Auckland University of Technology

School of Engineering, Computer and Mathematical Science

ENEL712 Embedded Systems Design

Mini Project Report

Submitted by: Min Choi

18 June 2020

Table of Contents

1 Introduction

This project is an important assessment component for the paper ENEL712 Embedded System Designs. Suppose you are working as an Embedded System Engineering in AnyWhere Inc., which is a leading manufacture for global positioning system for civil applications.

1.1 Project Overview

Recently, you are working with an automobile company to design a navigation and tracking system for their latest model of electrical vehicles. You are to develop a graphical user interface (GUI) for a Global Positioning System (GPS) data interpreter. Due to the COVID-19, you are not able to access the actual hardware, including GPS receiver and microcontroller board. However, you have decided to continue the development process in order to complete the project according to the mutually agreed deadline with your customer.

Luckily, you find the following items to help you during the development process:

GPS Simulator—In order to obtain GPS information without a physical GPS unit (or without driving around while testing the software!), you can use a free GPS simulator (SatGen NMEA) http://www.labsat.co.uk/index.php/en/free-gps-nmea-simulator-software. As shown in Figure 1, the software allows you to take a simulated track (or real if you have one) from Google Earth, convert it to a NMEA data file (see below), and then output it via a nominated serial port on a PC. [Note: To aid you understand the software, several NMEA data files are provided on AUT Online, however you can also generate your own if you wish.]

Virtual Serial Port Emulator—A simulator allows you to create multiple pairs of serial ports (URL: https://www.virtual-serial-port.org/articles/top-6-virtual-com-port-apps/). This allows you to test software development with serial ports without the actual physical connection of serial ports.

Tera Term—It is an open-source, free, software implemented, terminal emulator program (URL: https://tera-term.en.lo4d.com/windows). It emulates different types of computer terminals, from DEC VT100 to DEC VT382. It supports telnet, SSH 1 & 2 and serial port connections. It also has a built-in macro scripting language and a few other useful plugins. It is useful tool to simulate serial port transmitters and receivers.

2 Development Process

Suppose you have installed a copy of Virtual Studio 2019 (Remember to choose at least these three workloads: .NET desktop development, Desktop development with C++ and Universal Windows Platform development) on your computer. In addition, you have also installed Virtual Serial Port Emulator and Tera Term.

2.1 Requirement

Suppose you need to develop a GUI interface based on C# forms to emulate the actual GPS navigation and tracking product. The required functionality of the GUI includes:

A. The GUI will include three tabs, namely Serial Port, GPS Data and GPS Data Analysis.

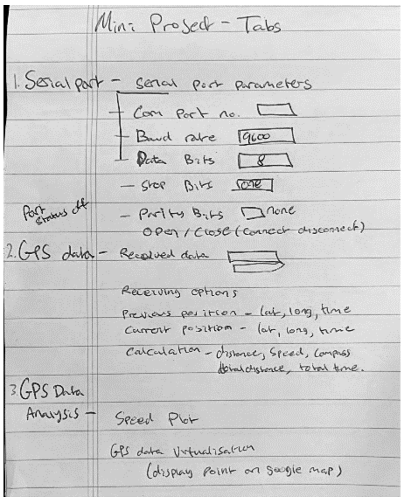
B. For the Serial Port Tab, it allows a user to set up a serial port by choosing parameters including COM port number, baud rate, data bits, stop bits and parity bits. Default values for baud rate, data bits, stop bits and parity bits are set as “9600”, “8”, “One”, “None”. When the serial port is off, the port status is “OFF”, the “close” button is disabled, and the “open” button is enabled. Similarly, when the port is on, the port status is “ON”, the “close” button is enabled, and the “open” button is disabled.

C. In the GPS Data tab (see Figure 6) , it includes three parts. On the top is the received raw GPS data. It can show the newly received data, or aggregated data by appending new data to the old data. In addition, it shows the latest new $GPGGA sentence in the textbox immediately below the raw GPS data textbox. In the middle, it shows two points with their sets of GPS coordinates and timestamps, including the current position and the previous position. By taking two points as input, the GUI can display the distance between them, the traveling speed at the moment, the compass bearing, the total distance travelled (mileage) and the total time elapsed (duration of the journey).

D. In the GPS Data Analysis Tab, it includes two parts. The first part shows a speed log chart, which display the speed of vehicle, recorded at the time interval obtained in the GPS Data Tab.
In the GPS Data Analysis tab, the second part is OPTIONAL: It captures a point in the journey and display it in Google Map (URL: https://www.google.co.nz/maps).

3 Methodologies

First, I drew and wrote down roughly the requirements for each tabs. (Figure 1)



Mini Project - Tabs

1. Serial port - serial port parameters
- Com Port no.
- Baud rate 9600
- Data Bits 8
- Stop Bits one
- Parity Bits none

Port status off

Open / Close (Connect disconnect)

2. GPS data - Received data

Receiving options

Previous position - lat, long, time

Current position - lat, long, time

Calculation - distance, Speed, Compass, total distance, total time.

3. GPS Data Analysis - Speed Plot

GPS data Virtualisation (display point on google map)

Figure 1

Then I created a new Visual Studio Project. Using Windows Forms App (.NET Framework). Before I started developing the application, I had to watch a few videos and read about Windows Forms App as this was my first time doing a project in this format. Once I found few good sources, I started developing the application.

3.1 Design process

I created a Tab Control component and created 3 tabs with required names. Then I added each elements of the requirements such as Combo Box for serial port parameters, all the required Buttons, Text Box and Rich Text Box to display the received and calculated data and a Chart for speed log in their respective tabs.

Before I started programming the required functionalities, I decided to complete the design of the user interface because I didn’t want to keep changing things as I added functions. So, I had to figure out how big each component needs to be. I figured this out by counting how many characters each box needed to display and what size font I wanted it to display at. Then I set the size and the location of all my tabs and their components.

3.1.1 Serial Port Tab (Tab 1)

Once I finished the design of the user interface, I started programming all the functions. I went step by step from top to bottom, tab 1 to tab 3. First function I coded was to find available COM ports then map it to the COM port selector box. Next I coded the options for baud rate, data bits, stop bits, parity bits and programmed it to set the selected option to be the setting for serial port connection.

Next I coded the function for when the open button is pressed, which took all the serial port settings and connected to the serial. (Figure 2). Then the close button which terminated the serial connection. As per requirement I made only one enabled a time.

```
private void button5_Click(object sender, EventArgs e)
        {
            try
            {
                comPort.PortName = comSel.Text; //set portname to what is selected from COM Port box
                comPort.BaudRate = baudRate;    //set baudrate
                comPort.DataBits = dataBits;    //set data bit
                comPort.StopBits = stopBits;    //set stopbit
                comPort.Parity = parity;        //set parity
                comPort.Open();                 //connect to serial

                if (comPort.IsOpen)             //when serial is conected
                {
                    offB.Enabled = true;        //enable close button
                    onB.Enabled = false;        //disable open button
                    portStatus.Text = "ON";     //change port status to ON
                    progressBar1.Value = 100;   //fill up progress bar
                }
            }
            catch (Exception err)               //if serial cannot be connected
            {
                MessageBox.Show(err.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error); //display
error message
                portStatus.Text = "OFF";    //set port status to OFF
                offB.Enabled = false;   //disable close button
                onB.Enabled = true;     //enable open button
            }
        }   //serial connect when OPEN button is pressed
```
Figure 2

The open button connects to serial port and when a connection is made open button is disable, close button is enabled. The status show on or off depending whether serial is connected or not. The progress bar is filled while serial port is on. This function can be seen in (Figure 3).

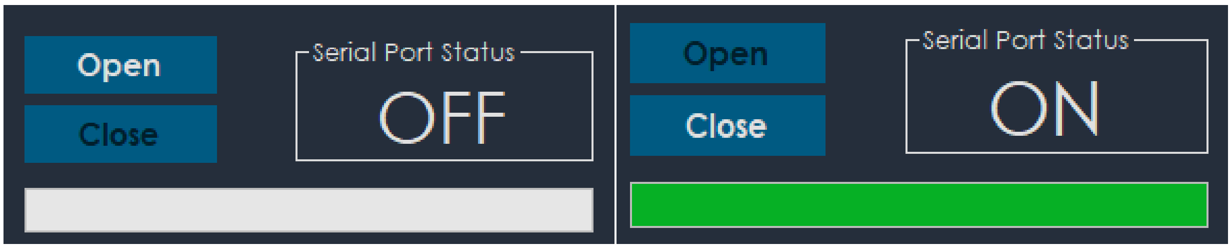


Figure 3

That was all that was needed for serial port tab, so I moved on to the next tab.

3.1.2 GPS Data Tab (Tab 2)

For GPS Data tab I needed to first be able to receive the serial data, then using the data received manipulate and calculate to be able to display in the respective boxes.

Implementing the serial receive function was quite easy as shown in (Figure 4) below.

```
private void comPort_DataReceived(object sender, SerialDataReceivedEventArgs e)
        {
            dataIn = comPort.ReadExisting();    //read serial port data
            this.Invoke(new EventHandler(updateData));  //goto update data function
        }   //when new data is received
```
Figure 4

The hard part for this tab came next. Which was using the received data to make the data which will be displayed in all the data boxes.

First I tested if I was getting the correct data from the SatGenNMEA by setting the raw data box to display the data in. once I checked that it was working I made the raw data box to show either the current line or all lines received depending which check box was selected (Figure 5). Then I set the box below it to display the raw data minus the $ sign in front. This was achieved by using the code in (Figure 6). The clear received data button clears the saved data and all the data boxes.

```
if (checkBox1.Checked)  //if always update is checked
{
    dataBox.Text = dataIn;  //display single line of received data to raw data box
}
else if(checkBox2.Checked)  //if add to old data is checked
{
    dataBox.Text = string.Join("",allData); //add to previous data and display on raw data box
}
```

Figure 5

```
dataB2 = dataIn.Remove(0, 1); //remove the first character of dataIn
```

Figure 6

Then I coded for the current and previous time. Next I realized all other boxes for this tab needed to be a calculation. For latitude I needed 3 parts. First 2 numbers of the latitude part of the GPGGA string which is the degree and the remaining numbers of the latitude which is the minutes. And third part was the N or S for the latitude. Longitude also needed 3 parts, but it was the first 3 numbers of longitude part which was degrees, remaining numbers were the minutes and E or W. I used (Figure 7) to calculate the latitude and longitude. First the S or N for latitude was converted to -1 or 1 and E or W also to -1 and 1. 60 minute is 1 degree so I divided the minute value of latitude and longitude by 60 then added to degree value then multiplied but either -1 or 1 depending on the S, N, E or W.

```
if (lastArray[3] == "S")
{
    lat2Dir = -1.0;
}
else if (lastArray[3] == "N")
{
    lat2Dir = 1.0;
}

if (lastArray[5] == "W")
{
    lon2Dir = -1.0;
}
else if (lastArray[5] == "E")
{
    lon2Dir = 1.0;
}
double lat1 = lat1Dir * (lat1D + (lat1M / 60));
double lon1 = lon1Dir * (lon1D + (lon1M / 60));
```

Figure 7

I did same calculation for previous location, current location and origin. (Figure 8) is used to calculate the distance. I converted current and previous position to GeoCoordinate then calculated the distance.

```
GeoCoordinate point1 = new GeoCoordinate(lat1, lon1); //previous lat and long to geocoordinate
GeoCoordinate point2 = new GeoCoordinate(lat2, lon2); //current lat and long to GeoCoordinate
double distance = Math.Round(point1.GetDistanceTo(point2),2); //distance between two points rounded to 2dp
```

Figure 8

Speed was calculated by simply using, $speed = \frac{\text{distance}}{(\text{currentTime}-\text{prevTime})} * 3.6$, the 3.6 was needed to be multiplied to convert from m/s to km/h. Total distance was calculated by comparing between origin location and time to current location and time. Method was same as above.

3.1.3 GPS Analysis Tab (Tab 3)

For the chart I used the total time and plotted it against the speed.

```
speedChart.Series["Speed"].Points.AddXY(time4, speed); //set time for X values, and speed for Y values
```

Figure 9

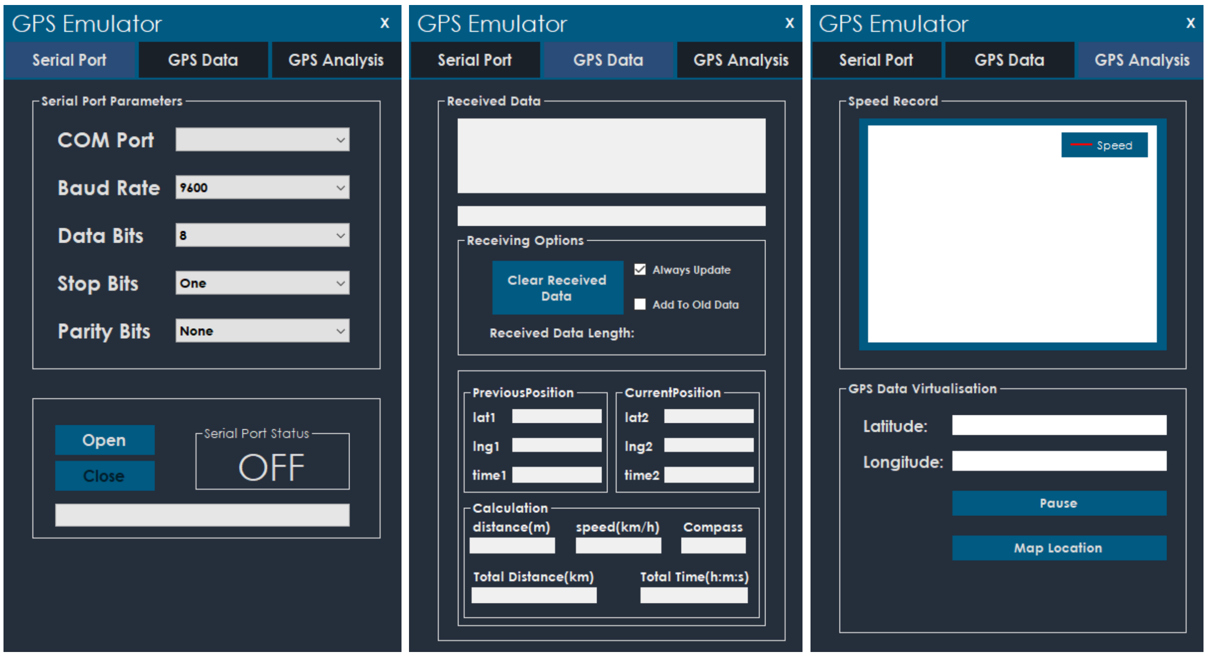
This was all the compulsory requirements but as I still had sometime left, I decided to do the optional requirement which was taking a point during the travel and then open google map of that point. I implemented the pause button to capture the current latitude and longitude then display them in their respective boxes. When the pause button is pressed it becomes disabled and resume button is shown instead. Then when the user presses the map location button browser opens up google map to the captured point.

```
System.Diagnostics.Process.Start("https://www.google.co.nz/maps/place/" + textBox13.Text + "," +
textBox12.Text); //textBox12 is longitude, textBox13 is latitude
```

4 Results

I have managed to Implement all the required functions including the optional task. Here are the snapshots of my application working.

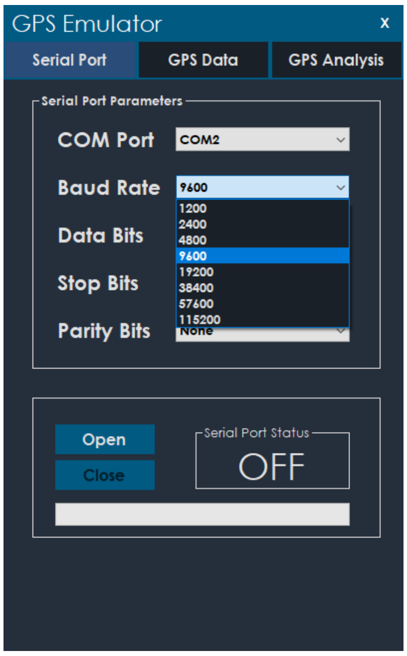
4.1 Tabs



COM Port:
Baud Rate: 9600
Data Bits: 8
Stop Bits: One
Parity Bits: None
Serial Port Status: OFF
Always Update: ✓

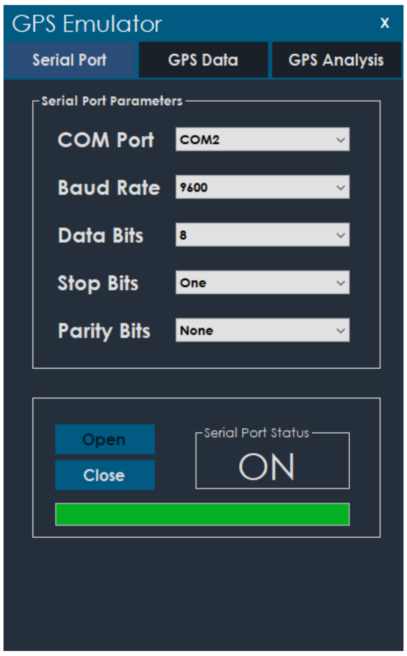
Figure 10

(Figure 10) above shows each tabs when the application is just started up. All the boxes are still empty because no data have been received yet. (Figure 11) below shows the drop down menu where the user can choose the serial connection option and (Figure 12) shows the application when it is connected to serial port.



COM Port: COM2
Baud Rate: 9600
1200
2400
4800
9600
19200
38400
57600
115200
Data Bits
Stop Bits
Parity Bits: None
Serial Port Status: OFF

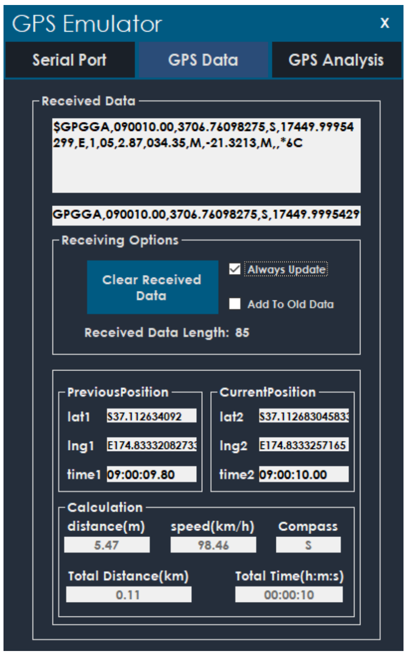
Figure 11



COM Port: COM2
Baud Rate: 9600
Data Bits: 8
Stop Bits: One
Parity Bits: None
Serial Port Status: ON

Figure 12

(Figure 13) is the GPS Data tab when it is receiving data from serial and (Figure 15) is the original GPGGA data in SatGenNMEA application. And (Figure 14) is when add to old data is checked instead of always update.



Received Data

```
$GPGGA,090010.00,3706.76098275,S,17449.99954
299,E,1,05,2.87,034.35,M,-21.3213,M,,*6C
```

```
GPGGA,090010.00,3706.76098275,S,17449.9995429
```

Receiving Options

[✓ Always Update]

Add To Old Data

Received Data Length: 85

PreviousPosition

lat1 S37.112634092

lng1 E174.83332082733

time1 09:00:09.80

CurrentPosition

lat2 S37.11268304583

lng2 E174.8333257165

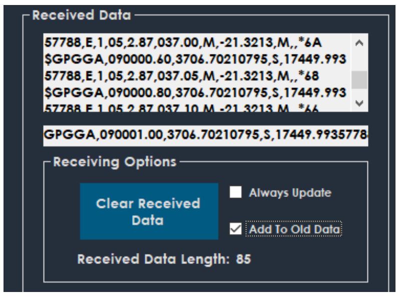
time2 09:00:10.00

Calculation

distance(m)	speed(km/h)	Compass
5.47	98.46	S

Total Distance(km)	Total Time(h:m:s)
0.11	00:00:10

Figure 13



Received Data

```
57788,E,1,05,2.87,037.00,M,-21.3213,M,,*6A
$GPGGA,090000.60,3706.70210795,S,17449.993
57788,E,1,05,2.87,037.05,M,-21.3213,M,,*68
$GPGGA,090000.80,3706.70210795,S,17449.993
57788,E,1,05,2.87,037.10,M,-21.3213,M,,*66
```

```
GPGGA,090001.00,3706.70210795,S,17449.9935778
```

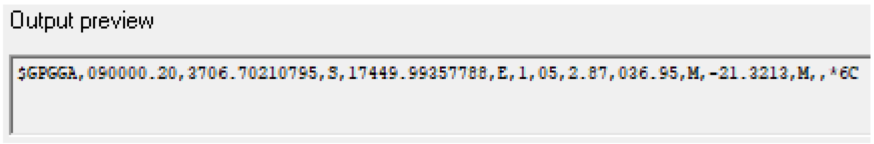
Receiving Options

Always Update

[✓ Add To Old Data]

Received Data Length: 85

Figure 14



Output preview

```
$GPGGA,090000.20,3706.70210795,S,17449.99357788,E,1,05,2.87,036.95,M,-21.3213,M,,*6C
```

Figure 15

(Figure 16) Shows the GPS Analysis tab while it is logging the speed history graph and while pause button haven't been pressed. (Figure 17) shows the tab after pause button have been pressed.

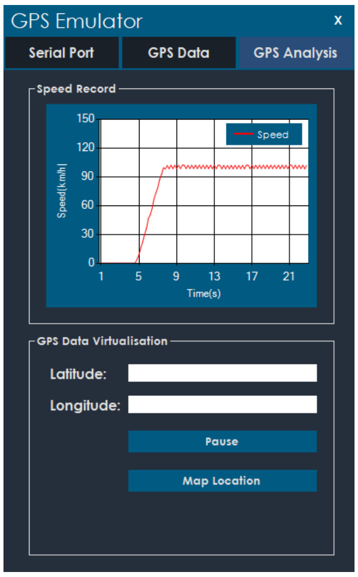
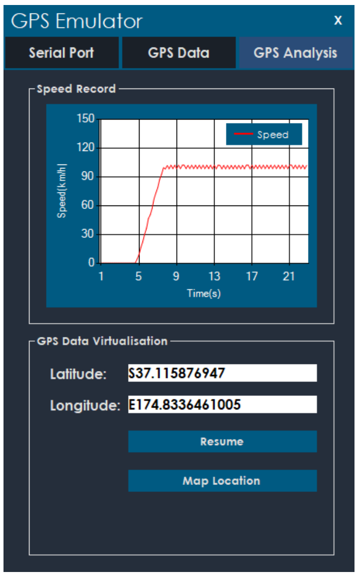


Figure 16



Latitude: S37.115876947

Longitude: E174.8336461005

Figure 17

We can see that the speed plot is working, and the pause button captures a point then becomes the resume button.

4.2 Verifying Values

From (Figure 13) we can see that the data received is

```
$GPGGA,090010.00,3706.76098275,S,17449.99954299,E,1,05,2.87,034.35,M,-
21.3213,M,,*6C
```

And from (Figure 15) the data sent is

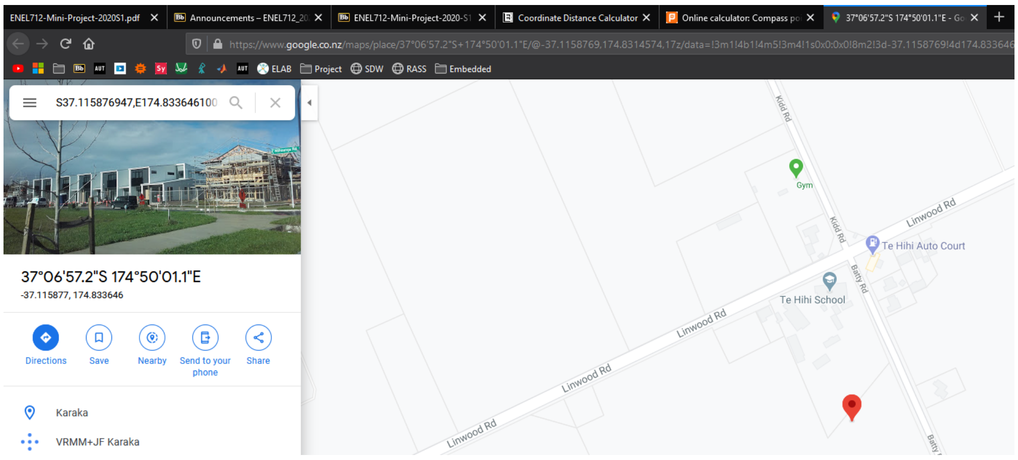
```
$GPGGA,090010.00,3706.76098275,S,17449.99954299,E,1,05,2.87,034.35,M,-
21.3213,M,,*6C
```

By using https://text-compare.com/ I checked that these are identical therefore the application is correctly receiving the data sent. We can see easily that the time is being displayed correctly because `090010` is being shown as 09:00:10.00(HH:mm:ss.ss). Next we should verify whether it is processing the latitude and longitude correctly. Lets use https://rl.se/gprmc to convert the GPGGA to latitude and longitude. 37.112683°S 174.833326°E is the result from the website. The application shows S37.1126830458333 E174.8333257165 this is correct but just unrounded. Now lets check the distance by copying the previous and current latitude and longitude to http://boulter.com/gps/distance/ and 5 meters was the result from the website and 5.47 meters is the result from the application so distance is accurate too. To calculate the speed, we divide the distance by the difference in time between current and previous.

speed $= \frac{5.47}{(900010.00-900009.80)} = \frac{27.35m}{s} = 98.46km/h$ the application shows 98.46 so that is accurate. To verify the compass heading I used https://planetcalc.com/7042/ plugged in the previous and current latitude and longitude and got S as the compass heading and the application shows S so everything is working correctly. Total distance and total time I have verified previously by using the text file of the NMEA and putting to distant data into my program.

4.3 Optional Requirement

(Figure 17) shows the application when the pause button is pressed. We can see that latitude = S37.115876947 and longitude = E174.8336461005 now () shows the resulting website when the map location button is pressed



We can see that the google map is showing the correct coordinate.

5 Discussions

I have made custom tab button and exit window button because the default looked too dated. As this was an exercise to develop a graphical user interface, I thought it was relevant to have a modern design. Especially for a latest model of electrical vehicles.

I was able to make everything work perfectly except the chart. I could not figure out how to make it not auto scale in X-axis while being able to update it to new maximum. As a result, it jumps a bit while logging the speed plot.

Also, I had a problem near the end where the visual studio started throwing all sorts or errors and I could not figure out exactly what line I made the mistake so I had to erase most of my calculation and serial function so my code is quite messy now and not in a neat order of operation as I have wrote first time.

6 Conclusion

I have managed to satisfy all the requirements of each tabs and the optional requirement. I have deviated from default windows forms elements and implemented some custom design to the GUI because I thought it looked more modern and was a better fit for the project objective.