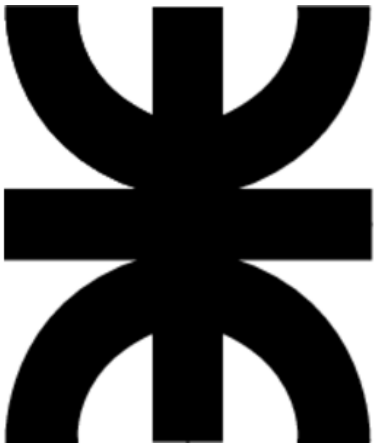
Universidad Tecnológica Nacional

Facultad Regional del Neuquén

Ingeniería Electrónica

Tesis de Proyecto Final:

Procesamiento de Imágenes orientado a Personas No Videntes



Alumna: Inda, María Lucrecia.

Profesor: Ing. Monte, Gustavo.

Año: 2017.

Índice:

❖ 1. Introducción:

En el presente informe se describirá el funcionamiento de un dispositivo móvil que tiene como objetivo la asistencia a personas no videntes en ambientes no estructurados.

Este dispositivo está conformado por una aplicación móvil dentro de un teléfono celular y un auricular conectado al mismo mediante conexión Bluetooth. Esta aplicación, programada en una plataforma Android y utilizando la librería OpenCV, es la encargada de procesar la información del ambiente tomada por la cámara y de comunicar la ocurrencia de eventos detectados, según lo requerido por el usuario.

Los eventos a detectar están pensados para orientar a una persona no vidente dentro del contexto de una ciudad, beneficiando a su desenvolvimiento y su autonomía en la misma. Las posibilidades que brinda este dispositivo abarcan el reconocimiento de colores, formas, caracteres y palabras. De este modo, puede alertarse a la persona mediante indicaciones sonoras, del estado de un semáforo, la presencia de una farmacia, una estación de subte o un banco; según lo requiera.

Este proyecto se presenta como Proyecto Final de la carrera Ingeniería Electrónica de la Universidad Tecnológica Nacional, Facultad Regional del Neuquén.

❖ 2. Desarrollo:

❖ 2.1 Descripción general:

La idea del presente proyecto es servir como herramienta guía para personas no videntes, dentro de ambientes no estructurados como lo es, por ejemplo, una ciudad. Esta herramienta asistirá a la persona en su cotidiano desenvolvimiento urbano, alertándola de eventualidades tales como el estado de los semáforos, o la presencia de lugares de interés, según lo requiera.

El proyecto consta de la programación de una aplicación móvil encargada de la obtención y procesamiento de imágenes tomadas del ambiente mediante la cámara incorporada en el dispositivo móvil. Se podrá elegir y cambiar la función a ejecutar mediante la posición y orientación del mismo, pudiéndose elegir entre la búsqueda del cartel indicador de una farmacia o de un subte, o la búsqueda del estado de un semáforo. Así, según la función que se active, se obtendrán diferentes resultados, los cuales serán comunicados al usuario mediante alertas sonoras reproducidas por un auricular inalámbrico. Éste estará comunicado con la aplicación mediante comunicación Bluetooth.

La programación de la aplicación móvil, se realizó bajo el sistema operativo Android, en la plataforma de programación Android Studio, con la utilización de la librería libre OpenCV para el procesamiento de imágenes y de la librería Tesseract para el reconocimiento de caracteres; mientras que el auricular inalámbrico fue programado en el entorno de desarrollo MPLAB X, en conjunto con Arduino para la configuración del módulo Bluetooth. Todos estos procedimientos, además de algunas funciones y resultados obtenidos, se describirán más detalladamente a continuación, así también como pruebas previas realizadas en el entorno de desarrollo Netbeans, donde se trabajó con cámaras web con la implementación de otras funcionalidades de dichas herramientas.

❖ 2.2 Esquema general:

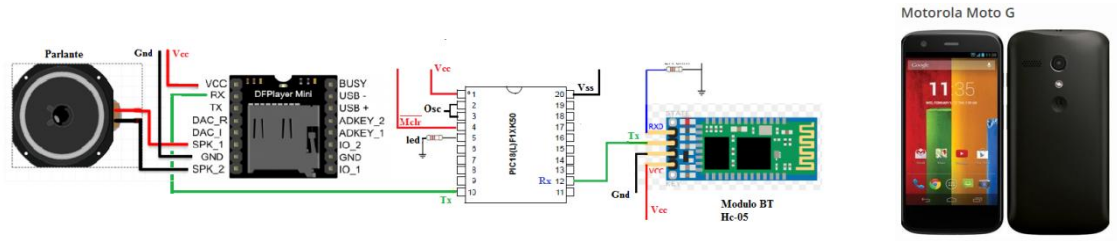


Fig 1: diagrama de bloques y conexión

❖ 2.3 Desarrollo de Aplicación Móvil:

Para el desarrollo de este proyecto se utilizó un teléfono móvil Motorola "Moto G", con un sistema operativo Android 4.4 "Kitkat" (nivel de API 19-20). El mismo cuenta con una cámara trasera de 5MP, una resolución de 2592x1944 pixeles, una resolución de video de 720p@30fps, Bluetooth v4.0.

Como ya se mencionó anteriormente, la aplicación se desarrolló en el entorno de programación Android Studio, el cual puede ser descargado gratuitamente. A modo de guía, se confeccionó un tutorial paso a paso, para la creación de un nuevo proyecto en dicho entorno el cual se encuentra al final de este informe, como ANEXO A: "Cómo crear un nuevo proyecto en Android Studio" .

Existen varias cosas a tener en cuenta a la hora de programar una aplicación: entre la mas importantes se encuentran: el manejo de diferentes archivos, los cuales configuran, compilan y manejan la aplicación; y lo relacionado con la versión del sistema operativo (API); así también como la utilización de permisos a la hora de hacer uso de los recursos del dispositivo tales como la cámara, almacenamiento de datos en memoria, GPS, Bluetooth, etc.

Los archivos mencionados, relacionados con el futuro funcionamiento y comportamiento de la aplicación se describen a continuación:

- AndroidManifest.xml: Proporciona información esencial sobre la aplicación al sistema Android para poder ejecutarla. Entre otras cosas: nombra el paquete Java identificador, describe los componentes como actividades y servicios y sus procesos, declaración de permisos para acceso a partes protegidas tanto por la API como de la misma aplicación, declara el nivel mínimo de API, etc. Se ubica dentro de: app→manifest.
- Activity_main.xml: define el diseño de la aplicación. Por defecto, contiene un elemento del tipo TextView; sin embargo en este archivo deberán definirse todos los elementos, tales como botones, cuadros de texto, barras desplazadoras, imágenes ventanas, etc., así también como sus tamaños, pesos, ubicación en el Layout, etc. este archivo puede encontrarse en: app→res→layout.

- MainActivity.java: Clase Principal. En este archivo se define la clase de la actividad.
- Build.gradle: Android utiliza Gradle para compilar la aplicación, la cual es una herramienta que automatiza la construcción de los proyectos. Se encarga no solo de su compilación, sino también de testing, empaquetado y despliegue. En la compilación, el proyecto se convierte en un paquete de Android : APK (Aplicación Empaquetada de Android), el cual es básicamente un archivo comprimido, una variante del formato JAR de Java y contiene todo el código del proyecto.

Este último archivo guarda una estrecha relación con la compilación en cada versión del sistema operativo: las **API** (*Interfaz de Programación de Aplicaciones*). Las API son el conjunto de funciones, subrutinas y procedimientos, y representa la capacidad de comunicación entre componentes de software. Incluye:

- Conjunto de atributos y elementos XML para declarar un archivo "Manifiesto".
- Atributos y elementos XML para declarar recursos y acceder a ellos.
- Conjunto básico de paquetes y clases.
- Conjunto de "Intents".
- Conjunto de permisos.

Las actualizaciones de las APIs están diseñadas para que la nueva API siga siendo compatible con las versiones de API anteriores; es decir, que los cambios en la API se agregan o presentan funcionalidades nuevas o de reemplazo. Las partes más antiguas reemplazadas dejan de estar disponibles pero no son eliminadas; de este modo las aplicaciones que hacen uso de ellas, aún pueden utilizarlas.

El identificador de cada API, un número entero llamado "nivel de API", es compatible con una versión de la plataforma de Android.

Version	Codename	API	Distribution
2.2	Froyo	8	0.1%
2.3.3 - 2.3.7	Gingerbread	10	2.0%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	1.9%
4.1.x	Jelly Bean	16	6.8%
4.2.x		17	9.4%
4.3		18	2.7%
4.4	KitKat	19	31.6%
5.0	Lollipop	21	15.4%
5.1		22	20.0%
6.0	Marshmallow	23	10.1%

Fig 2:Versiones de Android y API

El nivel de API principalmente, permite que el sistema negocie la instalación de aplicaciones en el dispositivo, de modo que no se instalen aplicaciones incompatibles con la versión del sistema operativo instalada. Por esto las aplicaciones pueden usar un elemento llamado SDK (Kit de Desarrollo de Software), el cual describirá los niveles de Api mínimo y máximo, dentro de los cuales podrá ejecutarse:

- ✓ Android:minSdkVersion: nivel mínimo de API en el cual puede ejecutarse. De ser ejecutada en una versión anterior, presentará errores en el funcionamiento de la aplicación.
- ✓ Android:TargetSdkVersion: nivel de API para el cual está diseñada la aplicación.
- ✓ Android:maxSdkVersion: nivel máximo de API. Es recomendable utilizar en este parámetro, el último nivel de API disponible, es decir, el más actual.

```
android {
    compileSdkVersion 24
    buildToolsVersion "25.0.0"
    defaultConfig {
        applicationId "com.example.lucrecia.btapp"
        minSdkVersion 15
        targetSdkVersion 24
        versionCode 1
        versionName "1.0"
```

Una vez que se creó correctamente el proyecto, se procedió con la programación de la aplicación comenzando por agregar y configurar las librerías correspondientes al procesamiento de imágenes como se describe a continuación:

❖ 2.3.1 Librerías:

❖ 2.3.1.1Librería para Procesamiento de Imágenes:

Para llevar a cabo la obtención de información específica del ambiente circundante mediante la cámara incorporada en el teléfono móvil, se utilizó la librería OpenCV en su versión 3.1.0 para Android. Esta librería libre de **visión artificial** y originalmente lanzada por Intel, soporta diversos sistemas operativos, incluyendo Android, y tiene interfaces para C++, C, Python y Java. Resulta de gran utilidad, ya que fue diseñada para la eficiencia computacional y enfocada a aplicaciones en tiempo real.

❖ 2.3.1.2. Librería para Reconocimiento de Caracteres:

Conjuntamente se utilizó "Tesseract", uno de los tres mejores motores de Reconocimiento Óptico de Caracteres (OCR), el cual fue liberado por Hewllet-Packard en 2005 y actualmente desarrollado y mejorado por Google. Puede procesar diversos idiomas, entre ellos: inglés, español, alemán, italiano, portugués. Desde la versión 3, Tesseract soporta el formato en el texto y el análisis del patrón de la página.

- *OpenCV y Tesseract en Andorid:*

Habiendo descargado la última versión de las librerías *OpenCV para Android*, y teniendo en cuenta los recaudos relacionados a la versión del sistema operativo a utilizar mencionados anteriormente, se confeccionó una guía con los pasos a seguir para su correcta compilación. La misma se encuentra disponible al final del presente informe, llamado "*ANEXO B: Agregar OpenCV y Tesseract en Android Studio*."

Para el caso de *Tesseract*, antes de poder hacer uso de la librería, primero se debe construirla de tal forma que pueda ser compilada posteriormente con Gradle. Para lograrlo, pueden seguirse los pasos descritos *en el mismo anexo*.

Una vez que se añadieron ambas librerías, se comenzó con la activación de la cámara y la obtención y visualización de los *frames* (de formato Mat), como se muestra a continuación:

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

    javaCameraView = (JavaCameraView) findViewById(R.id.java_camera_view);
    javaCameraView.setVisibility(SurfaceView.VISIBLE);
    javaCameraView.setCvCameraViewListener(this);
```

```
@Override
public Mat onCameraFrame(CameraBridgeViewBase.CvCameraViewFrame inputFrame) {
    mRgba = inputFrame.rgba();
    // Rotate mRgba 90 degrees
    Core.transpose(mRgba, mRgbaT);
    Imgproc.resize(mRgbaT, mRgbaF, mRgbaF.size(), 0, 0, 0);
    Core.flip(mRgbaF, mRgba, 1);
```

❖ 2.4 Funciones implementadas:

Habiendo configurado correctamente la cámara del celular se comenzó con el diseño de rutinas, la cuales debieron ser activadas y seleccionadas de manera tal de no hacer uso de Botones en la pantalla, ya que esto dificultaría el manejo de la aplicación por

la persona no vidente. El mecanismo encargado de elegir y activar una de las funciones disponibles en este proyecto, será el **movimiento del dispositivo**.

Esto significa que se activará una función, de acuerdo a la posición u orientación del dispositivo móvil, a diferencia de otras aplicaciones móviles comunes, ya que se precisa la independencia de mecanismos de activación habituales. La información requerida es brindada por el **acelerómetro** incorporado dentro del dispositivo. La misma es representada por valores de los ejes X, Y y Z, los cuales pueden adoptar un valor entre -10.0 y 10.0. La representación de los ejes se ve ilustrada a continuación; por ejemplo: los valores arrojados mientras el dispositivo descansa pantalla hacia arriba son: X=0.15, Y=0.2, Z=9.7. La disposición de los ejes se ilustra en la figura:

Fig 3: Disposición de Ejes Acelerómetro

Para la aplicación actual, se configuraron 3 movimientos opcionales tomando como posición de neutra o de referencia cuando el dispositivo está en posición vertical, con la pantalla de frente al usuario. Esta posición arroja los siguientes valores aproximados de los ejes: X=0.1, Y=9.8, Z=0.2.

Obteniendo los cambios en el valor de los ejes, se configuró la siguiente activación:

- ✓ Movimiento hacia la derecha: Activación función “Búsqueda de Farmacia”
- ✓ Movimiento hacia la izquierda: Activación función “Búsqueda de Subte”.
- ✓ Inclinación (cámara trasera) hacia arriba: Activación función “Búsqueda de Semáforo”.

La siguiente porción de código muestra dicha configuración, implementada en el MainActivity.java del proyecto:

```
synchronized (this) {
    Log.d("sensor", event.sensor.getName());
    switch (event.sensor.getType()) {
        case Sensor.TYPE_ACCELEROMETER:
            txt += "acelerometro\n";
            txt += "\n x: " + event.values[0] + " m/s";
            txt += "\n y: " + event.values[1] + " m/s";
            txt += "\n z: " + event.values[2] + " m/s";

            if (event.values[0] >= 4) {
                txt2 = "\n Izquierda (SUBTE)";
                layout.setBackgroundColor(Color.BLUE);
                mov = 1;
                //Llamar a una funcion pasando como parametro mov y que aplique el filtro que corresponda

            } else if (event.values[0] <= -4) {
                txt2 = "\n Derecha (FARMACIA)";
                layout.setBackgroundColor(Color.GREEN);
                mov = 2;
            } else {
                txt2 = "\n ESPERANDO INSTRUCCION";
                layout.setBackgroundColor(Color.WHITE);
            }

            if (event.values[2] >= 4) {
                txt2 = "\n Cámara hacia Abajo";
                layout.setBackgroundColor(Color.YELLOW);
            } else if (event.values[2] <= -4) {
                txt2 = "\n Cámara hacia Arriba (SEMAFORO)";
                layout.setBackgroundColor(Color.RED);
                mov=3;
            }
            orientacion.setText(txt2);
            acelerometro.setText(txt);
            break;
    }
```

Comenzando por la búsqueda del **estado del semáforo**, se diseñó una función que filtra la imagen por colores. Para esto se debe hacer una conversión de la representación de la imagen en la matriz mat. Opencv, al tomar un frame de la cámara la representa mediante 3 matrices correspondientes a los colores Rojo (R), Verde (G) y Azul (B) que contienen valores que van de 0 a 255 (esto se explicará más detalladamente mas adelante en este informe, precisamente en la parte dedicada al procesamiento de imágenes en la PC: página XX). Se debe utilizar la representación HSV: Hue (), Saturation (saturación) y Value (valor) la cual se ilustra de la siguiente manera:

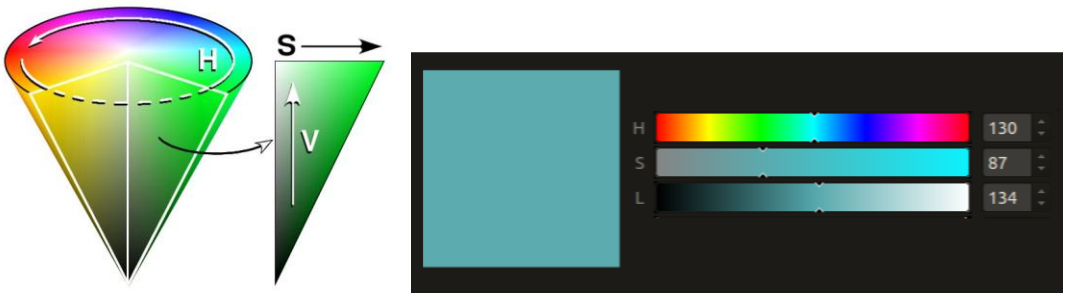


Fig 4: Modelo de Colores HSV

Para transformar una matriz RGB a una HSV y viceversa, se utilizan las siguientes fórmulas:

RGB a HSV:

$$H = \begin{cases} \text{no definido}, & \text{si } MAX = MIN \\ 60° \times \frac{G-B}{MAX-MIN} + 0°, & \text{si } MAX = R \text{ y } G \geq B \\ 60° \times \frac{G-B}{MAX-MIN} + 360°, & \text{si } MAX = R \text{ y } G < B \\ 60° \times \frac{B-R}{MAX-MIN} + 120°, & \text{si } MAX = G \\ 60° \times \frac{R-G}{MAX-MIN} + 240°, & \text{si } MAX = B \end{cases}$$

$$S = \begin{cases} 0, & \text{si } MAX = 0 \\ 1 - \frac{MIN}{MAX}, & \text{en otro caso} \end{cases}$$

$$V = MAX$$

HSV a RGB:

$$H_i = \left\lfloor \frac{H}{60} \right\rfloor \bmod 6; H \leq 360$$

$$f = \left(\frac{H}{60} \bmod 6\right) - H_i$$

$$p = V(1 - S)$$

$$q = V(1 - fS),$$

$$t = V(1 - (1 - f)S)$$

$$\text{si } H_i = \begin{cases} 0, & R = V \\ & G = t \\ & B = p \\ 1, & R = q \\ & G = V \\ & B = p \\ 2, & R = p \\ & G = V \\ & B = t \\ 3, & R = p \\ & G = q \\ & B = V \\ 4, & R = t \\ & G = p \\ & B = V \\ 5, & R = V \\ & G = p \\ & B = q \end{cases}$$

Con OpenCV este cálculo es transparente al programador, ya que puede obtenerse la matriz resultado de estos cálculos aplicando la función:

Imgproc.cvtColor(Mat origen, Mat destino, Imgproc.COLOR_RGB2HSV).

Ahora, para poder filtrar por colores se debe especificar el rango de valores HSV que corresponda al color deseado:

- Verdes: de [49, 50, 50] a [80, 255, 255]
- Azules: de [100, 65, 75] a [130, 255, 255]
- Rojos: de [0, 65, 75] a [12, 255, 255] y de [240, 65, 75] a [256, 255, 255]

Fig 5: Modelo de Colores HSV – Umbrales Rojo

La función utilizada para el obtención de los valores de un solo color es: inRange

Core.inRange(Mat origen, Scalar Lower, Scalar Upper, Mat destino)

A continuación se ilustra la porción de código del proyecto donde se hace uso de estas funciones para obtener una representación binaria de objetos de color VERDE:

```
} else if (mov == 3) {
    Imgproc.cvtColor(mRgba, mRgba, Imgproc.COLOR_RGB2HSV);
    Core.inRange(mRgba, new Scalar(49, 50, 50), new Scalar(80, 255, 255), mRgba); //Detecta verdes
    //mandar a detectar circulo
    mRgba = DetectarCirculo(mRgba);
}
```

Con el resultado obtenido, se desea detectar círculos, con el objetivo de asegurar la detección de la luz del semáforo. Se implementa la función Huogh.circles la cual fue evaluada en la implementación en PC que se describe más adelante en este informe (pág X)

```
public Mat DetectarCirculo(Mat img) {
    Mat circles = new Mat();

    Imgproc.cvtColor(mRgba, mCanny, Imgproc.COLOR_GRAY2RGB);
    Imgproc.cvtColor(mCanny,mCanny,Imgproc.COLOR_RGB2GRAY);

    Imgproc.HoughCircles(mCanny, circles, Imgproc.CV_HOUGH_GRADIENT, 1, mCanny.rows() / 8, 200, 100, 0, 0);

    if (circles.cols() > 0)
        for (int x = 0; x < circles.cols(); x++) {
            double vCircle[] = circles.get(0, x);

            if (vCircle == null)
                break;

            Point pt = new Point(Math.round(vCircle[0]), Math.round(vCircle[1]));
            int radius = (int) Math.round(vCircle[2]);

            // draw the found circle
            // circle center
            Imgproc.circle(mCanny, pt, 3, new Scalar(0, 255, 0), -1, 8, 0);
            // circle outline
            Imgproc.circle(mCanny, pt, radius, new Scalar(0, 0, 255), 3, 8, 0);
        }

    return mCanny;
}
```

Esto mismo, puede implementarse para la luz Roja del semáforo, cambiando sólo por los umbrales de Rojos Bajos ([0, 65, 75] a [12, 255, 255]), ya que no se obtuvieron los resultados esperados al implementar los umbrales altos.

❖ 2.4.1 OCR en Android:

Como ya se mencionó anteriormente, para el reconocimiento de caracteres se necesita la librería Tesseract. Y es importante saber debe utilizarse el entrenamiento de la librería para el idioma que quiera detectarse, ya que cada uno de ellos presenta caracteres diferente. De lo contrario, los resultados obtenidos distarán de los esperados.

Para esto, los entrenamientos de diferentes idiomas pueden encontrarse en la web para ser descargados; se tratan de archivos del tipo "***traineddata***". Para esto, se debe crear una carpeta con el nombre "*tessdata*" dentro "*assets*" y copiar allí el archivo "*eng.traineddata*" para inglés o "*spa.traineddata*" para español:

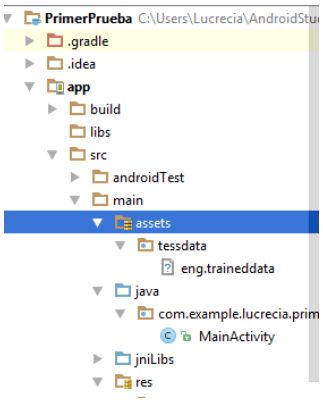


Fig 6: Ubicación de archivo "Traineddata"

Luego, deben tenerse ciertas consideraciones a la hora de su configuración: como lo es la ruta en donde queda guardado este archivo; de ahí la importancia de copiarlo en esa carpeta en específico. La configuración se lleva a cabo de la siguiente manera:

Dentro del método OnCreate del MainActivity: inicializar

```
//initialize Tesseract API
String language = "eng";
datapath = getFilesDir() + "/tesseract/";
mTess = new TessBaseAPI();

checkFile(new File(datapath + "tessdata/"));

mTess.init(datapath, language);
```

Y programar una función para el reconocimiento de caracteres:

```
//CODIGO DE TESSERACT

    public void processImage(){

        tmp=mRgba.clone(); //tmp = Mat declarada global, mRgba = frame actual

        try {

            image = Bitmap.createBitmap(tmp.cols(), tmp.rows(), Bitmap.Config.ARGB_8888);
            Utils.matToBitmap(tmp, image);
        }
        catch (CvException e){Log.d("Exception",e.getMessage());}

        String OCRresult = null;
        mTess.setImage(image);
        OCRresult = mTess.getUTF8Text();
```

Con esta función queda en la variable OCRresult del tipo String, todos los caracteres hallados. Alguno resultados de pruebas obtenidos se muestras en las siguientes figuras, las cuales son capturas de pantalla del celular mientras corría la aplicación:

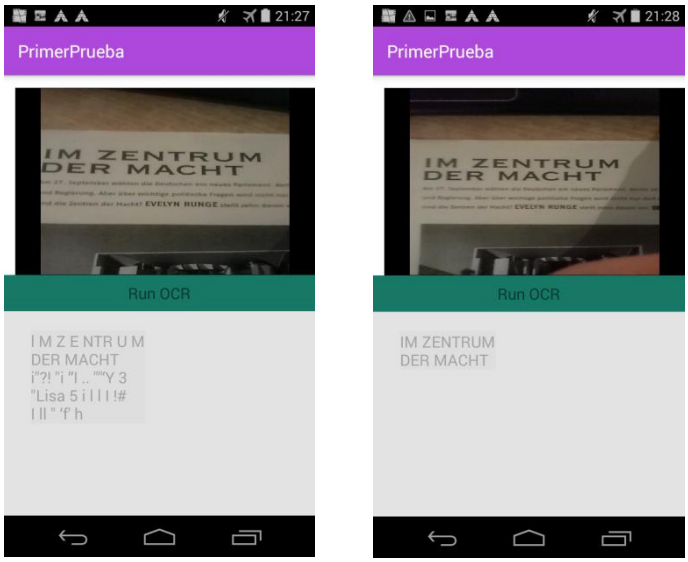


Fig 7: Resultados OCR

Una vez obtenido este resultado, se procede a la búsqueda de palabras específicas, cuyo acierto nos indicará que se encontró la palabra buscada. Para esto es necesario dividir la cadena de caracteres entregada por la función de Tesseract, lo cual se logra con la utilización de la función:

StringTokenizer(String str, String delim, boolean returnDelims).

Donde:

- *String str*: es la cadena a dividir. En nuestro caso, el resultado de la detección de Tesseract.
- *String delim*: carácter delimitador. La función dividirá la cadena original en cadenas mas chicas o subcadenas, y asignará a estas todo lo que se encuentre entre los caracteres delimitadores.
- *boolean returnDelims*: permite devolver los delimitadores como tokens. No es estrictamente necesario como parámetro.

Entonces, se crea un nuevo elemento del tipo StringTokenizer y se pasan los parámetros necesarios. A continuación se declaran dos cadenas, una para obtener los resultados de la división y otra, fija para realizar la comparación. Para realizar esto, se debe realizar una tarea que recorra la cadena original hasta el final comparando las cadenas. Para lograrlo se hace uso de tres funciones principales:

- *String nexToken()*: devuelve consecutivamente los resultados obtenidos de la división.
- *Boolean hasMoreTokens():* pregunta si hay mas tokens disponibles, por lo que devuelve *true* si quedan tokens por leer.

- *String equalsIgnoreCase (String str):* compara si las cadenas son iguales, ignorando las mayúsculas, por lo que es indistinto cómo se inicialice la variable a comprar.

Estas funciones fueron implementadas de la siguiente manera, alertando a la interfaz de que la comparación fue exitosa, por lo que se estaría en la presencia del cartel que se buscaba:

```
StringTokenizer tokens = new StringTokenizer(OCRresult, " ");
String second;
String third = "FARMACIA";

TextView Palab = (TextView) findViewById(R.id.Resul_COMP);

second=tokens.nextToken();// esta sentencia tiene que ir si o si,
// ya que sino el primer elemento es null
//lo cual es incorrecto y corresponde a la inicialización del String.

while (tokens.hasMoreTokens())
{

    if (second.equalsIgnoreCase(third))
    {
        Palab.setText("COINCIDEN: "+ second);
        //enviar a BT el caracter //
        contador=1;
    }
    else if(contador==0)
    {
        Palab.setText("NO coinciden");
        contador=0; //seguiría en 0 //no mandaría audio
    }
    second=tokens.nextToken();
}
```

❖ 2.5 Bluetooth en Android:

La plataforma de Android incluye compatibilidad con la pila de red Bluetooth, la cual permite que un dispositivo intercambie datos de manera inalámbrica con otros dispositivos Bluetooth. El framework de la aplicación proporciona acceso a la funcionalidad Bluetooth mediante las Android Bluetooth API. Estas API permiten a las aplicaciones conectarse de manera inalámbrica con otros dispositivos Bluetooth y habilitan las funciones inalámbricas punto a punto y de multipunto.

Con las Bluetooth API, una aplicación de Android puede realizar lo siguiente:

- ❖ Buscar otros dispositivos Bluetooth;
- ❖ Consultar el adaptador local de Bluetooth en busca de dispositivos Bluetooth sincronizados;
- ❖ Establecer canales RFCOMM;
- ❖ Conectarse con otros dispositivos mediante el descubrimiento de servicios;

- Transferir datos hacia otros dispositivos y desde estos;
- Administrar varias conexiones.

En este documento, se describe la manera de usar el sistema *Bluetooth clásico*.

Conceptos básicos:

Todas las Bluetooth API están disponibles en el paquete de *android.bluetooth*. A continuación, se ofrece un resumen de las clases e interfaces que necesitarás para crear conexiones Bluetooth:

- BluetoothAdapter: adaptador local de Bluetooth, es el punto de entrada de toda interacción de Bluetooth, permite visualizar otros dispositivos Bluetooth, consultar los dispositivos sincronizados, crear instancias de BluetoothDevice con dirección MAC conocida y crear un BluetoothServerSocket para oír comunicaciones de otros dispositivos.
- BluetoothDevice: representa un dispositivo Bluetooth remoto, se utiliza para solicitar una conexión Bluetooth mediante BluetoothSocket o consultar información sobre dispositivos, como nombre, dirección, clase y estado de conexión.
- BluetoothSocket: es la interfaz de un Socket de Bluetooth, el punto de conexión que permite a al aplicación intercambiar datos a través de InputStream y OutputStream.
- BluetoothServerSocket: un socket de servidor abierto recibe solicitudes entrantes. Es el encargado de mostrar un BluetoothSocket una vez que se haya establecido la conexión.
- BluetoothClass: conjunto de propiedades de solo lectura que define las clases de dispositivos y sus servicios.
- BluetoothProfile: interfaz inalámbrica para comunicación entre dispositivo. Ejemplo: Perfil de Manos Libres.

2.5.1 Permisos de Bluetooth:

Se requiere declarar el permiso de Bluetooth, necesario para cualquier tipo comunicación. Además se deberá declarar el permiso BLUETOOTH_ADMIN si la aplicación requiere de tareas específicas como detectar dispositivos o controlar los ajustes de Bluetooth; otras funciones que otorga este permiso sólo deberán ser utilizadas en caso de que la aplicación sea un “administrador de energía”.

```
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
```

❖ 2.5.2 Configuración de Bluetooth:

1. Obtener el *BluetoothAdapter*: el cual es obligatorio para toda actividad de Bluetooth, llamando al método estático getDefaultAdapter(). Este método devuelve el objeto *BluetoothAdapter* con el que la aplicación interactúa con el adaptador de Bluetooth del sistema. En caso de que devuelva *null* significa que el dispositivo no es compatible con Bluetooth.

```
BluetoothAdapter mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
if (mBluetoothAdapter == null) {
    // Device does not support Bluetooth
}
```

2. Habilitar Bluetooth: llamando al método *isEnable()* (true=habilitado; false=deshabilitado). Para solicitar la habilitación se deberá llamar a startActivityForResult() con un intent de acción, como se muestra en la siguiente porción de código; esto generará una solicitud de permiso para el usuario para que habilite Bluetooth (lo cual hace el mismo sistema).

```
if (!mBluetoothAdapter.isEnabled()) {
    Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
    startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
}
```

Donde:
REQUEST_ENABLE_BT: es un valor entero definido localmente que el sistema devuelve en la implementación de onActivityResult(), el cual debe ser mayor que cero.

❖ 3. Auricular inalámbrico:

Los resultados obtenidos por la aplicación móvil serán comunicados al usuario mediante alertas sonoras reproducidas por un auricular inalámbrico. El mismo, cuenta con tres elementos principales:

- ✓ Un módulo Bluetooth de Arduino (CZ-HC-05)
- ✓ Un Microcontrolador PIC 18F14K50
- ✓ Un módulo reproductor de MP3 de Arduino (DPFPlayer-Mini-MP3)

El esquema circuital implementado se ilustra a continuación:

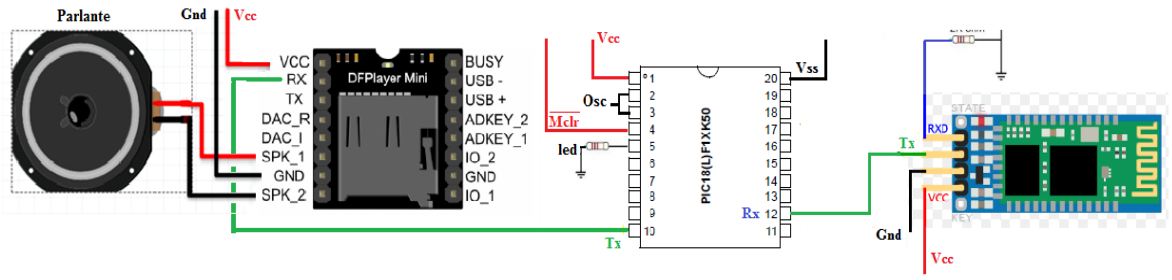


Fig 8: Diagrama Conexión Auricular

Cada uno de los elementos, así como la configuración adoptada para su funcionamiento, se describirán detalladamente a continuación:

❖ 3.1 Modulo Bluetooth CZ-HC-05:

Es un módulo Bluetooth fácil de utilizar el cual posee un protocolo de puerto serie, diseñado para una conexión serial wireless. Está calificado como Bluetooth V2.0+EDR1, modulación en 3Mbps con una tasa de transceptor y banda base de 2.4GHz.

Fig 9: Modulo Bluetooth CZ-HC-05

Especificaciones de Hardware:

- *Sensibilidad*: típica -80 dBm.
- *Potencia de transmisión* de RF de hasta +4dBm.

- *Operación en baja potencia*: 1.8V y puertos E/S de 3.3V y 5V.

El módulo Bluetooth viene configurando por defecto de la siguiente manera:

- ✓ Modo: esclavo.
- ✓ Nombre: HC-05.
- ✓ Velocidad (baud rate): 9600.
- ✓ Código de Verificación: 1234.

Este módulo tiene 3 estados posibles:

- *Estado DESCONECTADO*: estado inicial al poner en funcionamiento el módulo, sin haberse establecido ninguna conexión todavía. En este estado, el led parpadea a una velocidad de 2veces por segundo (XX Hz). El módulo no podrá interpretar los comandos AT encontrándose en este estado.
- *Estado CONECTADO*: cuando se establece una conexión con otro dispositivo; el led indicador hace un doble parpadeo 1 vez cada 2 segundos (YY Hz).
- *Modo AT*: Modo de configuración. Entra en este estado cuando se alimenta el pin "KEY" y luego se alimenta el módulo. Es en este estado en que módulo admite comandos AT2 para su configuración tanto como "Maestro" (o "Servidor") o como "Esclavo" (o "Cliente").

Para su configuración, puede consultarse el ANEXO C al final de este informe, donde se detalla las consideraciones a tener en cuenta, así como los comandos disponibles. Para este caso, se optó por la configuración a través del kit "Arduino UNO", para la realización de diferentes pruebas. Sin embargo, resultó óptima para esta aplicación, la siguiente configuración:

- ✓ Modo: esclavo.
- ✓ Nombre: HC-05.
- ✓ Velocidad (baud rate): 38400
- ✓ Código de Verificación: 1234

La velocidad de transmisión se refiere a la velocidad en que el módulo Bluetooth comunica los datos recibidos al Microcontrolador, a través de la UART. Se optó por un Baud Rate de 38400, por motivos de simplicidad a la hora de pasar del modo configuración al modo funcional.

❖ 3.2 Microcontrolador Microchip PIC 18F14K50:

Este Microcontrolador de 20 Pines, cuenta con 14 pines de E/S, con funcionalidad en 5.5V oscilador interno de hasta 16Mhz, conversor ADC de 10 bit de resolución, interrupciones externas programables, y una USART mejorada la cual soporta RS-485 y RS-232, entre otros parámetros.

El Pin-out del mismo se ilustra a continuación:

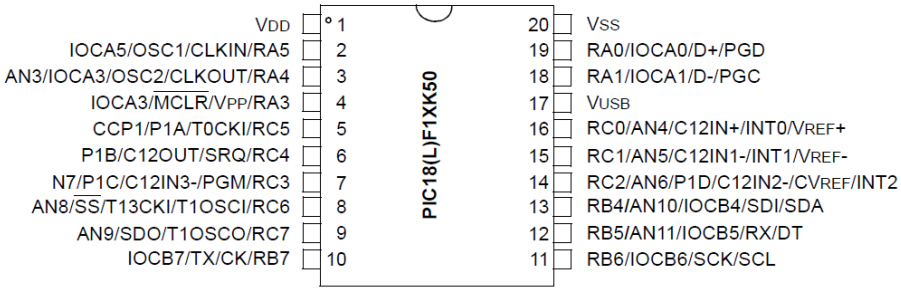


Fig 10: Pin-out microcontrolador

Los datos que recibe el módulo Bluetooth son enviados al microcontrolador, el cual luego de recibirlo y procesarlo, envía al reproductor de MP3 la configuración adecuada para que la alerta sea emitida. La comunicación entre estos dos últimos, se realizó mediante USART. Para la configuración de la USART del PIC, se utilizaron los siguientes registros:

- *Recepción:*

Nombre	n° de Bit	Registro	Estado	Acción
SPBRGH	-	SPBRGH	0	Configuran BaudRate
SPBRG	-	SPBRG	25	
BRGH	2	TXSTA	1	Determinan el multiplicador de BaudRate
BRG16	3	BAUDCON	0	
SPEN	7	RCSTA	1	Habilita el Puerto Serial
TRISB5	5	TRISB	1	Habilita el Puerto Como Entrada
CREN	4	RCSTA	1	Habilita la Recepción
SYNC	4	TXSTA	0	Recepción Asíncrona
RCIE	5	PIE1	0	Sin Interrupción Recepción
RCIF	5	PIR1	X	Bandera de Recepción
RCREG	-	RCREG	x	Guarda el Dato Recibido

Tabla

Para la obtención de los valores correspondientes a los registros de multiplicador de BaudRate, se utilizó la fórmula y recomendaciones proporcionadas en la hoja de datos:

TABLE 16-3: BAUD RATE FORMULAS

Configuration Bits			BRG/EUSART Mode	Baud Rate Formula
SYNC	BRG16	BRGH		
0	0	0	8-bit/Asynchronous	Fosc/[64 (n+1)]
0	0	1	8-bit/Asynchronous	Fosc/[16 (n+1)]
0	1	0	16-bit/Asynchronous	
0	1	1	16-bit/Asynchronous	Fosc/[4 (n+1)]
1	0	x	8-bit/Synchronous	
1	1	x	16-bit/Synchronous	

Legend: x = Don't care, n = value of SPBRGH, SPBRG register pair

Tabla

Para el caso de este proyecto, donde se cuenta con un oscilador de 4MHz, se obtuvo:

$$SPBRG = \left(\frac{4MHz}{9600 \cdot 16} \right) - 1 \Rightarrow SPBRG = 25$$

- *Transmisión:*

Nombre	n° de Bit	Registro	Estado	Acción
SPBRGH	-	SPBRGH	0	Configuran BaudRate
SPBRG	-	SPBRG	25	
BRGH	2	TXSTA	1	Determinan el multiplicador de BaudRate
BRG16	3	BAUDCON	0	
SPEN	7	RCSTA	1	Habilita el Puerto Serial
TRISB7	7	TRISB	0	Habilita el Puerto Como Salida
TXEN	5	TXSTA	1	Habilita la Transmisión
SYNC	4	TXSTA	0	Recepción Asíncrona
TXIE	4	PIE1	0	Sin Interrupción Recepción
TXIF	4	PIR1	X	Bandera de Transmisión
TXREG	-	TXREG	x	Guarda el Dato a ser Enviado

Tabla

❖ 3.3 Reproductor de MP3:

Se utilizó el módulo de reproducción DFPlayer Mini, el cual provee decodificación de MP3 y WMV, soporta formatos de archivos FAT16 y FAT32. Se puede configurar y manejar mediante comandos serie o haciendo uso de la librería de Arduino. Es fácil de usar, estable y confiable.

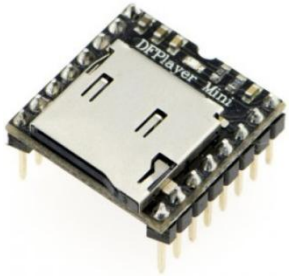


Fig 12: Módulo Reproductor de MP3

Características:

- Soporta decodificación MP3 y WMV.
- Posee tasa de muestreo desde 8kHz hasta 48kHz.
- Puede incluirse una tarjeta de memoria MiniSD de hasta 32GB.
- Variedad en modos de control: Comandos Serie o AD Key.
- Cuenta con un amplificador de 3W.
- Nivel de volumen ajustable: de 0 a 30.

Item	Description
MP3Format	1、Support 11172-3 and ISO13813-3 layer3 audio decoding
	2、Support sampling rate (KHZ):8/11.025/12/16/22.05/24/32/44.1/48
	3、Support Normal、Jazz、Classic、Pop、Rock etc
UART Port	Standard Serial; TTL Level; Baud rate adjustable(default baud rate is 9600)
Working Voltage	DC3.2~5.0V; Type :DC4.2V
Standby Current	20mA
Operating Temperature	-40~+70
Humidity	5% ~95%

Tabla X: características de módulo MP3

El Pin-Out del reproductor es el siguiente:

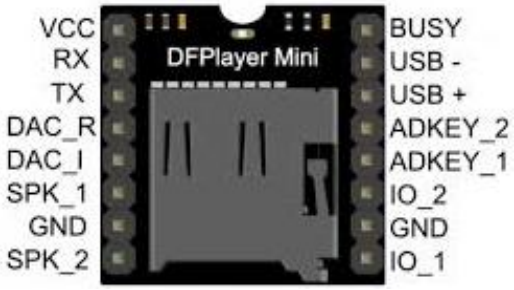


Fig 11: Pin-Out de módulo reproductor MP3.

En esta aplicación, se utilizaron comandos series y una tarjeta de memoria MiniSD.

Una vez configurado el módulo USART el microcontrolador debe enviar al reproductor su configuración inicial y esperar un momento a su correcta configuración. Los comandos serie deben tener un formato determinado, a continuación se ilustra en la siguiente tabla.

Formato: SS VER Len CMD Feedback Param1 Param2 Checksum S0		
SS	Bit de Arranque	0x7E
VER	Versión	0xFF
Len	Cantidad de Bits despues de Información	No se cuentan los bytes de Checksum
CMD	Comandos	Especifica operación a realizar
Feedback	Comando de Feedback	1:Feedback ; 0:Sin Fb.
Param1	Parametro 1	Parametros que dependen del comando
Param2	Parametro 2	
Checksum	Verificación	Suma de Verificacion
S 0	Bit de Stop	0xEF

Tabla X: formato de Comando Serie de módulo MP3

La lista de comando se encuentra detallada en la hoja de datos del reproductor:

CMD	Function Description	Parameters(16 bit)
0x01	Next	
0x02	Previous	
0x03	Specify tracking(NUM)	0-2999
0x04	Increase volume	
0x05	Decrease volume	
0x06	Specify volume	0-30
0x07	Specify EQ(0/1/2/3/4/5)	Normal/Pop/Rock/Jazz/Classic/Base
0x08	Specify playback mode (0/1/2/3)	Repeat/folder repeat/single repeat/ random

Tabla X: Comandos Serie de módulo MP3

Sin embargo, el comando de inicialización no se encuentra especificado en la hoja de datos, como tampoco se encuentra el método utilizado para realizar la *suma de verificación*, lo cuales fueron codificados en el programa:

```
//configuracion del reproductor:
data[0]=0x7E;  //bit de comienzo
data[1]=0xFF;  //Informacion de Version
data[2]=0x06;  //LEN = Numero de bits despues de LEN sin contar los checksum
data[3]=0x09;  //CMD = comando que yo especifico de acuerdo a lo que tiene que hacer el reproductor
data[4]=0x00;  //sin feedback
data[5]=0x00;  //Param1; //parametro 1 de acuerdo a comando
data[6]=0x02;   //Param2; //parametro 2 de acuerdo a comando  02=TF=tarjeta de memoria
data[7]=0xFE;   //checksum1;
data[8]=0xF0;  //checksum2;
data[9]=0xEF;  //Bit de Stop
```

```
uint16_t mp3_get_checksum (uint8_t *thebuf)
{
    uint16_t sum = 0;
    for (int i=1; i<7; i++)
    {
      sum += thebuf[i];
    }
    return -sum;
}
```

Dentro de la memoria SD se encuentran las pistas correspondientes para los eventos encuestados. Las pistas corresponden a:

"0001": Estación de Subte.
"0002": Farmacia.
"0003": No se encuentra
"0004": Semáforo Verde.
"0005": Semáforo Verde.

La selección del tema a reproducir se realizó de la siguiente forma:

```
if(recib=='f')//si se recibe una "f" se pende un led amarillo en pin15 (RC1)
 {
   data[3]=0x03; //CMD = 03 Especifico el tema que tiene que reproducir
   data[5]=0000; //Param1
   data[6]=0003;  // Param2= al segundo tema
   checksum = mp3_get_checksum(data);
   data[7]=(uint8_t)(checksum >> 8);
   data[8]=(uint8_t)(checksum);
   LATCbits.LATC1=1;
   __delay_ms(190);
   LATCbits.LATC1=0;
 }
```

❖ 4. Implementación de OpenCV en la PC:

Previamente al desarrollo en Android, se realizaron diversas pruebas en la PC, haciendo uso de diferentes funciones, con la utilización de una cámara web externa o en su defecto, la cámara web incorporada. El procesamiento de imágenes fue realizado en lenguaje C, implementando la librería OpenCV en su versión 2.4.9.

Se lograron principalmente rutinas de detección de **caracteres y palabras** aplicando diferentes umbrales de filtrado y detección de **círculos** en imágenes de carteles indicadores, tales como carteles de farmacias y de estaciones de subte.

A continuación se detallan las funciones y parámetros utilizados, seguido de la ilustración de algunos de los resultados obtenidos:

Imágenes: es importante destacar que las imágenes son tratadas como una matriz, por lo que deben ser declaradas de esta manera, haciendo uso del tipo de variable *Mat.* Ejemplo: *Mat imagen1*.

- **imread**: esta función carga una imagen desde un archivo. Si la imagen no puede ser leída, por falta del archivo, formato inválido, etc., devuelve una matriz vacía (NULL). Los parámetros necesarios para hacer uso de esta función son:

*Mat imread(**filename**, int **flags**=1)*

Donde *int **flags**=1* puede obviarse. Cabe aclarar que si la imagen que se pretende abrir se encuentra en el mismo directorio del proyecto creado, solo debe colocarse el nombre de la misma junto con su extensión entre comillas dobles; de lo contrario, deberá especificarse la ruta de acceso a la imagen.

- **cvtColor**: esta función convierte una imagen de entrada en RBG a otra en otro orden de colores o a grises. Sus parámetros son:

*void cvtColor(InputArray **src**, OutputArray **dst**, int **code**, int **dstCn**=0)*

Donde:

- ✓ *InputArray **src:*** representa el puntero de la imagen original previamente declarada.
- ✓ *OutputArray **dst:*** representa el puntero de la imagen destino, también previamente declarada; cabe destacar que puede utilizarse la misma variable como **scr** y **dst**, sobreescribiendo la imagen original como consecuencia.
- ✓ *int **code:*** código de conversión de espacio de color;
- ✓ *int **dstCn**=0:* representa el número de canales en la imagen destino; si el parámetro es 0 (cero) el número de canales se deriva de ***src*** y ***code*** automáticamente.

La función convierte una imagen de un espacio de color a otro. En caso de transformaciones desde y a RGB, los canales deben ser especificados explícitamente (RGB o BGR, siendo este último el formato color por defecto). El primer byte en una imagen de color estándar será un componente AZUL de 8-bits, el segundo byte será el componente VERDE y el tercero, el ROJO. Luego, el cuarto, quinto y sexto byte serán del segundo pixel y así sucesivamente.

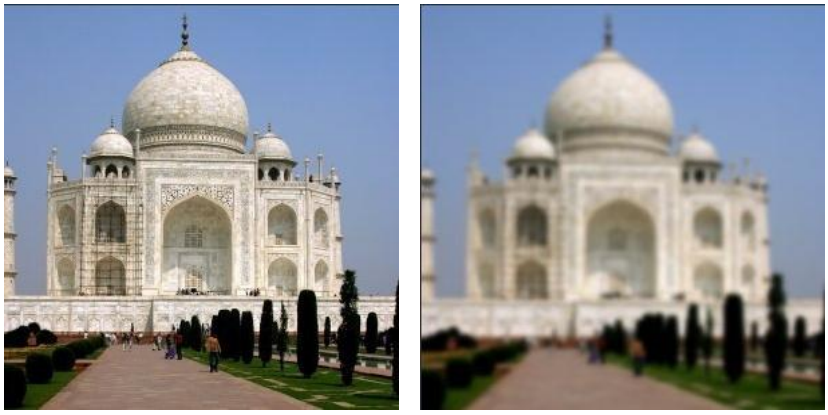
Los rangos convencionales para los canales R, G y B son:

- De 0 a 255 para imágenes *CV_8U*
- De 0 a 65535 para imágenes *CV_16U*
- De 0 a 1 para imágenes *CV_32F*

Para transformaciones lineales, los rangos no se ven afectados. Sin embargo, para transformaciones no lineales, se debe escalar la imagen previamente.

Al usar cvtColor en imágenes de 8 bits, la conversión perderá cierta información, la cual para la mayoría de las aplicaciones no perceptible; es recomendable utilizar de imágenes de 32 bits en aplicaciones que precisen el rango completo de color

- **GaussianBlur**: suaviza la imagen, aplicando un filtro Gaussiano. El efecto resultante es un difuminado, utilizado ampliamente para la eliminación de ruido y de detalles. Matemáticamente, aplicar *GaussianBlur* equivale a la convolución de cada punto del vector imagen con una función gaussiana, eliminando las componentes de alta frecuencia, análogamente a un filtro pasabajos.



Original *GaussianBlur*

Fig 12: Resultados de Filtro GaussianBlur.

Los parámetros que utiliza son los siguientes:

*void GaussianBlur(InputArray **src**, OutputArray **dst**, Size **ksize**, double **sigmaX**, double **sigmaY**=0, int **borderType**=**BORDER_DEFAULT**)*

Donde:

- ✓ *InputArray* ***src****:* imagen de entrada, representa el puntero de la imagen original previamente declarada; puede tener cualquier número de canales, los cuales son procesados independientemente, pero su profundidad deberá ser CV_8U, CV_16U, CV_16S, CV_32F or CV_64F.
- ✓ *OutputArray* ***dst****:* imagen de salida, representa el puntero de la imagen destino, también previamente declarada; cabe destacar que puede utilizarse la misma variable como **scr** y **dst**, sobreescribiendo la imagen original como consecuencia. Presenta el mismo tipo y tamaño que la imagen original.
- ✓ *Size* ***ksize****:* tamaño del núcleo gaussiano. Ksize.width y ksize.height pueden diferir, pero ambos deben ser positivos e impares.
- ✓ *double* ***sigmaX****:* Desviación estándar del núcleo Gaussiano en dirección X.
- ✓ *double* ***sigmaY=0****:* Desviación estándar del núcleo Gaussiano en dirección Y. En caso de ser 0, es seteado para ser igual a sigmaX.
- ✓ *int* ***borderType=BORDER_DEFAULT****:* método de exploración de píxeles.

- **HoughCircles**: esta función se encarga de encontrar un círculo en una imagen con escala de grises, utilizando la transformada de Hough. Dicha transformada es una técnica de detección de figuras dentro de una imagen considerando las relaciones globales entre píxeles de borde permitiendo encontrar ciertos patrones en la imagen como líneas y círculos, realiza agrupaciones de los puntos que pertenecen a los bordes de posibles figuras a través de un procedimiento de votación sobre un conjunto de figuras parametrizadas, para detectarlas aun cuando la imagen presente imperfecciones o desviaciones espaciales que hacen que falten puntos de la figura.

Sin embargo, *HoughCircles* no presenta inconvenientes para detectar el centro de los círculos, pero sí para encontrar el radio perfecto, por lo que es recomendable especificar un rango de radios probables (*minRadius y maxRadius*) dentro de lo posible.

Fig 13: Resultado Obtención de Círculos.

La función y sus parámetros asociados son:

void ***HoughCircles****(InputArray* ***image****, OutputArray* ***circles****, int* ***method****, double* ***dp****, double* ***minDist****, double* ***param1****=100, double* ***param2****=100, int* ***minRadius****=0, int* ***maxRadius****=0)*

- ✓ *InputArray* ***image:*** imagen de entrada. Debe ser de 8 bits, canal simple, en escala de grises.
- ✓ *OutputArray* ***circles:*** vectores de salida de círculos encontrados. Cada vector está codificado como un vector puntero a flotante de 3 elementos, los que corresponden a la posición del centro (X, Y) y el radio del círculo.
- ✓ *int* ***method:***método de detección a usar. Actualmente el único método implementado es `CV_HOUGH_GRADIENT`.
- ✓ *double* ***dp:*** taza inversa del acumulador de resolución a la resolución de la imagen. Por ejemplo si dp=2, entonces el acumulador tiene la mitad de tamaño del ancho y alto original. Si dp=1, cuenta con la misma resolución que la imagen de entrada.
- ✓ *double* ***minDist:*** distancia mínima entre el centro del círculo detectado. Si el parámetro es muy pequeño, múltiples círculos vecinos pueden ser falsamente detectados además del verdadero o correcto. Sin embargo, si este parámetro es muy grande, la detección de algunos círculos puede perderse.
- ✓ *double* ***param1****=100:* límite superior para el detector de bordes interno *Canny*.
- ✓ *double* ***param2****=100:* representa el límite del acumulador para la detección del centro de los círculos. Cuanto más pequeño sea este valor, mayor cantidad de falsos círculos serán detectados. Los círculos correspondientes a los valores mayores del acumulador serán devueltos primero.
- ✓ *int* ***minRadius****=0:* mínimo radio a ser detectado. En caso de ser desconocido, deberá colocarse 0 (cero) por defecto.
- ✓ *int* ***maxRadius****=0:* máximo radio a ser detectado. En caso de ser desconocido, deberá colocarse 0 (cero) por defecto.

- **Threshold**: aplica un umbral de nivel fijo a cada elemento de la matriz. Si el valor de un pixel está por encima del umbral, se le asigna un valor correspondiente al blanco, de otra marea el valor asignado corresponderá al negro. La imagen a filtrar ya debe encontrarse en escala de grises. Por esta razón, generalmente es utilizada para obtener una imagen binaria a partir de una en escala de grises o para remover el ruido en la imagen.

OpenCV ofrece diferentes tipos de umbrales:

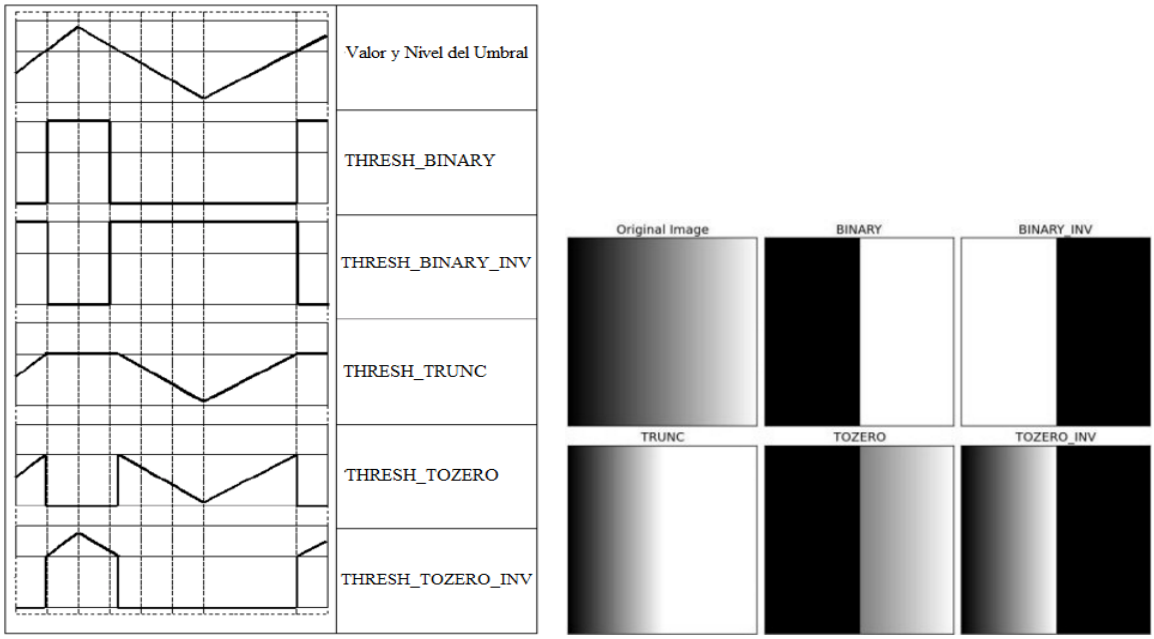


Fig 14:Umbrales de Threshold.

Se realizaron pruebas aplicando las diferentes umbralizaciones y se comprobó que para la detección de texto en imágenes el método más exitoso es "THRESH_TOZERO". Debido a que las imágenes presentan diferencias de exposición de luz, enfoque, dirección, etc. no es posible determinar un umbral mínimo fijo, por lo que se optó por emplear un bucle "for" en el cual se aumenta gradualmente el umbral mínimo, encontrando el umbral óptimo para la lectura en cada una de las imágenes.

Algunos **resultados obtenidos** con la implementación de estas funciones se ilustran a continuación:

- **4.1Resultados Detección de Círculos:**

Original

Circulo

Original *Circulo*

❖ 4.2 Detección de "Subte":

Original *THRESH_TOZERO (0-255)*

Original *THRESH_TOZERO (200-255)*

❖ 4.3 Detección de "Farmacia":

Original

THRESH_TOZERO (50-255)

Original

THRESH_TOZERO (0-255)

❖ 5. Conclusiones:

La realización de este proyecto me abrió las puertas a un mundo de oportunidades como el que presenta el procesamiento de imágenes. La información contenida en una imagen es basta y puede ser utilizada en una amplia gama de aplicaciones como pueden ser sistemas de seguridad, conteo de elementos, personas o vehículos, etc. Actualmente, esto es cada vez más factible y realizable gracias a la implementación de las librerías OpenCV y Tesseract, las cuales representan dos de las más potentes herramientas de programación.

Algo similar ocurre con la plataforma de programación Android, de la cual conocía muy poco antes de trabajar en este proyecto. Hoy en día pueden encontrarse aplicaciones de Android no solo en telefonía móvil y tablets, sino que ya son de masivo conocimiento los televisores y relojes inteligentes; mientras que "Android Auto", una aplicación que incorpora GPS y comandos por voz.

Sin embargo, me resultó arduo el aprendizaje en la utilización de estas herramientas a lo largo del proceso, ya que pocas veces pude resolver errores o detalles consultando la documentación o tutoriales que ofrece el sitio oficial de estas herramientas. Hay una total escasez de guías oficiales sobre la compilación previa de las librerías, lo cual fue posible realizar mediante videos tutoriales y consultas en foros disponibles en la web, que en su gran mayoría se encuentran en un elevado nivel de inglés técnico. Esto representa una desventaja, bajo mi punto de vista, ya que llega a resultar un largo proceso engorroso. Por ello, traté de documentar, detallar y aclarar lo mejor posible en el presente informe, para ser utilizado como guía al momento de continuar con esta idea.

Si bien, en un principio la idea de implementar el proyecto en un teléfono celular pareció óptimo, las políticas de seguridad de peticiones de permisos, como por ejemplo en Bluetooth, me generaron conflictos para esta aplicación en particular, ya que a una persona no vidente se le dificulta otorgar dichos permisos a través de la pulsación de botones ubicados en lugares específicos de la pantalla o el dispositivo.

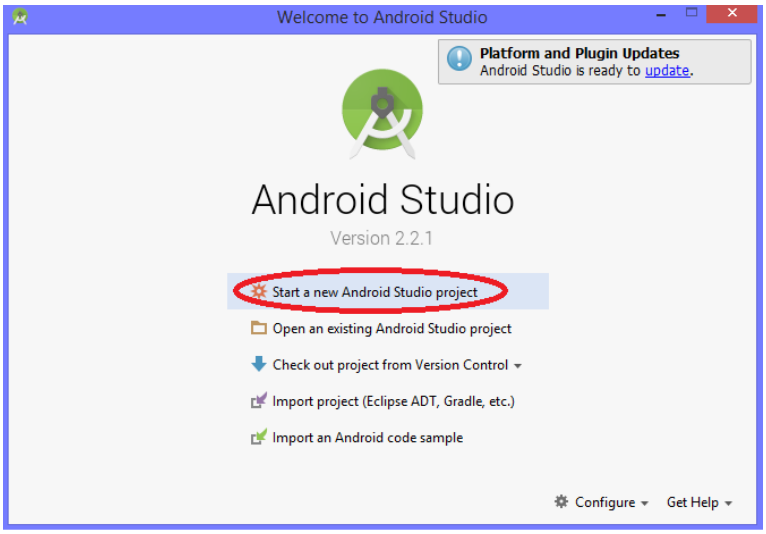
Esto representa un desafío para el futuro desarrollo y optimización de este proyecto, así también como la implementación de comandos por voz y la orientación a través de GPS para distancias mayores. Lo que haría que este proyecto sea más flexible y útil en otras situaciones adicionales a las resueltas hasta ahora.

Creo que con la realización de este proyecto pudieron y fueron implementados diversos conceptos adquiridos a lo largo de toda la carrera de Ingeniería Electrónica cursada en la Facultad Regional del Neuquén de la Universidad Tecnológica Nacional, la cual brinda una amplia visión e instrucción sobre el campo de acción que posee un Ingeniero Electrónico

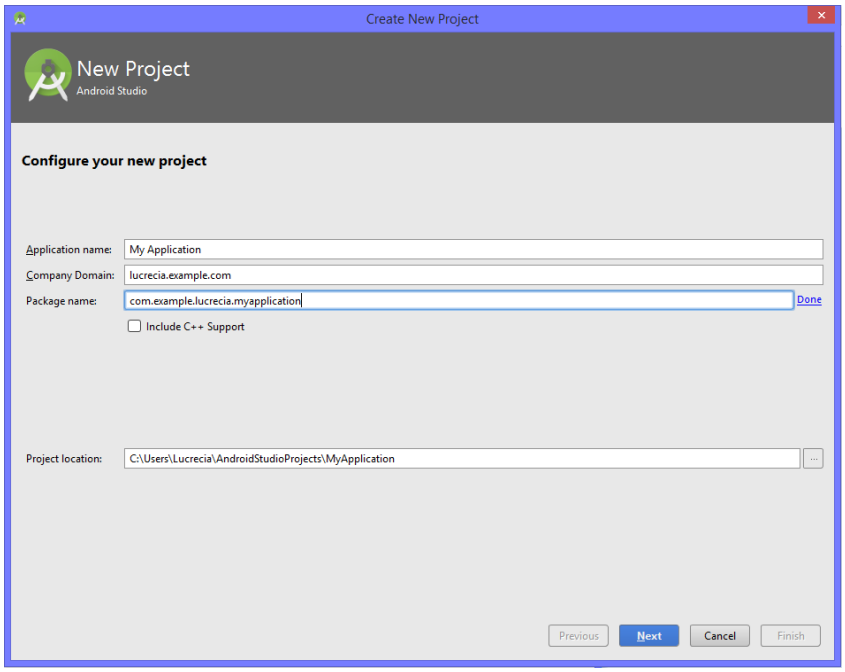
ANEXOS:

ANEXO A: Creando una Aplicación Android Studio:

- En la pantalla de inicio de "Android Studio" cliquear en "Start a New Android Studio Project" (empezar un nuevo proyecto de Android Studio).



- Ingresar un ***Nombre*** (Application Name) para la aplicación, el cual será mostrado en la pantalla de inicio de la aplicación. Luego, colocar el "***Dominio de la Compañía***" (Company Domain), el cual será usado para generar el "***Nombre del Paquete***" (Package Name) de la aplicación y es una cadena global única utilizada para identificar tu aplicación en las tiendas de aplicaciones y en dispositivos de Android.

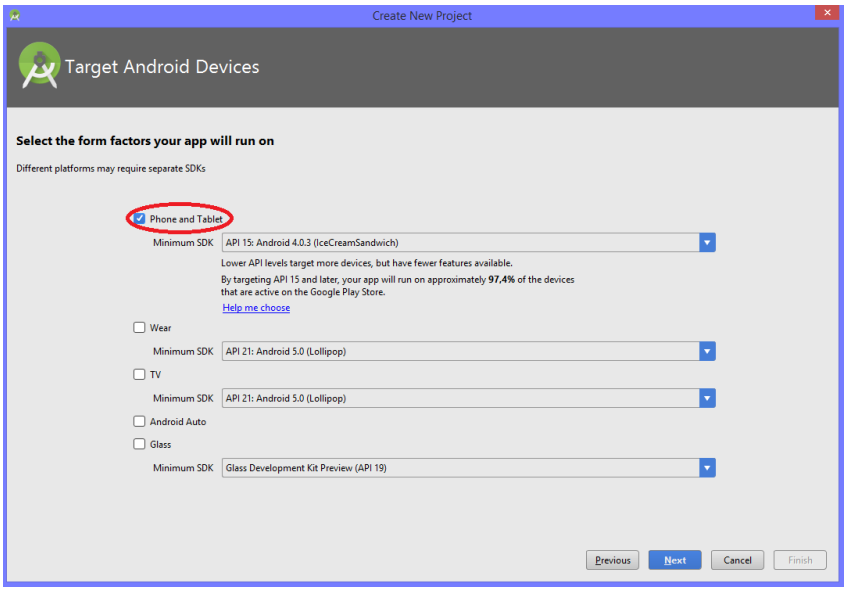


El Paquete de Aplicación está compuesto por el Dominio de Compañía en sentido inverso, más una cadena con el nombre propio de la aplicación. Sin embargo, éste puede ser modificado, cliqueando en la opción "Edit" en el costado derecho.

Al colocar el mismo nombre al Paquete para todas las aplicaciones, luego al ser cargadas en el dispositivo móvil, éstas serán automáticamente reemplazadas (las que tengan el mismo nombre) sin necesidad de tener que borrar numerosas aplicaciones obsoletas.

En la parte inferior se encuentra la locación del Proyecto en nuestro directorio por defecto, el cual también puede ser modificado. Luego, cliqueamos Next.

- A continuación se debe elegir dónde van a correr las aplicaciones. Por defecto, la opción "Teléfonos y Tablets" ya está chequeada.



"Minimun SDK" determina el límite inferior de versión de Sistema Operativo de Android en que podrá correr la aplicación. Por defecto se encuentra "API 15 Android 4.0.3 (IceCreamSandwich)", el cual no modificaremos (como se recomienda). Luego, cliqueamos Next.

La máxima SDK es seteada por defecto en la última versión de Android (actualmente Android 7.0 "Nougat" correspondiente a un API 24). Esto puede ser modificado luego en el archivo *"build.gradle"* del módulo de la aplicación.

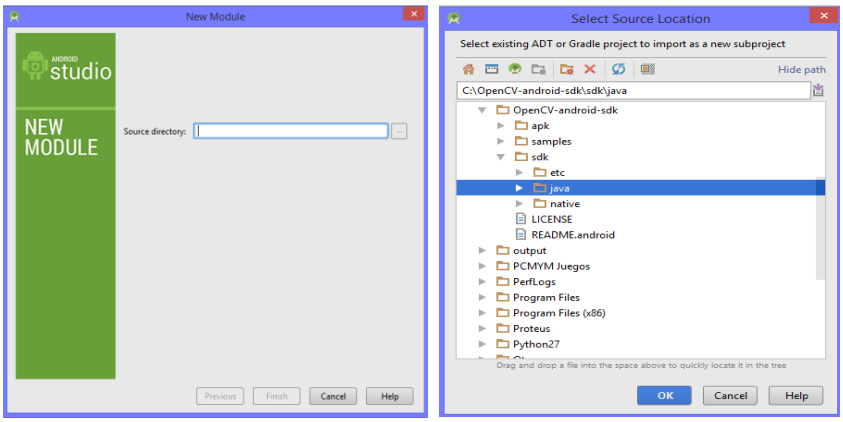
ANEXO B: Agregar Opencv y Tesseract en Android Studio.

OpenCV:

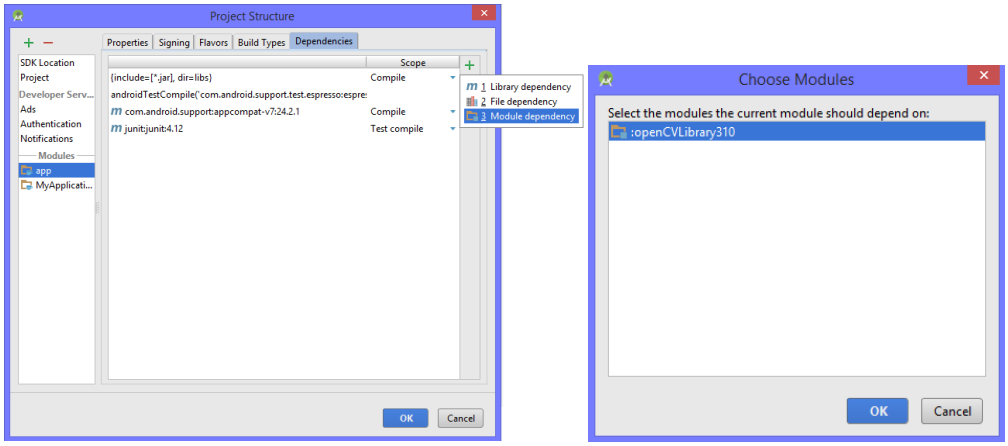
Para agregar la librería encargada del procesamiento de imágenes, pueden seguirse los siguientes pasos:

❖ Descargar de la página oficial (www.opencv.org) la última versión del SDK de "OpenCV para Android". En caso de no encontrarse en la página principal, puede encontrarse en "Downloads" (Descargas). Luego, se debe descomprimir esta carpeta.

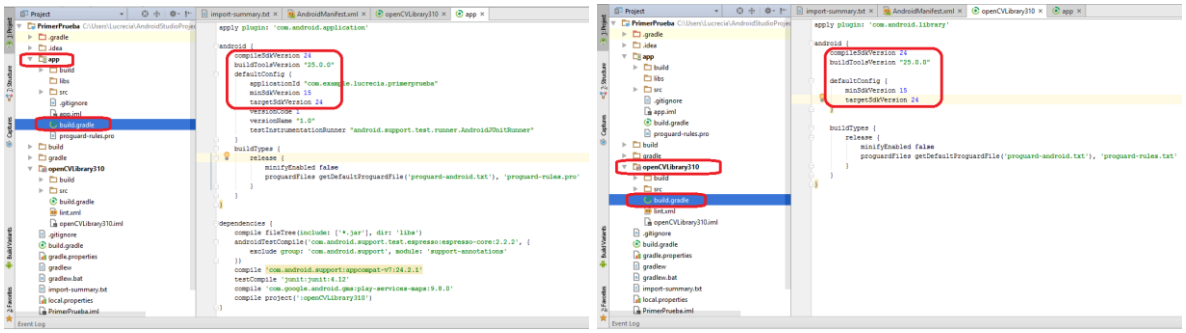
❖ En Android Studio se debe crear un Nuevo Proyecto en Android Studio cómo ya se explico en (AGREGAR NOMBRE DE ANEXO). Ir al menú "File", elegir la opción "New"→"Import Module". Esto abrirá la siguiente pantalla, donde elegiremos la ruta donde se encuentra la carpeta "Java", dentro de "sdk" de la carpeta descomprimida que descargamos anteriormente:



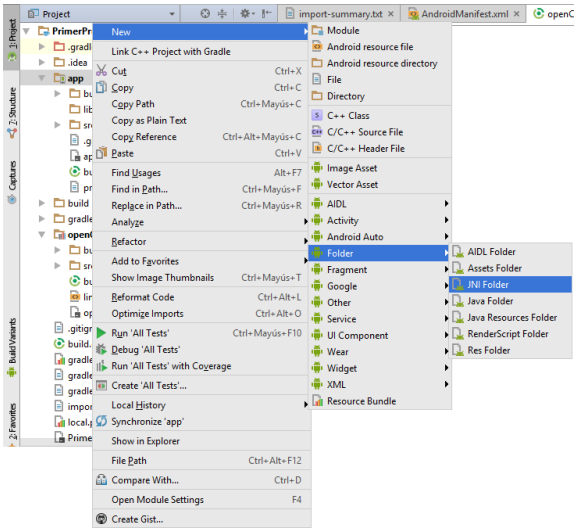
❖ Luego, declaramos las Dependencias. Para ello, vamos nuevamente al menú **File** y elegimos la opción **Project Structure**. En la pantalla emergente, seleccionando "**app**" en el cuadro de la izquierda, seleccionamos la pestaña "**Dependencies**". Luego de cliquear en el símbolo "+" agregaremos "**Module Dependency**". En la ventana emergente, seleccionamos "*openCVLibraries310*" (en este caso), damos click en **Ok**. Luego, dejamos la opción de "**Compile**" que viene por defecto y nuevamente aceptamos (click en Ok).



❖ **IMPORTANTE**: Una vez hecho lo anterior, debemos modificar las versiones de SDK con las que se compilará la librería OpenCV, de lo contrario, no compilará (o lo hará con un SDK diferente) y producirá errores. En la vista "Project" debemos ir a la pestaña desplegar la carpeta "app" y "openCVLibrary310" y abrir ambos archivos "**build.gradle**". A continuación se ilustran ambas pestañas. Es muy importante que ambos tengan los mismo valores de "Target SDK" y "Build tool Version":

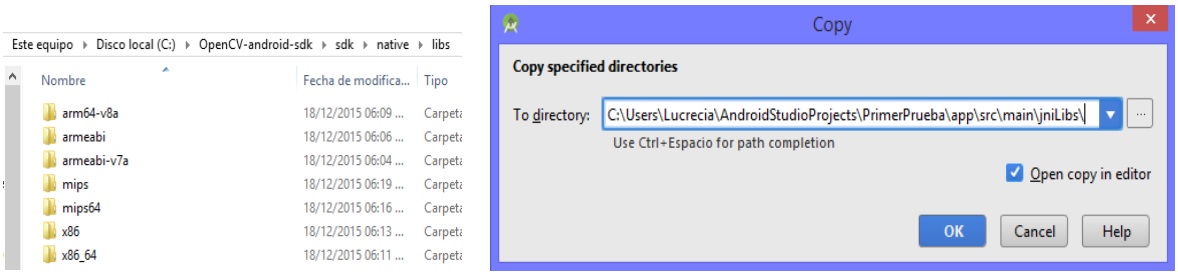


❖ Luego, debemos crear una nueva carpeta de JNI. Para ello, en la pantalla de Android Studio, en la vista "Android" en el recuadro izquierdo de la pantalla, hacemos click derecho sobre "app" y elegimos: New →Folder→JNI Folder:



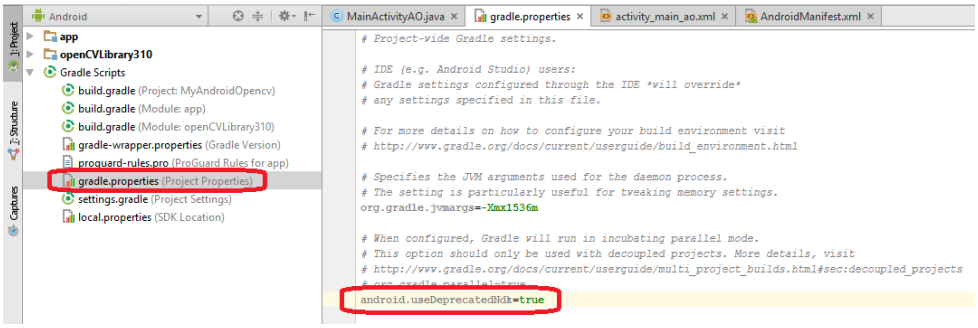
En la ventana emergente, habilitamos la opción "**Change Folder Location**" y cambiamos el nombre a "**JNILibs**".

❖ Nos dirigimos a la carpeta que descargamos "OpenCVLibraries310", entramos a **"sdk"→ "native"→ "libs"**, seleccionamos todos los elementos que contiene y los copiamos. Luego, volvemos a Android Studio, nuevamente hacemos click derecho en "app" como antes y elegimos "Paste"; como se muestra a continuación:



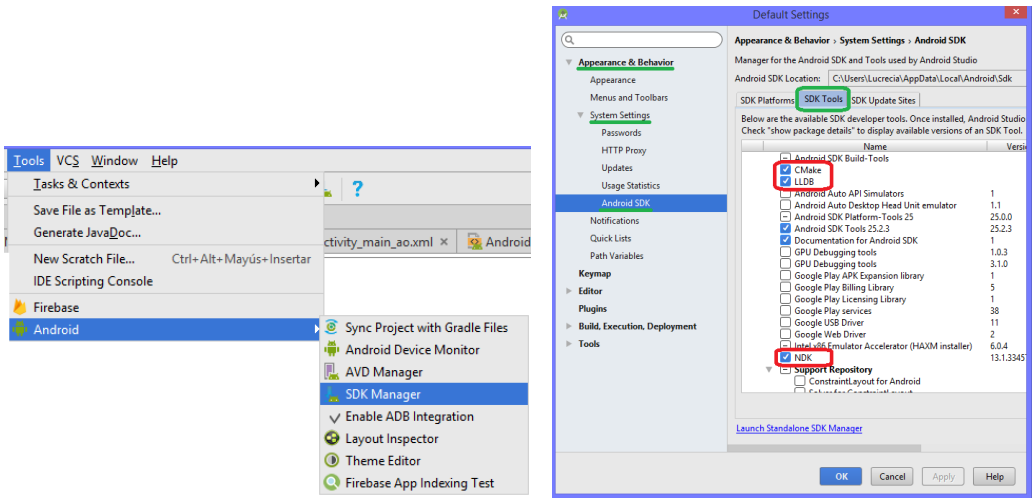
❖ Por último, debemos agregar una instrucción para compilar la librería. Abrimos la pestaña "**gradle.properties**" y escribimos la siguiente instrucción:

"android.useDeprecatedNdk=true"

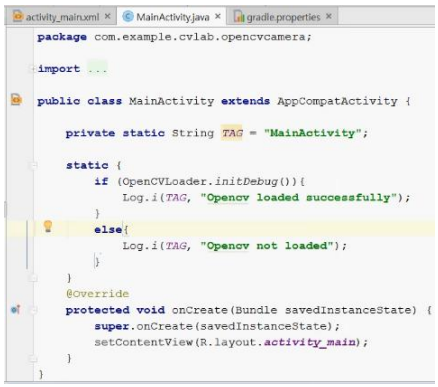


Esto utilizará las NDK obsoletas para compilar.

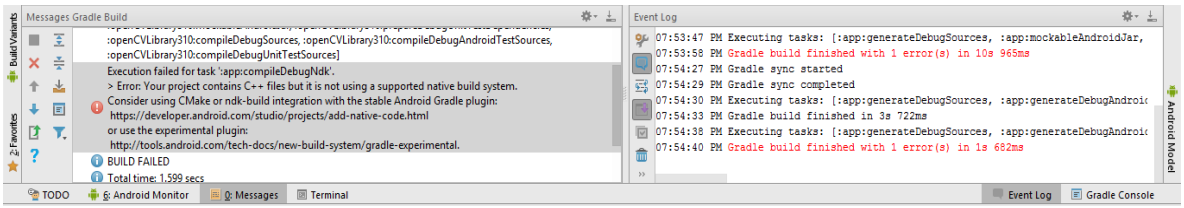
❖ En caso de ser la primera vez que se compile una librería externa, se deberá instalar las herramientas requeridas por Android Studio. Tales son: Cmake, LLDB, NDK las cuales pueden encontrarse en el menú **Tools→Android→SDK Manager**:



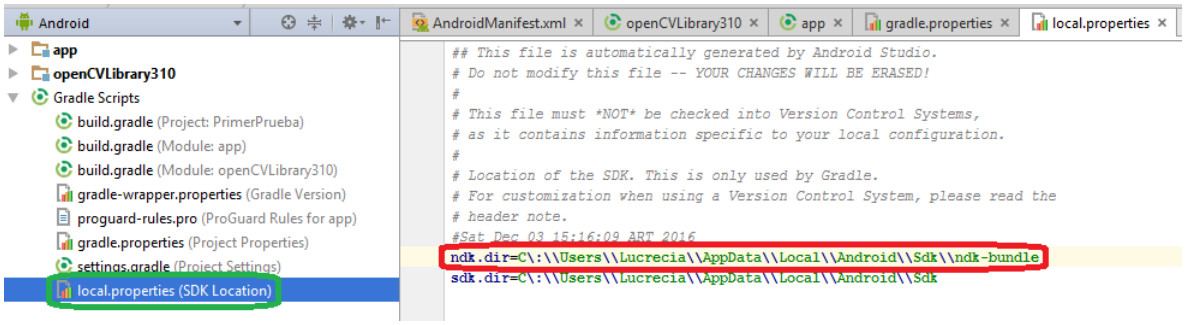
➢ Opcionalmente, se puede escribir la siguiente rutina en el archivo "MainActivity.java" para estar seguros de que la librería se cargó con éxito:



- En caso de que aparezca el siguiente error al querer compilar el proyecto:



Se deberá corroborar la correcta escritura de la instrucción: "android.useDeprecatedNdk=true" en el archivo "gradle.properties". Además, luego de la instalación de las herramientas requeridas anteriormente mencionadas, por defecto debe haberse agregado la siguiente ruta en el archivo "local.properties" de la pestaña "Gradle Scripts":



En caso de no ser así, asignar a la variable "ndk.dir" la ruta que lleva a los "ndk-bundle", como se ve en la figura anterior.

Tesseract:

Para el caso de la librería encargada del reconocimiento óptico de caracteres, se deberá construirla de manera tal que posteriormente pueda ser compilada por el sistema Gradle, y por ende, dentro de Android. Para ello deben cumplirse los siguientes pasos:

- En caso de no contar con los NDK anteriormente utilizados, descargar el paquete correspondiente, desde https://developer.android.com/ndk/downloads/index.html.

- Se utilizará el "Fork of Tesseract for Android" disponible en GitHub (https://github.com/rmtheis/tess-two), el cual contiene las funciones a compilar. Para esto, en modo consola (CMD) se deberán ejecutar los siguientes comandos:
  - git clone git://github.com/rmtheis/tess-two tess
  - cd tess
  - cd tess-two
  - ndk-build
  - android update project –path .
  - ant realise.

  Esto tomará varios minutos.
  El primer paso puede ser omitido en caso de haber descargado previamente el paquete desde Github. Solo hay que tener en cuenta la ruta en que se encuentra dicha carpeta, para poder acceder desde CMD.
  No hace falta compilar también "tess-two".

- Posteriormente, se debe copiar la carpeta "tess/tess-two" en la carpeta principal del proyecto de la aplicación. (ejemplo: suponiendo que la aplicación se llame "MiApp", copiar la carpeta en "MiApp/").

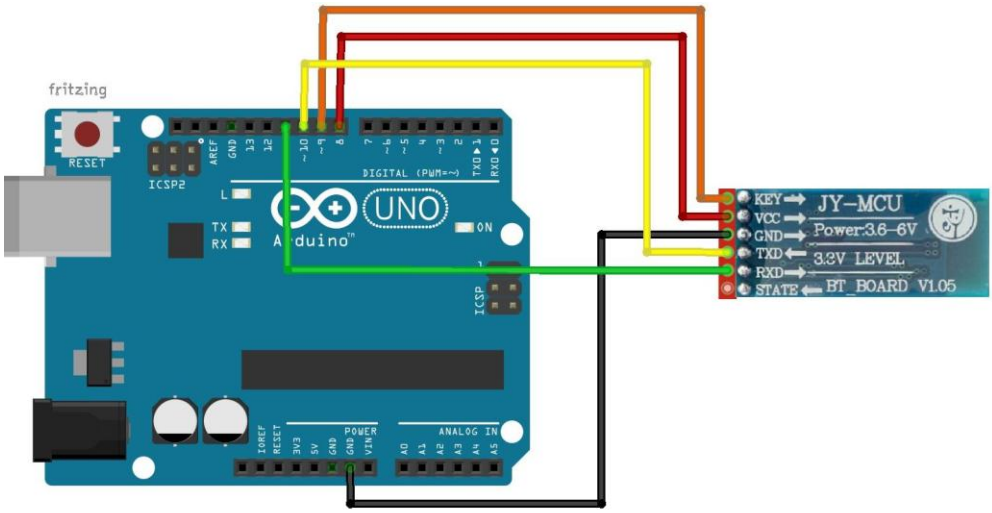
- Ahora, necesitamos agregar el archivo del tipo "build.gradle" para compilar la librería utilizando Gradle (la cual se encuentra ahora compilada en ANT). El mismo puede descargarse desde el siguiente enlace: https://github.com/priyankvex/Easy-Ocr-Scanner-Android/blob/master/easy_ocr_library/libs/tess-two/build.gradle. Este archivo deberá descargarse dentro de la carpeta "tess-two/" mencionada anteriormente.

- **IMPORTANTE**: modificar este archivo con las correspondientes versiones de SDK (compile, min y target) y de herramientas de construcción (buildToolsVersion), de manera que coincidan con las de nuestro proyecto, tal como se explicó previamente en este mismo Anexo para el caso de la librería Opencv. De lo contrario, la librería quedará erróneamente compilada.

- Por último, ir al archivo "project.settings" de nuestro proyecto y agregar la siguiente línea: "**include ':tess-two'**". Sólo resta construir (build) el proyecto, lo que puede tomar varios minutos.

ANEXO C: Configuracion de Módulo Arduino Bluetooth CZ-HC-05

Para la configuración del módulo de Bluetooth, se optó por utilización de la Arduino UNO, implementándose la siguiente conexión circuital:



Se debe tener en cuenta que los parámetros de configuración deben ser comunicados a través del puerto serie a una velocidad de 38400 Baud Rate, una vez que se haya iniciado el Modo AT (explicado en el informe), el cual se identifica mediante el parpadeo lento del led incorporado En el IDE de Arduino, se compiló y cargó el siguiente código, el cual se encuentra disponible y recomendado en la web:

sketch_mar23b

```
#include <SoftwareSerial.h>   // Incluimos la librería  SoftwareSerial
SoftwareSerial BT(10,11);    // Definimos los pines RX y TX del Arduino conectados al Bluetooth

void setup()
{
  BT.begin(38400);       // Inicializamos el puerto serie BT (Para Modo AT 2)
  Serial.begin(9600);   // Inicializamos  el puerto serie
}

void loop()
{
  if(BT.available())    // Si llega un dato por el puerto BT se envía al monitor serial
  {
    Serial.write(BT.read());
  }

  if(Serial.available())  // Si llega un dato por el monitor serial se envía al puerto BT
  {
     BT.write(Serial.read());
  }
}
```

Con este programa, puede configurarse el módulo mediante los comandos AT, los cuales sirven tanto para setear los parámetros como para encuestarlos, con sólo colocar el signo "?" al final. A continuación se enlistan los códigos AT disponibles:

AT COMMAND LISTING	
COMMAND	FUNCTION
AT	Test UART Connection
AT+RESET	Reset Device
AT+VERSION	Query firmware version
AT+ORGL	Restore settings to Factory Defaults
AT+ADDR	Query Device Bluetooth Address
AT+NAME	Query/Set Device Name
AT+RNAME	Query Remote Bluetooth Device's
AT+ROLE	Query/Set Device Role
AT+CLASS	Query/Set Class of Device CoD
AT+IAC	Query/Set Inquire Access Code
AT+INQM	Query/Set Inquire Access Mode
AT+PSWDAT+PIN	Query/Set Pairing Passkey
AT+UART	Query/Set UART parameter
AT+CMODE	Query/Set Connection Mode
AT+BIND	Query/Set Binding Bluetooth Address
AT+POLAR	Query/Set LED Output Polarity
AT+PIO	Set/Reset a User I/O pin

La velocidad de transmisión a través de la UART puede elegirse de entre las siguientes, y solo debe colocarse el número que las identifica:

1 configura	1200bps
2 configura	2400bps
3 configura	4800bps
4 configura	9600bps (Default)
5 configura	19200bps
6 configura	38400bps
7 configura	57600bps
8 configura	115200bps

❖ Bibliografía:

- Sitio oficial de Opencv: opencv.org
- Sitio oficial de Android: www.android.com
- Datasheet PIC 18F14k50.
- Datasheet DFPlayer Mini Manual.