UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

Grado en Ingeniería en Sonido e Imagen en Telecomunicación

Trabajo Fin de Grado

Sensorización de Parámetros Físicos con Microcontrolador

Alberto Hoyas Castro

Convocatoria de Junio

2016

UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica
Grado en Ingeniería en Sonido e Imagen en Telecomunicación

Trabajo Fin de Grado

Sensorización de Parámetros Físicos con Microcontrolador

Autor: Alberto Hoyas Castro

Fdo.:

Director: José Vicente Crespo

Fdo.:

Tribunal Calificador

Presidente:
Fdo.:

Secretario:
Fdo.:

Vocal:
Fdo.:

CALIFICACIÓN:
FECHA:

Índice general

Índice de figuras

Agradecimientos

En primer lugar agradecer a mis padres y familiares todo lo que han hecho por mí en todo este tiempo y todos los sacrificios que han surgido y que han tenido que realizar. Sin ellos, sin esos sacrificios y sin nada de lo que me han dado y aportado en toda mi vida, no podría haber llegado al lugar donde estoy ahora, porque ellos han sido los que me han guiado, educado y ayudado en todo, y los que también me han enseñado que nadie te da nada gratis y que todo lo que quieres se puede conseguir.

A mi pareja, que aunque haya sido un año distinto, raro y difícil, siempre ha estado ahí para lo que necesitara. Muchas gracias.

A todos mis amigos, tanto a los de la universidad como a los de siempre, por estar ahí, por ser como son, por ayudarme en todo, por dar esos ánimos por las mañanas de paseos, y esas tardes de música que tanto animaban, esos fines de semana en los que no se podía quedar antes de las 11 por motivos laborales y esos días interminables en la biblioteca. Gracias a todos vosotros, que solo vosotros sabéis quienes sois, y sin vosotros y sin vuestros ánimos, las cosas podrían haber sido diferentes.

Un apartado especial para unas personas con una fuerza impresionante y una vitalidad más fuerte todavía. Me gustaría agradecerles a todos los compañeros que he conocido del programa YUZZ todo, porque la recta final ha sido con ellos, y son ellos mediante charlas, coaching y viajes, los que me daban fuerzas para afrontar este último empujón.

Y, en general, a todas las personas que me han ayudado en este proyecto y en toda mi vida de estudiante, me habéis hecho ser como soy.

Resumen

En el presente trabajo se ha desarrollado una sensorización, que hemos acercado al ámbito de la meteorología, por su cercanía a ese mundo y por los datos que recoge el proyecto. Por ello lo hemos llamado estación meteorológica.

En este proyecto hemos diseñado todo lo referente a la obtención de datos, tratamiento y presentación, de los cuales se implementan con diferentes herramientas. Siendo estas:

- Arduino, para la obtención de datos desde los sensores.
- Raspberry pi, para albergar toda la estructura de almacenaje de los datos, siendo la herramienta sobre la cual la base de datos MySQL funcionará.
- Servidor Web Apache, para la representación de los datos en remoto, a la cual se conectará una aplicación de escritorio, así como, el administrador de la base de datos, PHPMyAdmin.

En Arduino se captan los datos mediante sensores conectados al microcontrolador y regidos por un firmware diseñado específicamente para este proyecto. Para almacenar la información en la base de datos se usará un firmware implementado en la Raspberry y escrito en Python, que almacenará los datos automáticamente.

Para administrar toda la base de datos mediante PHPMyAdmin, en este sistema solo será necesario estar conectado a la misma red WiFi del sistema Apache. La aplicación de escritorio desarrollada está hecha en Java, por tanto, es una multiplataforma y muestra los datos con un botón actuador para actualizar los datos recogidos en la Base de datos MySQL.

Capítulo 1

Introducción:

1.1. Objetivos

Monitorizar objetos, teniendo como meta obtener una serie de datos, es una práctica que el ser humano hace desde hace siglos. Antiguamente se recopilaban dichos datos a mano, es decir, un hombre observaba cómo funcionaba una máquina o un determinado artilugio y realizaba anotaciones al respecto. Este tedioso trabajo, desde hace unos años hasta ahora, lo realizan las máquinas, usando microcontroladores y microprocesadores y , a su vez, una serie de sensores para medir u "observar" cada parámetro de los que se quieren saber,como, por ejemplo, torsiones, presiones, fuerzas, etc.

Uno de los campos donde se usan sensores es la Fórmula 1, en la cual los coches usan millones de sensores para ver cada respuesta de cada pieza en todas las situaciones posibles en las que el monoplaza se pueda ver, con esos datos, cada equipo hace gráficas y los usa para mejorar dicha pieza.

En este proyecto vamos a realizar un prototipo básico de sensorización, que se puede escalar y ampliar para usarse para varios fines y en varias áreas. Los parámetros que vamos medir son la temperatura y la humedad, teniendo dos sensores de temperatura de diferente precisión.

Nuestro objetivo fue realizar el prototipo de nuestra estación meteorológica, que ésta fuera móvil y pequeña para poder ser transportada y que volcara todos los datos recogidos en un servidor web con una base de datos MySQL para tenerlos ordenados y poder consultarlos cuando se quiera.

Se ha intentado mantener los costes bajos, por ello solo consta que hacemos dos mediciones de todas las que suele tener una estación meteorológica.

1.2. Historia de la Meteorolgía

En la antigüedad los aparatos de medida del tiempo fueron de lo más subjetivos, en algunas civilizaciones estaban basados en mitología, vientos, migración de las aves, etc. Por ejemplo: el pueblo de Babilonia basaba sus predicciones en observaciones del movimiento planetario, fenómenos ópticos y aspectos del cielo

Las primeras recogidas de datos instrumentales están recogidas en el siglo XVII, el primer instrumento fue el termómetro, inventado por Galileo Galilei, seguido por el barómetro, inventado por Evangelista Torricelli, discípulo de éste.

Estos nuevos instrumentos causaron un gran revuelo entre la comunidad científica, y pronto se intentaron hacer mediciones simultáneas alrededor del mundo, la primera red de estaciones de observación meteorológica que se intentó establecer fue en 1653, financiados por el duque Fernando II de Toscana, se normalizaron los procesos y los instrumentos para la observación y se mandaron a los distintos observadores, se desarrollaron procesos para medir humedad, el higrómetro de cabello, dirección del viento, el anemómetro, y estado de los cielos, pero en 1667 la academia fundada por el duque cerró, pero hubo más intentos de realizar dicha red en los siglos XVIII y XIX.

La primera gran tormenta registrada y pronosticada fue en 1660 por Otto von Guerike, de Magdeburg (Prusia), quien mediante su barómetro registró una bajada de presión rápida dos horas antes.

El 21 de octubre de 1743, Benjamin Franklin realizó el primer estudio sinóptico meteorológico de América, debido a que una borrasca no dejaba ver un eclipse en Filadelfia pero en cambio, esa misma borrasca en Boston no empezó hasta pasadas las 11 de la noche, por tanto, sí pudo verse el eclipse. Así, con esos datos, dedujo que la tormenta, la lluvia y los vientos se habían desplazado desde Georgia a Nueva Inglaterra. Franklin también descubrió la dependencia del volumen de un gas y la presión, gracias a su famoso experimento de la cometa y el rayo, que dicho experimento lleva a la termodinámica.

George Hadley fue el primero en realizar una correcta explicación general de la circulación atmosférica global, con su estudio sobre los Alisios en 1735.

En 1778, Louis Cotte se comprometió a la creación y mantenimiento de una red extensa de estaciones de observación.

Antoine Lavoisier pensó que sería posible pronosticar el tiempo con uno o dos días de antelación por ello presionó para establecer una red de estaciones por toda Europa y por todo el mundo. Además, defendió la idea de difundir dichos datos una vez al día, cosa que no se pudo hacer hasta que no hubo un desarrollo en las comunicaciones.

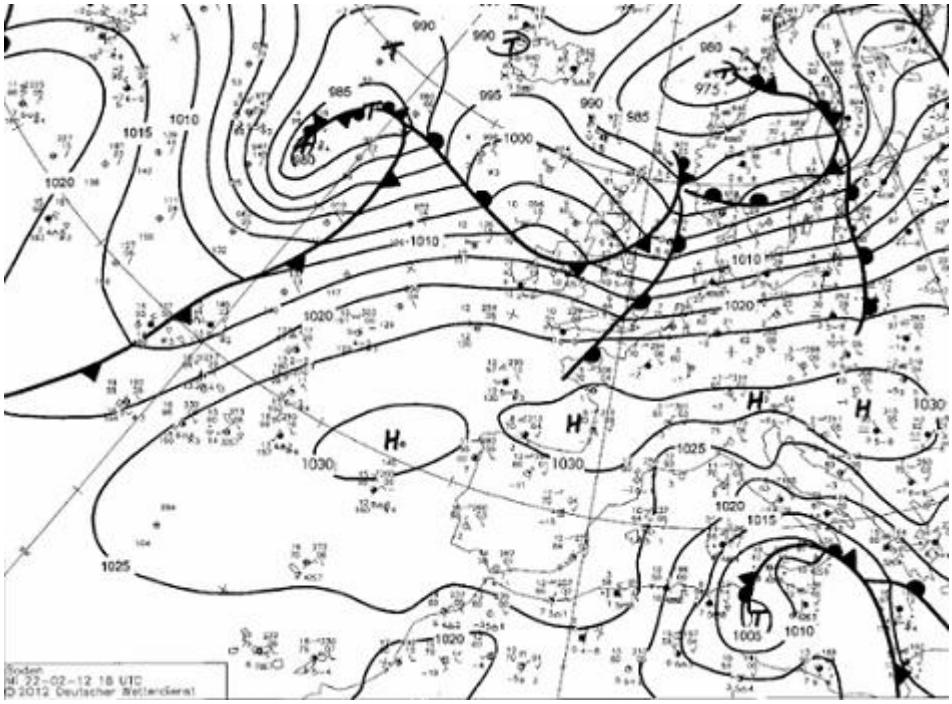

Figura 1.1: Mapa Sinóptico

A principios del siglo XIX se comprendió la necesidad de representar la temperatura y humedad en mapas. Así, mediante simbología se podía representar otros datos de interés, creándose, en consecuencia, la meteorología sinóptica

Otro de los grandes inventos del siglo XIX fue el telégrafo, ya que permitía el intercambio de pronósticos entre agencias y la mejora a la hora de reportar condiciones climatológicas.

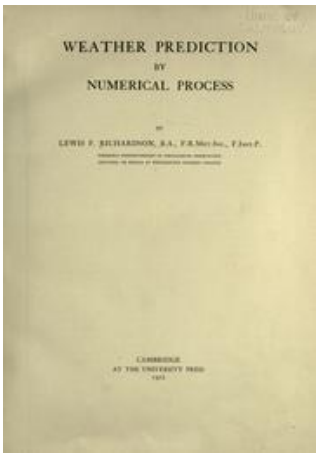

Figura 1.2: Portada Weather Predicction by Numerical Process

A inicios del siglo XX Lewis fry Richardson publica un artículo llamado Weather prediction by numerical process donde explica como pronosticar el tiempo eliminando variables y realizando ecuaciones de la dinámica de fluidos, permitiendo con eso llegar a soluciones numéricas, estas ecuaciones de valores iniciales ayudaron mucho a la comprensión y al pronóstico de la meteorología pero no llegaron a ser fiables y rápidas hasta que von Newman no crea la computadora, y con este invento se podían realizar estos cálculos rápidamente.

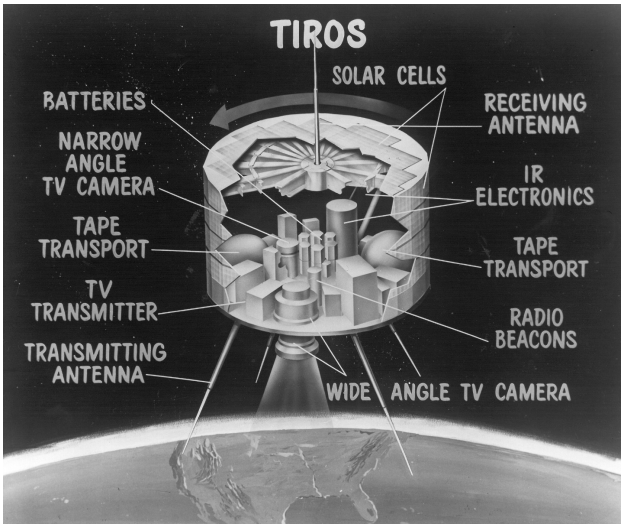

Figura 1.3: Satélite Tiros-1

Hasta aquí se utilizaban los antiguos artilugios de medida y de recolección de datos, pero todo esto cambia con la meteorología aeroespacial, en 1957 se lanza el primer satélite artificial, el TIROS-1, para la observación meteorológica, y con este avance empezaron los artilugios de medida a ser digitales o simplemente automáticos.

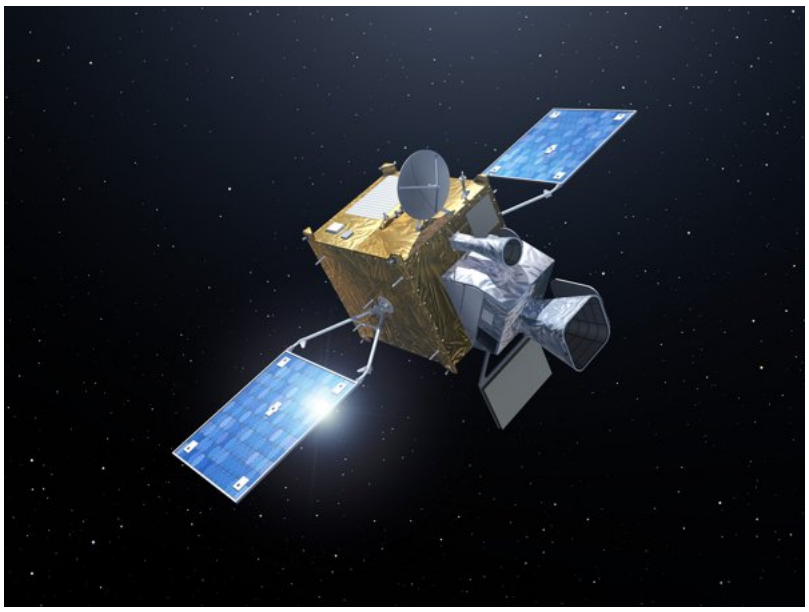

Figura 1.4: Satélite MeteoSat

Actualmente hay un sinfín de satélites meteorológicos en órbita, los cuales se dividen en dos grandes grupos, los TIROS de la agencia norteamericana NOAA (National Oceanic and Atmospheric Administration) y los Geoestacionarios, es decir, los sincronizados con el movimiento de la tierra, el MeteoSat es uno de ellos.

Los pronósticos meteorológicos actuales se obtienen a partir de los satélites y a partir de los datos recogidos en las numerosas estaciones meteorológicas repartidas por todo el mundo.

En España la AEMET tiene 783 estaciones repartidas y recogiendo datos actualmente en una extensa red de estaciones.

Figura 1.5: Estación AEMET Xábia-2

1.2.1. Termómetro de galileo

El termómetro inventado por Galileo se componía de un tubo cilíndrico de vidrio conteniendo un líquido transparente con un coeficiente de dilatación mayor que el del agua y cinco ampollas dentro del líquido. Sus mediciones eran debidas a la flotabilidad del líquido que variaba según la temperatura. Cada ampolla simbolizaba una temperatura, la ampolla que quedaba en la mitad del cilindro era la que decía a qué temperatura había. Tenía que haber un mínimo de cinco ampollas para un rango de no más de 6 grados, entre 18 y 26 grados Celsius aproximadamente y tenía una exactitud de ± 2 grados Celsius.

Figura 1.6: Termómetro de Galileo

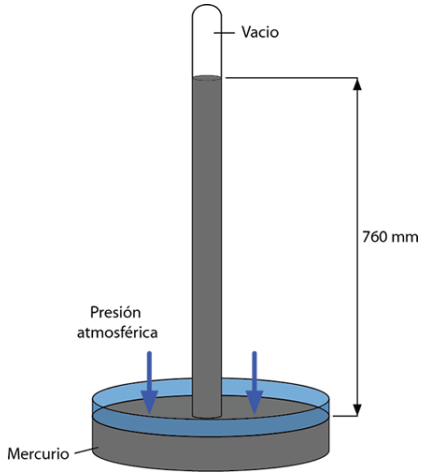

Figura 1.7: Experimento de Torricelli

1.2.2. Experimento de Torricelli

El Italiano Torricelli en 1643 consiguió medir por primera vez la presión atmosférica mediante este experimento de laboratorio, para el cual necesitó mercurio, un tubo de un metro de largo y cerrado por uno de sus extremos y una cubeta.
El experimento consistía en posicionar el tubo lleno de mercurio en la cubeta en la cual también había mercurio, en ese proceso el mercurio solo bajaba unos centímetros y se estabilizaba en los 76 cm de altura. En ese punto consiguió medir que la presión de la atmósfera es de 760 mm HG, esta medida también es igual a la llamada atmósfera, es decir, 760 mm HG = 1 atm.

1.2.3. Higrómetro de Cabello

Fue creado por Horacio de Saussure en 1780, funciona con la cualidad de algunos materiales de absorber el vapor, el cabello humano varía su longitud según el grado de humedad del aire.

1.2.4. Anemómetro

Aparato para medir la velocidad del viento. Fue inventado por Robert Hooke en 1667, consta de unas hélices unidas a un eje central, el cual con su giro se calcula la velocidad del viento

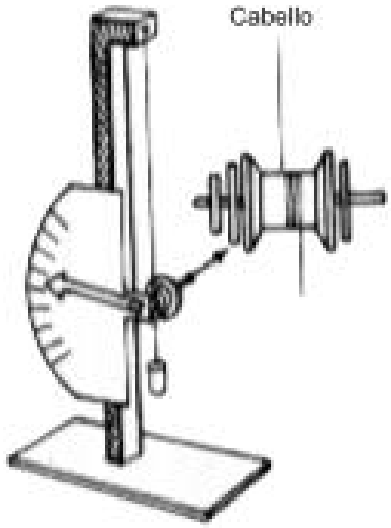

Figura 1.8: Higrómetro de Cabello

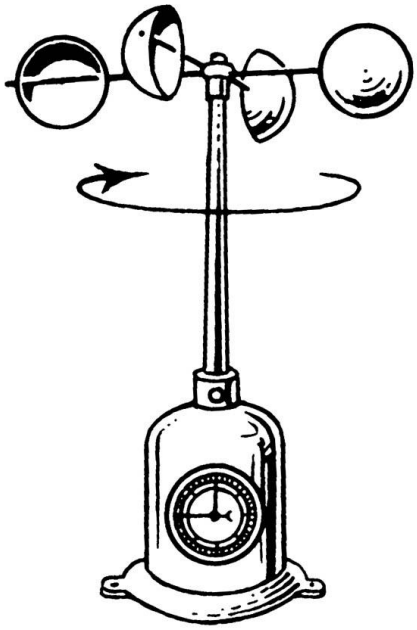

Figura 1.9: Anemómetro

Capítulo 2

Metodología

2.1. Descripción del hardware

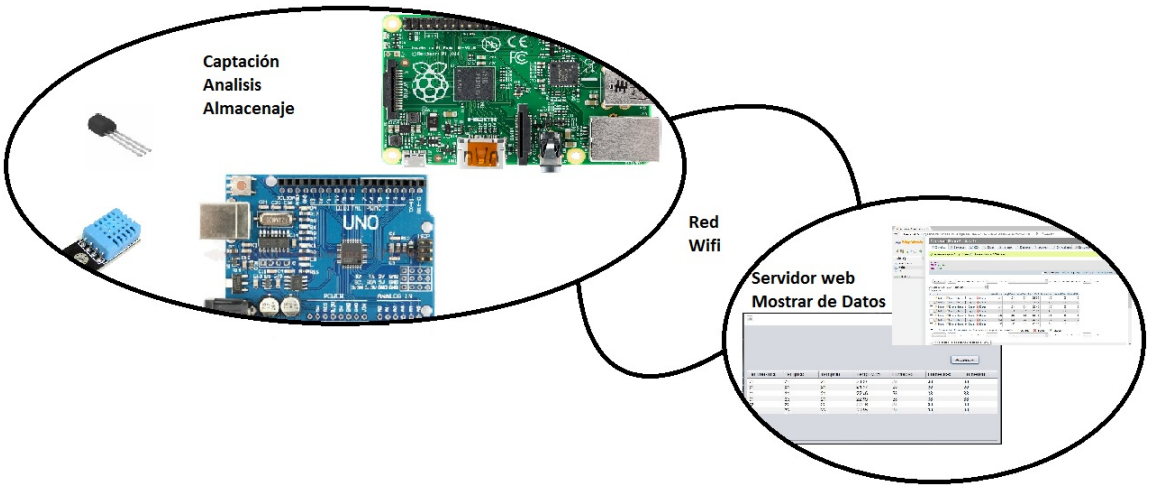

Figura 2.1: Esquema Proyecto

Para desarrollar este proyecto se han necesitado unos determinados componentes de hardware que detallamos a continuación.

2.1.1. Arduino

Figura 2.2: Logo Arduino

Arduino es una empresa italiana que desarrolla placas de microcontroladores con su propio entorno de desarrollo fácil de usar. Uno de sus fundadores Massimo Banzi, daba clases en el instituto italiano de IVERA, donde usaban el controlador Basic Stamp, el cual costaba 100 dólares americanos. Buscando un controlador asequible, nació el proyecto Arduino en el año 2006.

Su filosofía era que tenía que ser open source tanto en hardware como en software, fácil de usar, multiplataforma y su coste no podía superar los $ 50. Con estas reseñas crearon en 2006 la primera placa de desarrollo. Usa un lenguaje de programación de alto nivel llamado Processing, similar a C++. La placa que usaremos para este proyecto es la Arduino UNO Revision3 con el microcontrolador ATmega328. A continuación ponemos las características del microcontrolador y de la placa

Placa	Microcontrolador	Voltaje de entrada	Voltaje de salida	Frecuencia de reloj	Digital I/O
Arduino UNO R3	ATmega328	7-12V	5V	16MHz	14

Entradas Analógicas	PWM	UART	Memoria Flash	SRAM	EEPROM	Intensidad de corriente
6	6	1	32Kb (2Kb para Bootloader)	2Kb	1Kb	40 mA

Figura 2.3: Tabla Características Arduino

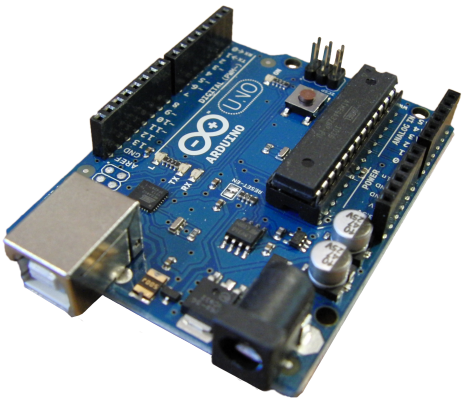

Figura 2.4: Arduino

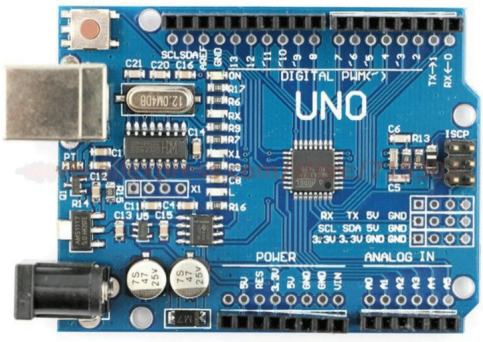

Figura 2.5: Mi Arduino

2.1.2. Raspberry pi

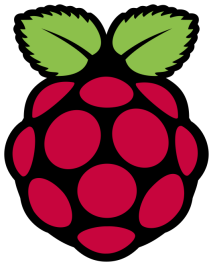

Figura 2.6: Logo Raspberry pi

Raspberry pi es una fundación que nació en 2009 en Caldecote, South Cambridgeshire, Reino Unido. Su objetivo era llevar un ordenador de tamaño reducido a las escuelas para estimular la enseñanza de las ciencias de la computación.

En 2011 crearon su primera placa, a la que llamaron igual que la fundación, de la cual montaron 50 “Alphas” .Poco después pusieron a la venta su primer lote de 100 placas en ebay.

Crearon dos modelos de Raspberry pi, el modelo A y el modelo B a lo que se sumó al tiempo el modelo B+, siendo este una mejora del antiguo modelo B. En 2015 sacaron al mercado un modelo más potente, la Raspberry pi 2 modelo B. En 2016 se comercia-

lizó la tercera versión de Raspberry pi, es solo un cambio respecto a la segunda versión, el procesador, que es más potente y soporta nuevos servicios como el Bluetooth 4.1 con la tecnología Bluetooth Low Energy (BLE) que ayuda al consumo de energía total del dispositivo. Cualquier modelo es compatible con los modelos anteriores a la hora de trabajar con más de uno a la vez.

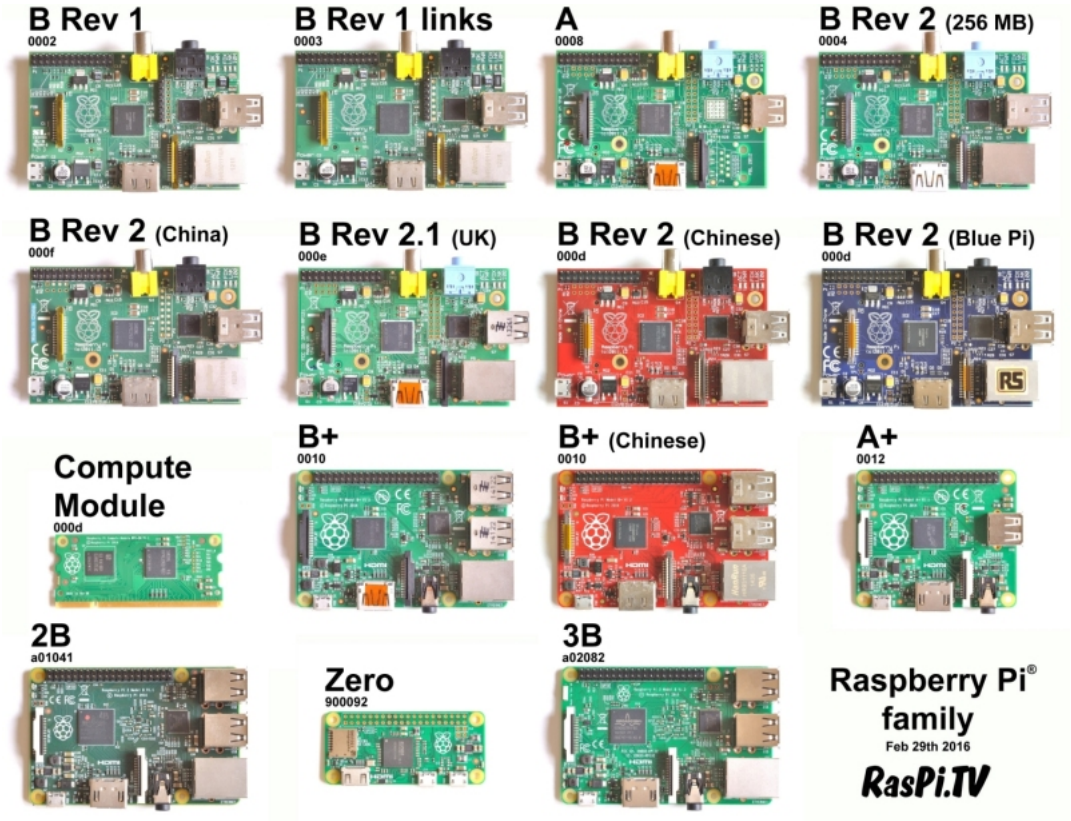

Figura 2.7: Familia Raspberry pi

Estos miniordenadores corren con varios sistemas operativos, uno de ellos y el más utilizado es Raspbian, distribución de Debian agilizada y desarrollada para funcionar en la Raspberry pi. Con la nueva Raspberry pi 2 en el mercado se ha incrementado el número de sistemas operativos, así como el número de distribuciones creadas para dichas placas, un nuevo SO es Windows 10, que con su versión IoT.

En este proyecto usaremos la Raspberry pi Model B+, con el microprocesador Broadcom BCM2835. A continuación pondremos un cuadro con las características de esta placa. Raspberry Pi Model B+

SoC	CPU	Juego de Instrucciones	GPU	Memoria (SDRAM)
Broadcom BMC2835	ARM 1176JZF-S a 700MHz	RISC de 32 bits	Broadcom VideoCore IV	512 MiB

Puertos USB 2.0	Entradas de video	Salidas de video	Salidas de audio	Almacenamiento integrado
4	Conectos MIPI CSI	Conector RCA (PAL y NTSC), HDMI, Interfaz DSI	Conector de 3,5 mm, HDMI	MicroSD

Conectividad	Perifericos a bajo nivel	Consumo Energetico	Fuente de alimentacion
10/100 Ethernet, via HubUSB	8xGPIO, SPI, UART	600 mA, (3.0W)	5V via MicroUSB o GPIO heather

Figura 2.8: Caracteristicas Raspberry pi B+

Figura 2.9: Mi Raspberry pi

2.1.3. Sensores

Para realizar el proyecto, en la parte de captar datos usamos dos sensores, el DHT11 y el LM35, que se expondrá sus características a continuación.

2.1.3.1. DHT11

Este sensor es un híbrido de temperatura y humedad, con señal de salida digital. Es muy estable a largo plazo, incluye un medidor de humedad de tipo resistivo y una resistencia del tipo NTC para la temperatura, todo ello está conectado a un microcontrolador de 8 bit ofreciendo una rápida respuesta y una buena relación calidad/precio con una precisión de 2ºC en el caso de la temperatura. Tiene un bajo consumo y un pequeño tamaño con interfaz de 4 pines.

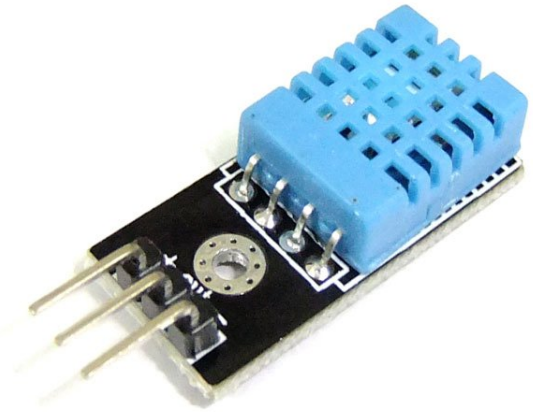

Figura 2.10: DHT11

2.1.3.2. LM35

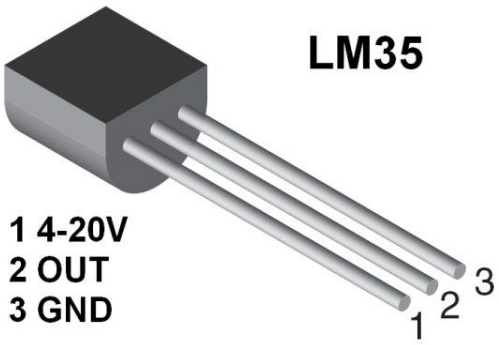

Figura 2.11: LM35

El LM35 es un sensor de temperatura de precisión, tiene un rango de -55ºC a 150ºC y una sensibilidad de 0.5ºC, este sensor está operativo entre 4 y 30 voltios, los cuales están calibrados con el rango de temperatura, no necesita de ninguna constante ni de ningún calibrador externo.

Specifications

Item	Measurement Range	Humidity Accuracy	Temperature Accuracy	Resolution	Package
DHT11	20-90%RH 0-50 ℃	±5%RH	±2℃	1	4 Pin Single Row

Parameters	Conditions	Minimum	Typical	Maximum
Humidity				
Resolution		1%RH	1%RH	1%RH
			8 Bit	
Repeatability			±1%RH	
Accuracy	25℃		±4%RH	
	0-50℃			±5%RH
Interchangeability	Fully Interchangeable			
Measurement Range	0℃	30%RH		90%RH
	25℃	20%RH		90%RH
	50℃	20%RH		80%RH
Response Time (Seconds)	1/e(63%)25℃, 1m/s Air	6 S	10 S	15 S
Hysteresis			±1%RH	
Long-Term Stability	Typical		±1%RH/year	
Temperature				
Resolution		1℃	1℃	1℃
		8 Bit	8 Bit	8 Bit
Repeatability			±1℃	
Accuracy		±1℃		±2℃
Measurement Range		0℃		50℃
Response Time (Seconds)	1/e(63%)	6 S		30 S

Item	Condition	Min	Typical	Max	Unit
Power supply	DC	3	5	5.5	V
Current supply	Measuring	0.5		2.5	mA
	Stand-by	100	Null	150	uA
	Average	0.2	Null	1	mA

Figura 2.12: Hoja de Características DHT11

LM35

Electrical Characteristics

(Notes 1, 6)

Parameter	Conditions	LM35			LM35C, LM35D			Units (Max.)
		Typical	Tested Limit (Note 4)	Design Limit (Note 5)	Typical	Tested Limit (Note 4)	Design Limit (Note 5)	
Accuracy, LM35, LM35C (Note 7)	T_A=+25°C	±0.4	±1.0		±0.4	±1.0		°C
	T_A=−10°C	±0.5			±0.5		±1.5	°C
	T_A=T_{MAX}	±0.8	±1.5		±0.8		±1.5	°C
	T_A=T_{MIN}	±0.8		±1.5	±0.8		±2.0	°C
Accuracy, LM35D (Note 7)	T_A=+25°C				±0.6	±1.5		°C
	T_A=T_{MAX}				±0.9		±2.0	°C
	T_A=T_{MIN}				±0.9		±2.0	°C
Nonlinearity (Note 8)	T_{MIN}≤T_A≤T_{MAX}	**±0.3**		**±0.5**	**±0.2**		**±0.5**	°C
Sensor Gain (Average Slope)	T_{MIN}≤T_A≤T_{MAX}	**+10.0**	**+9.8, +10.2**		**+10.0**		**+9.8, +10.2**	mV/°C
Load Regulation (Note 3) 0≤I_L≤1 mA	T_A=+25°C	±0.4	±2.0		±0.4	±2.0		mV/mA
	T_{MIN}≤T_A≤T_{MAX}	**±0.5**		**±5.0**	**±0.5**		**±5.0**	mV/mA
Line Regulation (Note 3)	T_A=+25°C	±0.01	±0.1		±0.01	±0.1		mV/V
	4V≤V_S≤30V	**±0.02**		**±0.2**	**±0.02**		**±0.2**	mV/V
Quiescent Current (Note 9)	V_S=+5V, +25°C	56	80		56	80		µA
	V_S=+5V	**105**		**158**	**91**		**138**	µA
	V_S=+30V, +25°C	56.2	82		56.2	82		µA
	V_S=+30V	**105.5**		**161**	**91.5**		**141**	µA
Change of Quiescent Current (Note 3)	4V≤V_S≤30V, +25°C	0.2	2.0		0.2	2.0		µA
	4V≤V_S≤30V	**0.5**		**3.0**	**0.5**		**3.0**	µA
Temperature Coefficient of Quiescent Current		**+0.39**		**+0.7**	**+0.39**		**+0.7**	µA/°C
Minimum Temperature for Rated Accuracy	In circuit of *Figure 1*, I_L=0	+1.5		+2.0	+1.5		+2.0	°C
Long Term Stability	T_J=T_{MAX}, for 1000 hours	±0.08			±0.08			°C

Figura 2.13: Hoja de Características LM35D

2.2. Descripción del software

2.2.1. Java

Figura 2.14: Logo Java

Figura 2.15: Sun Microsystem Logo

Java es un lenguaje de programación de alto nivel desarrollado en 1991 por Sun Microsystem. Su primer nombre fue OAK, luego Green y finalmente se acabó denominando Java. En 2009 Oracle compró la tecnología Java a Sun por casi 5710 millones de Dólares haciéndose así con el control de la tecnología y de la compañía Sun Microsystems.

Su estructura es muy similar a C y C++ pero eliminando las herramientas a bajo nivel. Es uno de los lenguajes más utilizados en la actualidad ya que sus predecesores C y C++, se dice que son difíciles en cuanto a estructura y sintaxis.

Las bases del lenguaje son:

- Programación orientada a objetos
- Multiplataforma
- Soporte para Internet
- Ejecutar en sistemas remotos
- fácil de usar

Figura 2.16: Sun Microsystem y Oracle

En Este proyecto usaremos el compilador NetBeans para crear los códigos Java. Por su simplicidad al programar y sus herramientas para diseñar interfaces de usuario.

Programación orientada a objetos: Es un método de programación en la que el código y los datos se agrupan en paquetes llamados objetos. Y el objetivo es separar las cosas permanentes de las alterables. Y también poder reutilizar algunos objetos en otros proyectos.

Multiplataforma: Esto significa que el código escrito en Java puede compilarse y utilizarse en cualquier tipo de sistema independientemente del hardware y del software. Para ello se compila el código fuente en Bytecode, instrucciones máquina específicas de la plataforma java, y estos se ejecutan en una máquina virtual java que se instala en el SO.

Código abierto: Oracle ha liberado gran parte del código bajo el tipo de licencia GNU GPL. Por tanto no incorpora ninguna inversión económica ya que el kit de desarrollo también es libre y gratuito

Figura 2.17: Logo NetBeans

En Este proyecto usaremos el compilador NetBeans para crear los códigos Java. Por su simplicidad al programar y sus herramientas para diseñar interfaces de usuario.

Figura 2.18: Oracle Logo

2.2.1.1. Arduino (Processing)

Figura 2.19: Logo Processing

Figura 2.20: Arduino IDE

Procesing es un lenguaje de programación específico para métodos de enseñanza multimedia y para programar en Arduino, está basado en Java en su totalidad y usa todas sus librerías. Fue creado por Ben Fry y Casey Reas en el MIT Media Lab y se distribuye bajo la licencia GNU GPL.

2.2.2. Python

Figura 2.21: Logo Python

Phython es un lenguaje de programación creado a finales de la década de los Ochenta por Guido Van Rossum en el CWI de los Países Bajos, es un lenguaje multiparadigma e interpretado con la filosofía de que su código sea legible.

La fundación encargada de su desarrollo es la Phyton Software Foundation y se distribuye con código de licencia GNU.

Una frase célebre que se suele decir es “la vida es corta, por eso programo en Python”, por tanto se define como un lenguaje sencillo, fácil de aprender y ágil.

En lo referente a multiparadigma, quiere decir que soporta programación orientada a objetos, programación imperativa y programación funcional en menor medida, es multiplataforma también, y es un lenguaje interpretado que usa un tipado dinámico.

Es uno de los 4 lenguajes oficiales de Google, siendo este el motor de toda la infraestructura de YouTube, como se suele decir también, "Java es el lenguaje de las aburridas consultorías de sistemas y de las mastodónticas multinacionales, Python es el lenguaje de las startups de Silicon Valley".

En 2012, Python cuenta con unos recursos para ciencia igual o mejor que los de Matlab, siendo estos recursos libres y gratuitos.

2.2.3. MySQL

Figura 2.22: Logo MySQL

Es un Sistema de gestión de bases de datos propiedad de Oracle Corporation, fue creado por David Axmark, Allan Larsson y Michael Widenius en la empresa MySQL AB en 1995, cuando fue fundada. MySQL AB fue comprada por Sun Microsystems en 2008 y finalmente por Oracle en 2010.

Tiene una doble licencia GPL/Licencia Comercial ya que al ser un proyecto avalado por una empresa comercial, que posee la mayor parte del copyright del código. Se distribuye en formato de versionado.

Se permite acceder a ella desde varios lenguajes de programación como Java, PHP, C, C++, Python, Real BASIC, etc. Es muy utilizado en aplicaciones web para mantener los datos organizados.

2.2.4. PHPMyAdmin

Figura 2.23: Logo PHPMyAdmin

Es una herramienta para manejar bases de datos tipo MySQL escrito en PHP y ejecutado por Internet. Se creó en 1998 por Tobias Ratshiller, el cual quería desarrollar una red administrativa basada en PHP para MySQL.

En la actualidad cuenta con una interfaz web, un organizador de Bases de Datos, administrador de varios servidores, creador de consultas Query-by-Example, búsqueda global en base de datos.

A continuación se expone un pantallazo de la interfaz web actual

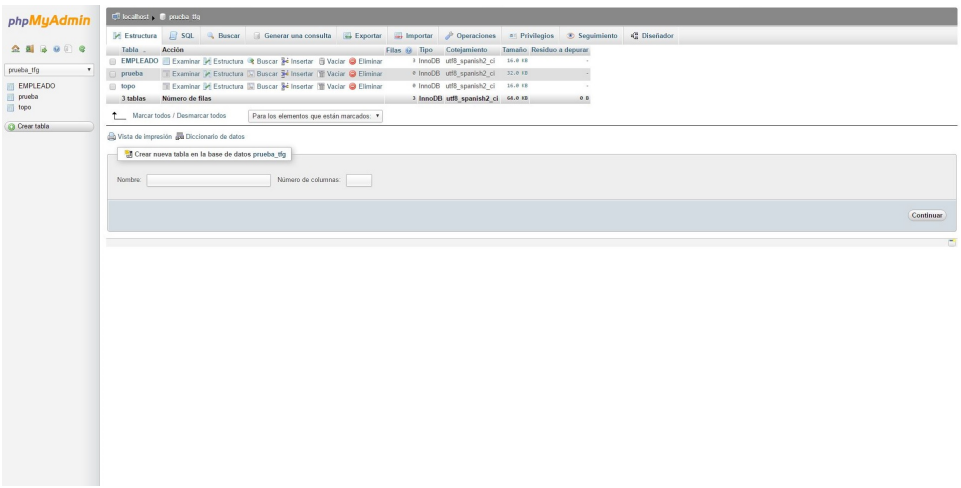

Figura 2.24: Captura PHPMyAdmin

2.2.4.1. PHP

Figura 2.25: Logo PHP

Es un lenguaje de programación de código abierto especializado en desarrollo web ya que se puede introducir en un archivo HTML. Sus siglas significan Pre Hypertext processor o Hypertext preprocessor. Fue creado por Rasmus Lerdorf en 1995 y forma parte del software libre de la licencia PHP incompatible con GNU.

Es un lenguaje del lado del servidor, es decir, que se ejecuta en el servidor y vuelca los datos en una página web que se le manda por HTML al usuario. Es un lenguaje muy simple para el programador principiante pero con unas características muy avanzadas para profesionales.

2.2.5. Servidor Apache

Figura 2.26: Logo Apache

Es un servidor web basado en HTML y el cual es multiplataforma, es código abierto con licencia Apache. Fue desarrollado en 1995 por Robert McCool y actualmente está supervisado por la Apache Software Foundation en el proyecto HTTP Server. En 1996 fue el servidor más usado de la World Wide Web y continuó aumentando hasta 2005.

Se suele distribuir junto con MySQL y se puede programar mediante PHP, Perl, Python y Ruby.

Capítulo 3

Implementación del proyecto

3.1. Inicios

La construcción de esta estación empiezan al principio con componentes prestados por compañeros, los cuales eran los mismos que se han detallado antes salvo por un pequeño cambio, la Raspberry pi que era un modelo B y no el modelo B+ que se usó al final.

En meses posteriores se adquirieron todos los componentes así como no solo los componentes, ya que todo se compró en packs de varios sensores y/o cables junto con las placas.

Figura 3.1: Kit Raspberry pi

Figura 3.2: Kit Arduino

En principio el proyecto iba encaminado a realizarlo todo en la Raspberry, es decir, sin conectarse a wifi ni a un posible Internet, pero posteriormente se vio la idea de conectarlo a una red wifi y que todos los dispositivos conectados a ella pudieran ver los datos, una de las líneas futuras que observamos y que detallamos más adelante es conectarlo a Internet definitivamente.

3.2. Parte 1: Raspberry Pi

Teniendo ya en posesión nuestros propios materiales lo primero que hicimos fue instalar la distribución Raspbian en la Raspberry. Dicho sistema operativo es una distribución muy liviana basada en Debian la cual está desarrollada específicamente para dispositivos Raspberry pi. A partir de ahí antes de nada instalamos todo lo referente al servidor web, instalando Apache en la Raspberry pi, junto con Apache también instalamos MySQL y PHPMyAdmin, los cuales tuvimos que configurar para que fuera todo orientado hacia las acciones posteriores con los datos.

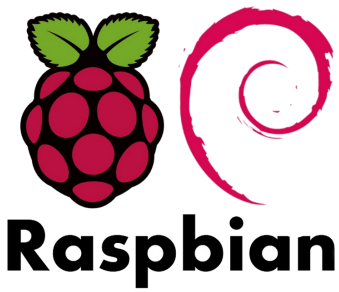

Figura 3.3: Logo Raspbian

Seguido de esto tuvimos que configurar la Raspberry para que se conectara mediante wifi al router automáticamente por el adaptador USB Vilros 802.11n con chip Atheros, para ello entramos en un terminal y con usuario ROOT abrimos mediante el editor de textos "$NANO$" el archivo "$/etc/wpa_supplicant/wpa_supplicant.conf$" y al cual le añadimos las líneas de la red wifi correspondiente como se muestra en la captura.

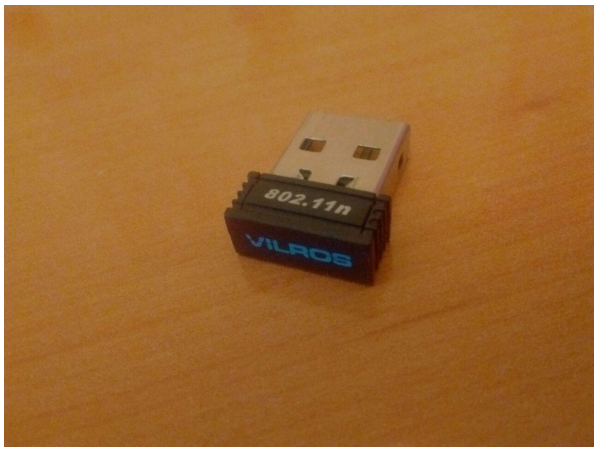

Figura 3.4: Adaptador USB 802.11n

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1

network={
        ssid="Orange-F332"
        psk="FA44A32E"
        proto=RSN
        key_mgmt=WPA-PSK
        pairwise=TKIP
        group=TKIP WEP104 WEP40
        auth_alg=OPEN
}
```

Figura 3.5: Pantallazo wpa supplicant

Con esta configuración ya guardada entramos en el archivo "$/etc/network/interfaces$" y escribimos lo siguiente para que tuviera ip estática.

```
auto lo
iface lo inet loopback
iface eth0 inet dhcp

allow-hotplug wlan0
auto wlan0
iface wlan0 inet manual
 wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
iface default inet manual
        address 192.168.1.110
        netmask 255.255.255.0
        gateway 192.168.1.1
```

Figura 3.6: Pantallazo Interfaces

Seguido de esto, creamos la base de datos con los campos que quería en MySQL desde PHPMyAdmin, los campos que pusimos fueron:

- Temperatura
- Temperatura máxima(TempMax)
- Temperatura mínima(TempMin)
- Temperatura del LM35(TempLM35)
- Humedad
- Humedad máxima(HumedMax)
- Humedad mínima(HumedMin)

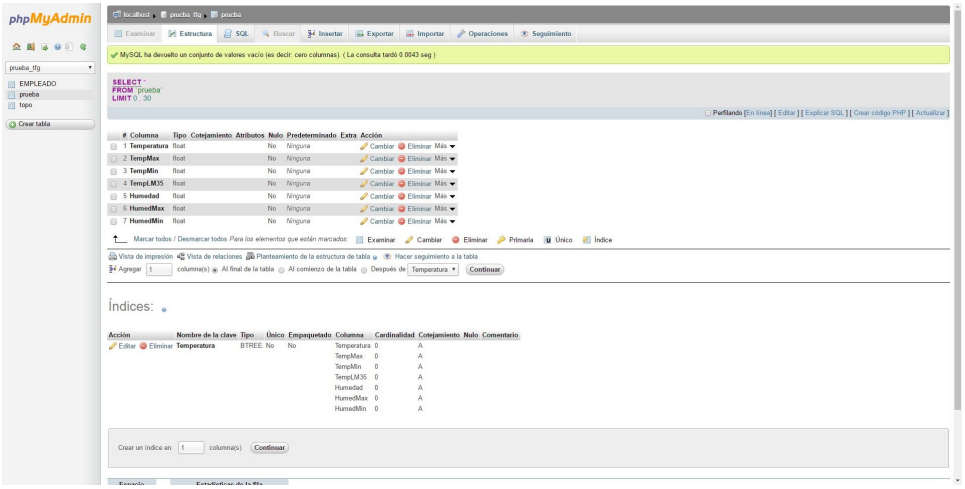

Figura 3.7: Pantallazo Base de Datos

Todos ellos con tipología Float para poder poner los decimales. Cuando todo lo referente al servidor estuvo configurado pasamos a diseñar el firmware para Arduino y la cogida de datos.

3.3. Parte 2: Arduino

Pasando al lado del microcontrolador Arduino, empezamos a crear el firmware mediante librerías, ya que el sensor DHT tiene librerías incorporadas que hacen que no sea necesario ninguna cuenta para saber la temperatura. Creamos el código basándonos en la librería

```
// Libreria para Sensores DHT
#include "DHT.h"
```

Figura 3.8: Librería DHT Arduino

De la cual sacamos la parte para detectar un error en la lectura, la parte de inicialización y las sentencias para leer los datos de temperatura y humedad.

```
// Obtiene la Humedad
float h = dht.readHumidity();
// Obtiene la Temperatura en Celsius
float t = dht.readTemperature();
```

Figura 3.9: Sentencias Arduino

Para el LM35 leímos los datos mediante entrada analógica ya que el sensor manda una señal analógica en la cual solo especifica la amplitud en milivoltios, para ello encontramos una fórmula que nos da la temperatura en grados Celsius, ya que la salida es proporcional a la temperatura en grados Celsius, dicha fórmula es:

```
// El sensor muestrea la temperatura del LM35 y realiza una cuenta para pasarla a celsius
temperatura1 = analogRead(analog_pin);
temperatura =  (5.0 * temperatura1 * 100.0)/1024.0;
```

Figura 3.10: Formula LM35

La relación básica es de 10mV/°C. Los demás factores son para ajustar la resolución de la salida a 10 bits, el 5 es el Vref y el 1024 la resolución ADC. La función analogRead es una función implementada en el IDE de Arduino que sirve para leer los datos de entrada analógica de sus pines desde el A0 al A5.

3.4. Parte 3: Almacenaje de datos

Cuando terminamos el firmware de Arduino para captar los datos y calcular los máximos y los mínimos buscamos la manera de conectar y guardar estos datos en la Raspberry pi, para ello en principio usamos los Puertos GPIO de esta placa para conectar directamente el Arduino a una línea de entrada mediante una herramienta que venía y especificaba cada pin de la Raspberry.

Figura 3.11: Puertos GPIO Raspberry pi

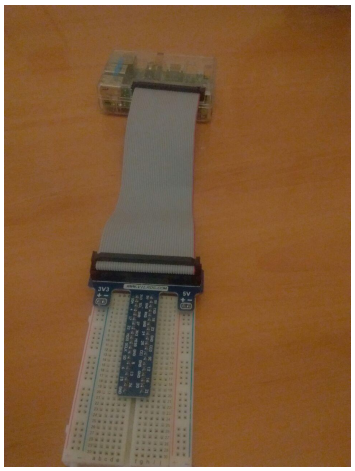

Figura 3.12: Cable y adaptador GPIO Raspberry pi

Pero esta vía no era muy fiable ya que tuvimos que diseñar un puente de resistencias para adaptar los voltajes de uno y de otro, ya que Arduino funciona a 5V y los GPIO de la Raspberry a 3,3V. El puente de resistencias que diseñamos tenía el siguiente esquema.

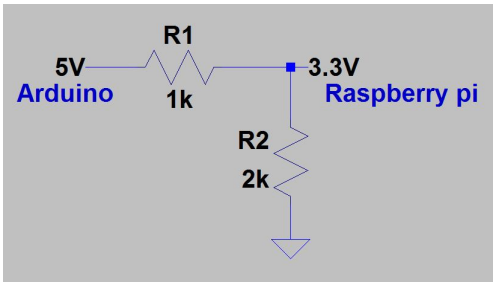

Figura 3.13: Divisor de tensión

La razón por la que desechamos este sistema fue en la calidad de los cables, ya que dependía mucho para la buena recolección de datos ya que los cables de los que disponíamos eran de mala calidad y había veces que no llegaba ningún dato a la Raspberry.

Las sentencias que usamos en primer lugar para leer los datos de Arduino en raspberry fueron:

```
pi@raspberrypi ~ $ stty -F /dev/ttyUSB0 9600
pi@raspberrypi ~ $ cat /dev/ttyUSB0
```

Figura 3.14: Sentencias para Obtener Datos Puerto Serie

Estas sentencias se escribían en un terminal y salía lo siguiente

```
pi@raspberrypi /dev $ stty -F ttyUSB0 9600
pi@raspberrypi /dev $ cat ttyUSB0
38.00|21.00|21|21|20.51|38|37||fpi@raspberrypi /dev $
```

Figura 3.15: Datos Puerto Serie en terminal

A partir de ahí, pasamos a diseñar un código para la recolección de datos y el almacenaje de estos en la base de datos. Decidimos hacer este código en el lenguaje de programación Python, por ser más sencillo que en otro lenguaje porque esta implementado en Raspbian de fábrica.

Para diseñar el código tuvimos que leernos e importar las librerías: PySerial, para leer el puerto serie por donde Arduino mandaba los datos; y MySQLdb, para poder insertar datos en la Base de datos.

Para mandar y separar varios datos usamos las sentencias Split y Join de Python, estas sentencias lo que hacen es separar datos de una cadena de caracteres poniendo símbolos específicos entre los datos.

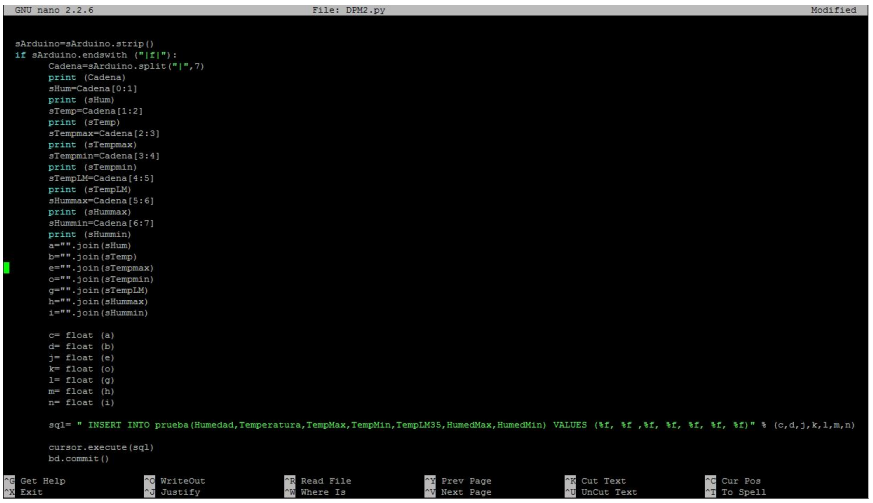

Figura 3.16: Código Split/Join Python

Finalmente todo esto manda los datos directos a la tabla de mi base de datos para guardar los datos y poder presentarlos.

Para poder manejar y organizar la base de datos en basta con poner la dirección: "$http://192{,}168{,}1{,}110/phpmyadmin$" en cualquier navegador de la Raspberry, y para poder acceder en remoto, el dispositivo en cuestión tiene que estar conectado a la misma red que la Raspberry y a partir de ahí, todo igual, abriendo cualquier navegador y poniendo la dirección anterior saldrá el panel de control de PHPMyAdmin donde podremos tener el control de la base de datos.

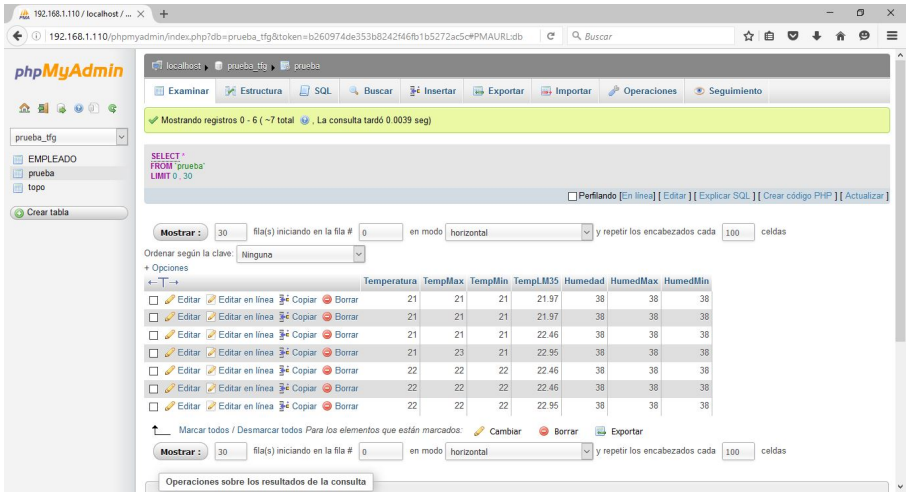

Figura 3.17: Base de Datos

3.5. Parte 4: Interfaz

Se ha creado una pequeña aplicación de escritorio en Java para poder visualizar los datos, el compilador usado es NetBeans, para que Java pueda entrar y mostrar los datos de MySQL, éste necesita un "Driver.º un controlador llamado "$com.mysql.jbdc_5$,1,5". Con este controlador Java puede entrar en la base de datos y acceder a cualquier dato que necesite para mostrar o modificar, para ello tenemos dos clases:

- Conexion.java
- Estacion2.java

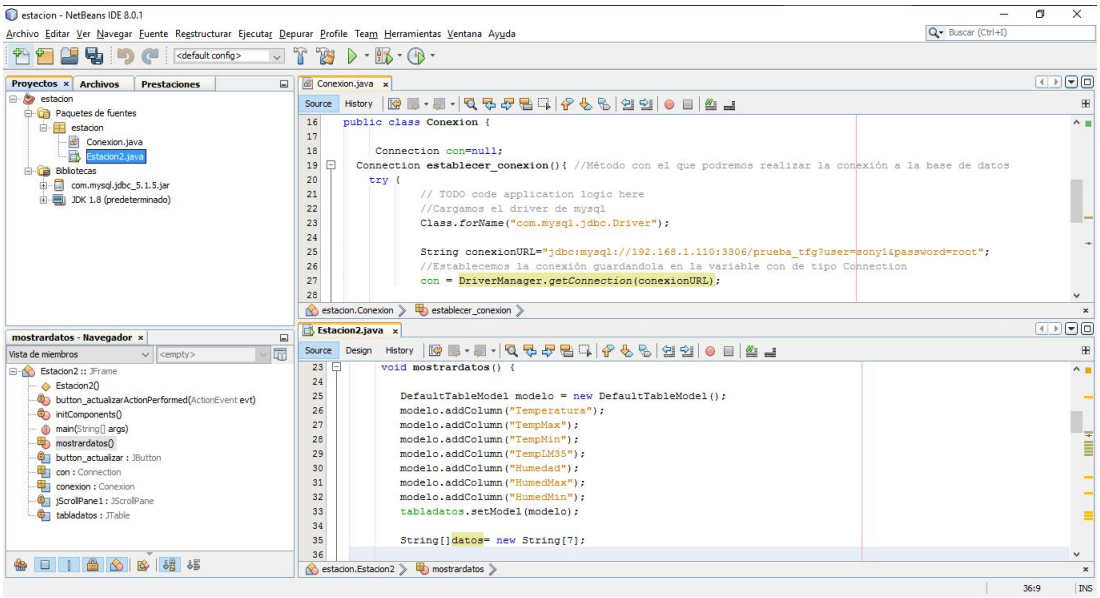

Figura 3.18: Parte del Código java Interfaz

Estas dos clases lo que hacen es: Conexion crea la ligación con la base de datos y guardarla en una variable. En la clase Estacion2 lo que se hace es, usando la variable anterior, crear la tabla y realizar la consulta "$select * fromprueba$" para mostrar los datos. A continuación se crea una interfaz de usuario con un simple botón de actualizar los datos

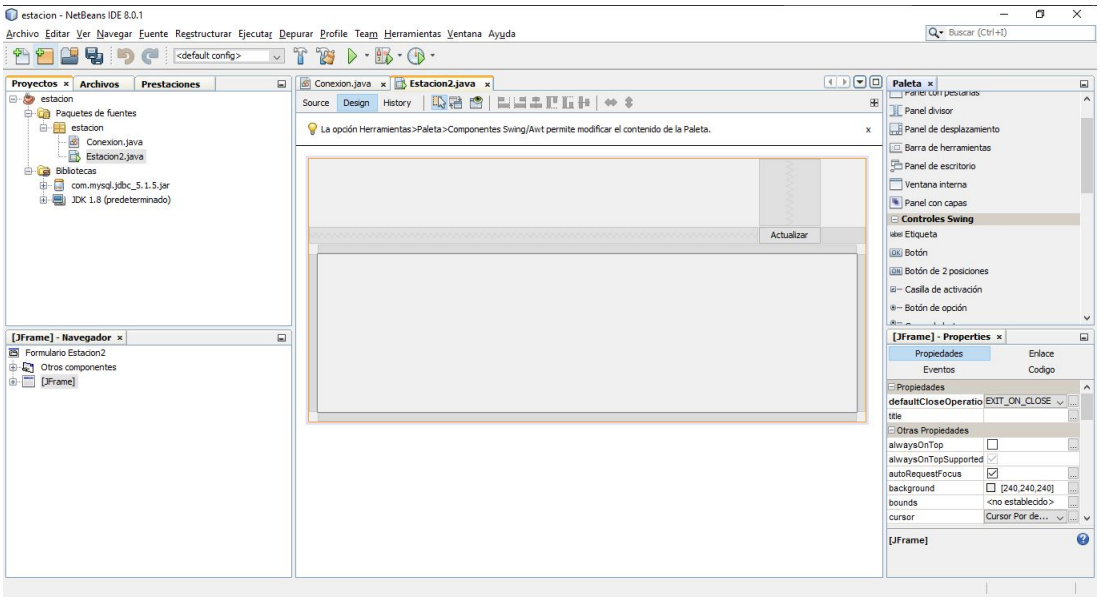

Figura 3.19: Herramienta de Creación de Interfaz NetBeans

Por último, el problema que tuvimos fue que para poder conectarnos a la base de datos tuvimos que modificar el archivo de configuración de MySQL "$/etc/mysql/my.cnf$", en el buscamos bind-address y cambiamos "$127{,}0{,}0{,}1$" por "$0{,}0{,}0{,}0$" para poder acceder a la base de datos desde cualquier ip, es decir, en remoto.

Figura 3.20: Bind address para acceso remoto

Capítulo 4

Resultados y Conclusiones

4.1. Resultados

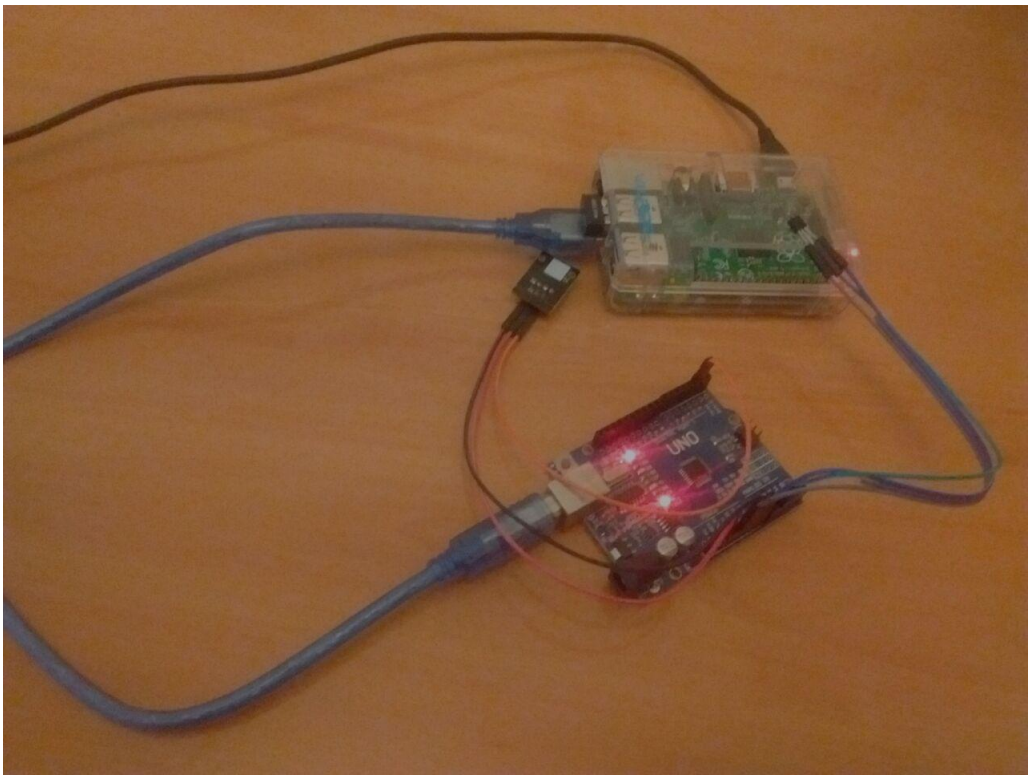

Figura 4.1: Maqueta terminada

Figura 4.2: Interfaz Java sin Datos

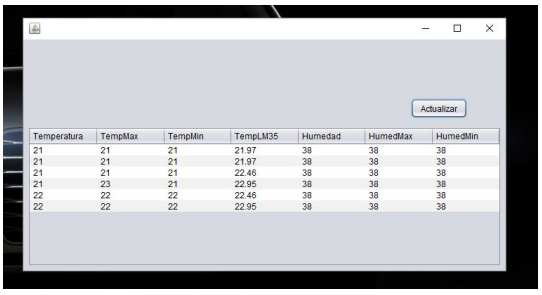

Figura 4.3: Interfaz Java con Datos

Como podemos ver el resultado final es bastante bueno, hay un pequeño error de medición entre los dos sensores de temperatura por tener diferente precisión ya que esta el DHT11 tiene una precisión de ± 2 grados y el LM35 de ± 0.5 grados.

4.2. Conclusiones

Hemos sentido una gran satisfacción al haber podido construir este proyecto con el conocimiento que ya disponíamos en todos los ámbitos que tocan sus fases, por tener que aprender nuevas líneas, lenguajes de programación y formas de hacer las cosas, como puede ser aprender el lenguaje Python y la forma en la que captamos los datos en la Raspberry pi.

Como proyecto ha superado los retos que nos pusimos en un principio, llegando a ser incluso mejor en algunos aspectos, como en el almacenaje de los datos con sus máximos y sus mínimos, que en un principio no contemplábamos.

Cada parte de este proyecto puede ser de gran utilidad para infinidad de cosas, aunque no tengan nada que ver con la sensorización y las estaciones meteorológicas, como podemos ver en el apartado del almacenaje en la base de datos, parte que se puede usar para almacenar cualquier tipo de datos que queramos.

Con este proyecto, en la parte personal, me siento muy contento de poderlo terminar, ya que para mi persona ha ayudado en ciertas maneras, como, por ejemplo, en la de superarse, resolver problemas y proponerse ciertos retos para después cumplirlos.

Capítulo 5

Líneas futuras

En este proyecto, puede haber infinidad de líneas futuras para llevar a cabo, la primera y más lógica es aumentar el número de sensores para poder medir más parámetros, en este proyecto solo hemos usado dos sensores por falta de presupuesto pero se le podrían añadir muchos más, tantos como parámetros y datos queramos medir.

Otra de las líneas es en lo referente a la red para visualizar los datos, en este proyecto nos quedamos en presentar los datos en una red de área local, pero una posible ampliación seria poder presentarlos en todo internet, es decir, que desde cualquier dispositivo conectado a internet en cualquier parte del mundo, puedas conectarte a la base de datos para poder verla, modificarla o borrarla. Con esto podríamos estar a cientos de kilómetros de la captación de datos y poder ver y copiar los datos fácilmente conectándose solo al servidor.

También sería interesante crear aplicaciones para poder visualizar los datos en cualquier dispositivo, ya sea móvil o fijo, en Android, IOS, Windows, Linux, etc. Con esta aplicación lo que haríamos seria mejorar la puesta de datos al usuario, adaptándola a la pantalla del dispositivo en cuestión. Con esto solucionaríamos el problema que muchas veces se nos puede plantear, y es el siguiente, saber ciertos datos sin levantarse de la silla, ahorramos tiempo, ya que con un simple dispositivo tendríamos toda esa información delante de nosotros.

Basado en la anterior ampliación, podría haber un botón en dicha aplicación, el cual sirviese para descargar los datos en múltiples formatos para ser procesados y analizados en hojas de cálculo.

Añadir un sistema de alimentación autónomo, como una batería, para poder tener mayor movilidad en lo referente a su uso. Con este sistema podríamos poner todos los aparatos en cualquier lugar que queramos medir parámetros. Y para tener mayor movilidad todavía, se podrían usar unos módulos inalámbricos para conectar Arduino con Raspberry pi, con esto lo que conseguiríamos seria poder alejar más el proyecto de cualquier enchufe, router, etc. Por ejemplo, cuando queremos saber la temperatura que hace en un campo, poder poner el captador de datos lo más alejado de la casa que se pueda.

Realizar un sistema de alarmas y notificaciones para tener controlado en todo momento los datos captados en tiempo real, es decir, cuando la temperatura exceda de tal valor, que nos salte una notificación o alarma en la aplicación en java, y que esa notificación o alarma pueda llevar a cualquier tipo de acción, por ejemplo en un congelador industrial, cuando la temperatura y la humedad salga de unos rangos, que salga la notificación y automáticamente se accionen los compresores para enfriar más la sala.

En el terreno más comercial, la tecnología de este proyecto, cambiando los tipos de los sensores, se podría usar para mejorar el mantenimiento y la logística, sensorizando todo lo que fuera necesario para ello, desde un simple frigorífico, en el tema de la temperatura, hasta una máquina de vending, poniendo todo tipo de sensores para detectar productos, el dinero del monedero, mal funcionamientos de botones e incluso vandalismo. En esta rama se podría usar esos mismos datos para realizar una técnica llamada BigData, que en lo que se basa esta técnica es en captación, administración y análisis de datos a gran escala, para usar esta técnica sería necesario implementar esta tecnología en varios sitios, varias máquinas y sensorizar varios parámetros, con estos datos y en el caso de que solo tuviéramos temperatura y humedad como ahora, podríamos predecir en que rango de temperatura y humedad es la idónea para que la máquina funcione perfectamente y ello conlleva en elegir el sitio idóneo para tener menos problemas con el mantenimiento debido a esos factores, etc. Si cambiamos los sensores, por ejemplo, a ponerlos de presión en cada producto, podríamos llegar a realizar patrones de consumo de ciertos productos y llegar a usar esos datos para poder poner las maquinas en los lugares idóneos, con los productos idóneos para esos lugares.

Como vemos, este proyecto puede tener infinidades de posibles ampliaciones y aplicaciones en la vida real, tanto comerciales como tecnológicas, por ello definir una línea futura de ampliación es muy difícil, ya que el abanico de posibilidades es tan amplio como necesidades y aplicaciones puede tener.

Apéndice A

Presupuesto

Para realizar el proyecto, a la hora de comprar los materiales, adquirimos dos Kits en los que venían más herramientas que no usamos pero nos salía más económico comprar estos kits que los componentes por separado.

Nota: Los precios son como se compraron los Kits, el Kit de Raspberry ya no se vende y el de Arduino varía según el día por la política de cambio de moneda, ambos se compraron en Amazon y Aliexpress respectivamente.

Item comprado	Componentes	Cantidad	Precio
Kit Raspberry pi	• Raspberry pi tipo B+ • Caja protectora • Cable HDMI • Mini Protoboard • 20 luces led • 2 pulsadores • Disipadores • Resistencias • Cable de alimentación • Cables para protoboard • Adaptador Wifi 802.11n • Tarjeta microSD de 8 gigas, clase 10 • Cable USB/pines • Adaptador Puertos GPIO • Cable adaptador GPIO	1	68.69 €
Kit Arduino	• Arduino UNO R3 • Cable USB • Cables protoboard • Protoboard • Leds de 5 colores • Resistencias • Potenciómetro • Zumbador • Circuito integrado 74HC595 • Receptor infrarrojos • LM35 • NTC • Interruptor • Botones • Mando control remoto • Pantalla 4 dígitos • Matriz de puntos leds 8x8 • Pantalla de un digito • Controlador de motor • Motor paso a paso • Servo • Pantalla LCD • Modulo Joystick • DHT11 • Prueba de agua • Modulo, llavero y tarjeta RFID • Módulo de sonido • Relé • Reloj a tiempo real • Teclado 4x4 • Luces RGB • Módulo de pila 9V	1	26.12 €
		total	94.81 €

Figura A.1: Presupuesto del Proyecto

Apéndice B

Códigos

B.1. Código Arduino

```
// Sketch de recojida de parametros fisicos mediante dth11
// lm35
// Escrito por Alberto Hoyas

// Libreria para DHT11
  #include "DHT.h"

// Pin del Arduino al cual esta conectado el sensor
  #define DHTPIN 2
  int analog_pin = A1;   //lm35 analogico
  float temperatura;    //temperatura para lm35

  #define DHTTYPE DHT11   // DHT 11

// Inicializa el sensor
  DHT dht(DHTPIN, DHTTYPE);
// variables para ir guardando las maximas de humedad y
// temperatura y las minimas de humedad y temperatura
  int maxh=0,minh=100,maxt=0,mint=100, i=0,  n=0;
```

```
// Configura Arduino
  void setup() {
  Serial.begin(9600);
  dht.begin();
  }

  void loop() {
// Espera dos segundos para realizar la primera medición.
  delay(3000);

// El sensor muestrea la temperatura del LM35 y
// realiza una cuenta para pasarla a celsius
  temperatura = analogRead(analog_pin);
  temperatura =  (5.0 * temperatura * 100.0)/1024.0;

// Obtiene la Humedad
  float h = dht.readHumidity();
// Obtiene la Temperatura en Celsius
  float t = dht.readTemperature();

// Control de error
  if (isnan(h) || isnan(t)) {
    Serial.println("Fallo al leer el sensor");
    return;
  }

  Serial.print(h);    //Temperatura
  Serial.print("|");
  Serial.print(t);    //Humedad
  Serial.print("|");
//Comprobacion de maximos y minimos de humedad y temperatura
   if (maxh<h)
     maxh=h;
   if (h<minh)
     minh=h;
```

```
   if (maxt<t)
     maxt=t;
   if (t<mint)
     mint=t;

  Serial.print(maxh);     //Humedad Máxima
  Serial.print("|");
  Serial.print(minh);     //Humedad Minima
  Serial.print("|");
  Serial.print(maxt);     //Temperatura Máxima
  Serial.print("|");
  Serial.print(mint);     //Temperatura Minima
  Serial.print("|");

  Serial.print(temperatura);   //TemperaturaLM35
  Serial.print("|");

// Marca de Fin de Línea en Python
  Serial.print("|f|");
  Serial.println();
}
```

B.2. Código Python

```
#!/usr/bin/python
import MySQLdb
# Importamos la libreira de PySerial
import serial
# Abrimos el puerto del arduino a 9600
PuertoSerie = serial.Serial('/dev/ttyUSB0', 9600)
# Establecemos la conexi??n con la base de datos
bd = MySQLdb.connect("localhost","root","1234","prueba_tfg")
# Preparamos el cursor que nos va a ayudar a realizar
# las operaciones con la base de datos
```

```
cursor = bd.cursor()
# Creamos un buble sin fin
while True:
# leemos hasta que encontarmos el final de linea
  sArduino = PuertoSerie.readline()


# Mostramos el valor leido y eliminamos el
                            salto de linea del final
  print "Valor Arduino: " + sArduino

  sArduino=sArduino.strip()
  if sArduino.endswith ("|f|"):
Cadena=sArduino.split("|",7)
print (Cadena)
sHum=Cadena[0:1]
print (sHum)
sTemp=Cadena[1:2]
print (sTemp)
sTempmax=Cadena[2:3]
print (sTempmax)
sTempmin=Cadena[3:4]
print (sTempmin)
sTempLM=Cadena[4:5]
print (sTempLM)
sHummax=Cadena[5:6]
print (sHummax)
sHummin=Cadena[6:7]
print (sHummin)
a="".join(sHum)
b="".join(sTemp)
e="".join(sTempmax)
o="".join(sTempmin)
g="".join(sTempLM)
h="".join(sHummax)
```

```
i="".join(sHummin)

        c= float (a)
d= float (b)
j= float (e)
k= float (o)
l= float (g)
m= float (h)
n= float (i)

  sql= " INSERT INTO prueba(Humedad,Temperatura,TempMax,
        TempMin,TempLM35,HumedMax,HumedMin) VALUES
          (%f, %f ,%f, %f, %f, %f, %f)" % (c,d,j,k,l,m,n)

   cursor.execute(sql)
   bd.commit()
PuertoSerie.close()
bd.close()
```

B.3. Script cron Raspberry pi

```
*/1 * * * 1-7 /home/pi/script.sh
```

B.4. Códigos Interfaz Java

B.4.1. Conexion.java

```
/*
 * To change this license header, choose License Headers in
 Project Properties.
 * To change this template file, choose Tools | Templates
```

```
 * and open the template in the editor.
 */
package estacion;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

/**
 *
 * @author Alberto
 */
public class Conexion {

 Connection con=null;
// Método con el que podremos realizar la
// conexión a la base de datos
 Connection establecer_conexion(){
 try {
  // TODO code application logic here
  // Cargamos el driver de mysql
  Class.forName("com.mysql.jdbc.Driver");

  String conexionURL="jdbc:mysql://192.168.1.110:3306/
                /prueba_tfg?user=sony1&password=root";
// Establecemos la conexión guardandola en la
// variable con de tipo Connection
  con = DriverManager.getConnection(conexionURL);


  }catch (SQLException e){
   System.out.println("SQL Exception: " + e.toString());
  } catch (ClassNotFoundException ex) {
   System.out.println("Exception: " + ex.toString());
  }
```

```
// Devolvemos un objeto Connection que representa la
// conexion con la base de datos.
  return con;
 }

}
```

B.4.2. Estacion2.java

```
/*
 * To change this license header, choose License Headers in
 Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package estacion;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import javax.swing.table.DefaultTableModel;

/**
 *
 * @author Alberto
 */
public class Estacion2 extends javax.swing.JFrame {
Conexion conexion = new Conexion();
 Connection con = conexion.establecer_conexion();
  void mostrardatos() {

    DefaultTableModel modelo = new DefaultTableModel();
    modelo.addColumn("Temperatura");
```

```
    modelo.addColumn("TempMax");
    modelo.addColumn("TempMin");
    modelo.addColumn("TempLM35");
    modelo.addColumn("Humedad");
    modelo.addColumn("HumedMax");
    modelo.addColumn("HumedMin");
    tabladatos.setModel(modelo);

   String[]datos= new String[7];

   try {
    Statement st = con.createStatement();
    ResultSet rs =st.executeQuery("select * from prueba");

      while (rs.next()) {
         datos[0]=rs.getString(1);
         datos[1]=rs.getString(2);
         datos[2]=rs.getString(3);
         datos[3]=rs.getString(4);
         datos[4]=rs.getString(5);
         datos[5]=rs.getString(6);
         datos[6]=rs.getString(7);
       modelo.addRow(datos);
      }

    tabladatos.setModel(modelo);

   } catch (SQLException ex) {
// Logger.getLogger
      (Estacion2.class.getName()).log(Level.SEVERE,
                                          null, ex);
   }

}
/**
```

```
 * Creates new form Estacion2
 */
 public Estacion2() {
 initComponents();
 mostrardatos();
 }

/**
* This method is called from within the constructor to
initialize the form.
* WARNING: Do NOT modify this code. The content of this
method is always
* regenerated by the Form Editor.
*/
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed"
// desc="Generated Code">
 private void initComponents() {

  jScrollPane1 = new javax.swing.JScrollPane();
  tabladatos = new javax.swing.JTable();
  button_actualizar = new javax.swing.JButton();

  setDefaultCloseOperation
          (javax.swing.WindowConstants.EXIT_ON_CLOSE);

   tabladatos.setModel(new
                javax.swing.table.DefaultTableModel(
    new Object [][] {
        {},
        {},
        {},
        {}
   },
    new String [] {
```

```
    }
  ));
 jScrollPane1.setViewportView(tabladatos);

button_actualizar.setText("Actualizar");
button_actualizar.addActionListener
                (new java.awt.event.ActionListener() {
public void actionPerformed
                   (java.awt.event.ActionEvent evt) {
    button_actualizarActionPerformed(evt);
  }
});

javax.swing.GroupLayout layout = new
             javax.swing.GroupLayout(getContentPane());
 getContentPane().setLayout(layout);
 layout.setHorizontalGroup(
  layout.createParallelGroup
            (javax.swing.GroupLayout.Alignment.LEADING)
   .addGroup(layout.createSequentialGroup()
   .addContainerGap()
   .addComponent(jScrollPane1,
                 javax.swing.GroupLayout.DEFAULT_SIZE,
                                  675, Short.MAX_VALUE)
   .addContainerGap())
   .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
                         layout.createSequentialGroup()
   .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
                                        Short.MAX_VALUE)
   .addComponent(button_actualizar)
   .addGap(55, 55, 55))
 );
 layout.setVerticalGroup(
  layout.createParallelGroup
```

```
                (javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
                               layout.createSequentialGroup()
    .addContainerGap(84, Short.MAX_VALUE)
    .addComponent(button_actualizar)
    .addPreferredGap
      (javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
    .addComponent(jScrollPane1,
             javax.swing.GroupLayout.PREFERRED_SIZE, 199,
                    javax.swing.GroupLayout.PREFERRED_SIZE)
    .addContainerGap())
);

  pack();
 }// </editor-fold>

private void button_actualizarActionPerformed
                            (java.awt.event.ActionEvent evt) {
// TODO add your handling code here:

   mostrardatos();
}

/**
* @param args the command line arguments
*/
public static void main(String args[]) {
/* Set the Nimbus look and feel */
//<editor-fold defaultstate="collapsed" desc=" Look and
//feel setting code (optional) ">
/* If Nimbus (introduced in Java SE 6) is not available,
stay with the default look and feel.
* For details see http://download.oracle.com/javase/
                         /tutorial/uiswing/lookandfeel/plaf.html
*/
```

```
  try {
   for (javax.swing.UIManager.LookAndFeelInfo info :
           javax.swing.UIManager.getInstalledLookAndFeels()) {
    if ("Nimbus".equals(info.getName())) {
     javax.swing.UIManager.setLookAndFeel(info.getClassName());
     break;
    }
   }
 } catch (ClassNotFoundException ex) {
  java.util.logging.Logger.getLogger
        (Estacion2.class.getName()).log
                (java.util.logging.Level.SEVERE, null, ex);
} catch (InstantiationException ex) {
 java.util.logging.Logger.getLogger
         (Estacion2.class.getName()).log
                 (java.util.logging.Level.SEVERE, null, ex);
} catch (IllegalAccessException ex) {
  java.util.logging.Logger.getLogger
          (Estacion2.class.getName()).log
                 (java.util.logging.Level.SEVERE, null, ex);
} catch (javax.swing.UnsupportedLookAndFeelException ex) {
  java.util.logging.Logger.getLogger
          (Estacion2.class.getName()).log
                 (java.util.logging.Level.SEVERE, null, ex);
}
//</editor-fold>

/* Create and display the form */
java.awt.EventQueue.invokeLater(new Runnable() {
  public void run() {
   new Estacion2().setVisible(true);
  }
 });
}
// Variables declaration - do not modify
```

```
 private javax.swing.JButton button_actualizar;
 private javax.swing.JScrollPane jScrollPane1;
 private javax.swing.JTable tabladatos;
// End of variables declaration
}
```

Bibliografía

[1] Historia de la meteorología http://www.tutiempo.net/meteorologia/historia.html

[2] Breve Historia de la Meteorología http://www.aemet.es/documentos/es/conocenos/nuestra_historia/breve_historia_meteorologia.pdf

[3] El pronóstico del tiempo a través de la historia http://astrofactoria.webcindario.com/Historia2.htm

[4] Bloq: Historia de la meteorología https://problemascolombianos.wordpress.com/ciencia-y-tecnologia/historia-de-la-meteorologia/

[5] Barómetro de Torricelli http://www.escuelapedia.com/barometro-de-torricelli/

[6] Higrómetro http://teleformacion.edu.aytolacoruna.es/AYC/document/atmosfera_y_clima/humedad/aparatosmedir.htm

[7] Anemoómetro http://www.infoagro.com/instrumentos_medida/doc_anemometro_velocidad_viento.asp?k=80

[8] Registros Climáticos AEMET http://www.aemet.es/es/idi/clima/registros_climaticos

[9] Satélites Meteorológicos http://www.proteccioncivil.es/catalogo/carpeta02/carpeta24/vademecum12/vdm031.html

[10] Datos AEMET http://datosclima.es/Aemet2013/DescargaDatos.html

[11] Web Oficial Arduino: https://www.arduino.cc

[12] Web Oficial Raspberry pi https://www.raspberrypi.org/

[13] Web Oficial Java https://www.java.com/es/

[14] Web Oficial Python https://www.python.org/

[15] Web Oficial MySQL https://www.mysql.com/

[16] Web Oficial Servidor Apache https://httpd.apache.org/

[17] Web Oficial PHPMyAdmin https://www.phpmyadmin.net/

[18] Web Oficial NetBeans https://netbeans.org/

[19] Hoja de características DHT11 http://www.micropik.com/PDF/dht11.pdf

[20] Librería DHT11 para Arduino https://docs.google.com/file/d/0B0hsUkhqWH97NnM5QWZlN0ZsYVE/edit

[21] Hoja de Características LM35 http://www.ti.com/lit/ds/symlink/lm35.pdf

[22] Medir Temperatura con Arduino y LM35 http://www.luisllamas.es/2015/07/medir-temperatura-con-arduino-y-sensor-lm35/

[23] Tutoriales para instalación en Raspberry pi https://geekytheory.com/

[24] Bloq: Conectar Arduino a Raspberry pi http://fuenteabierta.teubi.co/2012/12/conectando-la-raspberry-pi-al-arduino.html

[25] Bases de datos con Python http://codehero.co/python-desde-cero-bases-de-datos/

[26] Administrar bases de datos en Python http://codehero.co/python-desde-cero-bases-de-datos/

[27] Codificación de ficheros en linux http://codehero.co/python-desde-cero-bases-de-datos/

[28] Automatizar tareas en Raspberry pi http://miraquelodije.blogspot.com.es/2013/03/automatizar-tareas-en-raspberry-pi-cron.html

[29] Python para principiantes http://librosweb.es/libro/python/

[30] Definiciones en general https://es.wikipedia.org