UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO DE FIN DE GRADO EN INGENIERIA EN TECNOLOGÍAS INDUSTRIALES

IMPLEMENTACION Y DESARROLLO DE UNA RED DE SENSORES INALAMBRICA CON ALMACENAMIENTO DE LOS DATOS EN LA NUBE

AUTOR: ESTEBAN SANCHIS, ALEJANDRO

TUTOR: GARCÍA-DIEGO, FERNANDO J.

Curso Académico: 2015-2016

AGRADECIMIENTOS

Quiero aprovechar la ocasión para agradecer el apoyo de mi tutor Fernando Juan García Diego por darme la oportunidad de llevar a cabo un proyecto tan interesante a la par de instructivo. También me gustaría agradecerle su paciencia, ayuda, e implicación a la hora de guiarme a través de las tareas realizadas.

También me gustaría dar las gracias a mi hermano Borja Esteban Sanchis, quien me inspiro a realizar este proyecto con Fernando. Ya que, al haber realizado el proyecto predecesor a este, me ha servido de guía y facilitado información de su proyecto.

Quisiera dar las gracias también a todos mis compañeros de laboratorio; a Ángel Fernández Navajas colaborador del proyecto y diseñador del programa Ratón; Paloma Merello Giménez doctorada en economía y estadística, antigua colaboradora de Fernando Juan y del proyecto de Borja Esteban, y actual profesora en la Universidad de Valencia.

Para terminar, dedicarle el proyecto a mi familia, por todas las ayudas que me han ofrecido, la confianza que han puesto en mí y por ofrecerme y financiarme la oportunidad de sacarme el título de Ingeniero Industrial.

RESUMEN

El conocimiento del estado actual de una maquinaria, proceso, obra, economía e incluso población se alcanza debido al estudio de todas las variables y parámetros físicos que les rodean, interfieren y conforman.

Más concretamente, en el apartado de las ciencias y centrándose en la rama de la industria, es necesario conocer las variables que guían, interfieren o desgastan todo proceso productivo. La obtención de estas variables, a lo largo de los años, se ha llevado a cabo mediante el uso de sensores. Conocido este concepto, se procede a la elaboración de un proyecto centrado en el diseño de un *datalogger* para la adquisición e interpretación, de los datos proporcionados por dichos sensores.

Otro campo donde se está comenzando a instalar sensores, es en el área del patrimonio cultural, ya que variables como temperatura, humedad y radiación solar pueden afectar el estado de la obra, fachada, monumento… Por ello, la motivación para el diseño de este datalogger de utilidad para la conservación del patrimonio cultural.

En la actualidad, existe gran variedad dataloggers ya diseñados y fabricados, incluso aplicados ya a la conservación de patrimonio. Sin embargo estos dataloggers, a pesar de su amplia eficiencia a la hora de gestionar datos, requieren de cableado para abarcar superficies extensas, además de un técnico encargado de la recogida de datos cada cierto periodo de tiempo.

Para solventar estos inconvenientes, se decide incorporar las siguientes mejoras a los dataloggers instalados actualmente:

En primer lugar, no se dispondrá de un datalloger formado por un único componente, sino que será creada una red de dataloggers con autonomía propia para la adquisición de datos, todos ellos subordinados a un datalogger maestro, pudiendo de esta forma monitorizar grandes superficies. El diseño de esta red, consistirá en una red mallada o *mesh* coordinada por el modulo maestro, dentro de la cual circularán todos los datos adquiridos para posteriormente ser almacenados en un dispositivo de memoria externa instalado en la placa maestro.

En segundo lugar, se dotara al datalogger con conexión con la nube vía Wifi, de esta forma, todos los datos tomados en el trancurso del día estarán disponibles en la base de datos Dropbox, facilitando la interacción, recogida y análisis de datos por el usuario. Asimismo, una ventaja añadida, recae en que la conexión con la nube, también dotará al datalogger con la capacidad de avisar vía Twitter en caso de registrar datos anómalos o peligroso en algún sensor.

Con esta serie de mejoras; y proponiendo otras nuevas encaminadas a futuros proyectos, se pretende conseguir un prototipo de datalloger eficaz, cómodo, atractivo para el mercado, y que facilite las técnicas multivariable de adquisición de datos, en el entorno del patrimonio cultural y a ser posible en la Industria.

ÍNDICE TFG

TABLA DE ILUSTRACIONES

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO DE FIN DE GRADO EN INGENIERIA EN TECNOLOGÍAS INDUSTRIALES

MEMORIA

AUTOR: ESTEBAN SANCHIS, ALEJANDRO

TUTOR: GARCÍA-DIEGO, FERNANDO J.

Curso Académico: 2015-2016

Índice de la memoria

1. Motivación

Las normas internacionales (10/2001 112/1998 arte y DM, DL UNI 10829) de conservación del patrimonio cultural recomiendan unos umbrales de humedad relativa y temperatura que permitan la conservación de las obras de arte en unas condiciones óptimas. Por medio del análisis y monitorización de medidas multivariables, puede evitarse su efecto nocivo a largo plazo.

El proceso de conservación que permite salvaguardar el patrimonio cultural, viene respaldado por el diseño de varios dataloggers encargados de medir estas variables. Gracias a ello, se puede conocer el momento en el que es necesario tomar medidas de restauración o mejorar las condiciones en que se conserva.

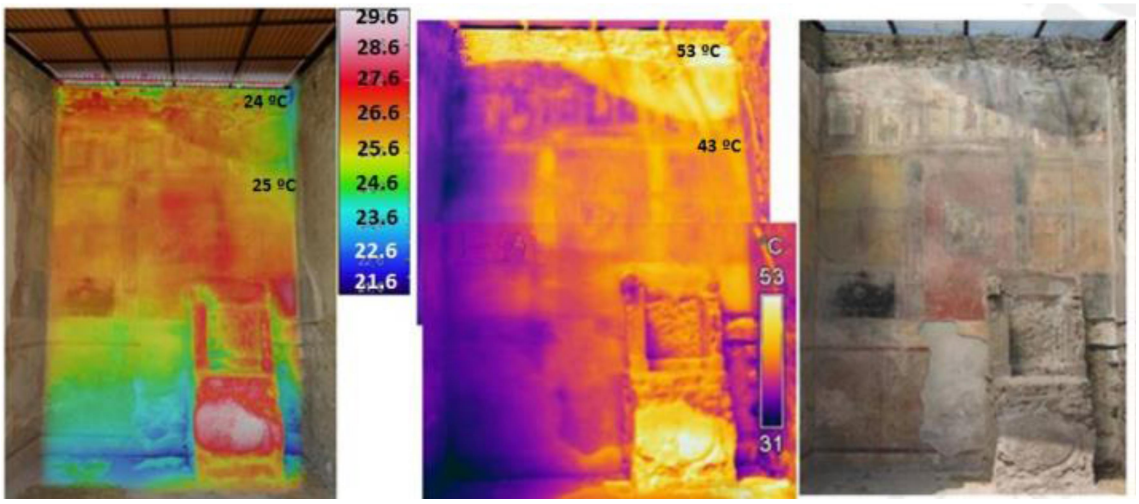


Ilustración 1. Estudio termografico de la casa de Ariadna (Pompeya) [1]

Las ilustraciones anteriores, muestran la humedad relativa y un termograma del murar de la Casa De Ariadna (Pompeya). En ellas puede observarse como las zonas más degradadas del mural coinciden con las zonas de mayor temperatura y humedad.

Es evidente la existencia de dataloggers diseñados y con un funcionamiento eficiente en la industria. Sin embargo, este proyecto pretende no solo llevar su uso a la conservación de patrimonio, sino mejorar esa adquisición de datos, permitiendo un análisis a tiempo real desde cualquier dispositivo con acceso a internet y abarcando grandes superficies gracias a la creación de una red Mesh.

Por otra parte, a pesar de que el diseño del prototipo está pensado para la conservación de obras, su empleo puede abarcar una gran diversidad de fines y tareas en el entorno industrial e incluso en el particular.

2. Antecedentes

2.1. Datalogger de partida

Conforme a lo anteriormente citado, actualmente coexiste diversidad de dataloggers destinados a la adquisición de datos. Uno de ellos es el diseñado en su proyecto final de carrera por Borja Esteban Sanchis para obtener su título en la Escuela Técnica Superior de Ingenieros Industriales (ETSII).

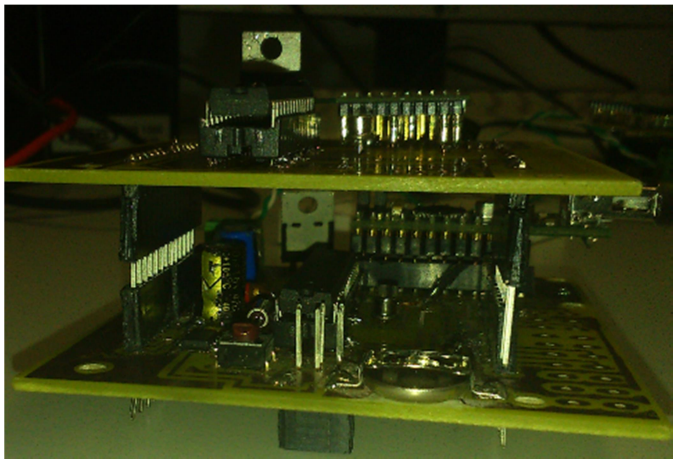
Este proyecto titulado **"Implementation and Design of a Datalogger with Microcontroller for Data Extraction from Sensors Using 1-WIre Protocol"** y publicado en la revista Sensors [1] así como premiado en 2015 por el colegio de ingenieros, es el antecesor a el presente proyecto. Por ello a continuación se realizara un resumen sobre su motivación, componentes y funcionamiento.

El PFC de Borja Esteban consistía en alcanzar la conservación preventiva de patrimonio mediante el uso de un datalogger inalámbrico programado mediante software Arduino, para la toma, transporte y posterior almacenamiento de datos. En un escueto resumen, el datalogger consta de las siguientes partes:

A- **Placa Maestro:**

Esta placa es la de mayor importancia, ya que se encarga de dirigir y coordinar el proceso de adquisición de datos, mediante los siguientes componentes:

- Microcontrolador ATMEGA328P.
- Dispositivo *Watchdog* o perro guardián, permite asegurar la correcta ejecución de las líneas de código. Reiniciándolo en caso de estancamiento en bucle.



- Ilustración 2. Placa Maestro PFC Borja Esteban Sanchis

- Un reloj DS1307 con su correspondiente cuarzo de 32kH, de esta manera el dispositivo conocía la hora exacta, siendo capaz de solicitar el suministro de datos cada minuto. Comunicado con el micro mediante el protocolo I_2C siguiendo el diagrama Arduino Uno, es decir, TWI pines SDA (A4) SCL (A5) y el uso de la librería 'Wire.h'.

- Módulo VDIP1 encargado de guardar los datos en una USB. Comunicado con el micro mediante el protocolo UART conectando pines Rx y Tx del microcontrolador con Tx y Rx del VDIP1 y usando la librería VDIP1_ART.h.

- Conexión directa con la placa Wireless, permitiendo la comunicación inalámbrica. Se realizara por medio del protocolo de comunicación I_2C usando la librería 'Wire.h' propia de Arduino.

B- Placa Esclavo:

La placa Esclavo es la encargada de recoger las lecturas de datos de humedad relativa y temperatura tomados por los sensores, renovando con ellos su buffer de memoria. Consta de los siguientes componentes:

- Microcontrolador ATMEGA328P.

- Dispositivo Watchdog para asegurar la correcta ejecución de las líneas de código.

- Convertidor I_2C t- 1Wire DS2482-800, dispositivo necesario en la recogida de los datos de los sensores permitiendo la colocación de una mayor cantidad y a mayor distancia mediante cableado.

- Conexión directa con placa Wireless mediante protocolo I_2C.

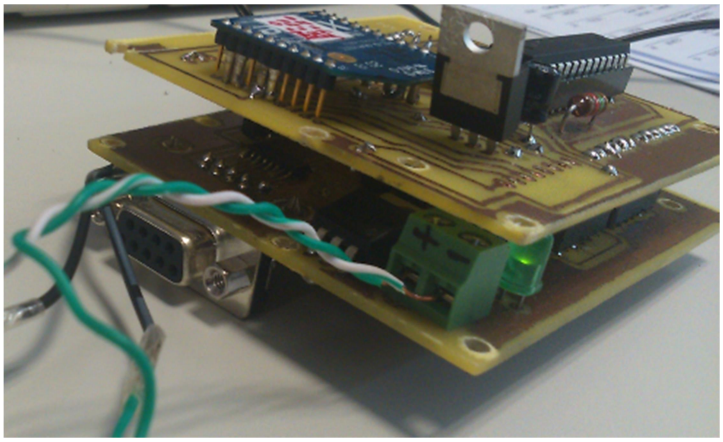


Ilustración 3. Placa Esclavo PFC Borja Esteban Sanchis

C- Placa Wireless:

La placa Wireless es la encargada de conseguir una comunicación inalámbrica entre las placas Maestro y Esclavo gracias a los siguientes componentes

- Microcontrolador ATMEGA328P

- Conexión directa con placa maestro y esclavo mediante comunicación I2C para conseguir comunicación con sus respectivas placas Wireless.

- Módulo inalámbrico Xbee S1 permitiendo una comunicación punto a punto mediante la especificación del canal y direcciones a través del programa X-CTU. Se encuentra comunicado con el microcontrolador mediante protocolo UART, de esta manera puede transmitir las órdenes del maestro al esclavo. También se encuentra comunicado con el esclavo mediante el canal programado, para así recibir los datos recogidos por el esclavo.

Ilustración 4. Modulo Xbee S1 [23]

Estas tres tipos diferente de placa trabajaban en conjunto, para conseguir un transporte de datos desde la placa esclavo hasta la placa maestro vía Wireless. Una vez los datos llegan a la placa maestro quien los guarda en un pen drive para la posterior recogida, transporte y estudio a través de un usuario.

2.2. Objetivo y alcance del proyecto

Como se recoge en la motivación, el alcance final del presente proyecto, como fue el de su predecesor, recae en la conservación del patrimonio cultural. Esta tarea será llevada a cabo indirectamente, ya que el proyecto no consiste en la manutención de las obras en sí, sino en la monitorización y toma de datos. De esta manera, mediantes los datos de temperatura y humedad obtenidos, se podrán tomar medidas paliativas para realentizar el deterioro de una determinada obra, escultura o edificio cultural.

Para la adquisición y posterior monitorización de datos, se necesita un datalogger. Actualmente ya se disponen de dataloggers monitorizando ciertas zonas culturales como la catedral de Valencia. Por ello el objetivo principal de este proyecto consistirá en diseñar un prototipo de datalogger capaz de mejorar las prestaciones y posibilidades que ofrecía el anterior.

El anterior datalogger, era capaz de monitorizar superficies alrededor de 1000 metros mediante cableado, por medio de la lectura de sensores mediante protocolo 1-wire. Sin embargo, solo era capaz de realizar una comunicación uno a uno, es decir, el maestro solo podía comunicarse con un único esclavo.

Observando las limitaciones ofrecidas tanto por el cableado como la comunicación uno a uno a la hora de abarcar terreno, estética y comodidad. Uno de los principales objetivos del proyecto consistirá en la creación de una Red Mallada o Red Mesh. De esta forma se podrán abarcar grandes superficies sin necesidad de gran cantidad de cableado.

Una Red Mallada o Mesh, consiste en una estructura inalámbrica, formada por varios módulos o nodos capaces de comunicarse unos con otros individualmente, o con todos a la vez de manera grupal, siempre y cuando ambos módulos se encuentren conectados a la misma red inalámbrica.

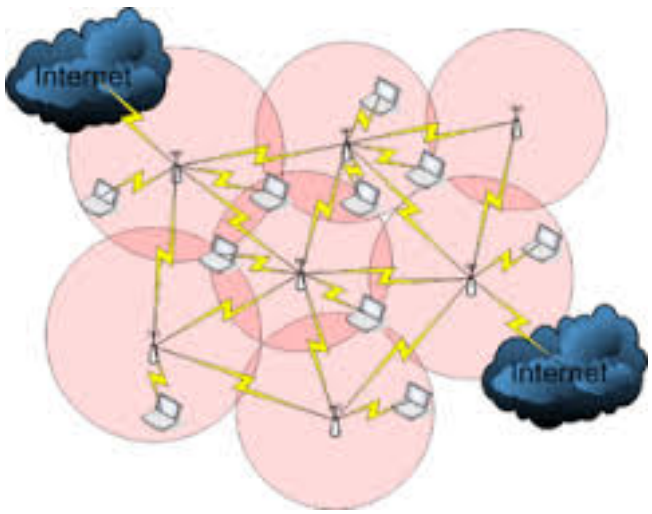


Ilustración 5. Red Mesh general [2]

Por lo general, existen tres tipos de modelo diferentes equipados todos con sensores. Un módulo maestro, encargado de dirigir y coordinar la red a la vez que almacena datos; varios módulos esclavo encargados de la toma de la toma de datos; y por ultimo varios módulos router, los cuales realizaran una función de puente para transmisión de datos entre distintos esclavos y el maestro, de esta forma será posible ampliar el área monitorizada incluso a kilómetros.

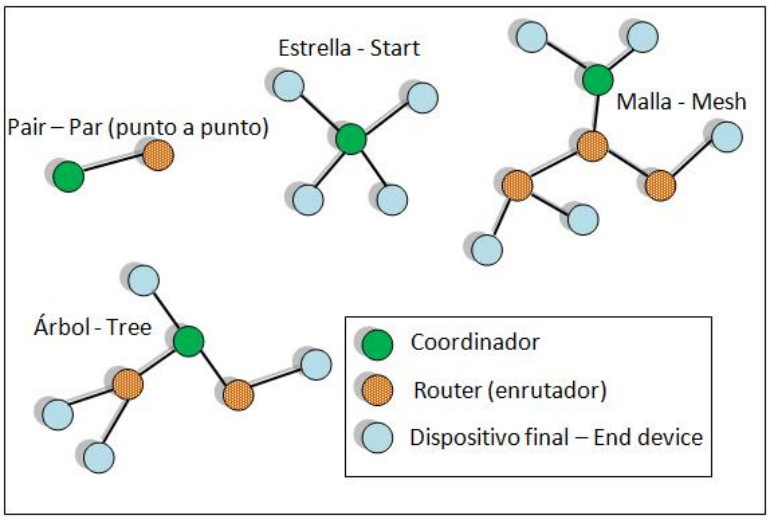


Ilustración 6. Tipos de redes y componentes

En esta ilustración, se puede observar varios tipos de redes, centrándose en la ofertada por este proyecto (Mesh) el coordinador correspondería a nuestra placa maestro, el router a la router y por último el dispositivo final o end device al módulo esclavo.

Para validar el presente prototipo de datalogger, siendo **objetivo** de nuestro proyecto será implementar la siguiente red mallada o Mesh:

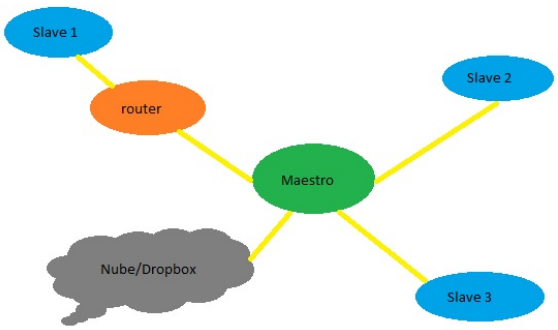


Ilustración 7. Red Mesh a diseñar

Otro de los problemas que presenta el datalogger predecesor, es la necesidad de que un usuario viaje a la ubicación geográfica donde se encuentre la obra en cuestión a extraer los datos. Esto un impacto económico y temporal debido al coste y tiempo destinado a la recogida y posterior procesado de datos.

El segundo objetivo del proyecto, consiste en solventar este problema. Para ello, se dotará a la placa maestro de conexión Wifi o un puerto Ethernet. Dicha aportación permitirá conectar la placa maestro con la nube. De esta manera siempre que el dispositivo disponga de un dispositivo con acceso a internet, se obtendrán unos registros de temperatura de nuestra obra en tiempo real.

El tercer objetivo, consiste en la realización de un mantenimiento adecuado de la red mesh, dotando al dispositivo con la capacidad de advertir vía twitter o Gmail (según decida el usuario) si un sensor proporciona un dato peligroso. En el caso de nuestro prototipo, si la temperatura medida por el sensor supera los 100 grados Celsius. Esto se podrá hacer gracias al objetivo primero, conectar nuestro modulo maestro con la nube.

Por último, el cuarto objetivo recae en la seguridad e integridad de los datos. Dado que el maestro necesitaría conexión Wifi para poder subir los datos a la nube, cabe la posibilidad de que esa conexión se desconecte, reinicie o que se averíe temporalmente. Por ello se dotará a la placa de un dispositivo de memoria externa o secundario, donde guardara los datos igual que lo hacia el datalogger anterior.

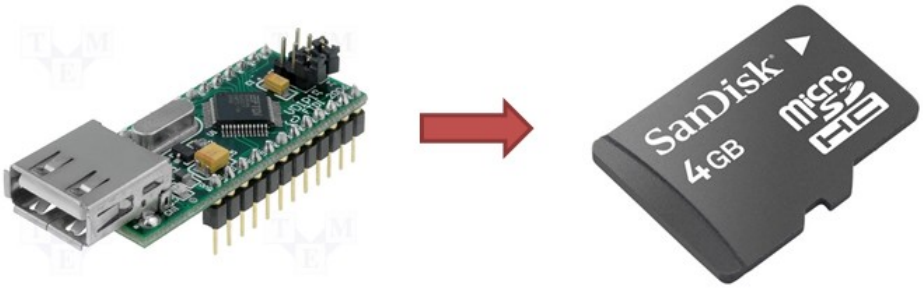


Ilustración 8. VDIP-External USB + Micro SD

A diferencia del Datalogger de Borja Esteban, que contaba con un puerto USB, a través del cual guardaba los datos; el nuevo datalogger, almacenará los datos en un micro SD por motivos comentados en el apartado "*4.1 Estudio y selección de componentes*".

2.3. Objeto de proyecto

Una vez definido el objetivo dedicado a la toma y adquisición de datos. El objeto de proyecto consistirá en la programación mediante software Arduino del presente datalogger.

La plataforma Arduino proporciona una amplia gama de placas que pueden emplearse como maestro, router y esclavo en nuestro prototipo. Por lo que el objeto de proyecto no consistirá en el diseño de placas, sino en su programación, e instalación del hardware adecuado para que el software pueda desempeñar su función.

Entre los componentes ofertados por la plataforma Arduino, se seleccionará como placa maestro el módulo Arduino YUN, mientras que para la tarea de router y esclavo se empleará y programarán módulos Arduino Nano IO Shield. La explicación de todos estos componentes, así como las posibilidades que ofrecen vienen explicadas en la web oficial de Arduino [3]. Por otra parte en el apartado "4. Componentes" se hará explicación sobre los integrantes utilizados en el proyecto, así como del hardware instalado.

En la ilustración 10, puede apreciarse un pequeño esquema del objeto del proyecto, observando tanto los módulos principales Arduino, como el hardware a instalar.

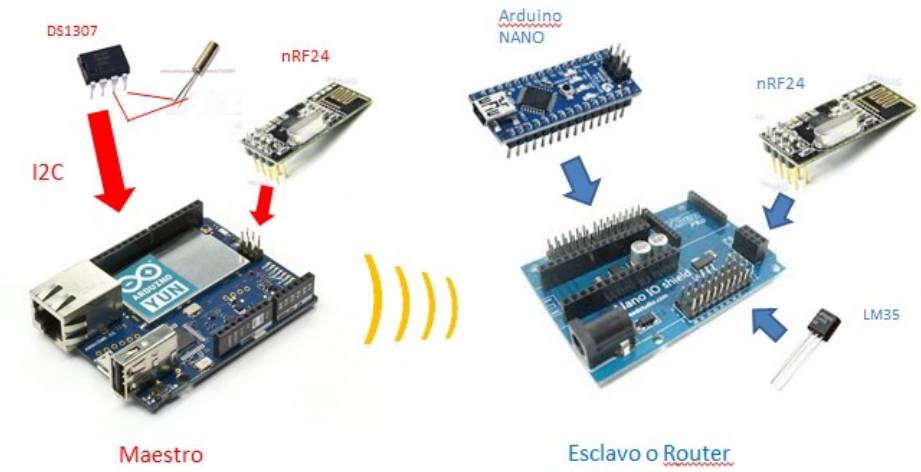


Ilustración 9. Objeto de proyecto

Finalmente, conviene señalar, que al ser un prototipo, únicamente se utilizarán sensores LM35 para la temperatura, por lo que solo se realizará el calibrado de estos. Sin embargo podría emplearse cualquier tipo de sensor o sensores, como podrían ser de humedad, temperatura, iluminación, movimiento… utilizándose el calibrado de fábrica.

2.4. Trabajo previo

2.4.1. Introducción a Arduino

Dado que toda la programación del proyecto va a ser llevada a cabo mediante software Arduino, la primera tarea previa a la realización del proyecto, será informarse acerca de que es y cómo funciona Arduino.

Según la plataforma Arduino [4]: "*Arduino is an open-source prototyping platform based on easy-to-use hardware and software. Arduino boards are able to read inputs and turn it into an output*"

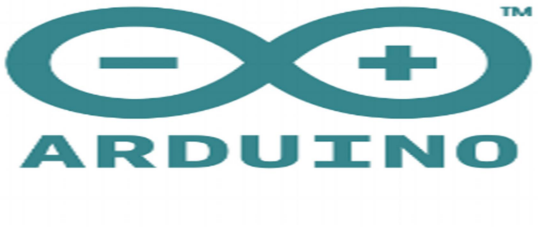


Ilustración 10. Logo de Arduino [3]

Como se observa, Arduino es una plataforma dedicada al tratamiento de señales de entrada y salida, tanto analógicas, como digitales mediante lenguaje C. Dadas estas condiciones, su software libre y gracias al hardware que ofrece, Arduino es una plataforma que se adapta perfectamente a la implementación de dataloggers y por tanto a este proyecto, ya que gracias a las entradas analógicas dispuestas en cualquier módulo Arduino, se podrá tomar datos de casi cualquier sensor.

Arduino dispone puertos Rx y Tx en sus pines 0 y 1 para una comunicación UART con el ordenador. De esta manera si lo deseamos, se puede observar cómo se ejecutan o donde se guardan las variables. Además dispone de una gran variedad de librerías, como 'Wire.h' para las cuales proporciona manuales de uso bien detallados en su web, además de varios ejemplos sencillos y básicos para su uso. Para averiguar más acerca de estas librerías se puede entrar en la página web de Arduino apartado librerías [5].

Continuando con el tema librerías, Arduino, también permite añadir librerías creadas por el usuario, o simplemente descargadas de una web como pueda ser Github [6]. Por ello fuera de la memoria, dentro del "Manual del programador" se encontrará una descripción de las librerías empleadas para este proyecto, ya sean propias de Arduino, descargadas o creadas por el usuario.

2.4.2. Tarea previa al proyecto

Como tarea previa y a modo de aprendizaje del uso de Arduino, sus componentes y el calibrado de sensores; se deberá recuperar el correcto funcionamiento del datalogger predecesor comentado en el punto "*2.1. PFC Borja Esteban Sanchis*".

Este datalogger ha pasado de funcionar en perfectas condiciones, a presentar errores de funcionamiento tras un largo periodo de funcionamiento. Estos errores consisten en que a la hora de guardar los datos en la USB por medio del VDIP1, proporciona líneas en blanco y líneas de ceros como se puede observar en la ilustración 11.

```
11;34;23;11;2015;  00.00;-26.70;  24.18; 09.80;
11;35;23;11;2015;  00.00;-26.70;  24.18; 09.80;
11;36;23;11;2015;  24.06;-26.70;  24.00; 09.80;

11;44;23;11;2015;  23.43;-26.65;  23.43; 11.20;
11;45;23;11;2015;  23.56;-26.70;  23.56; 10.71;
11;46;23;11;2015;  00.00;-26.70;  23.56; 10.71;
```

Ilustración 11. Datos anómalos

Tras una revisión del código, el cual no presenta nada anómalo; y debido a que el mal funcionamiento se ha ocasionado por el tiempo y no por el código. Realizamos la hipótesis de que el fallo es producido debido a deterioro del hardware. Por ello el procedimiento de revisión será el siguiente:

1- Mediante el multímetro en modo diodo se revisarán todas las conexiones de cada placa, de esta forma se descartarán desconexiones puntuales. No se detecta ningún fallo.
2- Siguiendo con el multímetro, pero esta vez en modo voltímetro, se comprobará que al VDIP1 le llegan los 5V necesarios para su correcto funcionamiento. Efectivamente la tensión proporcionada al VDIP1 es correcta.
3- Comprobar un posible fallo en cualquiera de los microcontroladores ATMEGA 328. Se sustituyen todos, y siguen proporcionando los mismo datos anómalos.
4- Dado que probablemente el problema nazca a la hora de almacenar datos se revisa el módulo VDIP1.
   Tras la revisión del VDIP1 mediante su programación extensiva (ASCII) y un uso de líneas de código básicas basadas en la creación de ficheros, escritura y cierre, se deduce que el VDIP1 empleado tiene deterioro, ya que no responde correctamente a algunas funciones básicas.

Sabiendo esto, sustituimos el VDIP1 por uno nuevo programado en hexadecimal (modo de programación corta). Una vez realizamos este cambio, observamos que ya no genera líneas en blanco, pero sin embargo sigue proporcionando ceros (ilustración 12)

```
12;46;23;11;2015;  00.00;-26.64;  24.37; 10.41;
12;47;23;11;2015;  24.37;-26.69;  24.31; 10.39;
12;48;23;11;2015;  00.00;-26.69;  24.31; 10.39;
12;49;23;11;2015;  00.00;-26.69;  24.31; 10.39;
12;50;23;11;2015;  24.31;-26.69;  24.31; 10.42;
12;51;23;11;2015;  00.00;-26.69;  24.31; 10.42;
```

Ilustración 12. Datos anómalos 2

5- Dado que a pesar de haber solucionado las líneas en blanco, ninguno de estos procedimientos ha solucionado el problema de los ceros, siendo un único dato de temperatura (TEM01 – temperatura sensor 1) el que falla, se supondrá que el fallo debe estar en el sensor 1 empleado en la toma de datos 1-Wire.
   Según esta hipótesis observamos que efectivamente el sensor 1 parece dañado, como puede observar en la ilustración 13.

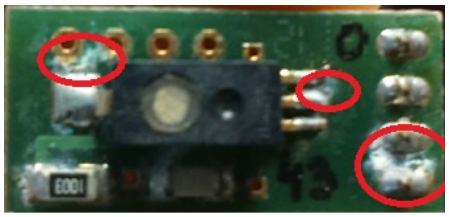


Ilustración 13. Sensor averiado

   Por ellos, se procederá a remplazarlo y calibrar el nuevo sensor. Tras esta labor, observamos que efectivamente ya todo vuelve a funcionar con normalidad (ilustración 14)

```
12;47;24;11;2015;  24.56;-26.81;  24.43; 09.33;
12;48;24;11;2015;  24.56;-26.70;  24.43; 09.10;
12;49;24;11;2015;  24.56;-26.77;  24.50; 09.21;
12;50;24;11;2015;  24.56;-26.70;  24.43; 09.27;
12;51;24;11;2015;  24.56;-26.77;  24.43; 09.08;
```

Ilustración 14. Datos correctos

Por último, y como innovación, ya que los VDIP1 no están preparados para almacenar datos de forma muy prolongada, se decide resetear el buffer del VDIP1, mediante hardware cada hora de funcionamiento para evitar posibles saturaciones del buffer de memoria. Para ello se añadirá un transistor Mosfet IRF-510, cambiando la tensión de la puerta entre 5V y 0V. De esta forma cuando recibe 0V, el canal drenador-surtidor se bloquea cortando la corriente al pin de reseteo el cual realizara su función, sin necesidad de software.

3. Ubicación y emplazamiento

3.1. Lugar de desarrollo

La investigación del proyecto comienza y termina en el Departamento de Física Aplicada (U.D. Industriales) Universidad Politécnica de Valencia, España; en el laboratorio de investigación número 3. Siendo llevada a cabo por Alejandro Esteban Sanchis y dirigida por el profesor Fernando Juan García-Diego, con la colaboración de Borja Esteban Sanchis y Ángel Fernández Navajas. Sin embargo, a mediados del desarrollo, el lugar de investigación, tuvo que ser desplazado a la residencia particular de Alejandro Esteban Sanchis.

Este desplazamiento fue ocasionado por la seguridad implantada en el servicio de red ofrecido por la Universidad Politécnica de Valencia (UPV). La UPV es un edificio público de enseñanza e investigación, con diversidad de usuarios conectados a su red, por ello, su red wifi dispone de una seguridad WPA/WPA2 Enterprise y PEAP con cifrado AES (Advanced Encryption Standard), la cual solicita: nombre de red, usuario y contraseña. Esta seguridad resulta infranqueable para el Arduino YUN, imposibilitando su conexión a la red. Por el consiguiente estado de incomunicación con la nube, el modulo maestro estará inhabilitado para subir los datos a Dropbox, por lo que no se podra comprobar si el prototipo es capaz de realizar su función.

En una residencia particular, por lo general, el servicio de protección otorgado suele ser de tipo WPA-PSK con cifrado AES, esta seguridad solicita nombre de red y contraseña. El módulo YUN sí es capaz de conectarse a este tipo de red, por lo que el prototipo fue desplazado a la residencia de Alejandro Esteban Sanchis hasta obtener una funcionalidad estable y adecuada.

3.2. Futuras instalaciones

Al igual que en un residencia, la red de nuestra futura instalación, solo solicita nombre de red y clave de red. Por lo que la conexión a red no supondrá ningún problema. De esta forma los datos llegaran perfectamente a Dropbox, y una vez allí, podrán ser monitorizados o manipulados desde cualquier usuario al que se le permita el acceso a dicha carpeta Dropbox.

Como se ha dicho anteriormente, la motivación de este proyecto viene dada por la conservación del patrimonio cultural. Por lo que su instalación va destinada a cualquier obra o edificio correspondiente al patrimonio cultural.

Un ejemplo para la futura instalación de este datalogger, puede ser la catedral de Santa María de Valencia, donde se puede encontrar ya instalado un datalogger anterior al de Borja Esteban Sanchis. Pero esta tarea será llevada más adelante tras una previa mejora del datalogger ofertada en el apartado 6 de la memoria "*6. Futuras mejoras*".

4. Componentes

4.1. Estudio y selección de componentes

Teniendo en cuenta los objetivos de nuestro proyecto, antes de comenzar con su diseño se debe seleccionar los componentes más adecuados para el propósito, es decir, se necesitará responder a las siguientes preguntas:

- **¿Por qué los módulos Arduino YUN y Arduino Nano (IO shield)?**

Para cumplir los objetivos de nuestro proyecto, necesitamos una placa maestro capaz de concitarse vía Wifi o Ethernet con la nube. Además deberá ser capaz de albergar dispositivos de memoria externa.

Estas características son cumplidas tanto por una raspberry, tanto por el módulo Arduino YUN, sería más eficaz utilizar la raspberry como maestro, pero para ello, serían necesarios lenguajes como Java o Python (desconocidos por mí en este momento) y podría ocasionar posibles problemas de compatibilidad con las placas sensor y router, ya que vendrían programadas en software Arduino (lenguaje de programación C#), por lo que tal vez implicaría rediseñar las placas esclavo y router. Ademas los modulos Arduino, como su propio lema dice, están específicamente diseñados para la adquisición de datos, lo cual en este campo supone una ventaja frente a la Raspberry.

Por ello, para realizar el prototipo de nuestro proyecto, se seleccionará como placa maestro el módulo Arduino YUN (características en el apartado *"4.3.1. Arduino YUN"*), sin embargo, no queda descartada de mejorar más adelante el prototipo, convirtiendo el maestro en una raspberry, ya que esta permitiría mayor maniobrabilidad de nuestros objetivos, y también estaría mejor visto por el mercado. Esta sustitución se comentara mejor más adelante en el apartado *"6. Futuras mejoras"*.

Por otro lado, para las placas sensor, necesitamos que sean capaces de tomar datos analógicos, esto se puede encontrar en infinidad de modelos, pero por seguir en la línea del uso de Arduino, y ya que en gran medida la plataforma está diseñada para el tratamiento de señales. Se seleccionará, de entre todos los ofertado por Arduino, el módulo Arduino NANO, ya que tiene un coste muy inferior al resto de módulos Arduino.

También, este modelo, puede ser acoplado a una placa IO Shield (de muy bajo coste también), la cual permitirá acoplarle el hardware necesario para la comunicación inalámbrica sin ningún problema.

Se definirá con más detalle, las características de esta placa en el apartado "4.3.2. Arduino NANO".

La selección de estas dos placas, también viene dada por la comodidad que proporciona Arduino, al tener las placas ya prediseñadas. Lo cual reduce el tiempo necesario para llevar a cabo el prototipo considerablemente, facilitando cumplir los plazos de entrega del proyecto.

- **¿Cómo conseguir la obtención de datos a tiempo real?**

Para conseguir que la obtención y almacenamiento de datos sea a tiempo real, se necesitará que la placa maestro cuente con un reloj. Para ello se tendrán dos opciones: Instalar un dispositivo DS1307 o que el Arduino YUN solicite la hora a la nube.

Al igual que los motivos por el que se decide instalar un dispositivo de memoria externo (posibles fallos de red), la opción de solicitar hora y fecha a la nube queda descartada. Por lo que se instalará el reloj DS1307, comunicandolo con el YUN mediante I_2C gracias a la librería 'Wire.h'.

El funcionamiento del DS1307 se explicara con más detalle en el apartado "4.3.4 DS1307 ".

- **¿Qué dispositivo Hardware se utilizará para llevar a cabo la comunicación inalámbrica?**

Para llevar a cabo otro de los objetivos de nuestro proyecto, se necesitará un dispositivo capaz de comunicar nuestros módulos maestros, esclavo y router formando la red Mesh.

Para ello la placa IO Shield junto al módulo Arduino NANO, cuenta con las ranuras de pines, tanto para conectar un dispositivo Xbee o un dispositivo NRF24L01. Por lo que la selección del hardware para crear la red Mesh se debatirá entre estos dos componentes.

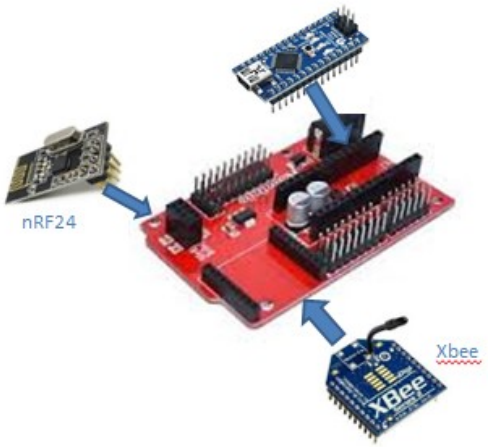


Ilustración 15. IO Shield; NRF24L01 y Xbee

El datalogger predecesor (El perteneciente a Borja Esteban Sanchis) realizaba una comunicación punto a punto gracias a un dispositivo llamado Xbee Serie 1 (XbeeS1), este dispositivo se puede ver en la ilustración 4.

El problema de este dispositivo, reside en que solamente permite crear una comunicación punto a punto, o multipunto, es decir, no es válido para este proyecto.

Como solución, los Xbee cuentan con una versión más avanzada llamada Serie 2 (XbeeS2) mostrado en la ilustración 16. Este dispositivo es capaz de formar una red mesh a costa de un nivel de programación más complicado colocando el XbeeS2 en modo API (application programming interface).

Ilustración 16. Xbee S2 [7]

La programación de los Xbee S2 se lleva a cabo en dos partes. Primero mediante el software de programación X-CTU, el cual permite modificar el modo de funcionamiento del Xbee S2, de esta forma se podrá programar en modo API, capacitándolo para llevar a cabo la red mesh; en el X-CTU también se indicaran los canales de comunicación por el que lee y escribe el Xbee S2. La segunda parte consiste en la implementación de código mediante software Arduino.

Por otra parte el NRF24L01, presenta una programación muy sencilla, mediante el uso de librerías en software Arduino. En la siguiente tabla, se podrá comparar las posibilidades que ofrecen los distintos dispositivos, para su posterior selección.

Tabla 1. NRF24L01 vs Xbee

	NRF24L01	Xbee S2	Xbee S1
Precio por unidad (euros)	(14.32/10) = 1.44	31.22	29.95
Alcance comunicación (m)	20 - 80	500 - 1000	500 - 1000
Consumo	Bajo	Bajo	Bajo
Complejidad	Baja	Alta	Media
Velocidad (Kbps)	250 - 2048	1024	256

De acuerdo, a estos datos y haciendo un análisis mediante *matriz comparativa* reflejado en "*Anexos punto 2. Estudio comparativo nRF24L01 vs Xbee*", llegamos a la conclusión de que es mucho mejor el uso del NRF24L01 para el diseño de nuestra red mesh.

El modo en que se comunica el NRF24L01 y sus protocolos de comunicación, serán explicados más adelante en el apartado *"4.3.3. NRF24L01".*

- **¿Qué dispositivo de memoria externa se usará?**

Como mencionamos en el punto *"2.2. Objetivo y alcance del proyecto"* por motivos de seguridad a la hora de almacenar datos, se dispondrá de un dispositivo de memoria externa. La placa maestro de nuestro prototipo, corresponde al Arduino YUN, este módulo proporciona dos alternativas para el almacenamiento externo. Primero un puerto USB, donde podría conectarse un pen drive y seguir con la filosofía del VDIP1; y en segundo lugar, una ranura micro SD; estas alternativas quedan visibles en la ilustración 17.

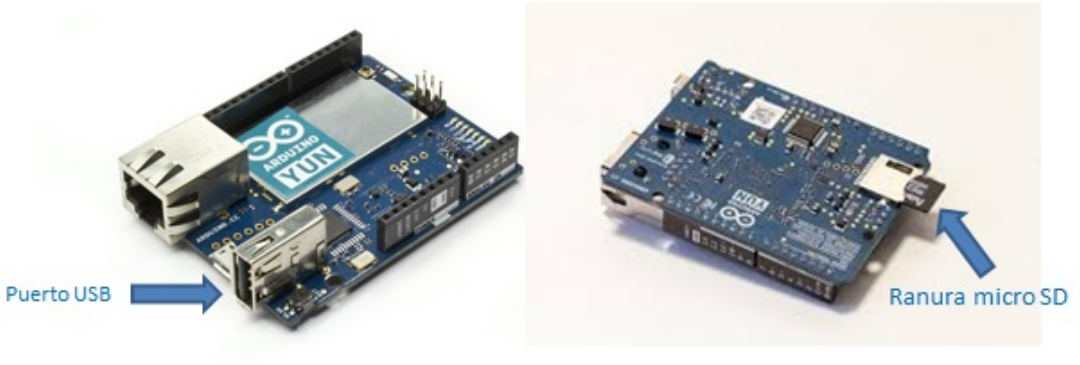


Ilustración 17. Almacenamiento externo Arduino YUN

A la hora de seleccionar uno, el pen drive, USB o Flash Drive tiene mejores características, ya que proporciona una mayor capacidad de almacenamiento, mayor velocidad de transmisión de datos así como mayor fiabilidad a la hora de su almacenamiento. Por otro lado, a la hora de haber trabajado con VDIP1, los colaboradores del proyecto no están muy contentos con su funcionamiento en dataloggers previos, ya que ha ocasionado problemas reiteradas veces al trabajar a altas horas de funcionamiento.

Teniendo en cuenta estos factores, aunque las características del pen drive sean mejores, se seleccionará para nuestro proyecto una memoria micro SD como dispositivo de almacenamiento externo. Como resultado, estamos muy contentos con los resultados que ofrece, ya que no ha ocasionado ningún problema en su instalación ni uso. Además tiene procedimientos de instalación y comunicación UART muy sencillos de incorporar en el programa. Estos procedimientos serán vistos más adelante en *"Anexos. 4.2. Guardar en la SD"* y en el *"Manual del programador y usuario".*

4.2. Plataforma Temboo

Como su lema dice: "*Your loT Software Stack. Build, drow, and extend your products for the internet of things with less fricction and more flexibility*" Tembo [8] podría ser considerada como una plataforma puente, ya que su función principal, es comunicar entre si una gran diversidad de plataformas web tanto para almacenamiento de datos, compra venta e incluso redes sociales; con la mayor flexibilidad posible. Entre las más conocidas se encuentran: Facebook, Twitter, Amazon, Google, Instagram y muchas más.

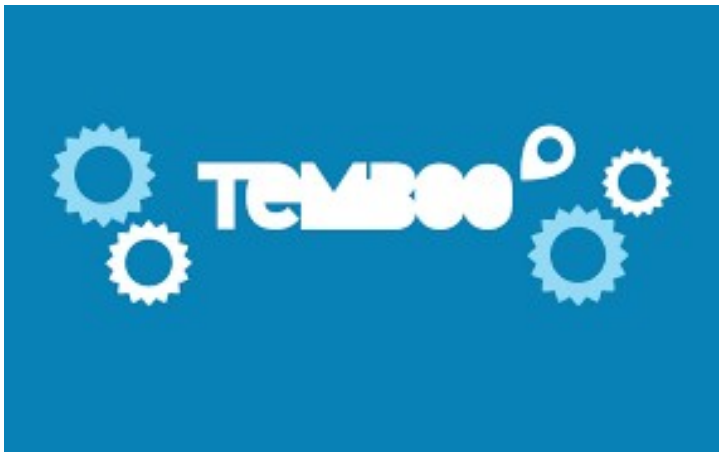


Ilustración 18. Icono Temboo [8]

La comunicación con estas plataformas, puede llevarse a cabo a través de las librerías que ofrece en [9], en el caso de nuestro proyecto se utilizarán las librerías para la comunicación con Dropbox, twitter y Google (Gmail).

Para llevar a cabo esta comunicación, no solo se necesitara de las librerías, sino que será necesario cumplir las condiciones que estas solicitan:

- En primer lugar, para llevar a cabo esta comunicación, las librerías necesitan una aplicación (App) puente diferente para comunicarse con cada plataforma web y en Temboo, por lo que para comunicar con Dropbox se necesitará crear una aplicación en Dropbox; para comunicar con Twitter otra en Twitter; y así con cada plataforma web con la que quiera comunicarse.

- En segundo lugar y para mayor seguridad, Temboo proporciona credenciales o contraseñas para llevar a cabo esta comunicación. De esta manera nadie ajeno a la nuestra cuenta Temboo podrá subir información a nuestro entorno personal.

A continuación se explica cómo conseguir las distintas Apps, así como las credenciales de Temboo [1]Este paso debe realizarse desde un computador a través de la web oficial de Temboo.

Creación App en Dropbox:

Para crear una App en Dropbox, debes seguir los siguientes pasos:

1- Debes tener o crearte una cuenta Dropbox. A continuación, debes ir a Dropbox/developers/apps y clicar en créate App.
2- Puedes crearte dos tipos de App. Una para acceder a todo Dropbox, y otra para solo tus ficheros. Se cogerá esta segunda llamada 'AppFolder', ya que solo interesa subir un fichero de datos a nuestra cuenta.
3-

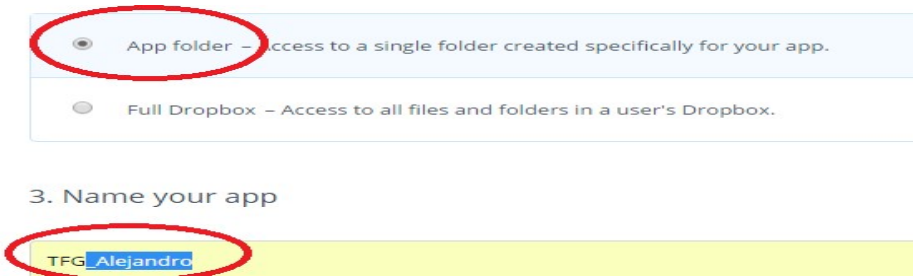


Ilustración 19. App Folder

4- Seguidamente se debe insertar el nombre que se quiere para la App Dropbox, en este caso 'TFG_Alejandro'.
5- Para terminar aceptamos, y Dropbox mostrará las características de nuestra App (Ilustración 20), entre estas características se encontrarán las credenciales 'AppKey' y 'SecretKey' necesarias para coordinar la App de Dropbox con la plataforma Temboo.

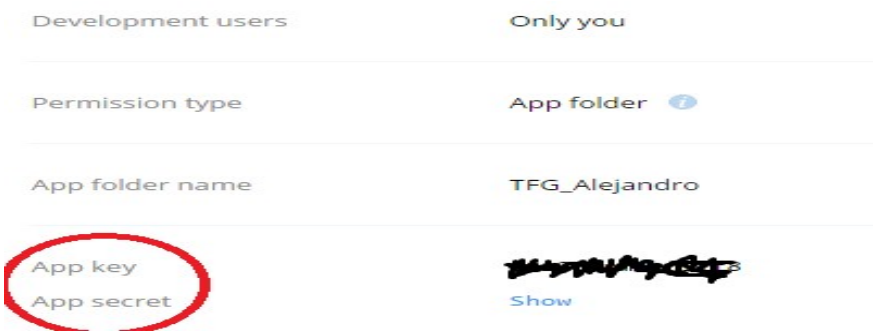


Ilustración 20. Keys

[1] Es recomendable apuntarse o guardar las credenciales proporcionadas por las Apps, tanto por la de Temboo como las de las App asociadas.

Creación App en Twitter:

En cuanto a la creación de la App Twitter, es muy parecida a la de Dropbox. Habrá que seguir los siguientes pasos:

1- Debes tener una cuenta Twitter, en este caso, 'alessantfg'. Seguidamente, debes entrar en https://apps.twitter.com/app/new y complementar los datos de la siguiente manera (Ilustración21):

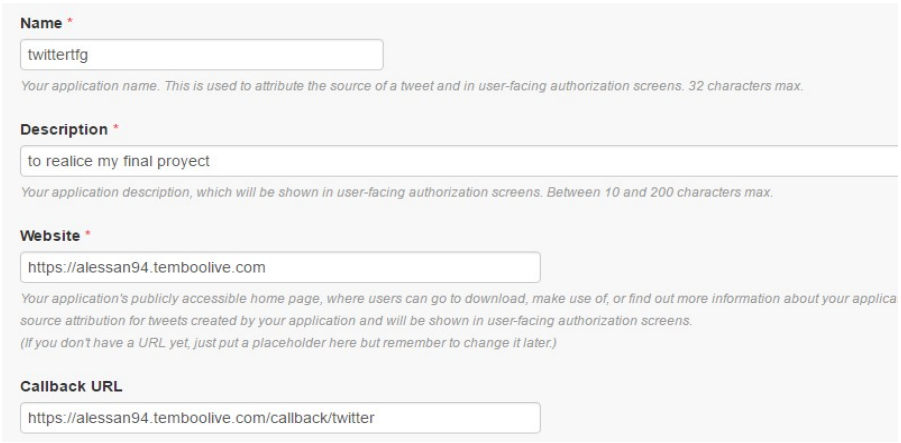


Ilustración 21. Creacion App Twitter

2- Al rellenar estos datos y aceptar los términos, se te creara la App Twitter con las respetivas credenciales 'Key' y 'SecretKey'. Ademas podras acceder a tu App twitter de una forma mucho más directa desde tu cuenta Twitter quedando de la siguiente forma (Ilustración 22):

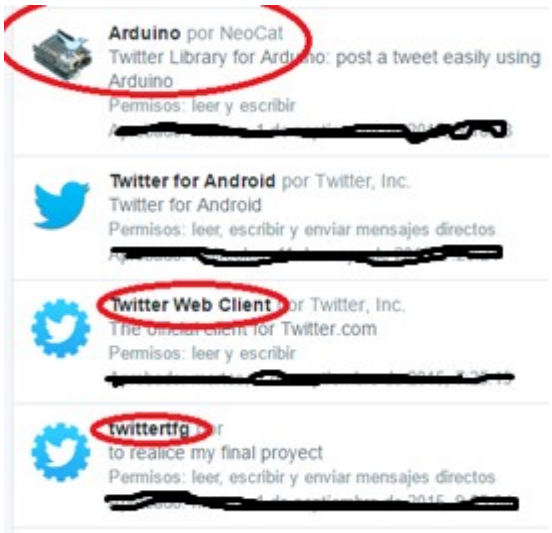


Ilustración 22. Apps Twitter

Post Data: No olvides guardar o anotar todas tus credenciales en algún lugar seguro, luego las necesitaras.

Creación App en Gmail o Google:

Se vinculará también Gmail con Temboo, para dar la opción al usuario, de si en caso de emergencia, quiere ser avisado vía twitter o Gmail. Para lograr esta vinculación, se necesita crear también una App en Google. Esta *App* se creara de forma muy parecida a las anteriores. Siguiendo los siguientes pasos:

1- Debes tener una cuenta Gmail en google, por ejemplo "a*lejandro.tfg@gmail.com*". Seguidamente para mayor seguridad, se le dotará con acceso de verificación a dos pasos (confirmación desde ordenador y móvil).

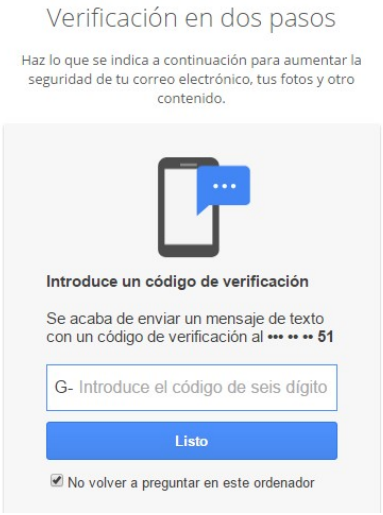


Ilustración 23. Two-Steps security

2- Una vez creada la cuenta y su seguridad, se procederá a la creación de la App. Para ello, debes ir a: inicio y seguridad de la cuenta/aplicaciones y sitios conectados a tu cuenta. Aquí podrás ver todas las Apps asociadas a google, como puede ser tu WhatsApp y podrás asociar o crear nuevas como se haría con nuestro Gmail y Temboo.

3- Por ultimo para crear la App y asociar nuestro Gmail con Temboo, dentro de este último apartado, se deberá clicar en contraseñas de aplicación, y generar la App de la siguiente forma, para asociarla correctamente con Temboo (ilustración 24).

Ilustración 24. Generar App Google

4- Una vez hecho esto, se proporcionara un URL de confirmación, al confirmarlo, ya se puede trabajar con Temboo y Gmail.

Credenciales Temboo:

1- Las 2 primeras credenciales ('AppKey' y 'AppSecret'), consistirán en las proporcionadas por tu App Dropbox, Twitter o Gmail. Para conseguir las otras 2 credenciales, lo primero que debes hacer es crearte cuenta Temboo. Automáticamente se te creara una App en Temboo con el nombre de cuenta y una clave de acceso; la cual, junto a las 2 credenciales ya obtenidas, te permite usar unas funciones específicas de la librería Temboo (OAuth) que te proporcionaran las correspondientes credenciales.
En mi caso la cuenta se llama alessan94, y la aplicación myFirstApp (Ilustración 25)

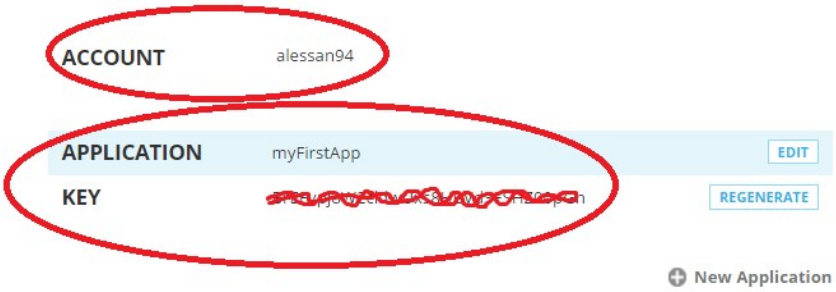


Ilustración 25. App Temboo

2- A continuación, debe usar estas librerías específicas OAuth en el orden apropiado, para así obtener las 2 credenciales necesarias para el resto de funciones de la librería Temboo (Ilustración 26).

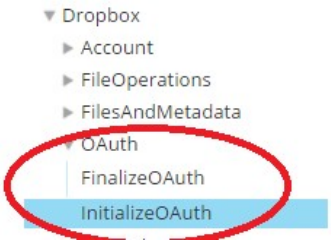


Ilustración 26.OAuth

3- Debes ejecutar la función InitializeOAuth, esta aplicación pide de entrada, las credenciales ('AppKey' y 'AppSecret') de tu App Dropbox, Twitter o Google; esta función te proporcionara como salida una autorizacionURL, una ID de llamada, y un OAuthTokenSecret necesario para ejecutar la siguiente función OAuth, así que no olvides guardar todos estos datos.

4- Entra en el URL ofertado (autorización URL) y confirma la autorización.

5- Por último, debes ejecutar la otra función OAuth, FinalizeOAuth. Esta pedirá como entrada la ID y el OAuthTokenSecret proporcionados por InicializeOAuth, asi como las credenciales de tu App Dropbox, Twitter o Gmail; seguidamente proporcionara como salida las 2 últimas credenciales ('AccesToken' y 'AccesSecretToken') necesarias para usar el resto de funciones Temboo y una ID de usuario.

Una vez obtenido todo esto, se podrán utilizar todas las funciones necesarias de nuestra plataforma Temboo. Se explicará cómo permitir el acceso al Arduino YUN a esta plataforma en el apartado "manual del programador y usuario" incluyendo en sus líneas de código las credenciales de nuestra App Temboo.

Como ha quedado implícito, se utilizará la plataforma temboo como plataforma puente para subir nuestros datos a la nube. En nuestro caso se comunicará nuestro Arduino YUN vía Temboo con Dropbox y twitter o Gmail, según elija el usuario, como puede observarse en la ilustración 27.

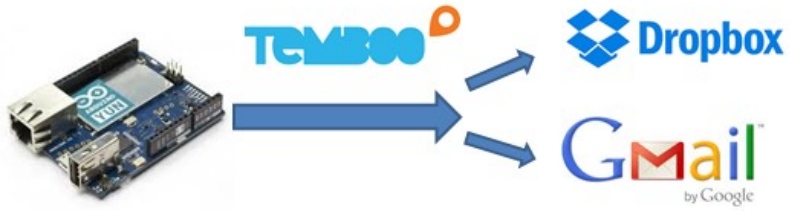


Ilustración 27. Comunicación plataformas

Se utilizará Dropbox para el almacenamiento de datos en nuestro ordenador, y twitter o Gmail a modo de seguridad del equipo, es decir, en caso de detectar una temperatura excesivamente alta, alertara al usuario con un mensaje directo o un e-mail.

Respecto a la plataforma temboo, tiene versión gratuita y de pago. Como este proyecto trata únicamente de diseñar un prototipo, se utilizará la versión gratuita, la cual restringe la capacidad de envíos a 10000 mensuales y 1Gigabyte (Gb).

Ecuación 1. Numero de envíos por mes

$$\frac{Envios}{mes} = 30dias * 24horas * \left(\frac{60}{Pm}\right) + Alertas\ Seguridad$$

- *P_m(Periodo de muestreo) = 1 minuto;* [2]*Alertas Seguridad = 0*

$$30 * 24 * 60 + 0 = 43200 > 10000$$

Como se ve en los cálculos (43200>10000) por lo que estos envíos no serían suficientes para un periodo de muestreo de 1 minuto en los datos, sin embargo con la versión de pago no existiría problema alguno con la cantidad de envíos ni el espacio ocupado por estos.

[2] Este envío solo lo realizara si la temperatura excede el límite puesto por algún problema. Debido a que la tarea del dispositivo debería llevarse a cabo sin ningún problema de seguridad el valor de 0.

4.3. Componentes principales

4.3.1. [3]Arduino YUN

A lo largo de toda la memoria, se ha hablado del Arduino YUN, esto es debido a que es el integrante mayoritario en nuestro proyecto; ya que conformara la placa maestro, siendo responsable de coordinar toda la comunicación y transmisión de datos, tanto entre dispositivos (red Mesh) como con la nube.

Arduino YUN [10] consiste en una protoboard o microcontroller board, basada en el microcontrolador ATmega32u4 (encargado del tratamiento de señales) y Atheros 9331(encargado de la comunicación Wifi, Ethernet y puertos externos). El módulo Arduino YUN, en su diseño, ha sido equipado con puerto Ethernet, soporte Wifi, puerto USB, ranura micro SD, 20 pines digitales de los cuales 7 pueden usarse como salidas PWM (Modulación por ancho de pulso) y otros 12 como entradas analógicas de 10bits (2^10 = 1024) valores que se aprovecharán en la conversión digital-analógica o analógica-digital a la hora de calibrar los sensores. Arduino YUN también dispone de un cuarzo o cristal oscilador de 16MHz (No compatible con el dispositivo reloj DS1307), conexión micro USB para ser programado desde el computador, un comunicador ICSP(Programación Serial En Circuito) y 3 botones de reseteo (para Wifi, micro y general). Estos integrantes pueden observarse en la ilustración 28.

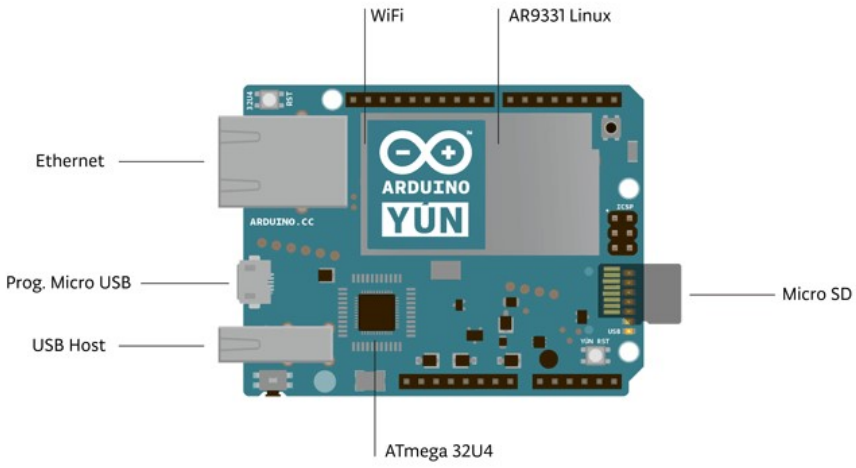


Ilustración 28. Componentes Arduino YUN [10]

[3] Se puede encontrar más información sobre Arduino YUN, buscando su datasheet [10]

En la tabla 2 se muestran las especificaciones técnicas para el correcto funcionamiento del Arduino YUN.

Tabla 2. Especificas técnicas Arduino YUN

Microcontrolador	ATmega32u4
Voltage	5V
Digital I/O pines	20
Canales PWM	7
Pines de entrada analogical	12
DC por pines I/O	40mA
DC por pines	50mA
Memoria flash	32KB
SRAM	2.5KB
EEPROM	1KB
[4]Velocidad del reloj (Clock Speed)	16MHz

Como ya se ha visto Arduino YUN es capaz de comunicarse por el mismo con la nube siempre que disponga de conexión a internet, sin embargo, será necesaria la instalación de cierto Hardware para que pueda cumplir el resto de las funciones propuestas en este proyecto y conectar Arduino YUN a la red disponible:

- Conectar Arduino YUN a la red

Como ya se ha dicho, Arduino YUN tiene la capacidad de conectarse a una red Wifi, en nuestro caso se conectará a una red de residencia particular, pero podría ser conectada a otras, cuya seguridad solo requiera nombre de red y clave de acceso.

Para conectar nuestro módulo Arduino a la red y pueda realizar sus funciones de comunicación con la nube, se deben seguir los siguientes pasos:

1- Conectar el Arduino YUN a una fuente de alimentación.
2- Desde tu computador, busca en el centro de redes. Ahí encontraras una red llamada "Arduino YUN-[5]numero hexadecimal". Conecta tu computador a la red.
3- Una vez conectado, perderás la conexión con internet, solo podrás conectarte al URL correspondiente a la configuración del Arduino YUN. Para encontrar este URL, debes escribir en el buscador "Arduino.local" o el numero [6]IP de tu placa Arduino, accediendo a la web de la ilustración 29.

[4] No confundir el reloj interno de Arduino YUN (con cristal de 16MHz) con el DS1307, el reloj interno del YUN coordina las acciones internas del micro y las salidas PWM, mientras que el DS1307 mide la hora y la fecha para tomar datos a tiempo real.
[5] El numero hexadecimal variara con cada Arduino YUN
[6] Anota este número en la placa, ya que lo necesitaras si quieres recalibrar su configuración.

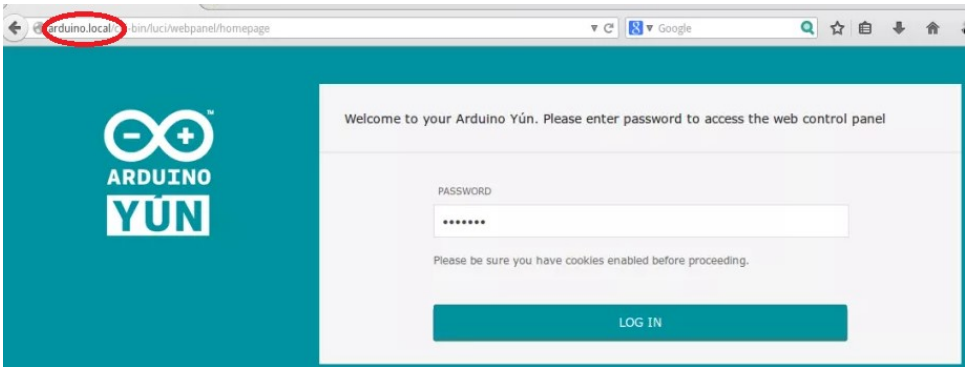


Ilustración 29. Arduino.local

4- La primera vez que configures el Arduino YUN la contraseña será siempre Arduino. Introdúcela y accederás a la ventana de características, clica en configuración y accederás a las opciones de configuración (Ilustración 30):

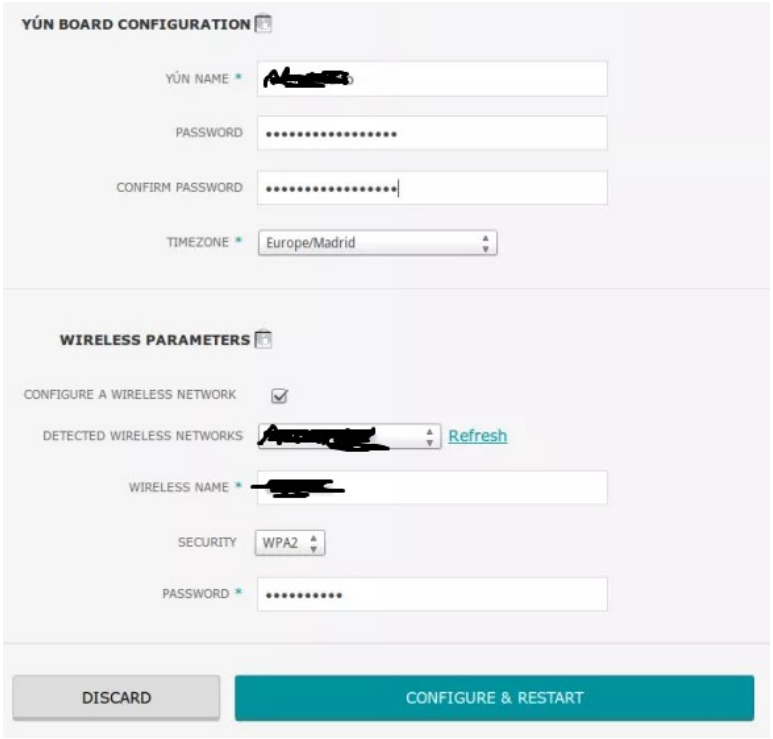


Ilustración 30. Configuracion Arduino YUN

5- Rellena los datos de tu arduinio y de la red Wifi que pretendas asociar, una vez rellenado, clica en “configure & restart”, espera, y tu Arduino YUN ya estará conectado a tu red local, listo para comunicarse con la nube.

- Instalación sensores LM35

Para la toma de datos, se podrá instalar un máximo de 12 sensores en el Arduino YUN (12 entradas analógicas) ya que de momento no está dotado con toma de datos 1-wire. Por ello se instalaran 2 dispositivos LM35 para medir la temperatura. El calibrado de estos sensores se verá en "*Anexos. 3. Calibración LM35*".

Las conexiones para realizar esta instalación serán las siguientes:

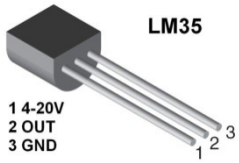


Ilustración 31. LM35 [24]

- Pin 1 a 5V
- Pin 2 conectado a entrada analógica
- Rango = [-55 ; 150] ºC
- Pin 3 conectado a tierra.
- 10mV/ºC

- Instalación DS1307

Para obtener datos en tiempo real, se le instalara el dispositivo DS1307. Esto será gracias a los pines TWI: SDA (pin 2) y SCL (pin 3). Los cuales junto a la librería 'Wire.h', permitirán llevar a cabo una comunicación I_2C con el DS1307, proporcionando la hora y fecha de la toma de datos.

Para llevar a cabo la instalación del DS1307 en el Arduino YUN será necesario conectar respectivamente los pines TWI de cada dispositivo (pines TWI del DS1307 reflejados en ilustración 43 apartado *"4.3.4. DS1307"* tal como se muestra en la ilustración 32.

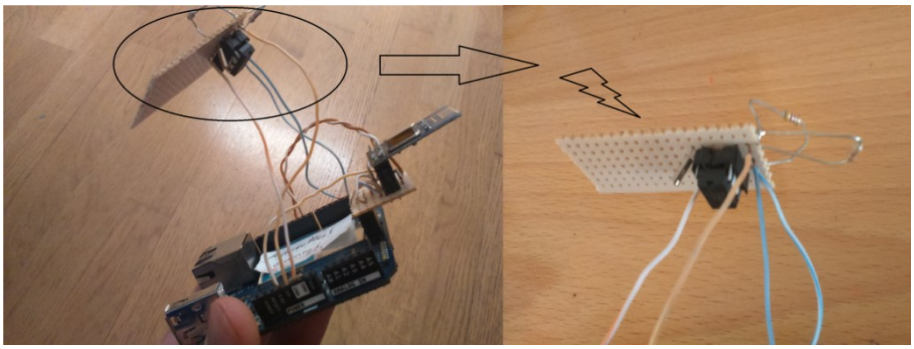


Ilustración 32. Instalacion DS1307

Una vez instalado el reloj DS1307, ya solo habrá que implementar una función en Arduino, que solicitando la hora al DS1307 tome datos cada 'X' periodo de tiempo. A la vez deberá concatenar la fecha leída por el reloj a los datos medidos, para conocer el día y hora en que se midió dicho dato, tomando así datos a tiempo real.

- Instalación SD

Tanto la SD y el USB, serian componentes externos al controlador de Arduino, por lo que sería necesario comunicarlos mediante la librería puente 'bridge.h'. Para que el almacenamiento de datos se realice correctamente, es necesario instalar previamente una carpeta vacía en el directorio raíz llamada "Arduino"y un archivo ISO "Arduino" como se muestra en la ilustración 33, de esta forma el micro será capaz de detectar la SD.

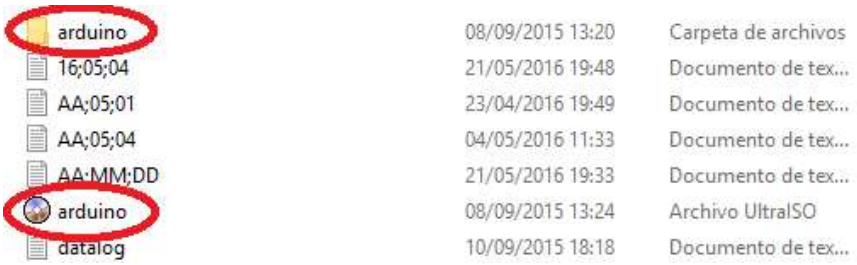


Ilustración 33. Configuracion SD

Por otra parte, para guardar los datos, el nombre del fichero donde vayan a ser guardados deberán ir precedidos por la extensión "/mnt/sd" para guardar en la micro SD como es nuestro caso, o "/mnt/usb" en el caso de la USB.

Las funciones para leer y escribir datos, pueden encontrarse en la librería 'FileIO.h', se explicará el uso de estas funciones, que proporcionan y que requieren en el manual del programador.

- Instalación NRF24L01

Como se ve de nuevo más adelante, los pines de la radio NRF24L01 son GND, VCC, CE, CSn, SCK, MOSI, MISO y IRQ. Para instalar la radio en el Arduino YUN, se aprovechará el puerto ICSP que contiene Arduino YUN. Esto lo se hará debido a que la radio funciona mediante los pines de comunicación MISO, MISO y SCK encontrados en el ISCP del YUN (Ilustración 34)

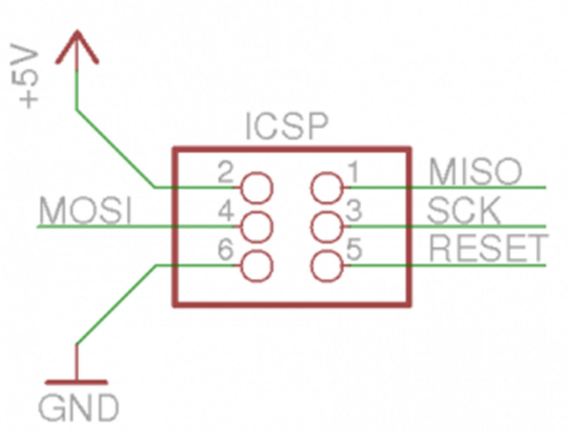


Ilustración 34. ICSP Arduino YUN [11]

Por tanto para acoplar el dispositivo NRF24L01 al Arduino YUN se soldarán los pines solicitado por el NRF24L01 con los respectivos en el Arduino YUN.

Las conexiones de pines realizadas se verán en la tabla 3 y en la ilustración 35.

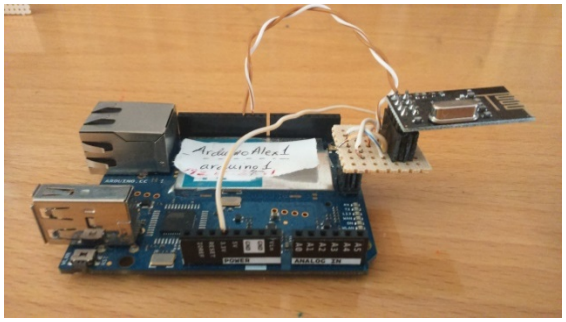


Ilustración 35. Instalacion NRF24L01

Tabla 3. Asignacion Pines NRF24L01 & YUN

Nombre PIN	[7]Pines NRF24L01	Pines YUN
CE	3	9
CSn	4	10
SCK	5	ICSP
MOSI	6	ICSP
MISO	7	ICSP
IRQ	8	No asignado

Evidentemente, también se conectará la radio a su tensión de funcionamiento VCC (3.3V) y a GND.

Una vez realizadas todas estas conexiones mediante soldadura con estaño, el NRF24L01 está listo para funcionar desde el módulo Arduino YUN.

[7] Los números asignados a los pines se verán reflejados en la ilustración 37. En el apartado "4.3.3 NRF24L01"

4.3.2. Arduino [8]NANO (IO Shield)

Respecto al conjunto Arduino NANO y placa IO Shield, estos dos módulos, junto al dispositivo NRF24L01, conformaran las placas esclavo y router. Las características técnicas tanto del Arduino Nano como la placa NanoIOShield son las siguientes:

- **Especificaciones técnicas Arduino NANO.**

Arduino NANO (Ilustración 36), consiste en un pequeño módulo breadboard basado en el microcontrolador ATmega328 o ATmega168. Consiste en un módulo de entradas y salidas analógicas y digitales muy convencional diseñado para la adquisición de datos.

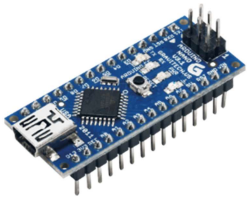


Ilustración 36. Arduino Nano [12]

A continuación,n se verá la tabla de especificaciones técnicas del módulo Arduino Nano:

Tabla 4 Especificaciones técnicas Arduino Nano

Microcontrolador	ATmega328
Voltaje (V)	5
Pines digitales I/O	14 (6 tipo PWM)
Pines analógicos de entrada	8
Corriente DC (mA)	40
Memoria Flash (Kb)	32
SRAM (Kb)	2
EEPROM (bytes)	1024
Clock Speed (MHz)	16
Largo (mm)	45
Ancho (mm)	18

- **Especificaciones técnicas Nano IO Shield.**

El IO Shield es una simple Proto board diseñada específicamente para albergar y comunicar entre si adecuadamente el módulo Arduino Nano, Xbee y NRF24L01 como se pudo observar en la ilustración 14 para conseguir una comunicación inalámbrica, de ahí que haya sido seleccionada para la realización de este proyecto.

[8] Se puede encontrar más información sobre estas cartas en su datasheet [12].

4.3.3. [9]NRF24L01

Como ya se ha hablado el NRF24L01 es el dispositivo de radio necesario para crear nuestra red mesh o mallada. Como se ha descrito en el punto *"4.1. Estudio y selección de componentes"* este chip, tiene una velocidad máxima de transmisión de 2.4GHz mediante el protocolo de comunicación 'Enhanced StockBurst™'. Para conseguir una explicación más clara y detallada del funcionamiento del NRF24L01, se detallará el diagrama de bloques en la ilustración 37 y en la página 9 del datasheet.

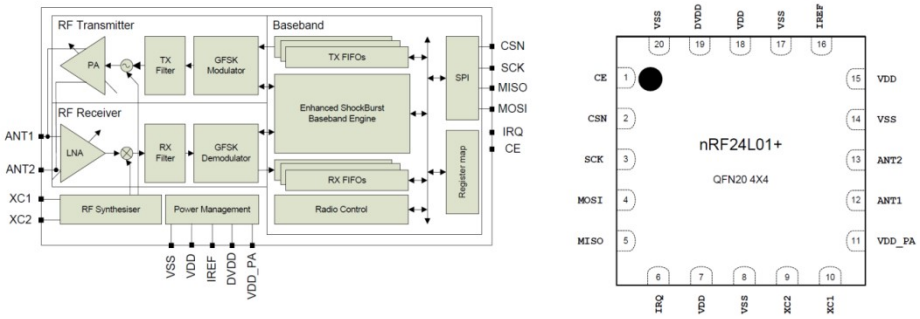


Ilustración 37. Diagrama de bloques y pines NRF24L01

Como vimos en el apartado "4.3.1 Arduino Yun/ instalación NRF24L01" entre todos estos pines, solo son accesibles los 8 primeros, los cuales deben ser conectados a nuestros módulos Arduino, son 8: los encargados de la SPI y comunicación FIFO, VDD y VSS. Estos pines pueden observarse en la ilustración 38, y son los encargados de llevar a cabo la comunicación inalámbrica como se ve más adelante.

Ilustración 38. Pines NRF24L01

En estos 8 pines. Los 6 pines FIFO se encargan de la comunicación inalámbrica junto al LNA t sus antenas de interfaz (ANT1 y ANT2). Los 2 pines VDD y VSS corresponderían al Power management y serían los correspondientes a la tensión de entrada y tierra (GND). En cuanto a la gestión del Power management, y por tanto para conseguir el correcto funcionamiento del NRF24L01, las especificaciones técnicas pueden ser vistas en la tabla 5, o con más detalle, en el datasheet [13] puntos del "2 Pin Information" al"5 Electrical specifications".

[9] Para más información acerca del NRF24L01, mirar en el datasheet [13]. En este punto se hablará de las especificaciones técnicas y pines de comunicación, del protocolo de comunicación y seguridad frente a fallo CRC, el diagrama de control, y la comunicación MultiCeiver

Tabla 5. Especificaciones técnicas NRF24L01

	Mínima	Máxima
VDD (V)	-0.3	3.6
VSS (V)	0	0
Energía de disipación (mV)	-	60
Temperatura de funcionamiento (ºC)	-40	85
Frecuencia cristal oscilador (MHz)	16	16
Frecuencia comunicación (MHz)	2400	2525
Resistencia equivalente (Ω)	100	100
Capacidad equivalente (pF)	8	16

Por ello, se conectará el NRF24L01 a la fuente de 3.3V ofertada por nuestros módulos Arduino.

- **Protocolo Enhanced StockBurst**

El protocolo [10]Enhanced StockBurst está basado en el automático reconocimiento, validación y reenvió de paquetes de datos de hasta 32 bytes; permitiendo una elevada capacidad y velocidad de comunicación por medio de un microcontrolador de bajo coste. Esta tarea la realiza concatenando los bits en el paquete de transmisión de manera automática.

Durante la recepción de datos establece un protocolo de seguridad frente a fallos. En este protocolo, lee constantemente la señal de validación del módulo con el que se esté comunicando. Cuando la señal de validación es correcta, procesa el resto de datos y procede a la validación del CRC. Si el CRC dictamina que el paquete es válido, el paquete de datos es movido al buffer RX [11]FIFOs (buffer de lectura de datos) para su posterior retransmisión, o almacenaje.

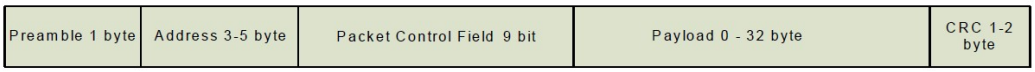


Ilustración 39. Paquete de datos. [13]

Como se muestra en la ilustración 38, en el paquete de transmisión de datos encontramos la carga, o información que se quiere transmitir de 32 bytes, junto con la dirección, byte de sincronización (preamble byte) y bytes de seguridad frente a fallo como el CRC o control:

Byte de sincronización: Consiste en una secuencia de bits para obtener la sincronización entre distintos módulos, para llevar a cabo una correcta comunicación entre estos. Para esta sincronía, un módulo manda un byte de ceros y unos alternados, y el otro le devuelve el mismo byte invirtiendo los ceros y unos. Por ejemplo:

Modulo1 → 01010101 Módulo 2→ 10101010

[10] Se puede encontrar en la página 26 del datasheet [13], punto 7.
[11] FIFO son buffers o archivos, que pueden ser usados o son accesible por distintos integrantes no relacionados.

Bytes de dirección: Con estos bytes se configura el canal de comunicación por el que se transmitirá el paquete de datos, de esta forma se enviaran los datos al módulo o módulos adecuados y mediante la seguridad de validación de la dirección, se podrá saber si el modulo o módulos han recibido correctamente los datos o no. Esto se implementara mediante chequeos de recepción y transmisión en las líneas de código reflejada en "Anexos. 4.4. Red Mesh", de esta manera se sabrá si hace falta volver a mandar el paquete de datos.

Bits de control: Estos bits se dividen en tres partes, como se refleja en la ilustración 39, entre los cuales el más significativo se encuentra a la izquierda. Los primeros 6 bits, especifican el tamaño de la carga o información que se desea transmitir (de 0 a 32 bytes); los dos siguientes corresponden a los de identificación PID, reflejando si el paquete recibido es nuevo o corresponde a un reenvió; por último, el noveno bit (llamado NO_ACK: No Acknowledgment flag), si este bit se encuentra en 1, significa que no ha habido acuse de recibo, por lo que algo en la comunicación va mal y debe ser retransmitido.

Ilustración 40. Bits de control [13]

Bytes de carga (payload): Contienen la información que se quiere transmitir.

Byte CRC (Cyclic Redundancy Check): Es un byte de detección de errores en el paquete de transmisión calculado mediante los bytes de dirección. Si el CRC falla, ha habido fallo en la transmisión por lo que el paquete no será aceptado.

- **Diagrama de control**

Ya se ha visto el protocolo de comunicación del NRF24L01, a continuación se hará una breve explicación de cómo llevar el control de esta comunicación mediante los 8 pines expuestos del NRF24L01, la cual se transformará a código más adelante. Esta explicación, se llevara a cabo con ayuda del diagrama visible en la ilustración 41 y en el datasheet [13] página 21 y que se explica a continuación:

Como se indica el paso necesario y previo al control serial conectar el dispositivo NRF24L01, a una fuente de tensión entre 0 y 3.6V, como se indica en las especificaciones técnicas. Una vez hecho esto, y mientras no reciba ninguna orden, pasaría a un estado de ahorro de energía o durmiente (power down) gastando el mínimo consumo posible.

El dispositivo permanecerá en este estado, hasta el el bite [12]PWM_UP sea activado, si en cualquier momento este bit vuelve a 0, nuestro dispositivo regresara al modo power down. Con PWM_UP=1, pasa al modo Standby o espera donde se activa también el cristal oscilador

[12] Este pin corresponde al pin de estado, simplemente indica si el modulo pasa a modo ahorro de energía o power down; o se encuentra latente a la espera de recibir un estímulo, ya que los pines SPI vistos en el diagrama de bloques siguen atentos.

donde permanecerá atento al bite CE y a la orden de transmitir o recibir datos dada por el bit [13]PRIM_RX.

PRIM_RX=1 →Recibir (start.listening)

PRIM_RX=0→Transmitir (stop.listening)

En cuanto al bit CE indica si permanece en modo Standby (CE=1) o está ejecutando alguna acción (CE=0) si CE=0, automáticamente vuelve a Standby desde cualquier punto.

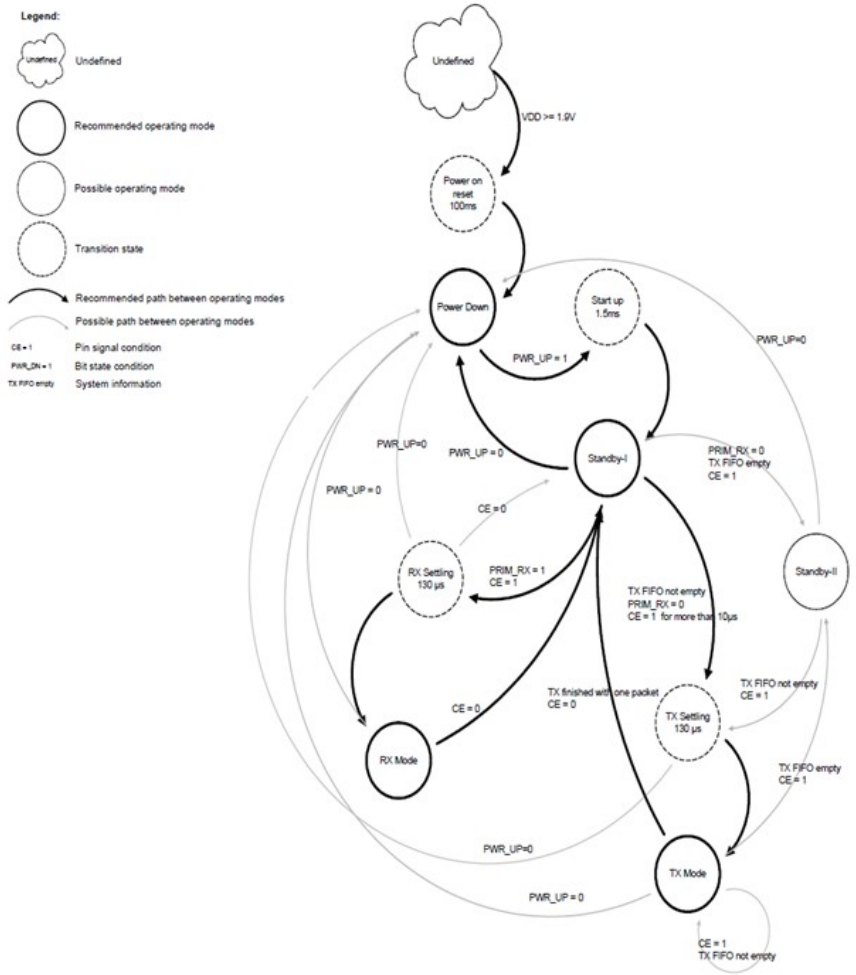


Ilustración 41. Diagrama de control nRF24L01 [13]

Por último, existe otro modo Standby 2, en el cual entrara si el buffer FIFO_TX se satura mientras esta en modo de transmisión de datos, entrando en un punto de espera hasta que este se vacíe o deje de estar saturado.

[13] Este bit, en el código corresponderá a la función 'start.Listening' y 'stop.listening' dadas por la Librería ''RF24.h' cual incluye la 'nRF24L01.h'.

El resto de la comunicación, conociendo la funcionalidad de los 3 bits reflejados, se puede seguir por medio del diagrama siendo TX lo referido a la transmisión de datos y RX lo referido a la recepción de datos.

- **Comunicación MultiCeiver**

Ya se sabe que los dispositivos nRF24L01, se comunican entre ellos a través de un canal, sin embargo para generar nuestra red mesh, se necesita la comunicación Multiceiver.

La comunicación MultiCeiver, permite la creación de 6 canales de comunicación por cada dispositivo nERF24L01. Gracias a esto se podrá comunicar cada dispositivo con otros 6 a la vez. Esto en el código se representara con la creación de un vector PIPE, el cual incluirá los 6 canales de comunicación dado en hexadecimal con un tamaño de 3 a 5 bytes como dirección o canal.

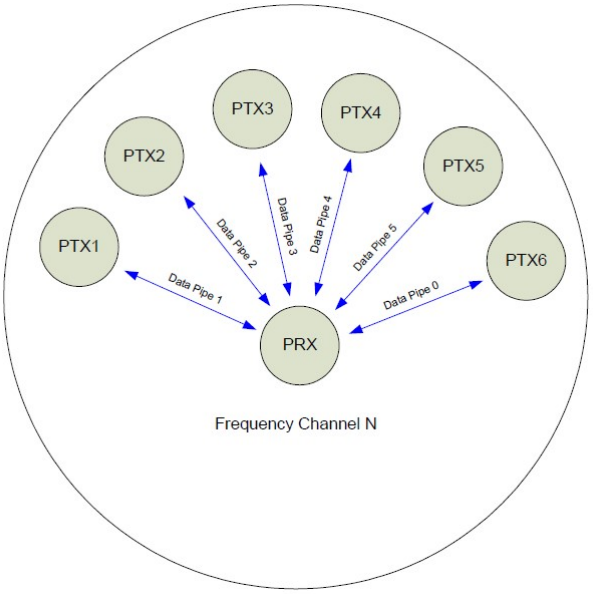


Ilustración 42. Comunicación MultiCeiver [13]

Según con el dispositivo que queramos hablar, se señalará con la función openReadingPipe (1, pipe[canal]) si se quiere que nuestro modulo lea por ese canal o openWritingPipe(pipe[canal]) si se quiere que escriba por ese canal, pero esto se explicara con mayor exactitud en el apartado “Manual del programador y usuario”.

4.3.4. DS1307

Como ya se ha dicho, el [14]DS1307 es el encargado suministrar la hora y la fecha en la que se tomó el dato. Así pues, en este sub apartado se hablará de sus integrantes, sus pines principales y sus especificaciones técnicas.

Este dispositivo deberá ser colocado en la palca maestro, de esta manera será capaz de coordinar todas las tareas a tiempo real. Ya hablamos de su instalación en el apartado "4.3.1. Arduino Yun". Así pues, la disposición de los pines del DS1307 necesario para realizar bien las conexiones es el de la ilustración 43.

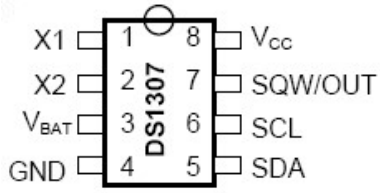


Ilustración 43. Pines DS1307 [14]

Viendo esta ilustración se sabe que los pines encargados de la comunicación I_2C son el 5 y 6, por lo que se conectaran a los pines TWI del Arduino YUN.

El dispositivo DS1307, es un dispositivo delicado, el cual necesita que se cumplan con gran precisión sus especificaciones para lograr un correcto funcionamiento, así como la adición del cuarzo adecuado.

En primer lugar los pines X1 y X2, van conectados al cristal oscilador de 32MHz exactamente; además esta conexión debe ir acompañada de un par de resistencias conectadas entre la entrada a 5V y los pines de comunicación I_2C de 10kΩ; por último, debe disponer de una batería auxiliar o pila de 3V conectada al pin 3, ya que si no se encuentra a un mínimo de 3V, el dispositivo DS1307 se reseteara, volviendo a la hora 0 del mes 0 del año 0. Todos estos aspectos, y las conexiones realizadas en nuestro proyecto se reflejan en el diagrama de la ilustración 44.

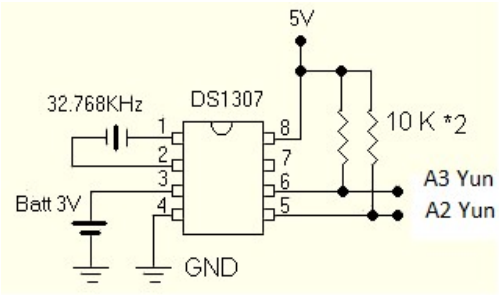


Ilustración 44. Diagrama conexiones DS1307 [15]

[14] Para más información, consultar el datasheet [14].

4.4. Programa auxiliar Ratón

Ratón consiste en un programa diseñado con software Python. No se va a perseverar demasiado en cómo se ha creado o diseñado Ratón, debido a que este programa no objeto de este proyecto. Sin embargo si debe ser mencionado, explicar en qué consiste y que funciona realiza, ya que condicionara el formato en el que se guardan los datos y donde.

Ilustración 45. Icono Ratón [16]

El programa Ratón ha sido diseñado por el colaborador de este proyecto Ángel Fernández Navajas durante un periodo de dos meses trabajando en el Departamento de Física Aplicada (U.D. Industriales) Universidad Politécnica de Valencia, España; en el laboratorio de investigación número 3 (de septiembre a noviembre de 2015).

La función de este programa consiste en leer una carpeta y sus ficheros internos llamada “Pendiente” en Dropbox. Si esta carpeta está llena, va leyendo los archivos en orden de fecha de instalación, y lo guarda en otra carpeta interna del ordenador llamada “Leídos”. Una vez el fichero está guardado en el disco duro del computador, elimina el correspondiente fichero de Dropbox. En caso de detectar algún fallo en el formato de lectura, elimina este fichero de Dropbox y lo guarda en una tercera carpeta llamada “errores”; por ello para la correcta lectura de los ficheros y el correcto funcionamiento de ratón, la realización de este proyecto estará guiada por las exigencias de formato que requiere Ratón, reflejadas en el “*Pliego de condiciones*”. Por otra parte será necesario que nuestro proyecto suba los datos a Dropbox, ya que nuestra carpeta de datos en la nube debe poder asociarse al ordenador.

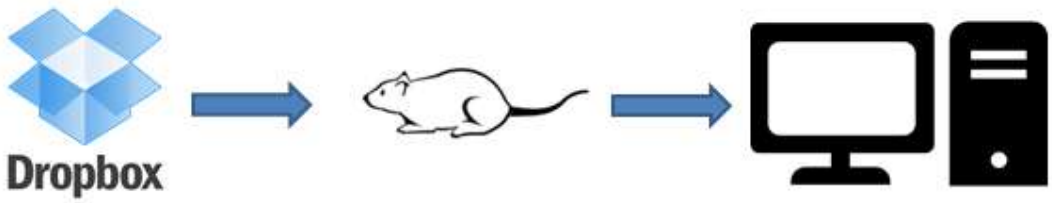


Ilustración 46. Proceso almacenamiento en computador

5. Resultados

En cuanto a los resultados que proporciona el prototipo, son los siguientes:

- **Guardados en la SD**

Al estar en la SD, no vienen guiados por las condiciones de formato impuestas por Ratón. Al guardar continuamente en la SD, sin generar nuevas carpetas en función de los días, también escribirá en el fichero.txt la fecha y hora en la que se tomaron los datos.

En la siguiente ilustración, se muestra un ejemplo de datos guardados en la SD mediante la toma a través de la Red mesh con 3 sensores, un repetidor y el maestro, que incluye un cuarto sensor.

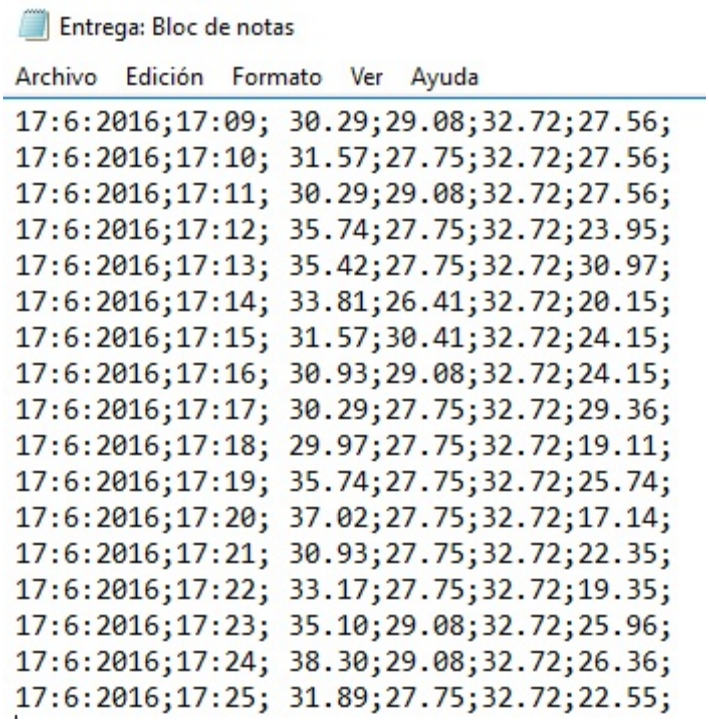


Ilustración 47. Resultados SD

- **Guardados en Dropbox**

Los resultados expuestos, no son los del programa oficial, ya que este tendría que estar encendida al final del dia para general la carpeta de datos y haber estado encendido todo el día, lo cual mostraría una lista de 1440 datos. Por lo que se ha decantado por mostrar los datos con una pequeña modificación del programa generando el fichero entrega mostrado anteriormente en la SD y creando el siguiente en Dropbox:

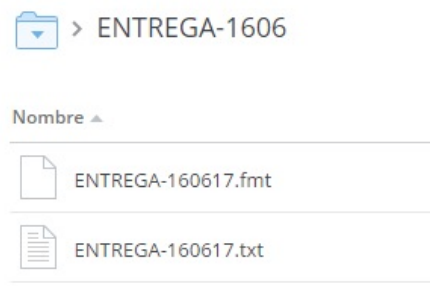


Ilustración 48. Nombre carpeta y ficheros .txt y .fmt

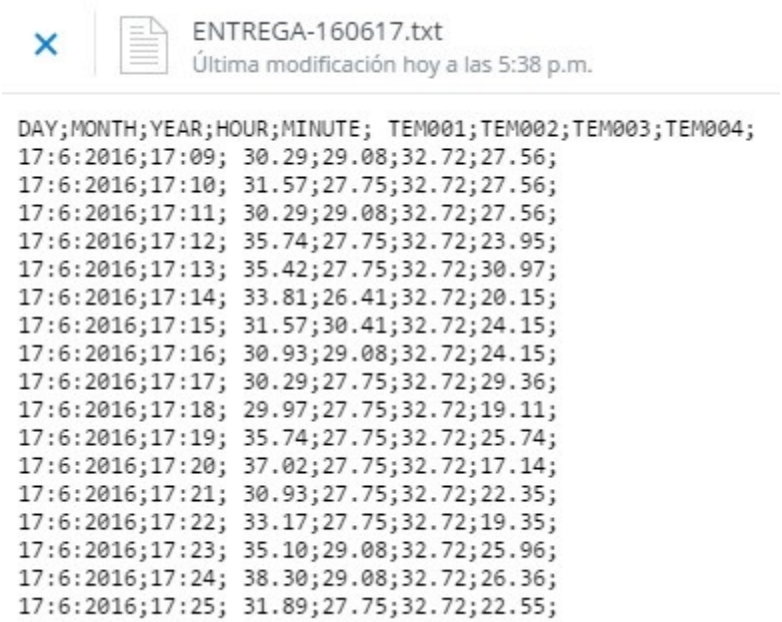


Ilustración 49. Datos Dropbox

- **Avisos vía Twitter**

Para simular el aviso de emergencia vía un Twitter, se ha sustituido un sensor LM35, por un potenciómetro, el cual simulara la calibración del LM35, con la diferencia de que se podrá controlar la temperatura simulada que devuelve.

Modificando la resistencia del potenciómetro, se podrá conseguir que Arduino lea temperaturas superiores a los 100ºC, y de esta forma entre en el bucle de comunicación con Twitter, alertando acerca del sensor averiado o que muestra temperaturas anómalas.

Un ejemplo de aviso viene mostrado en la siguiente ilustración:

Ilustración 50. Aviso vía Twitter

6. Futuras mejoras

En este apartado se hablara de posibles mejoras que se aplicaran al proyecto más adelante, para que tenga un funcionamiento más fiable, sea más atractivo y más económico. De esta forma podrá ser instalado en las respectivas instalaciones de patrimonio cultural e incluso ser vendido a empresas interesadas en su funcionalidad para monitorizar cualquier factor medible mediante un sensor; ya sean cámaras frigoríficas, túneles de viento, calderas, galgas…

Actualmente no cumple ninguna condición requerida para cumplir estos objetivos, ya que le faltan ciertas mejoras para su instalación y posible distribución. Por ello la conclusión de este proyecto oferta un nuevo, que consistiría en mejorar las características que ofrece mediante las siguientes alteraciones, e incluso alguna propuesta más en el futuro:

1- **Sustituir el módulo Arduino Yun por una Raspberry Pi**. Este cambio, además de mejorar la funcionalidad y el rendimiento de la placa maestro; proporcionar mayor diversidad de funciones y diversidad de lenguajes; también daría lugar a un prototipo mejor visto por el mercado.
   Por otra parte, y siendo un factor muy importante a tener en cuenta, gracias a una Raspberry podríamos prescindir de la plataforma Temboo, objetivo que resultaría también mejor visto por el mercado, a la vez que abarataría costes permanentes (cota mensual de Temboo)
   Como se ha dicho antes, esta mejora no se ha instalado en este proyecto, por intentar realizarlo todo mediante software Arduino, por falta de conocimiento en lenguajes como Java o Python y restricciones de tiempo para realizar este proyecto, además una raspberry no está específicamente diseñada para la adquisición de datos, en cambio Arduino gracias a las librerías que proporciona esta mayor capacitado para este fin.

2- **Diseño de las placas.** Este cambio, permitiría prescindir de Arduino, instalando únicamente los microcontroladores ATmega328 y programándolos en C. También, esta mejora, podría ahorrar cualquier problema de compatibilidad con la placa maestro, la cual estaría ahora formada por la Raspberry.
   Por último, esta mejora, supondría un considerable aumento en el atractivo del proyecto respecto al mercado, ya que serían productos propios y no productos provenientes de otra entidad.

3- **Incorporación sensores mediante protocolo de comunicación 1-wire.** Esta mejora viene ya diseñada en el datalogger predecesor a este proyecto de Borja Esteban Sanchis. Por lo cual no supondría una gran inversión de tiempo llevarla a cabo con gran repercusión a la hora de instalar sensores de humedad y temperatura en nuestro prototipo de datalogger.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO DE FIN DE GRADO EN INGENIERIA EN TECNOLOGÍAS INDUSTRIALES

ANEXOS

AUTOR: ESTEBAN SANCHIS, ALEJANDRO

TUTOR: GARCÍA-DIEGO, FERNANDO J.

Curso Académico: 2015-2016

Contenido Anexos

1. Codigo VDIP1 para la verificación de su funcionamiento

La realización de este código es algo muy simple, ya que todas las herramientas necesarias para su realización vienen dadas del proyecto antecesor **"Implementation and Design of a Datalogger with Microcontroller for Data Extraction from Sensors Using 1-WIre Protocol".**

Por ello, la realización de este código se basara simplemente en crear o modificar código a una versión básica, para que, en vez guardar datos leídos por los sensores, guarde una frase simple y comúnmente usada entre programadores "*Hola mundo*", y así poder comprobar el correcto funcionamiento del módulo VDIP1.

Para llevar a cabo esta comprobación se seguirán dos fases. Una primera 'Programación extensiva' en la que se programa el VDIP1 desde las funciones internas del DataSheet [17] y las ordenes de abrir, escribir, leer y cerrar fichero se le darán mediante puerto serial; y una segunda 'Librerías' en la que los comandos se ejecutaran directamente desde la librería, ya que pudiera ser que los errores fuesen proporcionados por la librería.

- **Programación extensiva**

En esta primera etapa, se comprobara si el VDIP1 es capaz de realizar sus funciones básicas, para ello le se mandarán comandos directamente desde su programación extensiva, según indica el datasheet [17] de comandos del VDIP1 en las páginas: 26 y 27.

Para llevar a cabo esta tarea se implementara el siguiente código, en una placa aparte que solo contenga un microcontrolador ATmega328 y el VDIP1.

Código 1. VDIP1_Version extensa.

```
//by Alejandro Esteban Sanchis

//COMPROBACION COMUNICACION DIRECTA MICRO CON VDIP1, FUERA DE PLACA

void setup() {
  Serial.begin(9600);
  pinMode(13, OUTPUT);
}

void loop() {

  Serial.write("ECS"); Serial.write(0x0D);
  digitalWrite(13, LOW);

  //abrir fichero
  Serial.write("OPW funciona.txt"); Serial.write(0xD);
  delay(100);
  //escribe en el fichero
  Serial.write("WRF 5"); Serial.write(0xD); delay(100); Serial.write("hola mundo"); Serial.write(0xD);
  delay(100);
  //cerrar fichero
  Serial.write("CLF funciona.txt"); Serial.write(0xD);
  delay(4000);
}
```

- **Librerías**

Una vez comprobado que el VDIP1 funciona correctamente, podría darse el caso de que el fallo este en la librería, por lo que se llevara a cabo esta segunda comprobación.

En esta parte, no se creara un código nuevo, sino que directamente se modificara programa maestro del antecesor, para que el comando Loop() solo contenga las funciones abrir, escribir, cerrar fichero, llamadas desde librería.

Las funciones de la librería que van a usarse, vendrán especificadas las tareas que realiza en el "Manual del programador y usuario", sin embargo no se incidirá más en ella, ya que pertenece al PFC de Borja Esteban Sanchis.

El código empleado para verificar el correcto funcionamiento mediante librerías es el siguiente:

Código 2. VDIP1_Librerías

```
//Modificado por Alejandro Esteban SAnchis
//Original de Borja Esteban Sanchis


#include <VDIP1ART.h>
#include <TPS3813ART.h>

// Prueba para ver si funciones VDIP OpenF, CloseF y writte funcionan

//1ºAbrir VDIP(OpenF)
//2ºEscribir en USB (VDIP.Write)
//3ºCerrar VDIP(closeF)

// VDIP1
const long VDIP_Serial_Speed = 9600; //Velocidad del protocolo Serial en Baudios
const byte PIN_LED_USB = 4;          //Pin donde se localiza el Led de uso de la USB

//Nombre y frase a guardar
byte FILE_NAME[] = "Prueba funcionamiento librerias";
byte TEXT_DATE[] = "Hola mundo\r\n";

void setup() {
}
```

```
void loop() {

  // Creacion de clases
  VDIP1ART VDIP1(VDIP_Serial_Speed);     //Objeto que contiene el control de la VDIP1

  // Asignacion de los pines
  pinMode(PIN_LED_USB, OUTPUT);
  int PIN_LED_USB_State = 0;
  digitalWrite(PIN_LED_USB, PIN_LED_USB_State);

  //Abre, guarda y cierra la frase en el VDIP1
  VDIP1.OpenF(FILE_NAME);
  VDIP1.Write(TEXT_DATE, sizeof(TEXT_DATE));
  VDIP1.CloseF(FILE_NAME);
}
```

2.Estudio comparativo nRF24L01 vs Xbee

Como ya hablamos de ello en la memoria apartado *"4.1. Estudio y selección de componentes"*. El propósito de esta matriz comparativa es seleccionar la mejor opción para llevar a cabo nuestra comunicación inalámbrica.

A continuación se vuelve a mostrar la tabla de funcionalidades:

Tabla 6. NRF24L01 vs Xbee numerada

	NRF24L01	Xbee S2	Xbee S1
A-Precio por unidad (euros)	(14.32/10) = 1.44	31.22	29.95
B-Alcance comunicación (m)	20 - 80	500 - 1000	500 - 1000
C-Consumo	Bajo	Bajo	Bajo
D-Complejidad	Baja	Alta	Media
E-Velocidad (Kbps)	250 - 2048	1024	256

A continuación, para la realización de nuestro proyecto se darán los siguientes valores numéricos de importancia a cada cualidad, según interesa para nuestro proyecto, es decir, primando el tiempo de transmisión y el coste:

Tabla 7. Ponderación

	A	B	C	D	E	Suma	%
A		0.7	0.6	0.8	0.5	2.6	26
B	0.3		0.3	0.6	0.2	1.4	14
C	0.4	0.7		0.8	0.5	2.4	24
D	0.2	0.4	0.2		0.2	1	10
E	0.5	0.8	0.5	0.8		2.6	26
						10	100

Tabla 8. Valores relativos nRF24L01 vs Xbee

Valores asignados: Bajo/baja = 3; Medio = 2; Alto/alta = 1.

Rango valores relativos = [0-3]

	NRF24L01	Xbee S2	Xbee S1
Precio por unidad (euros)	1.44≡**3**	3*1.44/31.22=**0.138**	3*1.44/29.95=**0.144**
Alcance comunicación (m)	80*3/1000=**0.24**	1000≡**3**	1000≡**3**
Consumo	bajo≡**3**	bajo≡**3**	bajo≡**3**
Complejidad	bajo≡**3**	alto≡**1**	medio≡**2**
Velocidad (Kbps)	2048≡**3**	1024*3/2048=**1.5**	256*3/2048=0.375

Por último, se realizaran los cálculos ponderados:

Ecuación 2. Cálculos ponderados

- **NRF24L01 = 3*0.26+0.24*0.14+3*0.24+3*0.1+3*0.26 = 2.61**
- **Xbee S2 = 0.138*0.26+3*0.14+3*0.24+1*0.1+1.5*0.26 = 1.67**
- **Xbee S1 →No puede llevar a cabo una red mesh → Descartado**

Como pude verse, la nota respecto a 3, del nRF24L01 es muy superior a la de Xbee S2, así como el Xbee S1 queda completamente descartado su uso debido a que no posee la capacidad de llevar a cabo una red mallada o mesh.

En las siguientes graficas el parámetro complejidad y consumo se invierten, para mayor comprensión por parte del usuario:

Alta = 3; Medio =2; Bajo =1.

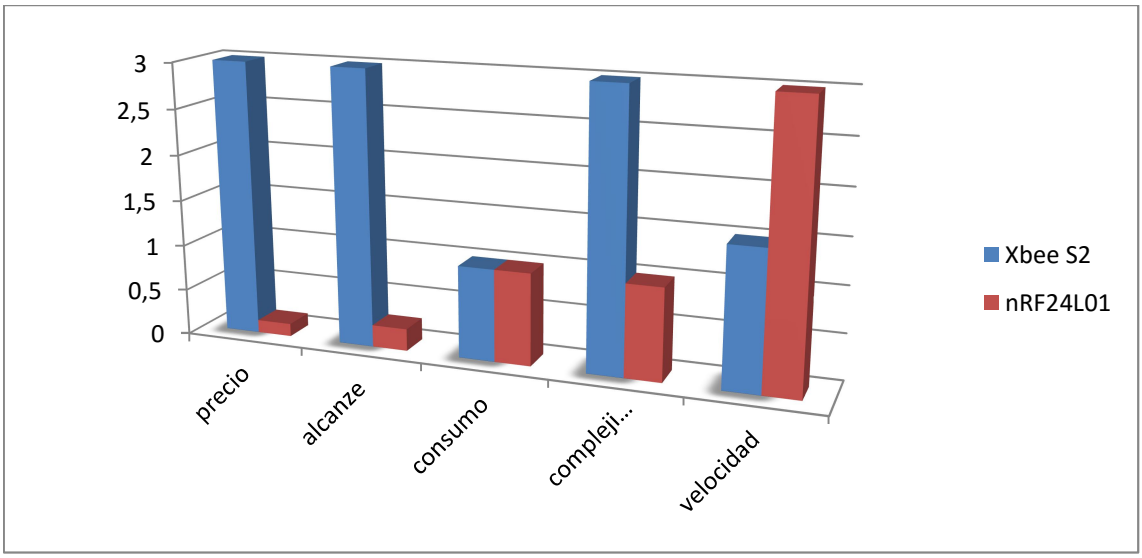


Ilustración 51. Grafico Xbee S2 vs nRF24L01

Como conclusión del estudio comparativo, la mejor opción según el criterio económico y el criterio de velocidad de transmisión es el nRF24L01.

3.Calibracion LM35

La toma de todos los datos de temperatura, como se ha dicho en la memoria, se realiza mediante un sensor LM35. Este sensor mide 1ºC por cada 10mV de tensiones ofertadas (Rangos [0-5]V; [0-500]ªC), siendo capaz de resistir temperaturas entre -55ºC y [15]150ºC Suficiente rango para las temperaturas que debe medir nuestro prototipo.

Arduino, dispone de 10 bits, por lo que → 2^10 =1024 equivaldrían a los 5V, teniendo en cuenta los 10mV/ºC, quedaría que por cada 1V equivaldría a 100ºC:

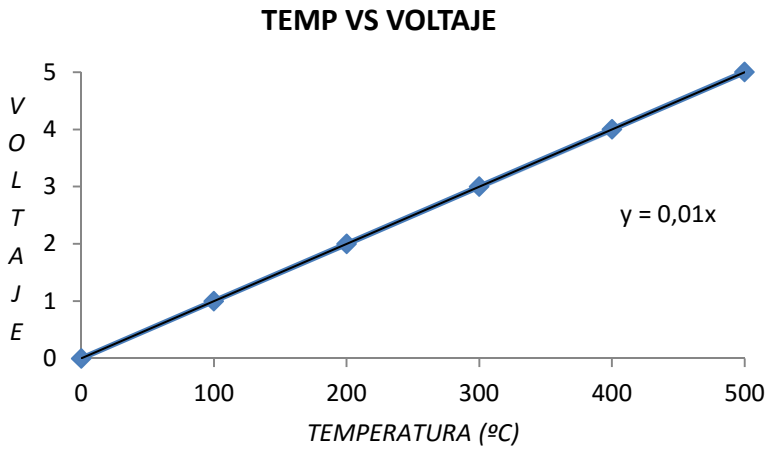


Ilustración 52. LM35 Tensión-Temperatura

Por otra parte, la curva tensión-bits de Arduino quedaría de la siguiente forma:

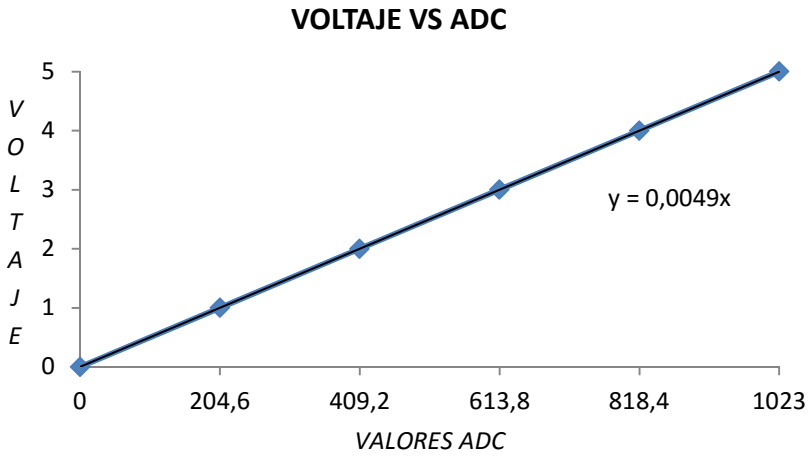


Ilustración 53. Arduino Tensión-Bits

[15] Por ello se establece el aviso de emergencia a 100ºC, proporcionando un margen de seguridad, dando tiempo para efectuar una correcta actuación.

Con ayuda de estas gráficas, el calibrado y por tanto la recta de conversión analógico digital queda de la siguiente forma:

Ecuación 3. Recta calibrado LM35

$$\boldsymbol{Temperatura}\ (\boldsymbol{^{o}C}) = \left(\frac{^{16}\boldsymbol{lectura} * \mathbf{500}}{\mathbf{1023}}\right)$$

Ejemplo:

Lectura = 100.

$$Temperatura\ = \left(\frac{100 * 500}{1023}\right) = 48.87\ ^{o}C$$

Una vez calculada la recta de calibrado, su forma es la siguiente:

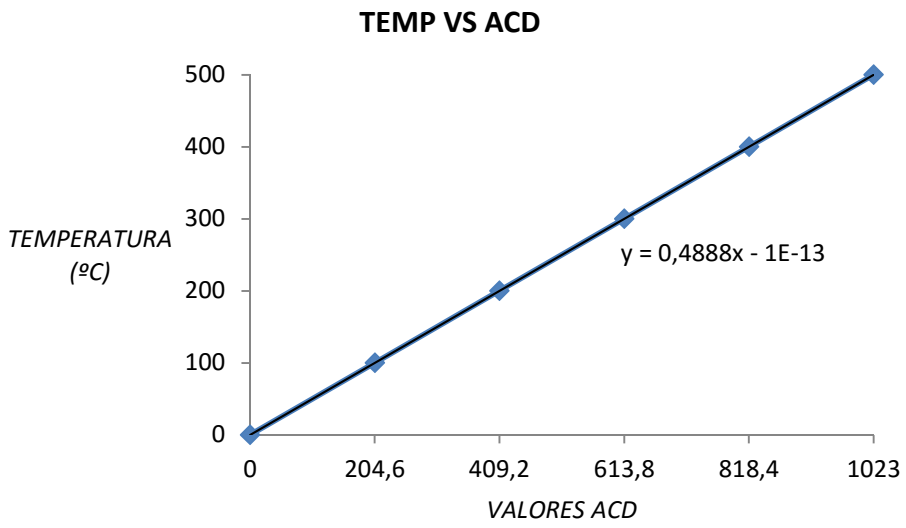


Ilustración 54. Temperatura vs Valores ADC

Cabe remarcar que a la hora de programar, los bits corresponderán a enteros, y la temperatura y tensión corresponderían a variables tipo '*float*'.

[16] Lectura es el valor leído por analogRead(), es decir un valor comprendido entre 0 y 1023, dentro del rango de bits ofertados.

4.Códigos prototipo

Todos los códigos mostrados en este apartado, incluirán al inicio de la función de bucle continuo '*Void Loop()*' la iteración del dispositivo DS1307 calibrado con la hora exacta. Mediante la librería '*DS1307ART.h*' y sus funciones expuestas en el "*Manual del programador y usuario*" guiara la adquisición de datos, para que se realice estrictamente cada minuto. Por otro lado otorgara la fecha y hora en la que se tomó el dato tanto al programa, como al dispositivo de memoria externo.

4.1. Guardar en la SD

Este programa es para testear si guarda correctamente en la SD los datos tomados por un sensor colocado en el Arduino Yun cada minuto.

Código 3. SD

```
//Por Alejandro Esteban Sanchis

//Librerias SD
#include <Bridge.h>
#include <FileIO.h>
//Librerias DS1307
#include <DS1307ART.h>
#include <Wire.h> //Necesita Wire.h para la comunicacion I2C

//Contructor reloj
DS1307ART External_Clock;  //Cuando se llame a una funcion del reloj se hara con este nombre

//Nombres archivos
char nameSD[] = "/mnt/sd/PruebaSD.txt";

//Variables para la fecha-reloj
byte fecha[7];
byte Current_Minute;
byte Last_Minute;


//Datos
float Adata1_1 = 0;
float Adata1_2 = 0;
float temp1=0;
float temp2=0;

//Variables para if's
int bucles=0;

//Datos a subir
String conc;

void setup() {
  Serial.begin(9600);
  Bridge.begin();
  FileSystem.begin();
  Wire.begin();
}
```

```
void loop() {
  //Primero de todo refrescamos la variable conc
  conc ="";

  //Comenzamos con la llamada al reloj
  if (bucles > 0) { //Se quedara aqui hasta el siguiente minuto
    do { //haz esCurrent_Minuteto mientras ultimo monuto y el actual sean iguales, cuando cambia el actual, sale
      Serial.print("entro en espera last minute y actual minute:   ");
      Serial.print(Last_Minute); Serial.println(Current_Minute);
      External_Clock.getDateInt(fecha);
      Current_Minute = fecha[1];
      External_Clock.checkDate();
      delay(10000); //sobra es para que no marque tanto
    } while (Last_Minute == Current_Minute);
  }
  External_Clock.getDateInt(fecha);
  Last_Minute = fecha[1];
  Current_Minute = Last_Minute;
  //Concatenamos la fecha
  conc.concat(fecha[4]); conc.concat(":"); conc.concat(fecha[5]); conc.concat(":20");
  conc.concat(fecha[6]); conc.concat(";"); conc.concat(fecha[2]); conc.concat(":");
  conc.concat(fecha[1]); conc.concat(":"); conc.concat(fecha[0]);conc.concat(";");
  Serial.print("la fecha es:  "); Serial.println(conc);
  External_Clock.checkDate();


  //Leemos datos desde arduino YUN
  Adata1_1 = analogRead(4);
  Adata1_2 = analogRead(5);
  //Los pasamos a temperatura
  temp1 = ((Adata1_1 * 5000) / 1024) / 10;
  temp2 = ((Adata1_2 * 5000) / 1024) / 10;
  //Los concatenamos
  conc.concat(" ");conc.concat(temp1);conc.concat(";");conc.concat(temp2);conc.concat(";");
  Serial.print("Los datos son:");Serial.println(conc);

  //Ya solo quea guardarlo en la SD
  Serial.println(nameSD);
  File  file = FileSystem.open(nameSD, FILE_APPEND);
  file.println(conc);
  file.close();

  //FIN
  bucles++;
}
```

4.2. Subida de datos a Dropbox y aviso por Twitter

4.2.1. Encriptación base 64

Una de las mayores complicaciones que se presentan en la realización de este proyecto aparece a la hora de llamar funciones de librería, ya que cada librería pide un formato específico de entrada.

El principal ejemplo reside en la librerías para Dropbox ofertadas por Temboo. Estas librerías piden la entrada de datos codificados a Base 64, por lo que previamente a la llamada de las funciones, habrá que codificar los datos.

Teóricamente la [17]encriptación a base 64, según [18], se realiza de la siguiente forma:

1- Introducimos el texto de entrada.
2- Este texto tendrá su correspondiente valor dentro de la tabla ASCII, por lo que sustituimos cada carácter por su equivalente numérico ASCII.

El código ASCII
sigla en inglés de American Standard Code for Information Interchange
(Código Estadounidense Estándar para el Intercambio de Información)
www.elcodigoascii.com.ar

Caracteres de control ASCII

DEC	HEX	Simbolo ASCII	
00	00h	NULL	(carácter nulo)
01	01h	SOH	(inicio encabezado)
02	02h	STX	(inicio texto)
03	03h	ETX	(fin de texto)
04	04h	EOT	(fin transmisión)
05	05h	ENQ	(enquiry)
06	06h	ACK	(acknowledgement)
07	07h	BEL	(timbre)
08	08h	BS	(retroceso)
09	09h	HT	(tab horizontal)
10	0Ah	LF	(salto de linea)
11	0Bh	VT	(tab vertical)
12	0Ch	FF	(form feed)
13	0Dh	CR	(retorno de carro)
14	0Eh	SO	(shift Out)
15	0Fh	SI	(shift In)
16	10h	DLE	(data link escape)
17	11h	DC1	(device control 1)
18	12h	DC2	(device control 2)
19	13h	DC3	(device control 3)
20	14h	DC4	(device control 4)
21	15h	NAK	(negative acknowle.)
22	16h	SYN	(synchronous idle)
23	17h	ETB	(end of trans. block)
24	18h	CAN	(cancel)
25	19h	EM	(end of medium)
26	1Ah	SUB	(substitute)
27	1Bh	ESC	(escape)
28	1Ch	FS	(file separator)
29	1Dh	GS	(group separator)
30	1Eh	RS	(record separator)
31	1Fh	US	(unit separator)
127	20h	DEL	(delete)

Caracteres ASCII imprimibles

DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo
32	20h	espacio	64	40h	@	96	60h	`
33	21h	!	65	41h	A	97	61h	a
34	22h	"	66	42h	B	98	62h	b
35	23h	#	67	43h	C	99	63h	c
36	24h	$	68	44h	D	100	64h	d
37	25h	%	69	45h	E	101	65h	e
38	26h	&	70	46h	F	102	66h	f
39	27h	'	71	47h	G	103	67h	g
40	28h	(	72	48h	H	104	68h	h
41	29h	)	73	49h	I	105	69h	i
42	2Ah	*	74	4Ah	J	106	6Ah	j
43	2Bh	+	75	4Bh	K	107	6Bh	k
44	2Ch	,	76	4Ch	L	108	6Ch	l
45	2Dh	-	77	4Dh	M	109	6Dh	m
46	2Eh	.	78	4Eh	N	110	6Eh	n
47	2Fh	/	79	4Fh	O	111	6Fh	o
48	30h	0	80	50h	P	112	70h	p
49	31h	1	81	51h	Q	113	71h	q
50	32h	2	82	52h	R	114	72h	r
51	33h	3	83	53h	S	115	73h	s
52	34h	4	84	54h	T	116	74h	t
53	35h	5	85	55h	U	117	75h	u
54	36h	6	86	56h	V	118	76h	v
55	37h	7	87	57h	W	119	77h	w
56	38h	8	88	58h	X	120	78h	x
57	39h	9	89	59h	Y	121	79h	y
58	3Ah	:	90	5Ah	Z	122	7Ah	z
59	3Bh	;	91	5Bh	[	123	7Bh	{
60	3Ch	<	92	5Ch	\	124	7Ch	|
61	3Dh	=	93	5Dh	]	125	7Dh	}
62	3Eh	>	94	5Eh	^	126	7Eh	~
63	3Fh	?	95	5Fh	_			

elCodigoASCII.com.ar

ASCII extendido

DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo
128	80h	Ç	160	A0h	á	192	C0h	└	224	E0h	Ó
129	81h	ü	161	A1h	í	193	C1h	┴	225	E1h	ß
130	82h	é	162	A2h	ó	194	C2h	┬	226	E2h	Ô
131	83h	â	163	A3h	ú	195	C3h	├	227	E3h	Ò
132	84h	ä	164	A4h	ñ	196	C4h	─	228	E4h	õ
133	85h	à	165	A5h	Ñ	197	C5h	┼	229	E5h	Õ
134	86h	å	166	A6h	ª	198	C6h	ã	230	E6h	µ
135	87h	ç	167	A7h	º	199	C7h	Ã	231	E7h	þ
136	88h	ê	168	A8h	¿	200	C8h	╚	232	E8h	Þ
137	89h	ë	169	A9h	®	201	C9h	╔	233	E9h	Ú
138	8Ah	è	170	AAh	¬	202	CAh	╩	234	EAh	Û
139	8Bh	ï	171	ABh	½	203	CBh	╦	235	EBh	Ù
140	8Ch	î	172	ACh	¼	204	CCh	╠	236	ECh	ý
141	8Dh	ì	173	ADh	¡	205	CDh	═	237	EDh	Ý
142	8Eh	Ä	174	AEh	«	206	CEh	╬	238	EEh	¯
143	8Fh	Å	175	AFh	»	207	CFh	¤	239	EFh	´
144	90h	É	176	B0h	░	208	D0h	ð	240	F0h	
145	91h	æ	177	B1h	▒	209	D1h	Ð	241	F1h	±
146	92h	Æ	178	B2h	▓	210	D2h	Ê	242	F2h	‗
147	93h	ô	179	B3h	│	211	D3h	Ë	243	F3h	¾
148	94h	ö	180	B4h	┤	212	D4h	È	244	F4h	¶
149	95h	ò	181	B5h	Á	213	D5h	ı	245	F5h	§
150	96h	û	182	B6h	Â	214	D6h	Í	246	F6h	÷
151	97h	ù	183	B7h	À	215	D7h	Î	247	F7h	¸
152	98h	ÿ	184	B8h	©	216	D8h	Ï	248	F8h	°
153	99h	Ö	185	B9h	╣	217	D9h	┘	249	F9h	¨
154	9Ah	Ü	186	BAh	║	218	DAh	┌	250	FAh	·
155	9Bh	ø	187	BBh	╗	219	DBh	█	251	FBh	¹
156	9Ch	£	188	BCh	╝	220	DCh	▄	252	FCh	³
157	9Dh	Ø	189	BDh	¢	221	DDh	¦	253	FDh	²
158	9Eh	×	190	BEh	¥	222	DEh	▀	254	FEh	■
159	9Fh	ƒ	191	BFh	┐	223	DFh		255	FFh	

Ilustración 55. Tabla ASCII [19]

3- Cada carácter ASCII puede ser transformado a binario, mediante su codificación en 8 bits, lo que equivale a **2^8 = 256; 256-1 = 255** caracteres. A continuación, como el nombre indica "Base64" (**2^6 = 64**) viene dada por codificación a 6 bits, lo que equivale a 63 caracteres. Sabiendo esto se codificará la frase en ASCII-Binario (8 bits) a 6 bits.
4- Por último, se ha obtenido un nuevo número binario de 6 bits, así que se buscará su equivalente a Base64 en la siguiente tabla:

[17] La Wikipedia no suele ser una fuente fiable, pero se ha comprobado que los resultados fueran todos correctos con distintos tipos de frase mediante el codificador online [22]

Value	Char	Value	Char	Value	Char	Value	Char
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/

Ilustración 56. Tabla Base 64 [20]

- Ejemplo:

- Frase de entrada: "*Hola*"

Frase entrada			H	o	l	a
Equivalente ASCII			72	111	108	97
Binario 8bits			0100 1000	0110 1111	0110 1100	0110 0001
Binario 6bits	010 010	000 110	111 101	101 100	011 000	010 000
Decimal	18	6	61	44	24	16
Equivalente B64	S	G	9	s	Y	Q

- Frase codificada 64: "SG9sYQ"

Para llevar a cabo esta tarea mediante código, se utilizará una librería descargada desde GitHub [6]. Debido a que esta librería pide entradas array tipo '*char*', y la de temboo pide objetos '*String',* a lo largo del código habrá que hacer las siguientes transformaciones de formato:

1- Los datos se tomaran con variable tipo '*float',* de esta manera podrán incluir todos los decimales necesarios, y la toma de datos será precisa.
2- Se declarará también la frase de aviso de emergencia en objeto tipo '*String*', para que pueda ser usada directamente desde Temboo.
3- Las variables dato' *float'*, se pasaran a '*String'* para concatenar todos los datos en un mismo objeto.
4- Una vez concatenados los datos, se pasaran a variables '*char*', de esta forma podrán ser guardados en la SD, y podrán incluirse como entrada para la codificación a base 64.
5- Una vez codificados a base 64, se volverán a transformar en '*String'*, de esta manera estarán disponibles como entrada para Temboo.
6- Se suben los datos a Dropbox como '*String'.*

4.2.2. Código

Como ya dijimos para subir los datos y utilizar la plataforma Temboo, se deben haber obtenido unas credenciales que se guardan en la App de Temboo. Para que la placa tengo acceso a estas credenciales, se guardará junto a nuestro programa un código auxiliar con las credenciales de nuestra App temboo. Es decir, de la siguiente forma:

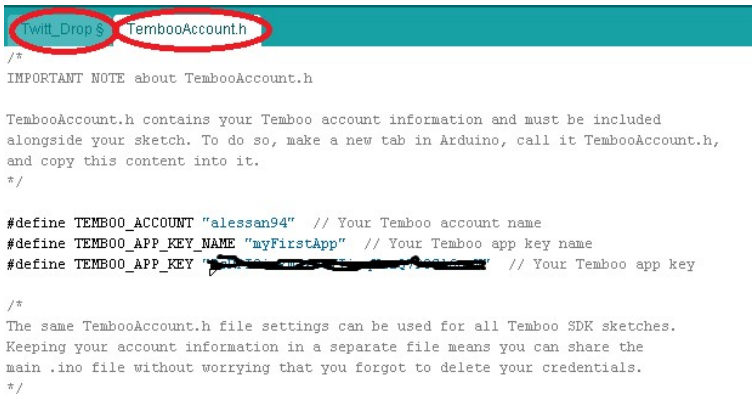


Ilustración 57. Llamamiento App Temboo

Una vez declarado el acceso a la App de Temboo, ya se puede ejecutar el código. Este código solo pretende subir dos datos a Dropbox, así como también, avisar vía twitter en el caso de que uno de los dos sensores de temperaturas extremadamente altas.

Código 4. Subida a Dropbox y aviso via Twitter.

```
//Por Alejandro Esteban Sanchis

//Librerias para Temboo
#include <Bridge.h>
#include <Temboo.h>
#include "TembooAccount.h" // contains Temboo account information, as described below
//Librerias para DS1307
#include <DS1307ART.h>
#include <Wire.h> //Necesita Wire.h para la comunicacion I2C
//Libreria B64
#include <Base64.h>

//Contructor reloj
DS1307ART External_Clock;  //Cuando se llame a una funcion del reloj se hara con este nombre
```

```
//Variables para la fecha-reloj
byte fecha[7];
byte Current_Minute;
byte Last_Minute;

//Variables para if's
int bucles = 0;
int ifgoto = 0;

//Variables contador
int maxCalls = 3;
int callID = 1;
int calls = 0;
int outputPin = 13;

//Variable dropbox chunck
String name;
String data;
String NextOffset;
String NextOffset_aux;
String UploadSessionID;
//Nombres
String nombrefolder = "/Twitt_Drop/";
String nombre;

//Variables Twitter
int sen_error = 0;
String twitt = "Se ha detectado un exceso de temperatura en el sensor: ";


//Datos
float Adata1_1 = 0;
float Adata1_2 = 0;
float temp1 = 0;
float temp2 = 0;

//Datos a subir
String conc;

void setup() {
  //pines
  pinMode(outputPin, OUTPUT);
  //begins
  Serial.begin(9600);
  Bridge.begin();
  Wire.begin();
}
```

```
void loop() {
  //Reinicia los objeto concatenados
  conc = "";
  nombre = "";

  //--------------------------------------------------------------------------------
  //Pedira hora al reloj, y cada minuto subira datos
  if (bucles > 0) { //Se quedara aqui hasta el siguiente minuto
    do { //haz esCurrent_Minuteto mientras ultimo monuto y el actual sean iguales, cuando cambia el actual, sale
      Serial.print("entro en espera last minute y actual minute:   ");
      Serial.print(Last_Minute);
      Serial.println(Current_Minute);
      External_Clock.getDateInt(fecha);
      Current_Minute = fecha[1];
      External_Clock.checkDate();
      delay(10000); //sobra es para que no marque tanto
    }
    while (Last_Minute == Current_Minute);
  }
  External_Clock.getDateInt(fecha);
  //concatenamos el nombre del archivo
  nombre.concat(nombrefolder);nombre.concat(fecha[4]);nombre.concat(":");nombre.concat(fecha[5]);
  nombre.concat(":20");nombre.concat(fecha[6]);nombre.concat(".txt");
  Serial.print("nombre=  ");Serial.println(nombre);
  //Terminamos de concatenar el nombre del archivo
  Last_Minute = fecha[1];
  Current_Minute = Last_Minute;
  //Concatenamos la fecha
  conc.concat(fecha[4]); conc.concat(":"); conc.concat(fecha[5]); conc.concat(":20");
  conc.concat(fecha[6]); conc.concat(";"); conc.concat(fecha[2]); conc.concat(":");
  conc.concat(fecha[1]); conc.concat(":"); conc.concat(fecha[0]); conc.concat(";");
  Serial.print("la fecha es:  "); Serial.println(conc);
  External_Clock.checkDate();


 //Luego leera datos del sensor.
 Adata1_1 = analogRead(4);
 Adata1_2 = analogRead(5);
 //Los pasamos a temperatura
 temp1 = ((Adata1_1 * 5000) / 1024) / 10;
 //Si es mayor de 100 acude a Twitter
 if (temp1 > 100) {
   sen_error = 1;
 }
 temp2 = ((Adata1_2 * 5000) / 1024) / 10;
 //Si es mayor de 100 acude a twitter
 if (temp2 > 100) {
   sen_error = 2;
 }
 //Los concatenamos
 conc.concat(" "); conc.concat(temp1); conc.concat(";"); conc.concat(temp2);
 conc.concat(";");conc.concat("\r\n");
 Serial.print("Los datos son:"); Serial.println(conc);
```

```
//Pasar a char para la transformacion a B64
char dato[conc.length() + 1]; //Declaro el char con la longitud justa IMP EL +1
conc.toCharArray(dato, conc.length() + 1); //muy IMP el lenght, sino no funciona.//IMP LOS +1

  //Codificacion a B64 para subir a Dropbox (Lo necesito en char)
NextOffset_aux = NextOffset; //MUY IMPORTANTE ESTA LINEA
int datoLen = sizeof(dato);
int encodedLen = base64_enc_len(datoLen);
char dato_encoded[encodedLen];
base64_encode(dato_encoded, dato, datoLen);

//Pasamos a String
String datostr(dato_encoded);
Serial.print("datostr =  ");
Serial.println(datostr);
NextOffset = NextOffset_aux; //MUY IMPORTANTE ESTA LINEA


 //Parte de comunicar con dropbox
 //Primero generamos la ID
 if (calls < callID) {
   Serial.println("Generando la ID");
   goto runChunkedUpload;
 }
 //Aqui empieza a concatenar datos en la ID
 if (calls >= callID && calls < maxCalls)  {
   Serial.println("Comienzo a concatenar datos");
   goto runChunkedUpload2;
 }
 //Se suben los datos concatenados a dropbox
 if (calls >= maxCalls) {
   Serial.println("Subo el concatenado de datos a dropbox");
   goto ChunkedUploadChoreo;
 }


  //----------------------------GOTO-->a partir de aqui solo funciones Temboo-----------
  if (ifgoto == 100) {  //ifs donde es imposible entrar si no es con el goto
runChunkedUpload:
    TembooChoreo ChunkedUploadChoreo;

    // Invoke the Temboo client
    ChunkedUploadChoreo.begin();

    // Set Temboo account credentials
    ChunkedUploadChoreo.setAccountName(TEMBOO_ACCOUNT);
    ChunkedUploadChoreo.setAppKeyName(TEMBOO_APP_KEY_NAME);
    ChunkedUploadChoreo.setAppKey(TEMBOO_APP_KEY);

    // Set Choreo inputs
    ChunkedUploadChoreo.addInput("AccessToken", "n5llzs4kvg2rtzic");
    ChunkedUploadChoreo.addInput("AppKey", "d0m28snmbknitf8");
    ChunkedUploadChoreo.addInput("AccessTokenSecret", "sshdwtwtmww6fly");
    ChunkedUploadChoreo.addInput("Chunk", datostr);
    ChunkedUploadChoreo.addInput("AppSecret", "qn7g33iv5l2i2hf");

    // Identify the Choreo to run
    ChunkedUploadChoreo.setChoreo("/Library/Dropbox/FilesAndMetadata/ChunkedUpload");
```

```
// Run the Choreo
unsigned int returnCode = ChunkedUploadChoreo.run();
Serial.print("choreo runl=   ");
Serial.println(returnCode);
if (returnCode != 0)  calls--;

// A return code of zero means everything worked
if (returnCode == 0) {
  while (ChunkedUploadChoreo.available()) {
    //Pone nombre a las salidas: NextOffset Expires UploadSessionID HTTP_CODE
    String name = ChunkedUploadChoreo.readStringUntil('\x1F');
    name.trim();
    //Serial.print("salida1=  "); Serial.print(name); Serial.print("     ");

    //Devuelve los datos que queremos
    String data = ChunkedUploadChoreo.readStringUntil('\x1E');
    data.trim();
    //Serial.print("salida2=  "); Serial.println(data);
    //Ahora hay que hacer que guarde los que nos interesan de data.
    if (name == "NextOffset") {
      NextOffset = data;
      Serial.print("NextOffset =   ");
      Serial.println(NextOffset);
    }
    if (name == "UploadSessionID") {
      UploadSessionID = data;
      Serial.print("UploadSessionID =   ");
      Serial.println(UploadSessionID);
    }


      //Pone el pin13 en alto si ha caducado la ID
      if (name == "Expires") {
        if (data == "") {
          digitalWrite(outputPin, HIGH);
        }
      }
    }
  }

  ChunkedUploadChoreo.close();
  calls++;
}
```

```
  if (ifgoto == 100) {
runChunkedUpload2:
    TembooChoreo ChunkedUploadChoreo;

    // Invoke the Temboo client
    ChunkedUploadChoreo.begin();

    // Set Temboo account credentials
    ChunkedUploadChoreo.setAccountName(TEMBOO_ACCOUNT);
    ChunkedUploadChoreo.setAppKeyName(TEMBOO_APP_KEY_NAME);
    ChunkedUploadChoreo.setAppKey(TEMBOO_APP_KEY);

    // Set Choreo inputs
    Serial.print("introduciomos inputs:  ");
    Serial.print(NextOffset);
    Serial.print("   ");
    Serial.println(UploadSessionID);
    ChunkedUploadChoreo.addInput("AccessToken", "n5llzs4kvg2rtzic");
    ChunkedUploadChoreo.addInput("AppKey", "d0m28snmbknitf8");
    ChunkedUploadChoreo.addInput("AccessTokenSecret", "sshdwtwtmww6fly");
    ChunkedUploadChoreo.addInput("Chunk", datostr); //con "blablabla\r\n" no falla
    ChunkedUploadChoreo.addInput("AppSecret", "qn7g33iv5l2i2hf");
    ChunkedUploadChoreo.addInput("UploadID", UploadSessionID);
    ChunkedUploadChoreo.addInput("Offset", NextOffset);

    // Identify the Choreo to run
    ChunkedUploadChoreo.setChoreo("/Library/Dropbox/FilesAndMetadata/ChunkedUpload");


   // Run the Choreo
   unsigned int returnCode = ChunkedUploadChoreo.run();
   Serial.print("returncode2 =   ");
   Serial.println(returnCode);
   if (returnCode != 0)  calls--;

   // A return code of zero means everything worked
   if (returnCode == 0) {
     while (ChunkedUploadChoreo.available()) {
       String name = ChunkedUploadChoreo.readStringUntil('\x1F');
       name.trim();
       //Serial.print("salidal=  ");Serial.print(name);Serial.print("     ");

       String data = ChunkedUploadChoreo.readStringUntil('\x1E');
       data.trim();
```

```
        if (name == "NextOffset") {
          NextOffset = data;
          Serial.print("NextOffset =   ");
          Serial.println(NextOffset);
        }
        if (name == "UploadSessionID") {
          UploadSessionID = data;
          Serial.print("UploadSessionID =   ");
          Serial.println(UploadSessionID);
        }

        if (name == "Expires") {
          if (data == "") {
            digitalWrite(outputPin, HIGH);
          }
        }
      }
    }

    ChunkedUploadChoreo.close();
    calls++;
  }


  if (ifgoto == 100) {
ChunkedUploadChoreo:
    TembooChoreo CommitChunkedUploadChoreo;

    // Invoke the Temboo client
    CommitChunkedUploadChoreo.begin();

    // Set Temboo account credentials
    CommitChunkedUploadChoreo.setAccountName(TEMBOO_ACCOUNT);
    CommitChunkedUploadChoreo.setAppKeyName(TEMBOO_APP_KEY_NAME);
    CommitChunkedUploadChoreo.setAppKey(TEMBOO_APP_KEY);

    // Set Choreo inputs
    CommitChunkedUploadChoreo.addInput("Path", nombre);
    CommitChunkedUploadChoreo.addInput("AccessToken", "n5llzs4kvg2rtzic");
    CommitChunkedUploadChoreo.addInput("AppKey", "d0m28snmbknitf8");
    CommitChunkedUploadChoreo.addInput("AccessTokenSecret", "sshdwtwtmww6fly");
    CommitChunkedUploadChoreo.addInput("AppSecret", "qn7g33iv5l2i2hf");
    CommitChunkedUploadChoreo.addInput("UploadID", UploadSessionID);

    // Identify the Choreo to run
    CommitChunkedUploadChoreo.setChoreo("/Library/Dropbox/FilesAndMetadata/CommitChunkedUpload");


    // Run the Choreo; when results are available, print them to serial
    unsigned int returnCode3 = CommitChunkedUploadChoreo.run();
    Serial.print("returncode3 =   ");
    Serial.println(returnCode3);


    while (CommitChunkedUploadChoreo.available()) {
      char c = CommitChunkedUploadChoreo.read();
    }
    CommitChunkedUploadChoreo.close();
    if (returnCode3 == 0) calls = 0;
  }
```

```
  //--------FIN GOTO

  if (sen_error != 0) {
    //concatenamos el twitter a subir
    twitt.concat(sen_error);
    Serial.print("twitt=   ");
    Serial.println(twitt);
    //Subimos un estado a twitter
    TembooChoreo StatusesUpdateChoreo;
    StatusesUpdateChoreo.begin();
    //credenciales de temboo.
    //Credenciales cuenta temboo.
    StatusesUpdateChoreo.setAccountName(TEMBOO_ACCOUNT);
    StatusesUpdateChoreo.setAppKeyName(TEMBOO_APP_KEY_NAME);
    StatusesUpdateChoreo.setAppKey(TEMBOO_APP_KEY);

    //Choreo inputs
    StatusesUpdateChoreo.addInput("AccessToken", "351013994l-GkndXVI0OHI3IrQiUNwZEHPzUXKA2Tsr3P2zNeC");
    StatusesUpdateChoreo.addInput("AccessTokenSecret", "jlQGBzX3AD5wui3P4ZAhA3abHudN2LSINoGQJpcQM6aum");
    StatusesUpdateChoreo.addInput("ConsumerKey", "YffihxevmS3YJqZ2D3caDEPeg");
    StatusesUpdateChoreo.addInput("ConsumerSecret", "SYfSKG2ew8f8pZYxmMAoxd2lvbS9zbxLGDxWGVYU7Cn7XYZKpv");
    //Tweet to notificate.
    StatusesUpdateChoreo.addInput("StatusUpdate", twitt);
    //identificar choreo to  run
    StatusesUpdateChoreo.setChoreo("/Library/Twitter/Tweets/StatusesUpdate");
    //run the choreo
    unsigned int returnCode4 = StatusesUpdateChoreo.run();
    Serial.print("returncode4=  ");
    Serial.println(returnCode4);
    StatusesUpdateChoreo.close();
    Serial.println("Se ha subido el tweet");
    //Volvemos a poner como que no hay ningun error y reiniciamos twitt
    twitt = "Se ha detectado un exceso de temperatura en el sensor: ";
    sen_error = 0;
  }

  //------------------------------------------------------------------------
  //FIN
  Serial.print("calls =  ");
  Serial.println(calls);
  bucles++;
}
```

4.3. Red Mesh

Código de diseño de la red mesh, con tres sensores, un repetidor y un maestro o coordinador.

Incluirá en el maestro, un almacenamiento en la SD, para verificar que los datos traspasados se almacenan correctamente.

- **Código Sensor 1**

Los sensores 2 y 3, tienen la misma estructura de programa, solo que cambiando los canales de escritura y lectura.

-S1: lectura=canal 1; escritura =canal 0.

-S2: lectura=canal 6; escritura=canal 5.

-S3: lectura=canal 3; escritura=canal 2 (Al repetidor).

Código 5. S1

```
// By Alejandro Esteban Sanchis


#include <SPI.h>
#include "nRF24L01.h"
#include "RF24.h"


//SPI bus pines 9 & 10
RF24 radio(9, 10);

// Topology
// Radio pipe addresses for the 2 nodes to communicate.
const uint64_t pipes[8] = { 0xF0F0F0F0E1LL, 0xF0F0F0F0D2LL, 0xF0F0F0F0D3LL, 0xF0F0F0F0D4LL, 0xF0F0F0F0D5LL,
0xF0F0F0F0D6LL, 0xF0F0F0F0D7LL, 0xF0F0F0F0D8LL }; //8canales de comunicacion, maximo puede usar 6 de ellos.

const unsigned int timeout_rwrite = 3000;

float temp;
float mV;
int PinTemp = 2;
bool check_r = false;
bool check_t = false;
bool a;
unsigned long T_Start;
unsigned long T_Now;
```

```
void setup() {

  pinMode(10, OUTPUT);
  Serial.begin(9600);

  radio.begin();

  // optionally, increase the delay between retries & # of retries
  radio.setRetries(15, 15);

  // optionally, reduce the payload size.  seems to
  // improve reliability
  //radio.setPayloadSize(8);

  // Open pipes to other nodes for communication
  //Declara por que canal habla y por cual escucha, se puede cambiar a lo largo del programa.
  radio.openWritingPipe(pipes[0]); //Escribe por el canal 0
  radio.openReadingPipe(1, pipes[1]); //Lee por el canal 1.
  Serial.println("comienza la transmision de datos");

}
```

```
void loop() {
  //Deja de escuchar el emisor para hablar
  radio.stopListening();
  mV = analogRead(PinTemp);
  Serial.print("mV =   ");Serial.println(mV);
  //temp = ((mV * 5000) / 1024) / 10;
  /*rango de -55 a 150°C 205=150+55*/
  temp = ((mV*500)/1023)   ;
  Serial.print("Enviando  ");

  // Envío de datos
  T_Start = millis();
  do {
    check_t = radio.write(&temp, sizeof(float)); //chequeo recepcion de dato
    T_Now = millis();
  } while ( check_t == false && T_Now - T_Start < timeout_rwrite);
  if(T_Now - T_Start >= timeout_rwrite) Serial.println("Se ha producido un timeout");
  Serial.print("ok, TIME:="); Serial.print(T_Now - T_Start); Serial.print(" Temp:="); Serial.println(temp);
  delay(20);
}
```

- Código Repetidor

Renvía el dato tomado por el sensor 3, hasta el Arduino Yun o maestro.

Código 6. Repetidor S3 a maestro (Yun)

```
//Por Alejandro Esteban Sanchis

#include <SPI.h>
#include "nRF24L01.h"
#include "RF24.h"

//SPI bus pines 9 y 10
RF24 radio(9, 10);

// Radio pipe addresses for the nodes to communicate.
const uint64_t pipes[8] = { 0xF0F0F0F0E1LL, 0xF0F0F0F0D2LL, 0xF0F0F0F0D3LL, 0xF0F0F0F0D4LL,
0xF0F0F0F0D5LL, 0xF0F0F0F0D6LL, 0xF0F0F0F0D7LL, 0xF0F0F0F0D8LL }; //8canales de comunicacion

const unsigned int timeout_rwrite = 3000;

float temp;
float mV;
float Adata;
int PinTemp = 2;
bool check_r = false;
bool check_t = false;
bool a;
unsigned long T_Start;
unsigned long T_Now;


void setup() {

  pinMode(10, OUTPUT);
  Serial.begin(9600);
  radio.begin();
  // optionally, increase the delay between retries & # of retries
  radio.setRetries(15, 15);
  Serial.println("comienza la transmision de datos repetirdor");

}
```

```
void loop() {
  //Primero escucha a 3
  radio.startListening();
  radio.openReadingPipe(1, pipes[2]); //Lee de 2 al S3
  T_Start = millis();
  do {
    if (radio.available()) {
      check_r = radio.read( &Adata, sizeof(float) );
    } else {
      check_r = false;
    }
    T_Now = millis();
  } while ( check_r == false && T_Now - T_Start < timeout_rwrite);
  if (T_Now - T_Start >= timeout_rwrite) Serial.println("Se ha producido un timeout");  //Timeout
  Serial.print("Dato Recibido, TIME:="); Serial.print(T_Now - T_Start); Serial.print(" Temp:="); Serial.println(Adata);
  radio.stopListening(); delay(100); //Para que carge la detencion

  //Ahora se prepara para escribir en el Yun
  radio.openWritingPipe(pipes[4]); //se prepara a mandar al yun los datos de 4
  T_Start = millis();
  do {
    check_t = radio.write(&Adata, sizeof(float));
    T_Now = millis();
  } while ( check_t == false && T_Now - T_Start < timeout_rwrite);
  if (T_Now - T_Start >= timeout_rwrite) Serial.println("Se ha producido un timeout");
  Serial.print("ok, TIME:="); Serial.print(T_Now - T_Start); Serial.print(" Temp enviada:="); Serial.println(Adata);
  delay(20);
}
```

- Código Maestro

Código 7. Maestro (Yun) solo con SD

```
// Por Alejandro Esteban Sanchis

//Librerias conversion
//#include <stdlib.h>
//Librerias SD
#include <Bridge.h>
#include <FileIO.h>
//Librerias Radio
#include <SPI.h>
#include "nRF24L01.h"
#include "RF24.h"
//Librerias DS1307
#include <DS1307ART.h>
#include <Wire.h> //Necesita Wire.h para la comunicacion I2C

//Contructor reloj
DS1307ART External_Clock;  //Cuando se llame a una funcion del reloj se hara con este nombre

RF24 radio(9, 10);

const uint64_t pipes[8] = { 0xF0F0F0F0E1LL, 0xF0F0F0F0D2LL, 0xF0F0F0F0D3LL, 0xF0F0F0F0D4LL,
0xF0F0F0F0D5LL, 0xF0F0F0F0D6LL, 0xF0F0F0F0D7LL, 0xF0F0F0F0D8LL }; //8canales de comunicacion
```

```
//Variables para guardar datos recibidos y leidos
float Adata1 = 0;
float Adata2=0;
float Adata3 = 0;
float Adata4 = 0;
//Variables timeout
const unsigned int timeout_rwrite = 3000;
unsigned long T_Start;
unsigned long T_Now;
//Variables chequeos
bool check_r = false; //Declarar inicialmente como 0.
bool check_t = false;

//Variables de reloj
byte fecha[7];
byte Current_Minute;
byte Last_Minute;

//Variables concatenado
String conc;

//Variable SD
char nameSD[] = "/mnt/sd/PruebaMESH.txt";

//Variables if's
int bucles = 0;

void setup() {
  //Inicializacion pines Radio
  pinMode(10, OUTPUT);
  //begins
  Serial.begin(9600);
  Bridge.begin();
  FileSystem.begin();
  Wire.begin();

  // Setup and configure rf radio
  radio.begin();
  // optionally, increase the delay between retries & # of retries
  radio.setRetries(15, 15);
  Serial.println("SetUp completado");
}

void loop() {
  //Primero reiniciamos el concatenado y datos
  conc = "";
  Adata1 = 0;
  Adata2 = 0;
  Adata3 = 0;
  Adata4 = 0;
```

```
//Ponemos el reloj
if (bucles > 0) { //Se quedara aqui hasta el siguiente minuto
  do { //haz esCurrent_Minuteto mientras ultimo monuto y el actual sean iguales, cuando cambia el actual, sale
    Serial.print("entro en espera last minute y actual minute:   ");
    Serial.print(Last_Minute); Serial.println(Current_Minute);
    External_Clock.getDateInt(fecha);
    Current_Minute = fecha[1];
    External_Clock.checkDate();
    delay(10000); //sobra es para que no marque tanto
  } while (Last_Minute == Current_Minute);
}
External_Clock.getDateInt(fecha);
Last_Minute = fecha[1];
Current_Minute = Last_Minute;
//Concatenamos la fecha
conc.concat(fecha[4]); conc.concat(":"); conc.concat(fecha[5]); conc.concat(":20");
conc.concat(fecha[6]); conc.concat(";"); conc.concat(fecha[2]); conc.concat(":");
conc.concat(fecha[1]); conc.concat(":"); conc.concat(fecha[0]); conc.concat(";");
Serial.print("la fecha es:  "); Serial.println(conc);
External_Clock.checkDate();

//Lee el dato del S1 y lo concatena
Serial.println("S1:");
radio.startListening(); //se inicia modo lectura
radio.openReadingPipe(1, pipes[0]); //Lee por el canal 0.
T_Start = millis();
do {
  if (radio.available()) {
    check_r = radio.read( &Adata1, sizeof(float) ); //si devuelve 1 el dato ha llegado, si devuelve 0 el dato no ha sido recibido.
  } else {
    check_r = false;
  }
  T_Now = millis();
} while ( check_r == false && T_Now - T_Start < timeout_rwrite);
//Timeout
if (T_Now - T_Start >= timeout_rwrite) Serial.println("Se ha producido un timeout");
Serial.print("Dato Recibido, TIME:="); Serial.print(T_Now - T_Start); Serial.print(" Temp:="); Serial.println(Adata1);
radio.stopListening();
delay(100); //Para que carge la detencion
conc.concat(" ");conc.concat(Adata1); conc.concat(";");


//Ahora lee su dato y lo concatena -->S2
Serial.println("S2:");
Serial.print("Adata2= ");Serial.println(analogRead(4));
/*rango de -55 a 150°C 205=150+55*/
Adata2 = ((analogRead(4) * 500) / 1023)      ;
conc.concat(Adata2); conc.concat(";");
```

```
  //Lee el dato del repetidor -->S3 + repetidor
  Serial.println("S3:");
  radio.startListening(); //se inicia modo lectura
  radio.openReadingPipe(1, pipes[4]); //Lee por el canal 0.
  T_Start = millis();
  do {
    if (radio.available()) {
      check_r = radio.read( &Adata3, sizeof(String) );
    } else {
      check_r = false;
    }
    T_Now = millis();
  } while ( check_r == false && T_Now - T_Start < timeout_rwrite);
  //Timeout
  if (T_Now - T_Start >= timeout_rwrite) Serial.println("Se ha producido un timeout");
  Serial.print("Dato Recibido, TIME:="); Serial.print(T_Now - T_Start); Serial.print(" Temp:="); Serial.println(Adata3);
  radio.stopListening();
  delay(100); //Para que carge la detencion
  conc.concat(Adata3); conc.concat(";");


  //Lee el dato S4 y lo concatena
  Serial.println("S4:");
  radio.startListening(); //se inicia modo lectura
  radio.openReadingPipe(1, pipes[5]); //Lee por el canal 0.
  T_Start = millis();
  do {
    if (radio.available()) {
      check_r = radio.read( &Adata4, sizeof(String) );
    } else {
      check_r = false;
    }
    T_Now = millis();
  } while ( check_r == false && T_Now - T_Start < timeout_rwrite);
  //Timeout
  if (T_Now - T_Start >= timeout_rwrite) Serial.println("Se ha producido un timeout");
  Serial.print("Dato Recibido, TIME:="); Serial.print(T_Now - T_Start); Serial.print(" Temp:="); Serial.println(Adata4);
  radio.stopListening();
  delay(100); //Para que carge la detencion
  conc.concat(Adata4); conc.concat(";");


  //Guardamos en la SD
  Serial.println("guardamos en la SD");
  Serial.print("Dato: ");Serial.println(conc);
  File  file = FileSystem.open(nameSD, FILE_APPEND);
  file.println(conc); //pongo solo print, porque conc ya tiene \r\n
  file.close();

  bucles++;

}
```

4.4. Programa final (Todo junto)

En este programa, tanto los sensores como el repetidor tendrán el mismo código reflejado anteriormente en el apartado "*4.4. Red mesh*". La diferencia, se encuentra en el código implementado en la placa maestro, ya que incluirá la subida de datos a Dropbox, y el aviso de emergencia vía Twitter., además de que no incluirá un número máximo de llamadas, sino que concatenara en una ID a lo largo del día, y cuando este acabe subirá todo a Dropbox.

Por otro lado, este programa será encargado de solventar pequeños errores de formato proporcionados por los códigos anteriores, ya que solo eran pruebas para un final ensamblaje de todas las partes en el código final mostrado a continuación. El nuevo código, seguirá el formato mostrado en el apartado "Protocolo para la creación de ficheros y su lectura por el programa ratón" reflejado en el "Pliego de condiciones/ Pliegos de otras entidades que afectan a este proyecto" para que los datos puedan ser transmitidos de Dropbox a la memoria del computador a posteriori.

Código 8. Maestro final

```
// Por Alejandro Esteban Sanchis

//Librerias conversion
#include <stdlib.h>
//Librerias SD
#include <Bridge.h>
#include <FileIO.h>
//Librerias Radio
#include <SPI.h>
#include "nRF24L01.h"
#include "RF24.h"
//Librerias DS1307
#include <DS1307ART.h>
#include <Wire.h> //Necesita Wire.h para la comunicacion I2C
//Librerias Temboo
#include <Temboo.h>
#include "TembooAccount.h" // Contains Temboo account information
//Libreria transformacion Base64
#include <Base64.h>

//Contructor reloj
DS1307ART External_Clock;  //Cuando se llame a una funcion del reloj se hara con este nombre

RF24 radio(9, 10);

const uint64_t pipes[8] = { 0xF0F0F0F0E1LL, 0xF0F0F0F0D2LL, 0xF0F0F0F0D3LL, 0xF0F0F0F0D4LL,
                            0xF0F0F0F0D5LL, 0xF0F0F0F0D6LL, 0xF0F0F0F0D7LL, 0xF0F0F0F0D8LL
                          }; //8canales de comunicacion
```

```
//Variables para guardar datos recibidos y leidos
float Adata1 = 0;
float Adata2 = 0;
float Adata3 = 0;
float Adata4 = 0;
//Variables timeout
const unsigned int timeout_rwrite = 3000;
unsigned long T_Start;
unsigned long T_Now;
//Variables chequeos
bool check_r = false; //Declarar inicialmente como 0.
bool check_t = false;

//Variables de reloj
byte fecha[7];
byte Current_Minute;
byte Last_Minute;


//Variables concatenado
String conc;

//Nombre ficheros SD y Drop
String nameSDstr = "/mnt/sd/"; //Fichero SD
char nameSDaux[] = "/mnt/sd/ENTREGA-DDHHAA.txt";
String nombre = "ENTREGA-";
String folderD = "ENTREGA-";

//Variables if's
int bucles = 0;
int goes = 0;

//Variables Temboo
String NextOffset;
String NextOffset_aux;
String UploadSessionID;
String name;


//Variables Twitter
int sen_error = 0;
String twitt = "Se ha detectado un exceso de temperatura en el sensor: ";

//Variables contador
int Check_Temboo = 0;

//Pines
int outputPin = 13;
```

```
void setup() {
  //Inicializacion pines Radio
  pinMode(10, OUTPUT);
  //begins
  Serial.begin(9600);
  Bridge.begin();
  FileSystem.begin();
  Wire.begin();

  // Setup and configure rf radio
  radio.begin();
  // optionally, increase the delay between retries & # of retries
  radio.setRetries(15, 15);
}


void loop() {
  //----------------Parte de inicialización-----------------------------------
  //Primero reiniciamos el concatenado y datos
  conc = "";
  nombre = "";
  folderD = "";
  nameSDstr = "";
  Adata1 = 0;
  Adata2 = 0;
  Adata3 = 0;
  Adata4 = 0;
  //Nombre SD y Dropbox, ajustarlo fuera del if
  String nameSDstr(nameSDaux);
  char nameSD[nameSDstr.length() + 1];
  nameSDstr.toCharArray(nameSD, nameSDstr.length() + 1);


 //----------------Parte del reloj-------------------------------------------
 //Ponemos el reloj
 if (bucles > 0) { //Se quedara aqui hasta el siguiente minuto
   do {
     External_Clock.getDateInt(fecha);
     Current_Minute = fecha[1];
     External_Clock.checkDate();
   } while (Last_Minute == Current_Minute);
 }
 External_Clock.getDateInt(fecha);
 Last_Minute = fecha[1];
 Current_Minute = Last_Minute;
 //Concatenamos la fecha
 conc.concat(fecha[4]); conc.concat(":"); conc.concat(fecha[5]);
```

```
//Guardamos la fecha en el nombre del fichero.txt
if (fecha[2] == 0 && fecha[1] == 0) { //Cada dia cambia el nombre del fichero
  nameSDstr.concat(conc); nameSDstr.concat(":"); nameSDstr.concat(fecha[6]);
  nameSDstr.concat(".txt");//Nombre para la SD completo
  //Pasarlo a char de nuevo para la SD --> nameSD[]
  char nameSD[nameSDstr.length() + 1];
  nameSDstr.toCharArray(nameSD, nameSDstr.length() + 1);
  //Nombre para el fichero Dropbox
  nombre.concat(conc); nombre.concat(":"); nombre.concat(fecha[6]); nombre.concat(".txt");
  //Nombre para carpeta Dropbox
  folderD.concat(fecha[5]); folderD.concat(fecha[6]); folderD.concat("/");
  folderD.concat(nombre);
}

conc.concat(":20"); conc.concat(fecha[6]); conc.concat(";");
if (fecha[2] >= 10) {
  conc.concat(fecha[2]); conc.concat(":");
}
if (fecha[2] < 10) {
  conc.concat("0"); conc.concat(fecha[2]); conc.concat(";");
}
if (fecha[1] >= 10) {
  conc.concat(fecha[1]); conc.concat(";");
}
if (fecha[1] < 10) {
  conc.concat("0"); conc.concat(fecha[1]); conc.concat(";");
}
External_Clock.checkDate();


//-------------------------Lectura y concatenacion de datos mesh-------
//Lee el dato del S1 y lo concatena
Serial.println("S1:");
radio.startListening(); //se inicia modo lectura
radio.openReadingPipe(1, pipes[0]); //Lee por el canal 0.
T_Start = millis();
do {
  if (radio.available()) {
    check_r = radio.read( &Adata1, sizeof(float) );
  } else {
    check_r = false;
  }
  T_Now = millis();
} while ( check_r == false && T_Now - T_Start < timeout_rwrite);

//Timeout
if (T_Now - T_Start >= timeout_rwrite) Serial.println("Se ha producido un timeout");
radio.stopListening();
delay(100); //Para que carge la detencion
conc.concat(" "); conc.concat(Adata1); conc.concat(";");
//Comprobacion que S1 no excede la temperatura
if (Adata1 > 100) {
  sen_error = 1;
}
```

```
//Ahora lee su dato y lo concatena -->S2
Adata2 = (((analogRead(4)) * 500 ) / 1023)     ;
conc.concat(Adata2); conc.concat(";");
//Comprobacion que S2 no excede la temperatura
if (Adata2 > 100) {
  sen_error = 2;
}


//Lee el dato del repetidor -->S3 + repetidor
Serial.println("S3:");
radio.startListening(); //se inicia modo lectura
radio.openReadingPipe(1, pipes[4]); //Lee por el canal 0.
T_Start = millis();
do {
  if (radio.available()) {
    check_r = radio.read( &Adata3, sizeof(String) );
  } else {
    check_r = false;
  }
  T_Now = millis();
} while ( check_r == false && T_Now - T_Start < timeout_rwrite);

//Timeout
if (T_Now - T_Start >= timeout_rwrite) Serial.println("Se ha producido un timeout");
radio.stopListening();
delay(100); //Para que carge la detencion
conc.concat(Adata3); conc.concat(";");
//Comprobacion que S1 no excede la temperatura
if (Adata3 > 100) {
  sen_error = 3;
}


//Lee el dato S4 y lo concatena
Serial.println("S4:");
radio.startListening(); //se inicia modo lectura
radio.openReadingPipe(1, pipes[5]); //Lee por el canal 0.
T_Start = millis();
do {
  if (radio.available()) {
    check_r = radio.read( &Adata4, sizeof(String) );
  } else {
    check_r = false;
  }
  T_Now = millis();
} while ( check_r == false && T_Now - T_Start < timeout_rwrite);
```

```
//Timeout
if (T_Now - T_Start >= timeout_rwrite) Serial.println("Se ha producido un timeout");
radio.stopListening();
delay(100); //Para que carge la detencion
conc.concat(Adata4); conc.concat(";");
conc.concat("\r\n");
//Comprobacion que S1 no excede la temperatura
if (Adata4 > 100) {
  sen_error = 4;
}


//----------------------Parte de guardar en la SD----------------------
//Guardamos en la SD
Serial.println("guardamos en la SD");
Serial.print("Dato: "); Serial.println(conc);
File  file = FileSystem.open(nameSD, FILE_APPEND);
file.print(conc); //pongo solo print, porque conc ya tiene \r\n
file.close();


//------------------------Parte de convertir a Base 64-----------------------------
//Pasamos a char para que pueda usarse la libreria B64
char dato_char[conc.length() + 1]; //Array un bit mas grande que String
conc.toCharArray(dato_char, conc.length() + 1); //Datos en char
//Convertimos a Base64
NextOffset_aux = NextOffset; //MUY IMPORTANTE ESTA LINEA PARA TEMBOO
int datoLen = sizeof(dato_char);
int encodedLen = base64_enc_len(datoLen);
char dato_encoded[encodedLen];
base64_encode(dato_encoded, dato_char, datoLen);
//Volvemos a pasar a String para que la libreria Temboo
String datostr(dato_encoded);
NextOffset = NextOffset_aux; //Recuperamos el valor original del Nextoffset

//---------------Comunicar con la Nube-------------------------------------
//Primero generar la ID, esto ocurre al inicio del dia a la hora 00
if (Check_Temboo == 0) {
  Serial.println("Generando la ID");
  goto runChunkedUpload;
}
//Concatenar datos a lo largo del día
if (Check_Temboo = ! 0)  {
  Serial.println("Comienzo a concatenar datos");
  goto runChunkedUpload2;
}
//Se suben los datos concatenados a dropbox
if ((Check_Temboo = ! 0) && (fecha[2] == 24) && (fecha[1] > 58)) {
  Serial.println("Subo el concatenado de datos a dropbox");
  if (fecha[2] == 0) Check_Temboo = 0; //Por si falla, no se quede atascado en esa ID
  goto ChunkedUploadChoreo;
}
```

```
  //----------------------GOTO TEMBOO--------------------------
  //Solicitar ID
  if (goes == 100) { //Condicion que nunca se cumple, solo puede entrar por el goto.
runChunkedUpload:
    //Para la primera subida sube el parrafo indicativo: DAY;MONTH;YEAR;...
    datostr = "DAY;MONTH;YEAR;HOUR;MINUTE;TEM01;TEM02;TEM03;TEM04";

    TembooChoreo ChunkedUploadChoreo;

    // Invoke the Temboo client
    ChunkedUploadChoreo.begin();


 // Set Choreo inputs
 ChunkedUploadChoreo.addInput("AccessToken", "n5l1zs4kvg2rtzic");
 ChunkedUploadChoreo.addInput("AppKey", "d0m28snmbknitf8");
 ChunkedUploadChoreo.addInput("AccessTokenSecret", "sshdwtwtmww6fly");
 ChunkedUploadChoreo.addInput("Chunk", datostr);
 ChunkedUploadChoreo.addInput("AppSecret", "qn7g33iv512i2hf");

 // Identify the Choreo to run
 ChunkedUploadChoreo.setChoreo("/Library/Dropbox/FilesAndMetadata/ChunkedUpload");


 // Run the Choreo
 unsigned int returnCode = ChunkedUploadChoreo.run();
 Serial.print("choreo runID=   ");Serial.println(returnCode);
 if (returnCode != 0)  Check_Temboo --;

 // A return code of zero means everything worked
 if (returnCode == 0) {
   while (ChunkedUploadChoreo.available()) {
     //Pone nombre a las salidas: NextOffset Expires UploadSessionID HTTP_CODE
     String name = ChunkedUploadChoreo.readStringUntil('\x1F');
     name.trim();


 //Devuelve los datos que queremos
 String data = ChunkedUploadChoreo.readStringUntil('\x1E');
 data.trim();
 //Ahora hay que hacer que guarde los que nos interesan de data.
 if (name == "NextOffset") {
   NextOffset = data;
   Serial.print("NextOffset =   "); Serial.println(NextOffset);
 }
 if (name == "UploadSessionID") {
   UploadSessionID = data;
   Serial.print("UploadSessionID =   "); Serial.println(UploadSessionID);
 }
```

```
        //Pone el pin13 en alto si ha caducado la ID
        if (name == "Expires") {
          if (data == "") {
            digitalWrite(outputPin, HIGH);
          }
        }
      }
    }

    ChunkedUploadChoreo.close();
    Check_Temboo ++;
  }

  //GOTO para concatenar en la ID o path
  if (goes == 100) {
runChunkedUpload2:
    TembooChoreo ChunkedUploadChoreo;

    // Invoke the Temboo client
    ChunkedUploadChoreo.begin();

    // Set Temboo account credentials
    ChunkedUploadChoreo.setAccountName(TEMBOO_ACCOUNT);
    ChunkedUploadChoreo.setAppKeyName(TEMBOO_APP_KEY_NAME);
    ChunkedUploadChoreo.setAppKey(TEMBOO_APP_KEY);

    // Set Choreo inputs
    ChunkedUploadChoreo.addInput("AccessToken", "n5l1zs4kvg2rtzic");
    ChunkedUploadChoreo.addInput("AppKey", "d0m28snmbknitf8");
    ChunkedUploadChoreo.addInput("AccessTokenSecret", "sshdwtwtmww6fly");
    ChunkedUploadChoreo.addInput("Chunk", datostr);
    ChunkedUploadChoreo.addInput("AppSecret", "qn7g33iv512i2hf");
    ChunkedUploadChoreo.addInput("UploadID", UploadSessionID);
    ChunkedUploadChoreo.addInput("Offset", NextOffset);

    // Identify the Choreo to run
    ChunkedUploadChoreo.setChoreo("/Library/Dropbox/FilesAndMetadata/ChunkedUpload");

  // Run the Choreo
  unsigned int returnCode = ChunkedUploadChoreo.run();
  Serial.print("returncode2 =    "); Serial.println(returnCode);

  // A return code of zero means everything worked
  if (returnCode == 0) {
    while (ChunkedUploadChoreo.available()) {
      String name = ChunkedUploadChoreo.readStringUntil('\x1F');
      name.trim();
```

```
    String data = ChunkedUploadChoreo.readStringUntil('\x1E');
    data.trim();
    //Sobreescribimos las variables que nos interesan.
    if (name == "NextOffset") {
      NextOffset = data;
      Serial.print("NextOffset =    "); Serial.println(NextOffset);
    }
    if (name == "UploadSessionID") {
      UploadSessionID = data;
      Serial.print("UploadSessionID =    "); Serial.println(UploadSessionID);
    }


        if (name == "Expires") {
          if (data == "") {
            digitalWrite(outputPin, HIGH);
          }
        }
      }
    }

    ChunkedUploadChoreo.close();
  }

  //GOTO para  subir los datos a Dropbox
  if (goes == 100) {
ChunkedUploadChoreo:
    TembooChoreo CommitChunkedUploadChoreo;

    // Invoke the Temboo client
    CommitChunkedUploadChoreo.begin();

    // Set Temboo account credentials
    CommitChunkedUploadChoreo.setAccountName(TEMBOO_ACCOUNT);
    CommitChunkedUploadChoreo.setAppKeyName(TEMBOO_APP_KEY_NAME);
    CommitChunkedUploadChoreo.setAppKey(TEMBOO_APP_KEY);


    // Set Choreo inputs
    CommitChunkedUploadChoreo.addInput("Path", folderD);
    CommitChunkedUploadChoreo.addInput("AccessToken", "n5l1zs4kvg2rtzic");
    CommitChunkedUploadChoreo.addInput("AppKey", "d0m28snmbknitf8");
    CommitChunkedUploadChoreo.addInput("AccessTokenSecret", "sshdwtwtmww6fly");
    CommitChunkedUploadChoreo.addInput("AppSecret", "qn7g33iv512i2hf");
    CommitChunkedUploadChoreo.addInput("UploadID", UploadSessionID);

    // Identify the Choreo to run
    CommitChunkedUploadChoreo.setChoreo("/Library/Dropbox/FilesAndMetadata/CommitChunkedUpload");
```

```
    // Run the Choreo; when results are available, print them to serial
    unsigned int returnCode3 = CommitChunkedUploadChoreo.run();
    Serial.print("returncode3 =    "); Serial.println(returnCode3);


    while (CommitChunkedUploadChoreo.available()) {
      char c = CommitChunkedUploadChoreo.read();
    }
    CommitChunkedUploadChoreo.close();
    if (returnCode3 == 0) Check_Temboo = 0;
  }

//-----------------------AVISO VÍA TWITTER----------------------------
//Subimos el estado de alerta a Twitter
if (sen_error != 0) {
  twitt.concat(sen_error);
  TembooChoreo StatusesUpdateChoreo;
  StatusesUpdateChoreo.begin();
  //Credenciales cuenta temboo.
  StatusesUpdateChoreo.setAccountName(TEMBOO_ACCOUNT);
  StatusesUpdateChoreo.setAppKeyName(TEMBOO_APP_KEY_NAME);
  StatusesUpdateChoreo.setAppKey(TEMBOO_APP_KEY);

  //Choreo inputs
  StatusesUpdateChoreo.addInput("AccessToken", "3510139941-GkndXVI00HI3IrQiUNwZEHPzUXKA2Tsr3P2zNeC");
  StatusesUpdateChoreo.addInput("AccessTokenSecret", "j1QGBzX3AD5wui3P4ZAhA3abHudN2LSINoGQJpcQM6aum");
  StatusesUpdateChoreo.addInput("ConsumerKey", "YffihxevmS3YJqZ2D3caDEPeg");
  StatusesUpdateChoreo.addInput("ConsumerSecret", "SYfSKG2ew8f8pZYxmMAoxd21vbS9zbxLGDxWGVYU7Cn7XYZKpv");
  //Tweet to notificate.
  StatusesUpdateChoreo.addInput("StatusUpdate", twitt);
  //identificar choreo to  run
  StatusesUpdateChoreo.setChoreo("/Library/Twitter/Tweets/StatusesUpdate");

  //run the choreo
  unsigned int returnCodeT = StatusesUpdateChoreo.run();
  Serial.print("returncodeT=  "); Serial.println(returnCodeT);
  StatusesUpdateChoreo.close();
  Serial.println("Se ha subido el tweet");
  //Volvemos a poner como que no hay ningun error y reiniciamos twitt
  twitt = "Se ha detectado un exceso de temperatura en el sensor: ";
  sen_error = 0;
}

 //----------------------FIN DE PROGRAMA----------------------------------------

 bucles++;
 datostr = "";
}
```

MANUAL DEL PROGRAMADOR Y USUARIO

En este apartado se explicara la tarea que llevan a cabo las distintas funciones de librerías utilizadas en este proyecto.

- **Temboo.h**

Esta librería viene incorporada en las últimas versiones de Arduino, y es la que permite comunicarse con la nube actuando de plataforma PHP o puente para la subida de datos a Dropbox y avisos de emergencia en Twitter.

Como se indica en el apartado "4.2. Plataforma Temboo" de la memoria. Esta librería incorpora gran cantidad de librerías capaces de comunicarse con distintas entidades web [9]. Sin embargo, en este proyecto se utilizaran 2: Una para comunicar nuestra plataforma con Dropbox, y otra para comunicarlo con Twitter o Google.

Para la comunicación con Dropbox se utilizarán las siguientes dos funciones:

- *Void ChunkedUpload:* Esta función se encarga del proceso de concatenar los datos o frases (tipo '*String' en base 64*) recibidos en una ID generada por la propia función. Proceso que requiere la realización de varias llamadas repetidas.
  En la primera llamada solo habrá que proporcionar las credenciales de las Apps Dropbox y Temboo. La función proporcionara una ID y subirá el dato a ella.
  Cuando se vuelva a llamar a la función, se debe introducir como dato de entrada la ID y la posición en la que se desea concatenar el siguiente dato.
- *Void CommitChunkedUpload*: Esta función recibe como entrada las credenciales de Dropbox, Temboo, una ID y un nombre de fichero. A continuación la información contenida en esa ID es subida a Dropbox en el fichero con denominación homónima (En caso de no existir ese fichero, lo genera automáticamente).

Para la comunicación con Twitter se utilizara la siguiente función:

- *Void StatusesUpdate:* Esta función sube un estado a Twitter, de esta forma el propio usuario y sus seguidores pueden ser alertados del fallo de algún sensor.
  Como datos de entrada solicita las credenciales de las Apps Twitter y Temboo, así como el estado que se desea subir (tipo *String*).
  Una vez recibidas las entradas se ejecuta y sube el estado a Twitter.

- **TembooAccount.h**

Esta librería únicamente contiene la cuenta y las claves de acceso para la cuenta Temboo.

- **Base64.h**

Como se ha mencionado en los anexos "4.3.1 Encriptación Base 64", los datos tipo '*String'* que desean ser subidos a Dropbox, necesitan estar codificados en Base 64. Por ello, esta librería recibe una entrada de datos tipo '*char'* y la codifica o descodifica.

Como en este proyecto únicamente se requiere codificar, no descodificar, solo se utilizara la siguiente función:

- *Void base64_encode(dato_encoded, dato, datoLen):* Esta función recibe, la frase a codificar (dato), su longitud medida con la función '*sizeof*()' y a cambio da como salida la frase o dato codificado (*dato_encoded*).

Es conveniente señalar que en esta librería, todas las entradas son de tipo '*char'*, por lo que al ser declarado el array *dato_encoded,* se debe conocer su longitud. Esta longitud será determinada mediante la siguiente función:

- *Int base64_enc_len(datoLen):* Esta función recibe como entrada el tamaño de la función sin codificar (dado por tabla ASCII, 8 bits) y por medio de cálculo de numero de bits (tabla base 64, 6 bits) devuelve el tamaño de la frase o dato codificado.

- **Bridge.h**

No se utiliza ninguna función contenida en este librería salvo su inicialización "Bridge.begin()", sin embargo es muy importante ya que relaciona y comunica los dos procesadores incluidos en el Arduino Yun: ATmega32U4 y Atheros 9331. El procesador Atheros, es el encargado de la comunicación con el módulo Wifi, puerto Ethernet y la abertura para la comunicación con puertos externos como la SD. Por ello esta librería será necesario incluirla para el correcto funcionamiento de '*Temboo*.h' y '*FileIO*.h'

- **FileIO.h**

Es la librería encargada de la comunicación con la SD, es capaz tanto de leer y escribir, como de borrar y desplazar ficheros dentro de ella. Sin embargo para nuestro proyecto se utilizará únicamente tres funciones: una para abrir o crear un fichero, otra para guardar datos, y otra para cerrar el fichero.

Lo primero de todo, para que esta librería funcione correctamente, hay que incluir la librería '*Bridge*.h' para generar la clase o constructor '*FileIO*' que marcara el inicio de la comunicación con la SD mediante la función '*FileSystem.begin()'.*

Una vez hecho esto, nuestro dispositivo estará listo para llevar a cabo las funciones de la librería.

- *File file = FileSystem.open(nameSD, FILE_APPEND):* Esta función lleva a cabo tres tareas:
  - Inicialización de la interacción con el almacenamiento de la SD.
  - Generar o abrir un fichero con el nombre introducido en la variable '[18]*nameSD*' (tipo '*char*').
  - Genera un objeto (file) necesario para interactuar con los datos de ese fichero.

- *Void file.print(dato):* Guarda una variable '*String*' en la SD (dato), sin salto de línea.

- *Void file.println(dato):* Guarda una variable '*String*' en la SD (dato), con salto de línea.

- *Void file.write(dato):* Guarda una variable '*char*' en la SD (dato), sin salto de línea.

- *Void file.close():* Cierra el fichero donde se guardan los datos.

- **Wire.h**

Permite la comunicación I_2C de Arduino necesaria para la comunicación con el DS1307. Es necesario inicializar el protocolo de comunicación I_2C para que funcione adecuadamente mediante la función "*wire.begin()".*

[18] Recordar que el nombre del archivo incluido en esta variable, debe ser nombrado con el prefijo "/mnt/sd/nombre.txt"

- **DS1307ART.h**

Es la librería necesaria para el uso del reloj DS1307. Contiene diversas funciones, pero solo se utilizarán las necesarias para solicitar la hora y sincronizar el reloj.

Previamente, para el correcto funcionamiento de la librería, se inicializará el constructor con la función "DS1307ART constructor" y activar la comunicación I_2C.

- *Void Constructor.getDateInt(fecha):* Solicita la hora y fecha al reloj y la guarda en un vector (fecha) compuesto por: fecha[0;6] tamaño 13→ segundos, minutos, hora, día de la semana, día, mes y año.
- *Void Constructor.setDateInt(fecha):* Sirve para cambiar o actualizar la fecha y hora del buffer del reloj, para que cuente desde ese segundo con ayuda del cuarzo. Se le envía un vector de entrada fecha de tamaño 13 con mismo formato que el anterior: segundos, minutos, hora, día de la semana, día, mes y año (dos dígitos).
- *Void Constructor.checkDate():* Sirve la para validar la fecha, debe ser llamada cuando se acabe de solicitar las prestaciones del reloj.

- **SPI.h**

Se utiliza para que Arduino comparta información con los circuitos integrados dentro de la placa NanoIOShield, activando el bus de datos SPI (Serial Peripheral Interface). De esta forma se permite que la comunicación y transmisión de datos entre módulos Arduino y el módulo nRF24L01.

Esta comunicación se lleva a cabo a través de los pines MOSI, MISI y SCK de ahí la importancia de las conexiones realizadas en el apartado "*4.3.3 nRF24L01*" al conectarse en el puerto ISCP del Arduino Yun.

- **RF24.h**

Para el correcto funcionamiento de esta librería, es necesario incluir la librería 'SPI.h', ya que como se ha mencionado, es necesaria para la comunicación entre Arduino y el módulo nRF24L01.

También, se requiere inicializar los pines CSN y CE, ya que dirigen el protocolo de comunicación y pautas, como se ha explicado en el apartado "*4.3.3. nRF24L01*", por medio de la siguiente función: '*void RF24 radio (PIN CE, PIN CSN)*'; los canales de comunicación o pipes mediante un vector de enteros constantes de 64bits hexadecimal ('*const uint64_t pipes[tamaño]'); y por ultimo inicializar también la radio mediante la función 'radio.begin()', la cual coloca el pin CE en alto.*

Para llevar a cabo la comunicación inalámbrica, esta librería incluye las siguientes funciones:

- *Void radio.startListening():* Habilita la radio o nRF24L01 para escuchar (PRIM_RX=1)información, mientras este escuchando no podrá enviar información por ningún canal.

- *Void radio.openReadingPipe(1, pipes[0]):* Indica el canal o pipe, por el que va a escuchar o recibir información. Mientras este escuchando este canal, no podrá atender uno distinto.

- *Boolean radio.read(&Adata, sizeof(float)):* Recibe la información del canal seleccionado en 'radio.openReadingPipe(1,pipe[])' y la guarda en la variable por referencia (Adata). Para almacenar correctamente en dato, se le ha de especificar como entrada el tamaño del mismo.
  Si la transmisión de datos ha sido exitosa, devuelve un 1, el dato ha sido enviado correctamente, por el contrario un 0 significaría que ha habido algún error en la transmisión.

- *Void radio.stopListening():* Coloca el PRIM_RX en 0 de esta forma vuelve a la posición *standby* a la espera de la siguiente instrucción.

- *Void radio.openWritingPipe(pipes[4]):* Para poder llevar a cabo esta función, el pin PRIM_RX debe estar en bajo, de esta forma se colocara el PRIM_TX en alto y la radio se dispondrá a transmitir información por el canal indicado (pipe[]).

- *Boolean radio.write(&Adata, sizeof(float)):* Trasmite la información contenida en la variable por referencia Adata, por el canal indicado en 'radio.openWritingPipe(pipe[])'.
  Devuelve un 1 si la transmisión se ha llevado a cabo exitosamente, si no es así, es que ha habido algún error y devuelve un 0.

PLIEGO DE CONDICIONES

Contenido

A lo largo del presente documento, se recoge la disposición de tareas para las que este proyecto ha sido diseñado, los materiales utilizados para su realización, el equipamiento, costes y sus especificaciones técnicas. Sin embargo, en este apartado se hará referencia a toda la normativa que envuelve proyecto.

La tarea que desempeña este prototipo, hay que recordar que no pretende una instalación, ni es el datalogger definitivo, sino que es un primer prototipo de datalogger con comunicación vía nube dando viabilidad a futuros proyectos que lo mejoren. Estas mejoras fueron explicadas con más detalle en el apartado "6. *Futuras mejoras*".

En caso de alterar alguno de los pasos o especificaciones dadas a lo largo de este proyecto sin el consentimiento del realizador o de algún coordinador, no supondrán de responsabilidad alguna por un incorrecto funcionamiento o problemas que puedan surgir.

Nomativa

- UNE-EN 61010-031: Medidas de seguridad, para el uso de equipamiento electrónico frente a personas como material de laboratorio.
- EN 61190-1-1: Requerimientos para la comprobación y realización de una correcta soldadura para la unión de materiales o dispositivos.
- UNE-EN 62008: Calibración de aparatos dedicados a la adquisición de datos.
- UNE-EN 61187: Documentación acerca del equipamiento electrónico utilizado.
- STD-003 : Test de comprobación de soldabilidad

Pliegos de otras entidades que afectan a este proyecto

Temboo

En primer lugar, debe cumplirse la normativa impuesta por la plataforma PHP Temboo [21]

Como ya se ha citado, en este prototipo se está empleando la versión gratuita de Temboo, limitándose al número de envíos (10000), tanto a la cantidad de datos enviados (1GB) cada mes.

PROTOCOLO PARA LA CREACION DE FICHEROS Y SU LECTURA POR EL PROGRAMA "RATÓN"

Este protocolo ha sido diseñado por el colaborador Ángel Fernández Navajas, como se ha explicado en el apartado "4.4. Programa auxiliar Ratón", restringe el formato en el que han de ser subidos los datos a la nube, para así de forma automática descargarlos de Dropbox y guardarlos en la memoria interna del ordenador donde esté instalado Ratón. Dicho protocolo es el siguiente:

1. Arduino, o cualquier otro procesador que se utilice para la toma de datos, creará un fichero cada mes, donde guardará los datos tomados entre el día 1 y el 31 del mes en curso.

El fichero será en formato texto, con datos separados por comas o por punto y coma (con preferencia a punto y coma) y con un punto para la separación de decimales. También podrá estar en formato base64.

Este fichero lo nombrará con el siguiente formato: ABCDEFGHAAMM.TXT

Donde ABCDEFGH (los 8 primeros caracteres) serán descriptivos del nombre del proyecto y siempre deberán ser 8 caracteres. En caso de que el nombre no utilice los 8 caracteres, se rellenará con el símbolo – (guion normal).

- AA será el año con dos dígitos.
- MM será el mes con dos dígitos.

2. Cada día, Arduino generará un fichero de datos que será una parte del fichero del mes, con los datos del día en curso, desde las 00:00 hasta las 23:59 horas.

Este fichero será el que envíe Arduino para ser leído por el programa.

Este fichero tendrá el siguiente formato: ABCDEFGHAAMMDD.TXT

Donde los caracteres significan lo mismo que en el punto 1 con el añadido del día, con dos dígitos, al final del nombre.

3. Junto con el fichero anterior, se enviará un fichero de formato, con el mismo nombre que el fichero TXT, pero con extensión FMT (de formato), que contendrá información de qué es cada una de las columnas del fichero de texto. Las cinco primeras columnas siempre deberán ser las variables de fecha/hora en idioma inglés, es decir YEAR, MONTH, DAY, HOUR y MINUTE. A partir de la 6ª columna, deberán ir los nombres descriptivos de los sensores, por ejemplo TEM001, HUM002, etc. Como mínimo, un fichero de formato, deberá llevar 6 columnas.

El orden de las cinco primeras columnas puede cambiar entre sí, por ejemplo, podría ser: **HOUR;MINUTE;DAY;MONTH;YEAR;**TEM001;TEM002;HUM001;HUM002

Pero NO podría ser: **HOUR;MINUTE;DAY;**TEM001**;MONTH;YEAR;**TEM002;HUM001;HUM002 (la columna 4 debe ser una variable de fecha/hora y en este caso es del dato del sensor TEM001)

BIBLIOGRAFÍA

[1] F. García-Diego, «digital,» bitstream, 09 2013. [En línea]. Available: http://digital.csic.es/bitstream/10261/84102/1/casa%20de%20Ariadna.pdf. [Último acceso: 01 07 2016].

[2] Sensors, «MDPI,» Sensors, 2015. [En línea]. Available: http://www.mdpi.com/1424-8220/15/4/7246. [Último acceso: 12 06 2016].

[3] «maswifi,» maswifi, [En línea]. Available: http://www.maswifi.com/blog/2012/05/redes-mesh-que-son-y-como-funcionan/. [Último acceso: 14 06 2016].

[4] «ArduinoHome,» Arduino, [En línea]. Available: http://www.arduino.cc. [Último acceso: 12 06 2016].

[5] «Introduccion Arduino,» Arduino, [En línea]. Available: https://www.arduino.cc/en/Guide/Introduction. [Último acceso: 12 06 2016].

[6] «Librerias Arduino,» Arduino, [En línea]. Available: https://www.arduino.cc/en/Reference/Libreries. [Último acceso: 12 06 2016].

[7] «GitHub,» [En línea]. Available: https://github.com/. [Último acceso: 12 06 2016].

[8] «Indiamart,» bombay electronics, [En línea]. Available: http://www.indiamart.com/bombayelectronics-mumbai/wireless-modules.html. [Último acceso: 14 06 2016].

[9] «Temboo,» Temboo, [En línea]. Available: https://temboo.com. [Último acceso: 12 06 2016].

[10] «Temboo Library,» Temboo, [En línea]. Available: https://temboo.com/library/. [Último acceso: 12 06 2016].

[11] «ArduinoBoardYun,» Arduino, [En línea]. Available: https://www.arduino.cc/en/Main/ArduinoBoardYUN. [Último acceso: 12 06 2016].

[12] J. Dourlens, «Jules.Doulens,» [En línea]. Available: http://jules.dourlens.com/arduino-yun-2-2-tft-display-ili9340/. [Último acceso: 14 06 2016].

[13] Arduino, «Arduno Nano,» Arduino, [En línea]. Available: https://www.arduino.cc/en/Main/ArduinoBoardNano. [Último acceso: 12 06 2016].

[14] N. SEMICONDUCTOR, «sparkfun,» [En línea]. Available: https://www.sparkfun.com/datasheets/Components/SMD/nRF24L01Pluss_Preliminary_Product_Specification_v1_0.pdf. [Último acceso: 12 06 2016].

[15] D. Semiconductor, «sparkfun,» Dallas Semiconductor, [En línea]. Available: https://www.sparkfun.com/datasheets/Components/DS1307.pdf. [Último acceso: 12 06 2016].

[16] arubia45, «blogspot,» 29 01 2013. [En línea]. Available: http://arubia45.blogspot.com.es/2013/01/reloj-rtc-ds1307-arduino.html. [Último acceso: 14 06 2016].

[17] «clipart.me,» [En línea]. Available: http://es.clipart.me/free-vector/sketch. [Último acceso: 14 06 2016].

[18] Vinculum, «ftdichip,» Vinculum, 31 05 2010. [En línea]. Available: http://www.ftdichip.com/Support/Documents/DataSheets/Modules/DS_VDIP1.pdf. [Último acceso: 12 06 2016].

[19] «Wikipedia,» Wikipedia, 02 02 2016. [En línea]. Available: https://es.wikipedia.org/wiki/Base64. [Último acceso: 13 06 2016].

[20] «Codigo ASCII,» [En línea]. Available: http://www.elcodigoascii.com.ar/. [Último acceso: 13 06 2016].

[21] «Ortizol,» Blogspot, [En línea]. Available: http://ortizol.blogspot.com.es/2014/05/receta-no-2-4-en-c-codificacion-de.html. [Último acceso: 13 06 2016].

[22] «Temboo,» Temboo, [En línea]. Available: https://temboo.com/terms. [Último acceso: 16 06 2016].

[23] «Base64 Encode and decode,» [En línea]. Available: https://www.base64encode.org/. [Último acceso: 13 06 2016].

[24] Silicio.MX, «silicio.MX,» [En línea]. Available: http://silicio.mx/antena-pcb-xbee-s1-802-15-4. [Último acceso: 14 06 2016].

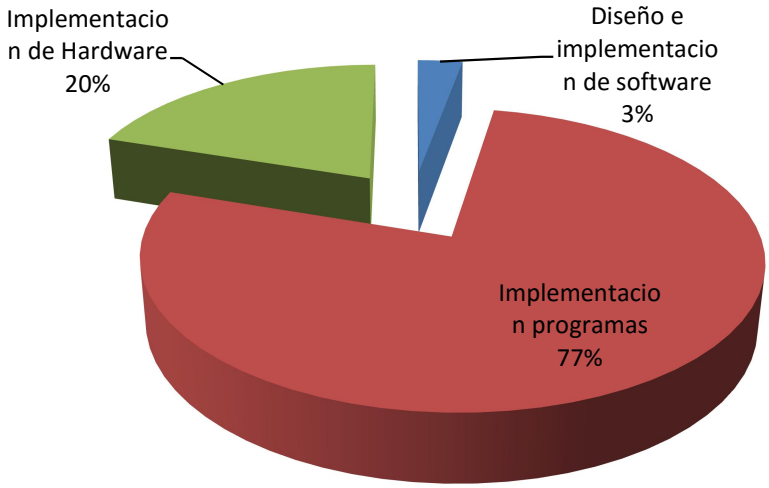
[25] «Instructables,» Instructables, [En línea]. Available: http://www.instructables.com/id/LM35-Temperature-Sensor/. [Último acceso: 14 06 2016].

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO DE FIN DE GRADO EN INGENIERIA EN TECNOLOGÍAS INDUSTRIALES

PRESUPUESTO



AUTOR: ESTEBAN SANCHIS, ALEJANDRO

TUTOR: GARCÍA-DIEGO, FERNANDO J.

Curso Académico: 2015-2016

Justificación de precios. Unidades de obra

Código	Cantidad	Ud	Descripción	Precio	Subtotal	Importe
Capítulo: 01			**IMPLEMENTACION Y DESARROLLO DE UNA RED DE SENSORES INALAMBRICA CON ALMACENAMIENTO DE LOS DATOS EN LA NUBE**			
Capítulo: 01.01			**Diseño e implementacion de software**			
01.01.01			**Diseño e implementacion software**			
ud						
	8,000	ud	Librerias	0,00		
	3,000	h	Computador	0,13	0,39	
	10,000	h	Graduado Ingenieria Industrial	35,00	350,00	
	1,000	ud	Software Arduino	0,00		
	0,020		Costes directos complementarios	350,39	7,01	
			Clase Mano de Obra			350,00
			Clase Maquinaria			0,39
			Clase Medio auxiliar			7,01
			Med. aux. y Resto obra			
			Total partida			**357,40**

Asciende el precio total a la expresada cantidad de: TRESCIENTOS CINCUENTA Y SIETE EUROS CON CUARENTA CÉNTIMOS

Capítulo: 01.02			**Implementacion programas**			
01.02.01			**Guardar en la SD**			
P2						
	25,000	h	Computador	0,13	3,25	
	30,000	h	Graduado Ingenieria Industrial	35,00	1.050,00	
	1,000	ud	Software Arduino	0,00		
	20,000	h	Conexion red placas para ensayo	0,10	2,00	
	0,020		Costes directos complementarios	1.055,25	21,11	
			Clase Mano de Obra			1.050,00
			Clase Maquinaria			5,25
			Clase Medio auxiliar			21,11
			Med. aux. y Resto obra			
			Total partida			**1.076,36**

Asciende el precio total a la expresada cantidad de: MIL SETENTA Y SEIS EUROS CON TREINTA Y SEIS CÉNTIMOS

01.02.02			**Subida datos a Dropbox y aviso via Twitter**			
P3						
	40,000	h	Computador	0,13	5,20	
	45,000	h	Graduado Ingenieria Industrial	35,00	1.575,00	
	1,000	ud	Software Arduino	0,00		
	35,000	h	Conexion red placas para ensayo	0,10	3,50	
	1,000	ud	Plataforma Temboo version gratuita	0,00		
	0,020		Costes directos complementarios	1.583,70	31,67	
			Clase Mano de Obra			1.575,00
			Clase Maquinaria			8,70
			Clase Medio auxiliar			31,67
			Med. aux. y Resto obra			
			Total partida			**1.615,37**

Asciende el precio total a la expresada cantidad de: MIL SEISCIENTOS QUINCE EUROS CON TREINTA Y SIETE CÉNTIMOS

Justificación de precios. Unidades de obra

Código	Cantidad	Ud	Descripción	Precio	Subtotal	Importe
Capítulo: 01.02.03			**Red Mesh**			
01.02.03.01			**Sensor**			
P4.1						
	10,000	h	Computador	0,13	1,30	
	10,000	h	Graduado Ingenieria Industrial	35,00	350,00	
	1,000	ud	Software Arduino	0,00		
	10,000	h	Conexion red placas para ensayo	0,10	1,00	
	0,020		Costes directos complementarios	352,30	7,05	
			Clase Mano de Obra			350,00
			Clase Maquinaria			2,30
			Clase Medio auxiliar			7,05
			Med. aux. y Resto obra			
			Total partida			**359,35**
			Asciende el precio total a la expresada cantidad de: TRESCIENTOS CINCUENTA Y NUEVE EUROS CON TREINTA Y CINCO CÉNTIMOS			
01.02.03.02			**Repetidor S3 a Maestro-Yun**			
P4.4						
	15,000	h	Computador	0,13	1,95	
	15,000	h	Graduado Ingenieria Industrial	35,00	525,00	
	1,000	ud	Software Arduino	0,00		
	15,000	h	Conexion red placas para ensayo	0,10	1,50	
	0,020		Costes directos complementarios	528,45	10,57	
			Clase Mano de Obra			525,00
			Clase Maquinaria			3,45
			Clase Medio auxiliar			10,57
			Med. aux. y Resto obra			
			Total partida			**539,02**
			Asciende el precio total a la expresada cantidad de: QUINIENTOS TREINTA Y NUEVE EUROS CON DOS CÉNTIMOS			
01.02.03.03			**Maestro - Yun**			
P4.5						
	50,000	h	Computador	0,13	6,50	
	50,000	h	Graduado Ingenieria Industrial	35,00	1.750,00	
	1,000	ud	Software Arduino	0,00		
	20,000	h	Conexion red placas para ensayo	0,10	2,00	
	0,020		Costes directos complementarios	1.758,50	35,17	
			Clase Mano de Obra			1.750,00
			Clase Maquinaria			8,50
			Clase Medio auxiliar			35,17
			Med. aux. y Resto obra			
			Total partida			**1.793,67**
			Asciende el precio total a la expresada cantidad de: MIL SETECIENTOS NOVENTA Y TRES EUROS CON SESENTA Y SIETE CÉNTIMOS			
01.02.04			**Maestro completo**			
P5						
			Incluye todos los programas anteriores, agrupados en uno.			
	80,000	h	Computador	0,13	10,40	
	100,000	h	Graduado Ingenieria Industrial	35,00	3.500,00	
	1,000	ud	Software Arduino	0,00		
	40,000	h	Conexion red placas para ensayo	0,10	4,00	
	0,020		Costes directos complementarios	3.514,40	70,29	
			Clase Mano de Obra			3.500,00
			Clase Maquinaria			14,40
			Clase Medio auxiliar			70,29
			Med. aux. y Resto obra			
			Total partida			**3.584,69**
			Asciende el precio total a la expresada cantidad de: TRES MIL QUINIENTOS OCHENTA Y CUATRO EUROS CON SESENTA Y NUEVE CÉNTIMOS			

Justificación de precios. Unidades de obra

Código	Cantidad	Ud	Descripción	Precio	Subtotal	Importe

Capítulo: 01.03 Implementacion de Hardware

01.03.01 **Placa Maestro - Yun**

Plac1

1,000	ud	Dispositivo nRF24L01 (Radio)	2,50	2,50
1,000	u	Sensor temperatura LM35	1,35	1,35
26,000	h	Soldador	0,20	5,20
30,000	h	Graduado Ingenieria Industrial	35,00	1.050,00
22,000	h	Cordinador proyecto	30,00	660,00
1,000	ud	Dispositivo DS1307 o reloj	2,09	2,09
1,000	ud	Cristal 32768 KHz	0,18	0,18
1,000	ud	Pila Litio CR1220 marca Duracel	1,30	1,30
1,000	ud	micro SD card 2Gb marca Transcend	4,85	4,85
1,000	ud	Módulo Arduino Yun	66,30	66,30
0,020		Costes directos complementarios	1.793,77	35,88

Clase Mano de Obra	1.710,00
Clase Maquinaria	5,20
Clase Material	78,57
Clase Medio auxiliar	35,88
Med. aux. y Resto obra	
Total partida	**1.829,65**

Asciende el precio total a la expresada cantidad de: MIL OCHOCIENTOS VEINTINUEVE EUROS CON SESENTA Y CINCO CÉNTIMOS

01.03.02 **Placa repetidor**

Plac2

1,000	ud	Dispositivo nRF24L01 (Radio)	2,50	2,50
1,000	ud	Placa NanoIOShield	5,99	5,99
1,000	ud	Modulo Arduino Nano	6,85	6,85
5,000	h	Graduado Ingenieria Industrial	35,00	175,00
0,020		Costes directos complementarios	190,34	3,81

Clase Mano de Obra	175,00
Clase Material	15,34
Clase Medio auxiliar	3,81
Med. aux. y Resto obra	
Total partida	**194,15**

Asciende el precio total a la expresada cantidad de: CIENTO NOVENTA Y CUATRO EUROS CON QUINCE CÉNTIMOS

01.03.03 **Placa Sensor**

Plac3

1,000	ud	Dispositivo nRF24L01 (Radio)	2,50	2,50
1,000	ud	Placa NanoIOShield	5,99	5,99
1,000	ud	Modulo Arduino Nano	6,85	6,85
1,000	u	Sensor temperatura LM35	1,35	1,35
5,000	h	Graduado Ingenieria Industrial	35,00	175,00
4,000	h	Soldador	0,20	0,80
2,000	h	Cordinador proyecto	30,00	60,00
0,020		Costes directos complementarios	252,49	5,05

Clase Mano de Obra	235,00
Clase Maquinaria	0,80
Clase Material	16,69
Clase Medio auxiliar	5,05
Med. aux. y Resto obra	
Total partida	**257,54**

Asciende el precio total a la expresada cantidad de: DOSCIENTOS CINCUENTA Y SIETE EUROS CON CINCUENTA Y CUATRO CÉNTIMOS

Presupuesto

Número	Cantidad	Ud	Descripción	Precio	Subtotal	Importe
Capítulo: 01			**IMPLEMENTACION Y DESARROLLO DE UNA RED DE SENSORES INALAMBRICA CON ALMACENAMIENTO DE LOS DATOS EN LA NUBE**			
Capítulo: 01.01			**Diseño e implementacion de software**			
01.01.01	1,00	ud	**Diseño e implementacion software** Diseño e implementacion software	357,4	357,4	
			Total capítulo: 01.01			**357,40**
Capítulo: 01.02			**Implementacion programas**			
01.02.01 P2	1,00		**Guardar en la SD** Guardar en la SD	1.076,36	1.076,36	
01.02.02 P3	1,00		**Subida datos a Dropbox y aviso via Twitter** Subida datos a Dropbox y aviso via Twitter	1.615,37	1.615,37	
Capítulo: 01.02.03			**Red Mesh**			
01.02.03.01 P4.1	3,00		**Sensor** Sensor	359,35	1.078,05	
01.02.03.02 P4.4	1,00		**Repetidor S3 a Maestro-Yun** Repetidor S3 a Maestro-Yun	539,02	539,02	
01.02.03.03 P4.5	1,00		**Maestro - Yun** Maestro - Yun	1.793,67	1.793,67	
			Total capítulo: 01.02.03			**3.410,74**
01.02.04 P5	1,00		**Maestro completo** Incluye todos los programas anteriores, agrupados en uno.	3.584,69	3.584,69	
			Total capítulo: 01.02			**9.687,16**
Capítulo: 01.03			**Implementacion de Hardware**			
01.03.01 Plac1	1,00		**Placa Maestro - Yun** Placa Maestro - Yun	1.829,65	1.829,65	
01.03.02 Plac2	1,00		**Placa repetidor** Placa repetidor	194,15	194,15	
01.03.03 Plac3	3,00		**Placa Sensor** Placa Sensor	257,54	772,62	
			Total capítulo: 01.03			**2.796,42**
			Total capítulo: 01			**12.840,98**
			Total presupuesto			**12.840,98**

Menfis 8.1.6 - Versión evaluación

		Pág.: 1
	RESUMEN DE CAPÍTULOS	Ref.: prores1
		Fec.:

Nº Orden	Código	Descripción de los capítulos	Importe
01	1	IMPLEMENTACION Y DESARROLLO DE UNA RED DE SENSORES INALAMBRICA CON ALMACENAMIENTO DE LOS DATOS EN LA NUBE	12.840,98
01.01	01	Diseño e implementacion de software	357,40
01.02	02	Implementacion programas	9.687,16
01.02.03	P4	Red Mesh	3.410,74
01.03	03	Implementacion de Hardware	2.796,42

TOTAL EJECUCIÓN MATERIAL	**12.840,98**
13 % Gastos Generales	1.669,33
6 % Beneficio Industrial	770,46
TOTAL EJECUCIÓN POR CONTRATA	**15.280,77**
21 % I.V.A.	3.208,96
TOTAL PRESUPUESTO C/IVA	**18.489,73**

Asciende el presupuesto proyectado, a la expresada cantidad de:
DIECIOCHO MIL CUATROCIENTOS OCHENTA Y NUEVE EUROS CON SETENTA Y TRES CÉNTIMOS

16 de Junio de 2016

LA PROPIEDAD	LA DIRECCIÓN TÉCNICA	LA CONSTRUCTORA
Fdo.:	Fdo.:	Fdo.: .....

Menfis 8.1.6 - Versión evaluación