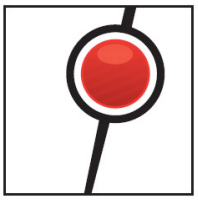


İstanbul
Bilgi Üniversitesi

LAUREATE INTERNATIONAL UNIVERSITIES

Istanbul Bilgi University

Faculty Of Engineering

Computer Engineering

Bluetooth - WIFI Communication Car

By:

Mehmetcan Güleşçi - 112200032

Supervised by:

Yrd. Doç Dr. Murat Orhun

May 21, 2017

Contents

List of Figures

List of Tables

1 Abstract

As we known, mobile platform has become a limb which all humanity from young to old cannot give up. Therefore, it is the case that an application made on this platform will be more interesting, enjoyable and more useful for them.

The development of smartphones offers a preliminary opportunity for low cost robots. Some of them are the WIFI, Bluetooth and camera systems which will be used in the project. Besides, it has a stronger CPU. This means that it expresses speed in terms of data communication.

In this project, Android supported device is used as a robot brain. At this point, it is possible to talk about two kinds of communication. One of these is the communication between our WIFI Module and the smartphone, the other one is on the basic Android application which we will use and Arduino with serial communication. When we consider these things, my first aim is to be able to command and control the Robot by using Bluetooth, the second one is to be able to taking live view from the camera by using WIFI Module. This was a simple abstract for this projects. These details will be discussed in this report clearly.

2 Introduction

A benefit of having a Bluetooth connection is that the establishment of the connection will be automatic once the device is paired with the Bluetooth node in the car. Another advantage with the Bluetooth technology is that it is an open standard, which means that if the Bluetooth devices all follow this standard they should be compatible with each other. Other advantages with the Bluetooth technology that are especially good for the car environment are the small physical size and the small power consumption.

In this way, we can say that it is possible to control a robot using a Bluetooth connection. The Bluetooth module in our robotic car will be connected to the L298P Motor Shield attached to the Arduino. In this system, the person who will manage the robot is the user who will use the phone. When the user sends control signals using the forward, backward, left and right buttons via android phones, these signals will be transmitted by Bluetooth and captured the current view from the HD Camera by using WIFI module which will be mounted on the robot. This signal will be transmitted to the controller which is connected to it so to the Arduino via the Motor driver shield. These controllers will analyze the signal. The robot will be moved according to the forward, backward, right, left commands which is sent by the user via Android. For this reason it will be possible to move the robot on four sides.

As a result, this project will be provided by two communication Bluetooth, WIFI. Thanks to Bluetooth, motion will be provided. The camera which is mounted on robot will capture the live video from WIFI and send it to the Android user. This will give the current position of the robot. Based on that video we can determine whether we need to move the robot forward, backward, right or left.

From/To Android	From/To HD Camera	On WIFI Module	On - From/To Motor Shield (UNO R3)	On Bluetooth
Signal to Android from HD Cam (1)	Signal from HD Cam to Android (2)	●	●	
Signal from Android to Motor Shield (3)			Signal to Motor Shield from Android (4)	●

Table 1: Communication Tables

2.1 Communication Between Devices

When we look at the table above:

1 To get Cam view from HD Cam on WIFI Module (to take IP) and Motor Shield (in terms of power).

2 To send Cam view to Android.

3 To send button motion to motor.

4 To get button motion from Android on Bluetooth

2.2 Objectives

- To enable inserting Arduino Code to give first motion to the car.
- To enable Android buttons to control the car as remoted.
- To enable getting live view on the HD Cam by using WIFI Module and display the robot eyes on Android Screen.

3 Processes of Car

I want to examine the car's processes in two parts. One of them is primarily hardware. The hardware side will include the controller cards that will be used in our remote controlled car, what they are and their detailed features.

The other part is the software that will give the necessary action to the car. I will include to this section in second semester. Now we will talk about hardware.

3.1 Hardware Parts

3.1.1 Requirements

I have tried to show the modules for motors which are necessary for their working mechanism and their numbers within the scope of this project on the table below.

Number	*Needed*
1	UNO R3
1	L298P Motor Shield
1	Digital WIFI Module
1	HD Camera
1	HC05 Bluetooth Module
1	DC Motors

Table 2: Requirements

Now I want to talk about these pieces. What are they? What do they do?

3.1.1.1 Arduino UNO R3

What is Arduino ?

We can think of it as an electronic unit like our brain, like our computer's processor. This unit is a microcontroller that makes what you want when you want it, or controls physical inputs with various sensors and converts these inputs into the desired output. Thanks to sensors and activators which are connected to Arduino, you can do whatever you want. By connecting the Arduino to our computer we can view the sensor data and transactions and control the equipment from the computer. This requires only a little electronic knowledge a little programming knowledge.

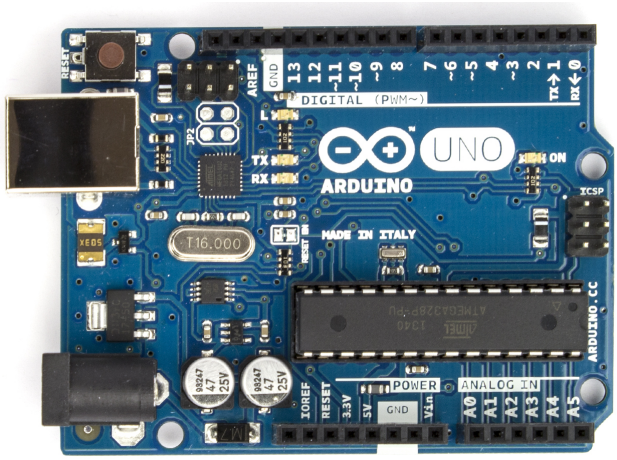


Figure 1: Arduino Uno R3

Arduino which is an open source electronic development environment spreads rapidly. This software is a combination of language C ++ and Java languages and provides us with many advantages and convenience. Writing a program with Arduino is simpler and thanks to the built-in interface you can develop the program without difficulty in the Arduino programming language.

Other features that highlight Arduino's from other development environments

- Easy to use.
- Communicating with everything by following various methods (computer, television, internet, GPS, loudspeaker, sensors, smart phones, cameras, motors ...).
- Rich library support and many example applications.
- Ideal for prototyping

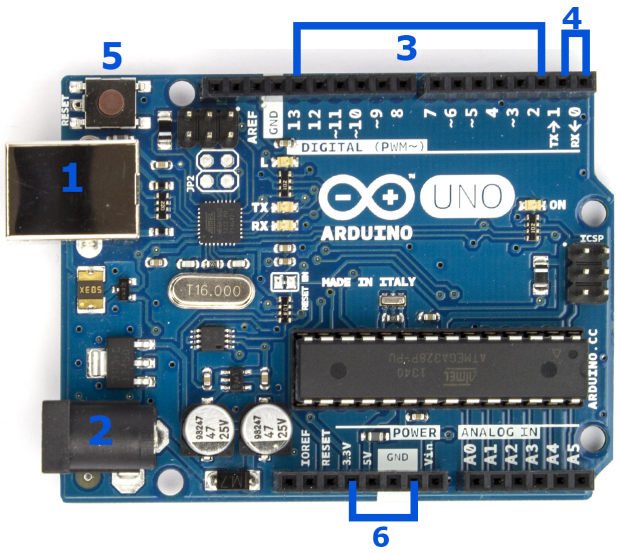


Figure 2: Arduino detailed Review

Power

(1) USB: We can send the code to the Arduino board thanks to it.

(2): Known as Arduino's external power supply (7-12V)

VIN: The voltage input used when an external power supply is connected to the Arduino Uno card.

(6) 5V: This pin provides 5 V output from the regulator on the Arduino board. The card can be powered from the DC power jack (part 2) with a 7-12 V adapter, 5 V from the USB jack (part 1) or 7-12 V from the VIN pin. The voltage supply from the 5V and 3.3V pins disregards the regulator and damages the card.

(6) 3.3V: The 3.3V output from the regulator on the Arduino card. The maximum is 50mA.

(6) GND: Ground pin.

Inputs and Outputs

All 14 digital input / output pins in Arduino Uno can be used as input or output with pinMode (), digitalWrite () and digitalRead () functions. These pins work with 5V.

(4) 0(RX) - 1(TX): These pins to receive data (receive - RX) and transmit (transmit - TX).

(3) PWM 3, 5, 6, 9, 10, and 11: These pins provide an 8-bit PWM signal with the analogWrite () function.

(5) RESET: To reset the microcontroller. It is usually used to add a reset button on the shield.

3.1.1.2 L298P Motor Shield

L298P Motor Drive integration is a full-bridge motor driver based on Arduino Uno. It can drive two separate 2A DC motors or one 2A stepper motor. It can be directly plugged into the Arduino.

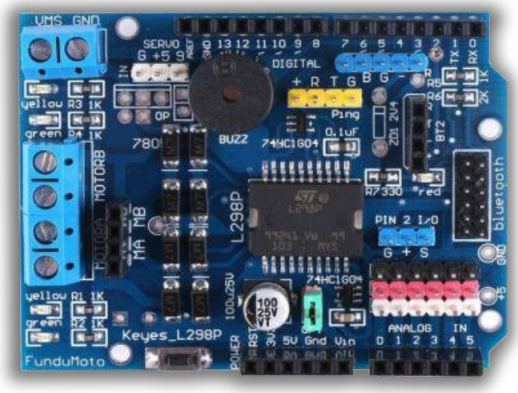


Figure 3: L298P Motor Shield

Features

- Take Arduino's data.
- On board buzzer (D4), you can set the astern alarm ringtone.
- Convenient motor interface can be two routes motor output.
- Two-way Bluetooth interface requires no wiring and you can plug directly.
- It has six analog interfaces (A0, A1, A2, A3, A4, and A5).
- It has seven digital interface that are not occupied (including D2, D3, D5, D6, D7, D8, and D9).

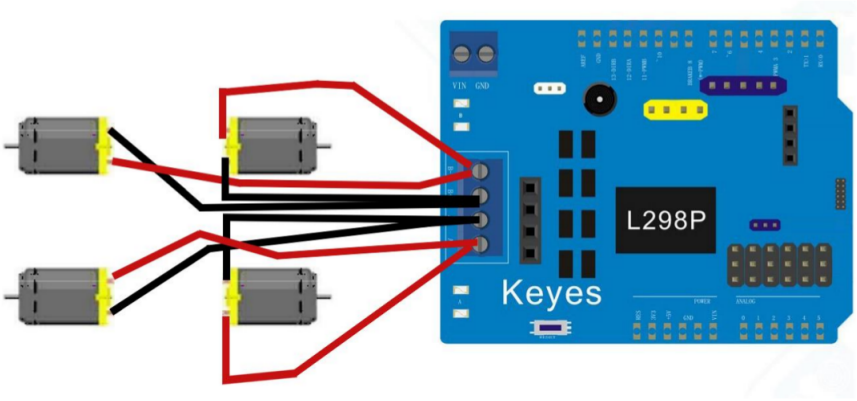


Figure 4: Connect L298P to DC Motors

Technical Specifications

- Operating Voltage: 5V-12V
- L298P Motor Controller: Drive 2 separate DC motors or 1 step motor
- Max. Current: Per 2A channel (with external supply)

Speed control function are connected with the 10, 11 interfaces. Direction control function are connected with the 12, 13 interfaces. And buzzer (alarm) function is expressed with the 4 interface. This is shown as the following table:

Function	*Channel A Pin*	*Channel B Pin*
Direction	D12	D13
PWM	D10	D11

Table 3: Shield pin usage table

3.1.1.3 Digital WIFI Module

- Thanks to this card, our Android application will be connected to Cam port thanks to IP which is in this module (may be 192.168.1.1).
- The usb input on it allows us to connect our HD Cam.

Figure 5: Digital WIFI Module

- The cables from the WIFI Module are connected to the relevant parts of the L298P Motor Shield which is integrated on the Arduino. (Just to provide a power etc. GND-5V)
- The connection will be provided to android phone on the WIFI Module's IP address.

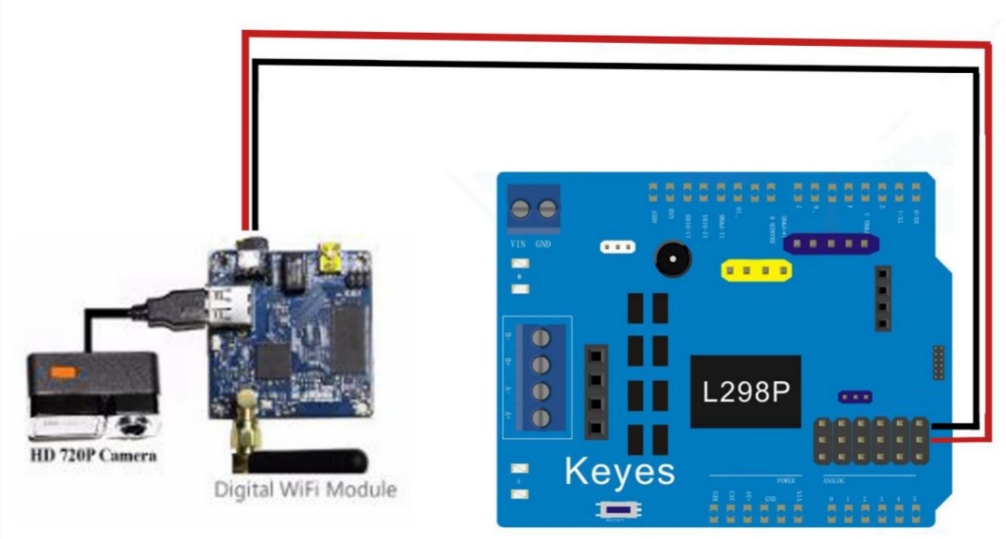


Figure 6: Digital WIFI Module and L298P Motor Shield

3.1.1.4 HD Camera

It will be possible to transfer live video to the phone with the HD camera when the necessary operation is done thanks to the usb connection which is connected to WIFI Module.

Figure 7: HD Cam

3.1.1.5 Bluetooth Module

There are 4 pins on VCC, GND, Rx and Tx on the Bluetooth module. From these VCC and GND are used to feed the Uno module. This project will be designed as sending data to Bluetooth module when certain button pressed from user. The Bluetooth module on Arduino receives the data and send to Arduino through the TX pin of Bluetooth module (RX pin of Arduino).

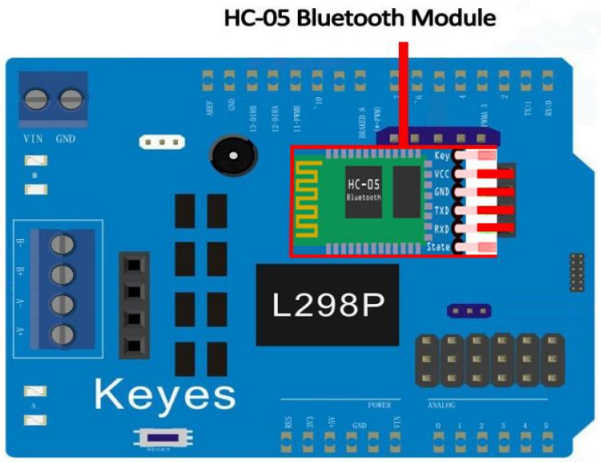


Figure 8: Bluetooth Module on L298P

3.1.1.6 DC Motors

The most preferred type of motor in robotics is DC motors. DC motors are cheap, small and effective. They are also very various in terms of size, shape and power. These are another reason to use them. Now we will explain dc motors in terms of direction, speed, voltage and current.

Direction: When a power supply is connected to the DC motors, the direction of rotation of the DC motor depends on the direction of the current. When the direction of the current is reversed, the direction of rotation of the DC motor is reversed.

Speed: The speed of a motor is measured by the number of revolutions completed in a minute. The speed of the motor depends on the voltage and the load.

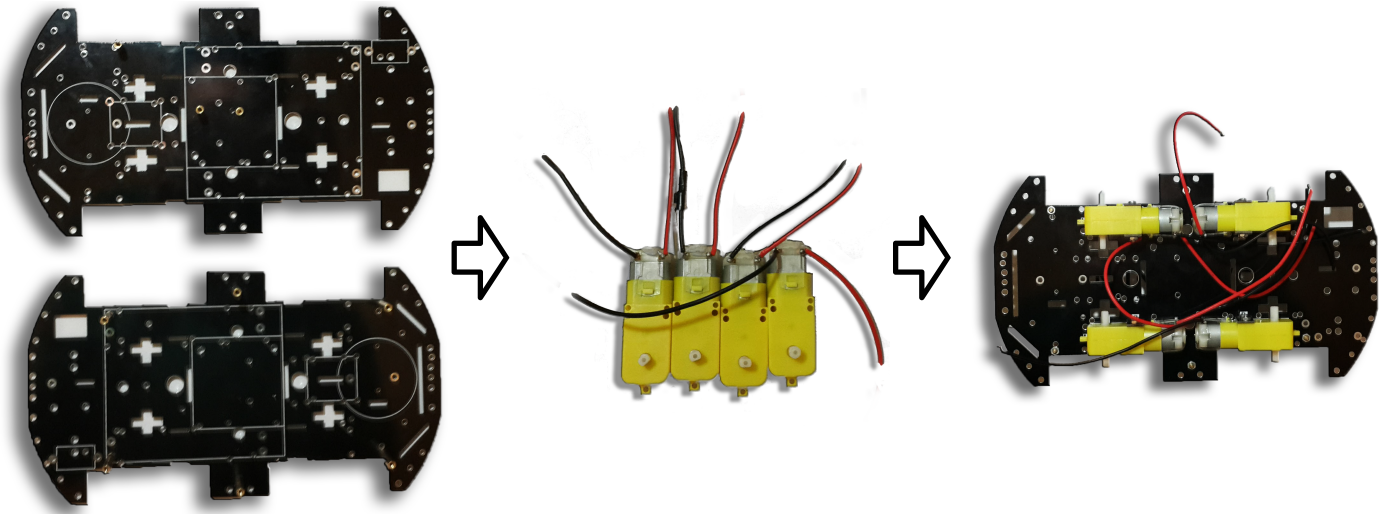
Figure 9: Dc Motor

Voltage: Small DC motors can be found with voltage values ranging from 1.5 V to 48 V. While using DC motors in robots and other systems, this voltage value is important because it determines the maximum operating voltage to be applied to the DC motor.

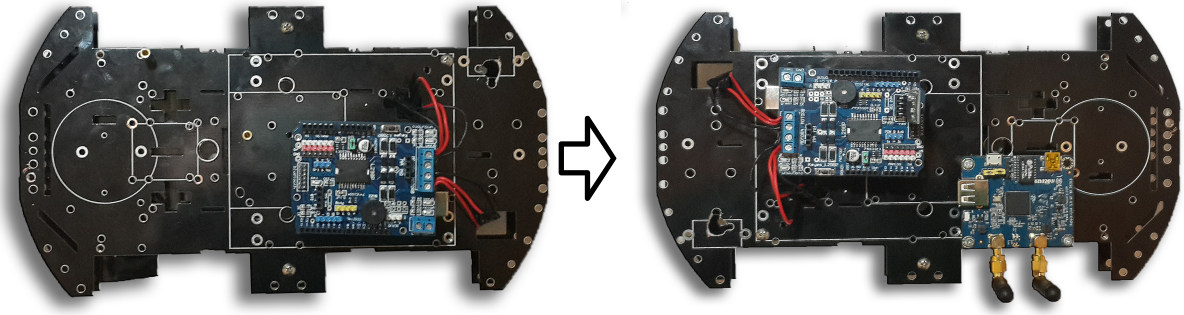
Current: When a DC motor is operated at the specified voltage, the current of the DC motor depends on the load. If the load increases, the current by the DC motor increases. The DC motor should not be overloaded to exceed the maximum current limit. In such a case, the DC motor is short-circuited and the applied power turns into heat. This may cause the DC motor to burn for long periods of time. Generally,the applied current range of DC motors can be up to 50mA and above 2A.

3.1.2 Car Building Process

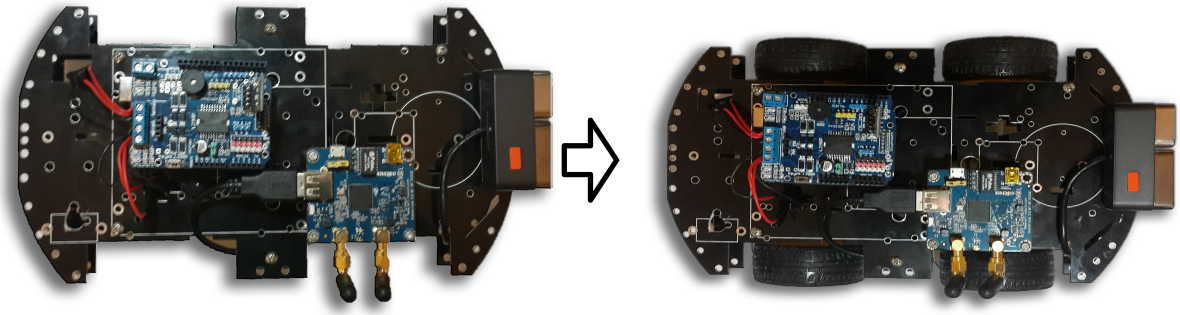
On the hardware side, the first step is to mount the dc motors which are wired, to the chassis.



The second step is to connect the dc motors to the motor driver shield. The third step is to mount the wifi shield on the chassis.



The fourth step is to connect the HD Camera to the chassis. In the fifth step, the wheels are finally mounted and the car is ready.



3.2 Software Parts

3.2.1 Arduino Programming Language

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. The environment is written in Java and based on Processing and other open-source software. This software can be used with any Arduino board.

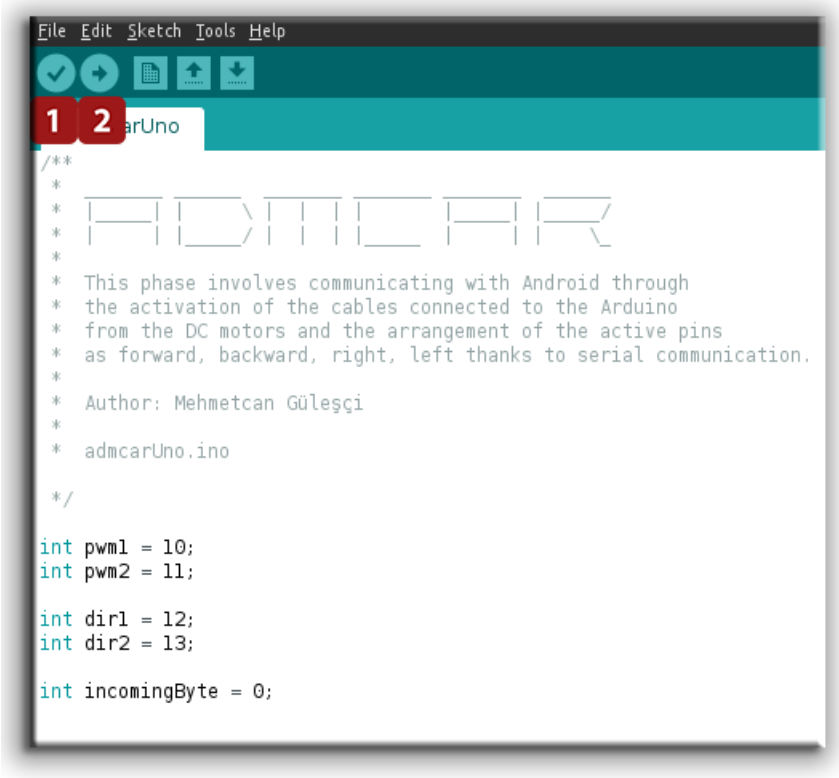


Figure 10: Arduino Usage

1. It allows us to check the Arduino code for errors.
2. It allows us to execute our Arduino code and upload to board.

Arduino Code

All my arduino codes for this project is shown below.

```
/**
 *  _______ ______  _______ _______ _______  ______
 *  |_____| |     \ |  |  | |       |_____| |_____/
 *  |     | |_____/ |  |  | |_____  |     | |    \_
 *
 *  This phase involves communicating with Android through
 *  the activation of the cables connected to the Arduino
 *  from the DC motors and the arrangement of the active pins
 *  as forward, backward, right, left thanks to serial communication.
 *
 *  Author: Mehmetcan Gulesci
 *
 *  admcarUno.ino

 */

// Initialize variables
int pwm1 = 10;
int pwm2 = 11;

int dir1 = 12;
int dir2 = 13;

int incomingByte = 0;

/*  The setup() function is called when a sketch starts. Use it to initialize
 *  variables, pin modes, start using libraries, etc. The setup function will
 *  only run once, after each powerup or * reset of the Arduino board.
 */

void setup()
{
  pinMode(pwm1, OUTPUT);
  pinMode(pwm1, OUTPUT);
  pinMode(dir1, OUTPUT);
  pinMode(dir2, OUTPUT);

  digitalWrite(pwm1, LOW);
  digitalWrite(pwm2, LOW);
  digitalWrite(dir1, LOW);
  digitalWrite(dir2, LOW);

  Serial.begin(9600);
}
```

```
45
46 // This was my method to call digitalWrite- analogWrite functions in merged file
47 void MotorKontrol(int mdir1, int mdir2, int pwmSpeed)
48 {
49   digitalWrite(dir1, mdir1);
50   digitalWrite(dir2, mdir2);
51   analogWrite(pwm1, pwmSpeed);
52   analogWrite(pwm2, pwmSpeed);
53 }
54
55 void loop()
56 {
57   // This expresses the motion part with serial communication
58   if (Serial.available() > 0)
59   {
60     incomingByte = Serial.read();
61
62     if (incomingByte == 10) // Forward
63     {
64       MotorKontrol(HIGH, HIGH, 170);
65     }
66     else if (incomingByte == 20) // Backward
67     {
68       MotorKontrol(LOW, LOW, 170);
69     }
70     else if (incomingByte == 30) // Left
71     {
72       MotorKontrol(HIGH, LOW, 170);
73     }
74     else if (incomingByte == 40) // Right
75     {
76       MotorKontrol(LOW, HIGH, 170);
77     }
78     else // Stop if another data comes
79     {
80       MotorKontrol(LOW, LOW, 0);
81     }
82   }
83 }
```

3.2.2 Android IDE

3.2.2.1 Android Application All Part of Code

This field contains everything from the birth to the end of the Android App. The design consists of 3 separate layouts.

- `activity_intro.xml`
- `activity_bluetooth_list.xml`
- `activity_car.xml`

≻ As we know, each screen designed in layout includes its own software. So it is necessary to open the class at least as much as it is. The first of these is the `SplashActivity.java` class, which corresponds to the `activity_intro.xml` layout.When we open the app , we encounter this activiy first time, and it goes to the main layout with the transition time of 3 seconds.

```
1  package com.example.mgulesci.admcar;
2
3  import android.content.Intent;
4  import android.os.Bundle;
5  import android.support.v4.app.FragmentActivity;
6
7  /**
8   * This class refers to intro screen of my application
9   */
10 public class SplashActivity extends FragmentActivity {
11     public Thread myThread;
12
13     @Override
14     protected void onCreate(Bundle savedInstanceState) {
15         super.onCreate(savedInstanceState);
16         setContentView(R.layout.activity_intro);
17
18         myThread = new Thread() {
19             @Override
20             public void run() {
21                 try {
22                     sleep(3000);
23                     // Open new intent after elapsed time
24                     Intent intent = new Intent(getApplicationContext(),
                           BluetoothList.class);
25                     startActivity(intent);
26                     finish();
27                 } catch (InterruptedException e) {
```

```
                    e.printStackTrace();
                }
            }
        };
        myThread.start();
    }
}
```

≻ The second of these is the `BluetoothList.java` class, which corresponds to the `activity_bluetooth_list.xml` layout. The first thing to do in this class is to see the bluetooth feature of the phone and to see the bluetooth paired devices with the adapter methods on related layout.

```
package com.example.mgulesci.admcar;

import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.Intent;
import android.os.Bundle;
import android.support.v4.app.FragmentActivity;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

import java.util.ArrayList;
import java.util.Set;


public class BluetoothList extends FragmentActivity {

    public static String EXTRA_ADDRESS = "device_address";
    private TextView text;
    private ListView listOfDevices;
    private Button btn_list;
    private BluetoothAdapter phoneBluetooth = null;
    private Set<BluetoothDevice> pairedDevices;

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

```

```
        this.initializeComponents();
        this.bluetoothControl();
        this.initializeListeners();

    }

    /**
     * Control the bluetooth whether it exists or not.
     */
    private void bluetoothControl(){
        // Take our phone's bluetooth
        phoneBluetooth = BluetoothAdapter.getDefaultAdapter();

        // Control, if your phone has bluetooth or not
        if (phoneBluetooth == null) {
            // If bluetooth is not available, give warning message and close application
            message("Your phone does not support bluetooth");
            finish();
        } else if (!phoneBluetooth.isEnabled()) {
            // If you have a bluetooth but it is not open, request to connect.
            Intent BTac = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
            startActivityForResult(BTac, 1);
        }
    }

    /**
     * Initialize the components
     */
    private void initializeComponents(){
        // Describing XML Widgets
        text = (TextView) findViewById(R.id.textView);
        listOfDevices = (ListView) findViewById(R.id.listView);
        btn_list = (Button) findViewById(R.id.ButtonTest);
    }

    /**
     * Initialize the listeners
     */
    private void initializeListeners(){
        btn_list.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                showPairedDevices();
            }
        });
    }

```

```
81      /**
82       * This method allows us to show paired devices and
83       * open new intent(CarActivity) by clicking anyone(list items).
84       */
85      private void showPairedDevices() {
86          // Take paired devices
87          pairedDevices = phoneBluetooth.getBondedDevices();
88          ArrayList list = new ArrayList();
89
90          if (pairedDevices.size() > 0) {
91              for (BluetoothDevice bt : pairedDevices) {
92                  // Add the name and address of the Bluetooth device to the list.
93                  list.add(bt.getName() + "\n" + bt.getAddress());
94              }
95          } else {
96              message("Paired devices not found, Be sure bluetooth connection is open..
                    ");
97          }
98
99          final ArrayAdapter adapter = new ArrayAdapter(this,
                android.R.layout.simple_list_item_activated_1, list);
100
101         listOfDevices.setAdapter(adapter);
102
103         // Method that allows us to select desired devices to connect.
104         listOfDevices.setOnItemClickListener(new AdapterView.OnItemClickListener() {
105             @Override
106             public void onItemClick(AdapterView<?> parent, View view, int position,
                        long id) {
107                 // We get the mac address, the last 17 characters in the view.
108                 String info = ((TextView) view).getText().toString();
109                 String address = info.substring(info.length() - 17);
110
111                 // We define an intent to start a new activity.
112                 Intent i = new Intent(BluetoothList.this, CarActivity.class);
113
114                 // Start the activity.
115                 i.putExtra(EXTRA_ADDRESS, address); // this will be received from
                        CarActivity class
116                 startActivity(i);
117             }
118         });
119     }
120
121     private void message(String msg) {
122         Toast.makeText(getApplicationContext(), msg, Toast.LENGTH_LONG).show();
123     }
```

```
}
```

➢ The third of these is the `MjpegStream.java`. This class begins by describing the Mjpeg Input Stream, which is not natively supported by Android. In doing so, it reads each next frame coming from the input stream. `URI` (Uniform Resource Identifier - `http://192.168.1.1:8080/?action=stream`) is sent to the client and an http entity is created with the http response incoming from the client. The input stream in this http entity is sent into the `Buffered Input Stream` which is refilled many bytes at a time and then to the `Data Input Stream`. At the end of this, the URI will complete the image acquisition in the background.

```
package com.example.mgulesci.admcar;

import android.graphics.Bitmap;
import android.graphics.BitmapFactory;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;

import java.io.BufferedInputStream;
import java.io.ByteArrayInputStream;
import java.io.DataInputStream;
import java.io.IOException;
import java.net.URI;
import java.util.Properties;

public class MjpegStream implements Runnable {

    // Typical max length of header data. (Maximum header length)
    private final static int HEADER_MAX_LENGTH = 100;

    // Expected length of an mjpeg frame. (Max frame length (100kB))
    private final static int FRAME_MAX_LENGTH = 40000 + HEADER_MAX_LENGTH;
    //private final String CONTENT_TYPE_PREFIX = "multipart/x-mixed-replace;boundary=";

    // Name of content length header.
    // Optional MJPEG frame header key used to indicate bytes of jpeg file data.
    // This header is optional and depends on the API call used with the camera.
    private final String CONTENT_LENGTH = "Content-Length";

    // The first two bytes of every JPEG stream are the Start Of Image (SOI) marker
        values FFh D8h.
    // Start Of Image marker. Size: 2 bytes The first two bytes of every image.
    private final byte[] SOI_MARKER = { (byte) 0xFF, (byte) 0xD8 };
```

```
35
36      // End Of Image (EOF), size: 2 bytes The last two bytes of every JPEG image. (FFh
            D9h)
37      private final byte[] EOF_MARKER = { (byte) 0xFF, (byte) 0xD9 };
38      //private String mBoundary;
39      private int mContentLength = -1;
40      private String mUrl;
41      private boolean mRun;
42      private Callback onFrameReadCallback;
43
44      public MjpegStream(String url) {
45          mUrl = url;
46          mRun = false;
47          onFrameReadCallback = null;
48      }
49
50      public void start() {
51          mRun = true;
52          new Thread(this, "MJPEG").start();
53      }
54
55      public void stop() {
56          mRun = false;
57      }
58
59      public void setCallback(Callback callback) {
60          onFrameReadCallback = callback;
61      }
62
63      /**
64       * @param in
65       * @param sequence (ID)
66       * @return The index of the first byte after the given sequence, or -1 if not found
67       * @throws IOException
68       */
69      private int getEndOfSequence(DataInputStream in, byte[] sequence) throws
            IOException {
70          int seqIndex = 0; //tracks number of sequence chars found
71          byte c;
72          for(int i=0; i < FRAME_MAX_LENGTH; i++) {
73              c = (byte) in.readUnsignedByte(); //read next byte
74              if(c == sequence[seqIndex]) {
75                  seqIndex++; //increment seq char found index
76                  //check if we have the whole sequence
77                  if(seqIndex == sequence.length)
78                      return i + 1;
79              } else
```

```
80                  //reset index if we don't find all sequence characters before breaking
81                  seqIndex = 0;
82              }
83              return -1;
84          }
85
86          /**
87           * @param in
88           * @param sequence (ID)
89           * @return Get the index of of the beginning of the sequence
90           * @throws IOException
91           */
92          private int getStartOfSequence(DataInputStream in, byte[] sequence) throws
                IOException {
93              int end = getEndOfSequence(in, sequence);
94              return (end < 0) ? (-1) : (end - sequence.length);
95          }
96
97
98          /**
99           * Get the content length from the input stream.
100          * Parse the content length string for a MJPEG frame from the given bytes. The
                string is parsed into an int and returned.
101          *
102          * @param headerBytes
103          * @return int
104          * @throws IOException
105          * @throws NumberFormatException
106          */
107         private int parseContentLength(byte[] headerBytes) throws IOException,
                NumberFormatException {
108             ByteArrayInputStream headerIn = new ByteArrayInputStream(headerBytes);
109             Properties props = new Properties();
110             props.load(headerIn);
111
112             return Integer.parseInt(props.getProperty(CONTENT_LENGTH));
113         }
114
115
116         /**
117          * Read the next MjpegFrame from the stream.
118          *
119          * @return The next MJPEG frame.
120          * @throws IOException (If there is an error.)
121          */
122         public Bitmap readFrame(DataInputStream in) throws IOException {
123             //int mContentLength = -1;
```

```
124
125         in.mark(FRAME_MAX_LENGTH);
126         int headerLen = getStartOfSequence(in, SOI_MARKER);
127         in.reset();
128         byte[] header = new byte[headerLen];
129         in.readFully(header);
130         try {
131             mContentLength = parseContentLength(header);
132         } catch (NumberFormatException nfe) {
133             mContentLength = getEndOfSequence(in, EOF_MARKER);
134         }
135         in.reset();
136         byte[] frameData = new byte[mContentLength];
137         in.skipBytes(headerLen);
138         in.readFully(frameData);
139         return BitmapFactory.decodeStream(new ByteArrayInputStream(frameData));
140     }
141
142     public void run() {
143         // Uniform Resource Identifier (URI)
144         // Uniform Resource Locator (URL)
145         URI uri = URI.create(mUrl);
146         DefaultHttpClient httpClient = new DefaultHttpClient();
147
148         HttpResponse httpResponse = null;
149         try {
150             httpResponse = httpClient.execute(new HttpGet(uri));
151         } catch (IOException e) {
152
153             return;
154         }
155
156         HttpEntity httpEntity = httpResponse.getEntity();
157
158         /**
159          * BufferedInputStream:
160          *
161          * As bytes from the stream are read or skipped, the internal buffer is
                refilled as
162          * necessary from the contained input stream, many bytes at a time.
163          */
164         BufferedInputStream in = null;
165         try {
166             in = new BufferedInputStream(httpEntity.getContent(), FRAME_MAX_LENGTH);
167         } catch (IOException e) {
168
169             return;
```

```
        }

        /**
         * DataInputStream:
         *
         * It lets an application read primitive Java data types from an underlying
         * input stream in a machine-independent way.
         */
        DataInputStream mjpeg = new DataInputStream(in);

        while (mRun) {
            Bitmap bitmap = null;

            try {
                bitmap = readFrame(mjpeg);
            } catch (IOException e) {

                break;
            }

            if (onFrameReadCallback != null) {
                onFrameReadCallback.onFrameRead(bitmap);
            }
        }

        try {
            mjpeg.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public interface Callback {
        public void onFrameRead(Bitmap bitmap);
    }
}
```

➢ The fourth of these is the `CarActivity.java`. We will use the relevant Bluetooth properties we selected in the previous class in this class. First of all, with the help of the sockets, the connection is provided with bluetooth which is connected to Arduino. We define individual sockets for each button (forward, backward, right, left) and we define the specifiers to distinguish them from each other. They can be string, integer (10, "w") etc.. We define each one here as we describe it in Arduino. So we get to Arduino data with bluetooth connection and provide the motion. A second occurrence in this class is to get a live image acquisition. We actualize the getting input stream process we defined in the previous class. How do we do it ? First we

start with the SurfaceView created in the xml file called by this java file. A surface is created through SurfaceHolder. Callback implemented in Java. This method creates a default image with the drawBitmap method that the Canvas object has. Activating this view is also done with MjpegStream which we call this class. Thus, the live image is taken in the direction of the given sequences.

```
package com.example.mgulesci.test;

import android.app.Activity;
import android.app.ProgressDialog;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Rect;
import android.os.AsyncTask;
import android.os.Bundle;
import android.view.MotionEvent;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.View;
import android.view.WindowManager;
import android.widget.ImageButton;
import android.widget.Toast;

import java.io.IOException;
import java.util.UUID;

public class CarActivity extends Activity implements SurfaceHolder.Callback {

    public static final UUID myUUID =
        UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
    public BluetoothAdapter phoneBluetooth = null;
    public BluetoothSocket btSocket = null;
    public String address = null;
    private ProgressDialog progress;
    private boolean isBtConnected = false;
    private ImageButton goForwardBtn;
    private ImageButton goBackwardBtn;
    private ImageButton goLeftBtn;
    private ImageButton goRightBtn;
    private SurfaceView surfaceView;
    private SurfaceHolder mSurfaceHolder;
    private MjpegStream mMjpegStream;

    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);

        setContentView(R.layout.activity_car);

        // Get the address of your bluetooth device.
        Intent newIntent = getIntent();
        address = newIntent.getStringExtra(BluetoothList.EXTRA_ADDRESS);

        new ConnectBluetooth().execute(); // Bluetooth connection

        this.initializeComponents();
        this.initializeListeners();
    }

    @Override
    public void onPause() {
        super.onPause();
    }

    @Override
    protected void onResume() {
        super.onResume();
    }


    @Override
    public void onBackPressed() {
        // Start the video stream next time the application is ran.
        this.finish();
    }

    /**
     * Initialize all view components.
     */
    private void initializeComponents() {
        surfaceView = (SurfaceView) findViewById(R.id.surfaceView);
        mSurfaceHolder = surfaceView.getHolder();
        mSurfaceHolder.addCallback(this);

        this.goForwardBtn = (ImageButton) findViewById(R.id.goForwardBtn);
        this.goBackwardBtn = (ImageButton) findViewById(R.id.goBackwardBtn);
        this.goLeftBtn = (ImageButton) findViewById(R.id.goLeftBtn);
        this.goRightBtn = (ImageButton) findViewById(R.id.goRightBtn);
    }

```

```
    /**
     * Bind all button listeners. (called during the initialization)
     */
    private void initializeListeners() {
        /*
        ******** Forward **********
        */
        this.goForwardBtn.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent event) {
                switch (event.getAction()) {
                    case MotionEvent.ACTION_DOWN:
                        goForward();
                        return true;
                    case MotionEvent.ACTION_UP:
                        stop();
                        return true;
                }
                return false;
            }
        });

        /*
        ******** Backward **********
        */
        this.goBackwardBtn.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent event) {
                switch (event.getAction()) {
                    case MotionEvent.ACTION_DOWN:
                        goBackward();
                        return true;
                    case MotionEvent.ACTION_UP:
                        stop();
                        return true;
                }
                return false;
            }
        });

        /*
        ******** Left **********
        */
        this.goLeftBtn.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent event) {
                switch (event.getAction()) {
```

```
                    case MotionEvent.ACTION_DOWN:
                        goLeft();
                        return true;
                    case MotionEvent.ACTION_UP:
                        stop();
                        return true;
                }
                return false;
            }
        });

        /*
        ******** Right **********
        */
        this.goRightBtn.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent event) {
                switch (event.getAction()) {
                    case MotionEvent.ACTION_DOWN:
                        goRight();
                        return true;
                    case MotionEvent.ACTION_UP:
                        stop();
                        return true;
                }
                return false;
            }
        });

    }

    /**
     * Send a request to the car to go forward.
     */
    private void goForward() {
        if (btSocket != null) {
            try {
                btSocket.getOutputStream().write(10);
            } catch (IOException e) {
                message("Bluetooth interrupt");
            }
        }
    }

    /**
     * Send a request to the car to go backward.
     */
```

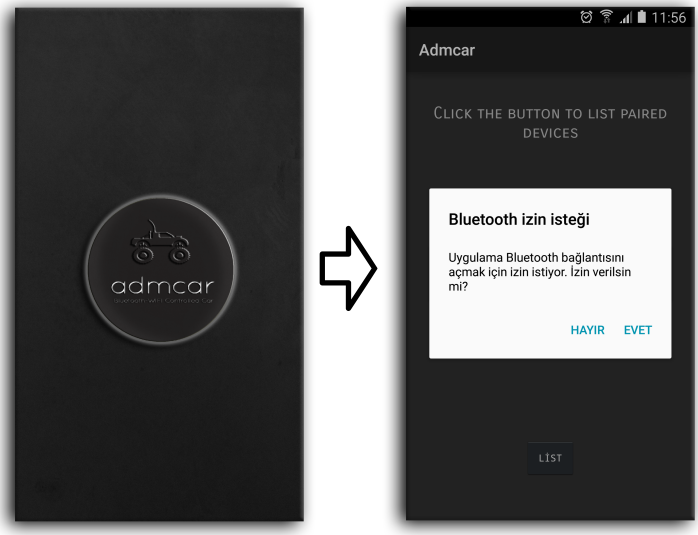
```
183     private void goBackward() {
184         if (btSocket != null) {
185             try {
186                 btSocket.getOutputStream().write(20);
187             } catch (IOException e) {
188                 message("Bluetooth interrupt");
189             }
190         }
191     }
192 
193     /**
194      * Send a request to the car to go to the left.
195      */
196     private void goLeft() {
197         if (btSocket != null) {
198             try {
199                 btSocket.getOutputStream().write(30);
200             } catch (IOException e) {
201                 message("Bluetooth interrupt");
202             }
203         }
204     }
205 
206     /**
207      * Send a request to the car to go to the right.
208      */
209     private void goRight() {
210         if (btSocket != null) {
211             try {
212                 btSocket.getOutputStream().write(40);
213             } catch (IOException e) {
214                 message("Bluetooth interrupt");
215             }
216         }
217     }
218 
219     /**
220      * Send a request to the car to stop.
221      */
222     private void stop() {
223         if (btSocket != null) {
224             try {
225                 btSocket.getOutputStream().write(50);
226             } catch (IOException e) {
227                 message("Bluetooth interrupt");
228             }
229         }
```

```
    }


    /**
     * Connecting and sending data via socket.
     */
    private class ConnectBluetooth extends AsyncTask<Void, Void, Void> {
        private boolean connectSuccess = true;

        @Override
        protected void onPreExecute() {
            progress = ProgressDialog.show(CarActivity.this, "Connecting...", "Please
                wait");
        }

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

        @Override
        protected void onPostExecute(Void result) {
            super.onPostExecute(result);
            if (!connectSuccess) {
                message("Connection error, Please try again");
                finish();
            } else {
                message("Connection successful");
                isBtConnected = true;
            }
            progress.dismiss();
        }
    }
    /**
     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
     * THIS PART INCLUDES CAMERA DESCRIPTION
```

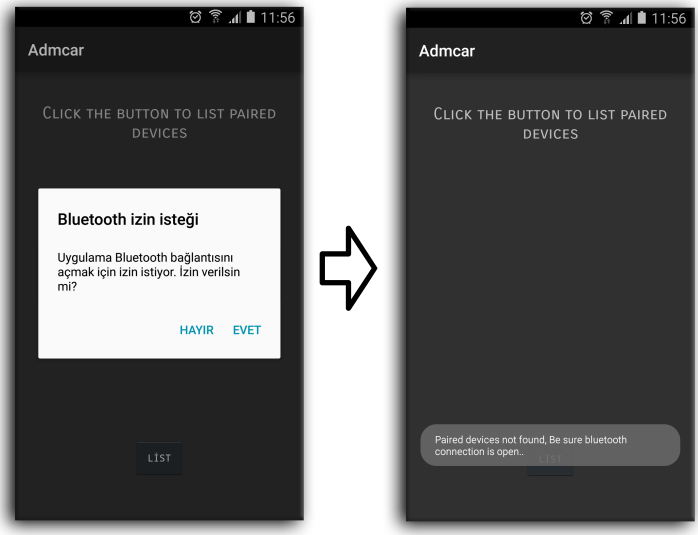
```
     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
     */
    @Override
    public void surfaceCreated(SurfaceHolder holder) {
        mMjpegStream = new MjpegStream("http://192.168.1.1:8080?action=stream");
        mMjpegStream.setCallback(new MjpegStream.Callback() {
            public void onFrameRead(Bitmap bitmap) {
                Canvas canvas = null;
                try {
                    canvas = mSurfaceHolder.lockCanvas();
                    if (canvas != null) {
                        try {
                            canvas.drawBitmap(bitmap, null, new Rect(0, 0,
                                canvas.getWidth(), canvas.getHeight()), null);
                        } catch (Exception e) {
                        }
                    }

                } finally {
                    if (canvas != null) {
                        mSurfaceHolder.unlockCanvasAndPost(canvas);
                    }
                }
            }
        });
        mMjpegStream.start();
    }

    @Override
    public void surfaceChanged(SurfaceHolder holder, int format, int width, int height)
        {
    }

    @Override
    public void surfaceDestroyed(SurfaceHolder holder) {
        mMjpegStream.stop();
    }
    /**
     * Error message method.
     *
     * @param msg
     */
    private void message(String msg) {
        Toast.makeText(getApplicationContext(), msg, Toast.LENGTH_LONG).show();
    }
}
```

3.2.2.2 User Interface

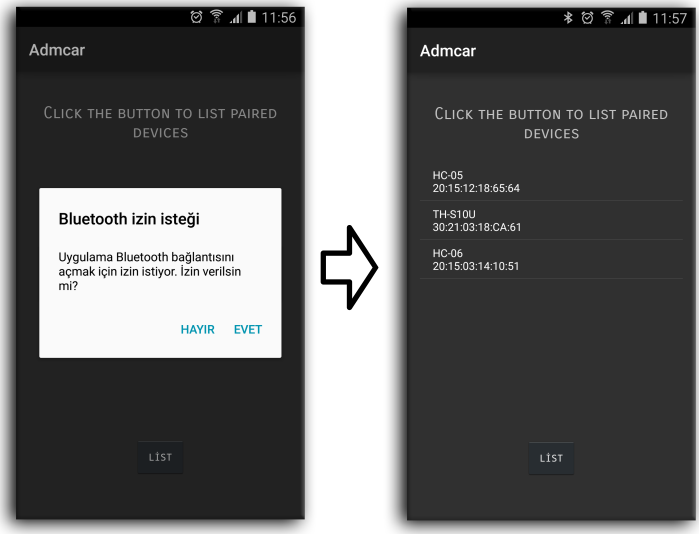
The application starts with the intro picture. Then the second intent comes out. This one requires connecting to the Bluetooth.



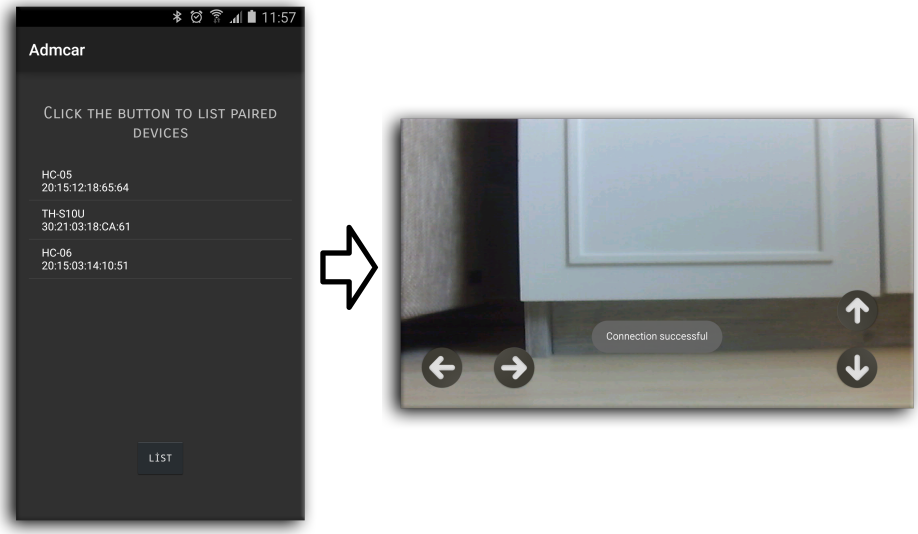
After the Bluetooth request has come out and when we click `No` option, the right window pops up and we get the warning of `'Paired devices not found. Be sure bluetooth connection is open.'`.



In the Bluetooth permission request, if you press `Yes` and then press the `List` button, we will see the list of paired devices.



As a final, when you click to bluetooth you are connected in, you will see the last intent to control the car. In this last intent, besides the view of the camera, it will be possible to control the car with the buttons.
Hint: Left side buttons express left-right direction of car and right side buttons express forward-backward direction of car



4 View Models for Application

4.1 Class Diagrams

This class that provides image transition time with the help of threads.

Figure 11: Intro Screen Class Diagram

It is a class that provides Bluetooth access to the Bluetooth feature of the phone with the help of Bluetooth adapters and makes some kind of communication with the Arduino Bluetooth via the Sockets.

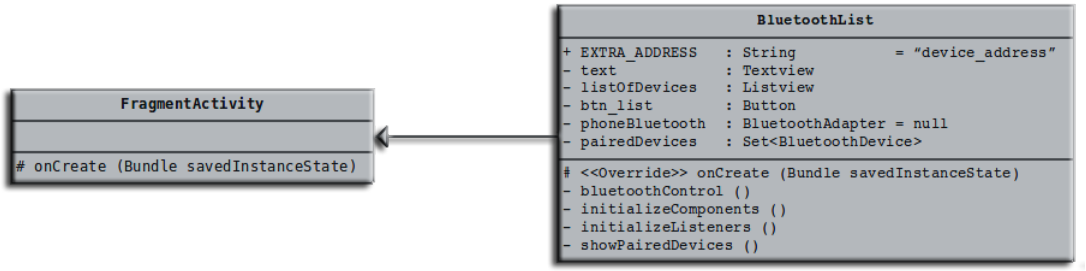


Figure 12: Bluetooth Class Diagram

This one contains classes that control the camera, including video stream. Basically, everything is managed from the MjpegStream, it's a view specialized to deal with Mjpeg video stream format, because Android doesn't support natively this format.

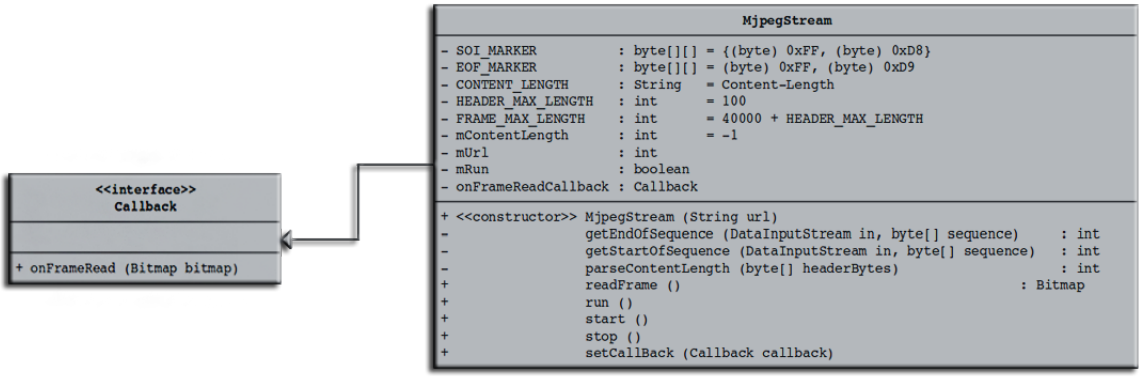


Figure 13: Camera Class Diagram

This class contains all classes in relationship with the car. The Car class is basically a controller for the car, it's this class that will control the direction via bluetooth sockets and monitor the live video stream.

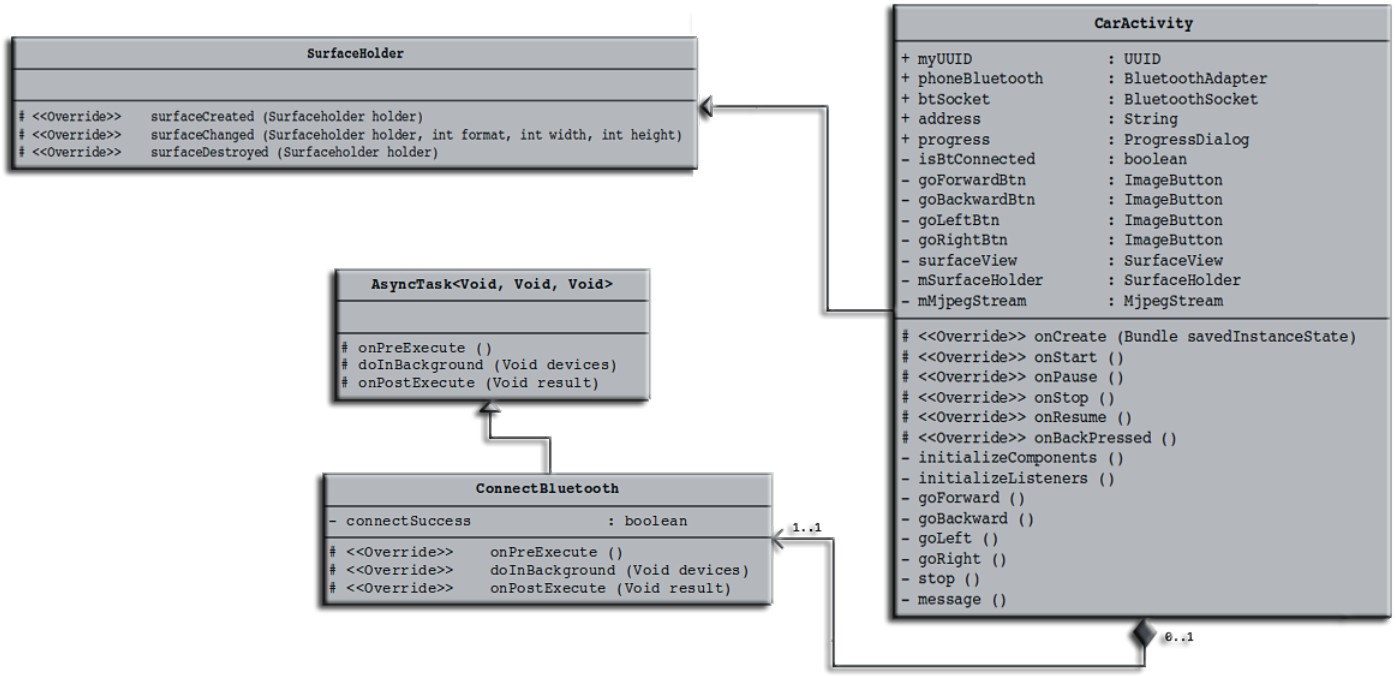


Figure 14: Car Activity Class Diagram

4.2 Use Case Diagram

To describe the system functionalities, we use the UML Use case diagrams. The Use Cases diagrams illustrated below show the actors that show also which kind of action they can use.

This is the first version of the system, which can basically only move, the details about all the possible movements are described too.

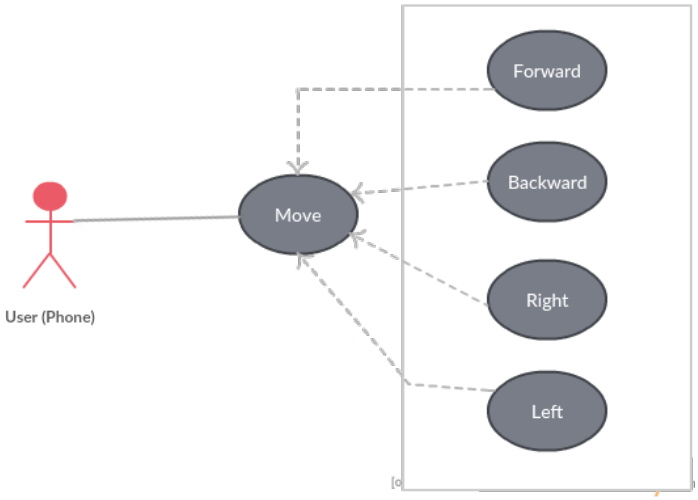


Figure 15: Use Case - Version1

The second version is a major release of the system because we will reach our goal that is to have a video stream on the phone and control the car almost in real time with good performances.

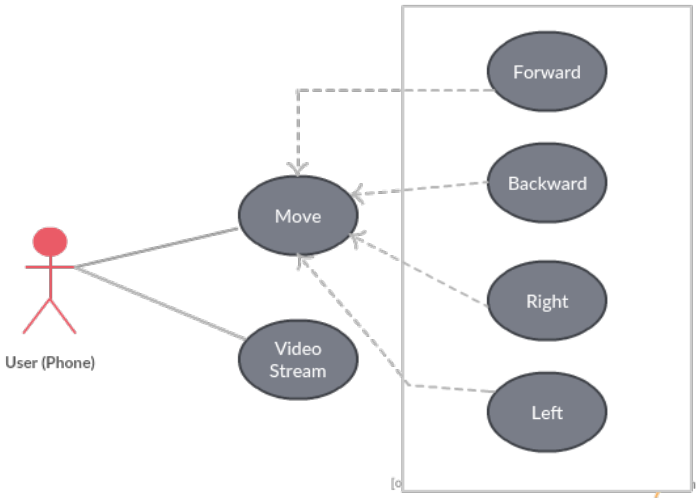


Figure 16: Use Case - Version2

5 Conclusion

Everything that is smart makes our life easier and more meaningful. Thanks to Arduino Technology, there are now so many systems that make people's lives easier and almost inventive, which is impossible to catch up with. In this project, which is considered to be my thesis, I accomplished to design my own car using Arduino and to use it in mobile environment and watch it.

Of course, we need communications to activate these clever things. I used Bluetooth and WIFI technologies that is supported by smartphones.

I have learned many things in several parts, hardware or software, even project management, this project helped me, to better understand the communication with hardware and software. I am used to develop software and I am not used to deal with hardware, it's also really complicated many information and documentation to read, to be aware of. You cannot really break software, it doesn't happen often, but it's really easy to break hardware for instance, I discovered that.

Managing a stand-alone project; It is very difficult to be both a manager and an employee of the project. During this process, I am really tired and confused. However, I have completed the project successfully and in the most efficient way.

5.1 User profile

The users that will be play with the system could be child or adults, they just have to know how to use a smartphone, launch an application and push some buttons. There is no really limitation about the age while they know how to use the phone.

5.2 Strengths and weaknesses

5.2.1 Strengths

- The project works, we are able to do what we wanted.
- The source code for both Android and Arduino is commented, well explained and documented with external documents.
- All the source code and the documentation are available for future usages for other people and free. (Git)
- Good and speed video stream.
- With the video streaming width designed to fill the user's phone screen, the user will be able to control the car comfortably.
- It connects to Bluetooth without difficulty and never breaks unless we are away from a certain distance.

5.2.2 Weaknesses

- The product needs two separated power sources.
- Sometimes WiFi can shut down automatically. In this case, the video is freezing because WiFi is off.
- Really short range with video stream. (8 meters)

5.3 Suggested improvements (Future of system)

- Use external antenna for greater range.
- Use only one source of power, so use the big battery used by the motor for both motor and Arduino using the regulator.
- Study and fix bugs on the applications.
- Improve the Android application to be able to get the video stream and the car control after lost them (out of range) automatically.
- Add sounds to warn the user on some events such as "out of range", "video lost", etc.
- Improve the application with new modules such as proximity detection using sensor.
- Improve Android application with a different way to control the car using sensors instead of buttons and propose the choice to the user between both.
- Some sensors of the Arduino can be added to bring joy to the car. *(honk-buzzer, distance-ultrasonic, headlights-lamp sensor etc.).*

6 Time Table and Work Schedule

6.1 Gantt Chart

Task Name	Start	Finish
Time and Working Schedule		
Decide Process	10/18/16	10/15/16
Research about the working mechanism	10/15/16	10/22/16
Learning of Materials	10/27/16	11/09/16
Research what I use	11/13/16	11/27/16
Decide what I use	12/01/16	12/05/16
Research about communication between Android & Arduino via WIFI	12/07/16	12/30/16
Process of Providing Materials	01/15/17	01/20/17
Building the Body	01/21/17	01/28/17
Integrating whole Parts of Body	02/03/17	02/04/17
Software Parts (Communication Android & WIFI)	02/04/17	02/09/17
Software Parts (Communication Android & Arduino Codes)	02/11/17	02/26/17
Combining all functions	02/28/17	03/04/17
Testing all parts and system (Margin of error and editions))	03/06/17	05/23/17
Last controls	05/23/17	05/25/17
Submitting and Presentation Process	05/27/17	06/15/17

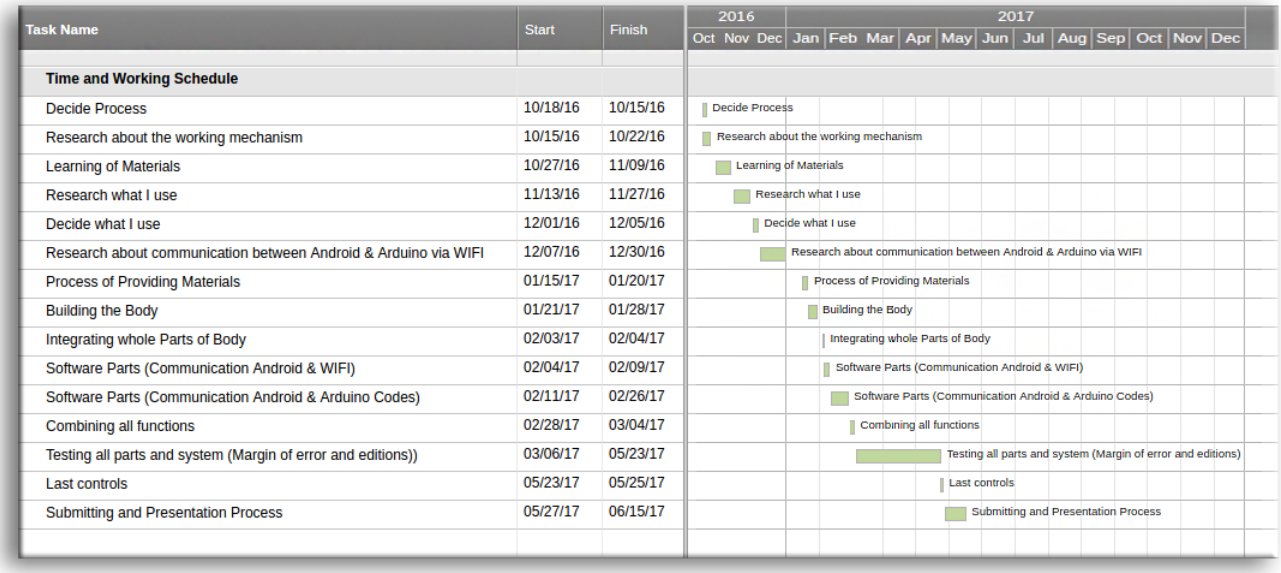


Figure 17: Time Table and Work Schedule

7 References

- http://electrotech.tv/arduinoya-giris-arduino-nedir-ne-degildir-1bolum/
- http://www.robotistan.com/arduino-smd-l298-cift-motor-suucu-shieldarduino-motor-shield#
- http://www.robotiksistem.com/dc_motor_ozellikleri.html
- http://android.serverbox.ch/?p=1039
- http://stackoverflow.com/questions/3205191/android-and-mjpeg
- http://forum.arduino.cc/
- https://sites.google.com/site/androidhowto/how-to-1/display-a-web-page
- http://www.bluecove.org/bluecove/apidocs/javax/bluetooth/UUID.html
- http://stackoverflow.com/questions/4032391/android-bluetooth-where-can-i-get-uuid