routed to the MLDP Bluetooth LED connection, if active.	Input; Edge triggered; Change from High to Low to enter CMD mode from MLDP mode
9	GND	Ground	Ground
10	CONNECTION LED PIO[1] SCK PWM1	Default state is output. Active-high indicates the module is connected to a remote device. Active-low indicates a disconnected state. Configurable as PIO[1] via software command. SCK for Diagnostics and Factory Calibration if pin 17 is asserted.	• Connection Status Indicator (Green LED) • PIO[1] • SCK • PWM1
11	MLDP_EV PIO[2] CS PWM2	Default function is output used for MLDP data event indicator (Red LED). Active-high indicates MLDP data received or UART console data pending. Low level indicates no events. Event is only triggered in MLDP mode, when CMD/MLDP (pin 8) is high. Configurable as PIO[2] via "[I" and "[O" commands. CS for Diagnostics and Factory Calibration if pin 17 is asserted.	• MLDP Data Indicator (Red LED) • PIO[2] • CS • PWM2
12	WS PIO[3] MOSI PWM3	Default function is an output used for Activity Indicator (Blue LED). High level indicates module is awake and active. Low level indicates module is in a Sleep state. Accessible as PIO[3] via "[>" and "[<" commands. MOSI for Diagnostics and Factory Calibration if pin 17 is asserted.	• WS (Blue LED) • PIO[3] • MOSI • PWM3
13	PIO[4] MISO	MISO for Diagnostics and Factory Calibration if pin 17 asserted.	• PIO[4] • MISO
14	CTS PIO[5]	Reserved for CTS if hardware flow control is enabled on the UART; active-low.	• CTS (input) • PIO[5]

TABLE 1-1: PIN DESCRIPTION (CONTINUED)

Pin	Name	Description	Function
15	WAKE_HW	Hardware wake from Dormant state. Setting the WAKE_HW (pin15) high wakes the module from Dormant mode. During the module power up, if WAKE_HW pin is flipped high and low for three cycles (putting the WAKE_HW pin into high, low, and then high again is considered as one flip cycle) in the first five seconds, then the module performs a factory Reset. If the WAKE_SW pin is high when a factory Reset is performed, the factory Reset is a full reset. Otherwise, it is a partial reset that retains the device name, private service and scripts. Set WAKE_HW pin to low in order to lower power consumption in Deep Sleep and Dormant modes. **CAUTION** A full factory Reset erases scripts and sets the device name to the serialized name. For more information, refer to the `SF` Command in the *RN4020 Bluetooth Low Energy User's Guide* (DS70005191).	Active-high; internal pull down
16	GND	Ground	Ground
17	SPI/$\overline{\text{PIO}}$	SPI/$\overline{\text{PIO}}$ for pins 10-13; active-high	Input with internal pull down; selects SPI on pins 10-13
18	RTS PIO[6]	Reserved for RTS if hardware flow control on UART is enabled. If the data transmission to RN4020 must be halted, assert RTS to high. RTS pin operates independently from the CTS (pin 14).	• RTS (output) • PIO[6]
19	PWM4 PIO[7]	Spare PIO	PIO[7]: Spare PIO configurable as input or output
20	RSVD	Do not connect. Factory diagnostics.	No Connect
21	SDA	SDA Data line of the I^2C interface. The RN4020 always acts as the I^2C Master.	SDA
22	SCL	I^2C Clock	SCL
23	VDD	Supply voltage	1.8 to 3.6V
24	GND	Ground	Ground

2.0 GENERAL SPECIFICATIONS

Table 2-1 provides the general specifications of the module. Table 2-2 and Table 2-3 shows the weight and dimensions, and electrical characteristics of the module. Table 2-4 and Table 2-5 specify the current consumption of the module.

TABLE 2-1: GENERAL SPECIFICATIONS

Specification	Description
Standard	Bluetooth 4.1
Frequency Band	2.4 ~ 2.48 GHz
Modulation Method	GFSK
Maximum Data Rate	1 Mbps
Antenna	PCB
Interface	UART, PIO, AIO, SPI
Operation Range	100 meters(1)
Sensitivity	-92.5 dBm at 0.1% BER
RF TX Power	-19.0 dBm to +7.5 dBm
Temperature (operating)	-30°C to +85°C
Temperature (storage)	-40°C to +85°C
Humidity	10% ~ 90% non-condensing

Note 1: Maximum range under ideal conditions such as RF matching, line of sight, maximum power. Actual results may vary depending on the customer's design.

TABLE 2-2: WEIGHT AND DIMENSIONS

Specification	Description
Dimensions	11.5 x 19.5 x 2.5 mm
Weight	1.2g

TABLE 2-3: ELECTRICAL CHARACTERISTICS

Specification	Description
Supply Voltage	1.8 to 3.6V DC
Working current	Depends on profiles, 12 mA typical

TABLE 2-4: CURRENT CONSUMPTION

Mode	Typical Current at 3V
Dormant	<900 nA
Deep Sleep	<5.0 µA
Idle	<1.5 mA
TX/RX active	16 mA

TABLE 2-5: CURRENT CONSUMPTION VS RF TX POWER

RN4020, V_{DD} = 3.3V, 25°C	
TX Power (dBm)	Id (mA)
-19.1	14.0
-15.1	14.4
-10.9	15.0
-6.9	15.9
-2.5	17.6
1.6	20.7
5.8	26.9
7.5	33.6