Course Name: Applied programming using JAVA.

Department: Computer and Systems.

Level: 2nd year electric.

Zagazig University

Faculty of Engineering

Name: Ahmed Mahmoud Mohamed El-Sayed Mohamed

Project Name: Interface with Arduino

Level: 2nd year electric

Year: (2019-2020)

Table of contents

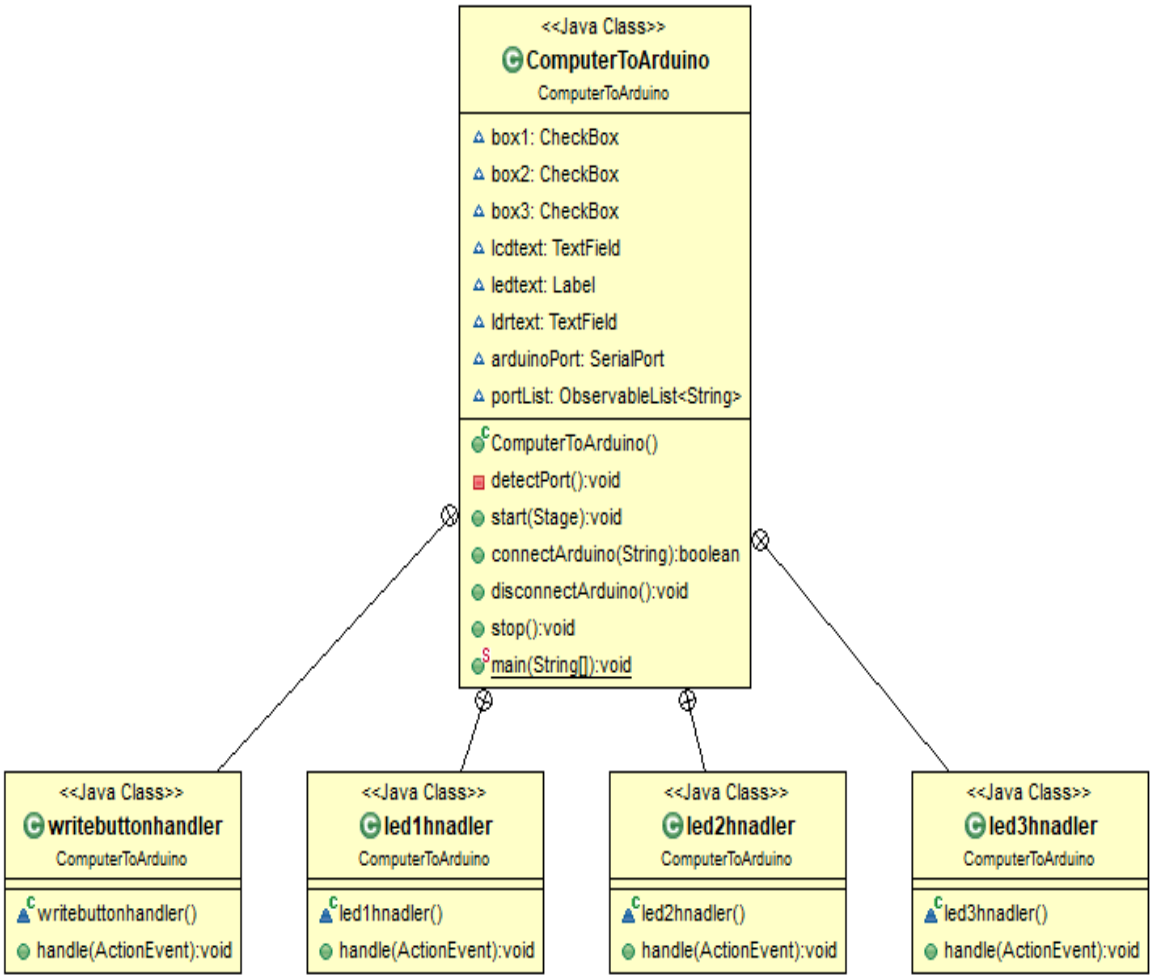
Introduction

Our main objective in this article to talk about JAVA programming language focusing on using OOP and JavaFX library to create a graphical user interface (GUI) which manages us to communicate with Arduino which in its turn react with the physical world. So, we can control some LEDs and measure some physical quantities like temperature, humidity, light intensity, and many other quantities through sensors. Hopefully by the end of this article we will be able to write a clean code with JavaFX application to design a GUI and run it on NetBeans and write an Arduino code which communicate with java code through a library we will insert in the java code and will do some circuit connection with the Arduino Board to test our program.

Components and Requirements

NetBeans IDE	To write, debug and run JAVAFX code.
Arduino IDE	To write and Arduino code.
Fritzing program	To make the schematic of the circuit
Intellij IDE	To design and generate class diagram (UML)
Arduino uno board	To run, execute and process Arduino code.
breadboard	To construct our project on it.
LEDS	To detect that all stuff and codes work by sending light.
LDR sensor	Reading light intensity and send it to serial monitor.
Resistors	To control Voltage and Ampere flow.
Jumper wires	For connection of the circuit.

UML diagram:


<<Java Class>>
ComputerToArduino
ComputerToArduino
box1: CheckBox
box2: CheckBox
box3: CheckBox
lcdtext: TextField
ledtext: Label
ldrtext: TextField
arduinoPort: SerialPort
portList: ObservableList<String>
ComputerToArduino()
detectPort():void
start(Stage):void
connectArduino(String):boolean
disconnectArduino():void
stop():void
main(String[]):void
<<Java Class>>
writebuttonhandler
ComputerToArduino
writebuttonhandler()
handle(ActionEvent):void
<<Java Class>>
led1hnadler
ComputerToArduino
led1hnadler()
handle(ActionEvent):void
<<Java Class>>
led2hnadler
ComputerToArduino
led2hnadler()
handle(ActionEvent):void
<<Java Class>>
led3hnadler
ComputerToArduino
led3hnadler()
handle(ActionEvent):void

JAVAFX code

First of all, we will write our JavaFX code using NetBeans IDE. We will open a NetBeans IDE and start a new project of category JavaFX Application. IDE automatically will insert a built-in example of a simple GUI that say, "Hello world!". I will start build and upgrade my own code as shown in the attached pictures.

```
package ComputerToArduino;

//importing the program libraries

import java.util.logging.Level;
import java.util.logging.Logger;
import javafx.animation.FadeTransition;  // A class that
import javafx.animation.KeyFrame;        // A class to define specifis
values at a certained point in time
import javafx.animation.Timeline;        // A class that is used to
define an animation for any written value
import javafx.application.Application;   // A class from which JavaFX
applications extend.
import javafx.application.Platform;      // a class that is responsible
for starting and ending our program
import javafx.beans.value.ChangeListener;// A class that is updated
when listener is changed
import javafx.beans.value.ObservableValue;// A class that is notified
whene the value of an ObservableValue changes.
import javafx.collections.FXCollections; // Utility class that consists
of static methods
import javafx.collections.ObservableList;// A class that Creates and
returns a typesafe wrapper on top of provided observable list.
import javafx.event.Event;               // A class which tell when an
event takes place
import javafx.event.EventHandler;        // A class which we use to can
handle our stuff
import javafx.geometry.Insets;           // A class which give the
capability to set spaces between our control objects
import javafx.geometry.Pos;              // A class that manage us to
edit the position of controls on the the program
import javafx.scene.Scene;               // the class which we can use
as a container for our all controls
import javafx.scene.control.*;           // class for all user
interface controls like buttons,labels and etc..
import javafx.scene.layout.HBox;         // A class in which we can add
our program controls like buttons,texts and..etc in a horizontal format
import javafx.scene.layout.VBox;         // A class in which we can add
our program controls like buttons,texts and..etc in a vertical format
import javafx.scene.paint.Color;         // A class that is used to add
colors to text,label..etc
import javafx.scene.text.Font;           // A class that manage us to
edit fonts
import javafx.scene.text.Text;           // A class which contain text
to our program
```

```
import javafx.stage.Stage;                // A class which contain our
program controls
import javafx.util.Duration;              // a class which add a
duration like delay
import jssc.*;                            // A library that provide all
information for serial communication
import static jssc.SerialPort.MASK_RXCHAR;// A class which is used to
set connection parameters

   public class ComputerToArduino extends Application
```

Comment:

I first imported all libraries and classes that I will need in my code and these libraries are:

- **JavaFx library**: we use this library to create windows and design GUI which contain three main classes (Stage, Scene, Control) so I imported this library first to can use it later and the library itself has a lot of other useful classes like:
    a) Stage: this is the main window of the GUI.
    b) Scene: the container that contains all components and controls.
    c) Control: and this is the class that has all components like Buttons, CheckBoxes, ComboBoxes, labels and many other components.
    d) Application class: creates the environment of windows.
- **Jssc library**: it stands for Java simple serial connection. This library makes it possible and easy to communicate with Arduino through its serial port. I download this library then added it to my project.

Then I start coding after importing all needed libraries and classes. In Application class I declared some objects that I will deal with in the program and these objects are checkboxes, TextFields, labels and serialPort as shown in the above piece of code.

After that using jssc library I made a class to detect all available ports that I can communicate with storing the results found in array of strings named serialPortsNames.

```
    //Declaring the gui components
    CheckBox box1;          // whe have four check boxes
    CheckBox box2;
    CheckBox box3;
    CheckBox box4;
    TextField lcdtext;
    Label ledtext;
    Label Commsg;
    TextField ldrtext;
    SerialPort arduinoPort = null;
    ObservableList<String> listPort;
    Timeline time;

    //creating searching for port to connect function
    private void searchingForPort(){
        listPort = FXCollections.observableArrayList();
        String[] foundedPorts; //storing ports name in a string array
        foundedPorts = SerialPortList.getPortNames();
        for(String name: foundedPorts){ //starting searching for the
port
            listPort.add(name);           //Adding searching result to
portList

        }
    }

    @Override
    public void start(Stage firstStage) throws Exception {

        searchingForPort(); // start searching for the port
        final ComboBox comboBoxPorts = new ComboBox(listPort);
//adding result for searching to the comboBox list
        comboBoxPorts.valueProperty().addListener(new
ChangeListener<String>() {

            @Override
            public void changed(ObservableValue<? extends String>
observable,  // a function to notify when the value of the port is
changed
                    String oldValue, String newValue) {
                connectToArduino(newValue);
// sending the new port value to the list
                Commsg.setText("Successfully connected\nto "+
newValue);       // tell the user that the connection doe successfully
            }
        });

        //program  gui components
        box1 = new CheckBox("LED1");                    // setting
names of our four check boxes
        box2 = new CheckBox("LED2 (Alert)");
        box3 = new CheckBox("LED3");
        box4 = new CheckBox("ALL LEDs");

        Text serialtext = new Text("Port conection");    // the serial
connection label and its style
```

Comment:

After declaring the detectPort class I override start method which will be automatically called when the program run. after that I called the detectPort function and it will store its searching results in the comboBox "comboBoxPorts". Then creating all components of my GUI and stacked them on layouts and then scene.

```
serialtext.setFill(Color.BLUE);
        serialtext.setFont(new Font("Arial",20));

        Text ldrlabel = new Text("LDR Reading");          // the LDR
Reading label and its style
        ldrlabel.setFill(Color.BLUE);
        ldrlabel.setFont(new Font("Arial",20));

        Text ledlabel = new Text("LEDs Control");
        ledlabel.setFill(Color.BLUE);
        ledlabel.setFont(new Font("Arial",20));

        Text lcdlabel = new Text("LCD Display ");
        lcdlabel.setFill(Color.BLUE);
        lcdlabel.setFont(new Font("Arial",20));

        Text msglabel = new Text("LEDs status");
        msglabel.setFill(Color.BLUE);
        msglabel.setFont(new Font("Arial",16));

        Text serialmsg = new Text("Connection Status");
        serialmsg.setFill(Color.BLUE);
        serialmsg.setFont(new Font("Arial",16));

        Text weltext = new Text("Welcome to L_Niwehy project");
        weltext.setFill(Color.RED);
        weltext.setFont(new Font("Arial",15));

        lcdtext = new TextField();                  // creating gui
components
        ledtext = new Label();
        Commsg = new Label();
        ldrtext = new TextField();
        ldrtext.setPrefColumnCount(5);
        ldrtext.setEditable(false);

        Button writebutton = new Button("Send");
        Button exitbutton = new Button("Exit");


       exitbutton.setOnAction(e -> {                    // Exit button
handler to exit from the program
           if(arduinoPort != null)
            try {
```

```
                arduinoPort.writeString("90");
            } catch (SerialPortException ex) {

Logger.getLogger(ComputerToArduino.class.getName()).log(Level.SEVERE,
null, ex);
            }
           Platform.exit();                              // Exit from the
program
        });
```

Comment: as shown in the above piece of code after creating all components and stacking it on layouts I stacked these layouts to the scene and the scene itself to the stage and show it. Then I did the connectArduino function which connect to the chosen port with the entered parameters and start listening to the orders.

```
         EventHandler h6 = new EventHandler() {        // handling the
fade of text on gui
            @Override
            public void handle(Event event) {
                FadeTransition fade = new
FadeTransition(Duration.millis(2500), weltext);  // set delay for the
display
                fade.setFromValue(1);
// start from displayed value
                fade.setToValue(0);
// end with fade
                fade.setCycleCount(Timeline.INDEFINITE);
                fade.setAutoReverse(true);
// smoothly display and disapper
                fade.play();
            }
        };

        time = new Timeline();                                        //
setting timeline for message display
        time.getKeyFrames().add(new
KeyFrame(Duration.millis(2500),h6));  // adding delay to the text
        time.play();

        //Setting actions for check boxes and writebutton

       box1.setOnAction(led1_handler);                               //
Setting actions for check boxes
       box2.setOnAction(led2_handler);
       box3.setOnAction(led3_handler);
       box4.setOnAction(all_handler);
       writebutton.setOnAction(writebuttonhandler);
```

```
        //Stacking the components on Layouts
        VBox layout1=new VBox(20);

layout1.getChildren().addAll(serialtext,comboBoxPorts,serialmsg,Commsg)
;   // the first line in the gui
        layout1.setPadding(new Insets(20, 20, 20, 20));

        VBox layout0=new VBox(20);
        layout0.getChildren().addAll(ldrlabel,ldrtext);
// second line in gui
        layout0.setPadding(new Insets(20, 20, 20, 20));

        VBox layout2 = new VBox(20);
        layout2.setPadding(new Insets(20, 20, 20, 20));
        layout2.getChildren().addAll(ledlabel,box1,
box2,box3,box4,msglabel,ledtext); // third line in the gui

        VBox layout3=new VBox(20);
        layout3.getChildren().addAll(lcdlabel,lcdtext,writebutton);
// lcd line
        layout3.setPadding(new Insets(20, 20, 20, 20));

        HBox layout4=new HBox(20);

layout4.getChildren().addAll(layout1,layout0,layout2,layout3,exitbutton
);     // adding all lines to the same layout

        VBox layout5=new VBox(20);
        layout5.getChildren().addAll(layout4,weltext);
        layout5.setAlignment(Pos.CENTER);
        Scene scene = new Scene(layout5, 900, 360);
        firstStage.setTitle("Computer to Arduino");
// Set the name or heder of the program
        firstStage.setScene(scene);
// set all layouts to the program stage
        firstStage.setResizable(false);
// closing the maximizing control of the program
        firstStage.show();
// launching the stage
    }

    //Connecting to the Arduino function
     public boolean connectToArduino(String port){

        boolean pass = false;
        SerialPort SP;
        SP = new SerialPort(port);
// creating a serial port
        try {
            SP.openPort();
// open the serial port
            SP.setParams(
//defining serial connectin parameters
                    SerialPort.BAUDRATE_9600,
// baud rate parameter
```

```
                    SerialPort.DATABITS_8,
// rate of bit exchange
                    SerialPort.STOPBITS_1,
                    SerialPort.PARITY_NONE);
            SP.setEventsMask(MASK_RXCHAR);
            SP.addEventListener((SerialPortEvent serialPortEvent) -> {
                if(serialPortEvent.isRXCHAR()){
                    try {

                        byte[] b = SP.readBytes();
                        int value = b[0] & 0xff;
//convert to integer
                        String st = String.valueOf(value);
                        //Update label in ui thread
                        Platform.runLater(() -> {
                            ldrtext.setText(st); //Showing LDR result
                        });

                    } catch (SerialPortException ex) {

Logger.getLogger(ComputerToArduino.class.getName())
                                .log(Level.SEVERE, null, ex);
                    }
                }
```

Comment: in the connectArduino function I told it to read data from serial port which Arduino send to the port in bytes form and this data represents the readings of of the LDR. After creating connectArduino function I now need to terminate this connection, so I made disconnectArduino function to terminate this connection if I removed the port and there is no port to communicate with.

```
            });

            arduinoPort = SP;
            pass = true;
        } catch (SerialPortException ex) {
            Logger.getLogger(ComputerToArduino.class.getName())
                    .log(Level.SEVERE, null, ex);
            System.out.println("SerialPortException: " +
ex.toString());
        }

        return pass;
    }
    //Disconnecting the Arduino function
      public void disconnectArduino(){

        if(arduinoPort != null){    // checking if there is no port to
connect
```

```
            try {
                arduinoPort.removeEventListener();    //stop connction
with port
                if(arduinoPort.isOpened()){
                    arduinoPort.closePort();           // closing
arduino port
                }

            } catch (SerialPortException ex) {
                Logger.getLogger(ComputerToArduino.class.getName())
                        .log(Level.SEVERE, null, ex);
            }
        }
    }

    @Override
    public void stop() throws Exception {
        disconnectArduino();
        super.stop();
    }


        EventHandler writebuttonhandler = new EventHandler() {
// controlling sending data to lcd
            @Override
            public void handle(Event event) {

                 try {
            if(arduinoPort != null)
            arduinoPort.writeString(lcdtext.getText());
// set text to lcd from text in the gui

            else
                Commsg.setText("Select port first!");
        } catch (SerialPortException ex) {

Logger.getLogger(ComputerToArduino.class.getName()).log(Level.SEVERE,
null, ex);
        }

            }
        };

        EventHandler led1_handler = new EventHandler() {
            @Override
            public void handle(Event event) {
              try {
            if(box1.isSelected()){
                if(arduinoPort != null){      //if there is connection
port
                    arduinoPort.writeString("10@");  //send string data
to Arduino serial
                    ledtext.setText("LED1 is ON");
                }else{
                    Commsg.setText("Select port first!");
                }
            }else {
```

```
                if(arduinoPort != null){     //if there is no
connection port
                    arduinoPort.writeString("20#");  //send different
data
                    ledtext.setText("LED1 is OFF");
                }else{
                    Commsg.setText("Select port first!");
                }
            }
        } catch (SerialPortException ex) {

Logger.getLogger(ComputerToArduino.class.getName()).log(Level.SEVERE,
null, ex);
        }
            }
        };

        EventHandler led2_handler = new EventHandler() {
            @Override
            public void handle(Event event) {
              try {
            if(box2.isSelected()){
                if(arduinoPort != null){
                    arduinoPort.writeString("30$");
                    ledtext.setText("LED2 is ON");
                }else{
                     Commsg.setText("Select port first!");
                }
            }else {
                if(arduinoPort != null){
                    arduinoPort.writeString("40%");
                    ledtext.setText("LED2 is OFF");
                }else{
                     Commsg.setText("Select port first!");
                }
            }
        } catch (SerialPortException ex) {
```

comment: After designing my GUI I made it interactive by making EventHandler for every ActionEvent that the user may do. Firstly, for the writebutton if the user clicked it (ActionEvent) the Handler will print the lcdtext text to the Arduino port if there is port if not it will Alert the user with message "arduinoPort not connected!".

Then Handler for every checkbox to send a string to ArduinoPort to tell it if it was chosen or not.

```
            });

            arduinoPort = SP;
            pass = true;
        } catch (SerialPortException ex) {
            Logger.getLogger(ComputerToArduino.class.getName())
                    .log(Level.SEVERE, null, ex);
            System.out.println("SerialPortException: " +
ex.toString());
        }

        return pass;
     }
     //Disconnecting the Arduino function
       public void disconnectArduino(){

        if(arduinoPort != null){    // checking if there is no port to
connect
            try {
                arduinoPort.removeEventListener();    //stop connction
with port
                if(arduinoPort.isOpened()){
                    arduinoPort.closePort();           // closing
arduino port
                }

            } catch (SerialPortException ex) {
                Logger.getLogger(ComputerToArduino.class.getName())
                        .log(Level.SEVERE, null, ex);
            }
        }
    }

    @Override
    public void stop() throws Exception {
        disconnectArduino();
        super.stop();
    }


        EventHandler writebuttonhandler = new EventHandler() {
// controlling sending data to lcd
            @Override
            public void handle(Event event) {

                 try {
            if(arduinoPort != null)
            arduinoPort.writeString(lcdtext.getText());
// set text to lcd from text in the gui

            else
                Commsg.setText("Select port first!");
        } catch (SerialPortException ex) {

Logger.getLogger(ComputerToArduino.class.getName()).log(Level.SEVERE,
null, ex);
```

```
        }

            }
        };

        EventHandler led1_handler = new EventHandler() {
            @Override
            public void handle(Event event) {
              try {
            if(box1.isSelected()){
                if(arduinoPort != null){      //if there is connection
port
                    arduinoPort.writeString("10@");  //send string data
to Arduino serial
                    ledtext.setText("LED1 is ON");
                }else{
                    Commsg.setText("Select port first!");
                }
            }else {
                if(arduinoPort != null){     //if there is no
connection port
                    arduinoPort.writeString("20#");  //send different
data
                    ledtext.setText("LED1 is OFF");
                }else{
                    Commsg.setText("Select port first!");
                }
            }
        } catch (SerialPortException ex) {

Logger.getLogger(ComputerToArduino.class.getName()).log(Level.SEVERE,
null, ex);
        }
            }
        };

        EventHandler led2_handler = new EventHandler() {
            @Override
            public void handle(Event event) {
              try {
            if(box2.isSelected()){
                if(arduinoPort != null){
                    arduinoPort.writeString("30$");
                    ledtext.setText("LED2 is ON");
                }else{
                     Commsg.setText("Select port first!");
                }
            }else {
                if(arduinoPort != null){
                    arduinoPort.writeString("40%");
                    ledtext.setText("LED2 is OFF");
                }else{
                     Commsg.setText("Select port first!");
                }
            }
        } catch (SerialPortException ex) {
```

Comment: every CheckBox send a different string to the ArduinoPort if it was selected and another string if it was not but considering if the there is port to connect or not.

```
Logger.getLogger(ComputerToArduino.class.getName()).log(Level.SEVERE,
null, ex);
        }
            }
        };

        EventHandler led3_handler = new EventHandler() {
            @Override
            public void handle(Event event) {
            try {
            if(box3.isSelected()){
                if(arduinoPort != null){
                    arduinoPort.writeString("50^");
                    ledtext.setText("LED3 is ON");
                }else{
                     Commsg.setText("Select port first!");
                }
            }else {
                if(arduinoPort != null){
                    arduinoPort.writeString("60&");
                    ledtext.setText("LED3 is OFF");
                }else{
                     Commsg.setText("Select port first!");
                }
            }
        } catch (SerialPortException ex) {

Logger.getLogger(ComputerToArduino.class.getName()).log(Level.SEVERE,
null, ex);
            }
            }
        };

        EventHandler all_handler = new EventHandler() {
            @Override
            public void handle(Event event) {
            try {
            if(box4.isSelected()){
                if(arduinoPort != null){
                    arduinoPort.writeString("70");
                    ledtext.setText("All LEDs are ON");
                }else{
                     Commsg.setText("Select port first!");
                }
            }else {
                    box1.setSelected(false);
                    box2.setSelected(false);
                    box3.setSelected(false);
                if(arduinoPort != null){
                    arduinoPort.writeString("80");
```

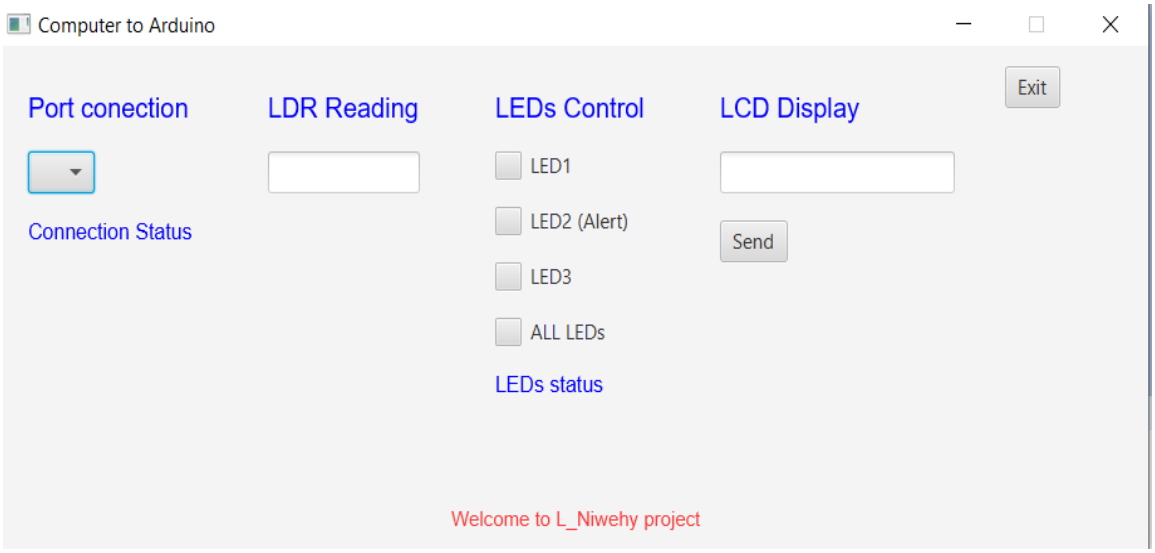
```
                    ledtext.setText("All LEDs are OFF");
                }else{
                     Commsg.setText("Select port first!");
                }
            }
        } catch (SerialPortException ex) {

Logger.getLogger(ComputerToArduino.class.getName()).log(Level.SEVERE,
null, ex);
        }
            }
        };

//Entry point to the program
public static void main(String[] args) {
        launch(args);  //make program go to execute Application class
first
    }

}
```

Comment: after declaring the EventHandler for every Event source object the program now is interactive. Finally I wrote the main function where the program start executing and its turn will make the program to execute application class first so start method will be executed and all written code go.

Run JavaFx code

Running JavaFx code we will get the following GUI as designed.

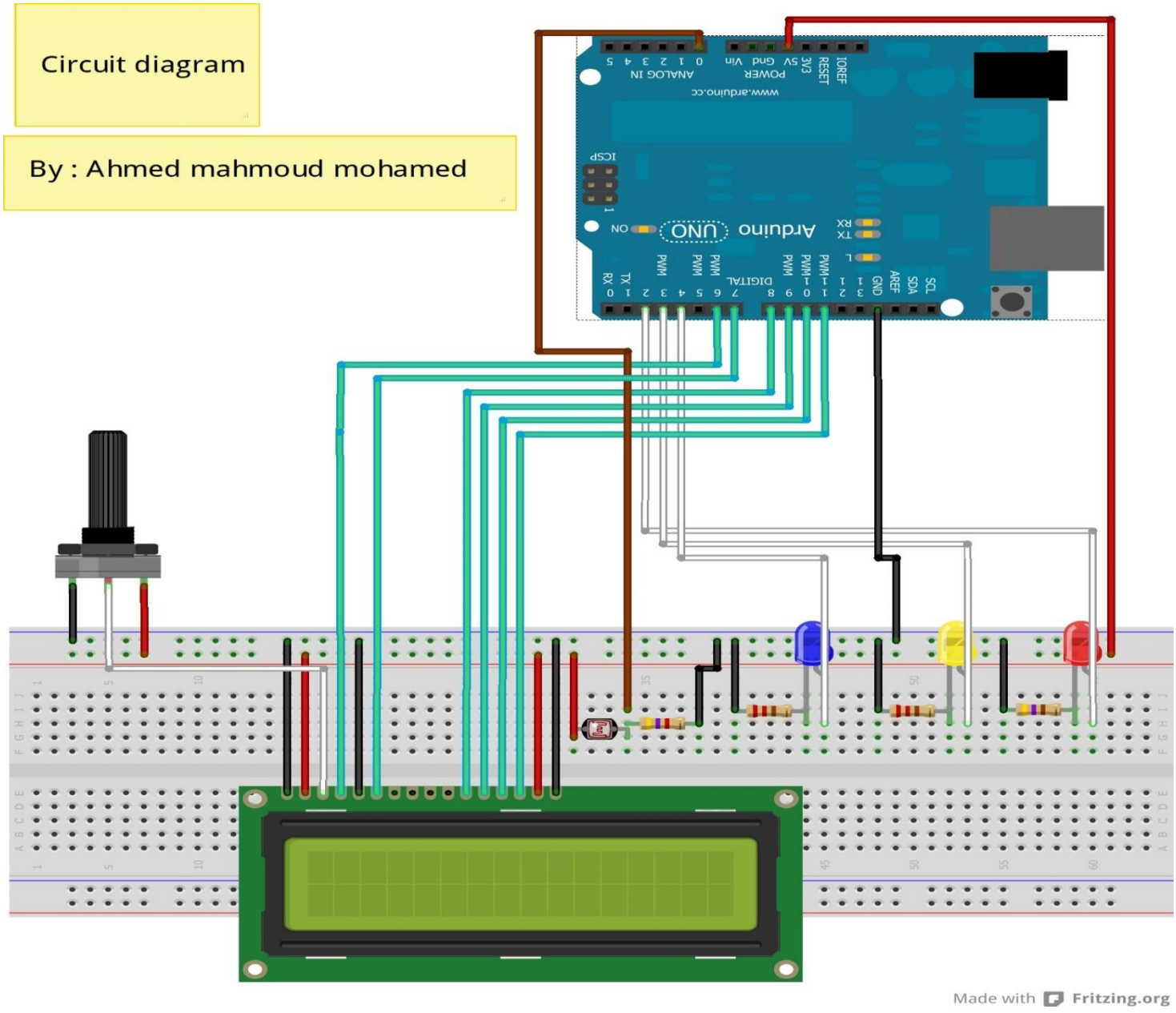
1. Combo box: labeled with "Serial connection" to check which port to connect.
2. Text field: labeled with "LDR Data " to receive and display LDR reading.
3. 3 check boxes labeled with " LED control " to control the three LEDs connected to the Arduino
4. Text field: labeled with " LCD 16x2 control" to can write a text on it and send it to display it on LCD.
5. Label: labeled with "LEDs control" to display a message about the 3 LEDs status.



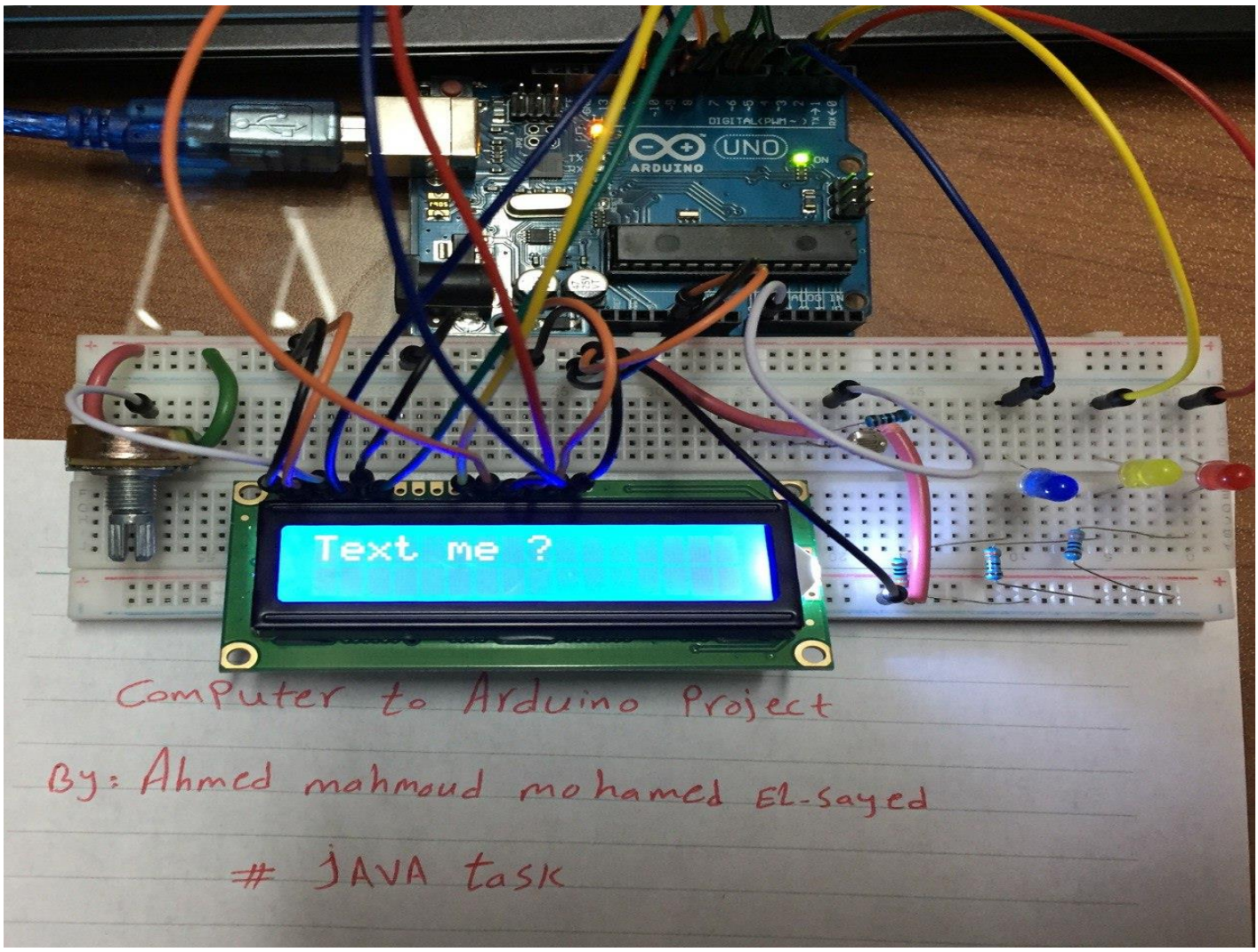
SCHEMATICS

I have designed my circuit using Fritzing program which contain:

1. Arduino Board
2. 3 LEDs
3. LER
4. LCD



Here is a photo of my circuit after connecting it:



Arduino code

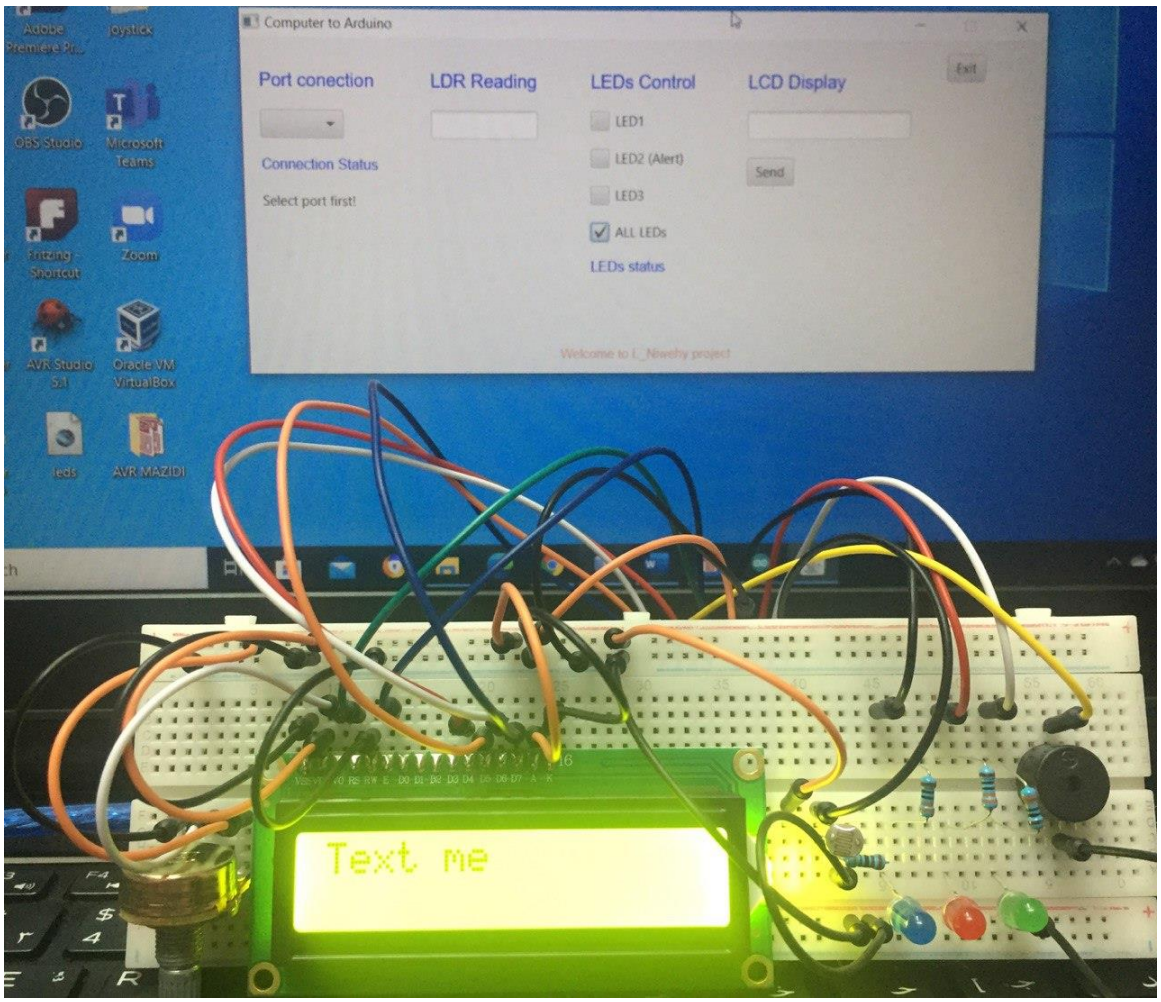
I used LCD to display the string I write in LCD textbox so I included the liquidCrystal.h library to can use it. I then define all Arduino pins I will deal with and set up these pins and the serial monitor with the same baud rate as I set on java code to communicate without any missing data.

```
#include <LiquidCrystal.h>
String stringVal;
int led1 = 2;
int led2 = 3;
int led3 = 4;
int buz = 12;
int ldrpin=A0;
int ldrVal=0;

// initialize the library by associating any needed LCD interface pin
// with the arduino pin number it is connected to
const int rs = 6, en = 7, d4 = 8, d5 = 9, d6 = 10, d7 = 11;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {
  Serial.begin(9600);
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(led3, OUTPUT);
  pinMode(buz,OUTPUT);

  lcd.begin(16, 2);
  lcd.print("Text me ");
}

void loop() {

  if (Serial.available() > 0) {
    stringVal = Serial.readString();      //reading serial data as
string
    if (stringVal == "10@" || stringVal == "20#" || stringVal == "30$"
        || stringVal == "40%" ||  stringVal == "50^" || stringVal ==
"60&"
        || stringVal == "70"  || stringVal == "80" || stringVal ==
"90")
    {
     int newVal = stringVal.toInt();
      switch (newVal) {
        case 10:
          digitalWrite(led1, HIGH);
          break;
```

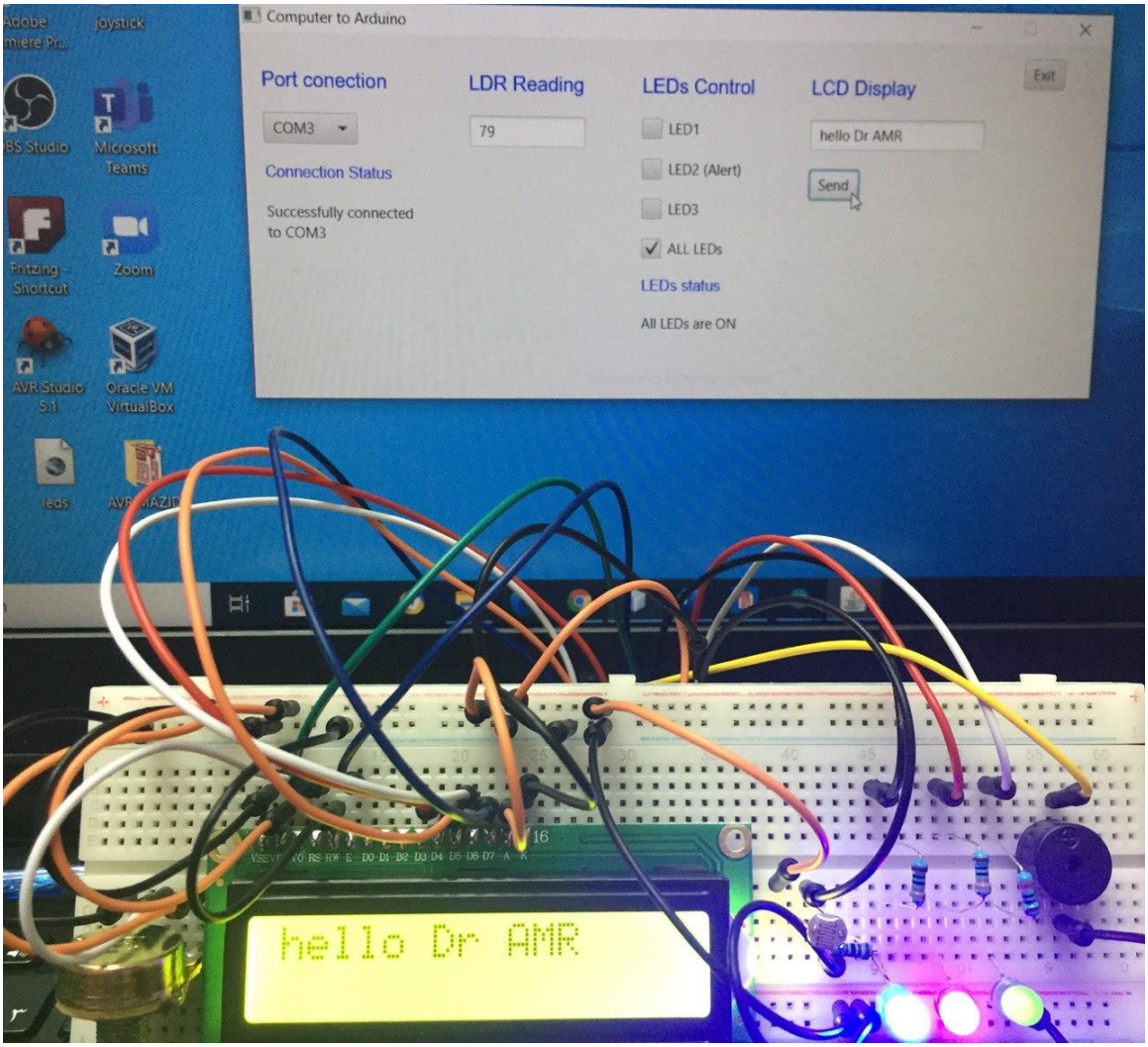
```
        case 20:
          digitalWrite(led1, LOW);
          break;

        case 30:
          digitalWrite(led2, HIGH);
          for(int i=0;i<=2;i++){
          tone(buz,450);
          delay(300);
          noTone(buz);
          delay(100);}
          break;

        case 40:
          digitalWrite(led2, LOW);
          break;

        case 50:
          digitalWrite(led3, HIGH);
          break;

        case 60:
          digitalWrite(led3, LOW);
          break;

        case 70:
          digitalWrite(led1, HIGH);
          digitalWrite(led2, HIGH);
          digitalWrite(led3, HIGH);
          break;

        case 80:
          digitalWrite(led1, LOW);
          digitalWrite(led2, LOW);
          digitalWrite(led3, LOW);
          break;

        case 90:
          digitalWrite(led1, LOW);
          digitalWrite(led2, LOW);
          digitalWrite(led3, LOW);
          lcd.clear();
          lcd.setCursor(0, 0);
          break;
      }
    }
    else {
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print( stringVal);
    }
  }
    ldrVal = analogRead(ldrpin);
    Serial.write(ldrVal);
    delay(250);
}
```

Comment: After setting up Arduino pins and serial monitor, I wrote the code in void loop to be executed all time it connected to the port. I wrote the code for LEDs, LDR and LCD all send and receive a string data to serial monitor depending on the procedure the user will do on GUI.

Simulation



This when we try to turn on the LEDs while unselecting port.



And this after selecting a port to connect

We got the reading of LDR sensor and turn on LEDs througth check boxes and send text to LCD.

Demo video for the simulation:

Here is a demo video for running and testing the project after I have finished in my YouTube channel: https://www.youtube.com/watch?v=JTH4baLzLqA&feature=youtu.be

Conclusion and proposal

We have designed a simple GUI using JavaFx to interface with Arduino to control three LEDs by choosing which one to be on and reading input data from LDR sensor through analog pin (A0) and display the result on text box in GUI. After all is done in the first, I faced a problem at the beginning it was that I couldn't make Jave code communicate with Arduino code. I did so research until I found jssc library which managed me to make this communication through serial monitor of the Arduino. I think that we can develop this project in the future by making this GUI in web server and this simple circuit could be a device we deal with daily like air conditioner, television, or any other device so we could control it through this web server using this simple GUI.

References

1. Deitel, P. J., Deitel, P. J., & Deitel, H. M. (2012). Java: how to program. Upper Saddle River, NJ: Prentice Hall.
2. JavaFX 9 by Example Kindle Edition
   by Carl Dea (Author), Gerrit Grunwald
3. Liang, Y. D. (2014). Intro to java programming, comprehensive version. Prentice Hall.
4. Arduino built in examples: https://www.arduino.cc/en/Tutorial/BuiltInExamples
5. How to design a GUI using javaFx: https://www.youtube.com/watch?v=dqvFVbwKdJI&list=PLR1KtmaCt9BlKEQlw2tPmMBSgTOTmCrKJ