UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Reinventando el control remoto

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: Adrián Quilis García

Tutor: Joan Josep Fons Cors
Vicent Pelechano Ferragud

Curso 2018-2019

Resum

En el present Treball de Fi de Grau (TFG) s'ha implementat un control remot el qual obri un canal d'integració de dispositius que solament utilitzen el infraroig o la radiofreqüència, mitjançant la xarxa d'àrea local, i oferir així una interfície per a la comunicació des de dispositius conectats a la xarxa local.

Aquest projecte ha sigut enfocat per a àrees amb dificultats d'accés a Internet e inclús a la xarxa elèctrica, pel que deu de complir les següents condicions: no deu requerir una conexió estable a Internet, pot ser alimentat amb una batería externa per a mòvils, i no deu requerir una configuració prèvia en la xarxa.

Paraules clau: arduino, Internet de les coses, xarxa d'àrea local, control remot

Resumen

En el presente Trabajo de Final de Grado (TFG) se ha implementado un control remoto el cual abre un canal de integración de dispositivos que solamente utilizan infrarrojos o radiofrecuencia, mediante la red de área local, y ofrecer así una interfaz para la comunicación desde dispositivos conectados a la red local.

Este proyecto ha sido enfocado para áreas con dificultades de acceso a Internet e incluso a la red eléctrica, por lo que debe cumplir las siguientes condiciones: no debe requerir una conexión estable a Internet, puede ser alimentado con una batería externa para móviles, y no debe requerir una configuración previa en la red.

Palabras clave: arduino, Internet de las cosas, red de área local, control remoto

Abstract

In the current Final Project Degree (TFG) a remote control has been implemented which opens a new way for integrating devices which only uses infrared or radiofrequency, through the local area network, and offer an interface to communicate from devices connected to the local network.

This project is oriented for areas with Internet access dificulties, even electricity dificulties, so some conditions should be accomplished: it should not require a stable Internet connection, it may be powered with phone power banks, and it should not require previous network configuration.

Key words: arduino, Internet of things, local area network, remote control

Índice general

Índice de figuras

CAPÍTULO 1
Introducción

En los últimos años, el sector del Internet de las Cosas (*IoT*) ha experimentado un incremento exponencial en cuanto a dispositivos disponibles y funcionalidades que abarca, desde el ámbito doméstico hasta ámbitos de caracter empresarial.

En este proyecto se creará un sistema con las características de un sistema IoT teniendo presente objetivos como el coste, velocidad, consumo y facilidad de instalación. Adicionalmente, como este tipo de dispositivos requieren de una aplicación para su manejo, se ha diseñado una aplicación en consola para el caso de prueba.

El sistema que ha sido creado es una interfaz que permite la comunicación a través de la red con dispositivos que emplean canales antiguos como el infrarrojo o la radiofrecuencia sin emplear el estándar 802.11 .

1.1 Motivación

El IoT ha supuesto una gran mejora en la calidad de vida de las personas que deciden apostar por estas tecnologías, las cuales crecen de forma cuantiosa cada año, con lo que se puede afirmar que estan marcando un progreso diferente en la vida diaria de las personas. Ejemplo de este progreso son productos tales como Google Home [1] o Apple TV [2]. El problema principal que presentan estos dispositivos anteriormente mencionados es que requieren de una conexión estable a Internet o no están pensados para zonas con problemas en el suministro eléctrico.

La motivación de este proyecto ha sido el llevar las comodidades del Internet de las Cosas a zonas con acceso a Internet restringido, como podría ser una zona rural o una zona conflictiva, las cuales pueden presentar dificultades incluso para acceder a la red eléctrica. Con este caso de estudio, se propone un dispositivo el cual solo requiera de una red de área local y una fuente de alimentación la cual puede ser fácilmente reemplazable por una batería externa para teléfonos móviles, las cuales son fáciles de encontrar en tiendas por un bajo precio.

Personalmente, el proyecto ha sido motivado al querer llevar la comodidad del Internet de las Cosas a un familiar de avanzada edad el cual se niega a contratar el servicio de Internet, y cuya casa presenta una instalación eléctrica obsoleta, con lo que se acciona el diferencial con relativa facilidad.

1.2 Objectivos

Debido a todo lo anteriormente mencionado, se presentan los siguientes objetivos o requisitos a cumplir en el diseño del dispositivo:

- Su consumo debe ser lo suficientemente bajo como para que una batería externa de teléfono móvil sirva para alimentarlo.
- No debe requerir de una conexión estable a Internet.
- Debe ser lo suficientemente rápido para no causar frustración a los usuarios de los mandos a distancia infrarrojos o radiofrecuencia.
- No debe requerir una configuración previa de la red.
- Debe ser fácilmente integrable para que se pueda desarrollar sin complicaciones un ámplio abanico de aplicaciónes para su control, como una aplicación en consola de comandos.

1.3 Estructura de la memoria

En primer lugar, se realizará una contemplación de otros dispositivos similares ya existentes en el mercado, junto a la exposición de sus deficiencias.

A continuación, se analizarán las tecnologías seleccionadas para este proyecto, aportando la justificación de la elección, tanto software como hardware y red. También se hará mención de las librerías empleadas en el desarrollo del dispositivo.

Posteriormente, se propondrá un diseño del dispositivo, donde se profundizará en detalles funcionales dentro del proyecto junto a las justificaciones de las decisiones tomadas a lo largo del proyecto. A continuación, se redactará la implementación del proyecto del dispositivo y el controlador del disposiivo.

Finalmente, se realizará un sumario de los pasos necesarios para realizar el ensamblaje del dispositivo junto a imágenes del resultado de cada paso. A continuación las pruebas realizadas y los dispositivos empleados, y el resultado de las mismas.

CAPÍTULO 2
Estado del arte

En el caso de los transformadores de señal WiFi a infrarrojo, empresas como Logitech [3] ya han apostado por crear este tipo de tecnologías. En cuanto a similitud, el producto que más se asimilaría al proyecto sería el Broadlink RM Mini3 [4].

Broadlink RM Mini3 [4] es un producto de la marca Broadlink [5] el cual dispone de una interfaz de conexión infrarroja a 38kHz de frecuencia. Su conectividad se realiza mediante WiFi, y se controla empleando una aplicación para smartphones de la cual es propietaria.

Figura 2.1: Broadlink RM Mini3 [4].

En el mercado podemos encontrar otro producto similar, el OWSOO eWeLink IR Hub.

Figura 2.2: OWSOO eWeLink IR Hub [6].

El eWeLink IR Hub [6] es un producto de OWSOO que, al igual que el producto anterior, proporciona una interfaz para conectar un dispositivo conectado a la red con un dispositivo que emplee la tecnología infrarroja a 38kHz de frecuencia. Este producto, en cambio, se ha integrado con una aplicación de terceros llamada eWeLink [7], la cual permite el control del dispositivo de forma remota.

2.1 Crítica al estado del arte

Todos los dispositivos anteriormente mencionados (Capítulo 2) solo entregan paquetes a travésde la interfaz de infrarrojos siempre y cuando la frecuencia de emisión, lo cualreduce el rango de dispositivos a protocolos como el NEC [8] que operan a dichafrecuencia, dejando sin conexión a dispositivos que emplean los protocolos SONY SIRC [9] o Philips RC-5 [10] que operan a la frecuencia de 40kHz y 36kHz, respectivamente.

Adicionalmente, estos dispositivos no soportan ningún tipo de conexón empleando el canal de la radiofrecuencia además del WiFi, con lo que dispositivos como aires acondicionados o enchufes de pared gestionables no podrán ser gestionados si no emplean infrarrojos y se pueden ver entre sí.

Otro problema que presentan es la poca utilidad que presentan ante la ausencia de conectividad del servidor de colas que los controla, debido a que estos dispositivos requieren de un servidor ajeno a la red para su correcto funcionamiento (véase Figura 2.3, lo que, sinentrar en temas de seguridad o privacidad, presenta notables retardos en lasoperaciones, además de una fuerte dependencia entre el funcionamiento del hogar con la conectividad a la red.

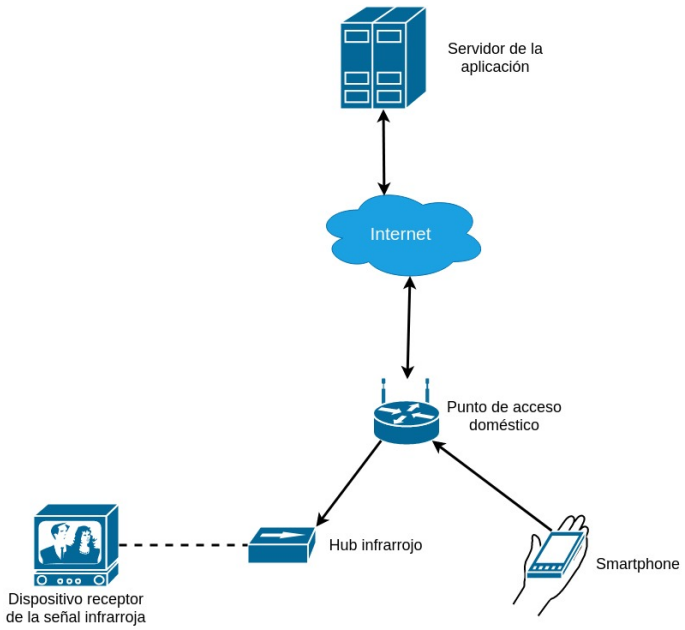

Figura 2.3: Esquema de red de los dispositivos del Capítulo 2

Como problema específico de la infraestructura del OWSOO eWeLink IR Hub, al depender de una aplicación de terceros, cada comando enviado requiere ser enviado primero al servidor de colas del tercero, después al servidor de colas del fabricante y de ahí al dispositivo final, lo cual conlleva a retardos mayores.

CAPÍTULO 3
Análisis del problema

El proyecto el cual se va a desarrollar debe de cubrir ciertos aspectos para cubrir los objetivos propuestos.

Para comenzar, el dispositivo debe poder modificar la frecuencia de funcionamiento del LED infrarrojo para poder abarcar también dispositivos que empleen otros protocolos para su funcionamiento. Es por ello que se establece que la frecuencia que debe soportar el dispositivo para emitir al LED debe situarse entre 36kHz y 40kHz, ambos inclusive. Además, para mayor flexibilidad ante el actual mercado de dispositivos siempre cambiante, los códigos no deben ser almacenados en el propio dispositivo, sino en el controlador.

Además de la interfaz infrarroja, debe disponer una interfaz de radiofrecuencia para poder abarcar dispositivos que emplean la radiofrecuencia como aires acondicionados o enchufes por control remoto que no empleen la interfaz infrarroja. Con esto se garantiza que el dispositivo será de un propósito más general y no exclusivo para televisores.

Como característica adicional, este dispositivo no debe requerir una conexión estable a Internet, solo a la red local mediante WiFi, con lo que no dispondrá de ningún servidor de colas ni interno ni externo a la red local. De esta forma se garantiza su funcionamiento ante un entorno en el que la conexión a Internet no sea estable o no exista. Tampoco debe requerir una configuración previa en el router, con lo que causa un problema con el hecho de encontrar el dispositivo dentro de la red ante un entorno con direcciones IP dinámicas, donde no se garantiza que la dirección IP que se empleó con anterioridad sea la misma que en la que se encuentre actualmente, con lo que se deberá diseñar un protocolo para permitir encontrar el dispositivo dentro de una red indeterminada.

Como problema final, el coste de la plataforma debe ser mínimo, lo suficiente para que permita ser asequible para la mayor parte de las familias, y que sea fácil de construir de nuevo y repararlo. Además debe ser lo suficientemente rápido como para que no cause frustración a los usuarios de controles remotos tradicionales.

3.1 Protocolo de red

Para la conexión a la red es necesario emplear ciertos protocolos que permitan la conexión entre dispositivos de forma fácil y rápida. Pero para decidir qué protocolo va a emplearse, se debe considerar la complejidad del protocolo, porque un mensaje difícil de procesar es un mensaje que costará tiempo adicional y, por lo tanto, consumo adicional.

Para definir los protocolos empleados, se empleará el modelo TCP/IP (véase Figura 3.1) y se explicará nivel a nivel la configuración seleccionada.

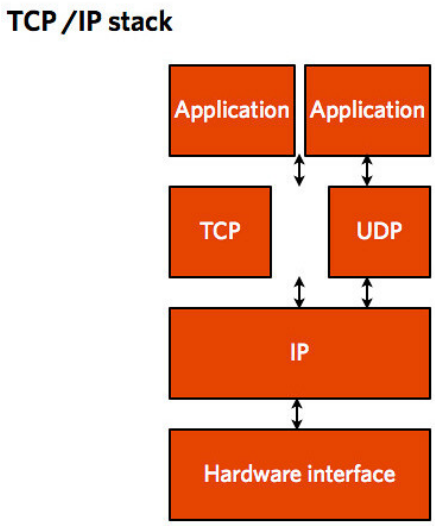

Figura 3.1: Modelo TCP/IP. Fuente: https://www.flickr.com/photos/purpleslog/5598305463.

3.1.1. Capa de enlace

En cuanto a interfaz hardware para realizar la conexión a la red, se ha optado por emplear una interfaz inalámbrica para así emplear la versatilidad y el bajo coste tanto en cuanto a precio como en instalación que ofrece dicho medio.

Al tratarse de un dispositivo de reducidas dimensiones que en algún momento puede ser necesario cambiar su ubicación para maximizar su efecto, emplear WiFi en lugar de una interfaz cableada para resulta notablemente conveniente. Adicionalmente, gracias a la interfaz WiFi, es posible realizar la instalación del dispositivo sin necesidad de depender del cableado del recinto.

3.1.2. Capa de red

En esta capa se encuentra el protocolo IP (*Internet Protocol*) el cual se configura para obtener direcciones de forma dinámica empleando un servidor DHCP (*Dynamic Host Configuration Protocol*), el cual normalmente se encuentra integrado dentro del software del router doméstico, para cumplir con el objetivo anteriormente mencionado (véase Sección 1.2) donde se especificaba que no debía requerir configuración previa en la red.

Esta característica supone un problema adicional en cuanto a la comunciación entre dispositivos debido a que las direcciones otorgadas no son permanentes, con lo que los dispositivos deben "encontrarse" entre si.

3.1.3. Capa de transporte

En cuanto al transporte de la información, se emplearán tanto el protocolo TCP (*Transmission Control Protocol*) [20] como el protocolo UDP [21] (*User Datagram Protocol*).

El protocolo TCP [20] será empleado para el envío de información y comandos entre dispositivos, para así garantizar la recepción del mensaje enviado. Adicionalmente, como parte del protocolo TCP, también se garantiza la integridad del mensaje recibido y, en caso de no ser correcto, dar la posibilidad a reenviar el mensaje.

El protocolo UDP [21] será empleado para el protocolo de descubrimiento entre dispositivos, el cual presenta notables ventajas al permitir enviar una difusión a la red y no ser necesaria una verificación como que el destinatario (en este caso, toda la red) ha recibido correctamente el mensaje. Por motivos de rapidez, este protocolo también es empleado para emitir al respuesta por parte del dispositivo buscado.

3.1.4. Capa de aplicación

En esta capa se hallan dos protocolos de desarrollo propio (véase Sección A.2) , uno para permitir el intercambio de mensajes entre la interfaz y el controlador, y otro para permitir el descubrimiento de la interfaz dentro de una red con asignación de direcciones dinámica. Ambos protocolos se han diseñado para ser simples tanto de entender como de procesar, para así permitir una mayor rapidez para integrar la interfaz con otras aplicaciones, además de facilitar el procesamiento del mensaje y así decrementar el tiempo hasta la emisión del código a través de la interfaz seleccionada.

3.2 Hardware

Como uno de los objetivos del proyecto es abaratar costes (véase Sección 1.2), pero a su vez que la interfaz sea lo suficientemente rápida como para no frustrar a los usuarios habituales del control remoto tradicional, se ha considerado la mejor opción del mercado la placa base *Arduino* [11], concretamente la Arduino MKR 1010 WiFi [12], debido a que posee integrada una interfaz WiFi que, tal y como se ha mencionado anteriormente (Subsección 3.1.1) se empleará para obtener conexión a la red. Adicionalmente, dicha placa presenta importantes modificaciones en la configuración hardware para minimizar el consumo de la misma.

3.2.1. Interfaz infrarroja

Para el uso de la interfaz infrarroja, es necesario un LED que permita la emisión de luz infrarroja, además de una resistencia la cuyo valor se calcula a 82 Ohms.

Como la salida de voltaje de los pines de esta Arduino es de 3,3 V, no es suficiente para alimentar suficientemente el LED y, por lo tanto, el dispositivo receptor de la señal infrarroja no es capaz de recibirla. Para solucionar este problema, es necesario un transistor que permita modular la señal a través de una corriente de 3,3 V y emplear la fuente de 5 V también incluida en la Arduino para otorgarle al LED la energía suficiente para que el

código emitido pueda ser leído por el dispositivo deseado. El transistor empleado es el IRF520 [13] junto a un circuito que facilita la saturación del mismo para permitirlo trabajar a altas frecuencias.

3.2.2. Interfaz de radiofrecuencia

Para poder emplear el uso de la radiofrecuencia como canal de transmisión de datos, se ha seleccionado la antena de emisión FS1000A [14] la cual puede operar con la alimentación estándar de Arduino (3,3 V), además de presentar un corto alcance (1,5 metros) el cual es suficiente para operar dentro de un entorno doméstico.

Esta antena opera a 433 MHz, que es la frecuencia seleccionada, dado que es la más usada dentro del ámbito de mandos a distancia para la apertura de puertas y otros dispositivos domésticos como enchufes controlables remotamente.

3.3 Software

Con el fin de permitir la comunicación entre controlador y dispositivo receptor de señal infrarroja o de radiofrecuencia, se hacen uso de ciertas librerías que garanticen un correcto y fiable funcionamiento de la interfaz.

Alguna de las librerías empleadas fueron desarrolladas por un equipo de desarrollo de Arduino y relacionados. Otras, sin embargo, fueron publicadas en repositorios públicos dentro de la red de repositorios GitHub [15] y son mantenidos por la comunidad.

3.3.1. Librería WiFi

Para poder hacer uso del chip WiFi integrado dentro del dispositivo, se hace uso de la librería WiFiNINA [16] la cual ha sido desarrollada por Arduino y relacionados.

Esta librería permite el control del chip ESP32 dentro de placas base Arduino o compatibles, el cual se encuentra presente dentro de la Arduino MKR 1010 WiFi [12].

3.3.2. Librería para infrarrojos

La librería empleada para usar la interfaz infrarroja es *IRLib2* [17] desarrollada y mantenida por un usuario de la plataforma GitHub [15] llamado *cyborg5*. Esta librería presenta una compatibilidad con una gran cantidad de protocolos para la interfaz infrarroja.

Debido a que esta librería no es compatible con la Arduino MRK 1010 WiFi, es necesario aplicar un parche propio para su correcto funcionamiento, sobre el cual se comentará con más profundidad posteriormente (véase Subsección 5.2.1).

3.3.3. Librería para radiofrecuencia

Para el desarrollo de la interfaz de radiofrecuencia a 433 MHz se ha requerido el uso de la librería *rc-switch* [18] desarrollada y mantenida por otro usuario de la plataforma GitHub [15] llamado *sui77*. Esta librería es empleada para la emisión de códigos a una frecuencia de 433 MHz y 315 MHz.

Esta librería no ha sido diseñada para el procesador SAMD21 presente en la Arduino MKR 1010 WiFi [12]; sin embargo, es totalmente compatible con él, pese al aviso en tiempo de compilación por parte del compilador de Arduino [19].

CAPÍTULO 4
Diseño del dispositivo

La propuesta de diseño de la solución se trata de un sistema el cual sea descentralizado, es decir, no requiere de ningún servidor o configuración dentro o fuera de la red mas que los datos de conexión a la red WiFi. Es por ello que la arquitectura de red propuesta presenta similitudes con una red P2P (*Peer-to-Peer*) [22] descentralizada pero a pequeña escala.

La complejidad adicional de esta arquitectura es que ambos dispositivos parten de un desconocimiento de la red, con lo que no tiene conocimiento alguno sobre la dirección de red del otro dispositivo, con lo que se precisa de un algoritmo de búsqueda de dispositivos en red según un identificador interno en la capa de aplicación.

Además de la red, también debe soportar varios protocolos infrarrojos, lo cual implica poder ajustar su frecuencia a diferentes valores, con lo que el dispositivo debe conocer los parámetros propios del protocolo para minimizar fallos en cuanto a la configuración del protocolo. No hay que descuidar la interfaz de radiofrecuencia, la cual requiere de una llamada desde el protocolo y sus parámetros deben estar previamente almacenados dentro del dispositivo para agilizar la transmisión de datos.

4.1 Arquitectura de red

En cuanto a la red, se propone una arquitectura sin servidores ni intermediarios donde el controlador se comunica directamente con la interfaz, quedando un esquema de red como el siguiente (véase Figura 4.1).

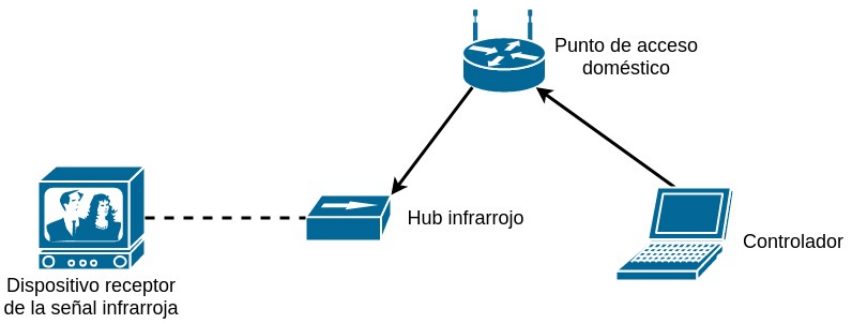

Figura 4.1: Esquema de red del envío de datos empleando infrarrojos del proyecto

Partiendo desde el caso en que ambos dispositivos han terminado la fase de inicialización (véase Sección A.1) y desconocen la dirección de red del otro dispositivo, realizan las siguientes fases para transmitir datos desde el controlador al dispositivo empleando el protocolo anteriormente mencionado (Subsección 3.1.4):

1. El controlador emite una difusión UDP donde solicita la respuesta del dispositivo en red que coincida con el identificador adjunto.

2. El controlador configura el puerto 836 para la escucha de datagramas UDP entrantes para recibir el mensaje del dispositivo con un temporizador de dos segundos.

3. El dispositivo recive el mensaje UDP y reenvía una copia del mensaje a la dirección origen del mensaje anterior, la cual será la dirección del controlador.

4. El controlador recibe la respuesta del dispositivo y termina la escucha del puerto 836 en UDP, y guarda la dirección de red origen de la respuesta, la cual es la del dispositivo.

5. El controlador envía un mensaje a la dirección 835 empleando TCP y se configura para recibir un mensaje de respuesta en dos segundos de margen. Si no recibe el mensaje en dos segundos de margen, se da por dispositivo caído y aborta el envío del mensaje.

6. El dispositivo recibe el mensaje, procesa el protocolo y el código a emitir y responde con un mensaje de respuesta indicando el estado del mensaje.

7. El controlador recibe y procesa el mensaje de respuesta e informa al usuario.

En el caso de que el controlador se haya conectado con anterioridad al dispositivo, primero realiza un intento de envío del mensaje empleando TCP y, en el caso de que el mensaje no haya podido ser entregado, se dispondrá a realizar el procedimiento anterior. En el caso de que haya sido entregado satisfactoriamente, se procederá al paso 6 y 7 del procedimiento anterior.

4.2 Hardware de la interfaz infrarroja

La interfaz infrarroja requiere de un LED con emisión de luz infrarroja, al que es necesario conectarle una resistencia de 82 Ohmios. Dado que la corriente del pin 0 de la Arduino MKR 1010 WiFi [12] no provee los 5V necesarios para funcionar, es necesario instalar adicionalmente un MOSFET para permitir que la alimentación del LED venga dada por el pin de 5V de la Arduino, pero permitiendo también modular la señal empleando el pin programable. Es por ello que se requiere como componente adicional el MOSFET IRF520 [13] con un circuito integrado para permitir una saturación del transistor más rápida.

El diseño final de la Arduino sigue el esquema siguiente (véase Figura 4.2).

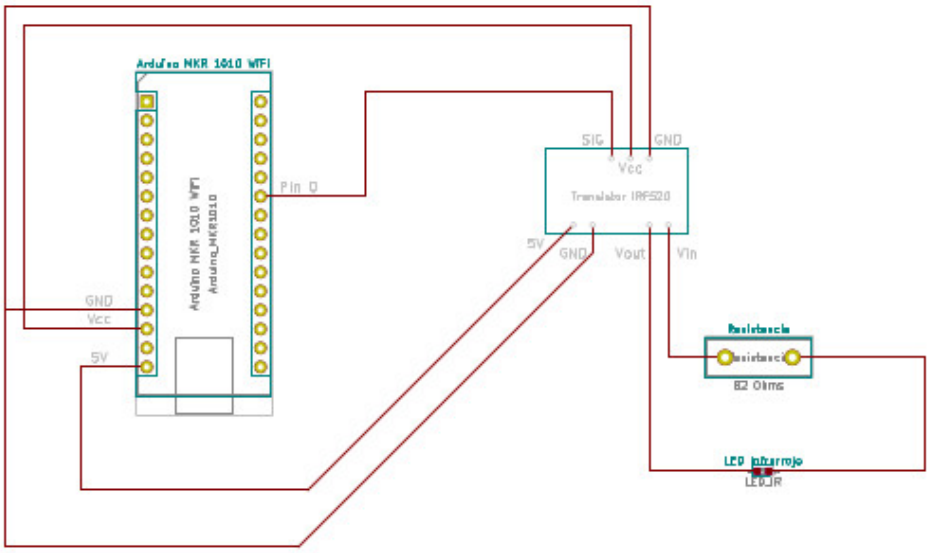

Figura 4.2: Diseño PCB de la interfaz infrarroja

En el diseño anterior (Figura 4.2) se observa que el pin 0 de la Arduino se conecta con la entrada de señal SIG del circuito de alimentación, junto con una conexión a la salida de 3,3V de la Arduino (Vcc) y la base (GND). Este circuito de alimentación se conecta con el transistor IRF520 en la puerta o *gate*. Se observa que el pin de 5V de la Arduino se conecta al circuito de alimentación, el cual conecta directamente con la resistencia del LED infrarrojo. El resistor del LED infrarrojo se conecta al LED infrarrojo y la base o GND se conecta al drenador del IRF520 o *drain*. Finalmente, la fuente del transistor se conecta con la base próxima a la conexión de 5V.

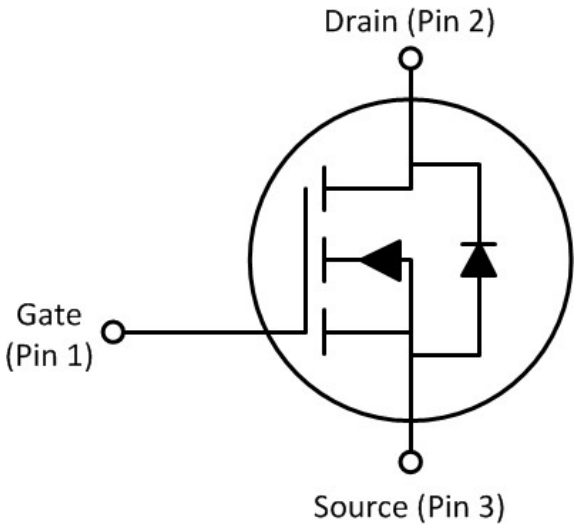

Figura 4.3: Esquema de un MOSFET. Fuente: http://www.ti.com/product/CSD19536KCS.

4.3 Hardware de la interfaz de radiofrecuencia

Para operar con la interfaz de radiofrecuencia, es necesario disponer de una antena de radiofrecuencia compatible. Para ello se dispone del chip de radiofrecuencia FS1000A [14] el cual permite ser alimentado con 3,3V, con lo que no requiere de un circuito de alimentación adicional para su funcionamiento.

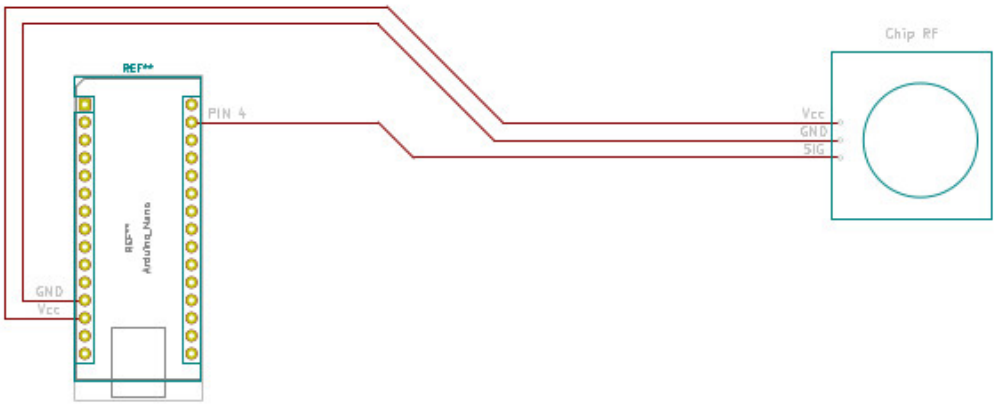

Figura 4.4: Diseño PCB de la interfaz de radiofrecuencia

Como se puede observar en la Figura 4.4, el chip es alimentado desde la alimentación estándar de la placa base Arduino y el pin es conectado de forma directa al chip sin necesidad de circuitos adicionales.

4.4 Software del dispositivo

En cuanto a software, la aplicación debe de estar a la escucha en dos puertos, el 835 en TCP y el 836 en UDP, para que el dispositivo pueda ser utilizado empleando el protocolo TCP, o descubierto con el protocolo UDP. Una vez recibido un mensaje, el programa deberá reaccionar en función del protocolo empleado.

Si el protocolo empleado es TCP, el programa deberá distinguir mediante el código de protocolo si va a emplear radiofrecuencia o infrarrojos e informar de que el mensaje ha sido recibido correctamente. En el caso de que el código del protocolo no coincida con ninguno de los soportados, informará de que el protocolo solicitado no es soportado.

Si el protocolo empleado UDP, procesará el mensaje recibido para determinar si el identificador de dispositivo coincide con el suyo. En caso de que coincidan, se copiará el mensaje de entrada al *buffer* de salida y enviar el mensaje a su origen. En el caso de que no coincida, el mensaje será descartado y no enviará respuesta alguna.

Para mostrarlo de forma gráfica, el software respeta el siguiente diagrama de flujo (véase Figura 4.5).

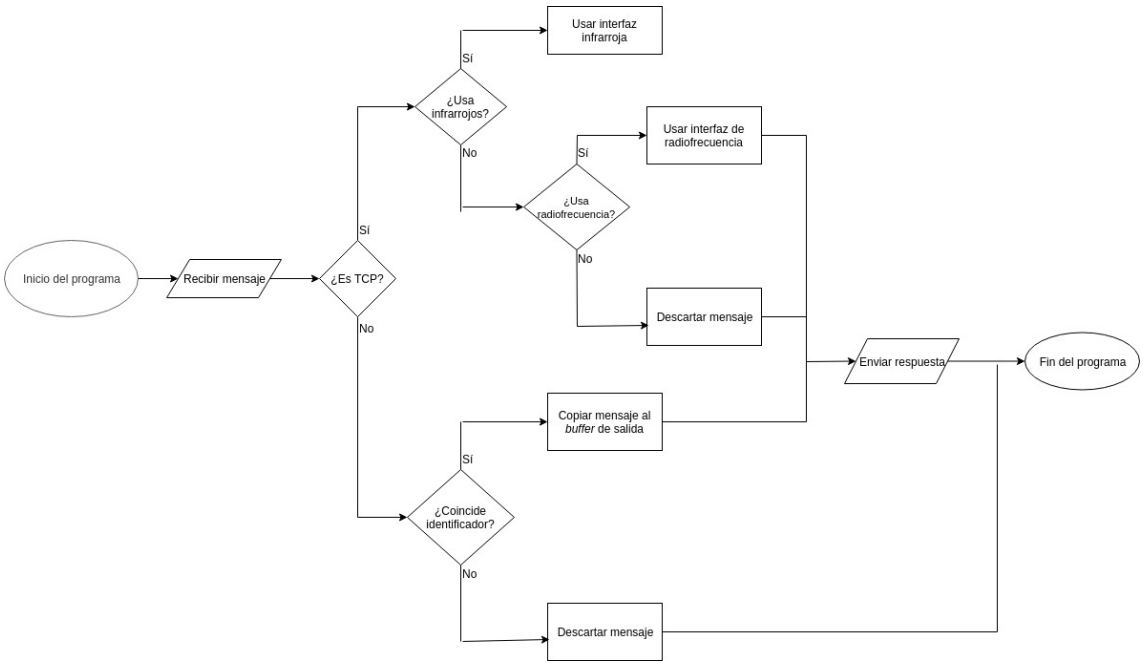

Figura 4.5: Diagrama de flujo del software del dispositivo

CAPÍTULO 5
Implementación de la solución

Este proyecto ha sido implementado empleando el kit de desarrollo de Arduino [19] y una aplicación en consola escrita en C para facilitar la interacción con el dispositivo.

La simplicidad de los requisitos a efectos de usuario dado que se perseguía como objetivo una fácil instalación del dispositivo (véase Sección 1.2).

5.1 Aplicación controladora

La aplicación creada con el fin de agilizar la comunicación con el dispositivo se trata de una aplicación en consola, la cual ha sido llamada *IUIC* o Interfaz unificada de intercomunicación (*Interfície Unificada d'InterComunicació*).

El control de la aplicación se realiza empleando una serie menús con opciones numeradas y una opción para regresar al menú anterior (véase Figura 5.1).

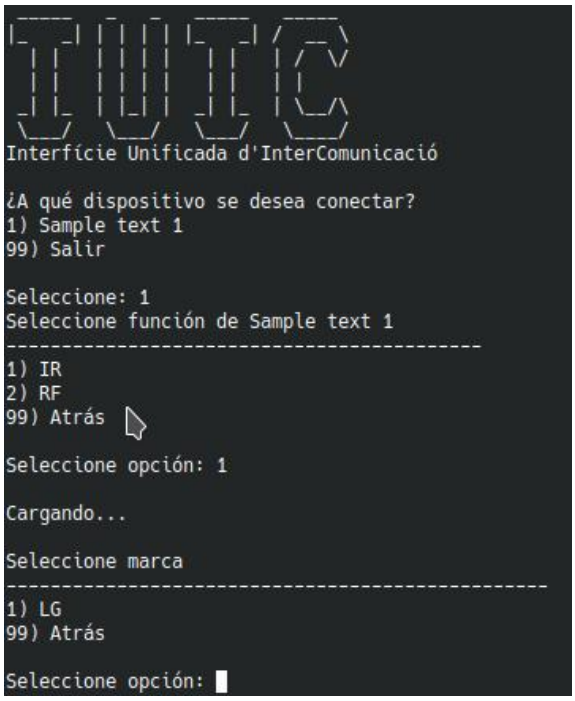

Figura 5.1: Captura de la pantalla de la Interfaz unificada de intercomunicaciones

La aplicación funciona a modo de cliente, dado que siempre es la aplicación quien inicia la comunicación del dispositivo. En cuanto a funcionalidad, la aplicación tiene almacenados todos los códigos de funcionamiento de los dispositivos a los que se puede conectar y el protocolo que requieren. Para ello han sido creadas ciertas estructuras de datos, como por ejemplo las que pueden verse para el control de los televisores por infrarrojos (véase Figura 5.2).

```
struct IRcode {
   char name[10];
   uint8_t code[4];
};
struct TVs {
   char brand[5];
   struct IRcode codes[MAX_CODES];
   uint8_t protocol;
   int num_codes;
} *codigos;
```

Figura 5.2: Estructuras de datos destinadas para el control de los televisores mediante infrarrojos en la aplicación

5.2 Dispositivo Arduino

Para el desarrollo del dispositivo, es imprescindible disponer del entorno de desarrollo de Arduino [19] e instalar las librerías especificadas en la Sección 3.3.

5.2.1. Parche a la librería IRLib2

Dado que la librería IRLib2 [17], tal y como se mencionó en la Sección 3.3, no es compatible con la Arduino que se va a emplear para el desarrollo del software, es necesario aplicar modificaciones en los archivos *IRLibProtocols/IRLibSAMD21.h* e *IRLibProtocols/IRLibSAMD21.cpp* para que funcione correctamente.

La modificación necesaria en el archivo terminado en ".h" es la siguiente

```
137c137
<   #define IR_SEND_PWM_PIN 9
---
>   #define IR_SEND_PWM_PIN 0
```

Y la modificación necesaria para el archivo fuente terminado en ".cpp" es la siguiente

```
30c30
<   uint8_t IR_PER_EorF;
---
>   short int IR_PER_EorF;
```

Con ambas modificaciones realizadas, es posible proceder a implementar el dispositivo sin recibir avisos o funcionamientos anómalos.

5.2.2. Protocolo de descubrimiento

Para permitir el descubrimiento del dispositivo, es necesario crear un *socket* de escucha para el protocolo UDP que permita la entrada de datagramas en difusión, debido a que el datagrama de descubrimiento se enviará en difusión al puerto 836.

Para poder habilitar la escucha en el puerto 836 es necesario recurrir a la librería WiFiNINA [16] empleando la función *WiFiUDP.begin(int port);*, donde *int port* debe ser reemplazado por el número del puerto que, en este caso, será 836.

Al procesar el datagrama, tal y como se menciona en la Sección 4.1, el dispositivo solo responderá en el caso de que el identificador solicitado coincida con el identificador del dispositivo. La respuesta a este mensaje será una copia idéntica del mensaje de descubrimiento y será enviado a la dirección de origen que coincidirá con la dirección del controlador solicitante.

5.2.3. Protocolo de transmisión de datos

Para poder recibir datos desde la controladora, es necesario crear otro *socket* de escucha en el puerto 835 para el protocolo TCP. En este caso, no es necesario aceptar paquetes en difusión, con lo que solo deberá recibir paquetes cuyo destinatario sea el dispositivo.

Para habilitar la escucha anterior, es necesario recurrir a la librería WiFiNINA [16] para usar la función *WiFiServer.begin()*. Para poder definir el puerto al que se va a escuchar es necesario crear un objeto en memoria empleando el constructor *WiFiServer(int port)*, donde *int port* deberá ser reemplazado por el número del puerto que se desea emplear, en este caso el 835.

Los mensajes que reciba serán procesados, donde el identificador del protocolo definirá el conjunto de parámetros que serán empleados, e informará al controlador sobre el estado del mensaje, tanto si ha sido aceptado como si no es posible procesarlo o posee una mala sintaxis.

CAPÍTULO 6
Montaje

De acuerdo a lo establecido anteriormente en la Sección 3.2, la plataforma sobre la cual se va a desarrollar el dispositivo será la Arduino MKR 1010 WiFi [12]. Con el fin de facilitar la visibilidad y claridad del ensamblado realizado y los componentes empleados, se ha optado por emplear una placa de prototipado más extensa de lo estrictamente necesario.

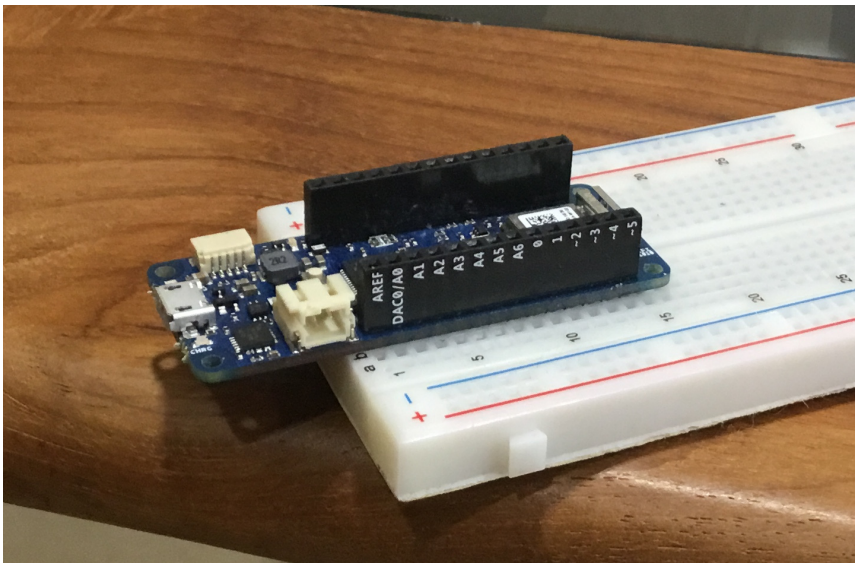

Figura 6.1: Arduino instalada en la placa de prototipado

Para mejor explicación, se explicará el ensamblado del dispositivo por partes antes de ensamblarlo completamente.

El ensamblado de la interfaz infrarroja de acuerdo al diseño PCB mostrado anteriormente (véase Figura 4.2), resultaría en el siguiente ensamblado (véase Figura 6.2).

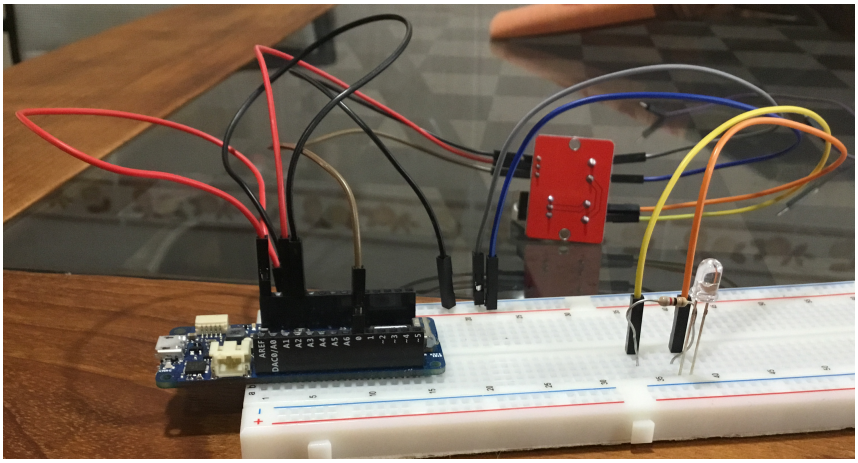

Figura 6.2: Ensamblado de la interfaz infrarroja

Nótese que al fondo hay un chip de alimentación con siete cables conectados al mismo. Las conexiones a este chip de alimentación tienen la siguiente diposición (véase Figura 6.3).

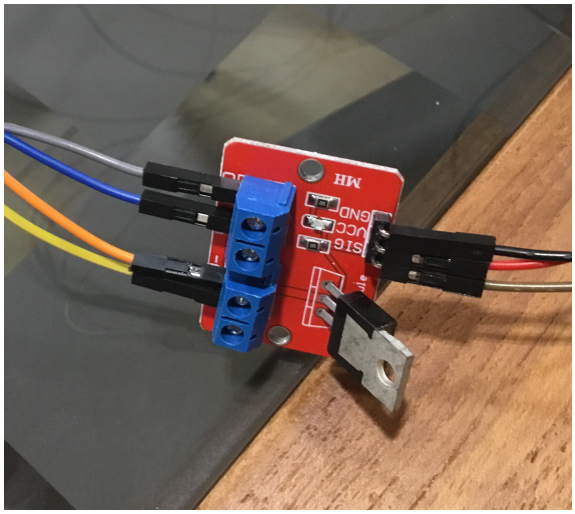

Figura 6.3: Ensamblado del chip de alimentación

En cuanto al ensamblado de la interfaz de radiofrecuencia, de acuerdo al diseño PCB mostrado anteriormente (véase Figura 4.4), resulta en el siguiente ensamblado (véase Figura 6.4).

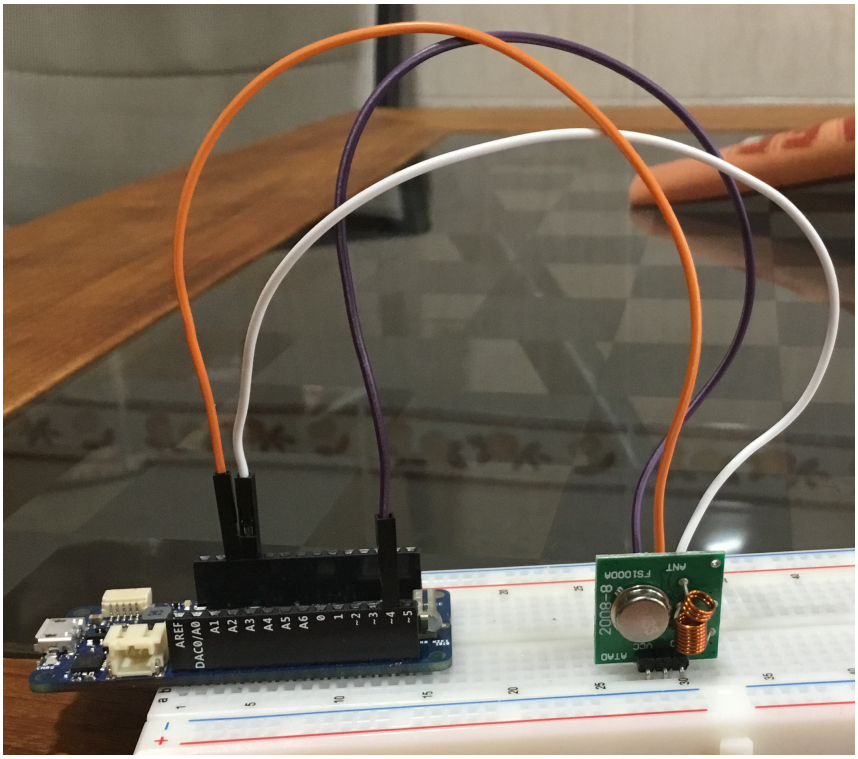

Figura 6.4: Ensamblado de la interfaz de radiofrecuencia

En cuanto a los pines empleados en la Arduino para conectar las interfaces, se emplean el pin 0 para la interfaz infrarroja y el pin 4 para la interfaz de radiofrecuencia (véase Figura 6.5), y en los pines de alimentación se emplean el de 5V, Vcc y GND para alimentar las interfaces.

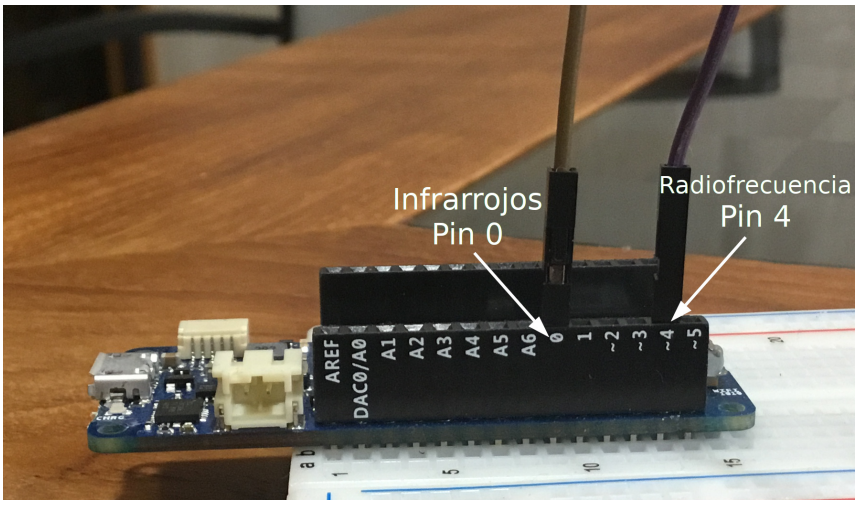

Figura 6.5: Pines GPIO utilizados en la Arduino

El resultado final del ensamblado completo, usando todos los diseños PCB y ensambla-

dos anteriores, derivaría a la siguiente disposición (véase Figura 6.6).

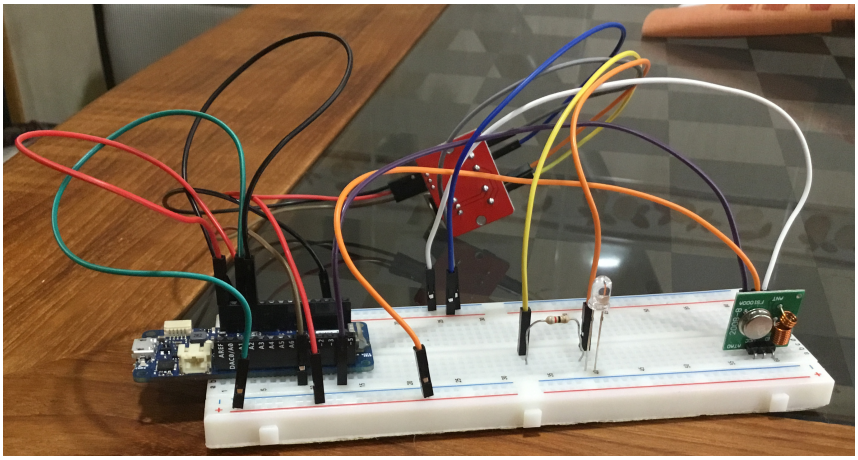

Figura 6.6: Ensamblado completo del dispositivo

CAPÍTULO 7
Pruebas realizadas

Tras realizar el ensamblaje y completar satisfactoriamente las pruebas que realiza el compilador previamente a realizar el compilado y la instalación del código ensamblado en la Arduino, se dispone a realizar las pruebas de funcionamiento.

Para ello se dispone de un televisor LG Flatron DM2350D [23] la cual se puede controlar remotamente con el control de infrarrojos de forma exclusiva y un enchufe RoHS PCH005723 [24] accionable exclusivamente desde el control remoto que emplea radiofrecuencia en la frecuencia de 433MHz. En cuanto a la instalación de red WiFi doméstica, se dispone de un punto de acceso WiFi desconectado de Internet. Para todas las pruebas realizadas, es necesario conectar dispositivo y controlador a la misma red WiFi.

Para realizar las pruebas de la interfaz infrarroja es necesario tener la televisión conectada a la red eléctrica y posicionar el LED infrarrojo apuntando directamente al receptor de infrarrojos del televisor, al igual que se realiza con los controles remotos convencionales.

En cuanto a las pruebas de la interfaz de radiofrecuencia, solo es necesario que el enchufe con control remoto esté dentro del alcance de la señal de radiofrecuencia de la antena de la interfaz, ya que la antena es omnidireccional, a diferencia del LED de infrarrojos que es unidireccional, es decir, requiere que apunte directamente al receptor.

Tras realizar las pruebas pertinentes con las funciones deseadas, se concluye con que la interfaz funciona perfectamente, y podría ser perfeccionado con la adición de nuevos protocolos soportados y mejorar su portabilidad con la adición de una batería integrada, aunque es posible alimentarlo con una batería externa de teléfono móvil.

CAPÍTULO 8
Conclusiones

Tras la realización de este proyecto se ha probado que los medios actuales permiten crear dispositivos IoT fiables y rápidos que permitan facilitar la utilización de los dispositivos que no permiten la conexión a la red empleando interfaces para transformar el código llegado desde la red a una emisión de infrarrojos y controlar dicho dispositivo como si estuviera conectado a la red.

También se ha probado que es posible disponer de una red de dispositivos IoT sin requerir de dispositivos externos, ni conexión a Internet, ni poner en riesgo la privacidad del hogar sin ser estrictamente necesario.

Este proyecto ha presentado dificultad del diseño, dado que el protocolo de descubrimiento ha requerido una implementación previa, además de que muchos de los métodos de identificación de dispositivos existentes requerían de configuraciones en la red o de un dispositivo con una dirección de red estática dentro del entorno doméstico. Es por ello que muchas empresas han decidido realizar el descubrimiento de dispositivos de forma remota en los servidores de la empresa o contratados en nube y, en algunos casos, la transmisión de datos también.

El diseño del proyecto en su fase inicial requería de un servidor de colas para identificar los dispositivos autorizados, es decir, los que se encontraban dentro de la red, pero se decidió descartar la idea tras el elevado coste que supondría el dispositivo a gran escala y la infraestructura tan súmamente compleja que se requeriría para realizar una operación simple como encender un televisor. A este hecho se unió el hecho de la rapidez de respuesta de los dispositivos, donde se decidió que sería necesario reducir tiempos de respuesta no solo para mejorar el tiempo de respuesta de dispositivos comerciales que disponen de dicha infraestructura, sino para poder competir contra los usuarios más "fieles" a las tecnologías obsoletas debido a la rapidez de respuesta del control remoto convencional.

Conociendo lo anterior, se decidió eliminar la mayor fuente de retardo del dispositivo, es decir, el servidor, con lo que resultó ser lo que se ha mostrado en este proyecto, un dispositivo que puede operar sin necesidad de un servidor.

Tras ver el diseño y los pocos requisitos que requería el dispositivo, se presentó la oportunidad de rediseñar este dispositivo pensando en personas de avanzada edad o países con dificultades de acceso a servicio técnico o Internet, por lo que el dispositivo solo requiere de conocer técnicas de soldadura básicas para poder modificarlo o repararlo, y que no sea necesario una persona especializada en electrónica para entender el funcionamiento

del dispositivo.

8.1 Relación con los estudios cursados

Se han requerido conocimientos de la asignatura de Integración de Aplicaciones para realizar el diseño de la solución tanto inicial como final, sobretodo en cuanto a conocer en profundidad el campo del *Internet of Things* y la integración de dispositivos.

Para realizar el diseño de la red, han sido necesarios los conocimientos adquiridos en la asignatura de Sistemas y Servicios en Red para poder diseñar un sistema empleando similitudes con los sistemas P2P, además de los conocimientos adquiridos en la asignatura de Redes Corporativas, los cuales han sido de gran ayuda para entender el funcionamiento de la red de forma no exclusiva en el nivel de aplicación.

Para diseñar la conectividad entre componentes del dispositivo y conocer los componentes necesarios fueron necesarios los conocimientos adquiridos en la asignatura Tecnología de Computadores y Arquitectura e Ingeniería de Computadores.

En cuanto al diseño y programación del dispositivo, fueron necesarios los conocimientos de Lenguajes, Teorías y Paradigmas de la programación para poder decidir con más exactitud las carácterísticas más óptimas para el desarrollo del software de la plataforma, además de la asignatura Hacking Ético por ayudar a la realización del software previniendo fallos de seguridad que podrían poner en riesgo la infraestructura completa, como el *Buffer overflow* [25].

Finalmente, se han empleado conocimientos tanto de la asignatura Integración de Aplicaciones como de los talleres educativos realizados por parte de la *Association for Computing Machinery* en la universidad (ACM-UPV) [26] para el conocimiento de la plataforma Arduino y el desarrollo sobre la misma, ademas de sus vastas posibilidades de desarrollo dentro del sector del IoT.

Bibliografía

[1] Asistente de voz Google Home https://store.google.com/es/product/google_home.

[2] Accesorio para la televisión Apple TV https://www.apple.com/es/tv/.

[3] Logitech https://www.logitech.com/.

[4] Broadlink RM Mini3 https://www.broadlink.com.es/broadlink-rm-mini3-domotica-mando-distancia-universal.html.

[5] Broadlink https://www.broadlink.com.es/.

[6] OWSOO eWeLink IR Hub en Amazon https://www.amazon.es/Habilitado-Infrarrojo-Universal-Acondicionado-Compatible/dp/B07KXFT86Q.

[7] eWeLink https://www.ewelink.cc/en/.

[8] SB-Projects protocolo NEC https://www.sbprojects.net/knowledge/ir/nec.php.

[9] SB-Projects protocolo SONY SIRC https://www.sbprojects.net/knowledge/ir/sirc.php.

[10] SB-Projects protocolo Philips RC-5 https://www.sbprojects.net/knowledge/ir/rc5.php.

[11] Sitio oficial Arduino https://www.arduino.cc/.

[12] Arduino MKR 1010 WiFi https://store.arduino.cc/mkr-wifi-1010.

[13] Transistor Vishay IRF520 https://www.vishay.com/docs/91017/91017.pdf.

[14] Emisor de radiofrecuencia FS1000A y receptor XY-MK-5V https://www.luisllamas.es/comunicacion-inalambrica-en-arduino-con-modulos-rf-433mhz/.

[15] Página oficial de GitHub https://github.com/.

[16] Arduino - WiFiNINA https://www.arduino.cc/en/Reference/WiFiNINA.

[17] Repositorio GitHub IRLib2 de *cyborg5* https://github.com/cyborg5/IRLib2.

[18] Repositorio GitHub rc-switch de *sui77* https://github.com/sui77/rc-switch.

[19] IDE oficial de Arduino https://www.arduino.cc/en/main/software.

[20] RFC 793 - Transmission Control Protocol https://tools.ietf.org/html/rfc793.

[21] RFC 768 - User Datagram Protocol https://www.ietf.org/rfc/rfc768.txt.

[22] RFC 5694 - Peer-to-Peer (P2P) architecture https://tools.ietf.org/html/rfc5694.

[23] Televisión LG DM2350D https://www.lg.com/es/monitores/lg-DM2350D-3d.

[24] Enchufe RoHS PCH005723 por Hidalgos Group http://www.hidalgosgroup.com/.

[25] CWE-121 Stack-based buffer overflow https://cwe.mitre.org/data/definitions/121.html.

[26] ACM UPV Chapter https://acmupv.webs.upv.es/.

[27] Licencia Apache 2.0 https://www.apache.org/licenses/LICENSE-2.0.

APÉNDICE A
Configuración del sistema

Antes de poder utilizar el dispositivo, es necesaria una configuración previa para realizar la conexión a la red y establecer un identificador para el mismo.

Adicionalmente, dado que el protocolo empleado es uno de desarrollo propio, es conveniente especificar el uso y la sintaxis del mismo.

A.1 Fase de inicialización

Previo a realizar la compilación del dispositivo es necesario crear un archivo llamado "secrets.h" el cual debe tener la siguiente información en su interior.

```
#define WIFI_NAME "Nombre del WiFi"
#define WIFI_PASS "Contraseña del WiFi"
#define DEVICE_NAME "Nombre del dispositivo"
```

Donde se deben reemplazar los valores entre comillas por sus respectivos valores.

Para configurar el controlador, es necesario reemplazar dentro de la función *generateDevices()* (véase Figura A.1) en la asignación de la variable "char devices[]", el valor establecido entre comillas por el nombre del dispositivo establecido anteriormente.Esta última modificación es necesaria debido a que el controlador aun se encuentra en fase experimental.

```
void generateDevices(){
   char devices[] = "Sample text 1";
   char *operations[] = {"IR", "RF"};
   char ipv4[] = "0.0.0.0";
   strcpy(devs[0].name, devices);
   for(int i = 0; i < (sizeof(devs[0].operations)/
sizeof(devs[0].operations[0])) && i < (sizeof(operations)/
sizeof(operations[0])); i++) strcpy(devs[0].operations[i],
operations[i]);
   strcpy(devs[0].ipv4, ipv4);
   devs[0].num_ops = 2;
}
```

Figura A.1: Código a modificar en el controlador

A.2 Protocolo de red

En cuanto al protocolo que permite el intercambio de mensajes entre la interfaz y el controlador, presenta la siguiente sintaxis (véase Figura A.2), donde se puede observar que tiene un campo para la versión del protocolo, otro campo para determinar el protocolo sobre el cual se enviará el mensaje de entre todos los soportados por el dispositivo, y por último se especifica el código que se emitirá a través de la interfaz seleccionada. Para separar dichos campos, se emplea el carácter *&*, el cual es sencillo de comprender su funcionalidad. Para finalizar el mensaje, se debe terminar con dos *bytes* nulos, es decir 0x0000.

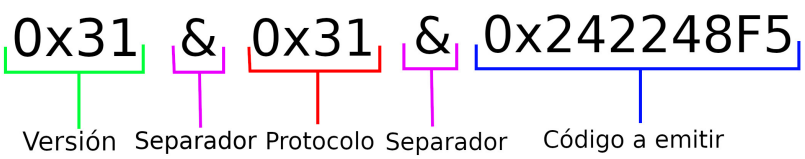

Figura A.2: Protocolo de transmisión de datos entre dispositivos

La respuesta presenta dos campos separados por el mismo separador que en el caso anterior. El primer campo presenta el código de error, los cuales son significativamente semejantes a los del protocolo HTTP, y el segundo campo es una descripción del error para que el error pueda ser legible por personas. El código enviado será correcto si y solo si el código devuelto es un 200.

Figura A.3: Mensaje de error en el protocolo de transmisión de datos entre dispositivos

En cuanto al protocolo de descubrimiento de dispositivos, solo se necesita un mensaje, el cual es copiado al generar una respuesta. Este mensaje dispone también de dos campos separados por el separador de los mensajes anteriores. El primer campo es la versión del protocolo, la cual actualmente es la 1, y el segundo campo es el identificador del dispositivo, el cual se estableció en Sección A.1. Si el identificador no coincide con el del dispositivo, el dispositivo no responderá al protocolo de descubrimiento, con lo que permanecerá oculto.

Figura A.4: Protocolo de descubrimiento de dispositivos

A.3 Respositorio del código

El código del dispositivo puede ser encontrado bajo el nombre de IRF-Controller en GitHub [15] y en la siguiente dirección `https://github.com/Bullcaos/IRF-controller`.

El código es *open-source* goza de una licencia Apache 2.0 [27].

APÉNDICE B
Proyectos futuros

Durante el desarrollo del dispositivo, se planteó la idea de transformar todos los dispositivos que no disponen de conectividad a la red, como cafeteras o robots de cocina, para permitir disfrutar de la comodidad proporcionada por la red IoT que se ha planteado en este proyecto sin necesidad de obtener costes descomunales e inasequibles, siguiendo el patrón planteado, es decir, simplicidad de componentes y código abierto. De esta forma es posible crear dispositivos IoT para cualquier tipo de entorno y ámbito.

Adicionalmente se planteó la idea de crear un controlador universal que permitiese la conexión de cualquier dispositivo de la casa con una aplicación unificada, idependientemente de la marca o la interfaz empleada, de modo que se unifican todas las operaciones domésticas y es posible realizar una interacción entre ellas, como, por ejemplo, establecer una alarma en un Amazon Echo desde un Google Home.