Universidad Nacional de La Matanza

Departamento de Ingeniería e Investigaciones Tecnológicas

Sistemas Operativos Avanzados

Año 2018 – 1º Cuatrimestre

Entrega final

Días de Cursada: Martes **Turno: Noche** **Comisión: 2900**

Integrantes:

Dezerio, Sandro	DNI: 35.946.018
Ibaceta, Leandro	DNI: 37.206.999
Jalid, Fernando	DNI: 36.083.826
Nestrojil, Lucas	DNI: 36.992.179
Trotta, Mauro	DNI: 36.070.412

Contenido

1. Objetivo 3
2. Descripción del Entorno 3
   a. Hardware 3
      Sistema Embebido 3
      Servidor 4
   b. Software 4
      Sistema Embebido 4
      Aplicación Android 4
3. Funcionamiento 4
   3.1 Acceso 4
   3.2 Control 5
   3.3 Diagrama funcional 8
   3.4 Diagrama físico 8
4. Alcance 9
   a. Sistema embebido 9
   b. Aplicación Android 9
5. Diseño 10
6. Disposición de sensores y actuadores 10

b. Android 11

7. Implementación 11
   a. Sistema Embebido 11
   b. Aplicación Android 12
   c. Servidor Web 12
8. Comunicación 13
9. Producto terminado 15
   a. Sistema Embebido 15
   b. Aplicación Android 16
10. Enlace 17

1. Objetivo

SelfieHouse es una solución integral que le permitirá controlar su casa de manera remota, brindando información en tiempo real del estado de la misma y administrando el acceso de manera fácil, rápida y segura.

El proyecto consiste en integrar un sistema embebido a una casa para obtener control de la misma de forma remota e información acerca de los distintos eventos (deseados o indeseados) que pudieran ocurrir en nuestra ausencia, aprovechando todas las aristas que nos ofrece el desarrollo de IoT (Internet de las cosas).

Con SelfieHouse el usuario podrá disponer toda la información que necesita saber acerca de su hogar cuando este se encuentre fuera de ella (temperatura del ambiente, detección de movimiento, detección de luz, detección de fuego, etc.) en tiempo real.

SelfieHouse ofrece también un sistema de alarma y monitoreo con avisos en tiempo real acerca del estado de su casa. Mediante la combinación de distintos sensores, actuadores y la lógica de Arduino (potenciado por el módulo NodeMCU – ESP8266), el sistema evaluará las condiciones de la misma y realizará una acción preventiva en caso que se determine una situación indeseada en la casa. Además del aviso a su celular, una alarma sonora se activará junto con un juego de luces para que cualquier individuo que pase por allí pueda saberlo y facilitar ayuda de un tercero según corresponda.

Otra gran funcionalidad que provee SelfieHouse es el acceso sin llaves. Esto le permitirá al usuario (o cualquier persona que este autorice) acceder a la casa vía Android con un código de acceso no reutilizable.

Como si esto fuera poco, cualquier individuo que desee ingresar y no tenga código, podrá solicitar acceso mediante una foto (selfie) de modo que el usuario residente evalúe si le concederá acceso o no.

2. Descripción del Entorno

a. Hardware

Sistema Embebido

i) MicroControlador
   (1) Módulo NodeMCU ESP8266
ii) Sensores / Actuadores
   (1) Sensor digital de movimiento (PIR)
   (2) Sensor digital de temperatura (DHT22)
   (3) Sensor digital de llama (KY-026)
   (4) Sensor analógico de luz (Light Sensor v1.0)
   (5) LEDs (Colores: Rojo, Verde y Azul)
   (6) Servo SG90
   (7) Cooler Fan
   (8) Buzzer Pasivo
iii) Fuentes
   (1) PowerBank 5V (Alimentación del micro)
   (2) Transformador 12V (Para alimentar al Cooler)
   (3) Fuente con Voltímetro 5V

iv) Otros
   (1) Transistor NPN BC337
   (2) Traba con pasador de puerta
   (3) Resistencia 10KΩ / 1KΩ
   (4) Cables

Servidor

- Notebook Coradir (Intel i3 – 4 GB RAM)
- Router TP-Link TL-WR740N

b. Software

Sistema Embebido

- Arduino IDE

Servidor

- XAMPP 3.2.2
  - Apache Server 7.2.4
  - MySQL Server 5.0

Aplicación Android

- Android Studio SDK

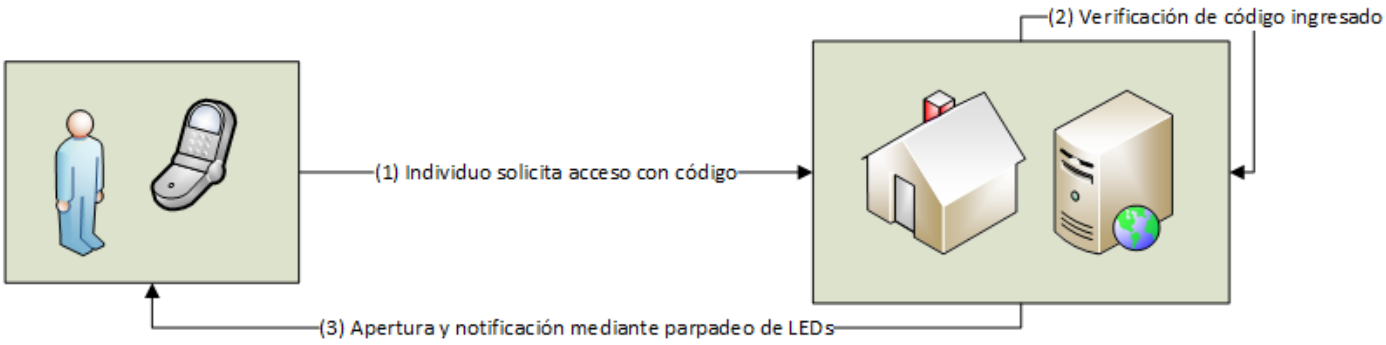
3. Funcionamiento

SelfieHouse consta de tres bloques funcionales: **un sistema embebido**, **un servidor web** y **una aplicación Android**.

La *aplicación Android* tiene dos ejes fundamentales: el **acceso** y el **control** a la casa.

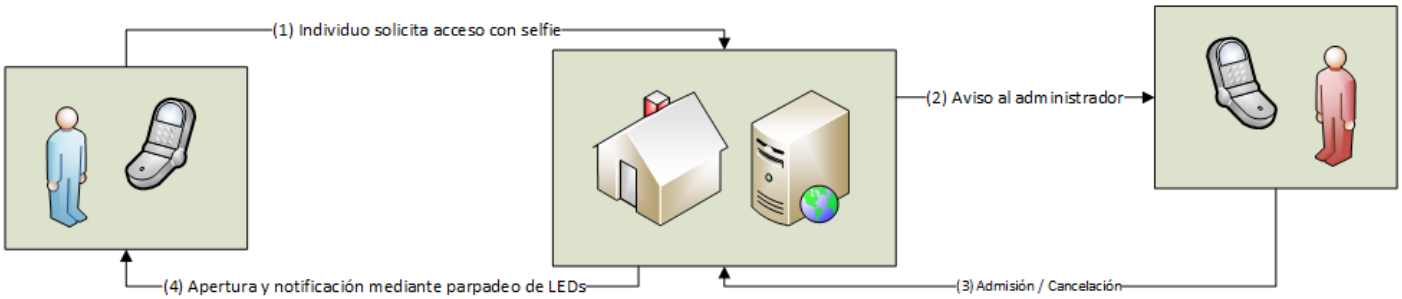
3.1 Acceso

Un individuo podrá acceder a la casa de dos maneras, de forma directa mediante la introducción de un código de acceso o bien mediante la aprobación de un tercero, previa solicitud. Esta solicitud, consta de una foto (*selfie*) que será recibida por un administrador (a través de un servidor) que será quien admita o deniegue el acceso a la casa.



(Figura 1 – Acceso sin selfie)

Esta acción es realizada cuando la persona ingresa un código de acceso valido, la cual será validada contra el servidor que se dispone.



(Figura 2 – Acceso con selfie)

En este caso la persona solicita acceso a la casa registrándose mediante una imagen de su persona, esta solicitud es almacenada en el servidor. Luego el administrador de la casa podrá atender la petición aceptando o denegando el acceso.

3.2 Control

Si se dispone de un código de acceso completo, se podrá tener control total de la casa. Desde allí se podrá ver los estados de los distintos sensores y actuadores, así como también podrá manipularse los distintos actuadores de la casa, de forma manual o automática.

El sistema de medición será activado cuando no se encuentre nadie en la casa. Este realizará el monitoreo de sus sensores y evaluará realizar una acción consecuente en caso que sea necesaria.

El *sistema embebido* se encontrará conectado a distintos sensores que medirán y evaluarán el estado el ambiente. Ante alguna situación no deseada se disparará una acción preventiva activando un actuador correspondiente. Este también reaccionará ante las solicitudes de la aplicación Android.

El *servidor web* hará de intermediario entre el sistema embebido y la aplicación Android. Aquí se almacenará una base de datos donde se persistirán las selfies de solicitud de acceso, los códigos de acceso y las notificaciones del sistema embebido.

A continuación, se presentan los distintos tipos de monitoreo que realizará SelfieHouse cuando el sistema de monitoreo se encuentre encendido.

3.1.1 Monitoreo de movimiento

Con el sistema activado, el sensor detecta movimiento dentro de la casa. Inmediatamente se enciende el buzzer y el led rojo que indican que la seguridad fue quebrantada. Luego el sistema embebido envía esa información al servidor y a la app.

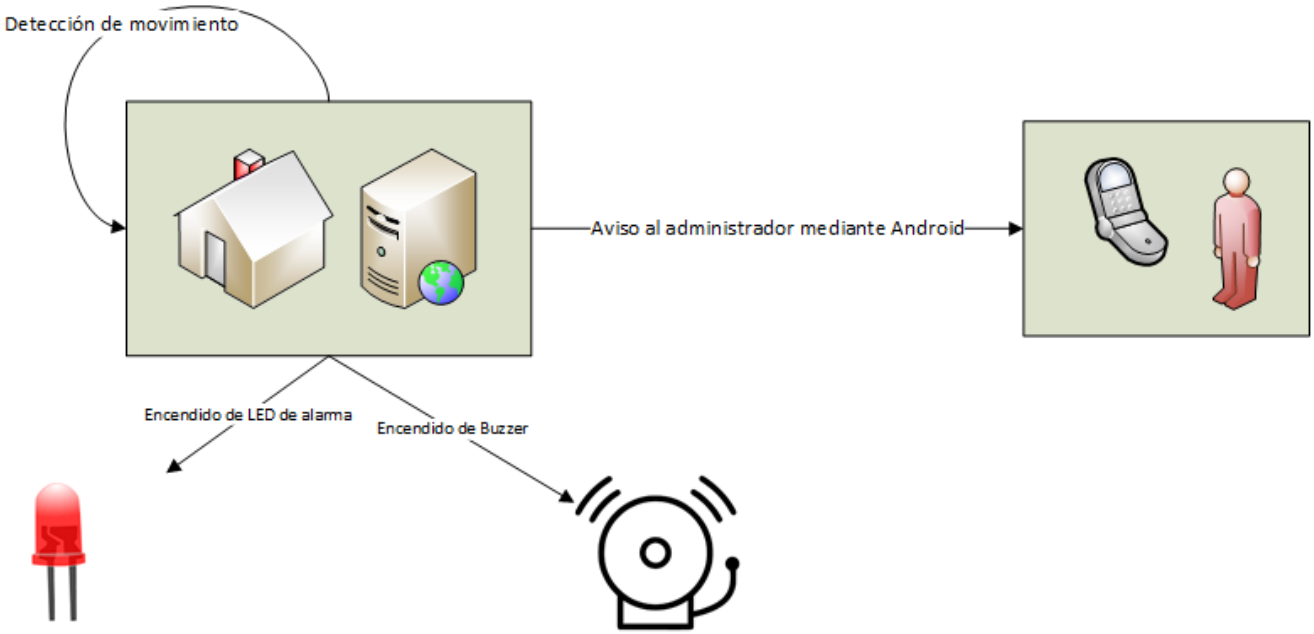


Figura 3 (Detección de movimiento)

3.1.2 Monitoreo de temperatura

Se determina un umbral de temperatura. Cuando el sensor detecta una temperatura mayor el sistema actúa enviando una notificación a la app. Además, enciende a manera de precaución el ventilador, junto con un aviso sonoro y encendido del led rojo.

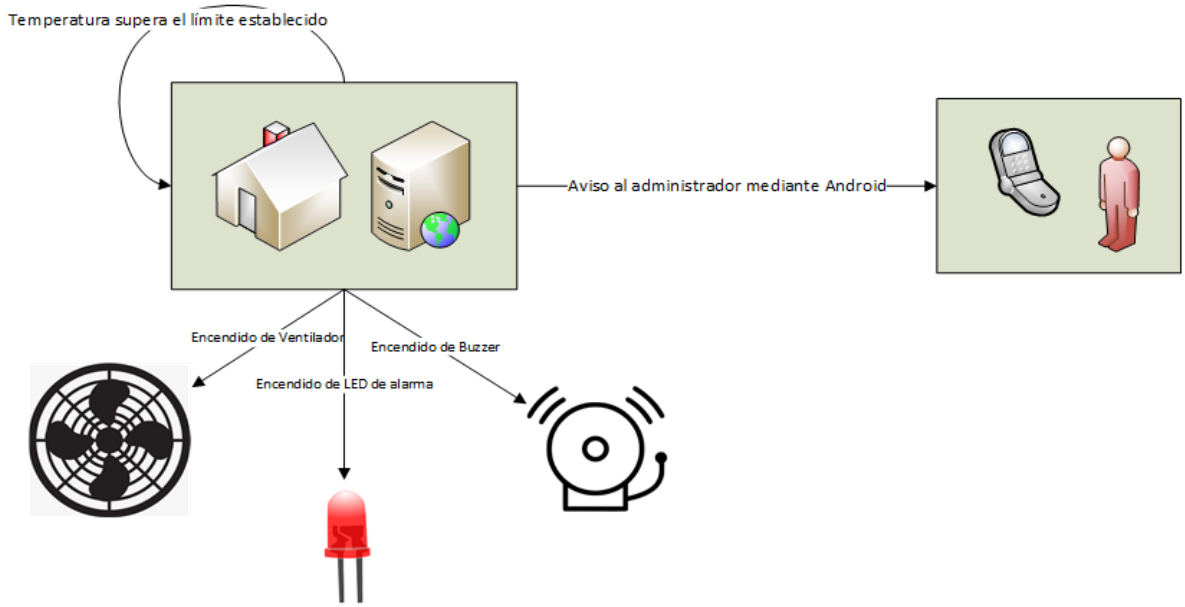


Figura 4 (Temperatura)

3.1.3 Monitoreo de luz y fuego

Para este evento se utilizan dos sensores: Sensor de luz y sensor de fuego. Cuando el sistema detecta presencia de fuego y un nivel alto de luminosidad se activa el buzzer y el led rojo. Inmediatamente se envía la información a la app dando aviso.

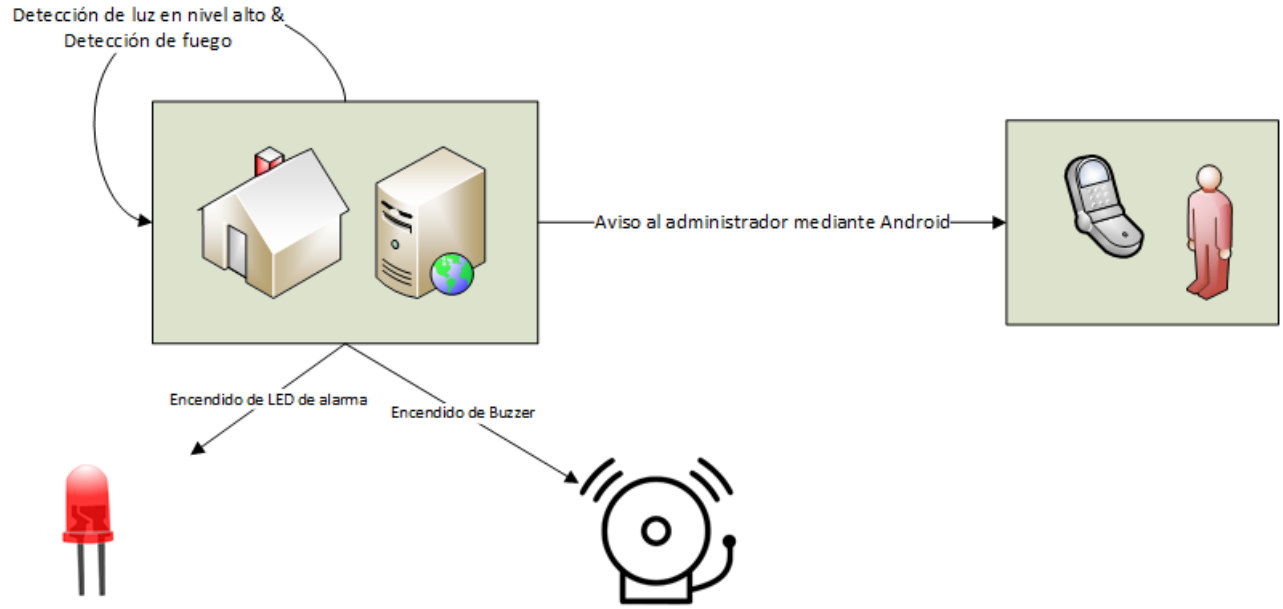
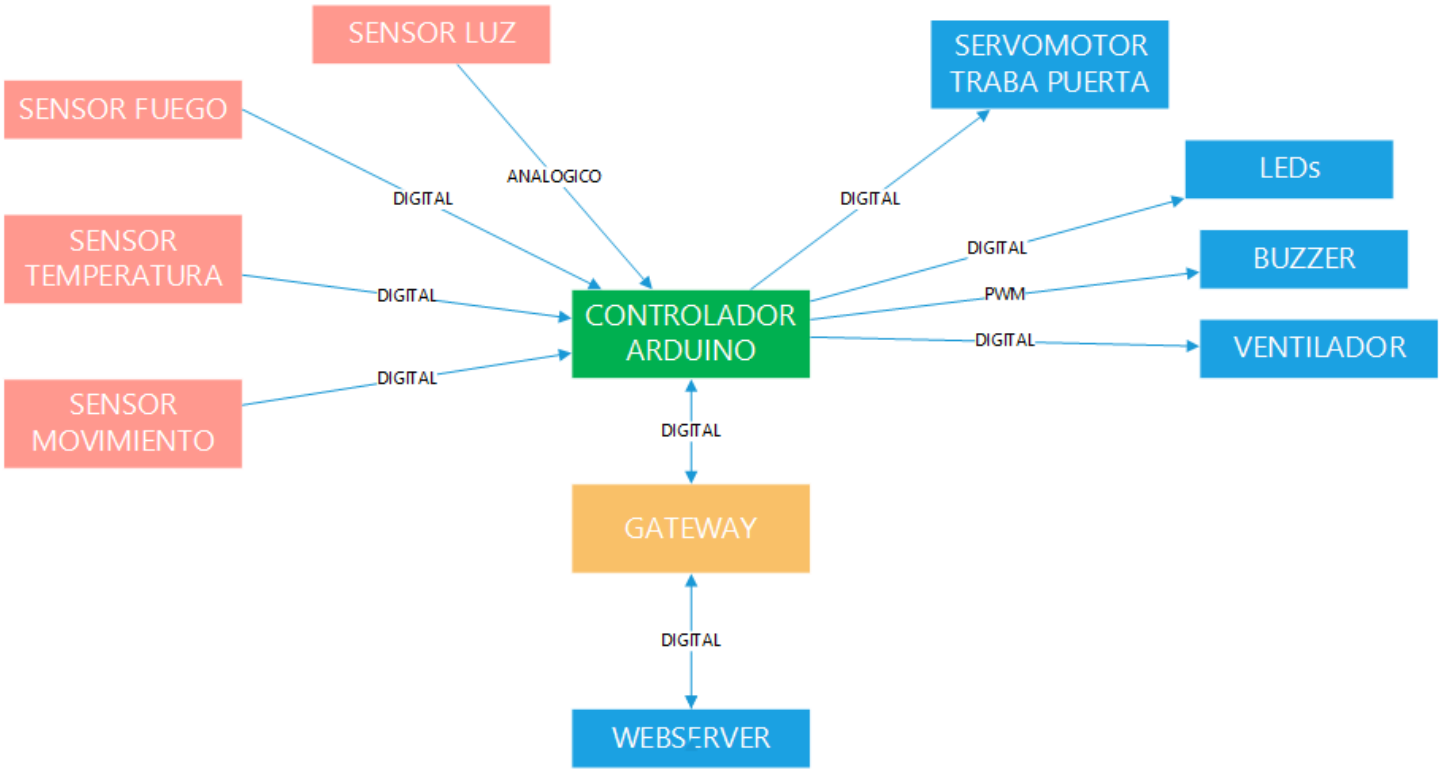


Figura 5 (Detección de fuego)

Para todos los casos, el administrador de la casa que sea notificado podrá ver mediante la app el estado de la casa, el motivo por el cual fue encendida una alarma y allí tener información certera para la futura toma de decisiones.

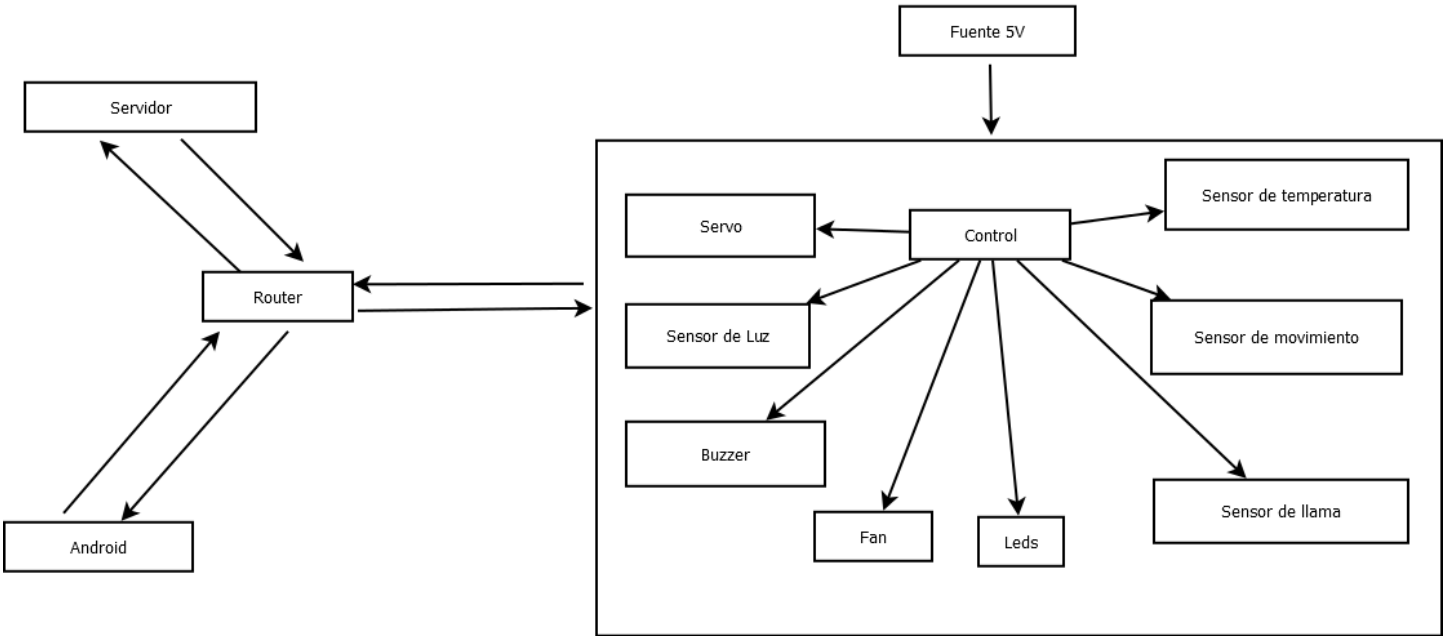
3.3 Diagrama funcional



3.4 Diagrama físico

Este diagrama representa las conexiones del hardware del sistema, como podemos observar, la placa Arduino, sensores y actuadores son alimentados por una fuente de 5V.

El Servidor, la aplicación de Android y el Arduino se conectan a la misma red para el intercambio de datos.



4. Alcance

a. Sistema embebido

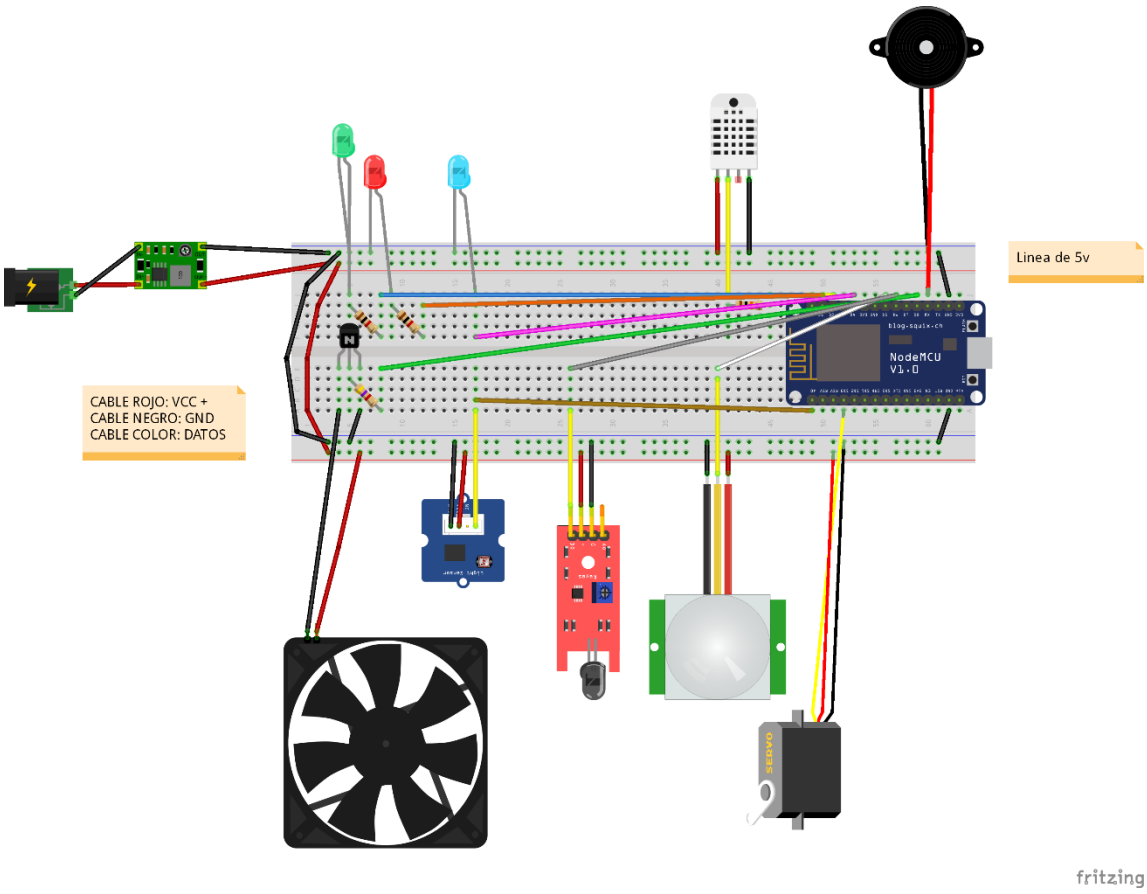
- El sistema embebido debe medir y evaluar la temperatura del ambiente.
- El sistema embebido debe medir y evaluar el nivel de luz del ambiente.
- El sistema embebido debe detectar el movimiento del ambiente.
- El sistema embebido debe detectar la presencia de fuego en el ambiente.
- El sistema embebido debe censar permanentemente el estado de sus mediciones.
- El sistema embebido debe establecer si alguna de sus mediciones implica la acción de una alerta.
- El sistema embebido debe encender el LED verde cuando inicie todos sus servicios.
- El sistema embebido debe encender el LED rojo cuando se detecte una medición fuera de rango.
- El sistema embebido debe encender el LED rojo cuando falle la inicialización de algún servicio.
- El sistema embebido debe encender el LED azul cuando se active el servicio de monitoreo.
- El sistema embebido debe sonar el buzzer y avisar al servidor Apache cuando se detecte movimiento.
- El sistema embebido debe sonar el buzzer y avisar al servidor Apache cuando se detecte la existencia de fuego y luz.
- El sistema embebido debe encender el ventilador y avisar al servidor Apache cuando se detecte una temperatura fuera de rango.
- El sistema embebido debe permitir el control de sus actuadores mediante WebService.
- El sistema embebido debe permitir obtener un informe de los estados de sus mediciones mediante WebService.

b. Aplicación Android

- La aplicación Android debe solicitar clave de acceso.
- La aplicación Android debe validar la clave de acceso contra el servidor Apache.
- La aplicación Android debe permitir tomar foto al solicitante.
- La aplicación Android debe verificar la posición geográfica de quien solicite acceso a la casa y verificar si se encuentra en un radio cercano de la casa.
- La aplicación Android debe permitir conceder el acceso a la casa cuando esta sea aprobada por un administrador de la casa.
- La aplicación Android debe mostrar las solicitudes de ingreso a la casa.
- La aplicación Android debe comunicarse con el Sistema Embebido permitiendo abrir y cerrar la traba de la casa.
- La aplicación Android debe ser notificada cuando el Sistema Embebido detecte una temperatura fuera de rango.
- La aplicación Android debe ser notificada cuando el Sistema Embebido detecte fuego en la casa.
- La aplicación Android debe ser notificada cuando el Sistema Embebido detecte movimiento en la casa.
- La aplicación Android debe comunicarse con el Sistema Embebido permitiendo cambiar el estado de las alarmas.
- La aplicación Android debe comunicarse con el Sistema Embebido permitiendo encender/apagar los actuadores de la casa.
- La aplicación Android debe permitir ingresar la ubicación de la casa.
- La aplicación Android debe permitir generar códigos de acceso a la casa.

5. Diseño

La siguiente imagen demuestra las conexiones del sistema embebido



6. Disposición de sensores y actuadores

a. Arduino

En la habitación principal disponemos de un sensor de movimiento, colocado en la parte superior de una de las esquinas de las paredes, en esa misma habitación disponemos de un buzzer que comienza a emitir sonido cada vez que el sensor de movimiento detecta movimiento.

En esta misma habitación disponemos de un servomotor conectado a una traba, que se activará cada vez que una solicitud de acceso enviada desde Android sea aceptada. Además, se encontrará un LED azul, que se usará de testigo si el sistema de alarmas está encendido o no.

Fuera de esa habitación disponemos de dos leds, uno rojo y otro verde, el rojo se enciende cada vez que se haya detectado movimiento o se haya detectado una temperatura alta dentro de la casa, el verde se enciende cada vez que se haya concedido acceso a la casa.

En la cocina disponemos de 3 sensores, luz, temperatura y llama, estos tres sensores funcionan en conjunto, en el caso de que haya fuego, estos sensores detectan alta temperatura, llama y luminosidad alta, por lo tanto, el sistema de alarma se activa, comienza a emitir sonido el buzzer, el fan y el LED rojo se encienden.

b. Android

Cada vez que una persona quiera solicitar acceso mediante una selfie se utiliza el GPS del dispositivo móvil para verificar la ubicación, se le permitirá tomarse la foto sólo en el caso de que se encuentre cerca de la casa.

Utilizamos la funcionalidad “shake”, que consiste en agitar el celular en un mismo eje, esta funcionalidad utiliza el acelerómetro del dispositivo, con esta funcionalidad apagamos el LED rojo cada vez que esté encendido.

Por último, hacemos uso del sensor de proximidad, con el cual luego de detectarse proximidad 5 veces sobre la cámara, se podrá resetear el embebido.

7. Implementación

a. Sistema Embebido

Hardware

Para la integración del hardware utilizamos un protoboard como se ve en la figura anterior. Utilizamos una línea de 5V que devuelve la placa para la conexión de los sensores de movimiento, luminosidad, fuego, temperatura y humedad, como para el servo, el cooler y el buzzer. Para la conexión del cooler utilizamos la misma fuente de 5V, a lo que agregamos un transistor para accionarlo desde un pin digital del microcontrolador.

Software Arduino

Se utilizaron las siguientes librerías: **DHT.h, ArduinoJson.h, ESP8266WebServer.h, ESP8266WiFiMulti.h, Servo.h**

Cuando el microcontrolador es encendido, realizamos la configuración de acceso a la red LAN a través de Wireless, lo que permite la comunicación tanto con el webserver, como con la aplicación de Android. También en esta instancia realizamos la configuración del servicio que nos permite realizar operaciones directas con el sistema embebido mediante protocolo GET, para esto utilizamos la librería *ESP8266WebServer.h*.

Determinamos un parámetro de tiempo de 150ms en el cuál vamos a censar todos los elementos conectados al microcontrolador y evaluar las peticiones recibidas debido a que consideramos un tiempo razonable para no sobrecargar el hardware evitando así errores en los datos devueltos. Esta acción solo se realiza cuando el estado de sistema de alarma esta activado.

Se programaron las funcionalidades de censado utilizando las librerías que nos brindan los fabricantes, tomando en cuenta las maneras de obtener información más precisa posible que brindan estos dispositivos. Por ejemplo, para el sensor de temperatura utilizamos la librería *DHT.h* que nos permite operar sobre una magnitud de grados Celsius. Luego para operar el servomotor se utilizó la librería *Servo.h* que nos permite modificar la posición del brazo de fuerza. También se debió implementar los métodos se lectura y acción que se reciben a través del servicio web.
Los datos devueltos por la placa, respondiendo a peticiones externas son devueltas con JSON, para esto utilizamos la librería *ArduinoJson.h* que nos provee las herramientas para este tipo de comunicación.

b. Aplicación Android

Librerías utilizadas: **Retrofit, Gson, OkHTTP, Glide**

Herramientas

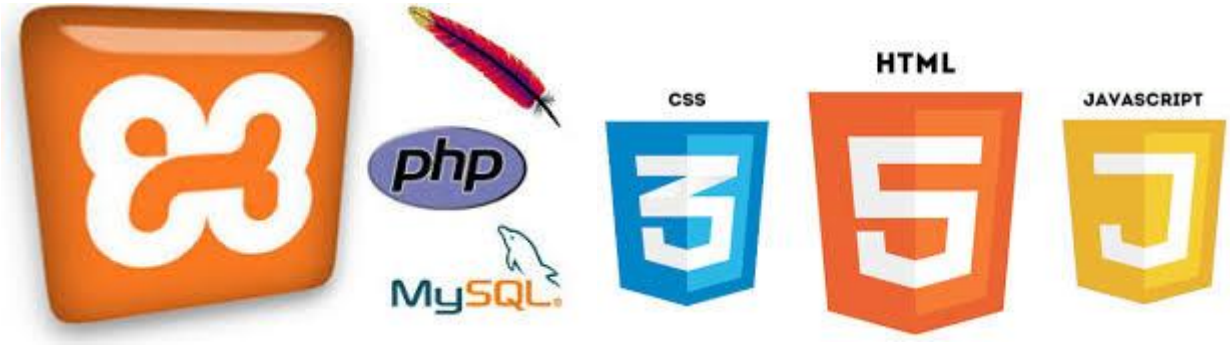
Para el desarrollo de la app utilizamos el IDE Android Studio SDK Versión XX para Windows 10. El debug fue realizado utilizando un celular MotoG en modo desarrollador conectado mediante USB a la PC.

Software

La aplicación fue desarrollada para versiones 4.4 KitKat en adelante. Utilizamos las distintas librerías proporcionadas por el IDE para el manejo de sensores, comunicación y uso de servicios que nos provee el SO. Para las funciones "*shake*" y sensor de proximidad utilizamos *SensorManager* (Herramienta proporcionada por Android), mientras que para la ubicación incorporamos al proyecto la librería *LocationManager* que nos brinda la información de nuestra ubicación utilizando el GPS del dispositivo. Para poder realizar la comunicación tanto con el servidor web como con el sistema embebido utilizamos la librería *Retrofit* que nos brinda herramientas de envío de documentos mediante el protocolo HTTP. Para la lectura la información mediante JSON que nos provee el sistema embebido usamos la librería *Gson*. También debimos incluir una librería para poder insertar GIF animados para mejorar la usabilidad del usuario (librería *Glide*).

c. Servidor Web

Utilizamos la herramienta XAMPP 3.2.2 que nos provee un servicio para servidores web (Apache) y base de datos MySQL montado sobre una Notebook Coradir.
Para almacenar las notificaciones de acceso, el estado de los sensores y el historial de eventos se creó una base de datos que se encuentra en dicho servidor. Por otro lado, implementamos un PHP, JavaScript y HTML para poder sacar la foto que utilizamos para el acceso que corre sobre un browser, este archivo se encuentra almacenado en el servidor y es accedido desde Android (**Librería HTML-5-web-php)**.
Este módulo sirve de interface para almacenar datos necesarios para obtener el estado de los sensores, claves de acceso e historial de eventos. La comunicación tanto con Android como con el sistema embebido se realiza a través de REST, enviando y recibiendo peticiones sobre GET.

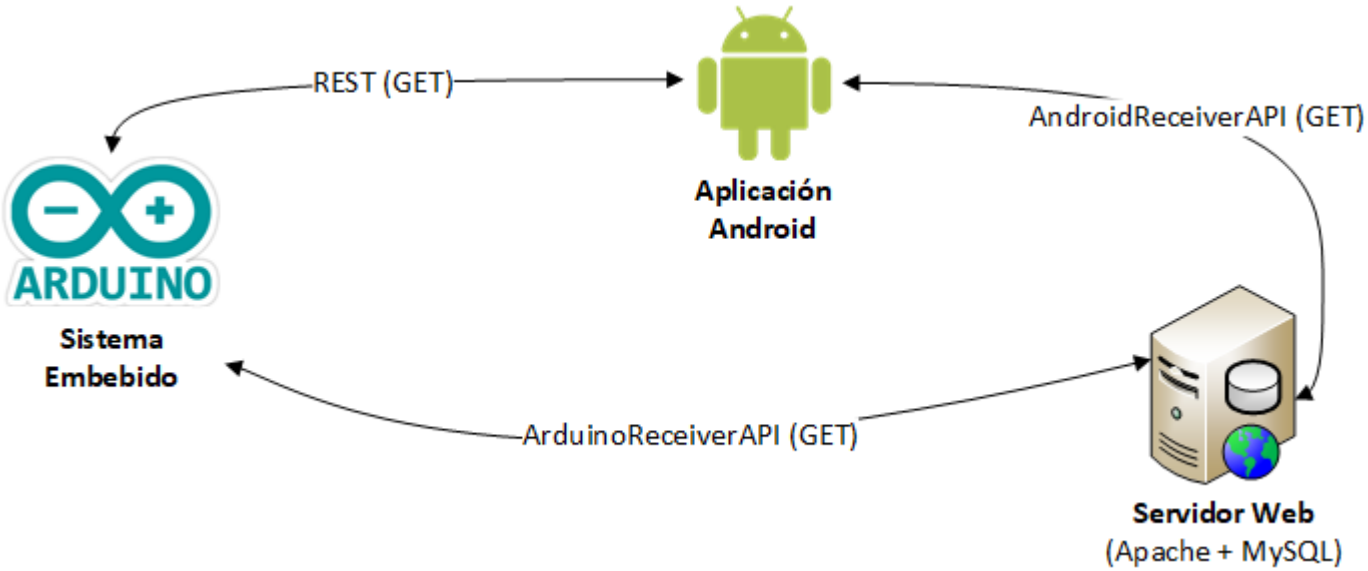


Librerías y Tecnología utilizadas:

- **HTML-5-web-php**
- **PDO:** Librería para realizar conexiones y consultas con distintos motores de base de datos.
- **jQuery:** Librería de código abierto, que simplifica la tarea de programar en JavaScript y permite agregar interactividad a un sitio web sin tener conocimientos del lenguaje.
- **AJAX:** (Asynchronous Javascript and XML) es una técnica de desarrollo web que, combina una serie de tecnologías independientes, permitiendo intercambiar información entre el servidor y el cliente (un navegador web) de forma asíncrona.
- **Bootstrap**: Framework que permite crear interfaces web con CSS y JavaScript, cuya particularidad es la de adaptar la interfaz del sitio web al tamaño del dispositivo en que se visualice.

8. Comunicación

La comunicación entre los sistemas se realiza mediante Webservices. El siguiente esquema muestra de manera clara como se presenta la comunicación:



La comunicación entre los dispositivos se realiza mediante servicios web mediante una arquitectura REST.

Se utiliza un servidor para poder almacenar datos sobre eventos y configuraciones que luego son consultados por la aplicación. En este servidor guardamos información como la ubicación de la casa (configurada inicalmente desde la app) , códigos de accesos, eventos que se lanzaron desde la casa, las peticiones de acceso junto con la imagen. También podemos obtener información sobre el ultimo estado de los sensores.

Por otro lado, la comunicación también es directa entre la app y el micotrocontrolador para poder obtener información en tiempo real de las mediciones tomadas cuando el sistema se encuentra activado, y poder activar/desactivar los actuadores y sensores que se encuentran en la casa.

El intercambio de información entre la aplicación Android y el Servidor Web se realiza mediante la API: **AndroidReceiverAPI**.

Descripción de los Webservices:

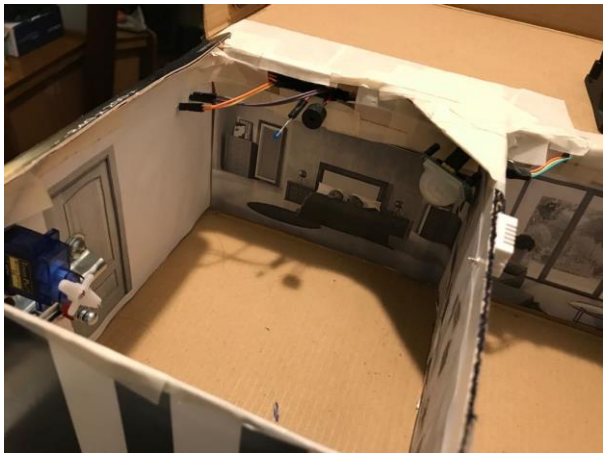
AndroidReceiverWs				
Método	**Entrada**	**Valor esperado**	**Descripción / Salida**	**Valores esperados**
GET	pull_solicitudes	True	Devuelve en JSON los datos de las solicitudes de acceso a la casa que no han sido atendidas. Los campos son: • id (int) • fecha (string) • foto (string)	No aplica
GET	pull_ubicacion	True	Devuelve en JSON las coordenadas de la ubicación del sistema embebido. Los campos son: • latitud (double) • longitud (double)	No aplica
GET	pull_notificaciones	True	Devuelve un JSON con las notificaciones de eventos en el sistema embebido. Los campos • id (int) • fecha (string) • comentario (string) • pendiente (int)	Pendiente 1 = No visto Pendiente 0 = Visto
GET	codigo_acceso tipo_acceso	Entero de 6 dígitos Entero: Acceso simple: 222 Acceso administrador: 777	Realiza una verificación para definir si el código recibido es apto para acceder al sistema con el tipo de acceso solicitado. En caso que el tipo de acceso sea simple solo se abrirá la puerta y se marcara el código como utilizado. En caso que el código sea de control no se abrirá la puerta pero contestara de forma afirmativa. Devuelve un String.	“Autorizado” “No autorizado” “Error”
POST	push_ubicacion latitud	True double	Graba en la base de datos la ubicación del sistema embebido.	“OK” “Error”

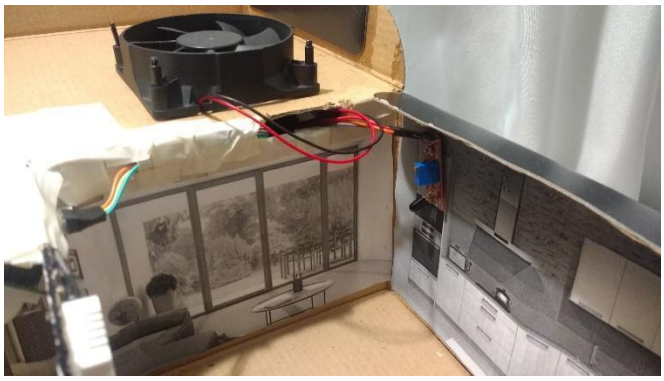
	longitud	double Entero: Acceso simple: 222 Acceso administrador: 777		

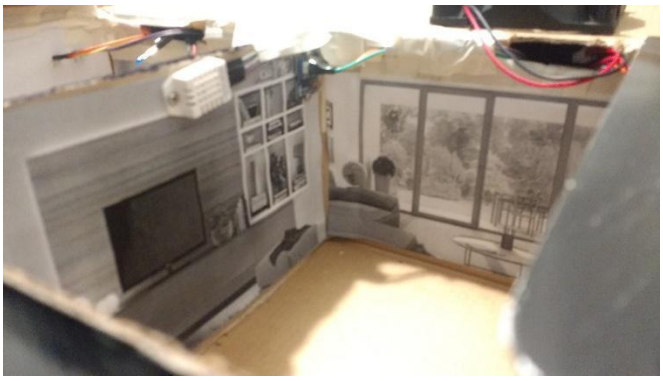
ArduinoReceiverWs				
Método	**Entrada**	**Valor esperado**	**Descripción / Salida**	**Valores esperados**
GET	acción disparador	Entero: PUERTA_TRABADA = 1000 PUERTA_DESTRABADA = 1001 BUZZER_ACTIVADO = 1002 BUZZER_DESACTIVADO = 1003 VENTILADOR_ACTIVADO = 1004 VENTILADOR_DESACTIVADO = 1005 SELFIEHOUSE_ACTIVADO = 1006 SELFIEHOUSE_DESACTIVADO = 1007 DEBUG_ACTIVADO = 1008 DEBUG_DESACTIVADO = 1009 LED_ROJO_ENCENDIDO = 1011 LED_ROJO_APAGADO = 1012 LED_VERDE_ENCENDIDO = 1013 LED_VERDE_APAGADO = 1014 REINICIO = 9999 Entero: DISPARADOR_MANUAL = 2003	Recibe acciones ocurridas en Arduino y actualiza la base de datos con estos eventos. También genera registros en la tabla Notificaciones de dichos eventos.	No aplica

9. Producto terminado

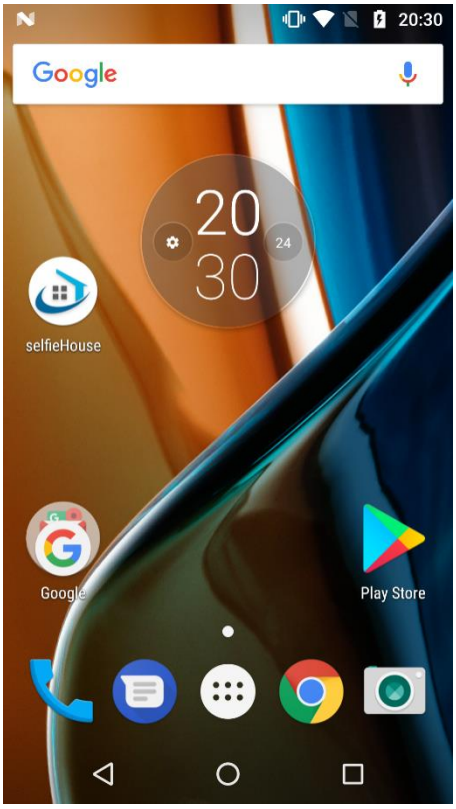
a. Sistema Embebido

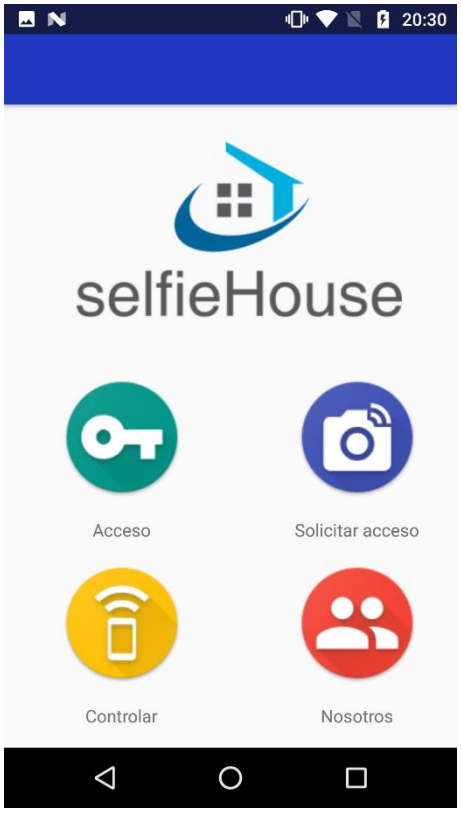


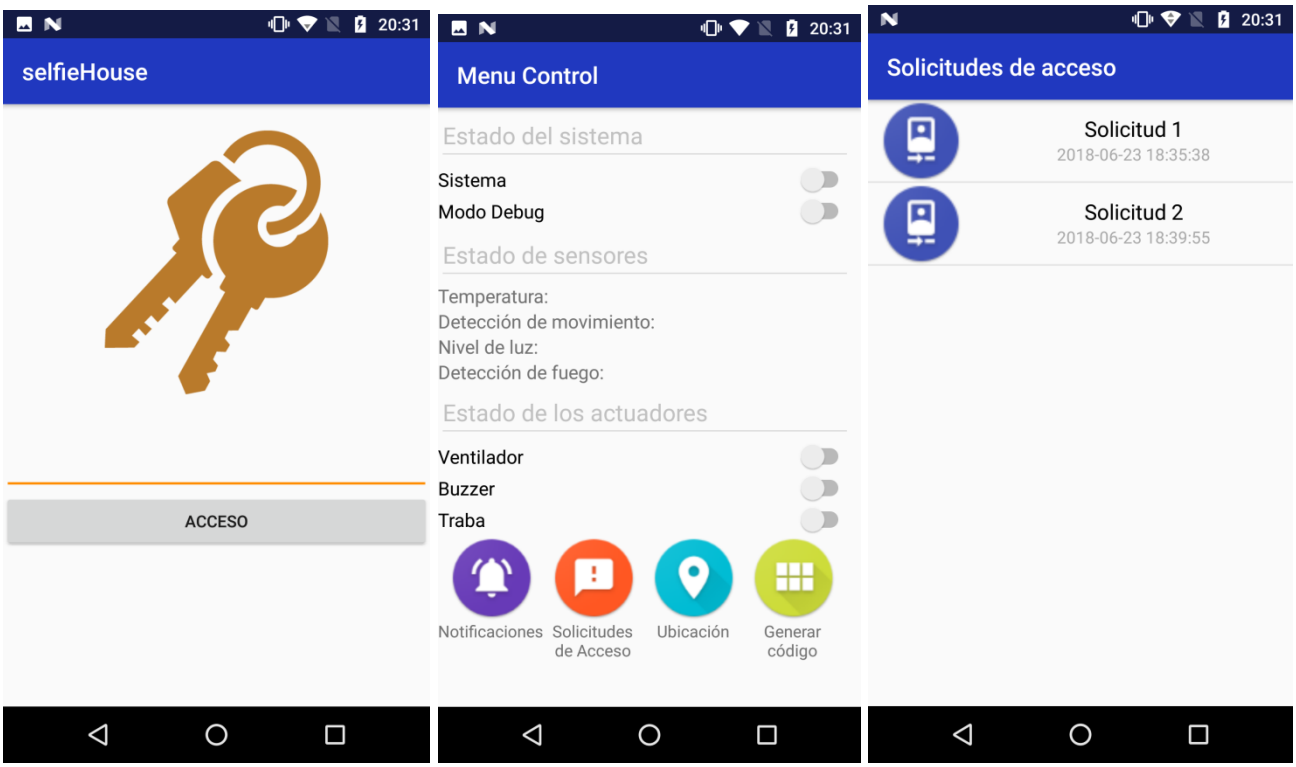




b. Aplicación Android







10. Enlace

https://github.com/SandroSD/selfieHouseSOA