



Sistemas Operativos Avanzados

Sistemas Embebidos - IoT

Segundo cuatrimestre - 2017

Días de Cursada: Lunes

Turno: Noche

Aula: 266

Docentes:

Graciela de Luca
Gerardo Garcia
Esteban Carnuccio
Mariano Volker
Waldo Valiente

Integrantes:

Broqua, Fernando	DNI: 34476433
Calapiña, Javier	DNI: 33329447
Cortez, Martín	DNI: 30440023
Gomez, Jorge	DNI: 31698426
Gonzalez, Gustavo	DNI: 33457235

Table of Contents

Introducción:	1
Componentes:	1
Modulos de procesamiento:	1
Sensores:	1
Actuadores:	1
Componentes IoT:	1
Descripción de Componentes	2
Arduino UNO	2
Raspberry Pi 3.....	2
Servomotor SG90	2
Sensor Infrarrojo FC-51 “Flying Fish”	2
Sensor de Sonido.....	3
LED’s	3
Parlante	3
Celular	3
Diagrama en bloques de la configuración:.....	4
Circuito	5
Instalación:	7
Vista Frontal:	7
Vista Vertical:	7
Funcionamiento	8
Sección Embebidos.....	8
Arduino:.....	8
Raspberry:	9
Sección IoT	10
Android:.....	10
Anexo I: Hojas de datos de componentes.....	13
Arduino.....	13
Raspberry Pi 3 model b	13
BCM2837	13
SERVO MOTOR SG90	14
IR Infrared Obstacle Detaction Sensor Module 2 - 30cm FC-51 “Flying Fish”	16
SENSOR DE SONIDO	17
AJUSTANDO EL LÍMITE DE DISPARO	17

Anexo II: Código principal de Apk Android	18
Anexo III: Base de datos Firebase.....	26

Proyecto de Puerta de Seguridad.

Introducción:

En este proyecto implementamos un diseño de Puerta de Seguridad basado en sensores y servomotores, controlados por dos módulos embebidos interconectados entre sí. Así como también una implementación IoT que permite controlar y monitorear la puerta a través de una aplicación móvil en Android.

Los objetivos principales del proyecto constan de:

- Permitir abrir la puerta con una secuencia de golpes.
- Detectar si hay alguien en frente de la puerta.
- Detectar si alguien está forzando la puerta.
- Obtener una respuesta visual y sonora cuando se ha abierto la puerta.
- Permitir abrir la puerta desde una aplicación móvil en Android.
- Obtener datos actualizados del estado de la puerta en la aplicación Android.

Componentes:

Modulos de procesamiento:

- Arduino UNO
- Raspberry Pi 3

Sensores:

- Servo SG90
- Sensor infrarrojo
- Sensor de Sonido con salida digital

Actuadores:

- Servo SG90
- Parlante
- LED's rojo, amarillo y transparente

Componentes IoT:

- Celular Android, con los siguientes sensores:
 - Luz
 - Proximidad
 - Acelerómetro
- Base de datos Firebase

Descripción de Componentes

Arduino UNO

Modulo embebido cuya principal función es la operación del Servomotor a través de señales de control codificadas.

También es utilizado como conversor analógico digital, a través de una de sus patas de entrada analógica se mide la caída de tensión al aplicarse esfuerzo sobre el servo.

Conecta con la Raspberry Pi a través del puerto USB enviándole y recibiendo comandos de control serie para el Servomotor.

Su programa principal está desarrollado en una variación de C llamada Sketch. El cual es crosscompilado a través del Arduino IDE.

Raspberry Pi 3

Mini computadora cuyo propósito es ser el centro de procesamiento de todo el sistema de Puerta Segurizada, el sistema contiene un Sistema Operativo Debian Linux adaptado para arquitectura ARM, llamado Raspbian, y utiliza Python como lenguaje de programación principal.

Esta unidad es utilizada como:

Módulo embebido: a través de su interfaz GPIO (General Purpose Input Output), se conectarán los sensores y actuadores que manejen señales digitales: Sensor infrarrojo, Sensor de Sonido, LED's.

Salida de audio: La conexión JACK Stereo de este sistema permite la reproducción de un archivo .mp3

Interfaz con Arduino: Debido a la incapacidad de Raspberry Pi de procesar señales analógicas conectamos con Arduino, a través de uno de los puertos USB, recibiendo y enviando señales de control pertenecientes al Servomotor.

Conexión con la base de datos Firebase: Las actualizaciones de valores a utilizar por la aplicación se depositan en el servicio de base de datos en la nube "Firebase". Estos datos serán consumidos por el dispositivo móvil desde la aplicación Android.

Servomotor SG90

Motor de rotación controlada a través de una señal codificada (PPM), su rotación está limitada a un ángulo de 180 grados (-90 a 0 a 90). Es utilizado para el control de movimiento de la puerta, como actuador y sensor.

La característica principal de este dispositivo es que siempre realizará el mayor esfuerzo posible para volver a la posición que se le ha ordenado, esto nos permite realizar un monitoreo de cuando alguien o algo esté intentando forzar su posición actual, en nuestro caso, cuando alguien esté intentando forzar la puerta. Este comportamiento se detecta como una baja de tensión medida en una de las entradas analógicas del Arduino.

Sensor Infrarrojo FC-51 "Flying Fish"

Sensor basado en dos componentes, un LED emisor infrarrojo y un LED receptor de la señal.

Cuando se antepone el sensor a un obstáculo este se activará. Desactivándose cuando ya no esté en frente a esa superficie.

El sensor infrarrojo conecta con una de las entradas digitales GPIO de la Raspberry Pi.

Su función es la de solo permitir la apertura de la puerta habiendo alguien frente a ella.

Sensor de Sonido

Sensor compuesto por un micrófono electret, un circuito integrado amplificador (LM386) y una resistencia variable que nos permite calibrar el dispositivo. Está compuesto de dos pines de alimentación y una salida digital (en algunos modelos también analógica).

La salida digital del sensor conecta con uno de los pines GPIO digitales de la Raspberry y en base al programa principal se le proporcionará una secuencia de golpes que funcionan como una contraseña de acceso a la apertura de la puerta.

LED's

Diodos emisores de luz utilizados como actuadores, para mostrar una respuesta visual a las diferentes acciones aplicadas a la puerta.

Parlante

Dispositivo electromecánico compuesto por un electroimán, un bobinado y una bocina cónica, utilizado como actuador para reproducir un sonido al realizar una apertura exitosa de la puerta.

Está conectado a la raspberry a través del Jack de audio.

Celular

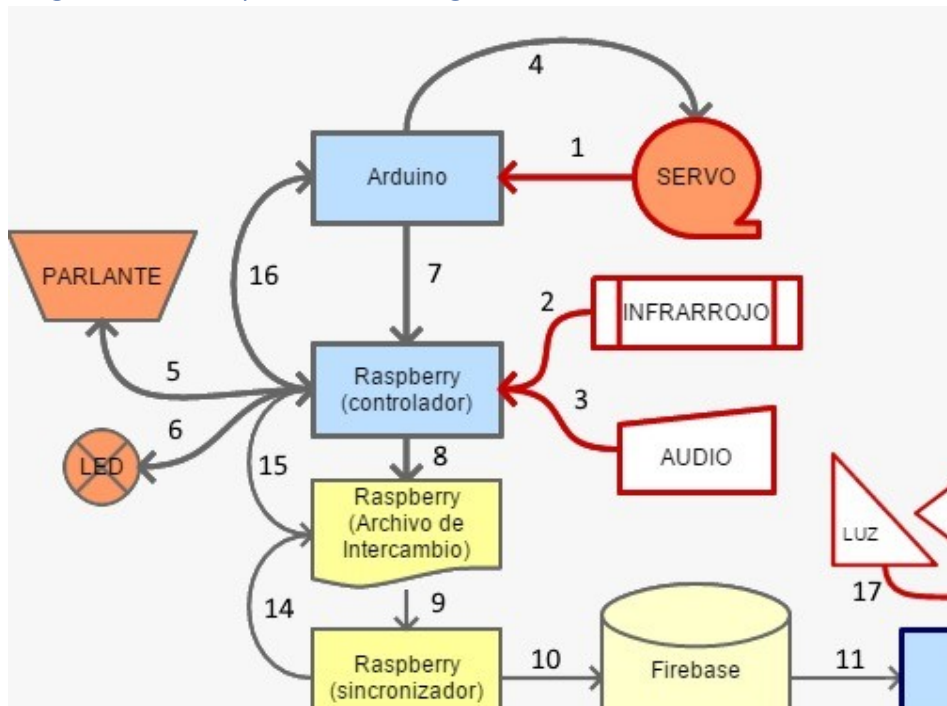
Dispositivo móvil con sistema operativo Android, contiene la aplicación dedicada al control de la puerta.

Comunica la aplicación con la base de datos en la nube "Firebase", consulta el estado actual de la puerta para así poder modificarlo.

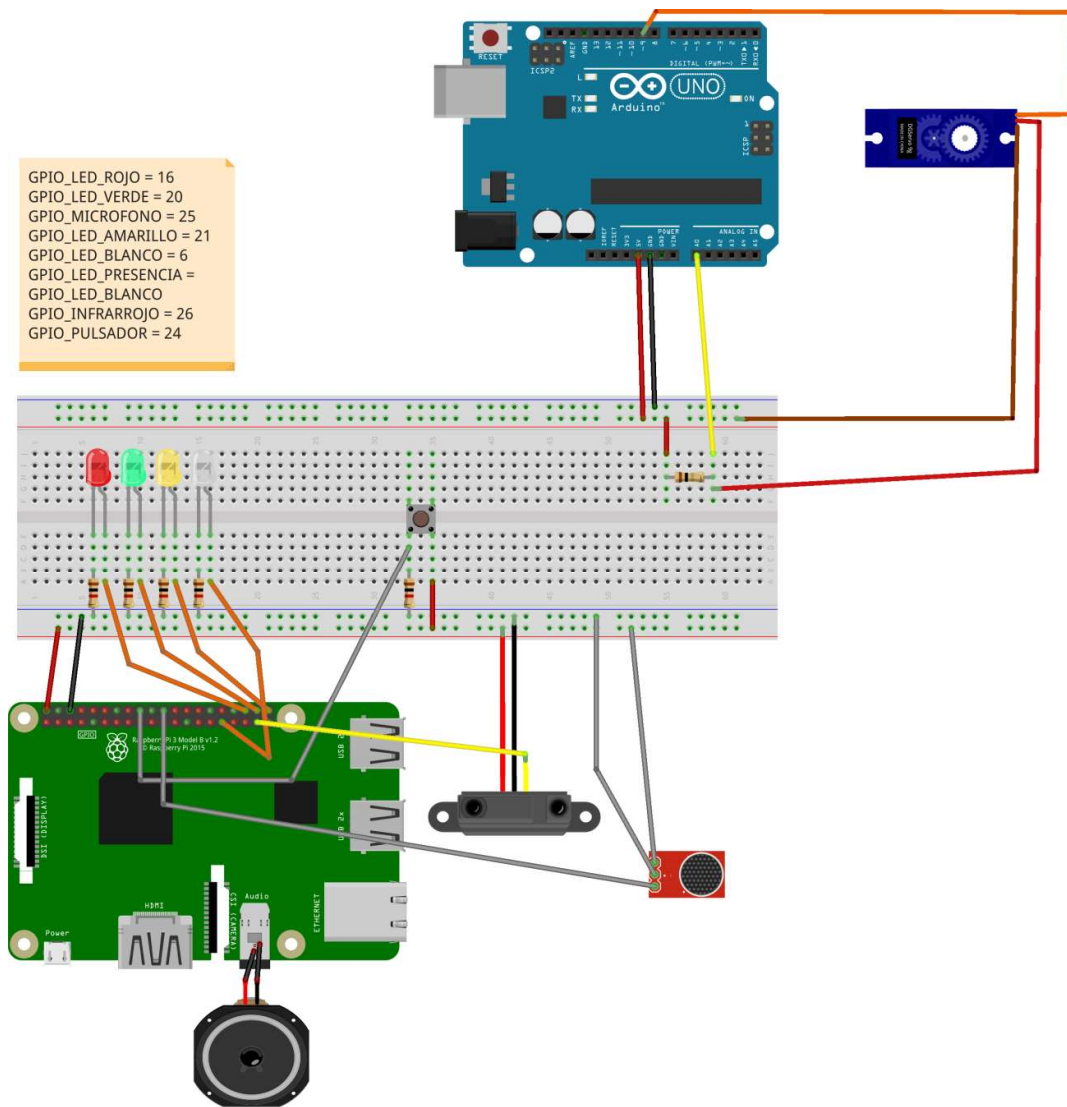
Utiliza 3 sensores para realizar las siguientes acciones:

- Luz: en caso de superar el umbral de luz definido, apaga el LED blanco. Caso contrario lo enciende.
- Proximidad: si se encuentra totalmente bloqueado enciende/apaga el LED blanco.
- Acelerómetro: Mediante el sensor de aceleración, calculamos las variaciones de movimiento del celular durante un determinado tiempo las cuales serán analizadas a través de una fórmula. Si el valor obtenido supera el valor "Shake" definido, abre o cierra la puerta.

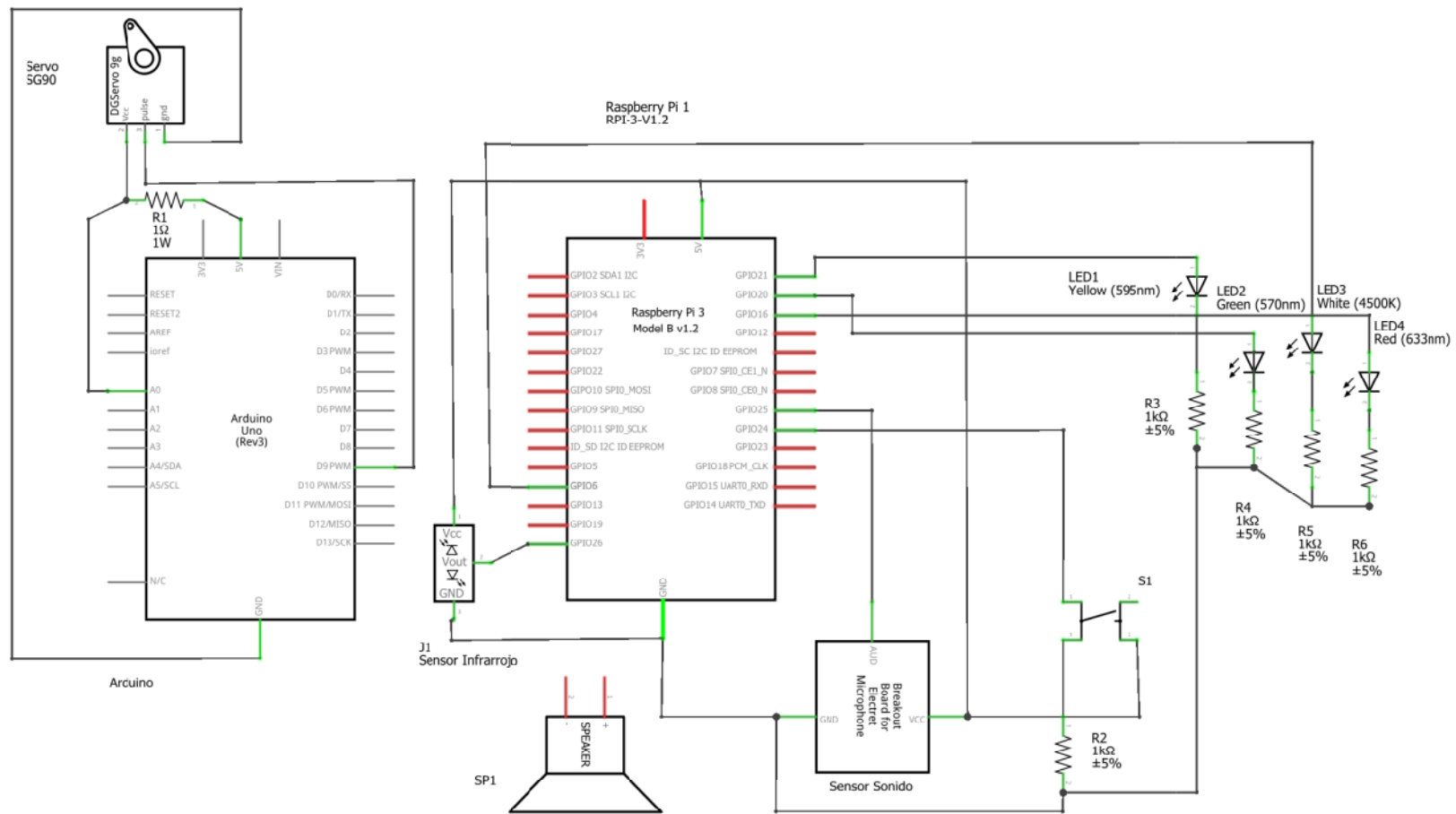
Diagrama en bloques de la configuración:



Circuito



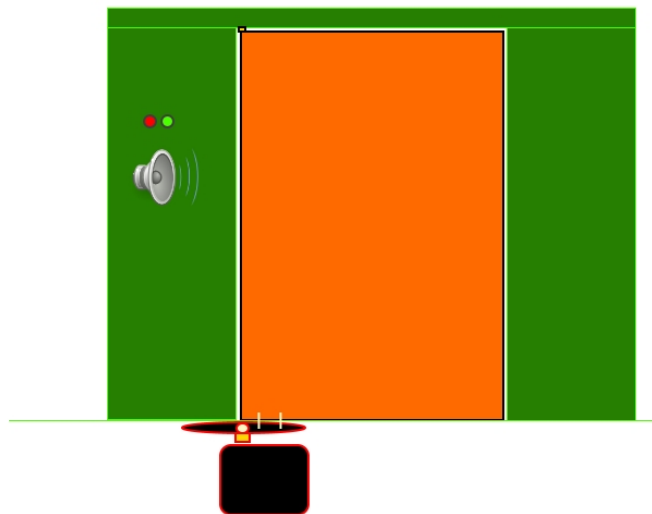
fritzing



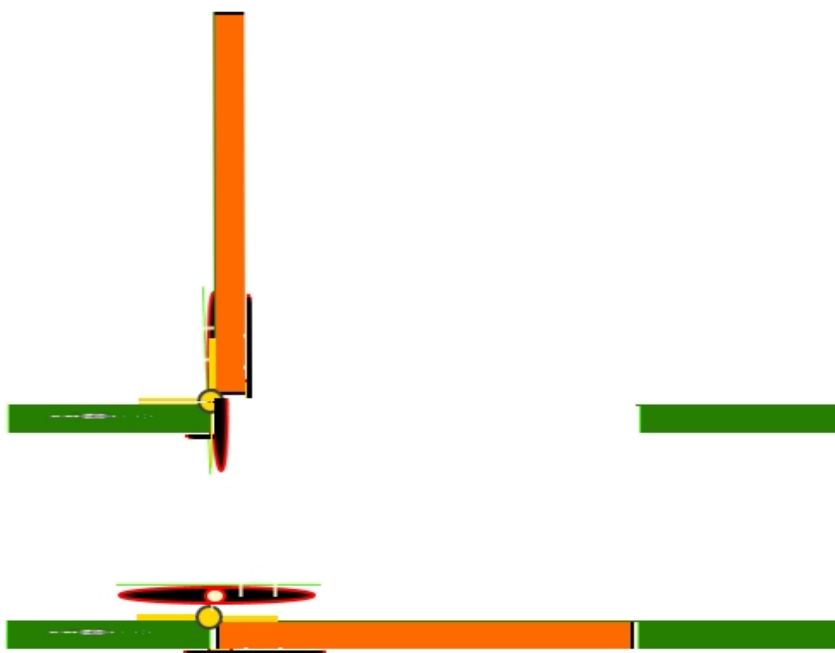
fritzing

Instalación:

Vista Frontal:



Vista Vertical:



Funcionamiento

Sección Embebidos

Arduino:

<https://github.com/SysEm/Arduino>

La principal función de este módulo es la operación del Servomotor a través de señales de control codificadas (PPM) utilizando la salida digital **PWM 9** del Arduino. Permitiendo la apertura y cierre de la puerta.

A través de la entrada analógica **A0** se mide la caída de tensión al aplicarse esfuerzo sobre el servo.

Estos valores y cambios sensados se transmiten mediante comunicación serial a través de USB con la Raspberry Pi.

La operación se basa en dos funciones principales:

accionarServo()

Para la operación del **servomotor** utilizamos la biblioteca "**Servo.h**" de Arduino la cual, a través de la función **write()**, nos permite manipular la posición del servo de forma angular.

Definimos como estados CERRADO = 180° y ABIERTO = 90°.

La función **accionarServo()** recibe un valor de tipo char específico, enviado al Arduino desde la Raspberry ,que le permite accionar el Servo en base a su estado actual.

Esta acción tiene un deadline de acción de 20 milisegundos aplicados según las especificaciones del servo.

Al aplicar un cambio de estado de apertura o cierre, el Arduino informa a la Raspberry.

verificarForzado()

Se define un "voltaje estacionario" **vEst** al inicializar el sistema, que es la tensión medida cuando el servo no recibe esfuerzo, se mide la tensión del entre el servo y la resistencia de 1 Ohm, a través de la entrada analógica A0.

Se define un valor de 3.3 Volts aproximados en base a la tensión del Arduino.

En base al valor medido en la pata A0, se mapea este valor proporcionalmente a los 3.3 volts medidos como tensión de Fuente:

$$vEst = \text{analogRead}(A0) * (vFuente / 1023);$$

Definimos la función "**verificarForzado()**" utilizando una variable llamada **voltajeMedido** que mapea proporcionalmente contra **vEst** en base a un **valorMedido** desde **analogRead(A0)**:

$$\text{float } \text{voltajeMedido} = \text{valorMedido} * (vEst / 1023);$$

Si el **voltajeMedido** es menor a la tensión estacionaria y a un umbral de tensión de forzado de 2.99 volts (obtenido en base a mediciones anteriores), definimos el estado de la puerta en FORZADO, sinó el estado es ESTACIONARIO. Esta medición se realiza cada 200 milisegundos y se envía a la raspberry a través del objeto Serial.

Raspberry:

<https://github.com/SysEm/Raspberry>

Utiliza 4 scripts python para realizar las siguientes acciones:

control-puerta.py

Script principal que se encarga del control de sensores y estados del sistema, las condiciones que evalúa son:

- Debe haber algo o alguien frente al sensor infrarrojo para que la puerta pueda abrirse del lado del afuera.
- La puerta se abrirá luego de una secuencia de golpes detectadas por el sensor de sonido.
- Se detectará si la puerta está siendo forzada, a través de una señal de control recibida desde el Arduino.
- El Pulsador puede abrir la puerta en cualquier circunstancia. No es necesario que exista alguien frente al sensor infrarrojo.

sincro-puerta.py

Script encargado de la actualización de la base de datos en la nube Firebase:

Básicamente utiliza dos objetos *json* que contienen los estados de la puerta, uno se utiliza para escribir desde la Raspberry hacia Firebase y el otro para leer las peticiones desde la Apk Android grabados en Firebase:

puertaDesdeRaspi="PuertaLecAppEscRas"

puertaHaciaRaspi="PuertaEscAppLecRas"

Los archivos utilizan el siguiente formato:

```
{"Led": {"estado": P_LED_EST_OFF}, "Presencia": {"estado": P_PRES_EST_OFF}, "Servo": {"angulo": P_SERV_ANG_CERRADO, "esfuerzo": P_SERV_ESFUERZO_NO}}
```

puertasfunciones.py

Funciones comunes utilizadas por los demás scripts:

- Leer archivo
- Escribir archivo
- Definir nivel de log: **DEBUG, INFO, WARNING, ERROR.**

genera-password-sonido.py

Aplicación utilizada para la generación de password sonora:

- Solicita una secuencia de golpes captados por el sensor de sonido, por defecto está definido:

cantidadGolpes = 3

- Cada golpe tiene un deadline de sensado, definido en:

tiempoEscucha = 3

Los valores son por defecto y pueden modificarse.

Luego del sensado de la contraseña se solicitará repetirla para validación. Si está OK se grabará en el archivo *password.txt*

Sección IoT

Android:

<https://github.com/SysEm/SoaTp>

La aplicación móvil actúa como un control remoto para poder controlar algunos componentes de la puerta física y mostrar información de ella. Estas funciones son: Apertura remota de puerta, encendido/apagado de luz y apagado/encendido de luz de acuerdo a la intensidad de luz que haya en el ambiente, notificar al usuario si es que la puerta esta forzada y mostrar el estado de la luz y la puerta y si es que hay gente frente a ella.

Para lograr esto se utilizó las clases `Sensor`, `SensorEvent`, `SensorManager` y la interfaz `SensorEventListener` del paquete `android.hardware` y una base de datos `Firebase`, mediante la cual estableceremos comunicación con la puerta real. También se necesitó incorporar el permiso del vibrador, el cual se ejecuta en la notificación de puerta forzada, en el archivo `Manifest` ya que al ser `hardware` está protegido.

Sensores de la aplicación:

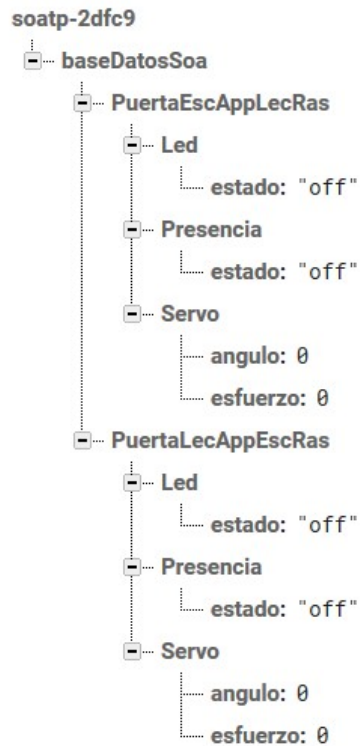
Mediante el uso de los sensores de luz, proximidad y aceleración, de los cuales se monitorean los cambios de valores, se hicieron las siguientes funcionalidades:

- **Activar luz inteligente:** Se determinó un umbral de luz y mediante el sensor de luz, en caso de que se registre un valor mayor a ese umbral, se enviara una solicitud para que la luz se apague. Caso contrario la solicitud es de apagado.
- **Activar apertura de puerta remota:** Se toman valores de los parámetros X, Y y Z del sensor de aceleración durante unos segundos. Mediante una fórmula se obtiene un valor y si ese valor supera nuestro umbral “*Shake*” entonces dependiendo del estado actual de la puerta, se envía una solicitud de apertura/cierre.
- **Activar encendido de luz remota:** Mediante el sensor de proximidad, si este es bloqueado completamente, se enviará la solicitud de apagado/encendido de la luz según corresponda.

Aclaración: Los cambios de los valores de los sensores usados son registrados mediante la función “**onSensorChanged**” propia de la interfaz **SensorEventListener**. Solamente se puede seleccionar una función a la vez.

Base de Datos Firebase:

La aplicación conecta a la base de datos `Firebase`, este es un servicio de BBDD con consultas de tipo `JSON`, que permite notificaciones en tiempo real. Nuestro esquema consta de 2 objetos “**PuertaLecAppEscRas**” (la aplicación lee y la puerta escribe) y “**PuertaEscAppLecRas**” (la puerta lee y la aplicación escribe).



Información mostrada en la aplicación:

Mediante la lectura de “*PuertaLecAppEscRas*” de la BBDD, la aplicación suministra en tiempo real el estado de la puerta como por ejemplo: abierta/cerrada. Además, en caso de que la puerta esté siendo forzada, la aplicación desplegará una notificación visual y sonora de tal acontecimiento. Esta notificación se realizará aun estando en segundo plano.

SoaTp

Luz:

Apagada

Puerta:

Abierta

Gente en puerta?

Si

Funcionalidades



Activar encendido de luz remota



Activar apertura de puerta remota



Activar luz inteligente

100.0

VALOR IIMI

Anexo I: Hojas de datos de componentes

Arduino

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Raspberry Pi 3 model b

The Raspberry Pi 3 is the third-generation Raspberry Pi. It replaced the Raspberry Pi 2 Model B in February 2016.

- Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
- 1GB RAM
- BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board
- 40-pin extended GPIO
- 4 USB 2 ports
- 4 Pole stereo output and composite video port
- Full size HDMI
- CSI camera port for connecting a Raspberry Pi camera
- DSI display port for connecting a Raspberry Pi touchscreen display
- Micro SD port for loading your operating system and storing data
- Upgraded switched Micro USB power source up to 2.5A

BCM2837

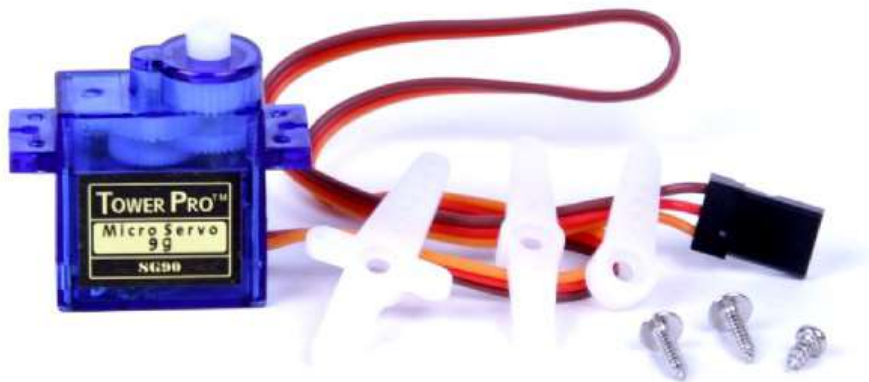
This is the Broadcom chip used in the Raspberry Pi 3, and in later models of the Raspberry Pi 2. The underlying architecture of the BCM2837 is identical to the BCM2836. The only significant

difference is the replacement of the ARMv7 quad core cluster with a **quad-core ARM Cortex A53 (ARMv8) cluster**.

The ARM cores run at 1.2GHz, making the device about 50% faster than the Raspberry Pi 2. The VideocoreIV runs at 400Mhz.

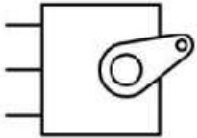
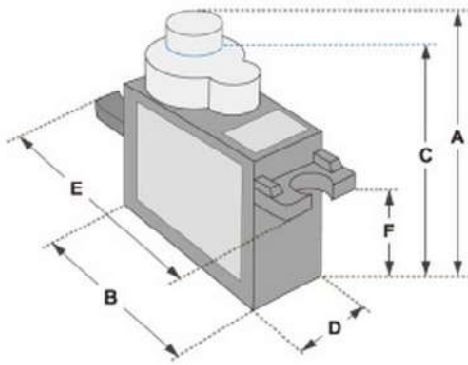
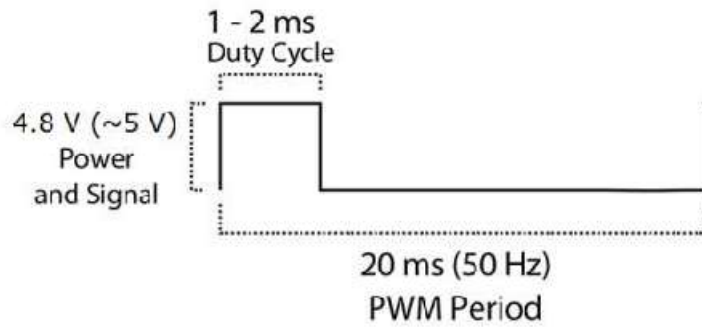
SERVO MOTOR SG90

Tiny and lightweight with high output power. Servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds but smaller. You can use any servo code, hardware or library to control these servos. Good for beginners who want to make stuff move without building a motor controller with feedback & gear box, especially since it will fit in small places. It comes with a 3 horns (arms) and hardware.



Position "0" (1.5 ms pulse) is middle, "90" (~2ms pulse) is middle, is all the way to the right, "-90" (~1ms pulse) is all the way to the left.

PWM=Orange (⏏)
Vcc = Red (+)
Ground=Brown (-)

Dimensions & Specifications	
A (mm) :	32
B (mm) :	23
C (mm) :	28.5
D (mm) :	12
E (mm) :	32
F (mm) :	19.5
Speed (sec) :	0.1
Torque (kg-cm) :	2.5
Weight (g) :	14.7
Voltage :	4.8 - 6

SPECIFICATIONS:

Torque	25.0 oz-in (1.80 kg-cm) at 4.8V
Speed	0.1sec/60° (4.8V)
Voltage	4.0V to 7.2V, 4.6V - 5.2V nominal
Running current with 5V supply (no mechanical load)	220 ±50mA
Stall current with 5V supply (horn locked)	650 ±80mA
Idle current with 5V supply	6 ±10mA
Dimensions	0.91in x 0.48in x 1.14in (23mm x 12.2mm x 29mm)
Weight	0.32oz (9g)
Dead band width	10µs
Operating Temperature range	-22°F to 140°F (-30°C to 60°C)
Universal "S" type connector fits most receivers	

IR Infrared Obstacle Detection Sensor Module 2 - 30cm FC-51 “Flying Fish”

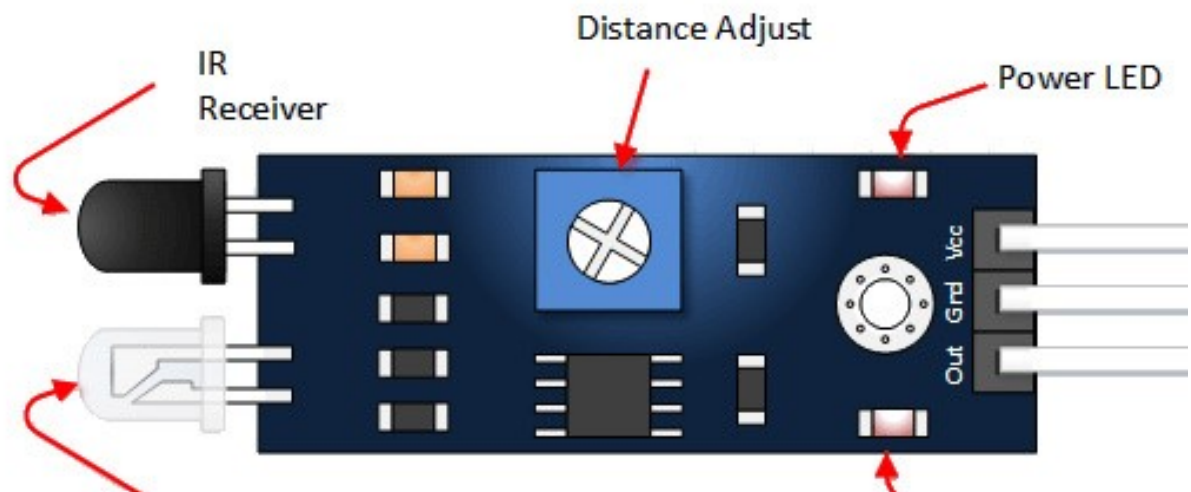
The basic concept of IR(infrared) obstacle detection is to transmit the IR signal(radiation) in a direction and a signal is received at the IR receiver when the IR radiation bounces back from a surface of the object.

Features:

- There is an obstacle, the green indicator light on the circuit board
- Digital output signal
- Detection distance: 2 ~ 30cm
- Detection angle: 35 ° Degree
- Comparator chip: LM393
- Adjustable detection distance range via potentiometer:
 - Clockwise: Increase detection distance
 - Counter-clockwise: Reduce detection distance

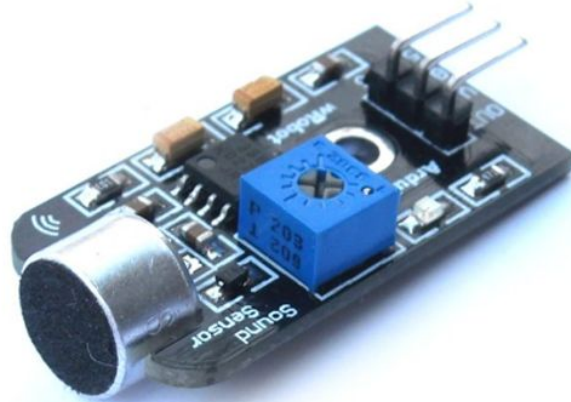
Specifications:

- Working voltage: 3 - 5V DC
- Output type: Digital switching output (0 and 1)
- 3mm screw holes for easy mounting
- Board size: 3.2 x 1.4cm



Pin, Control Indicator	Description
Vcc	3.3 to 5 Vdc Supply Input
Gnd	Ground Input
Out	Output that goes low when obstacle is in range
Power LED	Illuminates when power is applied
Obstacle LED	Illuminates when obstacle is detected
Distance Adjust	Adjust detection distance. CCW decreases distance. CW increases distance.
IR Emitter	Infrared emitter LED
IR Receiver	Infrared receiver that receives signal transmitted by Infrared emitter.

SENSOR DE SONIDO



En la parte de la izquierda vemos los pines de conexión:

- En el centro tenemos la conexión a 5V y a GND (+ y G).
- D0 es una salida digital que actúa a modo de comparador. Si el sonido captado por el micrófono supera un determinado nivel se pone a HIGH.

Tenemos un LED que nos indica si hay alimentación en el sensor.

El ajuste de sensibilidad del micrófono lo hacemos mediante un potenciómetro que tendremos que girar con un destornillador tipo phillips.

Sólo necesitamos conectar el pin D0 y los dos pines de alimentación, si hemos conectado bien el sensor, se debería iluminar el LED de alimentación.

AJUSTANDO EL LÍMITE DE DISPARO

Esta es seguramente la parte más complicada de esta sesión. Para ajustar el límite de disparo lo que hacemos es girar el potenciómetro con un destornillador. Tenemos que dejarlo de tal forma que el LED que marca si está accionada la salida digital esté apagado, pero lo más próximo posible al límite en el que se enciende.

Si lo ajustamos mal y el LED se está encendido, no detectaremos ningún cambio y no podremos reaccionar a ningún estímulo sonoro.

Si lo ajustamos de forma que esté apagado pero demasiado lejos del límite en el que se enciende, habrá que llamar al increíble Hulk para que dé una palmada por nosotros.

Anexo II: Código principal de Apk Android

```
package com.soa.javier.soatp;

import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.media.MediaPlayer;
import android.os.AsyncTask;
import android.os.Vibrator;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.Switch;
import android.widget.TextView;
import android.widget.Toast;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.soa.javier.soatp.Objetos.Puerta;

public class MainActivity extends AppCompatActivity implements
SensorEventListener{
    //region VARIABLES
    //region SENSORES
    SensorManager sm;
    Sensor sensorProx;
    Sensor sensorAcel;
    Sensor sensorLumi;
    //endregion
    //region VIEWS
    TextView tviewLed;
    TextView tviewPuerta;
    TextView tviewPresencia;
    TextView tviewNotificacion;
    Switch switchActivacion;
    RadioButton rBtnLed;
    RadioButton rBtnPuerta;
    RadioButton rBtnLuminosidad;
    RadioGroup radioGroup;
    Button btnOk;
    //endregion
    //region AUXILIARES
    Puerta PuertaLecAppEscRas = new Puerta();
    Puerta PuertaEscAppLecRas = new Puerta();

    String estadoPeticiónLed;
    TextView tviewProx;
    TextView tviewLumi;
    private static final float SHAKE_THRESHOLD = 2.1f;
    private static final int SHAKE_WAIT_TIME_MS = 500;
    private long mShakeTime = 0;
    Vibrator vibrator;
    MediaPlayer mp;
    //endregion
    //region BASE DE DATOS
    FirebaseDatabase database = FirebaseDatabase.getInstance();
```

```

DatabaseReference baseDatosSoaRef = database.getReference("baseDatosSoa");
//endregion
//endregion

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    Log.i("LOG:", "ESTOY EN CREATE");

    //INSTANCIO Y RELACIONO LOS ELEMENTOS QUE VOY A USAR EN LA PANTALLA.
    tviewLed = (TextView) findViewById(R.id.tviewLed);
    tviewPuerta = (TextView) findViewById(R.id.tviewPuerta);
    tviewPresencia = (TextView) findViewById(R.id.tviewPresencia);
    tViewNotificacion = (TextView) findViewById(R.id.tViewNotificacion);

    radioGroup = (RadioGroup) findViewById(R.id.radioGroup);
    for(int i = 0; i < radioGroup.getChildCount(); i++){
        ((RadioButton)radioGroup.getChildAt(i)).setEnabled(false);
    }

    rBtnLed = (RadioButton) findViewById(R.id.rBtnLed);
    rBtnPuerta = (RadioButton) findViewById(R.id.rBtnPuerta);
    rBtnLuminosidad = (RadioButton) findViewById(R.id.rBtnLuminosidad);

    //INSTANCIO LOS SENSORES
    sm = (SensorManager) getSystemService(SENSOR_SERVICE);
    sensorProx = sm.getDefaultSensor(Sensor.TYPE_PROXIMITY);
    sensorAcel = sm.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
    sensorLumi = sm.getDefaultSensor(Sensor.TYPE_LIGHT);

    //INSTANCIO SENSORES PARA QUE EMPIECEN A ESCUCHAR.
    activarSensores();

    //ACTIVO-DESACTIVO EL MENU DE FUNCIONALIDADES
    switchActivacion = (Switch) findViewById(R.id.switchActivacion);
    switchActivacion.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            if(switchActivacion.isChecked()){
                for(int i = 0; i < radioGroup.getChildCount(); i++){
                    ((RadioButton)radioGroup.getChildAt(i)).setEnabled(true);
                }
            }else{
                for(int i = 0; i < radioGroup.getChildCount(); i++){
                    ((RadioButton)radioGroup.getChildAt(i)).setEnabled(false);
                }
                radioGroup.clearCheck();
            }
        }
    });

    //BOTON PARA CANCELAR LA NOTIFICACION DE PUERTA FORZADA
    btnOk = (Button) findViewById(R.id.btnOk);
    btnOk.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            vibrator.cancel();
            mp.stop();
            tViewNotificacion.setVisibility(View.INVISIBLE);
            btnOk.setVisibility(View.INVISIBLE);
        }
    });
}

```



```

        btnOk.setVisibility(View.VISIBLE);
        mp.start();
        vibrator.vibrate(8000L);
    }

    }catch (NullPointerException e){
        // error, seguramente nombre de campos incorrectos
    devuelven objeto NULO
        tviewLed.setText("Error obteniendo datos BBDD");
    }

    }

    @Override
    public void onCancelled(DatabaseError databaseError) {

    }

    });
}

//region FUNCIONES DE CAMBIO DE ESTADO DE SENSORES
@Override
public void onSensorChanged(SensorEvent sensorEvent) {
    if(sensorEvent.sensor.getType()== Sensor.TYPE_ACCELEROMETER){
        if(rBtnPuerta.isChecked()){
            accionShake(sensorEvent);
        }
    }else{
        if(sensorEvent.sensor.getType() == Sensor.TYPE_PROXIMITY){
            if(rBtnLed.isChecked()){
                accionProximidad(sensorEvent);
            }
        }else{
            if(sensorEvent.sensor.getType() == Sensor.TYPE_LIGHT){
                if(rBtnLuminosidad.isChecked()){
                    accionLuminosidad(sensorEvent);
                }
            }
        }
    }
}

@Override
public void onAccuracyChanged(Sensor sensor, int i) {

}

//endregion

//region FUNCIONES PARA LOS SENSORES
public void accionProximidad(SensorEvent sensorEvent){
    String valProximidad = String.valueOf(sensorEvent.values[0]);
    Float valor = Float.parseFloat(valProximidad);
    tviewProx.setText(Float.toString(valor));

    //VALOR ES LO QUE ME DA EL SENSOR ES EL DE PROXIMIDAD
    if(valor == 1){
        sm.unregisterListener(this);
        if(PuertaLecAppEscRas.getLed().getEstado().equals("on")){
            Toast toast = Toast.makeText(getApplicationContext(),
"TRATANDO DE APAGAR LA LUZ", Toast.LENGTH_LONG);
            toast.show();
        }

        baseDatosSoaRef.child("PuertaEscAppLecRas").child("Led").child("estado").setValue("off");
    }
}

```



```

        PuertaEscAppLecRas.setLed("off");
    }else {
        Toast toast = Toast.makeText(getApplicationContext(),
"TRATANDO DE ENCENDER LA LUZ", Toast.LENGTH_LONG);
        toast.show();

baseDatosSoaRef.child("PuertaEscAppLecRas").child("Led").child("estado").setVa
lue("on");

        PuertaEscAppLecRas.setLed("on");
    }
    TareaLed tareaLed = new TareaLed();
    tareaLed.execute();
}

}

public void accionLuminosidad(SensorEvent sensorEvent){
    String valluminosidad = String.valueOf(sensorEvent.values[0]);
    Float valor = Float.parseFloat(valLuminosidad);
    tvviewLumi.setText(Float.toString(valor));

    if(valor > 150){
        if(PuertaLecAppEscRas.getLed().getEstado().equals("on")){
            sm.unregisterListener(this);
            Toast toast = Toast.makeText(getApplicationContext(),
"TRATANDO DE APAGAR LA LUZ", Toast.LENGTH_LONG);
            toast.show();

baseDatosSoaRef.child("PuertaEscAppLecRas").child("Led").child("estado").setVa
lue("off");

            PuertaEscAppLecRas.setLed("off");
            TareaLed tareaLed = new TareaLed();
            tareaLed.execute();
        }

    }else{
        if(PuertaLecAppEscRas.getLed().getEstado().equals("off")){
            sm.unregisterListener(this);
            Toast toast = Toast.makeText(getApplicationContext(),
"TRATANDO DE ENCENDER LA LUZ", Toast.LENGTH_LONG);
            toast.show();

baseDatosSoaRef.child("PuertaEscAppLecRas").child("Led").child("estado").setVa
lue("on");

            PuertaEscAppLecRas.setLed("on");
            TareaLed tareaLed = new TareaLed();
            tareaLed.execute();
        }

    }

}

public void accionShake(SensorEvent sensorEvent){
    long now = System.currentTimeMillis();

    if ((now - mShakeTime) > SHAKE_WAIT_TIME_MS) {
        mShakeTime = now;

        float gX = sensorEvent.values[0] / SensorManager.GRAVITY_EARTH;
        float gY = sensorEvent.values[1] / SensorManager.GRAVITY_EARTH;
        float gZ = sensorEvent.values[2] / SensorManager.GRAVITY_EARTH;

        double gForce = Math.sqrt(gX * gX + gY * gY + gZ * gZ);

        if (gForce > SHAKE_THRESHOLD) {
            sm.unregisterListener(this);
            Toast toast = Toast.makeText(getApplicationContext(), "SHAKE
DETECTADO", Toast.LENGTH_SHORT);
            toast.show();
        }
    }
}

```

```

        if(PuertaLecAppEscRas.getServo().getAngulo() == 0){
            toast = Toast.makeText(getApplicationContext(), "TRATANDO
DE ABRIR", Toast.LENGTH_LONG);
            toast.show();

baseDatosSoaRef.child("PuertaEscAppLecRas").child("Servo").child("angulo").set
Value(1);

            PuertaEscAppLecRas.setServo(1,0);
        }
        else {
            toast = Toast.makeText(getApplicationContext(), "TRATANDO
DE CERRAR", Toast.LENGTH_LONG);
            toast.show();

baseDatosSoaRef.child("PuertaEscAppLecRas").child("Servo").child("angulo").set
Value(0);

            PuertaEscAppLecRas.setServo(0,0);
        }
        TareaServo tareaServo = new TareaServo();
        tareaServo.execute();
    }
}

//endregion

//region OVERRIDE DE LOS ESTADOS DE LA APP
@Override
protected void onResume() {
    super.onResume();
    Log.i("LOG:", "ESTOY EN RESUME");
    activarSensores();
}

@Override
protected void onPause() {
    super.onPause();
    Log.i("LOG:", "ESTOY EN PAUSE");
    sm.unregisterListener(this);
}

@Override
protected void onStop() {
    super.onStop();
    Log.i("LOG:", "ESTOY EN STOP");
}

@Override
protected void onRestart() {
    super.onRestart();
    Log.i("LOG:", "ESTOY EN RESTART");
}

@Override
protected void onDestroy() {
    super.onDestroy();
    Log.i("LOG:", "ESTOY EN DESTROY");
    sm.unregisterListener(this);
}

//endregion

public void activarSensores(){
    sm.registerListener(this, sensorProx,
SensorManager.SENSOR_DELAY_NORMAL);
    sm.registerListener(this, sensorAcel,
SensorManager.SENSOR_DELAY_NORMAL);
}

```

```

        sm.registerListener(this, sensorLumi,
SensorManager.SENSOR_DELAY_NORMAL);
    }

    private class TareaLed extends AsyncTask<Void,Void,Void> {
        @Override
        protected Void doInBackground(Void... voids) {
            Log.i("ASYNC","Inicie");
            Long despues = System.currentTimeMillis()+5000;
            while(System.currentTimeMillis()<=despues){

            }
            return null;
        }

        @Override
        protected void onPostExecute(Void aVoid) {
            super.onPostExecute(aVoid);

            if(PuertaEscAppLecRas.getLed().getEstado().equals(PuertaLecAppEscRas.getLed().
getEstado())){
                Toast toast =
                Toast.makeText(MainActivity.this.getContext(),"Accion ejecutada
 exitosamente",Toast.LENGTH_SHORT);
                toast.show();
            }else{
                Toast toast =
                Toast.makeText(MainActivity.this.getContext(),"Accion no
 ejecutada",Toast.LENGTH_SHORT);
                toast.show();

                baseDatosSoaRef.child("PuertaEscAppLecRas").child("Led").child("estado").setVa
lue(PuertaLecAppEscRas.getLed().getEstado());
            }
            activarSensores();
            Log.i("ASYNC","Finalice");
        }
    }

    private class TareaServo extends AsyncTask<Void,Void,Void> {
        @Override
        protected Void doInBackground(Void... voids) {
            Log.i("ASYNC","Inicie");
            Long despues = System.currentTimeMillis()+5000;
            while(System.currentTimeMillis()<=despues){

            }
            return null;
        }

        @Override
        protected void onPostExecute(Void aVoid) {
            super.onPostExecute(aVoid);

            if(PuertaEscAppLecRas.getServo().getAngulo()==PuertaLecAppEscRas.getServo().ge
tAngulo()){
                Toast toast =
                Toast.makeText(MainActivity.this.getContext(),"Accion ejecutada
 exitosamente",Toast.LENGTH_SHORT);
                toast.show();
            }else{
                Toast toast =
                Toast.makeText(MainActivity.this.getContext(),"Accion no
 ejecutada",Toast.LENGTH_SHORT);
                toast.show();
            }
        }
    }

```

```

baseDatosSoaRef.child("PuertaEscAppLecRas").child("Servo").child("angulo").set
Value(PuertaLecAppEscRas.getServo().getAngulo());
    }
    activarSensores();
    Log.i("ASYNC","Finalice");
    TareaServo.this.cancel(true);
    return;
    }
}

```

Anexo III: Base de datos Firebase

<https://console.firebase.google.com/project/soatp-2dfc9/database/soatp-2dfc9/data>