# 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.

# Descripción del Entorno

## Hardware

#### Sistema Embebido

* Placa NodeMCU ESP8266
* Notebook Coradir (I3 – 4GB RAM)
* Router TP-LINK
* PowerBank 5V
* Transformador 12V
* Fuente con Voltímetro 5V
* Protoboard
* Sensor digital de movimiento
* Sensor digital de temperatura y humedad (DHT22)
* Sensor digital de llama (KY-026)
* Sensor analógico de luz (Light Sensor v1.0)
* Servo
* Transistor NPN BC237
* Buzzer
* Traba con pasador de puerta
* Fan 12v
* LED Verde
* LED Rojo
* Resistencia 10KΩ
* Resistencia 1KΩ
* Cables

#### Aplicación Android

* Celular MotoG4
* Sensor GPS
* Sensor Acelerómetro
* Sensor de proximidad

## Software

#### Sistema Embebido

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

#### Aplicación Android

* Android Studio IDE

# Alcance

## Sistema embebido

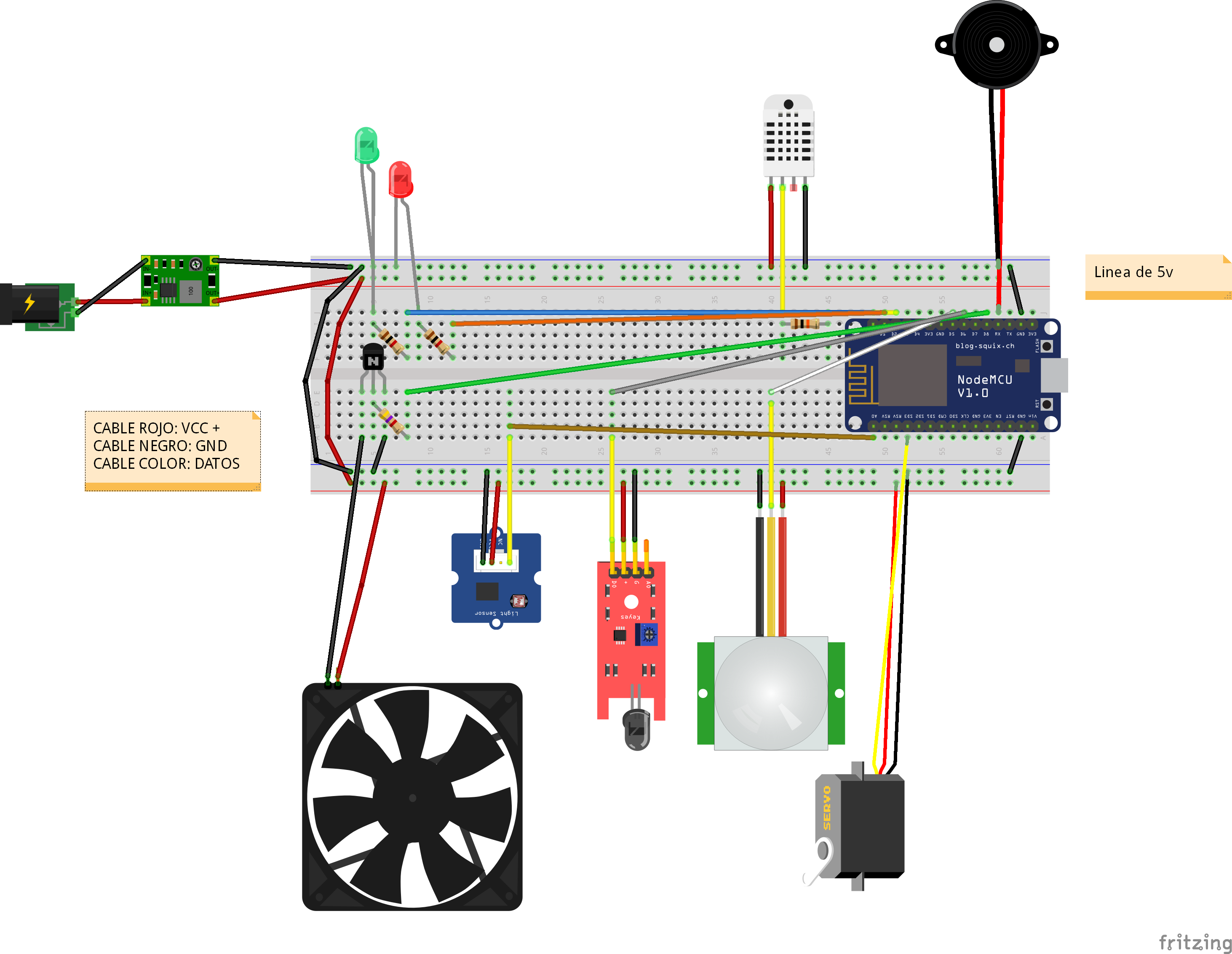
* El sistema embebido debe medir y evaluar la temperatura del ambiente.
* El sistema embebido debe medir y evaluar el nivel de luz del ambiente.
* El sistema embebido debe detectar el movimiento del ambiente.
* El sistema embebido debe detectar la presencia de fuego en el ambiente.
* El sistema embebido 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 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 resumen del estado de sus mediciones mediante WebService.

## Aplicación Android

* La aplicación Android debe solicitar contraseña de acceso.
* La aplicación Android debe capturar foto y enviarla al servidor Apache.
* La aplicación Android debe detectar la luminosidad del ambiente y encender el flash en caso que sea necesario para capturar una foto.
* La aplicación Android debe verificar la posición geográfica de quien solicite acceso a la casa y verificar si se encuentra en el radio de la casa.
* La aplicación Android debe permitir conceder el acceso a la casa cuando esta sea requerida mediante una sacudida de dispositivo.
* La aplicación Android debe ser notificada cuando exista una solicitud 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.

# Diseño

La siguiente imagen demuestra las conexiones del sistema embebido



# Implementación

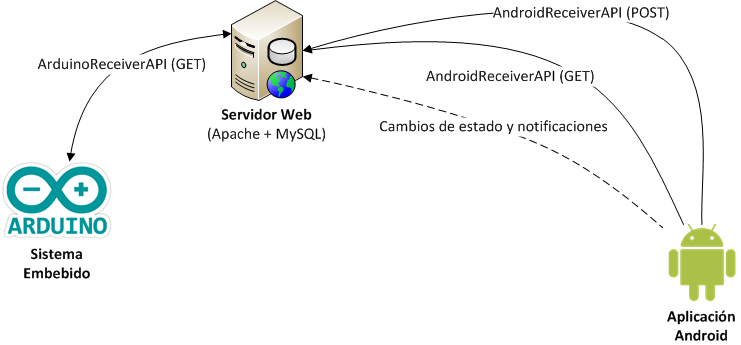
## Sistema Embebido

## Aplicación Android

## Servidor Web

# Comunicación

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



El servidor web dispone de dos webservices: uno para recibir las peticiones desde Android y otro para recibir las actualizaciones de Arduino. Estos funcionan como API REST (transferencia de estado representacional) y devuelven la información en formato JSON o texto plano según corresponda. La comunicación a estas APIs se realiza por

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\_estados | true | Devuelve en JSON los estados actuales de los actuadores del sistema embebido. Los campos son:   * id (int) * nombre (String) * estado (int) * fecha (String – último cambio de estado) | Estado 1 = Activado  Estado 0 = Desactivado |
| 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 |
| POST | 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 una acción desde Android y la envía al sistema embebido para que este la realice.  Devuelve un string con el resultado de la operación realizada. | “OK”  “Error” |
| Entero:  DISPARADOR\_MANUAL = 2003 |
| POST | 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 |
| POST | nuevo\_codigo  tipo\_codigo | true | Genera un código de 6 digitos con el permiso de acceso que sea informado.  Devuelve el código de acceso. | [código creado]  “Error” |
| 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.  Tambien genera registros en la tabla Notificaciones de dichos eventos. | No aplica |
| Entero:  DISPARADOR\_MANUAL = 2003 |

# Producto terminado

## Sistema Embebido

## Aplicación Android

# Anexos

## Funcionamiento de los sensores

# Enlaces

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