VNiVERSiDAD D SALAMANCA

CAMPUS DE EXCELENCIA INTERNACIONAL

Facultad de Ciencias

Grado en Ingeniería Informática

Anexo 4: Documentación técnica de programación

Comedero automático para Mascotas

Autor:

Álvaro Torijano García

Tutor:

Fernando de la prieta Pintado

Fecha de presentación: Enero 2019

1 Introducción:

En este anexo se presenta la documentación técnica del proyecto y se expondrá un a presentación del trabajo y el uso de las plataformas utilizadas, describiendo las librerías y la estructura del código, con la finalidad de que otro programador pueda modificar el proyecto a conveniencia.

2 Documentación de las bibliotecas:

2.1 Arduino:

2.1.1 LCD.h:

Esta biblioteca provee el uso fundamental de una pantalla LCD orientada a texto (en este caso una 16x2, lo que significa que tiene 16 caracteres por cada fila, y dos filas). Esta librería es la que ofrece los métodos para limpiar, mostrar texto, posicionar el cursor en una posición específica, y encender o apagar la retroiluminación.

2.1.2 LiquidCrystal_I2C.h:

Esta biblioteca expone una clase que extiende la clase que expone la librería LCD para poder controlar un LCD con un módulo LCM1602 que utiliza un microcontrolador PCF8574A que es un multiplexor de pines digitales para el bus I2C. Esta biblioteca lo que hace es proporcionar constructores para obtener objetos que se comunican con el PCF8574A por I2C enviando las operaciones que hace la librería LCD.

2.1.3 SoftwareSerial.h:

Esta librería permite crear interfaces UART sobre cualquier pin digital del microcontrolador. Es la librería que se utiliza para la comunicación con el módulo bluetooth.

2.1.4 EEPROM.h:

Esta librería permite leer y escribir de la memoria EEPROM del microcontrolador.

2.1.5 Wire.h:

Esta librería permite leer y escribir del bus I2C del microcontrolador

2.1.6 avr/pgmspace.h:

Esta librería permite almacenar variables estáticas en el espacio de programa, en vez de en la memoria principal. Se utiliza para poder mostrar textos largos.

2.2 Android:

2.2.1 Android.widget:

En este paquete es donde se encuentran los botones, cuadros de texto, selectores de fecha, listas, toast, vistas de imagenes y en definitiva cualquier elemento interactivo de la aplicación.

2.2.2 Android.view:

En este paquete se encuentran los elementos que permiten modelar las vistas a la hora de colocar los widgets. Aquí se encuentran elementos como gravity, menú, KeyEvent o LayoutInflater.

2.2.3 Android.os:

En este paquete aparecen los elementos que interactúan con el sistema operativo como por ejemplo Handler o Bundle.

2.2.4 Android.content:

Este paquete contiene los elementos "arquitectónicos" con los que interactuar para lanzar otros elementos de la aplicación. Estos elementos son por ejemplo: Intent, DialogInterface o BroadcastReceiver.

2.2.5 Android.app:

Aquí es donde aparecen los elementos que forman la aplicación y son los siguientes: Activity, datePickerDialog, AlertDialog, TimepickerDialog etc.

3 Funciones:

3.1 Arduino:

A continuación se explican las funciones mas relevantes del código. Se omitirán las conversiones.

Es importante destacar que en Arduino hay dos funciones que es obligatorio que nuestro código tenga:

Void setup(): Esta función se ejecuta solamente una vez al comienzo del código.

Void loop(): Esta función se ejecuta justo después de la anterior y se repite infinitamente.

No es posible modificar estos prototipos, puesto que si lo hacemos el compilador mostrará un error diciendo que la referencia a la función loop no se ha encontrado.

A parte de estas funciones, para este proyecto se utilizan las siguientes:

void resetearComedero(): Esta función lo que hace es mover el contador de programa a la primera posición de memoria. El resultado e sque el programa vuelve a empezar.

void setDS3231time(byte second, byte minute, byte hour, byte dayOfWeek, byte dayOfMonth, byte month, byte year): Esta funcion escribe en los registros del reloj de tiempo real para actualizer la fecha.

void readDS3231time(byte *second, byte *minute, byte *hour, byte *dayOfWeek, byte *dayOfMonth, byte *month, byte *year): Esta función lee lo registros del reloj de tiempo real para obtener la fecha y hora.

void displayTime(): Esta funcion muestra la hora en pantalla usando siempre el mismo numero de dígitos.

void encenderLCD(): esta función sirve para fijar el momento en el que el bucle principal apagará la pantalla.

void mostrarMensaje(): Esta funcion sirve para evitar que el buble principal sobreescriba el mensaje que se está mostrando en la pantalla, pero que pueda seguir haciendo comprobaciones.

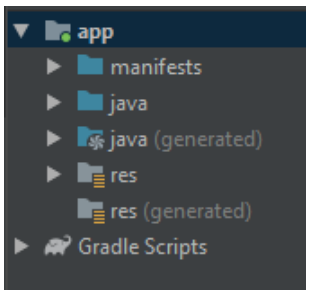
byte comparaOpcion(char *, String *, byte *, byte, byte *): Esta funcion recibe un vector de cadenas, su tamaño, un vector de tamaños con los tamaños de cada cadena, un entero por referencia para devolver el numero de caracteres desestimados y una cadena en la que buscar el resto de parámetros. Esta función devolverá un 0 en caso de fallo o el numero de la cadena perteneciente al array que encontró en la cadena en la que se buscaba.

void girarTolva (int vueltas, bool sentido): esta función hace girar la tolva el numero de veces que se le pase, y en el sentido horario o anti horario.

void dar_de_comer(): esta función hace girar la tolva el numero necesario de veces para suministrar todas las dosis que el animal tiene que recibir.

3.2 Android:

Para Android se explicará el árbol de directorios, y los métodos mas relevantes sobre los que se construye la funcionalidad de la aplicación.



De estas carpetas aquellas que muestran la etiqueta (generated) son carpetas con código autogenerado que no será necesario editar. Por tanto solamente quedan 4 elementos que debamos revisar:

En primerl lugar el árbol de directorios se divide en dos elementos: APP y Gradle Scripts.

App es la carpeta que contiene todo el código de la aplicación, mientras que Gradle Scripts es la carpeta que contiene las instrucciones para configurar la herramienta Gradle que es un generador de código que se utiliza para construir la aplicación a partir del código que ha escrito el programador.

Los scripts de Gradle tampoco se van a modificar por lo que solamente queda la carpeta APP:

Dentro de esta carpeta aparece:

3.2.1 Manifest:

Dentro de esta carpeta se ubica el manifiesto. Un manifiesto es una relación de elementos y describe las activities que la aplicación contiene y algunos de sus parámetros de funcionamiento.

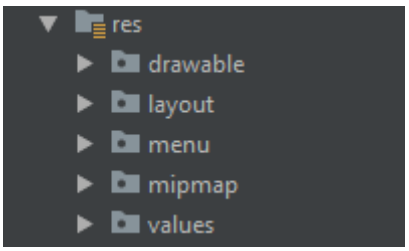
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="me.aflak.bluetoothterminal" >

    <application
        android:icon="@mipmap/ic_launcher"
        android:label="AutoSulivansFeeder"
        android:theme="@style/AppTheme" >

        <activity
            android:name=".Chat"
            android:label="AutoSulivansFeeder"
            android:screenOrientation="portrait">
        </activity>

        <activity
            android:name=".Scan"
            android:label="AutoSulivansFeeder"
            android:screenOrientation="portrait">
        </activity>

        <activity
            android:name=".Select"
            android:label="AutoSulivansFeeder"
            android:screenOrientation="portrait">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>

</manifest>
```

3.2.2 Java:

La siguiente carpeta que encontramos es “java”. En esta carpeta aparecen las clases de Java que se corresponden con cada una de las activities. Dentro de cada una de estas clases aparecen los métodos que se ejecutan cuando se produce un evento dentro de ese activity.

3.2.3 Res:

Dentro de esta carpeta se sitúan a su vez otras cinco:



3.2.3.1 Drawable:

Como su nombre indica contiene los recursos que son dibujables, esto es: imágenes definidas por el usuario, que aunque podrían ubiarse en cualquier parte del árbol de directorios, lo apropiado es que estén contenidas en esta carpeta.

3.2.3.2 Layout:

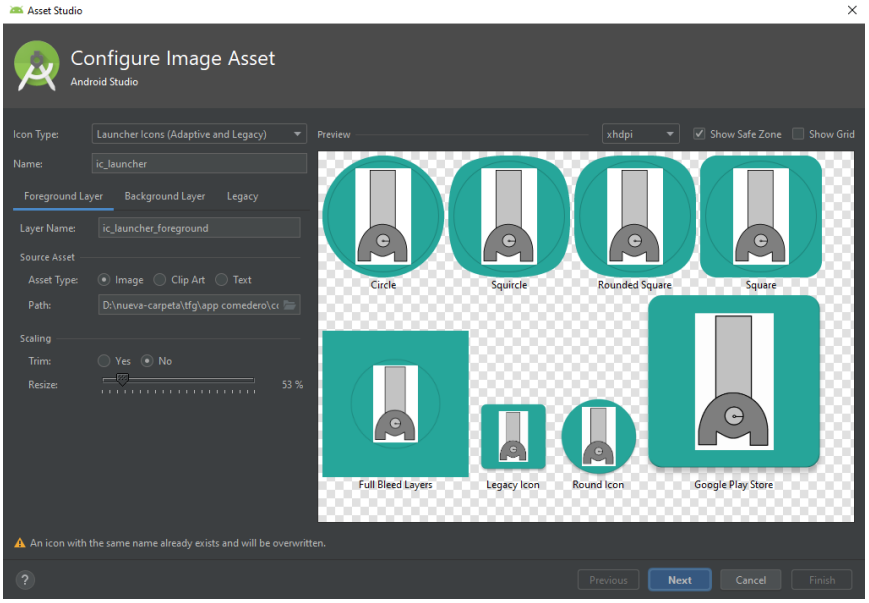
Junto con la carpeta, esta es la mas importante. Contiene los archivos XML que definen los elementos y su posición en las vistas. Estos archivos no contienen lógica, solamente reglas y propiedades de los objetos.

3.2.3.3 Menu:

En esta carpeta se coloca el correspondiente archivo XML para el menú de la aplicación (de ternerlo)

3.2.3.4 MipMap:

Al igual que la carpeta Drawable contiene imágenes, esta carpeta también contiene imágenes, con la salvedad de que está únicamente dedicada a contener el icono de la aplicación en diferentes resoluciones, formas o relaciones de aspecto. Esta carpeta junto con todos sus archivos se puede autogenerar usando el asistente que Visual Studio provee como se muestra a continuación:



3.2.3.5 *Values:*

Esta carpeta está dedicada a almacener archivos que contienen variables fijas. Seria el equivalente a guardar constantes o #defines. El objetivo de esta parte del proyecto es igual que en otros lenguajes de programación, salvo porque en JAVA este mecanismo no existe.

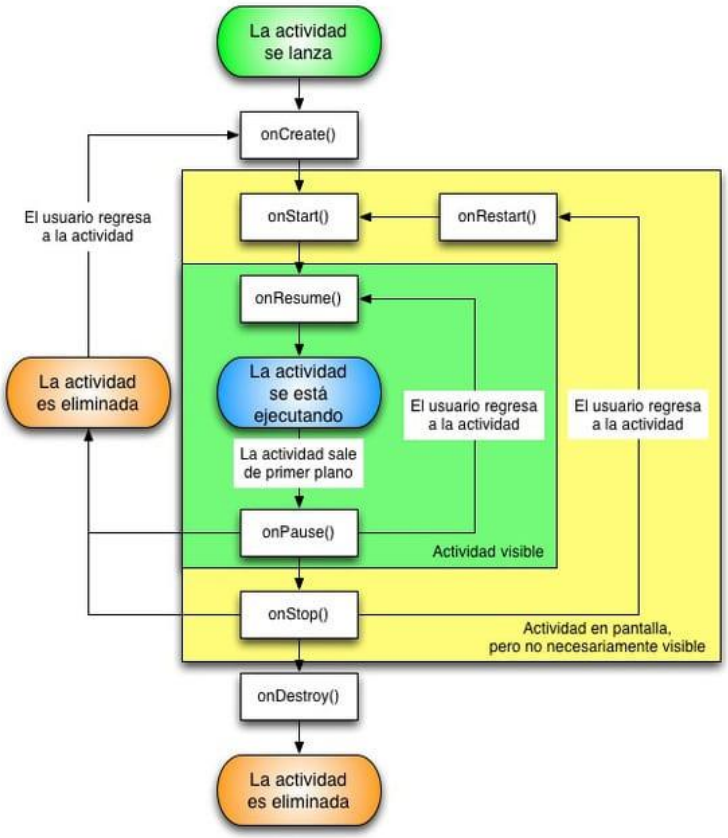
Nota: esta aplicación está hecha a partir de un ejemplo proporcionado por Omar Aflak en su github.

3.2.3.6 *Métodos en Android:*

En este apartado se explica el funcionamiento básico de una activity en Android.

Básicamente una activity es un objeto que hereda de la clase activity y que por tanto implementa los métodos onCreate(), onStart() onResume(), onRestart(), onPause(), onStop() y onDestro().

Estos métodos son llamados cuando la activity cambia de estado siguiendo el ciclo de vida que ya se mencionó anteriormente en la introducción y que sigue el siguiente esquema:



En este apartado se describirán los métodos mas relevantes utilizados en la construcción de la aplicación, todos estos métodos se ejecutan en el método onCreate() del activity para que sean ejecutados mientras el activity está vivo.

Widget.setOnClickListener()

Este método permite crear un escuchador, que es un método que se ejecuta cuando se dispara una acción, en este caso la acción es un click en el elemento Widget (siendo widget cualquier elemento que herede de esa clase, por ejemplo, un boton).

Widget.setOnEditorActionListener()

Este método al igual que el anterior, crea un escuchador que se dispara cuando se edita el valor que hay en el wiget.

void onMessage(String message)

esta es la funcion que se ejecuta cuando el dispositivo bluetooth recibe un mensaje.

public void Display(final String s)

Esta funcion muestra un texto en cuadro de texto que se utiliza como interfaz para la consola de texto de la interfaz serie.

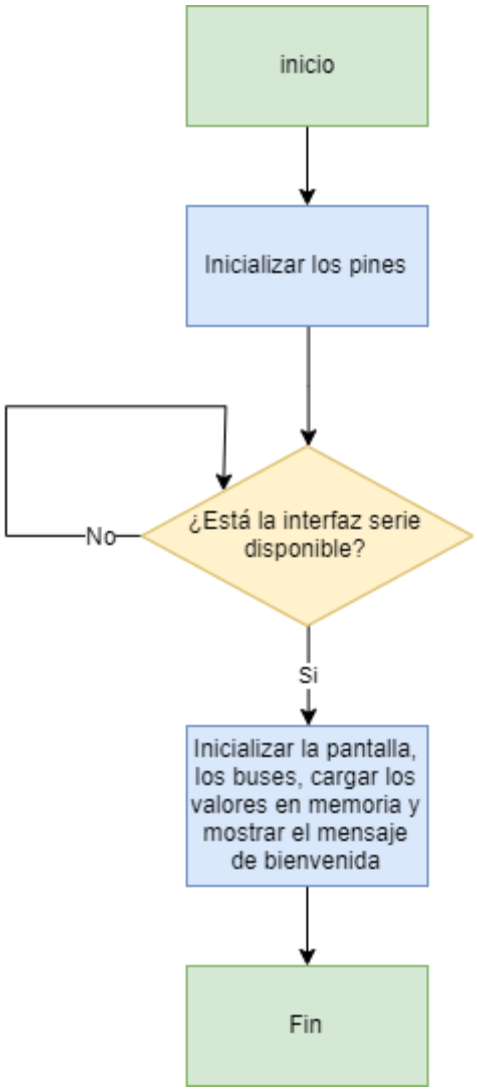
public boolean onOptionsItemSelected(MenuItem item)

Este metodo es llamado cuando se selecciona alguna opción del menú que Android provee para las aplicaciones y que suele representarse como tres puntos verticales.

4 Diagramas de flujo:

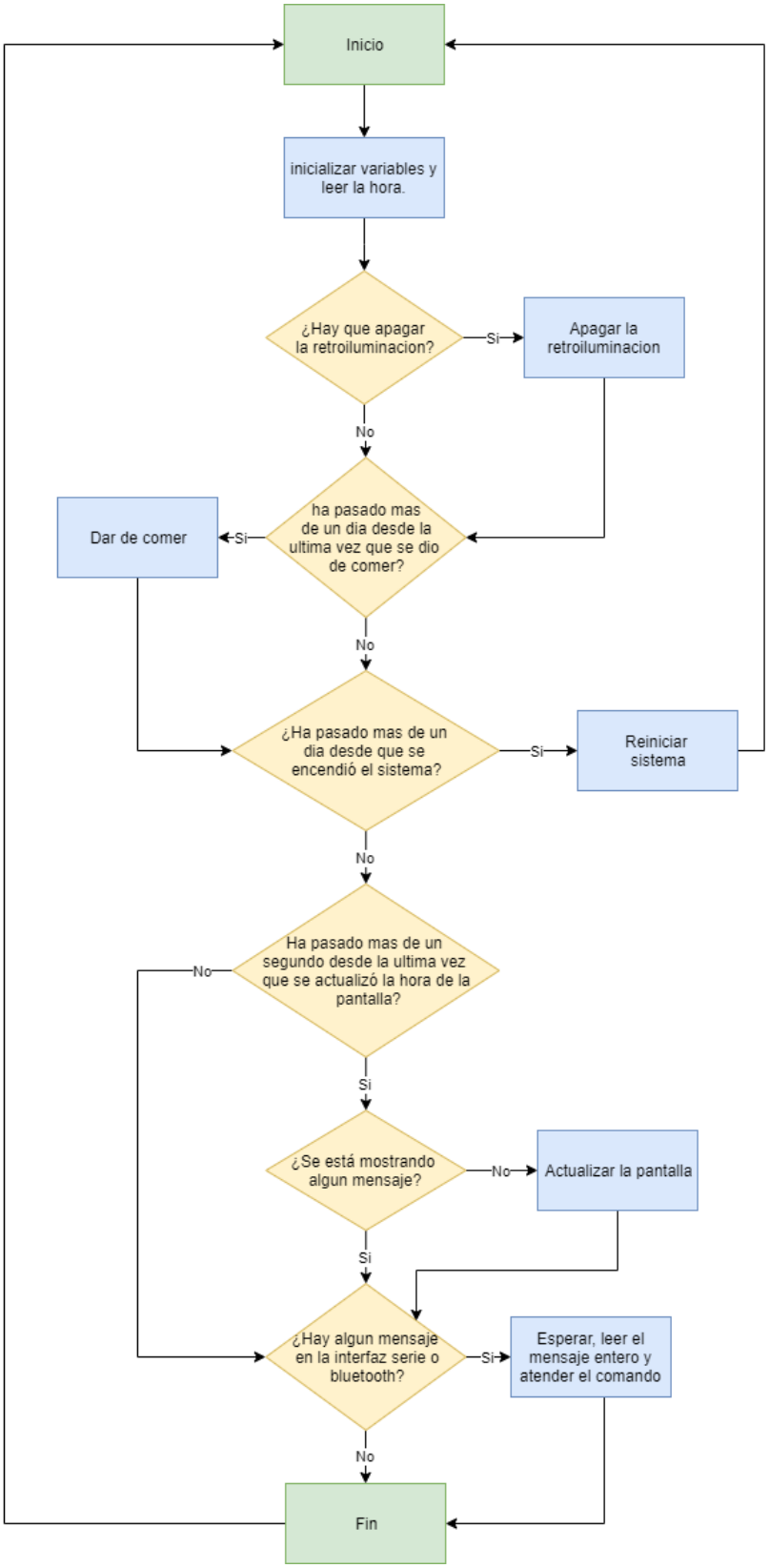
4.1 Función Setup

En este diagrama de flujo se resume la actividad de la función setup durante la que se hace la incializacion del hardware, se lee la hora por primera vez, y después se muestra el mensaje de bienvenida.



4.2 Función loop

En esta función es donde se encuentra toda la funcionalidad del dispositivo. Esta función ha sido diseñada para dar la sensación de simultaneidad en las tareas puesto que cuando este proyecto se diseñó, aun no estaba extendido el uso de sistemas operativos de tiempo real en el entorno Arduino.



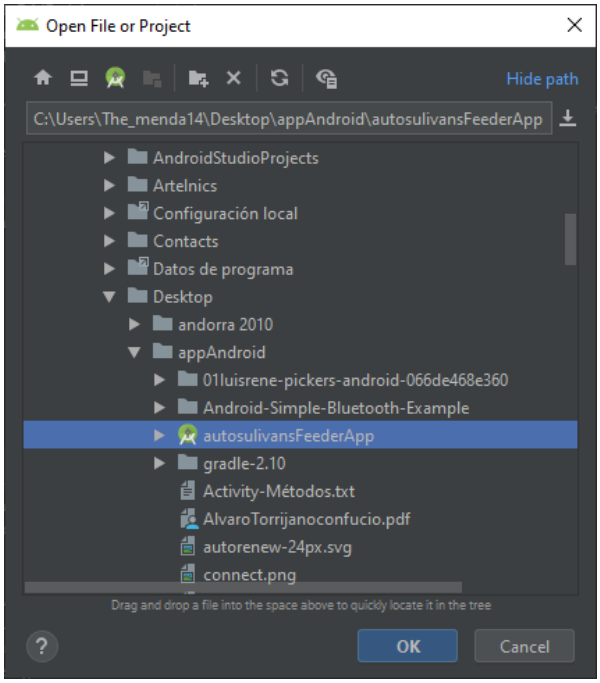
5 Manual del programador

En este apartado se trata de explicar los pasos que tendría que seguir un programador para modificar ampliar o reparar fallos si los hubiere en la aplicación.

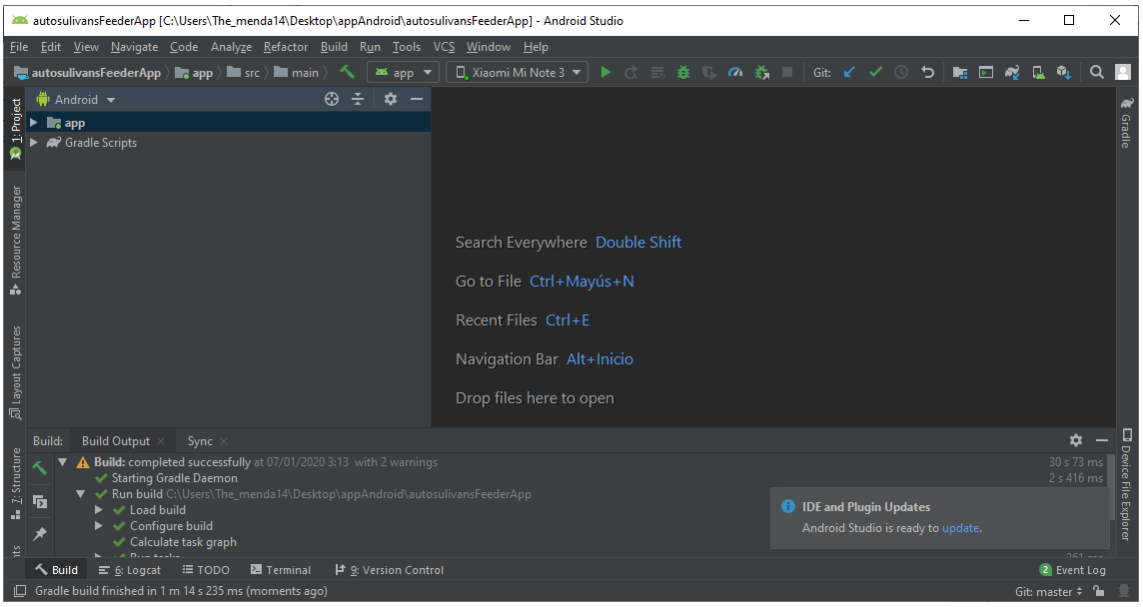
Para ello se asume que el programador cuenta con el entorno de desarrollo instalado, en este caso Android Studio que puede descargase haciendo click en el enlace anterior e instalarse normalmente en un sistema operativo Windows.

5.1 Importar proyecto:

Para importar el proyecto es necesario clonar primeramente el repositorio que puede encontrarse en https://github.com/alvarotorijano/autosulivansFeederApp abrir el entorno y hacer click en File -> Open y seleccionar la carpeta donde se ha clonado el repositorio.



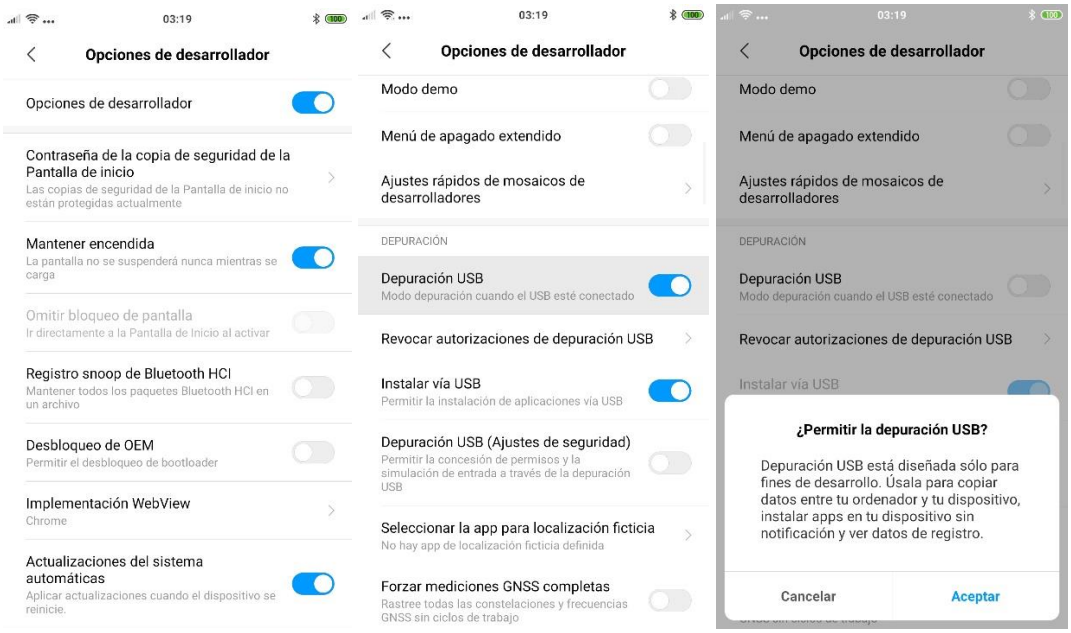
Hecho esto, se cargará el proyecto en el entorno de desarrollo y se mostrara en el explorador de archivos situado a la izquierda:



5.2 Depuracion:

Para depurar la aplicación es necesario contar con un dispositivo físico Android para evitar los problemas a la hora de acceder al hardware bluetooth puesto que, desde los dispositivos virtuales, a día de hoy, no es posible.

Para ello es necesario habilitar en el dispositivo las opciones de desarrollador, que dependiendo de la versión de Android o “aroma” que esté instalada en el dispositivo su ubicación dentro del menú de opciones varia o incluso puede llegar a estar oculta como es lo mas habitualmente últimamente. El proceso es el siguiente:



Eventualmente es posible que cuando se trate de instalar la primera aplicación se solicite permiso para autorizar al PC y posteriormente permiso para instalar la aplicación.

Una vez obtenidos los permisos, el proceso de depuracion es el habitual: Se colocará un punto de rupura que detendrá la ejecucion de la aplicación llegados a ese punto, mostrando el valor que tienen las variables en ese momento y permitiendo la ejecucion paso a paso.

6 Pruebas unitarias:

Las pruebas realizadas para la comprobación del correcto funcionamiento de la aplicación son las descritas en el apartado “Pruebas de aceptación” que se detalla anteriormente, y de las cuales, las que corresponden con la aplicación Android son las siguientes:

- Prueba de conectividad con la aplicación
- Prueba de interrupción de la conexión
- Prueba de interfaz y usabilidad
- Prueba de despliegue