

**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

# Objetivo

SelfieHouse es una solución integral que le permitirá controlar una 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.

# Descripción del Entorno

## Hardware

#### Sistema Embebido

* + 1. MicroControlador
       1. Módulo NodeMCU ESP8266
    2. 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 PC 12v
       8. Buzzer Pasivo 3V-5V
    3. Fuentes
       1. PowerBank 5V (Alimentación del micro)
       2. Transformador 12V (Para alimentar al Cooler)
       3. Fuente con Voltímetro 5V
    4. 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

## Software

#### Sistema Embebido

* Arduino IDE

***Servidor***

* XAMPP 3.2.2
  + Apache Server 7.2.4
  + MySQL Server 5.0

#### Aplicación Android

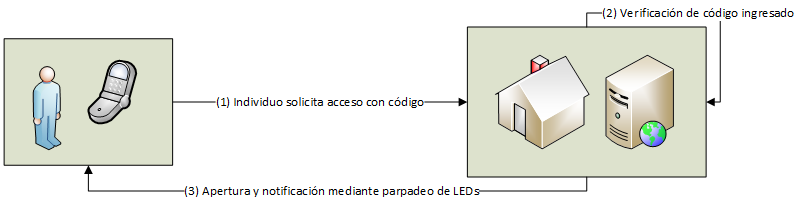
* Android Studio SDK

# Funcionamiento

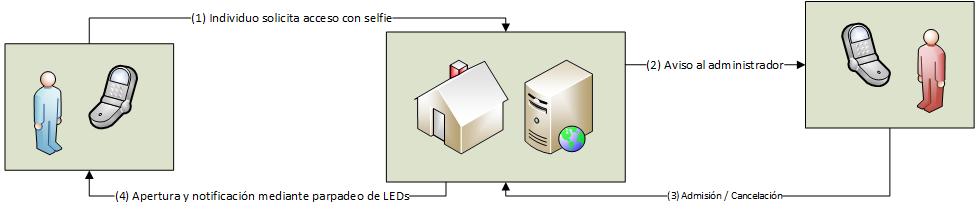
El proyecto consta de tres bloques funcionales: **un sistema embebido**, **un servidor web** y **una aplicación Android**.

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

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 que será quien admita o deniegue el acceso a la casa.



(Figura – Acceso sin selfie)



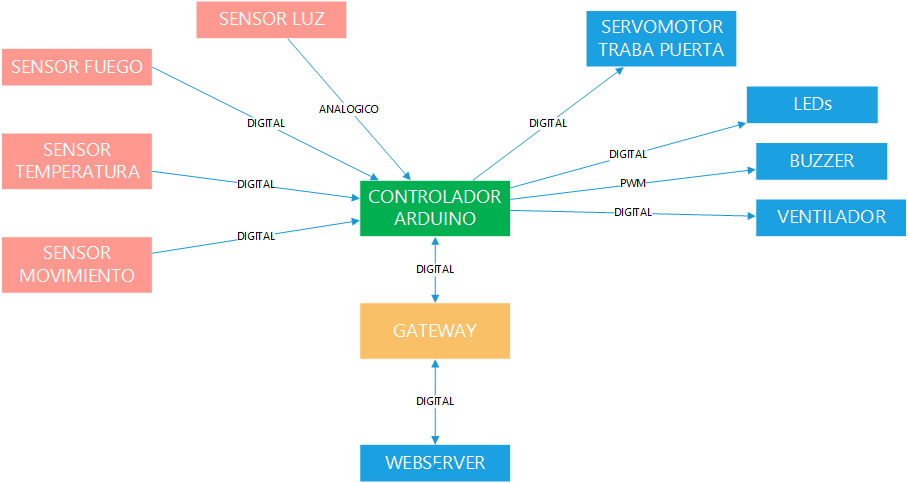
(Figura – Acceso con selfie)

Por otro lado, si se dispone de un código de acceso completo, con el cual 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.

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.

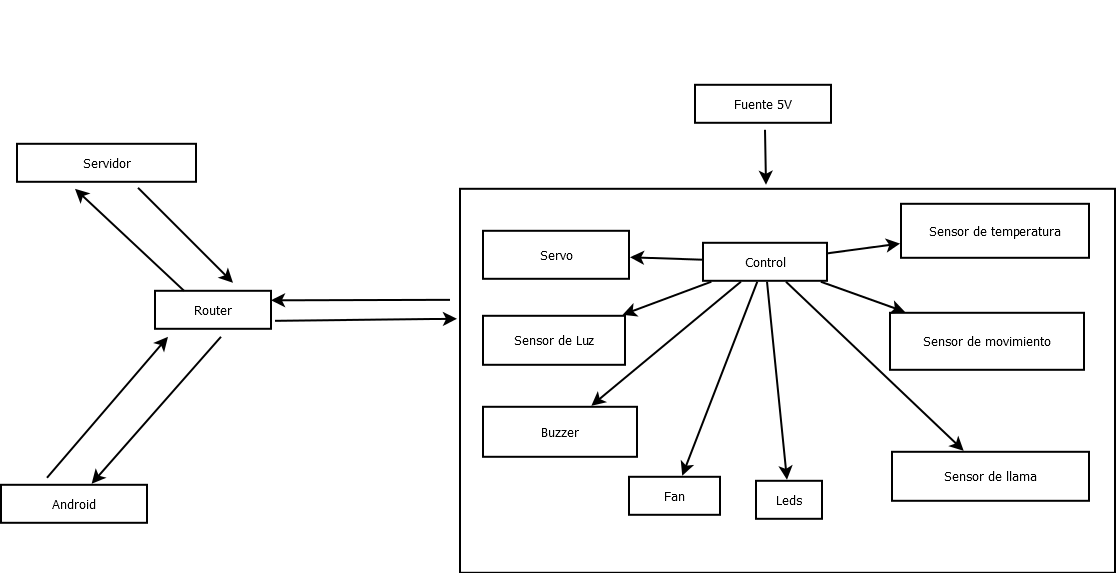
## Diagrama funcional



## 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.



# Alcance

## Sistema embebido

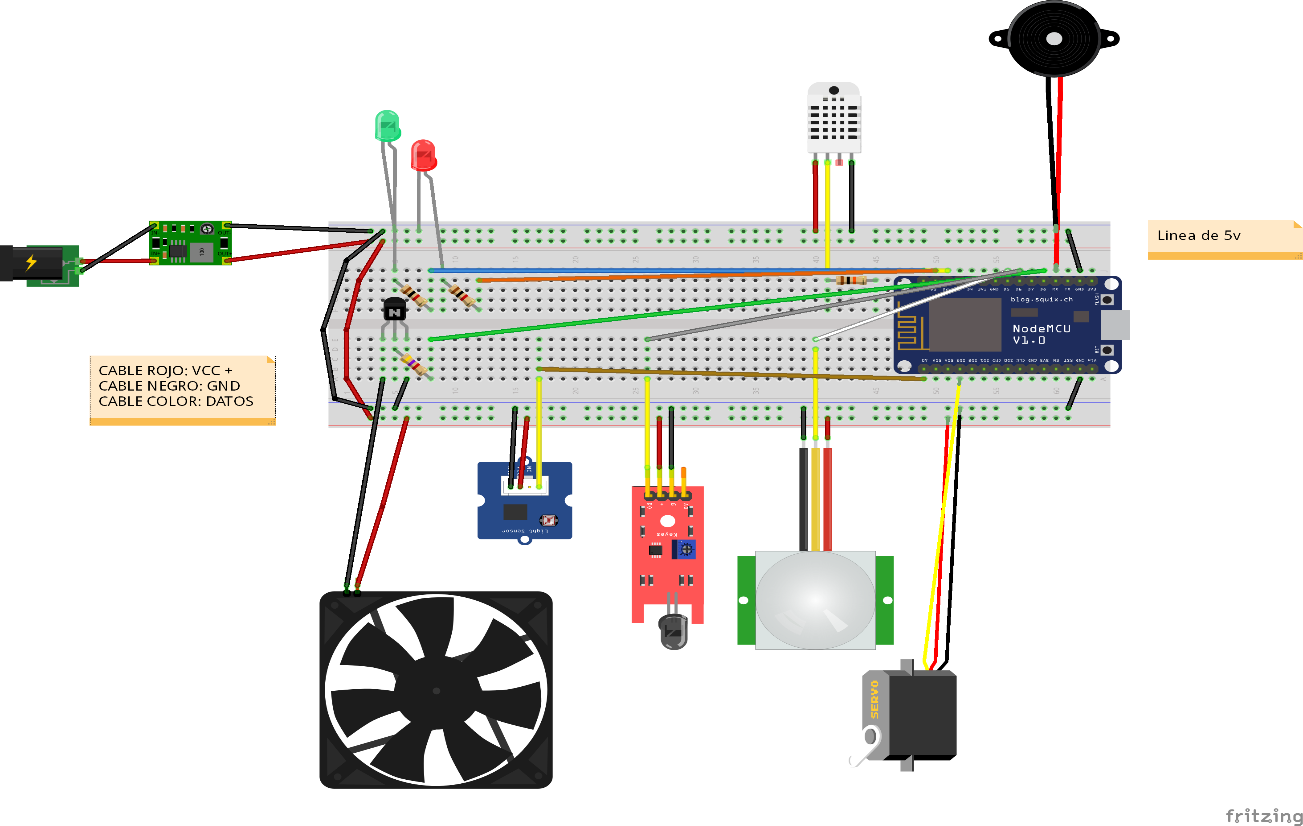
* El sistema embebido debe medir y evaluar la temperatura del ambiente.
* El sistema embebido debe medir y evaluar el nivel de luz del ambiente.
* El sistema embebido debe detectar el movimiento del ambiente.
* El sistema embebido debe detectar la presencia de fuego en el ambiente.
* El sistema embebido 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.

## 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.

# Diseño

La siguiente imagen demuestra las conexiones del sistema embebido



# Implementación

## 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 y el buzzer. Para la conexión del cooler utilizamos una fuente de 12V, utilizando 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.

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 celcius. Luego para operar el servomotor se utilizó la librería *Servo.h* que nos permite modificar la posición del brazo de fuerza. Tambien 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.

## 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*).

## 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 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)**.

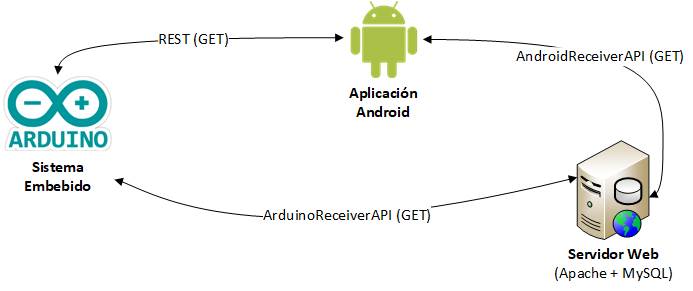
Este módulo sirve de interface para almacenar datos necesarios para obtener el estado de los sensores, claves de acceso e historial de eventos. La comunicación tanto con Android como con el sistema embebido se realiza a través de REST, enviando y recibiendo peticiones sobre GET.

***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.

# Comunicación

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



La aplicación Android se comunicará con el Servidor Web mediante la API: **AndroidReceiverAPI**.

Esta API se comunica mediante REST y brindará la siguiente información: solicitudes de acceso, ubicación del dispositivo, notificaciones de eventos ocurridos en la casa e información para realizar y validar los códigos de acceso en los intentos de acceso a la casa.

Por otro lado, la aplicación también podrá comunicarse directamente con el sistema embebido. Aquí obtendrá la información de los estados de los sensores y actuadores. También podrá enviar directamente información a este para tomar control de los dispositivos conectados este. La comunicación también es por REST y la información será otorgada en formato JSON.

De la misma forma, la aplicación Android estará consultando a el sistema embebido el estado de los sensores y actuadores conectado a este.

Descripción de los Webservices:

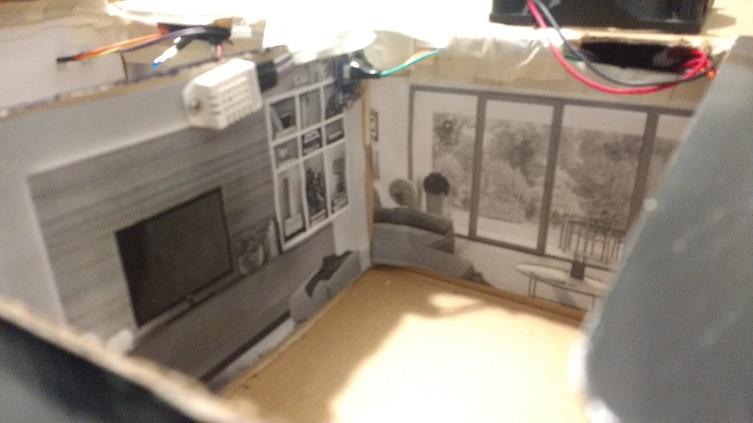
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **AndroidReceiverWs** | | | | |
| **Método** | **Entrada** | **Valor esperado** | **Descripción / Salida** | **Valores esperados** |
| GET | pull\_solicitudes | true | Devuelve en JSON los datos de las solicitudes de acceso a la casa que no han sido 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 | 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” |
| Entero:  Acceso simple: 222  Acceso administrador: 777 |
| POST | push\_ubicacion  latitud  longitud | True | Graba en la base de datos la ubicación del sistema embebido. | “OK”  “Error” |
| double |
| 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 | 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 |
| Entero:  DISPARADOR\_MANUAL = 2003 |

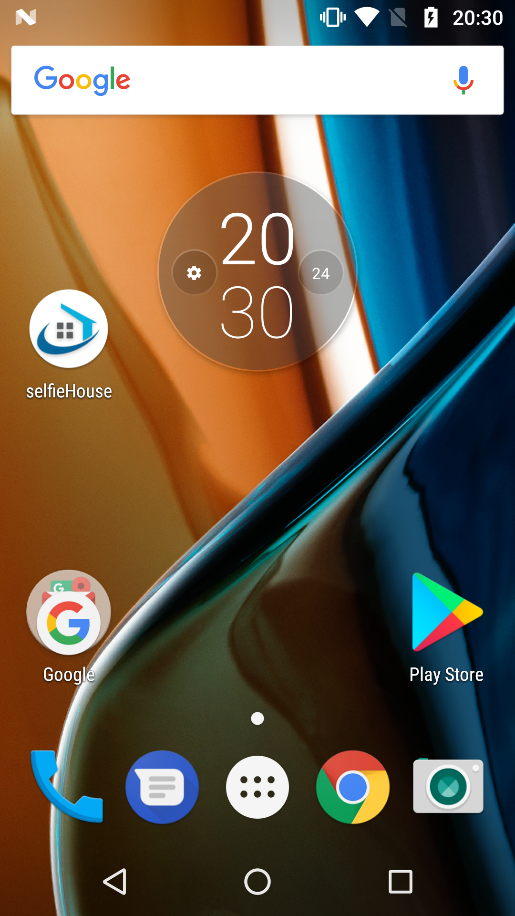
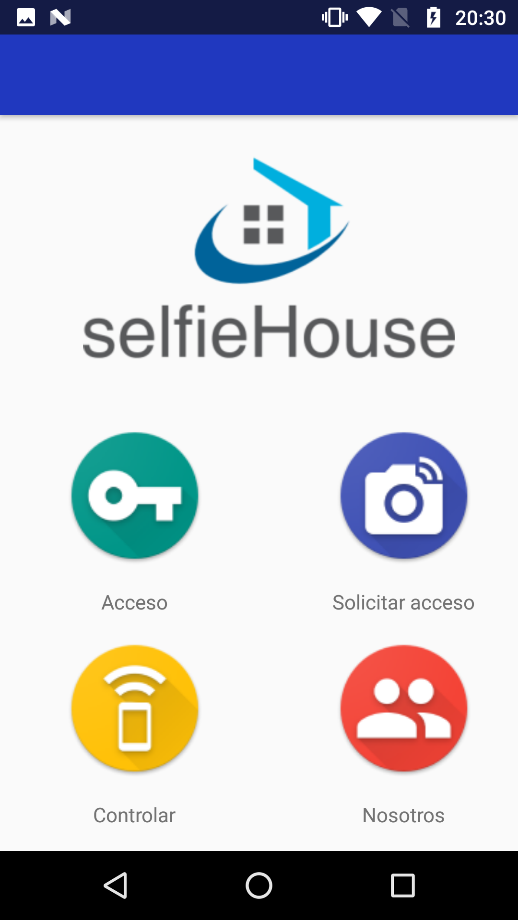
# Producto terminado

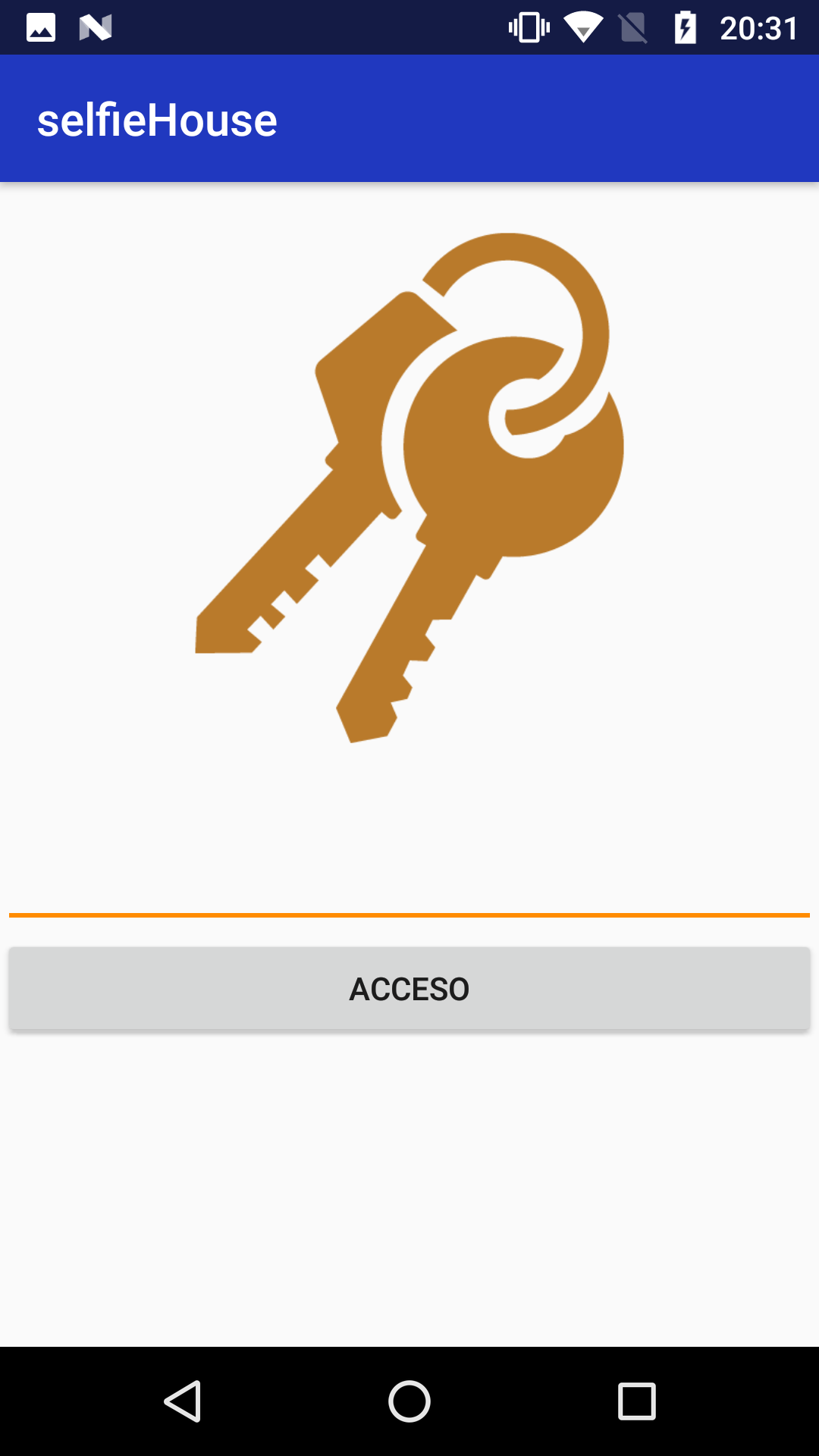
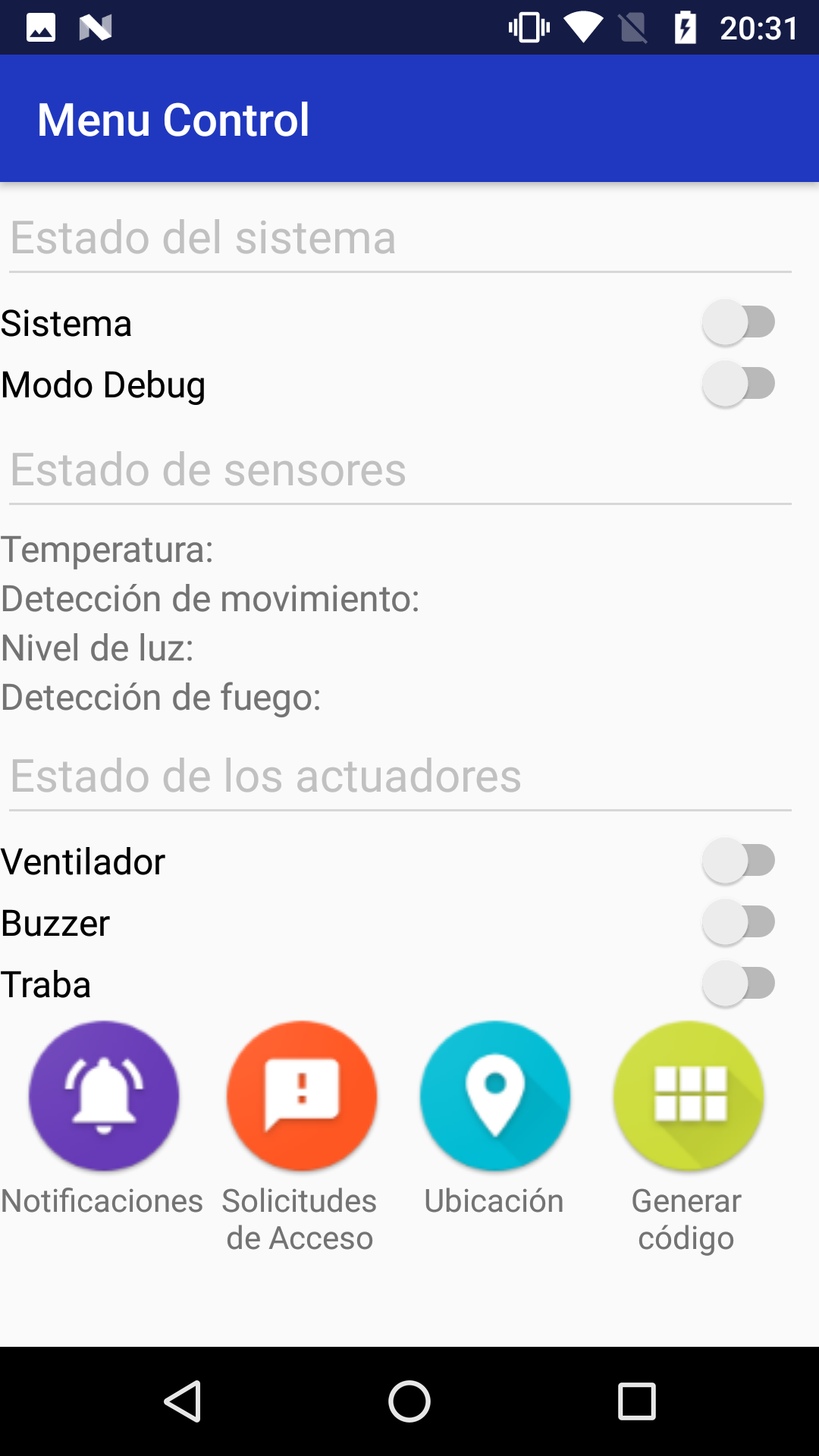
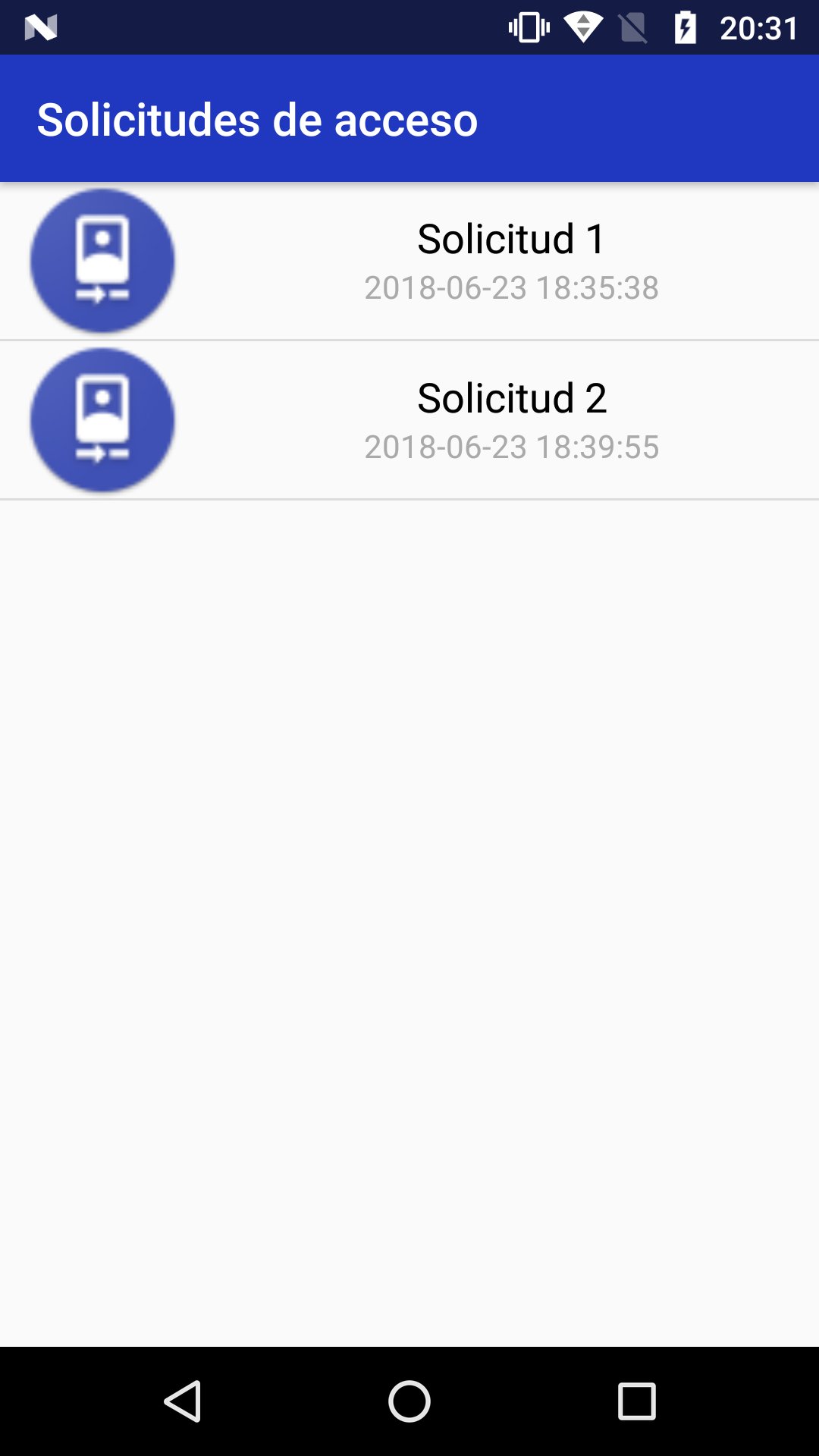
## Sistema Embebido

## Aplicación Android

# Enlace

<https://github.com/SandroSD/selfieHouseSOA>