DRIVE-01

Joe

Ugenteraan

April 11, 2017

Abstract

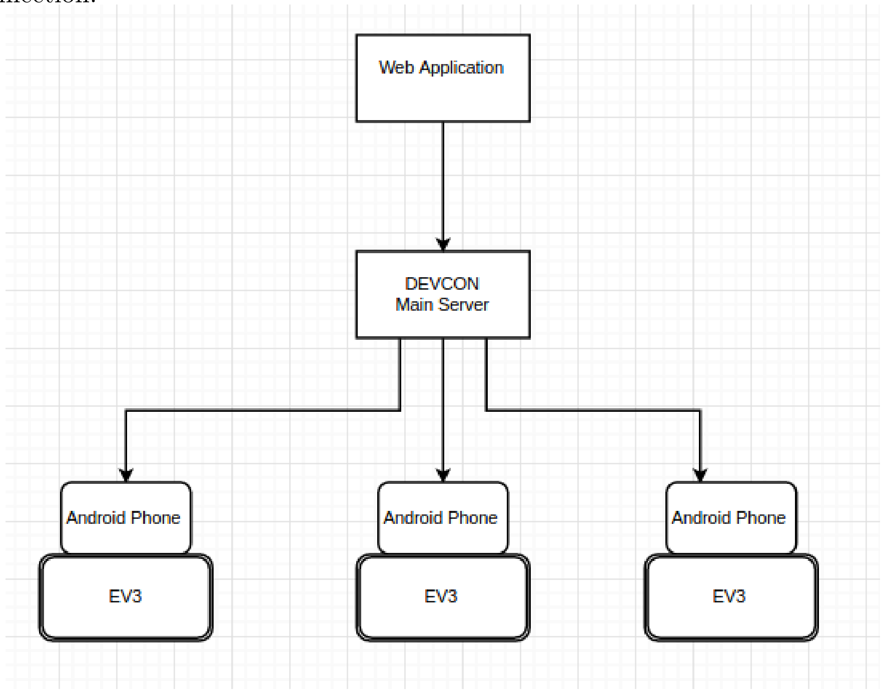
Since the EV3 does not possess high capability to perform calculations, a system will be developed whereby android phones will be the brain for the EV3s controlled by a main server. The system will be known as Democratized Robot Independent Vehicle Environment Version 1 (DRIVE - 01). This system can be developed easily by any programmer with basic knowledge of Java, JavaScript and concept of servers.

1 Introduction

Using a Lego Mindstorms EV3 to program and control it through Java (Programming Language) is entirely possible. However, the Lego Mindstorms EV3 does not have the capability to process a lot of information at once. This may not seem to be a big problem to many developers, but to those who intends to take their projects with EV3 to the next level might be concerned about this matter. To fix this problem, an android phone will be used as the brain to process the information and send instructions to EV3 on what has to be done.

2 Architecture

As can be seen in the diagram, a website will be created as a user interface to make the controlling of the EV3 possible. The user interface application will directly communicate with the main server. The main server will be connected to the android phones that are attached on the EV3 and finally each of the android phones will be communicating to their respective EV3 which completes the connection.



3 Technical Information

3.1 Connection between User Interface Application and Main Server

The user interface application in this example will be a web application that is written in HTML and AngularJS. The AngularJS will be responsible for the connection to the main server that is written in Node.js. The connection will be made through Node.js' Socket.io. This will ensure the speed of the data transfer as the data transfer through the socket connections are instantaneous.

3.2 Connection between Main Server, Android Phones and EV3s

The android application are written using Java (Android Studio). This application will be handling two types of socket connections. First is the Node.js' Socket.io that will be responsible for data transfer between the main server and the android phone. Second is the socket connection to the EV3 which will be acting as a server that is written in Java. Socket connections are used for the same purpose as before to ensure the data transfer speed.

4 Programming

4.1 Web Application

The web application is developed using simple HTML and AngularJS. The HTML part will be handling the presentation of the website while AngularJS handles the logical connection to the server.

4.1.1 HTML

```
<!DOCTYPE html>
<html ng-app="ev3">
<head>
<title>EV3</title>
</head>
<body>
<div ng-controller="movementController">
<form>
<button ng-click="movement('A')">Forward</button><input
        ng-model='time' type="number" name="time" />
<br />
<button ng-click="movement('B')">Backward</button>
<br />
<button ng-click="movement('C')">Left</button>
<br/>
<button ng-click="movement('D')">Right</button>
</form>
</div>
<script type="text/javascript" src='/socket.io/socket.io.js'></script>
<script type="text/javascript" src='app/connect.js'></script>
<script type="text/javascript" src="node_modules/angular/angular.js"></script>
<script type="text/javascript" src="app/app.js"></script>
</body>
</html>
```

4.1.2 AngularJS

```
(function(){
    angular.module('ev3', [])
```

```
        .controller('movementController', function($scope, $http){
            $scope.movement = function(letter){

                 var move = "";
                 var time = $scope.time;
                 if (time == null || time < 0){
                    time = 3.0;
                 }
                 switch(letter){
                     case "A" :
                         move = "forward";
                         window.sendinSocket(move, time);
                         break;
                     case "B" :
                         move = "backward";
                         window.sendinSocket(move, time);
                         break;
                     case "C" :
                         move = "left";
                         window.sendinSocket(move, time);
                         break;
                     case "D" :
                         move = "right";
                         window.sendinSocket(move, time);
                         break;
                }
              }
        });
}());
```

4.2 Main Server

```
var express  = require('express'),
app = express(),
http = require('http').Server(app),
client = require('socket.io').listen(http),
bodyParser = require('body-parser');


var port = 8000;

http.listen(port, function(){
    console.log("Listening at port : " + port);
});

app.use(bodyParser.json());
app.use('/node_modules', express.static(__dirname + "/node_modules"));
app.use('/app', express.static(__dirname + "/app"));


client.on('connection', function(socket){
    console.log('connected');
    socket.on("direction", function(data){
        sending(socket, data.test, data.time);
        console.log(data);
});

```

```
    socket.on('foo', function(data){
        console.log(data);
    });
});


app.get('/', function(req,res){
    res.sendFile('index.html', {root:__dirname});
});

var sending = function(socket, direction, time1){
    data = {
        test : direction,
        time: time1
    };

    client.emit('incoming', data);
}
```

4.3 Android Application

There will be two classes written in Java to separate the process of connecting to the sockets so that the codes will be more readable.

4.3.1 MainActivity Class

```
package com.example.root.myapplication;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;

import org.json.JSONObject;

import java.io.BufferedWriter;
import java.io.OutputStreamWriter;

import io.socket.client.Socket;
import io.socket.emitter.Emitter;

public class MainActivity extends AppCompatActivity {

    private java.net.Socket socketToEV3 = null;

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

    conn2ev3();
    conn2mainServer();
    }

    public void conn2ev3(){

        new Thread(){

            @Override
            public void run() {
                try{

                    socketToEV3 = new java.net.Socket("192.168.10.100", 3000);

                }catch(Exception e){
                    Log.d("Connection", e.toString());
                }
            }
        }.start();

    }



    public void conn2mainServer(){

        try{
            ConnectionToServer cn = new ConnectionToServer();
            final Socket socket = cn.getSocket();

            socket.on(Socket.EVENT_CONNECT, new Emitter.Listener(){

                @Override
                public void call(Object... args) {
                    socket.emit("foo", "Hello");
                }
            });


            socket.on("incoming", new Emitter.Listener(){

                @Override
                public void call(Object... args) {

                    Log.d("Connection", args[0].toString());
                    JSONObject jsonObj =(JSONObject) args[0];
                    String x = null;
                    int y = 0;
                    try{
                        x = (String) jsonObj.get("test");
                        y = (int) jsonObj.get("time");
                    }
                    catch (Exception e){
                        Log.d("Connection", e.toString());
                    }

                    if(x.equals("forward")){
                        sendData("F", y);
                    }
                    if(x.equals("backward")){
                        sendData("B", y);
                    }
                    if(x.equals("left")){
                        sendData("L", y);
                    }
                    if(x.equals("right")){
                        sendData("R", y);
                    }

                }

            }).on(Socket.EVENT_DISCONNECT, new Emitter.Listener(){
                @Override
                public void call(Object... args) {
                    Log.d("Connection", args[0].toString());
                }
            });


            socket.connect();
        }
        catch(Exception e){
            Log.d("Connection", e.toString());
        }

    }


    public void sendData(String x, int y){
        BufferedWriter bw = null;
        String s = x;
        int time = y * 1000;

        String timeString = Integer.toString(time);
        String sentence = s.concat(timeString);

        try{
            bw = new BufferedWriter(new OutputStreamWriter
            (socketToEV3.getOutputStream()));
            bw.write(sentence);
            bw.newLine();
            bw.flush();
        }
        catch(Exception e){
            Log.d("Connection", e.toString());
        }
     }
}
```

4.3.2 ConnectionToServer

```
package com.example.root.myapplication;

import android.util.Log;

import io.socket.client.IO;
import io.socket.client.Socket;

/**
 * Created by root on 4/11/17.
 */

public class ConnectionEV3 extends MainActivity {
    Socket socket;
    public Socket getSocket(){
        if(socket==null){
            try{
                socket = IO.socket("http://192.168.10.102:8000");
            }catch(Exception e){
                Log.d("Connection", e.toString());
            }
        }

        return socket;
    }

}
```

4.4 EV3 Server

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.ServerSocket;
import java.net.Socket;

import lejos.hardware.lcd.LCD;
import lejos.hardware.motor.EV3LargeRegulatedMotor;
import lejos.hardware.port.MotorPort;
import lejos.robotics.RegulatedMotor;
import lejos.utility.Delay;

public class Test {

     public static void main(String[] args) {
        try{
            ServerSocket server = new ServerSocket(3000);
            System.out.println("Started");
            Socket s = server.accept();
            System.out.println("Connected");

            RegulatedMotor m = new EV3LargeRegulatedMotor(MotorPort.A);
            RegulatedMotor m2 = new EV3LargeRegulatedMotor(MotorPort.C);

            BufferedReader br = new BufferedReader(new InputStreamReader
            (s.getInputStream()));
            String data = null;
            int time = 0;
            String data1 = null;

            while ((data = br.readLine()) != null){
                data1 = data.substring(0, 1);
                time = Integer.parseInt(data.substring(1));

                if(data1.equals("F")){
                    m.forward();
                    m2.forward();
                    Delay.msDelay(time);
                    m2.stop();
                    m.stop();
                }
```

```
                 else if(data1.equals("B")){

                     m.backward();
                     m2.backward();
                     Delay.msDelay(time);
                     m2.stop();
                     m.stop();
                 }
                 else if(data1.equals("R")){

                     m.forward();
                     m2.backward();
                     Delay.msDelay(time);
                     m.stop();
                     m2.stop();
                }
                     else if(data1.equals("L")){
                     m.backward();
                     m2.forward();
                     Delay.msDelay(time);
                     m2.stop();
                     m.stop();
                }

System.out.println(data);
               }
               System.out.println("Ended");
               Delay.msDelay(4000);

           }catch(Exception e){

         }
     }
}
```

5 Conclusion

In conclusion, the brain of the EV3 is successfully replaced with the android phone instead of the EV3 processor. With this architecture, further enhancement can be implemented in the system. Perhaps, a server that runs a machine learning algorithm while controlling the EV3s autonomously can be developed in the future.