Driver-Assistant Software Instruction

Jingwen Mo

Yiming Chen

Supervised by Prof. Quoc-Viet Dang

July 23th, 2018

1. Introduction

1.1 Purpose

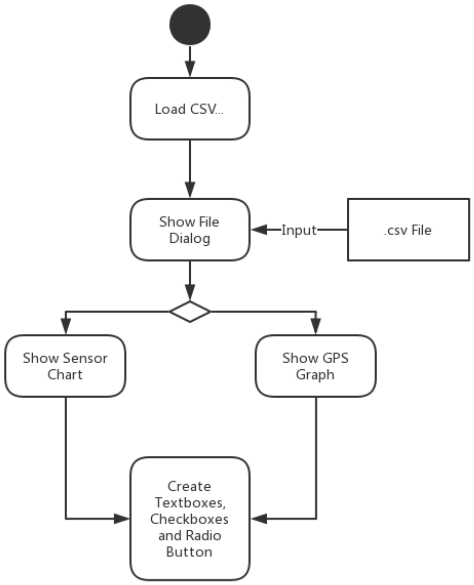
The Data-Graph Software is a PC based software developed to draw the graph which depicts different sensors for the racing car and show the GPS information. This instruction is for the software users and software tester.

1.2 Background

Racing car is one of the fiercest activities. It requires people to have access to different sensors in the car in order to evaluate the performance of the driver and train the driver by the feedback from the sensors. Our software is meant to help coach to transfer the data of different sensors to intuitive graphs easily.

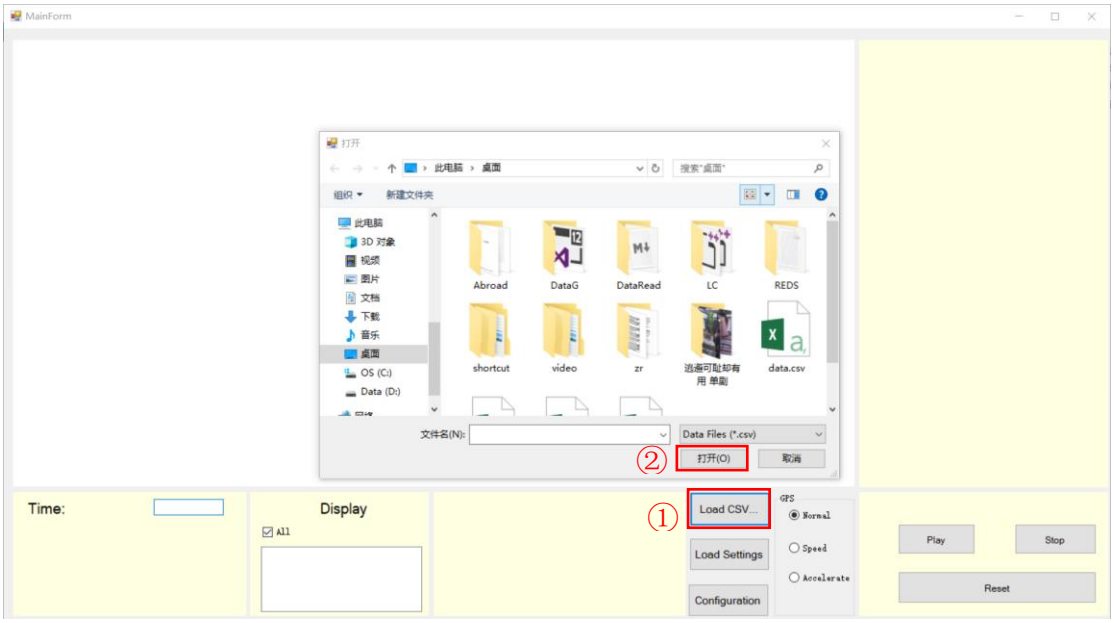
2. Single Run Form

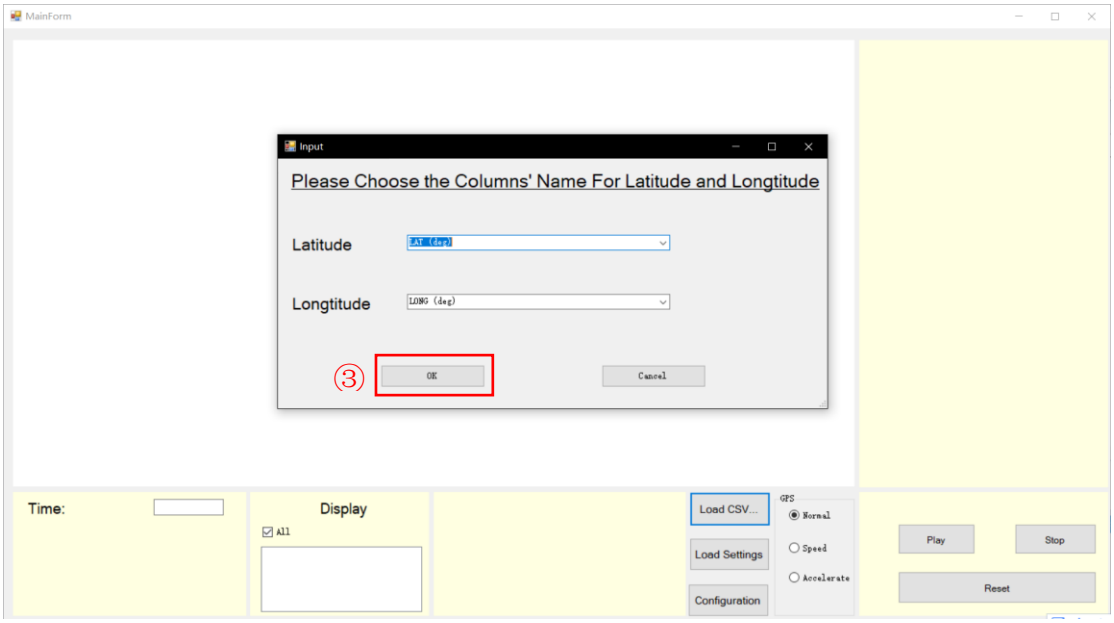
2.1 Load CSV Files

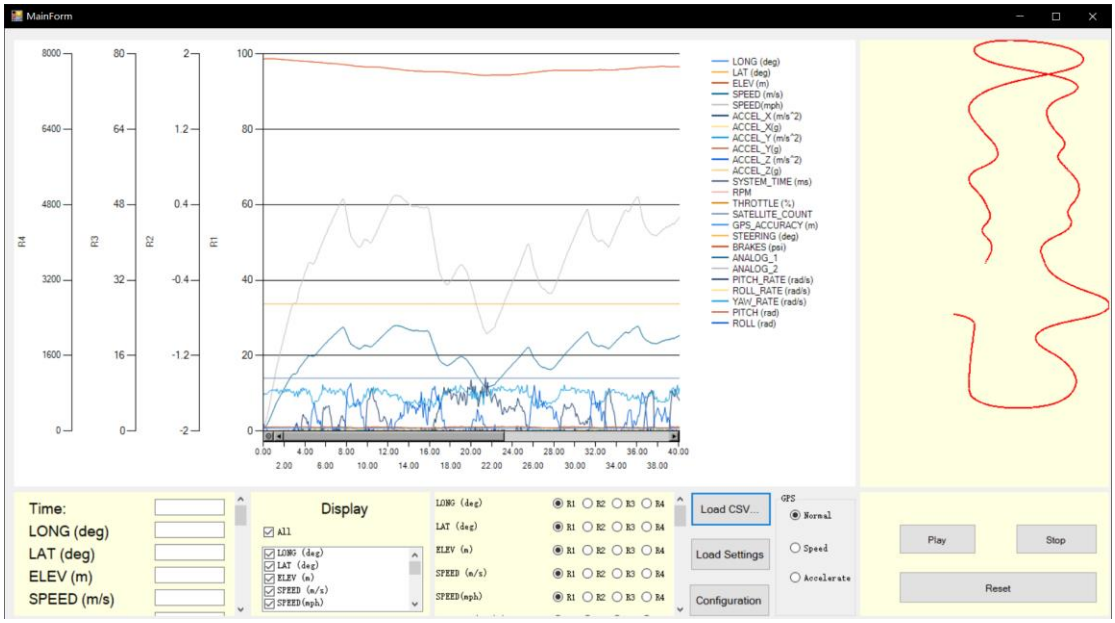


Call function of fileLoadingButton_Click and pop up a file dialog for users to choose their CSV file. After users choose their files and the columns for latitude and longitude, the software could draw graphs for different series of sensors and the GPS graph for the route of driving. Also, the software would dynamic create textboxes,

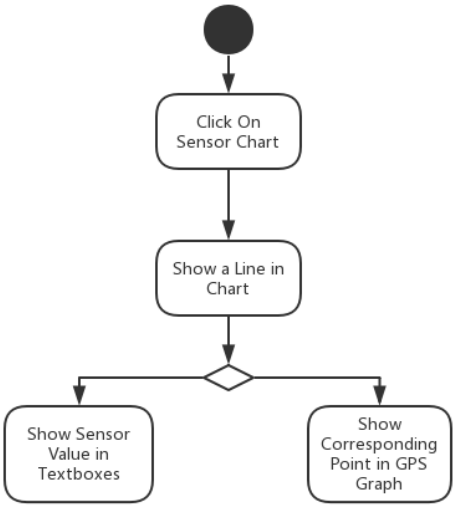
checkboxes and radio buttons for different sensors read from the CSV file.
Related Functions:
void fileLoadingButton_Click(object sender, EventArgs e)
DataTable OpenCSV(string filePath)







2.2 Show Specific Value of Points

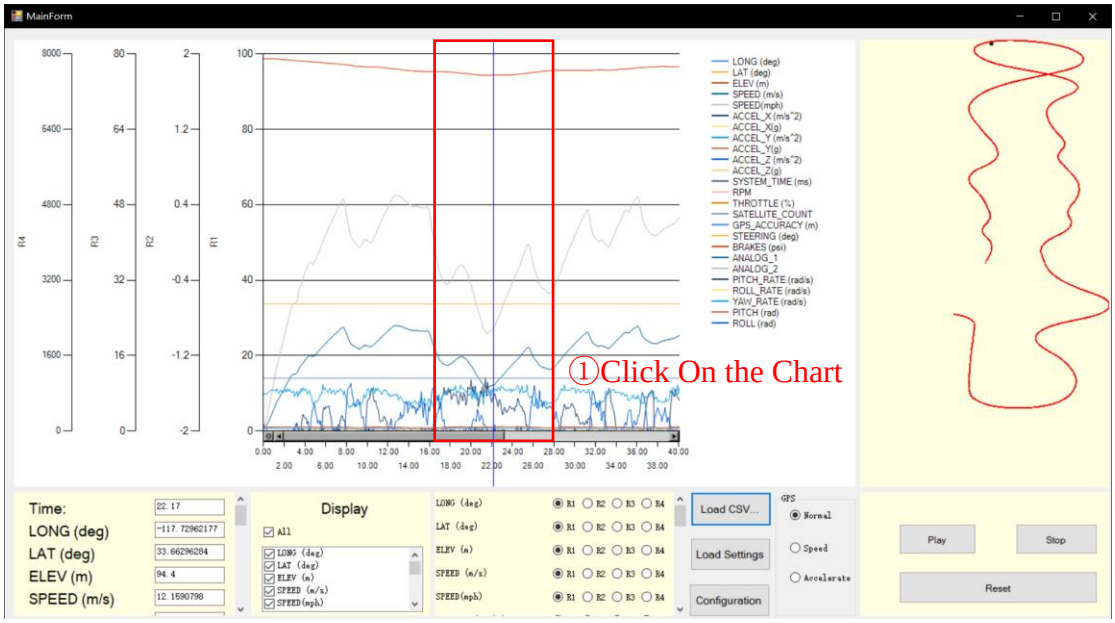


Users could click on the graph and show a vertical line at where the mouse clicked. Then the textboxes below would show the values of different sensors. Meanwhile, there would be a black point in the GPS graph to show the same place as you click on the chart.

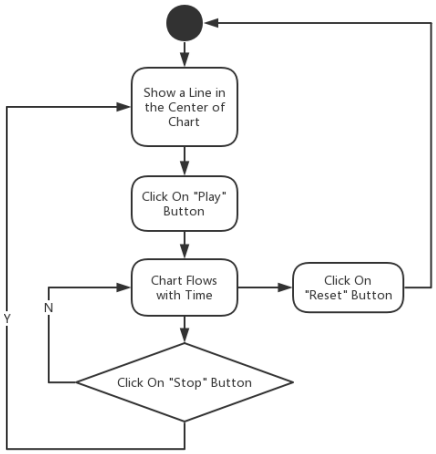
Related Functions:

void sensorChart_MouseClick(object sender, MouseEventArgs e)

int findLeftNear(double value, double[] array, int length)



2.3 Replay the Data



Users could click on “Play” button and call the function of buttonPlay_Click. This function enables chartTimer so that it could repeatedly execute the function of chartTimer_Tick at intervals. In the meantime, the textboxes below would show the values of sensors as the chart flowing. When users click on “Stop” button, the chart would stop flowing. After “Play” button is re-clicked, the chart would begin to flow from where it stops. The “Reset” button would let users to reset the chart to the original state.

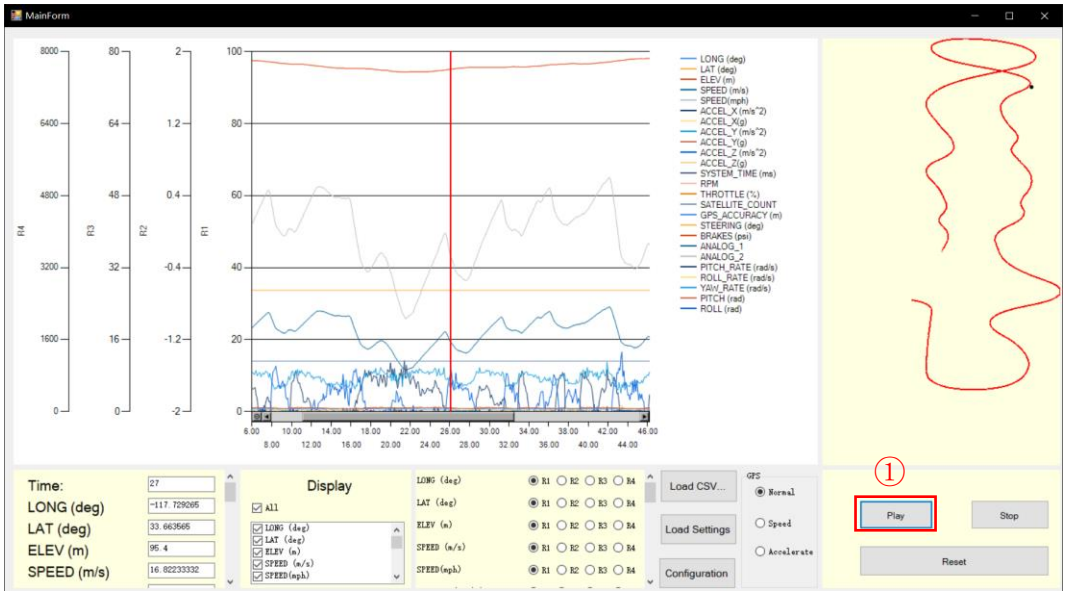
Related Functions:

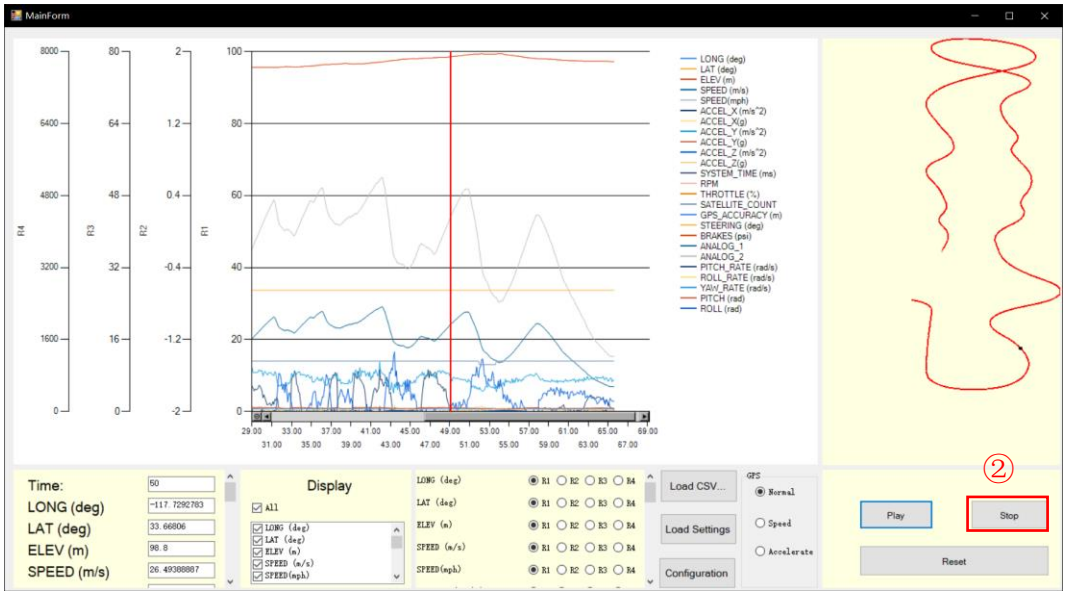
void buttonPlay_Click(object sender, EventArgs e)

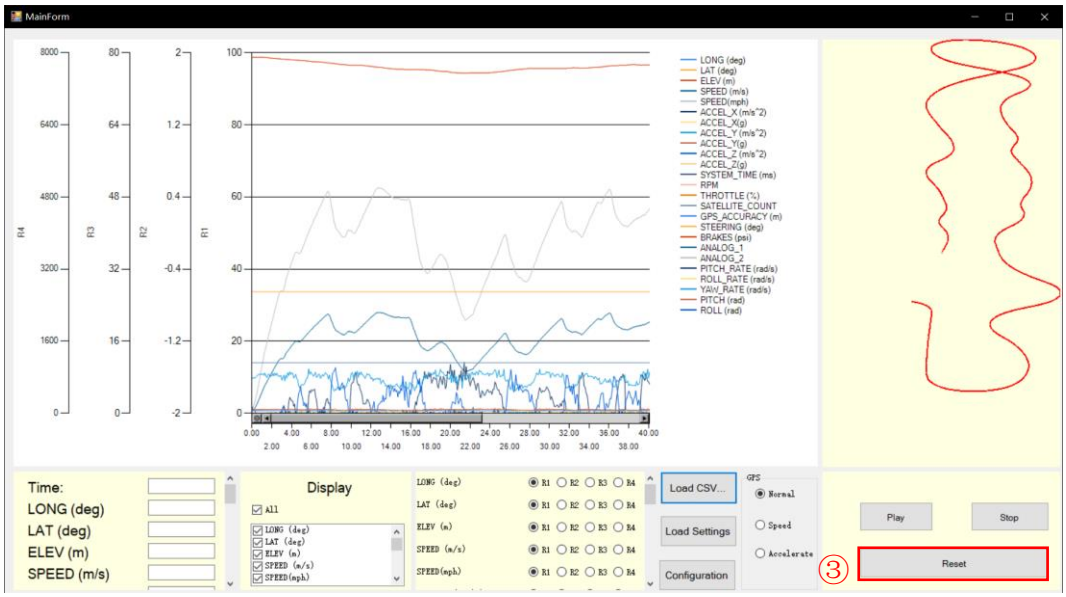
void buttonStop_Click(object sender, EventArgs e)

void resetButton_Click(object sender, EventArgs e)

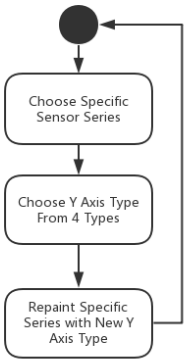
void chartTimer_Tick(object sender, EventArgs e)


MainForm
Time:
LONG (deg)
LAT (deg)
ELEV (m)
SPEED (m/s)
Display
All
Load CSV...
Load Settings
Configuration
GPS
Normal
Speed
Accelerate
Play
Stop
Reset
①


MainForm
Time:
LONG (deg)
LAT (deg)
ELEV (m)
SPEED (m/s)
Display
All
Load CSV...
Load Settings
Configuration
GPS
Normal
Speed
Accelerate
Play
Stop
Reset
②


MainForm
Time:
LONG (deg)
LAT (deg)
ELEV (m)
SPEED (m/s)
Display
All
Load CSV...
Load Settings
Configuration
GPS
Normal
Speed
Accelerate
Play
Stop
Reset
③

2.4 Choose Different Type of Y Axis



Users could change different series from one Y axis to another one. The software provides 4 different Y axes. We can switch between different axes by radio buttons dynamically created by fileLoadingButton_Click function.

Related Functions:

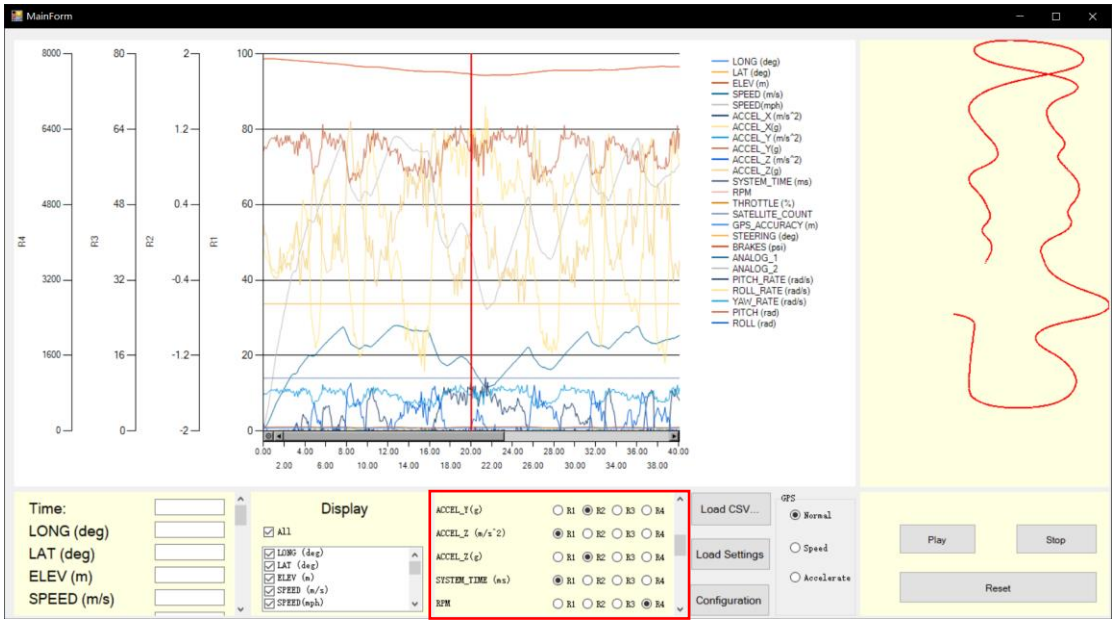
void change(int no, ChartArea caR)

void rb1_Click(object sender, EventArgs e)

void rb2_Click(object sender, EventArgs e)

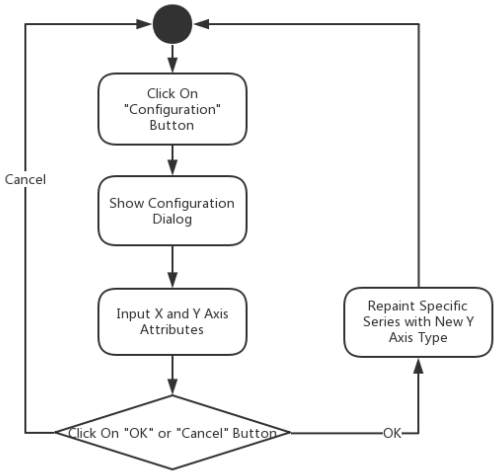
void rb3_Click(object sender, EventArgs e)

void rb4_Click(object sender, EventArgs e)



①Choose Different Y Types

2.5 Customize X axis and Y axis



Users could click on “Configuration” button to pop out a dialog for X axis and Y axis customization. Users could change the range of X axis range, scale and interval and Y axis range and type.

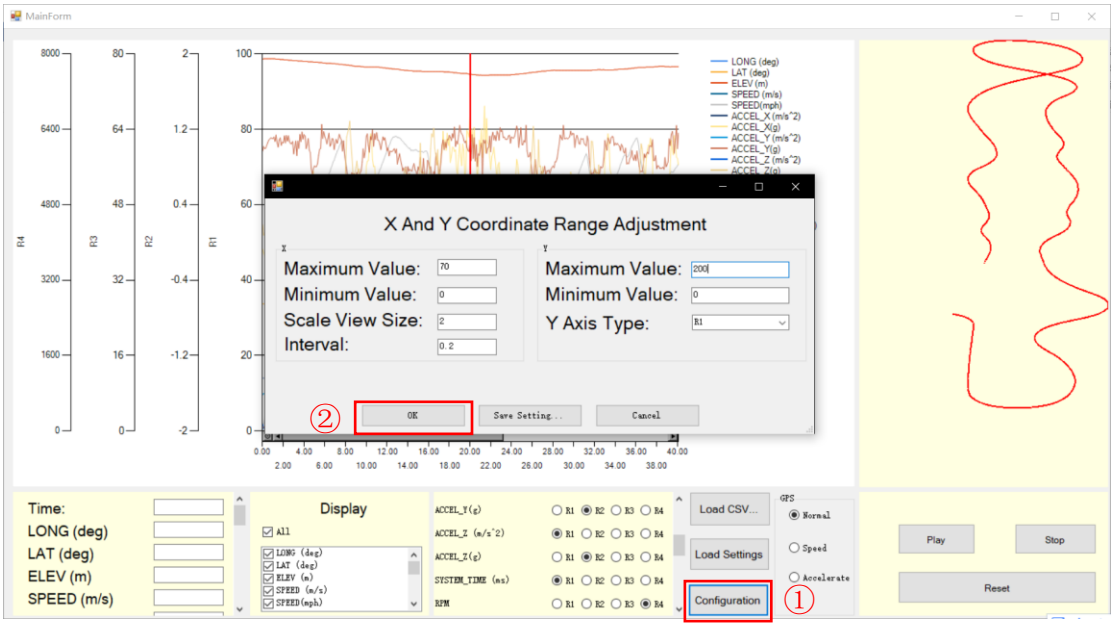
Related Functions and Forms:

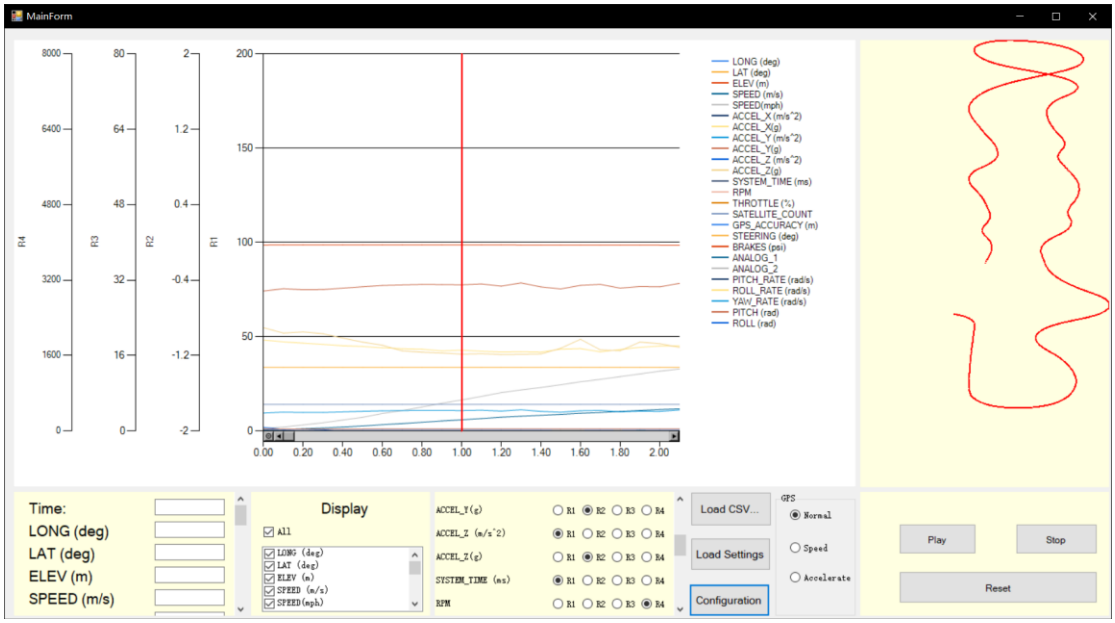
public partial class RangeForm : Form

void YRangeForm_Load(object sender, EventArgs e)

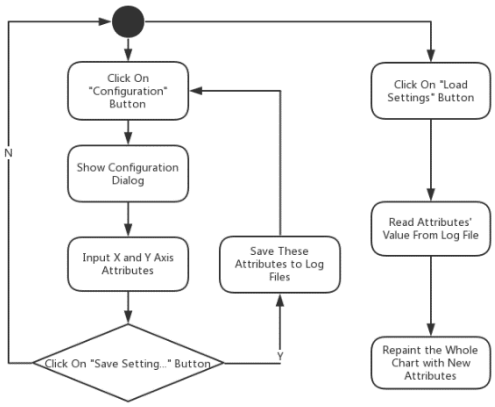
void confirmButton_Click(object sender, EventArgs e)

void ConfigureButton_Click(object sender, EventArgs e)





2.6 Save and Load Setting Log Files

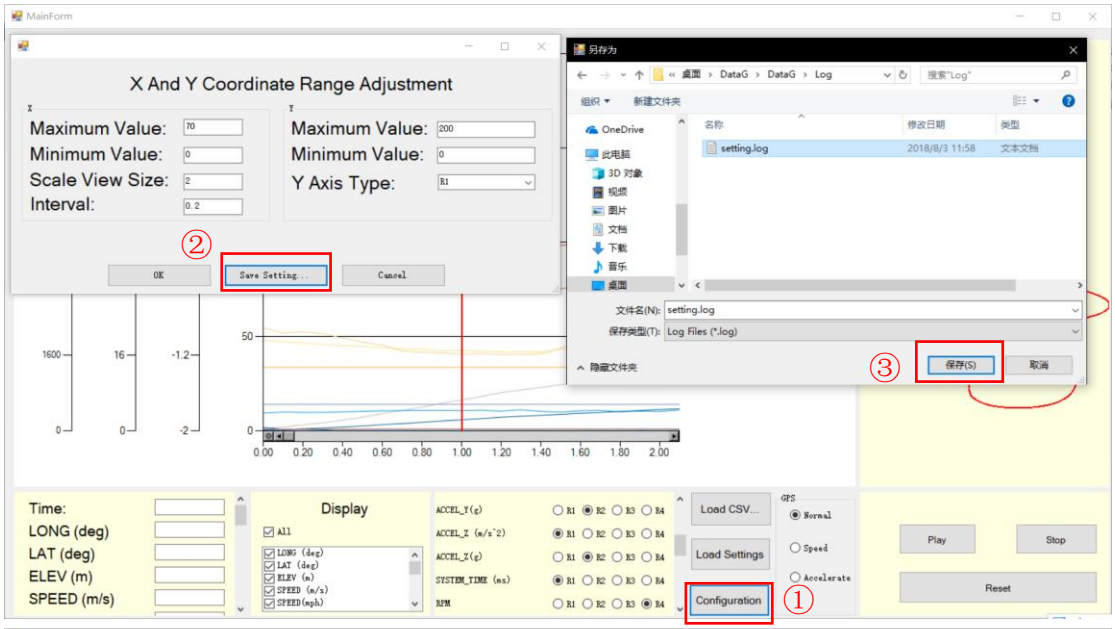


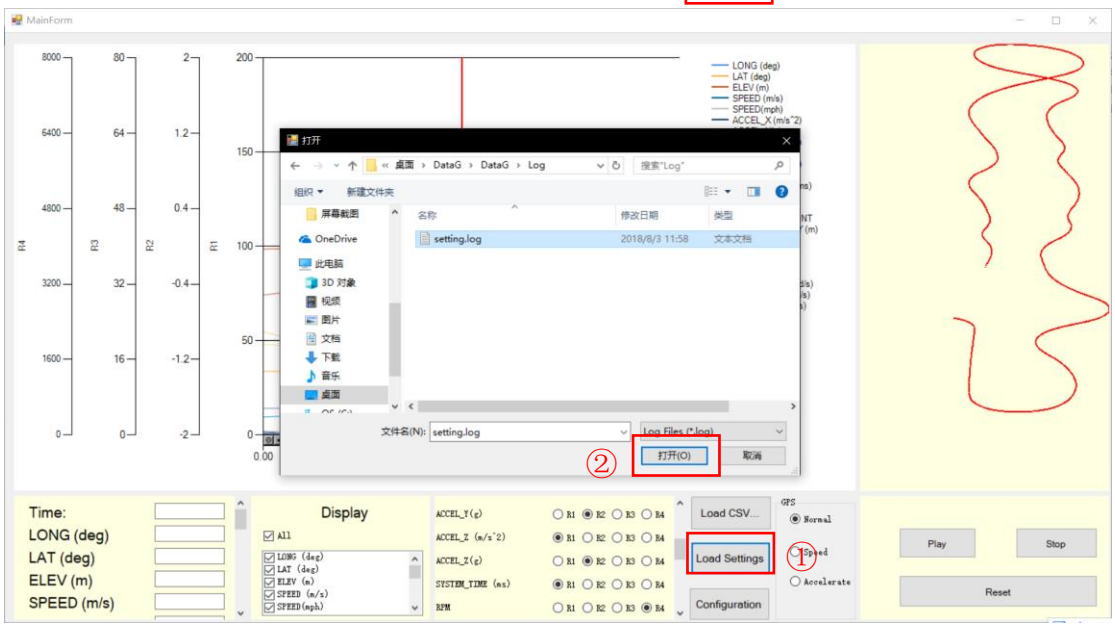
By clicking button of “Saving Setting…” in RangeForm, users could save log files with the X axis and Y axis configuration in specific form. Also, users could load log files created by them by clicking the button of “Loading Settings”.

Related Functions:

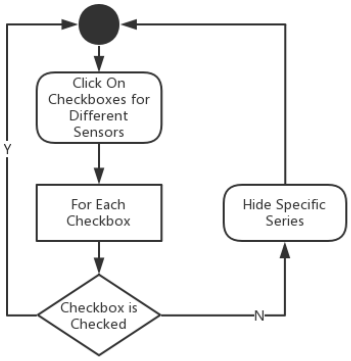
void settingSaveButton_Click(object sender, EventArgs e)

void settingButton_Click(object sender, EventArgs e)


MainForm
X And Y Coordinate Range Adjustment
Maximum Value:
Minimum Value:
Scale View Size:
Interval:
Maximum Value:
Minimum Value:
Y Axis Type:
OK
Save Setting...
Cancel
②
另存为
setting.log
文件名(N): setting.log
保存类型(T): Log Files (*.log)
保存(S)
取消
③
Time:
LONG (deg)
LAT (deg)
ELEV (m)
SPEED (m/s)
Display
All
Load CSV...
Load Settings
Configuration
①
Play
Stop
Reset


MainForm
打开
setting.log
文件名(N): setting.log
打开(O)
取消
②
Time:
LONG (deg)
LAT (deg)
ELEV (m)
SPEED (m/s)
Display
All
Load CSV...
Load Settings
Configuration
①
Play
Stop
Reset

2.7 Show or Hide Specific Series

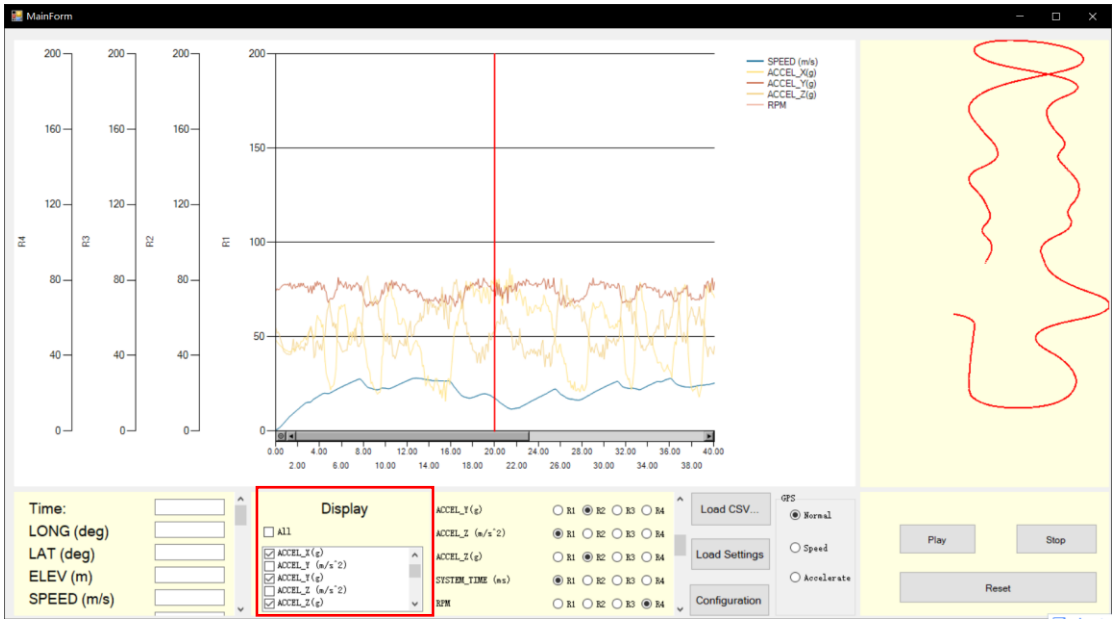


Users could click on different checkboxes created dynamically by fileLoadingButton_Click function. In addition, there is an allSelectedCheckBox for the selection of all.

Related Functions:

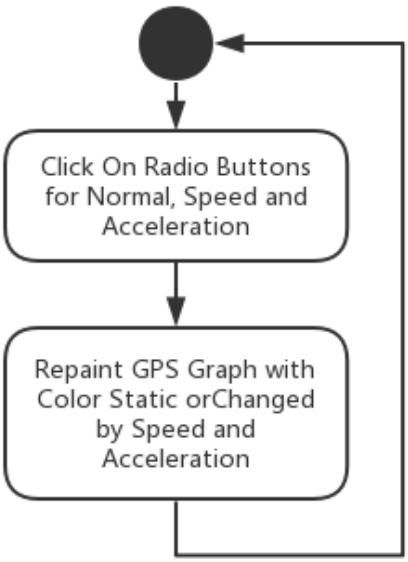
void sensorCheckedListBox_ItemCheck(object sender, ItemCheckEventArgs e)

void allSelectedCheckBox_CheckedChanged(object sender, EventArgs e)



①Click On Series Needed to be Show

2.8 Show Colored GPS Graph Changed by Speed or Acceleration

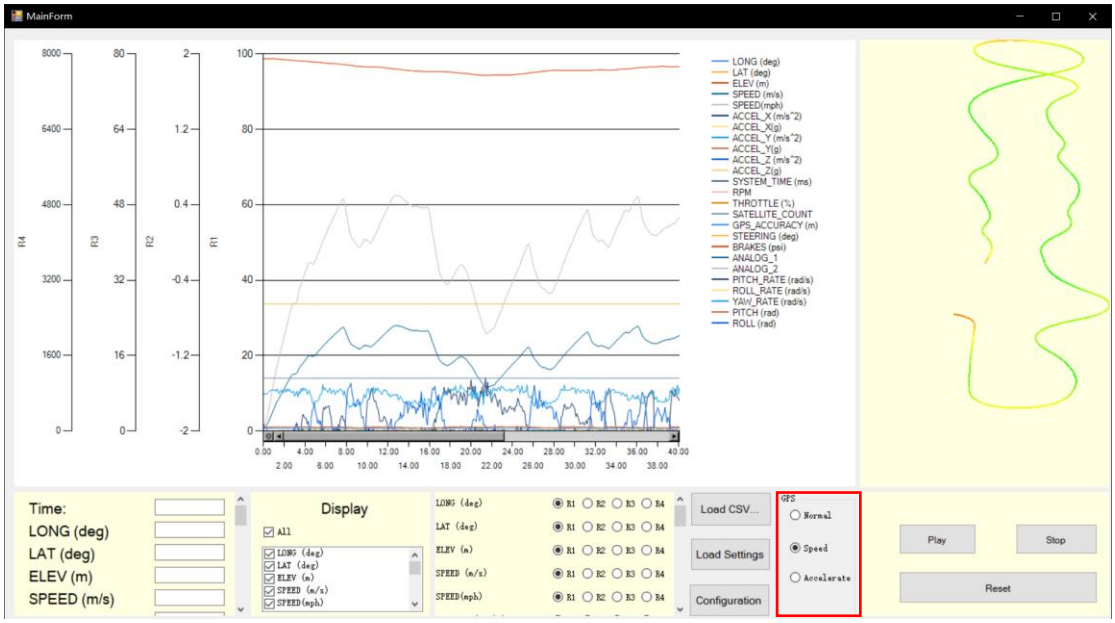


Users could click on "radioButton_Normal" to show normal GPS graph, "radioButton_Speed" to show GPS graph changed by speed where green represents high speed and red represents low speed and "radioButton_Accelerate" to show GPS graph changed by acceleration.

Related Functions:

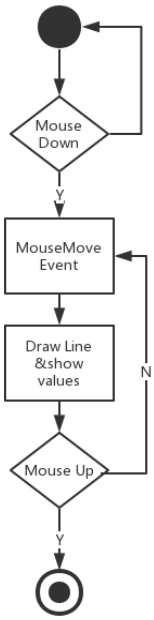
void radioButton_Normal_CheckedChanged(object sender, EventArgs e)

void radioButton_Speed_CheckedChanged(object sender, EventArgs e)
void radioButton_Accelerate_CheckedChanged(object sender, EventArgs e)
int colorRed(double x)
int colorGreen(double x)



① Click On Normal, Speed or Accelerate Radio Button

2.9 Drag Mouse on the Chart



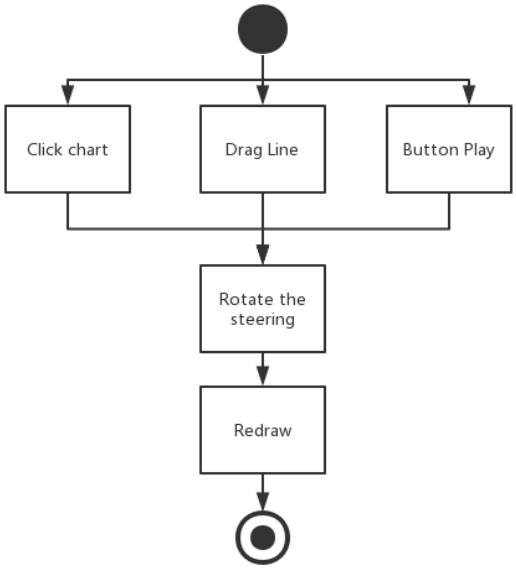
Users can drag the red line randomly in the chart. The chart will listen on "MouseDown" event to trigger "MouseMove" event and "MouseUp" event to end "MouseMove" event.

Related Functions:

void sensorChart_MouseMove(object sender, MouseEventArgs e)

void sensorChart_MouseDown(object sender, MouseEventArgs e)
void sensorChart_MouseUp(object sender, MouseEventArgs e)

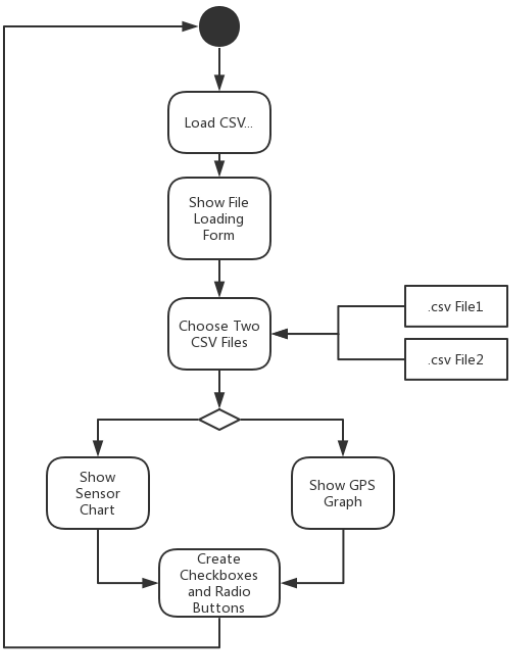
2.10 Show Steering Position



When user clicks on the chart, the steering wheel will be rotated a certain angle according to the CSV file. Similarly, if user clicks on Button Play, the angle of the steering wheel will be changed in real time.
Related Functions:
public static Image RotateImage(Image img, float rotationAngle);
private void sensorChart_MouseMove(object sender, MouseEventArgs e);
private void chartTimer_Tick(object sender, EventArgs e);
private void sensorChart_MouseClick(object sender, MouseEventArgs e);

3. Compared Run Form

3.1 Load CSV Files

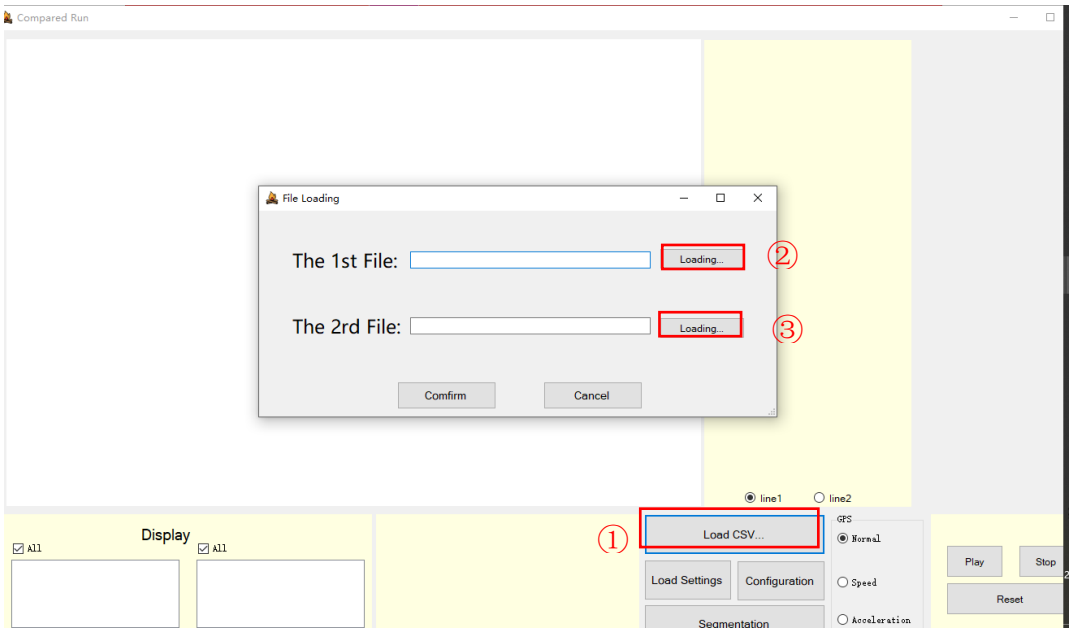


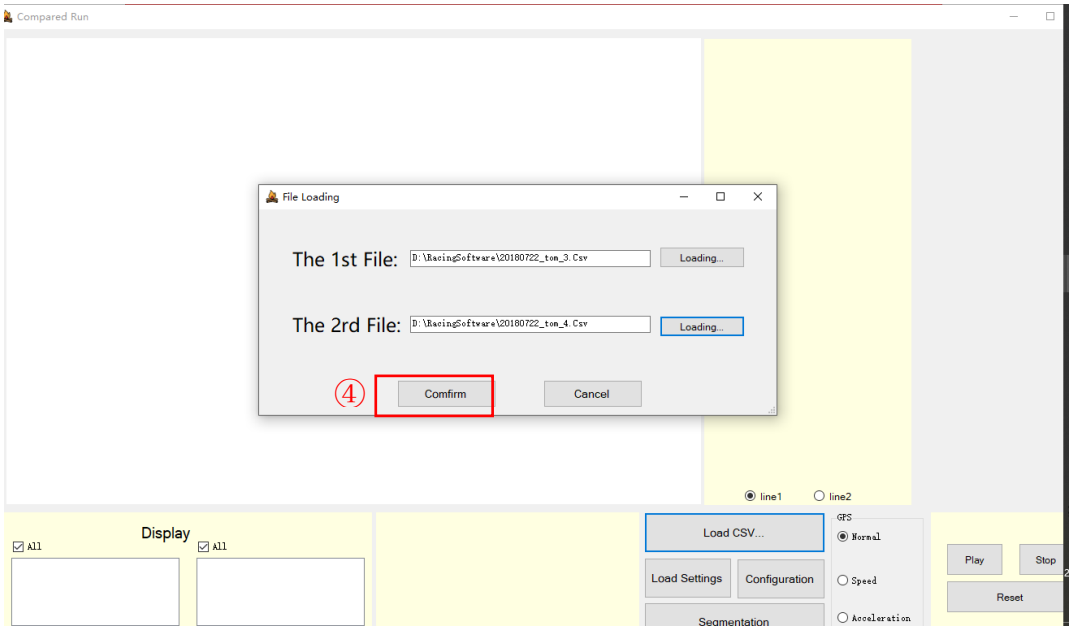
Call function of fileLoadingButton_Click and pop up a file dialog for users to choose their CSV files. After users choose their files and the columns for latitude and longitude, the software could draw graphs for different series of sensors and the GPS graph for two routes of driving. Also, the software would dynamic create checkboxes and radio buttons for different sensors read from the CSV file.

Related Functions:

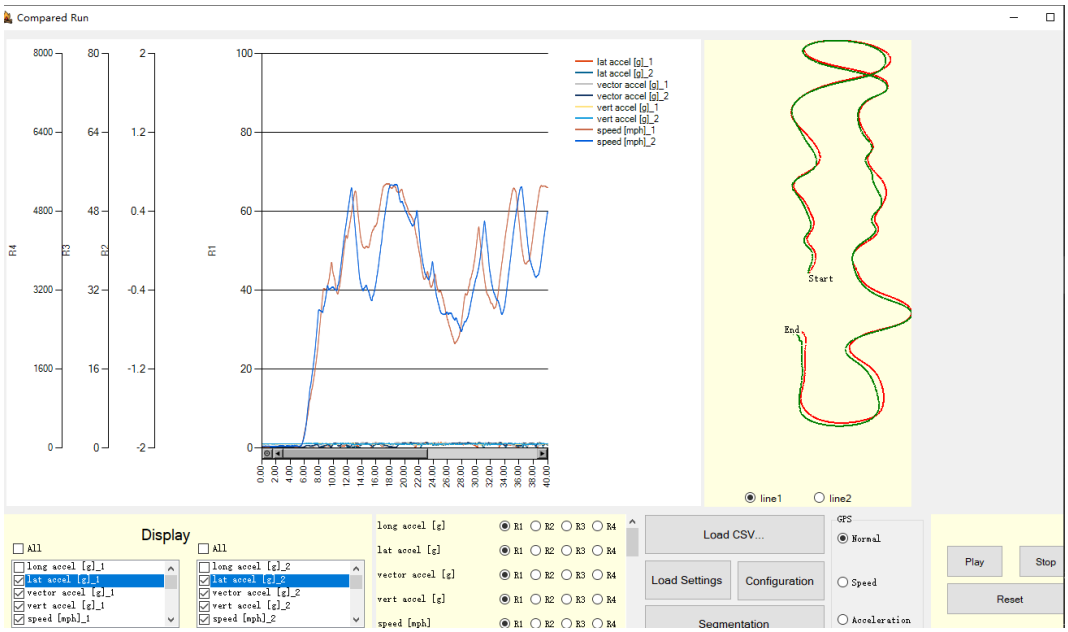
void fileLoadingButton_Click(object sender, EventArgs e)

DataTable OpenCSV(string filePath)

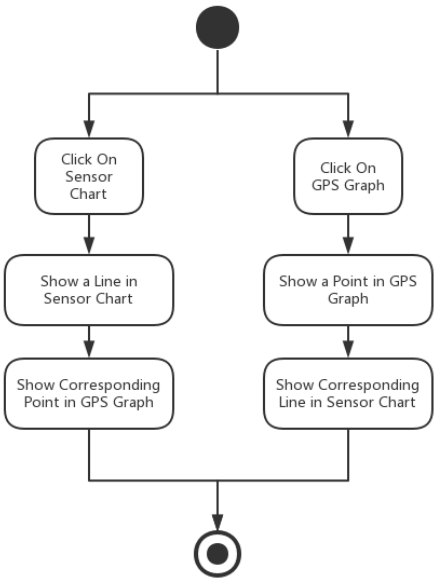




Now you can see the file is successfully readed into the software and it displays well.



3.2 Show Specific Value of Points



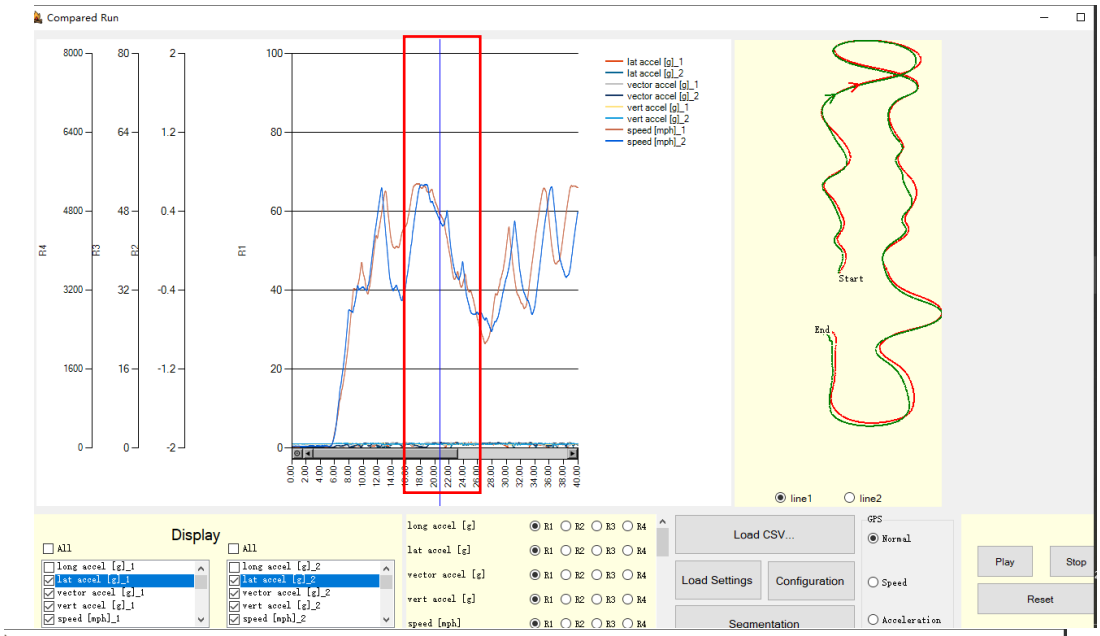
Users could click on the graph and show a vertical line at where the mouse clicked. Meanwhile, there would be a black point in the GPS graph to show the same place as you click on the chart. Also, users could click on the GPS graph and show the corresponding on the sensor chart.

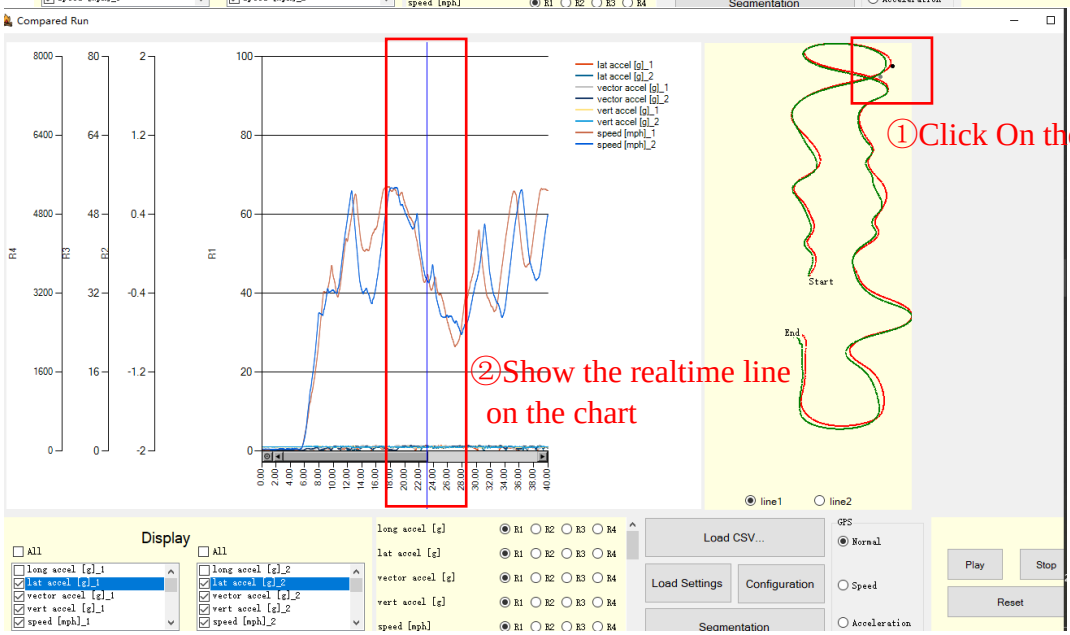
Related Functions:

void sensorChart_MouseClick(object sender, MouseEventArgs e)

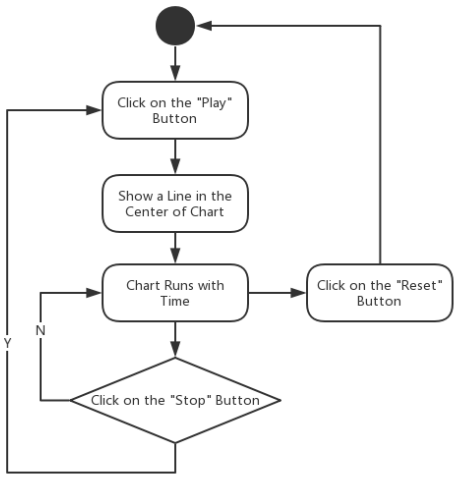
private void GPSPanel_MouseClick(object sender, MouseEventArgs e)

int findLeftNear(double value, double[] array, int length)


Compared Run
lat accel [g]_1
lat accel [g]_2
vector accel [g]_1
vector accel [g]_2
vert accel [g]_1
vert accel [g]_2
speed [mph]_1
speed [mph]_2
Start
End
line1
line2
Display
All
long accel [g]_1
lat accel [g]_1
vector accel [g]_1
vert accel [g]_1
speed [mph]_1
long accel [g]_2
lat accel [g]_2
vector accel [g]_2
vert accel [g]_2
speed [mph]_2
long accel [g]
lat accel [g]
vector accel [g]
vert accel [g]
speed [mph]
R1
R2
R3
R4
Load CSV...
Load Settings
Configuration
Segmentation
GPS
Normal
Speed
Acceleration
Play
Stop
Reset


Compared Run
①Click On the GPS Graph
②Show the realtime line on the chart
Start
End
line1
line2
Display
Load CSV...
Load Settings
Configuration
Segmentation
GPS
Normal
Speed
Acceleration
Play
Stop
Reset

3.3 Replay the Data

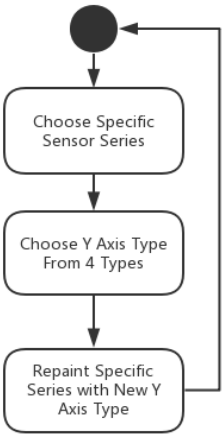


Users could click on “Play” button and call the function of buttonPlay_Click. This function enables chartTimer so that it could repeatedly execute the function of chartTimer_Tick at intervals. In the meantime, the textboxes below would show the values of sensors as the chart flowing. When users click on “Stop” button, the chart would stop flowing. After “Play” button is re-clicked, the chart would begin to flow from where it stops. The “Reset” button would let users to reset the chart to the original state. Meanwhile, there would be two arrows on the GPS graph, each of them represents the direction of car. They will also move with the same speed of sensor chart.

Related Functions:

```
void buttonPlay_Click(object sender, EventArgs e)
void buttonStop_Click(object sender, EventArgs e)
void resetButton_Click(object sender, EventArgs e)
void chartTimer_Tick(object sender, EventArgs e)
```

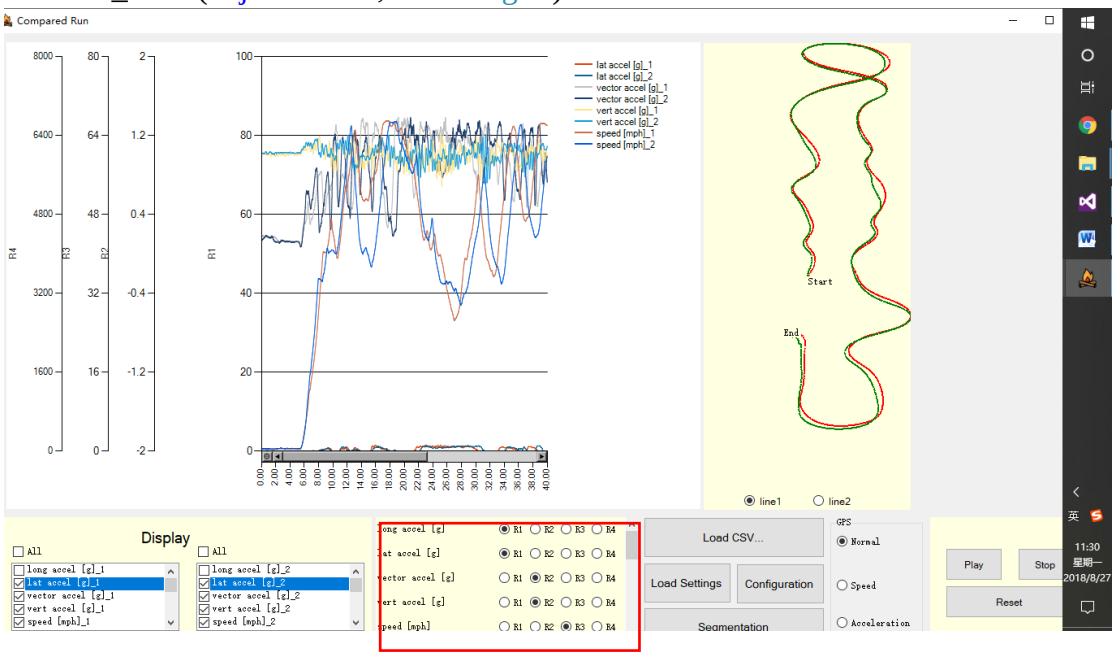
3.4 Choose Different Type of Y Axis



Users could change different series from one Y axis to another one to display the data in one chart more clearly. The software provides 4 different Y axes. We can switch between different axes by radio buttons dynamically created by fileLoadingButton_Click function. Both drivers' chart will change with different Y axis type.

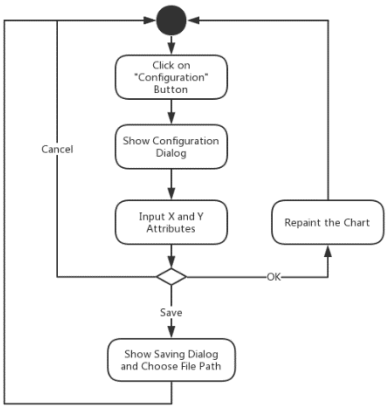
Related Functions:

void change(string no, ChartArea caR)
void change2(string no, ChartArea caR)
void rb1_Click(object sender, EventArgs e)
void rb2_Click(object sender, EventArgs e)
void rb3_Click(object sender, EventArgs e)
void rb4_Click(object sender, EventArgs e)

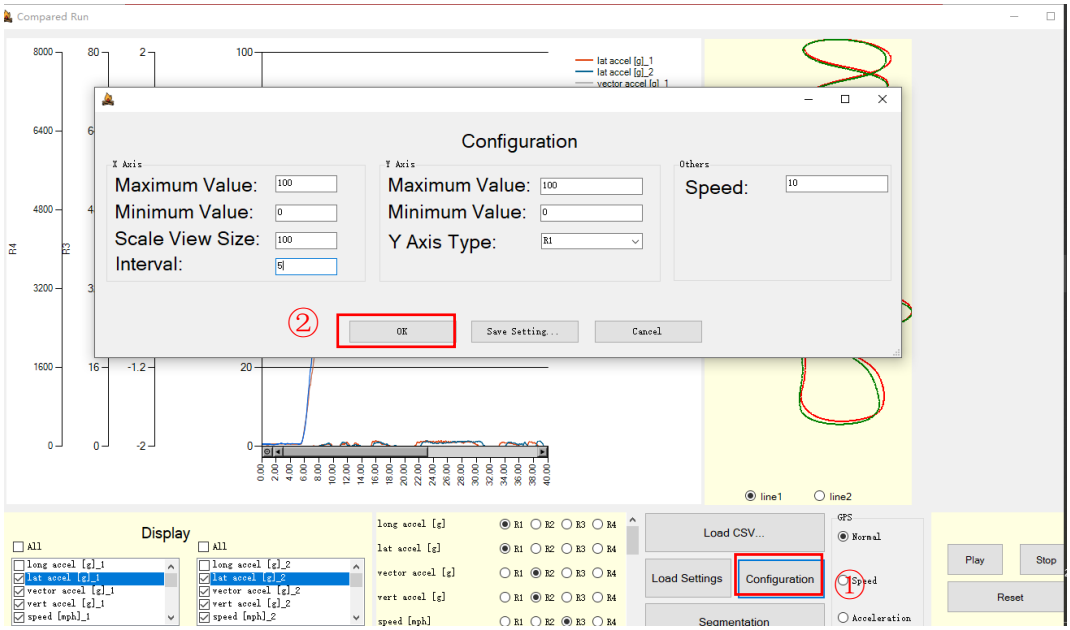


① Choose Different Y axis Types

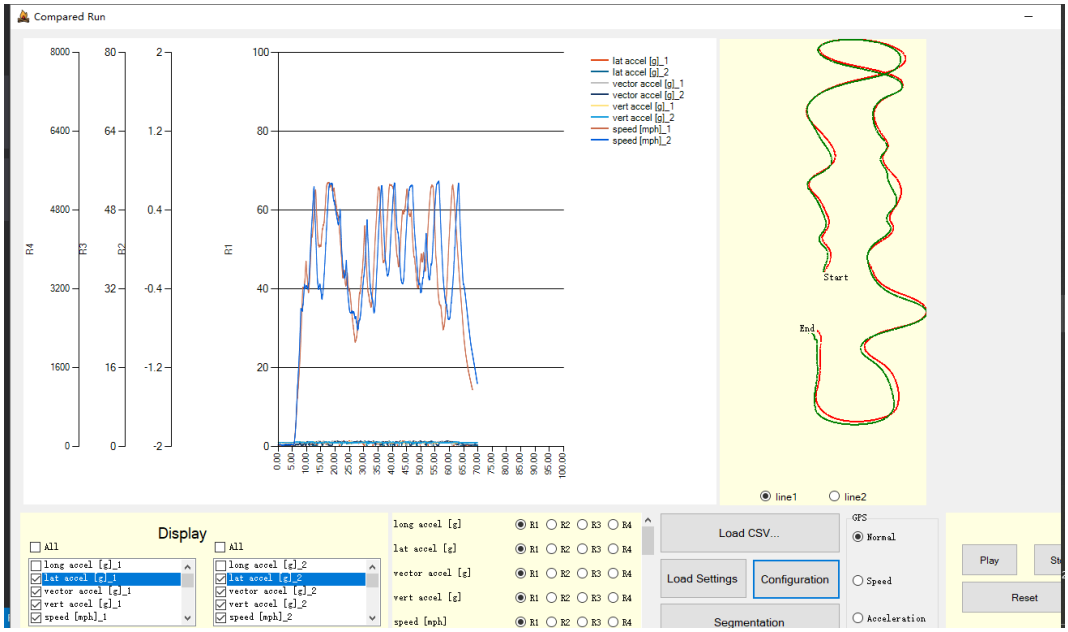
3.5 Customize X axis and Y axis



Users could click on “Configuration” button to pop out a dialog for X axis and Y axis customization. Users could change the range of X axis range, scale and interval and Y axis range and type.
Related Functions and Forms:
public partial class RangeForm : Form
void YRangeForm_Load(object sender, EventArgs e)
void confirmButton_Click(object sender, EventArgs e)
void ConfigureButton_Click(object sender, EventArgs e)



After change the max valve and scale view of X axis, we can see that the chart has changed as we expected.

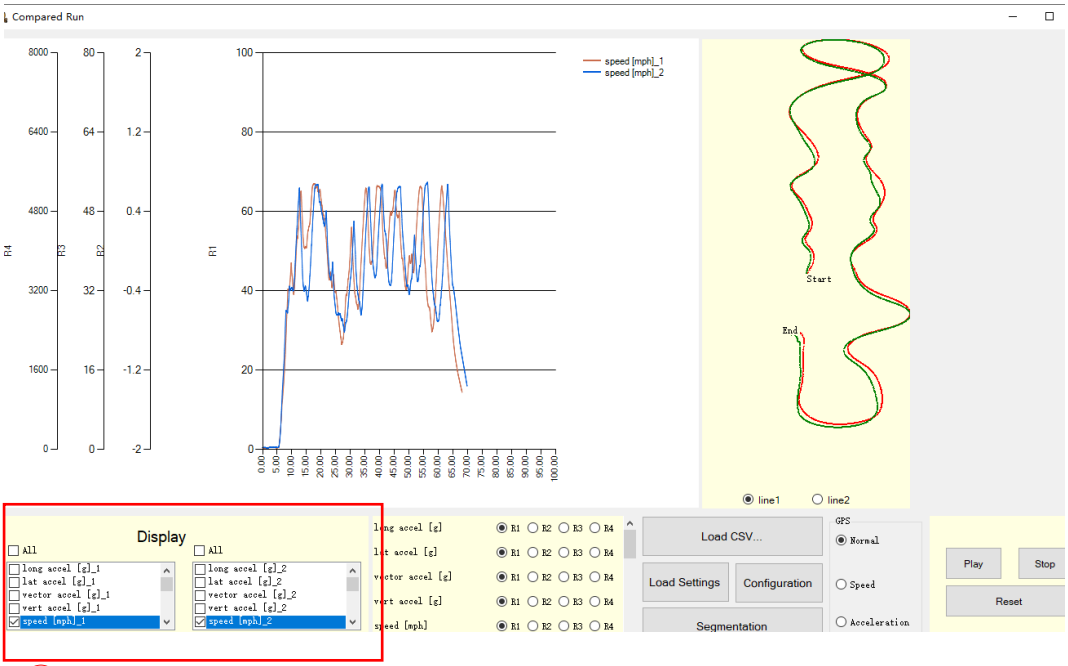


3.6 Save and Load Setting Log Files

This function is really similar to the single ride. See 2.6.

3.7 Show or Hide Specific Series

There are two checkboxeslists to show the data of two drivers. Users could click on different checkboxes created dynamically when reading the csv file. In addition, there is an allSelectedCheckBox for the selection of all.

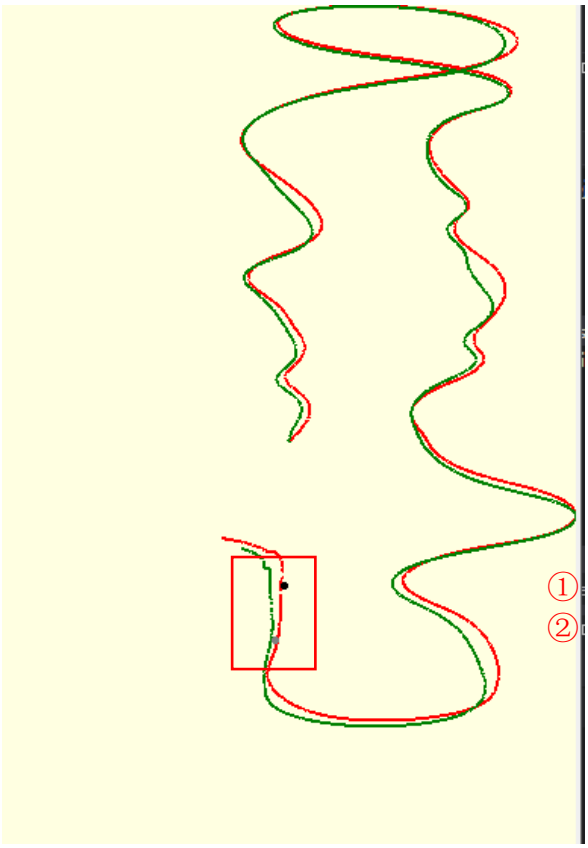


① Click On Series Needed to be Show

Related Functions:
void sensorCheckedListBox_ItemCheck(object sender, ItemCheckEventArgs e)
void allSelectedCheckBox_CheckedChanged(object sender, EventArgs e)

3.8 Click on the GPS map

When we click on one road, it can show where another driver is. With this, we can easily compare the performance of the two drivers.



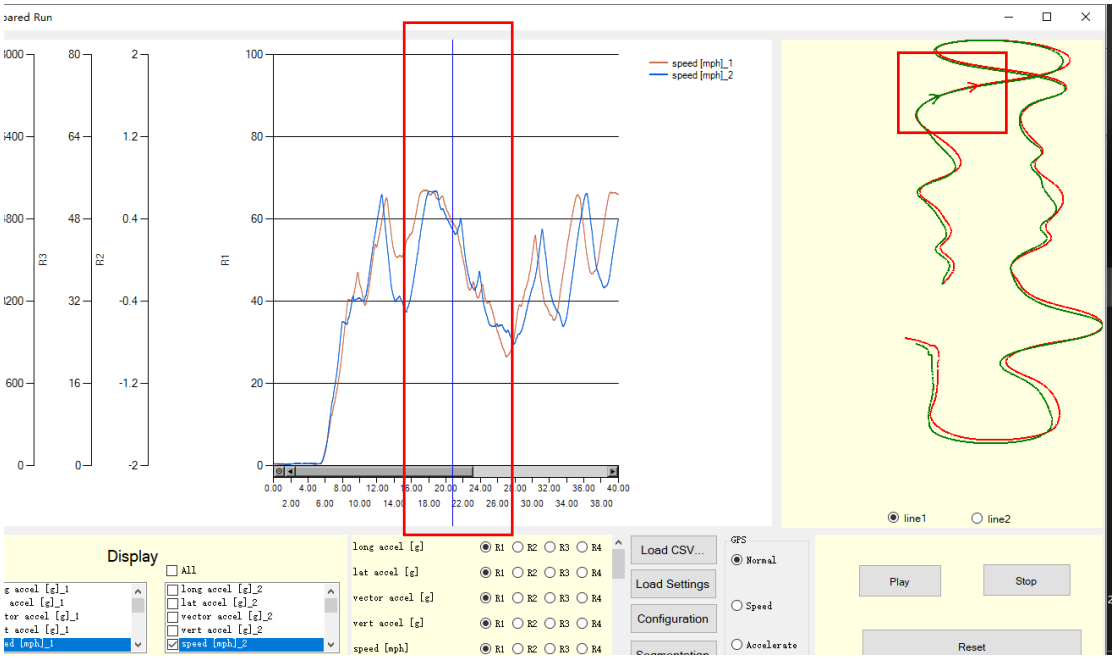
Related Functions:

```
private void GPSPanel_MouseClick(object sender, MouseEventArgs e)
```

3.9 Click/Drag Mouse on the Chart

-Users can drag the line randomly in the chart. The chart will listen on “MouseDown” event to trigger “MouseMove” event and “MouseUp” event to end “MouseMove” event.
- Users can click on the chart. It can show the locations & heading position of the two drivers.

①Click on the chart ②Show the locations



Related Functions:

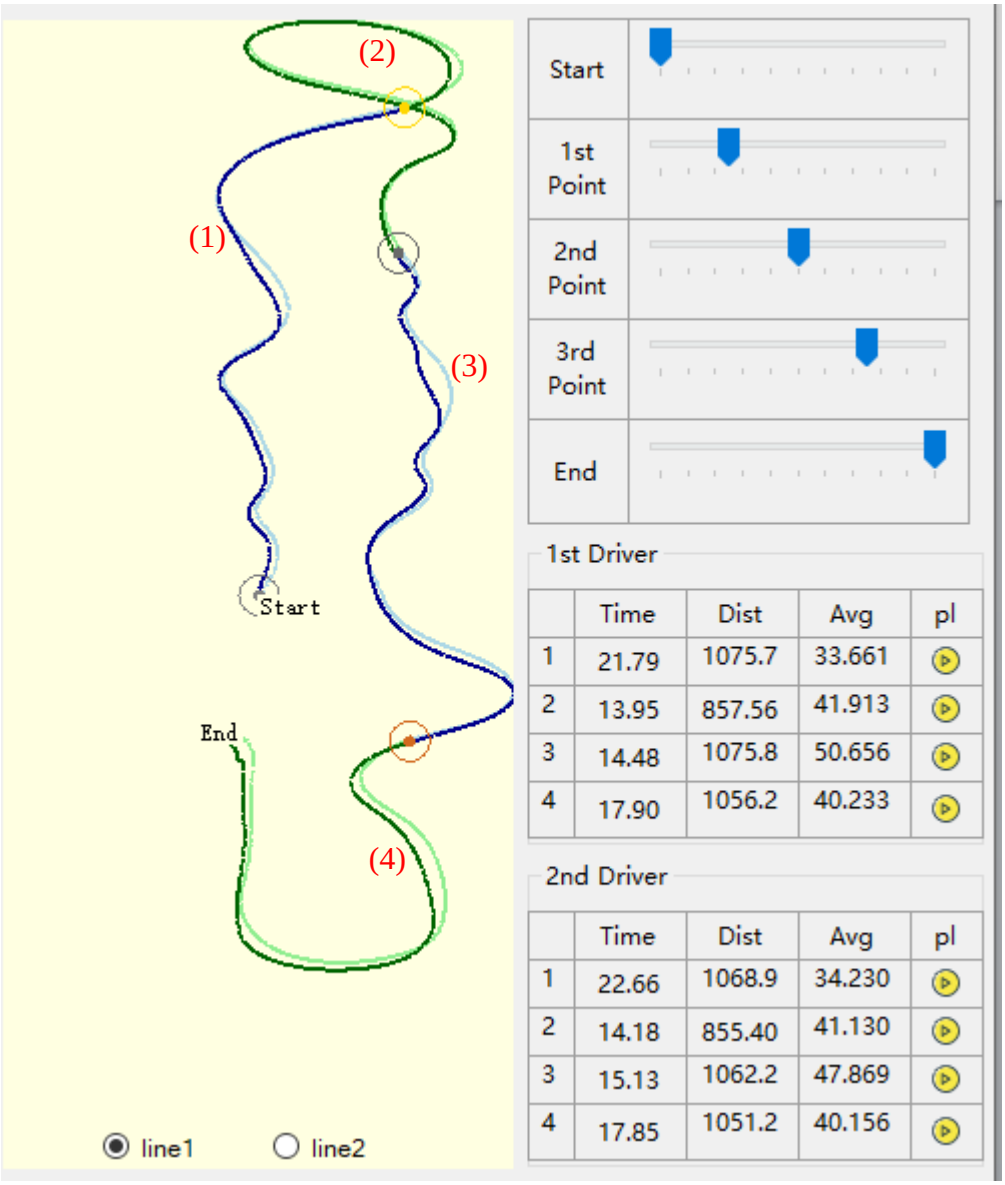
void sensorChart_MouseMove(object sender, MouseEventArgs e)

```
private void sensorChart_MouseClick(object sender, MouseEventArgs e)
private void sensorChart_MouseDown(object sender, MouseEventArgs e)
private void sensorChart_MouseUp(object sender, MouseEventArgs e)
```

3.10 Segmentation

When we click on “Segmentation” button, the segmentation part will come out, which shows the time, distance and average velocity it takes to complete every section for each driver. Also you can change the segmentation by sliding scroll bar. So we can easily compare the performance of the two drivers effectively. Besides, users can change the start point and the end point because there may be many useless data in the beginning and ending.

On the map, it can show the four segmentations. The lighter color represents the first driver and the darker color represents another driver. On the scroll bar, the default segmentation is 25%, 50%, 75%.



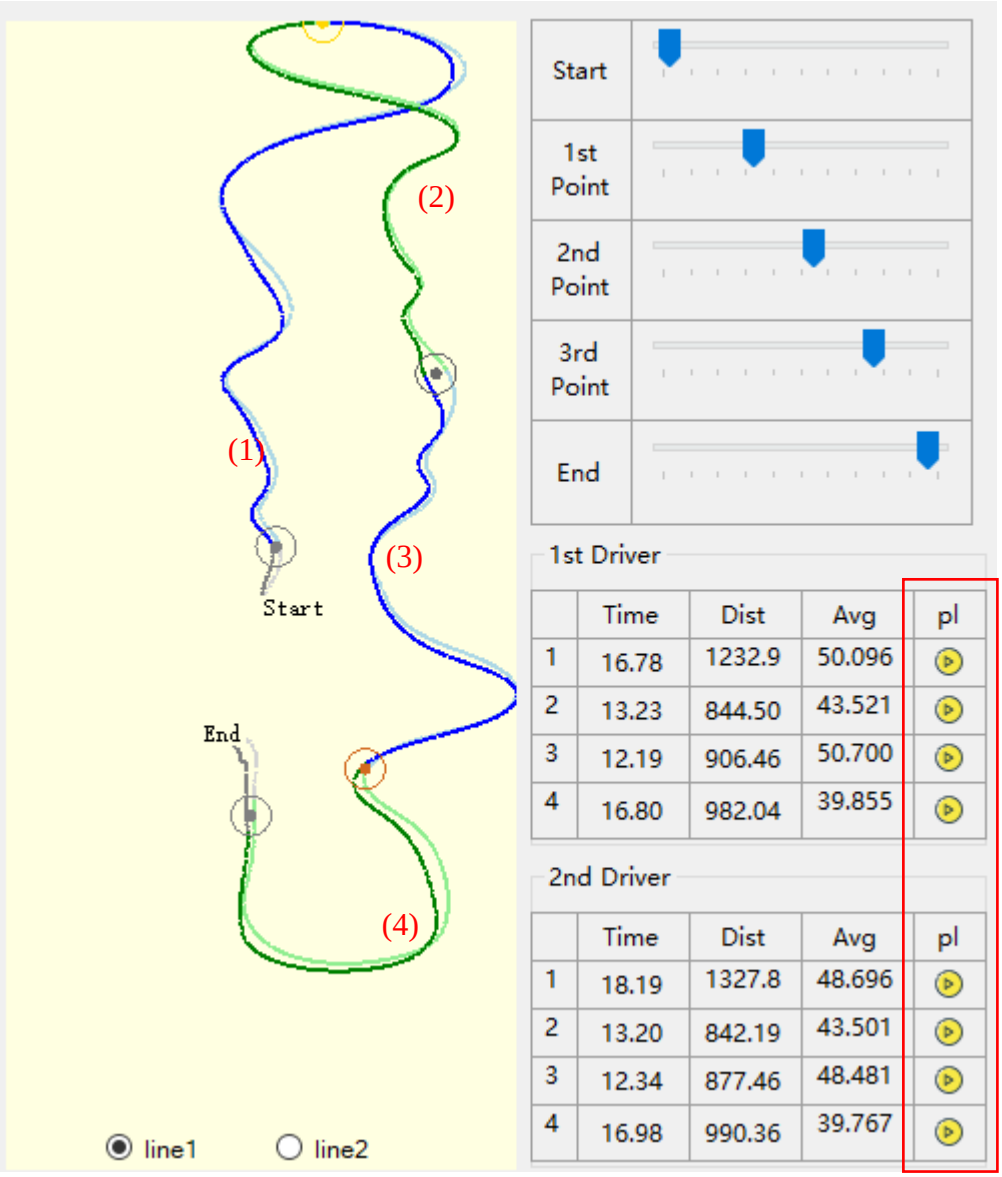
1st Driver

	Time	Dist	Avg	pl
1	21.79	1075.7	33.661	
2	13.95	857.56	41.913	
3	14.48	1075.8	50.656	
4	17.90	1056.2	40.233	

2nd Driver

	Time	Dist	Avg	pl
1	22.66	1068.9	34.230	
2	14.18	855.40	41.130	
3	15.13	1062.2	47.869	
4	17.85	1051.2	40.156	

After sliding the scroll bar, it will be like this:



Users can play every section so that users can see the movements of each driver in each section:
You can check it out on the youtube: https://youtu.be/af4zLfwI-Bc

Related Functions:

```
private void thirdTrackBar_ValueChanged(object sender, EventArgs e)
private void secondTrackBar_ValueChanged(object sender, EventArgs e)
private void firstTrackBar_ValueChanged(object sender, EventArgs e)
private void EndTrackBar_ValueChanged(object sender, EventArgs e)
private void StartTrackBar_ValueChanged(object sender, EventArgs e)
private void segmentationButton_Click(object sender, EventArgs e)
private void Section1PictureBox_Click(object sender, EventArgs e)
private void Section2PictureBox_Click(object sender, EventArgs e)
private void Section3PictureBox_Click(object sender, EventArgs e)
```

```
private void Section4PictureBox_Click(object sender, EventArgs e)
private void section1Timer_Tick(object sender, EventArgs e)
private void section2Timer_Tick(object sender, EventArgs e)
private void section3Timer_Tick(object sender, EventArgs e)
private void section4Timer_Tick(object sender, EventArgs e)
```

4. Testing Environment

- Visual Studio 2013&2015
- Programming Language: C#