**Driver-Assistant Software Instruction**

Jingwen Mo

Yiming Chen

Supervised by Prof. Quoc-Viet Dang

**July 23th, 2018**

# Introduction

## Purpose

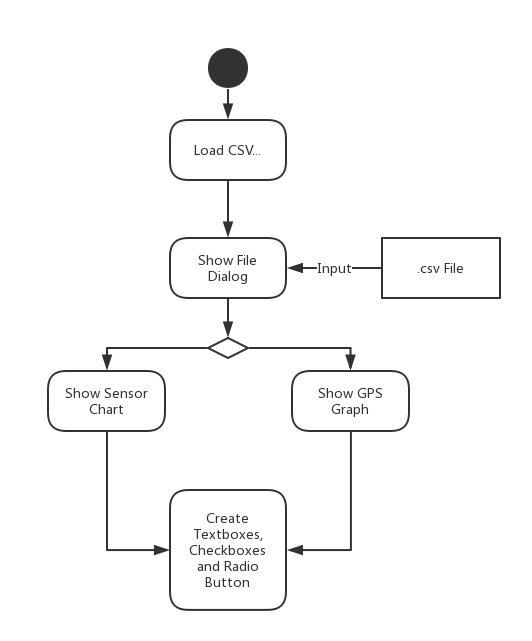
The Data-Graph Software is a PC based software developed to draw the graph which depicts different sensors for the racing car and show the GPS information. This instruction is for the software users and software tester.

## 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.

# Single Run Form

## 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)

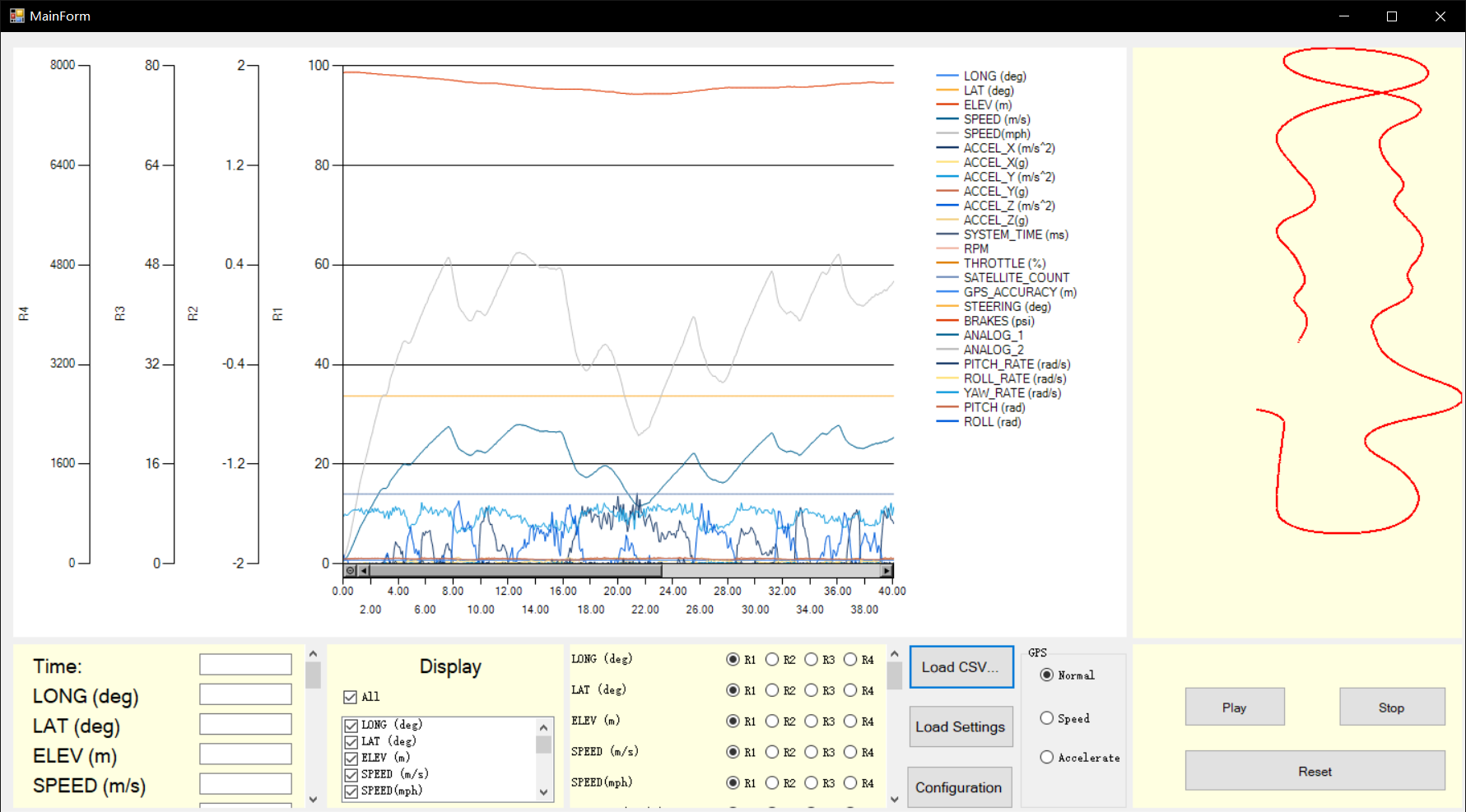


②

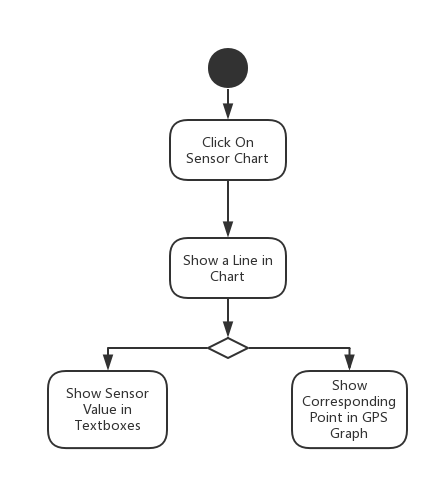
①



③



## Show Specific Value of Points

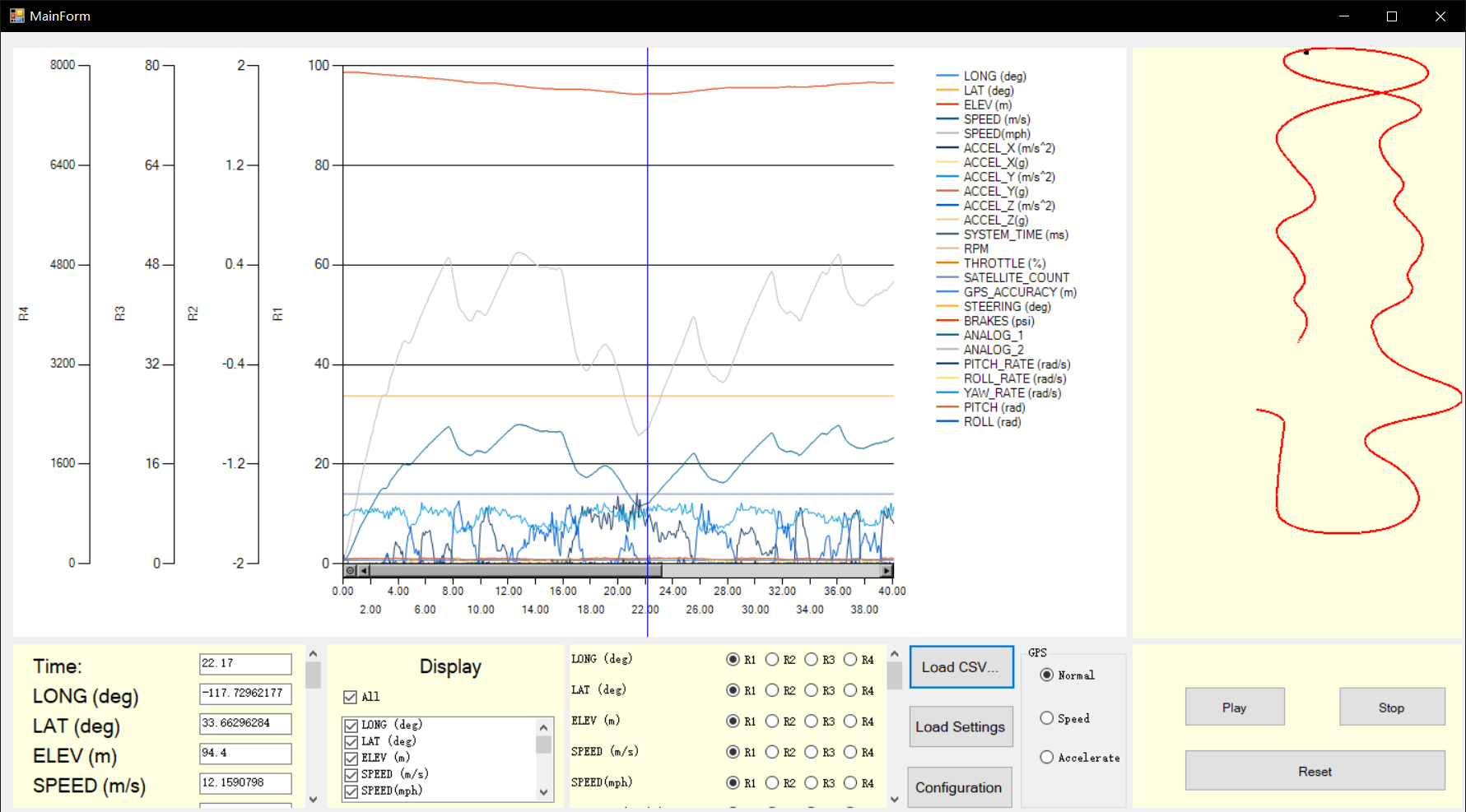


Users could click on the graph and show a vertical line at where the mouse clicked. Then the textboxes below would show the values of different sensors. Meanwhile, there would be a black point in the GPS graph to show the same place as you click on the chart.

Related Functions:

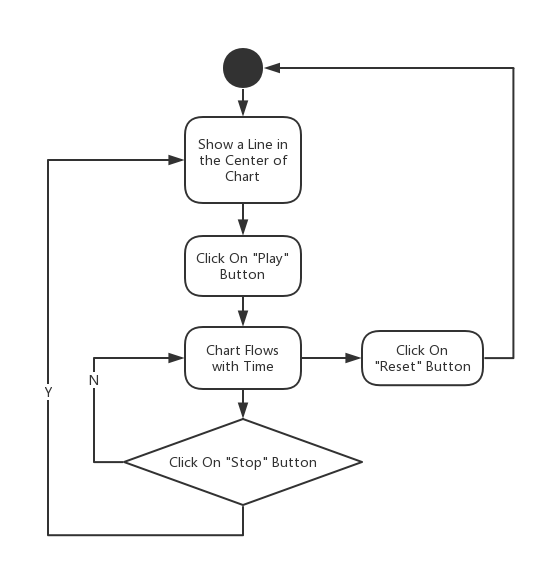
void sensorChart\_MouseClick(object sender, MouseEventArgs e)

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



①Click On the Chart

## 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)

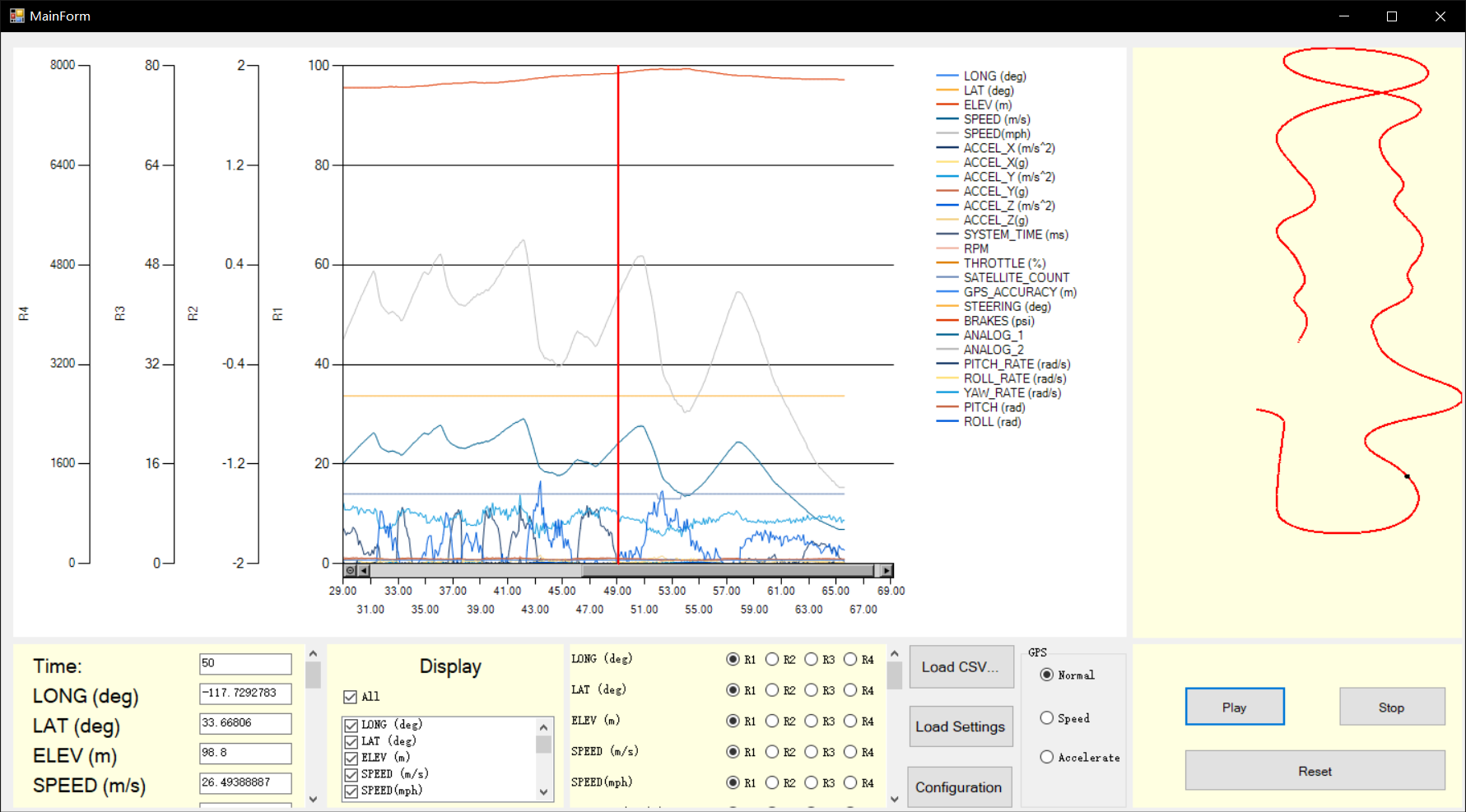
void buttonStop\_Click(object sender, EventArgs e)

void resetButton\_Click(object sender, EventArgs e)

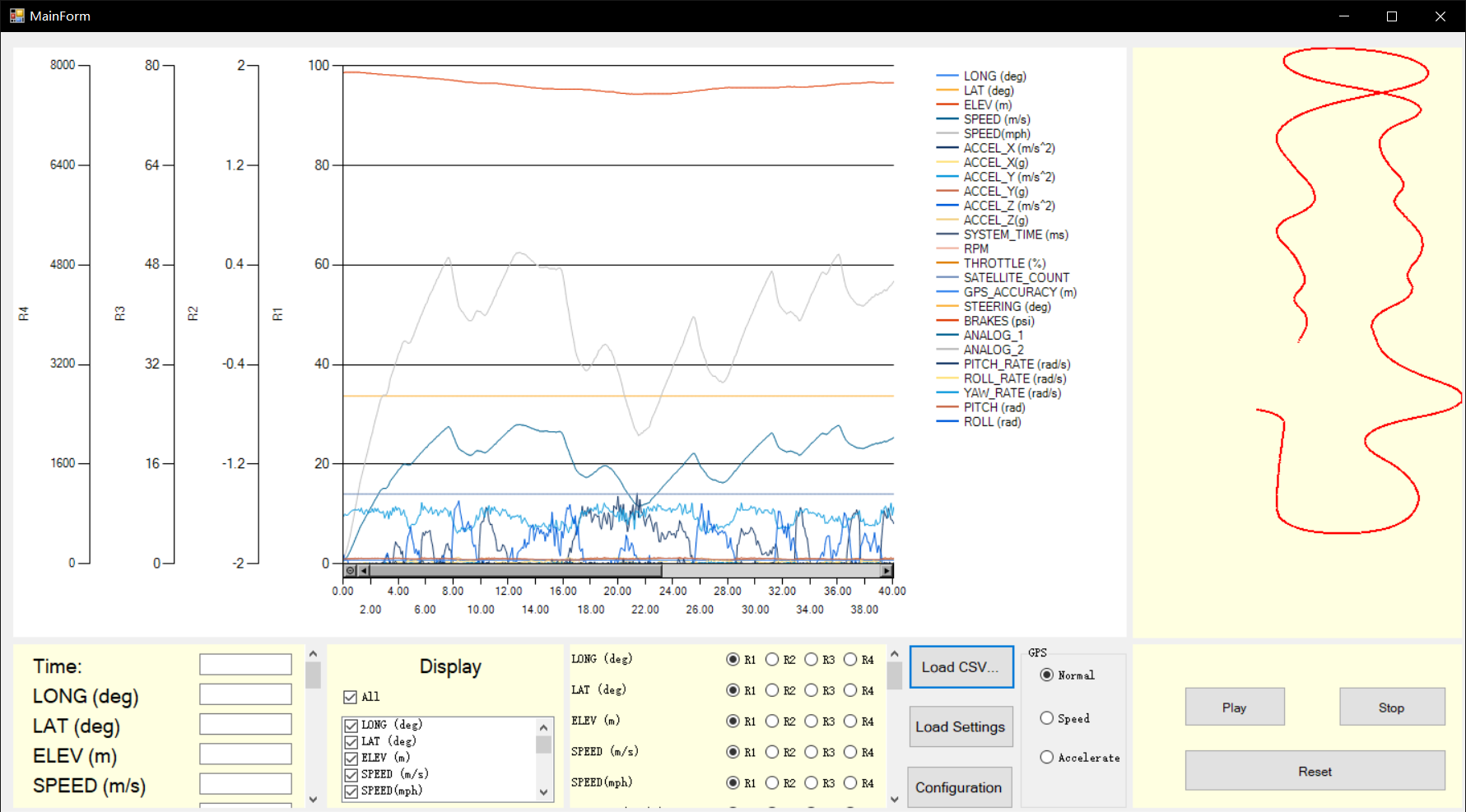
void chartTimer\_Tick(object sender, EventArgs e)



①



②



③

## 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)

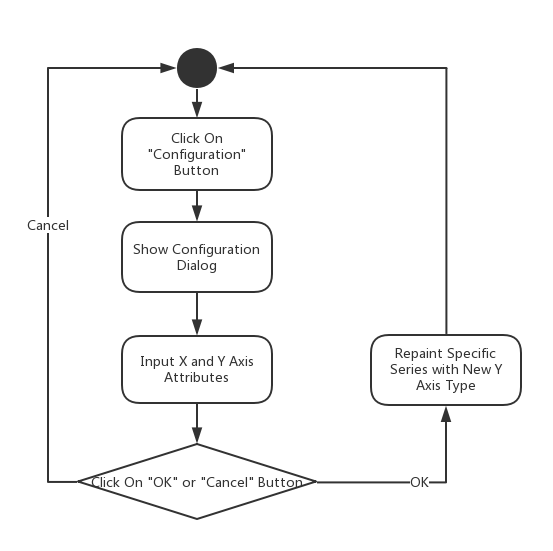
void rb3\_Click(object sender, EventArgs e)

void rb4\_Click(object sender, EventArgs e)



①Choose Different Y Types

## 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.

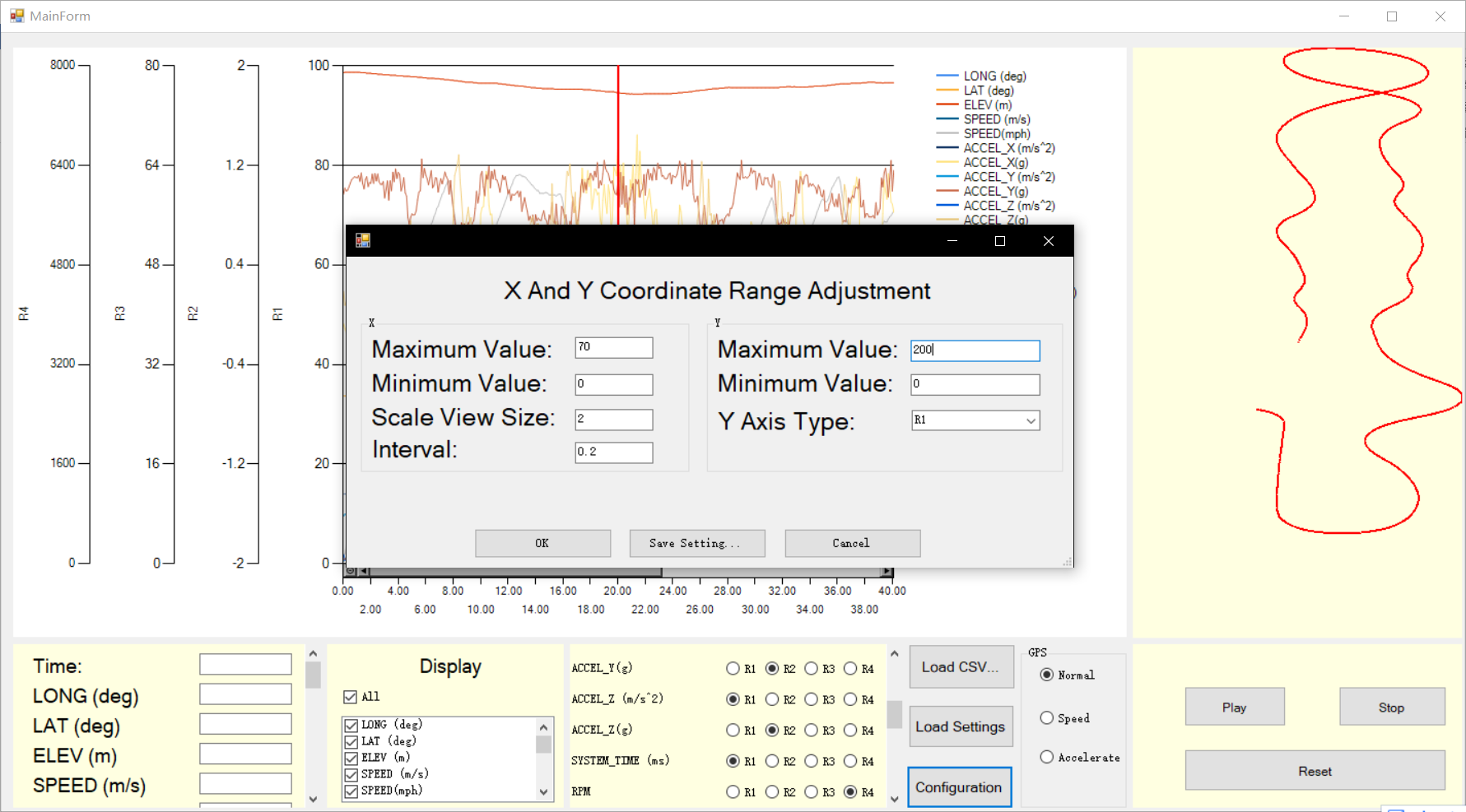
Related Functions and Forms:

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)

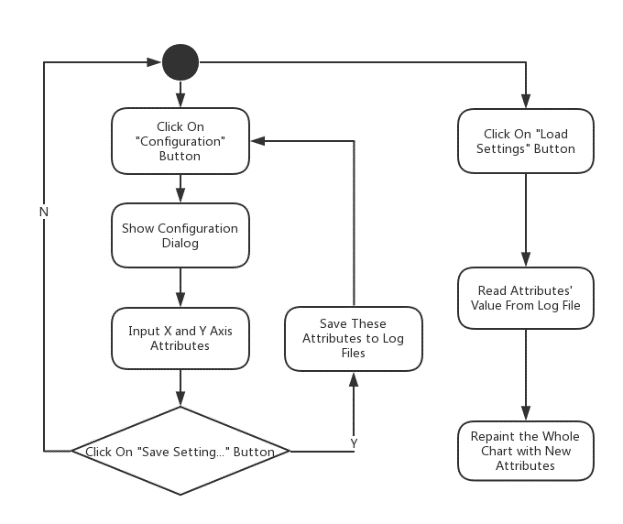


②

①



## Save and Load Setting Log Files

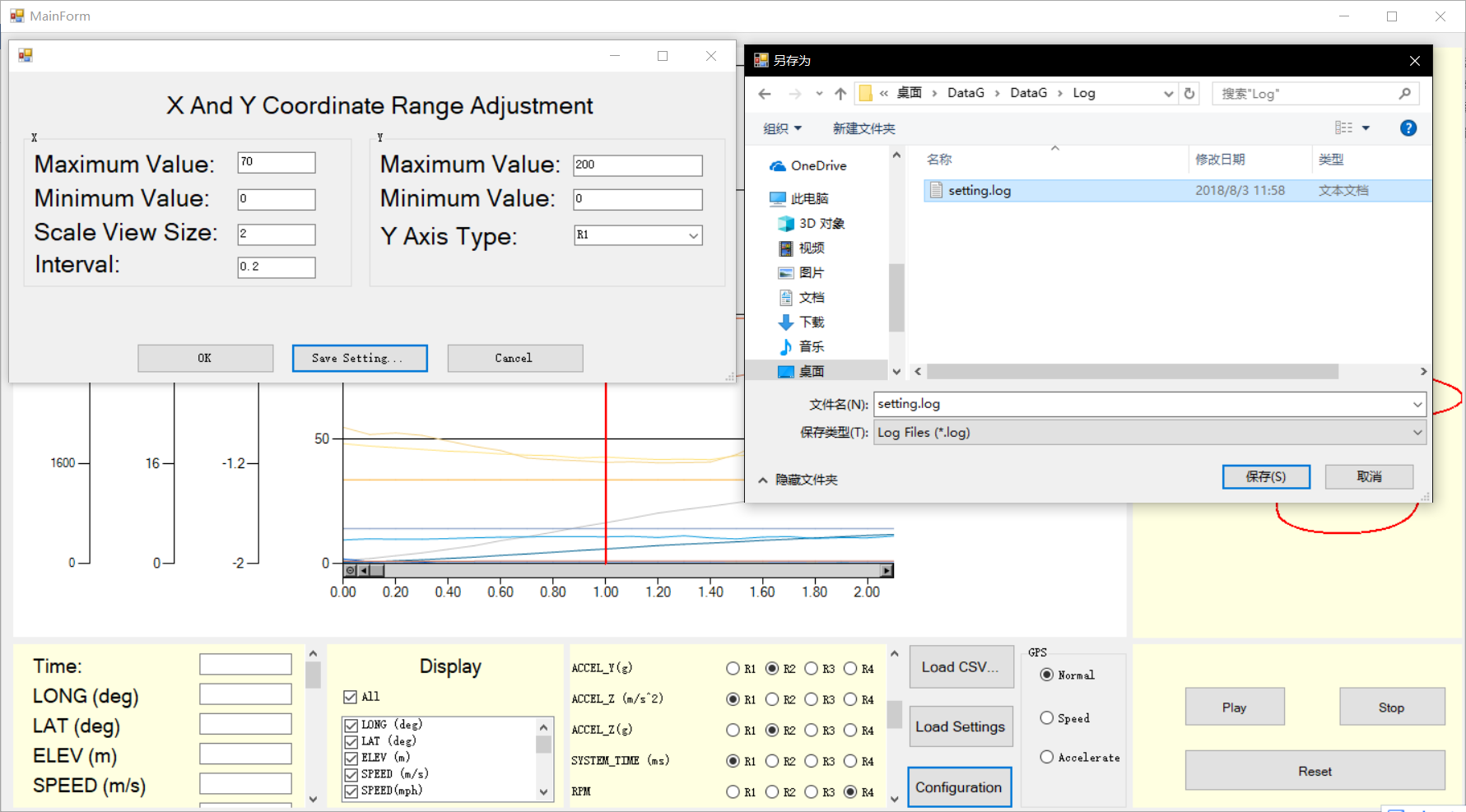


By clicking button of “Saving Setting…” in RangeForm, users could save log files with the X axis and Y axis configuration in specific form. Also, users could load log files created by them by clicking the button of “Loading Settings”.

Related Functions:

void settingSaveButton\_Click(object sender, EventArgs e)

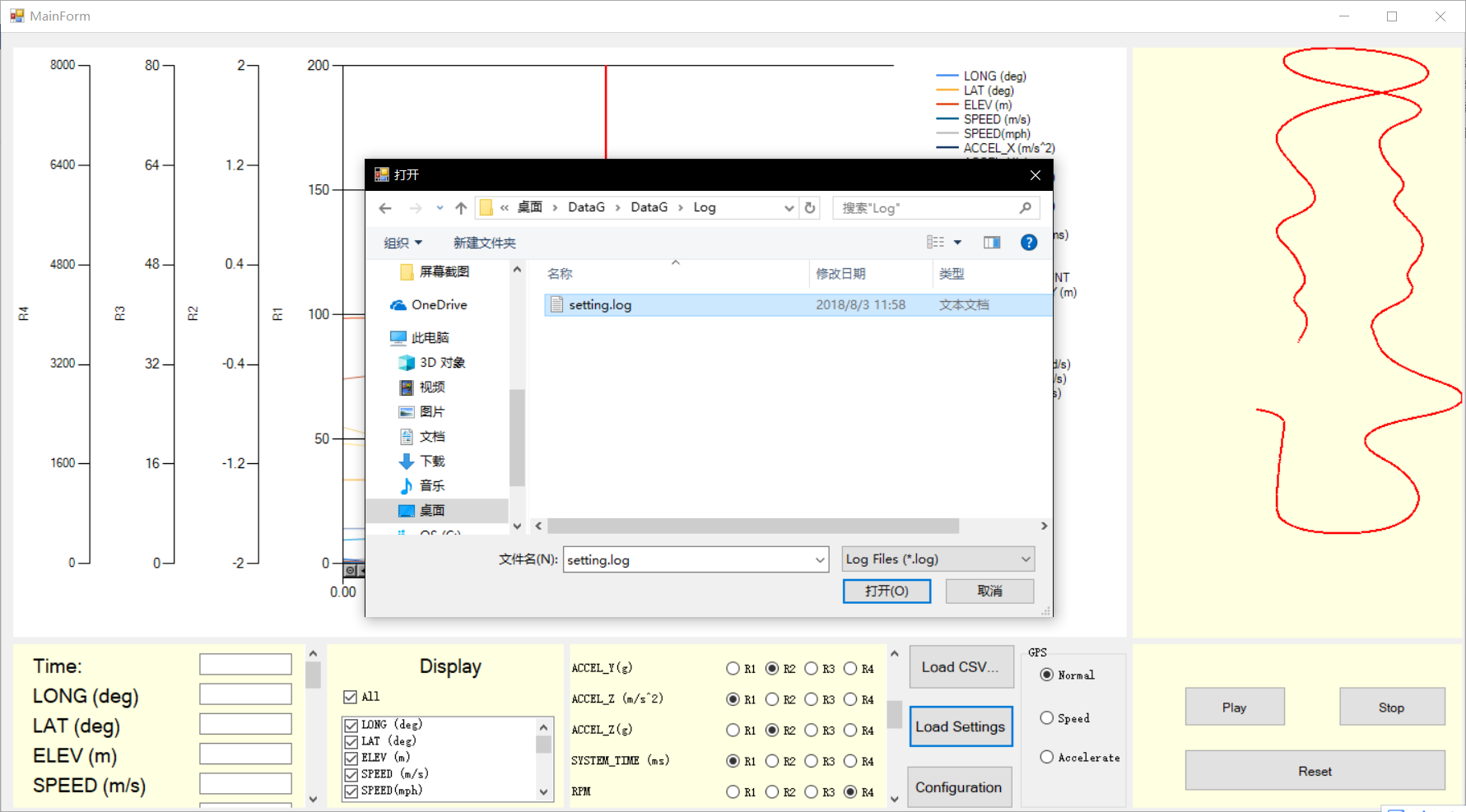
void settingButton\_Click(object sender, EventArgs e)



③

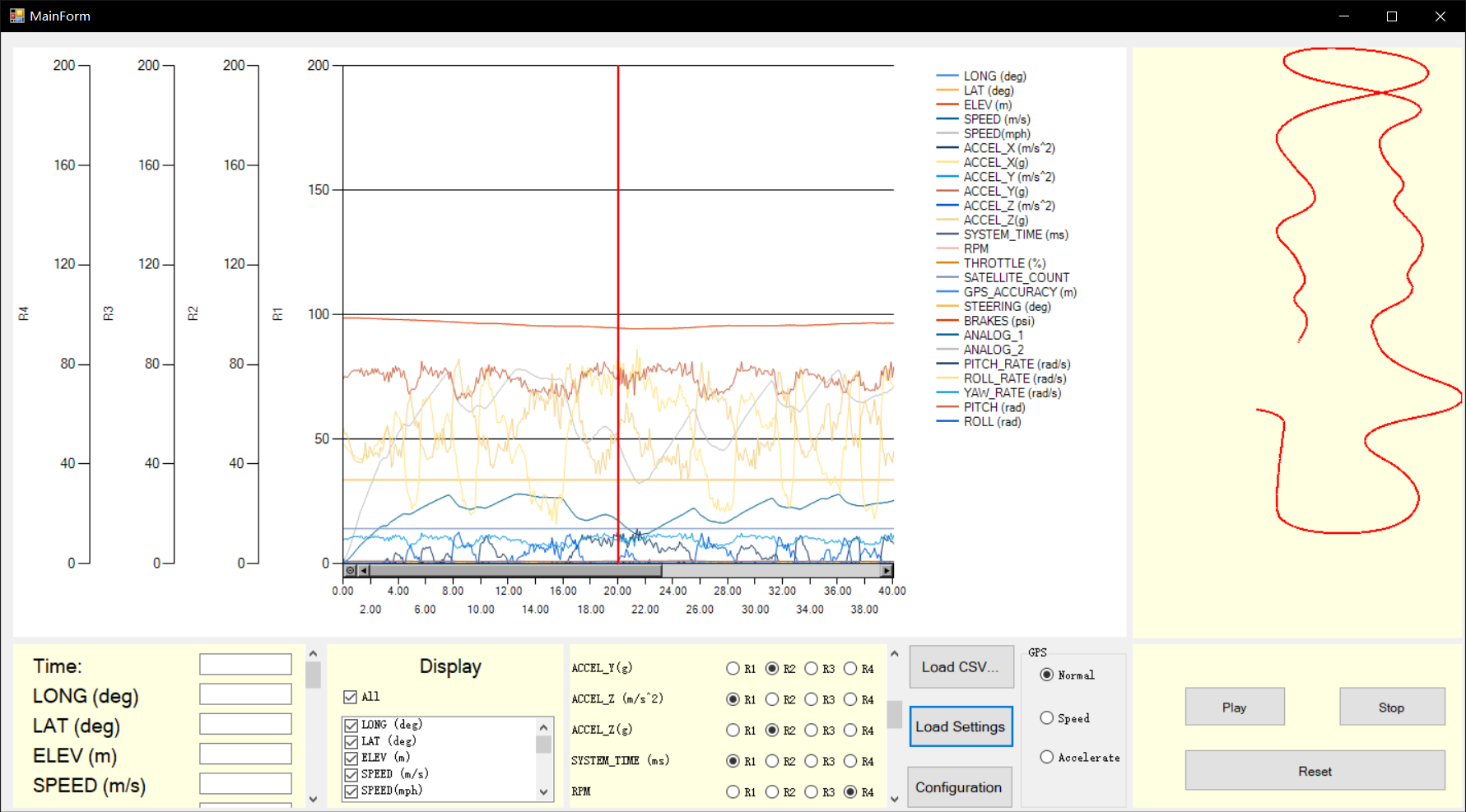
②

①

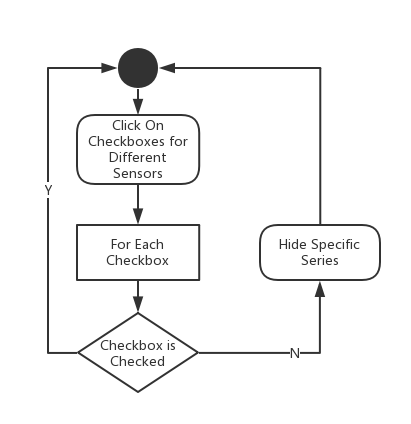


②

①



## Show or Hide Specific Series

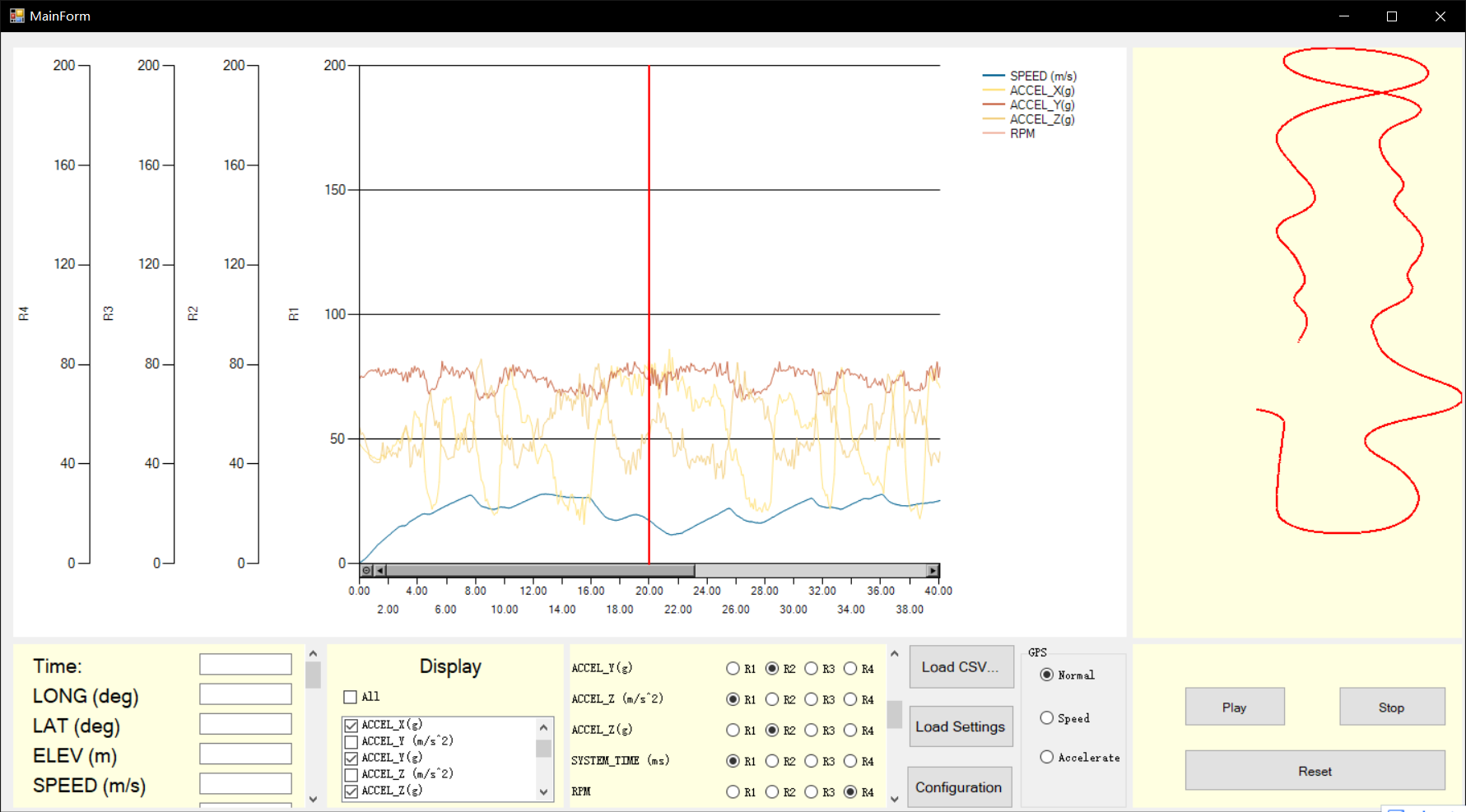


Users could click on different checkboxes created dynamically by fileLoadingButton\_Click function. In addition, there is an allSelectedCheckBox for the selection of all.

Related Functions:

void sensorCheckedListBox\_ItemCheck(object sender, ItemCheckEventArgs e)

void allSelectedCheckBox\_CheckedChanged(object sender, EventArgs e)



①Click On Series Needed to be Show

## 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:

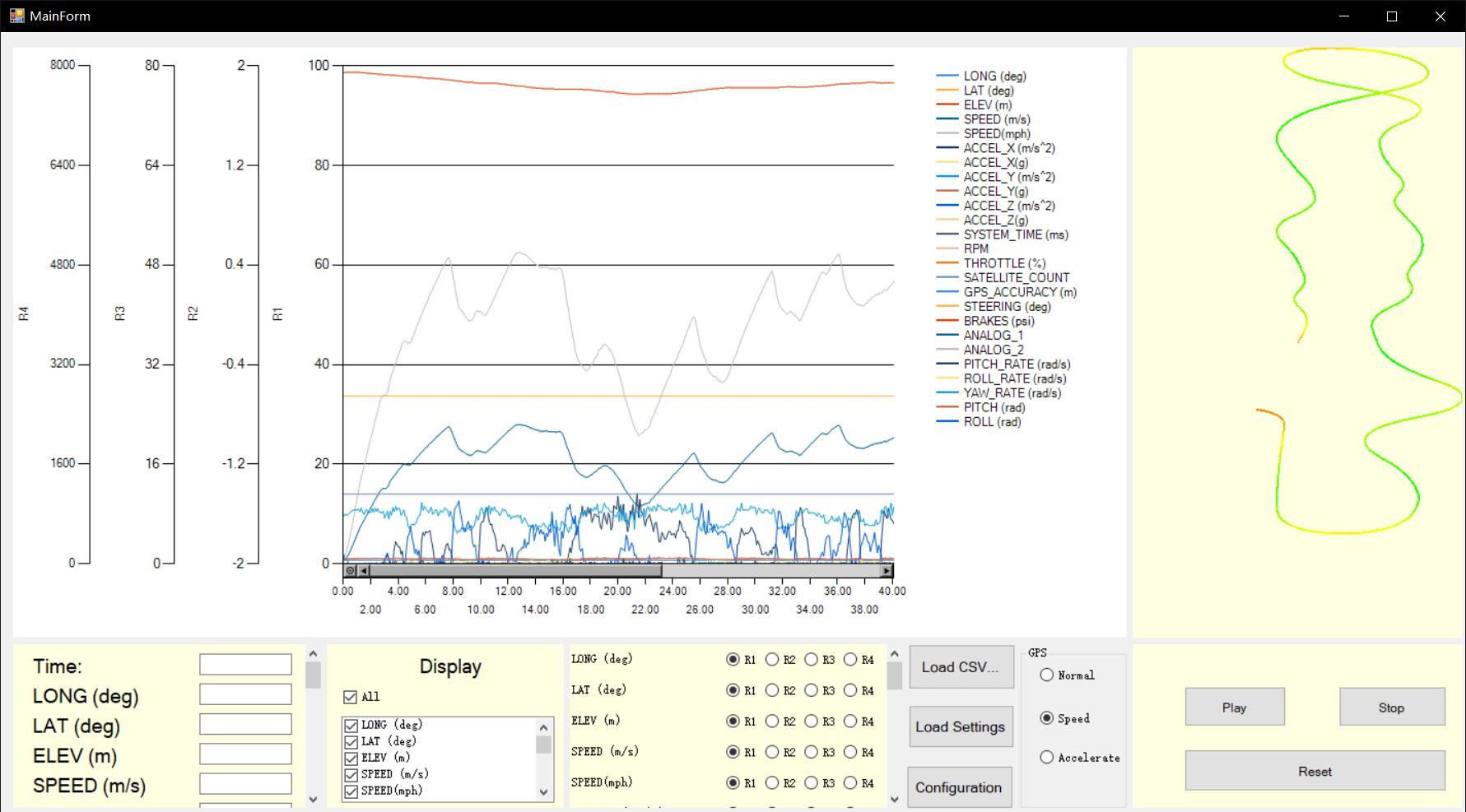
void radioButton\_Normal\_CheckedChanged(object sender, EventArgs e)

void radioButton\_Speed\_CheckedChanged(object sender, EventArgs e)

void radioButton\_Accelerate\_CheckedChanged(object sender, EventArgs e)

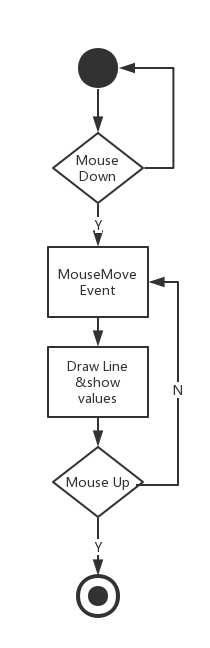
int colorRed(double x)

int colorGreen(double x)



## Drag Mouse on the Chart

①Click On Normal, Speed or Accelerate Radio Button



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.

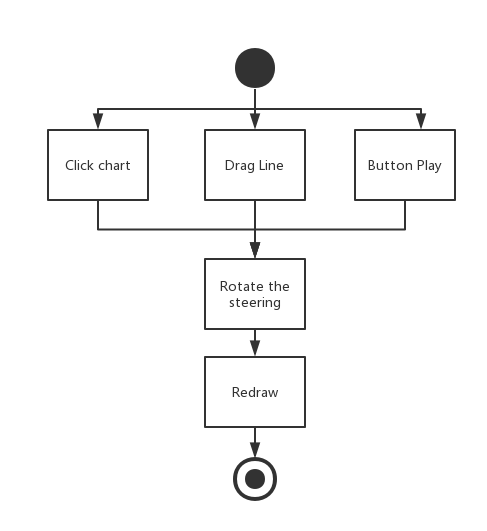
Related Functions:

void sensorChart\_MouseMove(object sender, MouseEventArgs e)

void sensorChart\_MouseDown(object sender, MouseEventArgs e)

void sensorChart\_MouseUp(object sender, MouseEventArgs e)

## 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);

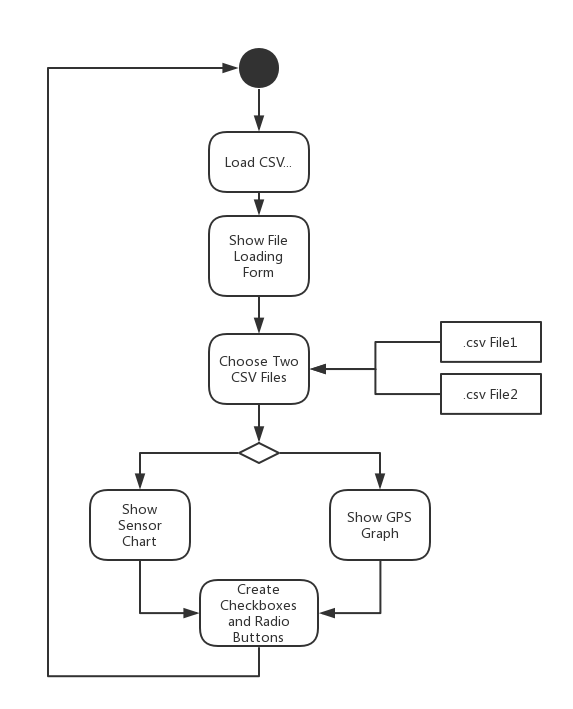
private void sensorChart\_MouseMove(object sender, MouseEventArgs e);

private void chartTimer\_Tick(object sender, EventArgs e);

private void sensorChart\_MouseClick(object sender, MouseEventArgs e);

# Compared Run Form

## Load CSV Files

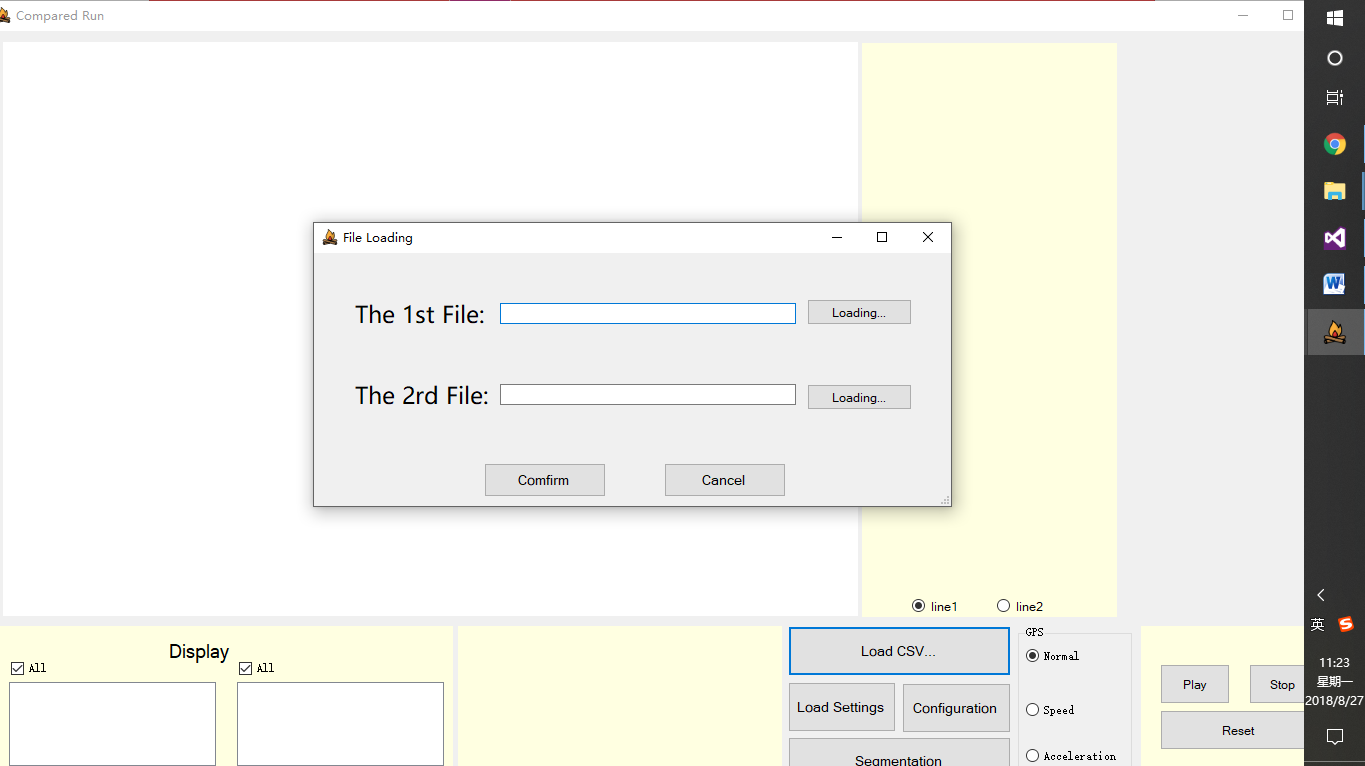


Call function of fileLoadingButton\_Click and pop up a file dialog for users to choose their CSV files. After users choose their files and the columns for latitude and longitude, the software could draw graphs for different series of sensors and the GPS graph for two routes of driving. Also, the software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

Related Functions:

void fileLoadingButton\_Click(object sender, EventArgs e)

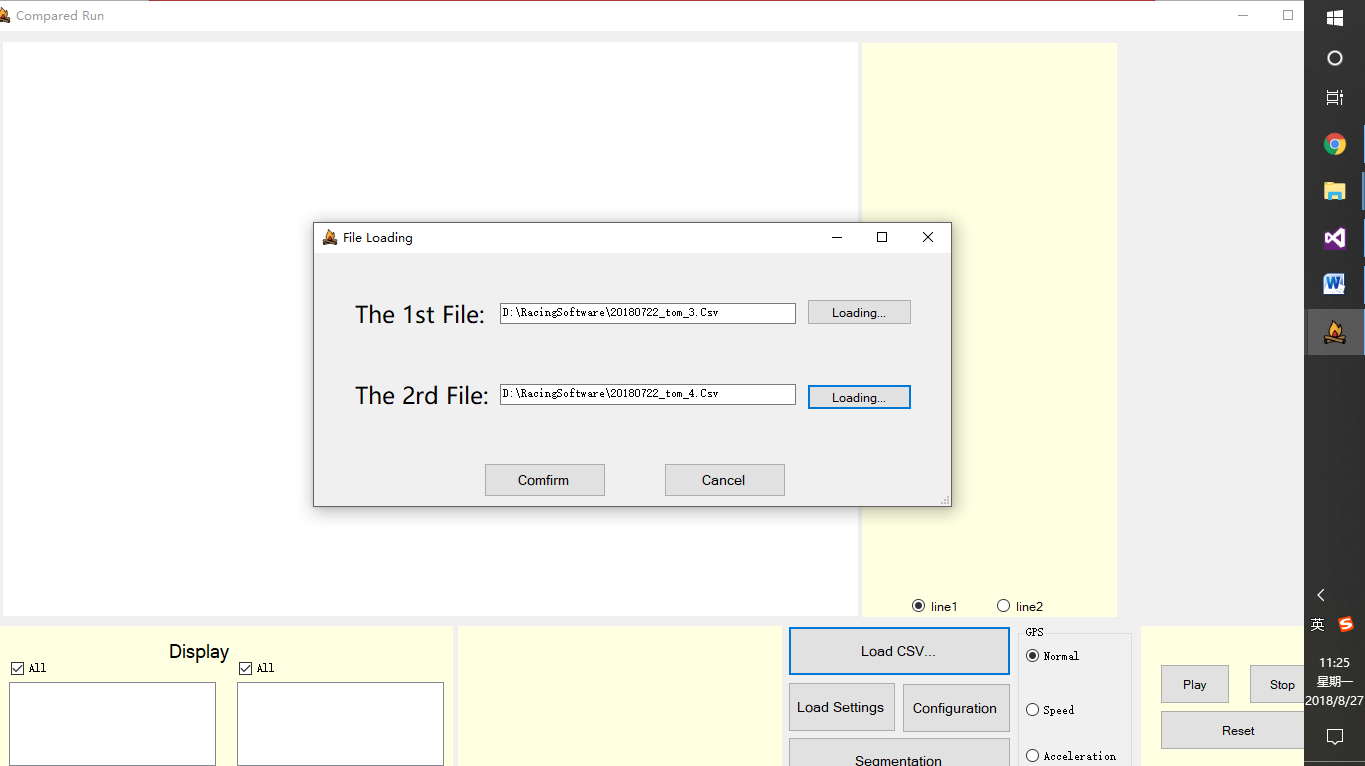
DataTable OpenCSV(string filePath)



③

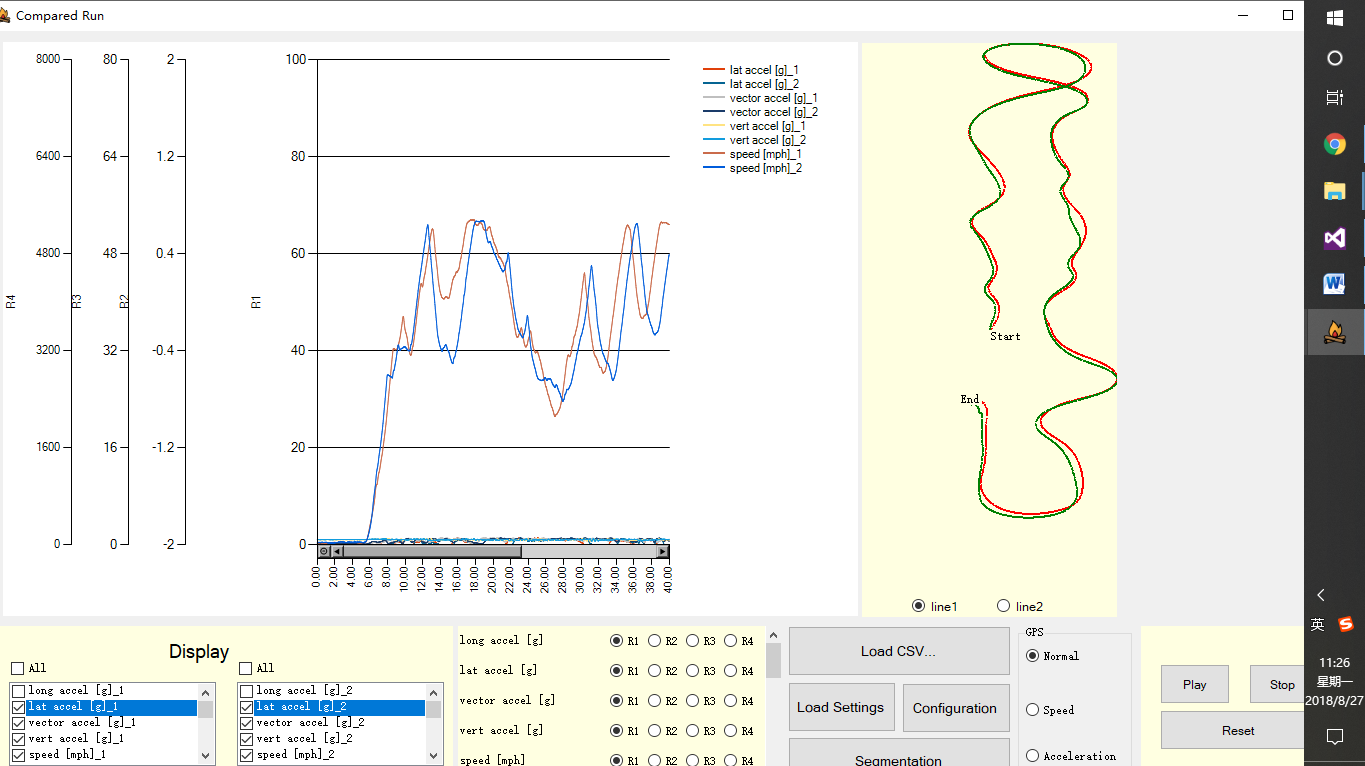
②

①

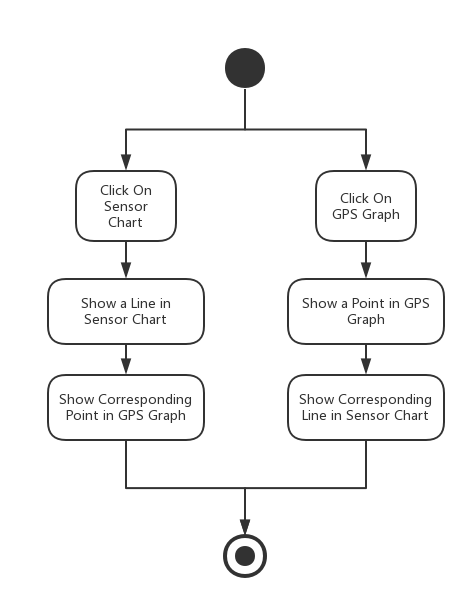


④

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



## Show Specific Value of Points



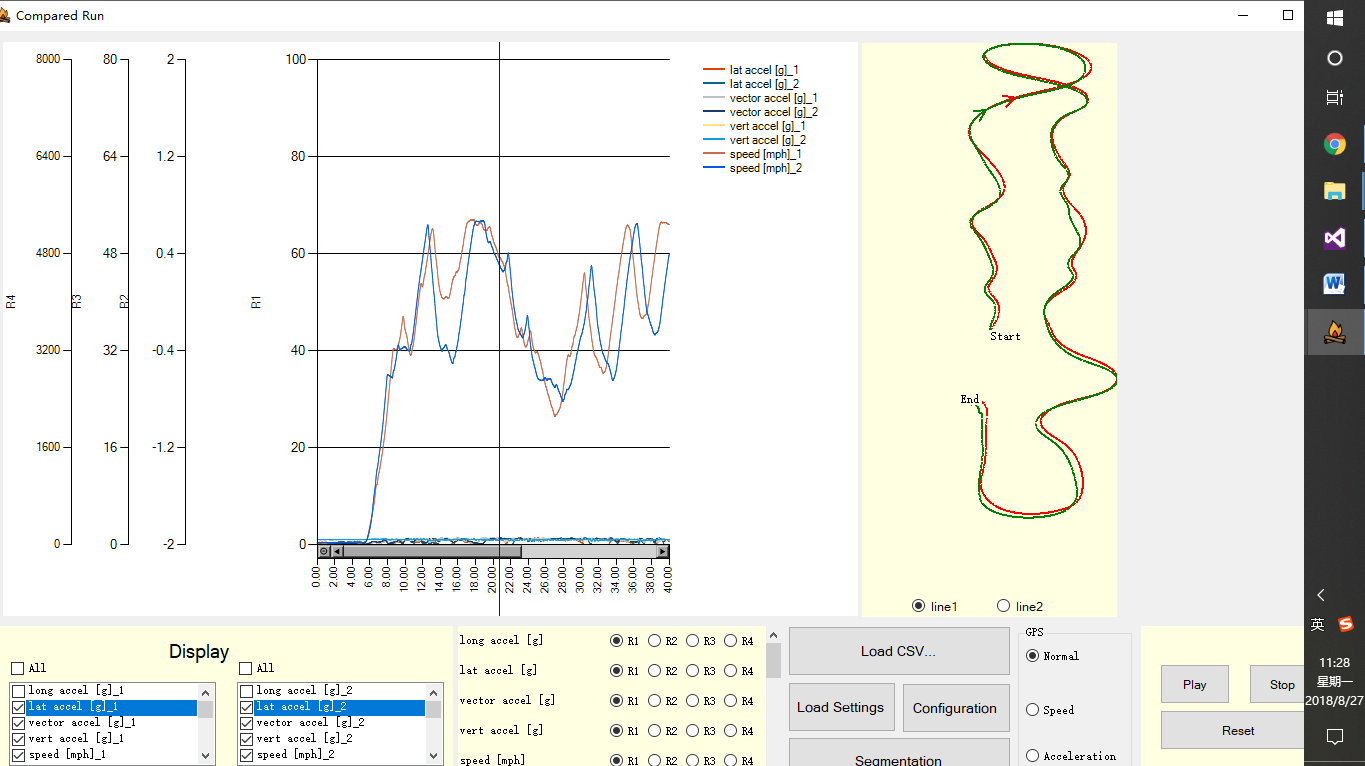
Users could click on the graph and show a vertical line at where the mouse clicked. Meanwhile, there would be a black point in the GPS graph to show the same place as you click on the chart. Also, users could click on the GPS graph and show the corresponding on the sensor chart.

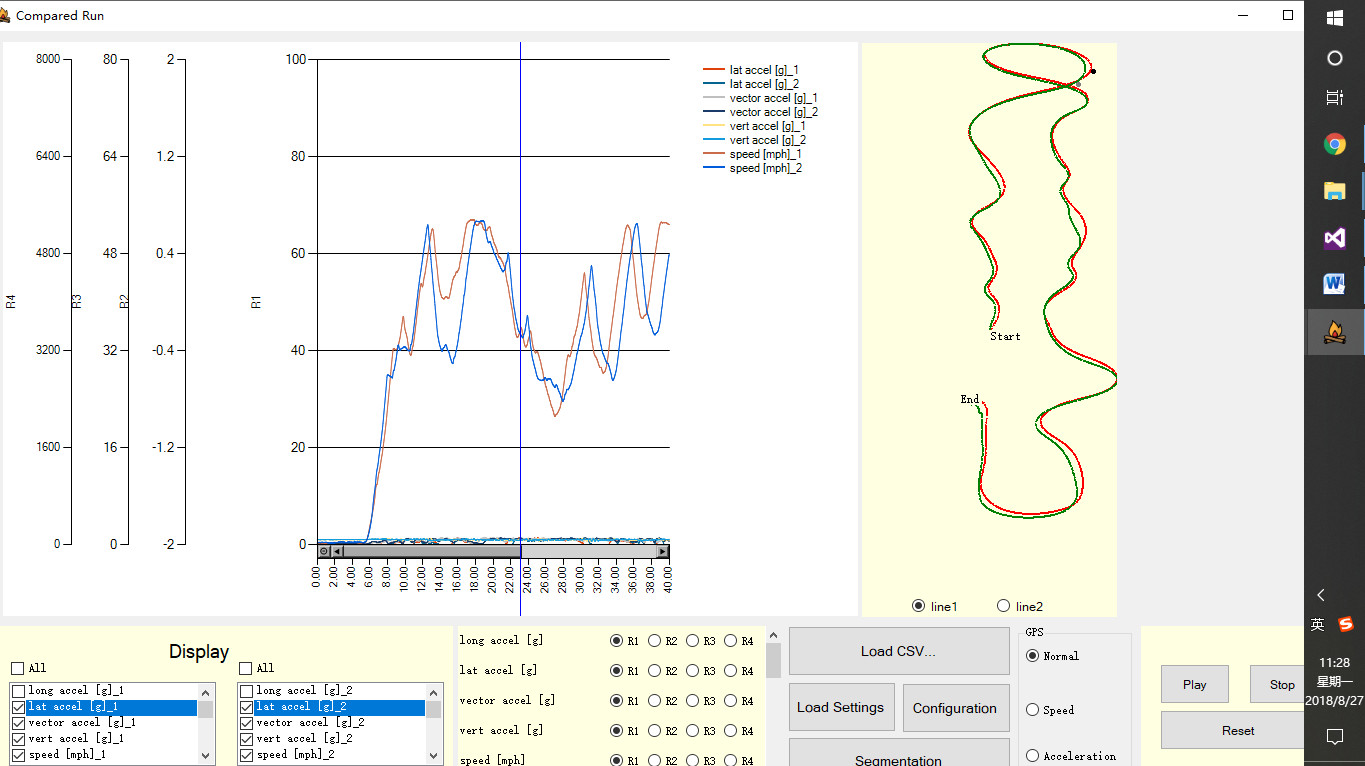
Related Functions:

void sensorChart\_MouseClick(object sender, MouseEventArgs e)

private void GPSPanel\_MouseClick(object sender, MouseEventArgs e)

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



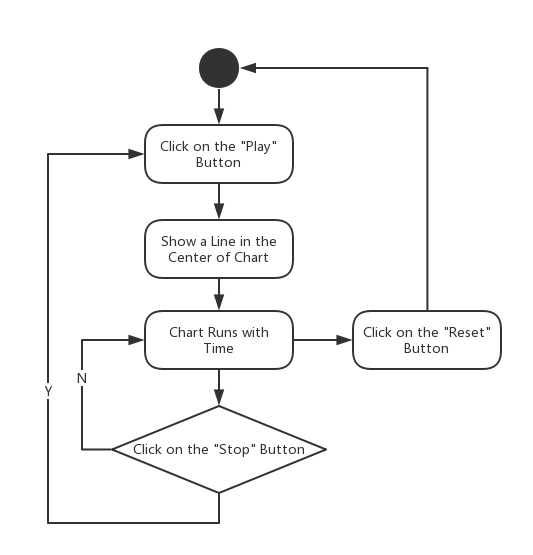


①Click On the GPS Graph

②Show the realtime line

on the chart

## 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)

## 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)

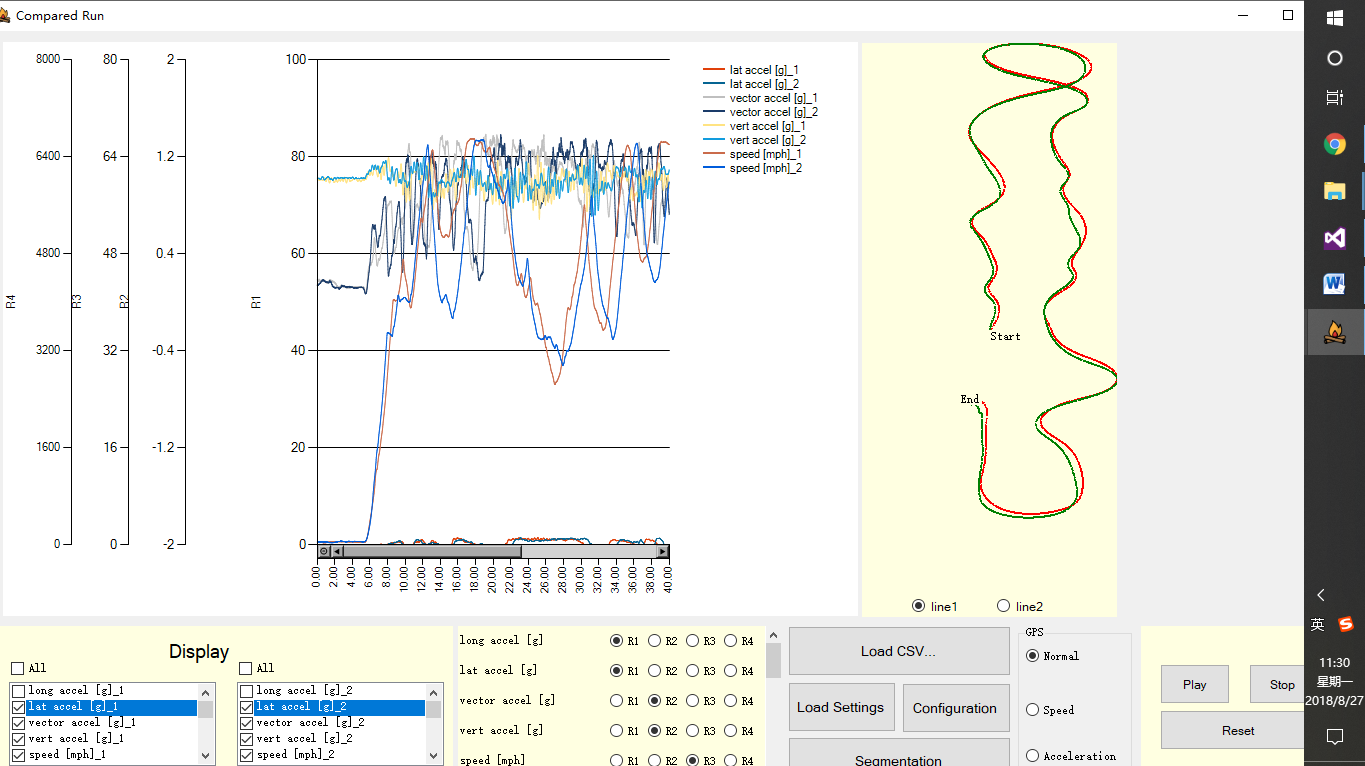
void change2(string no, ChartArea caR)

void rb1\_Click(object sender, EventArgs e)

void rb2\_Click(object sender, EventArgs e)

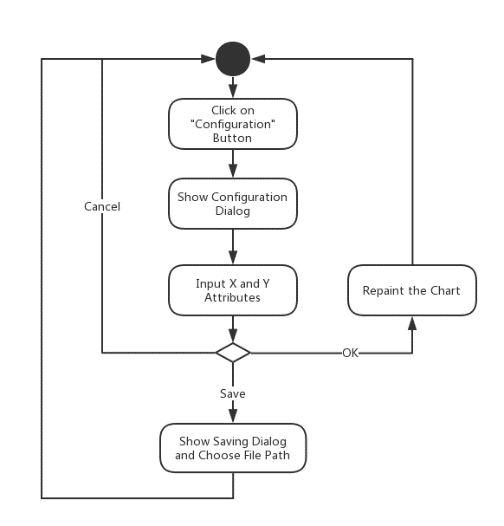
void rb3\_Click(object sender, EventArgs e)

void rb4\_Click(object sender, EventArgs e)



1. Choose Different Y axis Types

## 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.

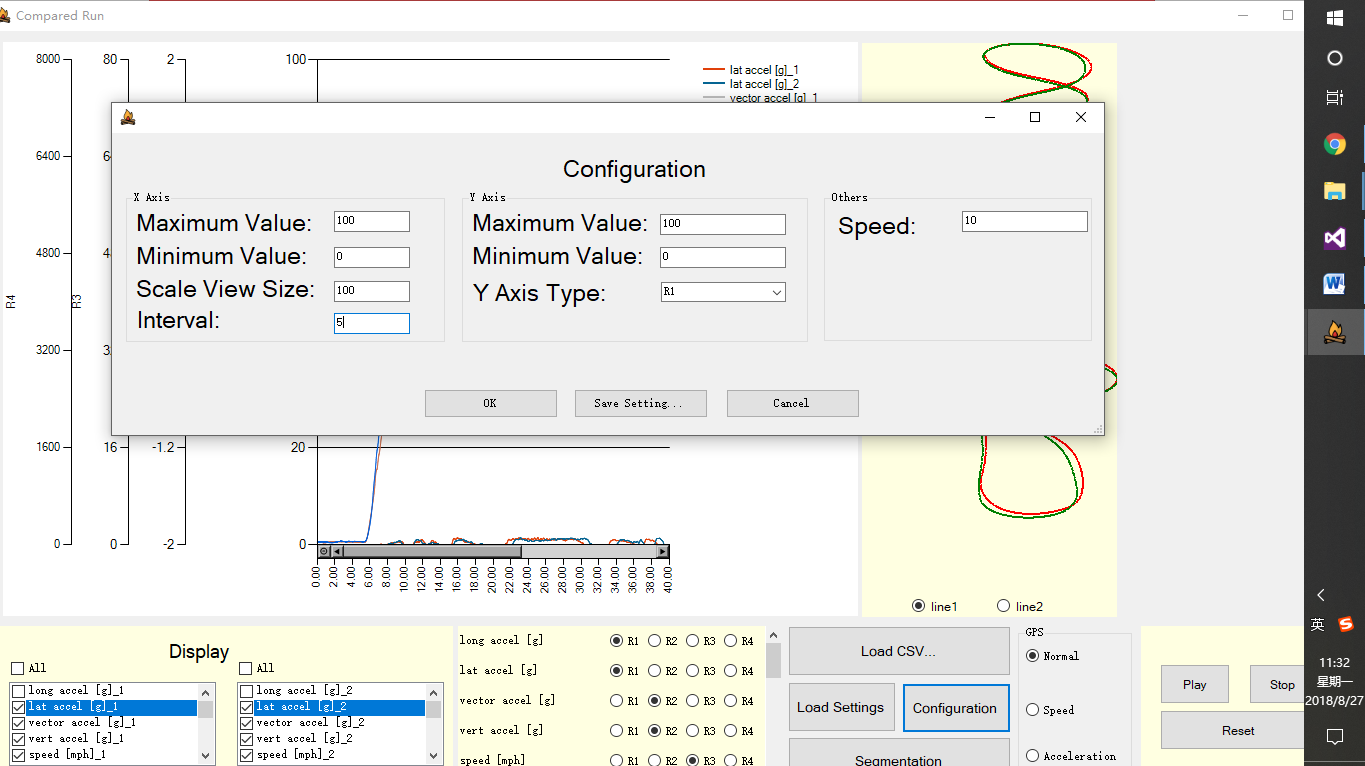
Related Functions and Forms:

public partial class RangeForm : Form

void YRangeForm\_Load(object sender, EventArgs e)

void confirmButton\_Click(object sender, EventArgs e)

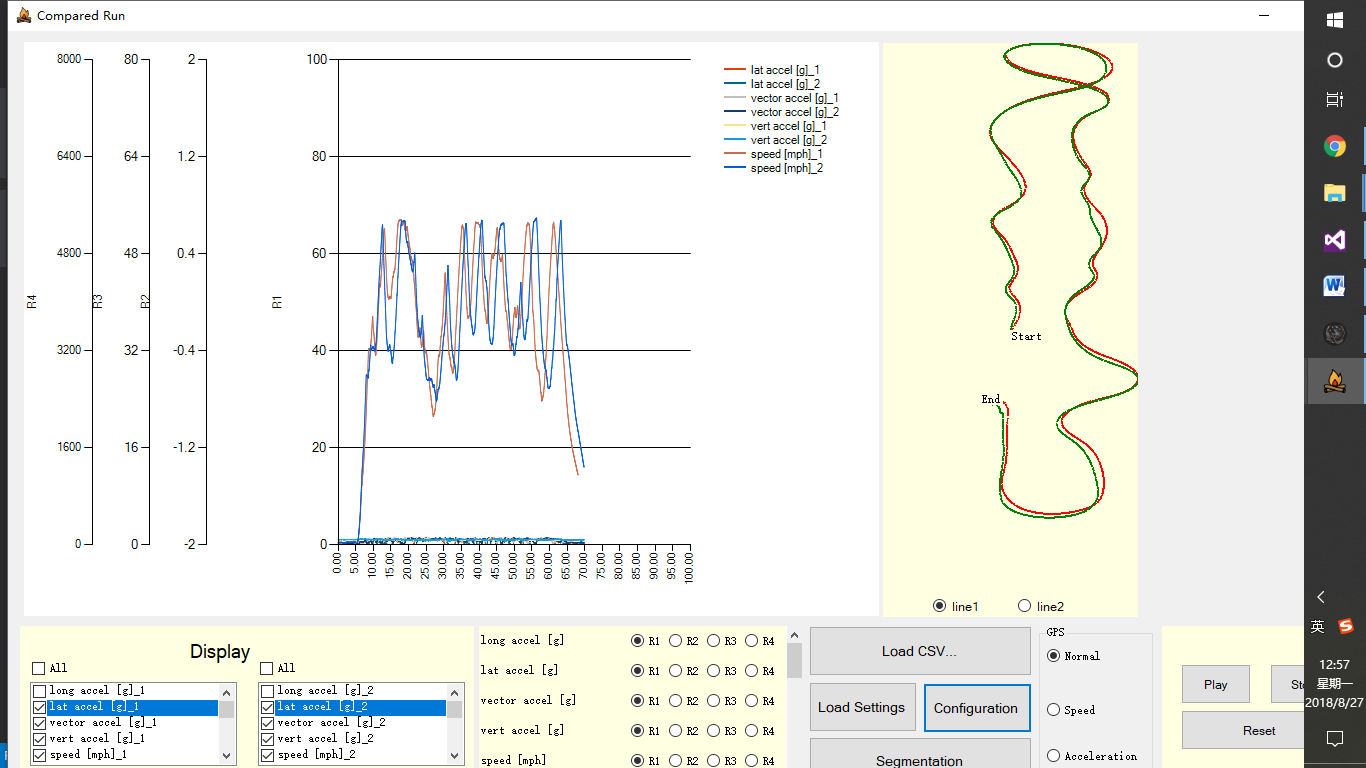
void ConfigureButton\_Click(object sender, EventArgs e)



①

②

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

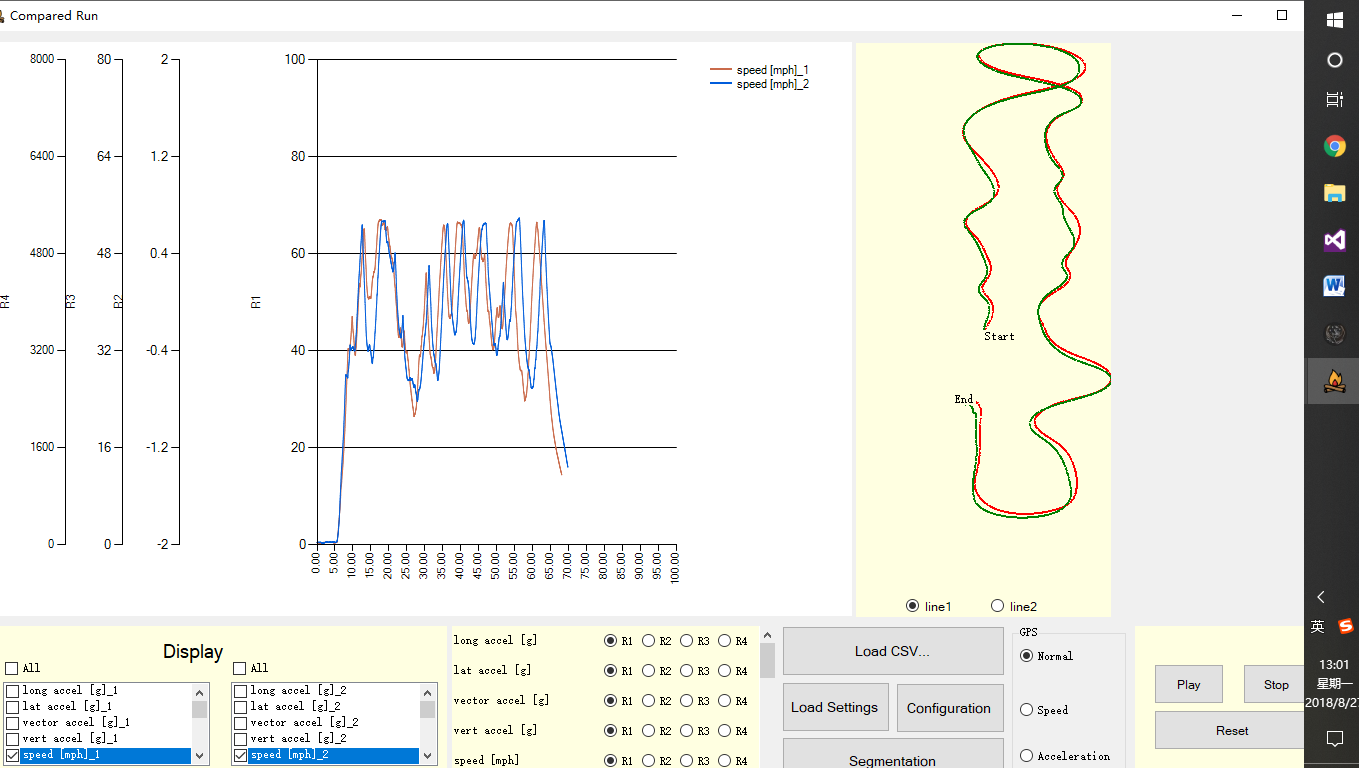


## Save and Load Setting Log Files

This function is really similar to the single ride. See [2.6](#_Save_and_Load).

## 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:

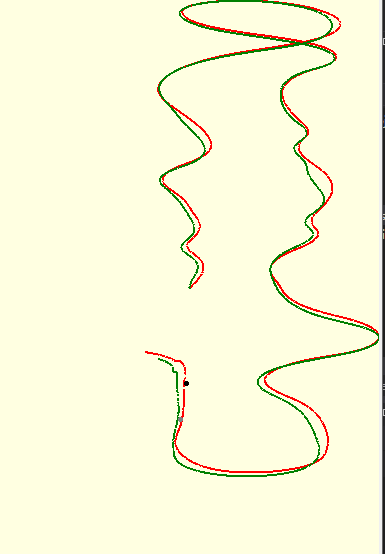
void sensorCheckedListBox\_ItemCheck(object sender, ItemCheckEventArgs e)

void allSelectedCheckBox\_CheckedChanged(object sender, EventArgs e)

## 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.



1. Click On the GPS map
2. Show where another driver is

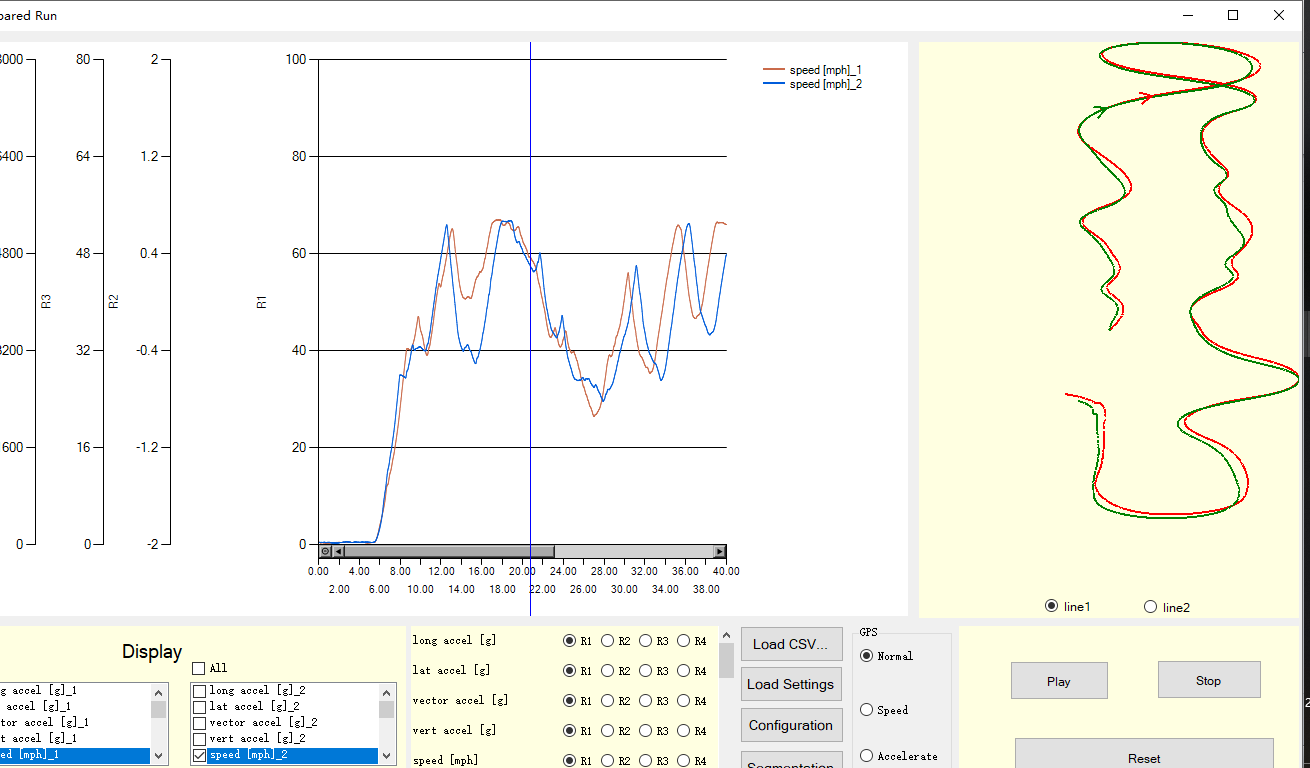
Related Functions:

private void GPSPanel\_MouseClick(object sender, MouseEventArgs e)

## 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)

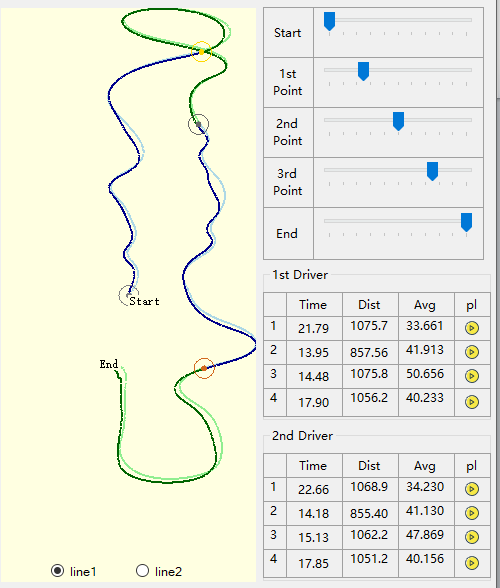
private void sensorChart\_MouseDown(object sender, MouseEventArgs e)

private void sensorChart\_MouseUp(object sender, MouseEventArgs e)

## 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%.



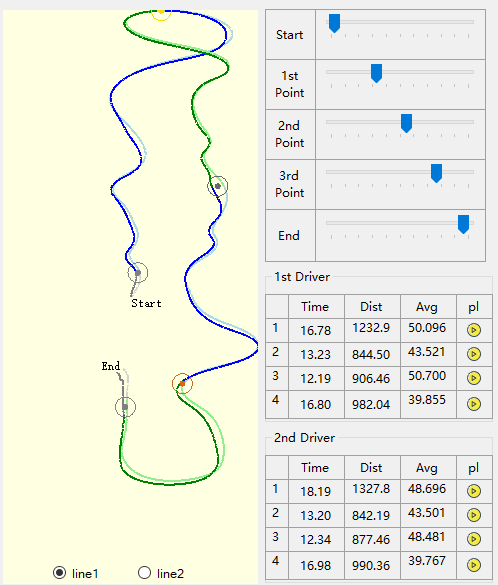
(4)

(3)

(2)

(1)

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



(3)

(4)

(2)

(1)

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)

# Testing Environment

- Visual Studio 2013&2015

- Programming Language: C#