UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Diseño e implementación de un sistema de alarma basado en beacons Bluetooth

Carlos Fernández Cofrades
Tutor: Oscar Delgado Mohatar
Ponente: Eloy Anguiano Rey

JULIO 2017

Diseño e implementación de un sistema de alarma basado en beacons Bluetooth

AUTOR: Carlos Fernández Cofrades
TUTOR: Oscar Delgado Mohatar
PONENTE: Eloy Anguiano Rey

Dpto. Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Julio de 2017

Resumen (castellano)

Un problema generalizado en los sistemas de alarmas actuales es la intrusión de ladrones en garajes comunitarios aprovechando la entrada de algún vecino: al entrar el vecino en el garaje, el ladrón se esconde tras de él y aprovecha el tiempo que tarda en cerrarse la puerta para entrar al garaje sin ser detectado. De este modo, cuando el vecino se haya ido, el ladrón podrá moverse por el garaje y realizar el robo. Para salir del garaje, el intruso podrá usar la misma técnica que usó al entrar.

En este Trabajo de Fin de Grado se ha diseñado e implementado un sistema de alarma basado en beacons Bluetooth. Este desarrollo estará enfocado en la detección de intrusos dentro de un garaje comunitario a través de una serie de sensores de movimiento conectados a una Raspberry Pi. Los usuarios legítimos para entrar al garaje (vecinos) estarán previamente registrados en el sistema y se les asignará un beacon individual e intransferible a cada uno.

El sistema contará con una "centralita" web en la que el administrador podrá consultar el registro de actividad (salidas/entradas de vecinos e intrusos), cambiar el beacon asociado a un vecino, dar de baja/alta a un vecino y configurar los usuarios que recibirán las notificaciones (mensajes a través de la aplicación telegram) cuando se detecte un intruso en el garaje. Estos posibles receptores serán 1) sólo el administrador, 2) el último vecino en entrar o 3) todos (el administrador y los vecinos).

La funcionalidad del sistema será la siguiente: si el sistema detecta un movimiento dentro del garaje se procederá a chequear si hay algún beacon cerca (vecino autorizado). En el caso de que se detecte un beacon, el sistema solamente actualizará el registro de actividad. En caso de que no detecte ningún beacon cerca se activará la alarma (luces y sonidos), se enviará un mensaje mediante la aplicación de telegram a los usuarios especificados previamente (mediante la web) y actualizará el registro de actividad.

De esta forma se resolverá el problema citado al principio: el sistema detectará la entrada de un vecino cuando el ladrón se cuele en el garaje, pero cuando el vecino se haya ido y el intruso se mueva dentro del garaje, el sistema ya no detectará al vecino y saltará la alarma.

Palabras clave (castellano)

Alarma, beacon, bluetooth, RaspberryPi, Python, PIR, sensor de movimiento, cherrypy, bootstrap.

Abstract (English)

At current alarm systems, there's a problem in community garages with thieves that take advantage of a neighbor's entrance: when the neighbor enters the garage, the thief hides behind him and take advantage while the door is closing to enter into the garage without being detected. In this way, the thief can move and steal in the garage when the neighbor is gone. To exit the garage, the thief can use the technique used at entrance.

In this thesis we have designed and implemented an alarm system based on bluetooth beacons. This system will use motion sensors connected to a Raspberry Pi. Firstly, neighbors will be logged in the system and will be assigned a beacon to each.

The system have a web in which the administrator can see the activity log (entrance and exit of neighbors and thieves), change the neighbor's beacon, unregister/register a neighbor and configure the users that will receive the notifications (messages via telegram app) when an intruder is detected inside the garage. The posible receivers will be 1) only the administrator, 2) the last neighbor who entered or 3) everyone (administrator and neighbors).

The system will work as follows: if a movement is detected inside the garage, the system will check for nearby beacons (authorized neighbor). If a beacon is detected, the system only update the activity log. Otherwise (no beacon nearby detected) the system active the alarm (lights and sounds), a message will be sent by the telegram app to the previously specified users (via web) and the activity log will be updated.

This will solve the problem mentioned at the beginning. The system will detect the entrance of a neighbor when the thief enter into the garage, but the alarm will be activated when the neighbor has left and the thief moves inside the garage.

Keywords (inglés)

Alarm, beacon, bluetooth, RaspberryPi, Python, PIR, motion sensor, cherrypy, bootstrap.

Agradecimientos

En primer lugar quisiera agradecer a mi familia por haberme apoyado, respetado y ayudado siempre en las decisiones que he ido tomando.

También querría agradecer a los compañeros que he ido conociendo en la carrera, gracias a los cuales he pasado unos cuatro años bastante amenos.

Y mención especial a mi tutor, al cual agradezco que me haya dado la oportunidad de llevar a cabo el desarrollo de esta gran idea.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

1 Introducción

1.1 Motivación

Hoy en día existen multitud de sistemas de alarmas para evitar que personas malintencionadas entren en un garaje comunitario privado y comiencen a sustraer objetos de valor (bicicletas, gps, piezas de coches, radios…). Comentando el problema con la Policía de Alcobendas se expuso un problema generalizado que solían tener los sistemas de alarmas actuales: los ladrones aprovechan la entrada y la salida de algún vecino autorizado para colarse justo detrás ("piggybacking") y evitar ser detectado por el sistema de alarma. De esta manera, con los sistemas de alarma actuales, un ladrón podrá moverse con relativa libertad por el garaje cuando no haya ningún vecino cerca.

A raíz de esta conversación surgió la idea de implementar un sistema de alarma que detectara de alguna manera si un movimiento dentro del garaje se correspondía con el de un vecino o el de un ladrón. La solución a este problema pasaría por que cada vecino tuviera en su posesión (colgado en el llavero con el mando del garaje) algún dispositivo con el que el sistema le identificara como usuario autorizado. De esta manera, si se detecta algún movimiento dentro del garaje y el sistema no encuentra ningún dispositivo de identificación cerca, el intruso será detectado por el sistema.

1.2 Objetivos

- El principal objetivo de la realización de este TFG es minimizar la probabilidad de que un robo en un garaje comunitario se lleve a cabo satisfactoriamente.

A partir de éste podemos desglosar otros dos objetivos más específicos de nuestro sistema:

- Por una parte minimizar las posibles vulnerabilidades de los sistemas de alarmas actuales. En particular, nuestro sistema minimizará el riesgo de que no sean detectados los intrusos que entren con la técnica de "piggybacking" mencionada anteriormente.
- Por otra parte, con este sistema se mantendrá informados al instante al administrador y a los vecinos de la actividad (legítima e ilegítima) dentro del garaje a través de un registro de actividad (web) y por notificaciones en el móvil (mensajes en la aplicación telegram).

1.3 Organización de la memoria

La memoria básicamente constará de los siguientes tres apartados:

- **Estado del arte:** En este apartado estudiaremos diferentes sistemas de alarmas que existen actualmente en el mercado y los compararemos con el nuestro mostrando ventajas y desventajas. En esta sección tendremos una segunda subsección en la que explicaremos las distintas tecnologías utilizadas en este proyecto.
- **Diseño:** Este apartado irá enfocado a explicar los requisitos funcionales y no funcionales de nuestro proyecto (tanto de la aplicación como de la web).
- **Desarrollo:** A continuación explicaremos con detalle todo lo relativo al desarrollo de los diferentes componentes de nuestro sistema: las tablas de la base de datos, las herramientas y secciones la web, las herramientas y funcionalidades de la aplicación y adjuntaremos el esquemático del circuito eléctrico del sistema.

2 Estado del arte

2.1 Sistemas actualmente en el mercado

2.1.1 Segurparking

2.1.1.1 Mandos Mio y SEGURphone

Este sistema propone solucionar el acceso de ladrones al garaje con llaves/mandos perdidos o copias de los mismos. La solución planteada consiste en que cada vecino posee un mando personal con su nombre, número de plaza y un código único e intransferible (ID) gracias al cual se irán registrando las entradas y salidas del parking. Este mando se podrá dar de baja (en caso de robo) y no se podrá copiar (ID único) [1].

Solución de nuestro sistema al problema planteado: nuestro sistema alertará de un intruso en el caso de que entre algún ladrón con llaves/mandos robados o copiados: se detectará el movimiento del ladrón y al no presenciar ningún beacon cerca, saltará la alarma. En el caso de que lo robado sea el beacon, éste se podrá desactivar inmediatamente desde la web de la administración. Los beacons son únicos (ID único), por lo que no se podrán copiar.

2.1.1.2 SEGURdoor

Esta misma empresa ha implementado una solución al "piggybacking" con la integración de las cámaras de seguridad del garaje. Lo que hará este sistema será disparar una señal de alarma cuando el intruso trate de colarse en el garaje aprovechando el paso detrás de un vehículo autorizado. En el caso de tener contratado un servicio de Central Receptora de Alarmas, al detectar un intruso, el sistema enviará una alerta a la central y se podrá realizar un seguimiento del incidente [2].

Parece ser que esta implementación está basada en una conexión de unas fotocélulas con las cámaras de seguridad del parking. Estas fotocélulas activarían las cámaras cada vez que la puerta automática se abra.

Pros de nuestro sistema respecto a SEGURdoor: respecto a la conexión con las fotocélulas no sería muy difícil saltárselas o desviar el rayo y entrar sin problemas. Nuestro sistema es seguro independientemente de cómo haya entrado el ladrón, ya que también podría entrar por las escaleras y no por la puerta de los coches.

2.1.2 Securitas Direct (alarmas para garajes)

Para garantizar la seguridad del garaje este sistema de alarma contará sensores de apertura para controlar el estado de la puerta del garaje y fotodetectores con sensores de movimiento y cámara de fotos [3].

Los sensores de apertura se instalarán en los puntos de entrada para evitar posibles manipulaciones de estos accesos de entrada. Al detectar la vibración producida por el intento de intrusión del ladrón, el sistema activará la alarma y así se evitarán las intrusiones por medio de la manipulación de cerraduras o perforación de cristales [4].

Los fotodetectores utilizados en este sistema cuentan con un sensor de movimiento por infrarrojos, con una microcámara y con un flash para las posibles intrusiones por la noche. Estos fotodetectores detectarán movimiento y enviarán el aviso y las imágenes captadas a la

Central Receptora de Alarmas (CRA) de Securitas Direct, y allí ya llevarán a cabo el determinado protocolo de actuación [5].

Pros de nuestro sistema: La pega que vemos al sistema citado de Securitas Direct es el método de detección de intrusos. Dado que el sistema no está enfocado a una comunidad de vecinos si no a un domicilio particular, la simple detección de un movimiento con el sistema activado, hará saltar la alarma y alertar a la CRA. Nuestro sistema no está enfocado a domicilios particulares, pero con nuestro sistema no haría falta activar la alarma: directamente al detectar un movimiento y no chequear ningún beacon cerca (no habrá ningún miembro de la familia cerca) se activaría la alarma detectando al intruso y no haría falta la existencia de ninguna CRA.

Contras de nuestro sistema: La principal debilidad de nuestro sistema respecto al de Securitas Direct es la posible detección del intruso antes de que entre (a través de los sensores de apertura).

2.1.3 Prevent Security Systems

En este sistema de alarma será el vecino directamente el que alertará de la intrusión de algún ladrón en el garaje. Este proceso se llevará a cabo mediante un mando SOS del garaje que activará la conexión de un operador de la central de alarmas. Este operador se conectará a las cámaras del garaje y actuará en consecuencia (llamando a la policía por ejemplo) guardando un registro de las imágenes de las cámaras [6]. En cuanto a los beneficios que presenta este sistema cabe destacar:

- Identifica los vehículos y personas que acceden al garaje de la comunidad.
- Elimina el descontrol de mandos originado por pérdidas y cambios de inquilinos.
- En situación de emergencia identifica a los autores de los actos vandálicos mediante las imágenes grabadas.
- Gestiona de manera sencilla la baja de los mandos.
- Detecta al instante si existe un intento de forzar la puerta del garaje.
- Solicita ayuda en caso de emergencia o asalto pulsando el botón SOS.
- Impide que los antiguos inquilinos/intrusos accedan al inmueble.

Nuestro sistema también impide el acceso de intrusos no autorizados que traten de entrar al garaje con mandos (beacons) perdidos/robados o que haya formado parte de la comunidad anteriormente, ya que desde la web del administrador se podrán dar de baja de inmediato.

Pros de nuestro sistema: El principal problema que vemos en este sistema es que la detección de intrusos no está automatizada (es un vecino el que tendrá que dar la señal de alarma a través del mando SOS). En nuestro sistema la alarma se dispara automáticamente en el caso de detectar el movimiento de un intruso. También podemos comprobar que este sistema no soluciona el problema del "piggybacking": el ladrón puede colarse detrás del coche de un vecino autorizado tanto para entrar como para salir del garaje. Nuestro sistema alertará cuando detecte un movimiento ilegítimo dentro del garaje.

Contras de nuestro sistema: Nuestro sistema no alertará hasta que el vecino se haya alejado lo suficiente.

2.1.4 Altatec Seguridad

Este sistema de alarma ofrece una solución bastante completa a los robos en viviendas garajes o trasteros, contando con 1) un sistema de videovigilancia con cámaras de alta resolución en las entradas de personas y coches que identificarán rostros y matrículas, 2) un control de acceso a través de mandos encriptados, 3) código de seguridad o biometría en las entradas al garaje y 4) una central de alarmas con conexión a la policía las 24h del día [7].

Pros de nuestro sistema: La posible solución de este sistema al problema planteado del "piggybacking" es la instalación de cámaras de seguridad. El sistema de videovigilancia además de poder ser vulnerado, ha de estar continuamente supervisado por una persona (la detección de intrusos no está automatizada). En nuestro sistema no haría falta ese personal adicional ya que la alarma saltaría automáticamente cuando el ladrón se quedara sólo en el garaje.

2.1.5 Resumen

Nombre	Segurparking	Securitas Direct	Prevent	Altatec
Descripción	Contiene un sistema que impide el acceso con llaves o mandos robados y otro que integra cámaras de seguridad y un servicio de Central Receptora de Alarmas.	Sistema basado en fotodetectores con sensor de movimiento y cámara conectados a una Central Receptora de Alarmas. Preveé la intrusión de un ladrón con sensores de apertura.	Sistema en el que el vecino alertará a un operador de la intrusión de algún ladrón en el garaje. Impide el acceso con mandos robados o perdidos.	Sistema de videovigilancia, con control de acceso y conexión con una central de alarmas.
Ventajas de nuestro sistema	Nuestro sistema es seguro independientemente de cómo haya entrado el ladrón	En nuestro sistema no hay ningún proceso de activación, la alarma alertará en cuanto se hayan alejado los usuarios legítimos	Nuestro sistema alertará de la intrusión de un ladrón automáticamente	Nuestro sistema está automatizado y no requiere de personal supervisando continuamente

Tabla 2-1: Comparativa de los sistemas actuales con el nuestro

2.2 Arquitectura del sistema

En este apartado detallaremos las distintas tecnologías utilizadas en nuestro sistema.

El sistema está formado, a grandes rasgos, de una parte software y otra hardware. La primera se compone de dos frameworks para la web (CherryPy y Bootstrap), de una API de telegram para integrar funcionalidad de esta aplicación al sistema y de una librería bluetooth para poder escanear los beacons de los usuarios. En cuanto a la parte hardware contaremos con una RaspberryPi como centro de nuestro sistema, con los beacons que llevarán los vecinos y con sensores de movimiento.

En el siguiente gráfico podemos ver un resumen de la arquitectura principal de nuestro sistema subdividida en elementos Software y elementos Hardware.

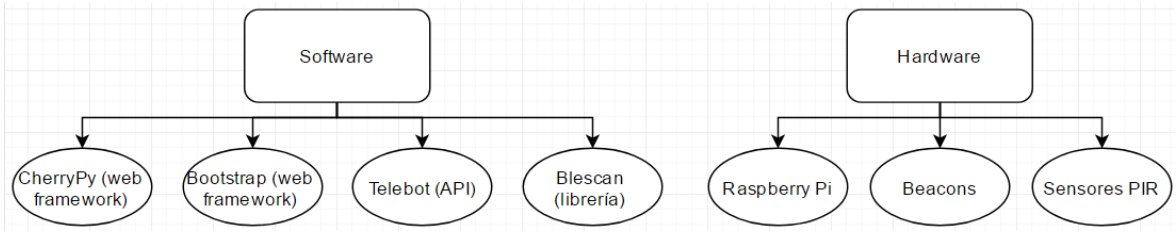

Figura 2-1: Arquitectura principal del sistema

2.2.1 Web

2.2.1.1 CherryPy

CherryPy es un framework de desarrollo web de Python que permite crear aplicaciones web de una manera similar al desarrollo de un programa en python [8]. Este framework nos va a permitir desarrollar nuestra web e integrarla con el código de nuestra aplicación de una manera mucho más sencilla.

Entre las principales ventajas de hacer uso de este framework están:

- CherryPy incluye su propio servidor HTTP para alojar la aplicación.
- La simpleza con la que es posible generar una página web.
- Ventajas relativas a Python: lenguaje dinámico, con un extenso catálogo de librerías y con una gran flexibilidad para integrar herramientas y plugins de terceros (por ejemplo en nuestro caso la librería 'blescan' para la conexión bluetooth).

2.2.1.2 Bootstrap

Bootstrap es el framework más popular para el diseño de sitios y páginas web basado en HTML, CSS y JavaScript [9]. Con este framework conseguiremos un mejor diseño en nuestra página web directamente enlazando las hojas de estilo CSS a nuestros archivos HTML.

Para usar este framework tan sólo habrá que descargar las hojas de estilo, elegir la más apropiada para nuestra página web y referenciarla en el HTML.

2.2.2 Aplicación (software)

2.2.2.1 Telebot (API Telegram)

La API de telegram que utilizaremos para enviar y recibir mensajes (comandos) será Telebot [10]. Esta API está desarrollada para Python y utilizará un bot creado previamente para enviar mensajes desde nuestro programa (al detectar un intruso) [11]. Otra funcionalidad que aprovecharemos de esta API será la gestión de comandos que permitirá realizar un programa en Python que responda a esos comandos (en nuestro caso: ejecutar el programa y salir del mismo).

Para utilizar esta API simplemente tendremos que instalar la librería, importarla e inicializar la variable de telegram, que utilizaremos para enviar mensajes, con el TOKEN de nuestro bot.

2.2.2.2 Librería blescan (bluetooth)

Esta librería nos permitirá escanear los beacons que haya dentro del rango bluetooth [12]. En particular la función de "parseo" de esta librería nos devolverá la lista de beacons que haya escaneado en el rango (UUID, RSSI…).

2.2.3 Aplicación (hardware)

2.2.3.1 Raspberry Pi

Raspberry Pi es un mini ordenador de tipo SBC (de placa reducida) que podremos utilizar para proyectos de electrónica, para usarlo como servidor o en general para realizar muchas de las acciones que se podrían realizar con un PC convencional [13].

En nuestro caso hemos usado la Raspberry Pi 3 Modelo B, con las siguientes características a destacar: [14]

- CPU ARMv8 de 4 núcleos a 1.2GHz de 64-bit
- WiFi 802.11n
- Bluetooth 4.1
- 1GB RAM
- 4 puertos USB
- 40 pins GPIO
- Puerto Ethernet
- Puerto HDMI
- Ranura para introducir una tarjeta microSD que actuará como memoria interna y como alojadora del Sistema Operativo que queramos usar (en nuestro caso Raspbian).

En nuestro proyecto utilizaremos la Raspberry Pi como "central" de nuestro circuito electrónico (recibiendo información de los sensores y mandando señales a los dispositivos de salida) y como servidor que alojará nuestra web.

El hecho de que ya tenga bluetooth integrado nos permite prescindir de un módulo bluetooth externo para la conexión con los beacons.

Figura 2-2: Raspberry Pi 3 Model B

2.2.3.2 Beacons

Un beacon es un pequeño dispositivo de bajo consumo basados en la tecnología bluetooth, que emiten una señal que recibirá otro dispositivo (en nuestro caso la RaspberryPi). El modelo de beacon que hemos utilizado para nuestro proyecto es el nRF51822 de Nordic Semiconductor [15].

Las características a destacar de este dispositivo son las siguientes:

- Su tamaño es muy pequeño (por lo que podrá llevarse en el llavero con las llaves del garaje)
- Su arranque y configuración son bastante sencillos (poner pila y dar al botón)
- La identificación unitaria de cada beacon por medio de su identificador único universal (UUID)
- Su consumo es ínfimo (la batería de la pila puede llegar a durar hasta dos años)
- La intensidad de la señal (RSSI) puede determinar a la distancia a la que se encuentra el beacon (aunque nosotros no la usamos en este proyecto, quizá estaría bien incorporar esta característica para trabajo futuro)

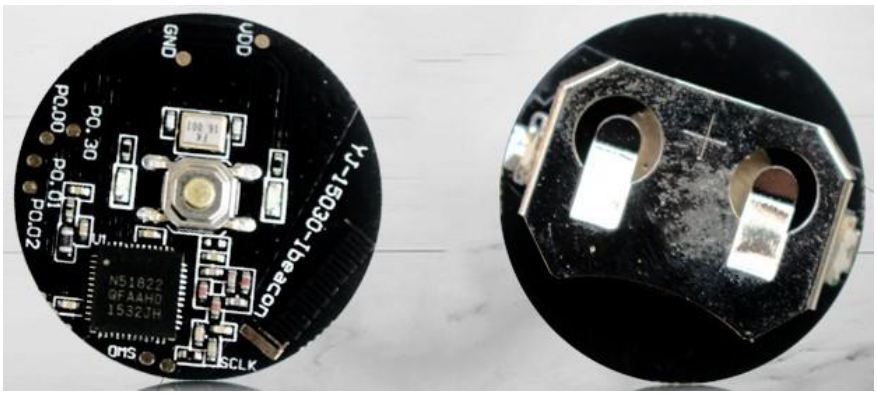

Figura 2-3: Beacon nRF51822

2.2.3.3 Sensores de movimiento

Un sensor de movimiento es un dispositivo electrónico que reacciona a un movimiento físico generando una señal que recibirá el programa central para actuar en consecuencia a ese movimiento.

Los sensores que utilizaremos en nuestro proyecto serán sensores pasivos infrarrojos (PIR). Estos sensores medirán la luz infrarroja emitida por los objetos situados en su campo de visión. En particular usaremos el modelo HC-SR501 cuyo rango de detección será de unos 120º a una distancia máxima de unos 7 metros [16].

Este sensor irá cableado hasta la Raspberry Pi debido a su mayor fiabilidad, mayor seguridad y mayor velocidad de transmisión con respecto a las ondas de radio inalámbricas.

Figura 2-4: Sensor PIR HC-SR501

3 Diseño

3.1 Requisitos funcionales

3.1.1 Activación de alarma al detectar intruso

El principal servicio que proveerá nuestro sistema es el propio de un sistema de alarma: detectar un intruso y hacer saltar la alarma.

Como ya hemos comentado anteriormente, el sistema detectará un movimiento a través del sensor PIR y si no chequea ningún beacon autorizado cerca se activará la alarma. Al activar la alarma lo que haremos será iluminar un led rojo y hacer sonar un buzzer de manera intermitente con el objetivo de disuadir al ladrón y alertar a cualquier persona que se percate del sonido y las luces.

3.1.2 Alta de vecinos

Cualquier vecino que vaya a formar parte del sistema, tendrá que haber sido dado de alta previamente por el administrador. El administrador dará de alta a un vecino a través del formulario de la web, en el que tendrá que rellenar los siguientes campos: nombre de usuario, información de contacto como la dirección de correo y el número de teléfono,

- Nombre de usuario
- Dirección de correo electrónico
- Número de teléfono
- Id del chat al que se enviarán las notificaciones de intrusión a través del bot de telegram.
- Beacon asociado
- Dirección: portal, piso y número

3.1.3 Baja de vecinos

Todo vecino que se marche de la comunidad o deje de usar el garaje comunitario deberá darse de baja del sistema. La comunidad notificará al administrador del sistema de la marcha del vecino en cuestión y desde la lista de vecinos de la web se le dará de baja. Este proceso será necesario como medida de seguridad para evitar posteriores intrusiones no autorizadas.

3.1.4 Cambio de beacon asociado

Ante el problema de las pérdidas, robos o avería de los beacons, el sistema ofrece la posibilidad de cambiar el beacon asociado de cada vecino. El vecino que quiera cambiar el beacon tendrá que notificar al administrador y éste realizará el proceso de cambio de beacon desde la lista de vecinos de la web.

3.2 Requisitos no funcionales

3.2.1 Registro de actividad

Para ofrecer una mayor seguridad, el sistema llevará a cabo un registro de actividad: registro de los movimientos detectados (legítimos e ilegítimos) por el sensor PIR. En este registro de actividad se mostrará la fecha y un mensaje informativo (acompañado del beacon detectado si se trata de un vecino).

3.2.2 Notificación de intrusos a través de telegram

Mediante la integración de la API de telegram a nuestro proyecto, el sistema alertará a los usuarios mediante la aplicación de telegram cuando se detecte algún movimiento ilegítimo dentro del garaje. Los usuarios que serán notificados los elegirá el administrador a través de la web. Los posibles receptores podrán ser: 1) sólo el administrador; 2) el último vecino en entrar; 3) el administrador y todos los vecinos.

Dicha notificación se llevará a cabo desde un bot de telegram creado previamente hacia los ids de los chats de los teléfonos móviles de los receptores (administrador y vecinos).

3.2.3 Manejo de la ejecución mediante comandos de telegram

Aprovechando la integración de la API de telegram hemos decidido manejar la ejecución de nuestro programa principal mediante comandos al bot de telegram. Los posibles comandos que se podrán enviar al bot serán ‘ejecutar’ y ‘salir’ para ejecutar el programa y salir de él. Esta funcionalidad nos permitirá realizar reparaciones y actualizaciones en el sistema con una mayor agilidad.

3.2.4 Cambio de contraseña

Dado que para acceder a la web, el administrador tendrá que iniciar sesión con su usuario y contraseña, éste tendrá una opción para cambiar de contraseña.

4 Desarrollo

4.1 Base de datos (alarm.db)

4.1.1 Tablas

4.1.1.1 Admin

Esta tabla contiene el nombre y la contraseña del administrador necesarios para iniciar sesión en la web. También guardaremos en esta tabla el id del chat por el que se enviarán los mensajes a través de la API de telegram.

4.1.1.2 Neighbors

Esta tabla contiene los datos de los vecinos registrados en el sistema: nombre de usuario, correo electrónico, teléfono, chat id, dirección (portal, piso y número) y beacon asociado.

4.1.1.3 Alerts

En esta tabla se irá guardando el registro de actividad (cada movimiento que detecte el sensor). Dispondremos de un campo para identificar si se trata de un intruso o no, otros dos campos que guardarán la fecha (el día y la hora) de dicha alerta y un último campo en el que se guardará el beacon detectado en caso de que el movimiento detectado sea el de un vecino autorizado.

4.1.1.4 Receiver

Por último, esta tabla determinará los usuarios que recibirán un mensaje a través de telegram en caso de que se detecte un intruso dentro del garaje. Los posibles receptores podrán ser: 1) sólo el administrador, 2) sólo el último vecino o 3) el administrador y todos los vecinos.

4.2 Web

A continuación describiremos detalladamente cada sección de la web del administrador.

4.2.1 Inicio de sesión (index.html)

Esta pantalla contendrá el típico inicio de sesión en el que sólo podremos acceder mediante el usuario y contraseña de la tabla *admin*.

- Funciones asociadas:
    - Cualquier función excepto *login* y *logout* redireccionará a esta pantalla en caso de que no esté la sesión iniciada.
    - *login*: Esta función comprobará que el usuario y la contraseña introducidos en el formulario son correctos.

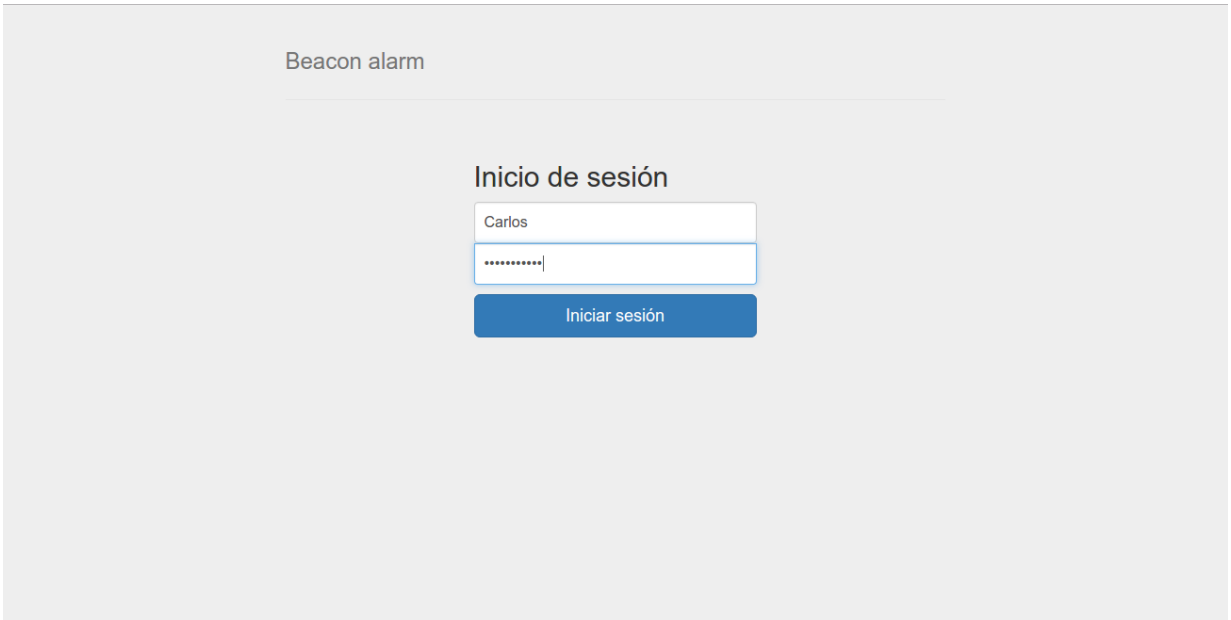

Figura 4-1: Inicio de sesión

4.2.2 Registro de actividad (activity.html)

Al iniciar sesión nos encontramos con una tabla con el registro de actividad del sistema. En esta tabla se imprimirán los registros de la tabla *alerts* de nuestra base de datos, ordenados de más recientes a más antiguos. El contenido de esta tabla será la fecha de la entrada/salida y un mensaje en función de si se trata de un intruso o un vecino.

- Funciones asociadas:
    - *index*: En esta función iteraremos sobre las alertas de la base de datos y las ordenaremos de más recientes a más antiguas. Luego seleccionaremos el mensaje a imprimir dependiendo si se trata de un intruso o no (campo *intruder* de la tabla *alerts*).

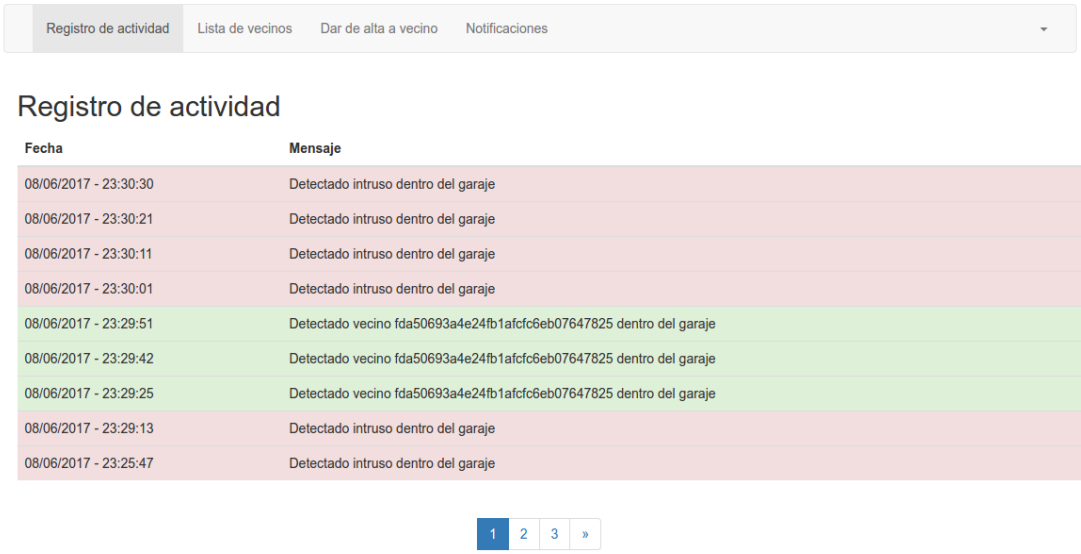

Figura 4-2: Registro de actividad

4.2.3 Lista de vecinos (users.html)

En esta sección podremos ver una tabla con la información de todos los vecinos registrados en el sistema. Esta información se obtendrá de la tabla *neighbors* de nuestra base de datos. A parte de mostrar los datos de los vecinos, también podremos cambiar el beacon asociado y dar de baja a un vecino.

- Funciones asociadas:
    - *users*: En esta función iteraremos sobre los vecinos de la base de datos guardando los diferentes datos de cada vecino que posteriormente imprimiremos en el html.
    - *deleteUser*: En esta función borraremos el vecino seleccionado de la base de datos. Esta acción se llevará a cabo cuando un vecino abandone la comunidad o solamente el garaje.
    - *changeBeacon*: Esta función modificará el beacon asociado al vecino seleccionado. Para llevar a cabo dicha acción, comprobaremos que el nuevo beacon no esté asociado a otro vecino y actualizaremos la tabla *neighbors* con el nuevo id del nuevo beacon.

Figura 4-3: Lista de vecinos

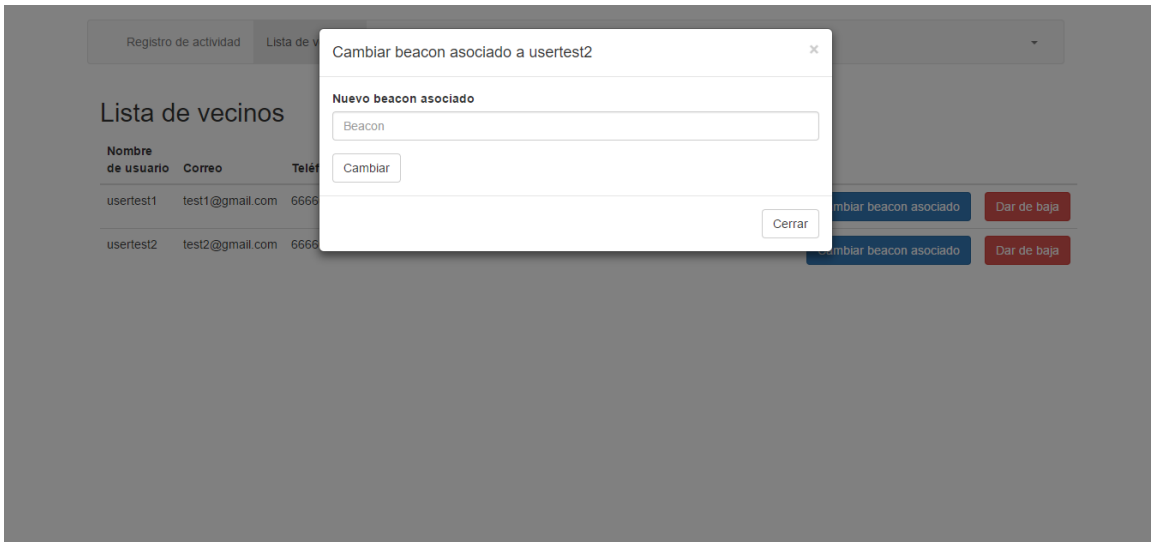

Figura 4-4: Cambiar beacon asociado a un vecino

4.2.4 Registro de vecinos (register.html)

En esta pestaña tendremos la opción de dar de alta a un nuevo vecino al sistema. Para ello habrá que rellenar un formulario con todos los datos necesarios: nombre de usuario, dirección de correo, número de teléfono, dirección y beacon asociado. Una vez dado de alta un nuevo vecino, éste aparecerá en la lista de vecinos mencionada anteriormente.

- Funciones asociadas:
    - *register*: Esta función simplemente redireccionará al html register.html.
    - *registerValidate*: Tras pulsar el botón de dar de alta, esta función comprobará que todos los campos del formulario hayan sido rellenados correctamente: que el nombre de usuario, el correo electrónico, el número de teléfono y el beacon asociado no estén registrados en la base de datos; que el número de teléfono tenga nueve dígitos; y que la dirección sea correcta (que sean dígitos y sea una dirección existente en la comunidad*)

      *Dado que no existe una comunidad para el sistema, esta comprobación está en fase *beta*.

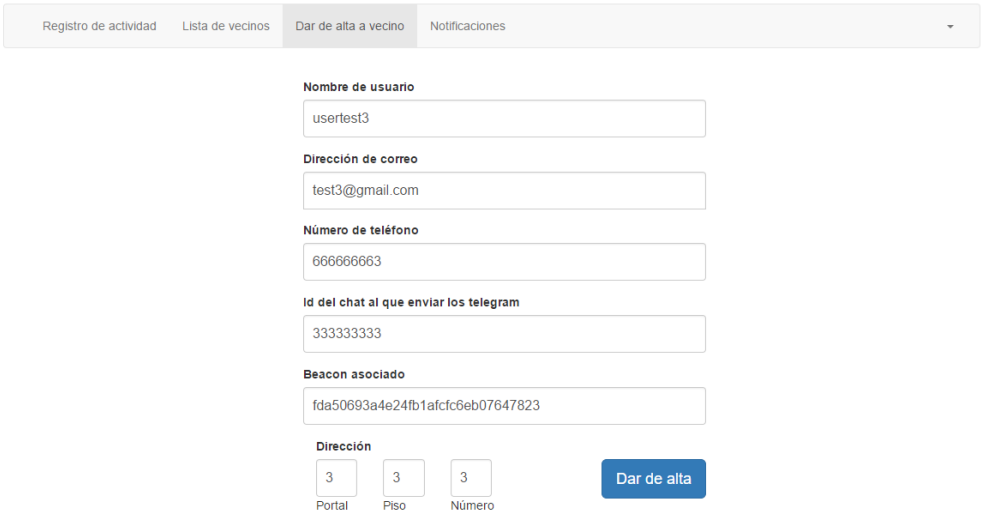

Figura 4-5: Dar de alta a un vecino

4.2.5 Configuración de las notificaciones (notifications.html)

En esta sección podremos seleccionar a través de un 'radio button' los usuarios que queremos que reciban los mensajes (vía telegram) cuando se detecte un intruso dentro del garaje.

- Funciones asociadas:
  - *notifications*: Esta función mostrará el 'radio button' dependiendo del contenido de la tabla *receiver* (chequeada la opción que tenga el valor 'checked').
  - *receiverNotifications*: Esta función actualizará la tabla *receiver* tras chequear una opción del 'radio button'.

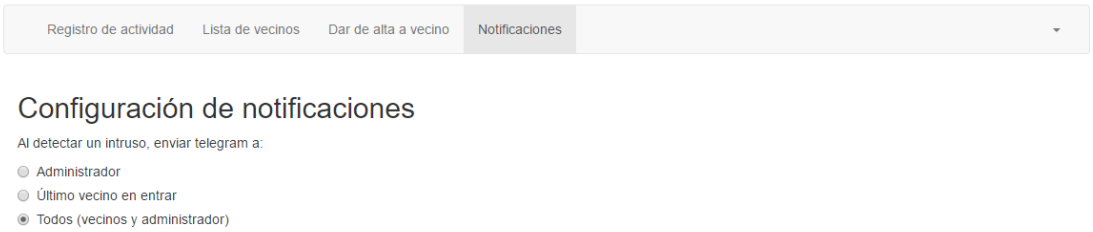

Figura 4-6: Configurar los receptores de las notificaciones

4.2.6 Otros

Por último tendremos un desplegable, visible desde cualquier sección excepto la de inicio de sesión, en el que podremos cambiar la contraseña y cerrar la sesión. Para realizar el cambio de contraseña tendremos que introducir la contraseña actual y la nueva dos veces.

- Funciones asociadas:
  - *changePassword*: Con esta función comprobaremos que la contraseña actual sea correcta, que la contraseña nueva introducida sea la misma en los dos campos y que tenga una longitud mínima de ocho caracteres.
  - *logout*: Esta función borrará la sesión 'userLogin' y volveremos a la sección de inicio de sesión.

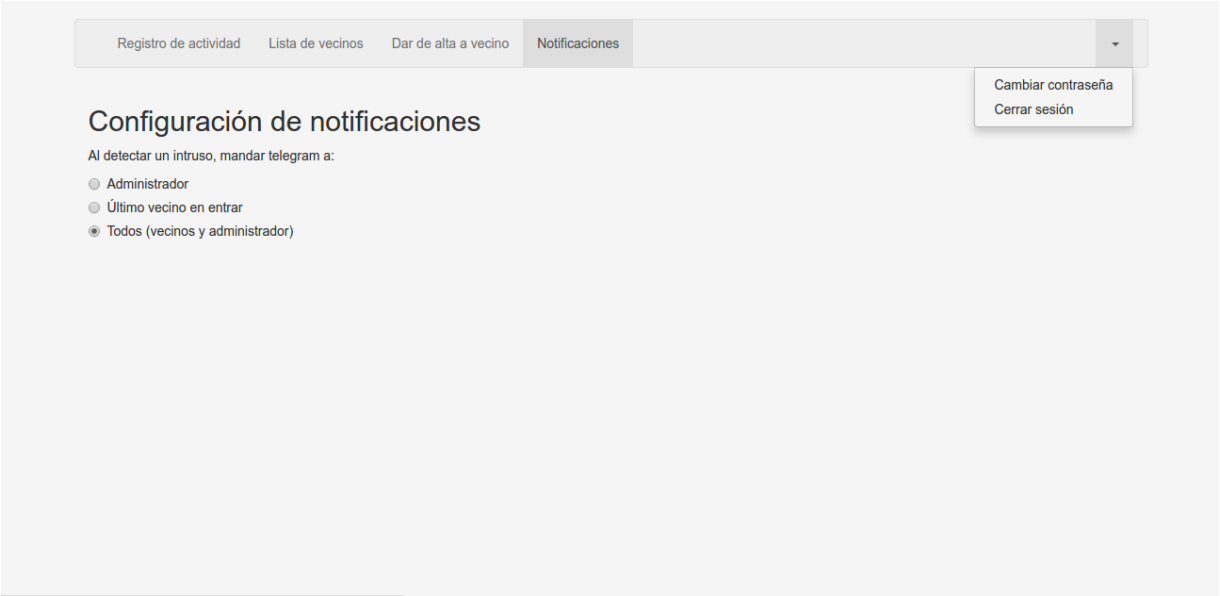

Figura 4-7: Desplegable

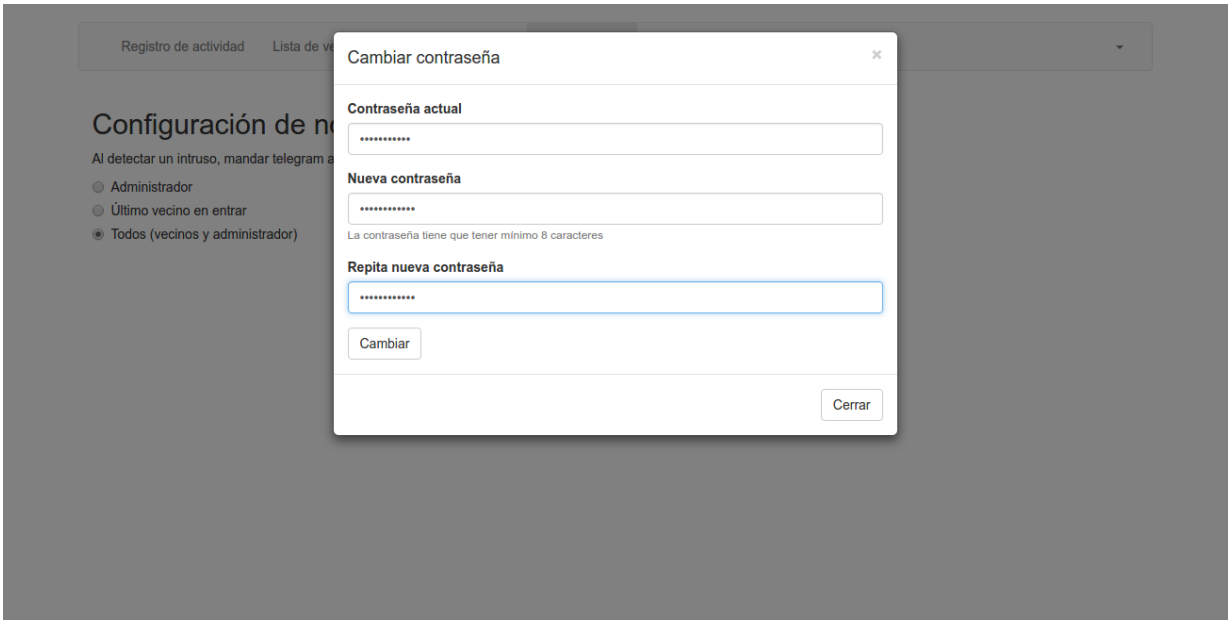

Figura 4-8: Cambiar contraseña

4.3 Aplicación

En este apartado pasaremos a explicar cómo funciona el código de nuestro proyecto alojado en github [17].

La funcionalidad que llevará a cabo nuestro programa se ha presentado en el siguiente diagrama de flujo:

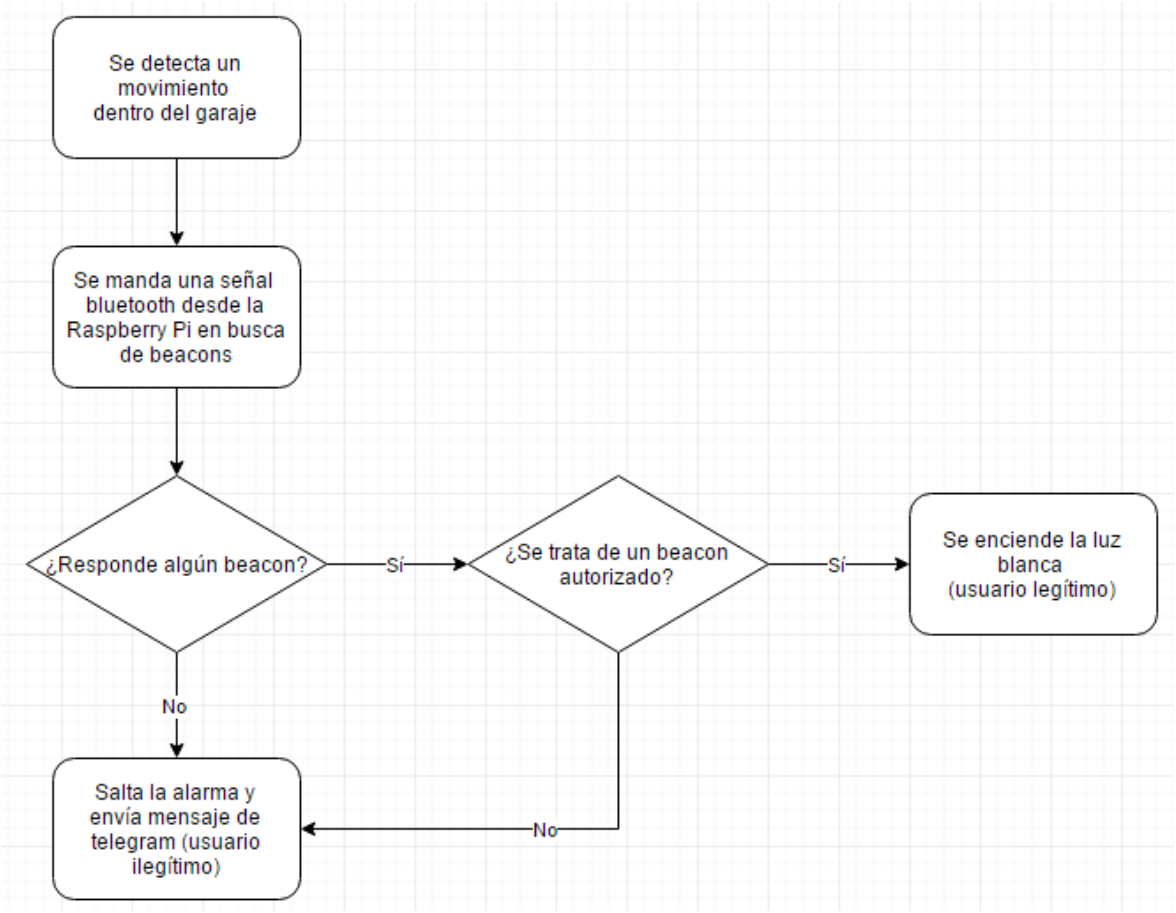

Figura 4-9: Diagrama de flujo

El programa que se ejecutará manualmente será *bot_telegram.py* para 'activar' la entrada de comandos a través de telegram por parte del administrador. Este programa irá leyendo los comandos que se le envíen a través del bot de telegram y actuará en consecuencia. Si un usuario que no es el administrador intenta ejecutar algún comando, se le enviará un mensaje denegándole el acceso (comprobación mediante el id del chat). Los comandos que admite el programa serán los siguientes:

- *ejecutar*: Ejecutaremos este comando para iniciar la ejecución del programa que activará el sistema (Proyecto.py).
- *salir*: Con este comando saldremos de la ejecución iniciada en el paso anterior.

Para no pisar una ejecución con otra utilizaremos una cola de ejecuciones manejada con contadores que se irán incrementando con cada ejecución.

En principio sólo sería necesario ejecutar el comando *salir* para realizar alguna reparación o actualización en el programa, en la web o en la base de datos.

El programa que ejecutaremos al mandar el comando *ejecutar* será *Proyecto.py*. En este programa entraremos en un bucle infinito hasta que el administrador mande el comando *salir* al bot de telegram.

Antes de iniciar la ejecución configuraremos los distintos parámetros de nuestro programa: 1) todos los GPIOs como BCM, los de los leds y el buzzer de salida y el del sensor PIR de entrada; 2) el bluetooth para chequear los beacons (a través de la librería blescan); y 3) una serie de inicializaciones como el bot de telegram a través del TOKEN o el id del chat para enviarle mensaje de 'debug' de la ejecución.

Una vez iniciada la ejecución comprobaremos el estado del sensor de movimiento. Si dicho sensor detecta un movimiento pasaremos a escanear los beacons cercanos (realizaremos varios chequeos para evitar posibles fallos). En caso de que el programa detecte un beacon cerca, se comprobará si se trata de un beacon autorizado (que se encuentre en la tabla *neighbors* de nuestra base de datos). Si no se detecta ningún beacon o se detecta un beacon no registrado en el sistema (no autorizado) se insertará la alerta en la tabla *alerts*, se enviará un mensaje a los destinatarios seleccionados en la web (pestaña *Notificaciones* / tabla *receiver*) y saltará la alarma encendiendo un led rojo y haciendo sonar un buzzer de manera intermitente. Si se detecta un beacon autorizado cerca se insertará la alerta en la tabla *alerts* y encenderemos un led blanco (en la fase final se prescindiría de este led).

4.4 Circuito eléctrico

A continuación mostraremos las conexiones entre los distintos componentes que hemos utilizado (Raspberry Pi, PIR HC-SR501, Leds y Buzzer).

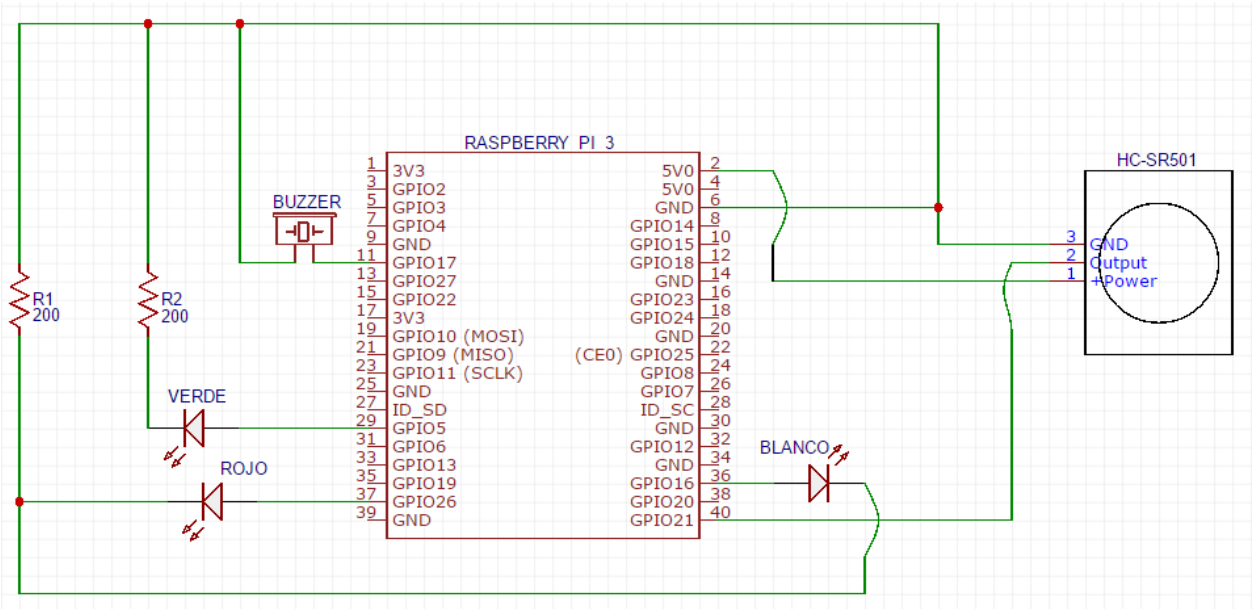

Figura 4-10: Circuito eléctrico

5 Integración, pruebas y resultados

A continuación vamos a detallar las pruebas que hemos realizado para asegurar que se cumplan los requisitos especificados en el apartado 3.

5.1 Activación de alarma al detectar intruso

Para probar esta funcionalidad básica del sistema, hemos realizado las siguientes pruebas unitarias:

- Primero hemos realizado un programa para probar el sensor PIR y su sensibilidad. Este programa imprimía una cadena de texto cada vez que detectaba un movimiento.
- Comprobación del correcto chequeo de beacons a través de la librería *blescan* sacando correctamente sus valores (UUID, RSSI…).
- Registramos en el sistema el beacon y vemos que al activarlo no salta la alarma y se enciende la luz blanca.
- Quitamos la pila del beacon y probamos que salte la alarma encendiéndose el led rojo y haciendo sonar el buzzer.
- Damos de baja al beacon, volvemos activarlo y vemos como salta la alarma al no detectar un beacon autorizado.

5.2 Alta de vecinos

Para probar esta funcionalidad realizaremos las siguientes pruebas en el formulario de la web:

- Al introducir un nombre de usuario, una dirección de correo, un número de teléfono, un chatid o un beacon que ya se encuentren en la base de datos (asociados a un vecino ya registrado) se nos deniega el alta.
- Al introducir datos incorrectos en el formulario se nos deniega la acción:
    - Número de teléfono que no tenga nueve dígitos.
    - Id del chat que no tenga nueve dígitos.
    - Dirección incorrecta (que no sean dígitos). En una implementación real habría que probar que la dirección exista dentro de la comunidad.

5.3 Baja de vecinos

Para probar esta funcionalidad simplemente comprobaremos que al dar al botón de ‘Dar de baja’ en la lista de vecinos, el vecino en cuestión desaparece de la lista y de la base de datos.

5.4 Cambio de beacon asociado

Desde la lista de vecinos de la web realizaremos las siguientes pruebas:

- Con varios vecinos dados de alta en el sistema, no nos dejará cambiar el beacon de un usuario por uno que ya esté asociado a otro vecino.
- Al introducir un beacon que esté libre, se actualizará la lista de vecinos con el nuevo beacon asociado.

5.5 Registro de actividad

Para probar que se muestre un correcto registro de actividad en la web:

- Probamos que cuando no hay ningún registro en la tabla *alerts* no aparece nada.
- Comprobamos que al detectar un movimiento la tabla se actualiza correctamente con ese nuevo registro en lo alto (la tabla estará ordenada de registro más recientes a más antiguos).
- Comprobamos que tras detectar a un vecino autorizado se actualiza la tabla con el registro en color verde y su mensaje correspondiente (con su beacon asociado).
- Comprobamos que tras detectar un intruso en el sistema se actualiza la tabla con el registro en color rojo y su mensaje correspondiente.

5.6 Notificación de intrusos a través de telegram

Para probar que la selección de receptores de las notificaciones de intrusión se realiza correctamente:

- Comprobamos que al marcar una opción en la web (radio button) se actualiza el contenido de la tabla *receiver* de nuestra base de datos.
- Probamos las distintas opciones:
    - Precondiciones: damos de alta a dos vecinos.
    - Tras marcar la opción 'Administrador' hacemos saltar la alarma y comprobamos que sólo envía el mensaje al administrador del sistema.
    - Tras marcar 'Todos' hacemos saltar la alarma y comprobamos que se envía el mensaje a los dos vecinos y al administrador.
    - Para probar la funcionalidad de la opción 'Último vecino en entrar' seguiremos los siguientes pasos:
        1) Marcaremos la opción 'Último vecino en entrar'.
        2) Haremos que el sistema detecte al vecino 1.
        3) Al hacer saltar la alarma se enviará un mensaje solamente al vecino 1.
        4) Haremos que el sistema detecte al vecino 2.
        5) Al hacer saltar la alarma se enviará un mensaje solamente al vecino 2.

5.7 Manejo de la ejecución mediante comandos de telegram

Para probar la correcta gestión de los comandos del bot de telegram:

- Comprobamos que no podemos ejecutar los comandos con otro usuario que no sea el administrador.
- Comprobamos que no podemos ejecutar el comando 'salir' si no se está ejecutando el programa.
- Comprobamos que no podemos ejecutar el programa si ya se está ejecutando.

5.8 Cambio de contraseña

Para probar el cambio de contraseña del administrador comprobaremos que el sistema no nos deje realizar el cambio en los siguientes casos:

- Introducimos la contraseña actual incorrecta.
- No coinciden los dos campos en los que hay que introducir la nueva contraseña.
- La longitud de la nueva contraseña es menor que 8.

6 Conclusiones y trabajo futuro

6.1 Conclusiones

Todos sabemos la importancia que conlleva la seguridad de cualquier propiedad privada, por lo que cualquier avance con el fin de tapar esos "agujeros de seguridad", con el cuidado de no abrir otros nuevos, siempre será bienvenido.

Además en un futuro cercano, donde la tecnología avanza a pasos agigantados (tanto para los "buenos" como para los "malos") es necesaria la investigación y creación de nuevos mecanismos de seguridad mucho más potentes que traten de acercarse a la invulnerabilidad (aunque siempre quedará algún agujero vulnerable).

En este trabajo se ha diseñado un sistema de alarma para garajes comunitarios que cubre una deficiencia general de los sistemas que hay actualmente en el mercado: el método del "piggybacking" (intrusión aprovechando la entrada de un vecino de la comunidad).

A lo largo de la realización del proyecto las principales decisiones que hemos ido tomando han sido las siguientes:

- A la hora de elegir la API de telegram, decidimos usar 'Telebot' porque es bastante sencilla de utilizar (apenas con el TOKEN del bot y los chatid de los vecinos) y permite la interacción con nuestro programa mediante comandos enviados al bot.
- Una de las principales funciones que tiene el sistema es la posibilidad de escanear beacons. Tras haber investigado, encontramos una librería bastante simple de usar y de incorporar a nuestro código: blescan. Para poder incorporar esta librería, tuvimos que modificar un poco el código porque no recogía la opción de que no encontrara ningún beacon cerca (pusimos un 'timeout').

También nos hemos ido encontrando algunos problemas que hemos ido solventando de la mejor manera posible:

- Al ir realizando las pruebas, el principal problema que nos encontramos fue que de vez en cuando el sensor PIR dejaba de responder a los movimientos. Este fallo se debía a la mala conexión que teníamos del circuito (algún cable quedaba suelto). Decidimos soldar los cables que pudimos, aunque para un futuro se tiene pensado usar un mejor cableado más consistente.
- También tuvimos algún problema al controlar la ejecución de varios programas en serie mediante los comandos de telegram. La solución a la que llegamos fue la utilización de colas en la que cada una iniciaría una ejecución y entonces no habría problemas de solapamiento.

6.2 Trabajo futuro

Lo diseñado hasta ahora sería solamente el principio. Tanto en la web como en la aplicación en sí se podría avanzar bastante.

A continuación enumeraremos una serie de mejoras en nuestro sistema de cara al futuro:

- Creación de una aplicación móvil destinada a que los vecinos estén más en contacto con el sistema y con lo que pasa en el garaje. Se podrían desplazar los mensajes de telegram a una sección de esta aplicación, aplicando algunas mejoras adicionales.
- Posibilidad de incorporar una cámara al sistema, de tal modo que cuando se detecte un intruso, el sistema hará una fotografía inmediatamente (o iniciará la grabación de un vídeo por un periodo de tiempo). Esta fotografía se adjuntaría junto al mensaje de intrusión.
- Mejora del código y de la seguridad de la aplicación y de la web.
- Mejora de los recursos hardware del sistema: sensores de movimiento inalámbricos, alarma y luces más potentes, mejor cableado, etc.

En definitiva se avanzaría en el desarrollo del sistema tratando de descubrir y atajar las nuevas vulnerabilidades que vayan surgiendo.

Referencias

[1] “Mandos Mio y SEGURphone. Sistema de seguridad para comunidades de vecinos”, Youtube https://www.youtube.com/watch?v=J-dBsF9yFZw

[2] “Cámaras de seguridad para garajes – Control de Accesos | Segurparking”, Segurparking http://segurparking.es/seguridad-garajes/control-de-accesos-para-garaje/

[3] “Alarma Verisure para garajes”, Alarmas Securitas Direct – Verisure https://www.securitasdirect.es/es/alarmas-hogar/garajes

[4] “Detector de apertura”, Alarmas Securitas Direct - Verisure, https://www.securitasdirect.es/es/verisure/detector-de-apertura

[5] “Detector de movimiento”, Alarmas Securitas Direct – Verisure https://www.securitasdirect.es/es/verisure/detector-de-movimiento

[6] “Protége-t Garaje. La innovación de la seguridad para garajes en comunidades de propietarios”, Prevent https://www.prevent.es/servicios-de-seguridad/control-de-accesos/protege-t-garaje-la-innovacion-de-la-seguridad-para-garajes-en-comunidades-de-propietarios

[7] “Videovigilancia para comunidades de vecinos”, Altatec Seguridad http://www.altatec-seguridad.com/videovigilancia-para-comunidades-de-vecinos/

[8] “CherryPy - A Minimalist Python Web Framework”, Docs.cherrypy http://docs.cherrypy.org

[9] “Bootstrap (framework)”, Wikipedia https://es.wikipedia.org/wiki/Bootstrap_(framework)

[10] “eternnoir/pyTelegramBotAPI”, Github https://github.com/eternnoir/pyTelegramBotAPI

[11] “Bots: An introduction for developers”, core.telegram.org https://core.telegram.org/bots

[12] “switchdoclabs/iBeacon-Scanner-”, Github https://github.com/switchdoclabs/iBeacon-Scanner-/blob/master/blescan.py

[13] “Raspberry Pi”, Wikipedia https://es.wikipedia.org/wiki/Raspberry_Pi

[14] “Raspberry Pi Model B”, RaspberryPi https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

[15] “nRF51822 / Ultra Low Power Wireless Solutions from NORDIC SEMICONDUCTOR”, Nordic Semiconductor https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF51822

[16] “HC-SR501 PIR Motion Sensor”, mpja.com https://www.mpja.com/download/31227sc.pdf

[17] “cfcGIT/beaconAlarm”, Github https://github.com/cfcGIT/beaconAlarm

Glosario

API	Application Programming Interface.
Piggybacking	Técnica de transmisión de datos en la que se envía el ACK dentro del próximo paquete. Nosotros usaremos este término para describir la técnica que usan los ladrones para entrar en un garaje detrás de un vecino.
Beacon	Dispositivo de emisión de señales bluetooth.
Cherrypy	Framework basado en Python orientado al desarrollo de aplicaciones web.
Bootstrap	Framework para diseño de aplicaciones web.
Telegram	Servicio de mensajería por internet (nosotros hablaremos de la aplicación para móvil).
Bot	Programa informático que imita el comportamiento de un humano.
PIR	Passive infrared sensor.
Buzzer o zumbador	Dispositivo que produce un sonido de un mismo tono.
Led	Light emitting diode.
GPIO	General Purpose Input/Output.
Debug	Depuración de programas con el fin de identificar y corregir errores.