Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERÍA INFORMÁTICA DE GESTIÓN Y SISTEMAS DE INFORMACIÓN

TRABAJO FIN DE GRADO

PLATAFORMA IOT PARA EL REGISTRO DE DATOS METEOROLÓGICOS Y SU VISUALIZACIÓN MEDIANTE UNA APLICACIÓN MÓVIL

DOCUMENTO I: MEMORIA

Alumno: Saavedra González, Alexander

Director (1): Casquero Oyarzabal, Oskar

Director (2): Equiraun Martínez, Harkaitz

Curso: 2017-2018

Fecha: Bilbao, a 23 de febrero de 2018

Resumen

En la actualidad, existe un concepto que está cobrando especial relevancia, el cual es conocido como IoT (Internet of Things, Internet de las Cosas).

En el IoT se define la interconexión digital de objetos cotidianos con internet, esto significa que no sólo “los humanos” tenemos la capacidad de conectarnos a internet, sino que caminamos hacia una nueva era donde prácticamente cualquier cosa podría ser conectada a internet, desde un reloj (smartwatch), como tenemos en la actualidad, hasta una nevera, una persiana, etc.

En este proyecto se ha querido aplicar ciertas fases del IoT, para convertir una información ambiental, proporcionada por una pequeña estación meteorológica, en un valor adicional a la hora de tomar decisiones basadas en las variables ambientales. Para ello utilizamos una serie de sensores que se encargan de darnos la información ambiental necesaria (como la temperatura, humedad y presión atmosférica), una fuente de procesamiento como puede ser un microcontrolador, una transmisión de datos entre estaciones, usamos las capacidades de la estación base para después poder manejar la información y procesarla en la nube, de forma remota, adquiriendo así el valor añadido que se espera en el IoT.

Para manejar todos estos conceptos y elementos, se hace uso de una aplicación multiplataforma, bases de datos, procesamiento, integrando todos los servicios en una misma plataforma que facilite la comunicación de todos los elementos involucrados.

Palabras Clave:

Internet de las Cosas, IoT, Sensores, Microcontrolador, Estaciones, Transmisión de datos, Procesamiento en la nube, Aplicación multiplataforma.

Abstract

Currently, there is a concept that is gaining special relevance, this concept is known as IoT (Internet of Things).

IoT defines digital interconnection of everyday objects to internet, this means humans will no longer be the only ones with the ability to connect to internet. We walk into a new era where anything could be virtually connected to Internet, from a clock (smartwatch), as we have today, to a refrigerator, a blind, etc.

In this project we have tried to apply certain phases of IoT, to convert a bit biased environmental information, in an additional value when making decisions about environmental conditions based on individual perceptions. We use different sensors that are responsible for giving the necessary environmental information (such as temperature, humidity, or atmospheric pressure), a source of processing such as a micro-controller, a data transmission between stations, we use the capabilities of the base station to later handle the information and process it in the cloud, remotely, thus acquiring the added value expected in the IoT.

To handle all these things, we make use of a multiplatform application, databases, automated processing, integrating all those services on a single platform that facilitates communication of all these elements.

Keywords:

Internet of Things, IoT, Sensors, Microcontroller, Environmental control, Data transmission, Cloud computing, Multiplatform Application.

Laburpena

Gaur egun, garrantzi berezia hartzen dabilen kontzeptu bat dugu gure artean, IoT deiturikoa (Gauzen Internet-a).

IoT-(e)n eguneroko objektuen eta Interneten arteko interkonexio digitala definitzen da, honek esan nahi du "gizakiok" ez garela Interneten konektatzeko gaitasuna dugun bakarrak, baizik eta gaur egun edozein motatako elementu batek gaitasun hau izan daitekeela, hala nola, erloju bat (smartwatch), gaur ezagutzen dugun bezala, edota hozkailu bat, pertsiana bat.. etab.

Proiektu honetan IoTeren zenbait aldi ezarri nahi izan dira ingurune-informazioa bihurtzeko meteorologia-estazio txiki batek emanarazia, erabakiak hartzerako orduan ingurune-aldagaietan oinarri harturik gehigarri gisa.

Horretarako, ezinbestekoa den ingurune-informazioa ematen diguten zenbait sentsore erabiltzen ditugu (hala nola, tenperatura, hezetasuna eta presio atmosferikoa), prozesamendu iturri bat mikro kontroladore bat izan daitekeen bezala, estazio arteko datu transmisio bat, funtsezko estazioaren gaitasunak erabiltzen ditugu ondoren informazioa maneiatu ahal izateko eta hodeian prozesatzeko, IoT(e)n espero den erantsitako balioa eskuratuz.

Kontzeptu eta elementu guzti hauek maneiatzeko, multiplataforma aplikazio bat erabiltzen da, datu-baseak, prozesamendua, zerbitzu guztiak plataforman integratuz parte-hartzen duten elementu guztien komunikazioa errazteko.

Gako-hitzak:

Gauzen Internet-a, IoT, Sentsore, Mikro kontroladore, Estazio, Datu transmisio, Hodeian prozesatu, Multiplataforma aplikazio.

Índice

Lista de figuras

Lista de tablas

Lista de Abreviaturas

IoT	*Internet of Things*
CSS	*Cascading Style Sheets*
SASS	*Syntactically Awesome Stylesheets*
I2C	*Inter-Integrated Circuit*
HTML	*Hypertext Markup Language*
JS	*JavaScript*
API	*Application Programming Interface*
HTTP	*Hypertext Transfer Protocol*
MQTT	*Message Queuing Telemetry Transport*
TFG	*Trabajo Fin de Grado*
IDE	*Integrated Development Environment*
CLI	*Command Line Interface*
MVC	*Model-View-Controller*
LCD	*Liquid Crystal Display*
XML	*eXtensible Markup Language*
CRC	*Cyclic Redundancy Check*
GPS	*Global Positioning System*
NMEA	*National Marine Electronics Association*
FTDI	*Future Technology Devices International*
USB	*Universal Serial Bus*
TTL	*Transistor-Transistor Logic*
IO	*Input/Output*
COM	*Puerto serie de una computadora*
IEEE	*Institute of Electrical and Electronics Engineers*
IP	*Internet Protocol*
SSH	***S**ecure **Sh**ell*
ES6	*ECMAScript 6*
UPV/EHU	*Universidad del País Vasco/Euskal Herriko Unibertsitatea*

1. Introducción

Este proyecto nace de la ambición e inquietudes de su autor por explorar las nuevas tecnologías y sus posibilidades en un mundo en continua evolución.

El objetivo principal consiste en poder monitorizar, almacenar y visualizar una serie de datos meteorológicos bajo el paradigma de lo que se conoce como Internet de las Cosas (IoT, Internet of Things), concepto de vanguardia y en continuo auge, cada vez con mayor implantación en todos los órdenes de la vida, y especialmente en el mundo de la Industria 4.0.

En la actualidad la demanda de proyectos basados en microcontroladores de hardware libre se ha consolidado tanto a nivel de aprendizaje como profesional. La sencillez tanto de implementación como de diseño de estos dispositivos permite al usuario realizar proyectos con rapidez y eficacia, dejando atrás lenguajes de bajo nivel los cuales dificultaban la programación del microcontrolador. La expansión creciente de este mercado ha permitido que el usuario pueda llevar a cabo cualquier tipo de proyecto. Ya que la gama de microcontroladores cada vez es mayor y más potente, así como la multitud de sensores y componentes electrónicos adaptados a éstos.

Por otra parte, los teléfonos inteligentes (smartphones) han irrumpido en nuestra vida diaria desde hace unos años atrás. Estos dispositivos que ya no son un simple sistema de comunicación si no que constituyen una herramienta de trabajo muy completa, asemejándose a la funcionalidad de un pequeño ordenador. La competencia en este mercado es encarnizada, lo que ha provocado que la calidad sea cada vez mayor ofreciendo numerosas opciones tanto de dispositivo como de sistema operativo.

En este contexto, este proyecto une estas dos tecnologías nombradas con anterioridad. Por una parte, se ha utilizado la plataforma Arduino para desarrollar las estaciones encargadas de muestrear los datos meteorológicos y subirlos a la nube.. Por otra parte, se ha desarrollado una aplicación móvil multiplataforma para visualizar los datos almacenados en la nube. Para dicho desarrollo se ha utilizado Ionic Framework, una tecnología capaz de crear aplicaciones híbridas con desarrollo web.

2. Objetivos y alcance del trabajo

El objetivo principal de este proyecto es el desarrollo de una plataforma IoT para el registro de datos meteorológicos y su visualización mediante una aplicación móvil. Para alcanzar este objetivo principal, se plantean los siguientes objetivos secundarios:

- Montaje y conexión de los elementos de cada estación en sendas protoboards.
- Muestreo de la temperatura, humedad, presión y geolocalización en la estación externa.
- Configuración de la estación remota para minimizar el consumo entre muestreos.
- Envío de datos de la estación externa a la estación base garantizando la detección de errores en esta última.
- Visualización de los datos correspondientes al último muestreo en una pantalla en la estación base.
- Envío de los datos desde la estación base a un repositorio de IoT en la nube.
- Visualización gráfica de los datos en una aplicación móvil multiplataforma.

3. Beneficios que aporta el trabajo

El “Internet of Things” (IoT) se apropia cada vez más del mundo que nos rodea, ya que en nuestro día a día, disponemos de dispositivos tecnológicos que conectan a Internet objetos de uso cotidiano a través de redes inalámbricas.

Toda la información generada por los dispositivos hiperconectados puede monitorizarse fácilmente a través de cualquier smartphone.

Al final, este tipo de tecnologías es de gran importancia ya que su uso puede dar nuevas perspectivas para el aprovechamiento de recursos y optimización de métodos.

De este modo, el IoT se plantea como una tecnología de enorme utilidad a la hora de mejorar la eficiencia de los proveedores de productos y servicios.

Este proyecto pretende facilitar información medioambiental y tener una visualización medida a través de un entorno gráfico como una aplicación móvil para la toma de decisiones del proveedor.

A pesar de que un ingeniero de software no haya acaparado términos electrónicos se obtendrá un nivel de entendimiento razonable, además de investigar nuevas tecnologías en el ámbito de la programación que son capaces de crear aplicaciones con un desarrollo web.

4. Descripción de requerimientos

4.1. Obtener parámetros ambientales con ayuda de sensores

Las tres magnitudes fundamentales que se miden en esta estación meteorológica son la temperatura, humedad y la presión atmosférica. Dichas magnitudes se miden mediante unos sensores digitales.

Para medir la temperatura y la humedad utilizamos el sensor DHT22, con el que se consigue una alta precisión (capacidad de un sensor de dar el mismo resultado en una misma medición durante un periodo de tiempo corto y en ausencia de cambios en las condiciones ambientales) y repetibilidad (capacidad de un sensor de dar el mismo resultado en diferentes mediciones realizadas en las mismas condiciones en distintos periodos de tiempo). Para este sensor se utilizará la librería DHT de Adafruit (https://github.com/adafruit/DHT-sensor-library), la cual ofrece una capa de abstracción respecto del protocolo de intercambio de datos con el sensor que facilita mucho la obtención de datos. La figura 1 muestra una vista general de este componente.

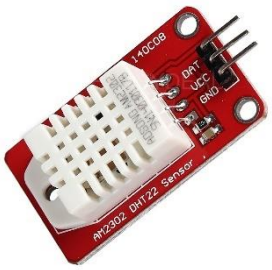


Figura 1: DHT22

Para el cálculo de la presión atmosférica utilizaremos el sensor BMP180, un sensor de amplio. Al igual que con el sensor DHT22, utilizaremos su correspondiente librería (https://github.com/LowPowerLab/SFE_BMP180) para abstraernos de las particularidades de la comunicación con el sensor.

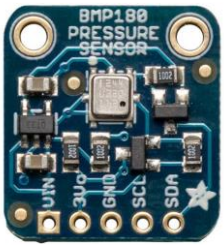


Figura 2: BMP180

A la temperatura, la humedad y la presión se le suman otros dos parámetros derivados de los anteriores: el punto de rocío y la sensación térmica. El punto de rocío es el valor al que debe descender la temperatura del aire para que el vapor de agua existente comience a condensarse. El punto de rocío puede calcularse directamente con los datos de temperatura y humedad relativa existentes en un momento dado. La sensación térmica es aquella sensación de frio o calor que

siente una persona según una combinación de parámetros meteorológicos. Se expresa en grados centígrados, al igual que la temperatura.

4.2. Localización de la estación remota (GPS)

La estación meteorológica que es planteada en este proyecto consta de una única estación remota, la cual, además, permanece estacionaria. Sin embargo, el proyecto podría extenderse fácilmente para dar cabida a varias estaciones remotas, las cuales podrían ser estaciones móviles (por ejemplo, una boya en el mar). En este contexto, se considera interesante conocer la geolocalización de la estación remota; por esta razón, se ha introducido un sensor de GPS G-622R en la estación remota. Para facilitar el procesamiento en Arduino de las sentencias NMEA que proporciona el GPS y que contienen la información de geolocalización, se ha utilizado la librería TinyGPSPlus (https://github.com/mikalhart/TinyGPSPlus).

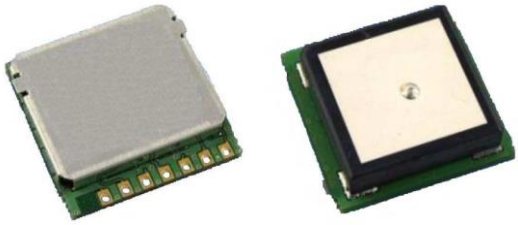


Figura 3: GPS G-622R

4.3 Comunicación entre la estación remota y el maestro

La estación meteorológica consta de dos elementos: estación base y estación remota. Para conseguir una comunicación inalámbrica robusta, de bajo consumo y largo alcance, y fácil de configurar y gestionar, se ha optado por módulos XBee Pro. Cada estación deberá de tener un módulo.

Figura 4: XBee PRO

4.4. Estación remota de bajo consumo

La estación remota se situará en exteriores y requiere una autonomía completa. Así, es necesario considerar ciertos requisitos y opciones:

- Por un lado, las fuentes de alimentación. Se ha de prescindir de fuentes de alimentación enchufables en pos de otras basadas en baterías recargables cuya recarga

provenga de la energía solar a través de una placa solar. De esta manera, la estación base tendrá dos posibles fuentes de alimentación -batería y energía solar- que se gestionarán de la siguiente forma: cuando la placa solar sea capaz de suministrar más corriente de la que la estación es capaz de consumir, el excedente se destinará cargar la batería; cuando la placa solar no sea capaz de alimentar la estación, el déficit de corriente será suplido por la batería; en ausencia de luz, la batería será la encargada de alimentar la estación. Un circuito integrado diseñado ad-hoc para gestionar este proceso será el encargado de realizar esta gestión.

- Por otro lado, minimizar el consumo de los diferentes circuitos integrados que componen la estación remota. En este sentido, el requerimiento que se establece es el de "dormir" el microcontrolador, el módulo XBee y el sensor de GPS (que son los elementos que mayor corriente consumen, 5mA, 250mA y 25mA, respectivamente) durante el periodo de muestreo y "despertarlos" únicamente durante el muestreo.

4.5. Almacenamiento de los datos en una plataforma IoT

Este proyecto se enmarca en el ámbito del Internet de las Cosas, por lo que se hace imprescindible subir los datos a la nube, a una plataforma diseñada ad-hoc para el almacenamiento de muestras de datos. Entre las diferentes alternativas existentes, se ha escogido la plataforma de IoT ThingSpeak por dos razones: primero, ThingSpeak tiene un conjunto de servicios (un *API*) muy completo y bien documentado; segundo, ThingSpeak es propiedad de MATLAB, lo cual favorece la integración de la plataforma de almacenamiento de datos en la nube (ThingSpeak) con la herramienta de análisis de datos (MATLAB) a través de una toolbox de MATLAB. Cada cliente tendrá su cuenta en ThingSpeak, y un script en la estación base se encargará de la creación de un canal para el almacenamiento de datos y de la subida de datos a ese canal; todo ello de forma automática, sin la intervención del usuario. La creación del canal se hará mediante protocolo HTTP (arquitectura REST) y la subida de datos mediante protocolo MQTT (arquitectura Industria 4.0), ya que este último es un protocolo bastante más ligero y simple que el anterior.

Figura 5: Logo ThingSpeak

4.6. Visualización de datos en una pantalla LCD

Como requerimiento adicional, añadiremos una pantalla LCD para poder visualizar los datos de temperatura, humedad y presión en la propia estación base. Una vez más, para abstraernos del protocolo de intercambio de datos con la pantalla, utilizaremos la librería U8g2 (https://github.com/olikraus/u8glib/wiki).

4.7. Visualización de los datos en una aplicación móvil multiplataforma

Uno de los principales requerimientos consiste en que el usuario pueda ver los datos registrados por la estación meteorológica desde cualquier ubicación a través de una conexión a Internet. Para ello, decidimos que lo más adecuado era crear una aplicación móvil, ya que este dispositivo es el que acompaña a la mayoría de los usuarios en su rutina diaria. Dado que crear una aplicación nativa para cada sistema operativo móvil -Android y iOS-, requería el aprendizaje de sus respectivos lenguajes de programación, se optó por programar una aplicación híbrida, es decir, una aplicación que se programa en un determinado lenguaje de programación y que se puede portar a diferentes plataformas móviles. Concretamente, se ha utilizado la plataforma Ionic para programar la aplicación híbrida, porque es una plataforma que permite programar aplicaciones móviles partiendo de lenguajes y/o plataformas de programación utilizadas para programar aplicaciones web (por ejemplo, Angular) y tiene una excelente documentación y una gran comunidad de soporte.

Figura 6: Logo Ionic

5. Análisis del estado del arte

5.1. Oregon Scientific BAR-218-HG

La estación meteorológica cuenta con una alerta de clima y meteorología, y combina una conexión inalámbrica Bluetooth que permite controlar también las condiciones climáticas locales a través de un teléfono inteligente. Esta estación meteorológica inalámbrica trae toda la información vital sobre el clima que uno pueda necesitar para planificar el día. La estación recibe información de un sensor exterior inalámbrico y muestra tanto la temperatura interior como exterior, al igual que el nivel de humedad. Pronóstico de 12-24 horas, y la fase de la luna. La función de alerta meteorológica también muestra la previsión del tiempo: calor, niebla, heladas, lluvia y viento/tormenta.

Figura 7: Oregon Scientific Bar-218-HG

La incorporación de Bluetooth permite comunicarse con la estación meteorológica directamente usando una aplicación. El único requisito es tener un software superior a iOS 9 o Android 4.3.

Muestra el tiempo actual, la temperatura y la humedad de hasta 5 lugares con un historial de 7 días y un registro de datos. Su rango de 30 metros de transmisión significa que uno puede moverse libremente alrededor de su casa al comprobar las actualizaciones más recientes.

La función de reloj se sincroniza automáticamente con el smartphone por lo que es exacta y se actualiza automáticamente para el cambio horario.

Figura 8: Características Oregon Scientific Bar-218-HG

5.2. Oregon Scientific LW301

Figura 9: Oregon Scientific LW301

Estación meteorológica para visualización de datos a través de internet y/o smartphone (iOS o Android) mediante la aplicación "Anywhere weather". A través de una conexión a internet se envían los datos a un servidor de Oregon Scientific. Desde aquí podremos visualizarlos en tiempo real en un ordenador o en un smartphone. También existe la posibilidad de ver los datos de nuestra estación a través del servidor de Oregon Scientific en la web www.osanywhereweather.com.

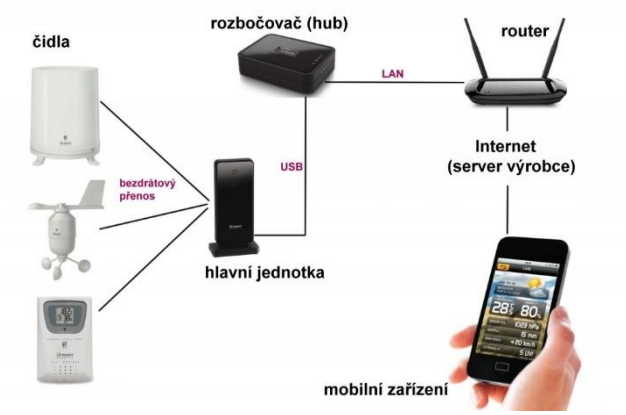


Figura 10: Esquema de conexiones de Oregon Scientific LW301

Esta estación mide los principales parámetros meteorológicos: temperatura, humedad, presión atmosférica, lluvia, velocidad y dirección del viento

	Temperatura	Humedad	Presión	Pronóstico	Gráfico presiones	Lluvia	Velocidad viento	Dirección viento	Conexión a ordenador
Interior	No	No	Sí	Sí	Sí	-	-	-	Sí
Exterior	Sí	Sí	-	-	-	Sí	Sí	Sí	-

Figura 10: Parámetros que mide Oregon Scientific LW301

Incorpora tres sensores exteriores: temperatura/humedad, velocidad/dirección del viento y pluviómetro. Todos ellos funcionan con pilas y transmiten los datos al receptor interno por radio y sin cables, con un alcance máximo en campo libre de 100 metros sin obstáculos. Este receptor

se conecta al hub suministrado. El hub se debe conectar a un router con conexión a internet. Para ver los datos es necesario tener un router y conexión a internet ya que la estación no trae pantalla.

Funciones meteorológicas:

- Temperatura:
  - Temperatura -30 y +60ºC y memoria de máxima y mínima.
  - Punto de rocío y memoria de máxima y mínima
  - Sensación térmica y memoria de máxima y mínima.
  - Transmisión de datos exteriores cada 60 seg. aproximadamente.
  - Opcionalmente se pueden añadir hasta ocho sensores más de temperatura y humedad THGR810.
- Humedad:
  - Humedad interior de 25 a 90%hr y memoria de máxima y mínima.
  - Transmisión de datos exteriores cada 60 seg. aproximadamente.
- Presión:
  - Presión atmosférica local y a nivel del mar.
  - Rango de 700 a 1050 mb, con una resolución de 1 mb y una precisión de ± 10 mb.
- Lluvia y viento:
  - Lluvia de las últimas 24 horas y hora actual.
  - Velocidad y dirección del viento y con memoria de máxima velocidad.
  - Transmisión de datos cada 14 seg. aproximadamente.
- Otras funciones:
  - Alcance máximo de transmisión de 100 mts. campo libre sin obstáculos.
  - Alimentación eléctrica.
  - Sensores exteriores con pilas alcalinas AA o AAA (incluidas).
  - Hub de red de 220 V.

5.3. National Geographic 90-68000

Esta estación meteorológica se compone de una consola principal y un sensor exterior termo-higro, que recoge y transmite los datos de temperatura y humedad exteriores. Sólo está incluido un único sensor exterior de un solo canal.

Figura 11: National Geographic 90-68000

Funciones meteorológicas:

- Pronóstico del tiempo: soleado, parcialmente nublado, lluvia ligera, lluvia, lluvias fuertes, condiciones meteorológicas inestables y nieve.
- Presión:
  - Presión actual o histórica (mBar / hPa, mmHg o inHg).
  - Ajuste de presión de altitud o en el nivel del mar para compensar la presión atmosférica.
  - Presión histórica a nivel del mar de los últimos 24 días.
  - Gráfico de barras con la presión histórica a nivel del mar.
- Fases de la luna:
  - 12 símbolos con las fases de la luna.
  - Escaneo de fases.
  - Fases de los últimos o los próximos 39 días.
- Reloj radio controlado:
  - La hora y el día se sincronizan a través de una radio señal con la precisión de un reloj atómico.
- Reloj y calendario:

    - 12hr/24hr y mes/día o día/mes.
    - Diferentes combinaciones de pantallas de reloj y calendario.
    - 6 idiomas diferentes para el día de la semana.
- Alarmas:
    - Alarma simple, semanal, pre-alarma, función de repetición.
- Horas de salida y puesta de sol:
    - Calcula la hora del amanecer con la información geográfica proporcionada por el usuario (133 puntos geográficos).
- Temperatura y humedad relativa remotas:
    - Muestra la temperatura y la humedad relativa tanto interior como exterior (ºC o ºF).
    - Memoria de máximos y mínimos de temperatura y humedad relativa.
- Sensación térmica:
    - Analiza las condiciones ambientales actuales (confort, húmedo y seco).

La transmisión remota de datos a la consola principal se efectúa a través de frecuencia de 433 MHz. La comunicación es sin cables y con un radio de operación de hasta 100 metros en un área abierta. El sensor puede colocarse en el interior o en el exterior.

5.4. Conclusiones

Estas alternativas tienen un precio desde 70 hasta 100 euros. Nos encontramos con productos que tienen características similares a nuestra estación meteorológica. Nuestro producto está compuesto por los mejores factores de cada una de ellas, entre otras:

- Una distancia más larga entre estaciones gracias a los módulos XBee.
- Histórico de datos almacenados en la plataforma ThingSpeak.
- Magnitudes fundamentales como temperatura, humedad relativa, presión atmosférica, sensación térmica y punto de rocío.
- Coordenadas de la estación remota sin necesidad de que el usuario intervenga introduciendo su geolocalización.
- Pantalla gráfica donde mostramos ciertos datos -temperatura, humedad, presión, IP, logo UPV/EHU, hora y fecha-.

- Posibilidad de ver los datos registrados desde cualquier ubicación a través de una conexión a internet y mediante una aplicación que facilitamos.
- Gráficas lineales y comparativas con otros puntos geográficos.
- Una autonomía mucho más duradera y económica gracias a una pila cargada mediante una placa solar y utilizando técnicas de consumo bajo.

6. Análisis de alternativas

6.1 Estación base

La mejor opción que se puede encontrar como alternativa es la Raspberry Pi, ya que la funcionalidad que realiza la estación base es bien sencilla (recibir datos y ejecutar script). Raspberry Pi es un mini ordenador de pequeño tamaño, bajo coste y bajo consumo. Generalmente este tipo de mini ordenadores ejecutan sistemas operativos basados en Linux. Además de un ordenador Raspberry incorpora funciones de electrónica. Con lo que pueda ser empleado en proyectos de electrónica y robótica interactuando con sensores y actuadores. En este sentido, podríamos decir que "comparte" ciertas capacidades con dispositivos como Arduino. No obstante, hay que remarcar que la potencia de estos mini PCs no es equiparable a la que disponemos en un ordenador "convencional".

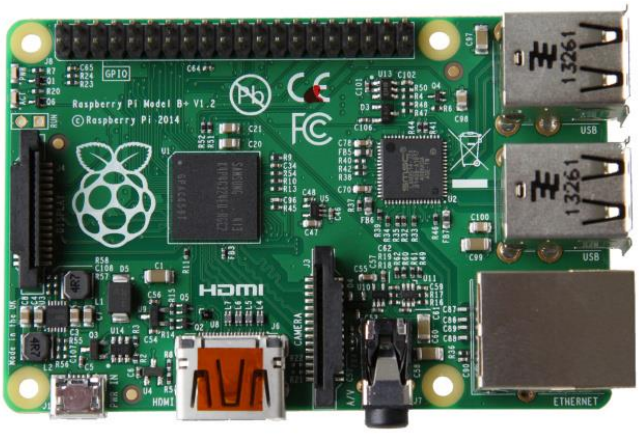


Figura 12: Raspberry Pi

Es una alternativa más cómoda ya que nos evitaríamos la comunicación entre Arduino y Linux en el caso del Arduino Yun.

6.2. Estación remota

6.2.1. Sensores de temperatura y humedad

El DHT11 y el DHT22 (el que utilizaremos) son dos modelos de una misma familia de sensores, que permiten realizar la medición simultánea de temperatura y humedad. Estos sensores disponen de un procesador interno que realiza el proceso de medición, proporcionando la medición mediante una señal digital, por lo que resulta muy sencillo obtener la medición desde un microprocesador como Arduino.

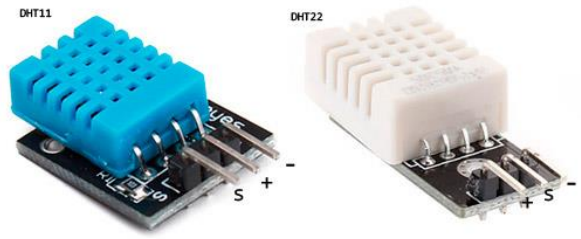


Figura 13: DHT11 y DHT22

Ambos sensores presentan un encapsulado de plástico similar. Podemos distinguir ambos modelos por el color de este. El DHT11 presenta una carcasa azul, mientras que en el caso del sensor DHT22 el exterior es blanco. De ambos modelos, el DHT11 es el hermano pequeño de la familia, y cuenta con peores características técnicas. El DHT22 es el modelo superior, pero, por contra, tiene un precio superior.

Las características del DHT11 son realmente escasas, especialmente en rango de medición y precisión.

- Medición de temperatura entre 0 a 50, con una precisión de 2ºC.
- Medición de humedad entre 20 a 80%, con precisión del 5%.
- Frecuencia de muestreo de 1 muestras por segundo (1 Hz).

El DHT11 es un sensor muy limitado que podemos usar con fines de formación, pruebas, o en proyectos que realmente no requieran una medición precisa. Por el contrario, el modelo DHT22 tiene unas características mucho más aceptables.

- Medición de temperatura entre -40 a 125, con una precisión de 0.5ºC.
- Medición de humedad entre 0 a 100%, con precisión del 2-5%.
- Frecuencia de muestreo de 2 muestras por segundo (2 Hz).

EL DHT22 (sin llegar a ser en absoluto un sensor de alta precisión) tiene unas características aceptables para que sea posible emplearlo en proyectos reales de monitorización o registro, que requieran una precisión media.

Por otro lado, se encuentra la posibilidad de usar sensores de temperatura y humedad individuales, con los cuales conseguiremos una medición más exacta pero que por el contrario tendremos un mayor consumo. Como alternativa al sensor de temperatura está el TMP32. Este es un sensor analógico muy popular y sencillo de utilizar. Se conecta una resistencia desde 5V a GND y el sensor dará en su salida un voltaje que se puede medir con el microcontrolador. La salida del sensor es lineal por lo que no tendrás que hacer cálculos de conversión.

Figura 15: TMP32

Respecto a la humedad, el HH10D es un sensor de humedad relativa compuesto por un sensor capacitivo de humedad, un convertidor capacitivo CMOS y una EEPROM interna para almacenar los valores de calibración. Debido a su funcionamiento interno, el sensor es capaz de reaccionar rápidamente a los cambios de humedad. La salida es un tren de pulsos cuya frecuencia es proporcional a la humedad mediante una formula.

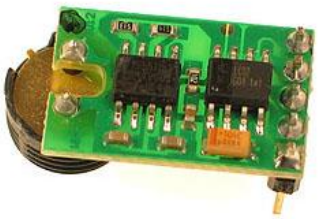


Figura 14: HH10D

6.2.2 **GPS 622R**

Dado que nuestro GPS G-62R requiere de una cantidad de tiempo algo elevada para la obtención de datos de los satélites, una alternativa mejor y mucho más conocida es el GPS NEO 6. Los dispositivos NEO-6 son una familia de receptores fabricados por U-Blox, que pueden ser conectados con facilidad a un autómata o procesador como Arduino. La familia de receptores GPS NEO-6 están diseñados para tener un pequeño tamaño, pequeño coste, y pequeño consumo. La intensidad de corriente necesaria es de unos 37mA en modo de medición continuo. El sensor tiene un tiempo de encendido cold y warm de unos 30s, y en hot 1 segundo.

Figura 16: GPS NEO-6

6.2.3 **Comunicación inalámbrica entre emisor y receptor**

- ***Bluetooth:*** Un módulo bluetooth como HC-05 y sus hermanos serían una alternativa perfecta para nuestros XBee Pro. Este módulo es sencillo e ideal para proyectos pequeños y poder encontrar una fácil comunicación entre varios micro controladores. Permite una conexión sencilla y sin problemas mediante comandos AT a través de un puerto serie.

Figura 17: Módulo Bluetooth HC-05

- ***ESP8266:*** Este es un microprocesador de bajo coste con WiFi integrado, fabricado por Espressif. Antes del ESP8266, las opciones disponibles para conectar un Arduino a WiFi (como el WiFi Shield) eran prohibitivamente caras. La aparición del ESP8266 supuso una pequeña revolución al ser el primer dispositivo realmente barato que proporcionaba conectividad WiFi. Aun así, la comunicación con Internet puede suponer una carga excesiva para Arduino. Tiene una tensión de alimentación de 3.3V. En ningún caso puede alimentarse a una tensión superior a 3.6V, o dañaremos el módulo. La comunicación con el firmware por defecto, al igual que HC-05, se realiza a través de comandos AT, que recordemos no son más que comandos de texto enviados por Serial.

Figura 18: ESP8266

6.2.3 **Barómetro**

Un barómetro digital es un dispositivo que mide la presión del aire y que puede usarse como altímetro. Podemos conectar este sensor a un autómata o procesador como Arduino para registrar la medición de la presión del aire o estimar la altitud del sensor respecto al nivel del mar.

El BMP180 es un sensor barométrico digital, que incorpora un sensor de temperatura para compensar sus efectos en la medición de la presión barométrica. Este es un sensor de alta precisión y baja potencia. Respecto a otras alternativas, están las opciones de utilizar BMP085 o SCP1000, que la diferencia entre ellos son números relativamente similares a la hora de obtención de datos. El BMP180 es el más utilizado hoy en día para este tipo de proyectos.

Figura 20: BMP085

Figura 19: SCP1000

6.3 Almacenamiento de datos

6.3.1 **Plataforma IoT**

Uno de los mayores rasgos significativos del proyecto es la supervisión de datos en tiempo real. Es por ello por lo que una manera sencilla y económica de poder ver qué sucede, es a través de una plataforma IoT, como en nuestro caso ThingSpeak. Dentro de esta categoría encontramos plataformas orientadas exclusivamente a proyectos del IoT y otras plataformas que ofrecen múltiples servicios para todo un sistema del mismo. Los protocolos de comunicación más utilizados son HTTP, MQTT y CoAP. Tener una *API* (Application Programming Interface) nos permitirá consultar, modificar y borrar la información desde otros dispositivos.

En cuanto a las alternativas, he incluido todas aquellas que nos permiten su uso de una manera gratuita, pero con limitaciones en cuanto al número de mensajes enviados y de dispositivos conectados.

- ***Arduino Cloud:*** Quizás sea la sencilla para utilizar en proyectos IoT con Arduino. No hace falta decir que es totalmente compatible con cualquier placa de Arduino que tenga conectividad. Para configurar un dispositivo es muy sencillo, solo tienes que seguir 4 simples pasos.

Figura 21: Arduino Cloud

- ***Artik Cloud:*** Es la apuesta de Samsung por el sector del IoT. También podemos adquirir hardware, el dispositivo Artik 1029, y el que pretende ser el competidor de Raspberry Pi.

Figura 22: Samsung Artik Cloud

- ***Thingerio:*** Es una plataforma de código abierto. La encontramos en su propio servidor como en GitHub para instalarla en una máquina propia.

Figura 23: Thingerio

ThingSpeak, es que además de ser una plataforma IoT muy reconocida, está enfocada exclusivamente a la construcción de aplicaciones del sector IoT. Como añadido, permite almacenar datos, visualizarlos y exponerlos a otras *APIs*. Dispone de una documentación muy extensa con ejemplos y es totalmente gratuita.

6.3.2 **Usuarios**

Firebase Google es una solución perfecta para el almacenaje de datos de usuarios. Es una plataforma en la nube muy completa que nos ofrece una integración total con Ionic. Ofrece desde registro de usuarios, base de datos, hosting, notificaciones, etc., y con un plan de pago totalmente gratuito.

Figura 24: Firebase Google

Por el contrario, podemos optar por otras alternativas aún más conocidas, pero quizás con menor integridad con Ionic.

- ***AWS (Amazon Web Services):*** Sin duda alguna, una de las grandes plataformas en la nube. Ofrece muchos servicios relacionados con la comunicación entre servicios y el

análisis de datos. No es una plataforma gratuita, el precio depende de los servicios que utilices.

Figura 25: Amazon Web Service

- ***Google Cloud Platform:*** Aunque Firebase y esta pertenezcan a Google, esta última está centrada para el ámbito profesional y empresarial.

Figura 26: Google Cloud Platform

- ***Microsoft Azure:*** Al igual que la plataforma de Amazon AWS y de Google Cloud, también se mete en el mercado Microsoft, con su plataforma en la nube Azure. Ofrece gestionar dispositivos, datos y todo lo que conlleva un sistema IoT.

Figura 27: Microsoft Azure

6.4 Visualización de datos

6.4.1 **Gráficas**

Una alternativa muy conocida en el ámbito gráfico es la librería D3, con la que cuenta con una diversidad infinita de tipos de gráficas y de funcionalidades, y quizás, una mayor interacción con sus gráficas.

Figura 28: D3.js

La librería Chart.js hace que su funcionalidad e integridad sea mucho más sencilla y fácil que la anterior. Es por ello que hemos recurrido a esta librería ya que su trabajo se ceñirá solamente a mostrar graficas lineales.

Figura 29: Chart.js

6.4.2 **Aplicación móvil**

Dado que crear una aplicación nativa para cada sistema operativo móvil -Android y iOS-, requería el aprendizaje de sus respectivos lenguajes de programación, se optó por programar una aplicación híbrida.

Contrapartida de las aplicaciones híbridas/nativas:

- ***Curva de aprendizaje:*** Al no utilizar HTML sino sus propios componentes (tipo XML), tendrás que aprenderlos todos, por lo que la curva de aprendizaje es más larga.
- ***Estilos:*** Siguiendo con la curva de aprendizaje, no solo deberías preocuparte por como mostrar datos en pantalla, sino de su diseño. Las dos alternativas que se propone tienen su propio sistema para darle estilo a tu aplicación y que se suma a la curva de aprendizaje.
- ***Plataformas:*** De momento, tanto React Native como Titanium Appcelerator están solo disponibles para Android y iOS. No es que la cuota de mercado de smartphones Windows sea muy elevada, pero seguro que quieres que tu aplicación esté en cuantos más smartphones mejor.
- ***No funciona en web****:* Es importante recalcar esto. Tu aplicación React Native o Appcelerator no es una web y por tanto no podrás usarla como tal.

Alternativas:

- ***Titanium Appcelerator:*** Es un kit de desarrollo de aplicaciones para dispositivos móviles que te permite crear aplicaciones multiplataforma basadas en JavaScript, pero con la apariencia y rendimiento de una aplicación nativa. Una de las ventajas de Appcelerator es que podemos programar nuestras aplicaciones siguiendo la arquitectura MVC (Modelo-Vista-Controlador) donde, utilizando XML y CSS podemos separar la interfaz de usuario, la lógica de negocio y los datos.

Figura 30: Titanium Appcelerator

- ***React Native:*** Es un framework desarrollado por Facebook y que usa la misma funcionalidad que Appcelarator, construir una aplicación móvil mediante JavaScript y un "puente" convirtiéndola en una aplicación nativa tanto para iOS como para Android. En React Native, en lugar de renderizar HTML sobre un WebView, se renderizan componentes nativos Android/iOS, que se pintan en su propio *thread*, ganando fluidez.

Figura 31: React Native

¿Por qué Ionic?

- ***Es web:*** A diferencia de las otras dos soluciones, aquí estás desarrollando con tecnología web pura y dura. Esto significa explotar Angular 2, HTML5 y CSS3 en toda su magnitud. Además, significa que tu aplicación funcionará también como *web mobile*.

Figura 33: Angular

Figura 32: HTML5

Figura 34: CSS3

- ***Más plataformas:*** Además de Android y iOS, también funciona perfectamente con Windows, así que llegarás a más usuarios.

Figura 35: Android, iOS y Windows Phone

- ***Eficiencia*:** Ionic 1 ya se movía con bastante fluidez en smartphones de gama media. Ionic 2, con Angular 2 rehecho de cero para solucionar los problemas de rendimiento de su predecesor cuando el número de *bindings* se disparaba, debería volar sobre la pantalla.
- ***WebWorkers*:** Gracias a Angular 2, ahora es más fácil trabajar con Web Workers que te permiten realizar tareas en otros *threads*.
- ***Potencia*:** Cada día los smartphones son más potentes y los navegadores para móviles están más trabajados, así que los problemas de rendimiento del pasado deberían quedar cada vez más enterrados.

6.5 Editor de texto/IDE

El primer editor de texto del que se va a hablar es ***Sublime Text***. Este es uno de los editores más completos en cuanto a complementos, funciones y personalización que podemos encontrar en toda la red.

Figura 36: Sublime Text

Listar todas las funciones y características que nos ofrece Sublime Text es bastante complicado, por lo que se va a intentar resumirlas:

- Cuenta con una función que nos permite seleccionar varias líneas a la vez y escribir a la vez en ellas. Muy útil a la hora de crear funciones o renombrar variables.
- Nos permite realizar modificaciones en lote de varios puntos de un documento.
- Cuenta con una paleta de comandos que nos da acceso rápido a una completa lista de funciones, como, por ejemplo, acceso a los marcadores o a la lista de sintaxis para colorear un fichero según el lenguaje de programación en el que esté escrito.
- Cuenta con un comando "GoTo" que nos permite desplazarnos muy fácilmente entre varios documentos, muy útil en proyectos grandes. Esta función, además, nos sirve para desplazarnos entre funciones, líneas e incluso palabras.
- La función "Find and Replace" nos permite seleccionar y remplazar rápidamente cualquier contenido del documento.
- Modo "sin distracciones" que nos permite escribir sin ningún otro elemento que pueda hacer que nos distraigamos.

- Cuenta con una completa *API* escrita en Python y, además, una consola que nos permite experimentar con nuestros proyectos muy fácilmente.

Además de todo eso, otra característica elemental de este editor es que es totalmente personalizable. Ya sea desde el propio editor como desde sus ficheros de configuración podemos cambiar todo lo que imaginemos, desde los atajos de teclado hasta las macros. Absolutamente todo. Por si fuera poco, este editor puede personalizarse aún más gracias a la gran variedad de extensiones, complementos y add-ons que podemos encontrar por la red.

Visual Studio Code es un editor de texto/IDE de programación desarrollado y lanzado por Microsoft en 2015 como una herramienta de código abierto. Este editor se caracteriza por ser bastante más modular que el anterior. Por defecto, el editor cuenta con las funciones más básicas y elementales y está preparado para adaptarse a los principales lenguajes de programación más utilizados, como HTML/CSS, JavaScript o C. Sin embargo, su mayor potencial se encuentra en las extensiones, y es que desde el mismo editor vamos a poder acceder a una gran variedad de extensiones y complementos que nos va a permitir dotarle de las funciones y características que necesitemos, sin sobrecargar el editor.

Figura 37: Visual Studio Code

Entre otras, las principales características que nos brinda Microsoft Visual Studio Code son:

- IntelliSense, una función que nos permite desde resaltar la sintaxis de nuestros proyectos hasta hacer uso de autocompletar funciones, controlar nuestras variables y, además, ver definiciones de las funciones de los distintos lenguajes de programación.
- Cuenta con un avanzado depurador de código que nos permite conocer todos los problemas y errores en tiempo real, ayudándonos a identificar estos problemas y solucionarlos.
- Se integra en Git. Nos permite versionar nuestros trabajos fácilmente desde esta plataforma.

Figura 38: Git

- Gracias a sus módulos, es compatible con prácticamente cualquier lenguaje de programación, desde los más conocidos hasta los más extraños de los que, probablemente, no hayamos oído hablar nunca.

El uso de Visual Studio Code se debe a la integridad con tantos diversos lenguajes de programación, en mi caso he obtenido mucha ayuda a la hora de implementar código en JavaScript y más aún en añadir elementos propios de Ionic. Además, cuenta con el añadido de que podemos tener la consola abierta en el mismo IDE.

Por otro lado, buscando por la red, se dice que Sublime Text es mucho más ligero y se ejecuta bastante más rápido que su rival, incluso es capaz de abrir archivos con más de 10 millones de líneas, algo que la mayoría de sus rivales no pueden hacer.

Pero, si ninguno de estos dos nos gusta, no podemos olvidarnos de otras opciones similares que podemos encontrar, como, por ejemplo, ***Atom***, el editor propio de GitHub especialmente diseñado para la programación web, ***Brackets***, una alternativa similar a Atom de la mano de Adobe, e incluso el potente y versátil ***Vim***.

Figura 39: Atom

Figura 41: Brackets

Figura 40: Vim

6.5 Editor de texto/IDE Arduino

Además del IDE estándar existen una buena variedad de alternativas para trabajar con Arduino. Su instalación y uso no es tan sencillo como el anterior, pero a cambio tendremos funcionalidades adicionales, mayor potencia, y mejor control del código ejecutado.

- El entorno de desarrollo por excelencia de Microsoft. ***Visual Studio*** puede ser configurado para trabajar con Arduino. Visual Studio añade localización de errores, predicción de texto, agrupación de proyectos, en fin, una maravilla.

Figura 42: Visual Studio

- El famoso entorno de programación ***Eclipse*** puede ser configurado para configurar procesadores AVR, disponiendo de todas las bibliotecas y comandos propios de Wiring, y realizar la escritura en el dispositivo desde el propio IDE. Además, pone a nuestra disposición todas las herramientas de C++ (objetos, constructores, librerías, entornos gráficos…). Sin embargo, el proceso de configuración es sustancialmente más complicado.

Figura 43: Eclipse

- Por último, ***Atmel Studio***, otras de las opciones más populares para el desarrollo en Arduino, especialmente por sus impresionantes funciones de *debug*.

Figura 44: Atmel Studio

7. Análisis de riesgos

Todo proyecto de cierta índole tiene unos riesgos asociados que conviene identificar para poder evitarlos (en la medida de lo posible) o, en caso de ser imposibles de evitar, estar preparado para actuar y solucionarlos lo más rápido posible y con la mayor eficacia.

En esta sección se detallarán los posibles riesgos detectados, la probabilidad de que ocurran, el impacto que supondría en el proyecto, así como las consecuencias que causarían. Se detallarán también las medidas de prevención tomadas, así como el plan de contingencia que se aplicará para arreglar el problema.

7.1. Planificación incorrecta del tiempo del proyecto

- ***Descripción*:** Al ser una de las primeras planificaciones que se realizan para un proyecto tan grande, es muy probable que las fechas y tiempos marcados en la planificación no sean correctos. Siendo estos tiempos demasiado optimistas, produciendo que la fecha de entrega no sea correcta.
- ***Probabilidad***: Alta.
- ***Impacto***: Alto.
- ***Consecuencia***: Una mala planificación del proyecto retrasaría todos los procesos y con ello la fecha de entrega se vería afectada.
- ***Prevención*:** La manera de prevenir este problema será hacer plazos de entrega más amplios para así tener margen de maniobra.
- ***Plan de contingencia*:** Se aumentarán las horas diarias invertidas a fin de encauzar los flujos temporales de nuevo para que la fecha final se vea afectada lo menos posible.

7.2. Baja por accidente o enfermedad

- ***Descripción*:** Debido al clima cambiante que sufrimos ahora en invierno es posible caer enfermo o también sufrir un accidente debido al periodo ocioso que representa esta estación.
- ***Probabilidad***: Baja.
- ***Impacto*:** Media.
- ***Consecuencia***: Sin ponernos en casos graves y teniendo en cuenta las enfermedades y accidentes más comunes, las consecuencias serían de pérdida de algunas horas de trabajo. En caso de problemas más graves el retraso o pérdida de horas será mucho mayor.
- ***Prevención*:** No existe una prevención contra este tipo de problemas pues son aleatorios. Si bien se pueden tener en cuenta a la hora de planificar tareas dándoles un poco más de margen.

- ***Plan de contingencia:*** Se aumentarán las horas de trabajo de otros días a fin de compensar las horas perdidas.

7.3. Pérdidas totales o parciales de documentos/ficheros

- ***Descripción*****:** Es posible que los ficheros y/o documentos sufran algún tipo de corrupción o perdida que imposibilite su lectura y trabajar con ellos.
- ***Probabilidad***: Baja.
- ***Impacto*****:** Baja.
- ***Consecuencia***: Perdida parcial del trabajo.
- ***Prevención*****:** Tener diferentes copias de seguridad alojadas en diferentes almacenamientos tanto en la nube (Dropbox, OneDrive, Google Drive…) como físicos (Disco duro, Disco duro externo.).
- ***Plan de contingencia:*** En caso de que produjera este accidente, se procedería a restaurar una copia de seguridad en el lugar afectado para seguir trabajando.

7.4. Sufrir fallos en el ordenador de trabajo

- ***Descripción*****:** Es posible que el entorno de trabajo sufra daños y deje de funcionar.
- ***Probabilidad***: Baja.
- ***Impacto*****:** Medio.
- ***Consecuencia***: Perdida del entorno de trabajo.
- ***Prevención*****:** Tener más dispositivos para poder seguir trabajando en caso de que un equipo falle.
- ***Plan de contingencia:*** Usar otro equipo para seguir con el trabajo hasta que el principal esté en condiciones de volver a ser usado.

7.5. Problemas a la hora de implementar algoritmo

- ***Descripción*****:** Es muy posible que aparezcan problemas a la hora de implementar un algoritmo nuevo.
- ***Probabilidad***: Alta.
- ***Impacto*****:** Medio.
- ***Consecuencia***: Retraso en las siguientes tareas.
- ***Prevención***: Dedicar el tiempo necesario a diseñar un código que sea fácil de implementar. También dedicar tiempo a la búsqueda de recursos para ayudarme con la tarea.
- ***Plan de contingencia:*** Dedicar más horas a la implementación a fin de no retrasar las demás tareas.

7.6. Problemas a la hora de implementar prototipos debido a cambios o errores en las *API/librerías*

- ***Descripción*:** Es posible que, durante la implementación de alguna funcionalidad o una vez ya implementada, la API/librería cambie dejando inservible el código actual.
- ***Probabilidad***: Baja.
- ***Impacto*:** Medio.
- ***Consecuencia***: Retraso en las demás tareas pudiendo afectar a la fecha de entrega del proyecto.
- ***Prevención***: Estar al día de todos los posibles cambios y actualizaciones que sufra la *API/librería*.
- ***Plan de contingencia:*** Considerar si los cambios de la *API/librería* son demasiado costosos de asumir a la hora de actualizar el código. Si el cambio obliga a demasiados cambios se puede buscar un servicio similar que no requiera tanto coste horario. Además, aumentar las horas de trabajo a fin de no retrasar las demás tareas y la fecha de entrega.

7.7. Fallos en los componentes eléctricos

- ***Descripción*:** Debido a que la gran mayoría de componentes son eléctricos puede que dejen de funcionar por sobrecarga, como, por ejemplo, a la hora de hacer pruebas.
- ***Probabilidad***: Alta.
- ***Impacto*:** Medio.
- ***Consecuencia***: Añadir más dinero al presupuesto.
- ***Prevención***: Hacer las pruebas con precaución y repasar tanto la documentación del componente como los medios conectados.
- ***Plan de contingencia:*** Comprar más de un mismo componente a la hora de hacer el pedido para prevenir que la estación meteorológica quede inservible hasta obtener un nuevo componente.

7.8. Error de conexión a internet

- ***Descripción*:** Perdidas de conexión a internet en todos los elementos que necesitan señal WiFi -módulos XBee: enviar y recoger datos, Arduino Yun: subida de datos a la nube, Aplicación móvil: recogida de datos de la nube-.
- ***Probabilidad***: Alta.
- ***Impacto*:** Medio.
- ***Consecuencia***: Aumenta el tiempo de espera.

- ***Prevención***: No existe una prevención contra este tipo de problemas pues son aleatorios.
- ***Plan de contingencia*:** Utilizar protocolos de comunicación a internet de bajo consumo.

7.9. Conexiones entre componentes errónea

- ***Descripción*:** Es posible que, haya equivocaciones a la hora de conectar componentes entre ellos.
- ***Probabilidad***: Alta.
- ***Impacto*:** Bajo.
- ***Consecuencia***: Perdida de tiempo invertido en la búsqueda del error.
- ***Prevención***: Repasar las conexiones con la ayuda del *datasheet* del componente.
- ***Plan de contingencia:*** Corregir las conexiones lo antes posible para no retrasarse en la planificación.

7.10. Presupuesto ajustado

- ***Descripción*:** Un presupuesto bajo conlleva comprar elementos de baja calidad y que los errores más comunes aumenten.
- ***Probabilidad***: Media.
- ***Impacto*:** Bajo.
- ***Consecuencia***: Errores comunes.
- ***Prevención***: Escoger componentes con buenas referencias -buenos comentarios en foros, etc-.
- ***Plan de contingencia:*** No se puede solventar ninguna contingencia ya que es aleatorio.

7.11. Falta de almacenamiento

- ***Descripción*:** La estación base tiene un almacenamiento interno muy escaso y se requiere la instalación de paquetes, librerías, etc.
- ***Probabilidad***: Alta.
- ***Impacto*:** Bajo.
- ***Consecuencia***: Error de instalaciones y actualizaciones futuras.
- ***Prevención***: Añadir una micro SD.
- ***Plan de contingencia:*** Prevenir el riesgo añadiendo un extra de almacenamiento desde el principio.

7.12. Datos erróneos

- ***Descripción*:** Sin un detector de errores la probabilidad de datos corruptos es más común.
- ***Probabilidad***: Alta.
- ***Impacto*:** Alto.
- ***Consecuencia***: Un producto final sin fiabilidad.
- ***Prevención***: Añadir un detector de errores como checksum para aplicar a los datos recogidos por los sensores.
- ***Plan de contingencia:*** Estudiar técnicas de detención de errores.

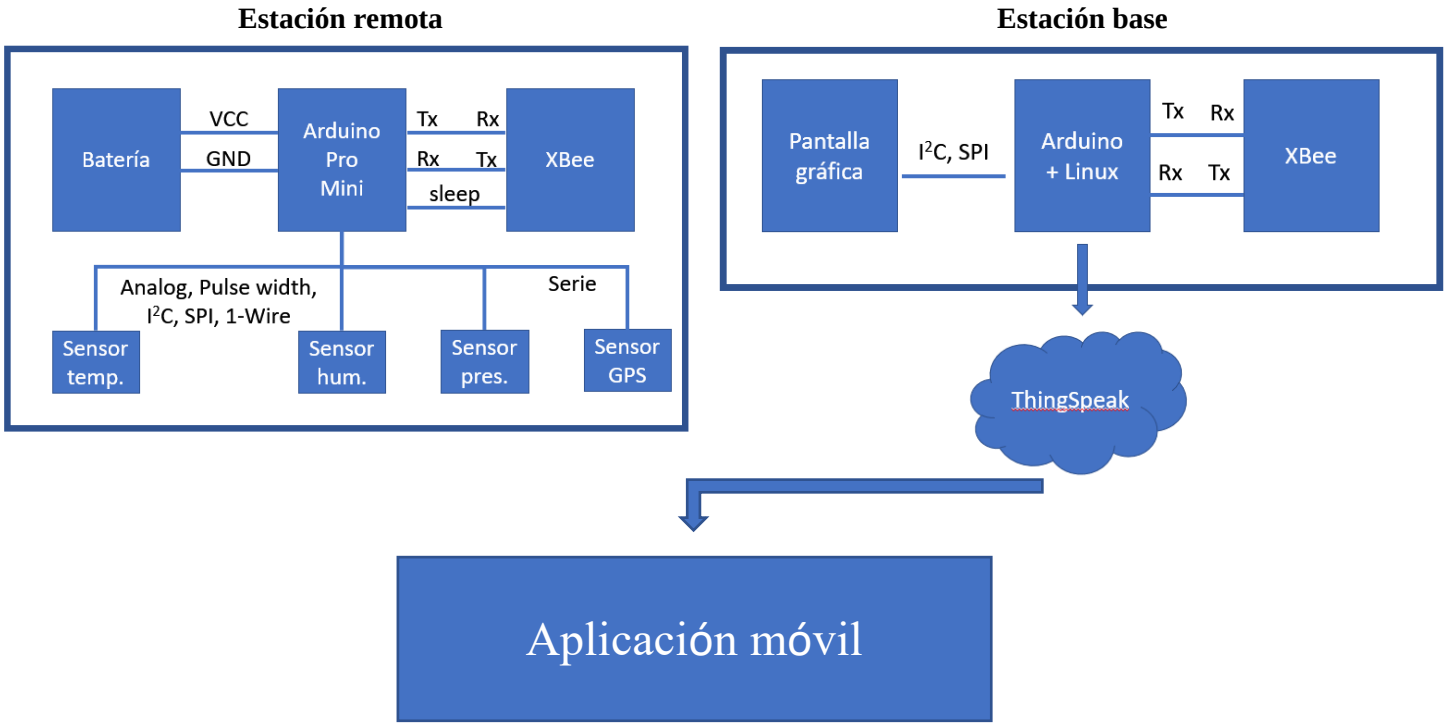
7.13. Autonomía insuficiente

- ***Descripción*:** La estación remota se situará en exteriores y requiere una autonomía completa. Sin un medio de prevención a este problema esta estación remota sería inservible.
- ***Probabilidad***: Alta.
- ***Impacto*:** Alto.
- ***Consecuencia***: La estación remota dejaría de funcionar.
- ***Prevención***: La estación base tendrá dos posibles fuentes de alimentación -batería y energía solar- que se gestionarán de la siguiente forma: cuando la placa solar sea capaz de suministrar más corriente de la que la estación es capaz de consumir, el excedente se destinará cargar la batería; cuando la placa solar no sea capaz de alimentar la estación, el déficit de corriente será suplido por la batería; en ausencia de luz, la batería será la encargada de alimentar la estación. Un circuito integrado diseñado ad-hoc para gestionar este proceso será el encargado de realizar esta gestión.
- ***Plan de contingencia:*** Planificar y estudiar la autonomía de esta estación remota para darle una vida mayor.

7.14. Incompatibilidad de versiones

- ***Descripción*:** Versiones de software que impidan la actualización de librerías, mejoras estables del software, etc.
- ***Probabilidad***: Media.
- ***Impacto*:** Medio.
- ***Consecuencia***: No mantener todos los equipos y librerías actualizadas implica que algunas funcionalidades imprescindibles queden obsoletas.
- ***Prevención:*** Actualizar todos los equipos y librerías actuando en consecuencia de ellas.
- ***Plan de contingencia:*** Mantenerse informado de dichas actualizaciones.

8. Diseño de alto nivel



La estación meteorológica que se plantea en este proyecto consta de cuatro bloques: estación remota, estación base, subida de datos y aplicación móvil.

La estación remota se encargará de enviar los datos recogidos por los sensores a la estación base. La estación meteorológica consta de tres sensores: temperatura/humedad, temperatura/presión y GPS. Una vez que los sensores capturan los datos, el microcontrolador crea una estructura de datos y se los pasa al módulo de conexión inalámbrica XBee para que los envíe. Dicha estructura está compuesta por los parámetros meteorológicos y un código de detección de errores denominado CRC. Este código no es más que el resultado de un cálculo con el que podemos verificar si en la transacción de datos ha habido algún error.

Una vez enviados los datos, la estación remota entrara en modo “sleep” de bajo consumo, ahorrando batería. Respecto a la autonomía de la estación remota, consta de dos posibles fuentes de alimentación -batería y placa solar- que se gestionan de la siguiente forma: cuando la placa solar sea capaz de suministrar más corriente de la que la estación es capaz de consumir, el excedente se destinará cargar la batería; cuando la placa solar no sea capaz de alimentar la estación, el déficit de corriente será suplido por la batería; en ausencia de luz, la batería será la encargada de alimentar la estación. Un circuito integrado será el encargado de realizar esta gestión.

Cuando el módulo XBee de la estación base haya recibido la estructura de datos y el código CRC no arroje indicios de que los datos están corruptos, mostraremos por pantalla los datos recibidos. A continuación, el microcontrolador Arduino y el microprocesador Atheros integrados en la estación base se comunicarán para que el primero le pase los datos al segundo, ya que la plataforma Linux que corre sobre el microcontrolador es la que tiene acceso a Internet por WiFi a través de la red "eduroam" para poder subir los datos a la plataforma de IoT (Internet of Things) ThingSpeak. El protocolo de HTTP se utilizará para la creación de un canal de almacenamiento de datos en ThingSpeak, mientras que la subida de datos al canal se realizará mediante el protocolo MQTT, dado que es un protocolo ligero diseñado para este tipo de operaciones en entornos IoT e Industria 4.0.

El consumo o la visualización de los datos se realiza a través de una aplicación móvil hecha con Ionic, un framework capaz de crear aplicaciones híbridas basadas en tecnologías web. Una vez el usuario haya iniciado sesión mediante su cuenta de Google, la aplicación se encargará de recoger los datos de ThingSpeak y de graficarlos. Además, y con el objetivo de explorar los beneficios del acceso a datos abiertos publicados por diferentes instituciones públicas como AEMET, la aplicación ofrece una comparativa de los datos recogidos por la estación remota con aquellos recogidos en el interior de Bizkaia, más concretamente en el Gorbea. La aplicación será capaz de actualizar los datos en tiempo real.

9. Planificación. Descripción de tareas, fases, equipos o procedimientos

A continuación, se describe la planificación llevada a lo largo del proyecto. A su vez se explicará las tareas y fases realizadas para conseguir el producto final. En el desarrollo de este TFG hemos contribuido dos personas: el director, quién actúa como jefe de proyecto; y el estudiante, que se encarga del análisis y el desarrollo.

9.1. Fases del proyecto

Este proyecto contiene 4 fases: investigación, desarrollo, testeo y memoria.

- ***Fase 1: Investigación.*** Investigación sobre proyectos similares que ofrecen resultados parecidos a los nuestros, junto con la búsqueda de herramientas innovadoras y aprendizaje de ellas.
- ***Fase 2: Desarrollo.*** Implementación de las herramientas y tecnologías para el desarrollo de los objetivos.
- ***Fase 3: Testeo.*** Evaluación de pruebas de la fase anterior.
- ***Fase 4: Memoria.*** Documentación de todo el trabajo realizado.

9.2. Descripción de tareas

Paquete de trabajo 1: Investigación

- ***Unidad de trabajo 1: Análisis de objetivos***

Duración	15 horas
Encargado	Alexander Saavedra
Descripción	Definir los objetivos que debe cumplir el TFG
Entradas	-
Recursos necesarios	Recurso ofimático
Salidas/Entregables	Objetivos y alcance del trabajo

Tabla 1: Análisis de objetivos

- ***Unidad de trabajo 2: Análisis de herramientas***

Duración	15 horas
Encargado	Alexander Saavedra
Descripción	Analizar herramientas nuevas e innovadoras que faciliten el trabajo
Entradas	-
Recursos necesarios	Internet
Salidas/Entregables	Tecnologías, herramientas

Tabla 2: Análisis de herramientas

- ***Unidad de trabajo 3: Formación***

Duración	40 horas
Encargado	Alexander Saavedra
Descripción	Formación de las herramientas nuevas y desconocidas
Entradas	Documentación
Recursos necesarios	Internet
Salidas/Entregables	Realizar el desarrollo del proyecto

Tabla 3: Formación

- ***Unidad de trabajo 4: Reunión semanal con el director***

Duración	9 horas
Encargado	Alexander Saavedra y Oskar Casquero
Descripción	Reunión semanal para la toma de decisiones
Entradas	Documentación
Recursos necesarios	Internet
Salidas/Entregables	Definir objetivos y requerimientos

Tabla 4: Reunión semanal con el director

Paquete de trabajo 2: Desarrollo

- ***Unidad de trabajo 1: Lectura de los sensores***

Duración	20 horas
Encargado	Alexander Saavedra
Descripción	Lectura de datos de los sensores que contiene la estación remota
Entradas	Datos meteorológicos
Recursos necesarios	Sensores, Arduino, Arduino IDE
Salidas/Entregables	Manejo de datos meteorlógicos

Tabla 5: Lectura de sensores

- ***Unidad de trabajo 2: Estructura de datos***

Duración	10 horas
Encargado	Alexander Saavedra
Descripción	Crear una estructura de datos para la transmisión de datos entre estaciones
Entradas	Datos y CRC
Recursos necesarios	Arduino Pro Mini, Arduino IDE
Salidas/Entregables	Envío de datos

Tabla 6: Estructura de datos

- ***Unidad de trabajo 3: Detección de errores en datos***

Duración	15 horas
Encargado	Alexander Saavedra
Descripción	Comprobar la integridad de los datos antes de ejecutar acciones
Entradas	Datos
Recursos necesarios	Arduino Pro Mini, Arduino IDE
Salidas/Entregables	Ejecutar acciones

Tabla 7: Detección de errores

- ***Unidad de trabajo 4: Modo "sleep" GPS***

Duración	10 horas
Encargado	Alexander Saavedra
Descripción	Activar el "modo" sleep del sensor GPS
Entradas	-
Recursos necesarios	Sensor GPS, Arduino Pro Mini, Arduino IDE
Salidas/Entregables	Consumo bajo

Tabla 8: Modo "sleep" GPS

- ***Unidad de trabajo 5: Modo "sleep" estación remota***

Duración	20 horas
Encargado	Alexander Saavedra
Descripción	Activar el "modo" sleep del Arduino Pro Mini
Entradas	-
Recursos necesarios	Arduino Pro Mini, Arduino IDE
Salidas/Entregables	Consumo bajo

Tabla 9: Modo "sleep" estación remota

- ***Unidad de trabajo 6: Transmisión de datos***

Duración	20 horas
Encargado	Alexander Saavedra
Descripción	Comunicación y transmisión de datos entre estaciones mediante el módulo XBee Pro
Entradas	Datos
Recursos necesarios	Arduino Pro Mini, Arduino Yun, Arduino IDE
Salidas/Entregables	Datos

Tabla 10: Transmisión de datos

- ***Unidad de trabajo 7: Pantalla gráfica***

Duración	30 horas
Encargado	Alexander Saavedra
Descripción	Visualización de logo UPV/EHU en una pantalla gráfica y muestreo de datos
Entradas	Bitmap logo UPV/EHU, Datos
Recursos necesarios	Pantalla gráfica, Arduino Yun, Arduino IDE
Salidas/Entregables	Muestreo de datos

Tabla 11: Pantalla gráfica

- ***Unidad de trabajo 8: Comunicación entre Arduino y Linux***

Duración	30 horas
Encargado	Alexander Saavedra
Descripción	Comunicar las dos caras que compone el Arduino Yun: Arduino + Linux
Entradas	-
Recursos necesarios	Arduino Yun, Arduino IDE
Salidas/Entregables	Acciones en Linux

Tabla 12: Comunicación entre Arduino y Linux

- ***Unidad de trabajo 9: Subida de datos a la nube***

Duración	50 horas
Encargado	Alexander Saavedra
Descripción	Realizar un script en Python para la subida de datos a la plataforma ThingSpeak
Entradas	Datos
Recursos necesarios	Python, datos, PyCharm Community
Salidas/Entregables	Datos almacenados en la nube

Tabla 13: Subida de datos a la nube

- ***Unidad de trabajo 10: Crear una aplicación multiplataforma***

Duración	100 horas
Encargado	Alexander Saavedra
Descripción	Crear una aplicación multiplataforma con Ionic Framework basado en tecnologías web
Entradas	-
Recursos necesarios	Internet, Navegador, Móvil, Visual Studio Code
Salidas/Entregables	Aplicación multiplataforma

Tabla 14: Crear una apliación multiplataforma

- ***Unidad de trabajo 11: Visualización de datos***

Duración	20 horas
Encargado	Alexander Saavedra
Descripción	Muestreo de datos mediante gráficas.
Entradas	Datos
Recursos necesarios	Datos, Internet, Navegador, Móvil, Visual Studio Code
Salidas/Entregables	Muestro de datos

Tabla 15: Visualización de datos

- ***Unidad de trabajo 12: Geolocalización***

Duración	10 horas
Encargado	Alexander Saavedra
Descripción	Mostrar mapa y la geolocalización de la estación remota en la aplicación móvil
Entradas	Coordenadas
Recursos necesarios	Internet, Navegador, Móvil, Visual Studio Code
Salidas/Entregables	Posición de la estación remota

Tabla 16: Geolocalización

- ***Unidad de trabajo 13: Reunión semanal con el director***

Duración	48 horas
Encargado	Alexander Saavedra y Oskar Casquero
Descripción	Reunión semanal para la toma de decisiones y ayuda
Entradas	-
Recursos necesarios	-
Salidas/Entregables	Problemas resueltos

Tabla 17: Reunión semanal con el director

Paquete de trabajo 3: Testeo

- ***Unidad de trabajo 1: Pruebas***

Duración	20 horas
Encargado	Alexander Saavedra y Oskar Casquero
Descripción	Testeo de la estación meteorológica y la aplicación móvil
Entradas	-
Recursos necesarios	Estación meteorológica, aplicación móvil
Salidas/Entregables	-

Tabla 18: Pruebas

Paquete de trabajo 4: Memoria

- ***Unidad de trabajo 1: Documentación del proyecto***

Duración	60 horas
Encargado	Alexander Saavedra
Descripción	Documentar todo el trabajo realizado
Entradas	-
Recursos necesarios	Recurso ofimático
Salidas/Entregables	Memoria TFG

Tabla 19: Documentación del proyecto

- ***Unidad de trabajo 2: Revisión de la memoria por parte del director***

Duración	10 horas
Encargado	Oskar Casquero
Descripción	Revisión de la documentación del proyecto
Entradas	Memoria
Recursos necesarios	Recurso ofimático
Salidas/Entregables	Memoria TFG Revisada

Tabla 20: Revisión de la memoria por parte del director

En la tabla 21 se muestra el total de horas que ha tenido este TFG.

Tareas	Estimación de horas
Investigación	**79**
Análisis de objetivos	15
Análisis de herramientas	15
Formación	40
Reunión semanal con el director	9
Desarrollo	**383**
Lectura de sensores	20
Estructura de datos	10
Detección de errores	15
Modo “sleep” GPS	10
Modo “sleep” estación remota	20
Transmisión de datos	20
Pantalla gráfica	30
Comunicación entre Arduino y Linux	30
Subida de datos a la nube	50
Aplicación multiplataforma	100
Visualización de datos	20
Geolocalización	10
Reunión semanal con el director	48
Testeo	**20**
Pruebas	20
Memoria	**70**
Documentación del proyecto	60
Revisión de la memoria por parte del director	10
TOTAL	**552**

10.Diagrama de Gantt

El total de horas invertidas en el trabajo ha sido de 552 horas. Ya que el trabajo comenzó a mediados de septiembre de 2017 y acaba a finales de febrero de 2018, se realiza una estimación de 24 horas semanales aprox. Dado que es un trabajo personal, está sujeto a mi disponibilidad diaria. Por lo tanto, se han incluido como días de trabajo los fines de semana, festivos y laborables indistintamente.

Para mostrar algo de claridad se ofrece a continuación un diagrama de *GANTT* adaptado a semanas.

Guía de colores:

- Naranja → Investigación
- Azul → Desarrollo
- Verde → Testeo
- Rojo → Memoria

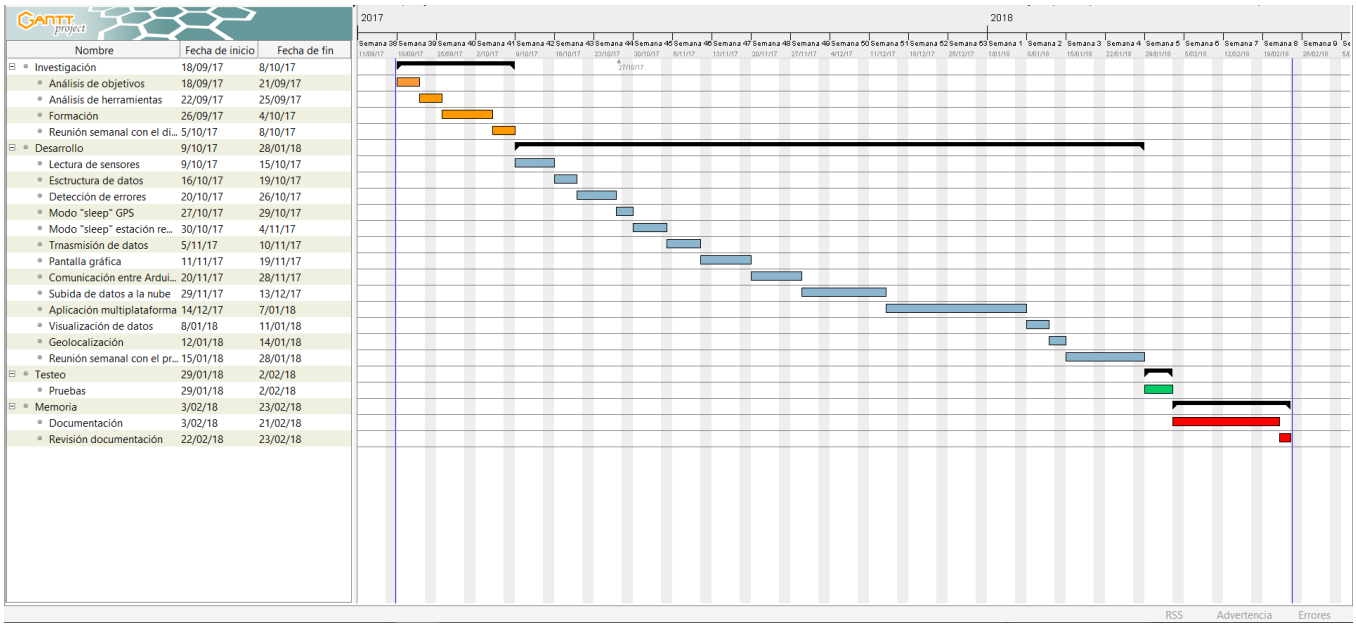


Figura 45: Diagrama de Gantt

11.Diseño de bajo nivel

La estación meteorológica está dividida en cuatro bloques: estación remota, estación base, subida de datos y aplicación móvil.

11.1. Estación remota

A continuación, se explica cada elemento de este primer bloque:

11.1.1. **Sensor temperatura/humedad**

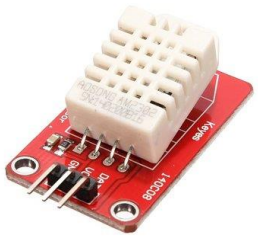


Figura 46: DHT22

***Descripción*:** El módulo DHT22 es un sensor de humedad y temperatura de bajo coste con una interfaz digital. El sensor está calibrado e incorpora un regulador de tensión para su conexión a fuentes de alimentación de 3.3V y 5V, por lo que no requiere componentes adicionales para la captura de los datos humedad relativa y temperatura.

Este sensor usa un protocolo de comunicación propio que ocupa sólo una conexión en uno de sus pines. Afortunadamente se han desarrollado librerías de Arduino que nos ahorran este trabajo y nos ofrecen funciones de comunicación sin necesidad de preocuparnos por la trama de datos que se envía y recibe. En este caso usamos la librería DHT de Adafruit (https://github.com/adafruit/DHT-sensor-library).

Características:

Especificaciones	Datos
Voltaje de alimentación	3.3 – 5.5V
Corriente durante medición	1 – 1.5mA
Corriente en modo espera	40 – 50µA
Humedad	0 – 100HR
Rango de temperatura	-40ºC a 80ºC
Precisión	*/.2% HR

Tabla 21: Características DHT22

Descripción de pines:

PIN	Nombre	Descripción
1	SDA	Datos
2	VCC	Fuente de alimentación
3	GND	Masa

Tabla 22: Descripción de pines DHT22

Conexiones con Arduino:

En la tabla siguiente se muestra la conexión del Arduino con el sensor DHT22:

Pin Arduino Pro Mini	Pin sensor DHT22
Pin digital 2	Pin 1 (SDA)
VCC (3.3V)	Pin 2 (VCC)
GND	Pin 3 (GND)

Tabla 23: Conexiones DHT22 con Arduino

Entradas/Salidas:

11.1.2. **Sensor de presión**

Figura 1.2: BMP180

***Descripción*:** Este sensor de precisión de BOSCH es una solución de bajo coste para detección y medida de la presión barométrica y temperatura. La presión atmosférica es la fuerza que ejerce el aire (atmósfera) sobre la superficie de la tierra. Por lo tanto, la presión atmosférica también varía con el clima, principalmente con la temperatura, pues ésta hace cambiar la densidad del aire.

Nuestro módulo BMP180 incluye, además del sensor, regulador de tensión para su conexión a fuentes de alimentación de 3.3V y 5V, resistencias de pull-up para I2C y condensadores de desacoplo. La comunicación con el microcontrolador se realiza a través del protocolo I2C.

Al igual que con el sensor DHT22, utilizaremos su correspondiente librería (https://github.com/LowPowerLab/SFE_BMP180) para abstraernos de las particularidades de la comunicación con el sensor.

Características:

Especificaciones	Datos
Voltaje de alimentación	5V
Rango de detección de presión	300 - 1100hPa
Precisión de temperatura	-40 a 85°C

Tabla 24: Características BMP180

Descripción de pines:

PIN	Nombre	Descripción
1	VCC	Fuente de alimentación
2	GND	Masa
3	SCL	I2C clock
4	SDA	I2C data

Tabla 25: Descripción de pines BMP180

Conexiones entre Arduino y módulo BMP180:

Las conexiones son como cualquier conexión I2C:

Pin Arduino Pro Mini	Pin sensor BMP180
VCC (3.3V)	Pin 1 (VCC)
GND	Pin 2 (GND)
Pin analógico 5 (A5)	Pin 3 (SCL)
Pin analógico 4 (A4)	Pin 4 (SDA)

Tabla 26: Conexiones entre Arduino Pro Mini y BMP180

Entradas/Salidas:

11.1.3. **GPS**

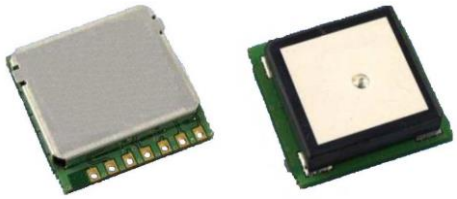


Figura 47: GPS-G622R

***Descripción*:** El GPS-622R es un módulo GPS inteligente integrado con una antena. El módulo contiene un motor de adquisición de 51 canales y un motor de seguimiento de 14 canales, que puede recibir señales de hasta 65 satélites GPS, permitiendo obtener una posición precisa. Como con todos los módulos GPS, se recomienda que el módulo tenga visión directa con los satélites para conseguir una correcta recepción de la señal.

El módulo de GPS tiene un tamaño pequeño y un relativo bajo consumo de energía tanto en modo adquisición como en modo seguimiento de satélites (unos 35mA). Es compatible con fuentes de alimentación de 3.3V y 5V.

El sensor GPS-622R se comunica con un microcontrolador mediante comunicación serie.

Dado que el módulo GPS no tiene modo “sleep” o de bajo consumo, es necesario diseñar una solución para evitar que el módulo GPS consuma corriente entre el periodo de muestreo de datos. Para ello, el módulo GPS se alimenta a través de un transistor BJT 2N2222A funcionando como conmutador controlado a través de un pin digital del Arduino.

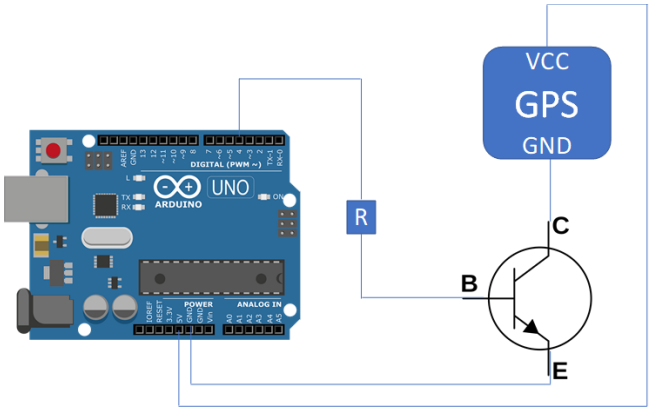


Figura 48: Esquema de conexiones entre GPS, transistor y Arduino

Cuando en el pin digital hay un cero lógico (0V), el transistor entra en corte y actúa como un interruptor abierto entre colector y emisor, de manera que el GPS queda desconectado. Sin embargo, cuando en el pin digital hay un uno lógico (3.3V), el transistor entra en saturación y actúa como un cortocircuito entre colector y emisor, de manera que el módulo de GPS se conecta a la alimentación. El valor de la resistencia R que hace que el transistor entre en saturación se calcula de la forma indicada a continuación. En este proyecto concreto el valor elegido para la resistencia es de 1K.

- **Transistor como conmutador:**

$$Ic = 35\text{mA}$$

$$Ib = \frac{Ic}{\beta} = \frac{35}{150} = 0.23mA$$

- **Red emisor-base:**

$$5 - Rb * Ib - 0.8 = 0 \rightarrow Rb = \frac{(5 - 0.8)}{0.23} = 18.26\ K\Omega$$

$$Rb \ < \ 18.26\ K\Omega$$

Cada vez que el módulo de GPS se apaga y se enciende para muestrear los datos de localización, se realiza una adquisición de satélites que conlleva un retardo en la obtención de los datos de unos 30 segundos (arranque “en frío”), lo cual repercute en el consumo de batería. Para reducir el retardo en la adquisición de los satélites, conectamos el pin VBAT del módulo de GPS a la alimentación del circuito. De esta forma, se consigue que el módulo de GPS recuerde los satélites de la última sesión y, así, se acelere la adquisición de los satélites al encender el módulo

de GPS (arranque “en caliente”), con lo que se minimiza el tiempo que el módulo de GPS debe estar encendido y, por ende, se reduce el consumo del corriente.

Para facilitar el procesamiento en Arduino de las sentencias NMEA que proporciona el GPS y que contienen la información de geolocalización, se ha utilizado la librería TinyGPSPlus (https://github.com/mikalhart/TinyGPSPlus).

Descripción de pines: En la siguiente tabla se describirá los pines de este módulo GPS. En este proyecto sólo se han usado del 1 al 5.

PIN	Nombre	Descripción
1	RX	Recepción
2	TX	Transmisión
3	GND	Masa
4	VCC	Fuente de alimentación
5	VBAT	Alimentación de backup
6	P1PS	Marca de 1 pulso por segundo
7	PSE_SEL	Selección de modo de motor de búsqueda

Tabla 27: Descripción de pines GPS G-622R

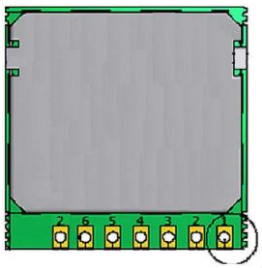


Figura 49: Pines GPS-G622R

Conexiones con Arduino:

Pin Arduino Pro Mini	Pin sensor GPS-G622R
Pin digital 5	Pin 1 (RX)
Pin digital 6	Pin 2 (TX)
GND	Pin 3 (GND)
VCC (3.3V)	Pin 4 (VCC)
Pin digital 7	Pin 5 (VBAT)

Tabla 28: Conexiones entre Arduino Pro Mini y GPS-G622R

Entradas/Salidas:

11.1.4. **Arduino Pro Mini**

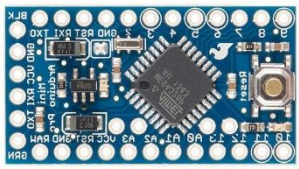


Figura 50: Arduino Pro Mini

Descripción: El Arduino Pro Mini es una tarjeta de desarrollo basada en el microcontrolador ATmega328. Cuenta con 14 pines de entradas/salidas digitales (de las cuales 6 se puede usar como salidas PWM), 6 entradas analógicas y un botón de RESET. Se puede conectar un conector de 6 pines para emplear un conversor FTDI de serie-USB a serie-TTL como interfaz para poder cargar los programas.

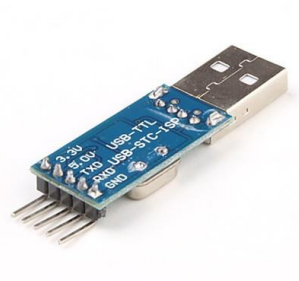


Figura 51: Conversor FTDI

El Arduino Pro Mini está diseñado para ser instalado de forma semi-permanente en los dispositivos. La tarjeta viene sin conectores, lo que permite usos diferentes, dependiendo de la aplicación que se le quiera dar.

Hay que mencionar que esta tarjeta cuenta con dos versiones: una que opera a 5V a una frecuencia de 16 MHz y otra que opera a 3.3V a una frecuencia de 8 MHz (la elegida en este proyecto).

***Características*:**

Especificaciones	Datos
Microcontrolador	Atmega328P
Voltaje de operación	3.3V
Voltaje de alimentación	7 – 9 V
Frecuencia de operación	16 MHz
Entradas/salidas analógicas	8/0
Entradas/salidas digitales	14/14
PWN	6
EEPROM (kB)	1
SRAM (kB)	2
Flash (kB)	32
UART	1
Puerto de programación y alimentación principal	Por medio de una tarjeta o un cable FTDI

Tabla 29: Características Arduino Pro Mini

***Descripción de pines*:**

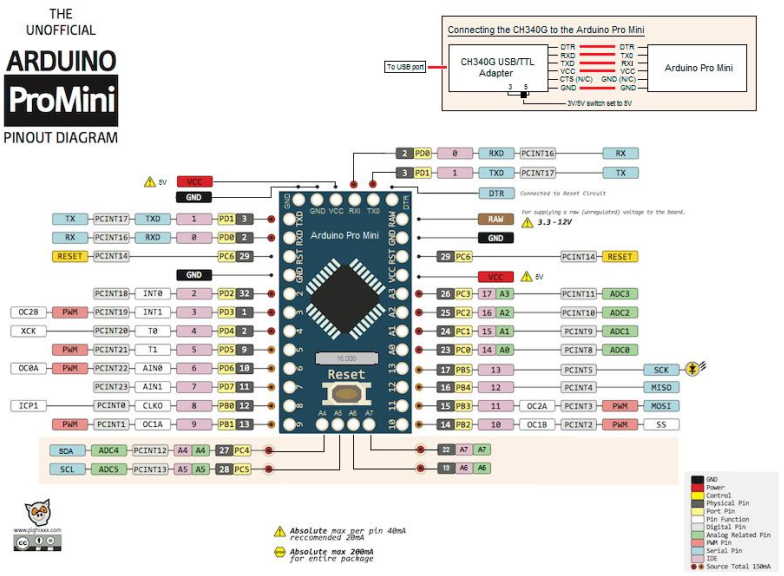


Figura 52: Descripción de pines Arduino Pro Mini

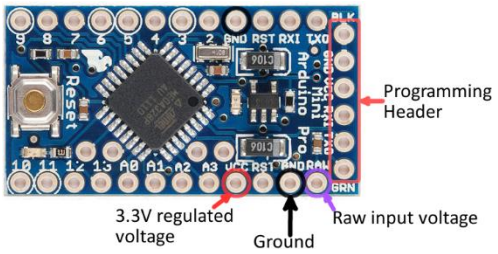


Figura 53: Pines para programar Arduino Pro Mini

Proceso en nuestra estación meteorológica:

Una vez descritos cada uno de los elementos de este primer bloque, a continuación, se describe el proceso que efectúa la estación remota.

Como todo programa en Arduino, la estructura está separada en tres partes principales:

- **Espacio de declaración inicial:** En esta primera parte se incluyen las variables globales para todo el código. Además, se incluyen las librerías externas que sean necesarias y se crean los objetos a partir de los constructores de las clases.

  En este espacio inicial se incluyen los archivos de cabecera de las librerías "SFE_BMP180.h", "Wire.h", "TinyGPS++.h", "DHT.h", "SoftwareSerial.h", así como los respectivos constructores. A continuación, se crean todas las variables necesarias en el programa, así como una estructura de datos y variables de tipo volatile que se utilizan tanto en el programa principal como en las rutinas de atención a la interrupción (ISR -Interrupt Service Routine-).
- **Función de configuración:** La función "setup()" alberga el código de ejecución única que sirve de configuración de los distintos elementos del programa.

  Inicializamos todos los sensores que previamente hemos declarado constructores. Además, definimos los pines que determinarán cuándo se deben dormir el GPS y el XBee. Por último, configuramos el modo "sleep" del XBee a través de los correspondientes comandos AT y el modo "sleep" del Arduino Pro Mini a través de los correspondientes registros asociados al "perro guardian" (WDT -Watch Dog Timer-).
- **Función de ejecución cíclica:** La función "loop()" es aquella cuyo código se ejecuta cíclicamente.

  Esta función comienza despertando el GPS y esperando la recepción de datos de localización válidos y actualizados. Cuando esto haya ocurrido, se desactiva la señal que activa el transistor BJT que actúa como conmutador del módulo GPS y el módulo GPS se apaga. A continuación, empieza la recogida de datos por parte de los sensores.

  Una vez se hayan obtenido todos los datos, estos se incluyen en una estructura a la que se añade un código CRC calculado mediante una XOR de dichos datos.

A continuación, se envían los datos por el puerto serie para que el XBee los retransmita de forma inalámbrica. Para finalizar, se duermen el XBee y el Arduino, por este orden. Este proceso se repetirá de forma periódica cada 60 segundos.

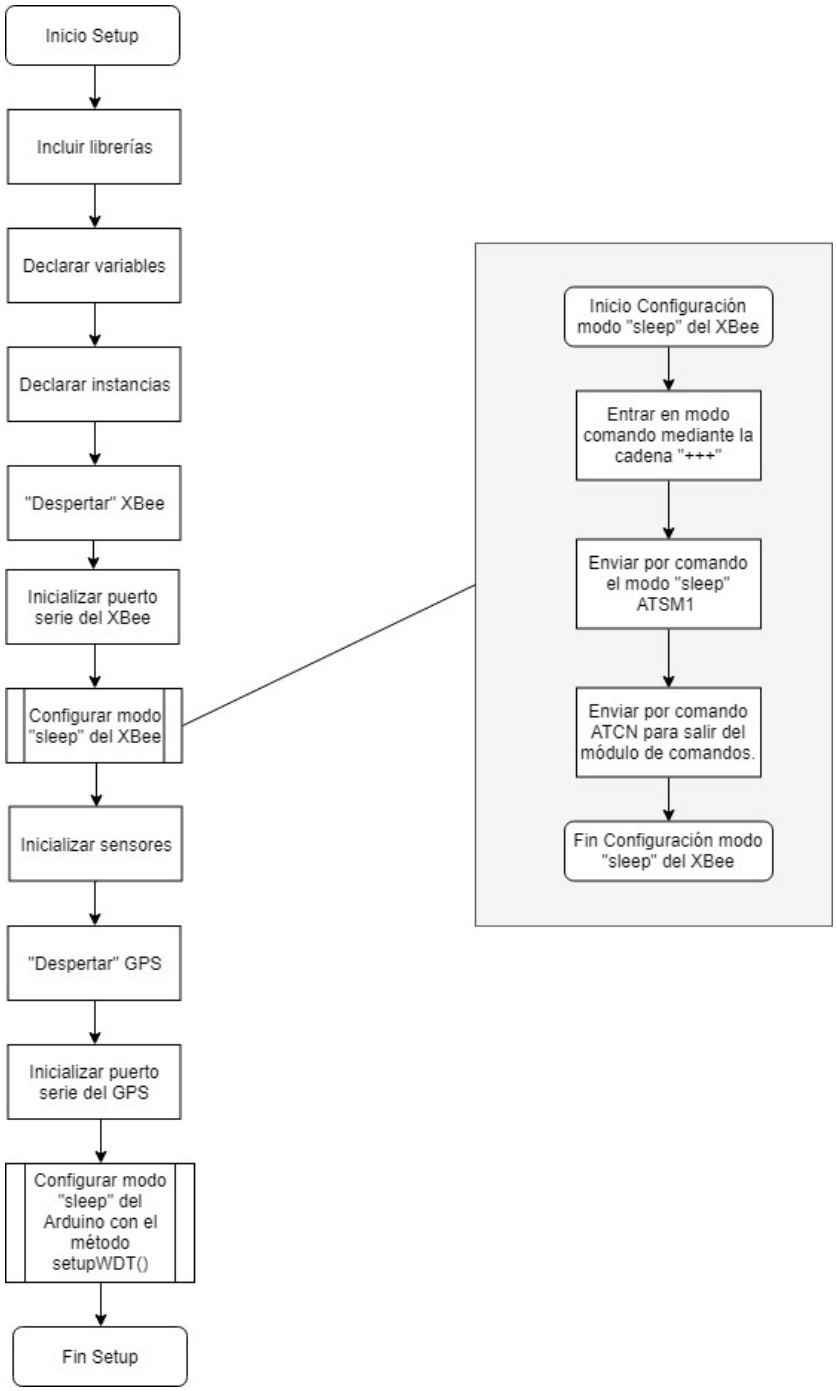


Figura 54: Diagrama de flujo de la declaración inicial y la función setup()

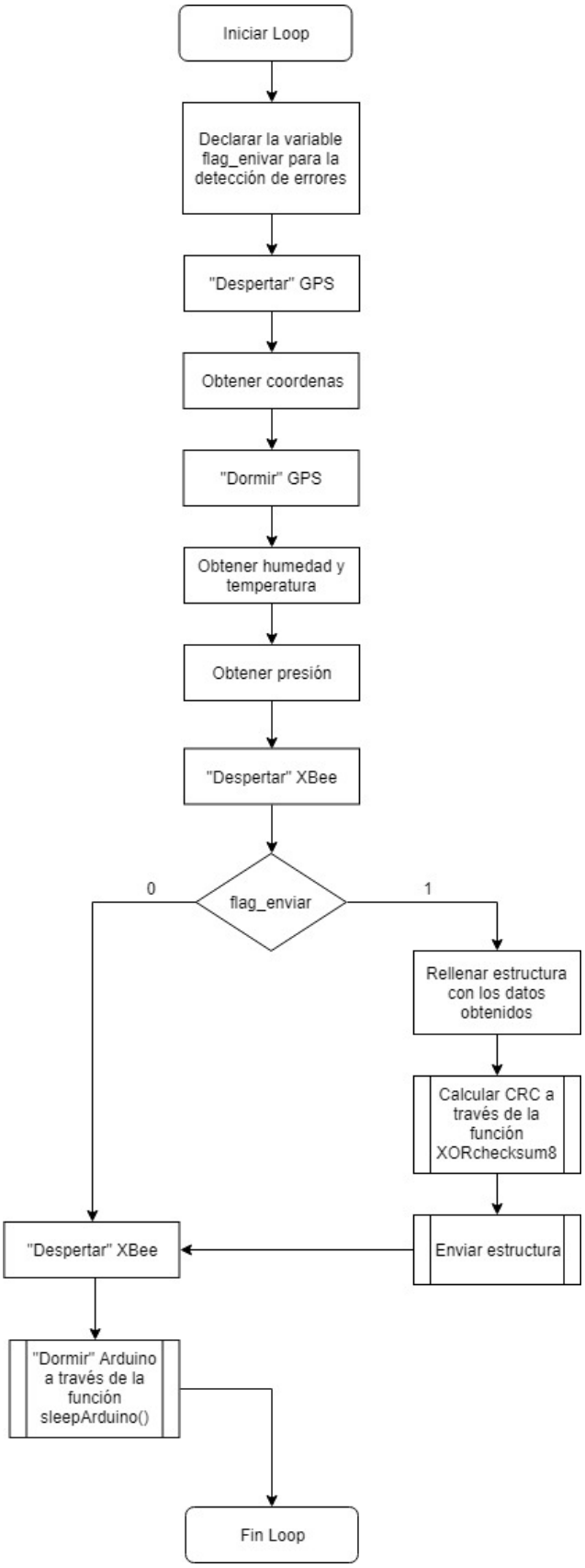


Figura 55: Diagrama de flujo de la función loop()

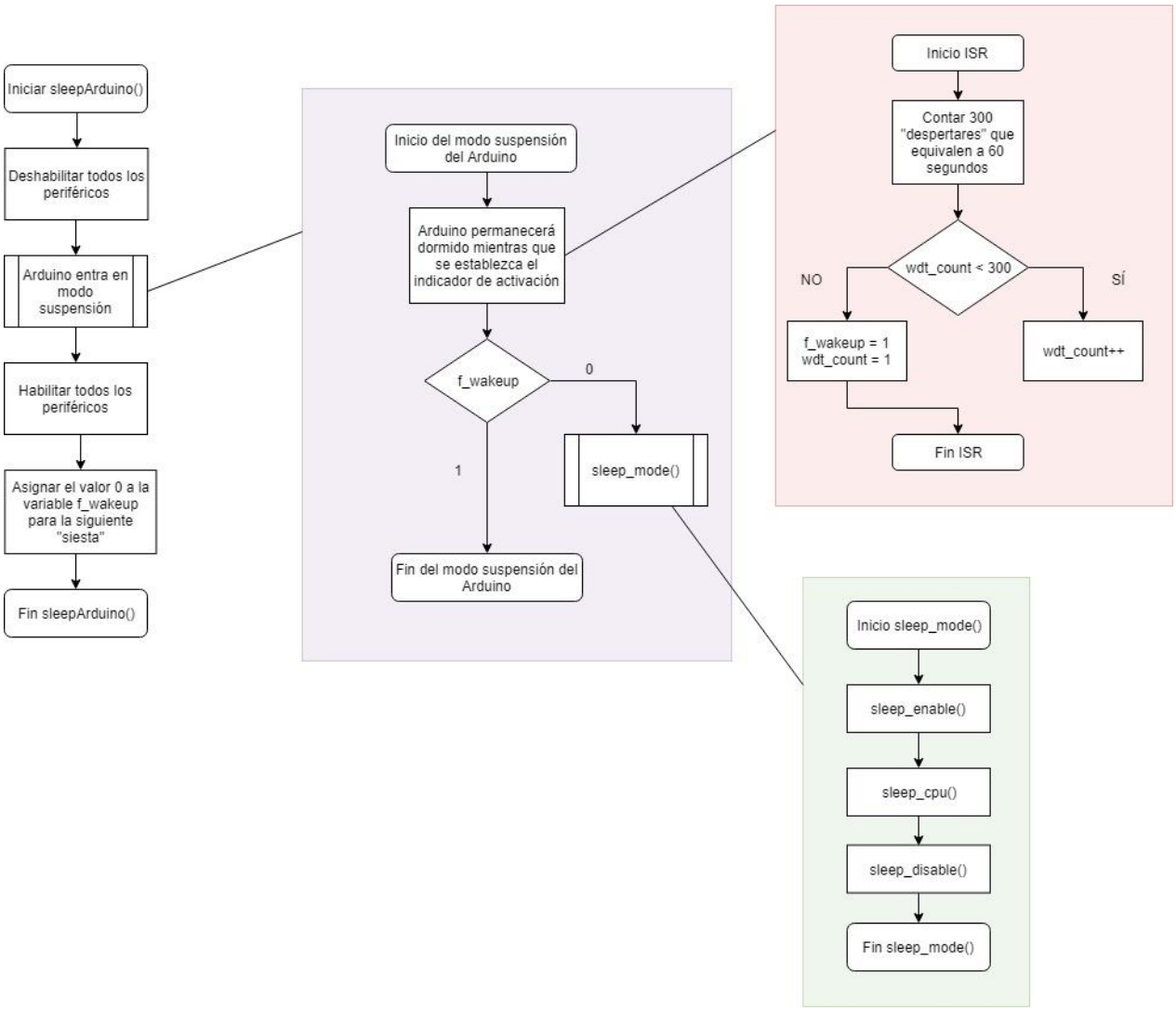


Figura 56: Diagrama de flujo de la función sleepArduino()

11.1.5. **Comunicación inalámbrica entre estaciones**

Figura 57: XBee PRO

***Descripción*:** Los módulos XBee son soluciones integradas que brindan un medio inalámbrico para la interconexión y comunicación entre dispositivos. Utilizan el protocolo de red llamado IEE 802.15.4 para crear redes FAST POINT-TO-MULTIPOINT (punto a multipunto) o para redes PEER-TO-PEER (punto a punto). Permiten una comunicación bidireccional entre microcontroladores u ordenadores.

Estos módulos han sido diseñados para aplicaciones que requieren de un alto tráfico de datos, baja latencia y una sincronización de comunicación predecible.

Hay pocas diferencias entre un XBee regular y un XBee PRO. La diferencia en cuanto a hardware es que el XBee PRO es un poco más largo. Con respecto a la comunicación, la versión Pro tiene un mayor alcance (1,6 km en línea directa), pero a costa de un mayor consumo de potencia. Los dos modelos se pueden mezclar entre sí; es decir, un XBee regular puede trabajar con un XBee Pro dentro de la misma red.

Funcionan a 3.3V y los pines no son tolerantes a 5V. Ya que la estación base contiene un Arduino Yun (una placa que funciona a 5V, es decir, cuyas señales funcionan con lógica de 0-5V), es necesario introducir un conversor bidireccional de tensión de 3.3V a 5V. Nosotros hemos utilizado un conversor de la marca Sparkfun.

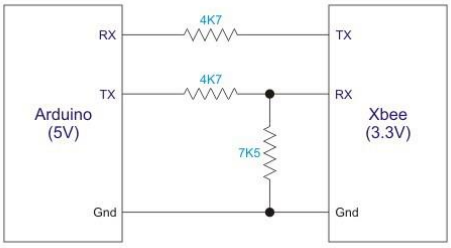


Figura 58: Divisor de tensión entre Arduino y XBee

***Características*:**

Especificaciones	Datos
Voltaje de alimentación	3.3V
Velocidad de transferencia	256kbps Max
Potencia de salida	60mW
Alcance	1500 metros aprox.
Antena integrada	Sí
Pines	6 pines ADC de 10-bit
Pines digitales	8 pines digitales IO
Configuración	Local o de forma inalámbrica
Comando	AT o API

Tabla 30: Características XBee PRO

Descripción de pines:

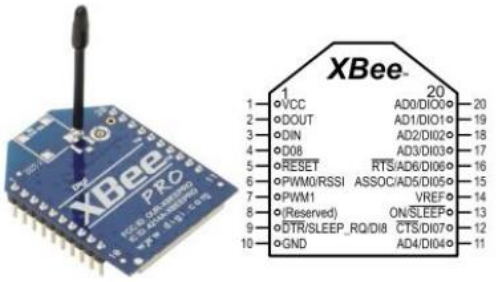


Figura 59: Descripción de pines XBee PRO

***Configuración*:** Para configurar y usar los módulos XBee es necesario descargar e instalar XBee Configuration and Test Utility (XCTU) que es un software multiplataforma que permite interactuar con los módulos mediante una interfaz gráfica. Esta aplicación incluye herramientas que hacen muy sencillo configurar y probar los módulos XBee.

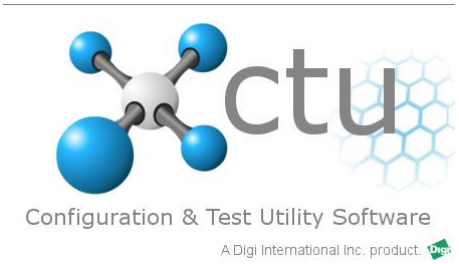


Figura 60: XCTU

XCTU nos sirve para configurar, inicializar, actualizar firmware y testear los módulos XBee, comunicándose por puerto serie a los módulos. Una ventaja de este software es que puedes ver rápidamente un resumen de todos los parámetros del módulo y una descripción de ellos.

Empezar a usarlo es tan simple como instalar el software XCTU, conectar el módulo a la placa de desarrollo Arduino, -en el caso de la estación remota conectar Arduino Pro Mini a un módulo FTDI que saca una interfaz USB- y luego enchufar el USB a nuestro PC.

Para configurar un módulo XBee con el XCTU, lo primero es poner el modo configuración y descubrir los módulos, seleccionando el puerto COM del USB al que he conectado la placa de desarrollo.

Para poder ver el módulo remoto es necesario configurar los parámetros DH y DL de la MAC del módulo remoto.

Cómo se comunican los dispositivos XBee:

Los dispositivos XBee se comunican entre ellos de forma inalámbrica enviando y recibiendo mensajes. Estos dispositivos no pueden gestionar los datos enviados o recibidos, sin embargo, pueden comunicarse con otros dispositivos a través del interfaz serie.

Los dispositivos XBee transmiten el aire los datos que llegan del puerto y transmiten al puerto serie cualquier dato que llega por el aire. Los microcontroladores o los PCs pueden controlar que envían los dispositivos XBee y gestionan los mensajes inalámbricos entrantes.

Por lo tanto, tenemos dos tipos de comunicación en los dispositivos XBee:

- Comunicación inalámbrica: es la comunicación entre los módulos XBee, estos módulos deben de ser parte de la misma red y usar la misma frecuencia de radio.
- Comunicación serie: es la comunicación entre el módulo XBee y el microcontrolador o el PC a través de un puerto serie.

En la comunicación inalámbrica los módulos transmiten y reciben información a través de la modulación de las ondas electromagnéticas. Para que se realice la transmisión ambos módulos deben estar en la misma frecuencia y en la misma red. Esto se determina por parámetros:

- **Channel (CH)** es la frecuencia usada para comunicar, es decir, el canal dentro de la red.
- **Personal Area Network Identifier (ID)** es un identificador único que establece que los módulos están en la misma red.

Un módulo XBee sólo recibirá y transmitirá datos a otros XBee dentro de la misma red (mismo ID) y usando el mismo canal (mismo CH).

Ahorro de energía:

Los módulos XBee tienen capacidades de ahorro de energía. Estos se pueden poner en estado sleep y apenas consumir energía, pudiendo llegar a una duración de batería de varios años.

El protocolo 802.15.4 contiene cuatro modos básicos para el modo sleep que se puede dividir en dos categorías: modo de espera controlado por pin y modo de velocidad cíclica. Por defecto el modo sleep está siempre deshabilitado.

- SM (Sleep Mode)
- ST (Time before sleep)
- SP (Cyclic sleep period)
- **Modo pin-controlled sleep:** este modo es controlado por el Sleep_RQ (pin 9) de forma que cuando es puesto a HIGH (3.3V) entre en modo sleep. (SM = 1)
- **Modo Cyclic Sleep mode:** el módulo se despierta y vuelve a modo sleep con una programación fija basada en el tiempo. Con SM = 4 se activa el modo y con SM = 5 además de activar el modo cíclico, permite activar a través del pin 9, siendo una mezcla de ambos modos. En estos dos modos se debe configurar los parámetros ST y SP.

Los pines de los módulos relacionados con el modo sleep son el pin 9 que pone el módulo en modo sleep cuando está a HIGH (3.3V) y el pin 13 es una salida que se pone HIGH cuando está despierto a LOW cuando está en modo sleep. Este pin se puede conectar a un led o a una entrada de microcontrolador.

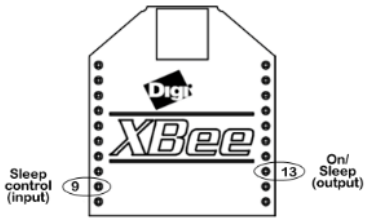


Figura 61: Pines sleep de XBee

Comandos AT:

Para entrar en modo comando a un XBee hay que poner la cadena "+++" y esperar a que nos devuelva un "OK" el módulo. El propósito es leer o cambiar la configuración del módulo Xbee.

El siguiente ejemplo muestra cómo se debería escribir una configuración:

- AT: comprueba la conexión con el módulo.
- ATCN: se sale del módulo de comandos.
- ATWR: Escribe la configuración actual a la memoria no volátil y persiste después de iniciar de nuevo el módulo.

Modelos de comunicación:

Existen dos tipos de comunicación y en ambos casos es bidireccional:

- **Punto a punto:** Para que se establezca la comunicación los módulos deben estar en el mismo canal (CH) y en la misma Network ID (ID), además para iniciar la comunicación es necesario saber la dirección MAC de 64-bit del destinatario.
- **Punto a multipunto:** En este modelo un módulo puede comunicarse con un módulo o múltiples módulos que estén en la misma red. Esta comunicación implica un nodo central coordinador con varios remotos conectándose al nodo central.

Señal y rango de frecuencia:

La distancia de alcance de la señal de los módulos XBee está afectada por diversos factores:

- Algunos materiales pueden reflejar las ondas de radio provocando interferencias. En particular materiales metálicos.
- Las ondas de radio pueden ser absorbidas por objetos en su camino.
- Las antenas pueden ajustarse para incrementar la distancia.
- La línea de visión puede ayudar a incrementar la fiabilidad de la señal.

Conexiones con Arduino:

Para la comunicación entre estación remota y estación base se ha implementado un módulo XBee Pro para cada una de ellas.

Las siguientes tablas muestran las conexiones de cada una de las estaciones:

- **Conexiones con Arduino Pro Mini:**

Pin Arduino Pro Mini	Pin sensor XBee PRO
VCC (3.3V)	Pin 1 (VCC)
Pin digital 9	Pin 2 (TX)
Pin digital 8	Pin 3 (RX)
Pin digital 4	Pin 9 (SLEEP)
GND	Pin 10 (GND)

Tabla 31: Conexiones Arduino Pro Mini con XBee Pro

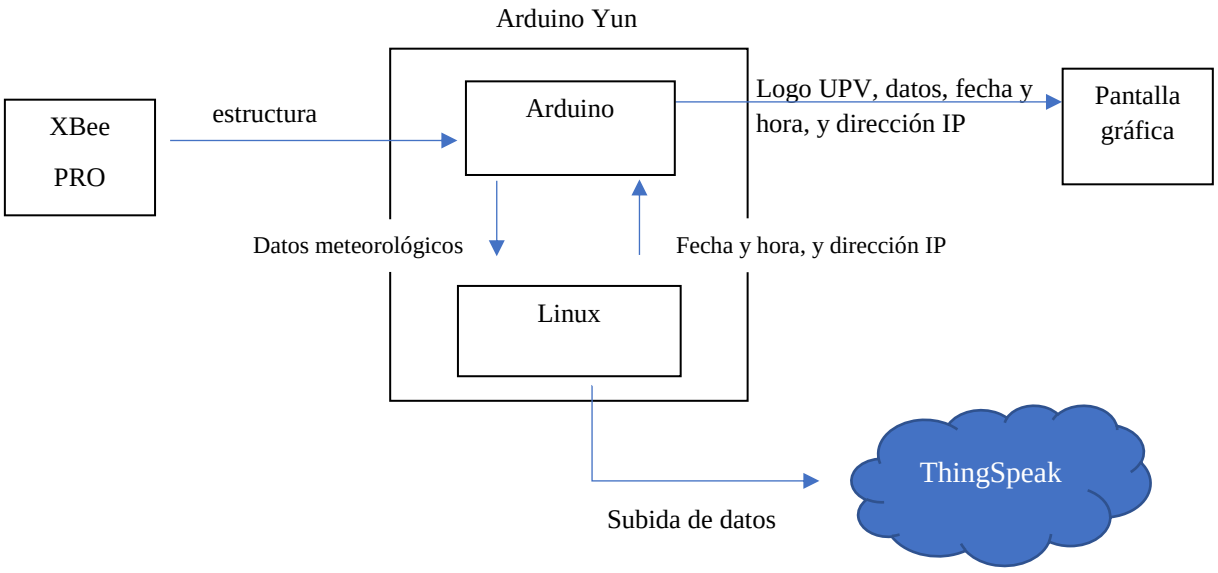
- **Conexiones con Arduino Yun:**

Pin Arduino Yun	Pin sensor XBee PRO
VCC (5V)	Pin 1 (VCC)
Pin digital 9	Pin 2 (TX)
Pin digital 8	Pin 3 (RX)
GND	Pin 10 (GND)

Tabla 32: Conexiones Arduino Yun con XBee Pro

El módulo XBee se encuentra en cada una de las estaciones, por lo tanto, antes de explicar el flujo de trabajo de este elemento mostramos el siguiente bloque de nuestra estación meteorológica, para a continuación analizar el trabajo de cada XBee.

11.2. Estación base



11.2.1. **Módulo XBee en estación remota**

Una vez hayamos recibido los datos meteorológicos de los sensores, el siguiente paso es añadirlos en una estructura de datos. Cuando trabajamos con sistemas de comunicación o datos almacenados, existe la posibilidad de que los datos se corrompan. Es decir, que uno o varios bytes de la secuencia cambien o incluso se pierdan por completo.

Con frecuencia necesitaremos comprobar la integridad de los datos antes de ejecutar acciones. Para eso podemos añadir un checksum, un valor de pequeño tamaño calculado a partir de un bloque de datos de tamaño mayor cuyo propósito es detectar errores introducidos en la secuencia.

Es por ello que este checksum lo enviaremos juntos a los datos, para que al realizar el envío la estación base calcule nuevamente el checksum con los datos recibidos y se comprueba la integridad de los datos.

Existen múltiples funciones de checksum, con diferentes tasas de detección de fallos y requisitos de cálculo. Por otro lado, cuando más largo sea el checksum mayor es la fiabilidad de este, pero mayores son las necesidades de comunicación/almacenamiento adicional para guardar el checksum.

Finalmente, ningún sistema de checksum es infalible. Como casi siempre, hay un compromiso entre fiabilidad, longitud, y carga de cálculo.

En esta ocasión hemos utilizado el checksum XOR, también llamado LCR (Longidtudinal Redundancy Check), el cual es ampliamente utilizado por su sencillez y rapidez de cálculo. Se calcula procesando la secuencia de datos en fragmentos de N bits (en nuestro caso de 8 bits) y ejecutando la operación XOR en todos los fragmentos.

```
uint8_t XORchecksum8(const byte *data, size_t dataLength) {
  uint8_t value = 0;
  for(size_t i = 0; i < dataLength; i++) {
    value ^= (uint8_t)data[i];
  }

  return ~value;
}
```

Figura 62: Longidtudinal Redundancy Check

Como decimos, el checksum XOR es sencillo y rápido de ejecutar, motivos por lo cual es muy extendido. Sin embargo, la tasa de detección de errores es bastante limitada.

Dicho esto, el XBee PRO mediante el método "write" enviará la estructura a otros módulos XBee.

11.2.2. **Módulo XBee en estación base**

Cuando el módulo XBee de esta estación base reciba información, a continuación, se realizará la detección de errores como anteriormente hemos mencionado -calcular checksum de los datos recogidos-.

11.2.3. **Arduino Yun**

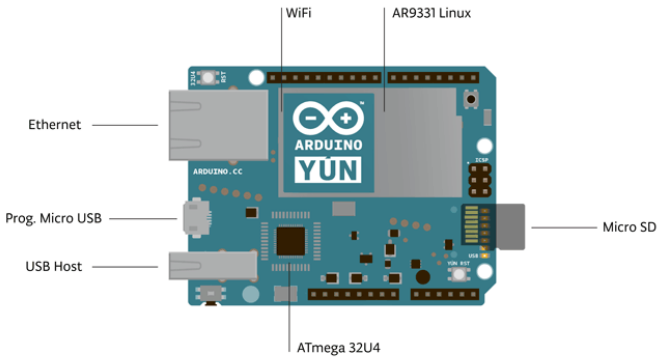


Figura 63: Arduino Yun

***Descripción*:** El Arduino Yún combina la potencia de un sistema Linux, junto con la sencillez característica de un microcontrolador. Combina el chip del modelo Leonardo (ATmega32U4) junto con un módulo SoC (System-on-Chip) Atheros AR9331 sobre el que corre una distribución de Linux llamada Linino, basada en OpenWRT. Una de las características más interesantes es que soporta red cableada Ethernet y WiFi.

El chip Arduino está conectado al módulo Linux mediante un puerto serie, por lo que es posible delegar procesos pesados a la máquina Linux integrada en la placa.

Características:

Especificaciones	Datos
Voltaje de alimentación	3.3V
Procesador	Atheros AR9331
Arquitectura	MIPS @400MHz
Puerto Ethernet	IEEE 802.3.10/100Mbit/s
Conexión WiFi	IEEE 802.11b/g/n
USB Type A	2.0 Host/Device
Lector de tarjetas	Micro-SD
RAM	64 MB DDR2
Memoria Flash	32 MB

Tabla 33: Características Arduino Yun

Conectividad:

Dispone de dos conexiones de red. Una red ethernet 10/100 Mbps y otra WiFi (IEEE 802.11 b/g/n, 2,4GHz) que puede montarse como cliente o como punto de acceso.

Conexión entre procesadores:

Para comunicar el pequeño ATMega32U4 con el módulo Linux, se utiliza la librería Bridge, que facilita mucho las cosas y es soportada directamente por el equipo Arduino. El puerto serie del Atheros AR9331 está conectado al puerto serie hardware del ATmega32U4. El puerto serie del Atheros AR9331 es un acceso a la consola Linux (CLI -Command Line Interface-), lo que permite lanzar procesos en Linux desde Arduino.

Varios paquetes de gestión del sistema de archivos y administración ya están preinstalados por defecto (incluso el intérprete de Python) y la librería Bridge permite también instalar y lanzar aplicaciones propias con ese mismo sistema

Una de las ventajas más interesantes, es que la placa (del lado del ATmega32U4) puede ser programada directamente por WiFi a través del módulo Linux.

Es una placa llena de posibilidades. También cabe destacar que dispone de un zócalo para memoria MicroSD que permite almacenar datos en ella como páginas web, datos logeados o cualquier otra cosa que necesitemos, ampliando aún más las posibilidades de la placa. En nuestro caso, hemos utilizado este recurso para la instalación de librerías y almacenaje de ficheros, como el script.

Alimentación:

Aunque se aconseje alimentar el Arduino Yun a través de su puerto micro USB con 5V, también lo podemos hacer a través de su pin Vin mediante una fuente de alimentación regulada a 5V, ya que Arduino Yun no tiene ningún regulador de tensión.

Memoria:

La parte clásica de Arduino Yun, el ATmega32u4 tiene 32KB, de los cuales 4KB están ocupados por el sistema de arranque –bootloader-. El bootloader es un programa que se ejecuta inmediatamente antes de ejecutar el programa que hay en la memoria flash al que cede el control cuando finaliza su ejecución. Del mismo modo cuenta con una SRAM de 2.5 KB y una EEPROM programable de 1KB.

Por su parte, el chip Atheros AR9331 cuenta con 64 MB de RAM DDR2 y 16MB de memoria flash, en la cual se ha cargado la distribución Linux Linino.

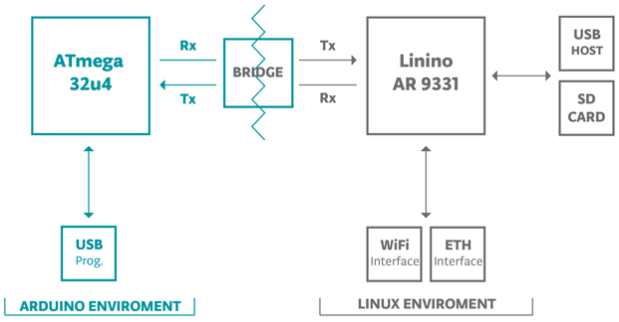


Figura 64: Las dos caras de Arduino Yun

Descripción de pines:

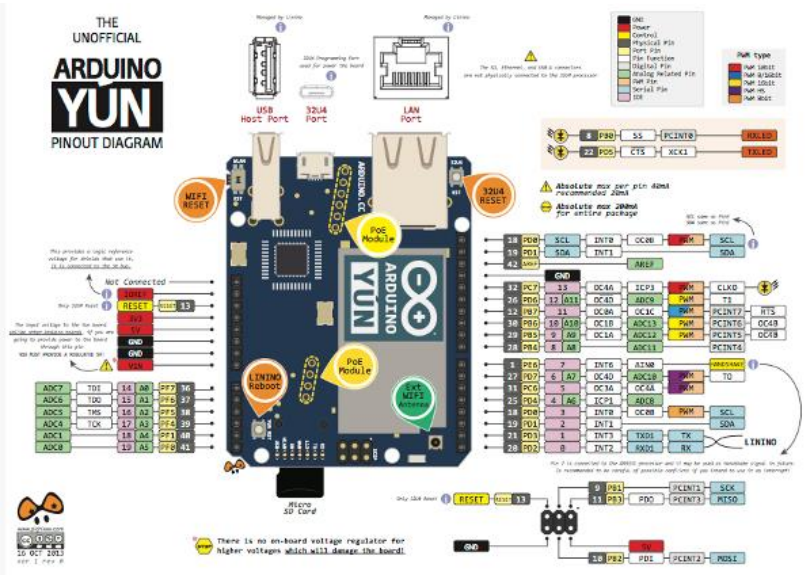


Figura 66: Descripción de pines Arduino Yun

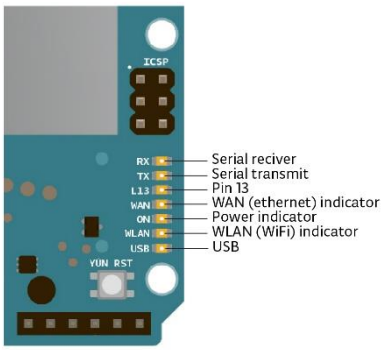


Figura 65: LEDs Arduino Yun

***Entradas/Salidas*:**

Si bien es cierto que el chip AR9331 tiene varios pines de entrada y salida, estos nos son accesibles, por lo que toda la parte de entrada y salida de datos a través de pines se hace como de costumbre, es decir, mediante los pines analógicos y digitales del chip ATmega32u4.

Al igual que en otras placas, existen pines con funciones particulares:

- **Serial:** El Arduino Yun dispone de dos puertos serie:
  - Un puerto serie que va sobre la conexión USB y que sirve para comunicar el ATmega32u4 con el ordenador o con otros dispositivos USB. Es el puerto serie a través del cual mandamos los mensajes de *debugging*.
  - Los pines 0 y 1. En este caso no se pueden utilizar porque este puerto serie está físicamente conectado al Atheros AR9331, y cuya comunicación puede gestionarse desde el ATmega32u4 a través de la librería Bridge.

- **TWI:** pin 2 (SDA) y pin 3 (SCL). Interrupciones externas, es decir, los pines 3, 2, 0, 1 y 7 actúan como interrupciones numeradas de 0 a 4, respectivamente.
- **PWM:** pines 3, 5, 6, 9, 10, 11, 13. La resolución de PWM es de 8 bits.
- **SPI**: en el conector marcado como ISCP/SPI.
- **LED:** al igual que ocurre en otros dispositivos, tenemos un led incorporado en la placa, conectado en este caso al pin 13.
- Entradas analógicas A0 – A11 en los pines digitales 4, 6, 8, 9, 10 y 12.
- **AREF:** referencia de voltaje para las entradas analógicas. Con este pin podemos cambiar el límite superior de 5V para adaptarlo a nuestras necesidades.
- Botones de RESET:
    - **Yun RST:** resetea el procesador AR9331, es decir, vacía la RAM y termina los programas en ejecución en el entorno Linux.
    - **32U4 RST:** reinicia el chip ATmega32u4.
    - **WLAN RST:** Presionado durante 5 segundos hará que el led azul de WLAN comience a parpadear, indicando que el entorno Linux y la configuración WiFi se están reiniciando, es decir, Yun actuará como un AP con IP 192.128.240.1 y SSID "Aurdinln Yun-direccionmac". Presionado durante 30 segundos dejaremos el entorno Linux tal y como venía el día que compramos nuestro Arduino.

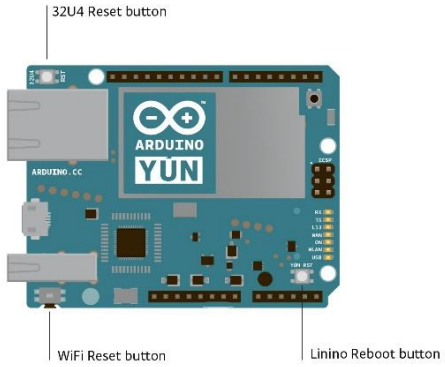


Figura 67: Botones RESET de Arduino Yun

Proceso en nuestra estación meteorológica:

Al igual que el Arduino Pro Mini, el Arduino Yun también tiene un proceso que debe ejecutar:

En esta ocasión la estructura también está separada en tres partes principales:

- **Espacio de declaración inicial:** Las librerías que son necesarias en este bloque son "Process.h" -para la comunicación entre Arduino y Linux-, "SoftwareSerial.h" -para la comunicación con XBee emulando un puerto serie- y "U8glib.h" -para la pantalla gráfica-

. Como constructores y/o variables que se declaren están una estructura de datos y la pantalla gráfica.

- **Función de configuración:** Una vez inicializadas las librerías, mostramos el logo de la UPH/EHU a través de la pantalla gráfica como bienvenida. Este logo está diseñado mediante un mapa de bits que se encarga de representar únicamente los bits necesarios para dicho logo. Para poder dibujar un mapa de bits es necesario tener en cuenta una serie de parámetros:
    - Posición x del inicio del bitmap.
    - Posición y del inicio del bitmap.
    - Número de bytes del bitmap en horizontal.
    - Y altura en pixeles del bitmap.

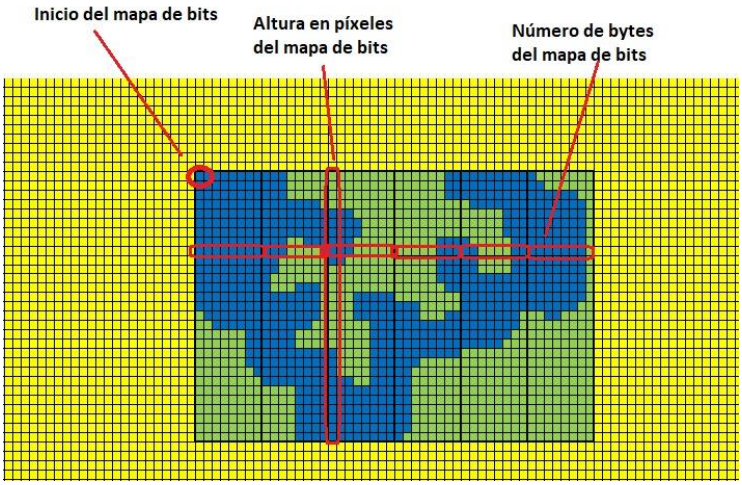


Figura 68: Parámetros Mapa de bits

Es decir, una vez decidido el gráfico que se quiere mostrar, es necesario definir los parámetros anteriores. Dentro de cada byte, cada bit tendrá un valor de 1 o 0. Si un bit vale 1 el programa lo interprete como que debe dibujar ese bit, y la celda se volverá de color negro. En caso de que valga 0, no lo dibuja.

Ya que para crear un bitmap manualmente requiere de mucho tiempo y esfuerzo, se crea una plantilla mediante Excel que permite realizar lo anterior y a la vez visualizar el gráfico que se está construyendo.

A continuación, para facilitar la conexión al Yun por SSH con el objetivo de realizar tareas de mantenimiento, al arrancar se muestra en pantalla la IP en lugar de la fecha de la última muestra hasta que se recibe la primera muestra. Para obtener la IP utilizamos un comando lanzado desde Arduino.

- **Función de ejecución cíclica:** Esta función comienza comprobando que el XBee dispone de algún mensaje pendiente enviado por la estación remota. Si no hay ningún mensaje la función cíclica seguirá hasta que reciba información.

Una vez recibida la estructura leemos los datos añadiendo cada parámetro recibido a la estructura creada en dicho programa. Como ya hemos comentado anteriormente, una vez leído el checksum, calculamos uno nuevo con los datos obtenidos. Todo irá bien siempre y cuando el checksum recibido y el calculado sean iguales. Si coinciden mostraremos los datos a través de la pantalla gráfica para, a continuación, subir los datos a la nube.

Es aquí cuando se comunican Arduino y Linux mediante la librería Bridge. Ya que este último es el único capaz de conectarse a Internet.

El siguiente bloque será el encargado de subir los datos a una plataforma IoT como ThingSpeak.

```
String date = getDateTime();
draw(date);

Serial.println(F("Ejecutando script python para subir datos a ThingSpeak..."));
long t1 = millis();
Process p;
p.begin(F("python"));
p.addParameter(F("/root/showdataV5.py")); // ejecutamos el script y la pasamos como parametro los datos obtenidos
p.addParameter(String(temp,2));
p.addParameter(String(hume,2));
p.addParameter(String(pressure,2));
p.addParameter(String(latitud,6));
p.addParameter(String(longitud,6));
p.run();

while(p.available() > 0) {
  char c = p.read();
  Serial.print(c);
}
p.close();

Serial.print(F("Tiempo de ejecución del script: "));
Serial.print((millis() - t1)/1000.0, 2); Serial.println("s");
```

Figura 69: Comunicación entre Arduino y Linux

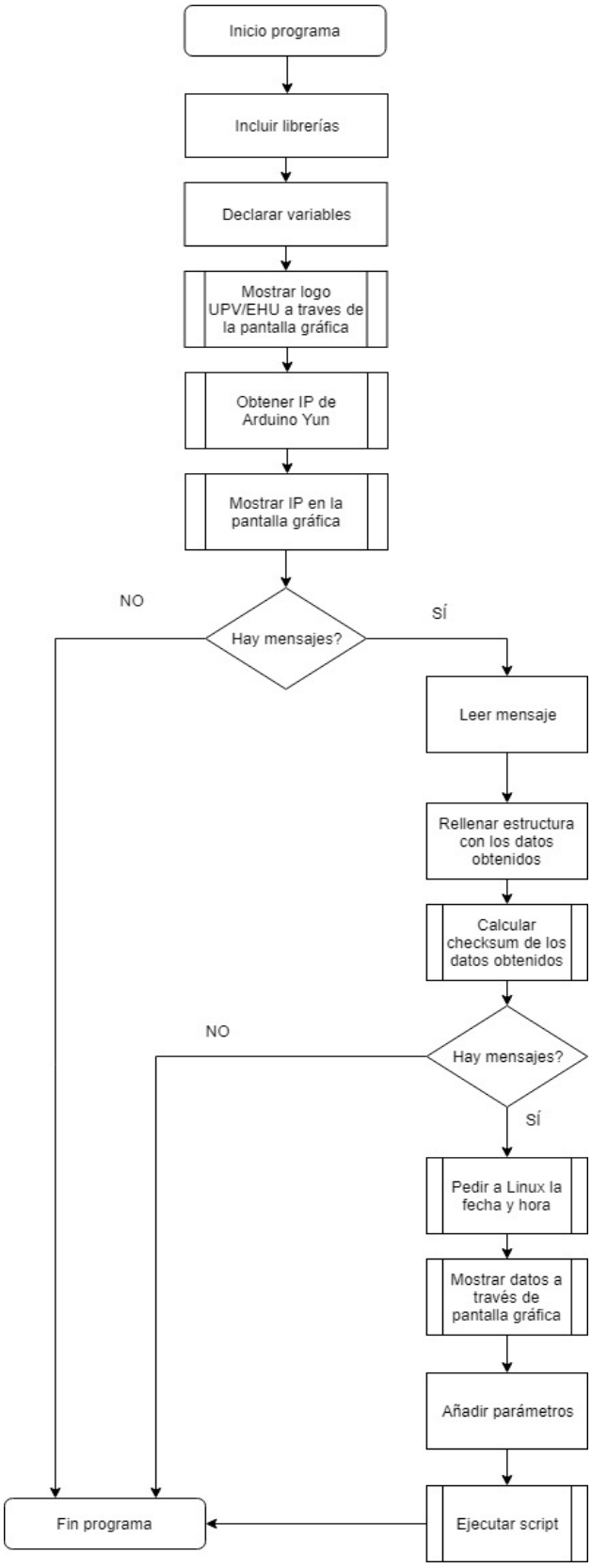
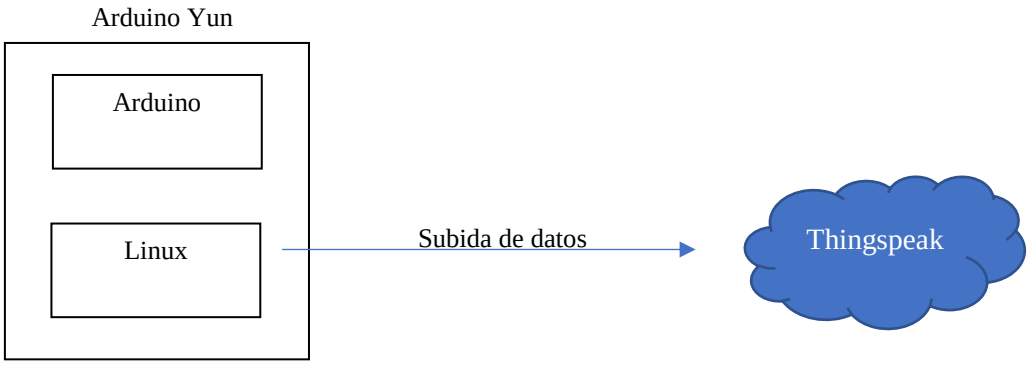


Figura 70: Diagrama de flujo de la recogida de datos de la estación base

11.3. Subida de datos



11.3.1. **Script para la subida de datos**

Este proceso lo hace un script en Python y el cual se repetirá siempre que los datos recibidos y la comunicación entre Arduino y Linux haya salido bien.

Figura 71: Python

Ya que este producto está pensado para que cada proveedor tenga su estación meteorológica y sus datos sean privados, el cliente debe crearse una cuenta gratuita en ThingSpeak, donde almacenaremos los datos. El usuario debe facilitarnos el *USER API* de dicha plataforma, o una vez registrado en la aplicación introducir ese campo en el apartado "Editar perfil". Este código *USER API* debe estar implementado en el script que ejecuta la estación base para poder realizar sus tareas.

Dicho esto, este script comenzará declarando las librerías necesarias para la subida de datos: la librería "requests" para la gestión del protocolo HTTP y "paho.mqtt" para la gestión del protocolo MQTT, y otras que serán necesarias para cálculos matemáticos, fechas, etc.

Las variables declaradas son los datos meteorológicos que el Arduino nos ha enviado. Una vez recogido dichos datos calcularemos la sensación termina y el punto de rocío mediante dos no tan simples formulas.

A continuación, se comprobará la existencia de un fichero "id.txt" en la carpeta /root de Linux. Si existe significa que no es la primera vez que se ejecuta este script y que el usuario ya tiene un canal propio creado en ThingSpeak.

Al leer este fichero obtenemos el ID del canal ya creado y usamos el protocolo HTTP para la obtención de datos del canal, ya que necesitamos las *APIs* WRITE y READ para la subida de datos. Estas *APIs* son un código de 16 dígitos que permite a una aplicación escribir y leer datos de un canal.

Si por el contrario "id.txt" no existe, se creará un canal para el usuario mediante el protocolo HTTP. Después de tener una respuesta correcta se llevará a cabo la creación del fichero antes nombrado y así no volver a ejecutar dicha acción.

Cualquiera de estos dos caminos conduce al mismo, es decir, si la ejecución es correcta comenzará la subida de datos mediante el protocolo MQTT. Se utiliza este protocolo ya que el envío de datos a aplicaciones requiere muy poco ancho de banda, además de tener un consumo realmente bajo comparado con el protocolo HTTP.

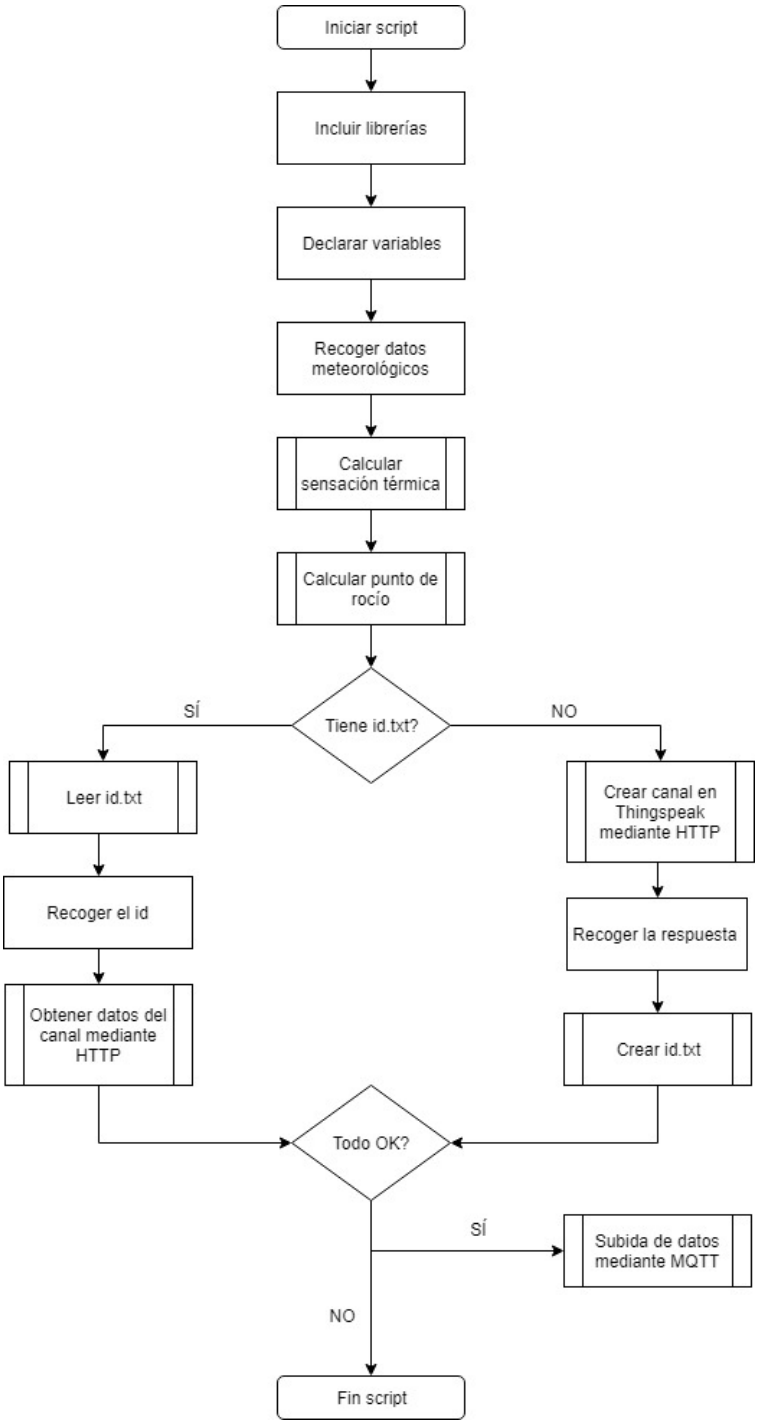


Figura 72: Diagrama de flujo de la subida de datos

11.4. Visualización de datos

Figura 73: Ejemplo de visualización de datos en la aplicación móvil

En este último bloque hemos optado por mostrar los datos obtenidos a través de una aplicación móvil híbrida -iOS, Android y Windows Phone-. La herramienta que se ha utilizado para poder llevar a cabo esto sin el aprendizaje nativo para cada uno de los sistemas ha sido Ionic.

11.4.1. **Ionic**

***Definición*:** Ionic es un framework para el desarrollo de aplicaciones híbridas, inicialmente pensado para móviles y tablets. Su característica fundamental es que usa por debajo Angular y una cantidad de componentes enorme, que facilita mucho el desarrollo de las aplicaciones.

Se trata de una estupenda herramienta para la creación de aplicaciones sorprendentes, pensada para obtener resultados de una manera rápida y con una menor inversión económica, ya que permite crear aplicaciones para distintas plataformas móviles con una misma base de código.

Qué es una aplicación híbrida: Una aplicación híbrida es aquella que permite desarrollar aplicaciones para móviles en base a las tecnologías web: HTML + CSS + JavaScript. Son como cualquier otra aplicación de las que puedes instalar a través de las tiendas de aplicaciones para cada sistema, por lo que en principio usuarios finales no percibirán la diferencia con respecto a otros tipos de aproximaciones diferentes, como las aplicaciones nativas.

Al ejecutarse con tecnologías web, los desarrolladores que ya tienen experiencia en el desarrollo en este medio pueden aprovechar sus conocimientos para lanzarse de una manera más rápida en el desarrollo de aplicaciones para móviles. Para hacer esto posible, las aplicaciones

híbridas se ejecutan en lo que se denomina un “web view”, que no es más que una especie de navegador integrado en el móvil y en el que solamente se ejecuta la aplicación híbrida.

Tecnología y herramientas:

- **Integración con Angular:** Ionic está desarrollado sobre el framework JavaScript Angular. Esto quiere decir que para el desarrollo con Ionic podemos apoyarnos en todas las ventajas de desarrollo con Angular, lo que nos permitirá contar con una excelente estructura de proyecto, uso de patrones de diseño de software y una buena gama de componentes y directivas.
- **Desarrollo basado en componentes:** Ionic nos ofrece ya de base una cantidad muy grandes de componentes que son capaces de trabajar perfectamente en dispositivos móviles con pantallas táctiles
- **TypeScript:** Otra cosa que viene dada por el desarrollo de Angular es el uso del lenguaje TypeScript, que no es más que un "superset" de JavaScript. Dicho de otra forma, TypeScript es JavaScript, pero con añadidos pensados para mejorar el trabajo por parte de los desarrolladores, haciéndonos más productivos. La mayor aportación de TypeScript al lenguaje JavaScript es la posibilidad de definición de tipos para las variables, pero en general aporta mucho más y además nos permite usar todas las mejoras de ES6 y algunas de ES7 en las aplicaciones.

Los componentes de Ionic ya vienen adaptados al dispositivo de manera estética. Quiere decir que, cuando se compila una aplicación para iOS el componente se visualizará de manera diferente que cuando se compila para Android. En Android usará Material Design mientras que en iOS usará las guías de diseño definidas por Apple.

Esto es una ventaja en sí, porque las personas disfrutarán de aplicaciones con una experiencia de usuario cercana a la que están acostumbrados en su teléfono y nos evita a los desarrolladores la necesidad de trabajar más para conseguir este efecto. Sin embargo, como autores de las aplicaciones con Ionic también somos capaces de alterar el diseño de las aplicaciones, proporcionando una experiencia de usuario específica y original para nuestra propia aplicación.

- **Desarrollo y compilado de aplicaciones:** Con Ionic desarrollamos aplicaciones con las tecnologías web. Durante toda la etapa de desarrollo usaremos el navegador para visualizar las aplicaciones, lo que permite un flujo de trabajo muy productivo, ya que no se tiene que compilar. Aunque lo cierto es que, por la necesidad de transpilar (transformar + compilar) el código de TypeScript a JavaScript compatible con el navegador, no será tan rápido de visualizar los cambios como simplemente refrescar la página del navegador.

Una vez que la aplicación está construida se tiene que realizar el proceso de compilación, en el que se producen los ejecutables específicos para cada dispositivo. Ese proceso sí es un poco

Figura 74: JavaScript

más pesado para el procesador, pero se hace solo cuando tenemos que lanzar una nueva versión que se subirá a las tiendas de aplicaciones.

- **Apache Cordova:** Ionic se basa también en Apache Cordova para la implementación de las aplicaciones. Hay partes, como el acceso a los componentes nativos del dispositivo, en las que se usan plugin que nos proporciona Apache Cordova principalmente. Actualmente también Ionic es proveedor de algunos plugin para trabajo con la parte nativa del teléfono. Aquí como nativo nos referimos a elementos como la cámara, acelerómetro, teclado virtual, etc. Todos esos elementos se pueden usar desde las aplicaciones de Ionic, con los correspondientes plugin nativos, que forman una especie de puente entre el desarrollo con JavaScript y el teléfono.

Apache Cordova también es el software que nos permite compilar el desarrollo realizado con Ionic con tecnologías web en aplicaciones para móviles instalables vía tiendas de aplicaciones.

Ionic CLI es el intérprete por línea de comandos de Ionic, que contiene una serie de herramientas útiles para realizar, de una forma sencilla, muchas tareas habituales en el desarrollo y producción de aplicaciones con Ionic.

11.4.2. **Aplicación de nuestra estación meteorológica**

Para poder adquirir las funcionalidades que ofrece esta aplicación debemos registrarnos o iniciar sesión si ya tenemos una cuenta creada. Tenemos varias opciones de registro: vía correo electrónico, Google, Facebook o Twitter.

Una vez dentro, lo que nos ofrece la pantalla principal son dos gráficos lineales de dichos datos meteorológicos ya obtenidos. La primera gráfica la forman los datos ambientales que hemos obtenido de los sensores, es decir, temperatura, humedad y presión atmosférica. En la segunda, visualizamos parámetros calculados en base a los parámetros recogidos.

Tendremos un menú lateral donde por una parte podemos hacer lo básico de una aplicación móvil, editar perfil, cambiar de contraseña o cerrar sesión. Por otro lado, ofrecemos al usuario la posibilidad de ver gráficas comparativas de nuestros datos con datos del interior

ofrecidos por AEMET. Además, de saber la última geolocalización de nuestra/s estación/es remota/s.

Figura 75: AEMET

Por último, añadir que la aplicación se encuentra de momento en tres idiomas disponibles: castellano, inglés y euskera.

12.Resultados

En este apartado, se detallan las partes más relevantes que han sido desarrolladas sin entrar en detalles en lo relativo al funcionamiento desde el punto de vista de un usuario final, ya que este aspecto se detalla en otros apartados.

Como bien se ha ido diciendo a lo largo del documento, esta estación meteorológica consta de dos elementos fundamentales:

- ***Estación remota.*** Su funcionalidad principal es obtener los datos recogidos por los sensores que la componen, además de transmitir los datos a la estación base. La siguiente figura muestra el resultado final de la estación remota.

Figura 76: Estación remota

- ***Estación base.*** Recoger la información enviada por la estación remota y transmisión de datos a la nube. La siguiente figura muestra el resultado final de la estación base.

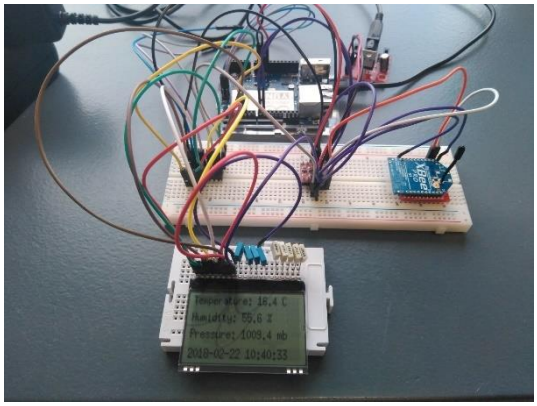


Figura 77: Estación base

Por otro lado, se describe e identifica la visualización y monitorización de los datos recogidos mediante los sensores a través de una aplicación móvil multiplataforma.

Una vez el usuario haya iniciado sesión en la aplicación obtenemos la monitorización de los datos.

En esta visualización se tiene, por un lado, la primera gráfica, que representa el histórico de valores de los parámetros asociados (temperatura, humedad y presión). Por otro lado, en la siguiente gráfica se representan la sensación térmica y el punto de rocío; resultados calculados a través de los parámetros recogidos por lo sensores. Como puede apreciarse en la siguiente figura, se visualizan rangos que varían desde cinco minutos a más de una hora. Esto es por las pruebas realizadas con las estaciones.

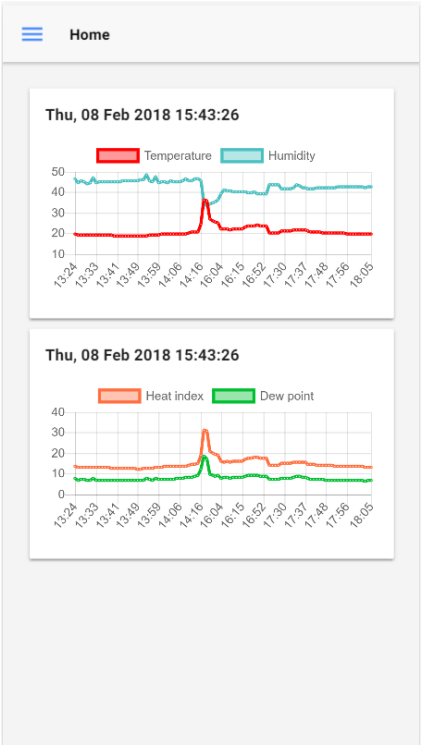


Figura 78: Monitorización de los datos recogidos por los sensores

Desde el menú lateral, que es el panel general para administrar la aplicación, se definen los componentes más importantes de la aplicación, aquí tenemos los siguientes módulos:

- ***Time series.*** Desde este módulo, podemos visualizar las entradas que nos llegan vía HTTP desde la nube y además tener una comparativa con los datos ofrecidos por el *API* de AEMET.
- ***Geolocation:*** Aquí existe la posibilidad de interactuar con un mapa y obtener la última localización y registro de la estación remota.
- ***About us.*** Módulo que muestra la documentación del equipo.
- ***Edit profile.*** Editar los datos del usuario, desde el nombre hasta el canal de ThingSpeak que quieres que se muestre en la aplicación.

- ***Change password.***

La siguiente figura muestra el acceso desde el menú lateral a estos módulos, describiéndose claramente la separación entre dichos módulos.

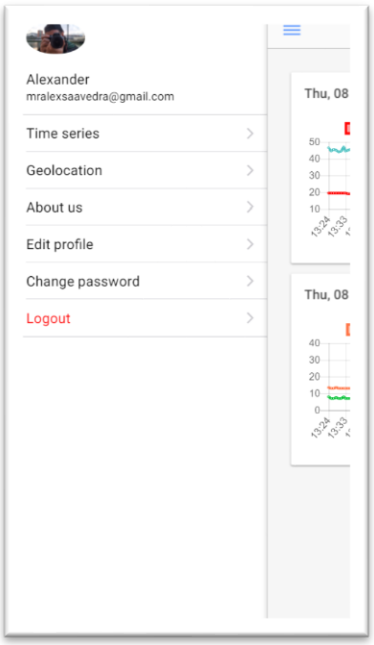


Figura 79: Menú lateral de la aplicación

Con base en el esquema de la figura anterior, a continuación, se describen estos cuatro componentes principales.

En el módulo de *Time series,* se pueden ver las entradas de datos procedentes de los sensores. En la figura 80, se muestra un simple ejemplo de parámetros recibidos en el sistema, que son temperatura y humedad. Se puede ver en dicha figura el valor de los datos y el tiempo transcurrido desde su última actualización. Este módulo lo componen los siguientes parámetros: temperatura, humedad, presión, sensación térmica y punto de rocío.

Además, en la misma gráfica encontramos los datos de la aplicación AEMET y así tener una comparativa con el interior de Bizkaia, más concretamente con el Gorbea.

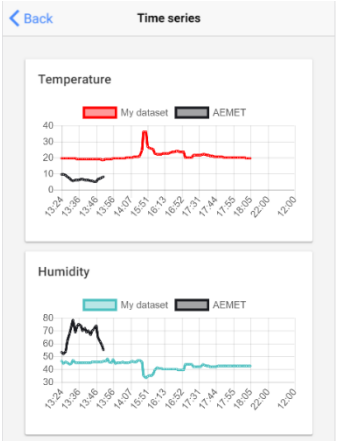


Figura 80: Time Series de la aplicación

Para ilustrar de forma representativa donde se encuentra la estación remota, en el módulo de *geolocalización* tenemos la posibilidad de interactuar con un mapa. Debe tenerse en cuenta, que tenemos la última posición mediante un *Google Mark* con el que podemos optar a los últimos parámetros registrados por los sensores.

Figura 81: Geolocalización de la estación remota

Los módulos restantes ya que no forman parte de la base principal de una plataforma IoT que es la visualización de datos obviamos a no comentar. Los módulos restantes sólo nos dan la oportunidad de saber o editar la información acerca de nuestro perfil.

La aplicación se puede descargar desde el siguiente enlace:

https://drive.google.com/drive/folders/1Jlytlye7AwdDOAVW9m4vW6V1sKpu0FVa

13.Descripción del presupuesto

En este punto se verá la evaluación económica del proyecto, incluido su coste total.

Se dividirá en los siguientes apartados:

13.1. Personal:

Para la elaboración del trabajo se necesita el trabajo de un desarrollador. Teniendo en cuenta que un programador junior cobra 6,5€/hora y que el proyecto dura 552 horas, el coste es de 3.588€.

13.2. Software:

- ***Licencia de Windows 10*** gratuita mediante *DreamSpark* (portal de la universidad junto a Microsoft que permite a los estudiantes descarga de software).
- ***Gantt Project:*** licencia gratuita.
- ***Google Chrome:*** licencia gratuita.
- ***Visual Studio Code:*** licencia gratuita.
- ***PyCharm Community:*** licencia gratuita durante 30 días.
- ***XCTU:*** licencia gratuita.
- ***MQTT.fx:*** licencia gratuita.
- ***Arduino IDE:*** licencia gratuita.
- ***GitHub IDE:*** licencia gratuita.
- ***Putty:*** licencia gratuita.
- ***WinSCP:*** licencia gratuita.
- ***Google Keep:*** licencia gratuita.
- ***Dropbox:*** licencia gratuita.
- ***ThingSpeak:*** licencia gratuita con limitación de datos.

Gastos totales de software: 0€

13.3. Hardware:

- ***Portátil MSI:*** valorado en 1000€ con una vida media de 1 año. Haciendo uso prácticamente total del ordenador durante la elaboración del TFG, tanto para desarrollo como para documentación. Con un uso dedicado al TFG del 33% a lo largo del proyecto.
    - Amortización anual: 1000/1 = 1000€.
    - Gasto del ordenador: ((1000 * 14semanas) *0.33/52semanas = **88,85 euros.**

- ***Samsung Galaxy S6:*** valorado hoy en día en 300€, con una vida de 2 años y medio, y utilizado para el testeo de la aplicación móvil de nuestra estación meteorológica. Uso de él del 5% del proyecto.
    - Amortización anual: 300/2,5 = 120€.
    - Gasto del ordenador: ((1000 * 7semanas) *0.05/52semanas = **7,69euros.**
- ***DHT22:*** Sensor de temperatura/humedad usado en la estación remota para la recogida de los mismos.
    - **Precio:** 2,40€/unidad
- ***BMP180:*** Barómetro digital que mide la presión del aire recogida por nuestra estación remota.
    - **Precio:** 1,70€/unidad
- ***GPS-G622R:*** Utilizado para obtener las coordenadas geográficas de la estación remota.
    - **Precio**: 3,10€/unidad
- ***Transistor****:* Cortar la alimentación del GPS entre muestreos mediante un transistor actuando como conmutador.
    - **Precio:** 0,41€/unidad
- ***Arduino Pro Mini 328 – 3.3V/8MHz:*** Circuito inteligente de la estación remota.
    - **Precio:** 20€/unidad
- ***Convertidor de USB a TTL:*** Para cargar los programas en el Arduino y configurar el XBee.
    - **Precio:** 8,99€/unidad
- ***Batería LiPo de 2200 mAh***
    - **Precio:** 12,85€/unidad
- ***XBee PRO 60mW con antena:*** Módulo colocado en cada una de las estaciones para la comunicación inalámbrica entre ellas.
    - **Precio:** 42€/unidad * 2 = 84€
- ***Adaptador DIP para XBee:*** Esta placa facilita la conexión de los módulos XBee a una protoboard.
    - **Precio:** 2,50€/unidad * 2 = 5€

- ***Módulo adaptador de 5V a 3.3V:*** Los módulos XBee funcionan a 3.3V y los pines no son tolerantes a 5V. Desde Arduino podemos alimentar un módulo XBee, pero la comunicación serie en Arduino es a 5V y en el módulo XBee es a 3.3V.
  - **Precio:** 12€/unidad
- ***Arduino Yun:*** Circuito inteligente de nuestra estación base.
  - **Precio:** 60€/unidad
- ***Pantalla gráfica:*** Visualización de logo UPV/EHU y muestro de datos.
  - **Precio:** 16,81€
- ***Protoboard y cables.***
  - **Precio:** 20€

Gastos totales de hardware: 343,8€

Gastos	Coste
Personal	3.588
Software	0
Hardware	343,8
TOTAL	**3.931,8**

Tabla 34: Gastos totales

14.Conclusiones

Llegados a este punto, se está en condiciones de afirmar que se ha cumplido con el objetivo principal del proyecto. Se ha desarrollado un sistema capaz de visualizar factores ambientales captados remotamente. Básicamente se ha desarrollado una aplicación bajo el paradigma del IoT. Se ha creado una estructura que permite la implementación de distintos módulos de una manera rápida y sencilla gracias al uso de patrones de diseño. El código generado a lo largo de la vida del proyecto es altamente reutilizable y flexible, lo que facilita la creación de distintas vías y oportunidades de desarrollo para continuar con el trabajo realizado en futuros desarrollos.

En este proyecto se han combinado diferentes tecnologías, algunas de ellas ciertamente novedosas. El adquirir los conocimientos necesarios sobre estas tecnologías para desarrollar la idea de proyecto ha sido menos tedioso de lo que pudiera esperarse en un principio. La base de conocimiento proporcionada a lo largo de estos años académicos ha sido de notable utilidad a la hora de abordar problemas desconocidos surgidos en el proyecto. Los paradigmas de la orientación a objetos, la implementación de patrones, mantener los factores de calidad del software han sido piezas claves para poder desarrollar este proyecto.

Hubiese sido deseable haber utilizado otra tecnología más eficaz para la creación de la aplicación multiplataforma, ya que Cordoba sólo es capaz de generar una *web view* y aunque a simple vista esto no dé problemas de eficiencia, siempre será mejor utilizar herramientas como React Native o lenguajes de programación nativos. A pesar de lo cual, la integración de esta tecnología ha resultado simple, fácilmente legible y modular.

El IoT parece ser un concepto que va a revolucionar la forma en la que vemos la tecnología en un corto periodo de tiempo, por lo que ha resultado muy interesante trabajar bajo un paradigma de tal naturaleza. Si bien es cierto que las fases de captura y transmisión de datos no encajan con las competencias de un ingeniero de software, por lo menos a niveles de investigación, las fases de análisis de datos y la conexión de servicios son áreas extensas que dan pie a una infinidad de usos para esos datos.

15.Bibliografía

Adafruit Industries, Unique & fun DIY electronics and kits. Recuperado de https://www.adafruit.com/

AEMET OpenData. *Sistema para la difusión y reutilización de la información de AEMET*. Recuperado de https://opendata.aemet.es/

AM Radio Transmitter Key Assembly Series. *GPS Smart Receiver with Antenna.* Recuperado de https://www.rfsolutions.co.uk/downloads/1456826946DS-GPS622F.pdf

Aprendiendo Arduino. *Aprendiendo a manejar Arduino en profundidad.* Recuperado de https://aprendiendoarduino.wordpress.com/

Atmel (11/2016). Recuperado de http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf

Bricogeek. Placas Arduino, Robótica, Electrónica, Raspeberry Pi. Recuperado de http://tienda.bricogeek.com/

Celemín, A (2009). El tiempo y la salud humana. *Meteorología Práctica.* Capítulo 18. Recuperado de http://www.paranauticos.com/notas/meteorologia/punto-de-rocio.htm

Ionic Dashboard. Recuperado de https://dashboard.ionicjs.com/

Del Valle Hernández, L. Proyectos loT con Arduino, las plataformas más importantes. *Programar fácil*. Recuperado de https://programarfacil.com/podcast/proyectos-iot-con-arduino/ [Consulta: 13/01/2018]

Digi international Inc (23/09/2003). P*roduct Manual v1.xEx - 802.15.4 Protocol*. Recuperado de https://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-Datasheet.pdf

Electronic Assembly (04/2011). *Dogm Graphic Series.* Recuperado de https://www.lcd-module.de/eng/pdf/grafik/dogm132-5e.pdf

Fabric. Recuperado de https://fabric.io/

Facebook for Developers. Recuperado de https://developers.facebook.com/

Firebase Google. Recuperado de https://firebase.google.com/docs/

HIVEMQ (Enterprise MQTT Broker). *The Seven Best MQTT Client Tools.* Recuperado de https://www.hivemq.com/blog/seven-best-mqtt-client-tools

Ionic Documentation. Recuperado de https://ionicframework.com/docs/

Ionic Forum. Recuperado de https://forum.ionicframework.com/

i18 next. Recuperado de https://www.i18next.com/

JavaScript Screencasts and Tutorials. Recuperado de https://www.javascripttuts.com/

Linino. Recuperado de http://download.linino.org/linino_distro/linino-2.0/avr/20170407.0/packages/

Mastering Ionic (30/05/2017). *Adding Charts to an Ionic project.* Recuperado de http://masteringionic.com/blog/2017-05-30-adding-charts-to-an-ionic-project/

Morony, J (25/01/2018) Adding Responsive Charts & Graphs to Ionic 2 & 3 Applications. Recuperado de https://www.joshmorony.com/adding-responsive-charts-graphs-to-ionic-2-applications/

Morrissey, D (25/04/2010). Sleeping Arduino-Part 1. *Arduino, Zigbee and Embedded Development*. Blogger. Recuperado de http://donalmorrissey.blogspot.com.es/2010/04/putting-arduino-diecimila-to-sleep-part.html

NGX-Translate. The internationalization (i18n) library for angular. Recuperado de http://www.ngx-translate.com/

NOAA (28/05/2014). National Weather Service. *The Heat Index Equation.* Recuperado de http://www.wpc.ncep.noaa.gov/html/heatindex_equation.shtml

OpenWrt (08/12/2017). *News.* Recuperado de https://forum.openwrt.org/viewforum.php?id=11

Python. Recuperado de https://www.python.org/doc/

Stack Overflow. Recuperado de https://stackoverflow.com/

Sparkfun Electronics (2003). Recuperado de https://www.sparkfun.com/

The MathWorks, Inc (1994-2018). Recuperado de https://es.mathworks.com/

The Official Ionic Blog. *Tips &Tricks for Ionic on Dekstop*. Recuperado de http://blog.ionic.io/

Twitter Application Management. Recuperado de https://apps.twitter.com/

Vergara, J. Integrate Firebase into your Ionic Apps. *Javebratt*. Recuperado de https://javebratt.com/

Vinduino affordable precision irrigation. Recuperado de http://www.vinduino.com/

Wikipedia. Recuperado de https://es.wikipedia.org