

MOTOR CONTROL SYSTEM

JAVA documentation

Contents

[0](#_Toc532112811)

[Contents 1](#_Toc532112812)

[INTRODUCTION 2](#_Toc532112813)

[PROGRAM INTERFACE 3](#_Toc532112821)

[Activating the program in ideal method 3](#_Toc532112822)

[Main scene with initial configuration 3](#_Toc532112823)

[Main scene 4](#_Toc532112824)

[Start scene 4](#_Toc532112825)

[Setting scene 5](#_Toc532112826)

[Deactivating the program in ideal method 5](#_Toc532112827)

[Error in activating and deactivating the program 6](#_Toc532112828)

[Program structure 7](#_Toc532112829)

[ConfigMotorInit class 7](#_Toc532112830)

[DataHandler class 8](#_Toc532112831)

[Maincontroller class 8](#_Toc532112832)

[Motorsystem class 9](#_Toc532112833)

[COMMUNICATION 9](#_Toc532112834)

[**Communication class:** 10](#_Toc532112835)

[**Serial Writer Class:** 10](#_Toc532112836)

[Hardware design 10](#_Toc532112837)

[References 11](#_Toc532112838)

1

# INTRODUCTION

The aim of the project is to build a graphic user interface with motor so we can control the direction, speed and the state of the motor. our system is a motor control using java application, it’s composed of 3 modules ,GUI module , communication module and hardware GUI for making user configure direction and speed ,while the motor starts working user can change configuration as we communicate through serial port using a communication module to send these changes to hardware.

Check if the motor connected

No

Run error and EXIT

# 

Yes

# 

Check if the motor config exists

# 

No

Create Motor config

Yes

# 

Start the GUI

The motor will run by the default speed 10 and clockwise

EXIT

It will hold the moto

It will load from the file and run

The speed will varies from 0->10

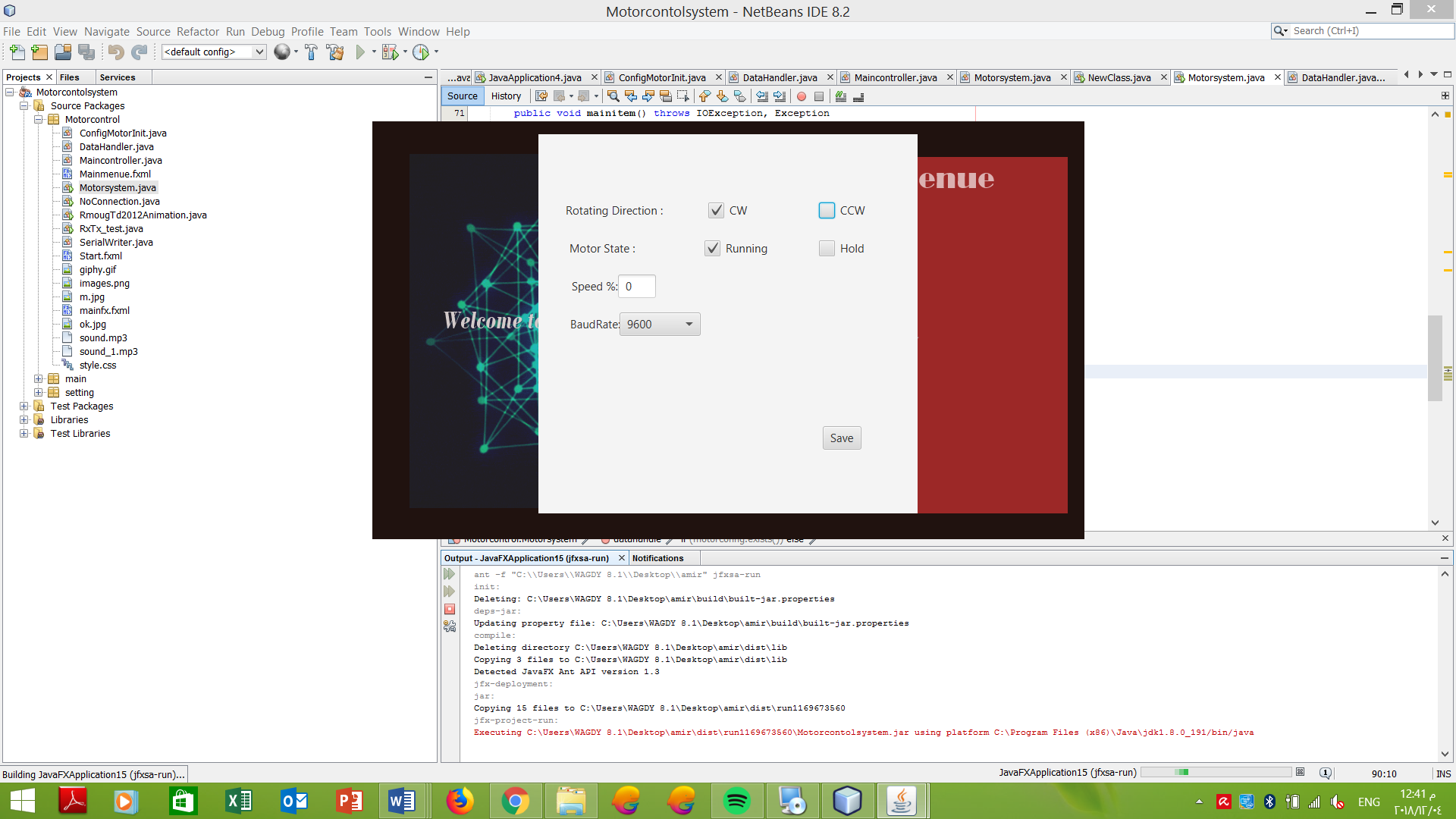
# PROGRAM INTERFACE

This section explains how the user communicates with the program. It must be stated how the user will start running the program, i.e. how the programming environment will be set and activated, what is the command to be entered to execute the program, and what are the parameters (if any) that must accompany the execution command. It must also be stated how the user terminates the program. This section is just for activating and deactivating the programming environment; the details of the communication steps will be explained in the following sections.

## Activating the program in ideal method

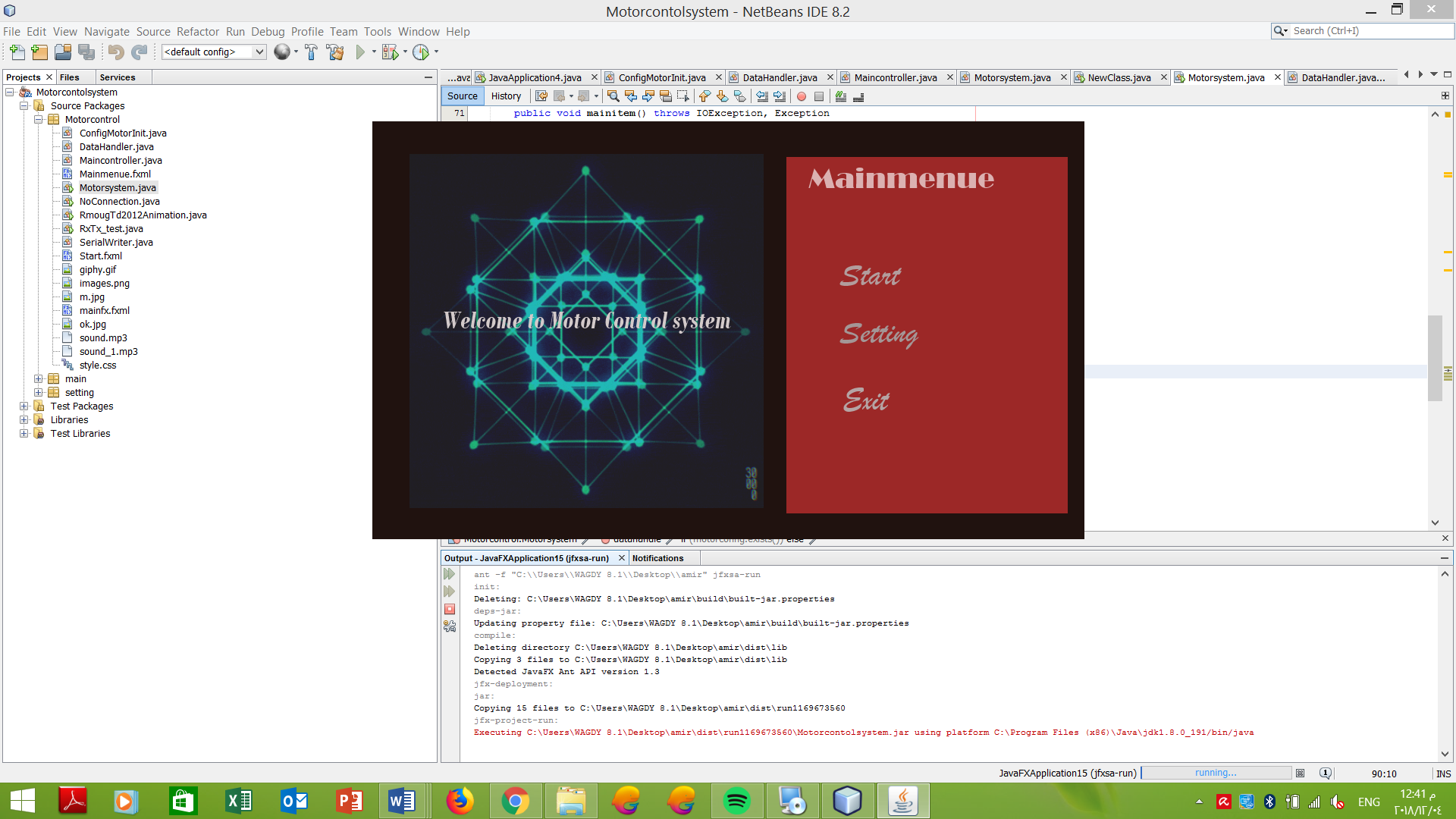
### Main scene with initial configuration

First you should connect the motor with the port in the computer(laptop) so you can run the program in the first run only you should configurate the intial state if you want a spacific configuration and load it in the main scene.



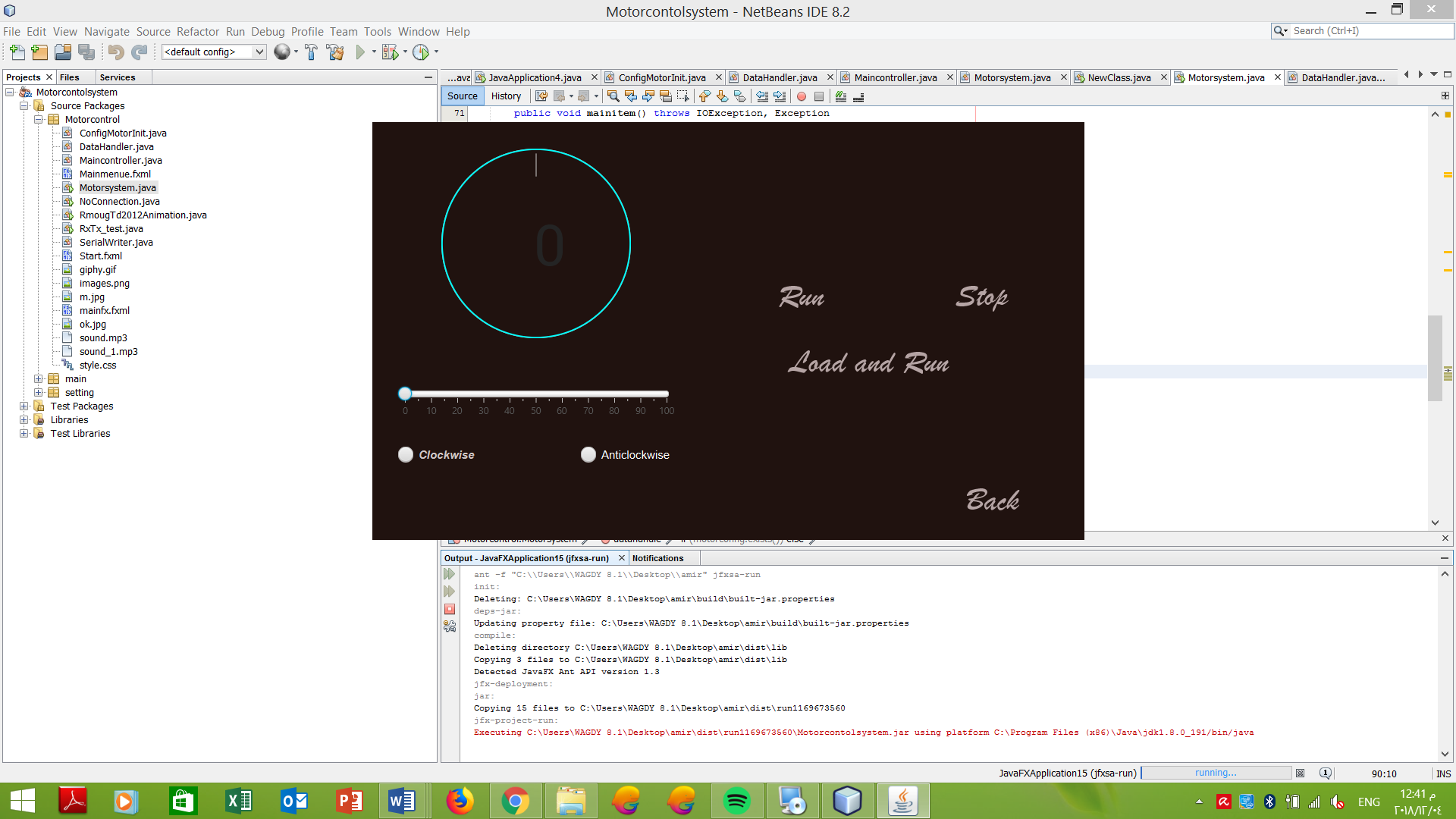
### Main scene

After you choose your initial configuration you are now at the mainmenue scene you have three options (Start Setting Exit)



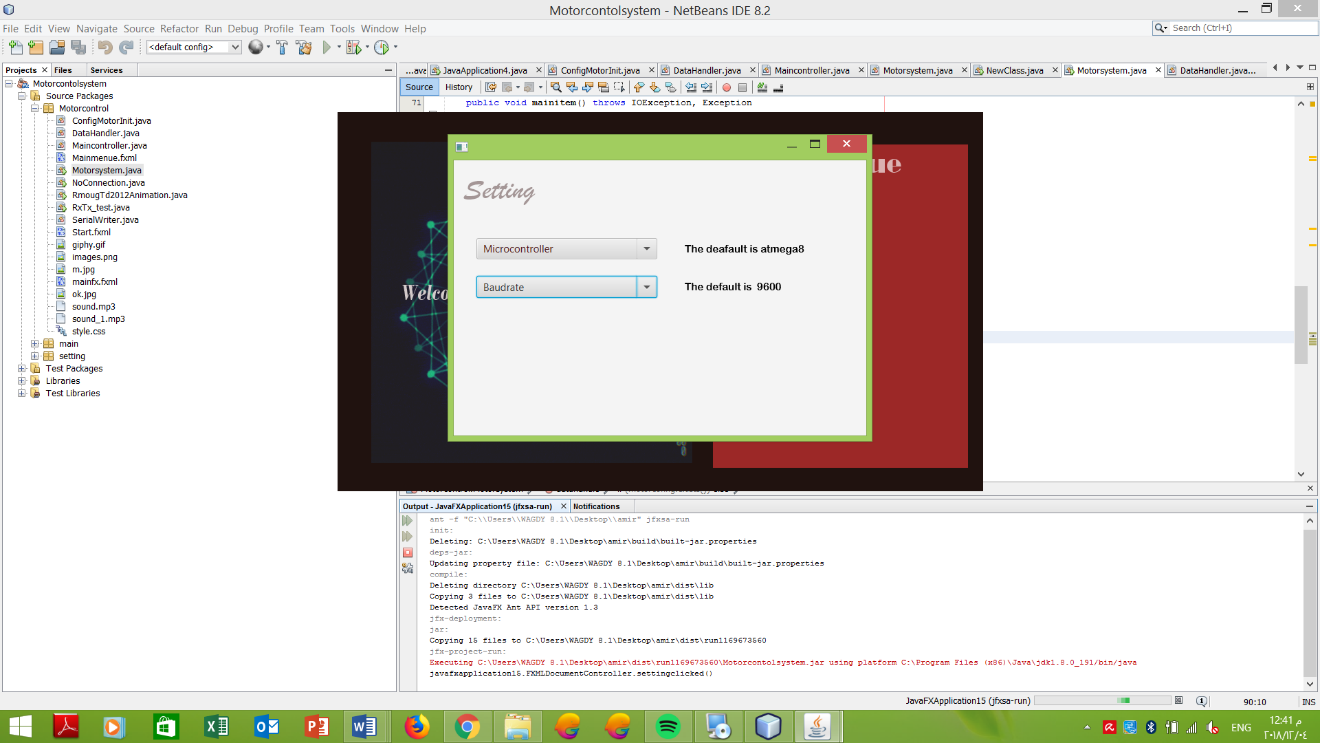
### Start scene

By clicking on the start scene, you are now in the main control system



You can now choose any preferences you need (the speed form 0-100, the direction (clockwise anticlockwise) and control your motor if you want to load the initial config you made or the last saved modifications you can click load and run

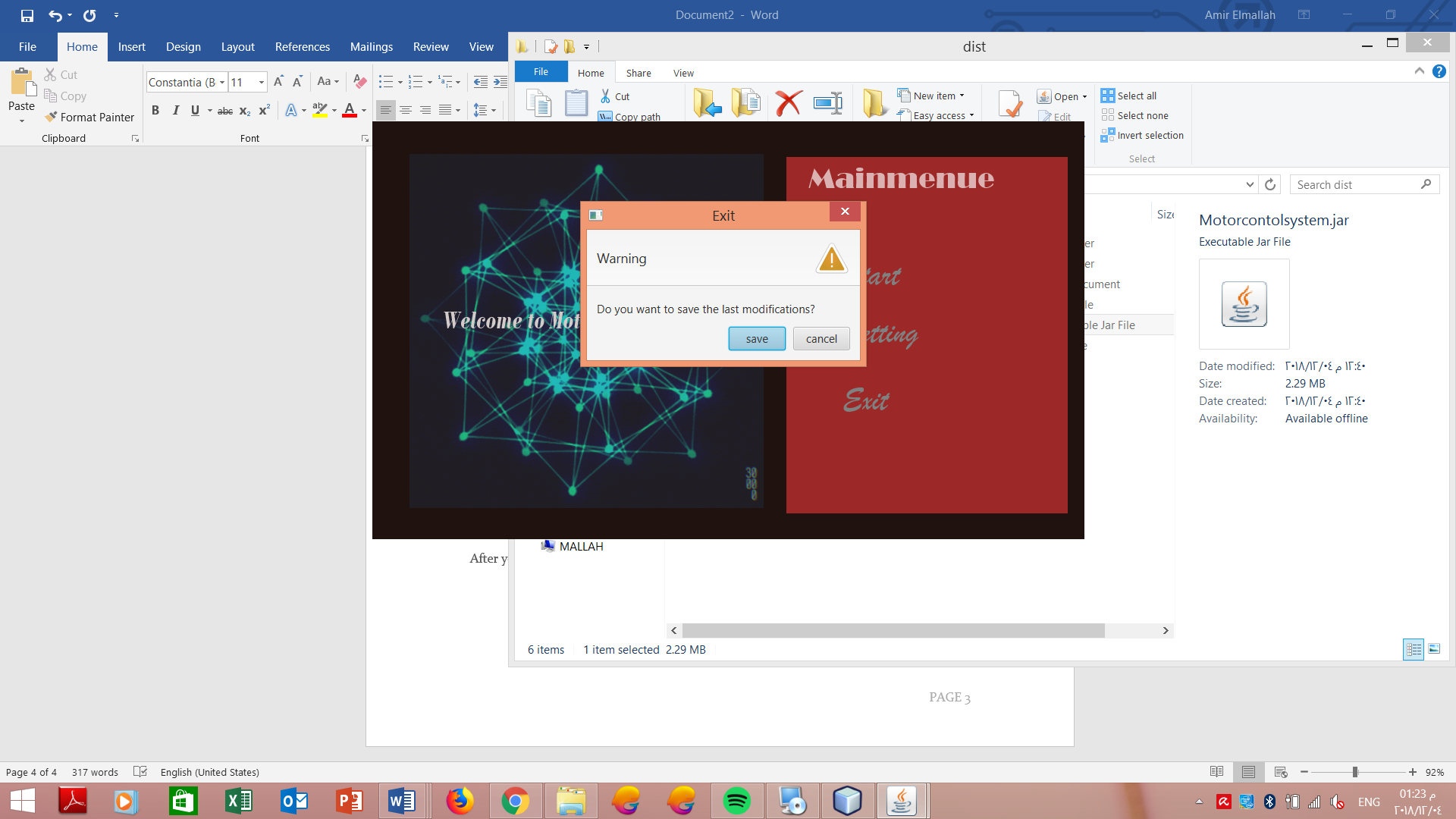
### Setting scene



We try to make our program more generic so it can be portable to any microcontroller and with baud rate you can skip it as we working on the default settings.

## Deactivating the program in ideal method

After you press on exit tab in the main menu

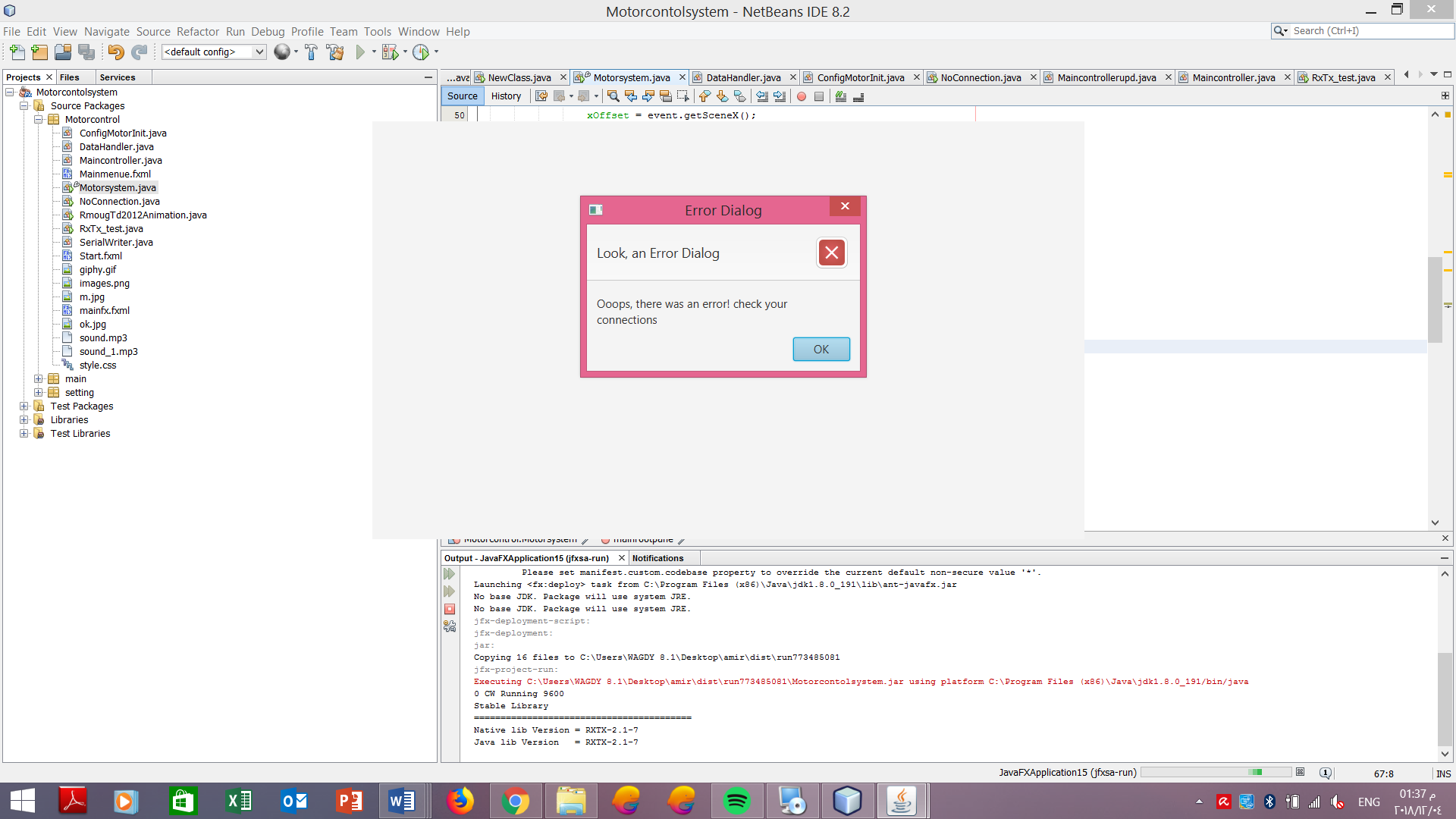
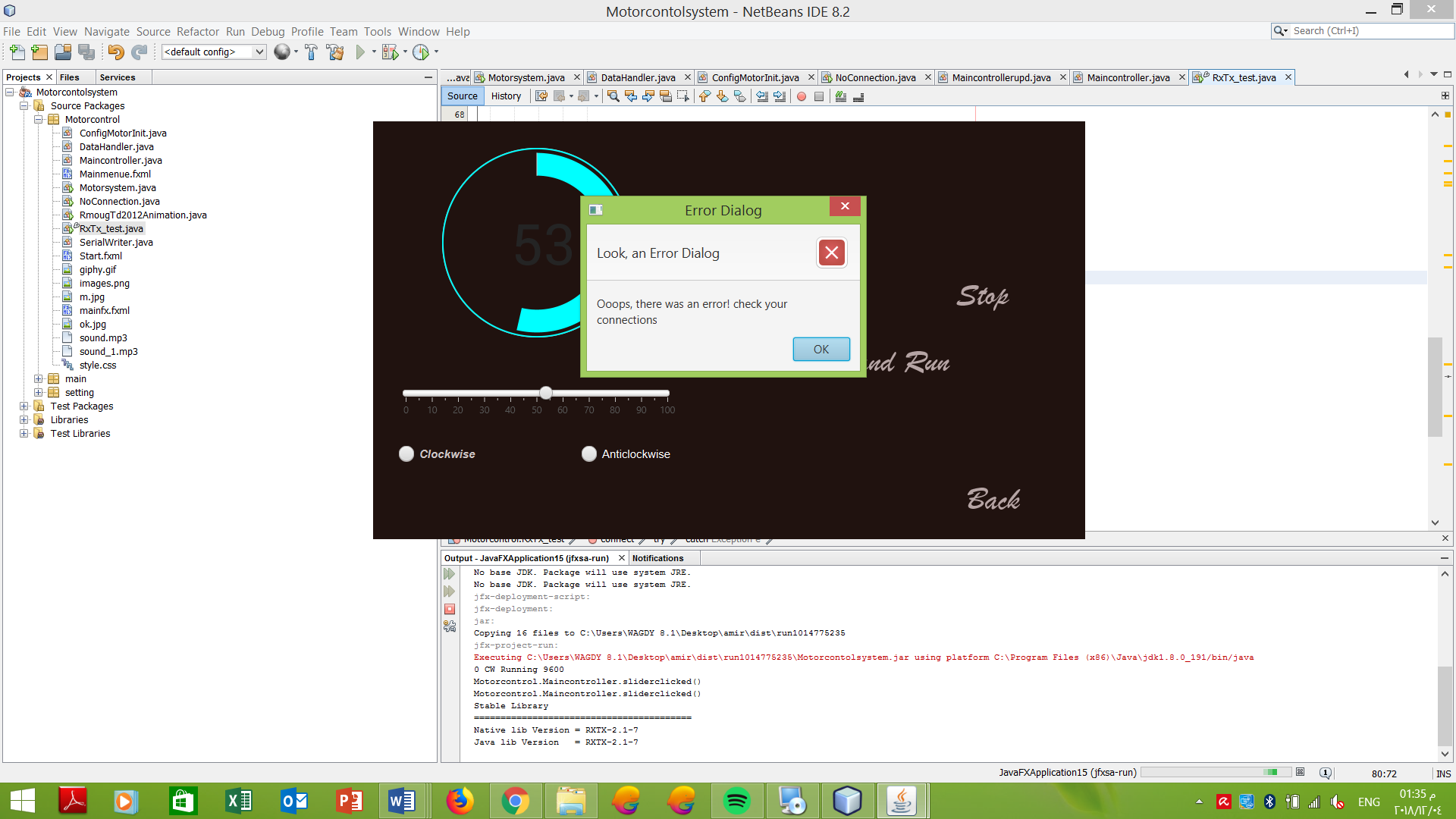


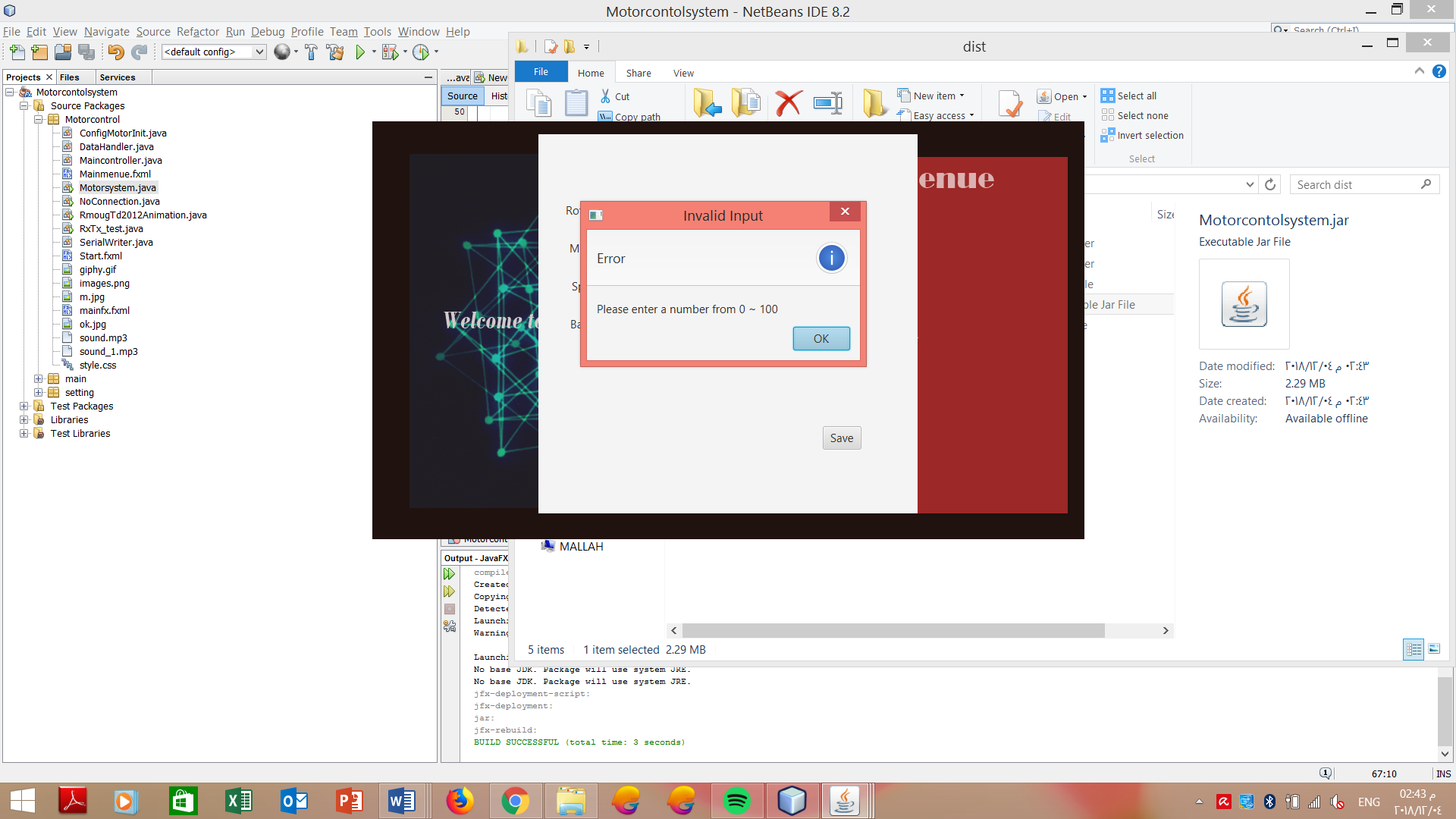
By clicking save button now you can save your last modification and in the start scene you can load and run your motor safely

## Error in activating and deactivating the program

If you try to run the program and the motor is not connected the program will give you an error message and it will close automatically OR you lost your connection with the motor while the program is running

In the first time of running the program and you put in the initial configuration speed that exceeds 100 it will give you an error message to pot an proper speed from 0-100





# Program structure

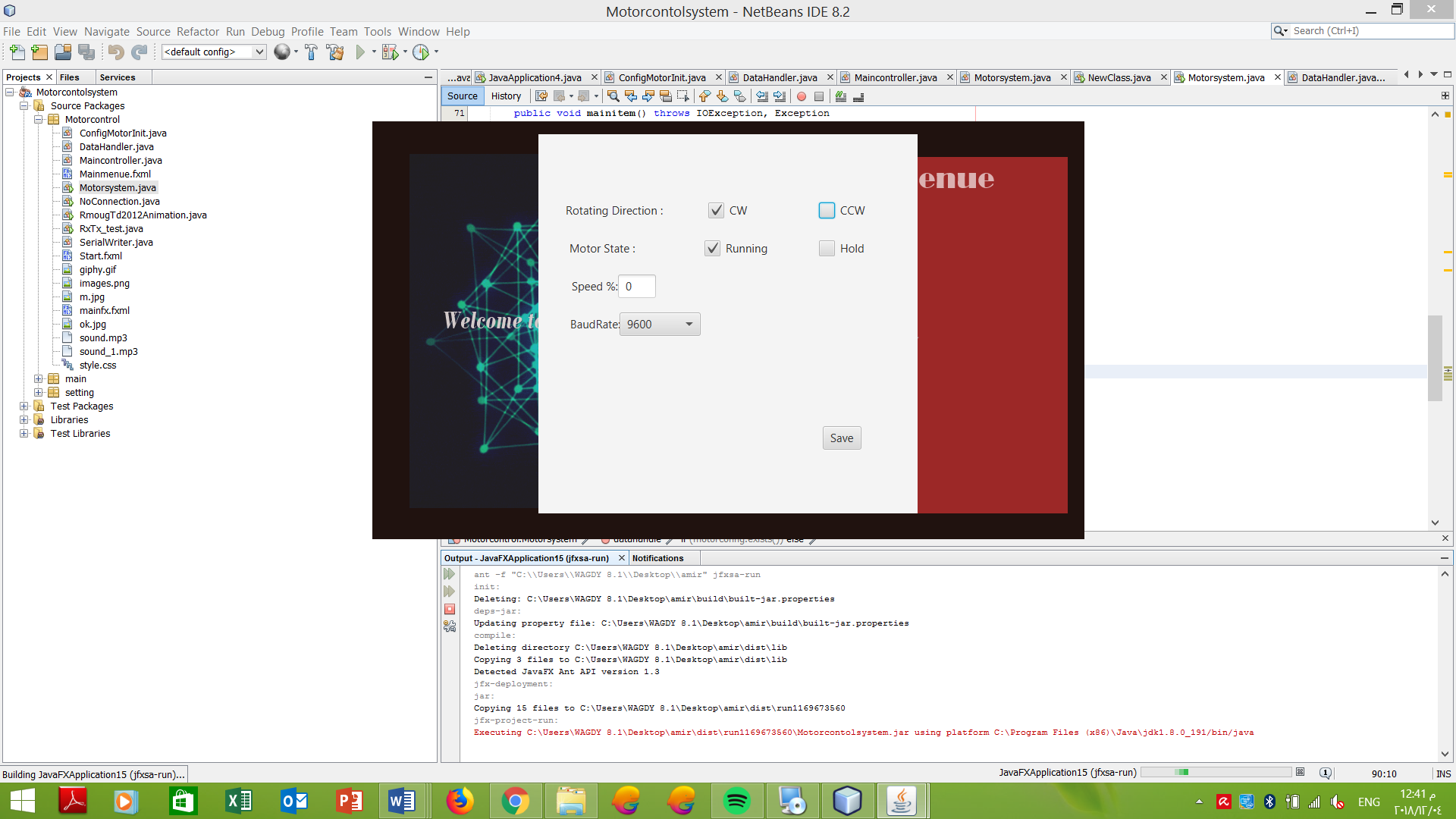
This is the “technical manual” of the program each part of the program (subprograms, modules, classes, etc.) will be explained in sufficient detail such that the reader can understand the statements, algorithmic logic, etc. The data structures (data types, variables, domains, etc.) . this section is reserved for detailing the program code.In our program we have 5 classes each class has its spacific functionality .these 5 classes are in the same package that called motorcontrolsystem.

## ConfigMotorInit class

İts a graphical user interface extends apllication made by javafx for the configuration file which its appear in the initial state only in the first time of running the program

The attributes of this class is the interface with the user the check button (clockwise anticlockwise, start, stop), text field (speed), save button and the stack pane to add them on it

The methods are the start method which contain anonymous class for the handlers



## DataHandler class

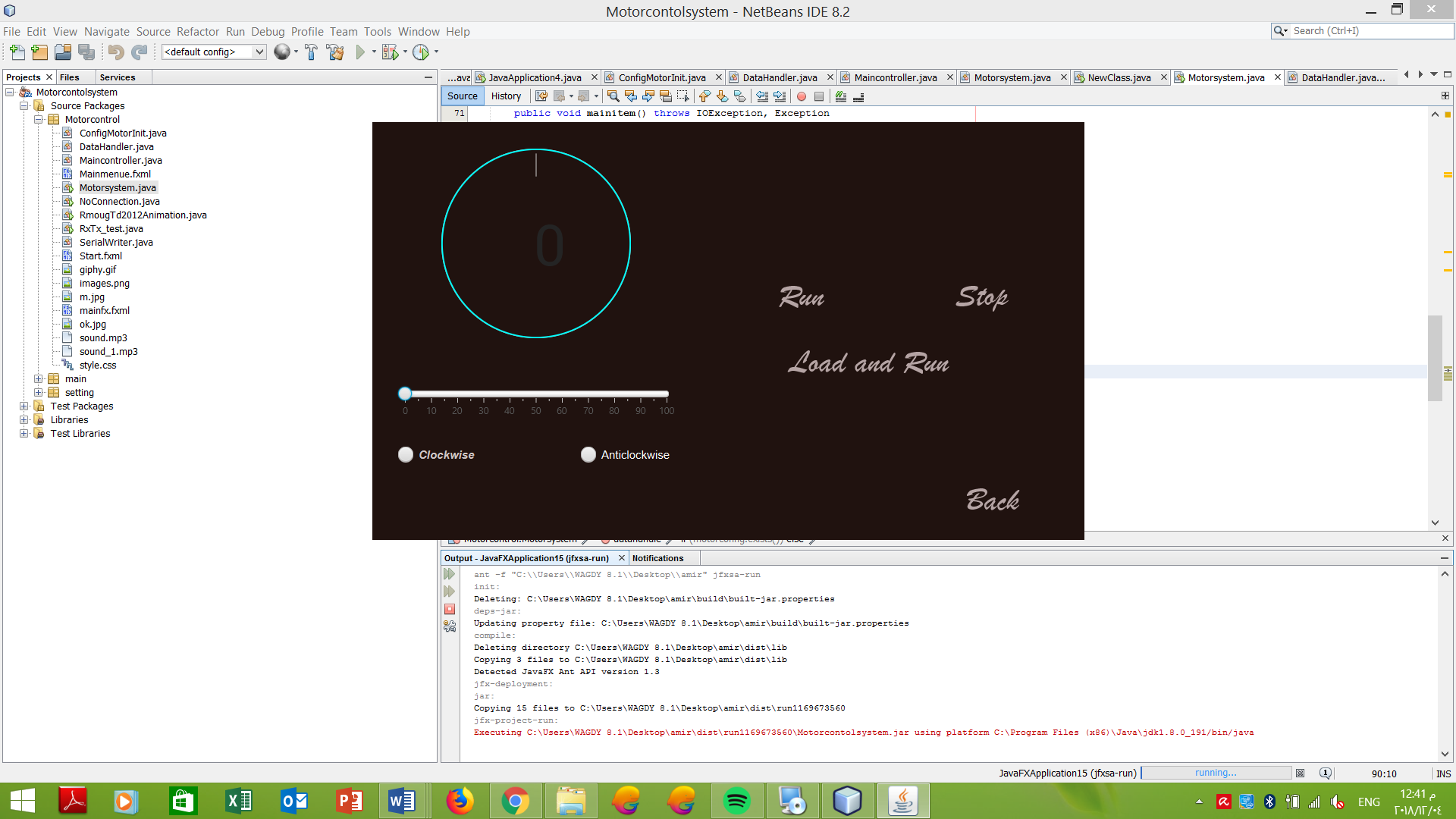
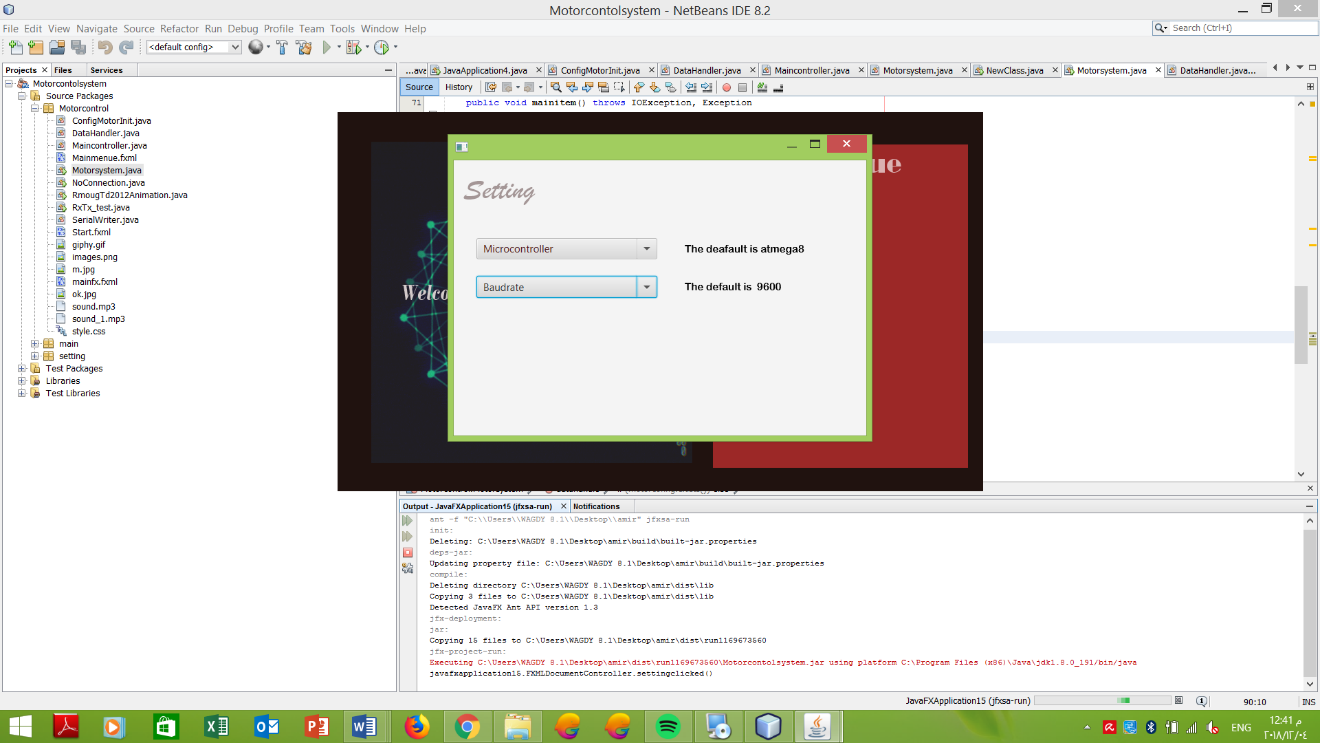
It’s a class that has the state of the motor, the direction, the speed and object from File class so we can make the configuration file and save our data.

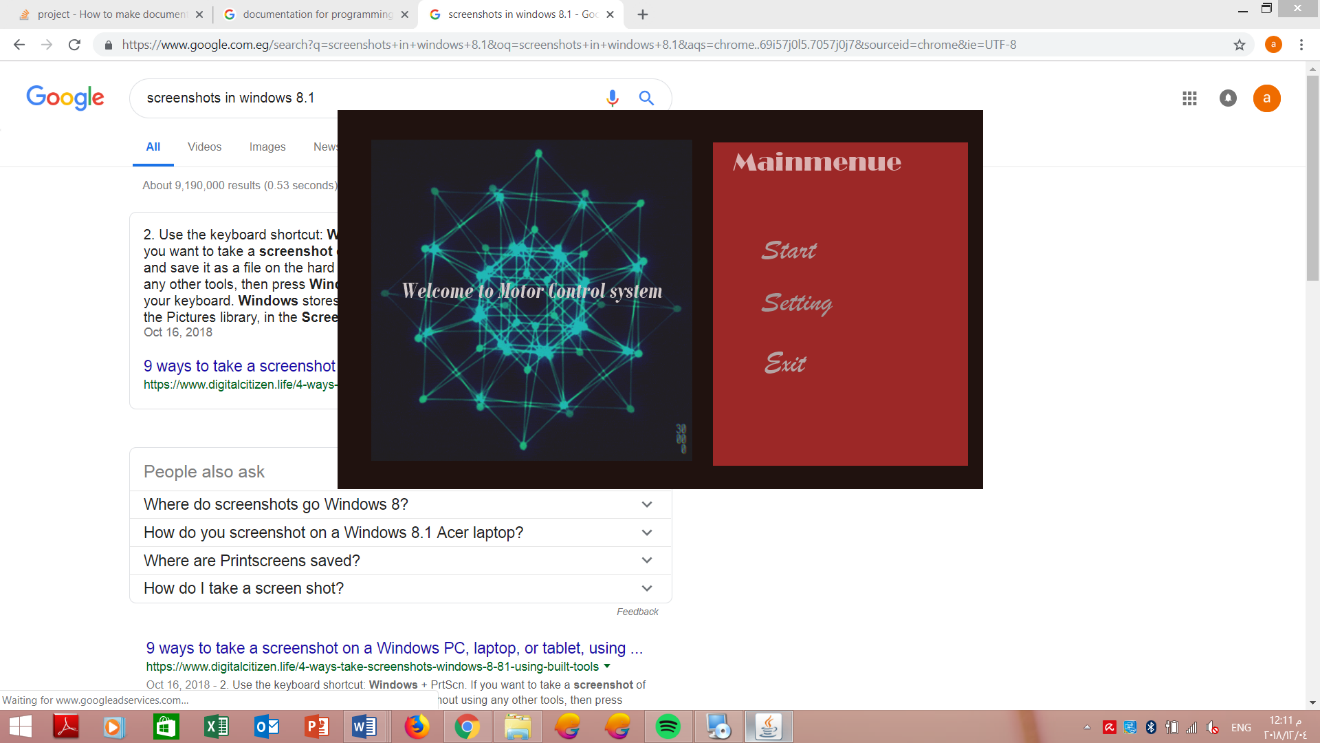
The constructor of this class read from the file and set the attributes with the new values in the configuration file

The methods in this to update on the file, to read from the file and public setters and getters so we can use them by making object from this class and set our speed and direction

## Maincontroller class

The maincontroller for the 3 fxml scenes (Main menu ,Start scene ,Setting scene)) its attributes is the 3 anchorpanes for the three scenes and all the controllers (slider gauge,button,label,checkbox,splitmenu button) that appear to the user while running the program

İts the main class for our program which it creat in its method object from Rxtx\_test class to send the data on action of clicking by the user and call the static attributes in Datahandler to save them befor exit the program and the static attributes in the Motorsystem class (which is the data object from data handler to use setters and getters that is mentioned before).



## Motorsystem class

İts the Main class the class that contains the main and all the program start from this class

İn this class its responsible for handling the scenes which will be the first scene and the show of the stages of file configurations FX file

The attributes of this class the manroot which the 2 FXML scenes load on it and it make a static attribute object from Data handler class to set them in the start of the program or to create the file if it’s the first time to run the program

## COMMUNICATION

We used RxTxcomm library that runs on a 32-bit JDK.

The module is composed of 2 classes:

**Communication class:**

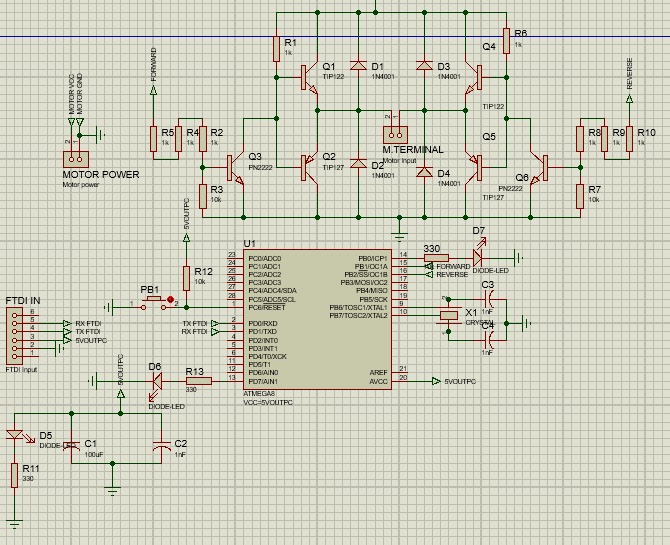
* The class constructor takes the serial port name, the baud rate, and the data to be as parameters then calls the “connect” function.
* The “connect” function checks if the serial port is not in use. After that it configures the serial port parameters (baud rate, stop bits number, data bits number and parity check bits), creates an Output Stream object and passes it to a Serial writer object constructor.

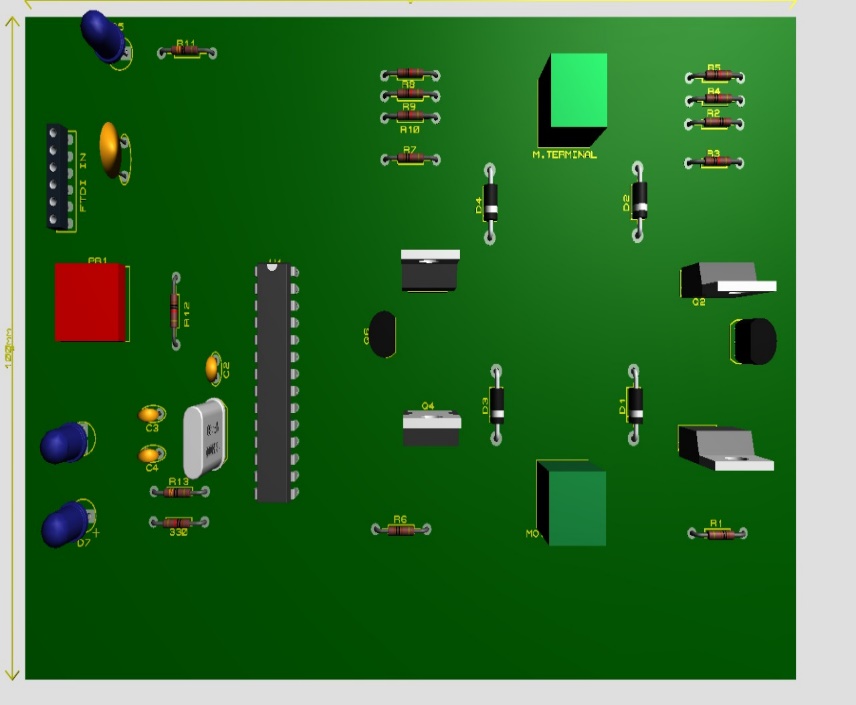
The class also contains “end Connection” function that closes the output stream object and the serial port after the session has been finished.

**Serial Writer Class:**

* The class implements a runnable thread which is responsible for writing on the serial port output stream .It‘s constructor takes an outputStream object and the data to be sent.

# Hardware design

* components:
* USB to TTL
* DC motor
* Resistance
* H bridge
* Power source
* Microcontroller



**Operation**:

-UART communication protocol is used to receive instructions from the user interface unit with rate of (9600).

- User chooses the direction of motor (clockwise or anti clockwise)

-user control speed of motor from the GUI interface and the microcontroller changes it using PWM (pulse width modulation).

# References

UART:<https://whatis.techtarget.com/definition/UART-Universal-Asynchronous-Receiver-Transmitter>

PWM :<https://learn.sparkfun.com/tutorials/pulse-width-modulation/all>

Mdusa library 8.0 jar

Rxtxcomm lbrary jar

Scene builder

Clickteam installer

Jdk 32bit