

COMUNICACIÓN BLUETOOTH ENTRE ARDUINO Y ANDROID APLICACIÓN “CONTROLBT”

Pablo Correa

Universidad EAFIT

Medellín, Colombia

pcorream2@eafit.edu.co

Tomás Navarro

Universidad EAFIT

Medellín, Colombia

tdnavarrom@eafit.edu.co

ÍNDICE

1.	Introducción	pag 3.
2.	Requisitos	pag 3.
3.	Descargas	pag 4.
3.1.	Android Studio IDE	pag 4.
3.2.	Arduino IDE	pag 5.
3.3.	Repositorio en GitHub	pag 6.
4.	Comunicación Bluetooth desde Arduino	pag 7.
4.1.	Conexión Hardware	pag 7.
4.2.	Conexión en el Software	pag 7.
5.	Comunicación Bluetooth desde Android Studio	pag 8.
5.1.	Dispositivo en modo Desarrollador	pag 8.
5.2.	Descomprimir el Proyecto de Android Studio	pag 8.
5.3.	Importar y correr el Proyecto en Android Studio .	pag 8.
6.	Ejecución de la Aplicación y Usos	pag 10.
7.	Explicación de los Códigos	pag 12.
7.1.	Código Arduino	pag 12.
7.2.	Código Android Studio	pag 13.
8.	Configuración Extra del Módulo HC-05	pag 18.
9.	Referencias	pag 22.

1. Introducción

Este manual trata acerca de la explicación paso a paso de cómo establecer una conexión bluetooth entre un Arduino Mega 2560 con módulo HC-05 y un dispositivo móvil Android (Ver Requisitos), utilizando las siguientes herramientas de programación: Arduino IDE y Android Studio, para su desarrollo de software.

La plataforma Android Studio permite integrar la comunicación bluetooth a las aplicaciones que se realizan en su entorno mediante la API Bluetooth. Habilitando la posibilidad de tener y desarrollar funciones inalámbricas, ya sean punto a punto, o multipunto.

2. Requisitos

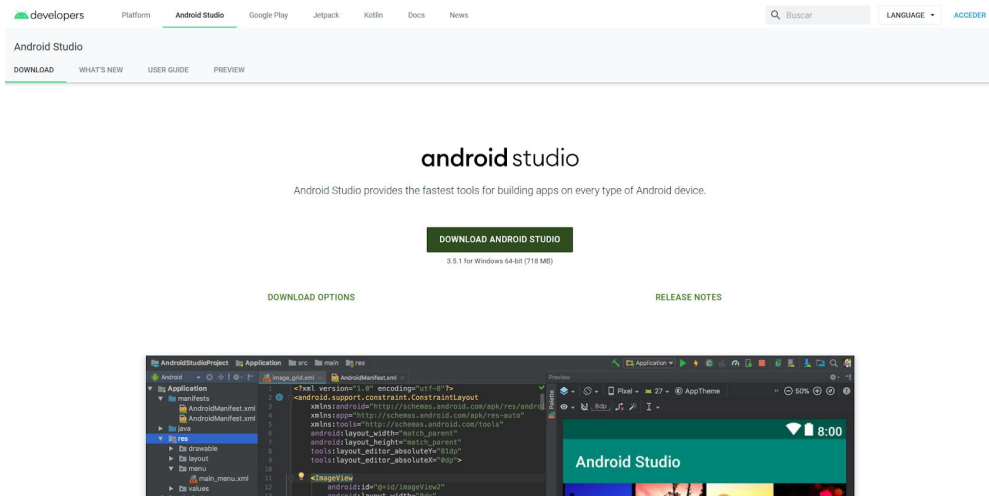
- 2.1.** Android Studio.
- 2.2.** Dispositivo Android con una versión mínima de 8.0 Oreo con soporte Bluetooth.
- 2.3.** Arduino IDE.
- 2.4.** Arduino MEGA 2510 (Recomendado) con modulo Bluetooth HC-05.
- 2.5.** Dispositivo Android con su respectivo cable USB.
- 2.6.** Dispositivo Android en Modo Desarrollador.

3. Descargas:

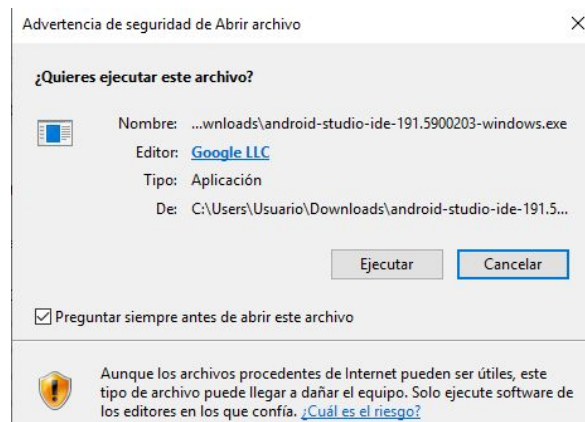
3.1. Android Studio IDE :

Para descargar Android Studio debemos acceder al siguiente link:

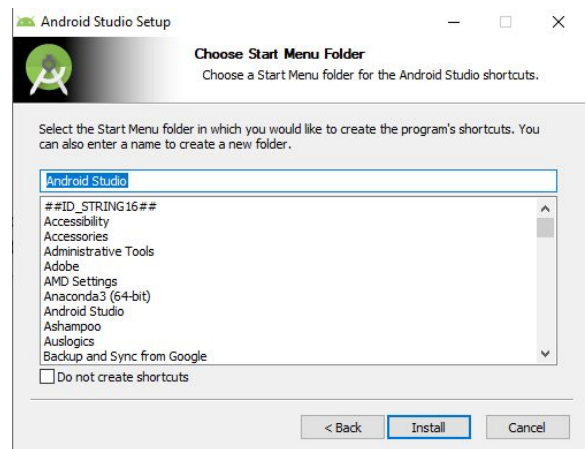
<https://developer.android.com/studio> . Ahí nos encontraremos en el inicio de la pagina, ahí daremos click en “DOWNLOAD ANDROID STUDIO”. Posteriormente tendremos que aceptar los términos y condiciones que nos pide el Software y comenzará su descarga.



Cuando acabe la Descarga abrimos la ubicación donde se descargo Android Studio y daremos click para empezar la instalación del software. Al instante nos pedirá permiso para ejecutar la instalación y daremos en “Ejecutar”.



Luego tendremos que dar unos permisos de administrador. Despues se abra una interfaz propia del Software (imagen derecha) y daremos click en el botón “Next” hasta que nos aparezca el botón “Install” (Imagen izquierda) y daremos click en él .




Tendremos que esperar unos minutos mientras se instalan todos los paquetes necesarios, cuando esto termine volveremos a dar click en “Next” y luego “Finish” para terminar la instalación.

3.2. Arduino:

Para descargar el IDE de Arduino iremos a el link: <https://www.arduino.cc/en/Main/Software> . Ahí nos aparecerán las diferentes versiones del Software para cada sistema Operativo. Donde escogeremos el que nos corresponden. Por temas de practicidad esta guia es para Windows, por ello daremos click en esa versión.



Download the Arduino IDE



ARDUINO 1.8.10
The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in java and based on Processing and other open-source software.
This software can be used with any Arduino board. Refer to the Getting Started page for installation instructions.

Windows installer, for Windows XP and up
Windows ZIP file for non admin install
Windows app Requires Win 6.1 or 10
[Get it](#)

Mac OS X 10.8 Mountain Lion or newer

Linux 32 bits
Linux 64 bits
Linux ARM 32 bits
Linux ARM 64 bits

[Release Notes](#)
[Source Code](#)
[Checkouts \(sha312\)](#)

HOURLY BUILDS LAST UPDATE 14 October 2019 4:24:23 GMT

Download a **preview of the incoming release** with the most updated features and bugfixes.

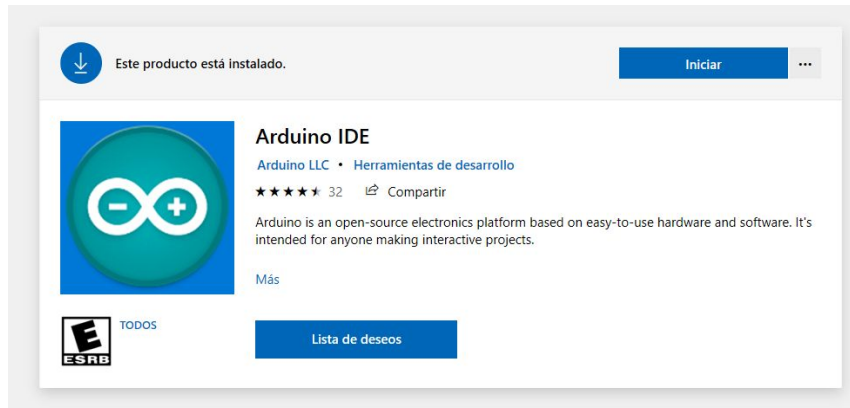
Windows
Mac OS X (Mac OSX Mountain Lion or later)
Linux 32 bit, Linux 64 bit, Linux ARM, Linux ARM64

BETA BUILDS BETA

Download the **Beta Version** of the Arduino IDE with experimental features. This version should NOT be used in production.

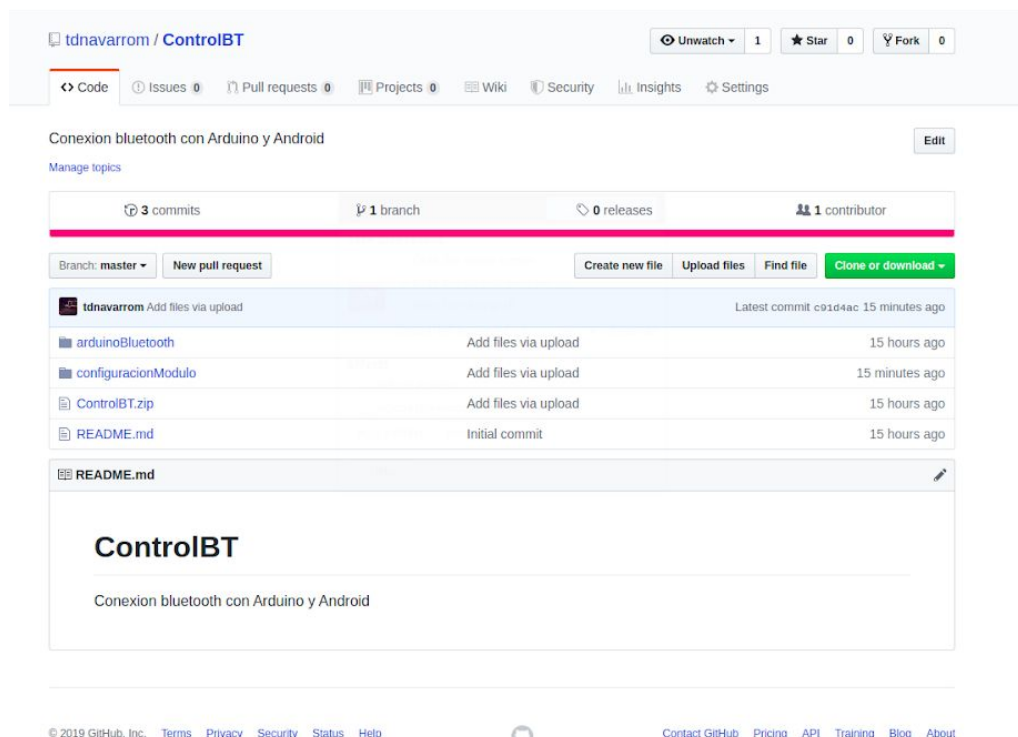
Windows
Mac OS X (Mac OSX Mountain Lion or later)
Linux 32 bit, Linux 64 bit, Linux ARM, Linux ARM64

Seguiremos los pasos para descargar hasta que nos envíe a la Microsoft Store en donde le daremos instalar (En nuestro caso dice iniciar puesto que ya se poseía el Software). Cuando termine la instalación le damos click en Iniciar.



3.3. Repositorio en GitHub:

Para descargar el código para la aplicación nos debemos ir al siguiente link de GitHub: <https://github.com/tdnavarrom/ControlBT>. En este daremos click en “Clone or download” y luego en “Download ZIP”.



4. Comunicación Bluetooth desde Arduino

En este apartado se explicará paso a paso, el procedimiento para la conexión del módulo bluetooth HC-05 y el Arduino MEGA 2560. Primero, desde la parte de hardware para conexión de los puertos. Luego, la conexión software desde el IDE de Arduino.

4.1. Conexión hardware

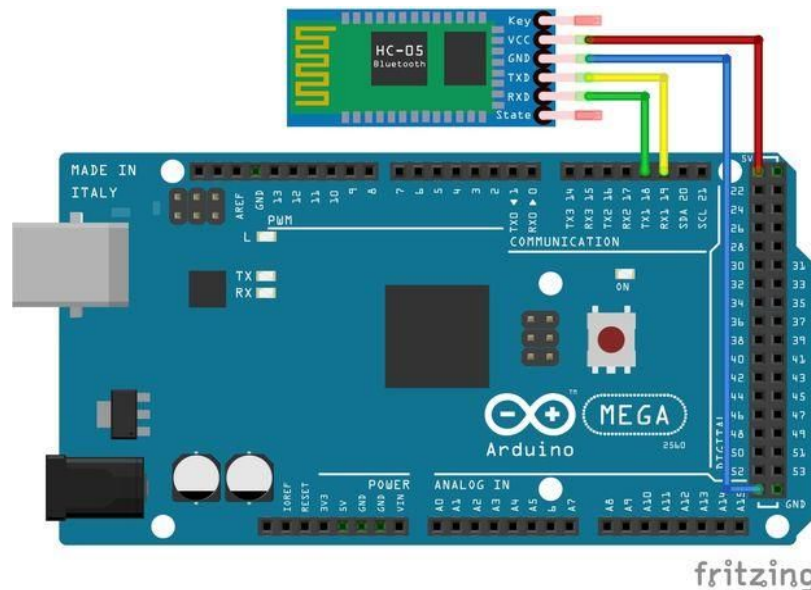


Imagen tomada de: <https://foto.askix.com/upload/2/a6/2a67ade68bd5921a5912fb236641d4c2.jpg>

La conexión del Arduino MEGA con el Módulo HC-05 es la siguiente:

- Cable Rojo: VCC (HC-05) hacía una salida 5V del Arduino.
- Cable Azul: GND (HC-05) hacía una salida GND del Arduino.
- Cable Amarillo: TXD(HC-05) hacia el pin 19 (RX1) del Arduino.
- Cable Verde: RXD(HC-05) hacia el pin 18 (TX1) del Arduino.

4.2 Conexión en el Software:

Para comenzar con el desarrollo del código necesario para la conexión del Arduino con el Módulo Bluetooth, se necesita primero descargar el repositorio de github (Ver Sección 3.3 del documento).

Una vez encontrado el archivo `arduinoBluetooth.ino` que se encuentra en la carpeta `arduinoBluetooth`, lo abrimos con Arduino IDE.

Conectamos nuestro dispositivo Arduino con el modulo HC-05, y el arduino al Computador. Una vez conectados, compilamos y ya estaría subido a la placa el programa.

5. Comunicación Bluetooth desde Android Studio:

5.1. Dispositivo en modo Desarrollador:

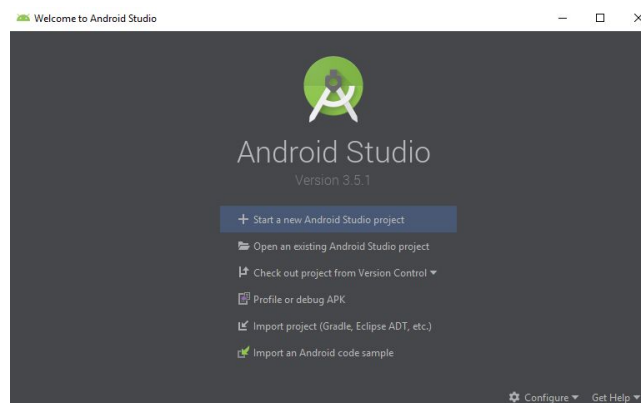
Al ser usuarios de Android tenemos la opción de poner el dispositivo en modo Desarrollador: Lamentablemente hay mucha variedad de dispositivos y para cada uno puede ser diferente el proceso, por ello recomendamos buscar la manera para hacerlo en tu dispositivo. Normalmente es en Ajustes, luego preferencias u opciones avanzadas.

5.2. Descomprimir el Proyecto de Android Studio:

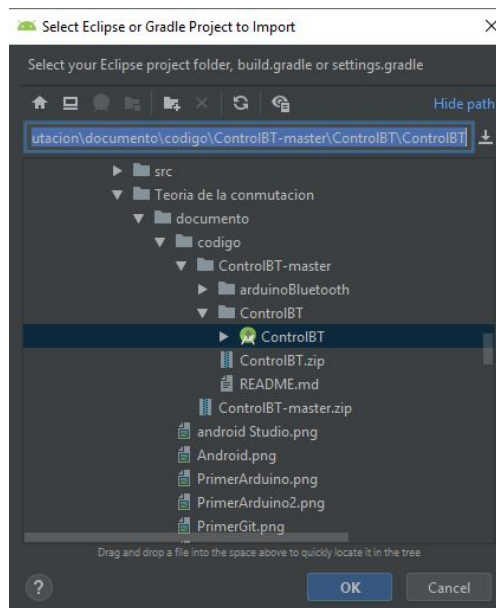
Para descomprimir un paquete zip descargado de GitHub ("ControlBT-master.zip") debemos darle click derecho en el archivo y seleccionar la opción "extraer aquí". Se generará una carpeta con el mismo nombre del archivo que descomprimimos, dentro de está carpeta encontraremos otro archivo ("ControlBT.zip") comprimido y repetimos el proceso anterior, esta carpeta será la que Android Studio reconocerá como proyecto y nos dejará importarla al IDE.

5.3. Importar y correr el Proyecto en Android Studio:

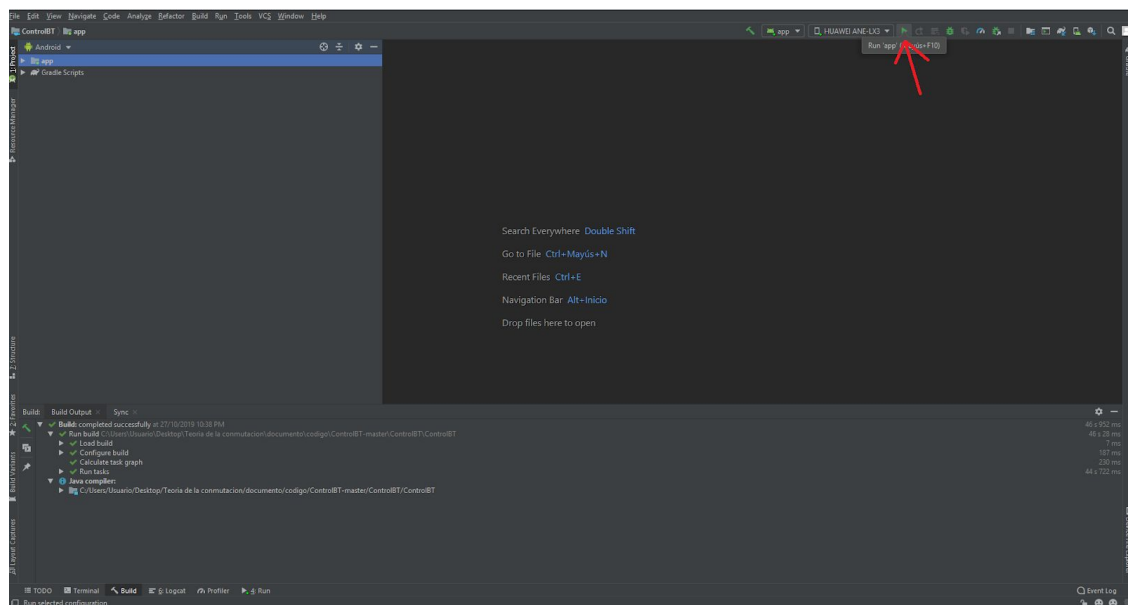
Para importar un proyecto en Android Studio debemos abrir la aplicación y darle Click en "Import project".



Una vez descomprimido el proyecto (ver sección 5.2) buscaremos en la ventana que nos aparecerá la carpeta que el IDE de Android Studio reconoce como proyecto, le damos Click para abrirlo. Si por alguna razón sale algún error de, ignorenlo y continúen, el software lo corrige automáticamente. A continuación una imagen de como se ve el el icono del proyecto de Android Studio.



Para correr el programa debemos conectar nuestro dispositivo Android al computador (Android Studio lo reconocera automaticamente). Recuerda qué se debe tener el modo Desarrollador de tu dispositivo activado (ver sección 5.1). Posteriormente le daremos click a la flecha verde en la parte superior de la pantalla, para compilar y subirle el programa al dispositivo. Y listo, debería aparecer la aplicación en el dispositivo.

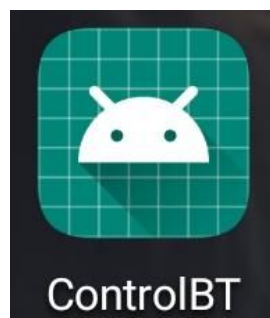


6. Ejecución de la Aplicación y Usos:

Usar la aplicación es muy sencillo, primero debemos conectar nuestro dispositivo con el módulo bluetooth conectado al arduino, esto desde la configuración de nuestro dispositivo. En nuestro ejemplo usamos el modulo HC-05, el cual nos aparecerá en la pantalla y lo seleccionaremos, a continuación nos pedirá un PIN este normalmente es 1234 o 0000. Una vez hayamos ingresado este, el módulo quedará apareado con nuestro dispositivo.



Una vez realizado esto buscaremos el icono de la aplicación “ControlBT” que debe aparecernos en nuestro dispositivo, una vez lo veamos lo seleccionamos e ingresamos a la aplicación.



Cuando estemos en la aplicación lo primero que veremos son los dispositivos que tenemos apareados al nuestro. Ahí, en nuestro caso, seleccionaremos el modulo HC-05.

ControlBT	
Dispositivos apareados	
PA15	A4:77:58:02:8C:15
BT ChevyStar - 3119	A0:14:3D:59:A4:0A
HC-05	98:D3:32:31:3B:4D
Car Kit	40:EF:4C:22:57:07
Hands Free System	0C:D9:C1:4C:DA:74
My Radio	18:17:14:6C:66:1B

Una vez seleccionado la aplicación se irá a un menú en donde aparecerán los pedidos, referencias, posición, canasta y cantidad que el módulo bluetooth nos está mandando.

ControlBT	
Control vehículo autónomo	
Pedido	Referencia
TextView	TextView
Posición	Canasta
TextView	TextView
Cantidad	
+	-
<div>ACEPTAR</div> <div>TextView</div> <div>UNIVERSIDAD EAFIT®</div>	

7. Explicación de los Códigos:

7.1. Código Arduino:

En esta sección abordaremos ya el código Arduino como tal, que se necesita para la aplicación:



```
arduinoBluetooth
#define SW 34
String s, value = "250";
void setup()
{
  pinMode(SW, INPUT);
  Serial1.begin(9600);
  Serial.begin(9600);
}

bool keyword(char c) {
  return c == 'a' or c == 'p' or c == 'm' or c == 's' or c == 'h' or c == 'E';
}

void loop() {
  if (Serial1.available()) {
    char c = Serial1.read();
    if (keyword(c)) {
      if (s == "") Serial.println(c);
      else {
        Serial.println(s);
        value = s;
      }
      s = "";
    } else s += c;
  }
  else if (digitalRead(SW)) {
    String string = "*4301,492383,222223,422," + String(value) + ",Bienvenido#";
    Serial1.println(string);
    Serial.println(string);
    delay(100);
  }
}
```

- “#define SW 34” lo utilizamos para poder usar el pin #34, que se encuentra en el Arduino, como un switch (Utilizando los Arduinos de la universidad).
- Se utilizan variables tipo “String” ya que es el tipo con el que bluetooth transmite la información.
- Para acceder a la información que se envía vía bluetooth utilizamos “Serial1.available()” que nos dirá si el dispositivo Android está conectado o no al Arduino. Se utiliza Serial1.read() para acceder la información que se está transmitiendo y las variables “s y value” son utilizadas para visualizar la información que el dispositivo android le está enviando vía bluetooth al Arduino.
- Si deseamos enviar información del Arduino al dispositivo Android, activamos el SW34, donde se enviará una cadena de caracteres separadas por “,” que se podrán visualizar en el dispositivo Android en sus respectivos lugares.

7.2. Código Android Studio:

A continuación explicaremos algunos detalles a tener en cuenta para el código de Android, puesto que gran parte de este se genera solo y no vale la pena mencionarlo, además de como modificar una interfaz gráfica en el mismo IDE.

La función que se puede visualizar en la imagen inferior es para lograr la conexión de bluetooth con Android. Lo primero que se hace es confirmar la conexión con bluetooth hacia el Arduino. Luego, se confirma la procedencia de la información que fue enviada por parte del Arduino hacia el Android. Una vez confirmada su procedencia, se evalúa sus datos, que son recibidos separados en “,” y son asignados respectivamente a sus IDs, para poder ser visualizados en la aplicación.

```
63 bluetoothIn = (Handler) handleMessage(msg) -> {
64
65     if (msg.what == handlerState) {
66         String readMessage = (String) msg.obj;
67         DataStringIN.append(readMessage);
68
69         int endOfLineIndex = DataStringIN.indexOf("#");
70         int beginOfLine = DataStringIN.indexOf("");
71
72         TextView[] texts = {IdText1, IdText2, IdText3, IdText4, IdValue, IdText5};
73
74         if (endOfLineIndex > 0 && beginOfLine == 0) {
75             System.out.println("holi");
76             String dataInPrint = DataStringIN.substring(1, endOfLineIndex);
77             String[] lines = dataInPrint.split(regex: ",");
78             if (lines.length == 6){
79                 for (int i = 0; i<lines.length; ++i) texts[i].setText(lines[i]);
80             }else {
81                 Toast.makeText(getBaseContext(), text: "Error en el string", Toast.LENGTH_LONG).show();
82                 MyConexionBF.write( input: "E");
83             }
84             DataStringIN.delete(0, DataStringIN.length());
85         }
86     }
87 }
```

De la línea 90 a la 179, los métodos que se encuentran son los de los botones que envían información al Arduino.

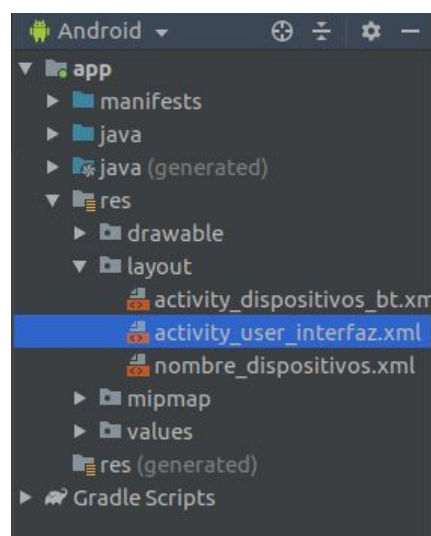
- IdEncender es el botón “+,” esto es para aumentar el valor del medio.
- IdApagar es el botón “-,” esto es para disminuir el valor del medio.
- IdAccept es para el botón “ACEPTAR,” esto es para enviar el valor al Arduino
- IdStop y IdHome, es para enviar cadenas a Arduino.
- IdExit y IdDesconectar, es para desconectarse a la conexión que hay entre el Arduino y el dispositivo Android.

```

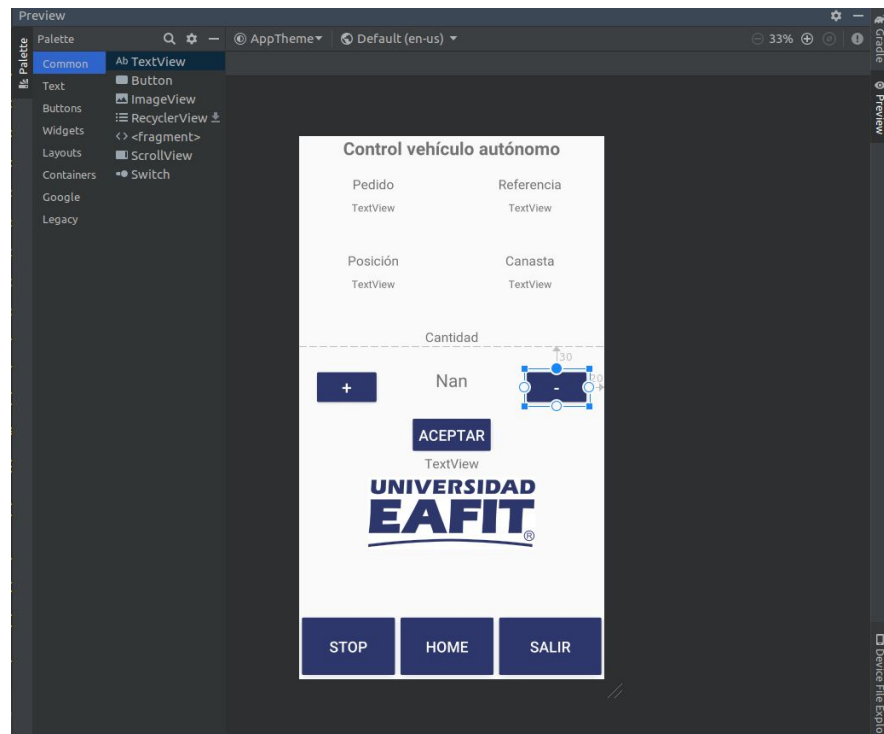
90 btAdapter = BluetoothAdapter.getDefaultAdapter(); // get Bluetooth adapter
91 VerificarEstadoBT();
92
93 // Configuración onClick listeners para los botones
94 // para indicar que se realizara cuando se detecte
95 // el evento de Click
96 IdEncender.setOnClickListener((v) -> {
99     String IdValueText = (String) IdValue.getText();
100     if (!IdValueText.equals("")) {...} else {...}
109 });
111
112 IdApagar.setOnClickListener((v) -> {
114     String IdValueText = (String) IdValue.getText();
115     if (!IdValueText.equals("")) {...} else {...}
125 });
127
128 IdAccept.setOnClickListener((v) -> {
131     String string = val == -1? "a" : val+"a";
132     MyConexionBT.write(string);
133     val = -1;
134     TextView[] texts = {IdText1, IdText2, IdText3, IdText4, IdValue, IdText5};
135     for (TextView t : texts) t.setText("");
136 });
138
139 IdStop.setOnClickListener((v) -> { MyConexionBT.write( input: "s"); });
145
146 IdHome.setOnClickListener((v) -> { MyConexionBT.write( input: "h"); });
152
153
154 IdExit.setOnClickListener((v) -> {
157     if (btSocket!=null)
158     {...}
159     finish();
163     moveTaskToBack( nonRoot: true);
164
165 });

```

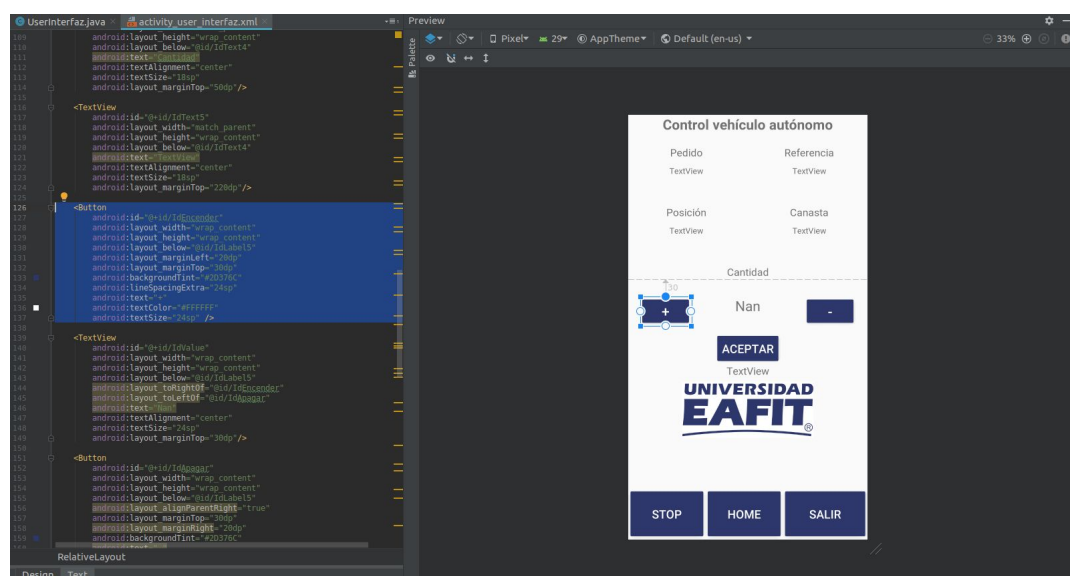
Para la creación y modificación de botones, nos dirigimos al archivo “activity_user_interfaz.xml” que se encuentra ubicado dentro de la carpeta “layout” que hace parte del paquete “app”



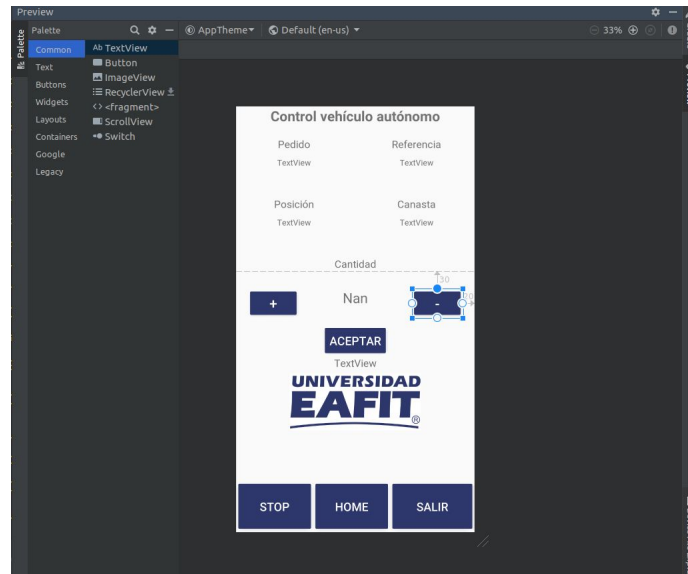
Allí nos aparecerá la siguiente interfaz de modificación, donde podremos crear y editar botones. En la barra de herramientas ubicada a la izquierda de la pantalla, encontraremos las diferentes “Views” que el Software nos permite usar.



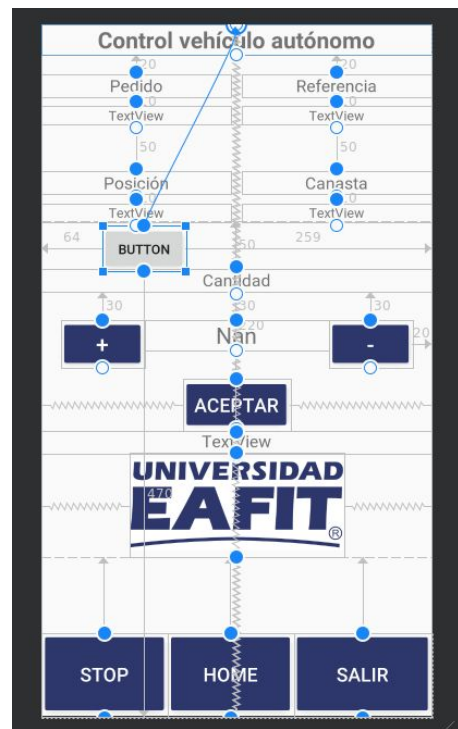
Al hacer doble click en cualquier botón de la interfaz que simula la aplicación, Android Studio nos mostrará en el código donde se encuentra la configuración visual del botón.



Para crear un botón, abrimos el menú “palette” ubicado en la parte izquierda de la pantalla, nos dirigimos a “Common”, seleccionamos “Boton” y lo arrastramos a la interfaz de la aplicación.



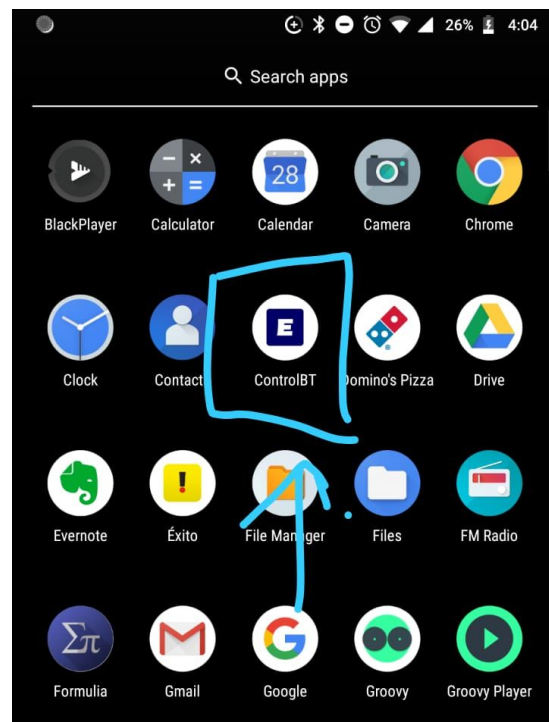
Al seleccionar el botón, nos aparecerán puntos blancos, los presionamos y los anclamos a las esquinas de la interfaz de la aplicación. Con esto hecho, ya podremos mover el botón en la aplicación y ponerlo donde se desea.



Para cambiar el icono de la aplicación el proceso es más fácil. El primer paso es añadir la imagen del nuevo ícono en la carpeta: ControlBT > app > src > main > res > mipmap-hdpi. Luego, entramos al directorio: ControlBT > app > src > main > AndroidManifest.xml. Allí en la etiqueta '<application>'. Para finalizar, modificamos los apartados: `android:icon="@mipmap/<el nombre del ícono>"`
`android:roundIcon="@mipmap/<el nombre del ícono>"`

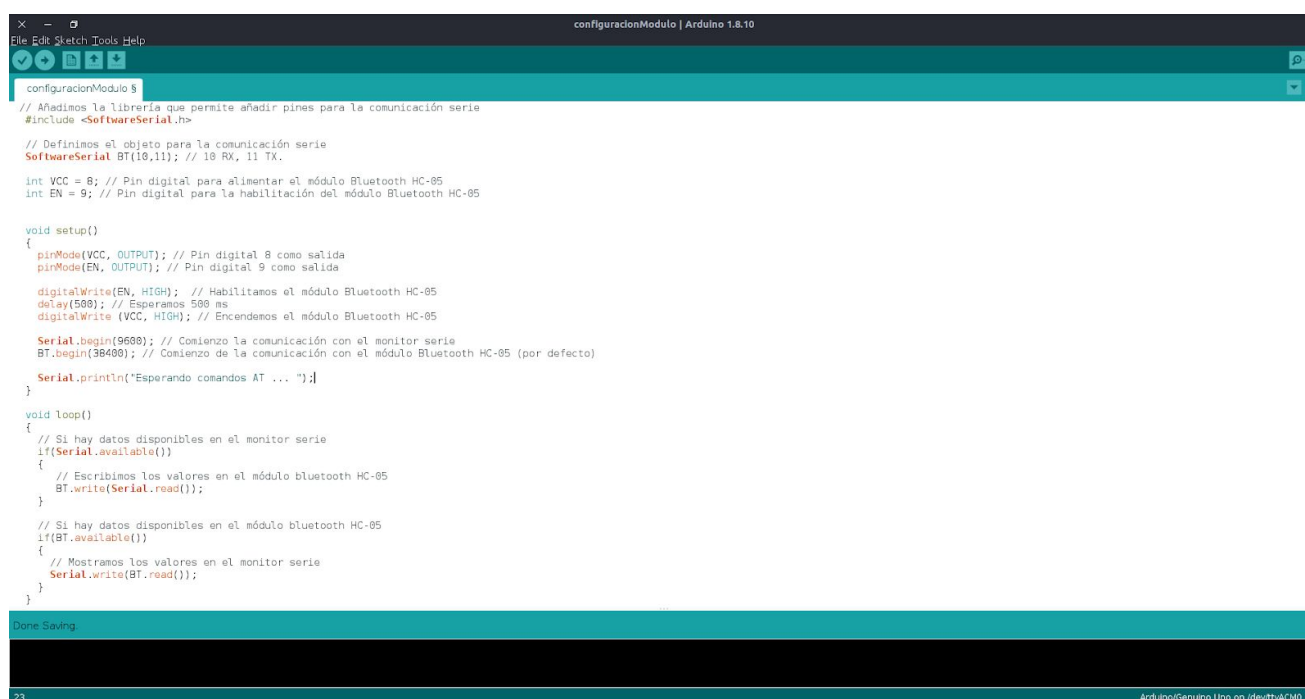
```
UserInterfaz.java x activity_user_interfaz.xml x AndroidManifest.xml x
4
5 <uses-permission android:name="android.permission.BLUETOOTH" />
6 <uses-permission android:name="android.permission.BLUETOOTH" />
7
8 <application
9     android:allowBackup="true"
10    android:icon="@mipmap/icon_logo"
11    android:label="ControlBT"
12    android:roundIcon="@mipmap/icon_logo"
13    android:supportsRtl="true"
14    android:theme="@style/AppTheme">
15
```

Una vez terminado lo anterior podremos encontrar que el icono cambio. En nuestro caso quedó de la siguiente manera :



8. Configuración Extra del Módulo HC-05

Si deseamos configurar nuestro módulo bluetooth HC-05, para cambiarle ya sea el nombre o la contraseña o la velocidad de los datos, podemos tomar el siguiente código como base, que se puede encontrar en el archivo descargado del repositorio de GitHub, en la carpeta “configuracionModulo”.



```
configuracionModulo
// Añadimos la librería que permite añadir pines para la comunicación serie
#include <SoftwareSerial.h>

// Definimos el objeto para la comunicación serie
SoftwareSerial BT(10,11); // 10 RX, 11 TX.

int VCC = 8; // Pin digital para alimentar el módulo Bluetooth HC-05
int EN = 9; // Pin digital para la habilitación del módulo Bluetooth HC-05

void setup()
{
  pinMode(VCC, OUTPUT); // Pin digital 8 como salida
  pinMode(EN, OUTPUT); // Pin digital 9 como salida

  digitalWrite(EN, HIGH); // Habilitamos el módulo Bluetooth HC-05
  delay(500); // Esperamos 500 ms
  digitalWrite(VCC, HIGH); // Encendemos el módulo Bluetooth HC-05

  Serial.begin(9600); // Comienzo la comunicación con el monitor serie
  BT.begin(38400); // Comienzo de la comunicación con el módulo Bluetooth HC-05 (por defecto)

  Serial.println("Esperando comandos AT ... ");
}

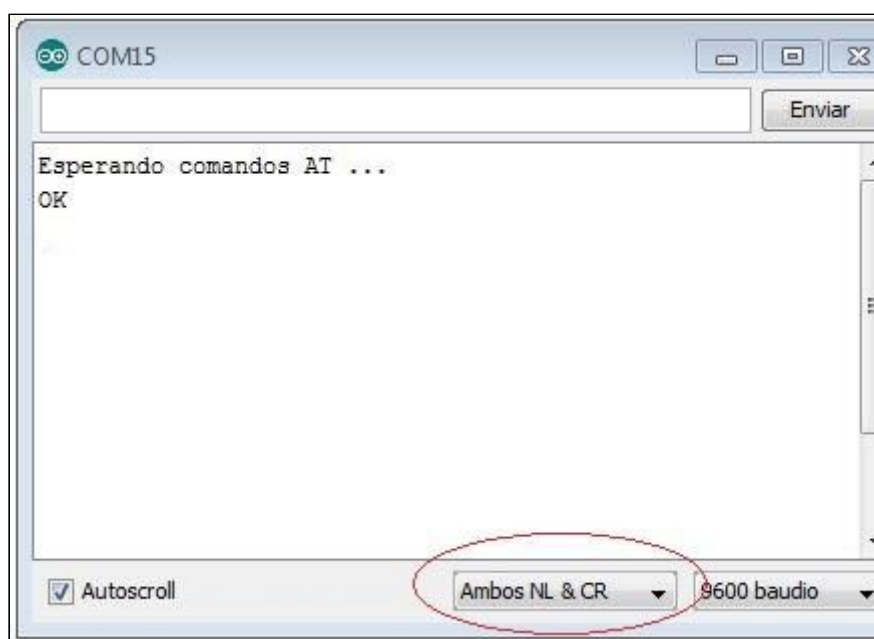
void loop()
{
  // Si hay datos disponibles en el monitor serie
  if(Serial.available())
  {
    // Escribimos los valores en el módulo bluetooth HC-05
    BT.write(Serial.read());
  }

  // Si hay datos disponibles en el módulo bluetooth HC-05
  if(BT.available())
  {
    // Mostramos los valores en el monitor serie
    Serial.write(BT.read());
  }
}
```

Arduino Mega	Modulo HC-05	Arduino Mega	Modulo HC-06
GND	GND	GND	GND
DIGITAL 8	VCC	+5V	VCC
DIGITAL 10	TX	DIGITAL 10	TX
DIGITAL 11	RX	DIGITAL 11	RX
DIGITAL 9	EN		
NO CONECTADO	STATE		

Ahora vamos a configurar el módulo Bluetooth HC-05 a través de comandos AT enviados desde el Monitor Serie del IDE de ARDUINO. Una vez compilado y cargado el código “Comandos_AT_ HC-05” en nuestro Arduino, abrimos el Monitor Serie, siempre manteniendo presionado el botón que trae el módulo.

Al enviar un comando AT desde el monitor serie, este módulo espera que la línea acabe en “\r\n” y entonces es cuando ejecuta el comando AT. En el Monitor Serie de IDE de Arduino tenemos que poner “Ambos NL&CR” para terminar las líneas , escribimos el comando “AT”, aunque este módulo no distingue entre mayúsculas y minúsculas, y pulsamos ENVIAR para comprobar que la comunicación es correcta. Como se puede observar en la imagen vemos que el módulo Bluetooth nos responde con un OK.



Monitor serie al enviar el comando “AT” en módulo HC-05

Ahora únicamente vamos a modificar el nombre del módulo insertando el comando “AT+NAME=Arduino_HC-05”, respondiéndonos el módulo HC-05 a través del monitor serie con “OK”. Los demás parámetros los vamos a dejar como nos vienen por defecto de fábrica. A continuación os dejo una tabla con los comandos AT más utilizados para el módulo HC-05.

Comando AT	Descripción	Respuesta
AT+ORGL	Restaura los valores por defecto del módulo Bluetooth HC-05	Responde con un OK . (+ROLE:0; +CMOD:0; +UART:38400,0,0; +PSWD:1234 y Nombre: "H-C-2010-06-01" o "HC-05").
AT	Test de comunicación.	Responde con un OK .
AT+VERSION	Retorna la versión del módulo	Responde con la versión (+VERSION:2.0-20100601) .
AT+UART=X,Y,Z	Configura la velocidad de transmisión del módulo según el valor de "x": 1200 bps, 2400 bps, 4800 bps, 9600 bps, 19200 bps, 38400 bps (por defecto), 57600 bps, 115200 bps, 230400 bps, 460800 bps, 921600 bps, 1382400 bps. Con el valor de "Y" se configura el bit de parada: 0 -> 1 bit. 1 -> 2 bits. Con el valor de "Z" se configura el bit de paridad: 0 -> Sin paridad. 1 -> Paridad impar. 2 -> Paridad Par.	AT+UART=38400,0,0 configura el módulo Bluetooth HC-05 con una velocidad de comunicación de 38400 bps, con un bit de parada y sin paridad. Responde con un OK . AT+UART. Preguntamos por el estado de la comunicación UART. Responde con +UART:38400,0,0 .
AT+NAME=XXXX	Configura el nombre con el que se visualizará el módulo, soporta hasta 20 caracteres.	AT+NAME=Arduino_HC-0 configura el nombre del módulo a Arduino_HC-05. Responde con un OK .
AT+PSWD=XXXX	Configura el Pin o la contraseña de acceso al módulo Bluetooth HC-05.	AT+PSWD=1234 configura el pin a 1234. Responde con un OK . AT+PSWD. Preguntamos por el pin. Nos responde con +PSWD:1234 .
AT+CMODE = X	Cambiamos el modo de conexión del módulo según el valor de "X": 0 -> Se conecta al dispositivo Bluetooth especificado. 1 -> Se conecta a cualquier dispositivo Bluetooth disponible.	AT+CMODE = 1 configura el módulo Bluetooth HC-05 para que pueda conectarse a cualquier dispositivo Bluetooth disponible (sin dirección requerida). Responde con un OK . AT+CMODE: Preguntamos por el modo de conexión del módulo. Responde con +CMOD:1 .
AT+ROLE = X	Cambiamos el modo de trabajo del módulo, según el valor de "X": 0 -> Modo Esclavo. 1 -> Modo Maestro.	AT+ROLE = 0 configura el módulo Bluetooth para que trabaje como Esclavo de otro dispositivo Bluetooth. Responde con un OK . AT+ROLE: Preguntamos por el modo de trabajo del módulo. Responde con +ROLE:0 .

// Añadimos la librería que permite añadir pines para la comunicación serie

```

#include <SoftwareSerial.h>

// Definimos el objeto para la comunicación serie
SoftwareSerial BT(10,11); // 10 RX, 11 TX.

int VCC = 8; // Pin digital para alimentar el módulo Bluetooth HC-05
int EN = 9; // Pin digital para la habilitación del módulo Bluetooth HC-05

void setup()
{
  pinMode(VCC, OUTPUT); // Pin digital 8 como salida
  pinMode(EN, OUTPUT); // Pin digital 9 como salida

  digitalWrite(EN, HIGH); // Habilitamos el módulo Bluetooth HC-05
  delay(500); // Esperamos 500 ms
  digitalWrite (VCC, HIGH); // Encendemos el módulo Bluetooth HC-05

  Serial.begin(9600); // Comienzo la comunicación con el monitor serie
  BT.begin(38400); // Comienzo de la comunicación con el módulo Bluetooth HC-05 (por defecto)

  Serial.println("Esperando comandos AT ... ");
}

void loop()
{
  // Si hay datos disponibles en el monitor serie
  if(Serial.available())
  {
    // Escribimos los valores en el módulo bluetooth HC-05
    BT.write(Serial.read());
  }

  // Si hay datos disponibles en el módulo bluetooth HC-05
  if(BT.available())
  {
    // Mostramos los valores en el monitor serie
    Serial.write(BT.read());
  }
}

```

9. Referencias

- Campo, M. del. (1970, January 1). Configurar módulos Bluetooth HC-05 y HC-06 mediante comandos AT. Retrieved from <https://miaridinounotieneunblog.blogspot.com/2016/12/configurar-modulos-bluetooth-hc-05-y-hc.html>.
- Download Android Studio and SDK tools : Android Developers. (n.d.). Retrieved from <https://developer.android.com/studio/>.
- (n.d.). Retrieved from <https://forum.arduino.cc/index.php?topic=545891.0>.
- Girao, D., & Girao, D. (2019, June 19). Cómo habilitar las opciones de desarrollador en Android. Retrieved from <https://www.movilzona.es/2019/06/19/como-habilitar-opciones-desarrollador-android/>