Bluetooth RC mini car

Bardi Bogdan, Group 30431/1

2020-2021

1 Scope

The aim of this project is to create a small car that can be remotely controlled using an smart-phone. The car can go forward or backward or turn in a certain direction whenever the user requests it via an app on its phone.

This is achieved by using an Arduino UNO board and a motor shield to control the robot itself and an HC-05 Bluetooth module to connect to the phone and receive commands. To make the project more interesting the user has the option to draw on the phone and the car will do its best to follow the trajectory.

2 Parts required

- Arduino UNO board
- L293D compatible motor shield
- 4x DC motors
- HC-05 Bluetooth module
- Plastic chassis
- Battery Holder which can have 6 batteries in series
- Wires
- a phone that can run Android applications and has Bluetooth capabilities

3 Electonic schematic and final assembly

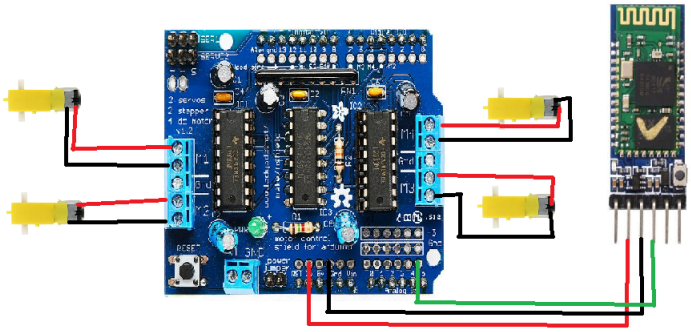

Figure 1: The electonic schematic

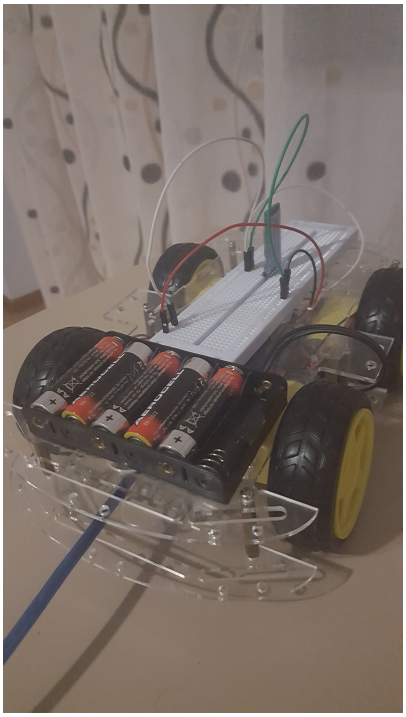

Figure 2: The fully assembled device

4 Implementation on the Arduino board

The Arduino board's job is to receive input from the HC-05's serial output(which is received from the phone) and to activate the motors according to the received command.

The phone can send the following to the RC car:

- u - go forward
- d - go backward
- l - turn left
- r - turn right

4.1 void setup()

In the setup procedure, the motors are initialized and the Software Serial port is initialized to communicate with the HC-05 board.

```
SoftSerial.begin(BAUDRATE);   // Init soft serial object
motor1.setSpeed(254);
motor1.run(RELEASE);
motor2.setSpeed(254);
motor2.run(RELEASE);
motor3.setSpeed(254);
motor3.run(RELEASE);
motor4.setSpeed(254);
motor4.run(RELEASE);
Serial.begin(9600);
```

4.2 void loop()

The loop procedure just checks if the HC-05 received anything and then process the signal by activating the motors in the directions intended.

```
if (SoftSerial.available())
{
  char a = SoftSerial.read();
  Serial.println(a);
  if(a == 'u'){
    /* engage the motors to go forward for 400ms */
  }
  if(a == 'd'){
    /* engage the motors to go backward for 400ms */
  }
  if(a == 'r'){
    /* engage the motors to turn the car right for 200ms */
  }
```

```
    if(a == '1'){
      /* engage the motors to turn the car left for 200ms */
    }
}
```

4.3 Implementation on Android

The Android application is implemented in Java using Android Studio and has to allow the user to connect to the Arduino and give commands to it.

4.3.1 Flowchart

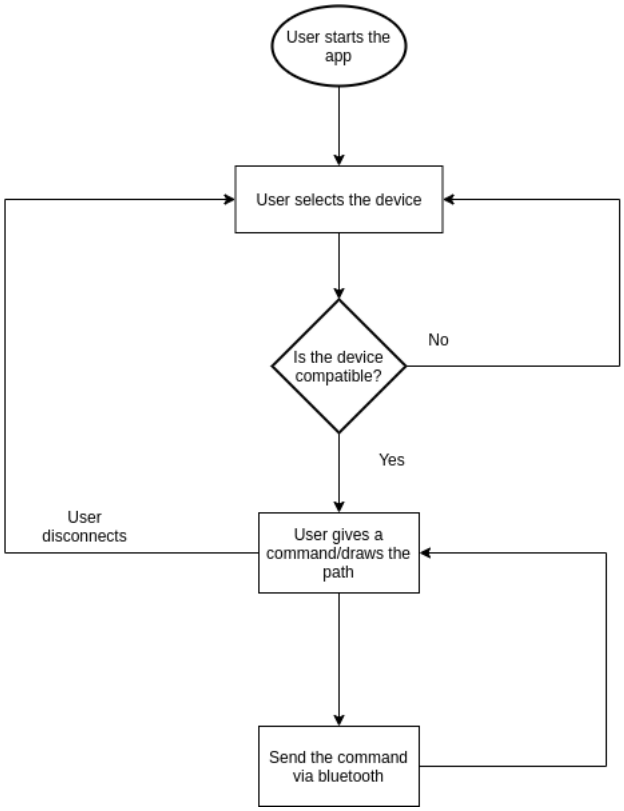

Figure 3: Android Application flowchart

4.3.2 User interaction

When the user opens the application the bluetooth capabilities of the phone are enabled and the user is presented with a list of paired devices.

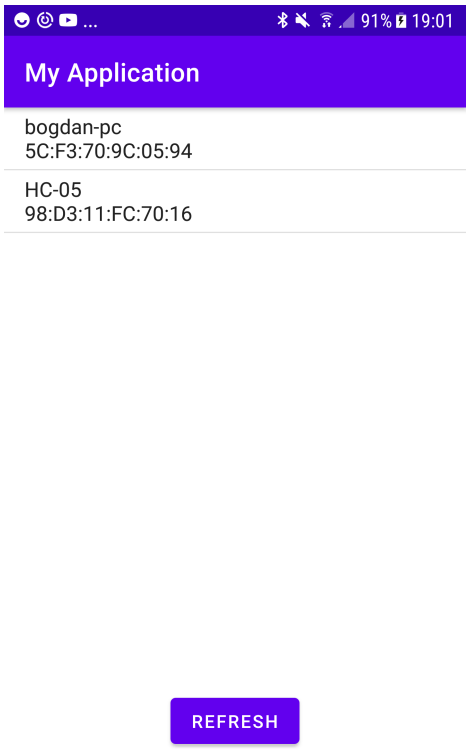

Figure 4: The device selection window

When he selects the device then the phone will attempt to connect to the device. If this fails it will create a notification on the bottom side and is redirected to the list. If the connection is established successfully then the user is presented with a couple of buttons and a space to draw.

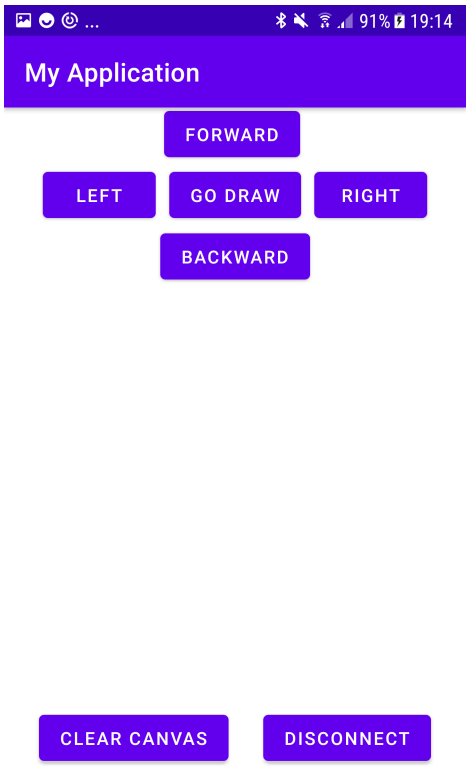

Figure 5: The command window

The buttons do the following:

- Forward -> makes the car move forward
- Backward -> makes the car move backward
- Left -> makes the car rotate to the counter clockwise(approximately 30°)
- Right -> makes the car rotate to the clockwise(approximately 30°)
- Go Draw -> it will make the device go into the trajectory drawn by the user
- Clear Canvas -> It will erase the existing drawing
- Disconnect -> it disconnects the phone from the device and allows a new device to be picked

4.3.3 Implementation

The Bluetooth communication is implemented as an Asynchronous task which maintains the opened connection to the device and is capable of sending simple messages just like any other serial console.

```
protected Void doInBackground (Void... devices) {
            try {
                if ( btSocket==null || !isBtConnected ) {
                    myBluetooth = BluetoothAdapter.getDefaultAdapter();
                    BluetoothDevice device = myBluetooth.getRemoteDevice(address);
                    btSocket = device.createInsecureRfcommSocketToServiceRecord(myUUID);
                    BluetoothAdapter.getDefaultAdapter().cancelDiscovery();
                    btSocket.connect();
                }
            } catch (IOException e) {
                ConnectSuccess = false;
            }

            return null;
}
```

The command buttons(Forward,Backward,Left,Right) are mapped to send the letter intended for that specific action as explained in the Arduino code.

```
up.setOnClickListener(v -> sendSignal("u"));
down.setOnClickListener(v -> sendSignal("d"));
left.setOnClickListener(v -> sendSignal("l"));
right.setOnClickListener(v -> sendSignal("r"));
...
private void sendSignal ( String command ) {
        if ( btSocket != null ) {
            try {
                btSocket.getOutputStream().write(command.getBytes());
            } catch (IOException e) {
                msg("Error");
            }
        }
}
```

However the drawing canvas is designed as a custom view which keeps track of all the points the user has inputed by using the touch screen and when the command is issued then the list of points is used to compute all the direction vectors it has to follow computes their angles using the scalar products and determine how many turns it has to take based on the angles and determines the orientation via the cross product between the previous and the current vector. It is assumed that the car is oriented forward.

5 Conclusions

The most challenging part of the project were creating the routines to compute the path as a series of commands to be transmitted to the robot, but overall it was a challenging experience having to work with both the Arduino micro controller and the Android software development process.