BLUETOOTH LOW ENERGY (BLE)

FINAL PROJECT - CARLOS ESTAY

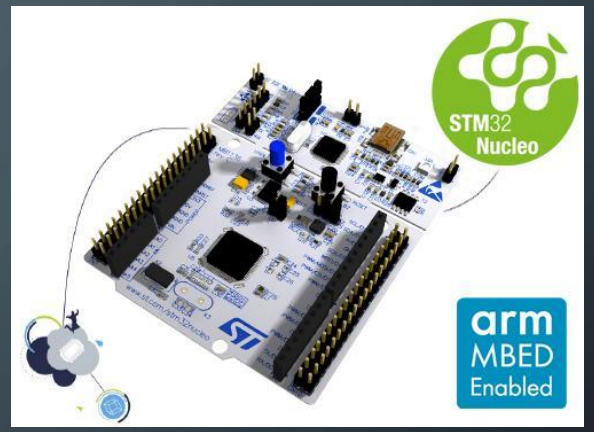
ANDROID DEVELOPMENT – DMIT2504

PROJECT OBJECTIVE

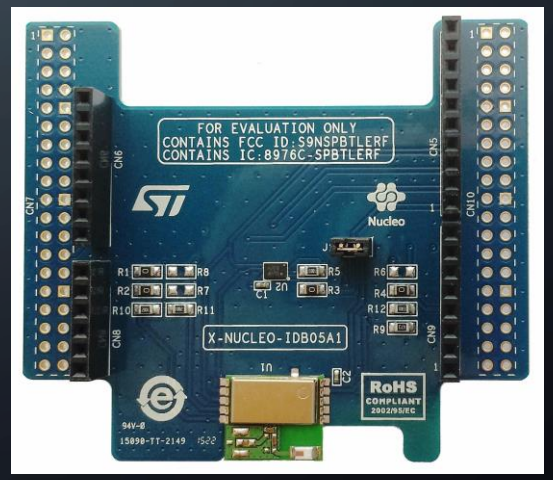
- Build an android app to remotely control a 4 wheel robot using BLE
- Control speed using slider (SeekBar)
- Control 4 possible moves with 4 buttons:
  - Forward
  - Reverse
  - Turn right
  - Turn left

PROJECT OBJECTIVE

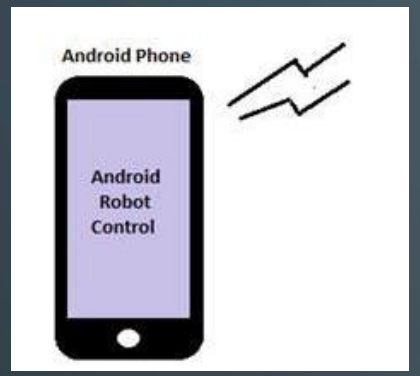
- Components to use:
  - Android Phone
  - Custom modified robot
  - Microcontroller module: NUCLEO-L476RG
    - https://www.st.com/en/evaluation-tools/nucleo-l476rg.html

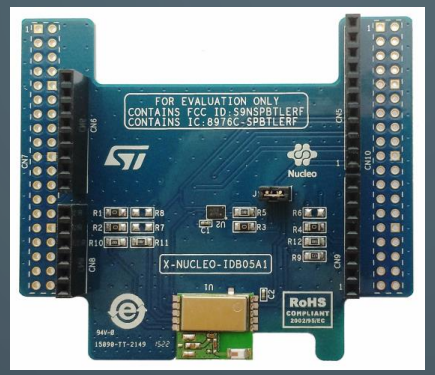


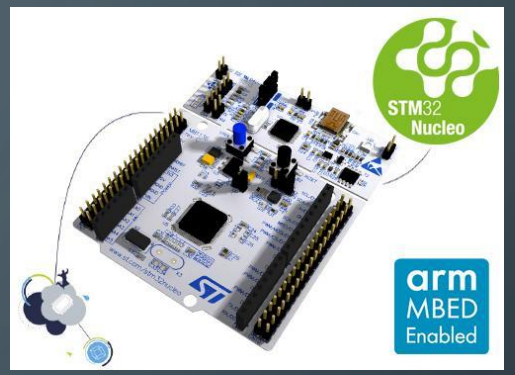
  - BLE module: X-NUCLEO-IDB05A1
    - https://www.st.com/en/ecosystems/x-nucleo-idb05a1.html

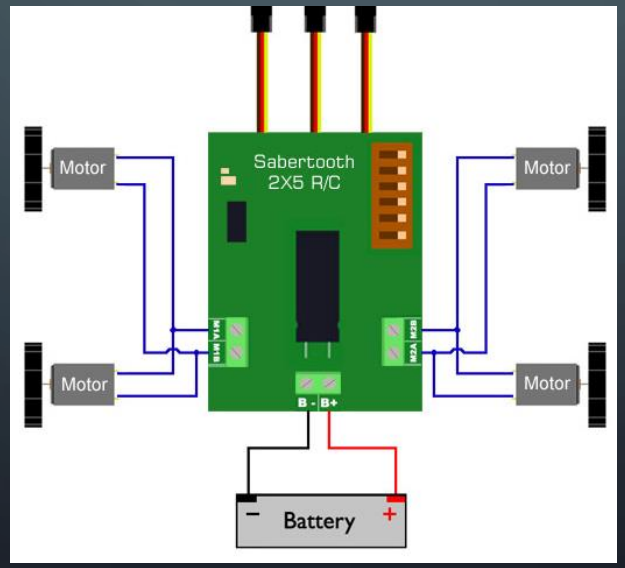


INTERACTION DIAGRAM



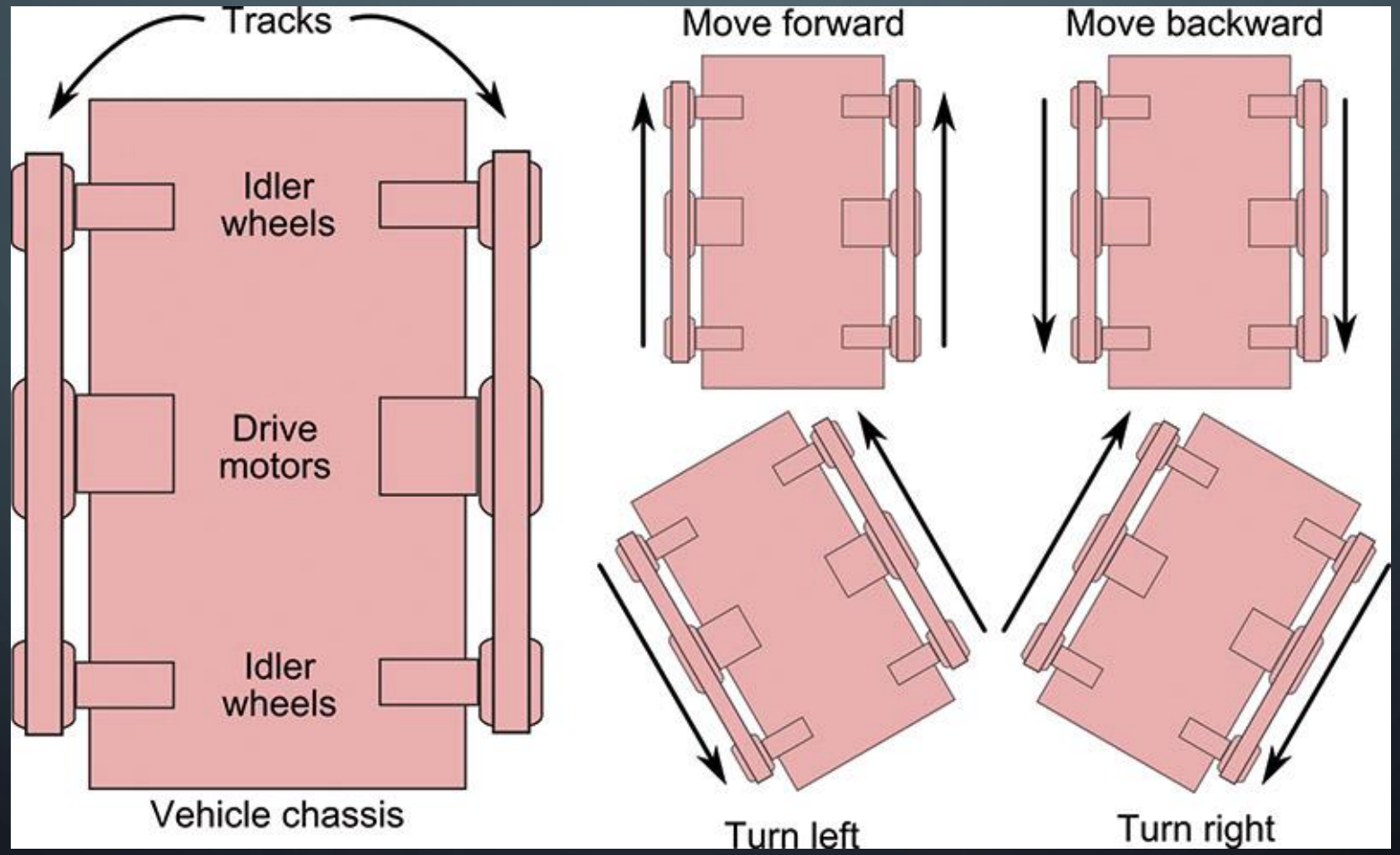






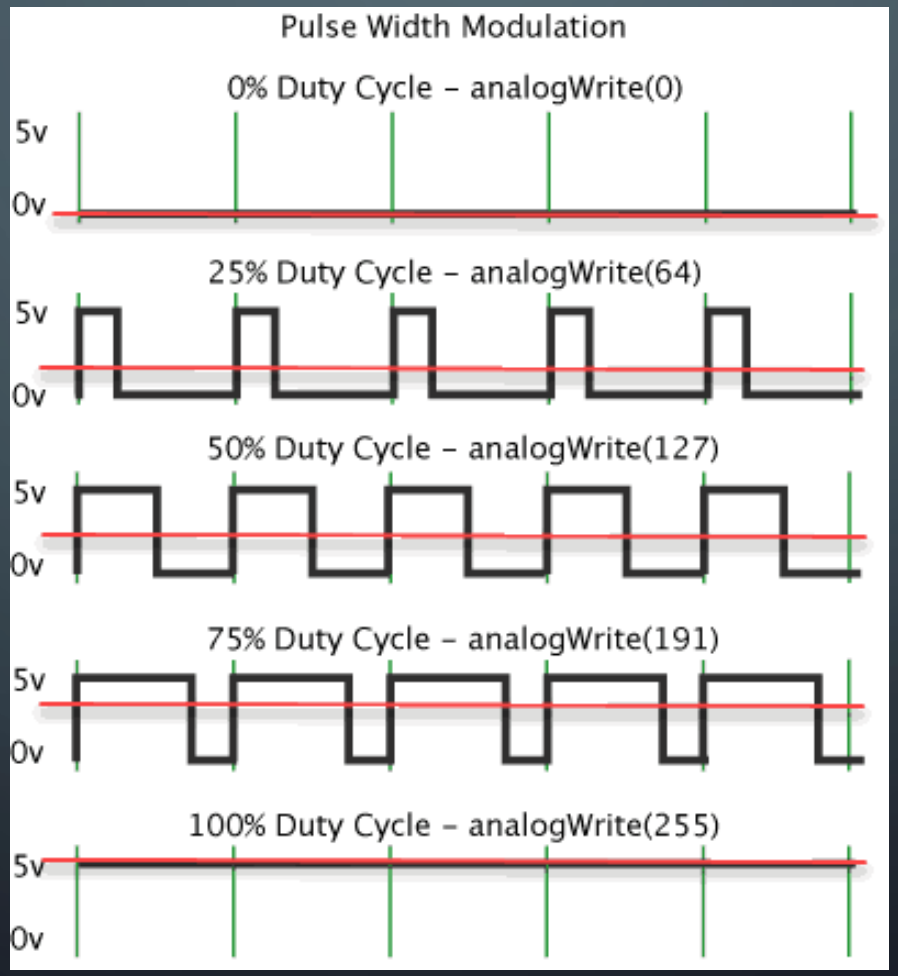
https://www.dimensionengineering.com/appnotes/simple_robot/

BASIC CONTROL OF A 4 WHEEL ROBOT



https://www.servomagazine.com/magazine/article/may2014_McComb

PULSE WIDTH MODULATION (PWM)



https://www.analogictips.com/pulse-width-modulation-pwm/

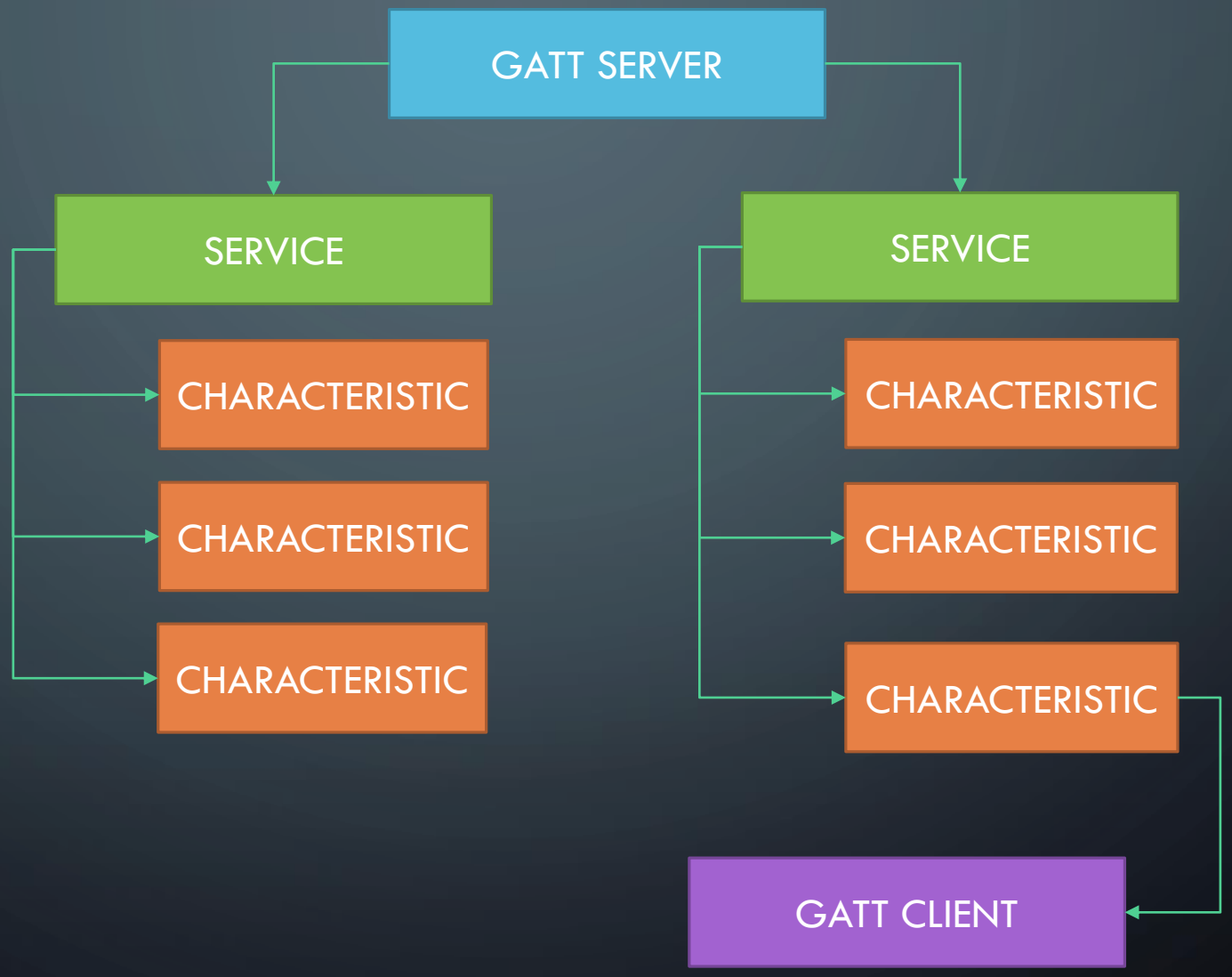
BLE FEATURES

- Transmit small amounts of data between devices
- It is designed for low power consumption
- Normally used for:
  - Proximity sensors
  - Heart rate monitors
  - Fitness devices
  - Remote control

BLE ACTORS

- GATT : Generic Attribute Profile
- GATT Server
- GATT Client
- Services: unique UUID identifier
- Characteristics: a service can have many characteristics. Also identified by a UUID

BLE ACTORS EXAMPLE



STEPS TO CONNECT TO A GATT SERVER

- STEP 1: enable permissions in android manifest

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

STEPS TO CONNECT TO A GATT SERVER

- STEP 2: Request for permission in run time, if not yet enabled

```
//Check and request permission if necessary
if (ContextCompat.checkSelfPermission(this,
        Manifest.permission.ACCESS_COARSE_LOCATION)
        != PackageManager.PERMISSION_GRANTED) {

    // Permission is not granted
    if (ActivityCompat.shouldShowRequestPermissionRationale(this,
            Manifest.permission.ACCESS_COARSE_LOCATION)) {
    } else {
        // No explanation needed; request the permission
        ActivityCompat.requestPermissions(this,
                new String[]{Manifest.permission.ACCESS_COARSE_LOCATION},
                1);
        // result of the request.
    }
} else {
    // Permission has already been granted
}
```

STEPS TO CONNECT TO A GATT SERVER

- Classes to utilize to scan devices and to connect
  - BluetoothManager
  - BluetoothAdapter
  - BluetoothGattServer : if you are setting device as a server
  - BluetoothGatt
  - BluetoothGattCharacteristic
  - BluetoothDevice
  - BluetoothLeScanner
  - Handler: to handle the Scan function

STEPS TO CONNECT TO A GATT SERVER

- STEP 3: initialize the Bluetooth adapter

```
//Initialize Bluetooth adapter
bluetoothManager = (BluetoothManager) getSystemService(Context.BLUETOOTH_SERVICE);
bluetoothAdapter = bluetoothManager.getAdapter();
```

```
//Enable Bluetooth if not yet enabled
if(bluetoothAdapter == null || !bluetoothAdapter.isEnabled()){
    Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
    startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
}
```

STEPS TO CONNECT TO A GATT SERVER

- STEP 4: Scan for devices – call scanLeDevice() function (or any name)

```
@TargetApi(21)
private void scanLeDevice(final boolean enable) {

    final BluetoothLeScanner bluetoothLeScanner = bluetoothAdapter.getBluetoothLeScanner();
    if (enable) {
        statusTextView.setText("Scanning...");
        // Stops scanning after a pre-defined scan period.

        handler.postDelayed(new Runnable() {
            @Override
            public void run() {
                mScanning = false;
                bluetoothLeScanner.stopScan(leScanCallback);
                statusTextView.setText("Scan stopped");
            }
        }, SCAN_PERIOD);
        mScanning = true;
        bluetoothLeScanner.startScan(leScanCallback);
    } else {
        mScanning = false;
        bluetoothLeScanner.stopScan(leScanCallback);
    }
}
```

STEPS TO CONNECT TO A GATT SERVER

- STEP 4: Scan for devices – Handle callback

```
private ScanCallback leScanCallback = new ScanCallback() {
    @Override
    public void onScanResult(int callbackType, ScanResult result) {
        super.onScanResult(callbackType, result);
    }

    @Override
    public void onBatchScanResults(List<ScanResult> results) {
        super.onBatchScanResults(results);
    }

    @Override
    public void onScanFailed(int errorCode) {
        super.onScanFailed(errorCode);
    }
```

STEPS TO CONNECT TO A GATT SERVER

- STEP 4: Scan for devices – Handle results
  - The result.getDevice() will return the Bluetooth Device to connect to.
    - BluetoothDevice myDevice = result.getDevice();
    - String myDeviceName = myDevice.getName;
    - String myDeviceAddress = myDevice.getAddress;

STEPS TO CONNECT TO A GATT SERVER

- STEP 4: Scan for devices – Handle results

```
@Override
public void onScanResult(int callbackType, ScanResult result) {
    super.onScanResult(callbackType, result);
    if (result.getDevice().getName() != null && result.getDevice().getName().contains(scanFilterTextView.getText())) {
        if(!deviceListAdapter.contains(result)){
            deviceListAdapter.addResult(result);//add found device to the listView if not already added
        }
        //register event for clicking items
        devicesListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
                ScanResult selectedResult = (ScanResult) deviceListAdapter.getItem(position);
                robotDevice = selectedResult.getDevice();
                bluetoothGatt = robotDevice.connectGatt(getActivity(), autoConnect: false, gattCallback);
                //Stop Scanning if connecting to a device
                scanLeDevice( enable: false);
                statusTextView.setText("Connecting to " + robotDevice.getName() + "...");
            }
        });
    }
}
@Override
```

STEPS TO CONNECT TO A GATT SERVER

- STEP 5: Select a device and connect to the Gatt server

```
myDevice.connectGatt(thisActivity, false, gattCallback);
```

STEPS TO CONNECT TO A GATT SERVER

- STEP 6: Handle gattCallback results

```
private final BluetoothGattCallback gattCallback = new BluetoothGattCallback() {
    @Override
    public void onConnectionStateChange(BluetoothGatt gatt, int status, int newState) {
        super.onConnectionStateChange(gatt, status, newState);
    }

    @Override
    public void onServicesDiscovered(BluetoothGatt gatt, int status) {
        super.onServicesDiscovered(gatt, status);
    }

    @Override
    public void onCharacteristicRead(BluetoothGatt gatt, BluetoothGattCharacteristic characteristic, int status) {
        super.onCharacteristicRead(gatt, characteristic, status);
    }

    @Override
    public void onCharacteristicWrite(BluetoothGatt gatt, BluetoothGattCharacteristic characteristic, int status) {
        super.onCharacteristicWrite(gatt, characteristic, status);
    }

    @Override
    public void onCharacteristicChanged(BluetoothGatt gatt, BluetoothGattCharacteristic characteristic) {
        super.onCharacteristicChanged(gatt, characteristic);
    }
};
```

STEPS TO CONNECT TO A GATT SERVER

- STEP 6: Handle gattCallback results - Discover services if connected

```
@Override
public void onConnectionStateChange(BluetoothGatt gatt, int
status, int newState) {
    //super.onConnectionStateChange(gatt, status, newState);
    connectedDevice = gatt.getDevice();
    switch(newState ){
        case BluetoothProfile.STATE_CONNECTED:
          bluetoothGatt.discoverServices();
            break;

        case BluetoothProfile.STATE_DISCONNECTED:
      break;

        default:
             break;
    }
```

STEPS TO CONNECT TO A GATT SERVER

- STEP 7: Get service and characteristic to use

```
@Override
public void onServicesDiscovered(BluetoothGatt gatt, int status) {
    //super.onServicesDiscovered(gatt, status);

    List<BluetoothGattService> services = gatt.getServices();

    String serviceUUID = "D973f2E0-B19E-11E2-9E96-0800200C9A66";
    String charUUID = "D973f2E2-B19E-11E2-9E96-0800200C9A66";

    BluetoothGattService service = gatt.getService(UUID.fromString(serviceUUID));
    customCharacteristic = service.getCharacteristic(UUID.fromString(charUUID));

}
```

STEPS TO CONNECT TO A GATT SERVER

- STEP 8: Write into custom characteristic

```
customCharacteristic.setValue("@MF0");
bluetoothGatt.writeCharacteristic(customCharacteristic);
```

- Pass Bluetooth device to a different activity if desired

```
//Robot connected, open remote control activity
Intent intent = new Intent(getApplicationContext(), RemoteControlActivity.class);
intent.putExtra("ROBOT", robotDevice);
startActivity(intent);
```

```
public final class BluetoothDevice
extends Object implements Parcelable
```

COMMANDS IMPLEMENTED FOR ROBOT

- "@MF0" : Move Forward
- "@MB0" : Move Reverse
- "@MR0" : Turn Right
- "@ML0" : Turn Left

- "@SXX" : Set speed to XX-> from 00 to 10

SUMMARY OF STEPS

1. Enable permissions in android manifest
2. Request permission ACCESS_COARSE_LOCATION in run time
3. Initialize Bluetooth Adapter
4. Scan devices: bluetoothLeScanner.startScan(leScanCallback);
   - Declare and handle (leScanCallback)
   - If onScanResult is called, add result to a list view
   - Once a result (device) is selected, go to step 5

SUMMARY OF STEPS

5. Connect to Gatt Server: myDevice.connectGatt(thisActivity, false, gattCallback);

6. Declare and handle gattCallback
   - if onConnectionStateChange is called and new state is STATE_CONNECTED, go to step 7

7. Get service and characteristic
   - If onServiceDiscovered is called, get desired service, if discovered, and characteristic

8. Write to custom characteristic selected
   - onCharacteristicWrite will be called as an acknowledge that the characteristic was written properly.

POSSIBLE OTHER IMPLEMENTATIONS OF BLE

- Create a Bluetooth Chat between 2 android devices, one configured as a Gatt Client and other one as a Gatt Server
- Create a chat between an android device and a computer that as a microcontroller with BLE module, which outputs the messages received in a terminal such as Tera Term, for instance
- Read other standard BLE devices/profiles. For instance, a heart rate monitor
- Use the android device as an extra or custom controller for a game

CONTACT DETAILS - QUESTIONS

carlos.estayo@gmail.com

cestayoyarzo1@studentmail.nait.ca

GitHub: cestayoyarzo1