UNIVERSIDAD DEL BÍO-BÍO

FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

INFORME DE TÍTULO

INGENIERÍA CIVIL EN AUTOMATIZACIÓN

DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA DE ADQUISICIÓN DE DATOS DE CONFORT TÉRMICO, DE BAJO CONSUMO ELÉCTRICO, PARA EL CENTRO DE INVESTIGACIÓN EN TECNOLOGÍAS DE LA CONSTRUCCIÓN CITEC.

AUTOR(ES): Felipe Matías Mora de la Vega.

CONCEPCIÓN – CHILE

2016

UNIVERSIDAD DEL BÍO-BÍO

UNIVERSIDAD DEL BÍO-BÍO

FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

UNIVERSIDAD DEL BÍO-BÍO

INFORME DE TÍTULO

DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA DE ADQUISICIÓN DE DATOS DE CONFORT TÉRMICO, DE BAJO CONSUMO ELÉCTRICO, PARA EL CENTRO DE INVESTIGACIÓN EN TECNOLOGÍAS DE LA CONSTRUCCIÓN CITEC.

PROFESOR GUÍA: Luis Vera Quiroga.

PROFESOR ADJUNTO: Ernesto Rubio Rodríguez

CONCEPCIÓN – CHILE

2016

UNIVERSIDAD DEL BÍO-BÍO

Índice.

Resumen

En este informe de título se describe el procedimiento realizado para la creación de un sistema de adquisición de datos de bajo consumo para realizar estudios de confort térmico en edificaciones. Estos estudios son realizados por el centro de investigación de las tecnologías de la construcción CITEC.

En una primera etapa se establecerá el tipo de placa microcontroladora con la cual se desarrollará el sistema, luego de haber escogido la placa se procederá a hacer un estudio acabado de su funcionamiento, analizando su programación y estructura. La placa seleccionada fue Intel Galileo, ya que es nueva en el mercado, está disponible para trabajar con ella y se necesita determinar sus capacidades.

Durante la realización del sistema se experimentó con tres lenguajes de programación y dos sistemas operativos. Se utilizaron lenguajes de programación como Arduino, C# y Python y con respecto a los sistemas operativos se utilizó Windows y Linux, escogiendo a Linux como el S.O y Python lenguaje de programación respectivamente para el desarrollo del sistema de adquisición de datos.

Una vez escogidos el S.O y el lenguaje de programación se procedió a realizar la comunicación entre la placa y los dispositivos adquisidores.

Posterior a realizar la comunicación entre la placa y la maleta se continuó con la integración de los requerimientos propuestos para el sistema, tales como definir tiempos de inicio y termino de los experimentos, tiempo de muestreo, realizar el estudio en presencia o no de internet y enviar correos electrónicos notificando a los encargados de los experimentos cuando el sistema inicia y termina la adquisición y además cuando se termina el estudio se envía automáticamente un archivo adjunto con los datos adquiridos durante el periodo de muestreo.

Finalmente se debió definir una batería con la cual se energizara la placa teniendo en consideración el tiempo de realización de los estudios y el consumo de la placa en mA, para así dejarlo independiente de la electricidad proporcionada por la red eléctrica.

1 Introducción

1.1 Planteamiento del problema

CITEC es un centro de investigación en tecnologías de la construcción donde su temática es la física de la construcción, la arquitectura y construcción sustentable.

Una de la áreas de trabajo del CITEC es el estudio del confort térmico en las estructuras habitacionales y/o casa habitación, donde se miden variables como CO2, temperatura, hermeticidad de la vivienda, eficiencia energética de la vivienda, etc. Algunos de estos estudios se llevan a cabo en lugares rurales donde no hay un suministro de energía adecuado o simplemente no existe, lo que imposibilita una adquisición de datos con un extenso periodo de muestreo.

Para el registro de estas variables hoy se utiliza una aplicación basada en S.O. Windows. Utilizar un computador para este caso presenta ciertas desventajas, tales como el tamaño del computador, su consumo energético, su robustez y autonomía ante perdidas de energía eléctrica, hecho que acontece con mayor frecuencia en un área rural. Fallas por esta causa impiden la adecuada adquisición de los datos y el posterior estudio.

Para minimizar la perdida de datos, se propone para la solución de este problema configurar un sistema de adquisición de datos basado en una placa microcontroladora, lo cual permitirá adquirir y registrar en un medio extraíble, en la propia placa de bajo consumo. Un suministro independiente de la red eléctrica permitirá implementar esta aplicación en los distintos escenarios en que CITEC deba efectuar sus estudios sobre confort térmico.

1.2 Objetivos del proyecto

1.2.1 Objetivo General

Desarrollo de un sistema de adquisición de datos de confort térmico, en estructuras habitacionales basado en microcontrolador.

1.2.2 Objetivos Específicos

- Estudio del proceso de monitoreo actual
- Determinación de placa microcontroladora adecuada para la nueva implementación.
- Estudio y desarrollo del programa con la placa microcontroladora escogida.
- Desarrollo e implementación de la aplicación definitiva.

1.3 Motivación

La motivación de realizar este proyecto es optimizar el sistema de captura de datos que posee CITEC, ya que el sistema actual posee ciertos problemas tales como la portabilidad del sistema de captura y la dependencia de un flujo constante de alimentación provisto por la red eléctrica, situación que lo limita en ciertas situaciones como por ejemplo ir a un lugar donde no hay un suministro constante de electricidad o se produce algún corte de energía que genera la perdida de datos importantes para realizar el estudio. Otra motivación es lograr que los datos tomados por el equipo no queden solamente en el hardware, como actualmente lo hace, sino que también se pueda almacenar en un medio electrónico y disponer de ellos todo el tiempo, además a través de un correo informar cuando se inicia y termina el proceso.

1.4 Descripción general del entorno de experimentación.

Este proyecto se llevará a cabo durante su etapa inicial totalmente dentro de las dependencias del CIMUBB. Se trabajará con una maleta provista por CITEC la cual contiene ocho sensores de temperatura (Termocuplas), conversor RS232 a RS485, una fuente de alimentación de 24 Volts y dos módulos de entradas análogas ADAM-4018, además se utilizará la placa Intel Galileo la cual será la base para desarrollar la aplicación que permitirá cumplir el objetivo de este trabajo de título.

1.5 Estado del arte

Un sistema de adquisición de datos está encargado de la medición de fenómenos físicos tales como temperatura, humedad, corriente, voltaje, etc. Este DAQ (Sistema de Adquisición de Datos) está compuesto por sensores, un acondicionador de señales o convertidor análogo-digital y un computador. Durante los años estos sistemas se han ido adecuando a las exigencias del mercado como por ejemplo que sea de bajo consumo.

En el trabajo realizado por Bob Steigerwald, Rajshree Chabukswar y Jun de Vega se postula que el ahorro energético se puede efectuar tanto modificando el hardware como el software también se examinan el desarrollo de metodologías y diseño del software para que aporte a la mejora de la eficiencia energética y en la extensión de la vida útil de la batería en sistema móviles. En este estudio también se presentan tópicos como la eficiencia computacional donde se muestran métodos para reducir el costo de energía mejorando la performance de la aplicación [1]. En este tópico se estudia como los algoritmos, el multithreading que se refiere a un programa que contiene dos o más partes que se pueden ejecutar de manera simultánea, los compiladores, las librerías y los sets de instrucciones influyen en la eficiencia computacional. Otro tópico que se muestra es la eficiencia en el manejo de datos donde se establece que si se minimiza el movimiento de datos se reduce el costo de energía, además se establece que se puede lograr la eficiencia de datos diseñando un software que minimice el movimiento de datos, que posea una jerarquía de memoria y un software de aplicación que utilice de manera eficiente la memoria cache. Otros tópicos

de interés que se estudia son como la elección del sistema operativo influye en el ahorro energético y otras herramientas y tecnologías que aportan a este propósito.

Un ejemplo de sistema de adquisición de datos es un data logger. Suman Nath presenta en su trabajo un data logger energéticamente eficiente y que está diseñado para sistemas que poseen memoria y procesamiento limitado. Este dispositivo utiliza un sistema amnésico esto se refiere a que prefiere mantener los datos antiguos en vez de descartarlos reduciendo su fidelidad para dejar espacio a datos nuevos. Según Suman un sistema amnésico está compuesto por dos cosas: una función amnésica y un algoritmo de compresión amnésico. El sistema diseñado por el autor lleva por nombre FLASHLOGGER y lo que hace es comprimir los datos y hacer "envejecer" los datos antiguos de acuerdo a una función amnésica y un algoritmo de compresión específico. Este sistema consiste en dos componentes principales: un módulo de compresión y el AM-STORE que según Suman Nath es el propósito de este trabajo. El sistema se probó en varios casos tomando temperatura, sonido y capturando imágenes. Se utilizó TinyOS. La evaluación del sistema muestra que hay un gran ahorro energético en la captura de datos y recuperación de datos con un tiempo de muestreo establecido [2].

Otro trabajo realizado para probar la utilidad y las característica de un data logger es el realizado por Jason Dunham, Gwynne Chandler, Bruce Rieman y Don Martin donde se midió temperatura en lugares con altas perturbaciones como por ejemplo un rio. En este estudio se realizaron comparaciones entre data loggers para elegir al que más se adecue a las circunstancias y se escogió a HOBBO como la mejor opción [3]. El objetivo de haber leído este trabajo es que entrega una visión de cómo poder escoger un sistema en cuanto a ciertos factores como la precisión, la portabilidad, el consumo energético, la programabilidad, la capacidad de memoria y la importancia de escoger un tiempo de muestreo adecuado al sistema [3].

Un punto importante de un sistema es el consumo de energía y más aún si el sistema es móvil. En el trabajo realizado por Tajana Simunic, Luca Benini y Giovanni De Micheli se hace uso de modelos de consumo de energía para analizar la optimización de código y compiladores con el fin de reducir el consumo de energía, además se simulan modelos de

baterías que entregan el estimado de la duración de una batería. El propósito de realizar este estudio es encontrar la forma de optimizar la disipación de energía en una smartbadge (Pulsera Inteligente) que es un sistema portátil con un microprocesador ARM desarrollado por HP. Durante el estudio se encontró que al optimizar el compilador se ahorra menos del 1% en energía, por el contrario la optimización de software es capaz de ahorra por sobre un 90% de energía, además se analizó la vida útil de una batería implementando un decodificador MPEG a la Smartbadge lo que arrojo que la eficiencia de la batería varía enormemente con las corrientes de descarga en base a un ciclo a ciclo y puede causar una reducción de un 16% en la vida útil de una batería [4].

En el estudio realizado por Ed Baker se presenta el desarrollo de un sistema que se encarga del monitoreo ambiental utilizando computación de bajo consumo y de bajo costo. El sistema es un data logger que se encarga de tomar datos del ambiente y está basado en la plataforma Arduino, además está integrado a un sistema web llamado Drupal que es una herramienta de biodiversidad. El objetivo de este trabajo es mostrar que se puede realizar un sistema de bajo consumo y que sea funcional con la plataforma antes mencionada [5].

Otro sistema desarrollado para que sea de bajo consumo y preciso en el control de temperatura y humedad es el realizado por Hua Xianzhe el cual está basado en un microcontrolador y control Fuzzy [6]. Según el estudio realizado por Xianzhe este sistema puede reducir el consumo de energía de una forma significativa.

Un trabajo de investigación realizado por Stavros Harizopoulos, Mehul, Justin Meza y Parthasarathy Ranganathan habla del consumo energético en los sistemas que manejan datos y dice que no necesariamente hay que considerar el hardware al momento de querer ahorrar energía sino que también el software juega un rol importante, además menciona que existen componentes o sistemas que ofrecen un consumo energético bajo pero a costa del rendimiento, por otra parte las memorias y discos no poseen mayor control de energía más que estados de Sleep. Un ejemplo que se nombra en el aspecto de eficiencia energética son los servers de Google que solo consumen energía cuando el sistema está en uso y no en caso contrario. Para probar la premisa puesta en este trabajo de que no solo el hardware es importante, sino también el software los autores realizaron dos experimentos [7]. El

primero consistió en variar el número de discos del sistema para ver cómo afecta al consumo de energía y el segundo muestra como el diseño de un algoritmo para la eficiencia energética no necesariamente resulta en el mejor desempeño.

Continuando con el desarrollo de sistema T.A. Jesha desarrolló un data logger de bajo costo para registrar temperatura y consumo energético en una casa. El dispositivo es capaz de registrar dos mediciones de temperatura. El sistema está basado en un microcontrolador PIC 18F4550. Todas las mediciones son almacenadas en una tarjeta SD. En este trabajo se hace un análisis de los datos obtenidos durante un año. El análisis de estos datos muestra que los datos reunidos durante el año y los datos reunidos mensualmente por el sistema entregan un valor similar con respecto al consumo energético [8].

En el desarrollo de este estado del arte no se encontró muchos trabajos relacionados al tema específico que se plantea en este trabajo de título o no estaban disponibles para ser leídos debido a que se necesitaba pagar por ellos, sin embargo se muestran trabajos y estudios de ciertos temas que corresponden al tópico de este proyecto como la eficiencia energética y el desarrollo de aplicación en tarjetas de desarrollo o micro controladores.

2 Determinación del sistema a desarrollar

El presente capitulo presenta tres aspectos del proyecto de implementación: 1) el levantamiento del actual sistema de adquisición de datos con el que CITEC desarrolla experimentación en terreno; 2) los requerimientos para el desarrollo del proyecto, en función de que el sistema propuesto tenga las características del anterior; y 3) la selección de la placa electrónica de adquisición para el desarrollo del sistema final.

2.1 Introducción

La necesidad de crear sistemas que sean de fácil uso, transporte y que sean de bajo consumo eléctrico está aumentando hoy en día, ya que se requiere tener el mejor rendimiento pero a bajo costo. El sistema que actualmente posee CITEC para realizar estudios de confort térmico se basa en un notebook y una maleta, los cuales poseen un peso considerable no contando los sensores que se deben transportar que se suma al peso anterior, además el sistema en general posee un consumo eléctrico considerable y aparte es totalmente dependiente de la energía eléctrica suministrada por la red eléctrica lo que en algunas ocasiones impide realizar estudios en lugares rurales donde no se dispone de una conexión eléctrica. Lo que propone este trabajo es que el sistema de registro de datos deje de depender de la red eléctrica, también que el notebook sea reemplazado por una placa microcontroladora, lo cual reducirá el peso del sistema y además entregar información de inicio y termino del proceso y que los datos obtenidos aparte de quedar almacenados en una micro SD, también se envíen al correo electrónico de los encargados del experimento, así podrán tener un mayor control del sistema.

2.2 Metodología actual de toma de datos.

En esta sección se describirá el proceso actual que utiliza CITEC para la adquisición de datos de temperatura en lugares habitados y no habitados para la realización del posterior estudio.

El sistema actual utilizado por CITEC entrega la posibilidad de medir dos tipos de variables: temperatura y transferencia de calor. Para este trabajo de título solo se medirá temperatura. Las variables de temperatura y transferencia de calor se miden a través de termocuplas y sensores de transferencia de calor respectivamente.

En la figura 1 se puede apreciar el equipo actual utilizado por CITEC.

Figura 1: Equipo utilizado por CITEC para toma de datos.

2.2.1 Maleta

La adquisición de datos se realiza a través de una maleta que está compuesta por sensores de temperatura en este caso termocuplas tipo J, también está compuesta por sensores de transferencia de calor y CO2, módulos de entradas análogas ADAM 4018 y un conversor RS232 a RS485.

La maleta está destinada para efectuar estudios a grandes escalas, pero solo se utiliza una parte de ella. En este caso solo los sensores de temperatura y de transferencia de calor son utilizados. En este trabajo de título solo de utilizaron los sensores de temperatura que están conectados a las ocho entradas de los módulos análogos de entrada ADAM-4018 [9], estos dispositivos están encargados de hacer la captura de los datos para luego enviarlos hacia el computador a través del conversor RS485 a RS232. En la figura 2 podemos apreciar la maleta con sus respectivos instrumentos de medición.

Figura 2: Maleta con sus respectivos instrumentos

A continuación se describirán cada uno de los componentes que componen esta maleta.

2.2.1.1 Conversor TRP-C06 RS232 a RS422/485

El TRP-C06 permite convertir desde el estándar RS232 al RS422 o RS485 y transmitir datos hasta los 1.2KM. Posee formato de datos automático y una función que detecta el baudrate automáticamente sin la necesidad de configurar nada. TRP-C06 está equipado con una aislación de 3000V DC y un protector interno en las líneas de datos que protege a cualquier computador que esté conectado a este dispositivo y al conversor contra los altos voltajes.

En la figura 3 se muestra el TRP-C06 y en la tabla 1 sus especificaciones técnicas.

Figura 3: Conversor TRP-C06

Voltaje de entrada	+10V a +30V
Conexión host	Standard D-Sub 9 conectores hembra
Interface RS-232	Standard D-Sub 9 conector hembra
Señal RS232	TXD, RXD y GND
Interface RS-422/485	Bornera industrial
Señal RS285	2 cables half-duplex diferencial (DATA+, DATA-)
Distancia de transmisión	RS422/485 hasta 1200M
Velocidad de comunicación	De 300bps a 115.2Kbps
Consumo de potencia	1.2 watt
Temperatura de operación	-10 a 50°C
Temperatura de almacenamiento	-20 a 70°C
Humedad	10-90% no condensado
Dimensión	151mm X 75mm X 26mm
Peso	375g

Tabla 1: Especificaciones técnicas Conversor TRP-C06

2.2.1.2 Módulo de entradas análogas ADAM-4018

El ADAM-4018 es un modelo de entradas análogas de 16 bit y 8 canales que provee rangos de entradas programables en todos sus canales. ADAM-4018 posee condicionamiento de señal, un conversor A/D, funciones de rango y comunicación digital RS485. Este dispositivo utiliza un microprocesador controlado de 16 bit que se encarga de convertir el voltaje o corriente del sensor en datos digitales. Este módulo envía los datos al host a través del standard RS485. En la figura 4 se muestra el modulo ADAM-4018 y en la tabla 2 sus especificaciones técnicas.

Figura 4: Modulo de entradas análogas ADAM-4018

Alimentación	10~30VDC
Canales	8 diferenciales
Corriente de entrada	+/- 20mA
Entrada de sensores	Termocuplas tipo J, K, T, E, R, S, B
Temperatura de operación	-10~70°C
Humedad	5~95% Humedad relativa
Temperatura de almacenamiento	-25~85°C
Interface	RS-485
Protocolo de comunicación	Comando ASCII
Velocidad de comunicación	Serial: de 1,200 a 115.2 kbps
Distancia de comunicación	Serial: 1.2 KM

Tabla 2: Especificación Técnica ADAM-4018

Además de los componentes antes mencionados la maleta también posee un batería de 24V que va conectada a la red eléctrica y que alimenta los dispositivos y también contiene termocuplas tipo J.

2.2.2 Software

El software actualmente utilizado esta realizado en lenguaje de programación QT el cual lleva por nombre monitoreo UBB V2.11 está compuesto por distintos módulos los cuales permiten configurar los módulos de entradas analógicas ADAM-4018, además permite configurar el monitoreo estableciendo una fecha de inicio y termino y tiempo de muestreo. Este software también proporciona la opción de visualizar los datos al momento de la captura y almacenarlos dentro del computador con una extensión CSV así como también permite visualizar los archivos guardados anteriormente y generar una gráfica con ellos. En las figuras 5 y 6 podemos apreciar el software utilizado.

Figura 5: Ventana principal Software.

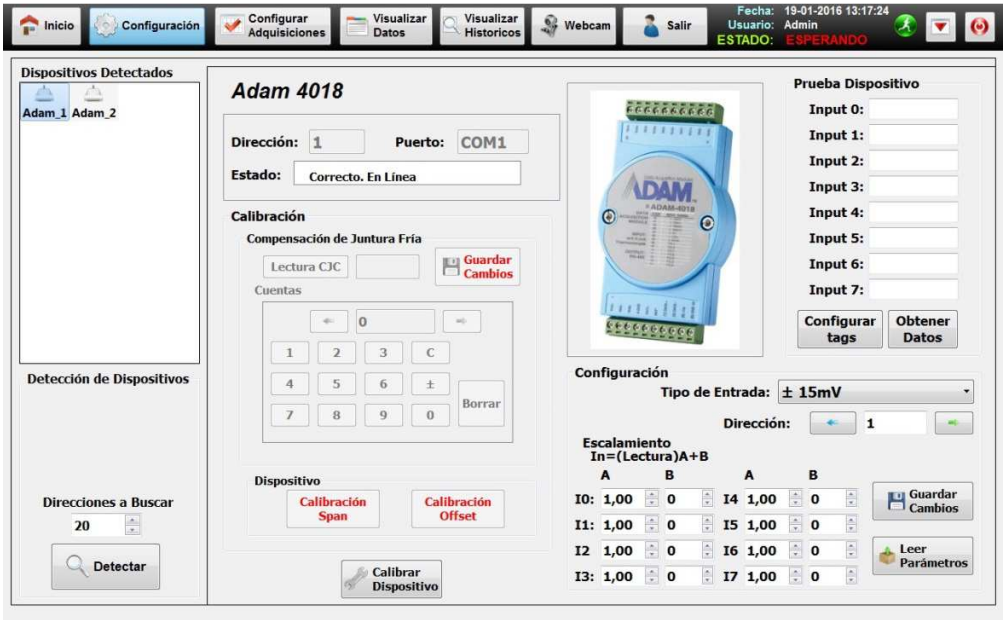

Figura 6: Configuración ADAM-4018

2.3 Actividades previas al desarrollo del nuevo sistema.

Para cumplir con los objetivos propuestos en este trabajo de título se realizaran ciertas actividades que intentaran replicar algunas funcionalidades del sistema actual, tales como: la adquisición de temperatura, el establecimiento de fechas de duración del experimento, la fijación de tiempos de muestreo y el almacenamiento de los datos en algún medio disponible. Se llevaran a cabo actividades como el estudio de la placa a utilizar para desarrollar el sistema, desarrollo de código en lenguaje de programación, realización de pruebas del sistema desarrollado para corregir errores, analizar la factibilidad técnica y económica del proyecto y finalmente se procederá a la implementación del sistema.

2.4 Determinación de requerimientos técnicos y económicos.

Para el desarrollo de este proyecto hay que tener en cuenta ciertos requerimientos técnicos y económicos para que este sea exitoso, los cuales se expondrán a continuación.

2.4.1 Requerimientos técnicos.

Los requerimientos técnicos se obtuvieron de las entrevistas con el personal de CITEC y con el profesor guía.

2.4.2 Requerimientos técnicos de software

Los requerimientos de software fueron planteados desde un principio por las personas interesadas en el desarrollo del proyecto, estos fueron que el sistema debía reemplazar al existente esto quiere decir que el sistema debe tomar datos, guardarlos dentro del sistema y se debe poder extraer los datos para realizar los estudios pertinentes.

El sistema debe tomar los datos y almacenarlos en una tarjeta microSD para su posterior extracción.

No se pide interfaz gráfica, ya que el sistema no necesitará un computador para adquirir y almacenar los datos.

2.4.3 Requerimientos técnicos de hardware

Los requerimientos de hardware son que el sistema se debe realizar en una placa microcontroladora Intel Galileo, debe prescindir de un computador para su configuración, también el sistema debe poseer una batería externa que alimente la placa microcontroladora y reduzca el consumo de energía del sistema y debe ser de fácil transporte, ya que en algunas ocasiones los estudios se realizan en lugares alejados.

2.4.4 Requerimientos económicos.

En este aspecto no existe algún requerimiento especifico, pero si el sistema debe ser de menor costo posible.

2.5 Estudio técnico y económico de la propuesta.

En la figura 16 se muestra el diagrama de conexión de la propuesta el cual consiste en la placa microcontroladora Intel Galileo a la cual se conectará una maleta por medio de un conversor RS232 a TTL, la cual contiene un conversor RS485 a RS232, 2 módulos análogos de entrada ADAM 4018 y 8 termocuplas. De esta maleta se utilizara solo un módulo ADAM 4018 al cual se le conectaran las 8 termocuplas en las 8 entradas análogas que el modulo dispone. El módulo ADAM 4018 estará conectado a un conversor RS485 a RS232. Este conversor se comunicara a la placa Intel galileo a través de un puerto serie. Cabe destacar que las ocho entradas del ADAM 4018 enviaran los datos adquiridos por los

sensores a través de un puerto serial. La placa estará conectada a una batería externa para su suministro de energía.

El sistema de adquisición de datos estará configurado de tal forma que tomara los datos de una cierta fecha a una cierta hora establecida por el personal que realizará el estudio, luego estos datos quedaran almacenados en la micro SD o también pueden ser enviados automáticamente por correo a la persona que esté a cargo de realizar el proyecto.

Esta propuesta está diseñada para que sea fácil de transportar, que sea reducida en consumo de energía en comparación con el sistema actualmente utilizado y que proporcione una mayor comodidad al momento de realizar los estudios.

La placa a utilizar y la batería se presenta en la figura 17 y 18 respectivamente, y la propuesta económica se presenta en la tabla 1.

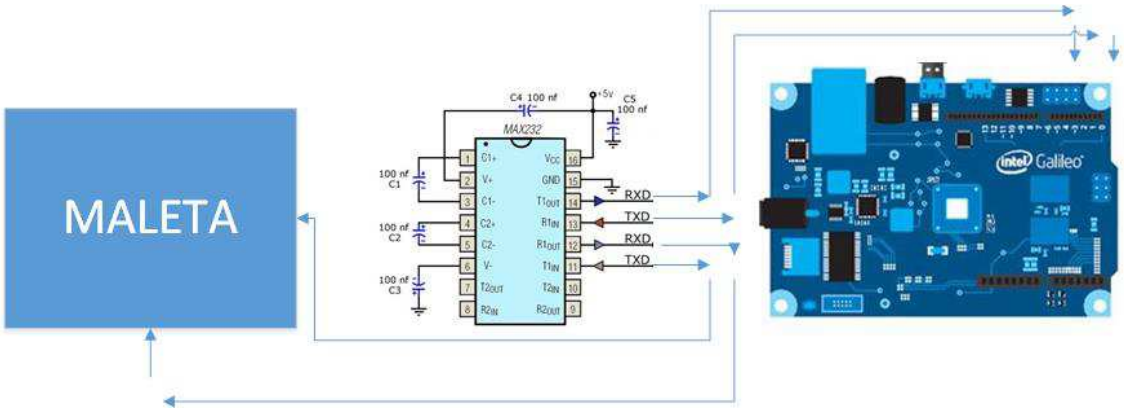

Figura 16: Esquema conexión de la propuesta

Figura 17: Tarjeta de desarrollo Intel galileo

Figura 18: Batería li-ion de 6000mAh

Elemento	Características	Cantidad	costo por unidad	Costo total
Tarjeta de desarrollo Intel Galileo	-Voltaje de Input 5v -14 Pins I/O Digitales -6 Pins analógicos -Corriente Total DC output en todas las líneas de 80mA -Corriente DC para 3.3V de 800mA -Corriente DC para 5v de 800mA	1	70.370	70.370
Conversor RS232	Max232	1	1.690	1.690
**Condensadores		5	500	500
DB9		2	924	924
*Cables		-	1000	1000
Batería	-Batería externa Genius 6000mAh -Tipo de conexión USB	1	18.990	18.990
MicroSD	Kingston 16GB	1	5.990	5.990
costo total				99.464

Tabla 3: Propuesta económica

*Los cables corresponden a cables de conexión que se utilizaron para armar el circuito del conversor RS232 a TTL.

**Los condensares al igual que los cables se utilizaron para armar el circuito del conversor RS232 a TTL.

2.6 Selección de placa para el desarrollo del sistema.

En esta sección se dará a conocer una parte importante del desarrollo de este proyecto que es la placa microcontroladora. Esta es Intel Galileo [10][11][12][13][14]. Desde un principio estaba planteado el desarrollo de este proyecto sobre una placa microcontroladora Intel Galileo, ya que era algo nuevo con grandes capacidades según los creadores y se propuso experimentar con ella.

Al ser algo relativamente nuevo en el mercado no se encontraba mucha información sobre las capacidades de Intel Galileo, por lo tanto hay que ser enfático en que en este trabajo de título se utilizó la placa Galileo con fines investigativos.

En la figura 7 y 8 se muestra la estructura de la placa y sus componentes.

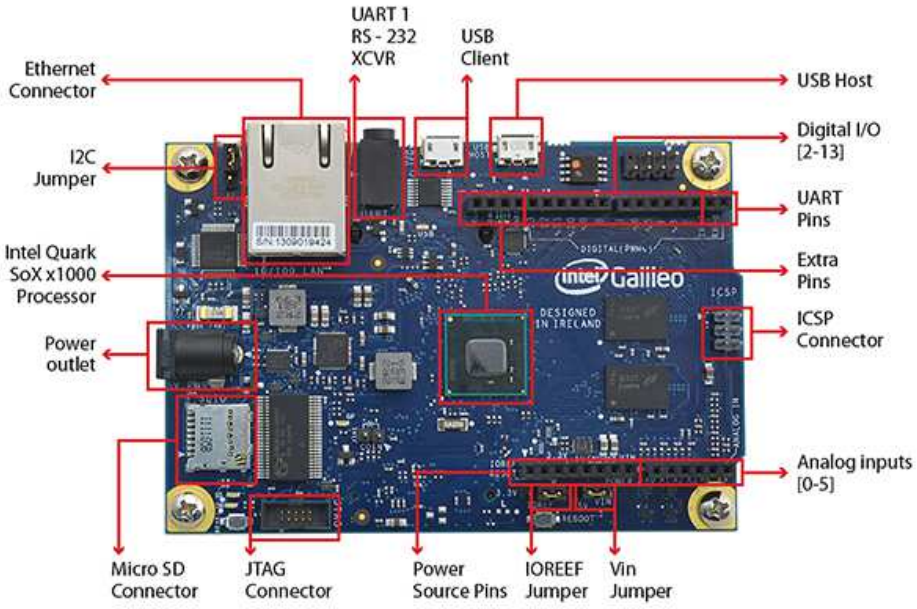

Figura 7: Lado superior placa Intel Galileo.

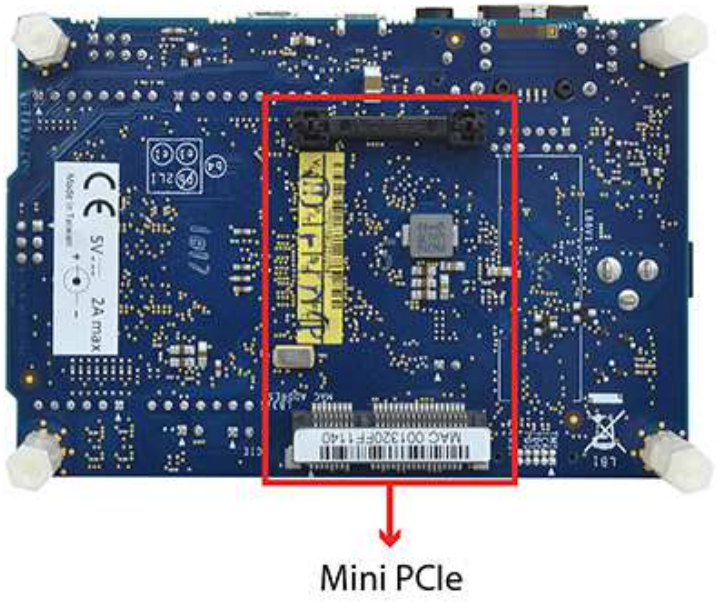

Figura 8: Lado inferior Intel Galileo.

2.7 Descripción y características Intel Galileo.

Intel galileo es la primera placa microcontroladora desarrollada por Intel que es compatible en hardware y en software con Arduino, pero el procesador es de propiedad de INTEL. Intel galileo posee características superiores a Arduino, tales como un procesador más rápido, puertos adicionales de I/O, un puerto Ethernet, un slot para micro SD, un puerto RS-232, un puerto USB host y un puerto USB cliente, además posee una mayor cantidad de memoria.

Lo que hace a Intel galileo interesante es su capacidad de procesamiento posee un procesador Intel Quark SoC x1000 mononucleo y una velocidad de 400Mhz lo que hace que haya un mejor desempeño a la hora de ejecutar código.

El software por defecto que utiliza Intel galileo para su programación es el software proporcionado por Arduino, pero también se puede cargar un sistema operativo en esta al

igual que otras placas de desarrollo disponibles en el mercado como Raspberry PI. Los sistemas operativos que se pueden cargar dentro de la placa son Windows y Linux.

2.8 Software a utilizar para el desarrollo de la aplicación.

Como se mencionó anteriormente Intel Galileo tiene dos formas de programación, una de estas es trabajar con el software proporcionado por Arduino y hacer que la placa actúe solo como un controlador tomando datos y registrándolos o hacer que la placa trabaje como un computador cargándole un sistema operativo como Windows o Linux y tenga mayores funcionalidades. En este proyecto se intentaron ambas formas obteniendo resultados muy variados y no esperados. Como primera instancia se trabajó con el software proporcionado por Arduino que según todos los manuales era totalmente compatible en programación y en recursos de programación como lo son las bibliotecas. Lamentablemente las bibliotecas no eran totalmente compatibles, ya que estas estaban hechas para Arduino, pero no compatibilizadas para Intel galileo por lo que se tuvo que desechar la opción de trabajar con Arduino, puesto que la programación de la aplicación tiene una complejidad que requiere ayuda de algunas bibliotecas para su desarrollo que no se encontraban disponibles. Como segunda opción se decidió trabajar con Windows, pero hubo muchas complicaciones al momento de querer ejecutar el programa, problema que retraso el avance del proyecto, por lo tanto se siguió con la siguiente opción: trabajar con Linux. Esta opción tampoco careció de dificultades, ya que era necesario encontrar la imagen de Linux correcta para Intel Galileo y que permitiera instalar todos los programas necesarios para la creación de la aplicación. Afortunadamente Linux después de todo funcionó perfecto, por lo tanto se escogió esta opción para la programación de la placa microcontroladora. Otros software que se utilizaron fueron PuTTY para visualizar dentro de Intel Galileo cuando se le instaló el sistema operativo Linux, SDFormatter para formatear la tarjeta SD y Raw Writer que sirvió para montar la imagen de Linux en la tarjeta SD. A continuación se describirá brevemente la configuración de estos softwares en Intel Galileo.

Otro punto a parte que dificultó encontrar el software adecuado para desarrollar la aplicación para este trabajo de título fue el hecho de que en un principio se había planteado

que el sistema estaría basado en el protocolo de comunicación MODBUS y todos los programas que se habían intentado realizar en un comienzo estaban basados en este protocolo, sin embargo mucho tiempo después se descubrió que los módulos de entrada y salida no eran compatibles con este protocolo, por lo tanto se debió plantear todo desde cero y buscar otras alternativas, pero afortunadamente como anteriormente se mencionó esta fue encontrada.

2.8.1 Arduino para Intel Galileo.

Este software se obtiene desde la página oficial de Arduino y su configuración es similar a cuando se trabaja con cualquier otro dispositivo Arduino. En la figura 9 se muestra la interfaz de Arduino para Intel Galileo y en el anexo I se encuentra la instalación y configuración completa de este software. Además en el anexo II se encuentra la instalación de drivers necesarios para poder utilizar Intel Galileo con Arduino.

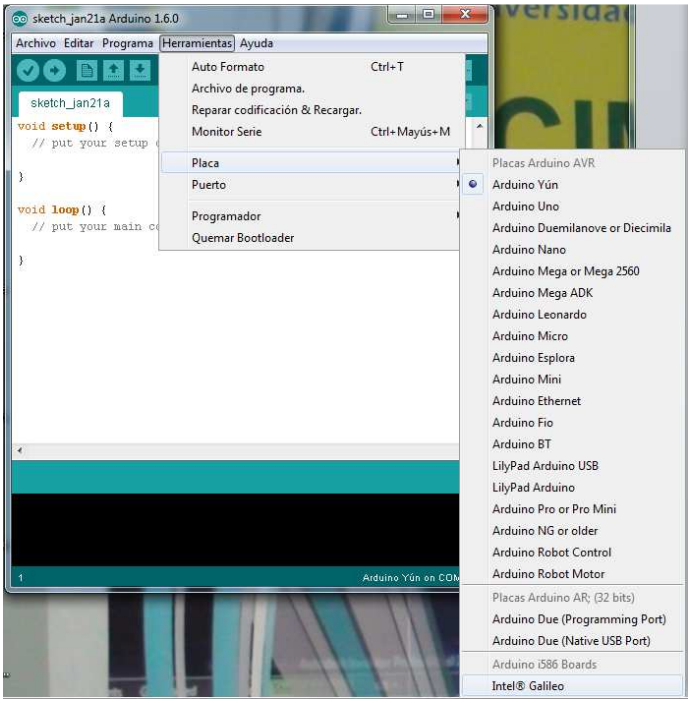

Figura 9: Interfaz Arduino para Intel Galileo

2.8.2 Windows para Intel Galileo

Para poder ejecutar un sistema operativo como Windows en Intel Galileo se necesita una tarjeta SD, tener una imagen del sistema operativo Windows que se puede obtener desde la página de Intel y se necesita que programas como Visual Studio 2013 [15] o superior y telnet estén instalados en el computador donde se conectará la placa. Para montar la imagen de Windows en la tarjeta SD se debe realizar por símbolo del sistema. En la figura 10 y 11 se aprecia como es montado Windows en la tarjeta SD y en el anexo III se muestra la instalación y configuración del software.

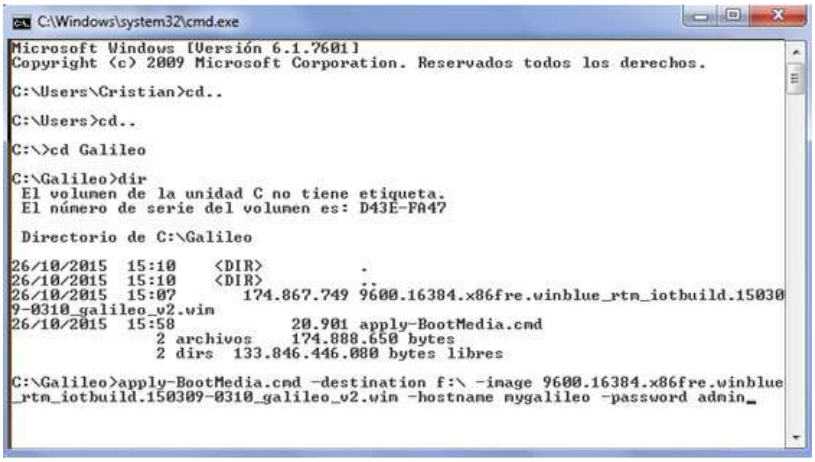

Figura 10: Proceso de grabación imagen de Windows en tarjeta SD.

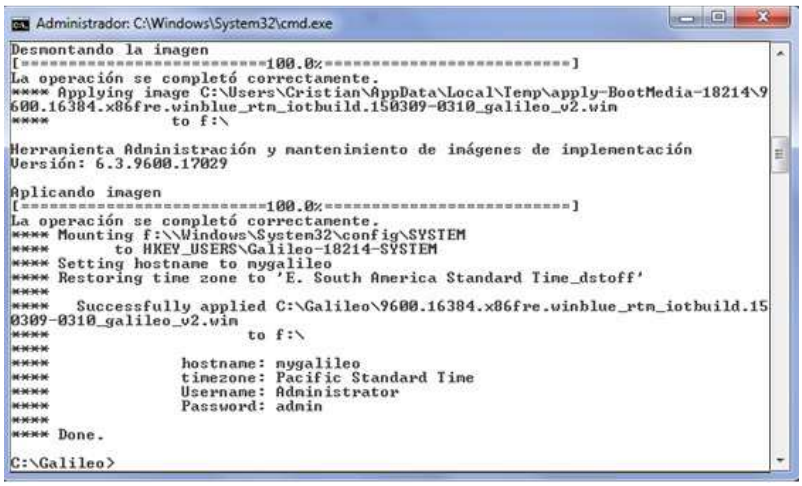

Figura 11: Proceso de grabación imagen de Windows en tarjeta SD.

2.8.3 Linux para Intel Galileo.

Instalar Linux en Intel Galileo [16] lleva casi el mismo proceso que Windows solo que con Linux se monta la imagen en la tarjeta SD con el programa Raw writer y se visualiza lo que se encuentra en la placa con PuTTY y no se necesitan programas como Visual Studio ni Telnet para funcionar. En la figura 12 se puede ver Linux instalado en Intel Galileo y en el anexo IV se muestra la configuración e instalación de este software.

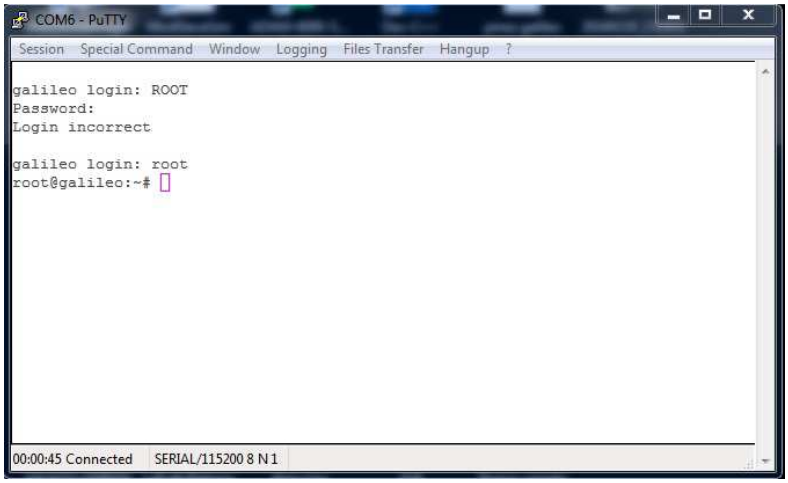

Figura 12: Linux en Intel Galileo

2.8.4 PuTTY

PuTTY es un cliente SSH, Telnet, rlogin, TCP y serial que se utilizó para visualizar todo lo que ocurría en Intel galileo y como interfaz para realizar la programación y visualización de resultados. En la figura 13 se puede ver la interfaz de configuración de PuTTY.

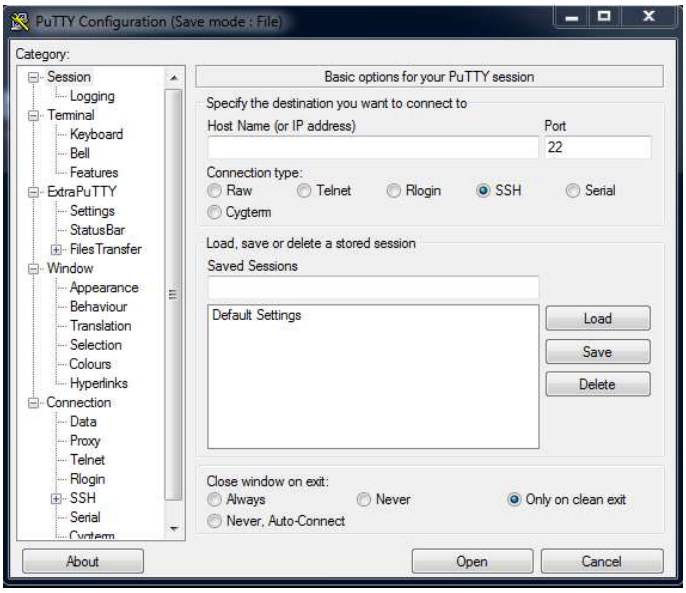

Figura 13: Interfaz de Configuración PuTTY.

2.8.5 SDFormatter y Raw Writer

Estos programas se utilizaron para formatear la tarjeta SD y montar el sistema operativo respectivamente. Son programas que se recomienda utilizar si se va a montar un sistema en Intel Galileo

2.8.6 WinSCP

WinSCP es una aplicación de software libre. WinSCP es un cliente SFTP gráfico para Windows que emplea SSH. Su función principal es facilitar la transferencia segura de archivos entre dos sistemas informáticos, el local y uno remoto que ofrezca servicios SSH. En la figura 14 se muestra la interfaz gráfica del programa. Este programa se utilizó para la transferencia de datos desde la placa Intel Galileo hacia un computador.

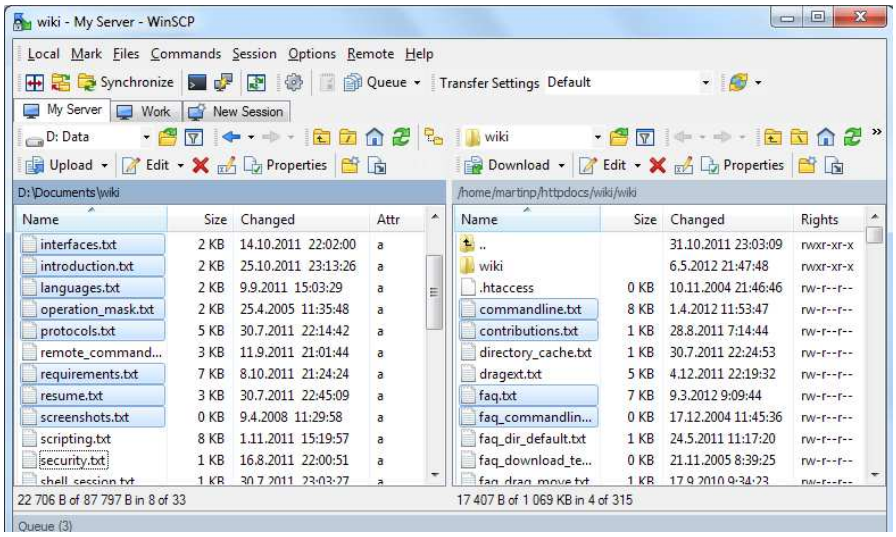

Figura 14: Interfaz gráfica WinSCP

2.8.7 Advanced IP Scanner

Advanced IP Scanner es un programa utilizado para escanear todos los dispositivos de red. En la figura 15 se muestra la interfaz gráfica del programa. Este programa se utilizó para poder conocer la dirección IP de la placa Intel galileo.

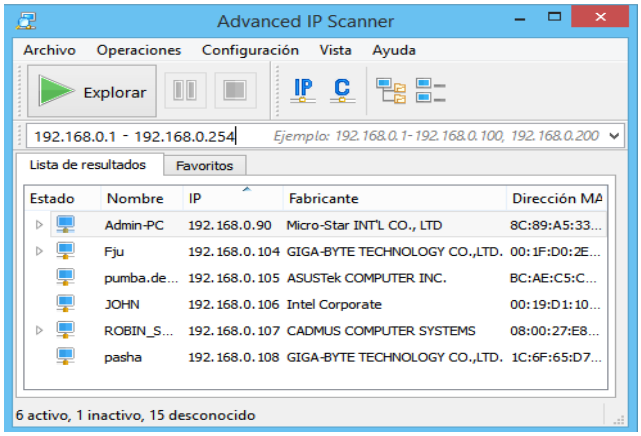

Figura 15: Interfaz Gráfica Advanced IP Scanner

3 Desarrollo de la aplicación en Python y conexión entre placa Intel Galileo y maleta adquisidora.

3.1 ¿Por qué Python?

A la hora de iniciarse en un lenguaje de programación Python tiene características que lo hacen muy atractivo para desarrollar programas básicos, intermedios o avanzados, ya que presenta líneas de código de más fácil comprensión que otros lenguajes similares como C++ o Java, además de tener menor cantidad de líneas de código en desarrollo en programas equivalentes, debido, por ejemplo, a la omisión de llaves o paréntesis que encierran ciertas funciones o ciclos.

La fácil comprensión de códigos en Python se debe a su estructura tabulada, lo que lo hace más intuitivo y ordenado.

Otra razón por la cual se utilizó Python es por su naturaleza multiplataforma, es decir, que se puede usar en más de un sistema operativo, en el caso de este trabajo de título el desarrollo del código fue principalmente en Windows, pero su ejecución final fue en YOCTO, que es una versión de Linux para su uso en la placa Intel Galileo.

Por último el número de módulos que proporciona Python para el desarrollo de proyectos es bastante amplio facilitando la programación del código.

3.2 ¿Qué es Python?

Python es un lenguaje de programación interpretado creado a finales de los ochenta por Guido Van Rossum en el Instituto Nacional de Investigación para las Matemáticas y la Informática (CWI Centrum Wiskude & Informática) en Holanda, como sucesor del lenguaje de programación ABC. Se trata de un lenguaje Multiparadigma, ya que soporta orientación a objetos, programación imperativa y en menor medida programación

funcional, además usa tipeado dinámico, y es multiplataforma, lo que facilita la migración entre sistemas operativos.

Python es administrado por Python Software Foundation. Posee licencia de código abierto, denominada Python Software License, que es compatible con la Licencia publica general de GNU apartir de la versión 2.1.1.

La filosofía de Python es particularmente atractiva a la hora de iniciar trabajos de programación, pues facilita considerablemente el entendimiento de códigos debido a su estructura tabulada. En la Figura 17 se muestra el logo del lenguaje.

Figura 19: Logo de Python.

3.3 Módulos de Python.

Como se ha mencionado anteriormente Python consta de una gran cantidad de módulos o librerías que permiten la conexión con otros sistemas de información o programas. Para el desarrollo del sistema se ocuparon varios módulos proporcionados por Python [17]. El primer módulo que se utilizo fue pySerial. Este módulo entrega la posibilidad de establecer comunicación serial con otros dispositivos, en este caso la placa Intel Galileo y la maleta. Esta librería se debe instalar previamente y se importa escribiendo la instrucción “Import serial” al principio del código.

El segundo módulo ocupado en el desarrollo de la aplicación fue el que te tenía que ver con los tiempos, retardos y sleep dentro del programa. Este módulo lleva por nombre “time” y se importa de la misma forma que el modulo anterior.

El tercer módulo utilizado fue “datetime” este módulo nos permite establecer fechas de término e inicio de un proceso, esta librería utiliza el reloj del sistema para realizar lo antes mencionado. Este módulo utiliza el mismo formato con el cual podemos ver la fecha y hora en cualquier computador, por lo tanto en el dispositivo en el cual se quiera utilizar este módulo es necesario establecer la fecha y hora con un formato estándar.

El cuarto modulo que se utilizo fue “smtplib” este módulo nos entrega la posibilidad de crear y enviar emails, también nos permite adjuntar archivos en un email. Para realizar esto se debe importar desde el módulo smtplib otros módulos tales como “email.MIMEMultipart”, “email.MIMEBase” y “email.MIMEText”, estos módulos nos permiten agregar texto al email y archivos adjuntos como antes se mencionó.

El quinto módulo ocupado fue “urllib2” este módulo sirvió para verificar si estaba disponible la conexión a internet.

La comunicación a Internet se realizara a través del puerto Ethernet de la placa Intel Galileo. Al cual se le conectara un cable Ethernet que estará conectado a un punto de red cualquiera.

Al módulo urllib2 se le entrega una dirección de internet específica y que sabemos que esta accesible como por ejemplo Google y lo que hace es verificar si se puede acceder a esta dirección, si esto es verdadero significa que hay internet disponible y si es falso no hay presencia de internet. Esto nos sirvió para que el sistema no colapsara en caso de que no hubiera internet, ya que este envía emails al momento de inicio y termino del estudio. En el caso de que no hubiera internet los datos recogidos solo se almacenan en la tarjeta SD.

3.4 Desarrollo del programa.

Para el desarrollo del programa como se mencionó anteriormente se utilizaron distintos módulos de Python. En la figura 20 se muestra a grandes rasgos la estructura del programa.

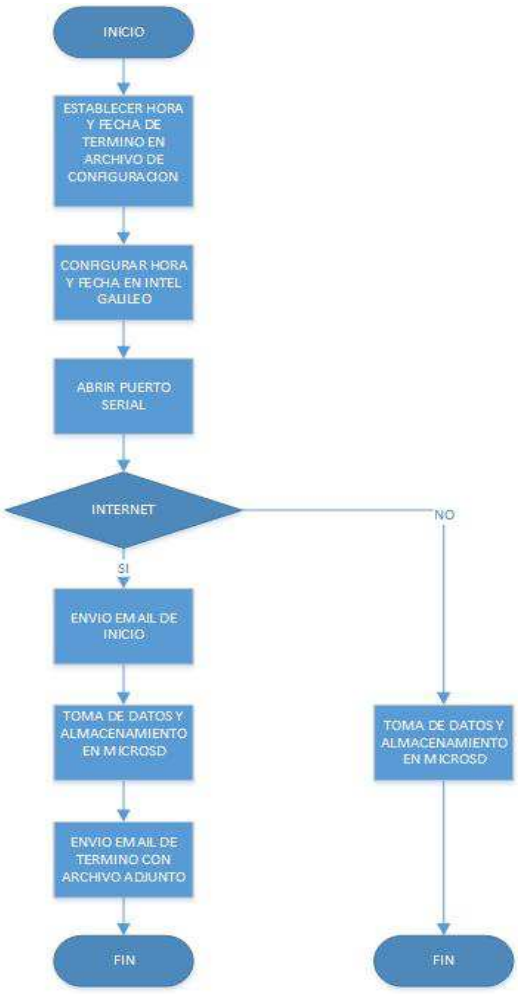

Figura 20: Estructura del programa

3.4.1 Desarrollo de la comunicación serial

Para el desarrollo de la comunicación serial como se mencionó anteriormente se realizó por medio del módulo pySerial de Python. En esta parte se explicara paso a paso el proceso por el cual se programó la comunicación serial.

Como primer paso se procede a declarar los módulos a utilizar en este caso se agrega el ya mencionado pySerial, pero además dentro del programa hay funciones de tiempo, por lo tanto también se debe agregar el módulo de tiempo que es “time”. Cada uno de estos módulos se importan con la palabra “import”, propia de Python. En la figura 21 se muestra cómo deben ir declarados los módulos.

```
import serial
import time
```

Figura 21: Declaración de módulos

Luego de haber realizado lo anterior se procede a definir las variables de un puerto serial como son: el puerto por el cual se recibirá y se enviará información, el baudrate, la paridad, el stopbit y también este módulo nos da la opción de definir un tiempo entre que se abre el puerto serial y comienza la toma de datos. Cabe destacar que todo esto se realiza dentro de una función. Luego de definir todas las variables se debe abrir el puerto para que comience la toma de datos. La función que se utiliza para realizar dicha acción se denomina “open”. Como dato extra se le debe dar un handshake time al hardware para se puede abrir el puerto, esto se realiza escribiendo “readline” antes de iniciar la toma de datos. En la figura 22 se muestra cómo va definido el puerto serial.

```
sp = serial.Serial()
sp.port = 'COM8'
sp.baudrate = 9600
sp.parity = serial.PARITY_NONE
sp.bytesize = serial.EIGHTBITS
sp.stopbits = serial.STOPBITS_ONE
sp.timeout = 0.5
sp.xonxoff = False
sp.rtscts = False
sp.dsrdtr = False

sp.open()

sp.readline()
```

Figura 22: Definición de variables del puerto serial en Python.

3.4.2 Desarrollo de programa para enviar y recibir datos de la maleta.

Para el desarrollo de la función que permitió el envío y recepción de datos, aparte de utilizar los módulos antes usados se agregó el módulo “datetime”, el cual se importa al igual que todos los módulos de Python agregando “import” y el nombre del módulo al principio del programa.

Para que se logre la conexión entre los ADAM-4018 y la placa el programa debe enviar un código específico para que se pueda enviar y recibir datos, en este caso la maleta consta de dos modulos ADAM-4018 y a ambos se le envían códigos diferentes para establecer la comunicación, en el caso de este proyecto se utilizara el modulo que se encarga de adquirir temperatura, al cual se accede enviando el código #02. Esto generaba problemas en Python al querer enviarlo, ya que el carácter # lo reconoce como comentario, por tanto fue necesario transformar el código en ascii, esto se realizó utilizando la función encode que proporciona Python para hacer transformaciones de datos. En el código se definió como sp.write(s.encode(‘ascii’)), donde sp es la definición del puerto serial y write es para enviar datos a través del puerto, s es una variable creada que contiene el código que será enviado de forma serial que como anteriormente se dijo es #02 y este dato se transforma a ascii por medio de la función antes mencionada encode. El siguiente paso es recibir los datos enviados por los modulos adquisidores ADAM. En un principio solo se enviaba el código y

se recibía lo que el ADAM enviaba, pero el primer bit del dato que entregaba el modulo traía basura, por lo tanto se debió intervenir en la trama y eliminar ese bit.

Para recibir un dato de forma serial en Python se utiliza la función “readline”. En el código se defino como “value = sp.readline()” donde value es una variable que almacena lo que entrega el puerto serial en este caso la temperatura adquirida por los 8 canales de los módulos de entrada y salida ADAM-4018. Una vez adquiridos los datos se almacenan en una nueva variable, la cual contendrá los datos, pero en forma de lista lo que permitirá separar cada bit y poder eliminar el primer bit como antes se mencionó. Esto se realizó utilizando el comando “list” de Python de la siguiente forma “s=list(value)” donde s es la variable y value la variable que contiene la temperatura adquirida por el ADAM, luego de realizar estos pasos se procede a eliminar el primer bit y volver a juntar todos los bits para armar la nueva trama. Para llevar a cabo esto en Python se hizo de la siguiente forma “s[0]=’ ’ ” y se utilizó el comando “join” para juntar todo otra vez.

Una vez adquiridos los datos, solucionado el problema del primer bit y unida toda la trama otra vez se le agregó la fecha y la hora a cada dato entregado por los sensores a través del puerto serial y también se creó un archivo de texto, el cual tiene el propósito de ir guardando estos datos automáticamente. Para añadir la fecha y hora a los datos se utilizó el comando “strftime” de Python y para la creación del archivo de texto se utilizaron funciones de Python también.

Para que la toma de datos se realizara dentro de un periodo establecido se integró todo lo anterior y se introdujo dentro de un ciclo while donde las restricciones son la fecha de inicio y termino del estudio. Para definir estas fechas se debió utilizar la librería de Python “datetime”, la cual nos permite definir la hora, los minutos, los segundos y la fecha en la cual se desee realizar una acción, en el caso de este trabajo de título adquirir la temperatura, también se estableció un tiempo de muestreo, lo que significa que el ciclo se ejecutara cada cierto tiempo. En la figura 23 se puede apreciar el código completo del envío y recepción de datos y el posterior almacenamiento de los datos en un archivo de texto.

```
import serial
import time
import datetime

text_file = open("temperatura.txt","w")

sp = serial.Serial()


    while comienzo<datetime.datetime.now()<termino:


            s = "#02\r\n"
            sp.write(s.encode('ascii'))

            value = sp.readline()
            s=list(value)

            s[0] = ''
            s
            "".join(s)

            tiempo=time.strftime("%c")
            print tiempo, "".join(s)

            text_file.write(tiempo)

            text_file.write(" ")
            text_file.write("".join(s))
            text_file.write("\n")
            text_file.flush()
            sleep(muestreo)
```

Figura 23: Envío de datos y almacenamiento en archivo de texto Python.

3.4.3 Desarrollo programa para enviar emails

Para el desarrollo de este programa se utilizó la librería "smtplib" de Python y otras sublibrerias de esta como MIMEMultipart, MIMEBase y MIMEText para enviar emails con archivos adjuntos.

En objetivo de los emails en este proyecto es dar a conocer cuando se inicia y termina el proceso, por lo tanto de debió desarrollar un programa para cada proceso.

Primeramente se creó un programa para enviar un email notificando cuando se inicia la toma de datos, en este caso se utilizó la librería antes mencionada y se definieron algunas variables como los destinatarios de los emails y el asunto. Cabe destacar que estas variables irán establecidas en un archivo de configuración, el cual será modificado por la persona encargada de realizar el estudio, además se utilizó la misma base para realizar el programa notificando el término del proceso cambiando solo el asunto del correo electrónico. En la figura 24 podemos ver como se definen los datos necesarios para el envío del email como: los destinatarios, el asunto que puede ser el termino o el inicio del estudio y quien lo envía y también se puede apreciar cómo se programa el envío del email.

```
import smtplib
from email.MIMEMultipart import MIMEMultipart
from email.MIMEBase import MIMEBase
from email.MIMEText import MIMEText
from email import Encoders

to = 'fmmora88@gmail.com'
gmail_user = 'mathiasmora88@gmail.com'
gmail_pwd = '@felipe123'
SUBJECT = 'EL SISTEMA HA INICIADO LA TOMA DE MEDICIONES DE TEMPERATURA'
SUBJECT2 = 'EL SISTEMA HA TERMINADO DE TOMAR MEDICIONES EXITOSAMENTE'
TEXT = 'EL SISTEMA HA INICIADO LA TOMA DE MEDICIONES DE TEMPERATURA'
TEXT2 = 'EL SISTEMA HA TERMINADO DE TOMAR MEDICIONES EXITOSAMENTE'

def mailcomienzo():

    smtpserver = smtplib.SMTP("smtp.gmail.com",587)
    smtpserver.ehlo()
    smtpserver.starttls()
    smtpserver.ehlo
    smtpserver.login(gmail_user, gmail_pwd)
    header = 'To:' + to + '\n' + 'From: ' + gmail_user
    header = header + '\n' + 'Subject:' + SUBJECT + '\n'

    msg = header + '\n' + TEXT + ' \n\n'
    smtpserver.sendmail(gmail_user, to, msg)
    smtpserver.close()
```

Figura 24: Desarrollo para enviar email de inicio y termino.

Para el desarrollo del programa para enviar un correo electrónico con un archivo adjunto aparte de utilizar las funciones antes mostradas en el envío del email de inicio y término se agregaron las funciones antes mencionadas como: MIMEBase que es una clase base para todos las subclases MIME de un mensaje, MIMEMultipart que es una subclase de MIMEBase que nos permite crear un mensaje que esté compuesto por muchas partes y MIMEText que es una subclase de MIMEMultipart que nos da la opción de generar archivos de texto. En la figura 25 se aprecia cómo se utilizan estas clases y se genera el email con el archivo adjunto.

```
import smtplib
from email.MIMEMultipart import MIMEMultipart
from email.MIMEBase import MIMEBase
from email.MIMEText import MIMEText
from email import Encoders

to = 'fmmora88@gmail.com'
gmail_user = 'mathiasmora88@gmail.com'
gmail_pwd = '@felipe123'
SUBJECT = 'EL SISTEMA HA INICIADO LA TOMA DE MEDICIONES DE TEMPERATURA'
SUBJECT2 = 'EL SISTEMA HA TERMINADO DE TOMAR MEDICIONES EXITOSAMENTE'
TEXT = 'EL SISTEMA HA INICIADO LA TOMA DE MEDICIONES DE TEMPERATURA'
TEXT2 = 'EL SISTEMA HA TERMINADO DE TOMAR MEDICIONES EXITOSAMENTE'
text_file = open("temperatura.txt","w")

def mail():
   msg = MIMEMultipart()

   msg['From'] = "mathiasmora88@gmail.com"
   msg['To'] = 'fmmora88@gmail.com'##"luisvera.cimubb@gmail.com"
   msg['Subject'] = "Mediciones de Temperatura"

   msg.attach(MIMEText("Registros de temperatura ADAM 4018-2"))

   part = MIMEBase('application', 'octet-stream')
   part.set_payload(open("temperatura.txt", 'rb').read())
   Encoders.encode_base64(part)
   smtpserver = smtplib.SMTP("smtp.gmail.com",587)
   part.add_header('Content-Disposition','attachment; filename="%s"' % os.path.b
   msg.attach(part)

   mailServer = smtplib.SMTP("smtp.gmail.com", 587)
   mailServer.ehlo()
   mailServer.starttls()
   mailServer.ehlo()
   mailServer.login("mathiasmora88@gmail.com","@felipe123" )
   mailServer.sendmail("mathiasmora88@gmail.com",'fmmora88@gmail.com', msg.as_st
   mailServer.close()
```

Figura 25: utilización de clase MIME y envío de email con archivo adjunto

Como se observa en la figura 25 se adjunta el archivo en el cual se fueron almacenando las temperaturas cuyo nombre es “temperatura.txt” y este se envía a los destinatarios que fueron definidos anteriormente.

Si se desea agregar más de una dirección de correos electrónicos estos deberán definirse en un archivo de configuración que está disponible en la placa Intel Galileo.

3.4.4 Desarrollo de programa para identificar conexión a internet

En el desarrollo del programa para verificar si existía conexión a internet se ocupó el modulo “urllib2” de Python. Su realización fue sencilla, ya que solo se necesita hacer un ping a una dirección de internet que sea válida y esté vigente En el caso de este proyecto se utilizó la dirección de Google. Si logra identificar que existe conexión a internet retornará verdadero y ejecutará los programas de envío de correos electrónicos, por el contrario si no identifica conexión a internet retronara falso y no enviara los correos electrónicos pero si tomara los datos de temperatura y los almacenará en la tarjeta SD. En la figura 26 se puede ver el programa con el cual se realizó lo anterior.

```
def internet_on():

   try :
     urllib2.urlopen("http://google.com")
     return True
   except urllib2.URLError,e:
     return False
```

Figura 26: Verificación conexión a internet

3.4.5 Ejecución de programas

Finalmente se integran todos los programas anteriormente señalados dentro de una función main para ser ejecutados de forma ordenada. Primeramente se debió poner un tiempo de espera antes de ejecutar el programa que está limitado por el comienzo de la toma de datos. Esto quiere decir que antes del inicio de la toma de datos el sistema estará en sleep y luego

que pase este periodo de espera se iniciara con la verificación de la conexión a internet. Como anteriormente se explicó si existe conexión a internet se ejecutaran los programas de envío de correos electrónicos junto con la adquisición de los datos y el almacenamiento de los datos en un archivo de texto, pero si no es así se ejecutara solo la adquisición de datos y el almacenamiento en el archivo de texto. En la figura 27 se puede apreciar el programa que define lo anteriormente planteado.

```
if __name__ == "__main__":

    while datetime.datetime.now() < comienzo :
        time.sleep(1)

    if internet_on() == 1:
        mailcomienzo()
        main()
        sp.close()
        text_file.close()
        mail()
        mailFin()
    else:
        main()
        sp.close()
        text_file.close()
```

Figura 27: Ejecución completa de los programas

El código completo de la aplicación se encuentra en el anexo V.

3.5 Conexión entre placa Intel Galileo y maleta adquisidora

Para la conexión entre la placa y la maleta se confeccionó un conversor RS232 a TTL macho, ya que la placa microcontroladora posee dos pines pin 0 (Rx) y pin 1 (Tx) que proveen comunicación serial UART TTL (5V/3.3V) y son los únicos pines en los cuales se puede establecer una comunicación serial con otro dispositivo.

Para la creación de este conversor se utilizó un circuito integrado max232, el que convierte señales RS232 a señales compatibles con niveles TTL. En la figura 28 se puede ver el dispositivo utilizado.

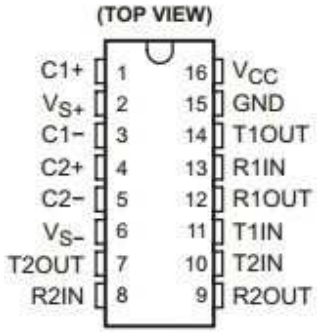

Figura 26: MAX232

El dispositivo encargado de enviar los datos y recibirlos desde y hacia la placa Intel galileo es el conversor RS232-RS485. A este dispositivo es al cual el conversor RS232 a TTL estará conectado y esto se realizó a través de un DB9 macho. En la figura 29 se muestra cómo deben ir conectado el DB9 al conversor RS232 a RS485 y la conexión desde el MAX232 hacia la placa Intel Galileo. En el capítulo 4.2 se muestra el conexionado eléctrico.

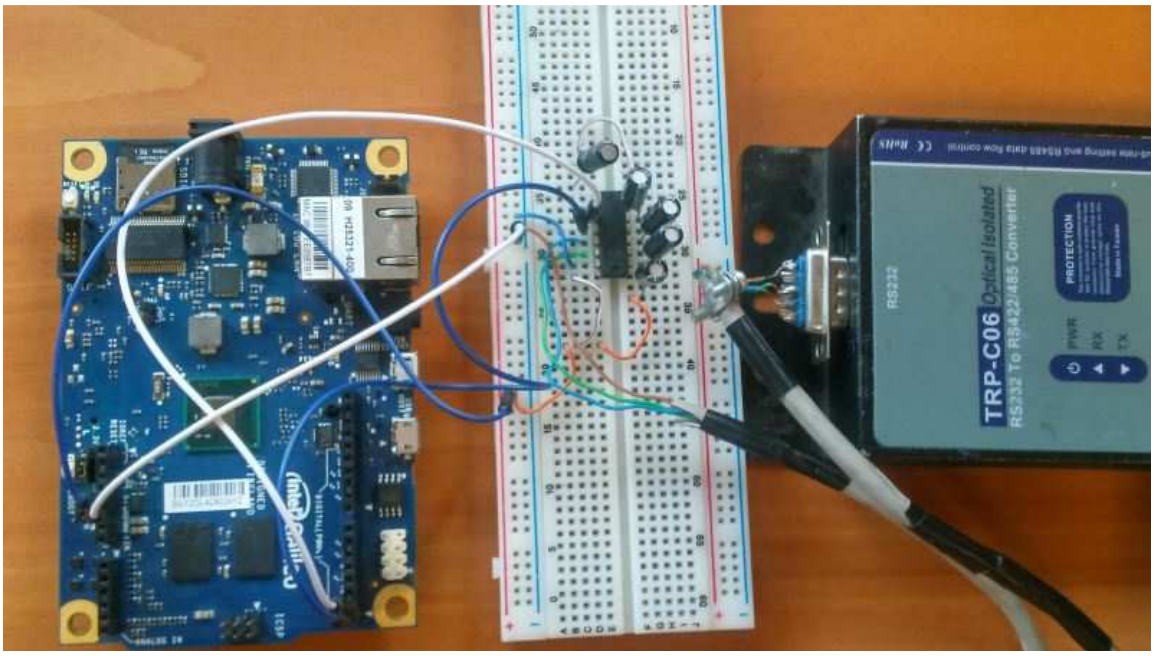

Figura 29: Conexión placa y maleta con Max232

4 Implementación y pruebas

4.1 Introducción

A continuación se detallaran las etapas para la implementación del sistema propuesto en este trabajo de título, desde la preparación del espacio físico hasta el sistema montado funcionando, además se incluirán pruebas de duración de la batería escogida para alimentar el sistema.

La implementación del sistema fue realizada en la sala de computación del segundo piso del CIMUBB, la figura 30 muestra como el sistema esta comunicado con los diversos dispositivos involucrados.

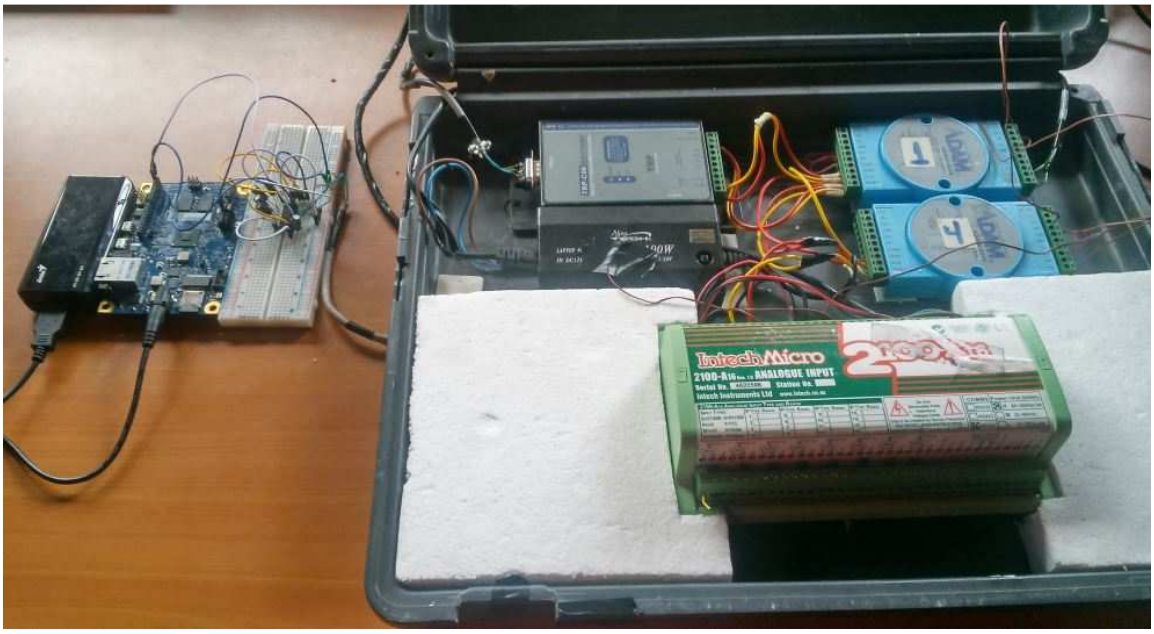

Figura 30: Sistema comunicado con los dispositivos

4.2 Preparación del entorno de prueba

Para llevar a cabo la implementación de este sistema fue necesario despejar la sala de computación del segundo piso del CIMUBB para así poder tener el espacio disponible para realizar las pruebas y además se necesitaba tener acceso a una red eléctrica para poder conectar la maleta.

Para realizar la instalación del sistema se utilizaron distintos espacios de la sala donde se ubicó el sistema y las termocuplas para tomar la información necesaria y comprobar que el sistema funcionaba de forma óptima además se modificó el número de termocuplas conectada a los ADAM, ya que en un inicio el sistema tenia conectada solo dos, luego de la modificación aumentaron a ocho termocuplas conectadas, esto con el objetivo de probar que sucedía con la batería si existían varios dispositivos conectados a los módulos de entrada y salida.

4.3 Instalación de dispositivos

La instalación del sistema constó de la placa microcontroladora Intel Galileo, el conversor RS232 a TTL, la maleta con todos sus componentes: Conversor RS232 a RS485, módulos de entrada y salida análogas ADAM-4018, las termocuplas y la batería externa de 6000 mAh alimentando el sistema.

En el caso de la placa Intel Galileo las características fueron presentadas en el capítulo 3 y ésta ira conectada a la batería externa la cual proporciona 5v y 6000mAh.

La figura 31 muestra un diagrama demostrativo de cómo deberían ir conectados todos los dispositivos anteriormente mencionados.

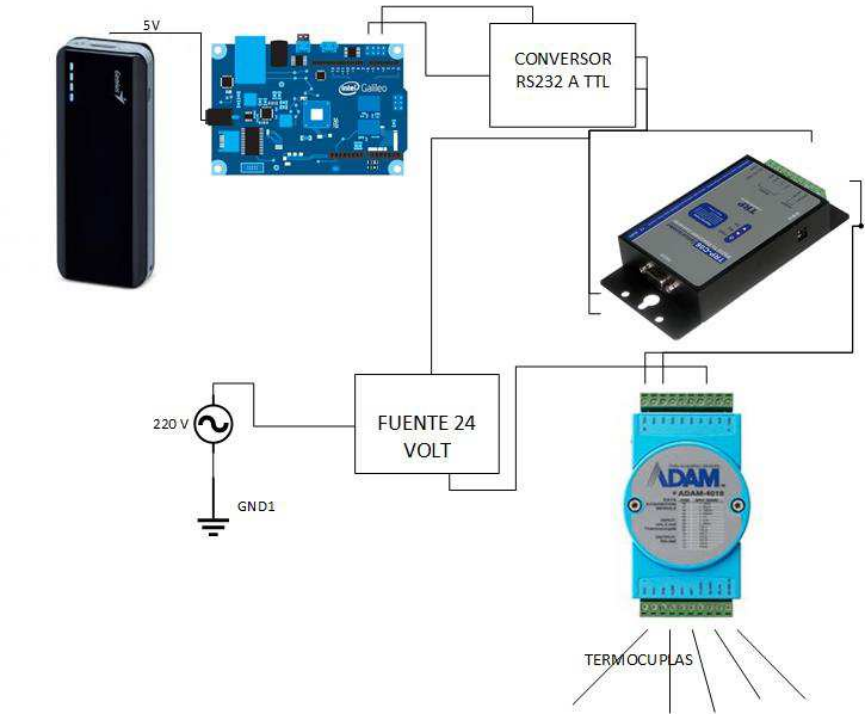

Figura 31: Diagrama demostrativo del conexionado implementado.

Como se puede apreciar en la figura 31 la batería está conectada directamente a la placa Galileo y esta a su vez va conectada a él conversor RS232 a TTL a través de los pines Tx y Rx de ambos. Esto se debe realizar con precaución, ya que si se conectan de mala forma el sistema no registrará los datos. El conversor RS232 a TTL a su vez está conectado al conversor RS232 a RS485 que va conectado al dispositivo adquisidor ADAM 4018, ambos se encuentran alimentados por una fuente de 24 Volts. Las termocuplas están conectadas al módulo de entradas ADAM 4018. Estas se distribuyeron de tal forma que tomaran distintas temperaturas algunas se ubicaron en los mismos dispositivos, una se ubicó para que midiera la temperatura ambiente y otra se ubicó a la intemperie para que tomara la temperatura externa.

4.4 Pruebas de funcionamiento

4.5 Software de adquisición de temperatura

Con el sistema ya instalado la fase de prueba consta de ejecutar el software de toma de datos en este caso temperatura, enlazado con el envío de correos electrónicos notificando el comienzo y término del proceso y además el envío de un correo electrónico con el archivo de texto adjunto con las temperaturas adquiridas durante el periodo de tiempo establecido.

Antes de ejecutar el programa principal de medición de temperatura es necesario modificar-lo parámetros de comienzo, termino y tiempo de muestreo del proceso. Esto se realiza en el archivo de configuración realizado exclusivamente para establecer estos parámetros y no intervenir en el código en la figura 34 se muestra el archivo de configuración que se encuentra en la placa Intel Galileo.

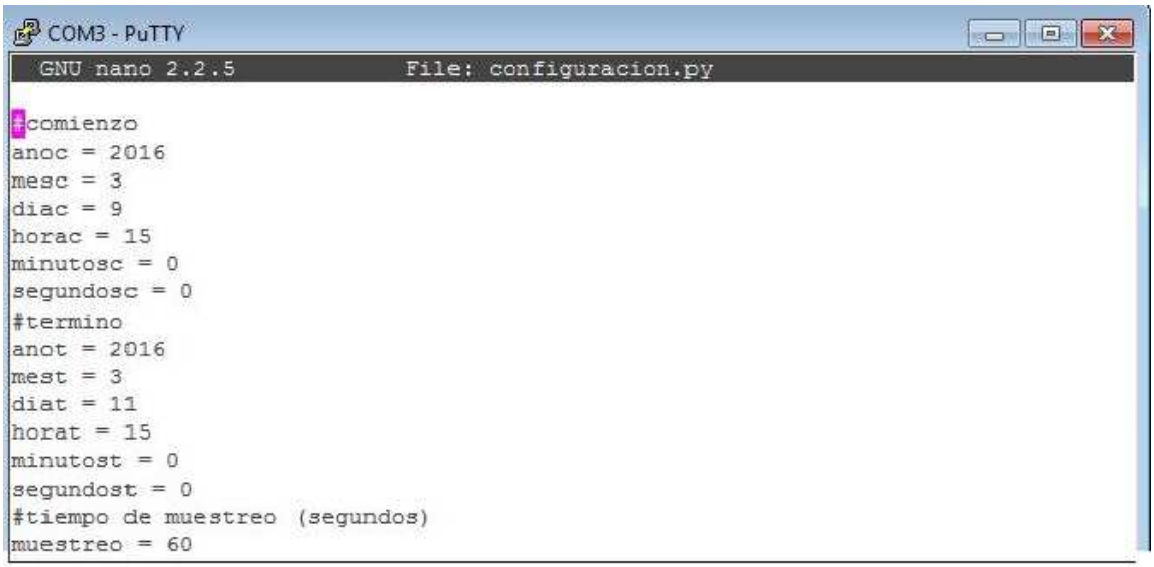

Figura 34: archivo de configuración

Para acceder a este archivo es necesario conectarse con la placa a través de la aplicación Putty descrita en el capítulo 3.3.4 y conectarse de forma serial a través de un cable Jack 3.5mm a USB (Anexo VI). Dentro de Intel Galileo se podrá ingresar al archivo de

configuración y cambiar los parámetros antes nombrados en la figura 35 se puede ver que comando utilizar para ejecutar el archivo de configuración.

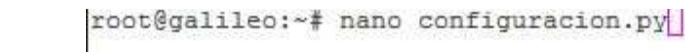

Figura 35: Comando para ejecutar archivo de configuración

Una vez establecidos los parámetros se procede a configurar la hora y fecha de la placa microcontroladora, ya que esta resetea estos parámetros dejándolos igual a la última vez que se utilizó. En la figura 36 se muestra como configurar la fecha y hora.

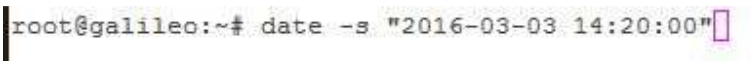

Figura 36: Configuración Hora y fecha

Después de establecer la hora y la fecha se debe tener cuidado de no desconectar la placa de la fuente de energía, ya que estos parámetros se volverán a resetear. Por esta razón lo recomendable es configurarlos en el lugar donde se realizará el estudio.

Luego de haber realizado todo lo anterior se puede ejecutar el programa principal encargado de adquirir los datos y enviar los correos correspondientes. En la figura 37 se muestra la instrucción para comenzar la ejecución del proceso. Como se mencionó anteriormente si la placa está conectada a internet se enviaran correos electrónicos notificando el comienzo y el término de la adquisición de datos y además un correo con el archivo adjunto correspondiente a los datos de temperatura adquiridos. En la figura 38, 39 y 40 se muestran los correos antes descritos respectivamente.

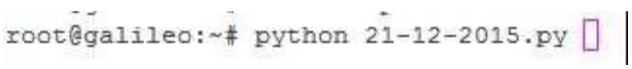

Figura 37: Instrucción para ejecutar programa de adquisición de temperatura.

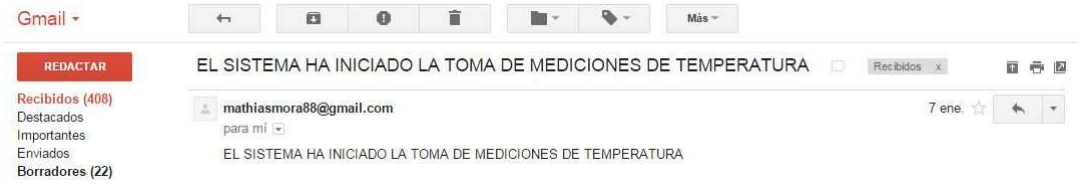

Figura 38: Correo de Inicio del proceso

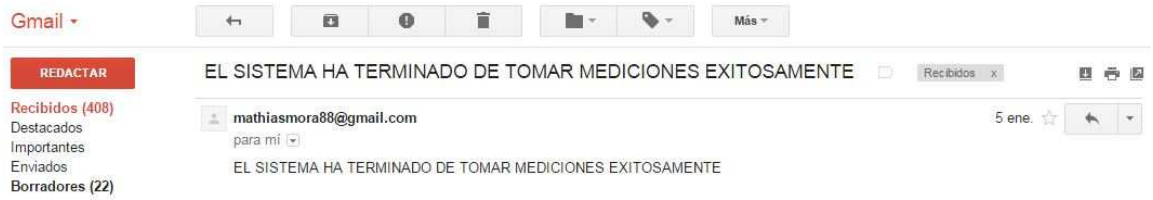

Figura 39: Correo de término del proceso

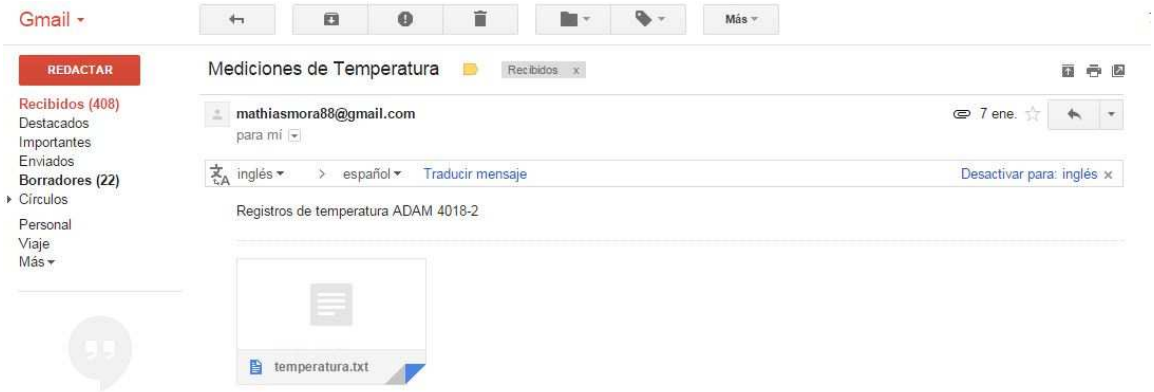

Figura 40: Correo con archivo de temperatura adjunto.

4.6 Organización de la información

Una vez realizada la prueba del software de adquisición de temperatura se debe comprobar que la información obtenida de los sensores, sea almacenada en la tarjeta MicroSD y si el sistema está conectado a internet que envié el correo con los datos de temperatura. En la figura 41 y 42 se muestra un extracto de los datos que almaceno en la tarjeta MicroSD y los datos que envió al correo respectivamente. En ambas figuras se logran ver varias columnas.

En la primera columna se muestra la fecha en la cual la temperatura fue tomada, en la segunda columna se muestra la hora en la cual fue tomada la temperatura y en las columnas siguientes se puede ver la temperaturas registradas en los ocho canales del módulo adquisidor.

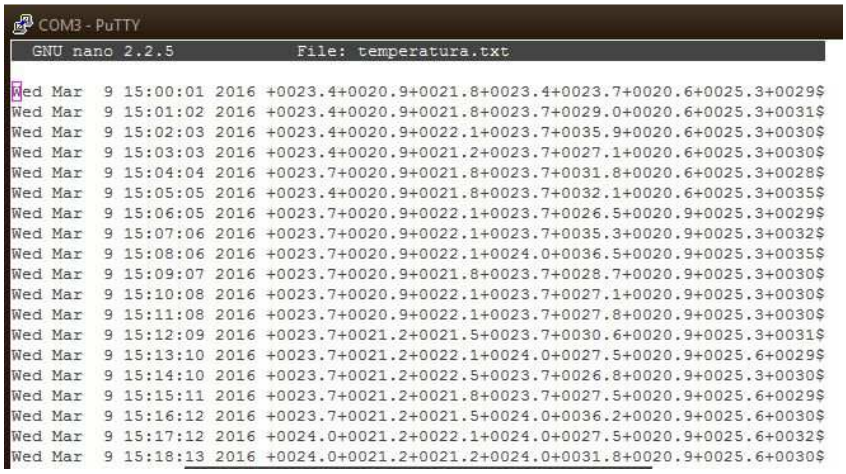

Figura 41: Datos almacenados en la tarjeta MicroSD en Intel galileo.

Figura 42: Datos enviados al correo en formato txt.

Ya con todas las etapas del software funcionando de manera correcta se puede decir que se cumplió parte de los objetivos, pero en el caso de que no tenga conexión a internet ¿Cómo se extraen los datos? De ese asunto tratará el siguiente capítulo.

4.7 Extracción de archivos desde Intel Galileo

Para la extracción de archivos de Intel Galileo se utilizará el programa WinSCP. La descripción de este programa está disponible en el capítulo 3.3.6.

Primeramente se debe le debe asignar una dirección IP a la placa Intel Galileo. Esto se realizó utilizando un router y conectando una salida del router hacia la placa microcontroladora y otra salida al computador que se esté utilizando. Esta conexión se debe realizar con cables Ethernet directos. Una vez realizado lo anterior se procede a identificar la IP asignada a la placa utilizando el programa Advanced IP Scanner. Programa que fue descrito en el capítulo 3.3.7. Se podrá identificar la IP mediante la MAC de la placa Intel que está disponible en la salida Ethernet. En la figura 43 se muestra como identificar la MAC y en la figura 44 se puede ver la identificación de la IP mediante la dirección MAC.

Figura 43: Identificación dirección MAC Intel Galileo

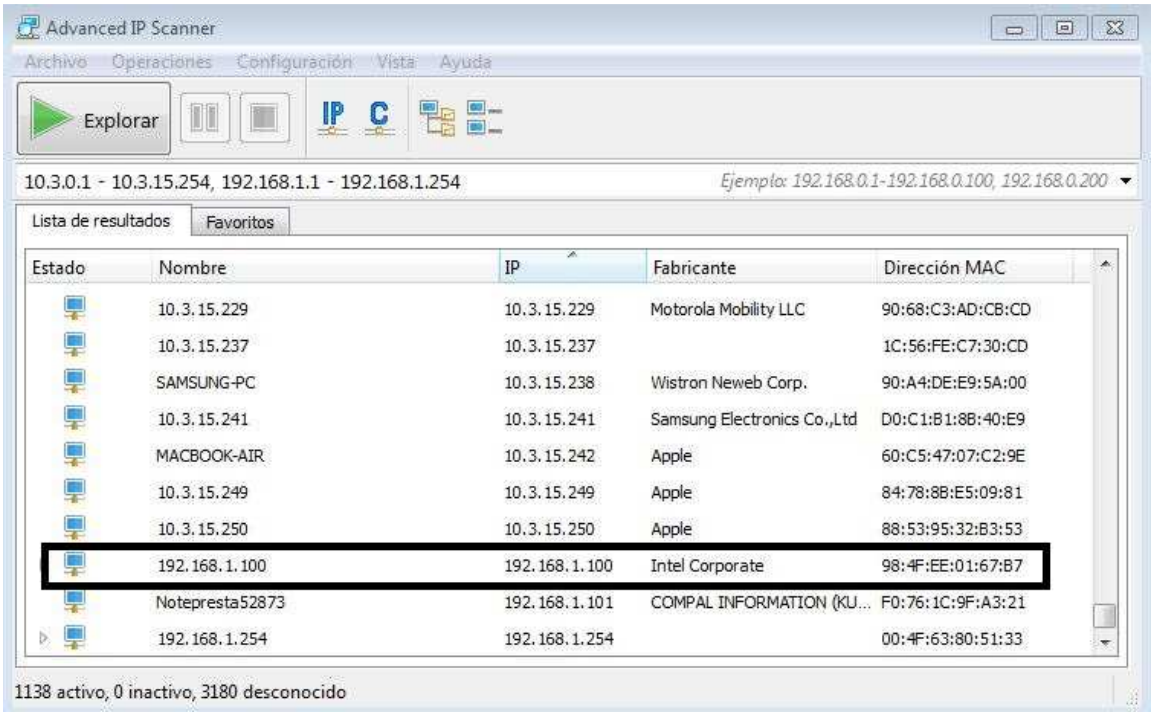

Figura 44: Identificación de IP Intel Galileo

Después de identificar la dirección IP de la placa se podrá ingresar a esta por medio del programa WinSCP en la figura 45 se muestra la pantalla principal del programa y que datos se necesitan.

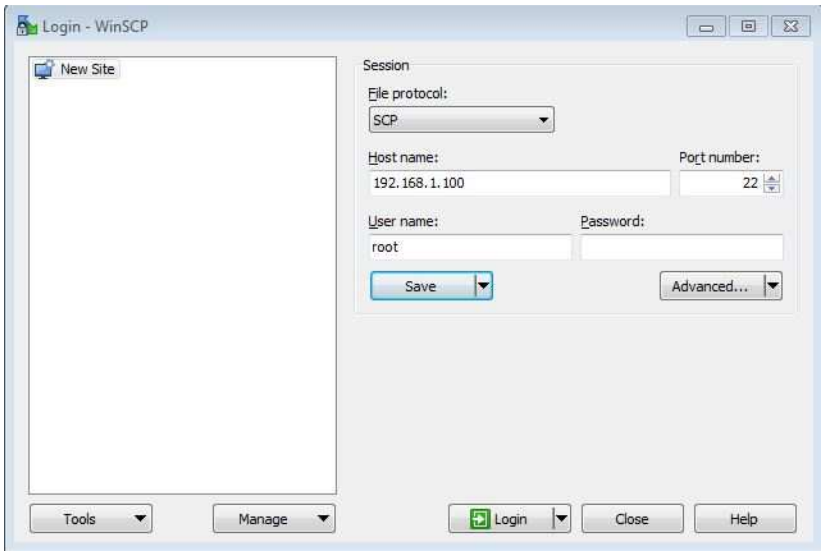

Figura 45: Conexión a WinSCP

En la figura 45 se puede notar que se piden algunos datos como la dirección IP que se obtuvo anteriormente, el protocolo de archivos donde se debe seleccionar SCP, el número de puerto que por defecto muestra el puerto 22 y el nombre de usuario que por defecto es root. La contraseña no es necesaria. Luego de completar estos datos se ingresa a la placa Intel Galileo a través del botón login.

Una vez dentro de la placa se mostrará algo similar a lo que se puede ver en la figura 46, donde se puede descargar los archivos que se deseen extraer, en este caso los valores de temperatura registrados durante el periodo de experimentación y además se puede cargar archivos desde el ordenador que se esté utilizando hacia la placa Intel Galileo.

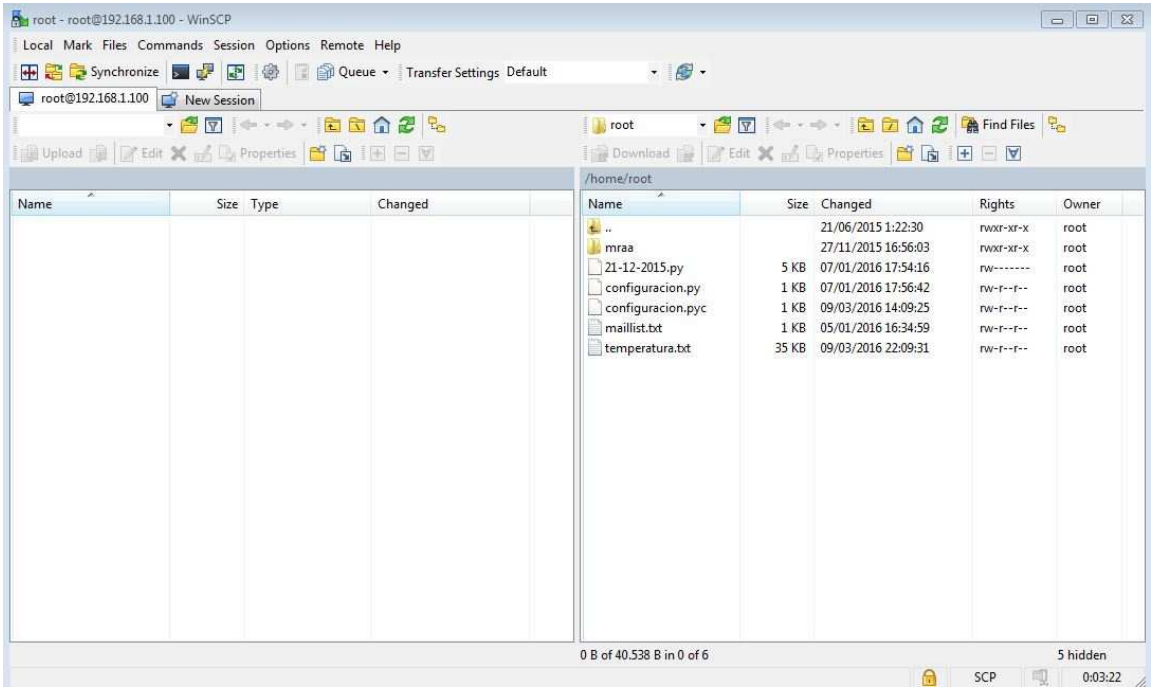

Figura 46: Descarga de archivo a través de WinSCP.

4.8 Resultado de las pruebas duración de la batería

Para efectuar este análisis se realizaron varios experimentos variando el tiempo de muestreo de la aplicación y así observar que sucede con la durabilidad de la batería en horas. Se mostrara un gráfico realizando la comparación entre la duración vs el tiempo de muestreo y además un análisis de los datos obtenidos durante la realización del experimento.

En el desarrollo de las pruebas se fue variando el tiempo de muestreo comenzando con una variación de 1 minuto, luego 5 minutos, 10 minutos, 15 minutos, 20 minutos, 30 minutos, 45 minutos y para finalizar 60 minutos.

La batería utilizada para realizar estos experimentos es descrita en el capítulo 4.2.

Todo está experimentación se realizó con el objetivo de verificar si la variación en el tiempo de muestreo afectaba la durabilidad de la batería. En la tabla 4 se muestra los experimentos realizados con sus respectivas fechas, tiempo de muestreo y durabilidad de la batería en horas.

Tiempo de muestreo	Hora inicio	Hora termino	Duración (Horas)
1 minuto	15:00	22:09	7 horas, 9 min
5 minutos	15:00	22:41	7 horas, 41 min
10 minutos	15:30	23:10	7 horas, 40 min
15 minutos	11:00	18:00	7 horas
20 minutos	14:10	21:50	7 horas, 50 min
30 minutos	11:40	19:10	7 horas, 30 min
45 minutos	13:43	20:43	7 horas 43 min
60 minutos	11:30	19:30	8 horas

Tabla 4: Datos Experimentos realizados con Batería.

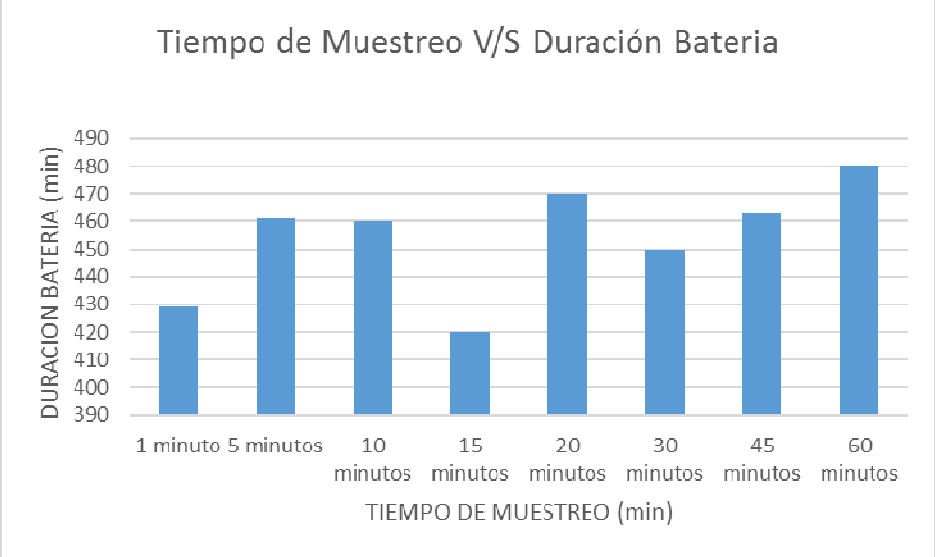

Grafico 1: Tiempo de Muestreo V/S Duración Batería.

Al analizar los resultados que se muestran en la tabla 4.2 y en el grafico 1 se puede concluir que el tiempo de muestreo no influye en la durabilidad de la batería, ya que la duración de esta se mantiene constante no importando el tiempo que se establezca entre toma de muestras del experimento. Las variaciones que se aprecian son por efecto del uso anterior a la toma de muestras del experimento. Esto quiere decir que se utilizó para realizar las configuraciones pertinentes para el desarrollo del experimento como: establecer la fecha, tiempo de muestreo y fecha de inicio y término del experimento, además existe un tiempo entre el inicio del experimento y el inicio de la toma de muestras que fue diferente en todos los experimentos realizados.

Al obtener estos resultados se buscó una explicación a porque ocurría esto y se plantearon algunas soluciones como por ejemplo disminuir los componentes conectados a la placa microcontroladora, tratar de poner algunas funciones de la placa en suspensión o alimentar Galileo con una batería de mayor amperaje por hora. La primera opción no era factible, ya que los componentes conectados a la placa son necesarios para el funcionamiento de esta, como la salida Ethernet, los pines de transmisión y recepción y la salida de alimentación para en conversor RS232 a TTL. La segunda opción parecía una buena solución, pero la placa Intel Galileo no permite deshabilitar componentes de su hardware ni tampoco es posible administrar el consumo de energía de su procesador Quark, ya que este según la bibliografía no posee algunos de los opcodes, más específicamente el MWAIT opcode, que es utilizado para poner los núcleos en un estado de bajo consumo hasta que ocurra un evento que los despierte, y la tercera opción descrita anteriormente sería la más factible de implementar, aunque se necesitaría una batería de un alta capacidad en corriente/hora para que abasteciese a la placa durante largos periodos de tiempo que es lo que dura realizar un experimento.

De acuerdo a las conclusiones obtenidas anteriormente, la batería propuesta para este trabajo de título solo sería utilizable en estudios cuyos periodos de prueba sean muy cortos máximo 7 horas.

Conclusiones

Durante el transcurso de este proyecto de título se buscó desarrollar un sistema de registro de datos de bajo consumo eléctrico que reemplazara en parte al sistema actual utilizado por CITEC, para el desarrollo de estudios de confort térmico en recintos habitacionales. Este sistema a desarrollar debía tener ciertas características como: replicar algunas funcionalidades del sistema actual, debía comunicarse con los dispositivos mediante un protocolo de comunicación Modbus, la alimentación eléctrica para el dispositivo de adquisición de datos debía ser una batería, y ser guardados en una tarjeta MicroSD. De acuerdo a lo anterior se propuso el reemplazo del computador que se utilizaba para la toma de datos y, el posterior almacenamiento debía ser reemplazado por una placa microcontroladora.

Durante el desarrollo de este trabajo de título ocurrieron varias complicaciones para poder lograr los objetivos.

En primer lugar la comunicación Modbus que se debía realizar en un principio nunca se logró, puesto que los dispositivos ADAM 4018 no estaban diseñados para comunicarse por medio de este protocolo de comunicación. Esto se debió a que no se hizo una correcta lectura del manual y la información entregada por parte de los encargados del sistema anterior era errónea.

En segundo lugar fue una gran dificultad y decepción trabajar con la placa microcontroladora Intel galileo, debido a que era nueva en el mercado, no había mucha información bibliográfica sobre su funcionamiento, su desarrollador Arduino, aseguraba que era totalmente compatible en hardware y en software con Arduino, afirmación que no era totalmente cierta, ya que existían librerías que se podían utilizar en Arduino, pero no así en Intel Galileo, su consumo energético es mayor que otras placas con las mismas características como por ejemplo Raspberry y un sinfin de complicaciones que surgieron en el desarrollo de este proyecto. Desafortunadamente el sistema estaba condicionado para

realizarse en Intel Galileo, si esto no hubiese sido de esta forma, claramente se hubiese escogido otra placa microcontroladora para su desarrollo.

En tercer lugar al momento de querer reducir costos de energía en la placa Intel galileo no se podía efectuar, ya que como se menciona en el capítulo 4.10, la placa microcontroladora no permite deshabilitar componentes de su hardware ni tampoco es posible administrar el consumo de energía de su procesador Quark.

Como conclusión general se puede decir que el sistema propuesto en este trabajo de título no satisface totalmente el objetivo propuesto inicialmente, sin embargo se logró comunicar la placa microcontroladora con la maleta de dispositivos, adquirir datos de los sensores y almacenarlos en una tarjeta microSD. También se logró el envío de los datos adquiridos por medio de un correo electrónico a los encargados de los experimentos y abastecer a la placa Intel con una fuente de independiente de energía, que eran parte de los objetivos de este proyecto de título.

Para finalizar cabe destacar que este proyecto posee un mérito innovador, ya que se desarrolló en una placa de desarrollo Intel, trabajo que no se había realizado anteriormente y de la cual no se encontró artículos y tampoco bibliografía al respecto.

Existe la posibilidad que a futuro, en próximos trabajos de título, se pueda mejorar y optimizar el proceso de adquisición y registro, es posible reemplazar la placa actualmente utilizada por una de mayor y mejor característica y capacidad, en donde se pueda integrar una pantalla LCD o un teclado. Además se podría buscar una forma de reducir su consumo modificando los estados de la CPU, donde se consuma energía solo cuando se esté utilizando y este en suspensión cuando no.

Bibliografía

[1] Bob Steigerwald, Rajshree Chabukswar, Karthik Krishnan, Jun De Vega. "Creating Energy-Efficient Software". Copyright 2007 Intel Corporation. All Rights Reserved.

[2] Suman Nath. "Energy-Efficient Sensor Data Logging with Amnesic Flash storage". 2009. Information Processing in Sensor Networks. IPSN. International Conference On.

[3] Jason Dunham, Gwynne Chandler, Bruce Rieman, Don Martin. 2005 "Measuring Stream Temperature with Digital Data Loggers: A User's Guide". Gen. Tech. Rep. RMRS-GTR-150WWW. FortCollins, CO: U.S. Department of Agriculture, Forest Service, Rocky Mountain Research Station. 15p

[4] Tajana Simunic, Luca Benini, Giovanni De Micheli. 2001 "Energy-Efficient Design of Battery-Powered Embedded Systems". IEEE Transactions on very large scale integration (VLSI) systems, VOL. 9. NO. 1.

[5] Ed Baker. 2014. "Open Source Data Logger for Low Cost Environmental Monitoring". Journal 2: e1059. Doi: 10.3897/BDJ.2.e1059.

[6] H. Xianzhe. 2011. "Room Temperature and Humidity Monitoring and Energy-Saving System" Computer Science & Education (ICCSE), 6th International Conference on.

[7] Stavros Harizopoulos, Mehul A. Shah, Justin Meza, Parthasarathy Ranganathan. 2009. "Energy Efficiency: The New Holy Grail of Data Management Systems Research". 4th Biennial Conference on Innovative Data Systems Research (CIDR), Asilomar, California, USA.

[8] T. A. Jesha. 2014 "Data Logging and Energy Consumption Analysis of Two Houses in St. John's, Newfoundland". Electrical and Computer Engineering (ICECE), International Conference on.

[9] "ADAM 4000 Series. Data Acquisition Modules. User's Manual", 1997. Advantech Co., Ltd Sarah Boyd, John Burfoot, Daniel Green, Cathie Howe. 2014

[10] Paolo Cocchi. 2015. "Analysing and Experimenting the Intel Galileo Board for the Internet of Things". Technical Report n.12, 2015.

[11] Miguel de Sousa, 2015. "Internet of Things with Intel Galileo". Published by Packt Publishing Ltd. Livery Place, 35 Livery Street, Birmingham B3 2PB, UK.

[12] "A teacher's Guide to the Intel Galileo" Macquarie ICT Innovations Centre (MacICT), Building C5B, Macquarie University, North Ryde, NSW, 2109.

.

[13] Agus Kurniawan. 2014. "The hands-on Intel Edison manual lab". ISBN: 978-1-312-73697-9, 1st Edition, 2014.

[14] "Programación de GPIO en Intel Galileo". Disponible en http://www.malinov.com/Home/sergey-s-blog/intelgalileo-programminggpiofromlinux. Enero Del 2016

[15] John Sharp. 2013. "Microsoft Visual C# 2013 Step by Step".

[16] "Instalación de Linux en Intel galileo". Disponible en https://software.intel.com/en-us/get-started-galileo-linux. Enero del 2016

[17] "Como instalar paquetes de Python en Windows" Disponible en: http://es.wikihow.com/instalar-paquetes-de-Python-en-Windows-7. Octubre del 2015.

Anexos

I. Arduino en Intel Galileo

Una de las formas de programar Intel Galileo es a través de Arduino. Este lenguaje se da de forma nativa en la placa, ya que posee la misma estructura en hardware y en software que los modelos de placa Arduino. A continuación se procederá a describir paso a paso la utilización de Arduino con la placa Intel Galileo.

Inicialmente se debe descargar la versión de Arduino que sea compatible con el sistema operativo que se esté utilizando y que sea para Intel Galileo, ya que hay versiones de Arduino que están diseñadas solo para placas Arduino. Luego de descargar el programa adecuado se procede a la instalación de este a continuación se describirá la instalación para los distintos sistemas operativos.

Para la instalación de Arduino en Windows se debe extraer el archivo previamente descargado en el disco C del computador como los muestra la figura 7. Como se aprecia en la figura se encuentran varios archivos dentro de la carpeta extraída, pero el archivo que interesa es el que lleva por nombre “Arduino.exe”. Este archivo es el ejecutable y es el que se utilizara para iniciar la interfaz de Arduino y poder programar en ella.

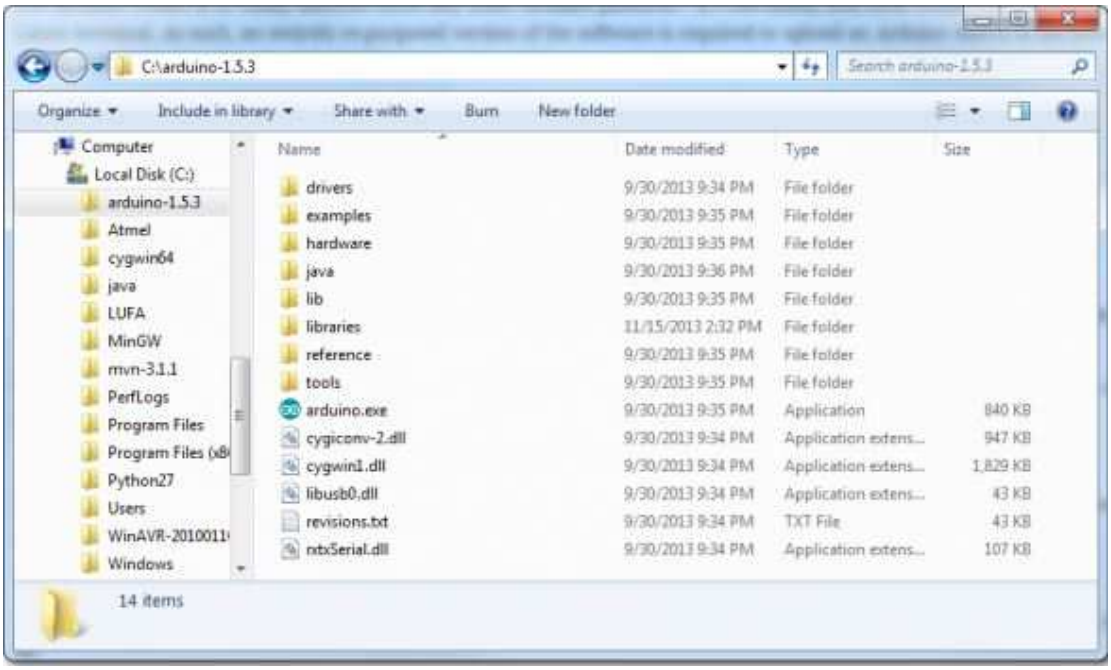

Figura 7: Arduino Windows S.O.

Para los usuarios de Mac OS se debe extraer la aplicación y ubicarla en la carpeta de aplicaciones. Luego se puede ejecutar haciendo doble click en la aplicación de Arduino.

En la figura 8 se muestra las aplicaciones de Arduino para usuarios Mac.

Figura 8: Aplicaciones de Arduino para Mac OS.

En el sistema operativo Linux se debe usar la herramienta “tar” para extraer el archivo “tar.gz”. El comando a utilizar se muestra en la figura 9.

```
tar -zxvf arduino-1.5.3-linux32.tar.gz
```

Figura 9: Comando para extraer archivos de Arduino en Linux.

También es necesario deshabilitar el modem manager en la mayoría de las distribuciones de Linux, para habilitar la carga de archivos a la placa Galileo. Las instrucciones para realizar esto s muestran en la figura 10 y estos comandos pueden variar de acuerdo a la distribución de Linux que se disponga.

```
sudo apt-get remove modemmanager
```

Figura 10: Comandos para deshabilitar el modem manager en Linux.

Una vez instalado Arduino en Linux, este se puede ejecutar ya sea ejecutando directamente el archivo ejecutable desde el directorio donde se extrajo o desde la terminal de Linux donde se debe ingresar la siguiente instrucción: “./arduino”.

II. Instalación de Drivers.

Una vez descargado e instalado el software de Arduino que corresponde al sistema operativo que se esté utilizando, el siguiente paso es conectar la placa e instalar los drivers. Este proceso difiere dependiendo del sistema operativo que se esté utilizando. A continuación se describirá el proceso instalación de drivers para Windows, Linux y Mac OS.

En Windows primeramente se debe conectar la placa a alimentación de 5V, luego se debe conectar un cable micro-B USB desde el puerto USB cliente de la Intel Galileo a un puerto USB disponible en el computador. Al momento de conectar la placa al computador Windows automáticamente intentara instalar el driver, pero fallara al hacerlo, por lo tanto se tendrá q realizar manualmente, esto se debe realizar en el administrador de dispositivos. En la figura 11 se muestra como actualizar el driver dentro del administrador de dispositivos.

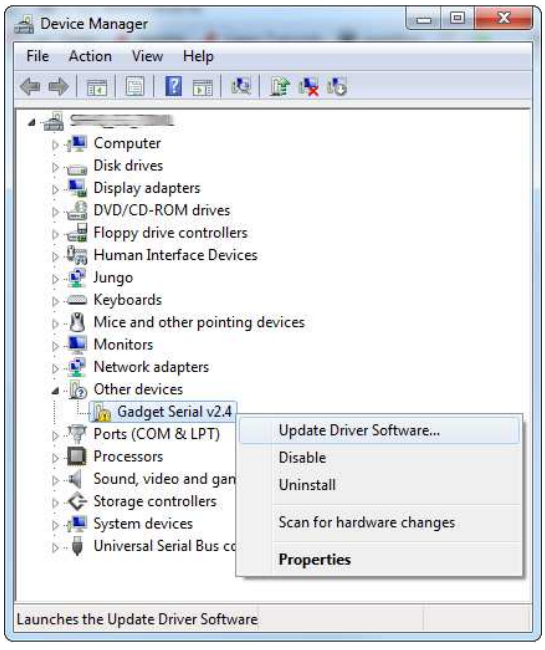

Figura 11: Actualización de Drivers Windows.

Luego de realizar lo anterior una ventana de Windows aparecerá. Se debe presionar el botón BROWSE y se tendrá que ir a la siguiente carpeta de Arduino: “hardware\arduino\x86\tools” y se debe continuar con la instalación. En la figura 12 se muestra lo anteriormente descrito.

Figura 12: Instalación Driver Arduino para Intel Galileo.

Después de la instalación de los drivers se tendrá que ingresar al administrador de dispositivos e identificar en que puerto COM se encuentra conectada la placa Intel galileo y memorizarlo, ya que será de suma importancia al momento de querer cargar programas en ella.

Para los usuarios de Mac el proceso de instalación de drivers es más sencillo, ya que Mac tiene soporte built-in driver para Galileo.

Primeramente como en todos los casos se debe conectar la placa a la alimentación, luego de esto conectar un cable micro-B USB desde el USB cliente de Galileo hacia un puerto USB disponible en el computador que se esté utilizado.

Se debe esperar un tiempo mientras la placa Intel se inicia. Para verificar que el sistema reconoció el dispositivo apropiadamente es necesario dirigirse a información del sistema y ver si en la opción de USB se encuentra “Gadget Serial v2.4”. En la figura 13 se muestra como verificarlo.

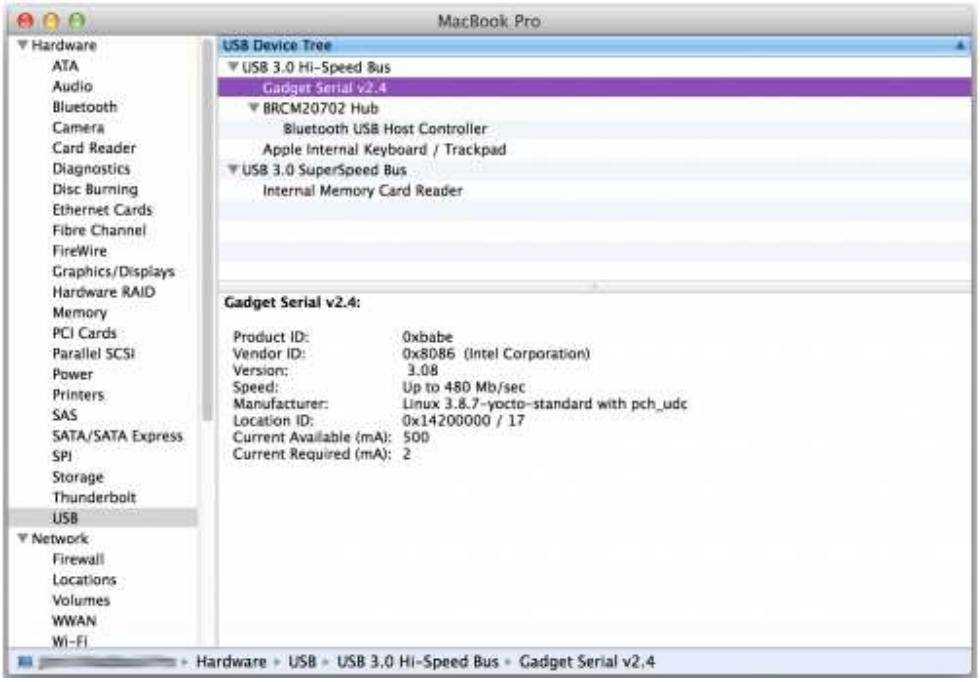

Figura 13: Verificación de Drivers de Intel Galileo Mac OS.

En el caso de los usuarios de Linux al Igual que los usuarios de Mac no es necesario descargar Drivers de instalación, ya que reconoce automáticamente el dispositivo.

En primer lugar es necesario conectar Intel Galileo a la alimentación y luego conectar un cable micro-B USB desde el cliente USB de la placa a un puerto USB disponible en el computador que se esté utilizando. Luego de realizar lo anteriormente descrito se debe abrir el terminal de Linux y escribir lo siguiente: ls/dev/ttyACM*.

Para finalizar es necesario verificar el número de puerto que se asignó a la placa y memorizarlo para la posterior carga de código que se quiera hacer a la Intel Galileo.

III. Instalación de Windows en Intel Galileo

Para poder descargar e instalar Windows para Intel Galileo primeramente se debe crear una cuenta en Microsoft Connect. Para esto se debe tener una cuenta Hotmail. Una vez descargada la imagen se debe cargar en una tarjeta microSD para luego ser introducida en la ranura microSD de Intel Galileo.

Para cargar la imagen en la tarjeta microSD primeramente se debe formatear eliminando cualquier rastro de información. Luego se abre la pantalla de símbolo del sistema y se escribe lo que se muestra en la figura 5.

```
cd /d %USERPROFILE%\Downloads
apply-bootmedia.cmd -destination {YourSDCardDrive} -image {.wimFile downloaded above} -hostname myg
alileo -password admin
```

Figura 5: Configuración para cargar imagen en Windows en MicroSD

Luego de cargar la imagen a la tarjeta microSD se procederá a ubicarla en la ranura microSD de Galileo y después de realizar esto se conecta la placa a la alimentación y al computador.

Para interactuar con Galileo con Windows como sistema operativo se necesita tener instalado TELNET programa descrito en este trabajo de título. En la figura 6 se muestra la interfaz de TELNET.

Figura 6: Interfaz de TELNET.

Después de haber utilizado Intel Galileo de debe apagar escribiendo los siguientes comandos en TELNET: “shutdown /s /t 0” y el sistema debería apagarse completamente.

Para crear programas con Windows en la placa se debe tener a disposición Visual Studio 2013 o superior. En esta plataforma se podrá diseñar todo tipo de programas que se quieran ejecutar en Galileo.

IV. Instalación de Linux en Intel Galileo

Para proceder a instalar Linux en Intel Galileo se necesita tener a disposición una tarjeta SD de al menos 1GB de memoria y menos de 32GB, además se necesita descargar la imagen de Linux proporcionada por Intel en su página oficial. Se debe descargar la última actualización del archivo.

Una vez descargada la imagen se deben extraer los archivos y cargarlos a la tarjeta SD, pero antes de efectuar esta acción la microSD debe estar totalmente formateada para eliminar cualquier rastro de información anterior. Luego de haber realizado el formateo se procede a cargar los archivos extraídos de la imagen descargada a la tarjeta SD.

En la figura 4 se muestran los archivos que contiene la imagen proporcionada por Intel.

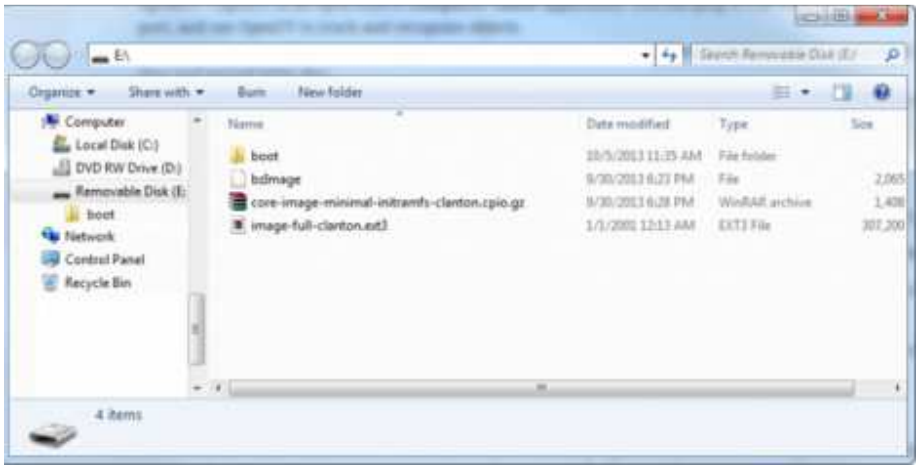

Figura 4: Archivos contenidos en la imagen de Linux para Intel Galileo

Una vez realizado lo anteriormente mencionado se debe poner la tarjeta SD dentro de la ranura microSD de la placa Intel Galileo y se conecta a la alimentación. Primeramente se cargara el sistema operativo que esta por defecto dentro de la placa y luego el que se ha cargado dentro de la tarjeta SD. Para visualizar lo que ocurre dentro de la placa se utiliza el programa antes descrito en esta tesis Putty en Windows o miniterm en Linux dependiendo del sistema operativo que se tenga instalado en el computador que se esté utilizando.

V. Código aplicación

```
import serial
import time
import datetime
from time import sleep
from configuracion import*
import smtplib
from email.MIMEMultipart import MIMEMultipart
from email.MIMEBase import MIMEBase
from email.MIMEText import MIMEText
from email import Encoders
import os
import winsound
import urllib2

to = 'fmmora88@gmail.com'
gmail_user = 'mathiasmora88@gmail.com'
gmail_pwd = '@felipe123'
SUBJECT = 'EL SISTEMA HA INICIADO LA TOMA DE MEDICIONES DE TEMPERATURA'
SUBJECT2 = 'EL SISTEMA HA TERMINADO DE TOMAR MEDICIONES EXITOSAMENTE'
TEXT = 'EL SISTEMA HA INICIADO LA TOMA DE MEDICIONES DE TEMPERATURA'
TEXT2 = 'EL SISTEMA HA TERMINADO DE TOMAR MEDICIONES EXITOSAMENTE'
text_file = open("temperatura.txt","w")

sp = serial.Serial()


comienzo = datetime.datetime(anoc,mesc,diac,horac,minutosc,segundosc)
termino = datetime.datetime(anot,mest,diat,horat,minutost,segundost)



def mail():
   msg = MIMEMultipart()

   msg['From'] = "mathiasmora88@gmail.com"
   msg['To'] = 'fmmora88@gmail.com'##"luisvera.cimubb@gmail.com"
   msg['Subject'] = "Mediciones de Temperatura"

   msg.attach(MIMEText("Registros de temperatura ADAM 4018-2"))

   part = MIMEBase('application', 'octet-stream')
   part.set_payload(open("temperatura.txt", 'rb').read())
   Encoders.encode_base64(part)
   smtpserver = smtplib.SMTP("smtp.gmail.com",587)
   part.add_header('Content-Disposition','attachment; filename="%s"' % os.path.basename("temperatura.txt"))
   msg.attach(part)

   mailServer = smtplib.SMTP("smtp.gmail.com", 587)
   mailServer.ehlo()
   mailServer.starttls()
   mailServer.ehlo()
   mailServer.login("mathiasmora88@gmail.com","@felipe123" )
   mailServer.sendmail("mathiasmora88@gmail.com",'fmmora88@gmail.com', msg.as_string())
```

```
    mailServer.close()

def mailcomienzo():

    smtpserver = smtplib.SMTP("smtp.gmail.com",587)
    smtpserver.ehlo()
    smtpserver.starttls()
    smtpserver.ehlo
    smtpserver.login(gmail_user, gmail_pwd)
    header = 'To:' + to + '\n' + 'From: ' + gmail_user
    header = header + '\n' + 'Subject:' + SUBJECT + '\n'

    msg = header + '\n' + TEXT + ' \n\n'
    smtpserver.sendmail(gmail_user, to, msg)
    smtpserver.close()

def mailFin():

    smtpserver = smtplib.SMTP("smtp.gmail.com",587)
    smtpserver.ehlo()
    smtpserver.starttls()
    smtpserver.ehlo
    smtpserver.login(gmail_user, gmail_pwd)
    header = 'To:' + to + '\n' + 'From: ' + gmail_user
    header = header + '\n' + 'Subject:' + SUBJECT2 + '\n'

    msg = header + '\n' + TEXT2 + ' \n\n'
    smtpserver.sendmail(gmail_user, to, msg)
    smtpserver.close()

def internet_on():

    try :
      urllib2.urlopen("http://google.com")
      return True
    except urllib2.URLError,e:
      return False



def main():
    sp = serial.Serial()
    sp.port = 'COM8'
    sp.baudrate = 9600
    sp.parity = serial.PARITY_NONE
    sp.bytesize = serial.EIGHTBITS
    sp.stopbits = serial.STOPBITS_ONE
    sp.timeout = 0.5 #tiempo de toma de datos
    sp.xonxoff = False
    sp.rtscts = False
    sp.dsrdtr = False
```

```
    sp.open()

    sp.readline() #to give the hardware handshake time to happen sp.open()


    while comienzo<datetime.datetime.now()<termino:


            s = A02 #"#02\r\n"
            sp.write(s.encode('ascii'))

            value = sp.readline()
            s=list(value)

            s[0] = ''
            s
            "".join(s)

            tiempo=time.strftime("%c")
            print tiempo, "".join(s)

            text_file.write(tiempo)

            text_file.write(" ")
            text_file.write("".join(s))
            text_file.write("\n")
            text_file.flush()
            sleep(muestreo)


if __name__ == "__main__":

    while datetime.datetime.now() < comienzo :
       time.sleep(1)

    if internet_on() == 1:
       mailcomienzo()
       main()
       sp.close()
       text_file.close()
       mail()
       mailFin()
    else:
       main()
       sp.close()
       text_file.close()
```

VI. Construcción de un cable Jack 3.5mm a DB9

Para realizar la comunicación serial entre la placa Intel Galileo y un computador existen dos maneras: la primera y la más engorrosa es tener a disposición un Router y efectuar una conexión SSH con la placa y la segunda es adquirir un cable 3.5mm a DB9 situación que es muy difícil puesto que no existen muchos vendedores de este producto, por lo tanto la solución a esta problemática es construirlo uno mismo. A continuación se explicara cómo fabricar este dispositivo.

Primeramente se debe tener a disposición una entrada Jack 3.5mm, tres cables de cobre, un conector DB9 hembra y artículos como cautín, soldadura, etc. En la figura 1 se muestra el conector DB9 y en la figura 2 se puede ver el conector Jack 3.5mm.

Figura 1: Conector DB9 Hembra.

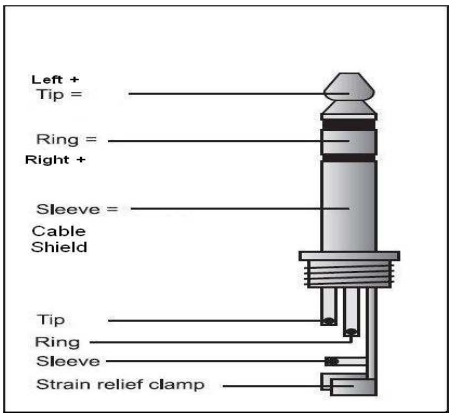

Figura 2: Conector Jack 3.5mm

Luego de tener todos los componentes a nuestro alcance se procede al armado del cable conector. En primer lugar se debe conectar uno de los extremos de los cables de cobre al pin Rx, Tx y GND del conector DB9 que son los pines 3, 2 y 5 respectivamente, luego el otro extremo se debe conectar al Rx, Tx y GND del conector Jack 3.5mm que son el anillo , la punta y la cubierta. En la figura 3 se aprecia el esquema del cable construido.

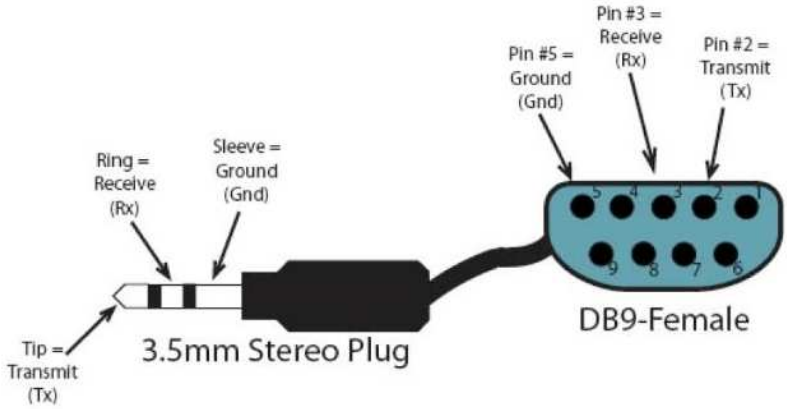

Figura 3: Cable Jack 3.5mm a DB9