El objetivo principal del proyecto es diseñar una casa inteligente, que cuente con un sistema capaz de gestionar los entornos básicos de la vivienda por medio de una aplicación Android.

DomoticApp

Casa inteligente con Arduino y Android

Miguel Rolle Ortega

Resumen

En esta memoria se han descrito los conocimientos básicos para entender qué es y cómo funciona un sistema domótico, y cómo, utilizando el hardware libre de Arduino, se puede crear un sistema estable con un presupuesto muy inferior al de las viviendas de alta categoría.

La memoria ha sido dividida en fases en las que se va a tratar cada tema de una forma amplia pero simple, es decir, se dará la información necesaria para entender el proceso de creación de un sistema.

En la introducción se va a poder dar un primer paso en el mundo de la domótica y conoceremos el porqué de la utilización de Arduino.

Después, aprenderemos cómo están construidas las placas Arduino y el entorno de trabajo que disponemos para programarla.

Es importante comentar también los distintos dispositivos que podemos acoplar a una placa para añadir funcionalidad al sistema domótico.

Para la parte de gestión del sistema se ha decidido utilizar un dispositivo móvil con sistema operativo Android, junto con un servidor conectado entre ellos mediante Internet.

El sistema que he implementado, permite controlar algunas funciones de una vivienda y hacer la vida cotidiana más cómoda.

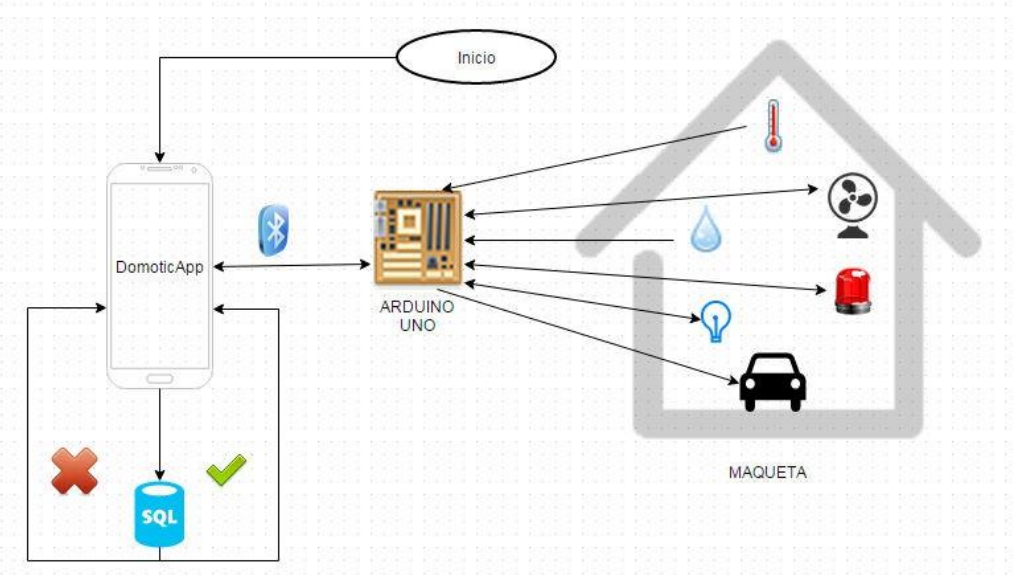


Tabla de Contenidos

Introducción

¿Qué es la domótica? Se podría definir como el conjunto de tecnologías aplicadas al control y la automatización inteligente de la vivienda, que permite una gestión eficiente del uso de la energía además de aportar seguridad, confort, y comunicación entre el usuario y el sistema. Para poder conseguir las propiedades comentadas anteriormente es necesario que los sistemas recojan la información de su entorno con sensores y dispongan de la lógica para actuar en consecuencia utilizando actuadores.

Actualmente, los sistemas domóticos tienen un precio de instalación muy alto, con lo cual solo es posible verlo en casas de lujo. Éstos suelen utilizar buses de transmisión de información que posibilitan una domótica robusta como son el EIB, X10, CEBus, LonWorks/LongTalk y ZigBee.
Una alternativa más barata y casera consiste en la utilización de placas de Arduino.

En este proyecto, se utilizará la plataforma Arduino, que a su vez se apoyará en otros dispositivos, para poder construir un sistema domótico simple con el cual se pretende controlar una vivienda a través de un dispositivo móvil y hacer la vida cotidiana más cómoda.

Arduino es una plataforma de hardware libre creada en 2005, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares.

Para crear el sistema domótico han de tenerse en cuenta varios aspectos. Hay que conocer el capital del que disponemos para invertir en el sistema y seleccionar los dispositivos que más se ajusten a nuestras necesidades. De poco sirve comprar un elemento con grandes prestaciones si luego no se va a aprovechar. Por ejemplo, en vez de utilizar como servidor del sistema un PC dedicado y muy simple que costase unos 150€ podríamos usar otras alternativas. Este año han aparecido alternativas muy baratas como la placa computadora Raspberry Pi que se puede obtener por unos 30€.

Como objetivo de este proyecto, nos propusimos crear un sistema domótico simple utilizando las placas de bajo coste Arduino y otros dispositivos, como sensores, actuadores y comunicadores. Es necesario dotar al sistema de la lógica necesaria para que puedan comunicarse las placas que estarán controlando las habitaciones en las cuales han sido instaladas.

Con los conocimientos adquiridos en el ciclo formativo de DAM, hemos podido desarrollar una parte importante de la aplicación móvil para Android que controla el sistema domótico de la vivienda, hemos podido programar las ordenes que se envían para el control de todos y cada uno de los sensores y por último, hemos creado la base de datos que guarda el estado de los dispositivos de control y de los usuarios.

El motivo para desarrollar este proyecto ha sido que algunos de nosotros contábamos de antemano con algunas nociones de electrónica y nos pareció un buen reto para una fusión de conocimientos. En cuanto a mis expectativas, pensaba que el diseño de la maqueta y el trabajo manual iba a ser lo más fácil, pero resultó ser más complicado.

Alcance Funcional del sistema

El principal objetivo del sistema es el de tener una vivienda automatizada, todo ello a través del móvil y estando dentro del hogar. La vivienda dispone de tres habitaciones incluyendo el salón, cada una de las habitaciones dispone de luces que pueden ser controladas desde el teléfono móvil.

La vivienda, a su vez, dispone de varios sensores, entre ellos, está el de temperatura y humedad el cual está comprobando permanentemente estos datos que podremos visualizar desde el teléfono móvil.

Este sensor entra en juego cuando queramos cambiar la temperatura del ambiente, ya que si queremos una temperatura inferior podemos indicarlo desde el móvil y esto hará que se active un ventilador que refrescara el ambiente del hogar.

El sensor de humos, que al igual que el sensor de temperatura y humedad, está comprobando el nivel de Co2 en el ambiente, en cuanto detecta una concentración superior a 500 ppm (partículas por millón), hace saltar una alarma, una luz roja de emergencia y todo esto a su vez activa un extractor de humos. Esta alarma se desactiva desde el teléfono móvil con una contraseña de seguridad.

Por último, la vivienda dispone de un servomotor que abrirá o cerrará la puerta del garaje, la apertura y cierre de la puerta también se indica a través del teléfono móvil.

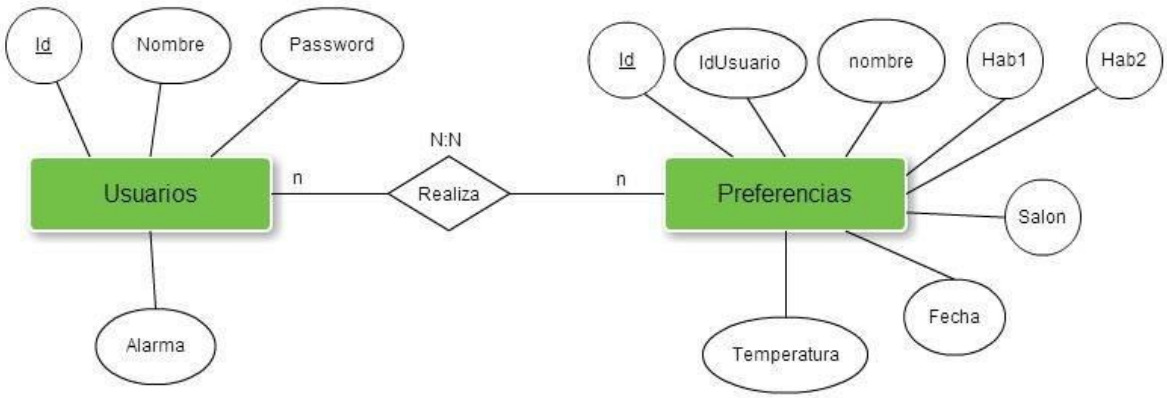
Cabe destacar que para tener acceso desde el móvil a todas las funcionalidades de la vivienda, previamente primero tenemos que ser estar logados en la aplicación.

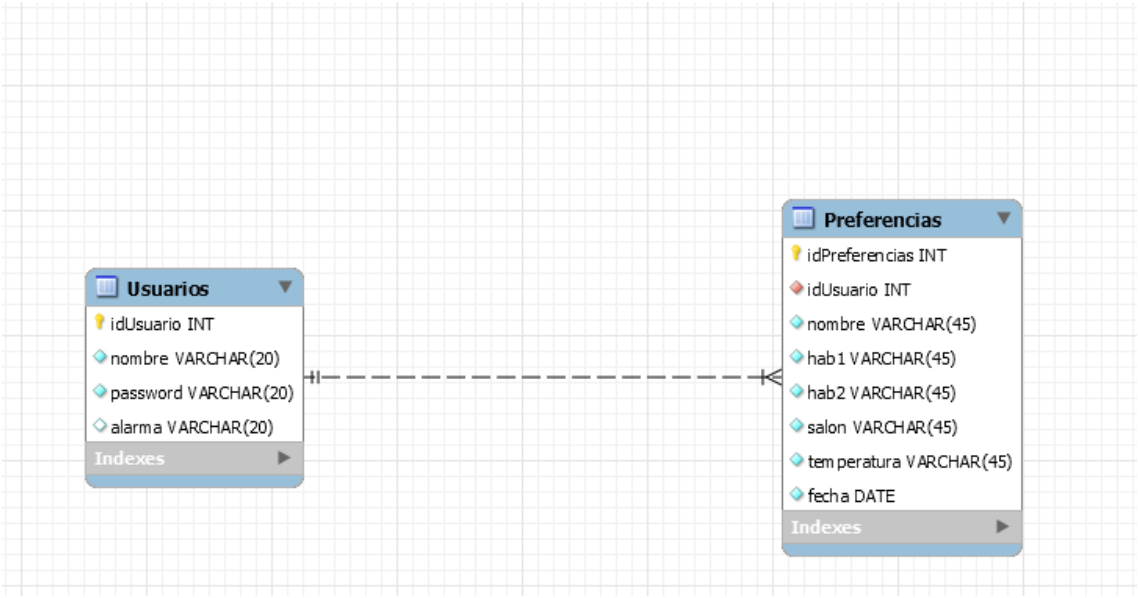
La aplicación dispone de un sistema de Log que nos permite comprobar el último usuario que usó la aplicación, además del estado de nuestra vivienda, es decir, que luces están encendidas y cuales están apagadas, la temperatura, etc.

Diseño Técnico

Esquema de base de datos:

Entidad - Relación





- Primary key
- foreign key

Significado de cada uno de los campos:

Tabla usuarios:

- idUsuario: Es un campo numérico autoincremental.
- Nombre: Campo que corresponde con el nombre de usuario.
- Password: Contraseña del usuario.
- Alarma: Contraseña para desactivar la alarma.

Tabla preferencias:

- idPreferencias: Campo autoincremental de tipo numérico.
- IdUsuario: Corresponde con el del usuario como clave foránea.
- Nombre: Campo que corresponde con el nombre de usuario.
- Hab1: Almacena el estado de la luz de la primera habitación.
- Hab2: Almacena el estado de la luz de la segunda habitación.
- Salón: Almacena el estado de la luz del salón.
- Temperatura: Almacena la temperatura de la vivienda.
- Fecha: Almacena la fecha de la última conexión.

Tecnologías utilizadas

Hemos utilizado Arduino, ya que es una tecnología que está siendo muy utilizada últimamente para el desarrollo de todo tipo de proyectos, como domótica, robótica, etc. También hemos optado por utilizar Android porque es la más usada actualmente en la mayoría de los dispositivos móviles.

Ambas tecnologías son compatibles con el desarrollo del proyecto y entre sí, personalmente pienso que esto será el futuro ya que cada vez usamos más los dispositivos móviles con aplicaciones personalizadas para el día a día, que nos ayudan a facilitarnos el nivel de vida, a ahorrar tiempo, y a mejorar nuestra comodidad.

Presupuesto

Para el desarrollo de este proyecto, todo el software utilizado fue Open Source, sin embargo no se contaba con ninguno de los materiales para la parte del hardware.

Teníamos pensado que con 100 euros bastaría para todo lo necesario, pero después de visitar varias tiendas de electrónica y ver los precios, nos percatamos de que ese presupuesto se quedaba corto.

A continuación se detalla el precio del material usado:

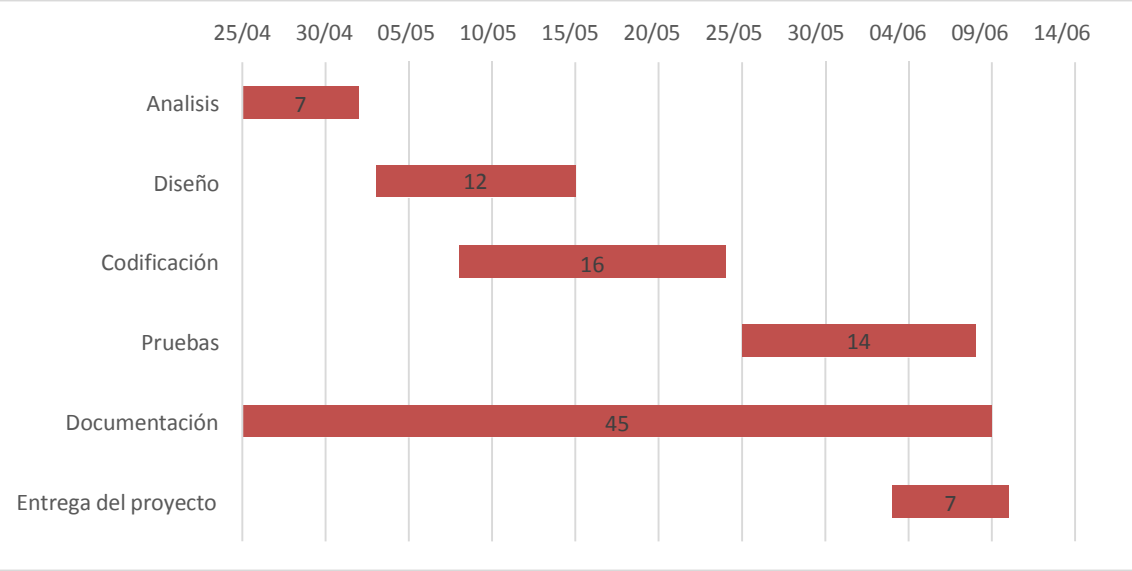
UNIDAD	MATERIAL	PRECIO/UNIDAD	PRECIO TOTAL
1	Placa de Arduino	27,49	27,49
1	Tableros de madera	15,95	15,95
10	Escuadras de ángulo	1,70	17,00
1	Sensor humedad y temperatura	11,73	11,73
1	Sensor temperatura	1,65	1,65
1	Zumbador electromecanico	1,57	1,57
2	Ventilador	2,72	5,44
1	Protoboard	8,14	8,14
1	Módulo Bluetooth	13,99	13,99
1	Sensor humo	4,50	4,50

	1	Módulo 4 reles	11,67	11,67
	1	Servo Motor	12,90	12,90
varios		Cables	4,95	4,95
	1	Polimetro	18,87	18,87
	7	Leds	0,87	6,09
	1	Soldador	14,05	14,05
	1	Estaño	4,60	4,60
varios		Otros	18,65	18,65
TOTAL SIN IVA				**199,24**
IVA				**21%**
TOTAL IVA INCLUIDO				**241,08**

En cuanto al coste humano, se estima que ha sido de aproximadamente 1000 euros, ya que han sido dos meses de dedicación durante casi todos los días, sacrificando fines de semana y tiempo libre.

Planificación

ACTIVIDAD	INICIO	DURACIÓN (DIAS)	FIN
Análisis	25/04/2015	7	02/05/2015
Diseño	03/05/2015	12	15/05/2015
Codificación	08/05/2015	16	24/05/2015
Pruebas	25/05/2015	14	08/06/2015
Documentación	25/04/2015	45	09/06/2015
Entrega del proyecto	03/06/2015	7	10/06/2015



Características básicas de Arduino

En este apartado vamos a describir los principales elementos que componen una placa Arduino y el entorno de desarrollo en el que se programa el código, es decir, la parte hardware y software que actúan sobre Arduino.

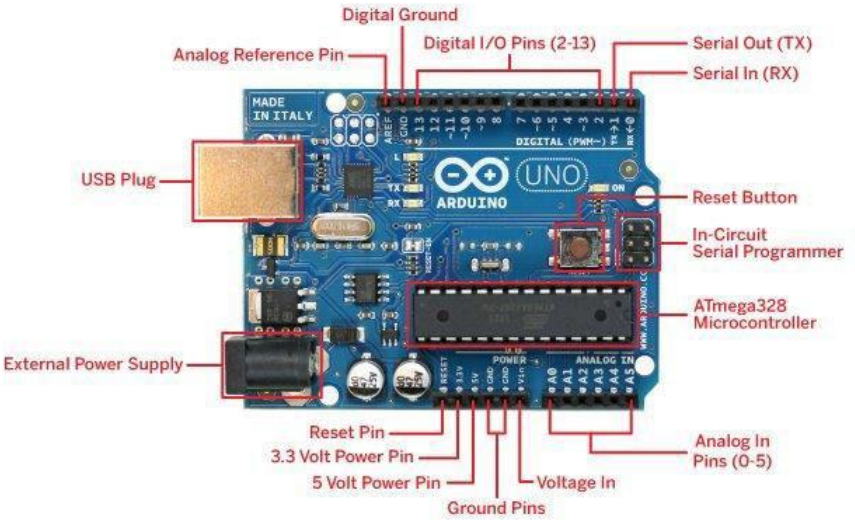
Hardware

Al ser Arduino una plataforma de hardware libre, tanto su diseño como su distribución puede utilizarse libremente para el desarrollo de cualquier tipo de proyecto sin haber adquirido ninguna licencia. Por eso existen varios tipos de placa oficiales, las creadas por la comunidad Arduino o las no oficiales creadas por terceros pero con características similares. En la placa Arduino es donde conectaremos los sensores, actuadores y otros elementos necesarios para comunicarnos con el sistema.

En el proyecto se ha utilizado una placa Arduino Uno que describiremos a continuación:

Arduino Uno

Estamos ante el último modelo diseñado y distribuido por la comunidad Arduino. Tiene un tamaño de 75x53mm. Su unidad de procesamiento consiste en un microcontrolador ATmega328 que puede ser alimentada mediante USB o alimentación externa, y contiene pines tanto analógicos como digitales.



Referencia para pines analógicos (AREF)
Tensión de referencia para entradas analógicas

Pines de tierra (GND)
Masa del circuito para pines, es decir, es la tensión de referencia de 0V.

Pines digitales de entrada y salida
En estos pines conectaremos la patilla de dato del sensor/actuador. Desde ellos podremos leer la información del sensor o activar el actuador.

Ciertos pines son reservados para determinados usos:

- Serie: 0(RX) y 1(TX). Utilizados para recibir (RX) y trasmitir (TX) datos serie. Utilizando estos pines podremos conectarnos con otras placas.

- Interrupciones externas: 2 y 3. Estos pines pueden ser configurados para activar interrupciones.

- SPI: 10-13. Estos pines soportan la librería de comunicación de dispositivos SPI.

- LED: 13. Este pin está conectado con un led de la placa. Cuando se le asigne un valor HIGH se encenderá. En cambio, si lo situamos en LOW, se apagará.

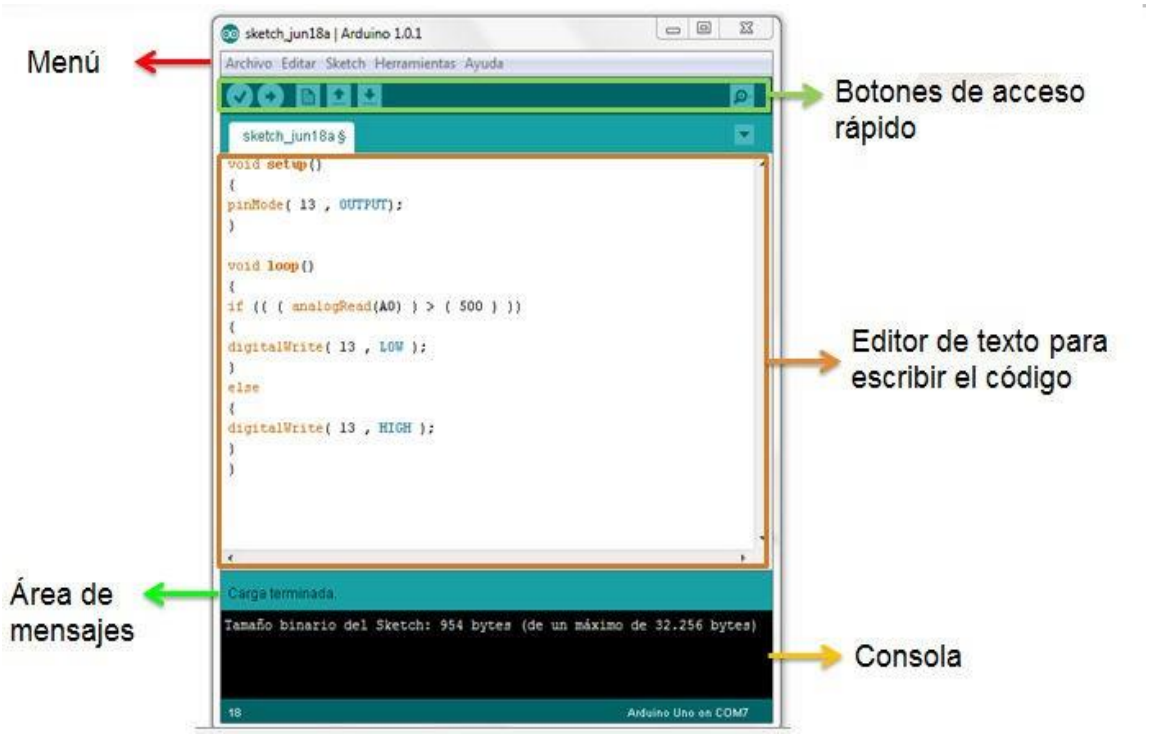
Conector USB
La placa se puede alimentar con la tensión de 5V que le proporciona el bus serie USB. Cuando carguemos un programa a la placa desde el software de Arduino se inyectará el código del ordenador por este bus.

Botón Reset
Utilizando este botón podremos reiniciar la ejecución del código del microcontrolador.

Software

La plataforma Arduino tiene un lenguaje propio que está basado en C/C++ y por ello soporta las funciones del estándar C y algunas de C++. Sin embargo, es posible utilizar otros lenguajes de programación y aplicaciones populares en Arduino como Java, Processing, Python, Mathematica, Matlab, Perl, Visual Basic, etc.

Esto es posible debido a que Arduino se comunica mediante la transmisión de datos en formato serie, que es algo que la mayoría de los lenguajes anteriormente citados soportan. Para los que no soportan el formato serie de forma nativa es posible utilizar software intermediario que traduzca los mensajes enviados por ambas partes para permitir una comunicación fluida. Es bastante interesante tener la posibilidad de interactuar con Arduino mediante esta gran variedad de sistemas y lenguajes, puesto que dependiendo de cuales sean las necesidades del problema que vamos a resolver podremos aprovecharnos de la gran compatibilidad de comunicación que ofrece.

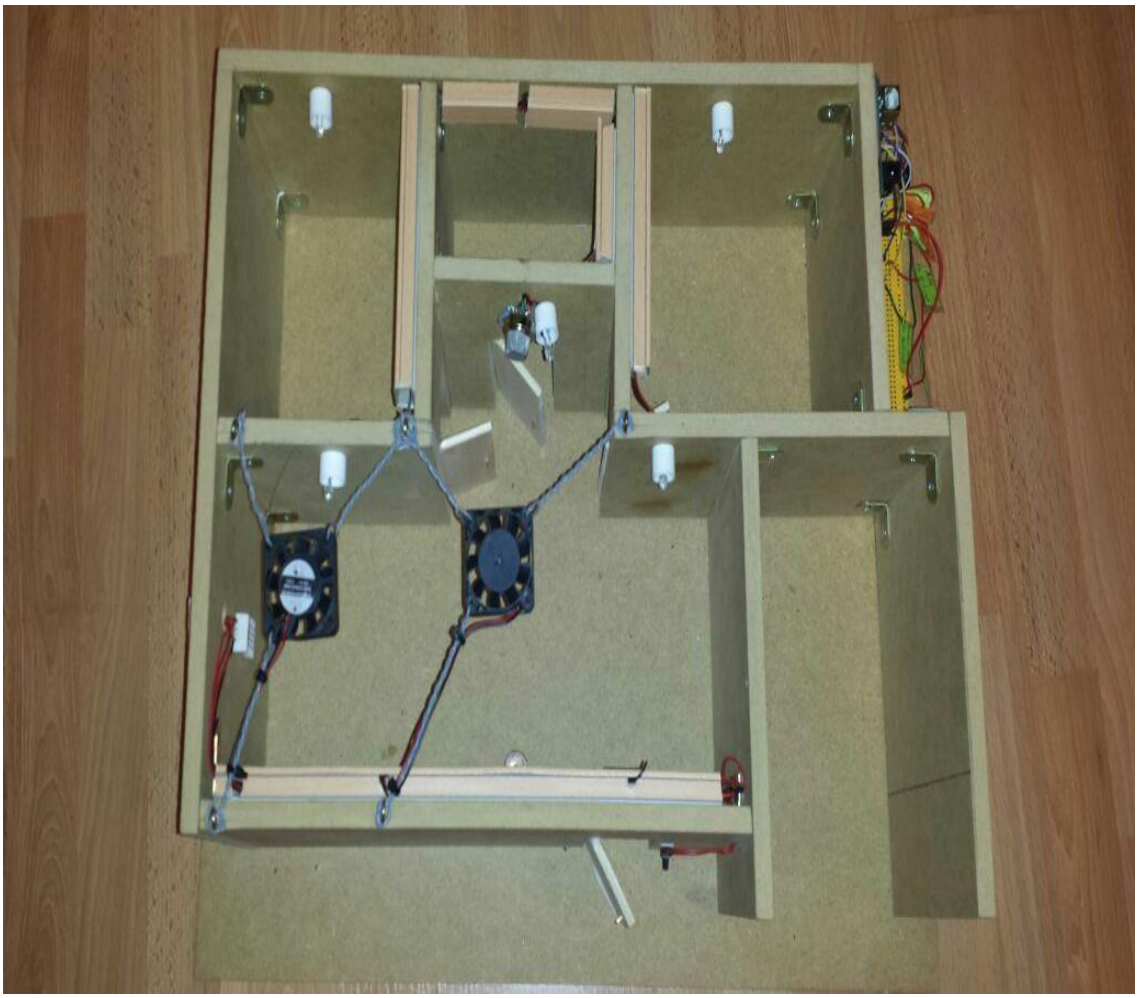


Maqueta

La maqueta está diseñada como una vivienda familiar de 3 habitaciones, un baño, salón con cocina americana, un garaje con una única plaza y un pequeño jardín.

Los materiales que se han utilizado para la realización de esta maqueta han sido:

- Tableros de madera MDF 800x400x10 mm.
- Escuadras de ángulo canto redondo 20X20 mm.
- Tornillos.

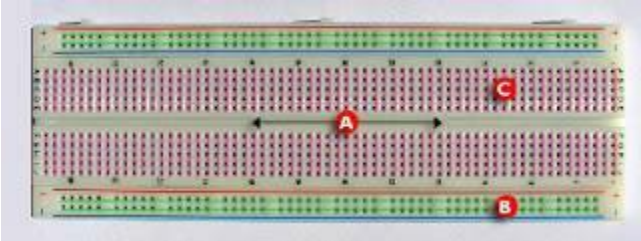


Arduino

Protoboard

Es la pieza que conecta todos los dispositivos de la casa a Arduino. Es una especie de tablero con orificios, en la cual se pueden insertar componentes electrónicos y cables para armar circuitos.

Estructura del protoboard: Básicamente un protoboard se divide en tres regiones:



A) Canal central: Es la región localizada en el medio, se utiliza para colocar los circuitos integrados.

B) Buses: Se representan por las líneas rojas (buses positivos o de voltaje) y azules (buses negativos o de tierra) y conducen de acuerdo a estas, no existe conexión física entre ellas.

C) Pistas: La pistas se localizan en la parte central del **protoboard**, se representan y conducen según las líneas rosas.

Comunicación con la aplicación

La conexión se establece con el módulo de Bluetooth de Arduino, es un estándar de comunicación inalámbrica que permite la transmisión de datos a través de radiofrecuencia en la banda de 2,4 GHz. Estos módulos contienen el chip con una placa de desarrollo con los pins necesarios para la comunicación serie.

A la hora de configurar el módulo Bluetooth utilizamos los comandos AT, son un tipo de comandos que sirven para configurar, a través de un microcontrolador, un ordenador o con cualquier dispositivo que posea una comunicación serie (Tx/Rx).

Son unas instrucciones que nos permiten cambiar los baudios del módulo, el PIN, el nombre, etc. Para usar los comandos AT, el módulo Bluetooth no debe estar vinculado a ningún dispositivo (led rojo del módulo parpadeando).

Según las especificaciones del módulo, el tiempo que se tiene que respetar entre el envío de un comando AT y otro tiene que ser de 1 segundo. Si se envía un comando AT y en menos de un segundo se envía otro, el módulo no devuelve respuesta.

Código de configuración:

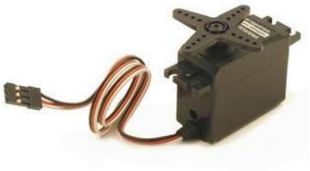
```
//Importa las librerías del sensor de temperatura-humedad y el bluetooth.
#include "DHT.h"
#include<SoftwareSerial.h>

//define el tipo de sensor de temperatura-humedad y asigna un pin en el arduino.
#define DHTTYPE DHT22
#define DHTPIN 3
DHT dht (DHTPIN, DHTTYPE);

//Asocia los pines 4 y 2 del arduino a los pines TX y RX del bluetooth.
SoftwareSerial BT1 (4,2);

//Asociación de pines de arduino a los sensores
int alarmaExtractor = 11;
int sensorHumos = A0;
int hab1 = 8;
int hab2 = 9;
int salon = 10;
int fan = 5;

//Almacena las lecturas del sensor de humos.
int sensorValue = 0;

//Variables varias
int temp = 30;
int swTono = 0;
char c;

//variables de humedad y temperatura
float h,t;
char dato;

//Inicialización de pines y puertos.
void setup() {

  //Inicializa el puerto serie a 9600 baudios.
  Serial.begin(9600);

  //Inicializa el puerto serie del bluetooth a 9600 baudios.
  BT1.begin(9600);

  //Define los pines del arduino con dos parametros, el primero es el número
 //del pin y el segundo representa si el terminal es de entrada o salida, en este caso es salida
  pinMode(fan,OUTPUT);
  pinMode(hab1,OUTPUT);
  pinMode(hab2,OUTPUT);
  pinMode(salon,OUTPUT);
  pinMode(alarmaExtractor,OUTPUT);
  //Inicializa el sensor de humedad-temperatura.
  dht.begin();
}
```

Partes de gestión autónoma

Son los dispositivos controlados por Arduino, sin poder modificar sus valores desde la aplicación, es decir, solo realizan acciones prediseñadas, ya se han devolver una información o realizar un acción mecánica.

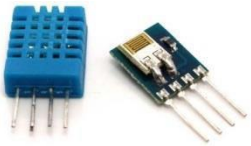
Servo Motor

Solo se podrá hacer petición para girar el motor de izquierda a derecha o al revés.



Sensor de Humedad y Temperatura

La aplicación solo podrá solicitar a Arduino que le muestre la humedad o temperatura.



Partes de gestión con la aplicación

La aplicación puede modificar sus valores, para cambiar el comportamiento de los dispositivos.

Leds

Se podrá modificar los valores de los leds pasándolos de on/off o de off/on.

Alarma de Humos

Avisara por medio acústico cuando se active, teniendo que desactivarla, también podrá ser modificado el valor máximo de humos para que salte la alarma.

Ventiladores

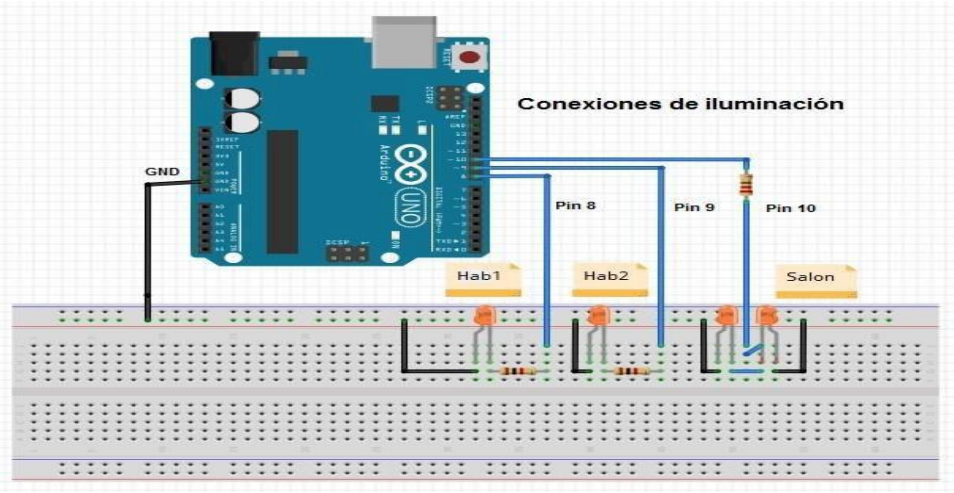
Se podrá asignar una temperatura para que se pongan en funcionamiento cuando sobrepasado ese parámetro.

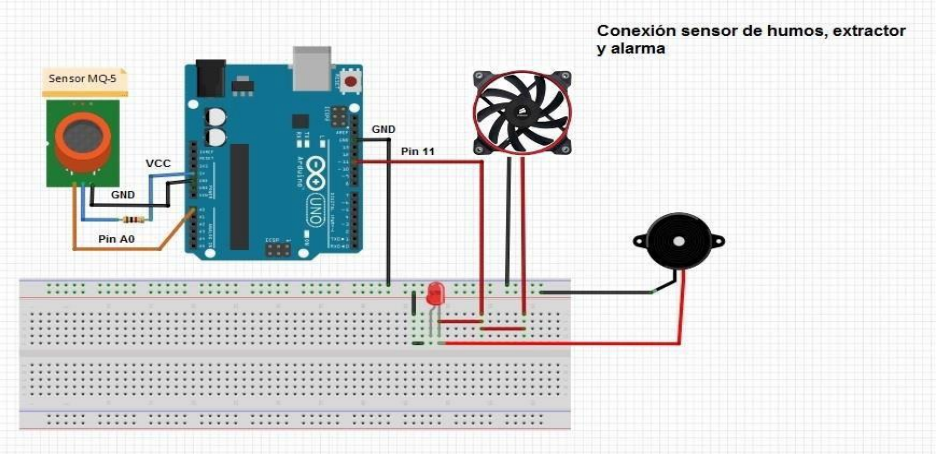
Funcionamiento:

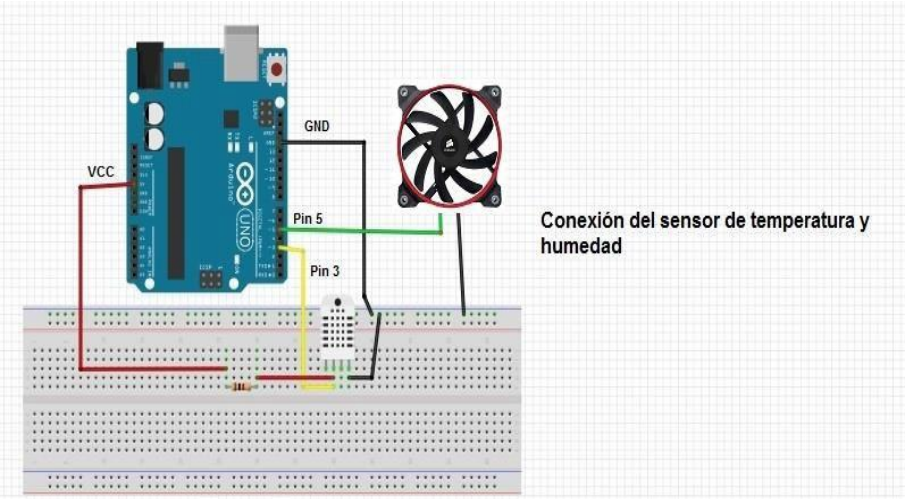
```
void loop() {
  //Metodos que leen la temperatura y la humedad y la envían al puerto serie del bluetooth.
EnviarTemperatura();
EnviarHumedad();
delay(1000);
   //Si el puerto serie del bluetooth está disponible lee lo que contenga el buffer
  //y lo guarda en una variable de tipo char  para después cumplir uno de todos los
  //casos posibles.
   if (BT1.available()){
       c = (char)BT1.read();
       Serial.println(c);
       //Luces
       switch(c){
       case 'a':
          digitalWrite(8,HIGH);
          break;
        case 'b':
          digitalWrite(8,LOW);
          break;
        case 'c':
          digitalWrite(9,HIGH);
          break;
        case 'd':
          digitalWrite(9,LOW);
          break;
        case 'l':
          digitalWrite(10,HIGH);
          break;
        case 'm':
          digitalWrite(10,LOW);
          break;
        case 'e':
          temp = 20;
          break;
        case 'f':
          temp = 22;
          break;
        case 'g':
          temp = 24;
          break;
        case 'h':
          temp = 26;
          break;
        case 'i':
          temp = 28;
          break;
        case 'p':
          EnviarHumedad();
          break;
        case 'j':
          swTono = 0;
          break;
    }
  }
  //fin del if (bluetooth disponible)
```

```
//Lee continuamente los datos que recibe del sensor de humos.
sensorValue = analogRead(sensorHumos);

  //Si la temperatura es superior de 500 (concentración de co2 en ppm que empieza a
  //ser peligrosa para las personas)
  if (sensorValue > 500){
    //Cambia el swich de la alarma y envía a la App un aviso
    swTono = 1;
    BT1.println('k');
  }

 //Tipo de tono que sonara en la alarma,tiene tres parametros (pin de arduino, frecuencia y
duración.)
  if(swTono == 1)
  {
    tone(13, 300, 500);
    delay(1000);
    digitalWrite(alarmaExtractor, HIGH);
    tone(13, 450, 500);
    delay(2000);
    digitalWrite(alarmaExtractor, LOW);
  }

  //Si la temperatura de la casa es superior a la que marca el sensor de temperatura saltara el
ventilador.
 if (t >= temp){
     digitalWrite(fan,HIGH);
   }else {
     digitalWrite(fan,LOW);
   }
}
//Método que lee la temperatura, y la envía al puerto serie del bluetooth.
void EnviarTemperatura(){
  t = dht.readTemperature();
  //Comprueba que el dato es un número.
  if (isnan(t)){
     Serial.print("Error");
  } else {
     BT1.print("c");
     BT1.print(t);
     /*Serial.print(t);
     Serial.print("C ");*/
  }
}
//Metodo que lee la humedad, y la envia al puerto serie del bluetooth.
void EnviarHumedad()
{
  h = dht.readHumidity();
   //Comprueba que el dato es un número
  if (isnan(h)){
    Serial.print("Error");
  } else {
    BT1.print("h");
    BT1.println(h);
  }
}
```

Esquemas de las Conexiones







Aplicación Android

Funcionamiento

Nada más iniciar la aplicación y tras una breve animación de carga, DomoticApp solicita el usuario y contraseña de los miembros con acceso a la vivienda.

Esta primera conexión será verificada conectando a una base de datos alojada un servidor externo través de Internet. Si los campos son correctos, se accede a la vista principal de la aplicación, donde se reflejan de los parámetros y constantes de la casa.

En caso de denegar el acceso, el usuario puede registrarse. Tan solo debe acceder al apartado 'Registrarse' y aportar los datos correspondientes. Aunque no haya sido incluido, se entiende que sólo podrán registrarse aquellos usuarios que dispongan de una clave especial de conexión a la vivienda.

Una vez nos logueamos en la aplicación, se abre un socket de comunicación entre el dispositivo móvil y la placa Arduino Uno, gracias al dispositivo Bluetooth 'My-Blue'. La comunicación, desde ese mismo instante, depende de un flujo de datos InputStream y OutputStream, que permanecerán abiertos para realizar las acciones pertinentes, hasta que se ejecuta el método OnStop(), al salir de la aplicación.

No obstante, y para ahorrar tiempo de respuesta en la conexión en caso de relanzarse de nuevo la aplicación, tanto el socket como los flujos de entrada y salida de datos permanecen en suspensión. Cuando transcurre un cierto tiempo, es el propio sistema operativo el que llama al método OnDestroy(), liberando la memoria del dispositivo.

Diseño y código

Splash Activity

La aplicación se inicia con una animación de tipo splashActivity. Las actividades tipo splashActivity son actividades que se muestran al iniciar la aplicación y tienen una duración determinada preestablecida por el programador. Una vez alcanzado el tiempo establecido se lanzará otra actividad llamada en el código. Suelen ser muy útiles para mostrar el logo o una imagen con el fin de informar al usuario sobre el tipo de aplicación o contenido con el que va a interactuar.

Iniciar splash Activity Manifest:

```
<activity android:name=".Splash">
        <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
</activity>
```

Ejecutar splash:

```
timerTask = new TimerTask(){
        @Override
        public void run() {
            Intent openMainActivity =  new Intent(Splash.this,cLogin.class);
            startActivity(openMainActivity);
        }
};
timer = new Timer();
timer.schedule(timerTask,splashDelay);
```

Se crea un objeto TimerTask con un Hilo, dentro de este se crea un intent con dos valores el primer valor es la actividad actual, y el segundo la actividad que abrirá al finalizar la duración establecida.

Después se crea un objeto timer, donde con su método schedule, se le pasa el timerTask y la duración que durara esta actividad ejecutándose.

Login

La actividad del login está conformada por dos campos y dos botones. El primer campo es el del usuario y el segundo el password. Para poder llevar a cabo el uso de la aplicación se debe haber creado un usuario previamente. Por eso están integrados dos botones, el primero es el de login, es el botón que te da acceso a la aplicación una vez hecho el login correctamente. El segundo botón es el de registro, el cual nos llevaría a otra vista.

Para poder llevar a cabo el sistema de logeo, hemos utilizado los siguientes recursos:

Un servidor donde alojar nuestra base de datos y los ficheros PHP que se comunican con ésta y con la aplicación.

Base de datos MySql, Ficheros PHP e Hilos AsyncTask de Android, estos últimos se ejecutan en segundo plano, para comunicarnos con la base de datos a través de unos ficheros php.

Estos ficheros recogen las variables que provienen de la aplicación mediante el método $_POST. Luego opera con los valores y devuelve un resultado hacia esta. Hemos optado por este tipo de hilos puesto que se ejecutan en segundo plano durante un periodo determinado sin parar el transcurso de la aplicación y además devuelven un resultado a la actividad principal y detienen su ejecución tanto si ha sido positiva como negativa.

En nuestra aplicación enviamos el campo usuario y password con el método $_POST, el fichero php realiza una select con estos valores, si existe devuelve 1, si no existe devuelve 0. El hilo Asynctask recibe 1 ó 0 y dependiendo del caso te deja acceder o te sugiere que introduzcas bien los datos.

El proceso de registro es prácticamente igual que el anterior, para registrarnos utilizamos la clase Asynctask, una vez intentamos registrarnos, le pasamos el campo nombre y password por método $_POST al fichero registro.php, este realiza una select con el campo usuario y password, si existe devuelve 0 a la aplicación, en caso contrario procede a realizar un insert, con los campos recibidos.

La aplicación mientras tanto permanece a la escucha del resultado, en caso de recibir un 1, lanza un mensaje a través del cual comunica que el registro se ha realizado correctamente y ejecutaría la actividad principal, si recibe un 0, lanzaría un mensaje comunicando el error o la existencia de este usuario ya en la base de datos y no daría paso a la siguiente actividad.

Hilo AsyncTask:

```
public class logIn extends AsyncTask<String, Void, Boolean> {
   ProgressDialog dialog = new ProgressDialog(cLogin.this);
   @Override
   protected void onPreExecute() {
      dialog.setMessage("Enviando    datos....");
      dialog.show();
   }
   @Override
   protected Boolean doInBackground(String... urls) {
      DefaultHttpClient client = new DefaultHttpClient();
      HttpEntity httpEntity = null;
      HttpResponse httpResponse = null;
      InputStream is = null;

      /* Creacion de valores a enviar */
      ArrayList<NameValuePair> pairs = new ArrayList<NameValuePair>();
      pairs.add(new    BasicNameValuePair("nombre",    nombre.getText().toString()));
      pairs.add(new    BasicNameValuePair("password",    password.getText().toString()));
      HttpPost httpPost = new HttpPost(urls[0]);
      if (pairs != null) {
         try {
            httpPost.setEntity(new     UrlEncodedFormEntity(pairs));
         } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
            return false;
         }
      }
      try {
         httpResponse = client.execute(httpPost);
      } catch (IOException e) {
         e.printStackTrace();
         return false;
      }
      httpEntity = httpResponse.getEntity();
      try {
         is = httpEntity.getContent();
      } catch (IOException e) {
         Log.i("myApp", "El inputStream no funciona" + e.toString());
      }
```

```
      try {
         BufferedReader reader = new BufferedReader( new InputStreamReader(is));
         String line = null;
         while ((line = reader.readLine()) != null) {
            resultado = line;
         }
         is.close();
         Log.i("myApp", "Resultado: " + resultado);
      } catch (Exception e) {
```

```
            Log.i("myApp", "Error converting result " + e.toString());
        }
        return true;
    }
    protected void onPostExecute(Boolean result) {
        if (result == true) {
            if (resultado.equals("1")){
                i.putExtra("nombre",    nombre.getText().toString());
                i.putExtra("password",    password.getText().toString());
                startActivity(i);
            }else{
                Toast.makeText(cLogin.this, "Se han intorducido mal los datos",
Toast.LENGTH_LONG).show();
            }
        } else {
                Toast.makeText(cLogin.this, "Fallida la conexion con la base de datos",
Toast.LENGTH_LONG).show();
        }
        dialog.dismiss();
    }
}
```

Fichero login.php:

```
<?php
$user = 'arduino';
$password = 'arduino';
$db = 'arduino';
$host = 'localhost';
$con=mysqli_connect("$host","$user","$password","$db");

if (!$con) {
    die("Connection failed: " . mysqli_connect_error());
}

$nombre = $_POST['nombre'];
$password = $_POST['password'];
$result=mysqli_query($con,"SELECT * FROM usuarios WHERE nombre='{$nombre}' AND
    password='{$password}'");
$num_rows = mysqli_num_rows($result);

if ($num_rows > 0){
   echo "1";
}else{
   echo "0";
}  ?>
```

Actividad Principal

La actividad principal está compuesta de varias funcionalidades tanto en el frontend como en el backend.

El frontend está basado en el sistema de comunicación entre fragments, los cuales mediante un fragment layout, se van intercambiando sin tener que volver a abrir otra activity.

La ventaja de utilizar fragments es la velocidad de intercambio de comunicación entre estos frente al primer método que apareció en Android de comunicación entre actividades. En nuestro caso al tener un menú con diversas funcionalidades para enviar a la placa de Arduino, hemos decidido utilizar fragmentos, puesto que si quieres ir realizando varias funciones si hubiésemos utilizado actividades, el proceso de abrir una actividad, cerrarla y luego abrir otra, sería muy lento y bastante costoso en cuanto a términos de batería.

En cuanto al diseño es bastante más agradable el intercambio de fragments mediante efectos de transición que el cambio de actividades que primero apaga por completo en la que te encuentras y después muestra la actividad llamada.

El diseño de las pantallas con los diferentes fragmentos es:

El backend está compuesto de varias fases:

La primera fase y la más importante es la conexión entre el dispositivo y la placa de Arduino a través del bluetooth. Para explicar el proceso hay que ir paso a paso en los ciclos de vida de la actividad principal.

OnCreate: Es el método que ejecuta el layout e inicializa tanto las variables como los contenidos a maneja el layout.

En nuestro caso, debemos inicializar un objeto BluetoothAdapter, el cual mediante su método getDefaultAdapter() recibimos el bluetooth del dispositivo en el que se está ejecutando la aplicación.

Una vez tenemos nuestro btAdapter, podemos realizar la conexión con nuestro Arduino, pero antes debemos comprobar si está conectado el bluetooth o no. Si no está conectado debemos lanzar una actividad para poder encenderlo.

```
btAdapter = BluetoothAdapter.getDefaultAdapter();  // Creamos adaptador del bluetooth
comprobarConexionBluetooth();

// Comprobar conexion del bluetooth
private void comprobarConexionBluetooth() {
   // Check for Bluetooth support and then check to make sure it is turned on
   // Emulator doesn't support Bluetooth and will return null
   if(btAdapter==null) {
      errorExit("Fatal Error", "Bluetooth not support");
   } else {
      if (btAdapter.isEnabled()) {
         Log.d(TAG, "...Bluetooth ON...");
      } else {
         //Prompt user to turn on Bluetooth
         Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
         startActivityForResult(enableBtIntent,    1);
      }
   }
}
```

OnResume: En este ciclo se pueden realizar inicializaciones pero a diferencia de onCreate este se ejecuta cada vez que inicias la aplicación,

Una vez se ejecuta el método onResume procedemos a la conexión, de la cual dependemos de varios procesos. El primero es que tenemos que estar conectados previamente al Arduino

mediante nuestro bluetooth incorporado en el dispositivo, después necesitamos un objeto ‘ device’, el cual lo obtenemos de la siguiente forma:

Creamos objeto Bluetooth device, este será igual a nuestro btAdapter.getRemoteDevice(Direccion), la dirección debemos conocerla previamente, existen varias aplicaciones que te dan la dirección de los dispositivos visibles a tu alcance o los propios a los que estas conectados.

Una vez tenemos el objeto device que en este caso ya es el Arduino, tenemos que crear un Bluetooth Socket, al cual le pasamos el device (Arduino).

Cuando tenemos el Socket utilizamos su método Socket.Connect() con un try y un catch, si no se interrumpe se conectará si no lanzara un error.

Conexión:

```
public void onResume() {
    super.onResume();
    BluetoothDevice device = btAdapter.getRemoteDevice(address);
      try {
        btSocket = createBluetoothSocket(device);
    } catch (IOException e) {
        errorExit("Fatal Error", "En onResume() y socket creacion fallida: " + e.getMessage());
    }
    btAdapter.cancelDiscovery();
    try {
        btSocket.connect();
        Log.i(TAG, "....Conexión ok...");
    } catch (IOException e) {
        try {
            btSocket.close();
        } catch (IOException e2) {
            errorExit("Fatal Error", "In onResume() and unable to close socket during connection
   failure" +  e2.getMessage() + ".");
        }
    }

    Log.i(TAG, "...Create Socket...");
    if(btSocket != null){
        mConnectedThread = new ConnectedThread(btSocket);
        mConnectedThread.start();
    }
}
```

Conectado:

Una vez conectado a la placa de Arduino, necesitamos establecer un proceso de comunicación entre el dispositivo y esta. Para poder llevarlo a cabo necesitamos de un hilo de ejecución. Es decir mientras que exista una conexión ambas partes deben estar en proceso de escucha constante, para recibir o enviar información. Por lo tanto creamos una clase ConnectedThread que extiende de Thread, a esta le pasamos el socket para que pueda realizar las operaciones de envío y escucha, y mediante InputStream y OutputStream realizamos estos procesos de la siguiente manera:

```
public class ConnectedThread extends Thread {
    private final InputStream mmInStream;
    private final OutputStream mmOutStream;
    public ConnectedThread(BluetoothSocket socket) {
        InputStream tmpIn = null;
        OutputStream tmpOut = null;

        try {
            tmpIn = socket.getInputStream();
            tmpOut = socket.getOutputStream();
        } catch (IOException e) { }
        mmInStream = tmpIn;
        mmOutStream = tmpOut;
    }
    public void run() {
        byte[] buffer = new byte[1024];  /
        int bytes = 0;
        int contador = 0;
        // Continua escuchando al inputstream hasta que ocurra una exception
            while (true) try {
                // Read from the InputStream
                    bytes = mmInStream.read(buffer);
                h.obtainMessage(RECEIVE_MESSAGE, bytes, -1, buffer).sendToTarget();
            } catch (IOException e) {
                break;
            }
    }

    public void escribir(String message) {
        Log.d(TAG, "...Data to send: " + message + "...");
        byte[] msgBuffer = message.getBytes();
        try {
            mmOutStream.write(msgBuffer);
        } catch (IOException e) {
            Log.d(TAG, "...Error data send: " + e.getMessage() + "...");
        }
    }
}
```

Esta clase, se ejecutara con su metodo .start(); desde la actividad principal una vez que la conexión del dispositivo y la placa se haya realizado satisfactoriamente. Se podría ejecutar de forma manual en un momento específico, pero en nuestro caso se realiza de esta forma.

Proceso de escucha

El proceso de escucha se realiza en el metodo run() de la clase ConnectedThread, pero la parte más importante se realiza una vez recibidos los datos. Estos se almacenan en un buffer mediante bytes, y se envían mediante un objeto Handler a la actividad principal, la cual tiene el objeto Handler declarado.

Run():

```
byte[] buffer = new byte[1024];  // buffer store for the stream
int bytes = 0; // bytes returned from read()
int contador = 0;
// Continua escuchando al inputstream hasta que ocurra una exception
   while (true) try {
      // Read from the InputStream
         bytes = mmInStream.read(buffer);
     h.obtainMessage(RECEIVE_MESSAGE, bytes, -1, buffer).sendToTarget();
   } catch (IOException e) {
      break;
   }

 Recibo los bytes en la actividad principal:

  h = new Handler() {
   public void handleMessage(android.os.Message msg) {
      switch (msg.what) {
         case RECEIVE_MESSAGE:
            byte[] readBuf = (byte[]) msg.obj;
            String strIncom = new String(readBuf, 0, msg.arg1);
            sb.append(strIncom);
            int endOfLineIndex = sb.indexOf("\r\n");
            if (endOfLineIndex > 0) {
               String sbprint = sb.substring(0, endOfLineIndex);
               sb.delete(0, sb.length());
               if (sbprint.equals("k")){
                  getFragmentManager().beginTransaction().add(R.id.fragmento,
                        alarmaFragment).commit();
               }
```

```
                try{
                    if(sbprint.length() > 1){
                        if (sbprint.contains("c")){
                            String temp = sbprint.substring(1,6);
                            temperatura.setText(temp);
                        }
                        if (sbprint.contains("h")){
                            String hum = sbprint.substring(7);
                            Log.i(TAG, "Humedad QUE LLEGA: " + hum);
                            bundle.putString("humedad", hum);
                            humedadFragment.setArguments(bundle);
                        }
                    }else{
                        Log.i(TAG,"LLega nulo");
                    }
                }catch(Exception e){
                }
            }
            //Log.d(TAG, "...String:"+ sb.toString() +  "Byte:" + msg.arg1 + "...");
            break;
        }
    }
};
```

Para poder recibir los datos realizamos un String Builder y va construyendo el String según va leyendo bytes hasta que encuentra un salto de línea. Después manejamos los datos y los tratamos en diferentes partes de la aplicación. En nuestro caso cuando recibimos una letra de tipo char, concretamente la letra “k” lanzamos el fragmento de la alarma, puesto que esto supone que ha saltado la alarma y debemos apagarla.

Proceso de envío

El proceso de envío se lleva a cabo a través del método enviar integrado en la clase ConnectedThread. Este recibe una cadena, la pasa a bytes, y la envía a través del OutputStream. La placa de Arduino recibe la cadena y la trata de tal forma que dependiendo lo que le llegue realiza una acción u otra.

Por ejemplo: nosotros enviamos una letra “a” y encendemos la luz de la habitación 1. Si enviamos la letra “b” apagamos la luz.

```
public void escribir(String message) {
    Log.d(TAG, "...Data to send: " + message + "...");
    byte[] msgBuffer = message.getBytes();
    try {
        mmOutStream.write(msgBuffer);
    } catch (IOException e) {
        Log.d(TAG, "...Error data send: " + e.getMessage() + "..."); }}
```

Este proceso se realiza con todos y cada uno de los fragmentos, estos envían parámetros a la actividad principal, y esta la gestiona mediante un switch. Dependiendo lo que llegue envía un tipo de información a la placa u otra.

```
private void reciboParametro(String p){
   Log.i(TAG, "Me llega al main el parametro... desde los fragments: " + p);
   switch (p){
      case "j":
         mConnectedThread.escribir("j");
         break;
      case "a":
         mConnectedThread.escribir("a");
         luz1 = "Encendida";
         break;
      case "b":
         mConnectedThread.escribir("b");
         luz1 = "Apagada";
         break;
      case "c":
         mConnectedThread.escribir("c");
         luz2 = "Encendida";
         break;
      case "d":
         mConnectedThread.escribir("d");
         luz2 = "Apagada";
         break;
      case "e":
         mConnectedThread.escribir("e");
         tempe = "20";
         break;
      case "f":
         mConnectedThread.escribir("f");
         tempe = "22";
         break;
      case "g":
         mConnectedThread.escribir("g");
         tempe = "24";
         break;
      case "h":
         mConnectedThread.escribir("h");
         tempe = "26";
         break;
      case "i":
         mConnectedThread.escribir("i");
         tempe = "28";
         break;
      case "l":
         mConnectedThread.escribir("l");
         luz3 = "Encendida";
         break;
      case "m":
         mConnectedThread.escribir("m");
         luz3 = "Apagada";
         break;  } }
```

Sistema de registros (Log)

El sistema de log es un proceso similar al que utilizamos al logearnos o registrarnos pero con una diferencia a la hora de comunicarnos con el fichero php para recibir los datos desde la base de datos.

Para ingresar los cambios realizados en la casa simplemente se le pasa un array de pairValues. Es decir con dos datos, el primero es un tag que le damos al valor a pasar, ej: 'id' : '1' el tag es id y el valor es 1. El código php recibirá con el método $_POST los valores y los insertara en la base de datos con la hora y fecha en el momento de la inserción.

Para recibir los datos y aquí la diferencia con los procesos anteriores, sigue utilizando la clase AsyncTask para realizar el proceso en segundo plano, esta clase la ejecutaremos en el método OnAttach específico de un fragment. Es decir cada vez que llamamos al fragmento ejecutamos el AsyncTask, llamamos al código php y aquí es donde empieza el proceso, el php simplemente realiza una select y pide la última fila que se insertó en la tabla, pero cada campo los debe guardar en un array que pasará a ser un objeto JSON (JavaScript Object).

Es decir cada valor tiene tambíen un tag y un valor. Ej: 'nombre': 'pepe','fecha':'2015:09:09','id_registro':'101'. Esto sería un objeto json, con la sentencia: print json_encode($array); devuelve a la aplicación todo el objeto JSON el cual debemos parsear para despedazarlo y poder utilizar cada uno de los valores con el código siguiente especifico de Android:

```
BufferedReader reader = new BufferedReader(
        new InputStreamReader(is, "iso-8859-1"), 8);
StringBuilder sb = new StringBuilder();
String line = null;
while ((line = reader.readLine()) != null) {
    sb.append(line + "\n");
}
is.close();
resultado = sb.toString();

 JSONObject obj = (JSONObject) new JSONTokener(resultado).nextValue();
 id  = obj.getString("id");
 nombre = obj.getString("nombre"); // Capitalizo para visualizar mejor
 nombre = nombre.substring(0,1).toUpperCase() + nombre.substring(1);
 luz1 = obj.getString("luz1");
 luz2 = obj.getString("luz2");
 luz3 = obj.getString("luz3");
 temperatura = obj.getString("temperatura");
 fecha = obj.getString("fecha");
```

Posibles Mejoras

Tanto la casa en la cual se alojan todos los sensores y dispositivos, como la aplicación para controlarlos, tienen un importante rango de mejora. Aunque la documentación presentada corresponde a un ejemplo básico, hay mejoras posibles a largo plazo tales como:

- Incluir cámaras en las estancias deseadas, para una video-vigilancia desde la aplicación móvil.

- Implementar el sistema de video-vigilancia mencionado para conectar la señal con los miembros de seguridad de la zona residencial, a través de WiFi.

- Añadir motores a las persianas para controlar la luz entrante hacia la casa.

- Instalar placas fotovoltaicas como forma de energía alternativa, sostenible y renovable, para la vivienda.

- Implementar la autonomía de la casa inteligente, añadiendo funciones como el apagado y encendido de luces, en función de lo captado por sensores de movimiento y luminosidad, riego automático o un sistema de gestión de un invernadero, entre otras posibilidades.

- Realizar una web con las funciones de la aplicación para poder controlar la casa desde cualquier pc o móvil sin tener instalada la aplicación.

- Pensando en los posibles fines comerciales, en la fase de registro, el usuario debería esperar un tiempo prudencial, hasta que el sistema garantice y autentique sus datos. Cuando esto ocurra, se devolvería una clave única de acceso con la que acceder por primera vez.

Conclusiones

Este proyecto nos ha dado la oportunidad de aprender mucho sobre el mundo de la domótica y un poco más sobre electrónica general.

En todo caso, se ha podido demostrar que es posible instalar un sistema domótico apoyándonos en la plataforma Arduino, con un coste muy inferior al que se utiliza en las viviendas de lujo, a cambio de dedicarle un poco de tiempo.

Según se avanzaba en el desarrollo del proyecto se hacía más necesario probar los distintos dispositivos que se pueden instalar. Es importante tener las herramientas adecuadas para trabajar dado que en algún momento del proyecto ha fallado algún dispositivo y tras comprobarlo a nivel físico con un tester eléctrico se detectaron los fallos, siendo reemplazado en cada caso y pudiendo continuar sin problemas, en consecuencia.

A la hora de programar los distintos elementos ayuda mucho realizar de antemano un esquema con las funciones que necesitamos y no alterarlo, ya que un proyecto se va haciendo cada vez más grande y tener que cambiar una cosa que, a priori, parece insignificante, puede dar mucho trabajo adicional.

Fuentes

Fórum Arduino
http://forum.arduino.cc/
25-Abril-2015

Fórum Arduino
http://forum.arduino.cc/?board=49.0
26-Abril-2015

YouTube
https://www.youtube.com/watch?v=mH2Q0KUMAI8
29-Abril-2015

El blog de Extremadura
http://www.extremadura-web.es/Blog/category/casa-inteligente-con-arduino/
5-Mayo-2015

Foro domótica con Arduino
http://excontrol.es/Arduino-Domotica-Foro/forumdisplay.php?fid=6
6- Mayo -2015

Taringa
http://www.taringa.net/posts/ciencia-educacion/17064197/Maqueta-de-una-casa-Domotica-Automatizada.html
10- Mayo -2015

Diego tecnology
http://www.diegotecnology.es/tutorial-arduino-domotica/
12- Mayo -2015

Casa domótica
http://es.slideshare.net/jenniferamadormartinez/casa-domotica-kboom?related=1
15- Mayo -2015

YouTube
https://www.youtube.com/watch?v=Is6pJyEv8s0
25- Mayo -2015

YouTube
https://www.youtube.com/watch?v=7ciWMxi4sSU
25- Mayo -2015