UNIVERSIDAD NACIONAL DE LA MATANZA

DEPTO. DE INGENIERIA E INVESTIGACIONES TECNOLOGICAS

Internet of Things Sistemas Embebidos + Android

Integrantes:

Romano Dario	DNI:	33.901.015
Gonzalez Gustavo	DNI:	94.490.934
Francini Lucas	DNI:	34.513.154
Calandra Checco Daniel	DNI:	30.444.305

Índice

Objetivo

Desarrollar los conocimientos y experiencias adoptados en la cursada acerca de Sistemas Embebidos, IoT y Android con el fin de implementar un sistema para cuna que controle la temperatura ambiente y con el sensor de sonido y movimiento detectar si el bebe se desperto.

Descripción del entorno de desarrollo

Hardware:

Sistema Embebido

- Notebook Lenovo B40 (Intel Core i3– 8 GB RAM – Windows 7)
- Arduino Uno
- Cables para la conexión entre placa y sensores (macho-macho y hembra-macho)
- 1 x Placa de desarrollo
- 1 x Sensor TILT
- 1 x Sensor PIR
- 1 x Sensor de Sonido KY-038
- 1 x Sensor de temperatura LM36
- 1 x Buzzer
- 3 x Resistencia
- 1 x Boton
- 1 x Módulo bluetooth HC-05
- 2 x LED

Aplicación Android

- Notebook Dell Latitude E5550 (Intel Core i5- 16 GB RAM - Windows 7)
- Celular one plus one

Software:

Sistema Embebido

- Arduino IDE 1.8.2
- Windows 7 Profesional x64

Aplicación Android

- Android SD version : API 18 Andriod 4.3 (Jelly Bean) Built tools version 25.0.3
- Windows 7 Enterprise 64

Alcance del Sistema (SE + Android)

Sistema Embebido:

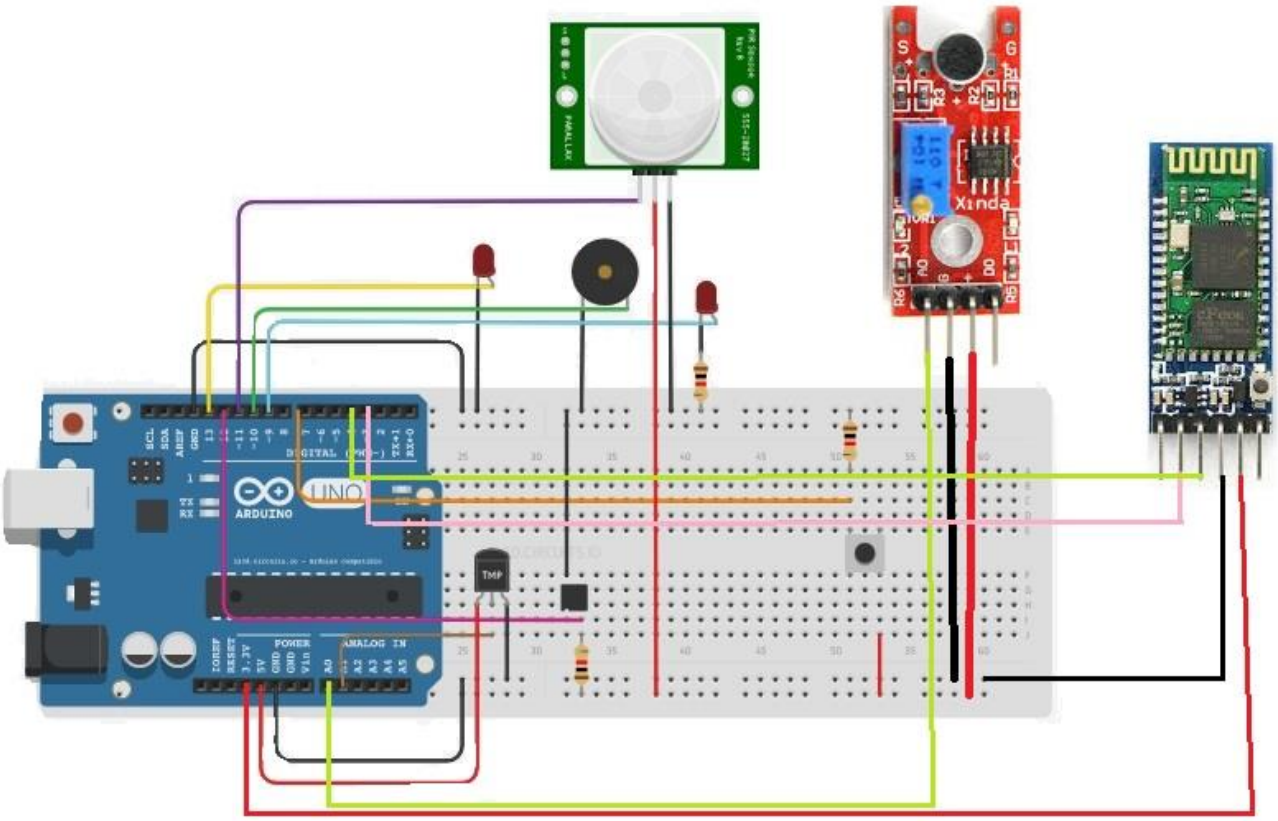
1. Debe medir la temperatura del ambiente.
2. Debe medir la el sonido del ambiente.
3. Debe controlar si hay movimiento frente a el.
4. En caso de que la temperatura exceda los rangos configurados, debe informar al usuario para que el mismo tome una acción.
5. En caso de que se detecte sonido, debe informar al usuario y además debe activar una música para tratar de lograr que el bebe se duerma nuevamente.
6. En caso de que detecte movimiento debe informarle al usuario para que realice una acción, adema lo informa con un led.
7. Además el sistema posee un sensor TILT, que se encarga de garantizar que el sistema este en la posición correcta. En caso de no estarlo, le informa al usuario y cancela la medición de movimiento ya que la misma puede ser errónea.

Sistema Android:

1. Debe conectarse por bluetooth al SE (por el módulo HC-05).
2. Debe poder encender y apagar el SE.
3. Debe poder configurar el rango de temperatura a controlar por el sensor del SE.
4. Recibe el estado de todos los sensores del SE y los informa por pantalla, además de mostrarlo en la barra de notificación y vibrar.

Diseño (ARDUINO)

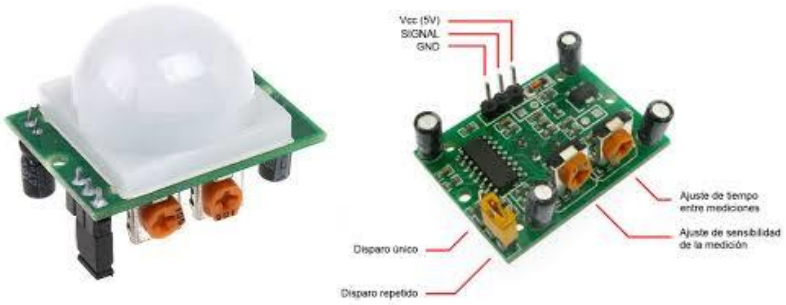
A continuación podemos ver un diagrama el cual hace referencia a las conexiones del dispositivos Arduino.



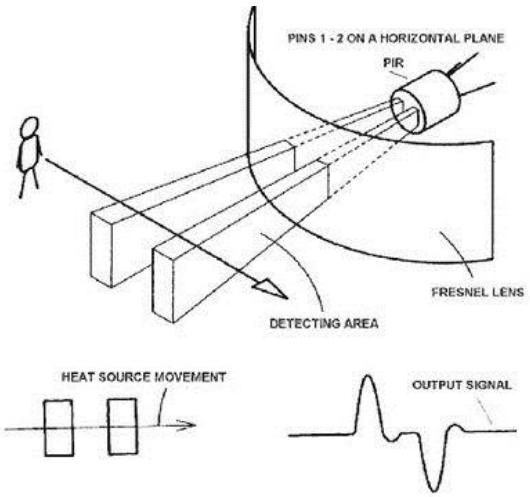
El sistema tiene cuatro sensores, el de temperatura, el de sonido, el de inclinación y el de movimiento. Además posee dos LED, un Buzzer y un botón. El botón se utiliza para encender y apagar el dispositivo. Un led informa si el sistema esta encendido, el otro es el encargado de informar si algún sensor se activo. El sensor de temperatura se encarga de ver si la temperatura ambiente se encuentra dentro de los rangos configurados y en caso de que no esté enciende el led e informa a la aplicación android.
El sensor de movimiento es el encargado de detectar si hay movimiento enfrente de el, en caso de detectarlo lo informa al android para que el usuario tome una acción al respecto. El sensor de sonido es el encargado de monitorear el ambiente y si detecta un sonido superior al 12% del sonido estándar, enciende un LED e informa al sistema android. Además enciende una música para tratar de que el bebe se duerma nuevamente.

Info de los sensores utilizados (ARDUINO)

SENSOR PIR



Los PIR más frecuentes son sensores de movimiento, y para ello están divididos en dos mitades de forma que detecten el cambio de radiación IR que reciben uno y otro lado, disparando la alarma cuando perciben ese cambio.

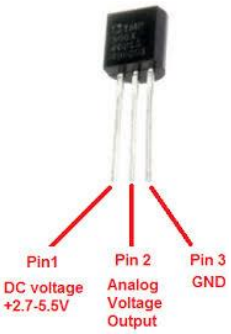


En el SE es el encargado de controlar el movimiento delante de él y lo informa a la aplicación android.

Código Arduino utilizado:

```
    int value= digitalRead(PIRPin);
    if (value == HIGH and tiltActivado==0){//si hay movimiento se activa
      digitalWrite(LEDPin, HIGH);//enciende el led
      BT1.println(4);//envía por bluetooth la acción que ocurrió
    }else{
      digitalWrite(LEDPin, LOW);//apaga el led
      BT1.println(5);//envía por bluetooth la acción que ocurrió
  }
```

SENSOR LM36



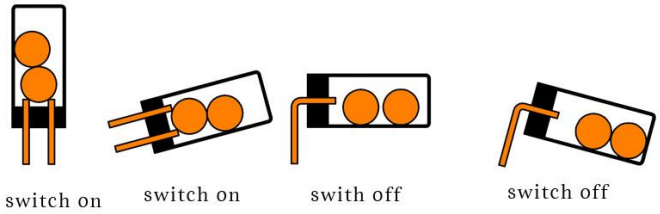
El TMP36 es un sensor de temperatura en grados centígrados de precisión y bajo voltaje. La salida de voltaje que proporciona es linealmente proporcional a la temperatura en grados Celsius. No requiere ninguna calibración externa para proporcionar una precisión típica de ± 1 °C a +25 °C y ± 2 °C por encima del Rango de temperatura de -40 °C a +125 °C. Este sensor es muy fácil de usar, basta con conectarlo a tierra y a un voltaje de 2.7 a 5.5 VDC y se podrán comenzar a tomar lecturas por el pin Vout. El voltaje de salida puede ser convertido fácilmente a temperatura utilizando el factor de escala de 10 mV/¼C.

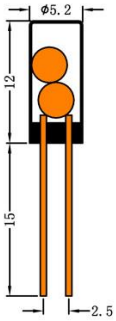
En el Se es el encargado de medir la temperatura ambiente y controlar que este dentro de los rangos configurados. En caso de no estarlo enciende un led e informa a la aplicación android del problema para que el usuario tome alguna acción.

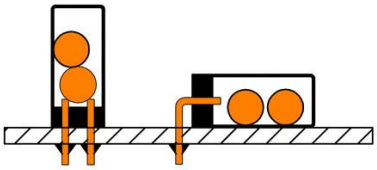
Código Arduino utilizado:

```
int lectura = analogRead(Temperatura);
  float voltaje = 5.0 /1024 * lectura ;
  float temp = voltaje * 100 -50 ;
  if (temp< tempMin or temp > tempMax) {
    digitalWrite(LEDPin, HIGH) ;//enciende el led, que significa que activo el AC
    BT1.println(2);//envía por bluetooth la acción que ocurrió
  }else{
    digitalWrite(LEDPin,LOW);//apaga el led
    BT1.println(3);//envía por bluetooth la acción que ocurrió
  }
```

SENSOR TILT







Cuando hacen contacto permiten el paso de la corriente y cierran el contacto exactamente igual que si fueran un interruptor (Y de hecho se manejan igual) pero que a partir de un cierto Angulo de inclinación dejan de hacer contacto y abren el contacto.

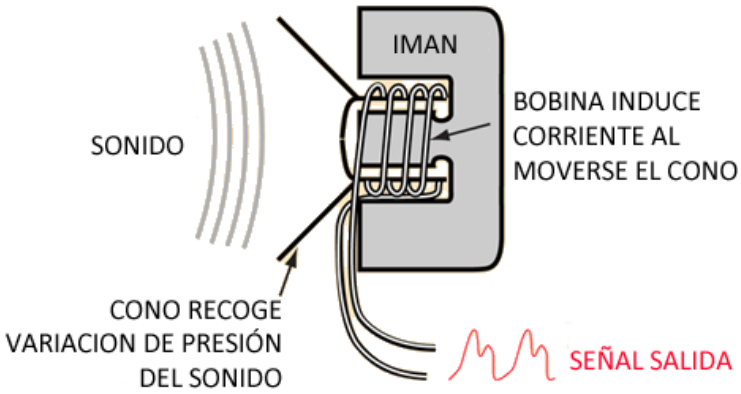
En el SE es el encargado de asegurar que el dispositivo este en la posición correcta y de esta forma asegurar el buen funcionamiento del mismo. Si está mal colocado, evita que el sensor de movimiento funcione hasta que esté bien acomodado.

Código Arduino utilizado:

```
  lecturaTilt = digitalRead(TiltPin);//lee el estado del sensor
    if (lecturaTilt==LOW){
      digitalWrite(LEDPin, HIGH);//enciende el led
      BT1.println(8);//envía por bluetooth la acción que ocurrió
      tiltActivado=1;
    }else{
      digitalWrite(LEDPin, LOW);//apaga el led
      BT1.println(9);//envía por bluetooth la acción que ocurrió
      tiltActivado=0;
}
```

SENSOR SONIDO (KY-038)

Un micrófono es un transductor que convierte las ondas sonoras en señales eléctricas. Podemos conectar un micrófono a un procesador como Arduino para detectar sonidos. La salida producida por un micrófono es una señal eléctrica analógica que representa el sonido recibido. Sin embargo, en general, esta señal demasiado baja para ser medida y tiene que ser amplificada.

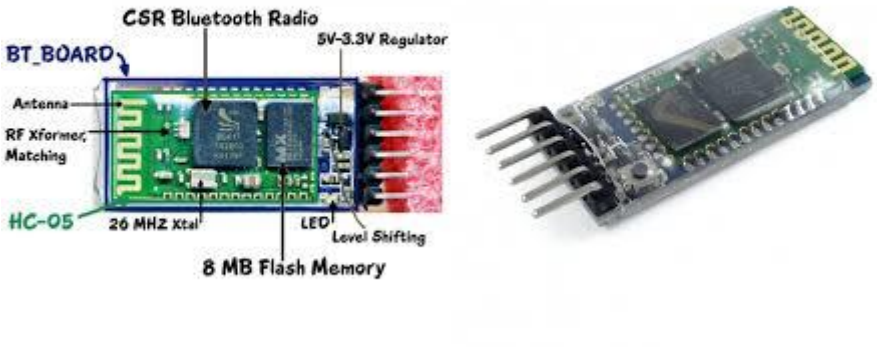


En el SE es el encargado de medir el sonido ambiente, en caso de detectar algún cambio lo informa a la aplicación android para que el usuario esté al tanto. Además enciende un LED y emite una música para tratar de que el bebe se vuelva a dormir.

Código Arduino utilizado:

```
volumen = analogRead(sonidoPin); //Se ha conectado el sensor a la placa por medio de la
entrada A0
  if (volumen > rangoSonido){
    digitalWrite(LEDPin, HIGH) ;//enciende el led
    BT1.println(6);//envía por bluetooth la acción que ocurrió
    beep(200,10);//suena 10 veces, simula un tema
  }else {
    digitalWrite(LEDPin,LOW);//apaga el led
    BT1.println(7);//envía por bluetooth la acción que ocurrió
}
```

Módulo bluetooth HC-05



Se utiliza para conectarse al sistema Android y a través del mismo enviar el estado de los sensores y a su vez recibir configuraciones de la aplicación.

Código Arduino utilizado:

```
//mientras reciba datos del bluetooth, lo va guardando para luego analizarlo
  while (BT1.available()) {
       char c = BT1.read();  //lee el buffer de bluetooth, carácter a carácter
      if (c == '\n') {
        break;  //cuando encuentra el salto de línea, da por finalizada la lectura del string
      }
      //va guardando los caracteres en el string
      cadena += c;
  }
   //si la cadena tiene valores, procede a analizar lo que recibió
  if (cadena.length() >0 ) {
      //si la cadena empieza con T, lo que se recibe es datos de temperatura
      if (cadena[0]=='T'){
        //extrae la temperatura minima que se envía desde el android
        String aux=cadena.substring(1,3);
        //asiga el valor a la temperatura mínima
        tempMin=aux.toInt();
        //extrae la temperatura máxima que se envía desde el android
        aux=cadena.substring(4,6);
        //asiga el valor a la temperatura máximo
        tempMax=aux.toInt();
      }else if (cadena[0]=='1' and encendido==0){//si envía uno y no esta encendido,
enciende el equipo
        encendido=1;//activa la bandera que dice que el sistema esta encendido
        sonido();//calcula el rango del sensor de sonido
        delay(200);
      }else if (cadena[0]=='0' and encendido==1){//si envía 0 y no está apagado, apaga el
equipo
        encendido=0;//activa la bandera que dice que el sistema está apagado
        delay(200);
      }
      cadena="";}
```

Info de los sensores utilizados (ANDROID)

Acelerómetro

El sensor se va a utilizar de la siguiente manera, al moverlo va a encender o apagar el SE.

Luz

El sensor se utilizara para controlar el brillo de la pantalla

Proximidad

Cuando el dedo tape el sensor de proximidad la aplicación va a abrir la pantalla de configuración, para que se puedan setear los rangos de temperatura a controlar.

Diseño (ANDROID)

Introducción

El sistema posee una pantalla principal, que se muestra primeramente al abrir la aplicación en donde se puede encender o apagar el dispositivo, hay una opción para entrar en la pantalla de configuración y además muestra el estado de todos los sensores del SE.
Para encender o apagar el SE, se puede hacer desde la pantalla principal oprimiendo el botón correspondiente a la acción o también con el acelerómetro, moviendo el celular de un lado para otro. EL sistema va a enviar la acción que corresponda al SE.
En la pantalla de configuración se puede setear los rangos de temperatura que debe controlar el sensor LM36. Para ingresar en esta pantalla se puede hacer desde la pantalla principal oprimiendo el botón de configuración o también tapando el sensor de proximidad.
Por último el sensor de luz del celular va a controlar el brillo de la pantalla.

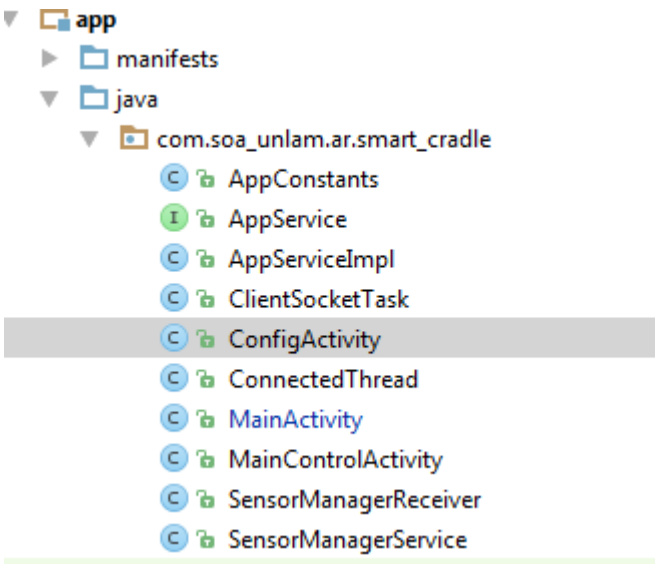
Vista Principal

En esta vista se puede observar en las screenshots que se tiene 3 rectángulos que representan botones de Encender, Apagar y Configuración del arduino. Al oprimir el Botón encender, se deshabilita esta opción y se habilita el de Apagar, mostrando así la sensación de cambio de estado y viceversa. El botón de configuración pasa a la vista de configuración.

Vista Configuración

Una vez llegado a esta vista a partir del botón de configuración inmediatamente se muestran los rangos de temperatura actual que posee el arduino, en caso de querer modificarlos hay que ingresar los correspondientes valores por los campos, "ingresar mínima" e "ingresar máxima". Una vez ingresados se puede actualizar el rango opromiendo Actualizar, en caso de no querer hacerlo, Cancelar, y ambas acciones unas vez realizada las operaciones, retornan a la vista principal.

Arquitectura de la Aplicación



En la arquitectura observamos clases e interfaces, las cuales describimos a continuación:

MainActivity, ConfigActivity: MainActivity maneja los componentes de la vista principal e inicializa los servicios necesarios, ya que es la primer clase que se invoca al iniciar la aplicación. ConfigActivity, se la invoca desde le MainActivity, mediante un botón de acción que llama al ConfigActivity y este muestra datos y componentes en una interfaz visual para la configuración de temperatura.

AppService, AppServiceImpl: Con la clase AppServiceImpl declaramos un singleton que implementa la interfaz AppService, para garantizarnos los accesos a los métodos que queremos mostrar en los distintos servicios. Se utiliza para compartir datos con las distintas partes de la aplicación, como los Threads, y servicios asincrónicos, que en algunos casos los cambios de algunas clases, no son visibles dentro del subproceso y no se tiene accesos desde las distitintas partes. Además usa algunas properties compartidas entre subprocesos para poder sincronizar, ya que dependiendo del proceso, puede entrar en condición de carrera y nunca ejecutar algún otro.

ConnectedThread: Una vez iniciada y realizada la conexión por bluetooth, este subproceso recibe por parámetro en el constructor el socket, del cual se obtiene el canal de transmisión y el canal de recepción. Una vez iniciado este thread, administra el envío de datos al arduino y la recepción. De la recepción de datos, invoca a un handler que está declarado en el main activity, y esté se encarga de informar el correspondiente cambio sobre la vista.

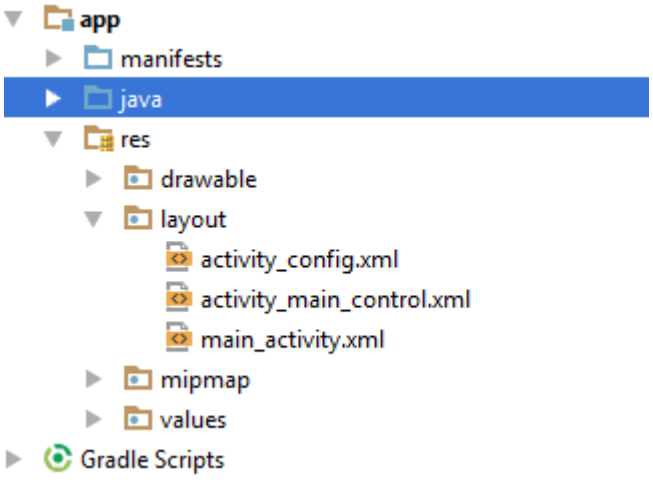
SensorManagerReceiver, SensorManagerService: SensorManagerService extiende de IntentService que es una clase proporcionada por Android para hacer que un servicio se vuelva subproceso y se ejecute en segundo plano, y a su vez implementa SensorEventListener que le da funcionamiento como manejador de cambios en los sensores internos del dispositivo android, obligando a implementar métodos que son llamados cuando ocurre algún evento en los sensores. Una vez determinado que sensor fue el que se modificó de acuerdo a ciertas reglas prestablecidas, se invoca al SensorManagerReceiver, que administrado por el MainActivity, y es un tipo de handler, que interpretará el mensaje de los enviado desde el servicio y realizará una acción específica: si es un Shake (Logrado con el

acelerómetro), se prende u apaga el dispositivo dependiendo del estado en el que se encontraba; si el estado del sensor de proximidad detecta una cercanía, entonces, cambia a configuración; si cambia la luz ambiente cambia el brillo de pantalla.

AppConstants: En esta clase se declaran las constantes generales de la aplicación, utilizadas entre los distintos subprocesos, como los códigos de los sensores enviados por arduino, estado en el que se encuentra el arduino (prendido u apagado), estados en los que se encuentran los sensores del dispositivo android.

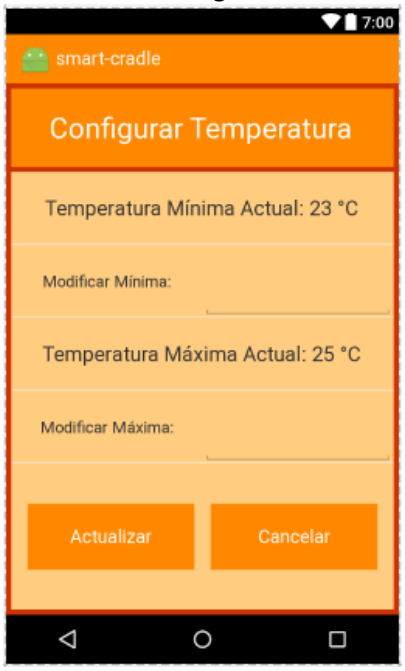
MainControlActivity, ClientSocketTask: Estas clases controlarían la conexión con un servidor remoto para una conexión por socket y protocolo http, actualmente en desuso.

Archivos de Configuración Visual



main_activity.xml: este archivo contiene el diseño de la vista, moldeado con las herramientas que proporciona android, a este xml es el que tiene acceso MainActivity. Java para controlarlo.

activity_config.xml: este archivo es controlado por ConfigActivity.java, y es para el diseño visual de la actividad de configuración de temperatura.



activity_main_control.xml: diseñado para configuración por internet, actualmente no se utiliza este vista dentro de la aplicación.