



**ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)**  
**GRADO EN INGENIERÍA ELECTROMECÁNICA**  
Especialidad Electrónica

**SISTEMA DOMÓTICO DE  
NOTIFICACIÓN DE ALARMAS POR  
SMS**

Autor: Inés Arnaiz Lázaro-Carrasco  
Director: Juan Luis Zamora Macho

Madrid

Julio 2018

Copyright © 2018 by Inés Arnaiz Lázaro-Carrasco

This dissertation was typeset with  $\text{\LaTeX}$  and compiled in  $\text{\TeX} \text{maker}$ . The font families used are Bitstream Charter, Utopia, Bookman, and Computer Modern. Unless otherwise noted, all figures were created by the author using Microsoft PowerPoint<sup>®</sup> and MATLAB<sup>®</sup>.

## **AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN RED DE PROYECTOS FIN DE GRADO, FIN DE MÁSTER, TESINAS O MEMORIAS DE BACHILLERATO**

### **1º. Declaración de la autoría y acreditación de la misma.**

El autor D. INÉS ARNAIZ LÁZRO-CARRASCO

DECLARA ser el titular de los derechos de propiedad intelectual de la obra: SISTEMA DOMÓTICO DE NOTIFICACIÓN DE ALARMAS POR SMS, que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual.

### **2º. Objeto y fines de la cesión.**

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad, el autor **CEDE** a la Universidad Pontificia Comillas, de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, los derechos de digitalización, de archivo, de reproducción, de distribución y de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra a) del apartado siguiente.

### **3º. Condiciones de la cesión y acceso**

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia habilita para:

- a) Transformarla con el fin de adaptarla a cualquier tecnología que permita incorporarla a internet y hacerla accesible; incorporar metadatos para realizar el registro de la obra e incorporar “marcas de agua” o cualquier otro sistema de seguridad o de protección.
- b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato.
- c) Comunicarla, por defecto, a través de un archivo institucional abierto, accesible de modo libre y gratuito a través de internet.
- d) Cualquier otra forma de acceso (restringido, embargado, cerrado) deberá solicitarse expresamente y obedecer a causas justificadas.
- e) Asignar por defecto a estos trabajos una licencia Creative Commons.
- f) Asignar por defecto a estos trabajos un HANDLE (URL persistente).

### **4º. Derechos del autor.**

El autor, en tanto que titular de una obra tiene derecho a:

- a) Que la Universidad identifique claramente su nombre como autor de la misma
- b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.
- c) Solicitar la retirada de la obra del repositorio por causa justificada.
- d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

### **5º. Deberes del autor.**

El autor se compromete a:

- a) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.
- b) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.
- c) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e intereses a causa de la cesión.

- d) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción de derechos derivada de las obras objeto de la cesión.

**6º. Fines y funcionamiento del Repositorio Institucional.**

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.
- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusive del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.
- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.
- La Universidad se reserva la facultad de retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a 16 de Julio de 2018

**ACEPTA**

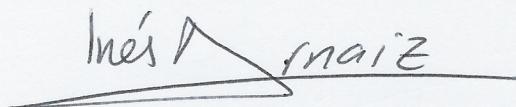
Fdo. .... Inés M. Martínez .....

Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional:

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título  
SISTEMA DOMÓTICO DE NOTIFICACIÓN  
DE ALARMAS POR SMS  
en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el  
curso académico 2017/2018 es de mi autoría, original e inédito y  
no ha sido presentado con anterioridad a otros efectos. El Proyecto no es  
plagio de otro, ni total ni parcialmente y la información que ha sido tomada  
de otros documentos está debidamente referenciada.

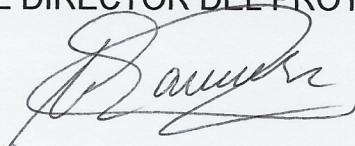
Fdo.: Inés Arnaiz Lázaro-Carrasco

Fecha: 16/07/2018



Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO



Fdo.: Juan Luis Zamora Macho Fecha: 16/07/2018



*A mis padres*



# SISTEMA DOMÓTICO DE NOTIFICACIÓN DE ALARMAS POR SMS

Autor: Arnaiz Lázaro-Carrasco, Inés

Director: Zamora Macho, Juan Luis.

Entidad colaboradora: ICAI – Universidad Pontificia Comillas

## RESUMEN DEL PROYECTO

### 1. Introducción

Gracias al avance de la informática y los sistemas de comunicación, la domótica surge en los años 70 y experimenta un gran desarrollo en los años 80 cuando se introduce a nivel comercial. La expansión se puede resumir en la estandarización, cada vez más extendida gracias al Open Source, y a la mejora de las comunicaciones. Además, permite un consumo más eficiente de energía y una mayor automatización de grandes edificaciones lo que está cobrando mucha importancia en los últimos años [1].

Este proyecto tiene como objetivo el desarrollo de un sistema de envío de alarmas por *Short Message Service* (SMS) en un ámbito doméstico y de pequeña empresa. El método de comunicación de estas alarmas, el SMS, se ha elegido por las siguientes razones:

- Es un canal eficaz y rápido puesto que se trata de mensajes pequeños [2]. Es además muy fiable ya que los SMS se transmiten a través del canal de señalización. Este canal es el más protegido por las operadoras debido a que es el utilizado para transmitir información sobre el servicio. Por esta misma razón es el que menos intensidad de señal necesita, incluso menos que las llamadas telefónicas o Internet [3].
- Se transmite directamente entre usuario y operadora, no depende de servidores de terceros como un mensaje enviado por Internet [4].

Las alarmas elegidas para este proyecto son: la salida de un rango de temperaturas y el fallo y restablecimiento del suministro eléctrico. Estas

dos alarmas se han elegido a raíz de una experiencia personal en la que, como consecuencia de un corte en el abastecimiento eléctrico, se desconfiguró la referencia de temperatura del sistema de climatización de una segunda vivienda subiendo la temperatura hasta 35 °C durante un mes.

### 2. Metodología

Para cumplir los requisitos enumerados en la Sección 1, los pasos a seguir son:

1. Diseño de una arquitectura capaz de cumplir con los requerimientos.
  - I Elección de los módulos que serán necesarios (sensores, actuadores, microcontrolador).
  - II Elección de la lógica y protocolos de comunicación entre ellos.
2. Elección y compra de los modelos de *hardware* para que realicen las funciones establecidas en el diseño.
3. Creación e implementación del software.
4. Pruebas de cada elemento *hardware* por separado y corrección del *software*.
5. Pruebas del dispositivo en su conjunto y corrección del *software*.

En la Figura 2.1 se puede muestra el diseño final, en el que se han utilizado:

- Un sensor de temperatura (modelo HDC1080) y un reloj en tiempo real (RTC,

modelo DS1307) conectados mediante el protocolo I<sup>2</sup>C al micro-controlador.

- Una Raspberry Pi 3B como micro-controlador.
- Un módulo GSM SIM800C conectado por puerto serie y controlado mediante comandos de Hayes con la Raspberry.
- Un conversor de 5 a 3.3V junto con un adaptador de micro USB para la detección de corriente (conectado al pin 18 del micro-controlador).
- Un router para la actualización de la hora del RTC, un *power bank* que permite el funcionamiento durante el fallo en el abastecimiento eléctrico y un *Shield* que permite una conexión limpia y cómoda de todos los sensores al micro-controlador

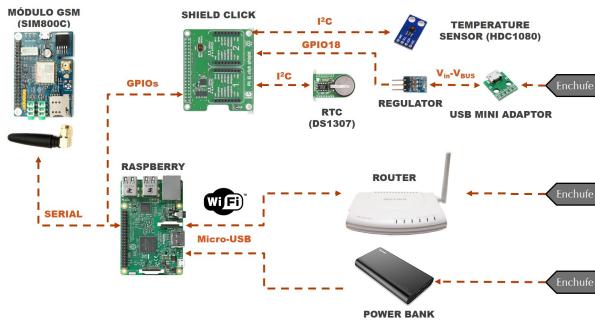


Figura 2.1. Arquitectura y conexiones del hardware.

### 3. Resultados

Tras la implementación del software y tal como se indica en la Sección 2, se somete el *hardware* a diferentes pruebas. Las pruebas y sus resultados se explican a continuación.

#### 3.1. Módulo GSM

Para comprobar el correcto funcionamiento de módulo GSM se le sometió a una prueba que con-

sistió en mandar un SMS con el texto *HOLA*. En esta prueba, a través del puerto serie se transmitieron los comandos necesarios para el envío del SMS y se decodificará la respuesta obtenida.

En la Figura 3.1 se muestra la secuencia ordenada de comandos y respuestas obtenidas. En la Figura 3.2 se muestra un ejemplo de los bytes recibidos en el puerto serie del micro-controlador durante el proceso. En este caso la respuesta al primer comando: *AT OK*.

| ENVÍO                 | RESPUESTA                           |
|-----------------------|-------------------------------------|
| AT                    | AT 13 10 OK                         |
| AT+CMGF=1             | AT+CMGF=1 13 10 OK                  |
| AT+CSCS="GSM"         | AT+CSCS="GSM" 13 10 OK              |
| A+CMGS="+34676217306" | AT+CMGS="+34642330167" 13 10 > HOLA |
| HOLA CRTL+Z           | +CMGS:202 13 10 OK                  |

Figura 3.1. Secuencia de comandos y respuestas a través del puerto serie durante el proceso de envío del SMS.

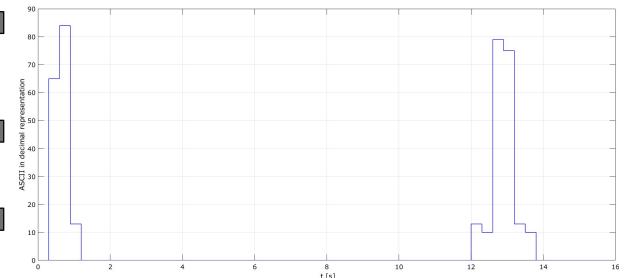


Figura 3.2. Representación decimal de la respuesta obtenida al comando *AT*.

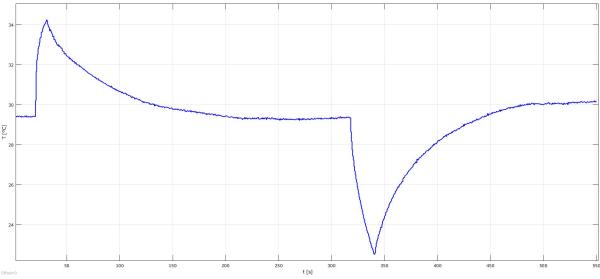
#### 3.2. Real Time Clock

Para probar el funcionamiento del reloj en tiempo real, se comparó la fecha y hora de dicho sensor con la fecha y hora del ordenador de trabajo conectado a Internet. El resultado de la prueba se puede ver en la Figura 3.3. Así pues, se puede ver que el RTC marca la hora correctamente sin diferencia con respecto a Internet.

La imagen está dividida en dos partes, la de la



**Figura 3.3.** RTC en funcionamiento. Fuente: elaboración propia gracias a <https://time.is/es/>.



**Figura 3.4.** Respuesta del sensor de temperatura a diferentes fuentes de calor.

izquierda que muestra la respuesta del RTC a Simulink y la de la derecha, una página web que muestra la hora en tiempo real. En el diagrama de Simulink los *displays* están en el siguiente orden:

1. Segundos: el diagrama de Simulink muestra 7, igual que en la página web.
2. Minutos: ambas partes muestran el mismo número, el 42.
3. Horas: ambas partes muestran el mismo número, el 23.
4. Día de la semana: el RTC indica que el día es el primero de la semana, en este caso el domingo (debido a la configuración).
5. Día del mes: ambas partes muestran el mismo número, el 15.
6. Mes: ambas partes muestran el mismo número, el 7 (julio).
7. Año: ambas partes muestran el mismo número, el 18.

### 3.3. Sensor de temperatura

Para probar el funcionamiento del sensor de temperatura, se le aportó una fuente de calor en el segundo 20, y un fuente de “frío” en el segundo 318 obteniendo el resultado de la Figura 3.4.

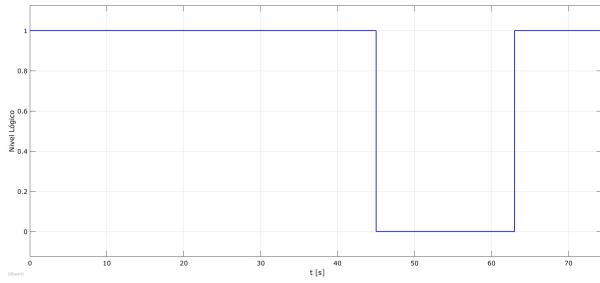
El sensor se estabilizó entorno a los 29.5 °C mientras no se le oportó ninguna fuente de calor. En el segundo 20, se le aportó una fuente de calor, que en este caso calor humano a través de un dedo de la mano. Se observó como a partir de este momento, la temperatura comenzó a ascender hasta llegar a 34.2 °C en el segundo 30. En este momento en el que se retira dicha fuente de calor, la temperatura comienza a descender se estabilizó de nuevo en torno a los 29 °C.

Se ensayó también el funcionamiento en temperaturas más bajas, acercando una fuente de “frío”, que en este caso fue un hielo protegido por un paño para no mojar el sensor. Esta fuente se aportó en el segundo 318 y se retiró tras unos instantes (segundo 340). En este momento la temperatura descendió descendido hasta 22.5 °C. Se apreció como la temperatura desciende cuando el sensor está en contacto con la fuente “fría”, y cuando esta se retira comienza de nuevo a ascender hasta su régimen estable.

### 3.4. Detección de la caída de la red eléctrica

Para comprobar que el *hardware* dedicado a la detección de la caída y restablecimiento de la corriente funciona correctamente, se sometió a la siguiente prueba: se dio tensión al dispositivo, se retiró la tensión y se volvió a alimentar. El resultado de este proceso muestra en la Figura 3.5.

La detección de la corriente se ha diseñado pa-



**Figura 3.5.** Respuesta del *hardware* dedicado a la detección de caída de la red eléctrica ante distintas situaciones.

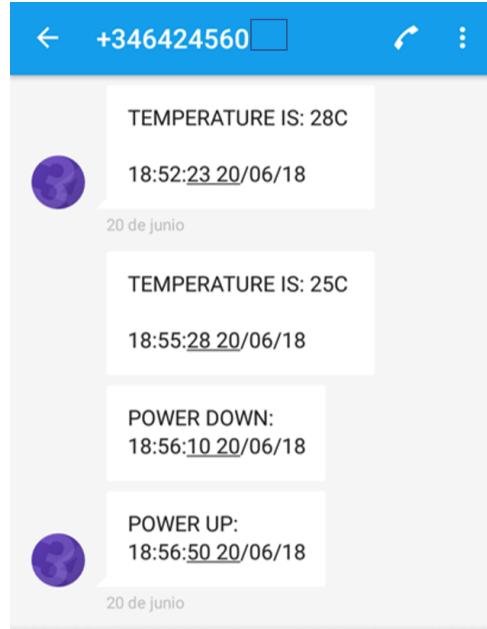
ra que solo dé aviso si la tensión de entrada se estabiliza en un valor (0 ó 1) durante más de 5 segundos. Así, en la gráfica de resultados se ve como habiéndose retirado la tensión en el segundo 40, la señal de caída de tensión no se estabiliza en 0 hasta el segundo 45. Del mismo modo, solo vuelve a estabilizarse en 1 en el segundo 63, habiéndose retirado la tensión en el segundo 58.

### 3.5. Funcionamiento del dispositivo

Se realizó una última prueba, es decir, la notificación tanto de caída y restablecimiento del suministro, como de salida y retorno al rango de temperatura establecido.

Se puede apreciar en la Figura 3.6 que primero se ha notificado que la temperatura ha rebasado el límite superior de 27 °C . En este caso la temperatura ascendió con tanta rapidez, que en el momento de la notificación la temperatura era de 28 °C. Más tarde, cuando descendió a 25 °C se volvió a notificar, ya que volvió a entrar en el “rango normal”. Después de someter al dispositivo a cambios de temperatura, se le quitó la corriente al mismo para simular una caída de tensión, y pasados unos segundos, se volvió a alimentar.

Como se puede ver, los mensajes contienen la hora del evento y la fecha; además del texto que indica la situación que ha provocado la notificación. En el caso de que haya sido debido a la temperatura, ésta también se indica en el mensaje.



**Figura 3.6.** Respuesta del dispositivo completo ante distintas situaciones.

## 4. Conclusiones

### 4.1. Conclusiones

#### 4.1.1. Módulo GSM

La base del proyecto fue la notificación por SMS, para ello se utilizó el módulo GSM SIM800C. Se implementó el protocolo de comunicación serie junto con los comandos de Hayes con éxito para que el micro-controlador se pudiera comunicar correctamente con dicho módulo.

Se consiguió decodificar todas las respuestas del módulo ante los comandos. Y así se pudo ordenar los comandos correctamente para el envío del SMS. Por último, se logró enviar SMS con texto a cualquier teléfono.

Durante la implementación del proceso para mandar el SMS surgieron varios problemas. Uno de ellos está relacionado con la velocidad de la comunicación. El módulo trabaja con unos baudios autoajustables en un rango bastante amplio; desde la Raspberry solo se consiguió comunicar-

se a una velocidad de 4800 baudios. Una vez conseguido el envío de SMS, estos estaban vacíos. La causa era el exceso de tiempo entre las dos últimas ordenes necesarias: el comando para mandar el SMS y el texto del mismo. El micro-controlador dejaba un tiempo de espera excesivo entre la transmisión de ambas.

Por último, cuando se consiguió enviar texto, se enviaba un numero exorbitado de @ indeseadas que tras varias pruebas se eliminaron. La causa era el envío de ceros por el puerto serie para dejarlo en reposo entre el comando de envío, el texto y la orden de que el texto está finalizado. Para ello se creó un buffer que permitía colocar dichos ceros donde mejor convenía — los ceros eran necesarios ya que el micro-controlador solo es capaz de mandar mensajes de la misma longitud por el puerto serie.

#### 4.1.2. Sensores

El sensor de temperatura HDC1080 ya había sido implementado anteriormente en un micro-controlador distinto. Al intentar implementar los bloques que ya habían sido usados, el sensor no respondía. Tras una serie de pruebas, se dedujo que el problema era que al mandarse la orden de “tomar medida” sin dejar ningún tiempo para leer dicha medida, nunca comunicaba el resultado al micro-controlador. Así, se diseño un nuevo bloque que ordenaba medir, y leía la respuesta por turnos.

El reloj en tiempo real DS1307 se implementó desde cero creando los bloques. Además, se sincronizó con Internet a través de la Raspberry de manera que si hay conexión, el reloj se pone automáticamente en hora.

Uno de los objetivos del proyecto fue detectar la caída y restablecimiento del abastecimiento eléctrico. Se logró detectar estas situaciones de una manera simple y barata y se implementó con éxito en el conjunto del proyecto.

#### 4.1.3. Presupuesto

El precio total del hardware utilizado asciende a 118 €, pero si se utilizan componentes industriales y en grandes cantidades este precio disminuye notablemente, a aproximadamente 60€

**Palabras clave:** SMS · Alarmas · Módulo GSM · Detección de fallo del abastecimiento eléctrico · Rango de temperaturas · Raspberry Pi · Matlab y Simulink

## Referencias

- [1] C. Reinisch, M. J. Kofler, and W. Kastner, “Thinkhome: A smart home as digital ecosystem,” in *4th IEEE International Conference on Digital Ecosystems and Technologies*, pp. 256–261, April 2010.
- [2] “Tia eia-637-a, short message service,” tech. rep., Telecommunications Industry Association, 1999.
- [3] C. Peersman, S. Cvetkovic, P. Griffiths, and H. Spear, “The global system for mobile communications short message service,” *IEEE Personal Communications*, vol. 7, pp. 15–23, June 2000.
- [4] X. Meng, P. Zerfos, V. Samanta, S. H. Y. Wong, and S. Lu, “Analysis of the reliability of a nationwide short message service,” in *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*, pp. 1811–1819, May 2007.



# DOMOTIC SMS ALARM NOTIFICATION SYSTEM

**Author:** Arnaiz Lázaro-Carrasco, Inés

**Director:** Zamora Macho, Juan Luis.

Collaborating entity: ICAI – Universidad Pontificia Comillas

## PROJECT ABSTRACT

### 1. Introduction

Domotics was born in the 70's due to advances in computer science and communications. It was greatly developed in the 80's when it was made commercially available. Standardization, which wouldn't have been possible without the Open Source community, and the development of telecommunications made the spread of domotics achievable. Domotics boosts energy consumption efficiency and supports the automation of sizable buildings, two things that are rapidly gaining importance. [1].

The main objective of this project is to notify alarms through *Short Message Service* (SMS) in the domestic and small business area. SMS has been chosen as the communication method for the following reasons:

- The messages that are transmitted are short, making this channel fast and effective [2]. It is also extremely reliable because these messages are transmitted through telecommunication's signaling. This is the most protected channel by the operators as it carries critical service information. As another consequence, it requires the least signal intensity, even less than phone calls or the Internet [3].
- It doesn't require any middleman servers like the Internet, the communication is made from operator to user [4].

Temperature range and power outage are the notifications selected for this project. They have been selected following a personal experience where a power outage caused a loss of the tem-

perature reference at a second residence raising the temperature up to 35 °C for a month.

### 2. Methodology

To achieve the requirements mentioned in the section 1, the steps to follow were:

1. Design of an architecture capable of meeting the requirements.
  - I Selection of the necessary modules (sensors, actuators, microcontrollers).
  - II Choice of the logic and communication protocols between them.
2. Selection and purchase of the hardware modules which are able to perform the functions declared in the design.
3. Software creation and deployment.
4. Tests to each separate component and subsequent software corrections.
5. Tests of the whole device and deployed software corrections.

Figure 2.1 shows the final design, in which the following components have been used:

- A temperature sensor (model HDC1080) and a Real Time Clock (RTC, model DS1307) connected through I<sup>2</sup>C protocol to the microcontroller.
- A Raspberry Pi 3B as microcontroller.
- A GSM SIM800C module connected through the serial port and controlled with Hayes commands with the Raspberry.

- A 5 to 3.3V converter together with a micro USB adaptor for power outage detection (connected to pin 18 of the Raspberry).
- A router to update the RTC's date and time, a *power bank* which supplies current to the microcontroller during a power outage and a *Shield* to connect cleanly and simply the sensors to the microcontroller.

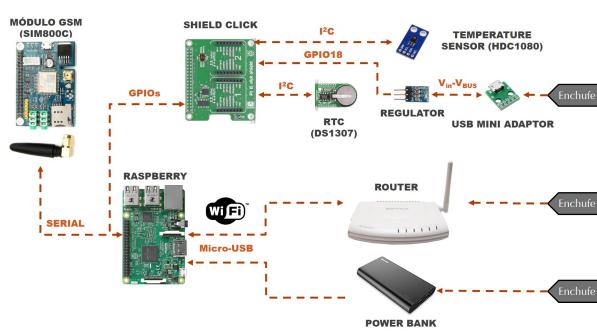


Figure 2.1. Hardware architecture and connections.

### 3. Results

After the software deployment and as it is indicated in section 2, different tests are performed. These tests and their results are explained in this section.

#### 3.1. GSM Module

To check the correct operation of the GSM module it will be put to a test in which it will have to send an SMS with the text *HOLA*. During the test, the necessary commands will be transmitted to the module and its responses will be decoded.

Figure 3.1 shows the sequence of commands transmitted and responses obtained and Figure 3.2 an example of the bytes received at the serial port during the process. In this case, the response to the first command: *AT*.

| ENVÍO                 | RESPUESTA                           |
|-----------------------|-------------------------------------|
| AT                    | AT 13 10 OK                         |
| AT+CMGF=1             | AT+CMGF=1 13 10 OK                  |
| AT+CSCS="GSM"         | AT+CSCS="GSM" 13 10 OK              |
| A+CMGS="+34676217306" | AT+CMGS="+34642330167" 13 10 > HOLA |
| HOLA CRTL+Z           | +CMGS:202 13 10 OK                  |

Figure 3.1. Command and response sequence during the SMS process.

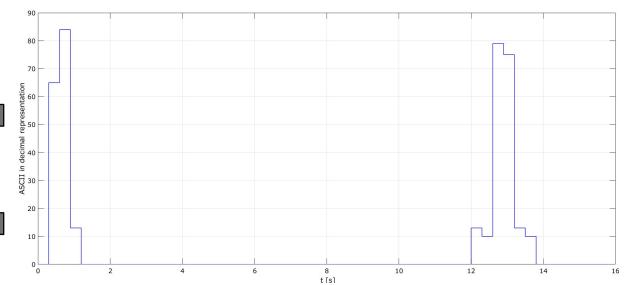


Figure 3.2. The decimal representation of the response obtained to the command *AT*.

#### 3.2. Real Time Clock

To verify the correct operation of the RTC, a comparison was made between the time and date extracted from it and the one from the Internet. The results can be seen in the Figure 3.3.



Figure 3.3. RTC running alongside the Internet. (<https://time.is/es/>).

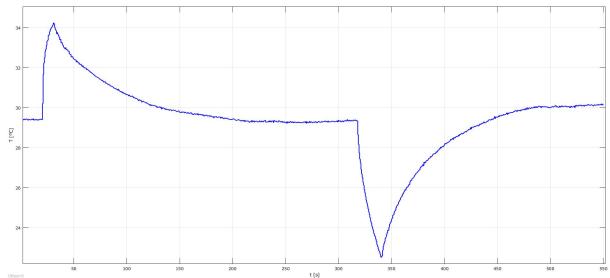
The image shows that the RTC works perfectly and the time is updated. The image is divided into two parts, the left-hand side shows the Simulink response of the RTC and the right-hand side a web page showing the current time. In the Simulink diagram the displays are in the follow-

ing order:

1. Seconds: the diagram shows a 7, same as the website.
2. Minutes: both sides show the same number, 42.
3. Hours: both sides show the same number, 23.
4. Day of the week: the RTC shows that it is the first day of the week, in this case Sunday (due to the settings).
5. Day of the month: both sides show the same number, 15.
6. Mes: both sides show the same number, 7 (July).
7. Año: both sides show the same number, 18.

### 3.3. Temperature Sensor

To test whether the temperature sensor works correctly, first a source of heat is brought close to the sensor, at second 20 and next, a "cold" source in the second 380 obtaining the response shown in the Figure 3.4. The sensor stabilizes at



**Figure 3.4.** Response of the temperature sensor to different heat sources.

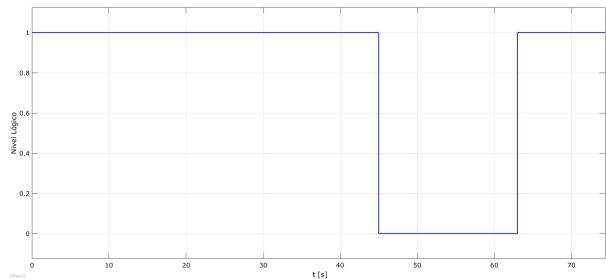
around 29.5 °C when no source is present. When the heat source is applied at the second 20, in this case, human heat through a finger, the temperature raises until 34.2 °C in 10 seconds. After

this time the source is withdrawn the temperature decreases until it stabilizes again at around 29 °C.

The decrease of temperature has also been tested, in this case by applying an ice cube protected by a cloth so that the sensor doesn't get wet. This "cold" source is applied at the second 318 and withdrawn seconds later (at second 340), at this time the temperature had dropped to 22.5 °C. Then the temperature starts to rise again.

### 3.4. Power outage detection

To test the power outage detection hardware the device was supplied with voltage during a short time, then the voltage was withdrawn and, finally, it was supplied again. The results of this process can be found in the Figure 3.5.



**Figure 3.5.** Response of the power outage detection hardware to the different situations of s

The detection has been designed to only notify when the power is down for more than 5 consecutive seconds. In the Figure 3.5 it can be seen that having withdrawn the power at second 40, the notification signal doesn't stabilize at 0 until the second 45. The same happens when the supply is applied at the second 58 and the signal stabilizes at 1 after 5 seconds.

### 3.5. Device as a whole

The last test was made to the whole device and deployed software, all notifications where

checked: power up, power down, temperature outside of range and temperature back to the range.

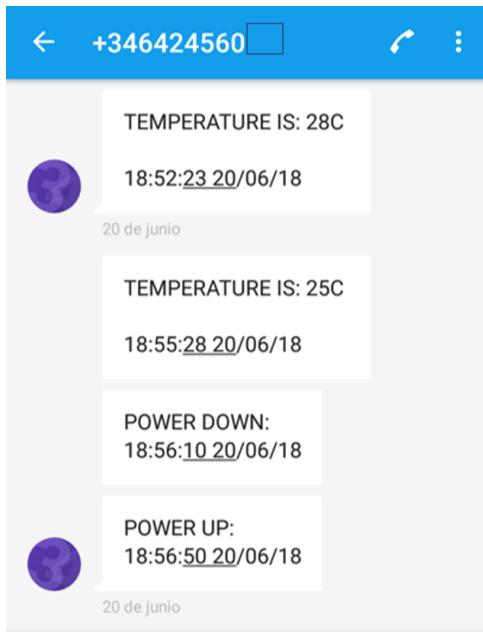


Figure 3.6. Device as a whole test.

As it can be seen in Figure 3.6 the first notification is that the temperature has surpassed the upper limit set at 27 °C. The temperature rose so fast that the notification was made at the moment when the temperature was 28 °C. The next notification shows that the temperature is back within range, 25 °C. After the temperature tests, the power outage notifications where successfully tested.

## 4. Conclusions

### 4.1. Conclusions

#### 4.1.1. GSM module

The main objective of the project was to send notifications through SMS for which a GSM SIM800c was used. Hayes commands through the serial port where implemented for the com-

munication of the module with the microcontroller.

The responses to the commands where decoded. This allowed the commands to be placed in order to correctly send the SMS. Lastly, the ability to send an SMS with text to any phone number was achieved.

Several problems arose during the development of the software. One of them was directly linked with the communication speed. The module works with wide auto-adjustable baud rate, but the Raspberry was only capable to communicate with the module at 4800 bits per second. Once the sending of the SMS was achieved, they were being sent blank. The cause was that the interval of time between the orders of “send SMS” and the actual text was too big.

Lastly, when a text was sent, a huge number of undesired @ where sent alongside the text. The cause where the zeros transmitted through the serial port with allowed to halt the communication between the SMS order, the text of the SMS and the finalized text command. A buffer was then created which allowed the zeros to be placed where it was most convenient — this zeros where necessary because the microcontroller could only send same length messages through the serial port.

#### 4.1.2. Sensors

The temperature sensor HDC1080 had already been used with another microcontroller. When the same blocks where deployed to the Raspberry it didn't respond well. After a series of tests, the problem was found: the request to take a temperature measurement was sent over and over again without leaving any time for the actual measurement to be transmitted. A new block was developed to measure and listen for the result in turns.

The Real Time Clock DS1307 was successfully implemented and synchronized with the Internet through the Raspberry. This means that if there

is Internet connexion, the time will update it selft.

One of the objectives of the project was to detect the power outage which was successfully met in a simple and unexpensive way.

#### 4.1.3. Budget

The hardware's total price is approximately 118 €, but by using industrial component and producing in big quantities, the cost can be lowered to roughly 60€.

**Key words:** SMS · Alarms · GSM module · Power outage detection · Temperature range · Raspberry Pi · Matlab y Simulink

## References

- [1] C. Reinisch, M. J. Kofler, and W. Kastner, "Thinkhome: A smart home as digital ecosystem," in *4th IEEE International Conference on Digital Ecosystems and Technologies*, pp. 256–261, April 2010.
- [2] "Tia eia-637-a, short message service," tech. rep., Telecommunications Industry Association, 1999.
- [3] C. Peersman, S. Cvetkovic, P. Griffiths, and H. Spear, "The global system for mobile communications short message service," *IEEE Personal Communications*, vol. 7, pp. 15–23, June 2000.
- [4] X. Meng, P. Zerfos, V. Samanta, S. H. Y. Wong, and S. Lu, "Analysis of the reliability of a nationwide short message service," in *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*, pp. 1811–1819, May 2007.



# Acknowledgment

Me gustaría agradecer en primer lugar a mi director, Juan Luis Zamora, por su apoyo indiscutible hasta en los momentos de mayor frustración. A mi familia por aguantar el estrés, no solo generador por este proyecto sino durante toda la carrera.

En segundo lugar, me gustaría agradecer su moral y apoyo a Julio y a Javi, por hacer los días en el laboratorio más agradables. A Jaime Boal porque sin él el L<sup>A</sup>T<sub>E</sub>X hubiera sido imposible. A Jose y Antonio, que siempre están dispuestos a echar una mano en el taller.

Por último a los amigos que no he nombrado, por ser mi punto de apoyo y mi ejemplo a seguir. Tanto los de siempre como los más recientes, porque en estos cuatro años he tenido la suerte de conocer a gente maravillosa.



# Índice general

|  |           |
|--|-----------|
| <b>1. Introducción</b>                                   | <b>1</b>  |
| 1.1. Introducción . . . . .                              | 1         |
| 1.2. Motivación . . . . .                                | 2         |
| 1.3. Estado del arte . . . . .                           | 3         |
| 1.3.1. Cellular Alarm . . . . .                          | 3         |
| 1.3.2. iSocket 3G . . . . .                              | 3         |
| 1.3.3. DD-5241 . . . . .                                 | 4         |
| 1.4. Objetivos . . . . .                                 | 4         |
| 1.5. Estructura del documento . . . . .                  | 4         |
| <b>2. Hardware</b>                                       | <b>5</b>  |
| 2.1. Estructura general del sistema . . . . .            | 5         |
| 2.2. Micro-controlador . . . . .                         | 6         |
| 2.2.1. Historia . . . . .                                | 6         |
| 2.2.2. Elección del modelo . . . . .                     | 7         |
| 2.3. Módulo GSM . . . . .                                | 7         |
| 2.3.1. Historia . . . . .                                | 7         |
| 2.3.2. Elección del modelo . . . . .                     | 7         |
| 2.3.3. Comunicación serie . . . . .                      | 9         |
| 2.3.4. Comunicación SMS . . . . .                        | 10        |
| 2.3.5. Comandos de Hayes . . . . .                       | 11        |
| 2.4. Real Time Clock (RTC) . . . . .                     | 11        |
| 2.4.1. Protocolo I <sup>2</sup> C . . . . .              | 12        |
| 2.5. Sensor de Temperatura . . . . .                     | 13        |
| 2.5.1. HDC 1080 . . . . .                                | 13        |
| 2.6. Pi 3 click Shield . . . . .                         | 14        |
| 2.7. Uninterrupted Power Supply (UPS) . . . . .          | 15        |
| 2.7.1. UPS HAT v1.2 Raspberry Pi . . . . .               | 15        |
| 2.7.2. Jackery Titan . . . . .                           | 16        |
| 2.8. Detección de la caída de la red eléctrica . . . . . | 16        |
| 2.8.1. Ethtool . . . . .                                 | 17        |
| 2.8.2. Adaptador de USB a DIP . . . . .                  | 17        |
| 2.8.3. Conversor de 5 a 3,3 V . . . . .                  | 17        |
| 2.9. Router . . . . .                                    | 18        |
| <b>3. Software</b>                                       | <b>19</b> |

## *Índice general*

|  |           |
|--|-----------|
| 3.1. Estructura general y buses de datos . . . . .       | 19        |
| 3.2. Módulo GSM . . . . .                                | 20        |
| 3.3. Sensores . . . . .                                  | 22        |
| 3.3.1. Real Time Clock . . . . .                         | 22        |
| 3.3.2. Sensor de temperatura . . . . .                   | 24        |
| 3.3.3. Detección de caída de corriente . . . . .         | 25        |
| 3.4. Máquina de estados general . . . . .                | 26        |
| 3.4.1. Rangos de temperatura . . . . .                   | 27        |
| 3.4.2. Cambio de modo del voltaje . . . . .              | 28        |
| <b>4. Resultados</b>                                     | <b>31</b> |
| 4.1. Módulo GSM . . . . .                                | 31        |
| 4.2. Real Time Clock . . . . .                           | 32        |
| 4.3. Sensor de temperatura . . . . .                     | 33        |
| 4.4. Detección de la caída de la red eléctrica . . . . . | 33        |
| 4.5. Funcionamiento del dispositivo . . . . .            | 34        |
| <b>5. Conclusiones y futuros desarrollos</b>             | <b>35</b> |
| 5.1. Conclusiones . . . . .                              | 35        |
| 5.1.1. Actuador . . . . .                                | 35        |
| 5.1.2. Sensores . . . . .                                | 36        |
| 5.1.3. Presupuesto . . . . .                             | 36        |
| 5.2. Futuros desarrollos . . . . .                       | 36        |
| <b>A. Presupuesto</b>                                    | <b>37</b> |
| A.1. Sumas parciales . . . . .                           | 37        |
| A.1.1. Componentes hardware . . . . .                    | 37        |
| A.1.2. Software . . . . .                                | 37        |
| A.1.3. Herramientas y equipos . . . . .                  | 38        |
| A.1.4. Mano de obra directa . . . . .                    | 38        |
| A.2. Presupuesto general . . . . .                       | 38        |
| <b>Bibliografía</b>                                      | <b>39</b> |

# Índice de figuras

|  |    |
|--|----|
| Figura 1.1. Cellular Alarm . . . . .   | 3  |
| Figura 1.2. iSocket 3G . . . . .   | 3  |
| Figura 1.3. DD-5241: Enchufe GSM/SMS . . . . .   | 4  |
| Figura 2.1. Esquema de conexión del <i>hardware</i> . . . . .  | 6  |
| Figura 2.2. Número de Raspberry Pi vendidas . . . . .  | 7  |
| Figura 2.3. Número de subscriptores únicos de GSM . . . . .  | 9  |
| Figura 2.4. Conexión serie de la Raspberry con el módulo GSM . . . . .   | 10 |
| Figura 2.5. Esquema de conexión del módulo GSM con la Raspberry Pi . . . . .   | 10 |
| Figura 2.6. Gráfico de las distintas capas de la señalización . . . . .  | 10 |
| Figura 2.7. Diagrama de bloques del RTC . . . . .  | 12 |
| Figura 2.8. Esquema de conexión del RTC con la Raspberry Pi . . . . .  | 13 |
| Figura 2.9. Registros del dispositivo DS1307 . . . . .   | 13 |
| Figura 2.10. Esquema de conexión del sensor de temperatura con la Raspberry P . . . . .  | 14 |
| Figura 2.11. Registros del dispositivo HDC 1080 . . . . .  | 14 |
| Figura 2.12. REsquema de conexión del <i>Shield</i> con la Raspberry Pi. . . . .   | 15 |
| Figura 2.13. Esquema de conexión del <i>Power Bank</i> con la Raspberry Pi . . . . .   | 16 |
| Figura 2.14. Esquema de conexión del adaptador de USB a DIP y del regulador con la Raspberry Pi . . . . .                                      | 17 |
| Figura 2.15. Conexiones entre el Regulador, el adaptador USB a DIP y la Raspberry . . .  | 18 |
| Figura 2.16. Esquema de conexión del router con la Raspberry Pi . . . . .  | 18 |
| Figura 3.1. Estructura del software . . . . .  | 19 |
| Figura 3.2. Esquema de buses del sistema . . . . .   | 20 |
| Figura 3.3. Diagrama de flujo del <i>software</i> del sensor de temperatura . . . . .  | 20 |
| Figura 3.4. Máquina de estados del módulo GSM . . . . .  | 21 |
| Figura 3.5. Diagrama de flujo del <i>software</i> del RTC . . . . .  | 23 |
| Figura 3.6. Diagrama de flujo del <i>software</i> del sensor de temperatura . . . . .  | 24 |
| Figura 3.7. Párametros del registro de configuración del HDC1080 . . . . .   | 25 |
| Figura 3.8. Diagrama de flujo del <i>software</i> de la detección de la caída de corriente . . .   | 26 |
| Figura 3.9. Máquina de estados general . . . . .   | 26 |
| Figura 3.10. Rangos de fluctuación de la temperatura . . . . .   | 27 |
| Figura 3.11. Diagrama de flujo del cambio del modo de voltaje . . . . .  | 28 |
| Figura 4.1. Imagen del <i>scope</i> de Simulink de la respuesta del módulo GSM a los comandos AT durante el proceso de mandar un SMS . . . . . | 31 |
| Figura 4.2. Imagen del <i>scope</i> de Simulink de la respuesta del módulo GSM a los comandos AT durante el proceso de mandar un SMS . . . . . | 32 |

## *Índice de figuras*

|   |    |
|---|----|
| Figura 4.3. RTC en funcionamiento . . . . .   | 32 |
| Figura 4.4. Respuesta del sensor de temperatura a diferentes fuentes de calor . . . . .   | 33 |
| Figura 4.5. Respuesta del <i>hardware</i> dedicado a la detección de caída de la red eléctrica ante distintas situaciones . . . . . | 33 |
| Figura 4.6. Respuesta del dispositivo completo ante distintas situaciones . . . . .   | 34 |

# Índice de tablas

|  |    |
|--|----|
| Tabla 2.1. Modelos de Raspberry Pi . . . . .                     | 8  |
| Tabla 2.2. Especificaciones del RTC . . . . .                    | 12 |
| Tabla 2.3. Especificaciones del <i>Shield</i> . . . . .          | 15 |
| Tabla 3.1. Rangos de temperatura y acciones necesarias . . . . . | 29 |
| TablaA.1. Sumas parciales: componentes HARDWARE . . . . .        | 37 |
| TablaA.2. Sumas parciales: <i>software</i> . . . . .             | 37 |
| TablaA.3. Sumas parciales: herramientas y equipos . . . . .      | 38 |
| TablaA.4. Sumas parciales: obra de mano directa . . . . .        | 38 |
| TablaA.5. Presupuestos generales . . . . .                       | 38 |



# 1

# Introducción

---

Este primer capítulo introduce la motivación del proyecto y sus objetivos. También proporciona un estudio del estado del arte y, finalmente, se explica la organización del documento.

---

## 1.1. Introducción

Hoy en día los ordenadores y la informática marcan la forma de vida, desde los parquímetros hasta las lavadoras. Gracias a la reducción del tamaño de los microprocesadores, a su estandarización y, por consiguiente, su abaratamiento, es posible el control de sistemas que hace unas décadas suponían un lujo. En 1971, época del primer microprocesador, el Intel 4004, 37 transistores equivalían a 1 dólar; actualmente con ese mismo dólar se pueden comprar 2 millones de transistores [1].

Esta reducción del precio y tamaño permite implementar microprocesadores en todo tipo de objetos y dispositivos desde juguetes de control remoto, aspiradoras autónomas, neveras, lavadoras, a cámaras de fotos e incluso cepillos de dientes. La implantación de estos sistemas para la automatización de los edificios deriva en lo que hoy conocemos como domótica. El término domótica viene de la unión de las palabras *domus* (que significa casa latín [2]) y *autónomo* (del griego: que se gobierna a sí mismo [3]).

Un sistema domótico es aquel que automatiza distintas instalaciones en una vivienda o edificio; ejemplos de ello pueden ser la energía, la intensidad de la luz, los sistemas de sonido, sistemas de seguridad y acceso, detección de incendios, mantenimiento, etc.

Gracias al avance de la informática y los sistemas de comunicación, la domótica surge en los años 70 y experimenta un gran desarrollo en los años 80 cuando se introduce a nivel comercial. Su expansión comienza con tecnologías como la Identificación por Radiofrecuencia (RFID), el sistema global para las comunicaciones móviles (GSM) y el Wireless Fidelity (WiFi), entre otras [4].

Finalmente, la expansión se puede resumir en la estandarización, cada vez más extendida gracias al Open Source, y a la mejora de las comunicaciones. La domótica experimenta un

crecimiento exponencial ya que permite un consumo más eficiente y una mayor automatización de grandes edificaciones.

En la actualidad, muchos dispositivos dependen de la energía eléctrica para funcionar, por tanto, un fallo en el abastecimiento eléctrico puede tener graves consecuencias (como se explicará en más detalle en la Sección 1.2). Por ello, en este proyecto se propone un diseño de un dispositivo que avisa y notifica la hora exacta de la pérdida y restauración de suministro.

La implementación se ha realizado con un microprocesador, específicamente una Raspberry Pi ya que está muy estandarizado, y un módulo de telecomunicaciones, en este caso se ha elegido un módulo GSM debido a que esta tecnología está muy extendida. Además, se ha utilizado un Real Time Clock (RTC) y un Uninterrupted Power Supply (UPS).

## **1.2. Motivación**

Este proyecto da solución a un problema muy usual. Al perder el suministro eléctrico, los sistemas de calefacción de las casas dejan de funcionar hasta que dicho suministro se restaura. Cuando se reestablece la electricidad, los termostatos suben la temperatura al máximo. Los dueños de dicha vivienda, si es una segunda residencia, solo se dan cuenta de la situación al recibir una alta factura de la luz.

La situación descrita puede derivar en lo siguiente:

- Si la temperatura de la casa baja de 8 °C el agua en las tuberías puede congelarse ocasionando graves desperfectos en las cañerías.
- Si la casa está sometida a altas temperaturas durante un tiempo prolongado los elementos de madera (muebles, puertas) pueden dilatarse provocando desperfectos.
- La alta factura de la luz supone un elevado gasto económico.

Otros ejemplos de problemas derivados del fallo en el abastecimiento de electricidad de un edificio son:

- La nevera/congelador con comida se queda sin potencia y por tanto la comida se estropea. Esto puede ser muy significativo cuando en segundas viviendas se guarda comida en el congelador o incluso en primeras viviendas si los dueños se van durante poco tiempo y dejan comida en la nevera.
- En el caso de un edificio con servidores, la notificación del fallo del abastecimiento de electricidad puede reducir las consecuencias de perder servicio ya que permite una pronta actuación. También, en la misma línea, en empresas en las que se dejan corriendo simulaciones, ya que si se va la luz aunque sea una duración muy corta, la simulación se detiene hasta que alguien la pone en marcha otra vez manualmente.
- En edificios de cultivo tanto de plantas como de insectos. En instalaciones de este tipo, la iluminación y el calor son claves para la calidad de su producto.

Para resolver las situaciones anteriores, y varias más, se busca un dispositivo simple y de bajo coste, ya que los elementos necesarios son básicos y específicos, combinado con la economía de escala. Uno de los objetivos es que sea barato, pequeño y simple de instalar.

## 1.3. Estado del arte

En este apartado se dará una visión general de los dispositivos con utilidades similares a las que plantea el proyecto.

### 1.3.1. Cellular Alarm

PumAlarm es una empresa con sede en Indianápolis, Estados Unidos. Esta empresa da servicio solo en EEUU y su propuesta es un producto base llamado Cellular Alarm al que le puedes conectar varios sensores vendidos por la misma empresa [5].

El llamado *Cellular Alarm* permite varias funciones con un precio de \$219:

- Power monitoring: envió de un mensaje a la caída y vuelta del suministro eléctrico.
- Baterías de emergencia: el aparato es capaz de autoalimentarse con batería ante la pérdida de alimentación por parte de la red eléctrica.
- Monitorización de la temperatura mediante un sensor incorporado.



**Figura 1.1.** Cellular Alarm. Fuente: [pumpalarm.com](http://pumpalarm.com).

Como se puede apreciar en la Figura 1.1, el sistema es relativamente discreto al ser negro. Es enchufable como se indica en la descripción, pero al distribuirse solo en Estados Unidos, el enchufe no es compatible con el europeo.

### 1.3.2. iSocket 3G

Con base en Londres, esta empresa presenta un dispositivo, que distribuye a cualquier parte del mundo, con un precio de 229 €. El dispositivo manda un mensaje a través de la red 3G cuando el suministro cae y cuando se restaura. Además, dispone de un sensor de temperatura que permite la monitorización de la temperatura y por tanto alertas de temperatura [6].



**Figura 1.2.** iSocket 3G. Fuente: [www.iosocket3g.com](http://www.iosocket3g.com).

Este sistema parece permitir tanto la opción de enchufe europeo como la del británico (Figura 1.2).

### 1.3.3. DD-5241

La empresa *Domo Desk* presenta su “Enchufe GSM/SMS alerta corte de corriente y temperatura”. Este dispositivo tiene un de precio de 65€.

El dispositivo notifica al usuario por SMS en caso de pérdida de corriente o recuperación de la actividad. También, permite enviar datos de la temperatura y notificar al usuario si la temperatura sale del rango especificado por dicho usuario [7]. Por ello, es el que más se adecúa al objetivo de este proyecto.



Figura 1.3. DD-5241:Enchufe GSM/SMS alerta de corte de corriente y temperatura. Fuente: Domo Desk.

Este dispositivo es similar al de la Sección 1.3.2 como podemos ver en la Figura 1.3. Es un dispositivo claramente grande y sigue una estética menos cuidada que el Sección 1.3.1 pero la diferencia de precio es muy notable.

## 1.4. Objetivos

La elaboración e implementación de este proyecto persigue los siguientes objetivos:

1. Detectar el corte y la restauración en el suministro eléctrico.
2. Detectar el momento en el que la temperatura se sale del rango especificado por el usuario.
3. Notificación mediante un SMS los eventos mencionados en los objetivos anteriores.

## 1.5. Estructura del documento

El documento está dividido en cinco capítulos: un primer capítulo de introducción, un capítulo de *hardware*, uno de *software*, uno de resultados y finalmente un capítulo de conclusiones. El capítulo de *hardware* explica uno por uno los componentes utilizado y su función en el dispositivo en conjunto. En el siguiente capítulo, el de *software*, se describen uno a uno los subsistemas y sus funciones. El capítulo de resultados presenta las pruebas que se han realizado y, finalmente, el capítulo de conclusiones presenta un resumen *a posteriori* del proyecto.

# 2

## Hardware

---

Este capítulo describe el dispositivo a nivel *hardware*, es decir, explica la estructura general del sistema y los componentes *hardware* uno por uno, explicando las funciones que desempeñan dentro del equipo.

---

### 2.1. Estructura general del sistema

En este proyecto se van a usar los siguientes elementos de *hardware*:

- **Micro-controlador:** comunica todos los elementos y procesa la información de cada uno, en este caso se ha elegido una Raspberry Pi 3 B.
- **Módulo GSM:** posibilita el envío de SMS con la información necesaria.
- **Real Time Clock:** permite monitorizar la fecha y hora para notificar al usuario.
- **Sensor de temperatura:** permite monitorizar la temperatura para poder notificar cuándo ésta rebasa los límites establecidos por el usuario.
- **USB-mini pinout y regulador de tensión:** permiten convertir la tensión de 220V de la red en una señal de 3,3 V para la detección del suministro eléctrico.
- **Power Bank:** alimenta el dispositivo desde su batería cuando la red eléctrica no está disponible.
- **Router:** permite la conexión a internet del reloj en tiempo real para su sincronización horaria.
- **Shield Click:** gracias a él se pueden conectar varios dispositivos a los pines del microcontrolador.

En la Figura 2.1 se muestra una representación visual del conexionado entre los distintos componentes utilizados en el dispositivo final:

- La Raspberry se conecta a través de sus pines (*GPIOs*) con el *Shield Click* para así permitir la conexión de dos dispositivos a sus pines.

- El reloj en tiempo real y el sensor de temperatura se conectan por el protocolo de comunicación I<sup>2</sup>C y a través de los pines del *Shield Click* a la Raspberry.
- El módulo GSM se conecta a través de los *GPIOs* a la Raspberry y ambos se comunican por el puerto serie mediante los comandos AT.
- El *USB-mini pinout* comunica la red eléctrica a través de un adaptador de 5V a USB. A su vez, su salida está conectada al GPIO 18 de la Raspberry por medio de un conversor de 5 a 3,3 V.
- El *Power Bank* alimenta a la Raspberry a través de un micro-USB y se carga continuamente de la red gracias a un adaptador de 5V a USB.
- La raspberry se conecta con el router a través de WiFi.

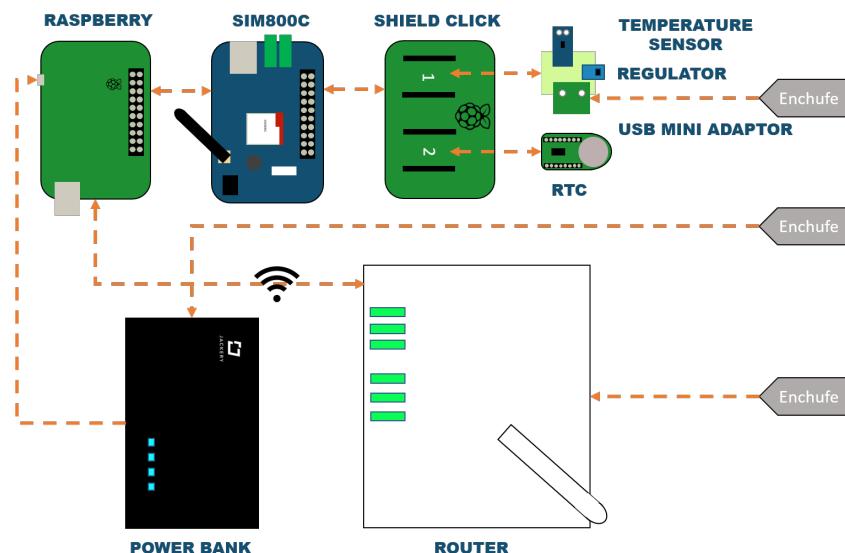


Figura 2.1. Esquema de conexión del hardware. Fuente: Elaboración propia.

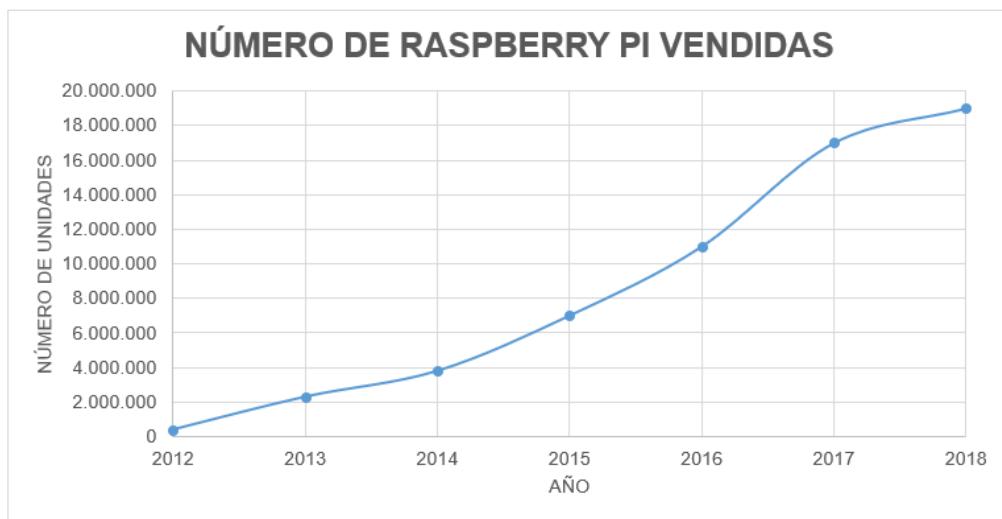
## 2.2. Micro-controlador

Como micro-controlador se utilizará una Raspberry Pi. La *Raspberry Pi Foundation* surge en el Reino Unido con el objetivo de acercar el poder de la programación digital a todo el mundo. Se trata de un microprocesador pequeño y de bajo coste pensado para la educación [8].

### 2.2.1. Historia

La primera generación salió al mercado en 2012 y constaba de una tarjeta de memoria SD estándar como la unidad local y dos puertos USB. Esto favoreció el desarrollo del *Internet of Things* (IoT) y el acercamiento de la electrónica a la comunidad, ya que por alrededor de 25€ aficionados de la electrónica podían tener acceso a un potente pero barato microprocesador [9].

Como se puede observar en la Figura 2.2, la venta de las distintas versiones de este dispositivo ha aumentado rápidamente a lo largo de los años. Desde la primera Raspberry Pi vendida en 2012 hasta el mes de marzo de 2018 el número total de ventas asciende a 19.000.000 unidades [8].



**Figura 2.2.** Número de Raspberry Pi vendidas. Elaboración propia Fuente: página web oficial de Raspberry Pi.

### 2.2.2. Elección del modelo

En la siguiente tabla (Tabla 2.1), se detalla la evolución de las prestaciones y modelos de la Raspberry Pi, en este proyecto se va a utilizar la Raspberry Pi 3 B por ser el más reciente y cumplir con los requisitos.

Como se puede observar en la Tabla 2.1 el modelo Raspberry Pi 3 consta de *wireless LAN* y puertos GPIO. En este proyecto se utilizarán ambos para conectar el micro-controlador con el resto de sensores y actuadores como se muestra en la Figura 2.1.

Los puertos GPIO cuentan con un interfaz I<sup>2</sup>C (Inter-Integrated Circuit) a través del cual se conectará con el reloj y el sensor de temperatura y una interfaz serie con el que se conectará el módulo GSM.

## 2.3. Módulo GSM

Global System of Mobile communications (GSM) es el Sistema de teléfono móvil estándar en Europa. Como se puede observar en la Figura 2.3, el número de subscriptores alcanza actualmente los 5.000 millones, repartidos en más de 100 países [10] [11].

### 2.3.1. Historia

El GSM nace en 1982 fundado por la *Confederation of European Posts and Telecommunications* (CEPT) ante la necesidad de crear una tecnología móvil estándar en Europa, aprobándose los parámetros básicos en 1987. En 1991 se hace la primera llamada GSM en Finlandia y un año más tarde, en 1992, se envía el primer SMS [10].

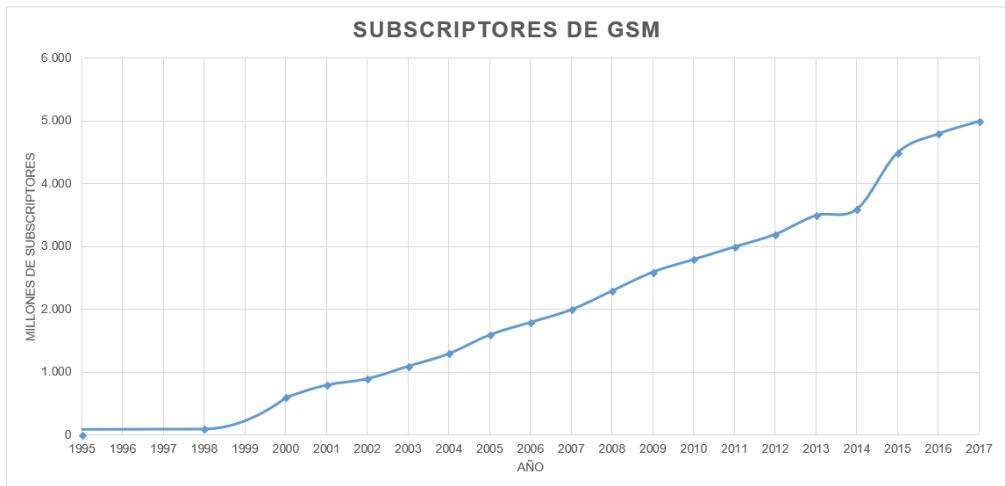
En el informe anual de la *GSM Asociation* (GSMA) se estima en 5.700 millones el número de individuos/objetos con su propia conexión móvil en el año 2020 (con el desarrollo del IoT las comunicaciones máquina/máquina se han extendido) [11].

### 2.3.2. Elección del modelo

En el proyecto, el fallo en el abastecimiento eléctrico interrumpe la conexión WiFi ya que el módem suele estar conectado directamente a la red eléctrica. Por esta razón, se ha utilizado

**Tabla 2.1.** Modelos de Raspberry Pi. Elaboración propia Fuente: página web oficial de Raspberry Pi.

| Fecha | Modelo                  | GPIO | Puertos USB   | Ethernet | CPU  | RAM   | Mejoras  | Precio (2018) |
|-------|-------------------------|------|---------------|----------|--|-------|--|---------------|
| 2012  | Raspberry Pi 1 Model B+ | 40   | 4             | SI       |  |       | Versión final de la primera generación   | 33,50 €       |
| 2014  | Raspberry Pi 1 Model A+ | 40   | 1             | SI       |  |       | Versión más económica de la primera generación                                   | 25,90 €       |
| 2015  | Raspberry Pi 2          | 40   | 4             | SI       | A 900MHz quad-core ARM Cortex-A7 CPU                       | 1GB   | Mejora de la RAM y CPU   | 36,50 €       |
| 2016  | Raspberry Pi 3          | 40   | 4             | SI       | Quad Core 1.2GHz Broadcom BCM2837 64bit                    | 1GB   | BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board, power source 2.5A | 36,90 €       |
| 2015  | Raspberry Pi Zero W     | 0    | 1 (Micro USB) | NO       | 1GHz single-core   | 512MB | Muy económica y funciones básicas  | 5,50 €        |
| 2017  | Raspberry Pi Zero W     | 1    | 0             | NO       | 1GHz single-core   | 512MB | 802.11 b/g/n wireless LAN, Bluetooth 4.1 (BLE)                                   | 11,00 €       |
| 2018  | Raspberry Pi 3B +       | 40   | 4             | SI       | Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit Soc @ 1.4GHz | 1GB   | 2.4GHz and IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE                 | 39,50 €       |



**Figura 2.3.** Millones de subscriptores únicos de GSM. Elaboración propia Fuente: GSMA página web oficial.

la comunicación GSM como alternativa, que como se ha explicado en la sección anterior está estandarizada y muy extendida. Además, la comunicación se hará a través de SMS ya que es la forma más fiable de comunicación — como se explica en más detalle en la Sección 2.3.4.

Así pues, en este trabajo se ha seleccionado el chip SIM800C en un formato adaptado para Raspberry cuyo precio ronda los 15€. Este módulo presenta un set de opciones para la comunicación móvil como: una interfaz UART TTL, un *baud rate* autoajustable desde 2400 hasta 115200. Se puede alimentar con 5V-20V y a través de los GPIOs o de un micro-USB. Tiene una ranura para tarjeta micro SIM card holder y una interfaz de antena IPX. Soporta un rango de temperatura de -45 °C to 80 °C y el interfaz de voz es un micrófono de 3,5 mm. [12]

En este proyecto el módulo GSM se conecta con el micro-controlador mediante la UART, como se puede observar en la Figura 2.5, y es el encargado de transmitir la información que le llega del micro-controlador a través de SMS mediante el set de comandos de Hayes. A continuación se explicarán los diferentes protocolos de comunicación utilizados por el módulo GSM y el set de comandos de Hayes.

### 2.3.3. Comunicación serie

La Raspberry Pi, como la mayoría de micro-controladores, posee *hardware* específico para el protocolo de conexión serie TTL llamado Universal Asynchronous Receiver-Transmitter (UART por sus siglas en inglés). El módulo GSM y la Raspberry se comunican a través de la UART usando el protocolo serie.

Los niveles de voltaje que usan las señales para puerto serie cumplen el estándar RS-232 y usan voltajes tanto positivos como negativos respecto a tierra. Los micro-controladores emplean el mismo protocolo de comunicación pero con niveles lógicos ya que la mayoría de sus pines suelen ser digitales y los voltajes negativos son muy poco convenientes. Este protocolo se denomina TTL Serie y se utiliza cada vez más con dispositivos a 3,3 V en vez de 5 V.

El protocolo TTL Serie se trata de una conexión punto a punto asíncrona, con dos pines de datos: Tx y Rx (*Transmit* y *Receive*) mostrados en la Figura 2.4. Al ser una comunicación asíncrona, se deben definir los baudios (incluyendo los bits de comienzo, final y, si es el caso, de paridad), el número de bits por palabra, el tipo de paridad si la hay y el número de bits de comienzo y final [13].

En este caso, la estructura y parámetros usados para la comunicación entre el módulo GSM y la Raspberry Pi tienen la estructura de los comandos de Hayes explicados en Sección 2.3.5.

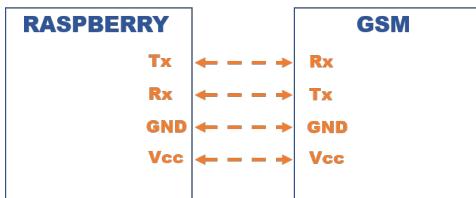


Figura 2.4. Conexión serie de la Raspberry con el módulo GSM. Fuente: Elaboración propia.

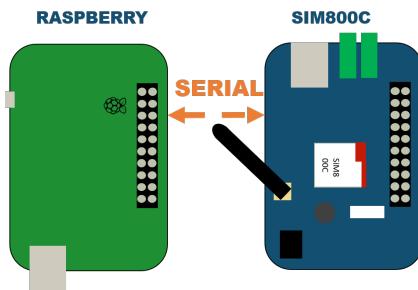


Figura 2.5. Esquema de conexión del módulo GSM con la Raspberry Pi. Fuente: Elaboración propia.

### 2.3.4. Comunicación SMS

Un SMS (*Short Message Service* por sus siglas en inglés), más conocido como mensaje de texto, es un servicio que permite mandar mensajes de texto desde casi cualquier dispositivo móvil. Está dentro de la capa de aplicación MAP (*Mobile Application Part* por sus siglas en inglés), que a su vez forma parte del sistema de señalización SS7 [14] [15]. (Las dependencias están claramente representadas en la Figura 2.6).

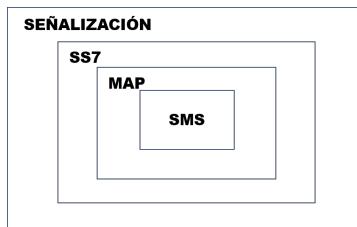


Figura 2.6. Gráfico de las distintas capas de la señalización. Fuente: Elaboración propia.

Al formar parte de un sistema de señalización, los SMS pueden ser mandados y recibidos en cualquier momento y con cobertura limitada. Esto es debido a que gracias a su pequeño tamaño (hasta 160 caracteres en el modo texto) pueden ser incrustados en pequeños paquetes trasladados en intervalos de tiempo reservados exclusivamente para este fin [15].

El SS7 es un sistema de señalización fuera de banda, de canal común y que incorpora tanto señalización de linea como señalización de registro . Ser una señalización fuera de banda implica que se le asigna un canal distinto al usado por las llamadas de teléfono y ser de canal común implica que se emplea un canal que a su vez agrupa varios canales de comunicación [15].

Con todo esto, se puede considerar que el SMS es una forma fiable de comunicar información ya que no comparte canal con las llamadas telefónicas y, de esta manera y al ser un mensaje pequeño, puede ser enviado y recibido con un nivel de cobertura mínimo [16].

### 2.3.5. Comandos de Hayes

Los comandos AT (*attention*) fueron desarrollados por Hayes inicialmente. También se conocen como comandos de Hayes por esta razón. Estos comandos se han extendido y hoy en día la mayoría de fabricantes de teléfonos como Motorola, LG, Samsung, Sony-Ericsson y Nokia los utilizan, haciendo que los programas puedan ser reutilizados en un amplio rango de dispositivos (estos comandos pueden variar levemente según el fabricante).

Estos comandos se han expandido para llevar a cabo tareas específicas a través de la interfaz de comunicación serie. Actualmente, son una serie de cortos *strings* de texto combinables para producir comandos que indican al módem las acciones de marcar, mandar SMS, usar el *General Packet Radio Service* (GPRS), mandar un fax, controlar el volumen, cambiar los parámetros de la conexión, comprobar el estado de la batería, etc.

Los comandos AT se caracterizan por empezar con las letras 'A', 'T' seguidos de *strings* específicos de cada operación y terminados con retorno de carro. Al finalizar cada comando, por el mismo protocolo serie, el módem manda una respuesta con los parámetros solicitados (si es el caso) o un 'O' 'K' si se ha realizado la acción correctamente [17].

En este trabajo, y siguiendo la *datasheet* de la familia de chips SIM800 se usarán los siguientes comandos:

- **AT**: permite "despertar" al módulo.
- **AT + CMGF**: selecciona el formato del mensaje, que puede ser o *Protocol Data Unit* (PDU) o modo texto siguiendo el formato GSM estándar. En este caso, se ha seleccionado el modo texto ya que los mensajes a enviar son cortos y no es necesario comprimirlos.
- **AT + CSCS**: ajusta el set de caracteres. En este caso se iguala al "GSM" para seleccionar el alfabeto GSM de 7 bit por defecto (3GPP TS 23.038).
- **AT + CMGS**: permite mandar un SMS.
- **AT + CPOWD**: permite reiniciar el módulo.

## 2.4. Real Time Clock (RTC)

Un RTC se emplea para poder llevar un registro del tiempo en unidades humanas. En casi todos los dispositivos electrónicos se pueden encontrar *hardware clocks*, señales que controlan electrónica digital, pero éstos no contabilizan en días, horas, minutos, etc.

En este caso, el RTC elegido funciona con un oscilador de cristal. Se trata de un oscilador electrónico que se vale de la resonancia de un cristal de material piezoelectrónico para crear una señal electrónica de una frecuencia determinada [18].

El modelo DS1307 fabricado por *Maxim Integrated* consta de calendario y reloj en decimal codificado en binario (BCD). Es un dispositivo de bajo consumo y con una salida de onda cuadrada programable. Cumple con los requerimientos del proyecto ya que se trata de un dispositivo de bajo coste y compatible con la Raspberry Pi. Además, mantiene segundos, minutos, horas, mes y año y funciona en un rango de -40°C a 85°C, perfecto para todo tipo de edificios. Cuenta con una batería externa de 3 voltios que le permite seguir funcionando cuando la fuente de 5V no está disponible. Se puede encontrar un resumen de las especificaciones en la Tabla 2.2.

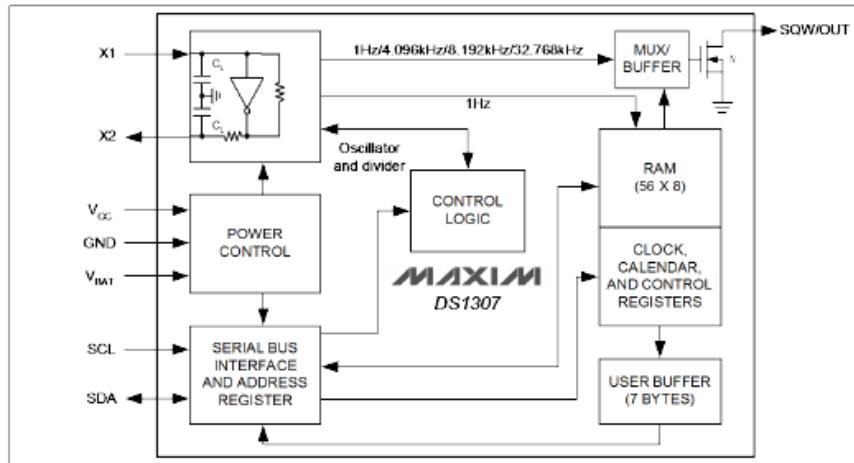


Figura 2.7. Diagrama de bloques del RTC. Fuente: Datasheet del fabricante.

El precio del DS1307 es de alrededor de 2,65€. El sensor que se usará en este proyecto es la placa RTC2 click ya que estaba disponible en el laboratorio pero este sensor contiene el mismo microchip DS1307.

La comunicación del RTC con el micro-controlador es mediante la comunicación I<sup>2</sup>C (como se puede observar en la Figura 2.7 y en la Figura 2.1). El módulo se encarga de mantener la hora exacta aún cuando el micro-controlador pierde la conexión a Internet. La información sobre la hora y fecha se almacena en siete registros a los que el micro-controlador accede para luego poder comunicárselos al usuario en caso de pérdida o restablecimiento de suministro eléctrico por SMS.

Tabla 2.2. Especificaciones del RTC. Fuente: el distribuidor ([www.mikroe.com](http://www.mikroe.com)).

|                  |  |
|------------------|--|
| Type             | RTC  |
| Applications     | Board is ideal for applications which require real-time clocks, calendars and programmed alarms.   |
| On-board modules | DS1307 serial real-time clock (RTC)  |
| Key Features     | Counts Seconds, Minutes, Hours, Date of the Month, Month, Day of the week, and Year with Leap-Year Clock function with four year calendar. |
| Key Benefits     | 3V/230mA lithium battery as a backup power supply  |
| Interface        | I2C,GPIO   |
| Input Voltage    | 5V   |
| Compatibility    | mikroBUS   |
| Click board size | M (42.9 x 25.4 mm)   |

#### 2.4.1. Protocolo I<sup>2</sup>C

El protocolo de comunicación *Inter-Integrated Circuit* (I<sup>2</sup>C) o también llamado *Two-Wire Interphase* (TWI) se vale de dos cables para la transmisión de datos. Es un bus al que se pueden conectar dispositivos a los mismos dos cables. Los dispositivos I<sup>2</sup>C pueden ser tanto *masters* o *slaves* y puede haber más de un *masters* por bus.

Los dos cables son: el *serial clock line* (SCL) y el *serial data line* (SDA). El SCL mantiene la sincronización del bus con el reloj y el SDA transmite los datos de forma bidireccional. Como se puede observar en la Figura 2.8, también es necesario un tercer cable a la misma tierra.

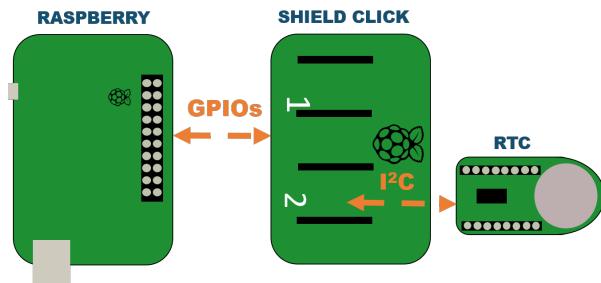


Figura 2.8. Esquema de conexión del RTC con la Raspberry Pi. Fuente: Elaboración propia.

Cada *slave* del sistema tiene asignado una dirección única dentro del bus. Esta dirección es la que usa el *master* con el que quiere interactuar, ya que es el *master* el que tiene el control y el que decide cuándo leer o escribir en los registros de los *slaves* del bus [13].

En este proyecto el dispositivo *master* es el micro-controlador y el *slave* es el RTC. La dirección del RTC dentro del bus es: '1101000'. Y la dirección de los registros para el reloj y el calendario se encuentran en la Figura 2.9. El bit 6 del registro 02h permite elegir entre el modo de 24 horas o el de 12 horas. En este caso se ha elegido el modo de 12 horas, por tanto dicho bit se inicializará a 1, el bit 5 indicará el período del día (PM/AM) y el 4 las decenas. (La imagen incluida en el datasheet contradice al texto incluido en la misma, por tanto, se ha modificado la imagen para ajustarse a la realidad).

| ADDRESS | BIT 7 | BIT 6   | BIT 5      | BIT 4    | BIT 3 | BIT 2   | BIT 1 | BIT 0 | FUNCTION   | RANGE                   |
|---------|-------|---------|------------|----------|-------|---------|-------|-------|------------|-------------------------|
| 00h     | CH    |         | 10 Seconds |          |       | Seconds |       |       | Seconds    | 00–59                   |
| 01h     | 0     |         | 10 Minutes |          |       | Minutes |       |       | Minutes    | 00–59                   |
| 02h     | 0     | 24      | 10 Hour    | 10 Hour  |       | Hours   |       |       | Hours      | 1–12<br>+AM/PM<br>00–23 |
|         |       | 12      | PM/AM      |          |       |         |       |       |            |                         |
|         |       | 0       | 0          | 0        | 0     |         | DAY   |       | Day        | 01–07                   |
| 03h     | 0     | 0       | 0          | 0        | 0     | Date    |       |       | Date       | 01–31                   |
| 04h     | 0     | 0       | 10 Date    |          |       |         | Date  |       |            |                         |
| 05h     | 0     | 0       | 0          | 10 Month |       | Month   |       |       | Month      | 01–12                   |
| 06h     |       | 10 Year |            |          |       |         | Year  |       | Year       | 00–99                   |
| 07h     | OUT   | 0       | 0          | SQWE     | 0     | 0       | RS1   | RS0   | Control    | —                       |
| 08h–3Fh |       |         |            |          |       |         |       |       | RAM 56 x 8 | 00h–FFh                 |

Figura 2.9. Registros del dispositivo DS1307. Fuente: Datasheet del fabricante (modificada).

## 2.5. Sensor de Temperatura

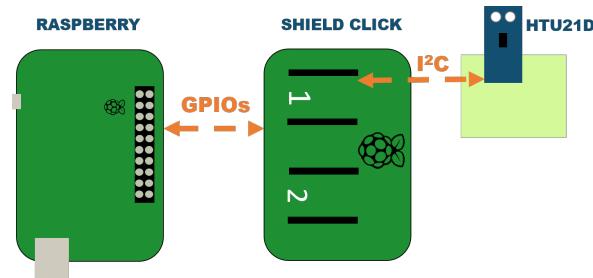
Para poder medir la temperatura es necesario un sensor, que en este caso, se comunicará con el micro-controlador por I<sup>2</sup>C aprovechando que el reloj en tiempo real también usa este tipo de comunicación.

### 2.5.1. HDC 1080

Se ha elegido el sensor HDC 1080, que también incorpora un sensor de humedad aunque para los objetivos de este trabajo no es necesario. Este sensor, fabricado por **Texas Instruments**, es un dispositivo de bajo consumo y bajo coste (su precio ronda los 9 €).

El sensor tiene una precisión de  $\pm 2\%$  para la humedad relativa y  $\pm 2^{\circ}\text{C}$  para la temperatura. Mide con una resolución de 14 bits y como se ha mencionado es compatible con la interfaz I<sup>2</sup>C de la Raspberry.

El sensor HDC se conecta a la Raspberry mediante la comunicación I<sup>2</sup>C a través del Shield Click como se puede observar en la Figura 2.10.



**Figura 2.10.** Esquema de conexión del sensor de temperatura con la Raspberry Pi. Fuente: Elaboración propia.

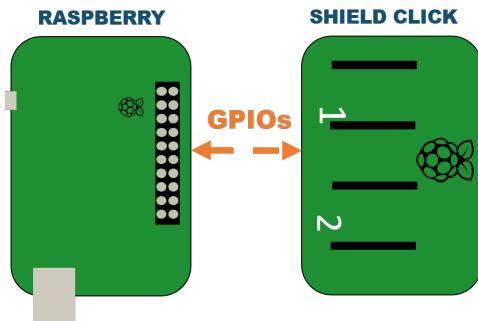
La dirección del sensor dentro del bus es ‘1000000’ y la dirección de los registros de temperatura y humedad, como se muestra en la Figura 2.11, son el 0x00 y el 0x01 respectivamente. El registro de configuración permite la adquisición de ambos datos con un solo comando poniendo el bit 12, correspondiente a *mode* en 1. Así pues, en el registro de configuración se elegirá tanto esta opción como la máxima resolución de ambas medidas (14 bits).

| Pointer | Name            | Reset value      | Description                                |
|---------|-----------------|------------------|--|
| 0x00    | Temperature     | 0x0000           | Temperature measurement output             |
| 0x01    | Humidity        | 0x0000           | Relative Humidity measurement output       |
| 0x02    | Configuration   | 0x1000           | HDC1080 configuration and status           |
| 0xFB    | Serial ID       | device dependent | First 2 bytes of the serial ID of the part |
| 0xFC    | Serial ID       | device dependent | Mid 2 bytes of the serial ID of the part   |
| 0xFD    | Serial ID       | device dependent | Last byte bit of the serial ID of the part |
| 0xFE    | Manufacturer ID | 0x5449           | ID of Texas Instruments                    |
| 0xFF    | Device ID       | 0x1050           | ID of the device                           |

**Figura 2.11.** Registros del dispositivo HDC 1080. Fuente: Datasheet del fabricante.

## 2.6. Pi 3 click Shield

El microcontrolador elegido, la Raspberry Pi solo tiene un set de GPIOs disponible. En este proyecto se van a usar tres sensores, dos de ellos conectados por I<sup>2</sup>C ( Sensor de temperatura y RTC) y uno conectado por serie ( GSM). Para poder hacer uso de todos los sensores se utiliza un *shield* que extiende los GPIOs a dos mikroBUS<sup>TM</sup> compatibles con los dispositivos click. Por ello, en un set se puede conectar el reloj en tiempo real y en el otro el sensor de temperatura por al bus I<sup>2</sup>C y el sistema de detección de la caída de corriente al GPIO disponible (como se puede apreciar en la Figura 2.12 y en la Figura 2.1) [19].



**Figura 2.12.** Esquema de conexión del *Shield* con la Raspberry Pi. Fuente: Elaboración propia.

El *shield* elegido, que tiene un precio de aproximadamente 12 € y las especificaciones se encuentran resumidas en la Tabla 2.3.

**Tabla 2.3.** Especificaciones del *Shield*. Fuente: el distribuidor ([www.mikroe.com](http://www.mikroe.com)).

| Type             | Shield  |
|------------------|---|
| Applications     | Pi 3 click shield allows you to use click boards on your Raspberry Pi®                            |
| On-board modules | Two mikroBUS™ sockets, MCP3204 12-bit Analog-to-Digital converter (ADC), 4.096V reference voltage |
| Key Features     | GND Oscilloscope probe pin. Four mounting holes.  |
| Input Voltage    | 3.3V or 5V  |
| Compatibility    | Raspberry Pi  |

## 2.7. Uninterrupted Power Supply (UPS)

EL UPS es un dispositivo que alimenta al micro-controlador con una batería externa en el caso de que falle el abastecimiento eléctrico. Esta prestación es clave ya que el sistema ha de seguir funcionando sin electricidad de la red para alertar del corte en el suministro.

### 2.7.1. UPS HAT v1.2 Raspberry Pi

El *HAT* v1.2 para la Raspberry es un UPS diseñado específicamente para este dispositivo. Según el fabricante tiene las siguientes características:

- Sus dimensiones son las estándar de la Raspberry Pi y, por tanto, es compatible con todos los dispositivos de extensión. (Sensores, actuadores, etc).
- Su diseño en cascada permite ahorrar espacio ya que alimenta la Raspberry y el resto de dispositivos enchufados a ella sin afectar a los GPIOs.
- Soporta 2A y con ello la tecnología *quick-charge*.
- Puede cargar la Raspberry mientras carga su propia batería.
- Cuenta con 4 LED para indicar su estado de carga [20].

Las especificaciones anteriores hacen de este un dispositivo perfecto para lidiar con la necesidad de poder seguir utilizando la Raspberry sin energía de la red eléctrica, pero en la

práctica se a demostrado que esto no es así. El dispositivo tiene un ligero tiempo de retardo que provoca el reinicio de la Raspberry entre la pérdida de energía de la red y la alimentación desde la batería. Esto hace que el UPS HAT v1.2 no cumpla con las especificaciones necesarias para este proyecto.

### 2.7.2. Jackery Titan

Ante el fallo del dispositivo de la Sección 2.7.1 se ha utilizado como alternativa una batería externa con suficiente amperaje como para alimentar todo el sistema (alrededor de 2A) y que, además, se pudiera recargar mientras alimentaba al sistema.

La batería externa de la marca **Jackery** y modelo **Titan** cumple estas especificaciones con un precio de alrededor de (40 €). Así pues, es capaz de alimentar a la Raspberry (a través del micro-USB de la Raspberry, Figura 2.13) y sus periféricos durante el “apagón” y, cuando vuelve a haber corriente eléctrica de la red, volver a recuperar la carga perdida mientras sigue alimentando al dispositivo.

Las especificaciones según el fabricante son:

- Carga eléctrica de 20100 mAh.
- Corriente máxima de salida de 3,4 A.
- Compatible con carga rápida.
- Protección ante cortocircuitos y sobre-tensiones.
- Protección ante sobre corrientes de salida [21].

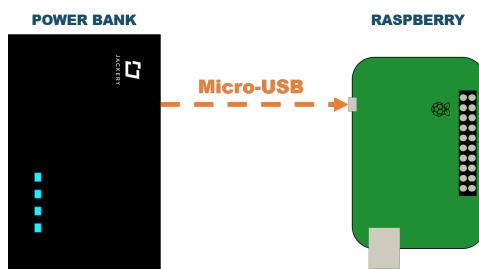


Figura 2.13. Esquema de conexión del *Power Bank* con la *Raspberry Pi*. Fuente: Elaboración propia.

## 2.8. Detección de la caída de la red eléctrica

La detección de corriente es clave para el objetivo de este proyecto y por tanto se han estudiado varias opciones.

Primero se valoró el detectarla a través del router, ya que este dejaría de funcionar durante el periodo en el que la electricidad de la red eléctrica no está disponible. Si se utiliza el router para la detección se pueden usar tanto el protocolo TCP (Transmission Control Protocol) como la comunicación SSH (Secure SHell) como formas inalámbricas y por Ethernet mediante este cable.

Ésta última forma se exploró profundamente, como se explica en la Sección 2.8.1, pero al no dar los resultados deseados, se exploraron opciones de *hardware*. De esta forma se consideró el uso de un relé de 220V y, finalmente, se decidió hacer uso de un dispositivo llamado “adaptador de USB a Dual in-line package (DIP)” combinado con un regulador de 5 a 3,3 V como se explica en la Sección 2.8.2 y Sección 2.8.3 respectivamente.

### 2.8.1. Ethtool

Ethtool es una herramienta de Linux/Unix que permite visualizar y modificar algunos parámetros de la tarjeta de red (NICs por sus siglas en inglés) de dicho dispositivo. Fue creada por David S. Miller, desarrollada por Ben Hutchings y publicada en 1998. Ya que el sistema operativo de la Raspberry está basado en Linux, esta herramienta está disponible aunque no viene preinstalada. [22] [23]

Una vez instalada, a través del comando `sudo ethtool eth0` la Raspberry analiza la conexión Ethernet y devuelve un informe que termina con: "Link detected: yes/no". Para este proyecto, esa línea es clave ya que en caso de responder sí — yes — significaría que la alimentación a través de la red eléctrica está disponible (el router estaría encendido), pero aunque Matlab sí que soporta los comandos de sistema de la Raspberry — `system(rpi, 'ethtool eth0')`, por ejemplo — Simulink no los soporta aún dentro de una función de Matlab para Simulink [24]. Por esta razón, esta vía de detección tuvo que ser descartada.

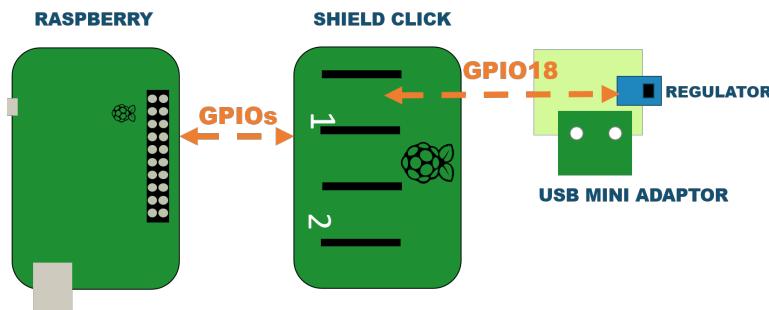
### 2.8.2. Adaptador de USB a DIP

Al no ser posible el uso de la herramienta Ethtool (Sección 2.8.1) se investigaron otras opciones que involucraban elementos *hardware*, finalmente llegando a la elección final: el adaptador de USB a DIP.

Este adaptador, comercializado por *Banggood* a un precio de alrededor de 1,5 € permite separar las salidas del micro-USB en pines [25]. Estas salidas son:

- Una señal de referencia de 5 V, **Vbus**.
- Una señal de referencia a tierra, **GND**
- Dos señales de datos.

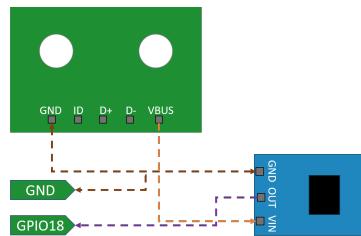
En el caso de una pérdida de alimentación en el cable conectado, la señal de 5 V cae a 0, si a su vez el cable está conectado a la red a través de un regulador de 5 V (cable estándar comúnmente empleado para cargar dispositivos electrónicos como móviles, tablets, etc) esta caída a 0 V indica la caída de la red eléctrica.



**Figura 2.14.** Esquema de conexión del adaptador de USB a DIP y del regulador con la Raspberry Pi. Fuente: Elaboración propia.

### 2.8.3. Conversor de 5 a 3,3 V

Como se explica en la sección anterior (Sección 2.8.2) la salida del adaptador de USB a DIP es de 5 V, pero los GPIOs de la Raspberry sólo soportan 3,3 V como máximo. Por esta razón, es necesario usar un conversor DC-DC para convertir los 5 V en 3,3.



**Figura 2.15.** Conexiones entre el Regulador, el adaptador USB a DIP y la Raspberry. Fuente: Elaboración propia.

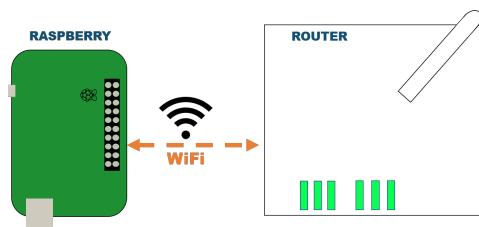
Se ha seleccionado el regulador *5V To 3.3V DC-DC AMS1117 800MA Step-Down Power Supply Buck Module* del distribuidor *Banggood* con un precio de 2 € por 5 piezas. Este regulador tiene una entrada de 4,2 a 10 V y una salida de 3,3 V y 800 mA, además, en el caso de que la entrada sea menor de 1,5 V su salida será 0 V [26].

Así pues, el conexiónado entre el regulador y el USB a DIP (Figura 2.15 es el siguiente:

- La entrada del regulador se conecta a la señal Vbus del USB DIP.
- La salida del regulador se conecta al GPIO 18 de la Raspberry.
- Tanto la referencia de tierra del regulador como la del USB DIP se conectan a la referencia a tierra de la Raspberry.

## 2.9. Router

En este proyecto es necesario que la Raspberry esté conectada a Internet por lo menos una vez para poder sincronizar correctamente la fecha y hora del reloj en tiempo real. Para que el router pueda conectarse a Internet a través de la red de la universidad ha de ser un router sin marca. En este caso la conexión con la Raspberry será por WiFi (como se muestra en la Figura 2.16) pero podría ser también por Ethernet. El modelo del router es *AirStation WHR-G125 Cable/DSL Router* de la marca *Buffalo*.



**Figura 2.16.** Esquema de conexión del router con la Raspberry Pi. Fuente: Elaboración propia.

# 3

## Software

---

Este capítulo describe el dispositivo a nivel *software*, es decir, la estructura general del sistema y una descripción de los subsistemas uno por uno explicando sus funciones.

---

### 3.1. Estructura general y buses de datos

La estructura del *software*, implementado con SIMULINK® en este proyecto, está dividida en dos grandes bloques, el bloque de *hardware* y el bloque de control. (En la Figura 3.1 se puede encontrar un esquema detallado de esta estructura).

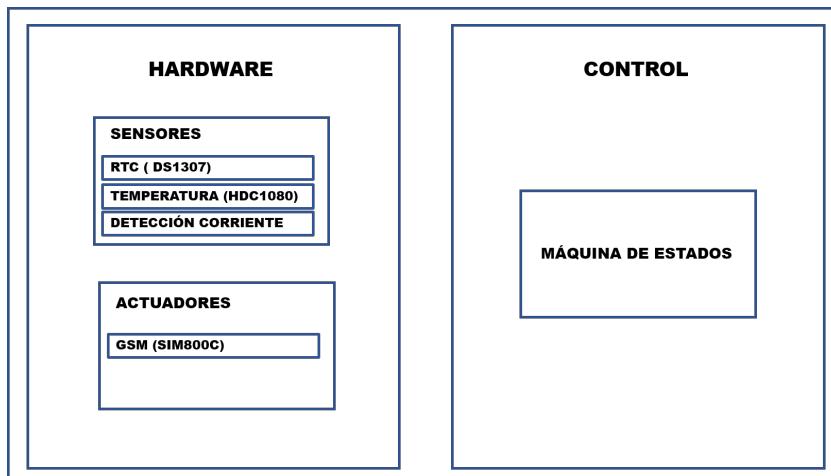


Figura 3.1. Estructura del software. Fuente: elaboración propia.

Dentro del bloque de *hardware* se encuentran dos sub-bloques:

- **Actuadores:** que en este caso solo contiene el módulo GSM (Sección 3.2).
- **Sensores:** que contiene la detección de la caída de corriente, el reloj en tiempo real y el sensor de temperatura (Sección 3.3).

El bloque de control tiene a su vez:

- **Máquina de estados:** controla el comportamiento de alto nivel del *software* (Sección 3.4).

Como medio de comunicación entre bloques se empleará un solo bus, el CONTROL\_BUS, que contiene a su vez los distintos buses y posiciones mostrados en la Figura 3.2. La labor de estas posiciones será explicada a lo largo de este capítulo en las secciones correspondientes.

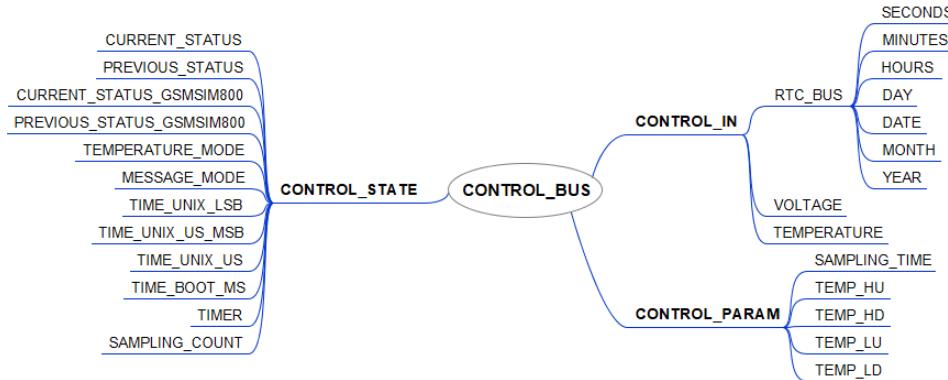


Figura 3.2. Esquema de buses del sistema. Fuente: elaboración propia.

## 3.2. Módulo GSM

Como se explica en la Sección 2.3, el micro-controlador manda los comandos al módulo GSM a través de la comunicación serie. Estos comandos tienen un orden determinado como se menciona en la Sección 2.3.5, a continuación se explicara cómo son mandados estos comandos.

En la Figura 3.4 se muestra el diagrama de flujo del bloque GSM. Este bloque es activado por la máquina de estados general cuando es necesario mandar el mensaje (Sección 3.4) y su funcionamiento es a su vez controlado por su propia máquina de estados.

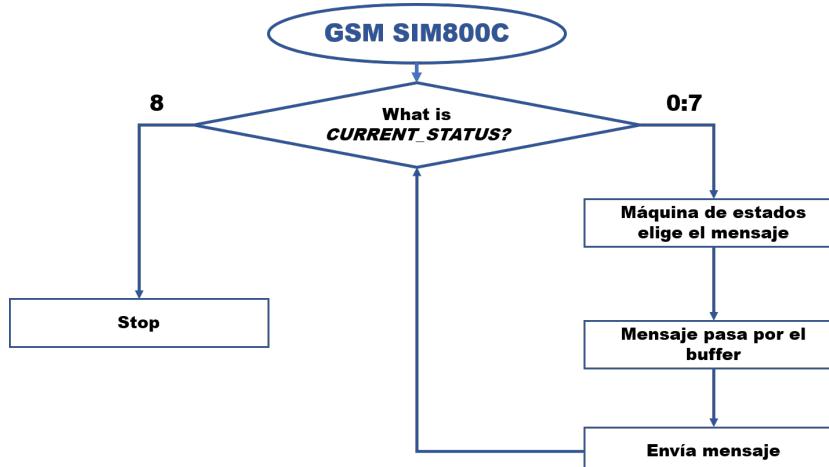
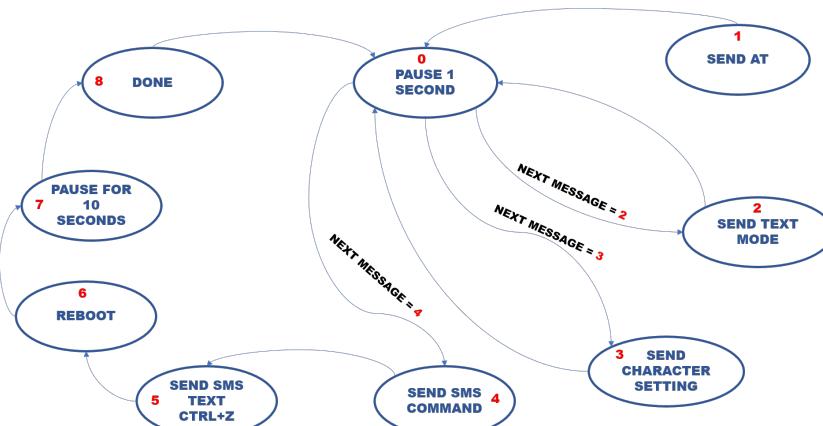


Figura 3.3. Diagrama de flujo del *software* del sensor de temperatura. Fuente: elaboración propia.

Así, lo primero que hace este bloque es comprobar el estado de su máquina de estados, entra dentro de éste código y se elige el mensaje que se ha de mandar. Estos mensajes se han de comunicar en lenguaje ASCII (*American Standard Code for Information Interchange*). Acto seguido, el mensaje pasa por una función que se encarga de crear el *buffer*.

La función de *buffer* es necesaria ya que el bloque de comunicación serie de Simulink para la Raspberry solo admite la transmisión de mensajes del mismo tamaño. Por esta razón, se deben

añadir ceros detrás de todos los comandos con una excepción, cuando el módulo se encuentra en el interfaz de escritura del SMS. Cuando el GSM recibe el comando de mandar el SMS, lo próximo que espera es el texto que ha de mandar. Por esta razón, entre el comando y el texto no puede haber ningún otro carácter ya que se tomaría como ASCII, por ejemplo en el caso del cero se recibiría una @. Lo mismo ocurre si se insertan caracteres no deseados entre el texto y el comando de enviar (CTRL+Z). Como solución, el texto del mensaje va seguido del número 26, que en ASCII representa el CTRL+Z y de tantos ceros como sea necesarios para llenar el *buffer*.



**Figura 3.4.** Máquina de estados del módulo GSM. Fuente: elaboración propia.

En la Figura 3.4 se pueden observar los distintos estados de la máquina de estados del módulo GSM. Cada estado se corresponde con un comando, excepto en el caso de las pausas y del estado *done* cuya función es indicar que el proceso de mandado del SMS ha finalizado.

Entre algunos de los comandos es necesario dejar un segundo de espera para asegurarse de que el módulo ha terminado de procesarlo antes de que llegue el siguiente. Por ello, después de los comandos **AT**, **AT + CMGF** y **AT + CSCS** se pasa directamente al estado de pausa 1 segundo. En este estado, ya que en Simulink se manda continuamente información por el bloque serie, se mandan ceros. Del mismo modo, después de reiniciar hay una espera de 10 segundos que permite asegurarse de que el módulo ha terminado este proceso que según el fabricante dura 5 segundos.

El proceso de reinicio se lleva a cabo para vaciar el *buffer* del puerto serie del módulo, ya que cuando ha enviado alrededor de 5 mensajes, empieza a mandar caracteres extraños. Así pues, el reinicio permite que cada vez que se termina el proceso de mandar el SMS el módulo limpie su *buffer*.

Por tanto, los estados de la máquina de estados del módulo GSM tienen las siguientes funciones:

- **Pause 1 second:** manda ceros a través del puerto serie para dar tiempo al módulo a procesar el comando recibido. Además, es el estado de reposo del módulo.
- **Send AT:** manda el comando **AT** que permite despertar al módulo y prepararlo para recibir otros comandos.
- **Send text mode:** manda el comando **AT + CMGF = 1** seleccionando así el modo texto para el SMS.

- **Send character setting:** manda el comando **AT + CSCS = "GSM"** que permite seleccionar el alfabeto GSM de 7 bit soportado por la Raspberry y el código ASCII.
- **Send SMS command:** manda el comando **AT + CMGS = -34numerodetelefono"** y comienza el modo texto del SMS.
- **Send SMS text CTRL+Z:** manda el texto del mensaje, que es seleccionado en la máquina de estados general según las condiciones que se quieren transmitir y que se elige en la máquina de estados del GSM con la variable MESSAGE\_MODE del bus de control. En el texto del mensaje, como se ha explicado antes, se incluye el carácter 26 y tantos ceros como sean necesarios para llenar el *buffer*.
- **Reboot:** manda el comando **AT + CPOWD = 1** que reinicia el módulo de forma segura limpiando su *buffer*.
- **Pause for 10 seconds:** envía ceros por el puerto serie durante 10 segundos para dejar tiempo a que el módulo se reinicie.
- **Done:** manda la señal a la máquina de estados general de que el módulo se ha reiniciado. Lo hace a través de la variable CURRENT\_STATE del SIM800C.

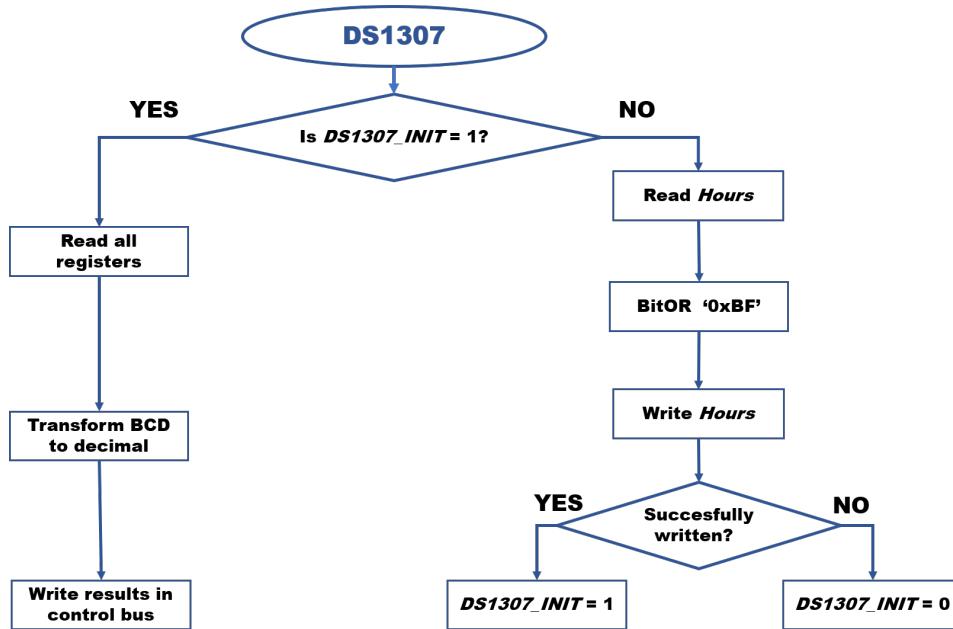
Ya que la máquina de estados esta estructurada con un *switch*, cuya variable de comparación es el estado actual, cuando se pasa por el estado *Pause 1 second* se necesita una segunda variable para saber qué comando se ha de mandar después.

## 3.3. Sensores

### 3.3.1. Real Time Clock

Como se explica en la Sección 2.4, el micro-controlador obtiene los datos sobre el momento exacto del evento (ya sea de caída o de restablecimiento) del RTC por el protocolo I<sup>2</sup>C. Para ello primero ha de inicializarse el componente para que el formato de la hora concuerde con el esperado, así, el bloque de Simulink creado lleva a cabo los procesos que se explican a continuación y que se pueden encontrar en el diagrama de flujo de la Figura 3.5.

1. Se lee el registro de la hora.
2. Se escribe en el registro de la hora la hora leída en el paso anterior pero con el bit 6 a cero y así seleccionar el modo de 12 horas.
3. Una vez inicializado no se volverán a realizar los dos pasos anteriores a no ser que haya un reinicio inesperado. Esto es posible ya que las acciones anteriores se llevan a cabo si el registro llamado **DS1307\_INIT** contiene un 0, y al terminar estas acciones correctamente, se pone el registro a 1 para que solo se lleven a cabo los procesos que se describen a continuación.
4. Se leen los registros que contienen la hora y fecha en orden tal como indica la datasheet: segundos, minutos, horas, día de la semana (del 1 al 7), día, mes y año.
5. La información en bruto entra en la función **READ\_TIME** donde se procesa.
6. Se carga la información procesada en el bus de control en la parte de RTC en las posiciones designadas para cada dato. (Figura 3.2)



**Figura 3.5.** Diagrama de flujo del *software* del RTC. Fuente: elaboración propia.

La Función READ TIME, como se ha mencionado anteriormente, tiene como entrada la salida del bloque de lectura I<sup>2</sup>C, es decir, un vector de uint8. Siguiendo la información de la data sheet, se extraen los bytes de cada registro — se leen 7 registros de golpe — y se dividen en dos grupos de 4 bits. Esto se debe a que los registros están codificados en BCD.

Así pues, para poder tener las decenas y las unidades por separado se hace un *bitand* seguido de un *bitshift*, la primera genera una máscara de AND con unos para hacer cero los cuatro primeros bits (o los 4 bits más significativos según proceda) y la segunda desplaza los cuatro bits más significativos cuatro posiciones a la derecha en el caso de la extracción de las decenas. De esta manera, los bits que no interesan se convierten en 0 y seguidamente se mueven a las posiciones más a la derecha para poder multiplicar (por 10 si son decenas). Acto seguido se suman ambas cantidades y gracias al comando *typecast* se genera un uint8 con la representación decimal de cada número.

En el caso de las horas, este proceso difiere ligeramente del anteriormente explicado ya que se reciben 8 bits de los cuales los 2 más significativos indican no indican decenas, si no que el bit 7 siempre es 0 y el 6, en este caso, también — modo 24 horas — Así, se extraen las unidades como se ha explicado en el párrafo anterior y las decenas del mismo modo pero poniendo a 0 los bits 6 y 7 (aunque ya son 0 siempre).

Para poder sincronizar el RTC correctamente extrayendo la fecha de Internet, se han seguido los siguientes pasos directamente en el sistema operativo de las Raspberry (Raspian), estos pasos basta con hacerlos una vez [27]. Para ello se empleó una pantalla y un teclado.

- (sudo apt-get update)
- (sudo apt-get -y upgrade)
- Dentro del archivo modules (sudo nano /etc/modules) hay que añadir *rtc-ds1307* en la última línea.
- Se crea el archivo *rtc* con el comando *sudo nano /etc/rc.local* y en el se incluyen la siguientes líneas:

```
echo ds1307 0x68 >/sys/class/i2c-adapter i2c-1/new_device
hwclock -s
```

### 3.3.2. Sensor de temperatura

Como se explica en la Sección 2.5.1, el micro-controlador obtiene los datos sobre la temperatura por el protocolo I<sup>2</sup>C. El bloque de Simulink creado lleva a cabo los procesos que se explican a continuación y que se pueden encontrar en el diagrama de flujo de la Figura 3.6.

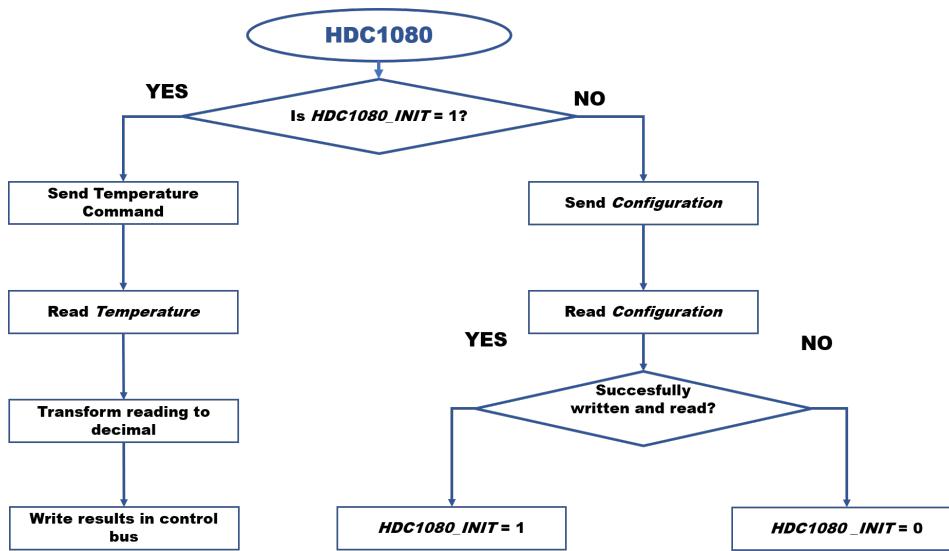


Figura 3.6. Diagrama de flujo del software del sensor de temperatura. Fuente: elaboración propia.

1. Se escribe el byte de configuración en el registro correspondiente.
2. Se lee el registro de configuración.
3. Si tanto la escritura como la lectura se realizan con éxito, el registro llamado **HDC1080\_INIT**, que contiene un 0 en el inicio, se pone a 1 para que solo se lleven a cabo los procesos que se describen a continuación. De no ser así se repetirán los pasos 1 y 2 hasta que se completen correctamente.
4. Se manda el comando de temperatura, que en el caso de simulink para la raspberry se hace enviando la dirección del registro al sensor y desmarcando la opción “enable register access” en el bloque de I<sup>2</sup>C.
5. Una vez se ha comprobado que dicho comando ha sido enviado correctamente el dispositivo espera a recibir los datos. Esto se hace a través de un *enable* de Simulink, ya que si se envía la orden de temperatura constantemente nunca llega el resultado porque el dispositivo pasa automáticamente al proceso de medida. Así pues, una vez se ha comprobado que se ha leído correctamente los datos, (temperatura y humedad aunque el de humedad se descarte) se pasa la información a la función *READ\_TEMPERATURE\_HUMIDITY*.
6. La información en bruto se procesa en la función *READ\_TEMPERATURE\_HUMIDITY*.
7. La información procesada se carga en el bus de control en el espacio TEMPERATURE. (Figura 3.2).

| NAME | Bits  | DESCRIPTION                        |   |
|------|-------|------------------------------------|---|
| RST  | [15]  | Software reset bit                 | 0 Normal Operation, this bit self clears<br>1 Software Reset  |
|      |       | Reserved                           | 0 Reserved, must be 0<br>1 Heater Enabled   |
| HEAT | [13]  | Heater                             | 0 Heater Disabled<br>1 Heater Enabled   |
|      |       | Mode of acquisition                | 0 Temperature or Humidity is acquired.<br>1 Temperature and Humidity are acquired in sequence, Temperature first. |
| BTST | [11]  | Battery Status                     | 0 Battery voltage > 2.8V (read only)<br>1 Battery voltage < 2.8V (read only)                                      |
|      |       | Temperature Measurement Resolution | 0 14 bit<br>1 11 bit  |
| HRES | [9:8] | Humidity Measurement Resolution    | 00 14 bit<br>01 11 bit<br>10 8 bit  |
|      |       | Reserved                           | 0 Reserved, must be 0   |
|      |       | Reserved                           | 0 Reserved, must be 0   |

Figura 3.7. Párametros del registro de configuración del HDC1080. Fuente: datasheet del fabricante.

El byte de configuración que se ha de escribir en el paso 1 concuerda con la información sobre el registro de configuración mostrado en la Figura 3.7. Se escogen las siguientes opciones:

- Software reset bit [15]: normal operation (0).
- Heater [13]: desabilitado (0).
- Mode of adquisition [12]: temperatura y humedad concatenados con la temperatura primero (1).
- Temperature measurement resolution [10]: resolución de 14 bits (0).
- Humidity measurement resolution [9:8]: resolución de 14 bits (00).
- El resto de bits son 0 ya que son solo de lectura o están reservados.

Así el byte de configuración es: ‘0001000000000000’.

La Función READ TEMPERATURE HUMIDITY, como se ha mencionado anteriormente, tiene como entrada la salida del bloque de lectura I<sup>2</sup>C, en este caso (por la configuración) dos vectores de uint16. Siguiendo la información de la datasheet, el primer vector — el de la temperatura— se inserta en la Ecuación 3.1 para obtener la temperatura en °C. Del mismo modo, el segundo vector se inserta en la Ecuación 3.2 para obtener la humedad en % (humedad relativa).

$$Temperature = \frac{Valor}{2^{16}} * 165 - 40 \quad (3.1)$$

$$Humidity = \frac{Valor}{2^{16}} * 100 \quad (3.2)$$

### 3.3.3. Detección de caída de corriente

La detección de la caída de suministro eléctrico se hace gracias al USB DIP (Sección 2.8.2), que conectado al GPIO 18 de la Raspberry da una señal de nivel alto cuando hay 5 V en el USB-mini y una señal de nivel bajo cuando hay 0 V. Así, y como se ha resumido en la Figura 3.8, si la señal es alta se escribe un 1 en el bus de control en el espacio VOLTAGE y un 0 en el caso contrario.

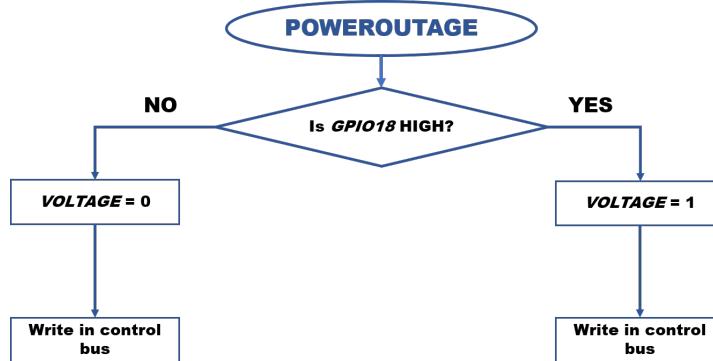


Figura 3.8. Diagrama de flujo del software de la detección de la caída de corriente. Fuente: elaboración propia.

### 3.4. Máquina de estados general

La máquina de estados general es la encargada de todo el funcionamiento del dispositivo, se ocupa de comparar los niveles de temperatura y voltaje y de activar el proceso del módulo GSM. Tiene tres estados: *WAITING*, *SEND SMS* y *WAIT FOR OK* (Figura 3.9). En cada estado se realizan los siguientes procesos:

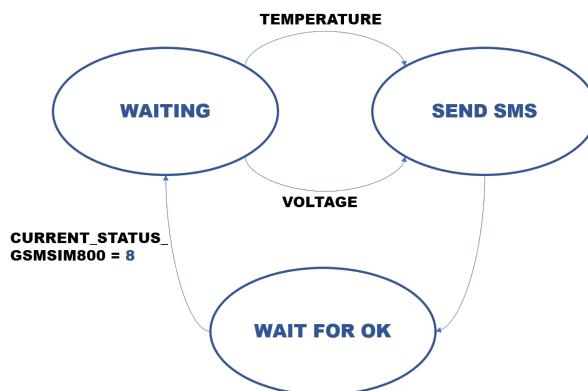


Figura 3.9. Máquina de estados general. Fuente: elaboración propia.

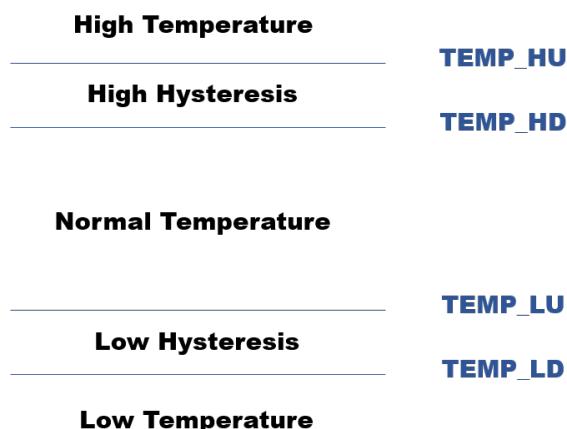
- **WAITING:** en este estado se comprueba si es necesario mandar un SMS, es decir, si la temperatura cambia de rango (los rangos se explican en la Sección 3.4.1 o el voltaje se puede considerar que ha cambiado de modo como se explica en la Sección 3.4.2). Además, en este estado se actualizan tanto la variable *VOLTAGE\_MODE* como la variable *TEMPERATURE\_MODE*. La transición al estado *SEND SMS* se da cuando hay un cambio en el voltaje o temperatura que es necesario notificar al usuario.
- **SEND SMS:** este estado activa el bloque del módulo GSM poniendo la variable *CURRENT\_STATUS\_GSMSIM800* en 1. (Sección 3.2). Al terminar la actualización de dicha variable, se pasa directamente al siguiente estado (*WAIT FOR OK*).
- **WAIT FOR OK:** en este estado se espera a que la máquina de estados del módulo GSM termine de mandar el SMS y se reinicie, como se explica en la Sección 3.2 esto se indica poniendo la variable *CURRENT\_STATUS\_GSMSIM800* a 8. Una vez se ha detectado que dicha variable tiene el valor 8 se pasa directamente al estado de *WAITING*.

### 3.4.1. Rangos de temperatura

Debido a que la temperatura puede fluctuar entre valores muy cercanos, esto podría significar que si solo se define un nivel a partir del cual se da un aviso, se podría estar avisando sin motivo. Por ejemplo, si el usuario necesita un aviso cuando la temperatura supera los 27 °C , la primera vez que la temperatura sube a ese nivel se mandará el SMS correctamente, pero si la temperatura se oscila entre 26,9 y 27 se estará avisando al usuario sin motivo.

Para evitar la situación anterior se definen unos niveles de histéresis, que de forma simplificada se puede entender que los niveles de encendido y apagado son distintos. En la Figura 3.10 se muestran las distintas zonas limitadas por estos niveles, las temperaturas mostradas son elegidas por el usuario y se denominan:

- TEMP\_ HU: “high up”, es el nivel al que el usuario desea recibir la notificación que indica una temperatura superior al rango deseado.
- TEMP\_ HD: “high down”, es el nivel al que el usuario desea recibir la notificación que indica que se ha vuelto al rango deseado desde una temperatura superior a la TEMP\_ HU.
- TEMP\_ LU: “low up”, es el nivel al que el usuario desea recibir la notificación que indica que se ha vuelto al rango deseado desde una temperatura inferior a la TEMP\_ LD.
- TEMP\_ LD: “low down”, es el nivel al que el usuario desea recibir la notificación que indica una temperatura inferior al rango deseado.



**Figura 3.10.** Rangos de fluctuación de la temperatura. Fuente: elaboración propia.

De esta manera se pueden dar las siguientes situaciones (y resumidas en la Tabla 3.1):

1. La temperatura entra en la zona *high temperature* y todavía no se ha dado el aviso: el sistema cambiará el modo de temperatura a 1 (aviso), el modo de mensaje a 1 (temperatura) y pasará al estado *SEND SMS*.
2. La temperatura está en la zona *high temperature* y se ha dado el aviso: el sistema seguirá en el modo de temperatura 1 (aviso), el modo de mensaje será 0 (NAN) temperatura y seguirá en el *WAITING*.
3. La temperatura está en la zona *high hysteresis* y el modo de temperatura es 1 (aviso), esto significa que la temperatura ha subido de TEMP\_ HU pero todavía no ha bajado al rango normal, por tanto el modo de temperatura seguirá siendo 1 y el modo mensaje seguirá siendo 0. La máquina de estados permanecerá en el estado *WAITING*.

4. La temperatura está en la zona *high hysteresis* y el modo de temperatura es 0 (no aviso), esto significa que la temperatura no ha subido de TEMP\_HU, por tanto el modo de temperatura seguirá siendo 0 y el modo mensaje seguirá siendo 0. La máquina de estados permanecerá en el estado *WAITING*.
5. La temperatura está en la zona *normal temperature* y el modo de temperatura es 1, esto significa que la temperatura ha subido de TEMP\_HU pero todavía ha bajado al rango normal, por tanto hay que notificar al usuario, el modo de temperatura seguirá actualizarse a 0 y el modo mensaje a 1. La máquina de estados pasará al estado *SEN SMS*.
6. La temperatura entra en la zona *normal temperature* y el modo de temperatura es 1, esto significa que no es necesario notificar al usuario, por tanto el modo de temperatura seguirá siendo 0 y el modo mensaje seguirá siendo 0. La máquina de estados permanecerá en el estado *WAITING*.
7. La temperatura entra en la zona *low temperature* y todavía no se ha dado el aviso: el sistema cambiará el modo de temperatura a 1 (aviso), el modo de mensaje a 1 (temperatura) y pasará al estado *SEND SMS*.
8. La temperatura está en la zona *low temperature* y se ha dado el aviso: el sistema seguirá en el modo de temperatura 1 (aviso), el modo de mensaje será 0 (NAN) temperatura y seguirá en el *WAITING*.
9. La temperatura está en la zona *low hysteresis* y el modo de temperatura es 1 (aviso), esto significa que la temperatura ha bajado de TEMP\_LD pero todavía no ha subido al rango normal, por tanto el modo de temperatura seguirá siendo 1 y el modo mensaje seguirá siendo 0. La máquina de estados permanecerá en el estado *WAITING*.
10. La temperatura está en la zona *low hysteresis* y el modo de temperatura es 0 (no aviso), esto significa que la temperatura no ha bajado de TEMP\_LD, por tanto el modo de temperatura seguirá siendo 0 y el modo mensaje seguirá siendo 0. La máquina de estados permanecerá en el estado *WAITING*.

### 3.4.2. Cambio de modo del voltaje

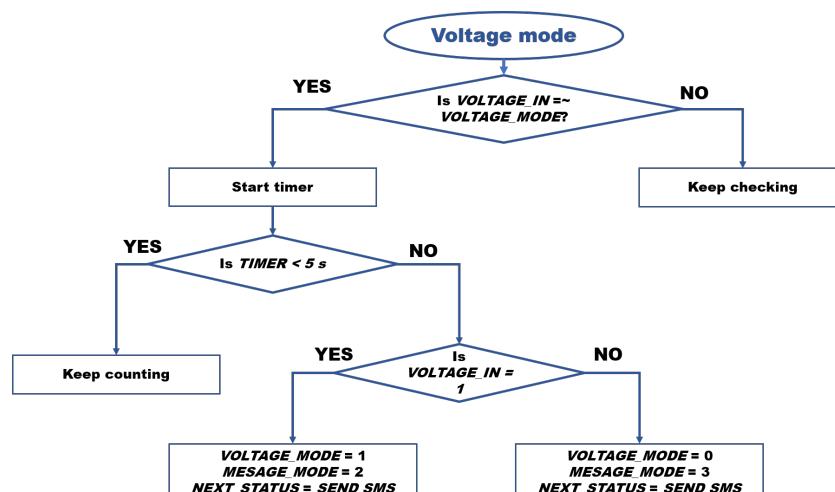


Figura 3.11. Diagrama de flujo del cambio del modo de voltaje. Fuente: elaboración propia.

Para el cambio del modo de voltaje, que significa notificar al usuario, ocurre lo mismo que con la temperatura. Si la tensión se va y vuelve enseguida, se notificará al usuario de forma

**Tabla 3.1.** Rangos de temperatura y acciones necesarias. Fuente: elaboración propia.

| Temperature Zone   | Temperature mode | Acciones   |
|--------------------|------------------|--|
| High Temperature   | 0                | TEMPERATURE_MODE = 1<br>MESSAGE_MODE = 1<br>Próximo estado: SEND SMS |
| High Temperature   | 1                | TEMPERATURE_MODE = 1<br>MESSAGE_MODE = 0<br>Próximo estado: WAITING  |
| High Histeresis    | 0                | TEMPERATURE_MODE se mantiene<br>Próximo estado: WAITING              |
|                    | 1                | MESSAGE_MODE = 0   |
| Normal Temperature | 0                | TEMPERATURE_MODE = 0<br>MESSAGE_MODE = 0<br>Próximo estado: WAITING  |
| Normal Temperature | 1                | TEMPERATURE_MODE = 0<br>MESSAGE_MODE = 1<br>Próximo estado: SEND SMS |
| Low temperature    | 0                | TEMPERATURE_MODE = 1<br>MESSAGE_MODE = 1<br>Próximo estado: SEND SMS |
| Low temperature    | 1                | TEMPERATURE_MODE = 1<br>MESSAGE_MODE = 0<br>Próximo estado: WAITING  |
| Low Histeresis     | 0                | TEMPERATURE_MODE se mantiene<br>Próximo estado: WAITING              |
|                    | 1                | MESSAGE_MODE = 0   |

incorrecta. Por ello, se esperará a que la tensión se haya estabilizado por lo menos 5 segundos para dar el aviso.

Así, si el voltaje leído y el modo de voltaje difieren, se empezará a contar, si se cuenta hasta 5 segundos y siguen siendo distintos se notificará al usuario poniendo el modo de voltaje igual al del voltaje de entrada ( 1 si hay tensión y 0 si la tensión es 0, el modo de mensaje se actualizará a 2 si el modo de voltaje ha pasado a ser 1 (*Power up*) o a 3 si ha pasado a ser 0 (*Power down*) y se pasará al estado *SEND SMS* (Figura 3.11).



# 4

## Resultados

---

Este capítulo presenta los resultados de las pruebas que se han realizado a los distintos componentes por separado y a todo el dispositivo en junto.

---

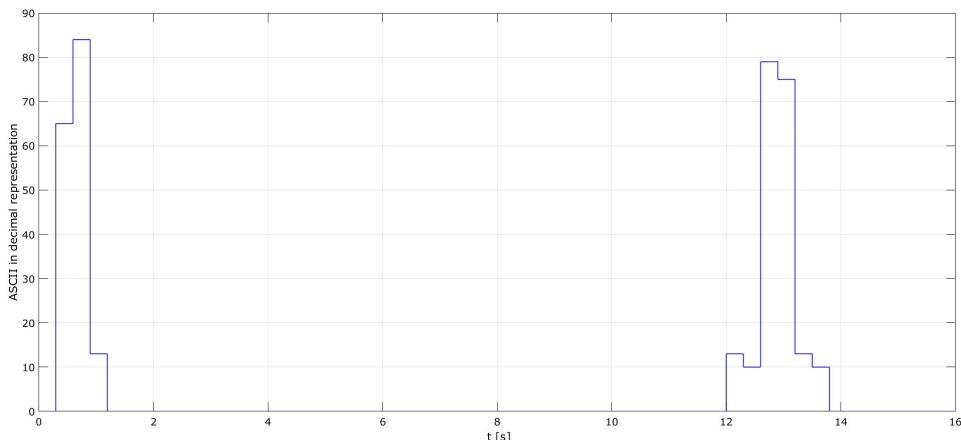
### 4.1. Módulo GSM

Para comprobar el correcto funcionamiento de módulo GSM se le sometió a una prueba que consistió en mandar un SMS con el texto *HOLA*. En esta prueba, a través del puerto serie se transmitieron los comandos necesarios para el envío del SMS y se decodificará la respuesta obtenida.

En la Figura 4.1 se muestra la secuencia ordenada de comandos y respuestas obtenidas. En la Figura 4.2 se muestra un ejemplo de los bytes recibidos en el puerto serie del micro-controlador durante el proceso. En este caso la respuesta al primer comando: *AT OK*.

| ENVÍO                 | RESPUESTA                                  |
|-----------------------|--|
| AT                    | AT <b>13 10</b> OK                         |
| AT+CMGF=1             | AT+CMGF=1 <b>13 10</b> OK                  |
| AT+CSCS="GSM"         | AT+CSCS="GSM" <b>13 10</b> OK              |
| A+CMGS="+34676217306" | AT+CMGS="+34642330167" <b>13 10</b> > HOLA |
| HOLA CRTL+Z           | +CMGS:202 <b>13 10</b> OK                  |

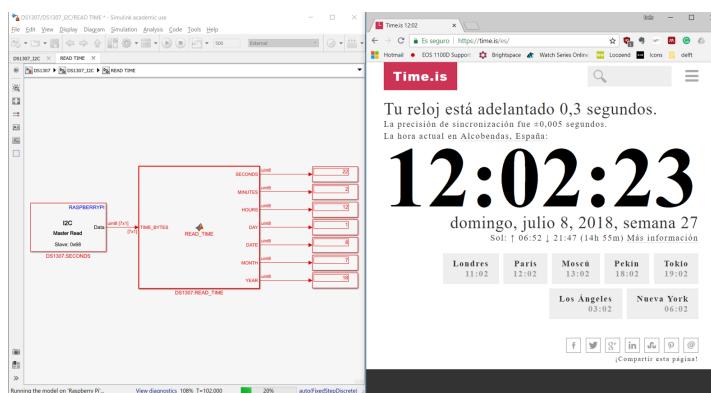
**Figura 4.1.** Imagen del scope de Simulink de la respuesta del módulo GSM a los comandos AT durante el proceso de mandar un SMS. Fuente: elaboración propia.



**Figura 4.2.** Imagen del scope de Simulink de la respuesta del módulo GSM a los comandos AT durante el proceso de mandar un SMS. Fuente: elaboración propia.

## 4.2. Real Time Clock

Para probar el funcionamiento del reloj en tiempo real, se comparó la fecha y hora de dicho sensor con la fecha y hora del ordenador de trabajo conectado a Internet. El resultado de la prueba se puede ver en la Figura 4.3. Así pues, se puede ver que el RTC marca la hora correctamente sin diferencia con respecto a Internet.



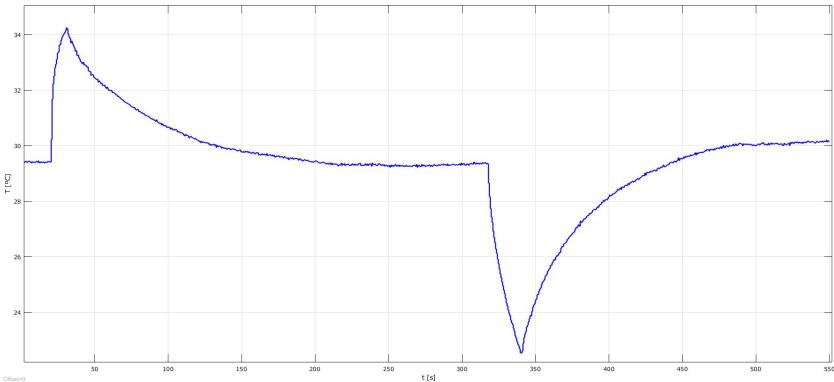
**Figura 4.3.** RTC en funcionamiento. Fuente: elaboración propia gracias a <https://time.is/es/>.

La imagen está dividida en dos partes, la de la izquierda que muestra la respuesta del RTC a Simulink y la de la derecha, una página web que muestra la hora en tiempo real. En el diagrama de Simulink los *displays* están en el siguiente orden:

1. Segundos: el diagrama de Simulink muestra 7, igual que en la página web.
2. Minutos: ambas partes muestran el mismo número, el 42.
3. Horas: ambas partes muestran el mismo número, el 23.
4. Día de la semana: el RTC indica que el día es el primero de la semana, en este caso el domingo (debido a la configuración).
5. Día del mes: ambas partes muestran el mismo número, el 15.
6. Mes: ambas partes muestran el mismo número, el 7 (julio).
7. Año: ambas partes muestran el mismo número, el 18.

## 4.3. Sensor de temperatura

Para probar el funcionamiento del sensor de temperatura, se le aportó una fuente de calor en el segundo 20, y un fuente de “frío” en el segundo 318 obteniendo el resultado de la Figura 4.4.



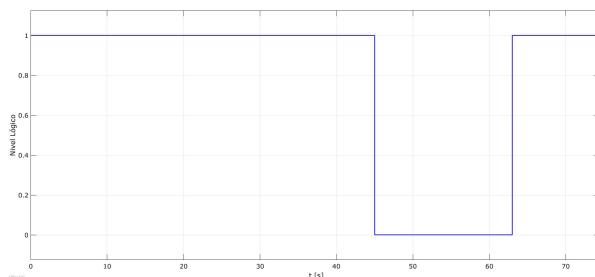
**Figura 4.4.** Respuesta del sensor de temperatura a diferentes fuentes de calor. Fuente: elaboración propia.

El sensor se estabilizó entorno a los 29.5 °C mientras no se le oportó ninguna fuente de calor. En el segundo 20, se le aportó una fuente de calor, que en este caso calor humano a través de un dedo de la mano. Se observó como a partir de este momento, la temperatura comenzó a ascender hasta llegar a 34.2 °C en el segundo 30. En este momento en el que se retira dicha fuente de calor, la temperatura comienza a descender se estabilizó de nuevo en torno a los 29 °C.

Se ensayó también el funcionamiento en temperaturas más bajas, acercando una fuente de “frío”, que en este caso fue un hielo protegido por un paño para no mojar el sensor. Esta fuente se aportó en el segundo 318 y se retiró tras unos instantes (segundo 340). En este momento la temperatura descendió descendido hasta 22.5 °C. Se apreció como la temperatura desciende cuando el sensor está en contacto con la fuente “fría”, y cuando esta se retira comienza de nuevo a ascender hasta su régimen estable.

## 4.4. Detección de la caída de la red eléctrica

Para comprobar que el *hardware* dedicado a la detección de la caída y restablecimiento de la corriente funciona correctamente, se sometió a la siguiente prueba: se dio tensión al dispositivo, se retiró la tensión y se volvió a alimentar. El resultado de este proceso muestra en la Figura 4.5.



**Figura 4.5.** Respuesta del *hardware* dedicado a la detección de caída de la red eléctrica ante distintas situaciones. Fuente: elaboración propia.

La detección de la corriente se ha diseñado, como se explica en la Sección 3.3.3, para que solo dé aviso si la tensión de entrada se estabiliza en un valor (0 ó 1) durante más de 5 segundos. Así, en la gráfica de resultados podemos ver cómo habiéndose retirado la tensión en el segundo 40, la señal de caída de tensión no se va a 0 hasta el segundo 45. Del mismo modo, solo vuelve a estabilizarse en 1 en el segundo 63, habiéndose retirado la tensión en el segundo 58.

## 4.5. Funcionamiento del dispositivo

Se realizó una última prueba, es decir, la notificación tanto de caída y restablecimiento del suministro, como de salida y retorno al rango de temperatura establecido.

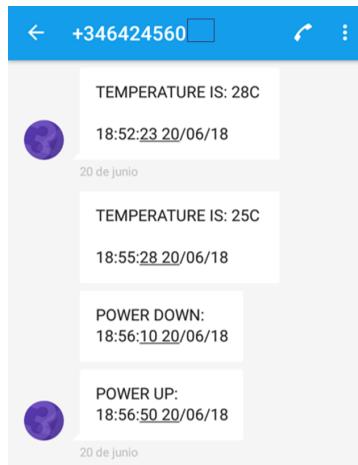


Figura 4.6. Respuesta del dispositivo completo ante distintas situaciones. Fuente: elaboración propia.

Se puede apreciar en la Figura 4.6 que primero se ha notificado que la temperatura ha rebasado el límite superior de 27 °C . En este caso la temperatura ascendió con tanta rapidez, que en el momento de la notificación la temperatura era de 28 °C. Más tarde, cuando descendió a 25 °C se volvió a notificar, ya que volvió a entrar en el “rango normal”. Despues de someter al dispositivo a cambios de temperatura, se le quitó la corriente al mismo para simular una caída de tensión, y pasados unos segundos, se volvió a alimentar.

Como se puede ver, los mensajes contienen la hora del evento y la fecha; además del texto que indica la situación que ha provocado la notificación. En el caso de que haya sido debido a la temperatura, ésta también se indica en el mensaje.

# 5

# Conclusiones y futuros desarrollos

---

Este capítulo presenta las conclusiones extraídas de este proyecto así como posibles futuros desarrollos.

---

## 5.1. Conclusiones

Durante el desarrollo del proyecto se han cumplido los objetivos marcados en la Sección 1.4.

### 5.1.1. Actuador

La base del proyecto fue la notificación por SMS, para ello se utilizó el módulo GSM SIM800C. Se implementó el protocolo de comunicación serie junto con los comandos de Hayes con éxito para que el micro-controlador se pudiera comunicar correctamente con dicho módulo.

Se consiguió decodificar todas las respuestas del módulo ante los comandos. Y así se pudo ordenar los comandos correctamente para el envío del SMS. Por último, se logró enviar SMS con texto a cualquier teléfono.

Durante la implementación del proceso para mandar el SMS surgieron varios problemas. Uno de ellos está relacionado con la velocidad de la comunicación. El módulo trabaja con unos baudios autoajustables en un rango bastante amplio; desde la Raspberry solo se consiguió comunicarse a una velocidad de 4800 baudios. Una vez conseguido el envío de SMS, estos estaban vacíos. La causa era el exceso de tiempo entre las dos últimas órdenes necesarias: el comando para mandar el SMS y el texto del mismo. El micro-controlador dejaba un tiempo de espera excesivo entre la transmisión de ambas.

Por último, cuando se consiguió enviar texto, se enviaba un número exorbitado de @ indeseadas que tras varias pruebas se eliminaron. La causa era el envío de ceros por el puerto serie para dejarlo en reposo entre el comando de envío, el texto y la orden de que el texto está

finalizado. Para ello se creó un buffer que permitía colocar dichos ceros donde mejor convenía — los ceros eran necesarios ya que el micro-controlador solo es capaz de mandar mensajes de la misma longitud por el puerto serie.

### 5.1.2. Sensores

El sensor de temperatura HDC1080 ya había sido implementado anteriormente en un micro-controlador distinto. Al intentar implementar los bloques que ya habían sido usados, el sensor no respondía. Tras una serie de pruebas, se dedujo que el problema era que al mandarse la orden de “tomar medida” sin dejar ningún tiempo para leer dicha medida, nunca comunicaba el resultado al micro-controlador. Así, se diseñó un nuevo bloque que ordenaba medir, y leía la respuesta por turnos.

El reloj en tiempo real DS1307 se implementó desde cero creando los bloques. Además, se sincronizó con Internet a través de la Raspberry de manera que si hay conexión, el reloj se pone automáticamente en hora.

Uno de los objetivos del proyecto fue detectar la caída y restablecimiento del abastecimiento eléctrico. Se logró detectar estas situaciones de una manera simple y barata y se implementó con éxito en el conjunto del proyecto.

### 5.1.3. Presupuesto

Como se refleja en el Apéndice A, el precio total del *hardware* asciende a 118€ aproximadamente. Este precio corresponde a la compra de los componentes al por menor, por tanto se podría reducir notablemente el precio si se fabricara de forma industrial.

Un claro ejemplo es el *Shield*, únicamente utilizado para que la conexión de los sensores fuera cómoda y limpia sin necesidad de cables.

## 5.2. Futuros desarrollos

En este proyecto se ha utilizado la comunicación SMS para la notificación tanto de caída como de restablecimiento de la red eléctrica, y de salida de un rango de temperatura. En proyectos futuros, se podría utilizar para también para recibir datos por SMS, no solo para transmitirlos.

Así, el proyecto se podría integrar en un entorno domótico, y por ejemplo, controlar la temperatura a través de actuadores. Además, con el mismo módulo GSM se podrían enviar MMS con imágenes para comprobar la identidad de las personas que llaman a un domicilio o verificar que las personas que están dentro de la casa están bien – como niños o personas de la tercera edad.

Otro tipo de notificación posible con el módulo es la conexión por GPRS, que en este proyecto no se ha desarrollado ya que el SMS puede resultar más fiable. A través de GPRS se pueden transmitir todo tipo de datos a cualquier dispositivo con conexión a Internet.

# A

## Presupuesto

---

En este anexo, se hace una estimación de los costes que ha supuesto este proyecto. El presupuesto está dividido en cuatro grupos: *hardware*, *software*, herramientas y equipos y mano de obra directa

---

### A.1. Sumas parciales

#### A.1.1. Componentes hardware

Tabla A.1. Sumas parciales: comonentes HARDWARE

| Componente                    | Cantidad | €/unidad | Coste Parcial (€) |
|-------------------------------|----------|----------|-------------------|
| Raspberry Pi 3B               | 1        | 39       | 39                |
| DS1307 (RTC2 click)           | 1        | 15       | 15                |
| HDC1080                       | 1        | 9        | 9                 |
| USB a DIP                     | 1        | 1,5      | 1,5               |
| 5 to 3.3 VDC-DC AMS1117 800MA | 1        | 2        | 2                 |
| Jackery Titan                 | 1        | 40       | 40                |
| Pi 3 Click Shield             | 1        | 12       | 12                |
| GSM SIM800C                   | 1        | 15       | 15                |
| <b>TOTAL</b>                  |          |          | <b>133,5</b>      |

#### A.1.2. Software

Tabla A.2. Sumas parciales: *software*

| Programa        | Cantidad | €/unidad | Coste Parcial (€) |
|-----------------|----------|----------|-------------------|
| Matlab/Simulink | 1        | 700      | 700               |
| Putty           | 1        | 0*       | 0*                |
| TEXMaker        | 1        | 0*       | 0*                |
| <b>TOTAL</b>    |          |          | <b>700</b>        |

\* Software gratuito.

### A.1.3. Herramientas y equipos

**Tabla A.3.** Sumas parciales: herramientas y equipos

| Elemento     | Cantidad | €/unidad     | Coste Parcial (€) |
|--------------|----------|--------------|-------------------|
| Ordenador    | 1        | 600          | 600               |
| Osciloscopio | 1        | 519          | 519               |
|              |          | <b>TOTAL</b> | <b>1.119</b>      |

### A.1.4. Mano de obra directa

**Tabla A.4.** Sumas parciales: obra de mano directa

| Nombre                                   | Horas | €/hora       | Coste Parcial (€) |
|--|-------|--------------|-------------------|
| Estado del Arte                          | 5     | 17           | 85                |
| Implementación del sensor de temperatura | 5     | 35           | 175               |
| Implementación del reloj en tiempo real  | 4     | 35           | 140               |
| Implementación del módulo GSM            | 175   | 70           | 12.250            |
| Puesta en conjunto                       | 30    | 65           | 1.950             |
| Documentación del proyecto               | 100   | 40           | 4.000             |
|  |       | <b>TOTAL</b> | <b>18.600</b>     |

## A.2. Presupuesto general

**Tabla A.5.** Presupuestos generales

| Nombre                 | Coste Parcial (€) |
|------------------------|-------------------|
| Componentes            | 118,5             |
| Software               | 700               |
| Herramientas y equipos | 1.119             |
| Obra de mano directa   | 18.600            |
|                        | <b>TOTAL</b>      |
|                        | <b>20.537,5</b>   |

# Bibliografía

- [1] I. Corporation, “Infographic 40 years of innovation.” <https://newsroom.intel.com/press-kits/40th-anniversary-of-the-microprocessor/#> Infographic 40 years of innovation. Accessed: 11-04-2018.
- [2] Wikipedia, “Domus definición.” <https://es.wiktionary.org/wiki/domus#Lat%C3%ADn>. Accessed: 11-04-2018.
- [3] Definiciona, “Definición de autonomo.” <https://definiciona.com/autonomo/>. Accessed: 16-02-2018.
- [4] C. Reinisch, M. J. Kofler, and W. Kastner, “Thinkhome: A smart home as digital ecosystem,” in *4th IEEE International Conference on Digital Ecosystems and Technologies*, pp. 256–261, April 2010.
- [5] Pumpalarm.com, “Cellular alarm.” <https://www.pumpalarm.com/>. Accessed: 16-02-2018.
- [6] iSocket Systems, “isocket 3g.” <https://www.isocket3g.com/en/>. Accessed: 16-02-2018.
- [7] DomoDesk, “Enchufe gsm/sms alerta de corte y temperatura.” <http://www.domodesk.com/899-enchufe-gsmsms-alerta-corte-de-corriente-y-temperatura.html>. Accessed: 16-02-2018.
- [8] RaspberryPiFoundation, “Raspberry pi.” <http://www.raspberrypi.org>. Accessed: 16-02-2018.
- [9] T. Petterson, “Raspberry pi just turned 5.” <https://medium.freecodecamp.org/raspberry-pi-just-turned-5-d4210cc29230>. Accessed: 16-02-2018.
- [10] GSMAssociation, “Gsma history.” <https://www.gsma.com/aboutus/history>. Accessed: 16-02-2018.
- [11] G. H. Office, “Annual report 2017,” tech. rep., GSM Association, London, UK, 2017.
- [12] Banggood, “Gsm module specifications.” <https://www.banggood.com/SIM800C-GPRS-GSM-Module-Development-Board-For-Raspberry-Pi-p-1062138.html?rmmds=search>.
- [13] P. Scherz, *Practical Electronics for Inventors*. The McGraw-Hill Companies, 2006.
- [14] “Tia eia-637-a, short message service,” tech. rep., Telecommunications Industry Association, 1999.

## Bibliografía

- [15] C. Peersman, S. Cvetkovic, P. Griffiths, and H. Spear, “The global system for mobile communications short message service,” *IEEE Personal Communications*, vol. 7, pp. 15–23, June 2000.
- [16] X. Meng, P. Zerfos, V. Samanta, S. H. Y. Wong, and S. Lu, “Analysis of the reliability of a nationwide short message service,” in *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*, pp. 1811–1819, May 2007.
- [17] N. P. Jawarkar, V. Ahmed, and R. D. Thakare, “Remote control using mobile through spoken commands,” in *2007 International Conference on Signal Processing, Communications and Networking*, pp. 622–625, Feb 2007.
- [18] P. A. Laplante, *Comprehensive Dictionary of Electrical Engineering*. CRC Press, 1999.
- [19] MikroElektronika, “Click shield specifications.” <https://www.mikroe.com/pi-3-click-shield>. Accessed: 24-05-2018.
- [20] RaspberryPiWiKi, “Raspi ups hat board.” [http://www.raspberrypiwiki.com/index.php/Raspi\\_UPS\\_HAT\\_Board](http://www.raspberrypiwiki.com/index.php/Raspi_UPS_HAT_Board). Accessed: 25-06-2018.
- [21] Amazon, “Jackery titan 20100 mah portable charger battery pack 3.4a (max).” <https://www.amazon.com/Jackery-Portable-Superior-Charging-Technology/dp/B015QGW4BU>. Accessed: 01-07-2018.
- [22] Wikipedia, “Ethtool.” <https://en.wikipedia.org/wiki/Ethtool>. Accessed: 26-06-2018.
- [23] J. W. Linville, “Ethtool - utility for controlling network drivers and hardware.” <https://en.wikipedia.org/wiki/Ethtool>. Accessed: 26-06-2018.
- [24] Matlab, “Getting started with matlab support package for raspberry pi hardware.” <https://es.mathworks.com/help/supportpkg/raspberrypiio/examples/getting-started-with-matlab-support-package-for-raspberry-pi-hardware.html>.
- [25] Banggood, “Usb to dip female head mini-5p patch to dip.” [https://www.banggood.com/es/USB-To-DIP-Female-Head-Mini-5P-Patch-To-DIP-2\\_54mm-Adapter-Board-p-1167632.html?cur\\_warehouse=CN](https://www.banggood.com/es/USB-To-DIP-Female-Head-Mini-5P-Patch-To-DIP-2_54mm-Adapter-Board-p-1167632.html?cur_warehouse=CN). Accessed: 26-06-2018.
- [26] Banggood, “5pcs 5v to 3.3v dc-dc ams1117 800ma step-down power supply buck module.” [https://www.banggood.com/es/5Pcs-5V-To-3\\_3V-DC-DC-AMS1117-800MA-Step-Down-Power-Supply-Buck-Module-p-944686.html?cur\\_warehouse=CN](https://www.banggood.com/es/5Pcs-5V-To-3_3V-DC-DC-AMS1117-800MA-Step-Down-Power-Supply-Buck-Module-p-944686.html?cur_warehouse=CN). Accessed: 26-06-2018.
- [27] Matt, “Adding a ds3231 real time clock to the raspberry pi.” <https://www.raspberrypi-spy.co.uk/2015/05/adding-a-ds3231-real-time-clock-to-the-raspberry-pi/>. Accessed: 20-04-2018.



