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

1. Objetivo

Implementar los conocimientos obtenidos sobre IOT a lo largo de la cursada desarrollando un sistema de casa inteligente utilizando Sistemas Embebidos y una aplicación Android.

2. Descripción del Entorno

a. Hardware

i. Sistema Embebido

- Módulo NodeMCU ESP8266
- Notebook Coradir (I3 – 4GB RAM)
- Router TP-LINK TL-WR740N
- PowerBank 5V
- Transformador 12V
- Fuente con Voltímetro 5V
- Protoboard
- Sensor digital de movimiento
- Sensor digital de temperatura (DHT22)
- Sensor digital de llama (KY-026)
- Sensor analógico de luz (Light Sensor v1.0)
- Servo SG90
- Transistor NPN BC337
- Buzzer
- Traba con pasador de puerta
- Fan 12v
- LED Verde
- LED Rojo
- Resistencia 10KΩ
- Resistencia 1KΩ
- Cables

ii. Aplicación Android

- Celular MotoG4

b. Software

i. Sistema Embebido

- Arduino IDE
- Notepad ++
- XAMPP 3.2.2
  - Apache Server 7.2.4
  - MySQL Server 5.0

ii. Aplicación Android

- Android Studio SDK

3. 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.

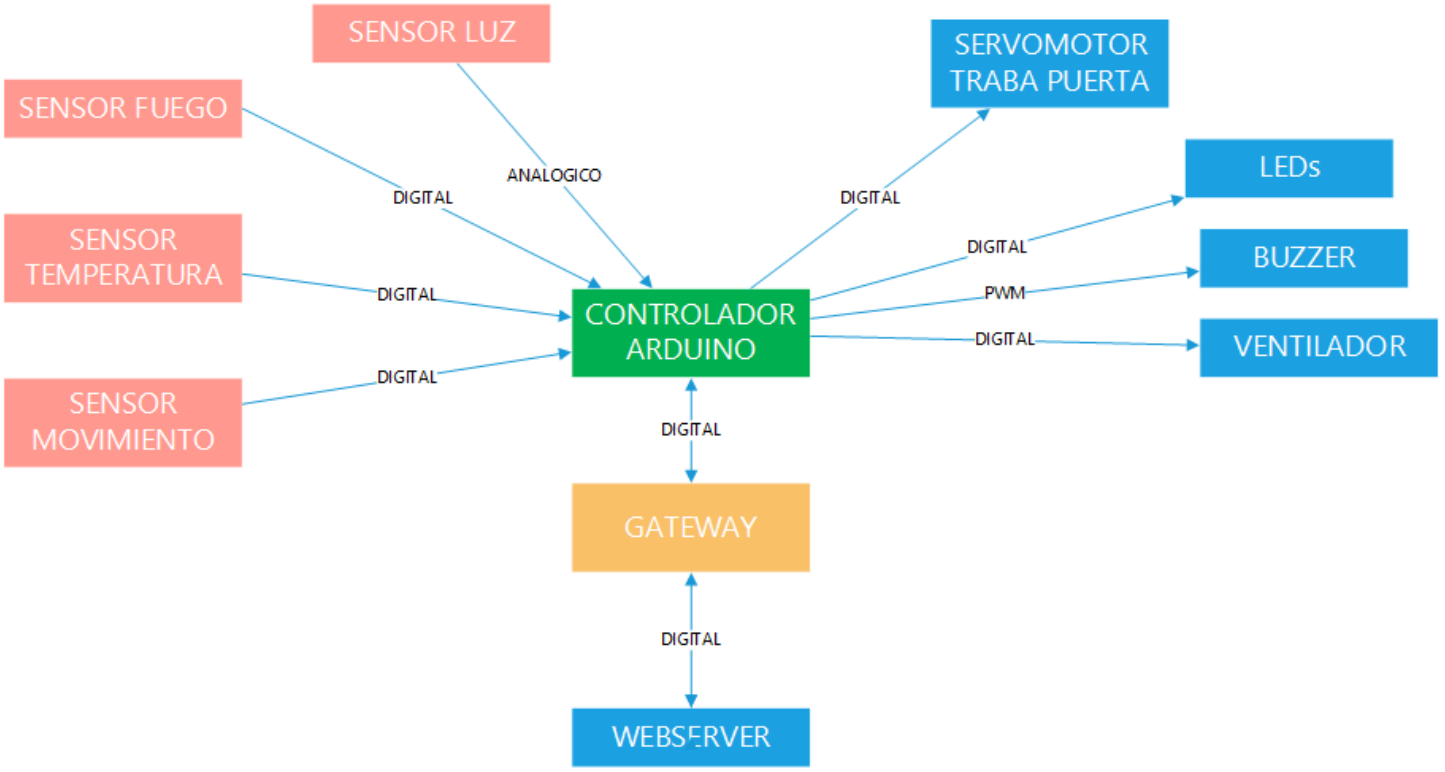
Un individuo podrá acceder a la casa de dos maneras, de forma directa mediante la introducción de un código de acceso o bien mediante la aprobación de un tercero, previa solicitud. Esta solicitud, consta de una foto (*selfie*) que será recibida por un administrador que será quien admita o deniegue el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

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.

3.1 Diagrama funcional



4. Alcance

a. Sistema embebido

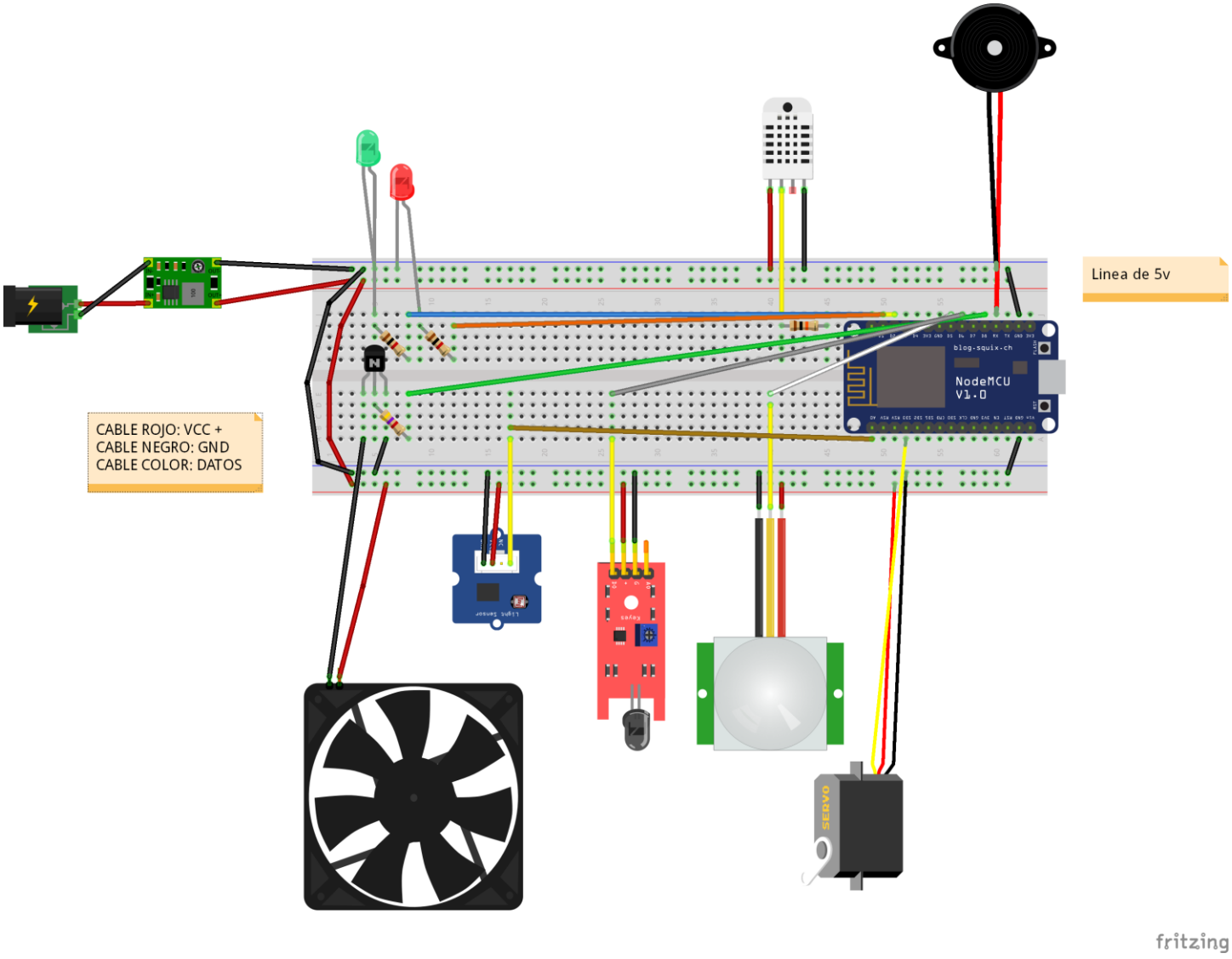
- El sistema embebido debe medir y evaluar la temperatura del ambiente.
- El sistema embebido debe medir y evaluar el nivel de luz del ambiente.
- El sistema embebido debe detectar el movimiento del ambiente.
- El sistema embebido debe detectar la presencia de fuego en el ambiente.
- El sistema embebido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

a. Sistema Embebido

Para compilar esta aplicación se utilizaron las siguientes librerías:

- **DHT.h:** Utilizada para leer e interpretar los valores del sensor de temperatura DHT22.
- **ArduinoJson.h:** Utilizado para generar objetos JSON y posteriormente enviar mensajes a través de la red.
- **ESP8266WebServer.h:** Utilizado para incorporar un webserver al sistema embebido.
- **ESP8266WiFiMulti.h:** Utilizado para enviar respuestas a través de del webserver.
- **Servo.h**: Utilizado para controlar el servo.

b. Aplicación Android

Para compilar esta aplicación se utilizaron las siguientes librerías:

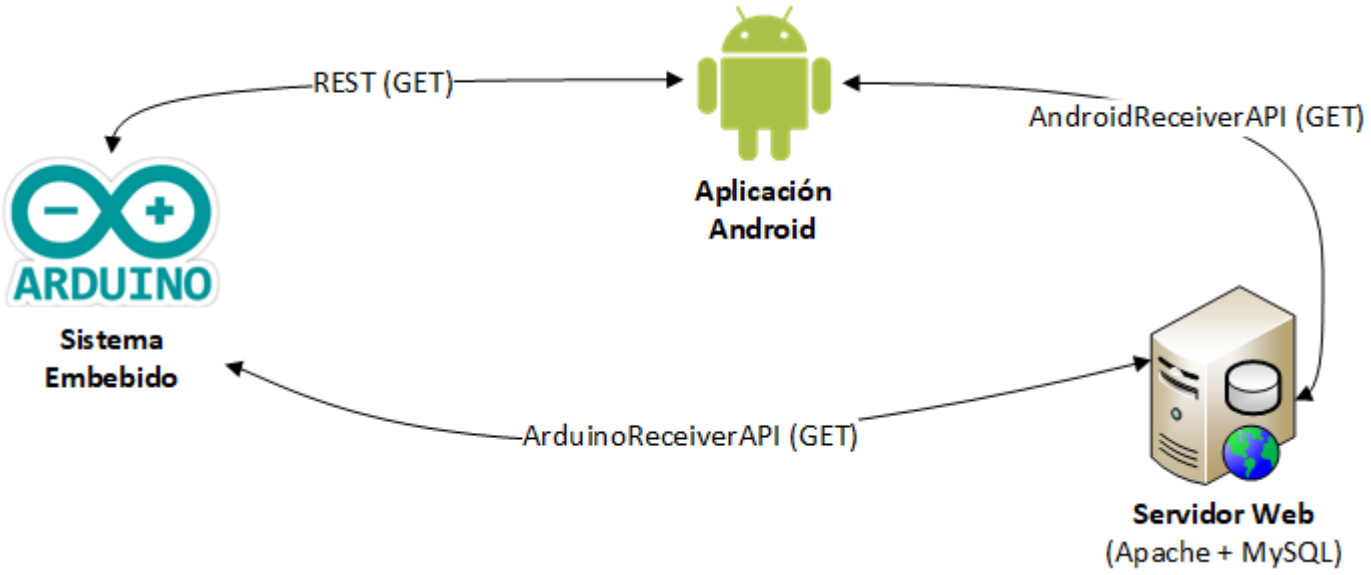
- **Retrofit:** Esta librería se utilizó para realizar peticiones REST utilizando el protocolo HTTP tanto al webserver como al sistema embebido.
- **Gson:** Esta librería se utiliza para decodificar las respuestas del webserver y del sistema embebido en formato JSON. Gracias a ella se puede interpretar de forma correcta y manipularla para su uso.
- **OkHTTP**: Se utiliza para setear el tiempo de timeout para peticiones HTTP con Retrofit.
- **Glide:** Esta librería se utiliza para visualizar correctamente GIF's animados.

c. Servidor Web

- **HTML-5-web-php:** Librería que permite utilizar la cámara del celular para tomar fotos y manipularlas desde 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.

7. 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:

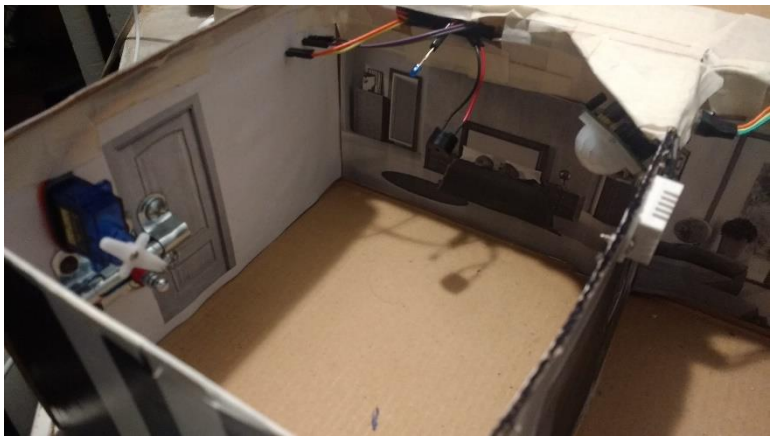
AndroidReceiverWs				
Método	**Entrada**	**Valor esperado**	**Descripción / Salida**	**Valores esperados**
GET	pull_solicitudes	true	Devuelve en JSON los datos de las solicitudes de acceso a la casa que no han sido atendidas. Los campos son: • id (int) • fecha (string) • foto (string)	No aplica
GET	pull_ubicacion	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 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 longitud	True double double Entero: Acceso simple: 222 Acceso administrador: 777	Graba en la base de datos la ubicación del sistema embebido.	“OK” “Error”

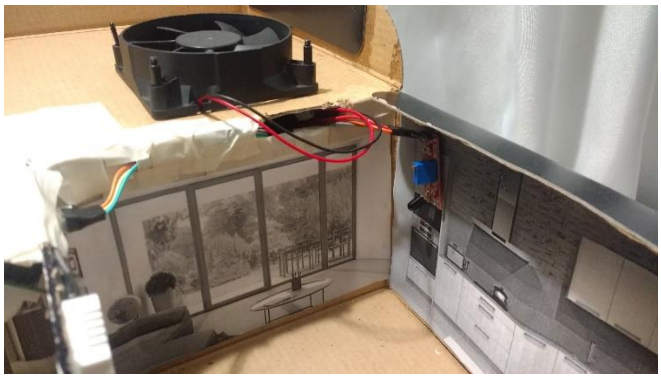
ArduinoReceiverWs				
Método	**Entrada**	**Valor esperado**	**Descripción / Salida**	**Valores esperados**
GET	acción disparador	Entero: PUERTA_TRABADA = 1000 PUERTA_DESTRABADA = 1001 BUZZER_ACTIVADO = 1002 BUZZER_DESACTIVADO = 1003 VENTILADOR_ACTIVADO = 1004 VENTILADOR_DESACTIVADO = 1005 SELFIEHOUSE_ACTIVADO = 1006 SELFIEHOUSE_DESACTIVADO = 1007	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

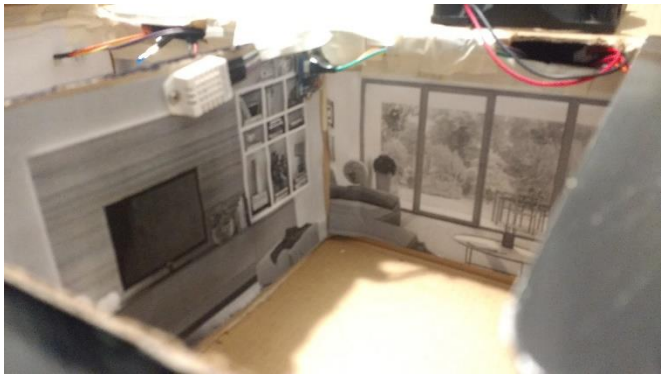
		DEBUG_ACTIVADO = 1008 DEBUG_DESACTIVADO = 1009 LED_ROJO_ENCENDIDO = 1011 LED_ROJO_APAGADO = 1012 LED_VERDE_ENCENDIDO = 1013 LED_VERDE_APAGADO = 1014 REINICIO = 9999		
		Entero: DISPARADOR_MANUAL = 2003		

8. Producto terminado

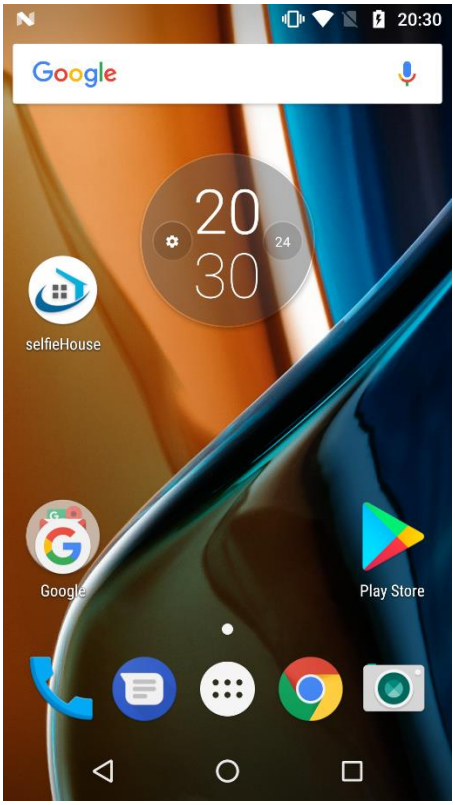
a. Sistema Embebido

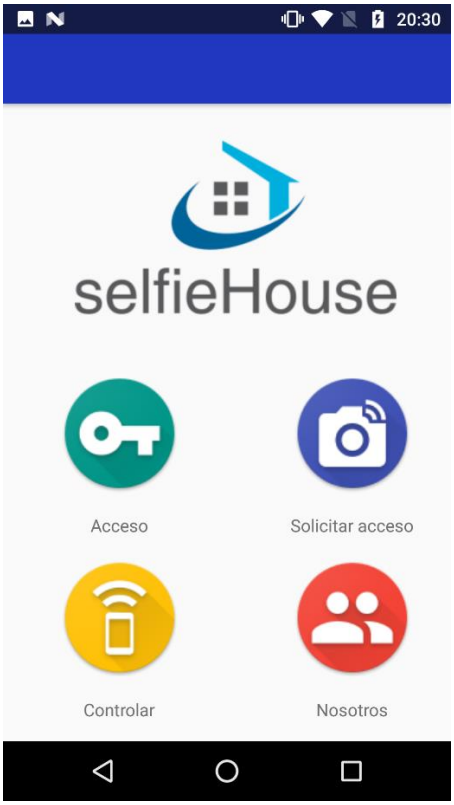


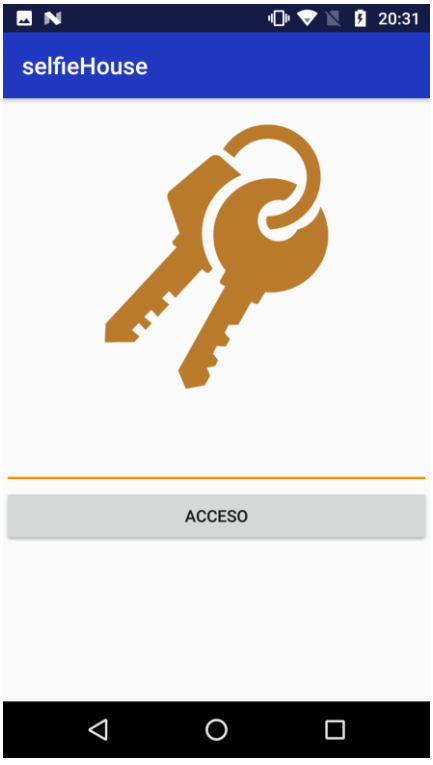


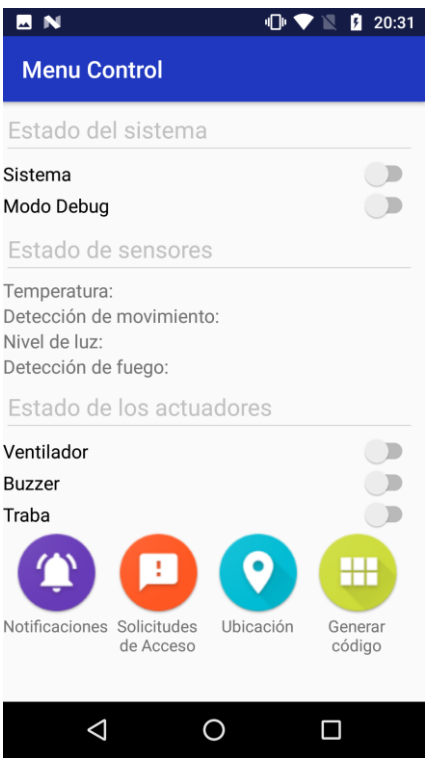


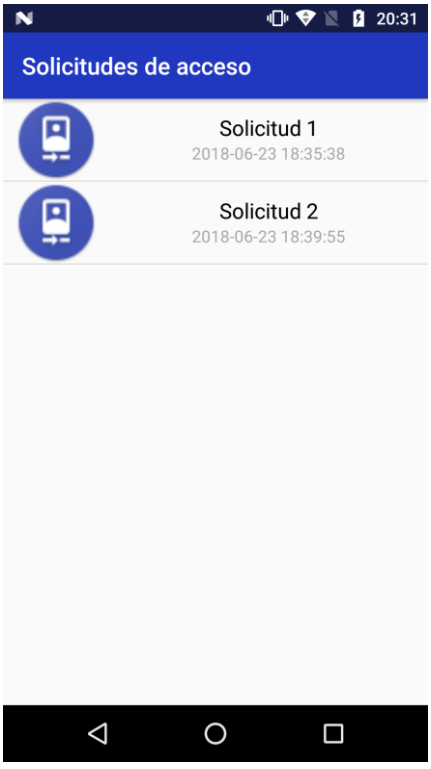
b. Aplicación Android











9. Enlaces

https://github.com/SandroSD/selfieHouseSOA