Building Rail-Vehicle Braking Distance Calculation Tool Using C# Programming Language

Andinet Kumella

School of Mechanical and Industrial Engineering, Addis Ababa Institute of Technology

Abstract- This Research work presents the development of computer programming software namely; IBM PC® based modular tool for the prediction of vehicle deceleration, and braking time for various position of the transport vehicle at any time. Finally, particular attention is devoted to the accurate prediction of vehicle stopping distance. These will help to improve the braking system design in the rail vehicle industry, with respect to the safe movement of the transportation industry.

Index Terms- Vehicles, Braking, acceleration, distance, time.

I. Introduction

For trains to safely travel on a railway, trains must be provided with sufficient distance in which to stop. Allowing too long a distance reduces the capacity of the line and hence the return on rail infrastructure investment. Too short a distance and collisions would occur, because the train would not be able to stop within the available distance and would therefore occupy a section of track that could be allocated to another train. Consequently it is important that the distance be adequate, but not overly so [1].

The purpose of this IBM PC® based tool is to:

- Assist the signal designer to provide an adequate distance for the safe stopping of trains operating on a given line whilst at the same time maximizing line capacity;
- Put in place controls to ensure that the data used for the calculation is verified and traceable, and any changes controlled;
- Allow the ready evaluation of a new train by performing multiple calculations in one execution;
- Readily highlight those Limits of Authority that need to be relocated to allow for changes in speed or train braking performance.

II. CALCULATING BRAKING DISTANCE

Influencing Factors for Braking Distance

- The speed of the train when the brakes are applied;
- The deceleration rate available with a full-service brake application, which varies according to the coefficient of friction between wheel and rail;
- The delay from when the brakes are commanded by the train driver to when they are actually become effective (brake delay time);
- The state of the wear of the brake pads and the air pressure available in the brake cylinders;
- The geography of the track, in particular the track gradient the train travels over from when the brakes are commanded to where the front of the train stops;
- The mass distribution of the train.

III. C# PROGRAMMING TOOL

C# is a modern, general-purpose, object-oriented, high-level programming language. Its syntax is similar to that of C and C++ but many features of those languages are not supported in C# in order to simplify the language, which makes programming easier.

The C# programming language is a good choice, because it is an elegant language through which the program's representation in the computer memory is of no concern to us and we can concentrate on improving the efficiency and elegance of our program.

Advantage of C#

C# is object-oriented programming language. Such are all modern programming languages used for serious software systems (like java and c++). The advantages of object-oriented programming are brought up in many passages throughout the book, but, for the moment, you can think of object-oriented languages as languages that allow working with objects from the real world.

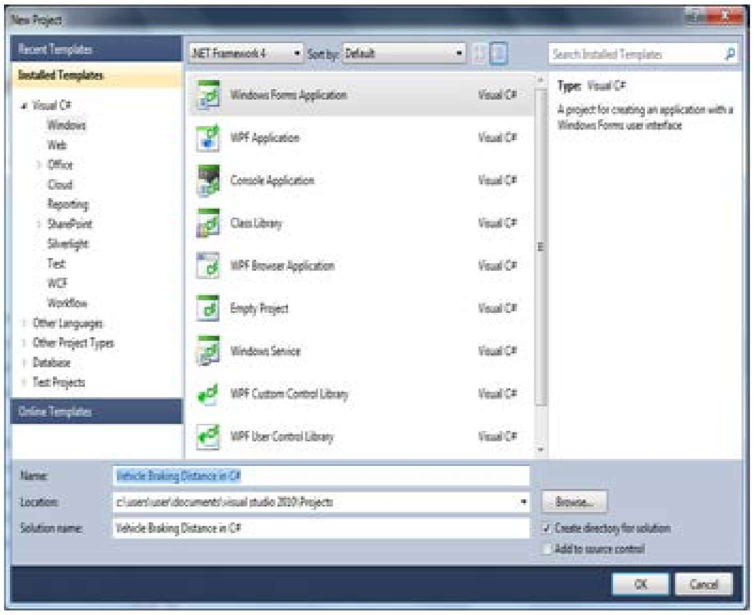
Development of C# Modular Tool

The tool is used in the design process for a railway signaling application. It is only used when required to calculate train braking distances. There are two versions of the tool;

1. Microsoft Windows®2 environment
2. IBM OS/2®3 environment.

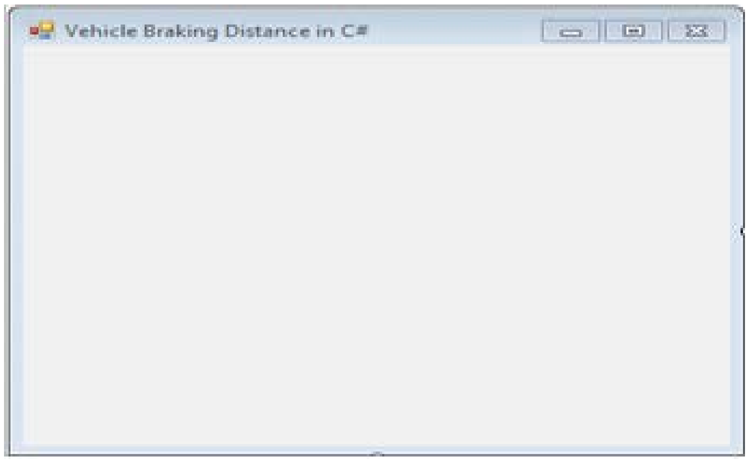
Both versions are written using the C++ language. However C++ compilers from different suppliers are used. The two versions have been designated as “ISAAC” and “NEWTON”. Both ISAAC and NEWTON versions of the tool need to be used before any calculated result can be accepted.

New window forms

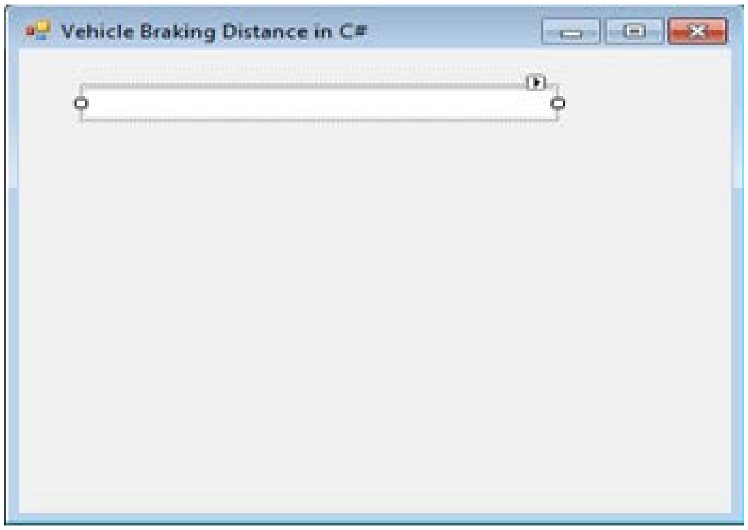

New Project
Recent Templates
Installed Templates
Visual C#
Windows
Web
Office
Cloud
Reporting
SharePoint
Silverlight
Test
WCF
Workflow
Other Languages
Other Project Types
Database
Test Projects
Online Templates
.NET Framework 4
Sort by: Default
Search Installed Templates
Windows Forms Application Visual C#
WPF Application Visual C#
Console Application Visual C#
Class Library Visual C#
WPF Browser Application Visual C#
Empty Project Visual C#
Windows Service Visual C#
WPF Custom Control Library Visual C#
WPF User Control Library Visual C#
Type: Visual C#
A project for creating an application with a Windows Forms user interface
Browse...
OK
Cancel

Name: Vehicle Braking Distance in C#
Location: c:\users\user\documents\visual studio 2010\Projects
Solution name: Vehicle Braking Distance in C#
✓ Create directory for solution
Add to source control

Name of the form

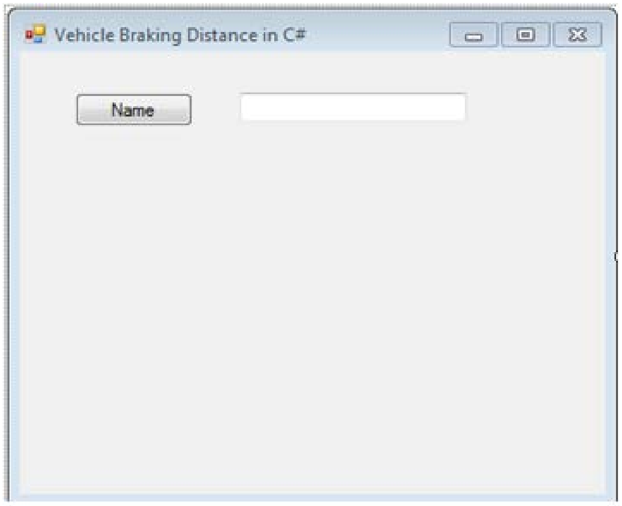


Add a text box for the display



Create buttons

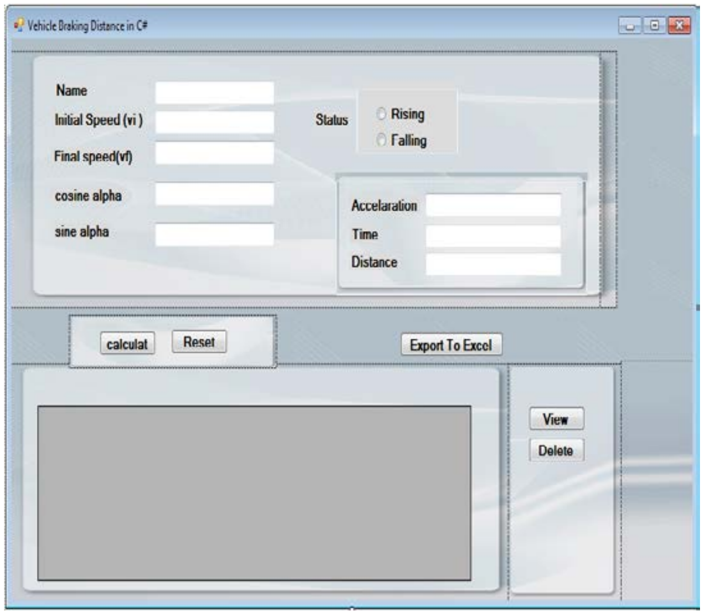
- The number buttons should have their text be the actual numbers. I will use this text in the program.
- The function buttons names are not important.



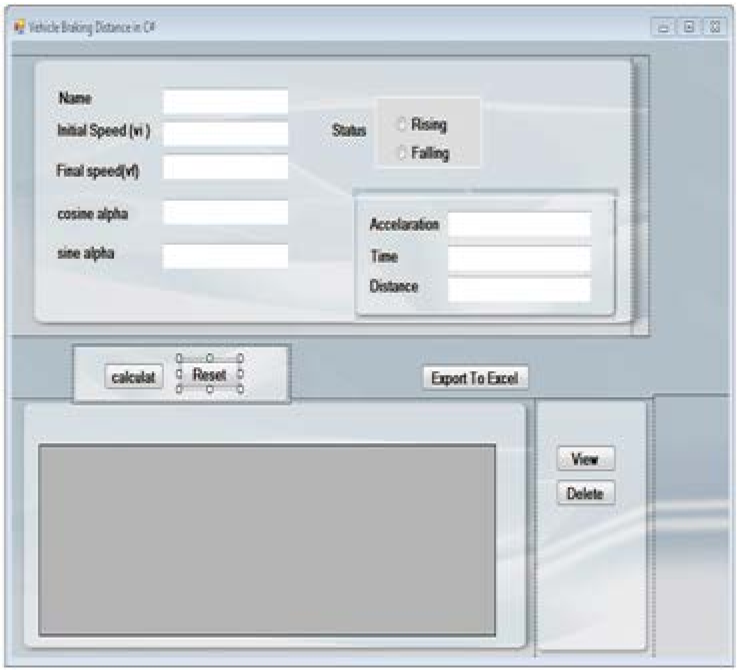
Copy / Paste, make it look nice

- Copy and paste the buttons to keep size and placement consistent.
- Alter the text values to correspond to the buttons
- Adjust the size of the form and placement of the elements
- Change the background color of the form

The Finished Layout

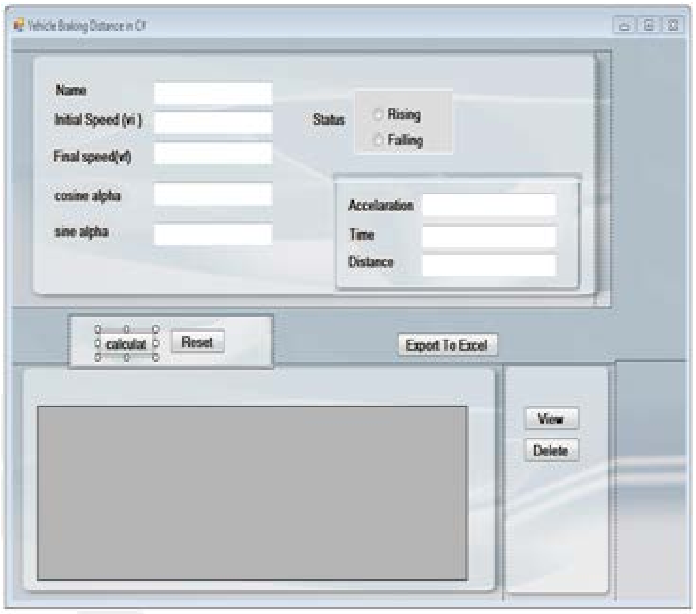


Let's Program the Reset Button



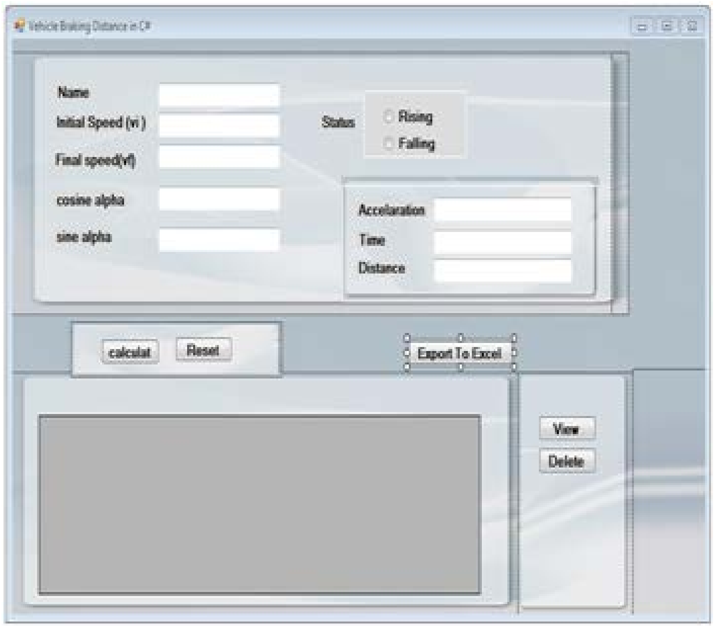
```
privatevoid btnReset_Click(object sender, EventArgs e)
        {
        txtaccelaration.Text = "";
        txtdistance.Text = "";
        txtvf.Text = "";
        txtvi.Text = "";
        txtcos_alpha.Text = "";
        txtsine_alpha.Text = "";
        radiobtnFalling.Checked=false;
        radiobtnRising.Checked=false;
        groupBoxCalculate.Visible=false;

        }
```

Program the Calculat Button



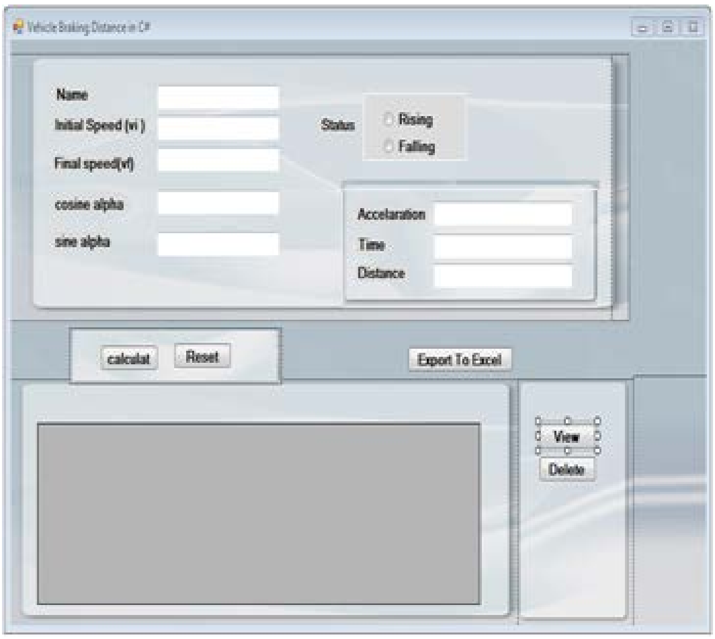
```
}

privatevoid
calculate_Click(object sender, EventArgs e)
  {
if (CheckTxts())
return;
else
    {
String name = txtName.Text;
double vf = Convert.ToDouble(txtvf.Text);
double vi = Convert.ToDouble(txtvi.Text);
double cos_alpha =
Convert.ToDouble(txtcos_alpha.Text);
double sine_alpha = Convert.ToDouble(txtsine_alpha.Text);

//calculate accelaration
if(radiobtnRising.Checked                    ==                    true)
       {
acc = (1 + X + (Y * cos_alpha) - (Y * sine_alpha)) / 100;
status = "Rising";
       }
if (radiobtnFalling.Checked == true)
    {
acc = (1 - 1 - (Y * cos_alpha) - (Y * sine_alpha)) / 100;
status = "Falling";
    }
txtaccelaration.Text =Convert.ToString(acc);

//calculate distance
if(acc                              ==                              0)
    {
dist = 0;
time = 0;
}
else
{
dist = ((vf * vf) - (vi * vi))/(2 * acc);
time = (vf-vi)/acc;
}
txtdistance.Text = Convert.ToString(dist);
txttime.Text = Convert.ToString(time);

}
```

Let's Program Export To Excel Button



```
}
        }
        }
publicOleDbConnection connection =
newOleDbConnection(@"Provider=Microsoft.ACE.OLEDB.12.
0;Data Source=D:\calculation.accdb");


privatevoidexport_Click(object sender, EventArgs e)
        {

try
            {
// creating Excel Application
                Microsoft.Office.Interop.Excel._Application app =
newMicrosoft.Office.Interop.Excel.Application();

// creating new WorkBook within Excel application
                Microsoft.Office.Interop.Excel._Workbook workbook
= app.Workbooks.Add(Type.Missing);

// creating new Excelsheet in workbook
                Microsoft.Office.Interop.Excel._Worksheet worksheet
= null;

// see the excel sheet behind the program
                app.Visible = true;

// get the reference of first sheet. By
      default its name is Sheet1.
// store its reference to worksheet
worksheet = workbook.Sheets["Sheet1"];
worksheet = workbook.ActiveSheet;

// changing the name of active sheet
worksheet.Name = "Exported from Student";

// storing header part in Excel
for (int i = 1; i < dataGridView1.Columns.Count + 1; i++)
                {
worksheet.Cells[1, i] = dataGridView1.Columns[i -
1].HeaderText;
                }

for (int i = 0; i < dataGridView1.Rows.Count - 1; i++)
                {
for (int j = 0; j < dataGridView1.Columns.Count; j++)
                    {
worksheet.Cells[i + 2, j + 1] =
dataGridView1.Rows[i].Cells[j].Value.ToString();
                    }

// storing Each row and column value to
      excel sheet
for (i = 0; i < dataGridView1.Rows.Count; i++)
                    {
for (int j = 0; j < dataGridView1.Columns.Count; j++)
                        {
worksheet.Cells[i + 2, j + 1] =
dataGridView1.Rows[i].Cells[j].Value.ToString();
                        }
                    }
                }
            }
catch
            {

            }
        }
```

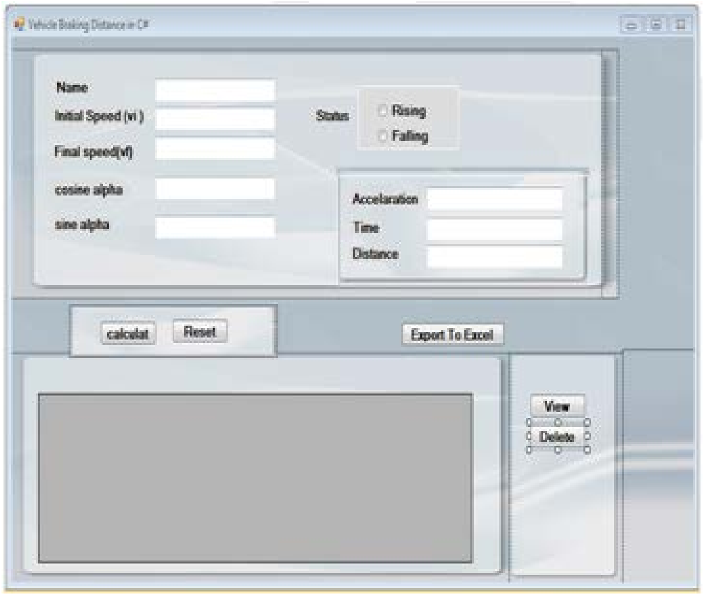
Let's Program To View Button



}

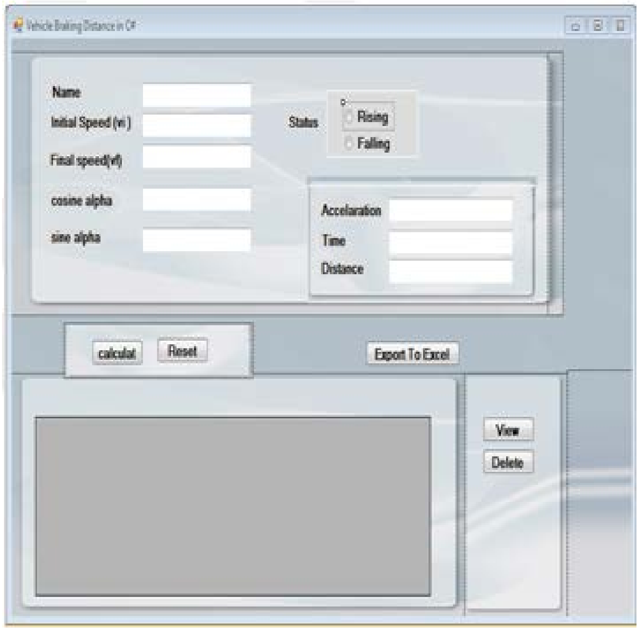
```
privatevoid button1_Click(object sender, EventArgs e)
        {
displayResult();

        }
}
```

Let's Program To Delete Button



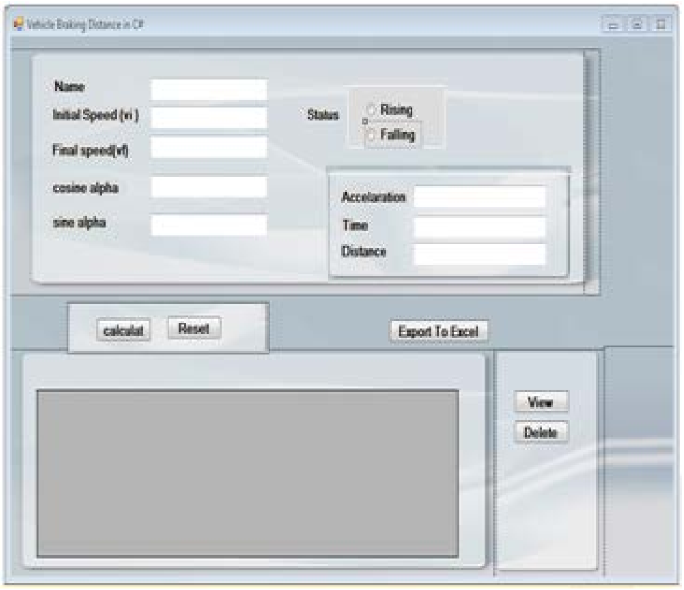
```
}

privatevoid btndelete_Click(object sender, EventArgs e)
        {
this.panel1.Visible = true;
this.btnexport.Visible = true;
if (connection.State == ConnectionState.Closed)
connection.Open();
DataSet ds = newDataSet();
DataTabledt = newDataTable();
ds.Tables.Add(dt);
OleDbDataAdapter da = newOleDbDataAdapter();

da = newOleDbDataAdapter("DELETE * from CALCULATE", connection);
da.Fill(dt);
            dataGridView1.DataSource = dt.DefaultView;
connection.Close();

displayResult();
        }
    }
}
```

Let's Program To Rising Button



```
}

privatevoid radiobtnRising_CheckedChanged(object sender, EventArgs e)
{
```

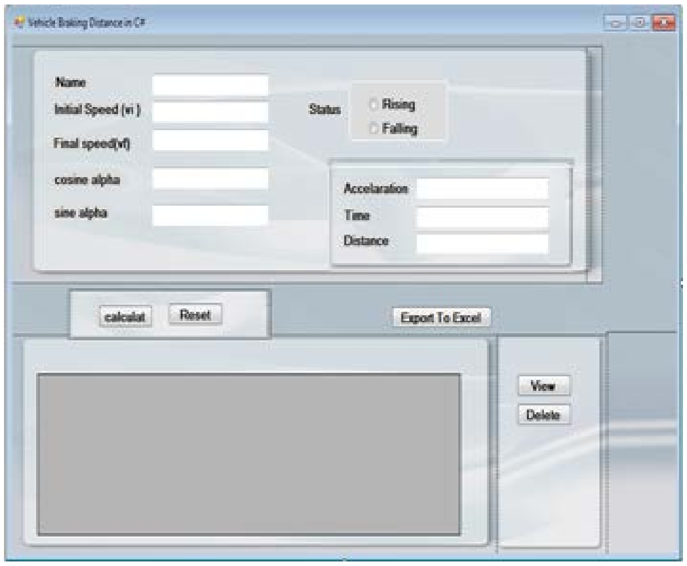
Let's Program To Falling Button



```
}

privatevoid radiobtnFalling_CheckedChanged(object sender, EventArgs e)
{
```

IV. CONCLUSION

The Finished product



ACKNOWLEDGMENT

I am using this opportunity to express my gratitude to everyone who supported me throughout the course of this research paper. I am thankful for their aspiring guidance, invaluably constructive criticism and friendly advice during this paper work. I am sincerely grateful to them for sharing their truthful and illuminating views on a number of research issues.

REFERENCES

[1] David Barney, David Haley and George Nikandros, Signal and Operational Systems, Queensland RailPO Box 1429,Brisbane 4001,Queensland, Australia.

[2] RAILTRACK PLC (Safety & Standards Directorate), Railway Group Standard (May 1996), Lineside Signal Spacing, GK/RT0034, Issue 1.0, May 1996, London.

[3] CENELEC (European Committee for Electro technical Standization EN 50128, Railway Applications --- Software for Control and Protection Systems.

[4] Fundamental of computer programming with C#.

AUTHORS

First Author – Andinet Kumella, M.Sc in Mechanical Engineering under Railway Engineering, Addis Ababa Institute of Technology(AAiT), andinet_kumella@yahoo.com/ andinetkumella48@gmail.com