RC Car controlled by Java desktop Application

Supervised By: Eng. Eman Hesham

Contributors

Marwa Mustafa

Ayman Youssef

Ahmed M. Soliman

Mohamed Abdel Fatah

Ahmed Yehia

Table of content

1. Abstract

The main idea of the project is to control an arduino based remote controller Car using a desktop Java application that uses Javafx as it is gui framework, the connection between the project main two components (the hardware and software) is done through Bluetooth that works on top of serial communication protocol or by connecting the arduino directly to any port of the lab ports.

2. Software Implementation

The graphical user interface (gui) was built using javafx, a graphical toolkit that provides a set of functionalities in order to make gui development easier that before (in applet and swing)

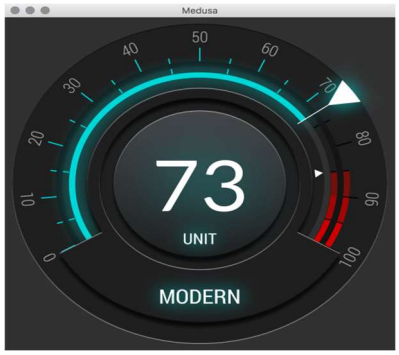
We can classify the gui built on this project into 3 main parts:

- **Gauge** to create the speedometer object
- **Slider** to control the speed of the car
- **Keyboard event listener** to control the motion of the car

We will go in more details of each part along with other components used

2.1. Gauge

It is a library that attempts to fill the gap in the javaFx list of features, the library was made by a german software engineer called Gerrit Granwald (Hansolo).
Gerrit created the Medusa library in 2015, it is a great library that includes many gauges like speedometers and watches and many other gauges.



2.2. Slider

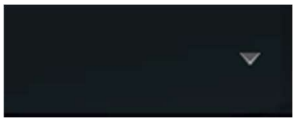
It is a library offered by JavaFx to create a slider and use it to control whatever values you want to control.
We are using a horizontal slider to control the speed of the car.

2.3. Start Button

JavaFx offers a great Button library to create a button and set its shape and event handlers.

2.4. ComboBox

Great library to create a combobox to store our the system ports so the user can choose the port that is connected to arduino.



2.5. Photos (Image View)

The project design contains 7 images to make the final form more friendly to users.

2.6. Event Handlers

2.6.1. Slider mouse drag handler

When the slider value is changed due to dragging the mouse over it, the speedometer value and RPM gauge values are changed also.

```
278
279         @FXML
280         void dragMouseSlider(MouseEvent event
281         ) {
282
283             int sliderValue = (int) hSlider.getValue();
284
285             //value to be sent to the Spedo Meter
286             float SpedoMeterValue = (float) (sliderValue * 50) / 255;
287             spedometer.setValue(SpedoMeterValue);
288
289             //value to be sent to the RPM guage    (RPM = %duty-cycle * 5.5)
290             //duty-cycle=slidervalue/1(s)*100
291             float RPMValue = (float) ((sliderValue * 5.5) / 60);
292             rpm.setValue(RPMValue);
293
294             //value to be sent to the Feul consumption
295             //Assume the car fuel tank has 12 gallon of fuel
296             i = i + RPMValue;
297             float fuelValue = (float) ((i / 1.6) / 12);
298             fuel.setValue(fuelValue);
299             if (RPMValue >= 50) {
300                 fuel.setValue(50);
301             } else {
302                 fuel.setValue(fuelValue);
303             }
304
```

```
304
305             //value to be sent to the Heat guage
306             int heatValue = sliderValue / 20;
307             heat.setValue(heatValue);
308
309         }
```

2.6.2. Anchor pane Key pressed handler

When any of the control keys are pressed a value is sent to the arduino, and according to this value the car will take an action.

For further information about the communication algorithm check connections section

2.6.3. Anchor pane Key release handler

When any of the control keys are released a value is sent to the arduino, so the car will stop.
We want to move the car only when the control keys are pressed.

2.6.4. Start button pressed handler

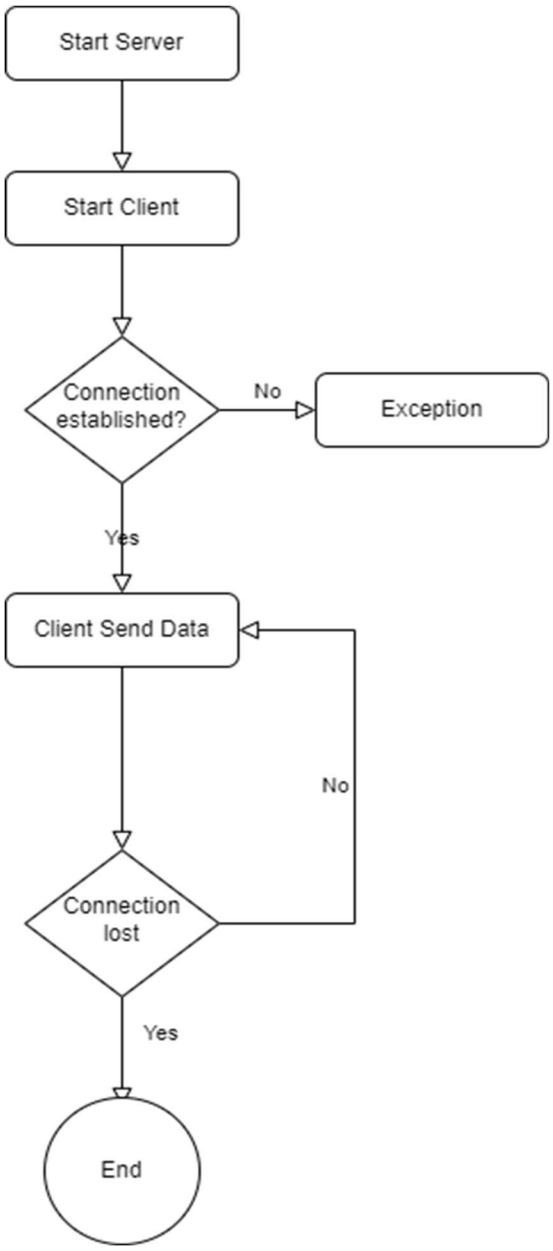
When the button is pressed the application will connect to the arduino so users can control the car.
If there is no chosen port or there is a problem in the connection, an alerting message will pop-up to inform the user of what he should do.
If the application connects successfully to the MCU the button color will be red and its text will be “End”.
If the user clicks on it again, the connection will terminate and “Start” shows up again.

2.7. Socket Programming

As an extra feature, Socket programming was used to create a server that communicate with all connected cars through their main application (what we can call clients on this context) and plot their paths

Program flow:

1. The server starts to run before any other client.
2. Then it starts a blocking state (on a seperate thread) waiting for connection with a client.
3. When a client is started the connection is established between the server and the client.
4. Every time the client application sends an order to the car it also sends the same order to the server so that it tracks the orders given to the car and maps it’s path.



3. Hardware Implementation

Moving on to the hardware implementation the project used arduino as the microcontroller the controls the car, The microcontroller receives orders from the desktop application through the communication channel (will be covered in the next section) and processes those orders correspondingly.

The main components of the hardware side is:

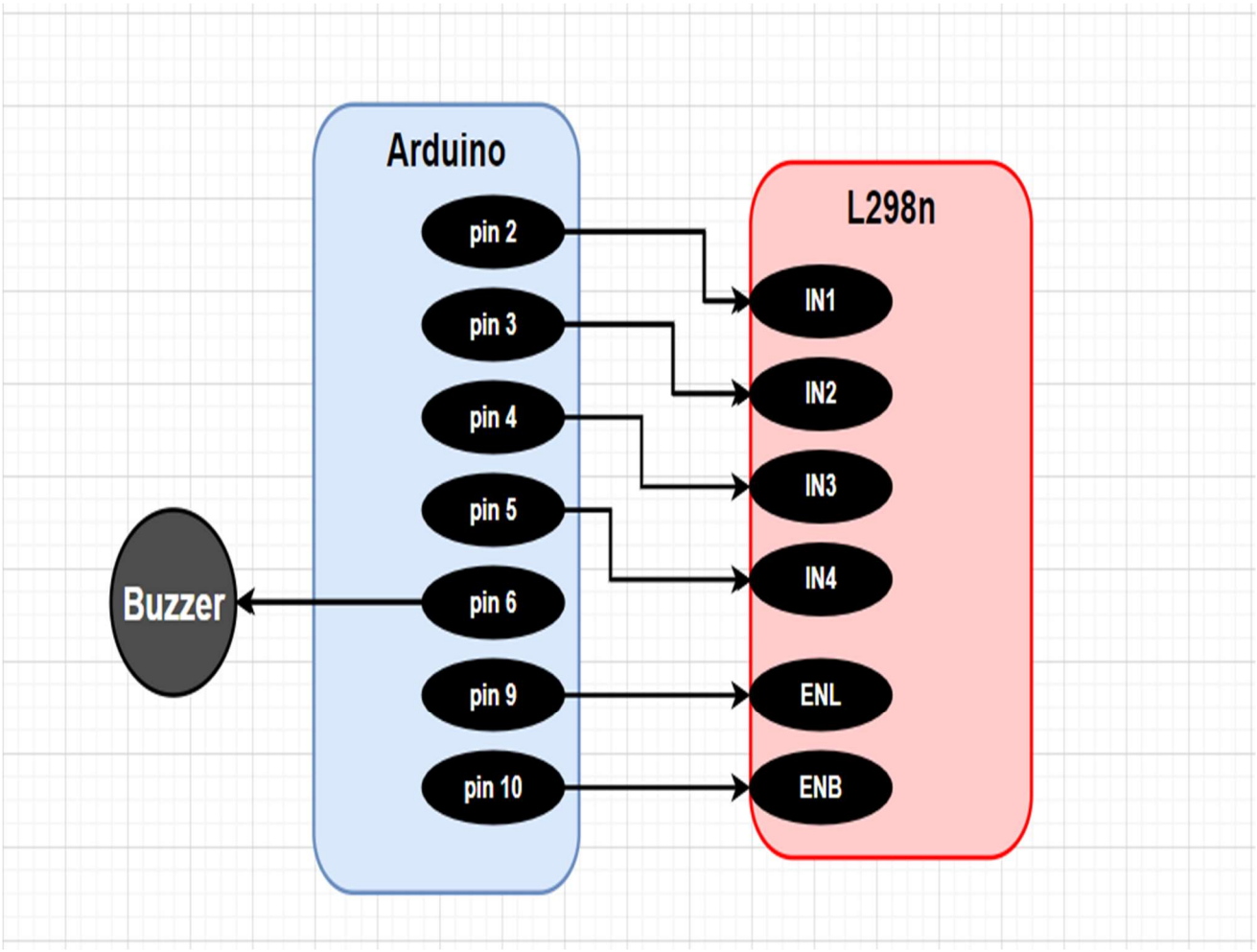
- The physical components (arduino - car body - motor driver - adopter)
- Connections
- Arduino code

3.1. Physical components

The components used on the project can be listed as follows

Component name	Purpose
Arduino uno	The controlling brain
Motor driver (H-Bridge)	To control the car motors
HC-05 Bluetooth Module	To achieve wireless comm.

3.2. Connections



3.3. Arduino code


```
/* set up l298 input pin on IN1 - 2 , IN2 - 3 , IN3 - 4 , IN4 - 5 */
#define IN1    (2)
#define IN2    (3)
#define IN3    (4)
#define IN4    (5)
#define buzz   (6)
#define ENL    (9)
#define ENR    (10)

int f = 0;
int speedValue;
void setup()
{
  Serial.begin(9600);
  pinMode(13, OUTPUT);
  /* set up l298 input pin as Output pin*/
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  pinMode(buzz,  OUTPUT);

  pinMode(ENL, OUTPUT);
  pinMode(ENR, OUTPUT);

  analogWrite(ENL, 0);
  analogWrite(ENR, 0);

}
```

arduino | Arduino 1.8.13

File Edit Sketch Tools Help

arduino

```
void loop()
{

  f = Serial.read();

  /* move forward */
  if (f <= 20 && f >= 0)
  {
    f = f * 12.75;
    analogWrite(ENL, f);
    analogWrite(ENR, f);
    Forward_Motion();
  }
  /* move Downward */
  else if (f <= 42 && f >= 22)
  {
    f = (f - 22) * 12.75;
    analogWrite(ENL, f);
    analogWrite(ENR, f);
    Backward_Motion();
  }
  /* move left */
  else if (f <= 64 && f >= 44)
  {
    f = (f - 44) * 12.75;
    analogWrite(ENL, f);
    analogWrite(ENR, f);
    Left_Motion();
  }
```

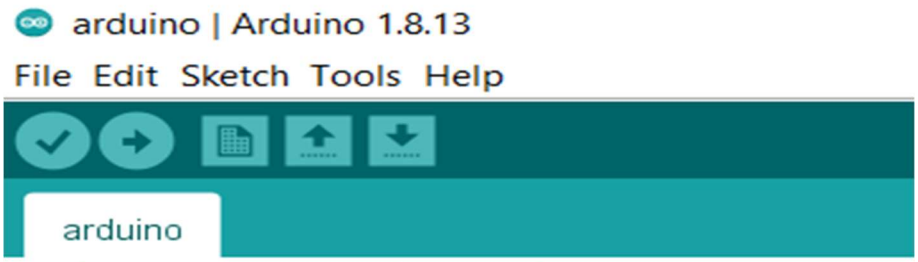
arduino | Arduino 1.8.13

File Edit Sketch Tools Help

arduino

```
  /* move right */
  else if (f <= 86 && f >= 66)
  {
    f = (f - 66) * 12.75;
    analogWrite(ENL, f);
    analogWrite(ENR, f);
    Right_Motion();
  }
  /* buzzer on */
  else if (f == 100)
  {
    digitalWrite(buzz, HIGH);
  }
  /* buzzer off */
  else if (f == 120)
  {
    digitalWrite(buzz, LOW);
  }
  /* stop the car */
  else if (f >= 130)
  {
    Stop_Motion();
    analogWrite(ENL, 0);
    analogWrite(ENR, 0);
  }

}
```




```
void Forward_Motion(void)
{
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
}
void Backward_Motion(void)
{
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
}
void Left_Motion(void)
{
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
}
void Right_Motion(void)
{
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
}
void Stop_Motion(void)
{
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
}
```

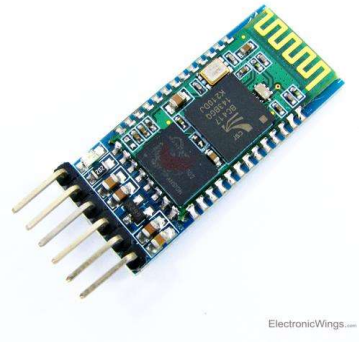
4. Communications

The Arduino board MCU is ATmega328p so the communication protocol registers are 8-bits, that means we are only limited to use a range from 0 to 255 to send data from the PC to the arduino.
We have 7 keyboard keys to control the car (W - S - D - A - O - L - K), data is transmitted when only 5 of them are pressed (W - S - D - A) and K.
When K is pressed 100 is sent to the arduino and it is translated there to switch the buzzer on, when K is pressed again 120 is sent and it is translated to switch the buzzer off.
we need to distinguish between the direction controllers (W - S - D - A) so there are 4 ranges, each range refers to one direction and the speed is embedded into that direction.
The slider value is divided by 12.75 and added to the start point of each direction.
Forward (W): the range is (from 0 to 20), one value is sent according to the speed or the slider value.
Downward (S): the range is (from 22 to 42), one value is sent according to the speed or the slider value.
Left (A): the range is (from 44 to 64), one value is sent according to the speed or the slider value.
Right (D): the range is (from 66 to 86), one value is sent according to the speed or the slider value.

To stop the car in the Key released handler 205 is sent and that value is translated on the arduino side to stop the car.

4.1. Serial Communications

Serial Communications were integral to the project, to send commands to the Arduino to control the car wirelessly to guarantee the freedom of the car movement, the communications were done by connecting to a serial computer port, and connecting the bluetooth of the computer to the HC - 05 bluetooth module, that communicates with the Arduino through UART.



4.2. End-User-Interface

The user interface is streamlined efficiently enough that the user simply chooses the COM port the Arduino is connected upon, and press on the button start, by these few steps the connection between the computer and the car is already established, and the car is waiting for the user’s commands, the commands are sent to the Arduino when the user presses on the keys that control direction or the speed of the car, the user can check which key does what function in the user guide.