Universidad Zaragoza
1542

Trabajo Fin de Grado

Sistema remoto para el control automático de la temperatura de una vivienda

Autor/es

Diego Polo Pérez

Director/es

Eduardo Gil Herrando

Escuela Universitaria Politécnica de Teruel

2017

"Sistema remoto para el control automático de temperatura de una vivienda"

Resumen

En este Trabajo Fin de Grado se ha diseñado e implementado un sistema de control de temperatura de una vivienda de manera remota mediante la comunicación entre un dispositivo móvil y un microcontrolador. Dicho control se ejerce sobre los actuadores disponibles (calefacción y persiana).

Existe un intercambio de información bidireccional entre el dispositivo móvil y la placa Arduino que controla el sistema mediante un módulo GSM. Esta interacción entre dispositivos se realiza de dos formas diferentes: mediante SMS o llamadas perdidas.

Además, el Arduino realiza una identificación del número de teléfono como elemento de seguridad; de este modo únicamente pueden acceder a la información y el control del sistema aquellos números autorizados para ello.

Palabras clave: Arduino, calefacción, persiana, relé y sensor.

"Remote control system for the automatic control of the temperature of a house"

Abstract

This project shows a sketch of remote control system. It has been designed to manage the temperature of the house with a mobile device. The management mentioned works controlling the temperature equipment (heating and blinds).

Particularly, the information exchange is produced between a mobile phone and a microprocessor (Arduino) which controls the technique by a GSM section. That interaction is made in two different ways: by missed calls or SMS.

In addition, the Arduino produces a telephone number identification as a security; in order to allow only the authorized numbers access to the information.

Key words: Arduino, blind, heating, relay and sensor.

ÍNDICE GENERAL

Índices

Memoria

Anexos

Índices

GUÍA DE ABREVIATURAS Y SIGLAS

C: Common (Común)

CA: Corriente Alterna

CC: Corriente Continua

GND: Ground (Tierra)

GPRS: General Packet Radio Service (Paquete general de radio servicio)

GPS: Global Positioning System (Sistema de posicionamiento global)

GSM: Global System for Mobile communications (sistema global para comunicaciones móviles)

KB: Kilobit

KB/s: Kilobit por segundo

KΩ: Kiloohmios

LED: Light Emitting Diode (Diodo emisor de luz)

MHZ: Megahercio

Min: Minuto

mV: Milivoltio

NC: Normally Closed (Normalmente cerrado)

NO: Normally Open (Normalmente abierto)

Nº: Número

PWM: Pulse-Width Modulation (Modulación por ancho de pulsos)

SAI: Sistema de Alimentación Ininterrumpido

SIM: Subscriber Identity Module (Módulo de identidad de suscripción)

SMS: Short Message Service (Servicio de mensajería instantánea)

SSR: Solid State Relay (Relé de estado sólido)

TFG: Trabajo Fin de Grado

UART: Universal Asynchronous Receiver-Transmitter (Transmisor-receptor asíncrono universal)

USB: Universal Serial Bus (Bus universal en serie)

V: Voltio

WAP: Wireless Application Protocol (Protocolo de aplicaciones inalámbricas)

3G: Tercera Generación

ºC: Grado Celsius

€: Euro

LISTA DE FIGURAS

LISTA DE TABLAS

Memoria

1. INTRODUCCIÓN

A día de hoy la domótica está todavía en auge a pesar del costo que conlleva la contratación e instalación de los diferentes componentes en relación con dicho campo.

El sistema de control desarrollado en este Trabajo Fin de Grado se aplicará en una vivienda de la localidad de Bronchales, situada en la Sierra de Albarracín al suroeste de la provincia de Teruel. Este pueblo se caracteriza por ser uno de los más altos de España con una altura de 1569 metros sobre el nivel del mar.

Debido a dicha altura, en Bronchales existen temperaturas bastante bajas sobre todo durante los meses de invierno. Un ejemplo de ello se puede encontrar en la Tabla 7 del Anexo 2. En ella, se muestra que en el mes de enero la temperatura media es de -0.2 ºC.

Por otra parte, cabe destacar que la red de telefonía móvil de la compañía telefónica a la que se le ha comprado la tarjeta SIM es aceptable en dicha población. En la Figura 16 del Anexo 2 se muestra la cobertura 3G que sería suficiente para una buena comunicación.

El principal motivo por el cual se ha desarrollado este proyecto es la necesidad de controlar la temperatura de la vivienda a distancia, ya que hoy en día es necesario estar *in situ* para encender la calefacción y abrir o cerrar las persianas. Además, también influye el tiempo de respuesta del sistema porque es una calefacción de tipo suelo radiante y, por lo tanto, el tiempo necesario hasta alcanzar la temperatura deseada es considerable.

En definitiva, estos problemas son los que no hacen atractivo gastar un periodo corto de vacaciones (como puede ser el fin de semana) en esta localidad. Por ello, este proyecto ha sido diseñado con el fin de solventar dichas dificultades.

2. OBJETIVOS

Los objetivos que se plantean en este TFG son principalmente de dos tipos: objetivo general y los objetivos específicos.

Para que alcanzar el objetivo general, se tendrán que cumplir previamente todos los objetivos específicos en los cuales se emplearán herramientas computacionales, tales como el programa de Arduino, Eagle, Cacoo y notepad++. Igualmente se necesitan una

serie de dispositivos físicos que habrá que configurar de tal manera que se adapten a los requerimientos necesarios.

2.1 OBJETIVO GENERAL

- Control de la temperatura de una vivienda de manera remota comunicándose a través del uso de la red de telefonía móvil.

2.2 OBJETIVOS ESPECÍFICOS

- Consultar y examinar el manual del termostato (centralita E20) para conocer sus características, y estudiar si cumple las especificaciones necesarias para la realización de este proyecto.
- Analizar el funcionamiento de una persiana eléctrica para poder manejarla tanto de forma remota como manual.
- Elaboración de los diferentes esquemas eléctricos necesarios.
- Conseguir una comunicación continua con el microcontrolador véase Figura 17 del Anexo 2
- Desarrollar un código en Arduino para el microcontrolador.

3. ESPECIFICACIONES DE DISEÑO

Las funciones principales que lleva a cabo el sistema son:

1. Modo manual: permite subir y bajar persianas. Una vez finalizado el movimiento, el relé principal vuelve al estado de reposo pudiendo así controlarse las persianas manualmente (con el interruptor de la pared).
2. Modo automático: accede al control de apertura y cierre de las persianas en función de la luz. El interruptor de la pared quedará anulado hasta que no se cambie al modo manual y haya finalizado el movimiento.
3. Programación de la calefacción a la temperatura de climatización deseada.
4. Aviso de temperaturas extremas: el sistema realiza una llamada perdida al último número que se ha comunicado con él, o en su carencia la realiza a un número inicializado por defecto. El significado de esta llamada es que la temperatura ha excedido unos valores límites de temperatura, puede ser tanto superior como inferior.

5. Reinicio del módulo GSM por fallo en la conexión a la red: el dispositivo cada cierto tiempo (se ha programado un intervalo de 10 segundos en el código de Arduino) comprueba su estado de conexión y en caso de fallo se reinicia.
6. Información de la temperatura actual y la temperatura diaria (máxima y mínima): posibilita el seguimiento continuo de las temperaturas durante un periodo de 24 horas.
7. Información de la temperatura de climatización y del estado de la calefacción (encendida o apagada).
8. Información del modo y estado de la persiana.

Otras características destacadas son:

1. Histéresis de la temperatura y luminosidad: es un método eficaz y usado de forma general para evitar pequeñas variaciones o desviaciones en las medidas que provocarían una rápida y repetida conmutación del relé. Esto se produce cuando se alcanzan valores próximos al valor deseado en un sistema de control de tipo ON-OFF.
2. Detección de números de teléfonos autorizados: en el código de Arduino está implícito una serie de números telefónicos autorizados. Si un número no autorizado intenta mandar un SMS o llamar, la información recibida será rechazada. Esto resulta interesante ya que se impide que mensajes de publicidad, llamadas erróneas o incluso números no autorizados puedan acceder a la información del domicilio u ordenar cualquier acción indeseada.
3. Comunicación vía SMS además de llamadas perdidas: como se ha explicado anteriormente, el dispositivo atiende tanto a llamadas perdidas como a mensajes a excepción de la climatización, ya que en ella se debe especificar los grados. En la Figura 24 del Anexo 2 se contempla este último caso así como el modo y el estado de la persiana. Mientras, en la Figura 25 del Anexo 2, no se muestra ninguna de estas características porque el Arduino no había recibido ninguna orden previa (acababa de encenderse).
4. Capacidad de incrementar el número de sistemas a controlar como por ejemplo la adición de persianas, o un sensor de temperatura exterior.

En la Tabla 1 y Tabla 2 se muestran los SMS y el número de tonos (llamadas perdidas) que se pueden enviar desde un teléfono autorizado con sus respectivas

acciones. Cabe destacar que los mensajes de texto se escriben todos caracteres con mayúscula y sin tildes.

Número de tonos	Acción
1	Activar el modo automático
2	Recibir un SMS con información actual
3	Subir persiana (modo manual)
4	Bajar persiana (modo manual)

Tabla 1: Número de tonos desde teléfono autorizado

SMS a enviar	Acción
AUTOMATICO	Activar el modo automático
INFO	Recibir un SMS con información actual
SUBIR	Subir persiana (modo manual)
BAJAR	Bajar persiana (modo manual)
T-9	Temperatura de climatización a alcanzar -9
T-5	Temperatura de climatización a alcanzar -5
T00	Temperatura de climatización a alcanzar 0
T15	Temperatura de climatización a alcanzar 15
T16	Temperatura de climatización a alcanzar 16
T17	Temperatura de climatización a alcanzar 17
T18	Temperatura de climatización a alcanzar 18
T19	Temperatura de climatización a alcanzar 19
T29	Temperatura de climatización a alcanzar 29

Tabla 2: Mensaje a enviar desde número de teléfono autorizado

4. COMPONENTES UTILIZADOS

En este apartado se especifican las características de cada uno de los componentes (un total de 8 dispositivos) utilizados para llevar a cabo el proyecto. Asimismo. Se detallan los motivos que han incitado a decantarse por cada uno de ellos.

4.1 ARDUINO MEGA 2560

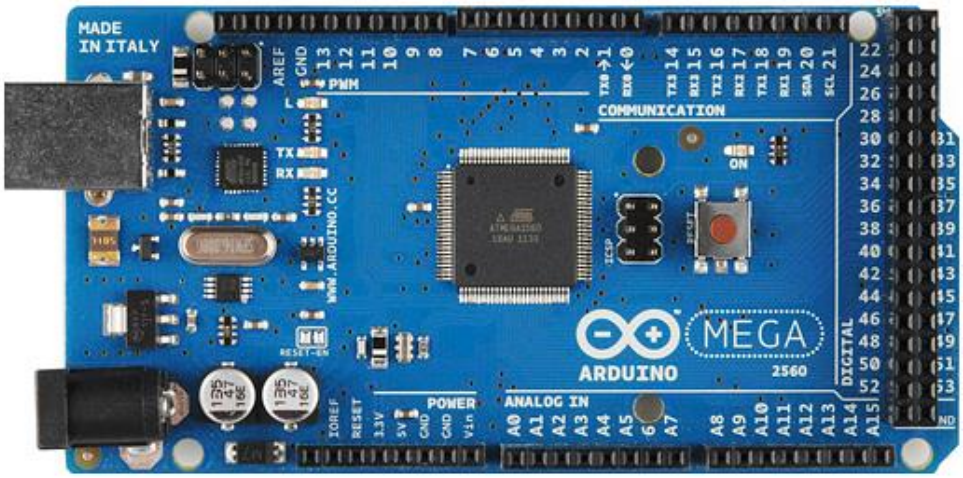

Figura 1: Arduino Mega[1]

"La Mega 2560 es una placa electrónica basada en el Atmega2560. Cuenta con 54 pines digitales de entrada / salida (de los cuales 15 se pueden utilizar como salidas PWM), 16 entradas analógicas, 4 UARTs (puertos serie de hardware), un oscilador de 16MHz, una conexión USB, un conector de alimentación, un conector ICSP, y un botón de reset. Contiene todo lo necesario para apoyar el microcontrolador; basta con conectarlo a un ordenador con un cable USB o a la corriente con un adaptador de CA a CC o una batería. La placa Mega 2560 es compatible con la mayoría de los módulos para el Uno y las placas anteriores Duemilanove o Diecimila", como menciona Manuel Delgado Crespo en su blog[1].

Arduino trabaja con un software compatible con la mayoría de sistemas operativos conocidos como pueden ser Microsoft Windows, Linux o Mac OS X.

Otra de las ventajas de utilizar este microcontrolador es la facilidad de búsqueda de información debida a la abundante cantidad de páginas web que posee Internet en relación con Arduino: videos, ejemplos, proyectos, etc.

Existe una extensa variedad de placas con las que se puede realizar este trabajo, por ello no siempre se da una solución única. A continuación, se exponen algunas de las especificaciones de las placas seleccionadas para obtener una comparativa global más clara:

Placa	Arduino Uno i2c Atmega 328	Arduino Mega 2560	Raspberry Pi A+
Precio (€)	7	12	22
Tipo de circuito integrado	microcontrolador	microcontrolador	microprocesador
Sistema procesador	ATmega328	ATmega2560	ARM1176JZF-S
Memoria	SRAM 2KB - EEPROM 1KB	SRAM 8KB - EEPROM 4KB	RAM 512 MB
Memoria flash (KB)	32	256	microSD
Frecuencia (MHZ)	16	16	700
Cantidad de pines analógicos	6	16	-
Cantidad de pines digitales	14	54	8 GPIO Digital

Tabla 3: Comparación de diferentes placas[2], [3], [4]

En la Tabla 3 se muestran las principales características que se han tenido en cuenta para esta elección. Se ha seleccionado Arduino Mega 2560 frente a Arduino Uno debido a su gran número de pines. En este trabajo se utilizan sólo 2 pines analógicos y 4 digitales porque se ha tenido en cuenta el control de una sola persiana, sin embargo, ya fuera del ámbito del presente TFG, se plantea extender el control al resto de persianas de la vivienda lo que conllevará un incremento del número de pines.

Por otro lado, Raspberry Pi A+ es un microprocesador que, en comparación con Arduino Mega 2560, tiene unas características muy superiores pero que no son imprescindibles para este trabajo en concreto. Además, como principales desventajas están el precio y el consumo energético.

4.2 MÓDULO GSM/GPRS SIM900

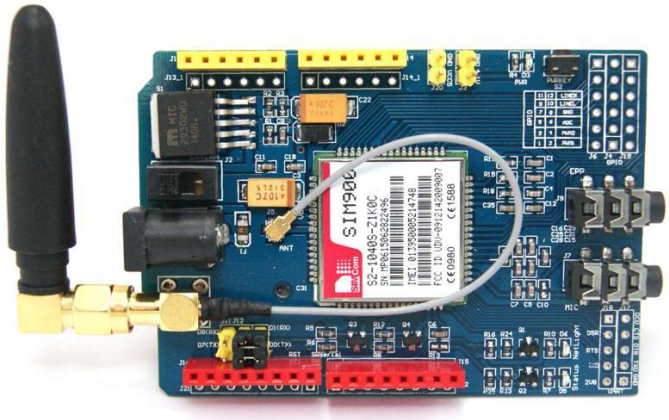

Figura 2: Módulo GSM/GRPS[5]

El módulo GSM/GPRS SIM900 se encarga de la comunicación a través de la red de telefonía móvil. No obstante, se debe colocar una tarjeta SIM que permite el acceso a dicha red con funciones propias de un teléfono móvil. Algunas de estas funciones son

enviar y recibir tanto llamadas como mensajes, conectarse a Internet mediante uso de datos, localización GPS, etc.

Entre otros elementos, dispone de entrada para micrófono y salida para auriculares, una antena, y la posibilidad de colocar una pila.

Por otro lado, las ventajas que aporta GPRS respecto a GSM son, además de una mayor velocidad de transmisión (9,6 KB/s frente a 171 KB/s), "la conexión permanente y la tarificación por tráfico, convirtiéndolo en el portador ideal para los servicios WAP, el acceso a Internet y el acceso a intranets de empresas" como menciona Elnaanox en su web[6].

Asimismo, cabe destacar el conjunto de comandos hayes o también denominados comandos AT (Atención) que inicialmente fueron usados para configurar módems[7], son utilizados como principal lenguaje de programación,.

Algunos de los comandos AT utilizados en el trabajo[8]:

AT → Comprueba estado del módulo.

AT+CPIN="XXXX" → Introducir el PIN de la SIM.

AT+CREG? → Comprueba la conexión a la red.

ATDXXXXXXX → Realiza una llamada.

ATH → Finaliza la llamada.

AT+CMGF=1 → Configura el modo texto para enviar o recibir mensajes. Devuelve ">" como inductor.

AT+CMGS="XXXXXXXXX" → Nº al que vamos a enviar el mensaje.

AT+CLIP=1 → Activamos la identificación de llamada.

AT+CNMI=2,2,0,0,0 → Configuramos el módulo para que muestre los SMS por el puerto serie.

4.3 MÓDULO DE 8 RELÉS ELECTROMECÁNICOS OPTOACOPLADOS

Figura 3: Módulo de relés electromecánicos[9]

"Un *optoacoplador* es un dispositivo de emisión y recepción que funciona como un interruptor activado mediante la luz emitida por un diodo LED que satura un componente optoelectrónico, normalmente en forma de fototransistor o fototriac"[10].

Este componente se emplea en el proyecto porque se pretende garantizar el aislamiento entre los circuitos de mando y fuerza. El siguiente esquema eléctrico muestra una manera muy simple de entender el funcionamiento de estos circuitos.

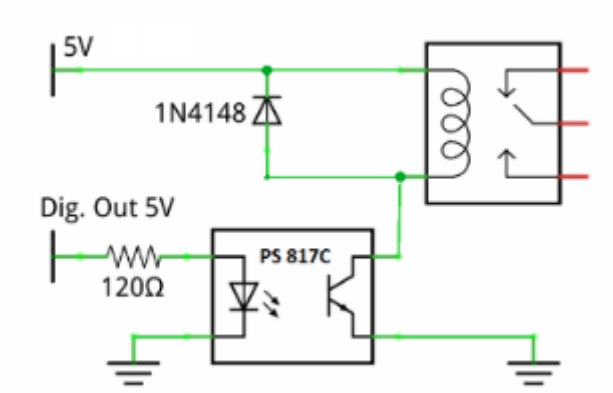

Figura 4: Esquema eléctrico de un optoacoplador y un relé[10]

Características principales[11]:

-Soporta 10A a 220VAC o 10A y hasta 30VDC.

-El control se realiza entre 3 y 6V.

-Dispone de un puente para seleccionar el aislamiento completo del circuito mando y fuerza.

Cabe destacar, como detalle fundamental a tener en cuenta, que estos relés funcionan con lógica inversa; es decir, a la hora de programar se activan con la constante LOW y se "relajan" con HIGH.

Por otro lado, existen los relés de estado sólido (SSR). Estos utilizan semiconductores de potencia, tales como tiristores y transistores para conmutar. Cabe destacar que su velocidad de conmutación es muy elevada en comparación con los electromecánicos. Los relés de estado sólido al conmutar no tienen contactos mecánicos y por lo tanto no se desgastan. Esto implica una vida útil superior a costa de un mayor precio.

En este trabajo en concreto, la velocidad de conmutación no es un factor crítico que sea necesario tener en cuenta. Asimismo, el precio de estos relés es algo superior, aunque, tampoco es excesivo. El principal motivo por lo que se han utilizado los relés electromecánicos es porque poseen 3 contactos: C, NO y NC. Esto ha sido idóneo para la aplicación ya que los SSR tienen únicamente dos contactos y por lo tanto habría que usar un relé más que diera paso al circuito de accionamiento manual o al circuito accionado mediante relés para subir o bajar la persiana automáticamente.

4.4 SENSOR DE TEMPERATURA LM35

Figura 5: Sensor LM35[12]

El sensor LM35 es un circuito integrado que proporciona una salida de voltaje proporcional a la temperatura.

Se caracteriza porque toma valores digitales y posee una salida lineal con la temperatura, incrementando el valor a razón de 10mV por cada grado Celsius. El rango

de medición es de -55ºC (-550mV) a 150ºC (1500mV), con una precisión a temperatura ambiente de ±0,5ºC[12]. Este sensor es preciso y económico, por lo tanto, es ideal para este tipo de proyectos.

El cálculo para obtener la temperatura final en grados Celsius es el siguiente:

$$\frac{Lectura\ sensor}{1023} * 5000[mV] * \frac{1}{10}[\frac{ºC}{mV}]$$

Donde la lectura del sensor puede ser de 1024 valores cuyo rango es de 0 a 1023. $5000[mV]$ son los 5V que se le aplica a una de las patillas del sensor. $\frac{1}{10}[\frac{ºC}{mV}]$ es debido a que la salida incrementa a razón de 10mV por cada grado Celsius.

Sensor de temperatura	LM35	TMP36	TC74	DHT11	DHT22
Precio[€]	0,9	6	5	1,9	5,31
Voltaje de operación[V]	Entre 4 y 30	Entre 2,7 y 5,5	Entre 2,7 y 5,5	Entre 3 y 5,5	Entre 3,3 y 6
Rango de temperaturas[ºC]	Entre -55 y 150	Entre -40 y 150	Entre -40 y 150	Entre 0 y 50	Entre -40 y 80
Precisión[ºC]	±0,5	±0,2	±2 y ±3	±2	±0,5
Conversión[mV/ºC]	10	10	-	-	-
Tiempo de respuesta(100%)[min]	4	8	-	-	-
Offset[V]	0	0,5	-	-	-
Resolución[bits]	-	-	8	8	16
Muestras/segundo	-	-	8	0,5	0,5

Tabla 4: Comparación de sensores de temperatura[13]

En la Tabla 4 se exponen las características mencionadas previamente. Por el contrario, el TMP36 también es bastante preciso, pero es caro y a partir de 125ºC pierde linealidad. Del mismo modo, el sensor elegido está regulado directamente en grados Celsius y no se debe realizar ningún tipo de conversión ni de calibración externa.

4.5 FOTORRESISTENCIA LDR (GL55)

Figura 6: Sensor LDR[14]

Un *fotorresistor*, o LDR es un dispositivo cuya resistencia varía en función de la luz recibida; más concretamente, su resistencia disminuye conforme aumenta la luz sobre él. Dicha variación se puede usar para medir a través de las entradas analógicas una estimación del nivel de luz.

No obstante, los fotorresistores no son muy precisos ya que tienen una gran dependencia a la temperatura, por lo que "no resultan adecuados para proporcionar una medición de la iluminancia, es decir, para servir como luxómetro" como menciona Luis Llamas en su página web[14].

Por último, se ha elegido este tipo de LDR además de por su bajo precio (en torno a 10 céntimos) por su facilidad de uso y configuración.

Specification	Type	Max. Voltage	Max. power	Environmental temp.	Spectrum peak value
Φ5 series	GL5516	150	90	-30~+70	540
	GL5528	150	100	-30~+70	540
	GL5537-1	150	100	-30~+70	540
	GL5537-2	150	100	-30~+70	540
	GL5539	150	100	-30~+70	540
	GL5549	150	100	-30~+70	540

Tabla 5:Características del sensor LDR[15]

4.6 POTENCIÓMETRO

Figura 7: Potenciómetro

En este proyecto se ha incorporado un potenciómetro de 1KΩ en serie con el LDR. Con ello se pretende regular de forma manual la lectura lumínica, que se efectúa a través del pin analógico. Posteriormente, esta información es tratada por el conversor analógico/digital del Arduino.

4.7 PERSIANA CON MOTOR ELÉCTRICO

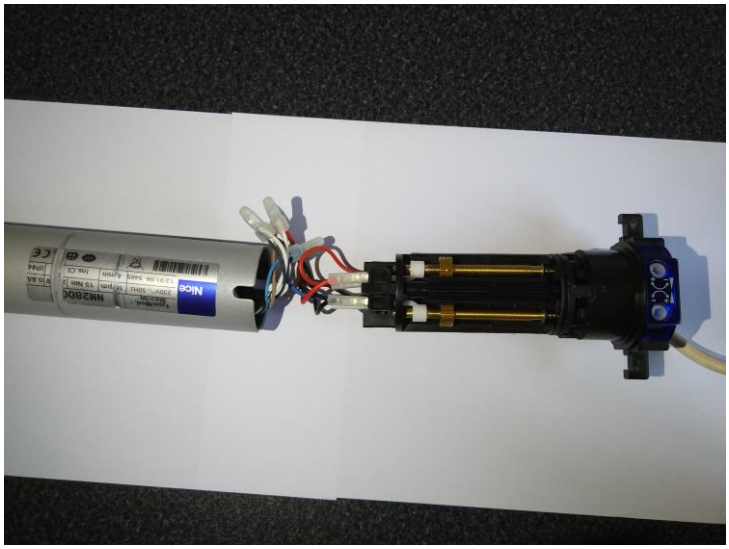

Figura 8: Motor eléctrico de la persiana

Una *persiana eléctrica* es simplemente una persiana accionada mediante un motor eléctrico. Además, posee unos finales de carrera ajustables encargados de cortar el suministro eléctrico cuando son pulsados.

Las principales características del motor de la persiana se encuentran en la pegatina de su carcasa, tal y como se muestra en la siguiente imagen.

Figura 9: Características del motor de la persiana

Esta persiana se encontraba anteriormente instalada en el domicilio debido al escaso espacio disponible en la pared para colocar una manivela (véase Figura 18 del Anexo 2).

4.8 CENTRALITA E20

Figura 10: Centralita E20

"El regulador E20 permite una cómoda introducción y visualización de valores de ajuste y de valores de medición de la instalación de calefacción en la vivienda del usuario. De esta manera se puede controlar y optimizar constantemente la instalación de calefacción" como se menciona en el manual de regulador de ambiente [16].

Además, haciendo uso de esta centralita, se puede efectuar la regulación y visualización de la temperatura del ambiente. También incluye conexiones opcionales como una sonda exterior para la visualización de la temperatura de la calle y la posibilidad de conectar un selector remoto de teléfono. Ambas opciones están implementadas, aunque se ha profundizado en la segunda.

5. DISEÑO Y DESARROLLO

Este punto versa sobre una descripción detallada del diseño y desarrollo del hardware. En primer lugar, se va profundizar en la parte material del sistema implementado. Una vez realizado esto, se procede al desarrollo del código donde se muestran los diferentes diagramas de flujo.

5.1 DISEÑO Y DESARROLLO HARDWARE

En cuanto al diseño y desarrollo del hardware se explica la forma en la que se han conectado los componentes, los problemas que han surgido y cómo se han solventado.

5.1.1 CONEXIÓN ENTRE EL MÓDULO GSM Y EL ARDUINO Y PUESTA EN MARCHA

Para llevar a cabo el proyecto se ha realizado una soldadura en la patilla "R13" del módulo GSM, tal y como menciona Raúl Moreno en el blog de Eduardo Lara[17]; con ello se ha conseguido poder encender mediante software el módulo aplicando un pulso de duración 2 segundos al pin 9 y así no tener que depender del botón del encendido manual. Esto resulta atractivo para el proyecto debido a que, si hay algún corte del suministro de la red eléctrica en la vivienda, el Arduino se reiniciará y el módulo podrá ser encendido sin necesidad de presionar dicho botón. Por tanto, no sería necesaria la presencia ni la acción humana para inicializarlo.

Igualmente, en la programación se ha tenido en cuenta tanto la forma de encendido como el reinicio (fallo en la alimentación del módulo GSM). En el caso de que dicho módulo estuviese encendido anteriormente no se le aplicaría el pulso, ya que este se apagaría.

Por otro lado, la tarjeta SIM900 se configura vía UART. Este dispositivo es el encargado de controlar los puertos y dispositivos en serie. Para lograr dicha comunicación serie, es necesario unir los pines 7 y 8, lo cual se ha llevado a cabo mediante el uso de puentes eléctricos.

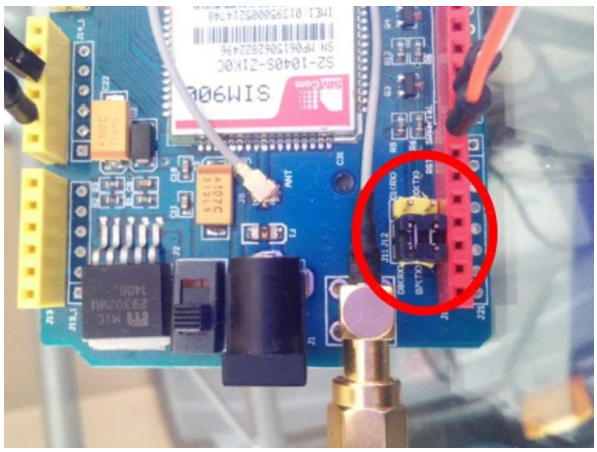

Figura 11: Colocación de los puentes eléctricos en el módulo GSM/GRPS[17]

Los pines 7 y 8 del módulo GSM/GPRS se han conectado al Arduino Mega a los pines 10 y 11. Además, también se ha unido el pin 9 del módulo GSM/GPRS al 9 del Arduino Mega y el GND de ambas tarjetas. El módulo GSM/GPRS se ha alimentado directamente de la salida de 5V del Arduino. No obstante, será insuficiente si únicamente se alimenta el Arduino con el USB. Este problema se resuelve usando una alimentación externa, que es exactamente como se ha solucionado en este trabajo ya que

se hace uso de una alimentación externa de corriente continua de 12V y 1A a través del conector Jack.

5.1.2 ESQUEMA ELÉCTRICO Y FUNCIONAMIENTO DE LA PERSIANA

En primer lugar, cabe mencionar que la persiana a automatizar poseía motor eléctrico, por tanto, hubo que modificar la instalación ya existente. Y con ello, se ha desarrollado un esquema eléctrico eficiente para el manejo de la persiana.

El primer problema que se planteó fue que alguien quisiera accionar manualmente la persiana mientras que la estuviera controlando el Arduino. Esto supondría que al motor se le ordenara girar tanto a favor de las agujas del reloj como en contra en un mismo instante de tiempo. Por ello, se adoptaron medidas de seguridad dando prioridad en modo manual al usuario que quiere accionar manualmente la persiana y en modo automático al Arduino (relés subir y bajar a la derecha de la Figura 12). Por un lado, se considera modo manual cuando el Arduino está totalmente desconectado, cuando acaba de encenderse, o también cuando se le ordena o mediante SMS o llamada perdida que suba o baje la persiana y ha finalizado dicha acción. Por otro lado, se considera modo automático únicamente cuando se le ordena mediante SMS o llamada perdida que cambie a modo automático.

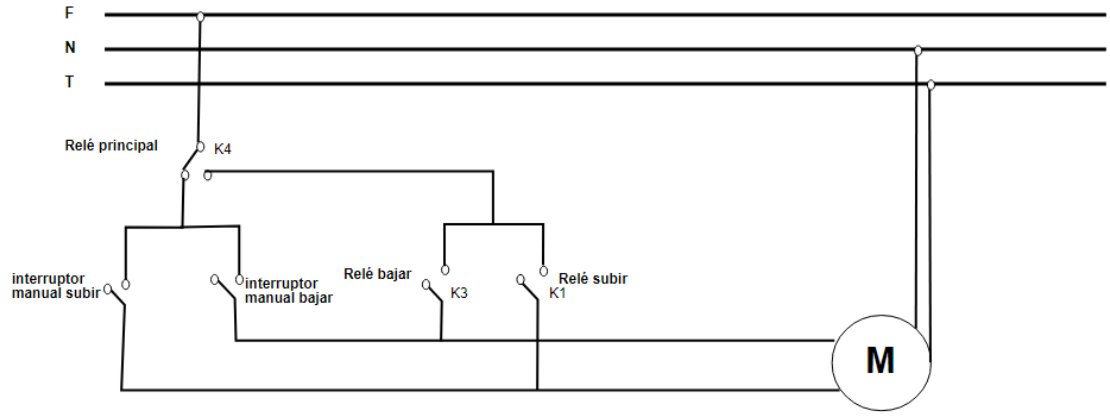

Figura 12: Esquema eléctrico persiana

Igualmente, también se ha estudiado el caso de que el Arduino se apagara, o se estropease el relé. En esta situación el relé se queda en su posición de "reposo" y por tanto la persiana podría controlarse de manera manual.

Las conexiones entre el interruptor manual y el motor de la persiana antes de la modificación se puede ver en la Figura 19 del Anexo 2.

Una persiana eléctrica en su funcionamiento normal se detiene mediante finales de carrera. Así pues, consta de dos finales de carrera, uno para cada dirección a la que gira. Este elemento al ser mecánico, sufre un desgaste. Por ello para alargar su vida útil, en la programación del código en el modo automático, se han tenido en cuenta las mediciones de tiempo de elevación y descenso de la persiana para que se detenga justo antes de tocarlo dejando así el final de carrera como un sistema de seguridad en caso de fallo del Arduino.

Como se menciona en el párrafo anterior, el motor gira durante un tiempo predefinido y no se actúa sobre los finales de carrera. Esto implica no saber con exactitud la posición de la persiana, ya que la forma idónea es acceder a los finales de carrera y detectar si están pulsados o no. En un principio se trató de hacer así, sin embargo, suponía un mayor coste tanto si se usaban finales de carrera externos como si se usaban los de la propia persiana. Además, también se complicaba en cuanto a la adición de los nuevos cables que tenían que acceder hasta los finales de carrera debido al reducido espacio del que se dispone como puede verse en la Figura 20 del Anexo 2.

5.1.3 PUESTA EN FUNCIONAMIENTO DE LA CENTRALITA E20

La posibilidad de conectar un selector remoto de teléfono ha sido consultada tanto en la hoja de especificaciones técnicas del regulador como vía email al fabricante. Una vez se tiene claro cómo realizar las conexiones se abre la tapa de la centralita y se colocan los cables, tal y como se ve en las Figura 21 y Figura 22 del Anexo 2.

Estando inicialmente la calefacción en modo apagado, cortocircuitando los pines 3 y 4 de la centralita como puede verse en la Figura 23 del Anexo 2, el termostato pasaría a modo calefacción y realizaría las funciones configuradas en temperatura deseada ambiente I. Se sabe que el modo calefacción está activo cuando parpadea cualquiera de los símbolos que representan el modo en el que se encuentra la centralita (modo apagado, diurno, nocturno…).

Este cortocircuito se va a realizar mediante un relé el cual será el encargado de cambiar a modo calefacción la centralita una vez conmute.

Por otro lado, la programación de la temperatura del termostato es la que pone en marcha la calefacción. Por lo tanto, aunque el Arduino haya dado orden de encender la calefacción para alcanzar una temperatura superior a la configurada previamente en la

centralita, el cortocircuito de los pines 3 y 4 se mantendrá; sin embargo, la calefacción permanecerá apagada.

5.1.4 ESQUEMA ELÉCTRICO-ELECTRÓNICO DEL CIRCUITO

A continuación, en la Figura 13 se muestra el esquema final del sistema.

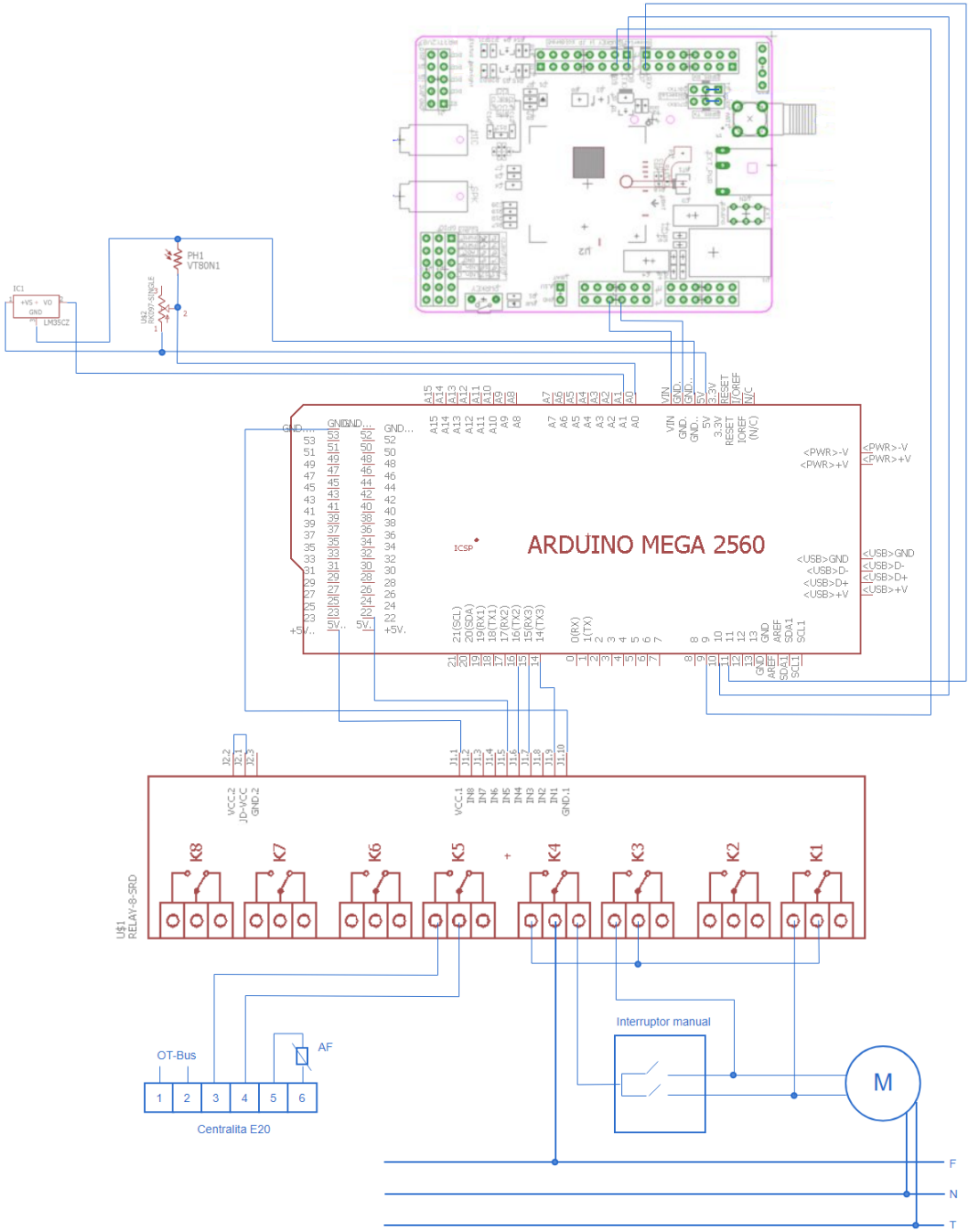

Figura 13: Esquema eléctrico-electrónico del circuito

5.2 DISEÑO Y DESARROLLO SOFTWARE

En este apartado se expone el desarrollo del código (véase Anexo 3).

Cabe destacar que el lenguaje de la programación de Arduino es C++. Asimismo, la estructura de Arduino está claramente diferenciada en dos bloques o funciones fundamentales que son: setup() y void loop().

5.2.1 DIAGRAMA DE FLUJO DE LA FUNCIÓN LOOP

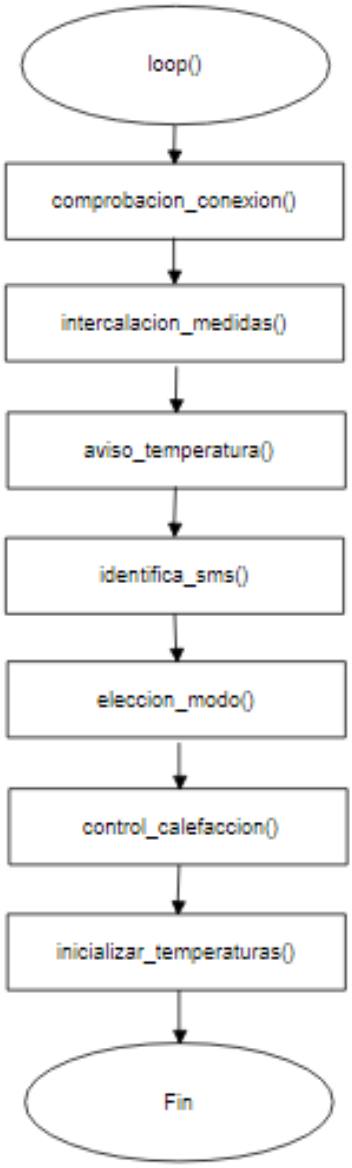

Figura 14: Diagrama de flujo de la función loop()

5.2.2 DIAGRAMA DE FLUJO DE LA FUNCIÓN ENVIAR COMANDOS AT

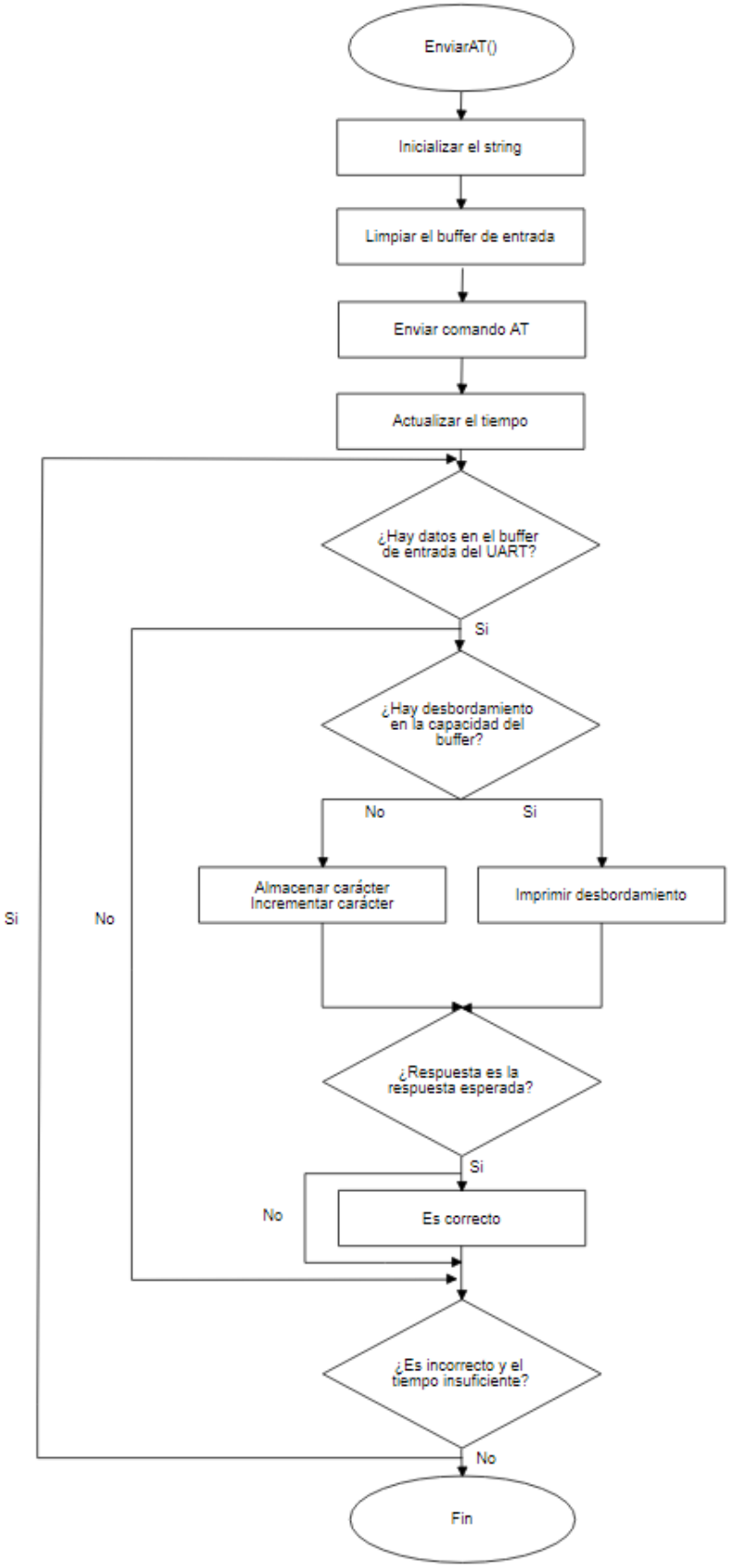

Figura 15: Diagrama de flujo de la función EnviarAT()

6. PRESUPUESTO TOTAL DEL PROTOTIPO

En este punto se hace un balance económico en cuanto al coste total de los materiales usados en el proyecto, el cual puede resumirse en la siguiente tabla.

Componentes	Precio (€)
Arduino Mega 2560	12,00
Módulo GSM/GRPS SIM900	24,00
Módulo de 8 relés electromecánicos y optoacoplados	8,00
Sensor de temperatura LM35	0,90
Fotorresistencia LDR (GL55)	0,10
Potenciómetro 1KΩ	0,34
Protoboard	3,00
Cableado	14,00
Tarjeta SIM	3,00
TOTAL	65,34

Tabla 6: Presupuesto total del prototipo

A este presupuesto hay que sumarle el coste que conlleva enviar cada mensaje de texto, aunque hoy día existe una gran variedad de tarifas asequibles económicamente, que pueden incluir packs de cientos de mensajes o incluso SMS ilimitados. Todo esto va en función del uso que se le vaya a dar y de la compañía telefónica que se tenga contratada.

7. CONCLUSIÓN Y MEJORAS FUTURAS

Desde mi punto de vista, este trabajo final ha englobado una importante cantidad de conceptos que han sido estudiados previamente a lo largo del Grado. Además, se han podido poner en práctica los conocimientos de diferentes asignaturas, tales como la automatización industrial, instalaciones eléctricas, instrumentación electrónica o fundamentos de informática.

Además, la realización de este trabajo implica una búsqueda profunda en Internet, sin embargo, no de la manera en la que habitualmente estaba acostumbrado, ya que la información tenía que ser cuidadosamente tratada y como es obvio, fiable.

Por otro lado, se puede afirmar que se han cumplido tanto el objetivo general de control de la temperatura de una vivienda de manera remota comunicándose a través del

uso de la red de telefonía móvil como los específicos planteados en el apartado 2 de la memoria.

Además, se ha pensado una serie de mejoras futuras como crear una aplicación móvil para una mejor visualización y manejo para no tener que recordar las palabras exactas de los mensajes de texto, diseñar un SAI (Sistema de Alimentación Ininterrumpido), obtener la hora mediante el módulo GSM para una mayor precisión en cuanto a las temperaturas máximas y mínimas del día, ampliar el código para el control de más ventanas, que funcionarían en modo automático de forma individual y aleatoria (para que cada vez que suban y bajen no sigan siempre el mismo orden de apertura). Con ello se evita un consumo elevado en un instante de tiempo determinado. A su vez, también simularía una sensación de habitabilidad en la vivienda y por lo tanto la persuasión de posibles delincuentes.

Para concluir, pienso firmemente que, sin la formación recibida a lo largo de estos últimos años, no hubiera sido capaz de comprender de igual manera todos los conceptos aplicados en el trabajo ni haber conseguido resolver todos los problemas de una forma eficiente.

8. BIBLIOGRAFÍA

[1] Manuel Delgado Crespo, «Arduino en español: Arduino Mega 2560», *Arduino en español*, 29-mar-2016.

[2] Francisco Moya Fernández, «La Raspberry Pi · Taller de Raspberry Pi», *La Raspberry Pi*, 17-ene-2017. [En línea]. Disponible en: https://franciscomoya.gitbooks.io/taller-de-raspberry-pi/content/es/intro/rpi.html. [Accedido: 02-ago-2017].

[3] «Datasheet de Arduino UNO i2c ATmega 328», *Datasheet de Arduino UNO i2c ATmega 328*. [En línea]. Disponible en: http://www.tiendaarduino.com/datasheet/arduino-uno_atmega328.htm#.WgtGrUribIU. [Accedido: 02-ago-2017].

[4] «Datasheet de Arduino MEGA 2560», *Datasheet de Arduino Mega 2560*. [En línea]. Disponible en: http://www.tiendaarduino.com/datasheet/arduino-mega_2560.htm#.WgtHJUribIU. [Accedido: 11-ago-2017].

[5] Emiliano Gioia, «Shield GSM GPRS SIM900 Celular Arduino con Antena», *Fematech*.

[6] elnaanox, «Diferencias entre GSM, GPRS y 3G», *Diferencias entre GSM, GPRS y 3G*, 09-may-2011. [En línea]. Disponible en: https://www.taringa.net/posts/ciencia-educacion/10530547/Diferencias-entre-GSM-GPRS-y-3G.html. [Accedido: 24-ago-2017].

[7] «Conjunto de comandos Hayes», *Wikipedia, la enciclopedia libre*. 19-may-2017.

[8] designthemes, «ANEXO COMANDOS AT PARA GSM/GPRS Y GPS | Tutoriales Arduino», *prometec*.

[9] designthemes, «Módulos de relés y Opto acopladores | Tutoriales Arduino», *prometec*.

[10] «Optoacoplador», *Wikipedia, la enciclopedia libre*. 21-feb-2017.

[11] «Módulo de 8 relés – Tutoriales Arduino», *prometec*.

[12] Luis Llamas, «Medir temperatura con Arduino y sensor LM35», *Luis Llamas*, 15-jul-2015.

[13] «Sensor de temperatura, escoge el mejor para tus proyectos con Arduino», *Programar fácil con Arduino*, 06-jun-2016.

[14] Luis Llamas, «Medir nivel de luz con Arduino y fotoresistencia LDR (GL55)», *Luis Llamas*.

[15] «GL55 Series Photoresistor.pdf».

[16] «CENTRALITA-E20.pdf».

[17] Eduardo Lara, «SIM900 GSM Shield con Arduino UNO GPRS SD 2G», *HETPRO/TUTORIALES*, 13-oct-2015.

[18] «Cobertura movil | Simyo». [En línea]. Disponible en: /mapa-cobertura.html. [Accedido: 01-sep-2017].

[19] «UISRAEL-EC-ELDT-378.242-29.pdf».

[20] «Clima Bronchales: Temperatura, Climograma y Tabla climática para Bronchales - Climate-Data.org». [En línea]. Disponible en: https://es.climate-data.org/location/283082/. [Accedido: 07-sep-2017].

ANEXOS

ANEXOS 1: ESPECIFICAIONES TÉCNICAS DE LOS COMPONENTES

- Especificaciones técnicas del Arduino Mega

http://www.mantech.co.za/datasheets/products/A000047.pdf

- Especificaciones técnicas del Módulo GSM/GRPS

http://linksprite.com/wiki/index.php5?title=SIM900_GPRS/GSM_Shield

- Comandos AT de la SIM900

http://simcom.ee/documents/SIM900/SIM900_AT%20Command%20Manual_V1.11.pdf

- Especificaciones técnicas del Módulo de 8 relés electromecánicos

https://www.prometec.net/producto/modulo-8-reles/

- Especificaciones técnicas del sensor LM35

http://www.ti.com/lit/ds/symlink/lm35.pdf

- Especificaciones técnicas del sensor LDR

http://akizukidenshi.com/download/ds/senba/GL55%20Series%20Photoresistor.pdf

- Especificaciones técnicas de la centralita E20

https://www.suministrosguallar.com/documentacion/tarifas/domusa/instrucciones-de-instalacion/CENTRALITA-E20.pdf

ANEXOS 2: TABLAS Y FIGURAS

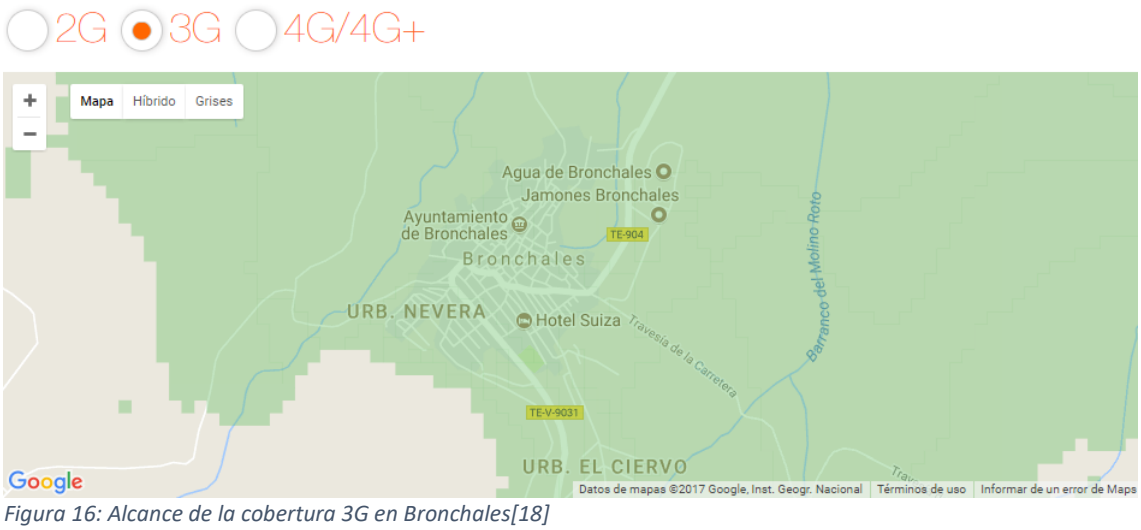

Figura 16: Alcance de la cobertura 3G en Bronchales[18]

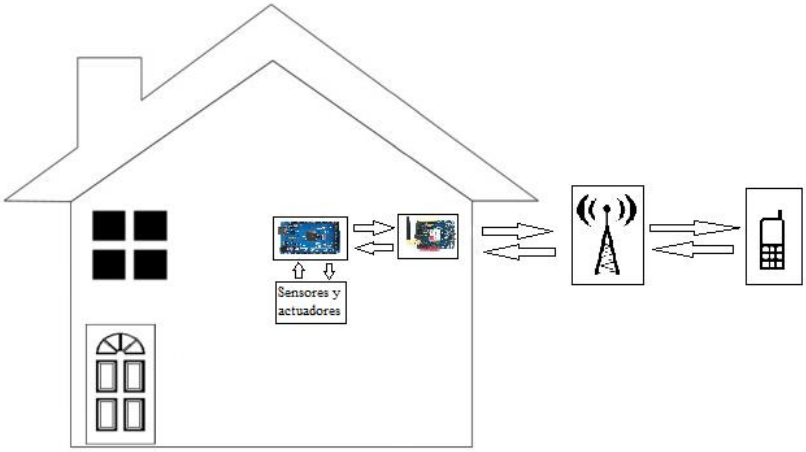

Figura 17: Esquema de comunicación[19]

TABLA CLIMÁTICA // DATOS HISTÓRICOS DEL TIEMPO BRONCHALES

	Enero	Febrero	Marzo	Abril	Mayo	Junio	Julio	Agosto	Septiembre	Octubre	Noviembre	Diciembre
Temperatura media (°C)	-0.2	0.8	3.4	5.8	9.5	14.3	17.8	17.5	14	8.4	3.8	1
Temperatura mín. (°C)	-4.3	-4	-1.8	0.4	4.1	8.3	11.3	11.1	8.1	3.1	-1.1	-3.3
Temperatura máx. (°C)	4	5.6	8.6	11.2	14.9	20.3	24.3	23.9	20	13.7	8.7	5.4
Temperatura media (°F)	31.6	33.4	38.1	42.4	49.1	57.7	64.0	63.5	57.2	47.1	38.8	33.8
Temperatura mín. (°F)	24.3	24.8	28.8	32.7	39.4	46.9	52.3	52.0	46.6	37.6	30.0	26.1
Temperatura máx. (°F)	39.2	42.1	47.5	52.2	58.8	68.5	75.7	75.0	68.0	56.7	47.7	41.7
Precipitación (mm)	38	39	55	56	82	69	35	39	53	51	47	48

Tabla 7: Datos históricos temperaturas en Bronchales[20]

Figura 18: Espacios persiana

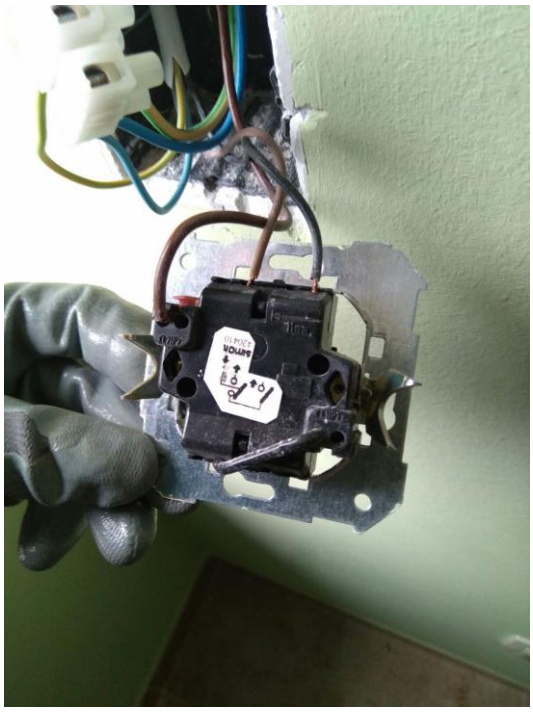

Figura 19: Conexiones previas al trabajo entre interruptor y persiana

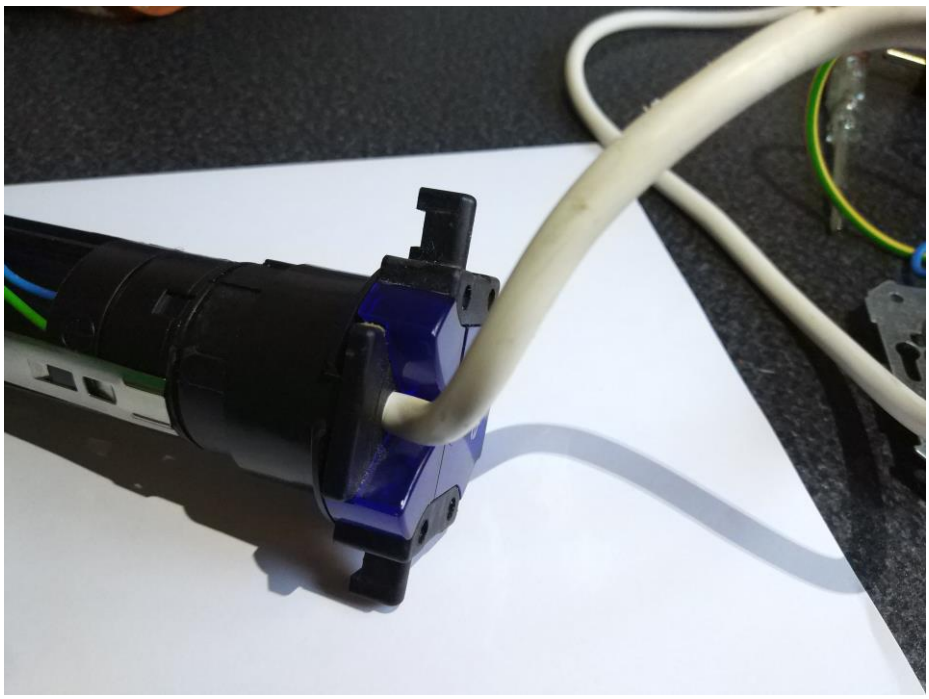

Figura 20: Espacio reducido para más cableado en el motor de la persiana

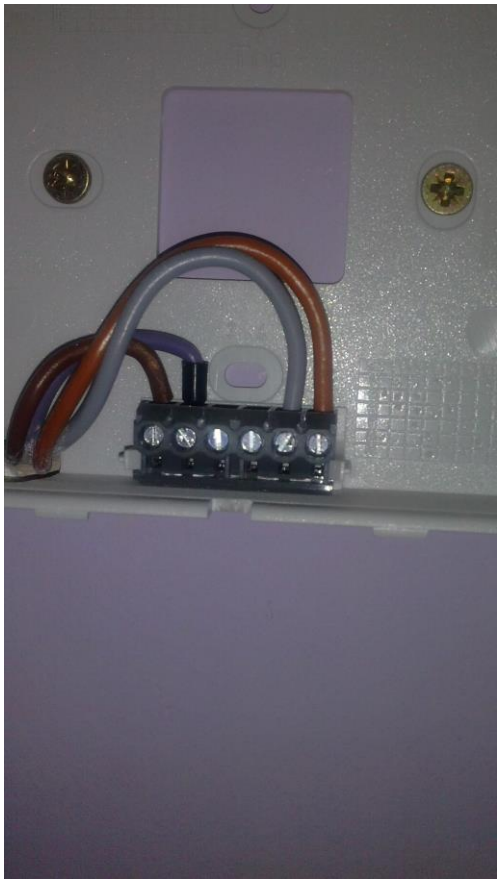

Figura 21: Conexión centralita previa al trabajo

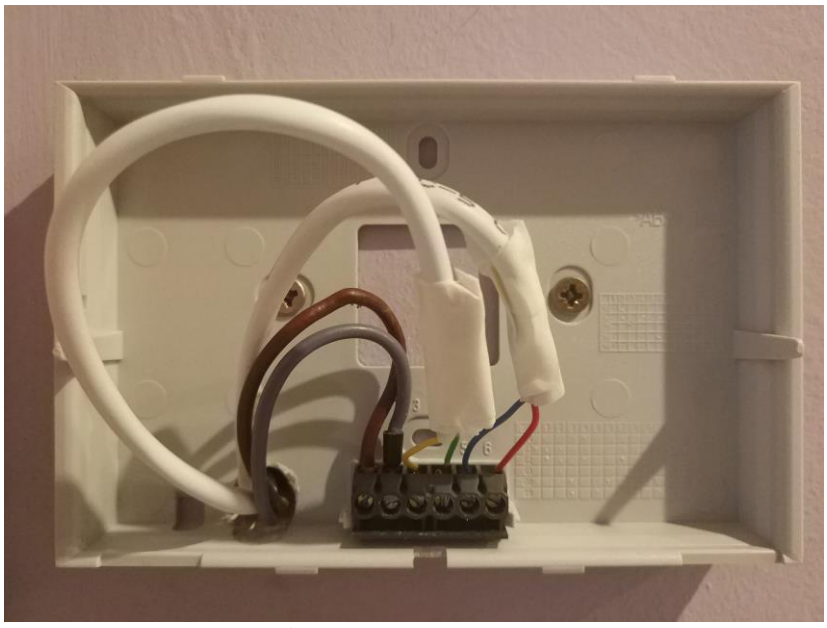

Figura 22: Conexiones centralita

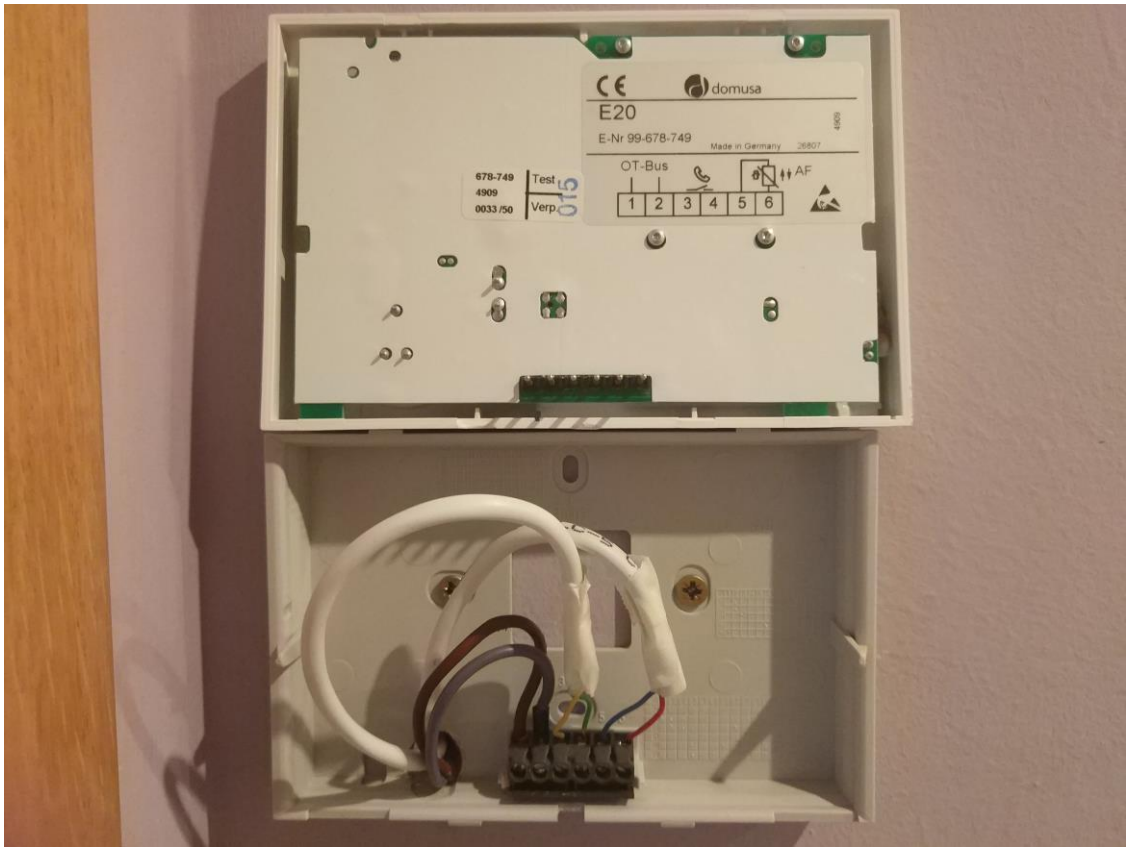

Figura 23: Esquema de conexión de la centralita

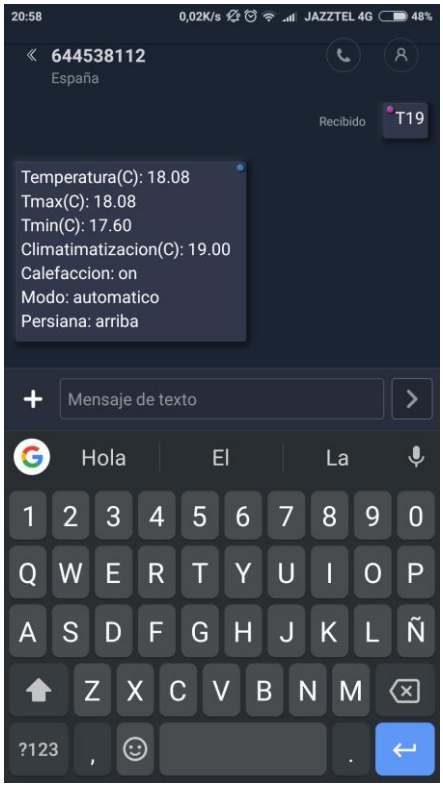

Figura 24: Información enviada y recibida desde el Arduino

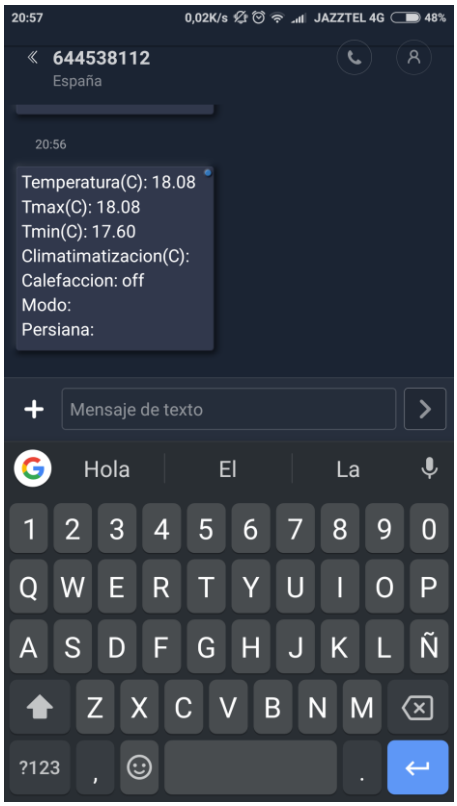

Figura 25: Información recibida desde el Arduino

ANEXOS 3: CÓDIGO ARDUINO

```
/*********************************************************************************
 * ESCUELA UNIVERSITARIA POLITECNICA DE TERUEL                                   *
 * GRADO EN INGENIERIA ELECTRONICA Y AUTOMATICA                                  *
 * TRABAJO DE FINAL DE GRADO                                                     *
 *                                                                               *
 * Sistema remoto para el control automatico de temperatura de una vivienda      *
 *                                                                               *
 * Autor: Diego Polo Perez                                                       *
 * Fecha: Noviembre 2017                                                         *
 *                                                                               *
 * Breve descripcion: Programa encargado del control de la calefaccion y         *
 * persiana en funcion de las ordenes de la sim900 y de los sensores             *
 *                                                                               *
 * Acentos omitidos                                                              *
 *                                                                               *
 *********************************************************************************/
#include <SoftwareSerial.h>

SoftwareSerial SIM900(10, 11); // Configura el puerto serial para el SIM900. Para el
Arduino MEGA utilizar pines 10 y 11

// Configuracion de entradas y salidas del Arduino
int ldr = A0;                   //El LDR esta conectador en el pin A0 del Arduino
int lm35 = A1;                  //El LM35 esta conectador en el pin A1 del Arduino
int subir_persiana = 14;        //La salida del pin 14 hace girar al motor enrollando la
persiana
int bajar_persiana = 15;        //La salida del pin 15 hace girar al motor desenrollando
la persiana
int control_arduino = 16;         //La salida del pin 16 alimenta el control automático
del sistema
int encender_calefaccion = 22;  //La salida del pin 22 enciende el modo calefaccion

char incoming_char = 0; //Variable que guarda los caracteres que envia el SIM900
String mensaje = "";    //Cadena mensaje inicialmente vacia

//Deteccion de llamada
const char SONANDO[] = "RING";        //caso de descolgar llamada
const char COLGADO[] = "NO CARRIER";  //caso de colgar llamada

//Ordenes a escribir con el telefono movil
const char AUTO[] = "AUTOMATICO";
const char INFO[] = "INFO";
const char SUBIR[] = "SUBIR";
const char BAJAR[] = "BAJAR";
String TEMPERATURAS_ELEGIDA[] = {"T-9", "T-5", "T00", "T15", "T16", "T17", "T18", "T19",
"T29"}; //Temperaturas de climatizacion

int numero_posibles_temperaturas = 9; //Numero de TEMPERATURAS_ELEGIDA autorizadas

char num_telefono[9];  //Almacena los 9 digitos de un telefono

String agenda[] = {"610976930", "978604811", "644538112" , "689275192", "648193018",
"659852174"}; //Cadena con los numeros autorizados
int numeros_autorizados = 6; //Cantidad de numeros de telefono autorizados
char codigo_llamada[14] = "ATD648193018;"; //Inicializacion de un numero de telefono
para el caso de temperatura extrema
char codigo_mensaje[22]; //= "AT+CMGS=\"648193018\""; //descomentar si se quiere que
cuando haya una temperatura extrema envie sms

int temperatura_elegida = 0;
int temperatura_coincide = 0;

float valor_sensor_temperatura_anterior = 0;
float valor_temperatura_real = 0;
float tmax = -99.00;
float tmin = 99.00;

int luz_coincide = 0;
int valor_sensor_luz_anterior = 0;
int valor_luz_real = 0;

int calefaccion = 0;
int tono = 0;
```

```
int contador_aviso = 0;
int contador_llamadas = 0;
int opciones = 0;
int valor_sensor_luz = 0;
int valor_limite_sensor = 1000;      //Con este valor se ajusta la sensibilidad del ldr
const int ajuste_luminosidad = 10;  //Histeresis de luminosidad
const int ajuste_temperatura = 2;   //Histeresis de temperatura
bool persiana_subida_automatico = false;
bool persiana_bajada_automatico = false;
bool persiana_subida = false;
bool persiana_bajada = false;
int modo = 0;
const int automatico = 1;
const int manual = 2;
const int subir = 1;
const int bajar = 2;
int contador_primera_posicion = 0;

char aux_str[50];
int luz_temperatura = 0;
unsigned long tiempo_aviso_anterior = 0;
unsigned long tiempo_comprobacion_anterior = 0;
unsigned long tiempo_dia_anterior = 0;

bool contador_calefaccion_apagada = false;
bool contador_calefaccion_encendida = false;
int temperatura_variable = 0;


void setup()
{
  pinMode(subir_persiana, OUTPUT);
  pinMode(bajar_persiana, OUTPUT);
  pinMode(control_arduino, OUTPUT);
  pinMode(encender_calefaccion, OUTPUT);
  pinMode(ldr, INPUT);
  pinMode(lm35, INPUT);
  digitalWrite (subir_persiana, HIGH);
  digitalWrite (bajar_persiana, HIGH);
  digitalWrite (control_arduino, HIGH);
  digitalWrite (encender_calefaccion, HIGH);
  SIM900.begin(19200); //Configura velocidad del puerto serie para el SIM900
  Serial.begin(19200); //Configura velocidad del puerto serie del Arduino
  delay(1000);
  Serial.println("Iniciando...");
  encender_gsm();
  iniciar_gsm();
}


void loop()
{
  comprobacion_conexion();    //Comprueba si el sim900 esta conectado a la red
  intercalacion_medidas();    //Omite el primer valor de medicion de la temperatura
  aviso_temperatura();        //Avisa si la temperatura es extrema
  identifica_sms_y_llamada(); //Identifica el mensaje o los tonos de llamada
  eleccion_modo();            //Elige entre modo manual y automatico
  control_calefaccion();      //Climatizacion
  reinicio_temperaturas();    //Reinicio diario de las temperaturas maximas y minimas
}

void comprobacion_conexion()
{
  const int tiempo_comprobacion = 10000;     //Si se sube a mas de 32767, emplear const
long
  if (millis() - tiempo_comprobacion_anterior >= tiempo_comprobacion){
    while (enviar_AT("AT", "OK", 2000) == 0) //comprueba la conexion a la red
    {
      reiniciar_gsm();
      iniciar_gsm();
    }
    tiempo_comprobacion_anterior = millis();
  }
}
```

```
void intercalacion_medidas()
{
  if (luz_temperatura < 2)
  {
    medir_temperatura();
    luz_temperatura++;
  }
  else
  {
    medir_luz();
    luz_temperatura = 0;
  }
}

void reinicio_temperaturas()
{
  const long tiempo_dia = 86400000;  //un dia 86400000 milisegundos
  if (millis() - tiempo_dia_anterior >= tiempo_dia)
  {
    tmax = -99.00;
    tmin = 99.00;
    tiempo_dia_anterior = millis();
    Serial.println(tmax);
  }
}

void medir_temperatura()
{
  int value = analogRead(lm35);                        //Lectura del sensor lm35
  float millivolts = (value / 1023.0) * 5000;          //conversor a milivoltios
  float valor_sensor_temperatura = millivolts / 10;    //temperatura en grados
  if (luz_temperatura != 0)
  {
    if ((valor_sensor_temperatura == valor_sensor_temperatura_anterior))
    {
      temperatura_coincide++;
    }
    else
    {
      temperatura_coincide = 0;
    }
    if (temperatura_coincide == 10)
    {
      temperatura_coincide = 0;
      valor_temperatura_real = valor_sensor_temperatura;
      if (valor_temperatura_real > tmax)
      {
        tmax=valor_temperatura_real;
      }
      if (valor_temperatura_real < tmin)
      {
        tmin=valor_temperatura_real;
      }

    }
    valor_sensor_temperatura_anterior = valor_sensor_temperatura;
  }
}

void aviso_temperatura()
{
  const int tiempo = 20000;     // Espera 20 segundos desde que cambia de modo hasta que
avisa. Si se sube a mas de 32767, emplear const long
  unsigned long tiempo_real = 0;

  if   (((valor_temperatura_real   <   1.00)   ||   (valor_temperatura_real   >   31.00))   &&
(valor_temperatura_real != 0.00))
  {
    if (contador_aviso == 0)
    {
      tiempo_aviso_anterior = millis();
      contador_aviso++;
    }

    tiempo_real = tiempo_aviso_anterior + tiempo;
```

```
    if ((tiempo_real < millis())   && (contador_llamadas == 0))   //Para que unicamente
haga una perdida
    {
      Serial.println("Temperatura extrema");
      //enviar_sms();
      llamar();
      contador_llamadas++;
    }
  }
}

void medir_luz()
{
  valor_sensor_luz = analogRead(ldr); //lee el valor del sensor ldr
  if (valor_sensor_luz == valor_sensor_luz_anterior)
  {
    luz_coincide++;
  }
  else
  {
    luz_coincide = 0;
  }
  if (luz_coincide == 3)
  {
    luz_coincide = 0;
    valor_luz_real = valor_sensor_luz;
  }
  valor_sensor_luz_anterior = valor_sensor_luz;
}

void identifica_sms_y_llamada()
{

  char elegido[4];
  char elegido_nuevo[3];
  int llamada = 0;
  int fin_llamada = 0;
  int sms_automatico = 0;
  int sms_informacion = 0;
  int sms_subir = 0;
  int sms_bajar = 0;
  int sms_controlar_calefaccion = 0;

  //vaciamos cadena
  elegido_nuevo[0]='\0';
  elegido_nuevo[1]='\0';
  elegido_nuevo[2]='\0';

  if (SIM900.available() > 0)          // SIM900 disponible
  {
    incoming_char = SIM900.read(); //Guardamos el carácter del GPRS
    mensaje = mensaje + incoming_char ; // Añadimos el carácter leído al mensaje

    //Da un valor positivo a la variable cuando mensaje coincide con la cadena enviada
por el telefono
    llamada = mensaje.indexOf(SONANDO);
    fin_llamada = mensaje.indexOf(COLGADO);
    sms_automatico = mensaje.indexOf(AUTO);
    sms_informacion = mensaje.indexOf(INFO);
    sms_subir = mensaje.indexOf(SUBIR);
    sms_bajar = mensaje.indexOf(BAJAR);


    for (int j = 0; j < numero_posibles_temperaturas; j++)
    {
      sms_controlar_calefaccion = mensaje.indexOf(TEMPERATURAS_ELEGIDA[j]);

      if ((sms_automatico >= 0) || (sms_informacion >= 0) || (sms_subir >= 0) ||
(llamada >= 0) || (fin_llamada >= 0)
      || (sms_bajar >= 0) || (sms_controlar_calefaccion >= 0)) //si existe algun mensaje
      {
        for ( int i = 0; i < numeros_autorizados; i++)
        {
          int pos = mensaje.indexOf(agenda[i]);      //pos devolvera -1 si no encuentra
coincidencias
```

```
          //Identificamos el numero de telefono para enviar los mensajes o llamadas al
ultimo numero que se comunico
          if  ((tono  ==  0)  ||  (sms_automatico  >=  0)  ||  (sms_informacion  >=  0)  ||
(sms_subir >= 0) ||
          (sms_bajar >= 0) || (sms_controlar_calefaccion >= 0))
          {
           agenda[i].toCharArray(num_telefono,agenda[i].length()+1);
          }
          //Debido a que NO CARRIER no lleva asociado ningun numero
          //Todas llamadas (RINGS) y sms pasaran por pos>=0
          if ((pos >= 0) || ((fin_llamada >= 0) && (tono > 0)))
          {
            if (sms_controlar_calefaccion >= 0)
            {
              TEMPERATURAS_ELEGIDA[j].toCharArray(elegido,
TEMPERATURAS_ELEGIDA[j].length()+1);
              //Obtenemos el numero entero descartando el caracter T
              elegido_nuevo[0] = elegido[1];
              elegido_nuevo[1] = elegido[2];
              //Conversion de cadena a entero
              temperatura_elegida = atoi(elegido_nuevo);
              Serial.println(elegido_nuevo);
              Serial.println(temperatura_elegida);
            }
            // Guarda el numero de cualquier mensaje recibido o del primer tono
            if ((sms_automatico >= 0) || (sms_informacion >= 0) || (sms_subir >= 0) ||
            (sms_bajar >= 0) || (sms_controlar_calefaccion >= 0) || tono == 0)
            {
            sprintf(codigo_mensaje, "AT+CMGS=\"%s\"",num_telefono);
            sprintf(codigo_llamada, "ATD%s;",num_telefono);
            Serial.println(codigo_mensaje);
            Serial.println(codigo_llamada);
            }
            contador_aviso = 0;
            contador_llamadas = 0;
            while (llamada >= 0)
            {
              mensaje = "";
              tono++;
              llamada = -1;
            }
            if ((fin_llamada >= 0) || (sms_automatico >= 0) || (sms_informacion >= 0) ||
(sms_subir >= 0)
            || (sms_bajar >= 0) || (sms_controlar_calefaccion >= 0))
            {
              mensaje = "";
              if ((sms_automatico >= 0) || (tono == 1))
              {
                Serial.println("\nautomatico");
                contador_primera_posicion = 0;
                persiana_subida_automatico = 0;
                persiana_bajada_automatico = 0;
                tono = 0;
                modo = automatico;
                sms_automatico = -1;
              }
              if ((sms_informacion >= 0) || (tono == 2))
              {
                tono = 0;
                enviar_sms();
                sms_informacion = -1;
              }
              if ((sms_subir >= 0) || (tono == 3))
              {
                Serial.println("\nmanual");
                tono = 0;
                modo = manual;
                opciones = subir;
                sms_subir = -1;
              }
              if ((sms_bajar >= 0) || (tono == 4))
              {
                Serial.println("\nmanual");
                tono = 0;
                modo = manual;
                opciones = bajar;
```

```
                sms_bajar = -1;
              }
              if (sms_controlar_calefaccion >= 0)
              {
                tono = 0;
                temperatura_variable = temperatura_elegida;
                sms_controlar_calefaccion = -1;
              }
              if (tono >= 7)tono = 0;
            }
          }
        }
      }
    }
  }
}

void eleccion_modo()
{
  switch (modo)
  {
    case automatico: //modo automático

    digitalWrite (control_arduino, LOW);

    if  ((valor_limite_sensor  >  valor_luz_real)  &&  !(persiana_subida_automatico)  &&
(valor_luz_real != 0)) //Si es mayor que el valor limite
    {
      if (contador_primera_posicion == 0)
      {
        Serial.println("Subiendo persiana");
        digitalWrite (subir_persiana, LOW);
        delay(15000);   //tiempo en movimiento de la persiana
        digitalWrite (subir_persiana, HIGH);
        Serial.println("Persiana subida");
        persiana_subida_automatico = true;
        persiana_bajada_automatico = false;
        persiana_subida = true;
        persiana_bajada = false;

        //bajar la persiana para que no toque el final de carrera y asi evitar desgaste
        digitalWrite (bajar_persiana, LOW);
        delay(400);   //0.4 segundos
        digitalWrite (bajar_persiana, HIGH);
        delay(300);
      }
      if (contador_primera_posicion != 0)
      {
        SubiendoPersianaAutomatico();
      }
      contador_primera_posicion++;
    }
    if  ((valor_limite_sensor  <  valor_luz_real)  &&  !(persiana_bajada_automatico)  &&
(valor_luz_real != 0))
    {
      if (contador_primera_posicion == 0)
      {
        Serial.println("bajando persiana");
        digitalWrite (bajar_persiana, LOW);
        delay(15000);   //17000 ms
        digitalWrite (bajar_persiana, HIGH);
        Serial.println("Persiana bajada");
        persiana_bajada_automatico = true;
        persiana_subida_automatico = false;
        persiana_bajada = true;
        persiana_subida = false;

        //subir la persiana 1 segundo para que no toque el final de carrera
        digitalWrite (subir_persiana, LOW);
        delay(400);   //0.4 segundos
        digitalWrite (subir_persiana, HIGH);
        delay(300);
      }
      if (contador_primera_posicion != 0)
      {
```

```
        BajandoPersianaAutomatico();
      }
      contador_primera_posicion++;
    }
    break;

    case manual: //modo manual
    if (opciones == subir)
    {
      SubiendoPersianaManual();
      opciones = 0;
    }
    if (opciones == bajar)
    {
      BajandoPersianaManual();
      opciones = 0;
    }
    break;
    default:
    manual;
    break;
  }
}

void SubiendoPersianaManual()
{
  Serial.println("Subiendo persiana");
  digitalWrite (control_arduino, LOW);
  delay(300);
  digitalWrite (subir_persiana, LOW);
  delay(14000);   //tiempo en movimiento de la persiana
  digitalWrite (subir_persiana, HIGH);
  delay(300);
  digitalWrite (control_arduino, HIGH);
  Serial.println("Persiana subida");
  persiana_subida = true;
  persiana_bajada = false;

}

void BajandoPersianaManual()
{
  Serial.println("bajando persiana");
  digitalWrite (control_arduino, LOW);
  delay(300);
  digitalWrite (bajar_persiana, LOW);
  delay(14000);   //tiempo en movimiento de la persiana
  digitalWrite (bajar_persiana, HIGH);
  delay(300);
  digitalWrite (control_arduino, HIGH);
  Serial.println("Persiana bajada");
  persiana_bajada = true;
  persiana_subida = false;
}

void SubiendoPersianaAutomatico()
{
  Serial.println("Subiendo persiana");
  digitalWrite (subir_persiana, LOW);
  delay(12200);   //tiempo en movimiento de la persiana
  digitalWrite (subir_persiana, HIGH);
  Serial.println("Persiana subida");
  persiana_subida_automatico = true;
  persiana_bajada_automatico = false;
  persiana_subida = true;
  persiana_bajada = false;
  valor_limite_sensor  =  valor_limite_sensor  +  ajuste_luminosidad;       //Histeresis
luminosidad
  // Al aumentar el valor limite permite que no cambie de estado
  //la persiana ante pequeñas variaciones de iluminación
}

void BajandoPersianaAutomatico()
{
  Serial.println("bajando persiana");
  digitalWrite (bajar_persiana, LOW);
```

```
  delay(12200);   //tiempo en movimiento de la persiana
  digitalWrite (bajar_persiana, HIGH);
  Serial.println("Persiana bajada");
  persiana_bajada_automatico = true;
  persiana_subida_automatico = false;
  persiana_bajada = true;
  persiana_subida = false;
  valor_limite_sensor  =  valor_limite_sensor  -  ajuste_luminosidad;      //Histeresis
luminosidad
  // Al aumentar el valor_limite permite que no cambie de estado
  //la persiana ante pequeñas variaciones de iluminación
}

int enviar_AT(char* comando_AT, char* respuesta_correcta, unsigned int tiempo_AT)
{
  int x = 0;
  bool correcto = false;
  char respuesta[159];
  unsigned long anterior;

  memset(respuesta, '\0', 100); // Inicializa el string
  delay(100);
  while ( SIM900.available() > 0) SIM900.read(); // Limpia el buffer de entrada
  SIM900.println(comando_AT); // Envia el comando AT
  x = 0;
  anterior = millis();  //milisegundos desde que se enciende Arduino
  // Espera una respuesta
  do
  {
    // si hay datos el buffer de entrada del UART lee y comprueba la respuesta
    if (SIM900.available() != 0)
    {
      //Comprueba que no haya desbordamiento en la capacidad del buffer
      if (x < 158) {
        respuesta[x] = SIM900.read();
        x++;
      }
      else Serial.println("Desbordamiento!");
      // Comprueba si la respuesta del modulo es la esperada
      if (strstr(respuesta, respuesta_correcta) != NULL)
      {
        correcto = true;
      }
    }
  }
  // Espera hasta tener una respuesta
  while ((correcto == false) && ((millis() - anterior) < tiempo_AT));
  return correcto;
}

void encender_gsm()
{
  bool respuesta = false;

  // Comprueba que el modulo SIM900 esta arrancado
  if (enviar_AT("AT", "OK", 2000) == 0) //comprueba la conexion a la red
  {
    Serial.println("Encendiendo el GPRS...");

    pinMode(9, OUTPUT);
    digitalWrite(9, HIGH);
    delay(2000);
    digitalWrite(9, LOW);
    delay(1000);

    // Espera la respuesta del modulo SIM900
    while (respuesta == false)
    {
      // Envia un comando AT cada 2 segundos y espera la respuesta
      respuesta = enviar_AT("AT", "OK", 2000);
      SIM900.println(respuesta);
      if(enviar_AT("AT", "OK", 2000) == 0)
      {
        reiniciar_gsm();
      }
```

```
    }
  }
}

void apagar_gsm()
{
  digitalWrite(9, HIGH);
  delay(1000);
  digitalWrite(9, LOW);
  delay(1000);
}

void reiniciar_gsm()
{
  Serial.println("Reiniciando...");
  apagar_gsm();
  delay (5000);
  encender_gsm();
}

void iniciar_gsm()
{
  enviar_AT("AT+CPIN=\"6523\"", "OK", 1000);
  Serial.println("Conectando a la red...");
  delay (5000);

  //espera hasta estar conectado a la red movil
  while ( enviar_AT("AT+CREG?", "+CREG: 0,1", 1000) == 0 )
  {
  }
  Serial.println("Conectado a la red.");
  enviar_AT("AT+CLIP=1\r", "OK", 1000); //Activar la identificacion de llamadas
  enviar_AT("AT+CMGF=1\r", "OK", 1000); //Configurar el modo texto para enviar o recibir
mensajes
  enviar_AT("AT+CNMI=2,2,0,0,0\r", "OK", 1000); //Configurar el modulo para que nos
muestre los SMS recibidos por comunicacion serie
  Serial.println("Preparado.");
}

void control_calefaccion()
{
  if ((valor_temperatura_real <= temperatura_variable) && (valor_temperatura_real !=
0.00))
  {
    contador_calefaccion_apagada = false;
    if (contador_calefaccion_encendida == false)
    {
    digitalWrite (encender_calefaccion, LOW);
    Serial.println("Calefaccion encendida");
    calefaccion = true;
    temperatura_variable = temperatura_elegida + ajuste_temperatura;    //Histeresis
temperatura
    contador_calefaccion_encendida = true;
    }
  }
  if ((valor_temperatura_real >= temperatura_variable) && (valor_temperatura_real !=
0.00))
  {
    if (contador_calefaccion_apagada == false)
    {
    contador_calefaccion_encendida = false;
    digitalWrite (encender_calefaccion, HIGH);
    Serial.println("Calefaccion apagada");
    calefaccion = false;
    temperatura_variable = temperatura_elegida - ajuste_temperatura;    //Histeresis
temperatura
    contador_calefaccion_apagada = true;
    }
  }
}

void enviar_sms()
{
  char no_disponible[] = " ";
  char temperatura[6];
  char temperatura_maxima[6];
```

```
  char temperatura_minima[6];
  char estado_climatizacion[6];
  char persiana_arriba[] = "arriba";
  char persiana_abajo[] = "abajo";
  char estado_persiana[7];
  char modo_automatico[] = "automatico";
  char modo_manual[] = "manual";
  char estado_modo[11];
  char calefaccion_encendida[] = "on";
  char calefaccion_apagada[] = "off";
  char estado_calefaccion[4];
  char cadena[159];
  char t_elegida[5];
  int tamano_numero_caracteres = 5;
  int numero_decimales = 2;



   Serial.println("Enviando SMS...");
   enviar_AT("AT+CMGF=1\r", "OK", 1000); //Comando AT para mandar un SMS

   //Convierte de un valor float a ASCII
   dtostrf( tmax, tamano_numero_caracteres, numero_decimales, temperatura_maxima);
   dtostrf( tmin, tamano_numero_caracteres, numero_decimales, temperatura_minima);

   //TEMPERATURA
   if (valor_temperatura_real == 0.00)
   strcpy(temperatura, no_disponible);
   if (valor_temperatura_real != 0.00)
   //Convierte de un valor float a ASCII
   dtostrf(    valor_temperatura_real,    tamano_numero_caracteres,    numero_decimales,
temperatura);

   //CLIMATIZACION
   dtostrf( temperatura_elegida, tamano_numero_caracteres, numero_decimales, t_elegida);
   if (temperatura_elegida == 0)
   strcpy(estado_climatizacion, no_disponible);
   if (temperatura_elegida != 0 )
   strcpy(estado_climatizacion, t_elegida);

   //ESTADO CALEFACCION
   if (calefaccion == 0)
   strcpy(estado_calefaccion, calefaccion_apagada);
   if (calefaccion == 1)
   strcpy(estado_calefaccion, calefaccion_encendida);

   //MODO AUTOMATICO Y MANUAL
   if (modo == automatico)
   strcpy(estado_modo, modo_automatico);
   if (modo == manual)
   strcpy(estado_modo, modo_manual);
   if ((modo != automatico) && (modo != manual))
   strcpy(estado_modo, no_disponible);

  //ESTADO PERSIANAS
  if (persiana_subida)
  strcpy(estado_persiana, persiana_arriba);
  if (persiana_bajada)
  strcpy(estado_persiana, persiana_abajo);
  if ((!persiana_subida) && (!persiana_bajada))
  strcpy(estado_persiana, no_disponible);

   sprintf(cadena, "Temperatura(C): %s\nTmax(C): %s\nTmin(C): %s\nClimatimatizacion(C):
%s\nCalefaccion:          %s\nModo:          %s\nPersiana:          %s\x1A          \r\n",
temperatura,temperatura_maxima,temperatura_minima,                estado_climatizacion,
estado_calefaccion, estado_modo, estado_persiana);
   Serial.println(cadena);
   sprintf(aux_str, codigo_mensaje, strlen(cadena)); //Numero al que vamos a enviar el
mensaje (95 caracteres de cadena)
   //Texto del mensaje
   if (enviar_AT(aux_str, ">", 10000) == 1)
   {
     enviar_AT(cadena, "OK", 10000);
   }
   Serial.println("SMS enviado");
```

```
}

void llamar()
{
  {
    Serial.println("Realizando llamada...");
    enviar_AT(codigo_llamada, "OK", 1000);
    delay(7000); // Duracion de llamada
    enviar_AT("ATH", "OK", 1000); // Cuelga la llamada
    Serial.println("Llamada finalizada");
```