PC based new software developed to create an input pilot balloon data file to an alternative to Hand Held Data Logger (HHDL) for using PC based SAMEER Pibal computation software

N. MEENATCHI NATHAN, CHANABASANAGOUDA. S. PATIL and J. P. IMMANUEL JAYAPRAKASH

Aerodrome Meteorological Office, Chennai, India

(Received 3 October 2013, Accepted 7 September 2015)

e mail : nathan.imd@gmail.com

1. Introduction

The pilot balloon observations, whose observations dates back to early 20th century, are still used today in mostly developing countries in Asia and Africa. Although PB observations are having limitations due to cloudiness, their low cost has historically allowed for more frequent and widespread used than radio sonde.

Various instruments and methods are used to measure wind, an inevitable parameter in meteorology and forecasting weather, at surface and at upper air level. Pilot balloon observation is one of the techniques of upper air wind measurement. A spherical rubber balloon, filled with hydrogen gas, rises due to its buoyancy forces and gets drifted by wind when it is released. The balloon is followed with the help of optical theodolite and its horizontal and vertical angles are noted with the graduated scale fixed on the theodolite. At the interval of every minute the height of the balloon above ground is calculated either assuming fixed rate of ascent or with the help of tail attached to the balloon. The angular position of
the balloon is determined in terms of two quantities viz., azimuth and angle of elevation. The azimuth angle is considered horizontally from the true north and the angle of elevation measures the position in the upward direction. The azimuth angle is denoted by H and the angle of elevation is by ‘V’. In the field of view of the telescope of the theodolite the flag readings are noted which are known as graticule reading. With the help of these three parameters the computational work is done to get the height drift of the balloon at every interval manually by drawing the trajectory and using suitable velocity scale for measuring the wind velocity.

In 2007 under modernization programme of IMD, all the PB stations are supplied with micro controller based Hand Held data logger. The Hand Held Data Logger (HHDL) is a sleek, microcontroller based battery operated unit for processing Pilot balloon data. Using Pilot balloon flight data, the HHDL computes wind speed and direction at 1-minute interval, at standard height and standard pressure levels. At the end, it also generates the Pilot message. The operator can print the processed data in the tabular form on the serial printer.

It has built-in battery backed memory to archive 100 numbers of the flight data. HHDL Accessories are RJ-9 & RJ-11 Coiled cable, RJ to D type 9 pin connector PC cable, Printer cable, Battery charger, three numbers of rechargeable batteries, etc. The theodolite observed parameters are keyed to the HHDL through Alpha numeric keypad which contains total 24 keys which are used for entering alphanumeric values. Various combinations of these keys are used for entering data, processing, connecting to PC etc. Hence these operations are time consuming and needs a good amount of practice for using the data logger. Moreover, these accessories and data logger is getting old, gives frequent problems and become unserviceable in some stations. SAMEER has supplied a Pibal Data Processing software (PC) to process the HHDL raw data files after transferring in to PC. This PC based software is capable of generating Pibal code, NDC data archival format and monthly climate etc. If the HHDL goes unserviceable then (PC based data processing software) this cannot be used and the total Pibal computation has to revert back to Manual computations.

These drawbacks may be overcome by using newly developed PC based software for pilot data inputting using visual studio built in language visual C#. Visual Studio is an Integrated Development Environment (IDE) from Microsoft. It is used to develop console and graphical user interface along with windows form applications etc. The Visual Studio built-in tools include a forms designer for building Graphical User Interface (GUI) applications, web designer, class designer and database schema designer etc. Visual Studio includes a full text generation framework called T4 which enables Visual Studio to generate text files from templates either in the IDE or via code. Using these features GUI based PB data input forms created and a suitable text file has been created for using the PC based PB computation software supplied by SAMEER.

2. Data and methodology

Pibal computation needs various information such as station details, balloon and attachments, preflight, post flight cloud information and theodolite angle information’s of the flight at every interval of one minute. The visual studio GUI tool consists of various options of text box, combo box, date time picker, Label, and so on. Different information collected through a tab control
windows form application, which gives easy access to moves around. The hierarchy of data feeding explained in the flowchart.

2 (a). Station information

The station information consists of basic details of the station which is used to identify the station as well as wind computation. The station information gives default values for specific stations, correction if needed; it permits the correction and formats the inputs automatically. The sample programme explained for pillar height. The pillar height, being a floating point number, has to accept only the numbers with one decimal point and not an alphabet. Hence, the key press event & mouse leave event are programmed, as follows, for accepting the inputs and automatically converts to the specified formats.

```csharp
private void PillarHtASL_KeyPress(object sender, KeyPressEventArgs e)
{
    // string oldText = string.Empty;
    if ((char.IsControl(e.KeyChar)
     && !char.IsDigit(e.KeyChar)
     && e.KeyChar != '.')
     {
        e.Handled = true;
    }
    // only allow one decimal point
    if (e.KeyChar == '.')
       && (sender as TextBox).Text.IndexOf('.') > -1)
    {
        e.Handled = true;
    }
}
```

2 (b). Balloon and attachments

The balloon and type of attachment information is important for deciding the fixed rate of ascent for night and tail attachments for day time wind calculation. This window provides an option to select the type of attachment as lantern for night and flag for day time. If flag is selected, other relevant information pertaining to the flag such as length of the different flags, drum number of the flags, flag change over time for graticule measurements are needed. Depending on the type of attachments, different data grid view created for day and night observations. The coding is as follows:

```csharp
private void BalloonAttachType_SelectedIndexChanged(object sender, EventArgs e)
{
    if (BalloonAttachType.Text.Contains("F"))
    {
        FlagInformation.Enabled = true;
        dataGridView2.Enabled = true;
        dataGridView2.Visible = true;
        dataGridView1.Enabled = false;
        dataGridView1.Visible = false;
    }
    else
    {
        FlagInformation.Enabled = false;
        dataGridView2.Enabled = false;
    }
```
dataGridView2.Visible = false;
dataGridView1.Enabled = true;
dataGridView1.Visible = true;
}

2. (c). Cloud information

The preflight and postflight window gives the surface wind, visibility and various cloud information before and after the ascent. This is achieved using combo box; it displays an editable text box with a dropdown list of permitted values. If the sky condition is partly cloudy then only it permits to enter the other cloud information otherwise totally disabled the cloud information. The following is the sample coding and Preflight window.

```csharp
private void SkyStausBA_SelectedIndexChanged(object sender, EventArgs e)
{
    if (SkyStausBA.Text.Contains("PRC"))
    {
        PreFlightCloudInfo.Enabled = true;
    }
    else
    {
        PreFlightCloudInfo.Enabled = false;
        PrC1Type.SelectedIndex = 0;
        PrC1Amt.SelectedIndex = 0;
        PrC1Dir.SelectedIndex = 0;
        PrC1Ht.SelectedIndex = 0;
    }
}
private void PoC1Type_SelectedIndexChanged(object sender, EventArgs e)
{
    if (PoC1Type.Text.Contains("AB"))
    {
        PoCloud1.Enabled = false;
        PoCloud2.Enabled = false;
        PoCloud3.Enabled = false;
        PoCloud4.Enabled = false;
        PoCloud5.Enabled = false;
        PoC2Type.Enabled = false;
        PoC3Type.Enabled = false;
        PoC4Type.Enabled = false;
        PoC5Type.Enabled = false;
    }
    else
    {
        PoCloud1.Enabled = true;
        PoCloud2.Enabled = false;
        PoCloud3.Enabled = false;
        PoCloud4.Enabled = false;
        PoCloud5.Enabled = false;
        PoC2Type.Enabled = true;
    }
}
```

2. (d). Angle values of theodolite azimuth/elevation data

On completion of the pre-ascent data entry, it is ready to accept balloon position data at one minute interval. The balloon position is defined in terms of graticule, azimuth and elevation angles for day time flight and no graticule values for night ascent (assuming uniform rate of ascent (9.0 km/hr)). This data is used to compute the wind speed and direction. This data entry is always starts from 0.5 minutes of elapsed time, and ready to accept Azimuth, Elevation and Graticule values in the data grid view window. The data grid view is configured to accept the angle values of the Pibal information's sequentially with auto increment of the Elapsed time with one minute interval using visual C# Event handler of the key press and row added event.

```csharp
private void dataGridView2_KeyPress(object s1, KeyEventArgs sd)
{
    if (sd.KeyCode == Keys.Enter)
    {
        int c = this.dataGridView2.CurrentCell.
        ColumnIndex;
        int ct = this.dataGridView2.ColumnCount - 1;
        int r = this.dataGridView2.CurrentRow.Index;
        if (c == ct)
        {
            if (r >= 0)
                this.dataGridView2.CurrentCell =
                dataGridView2.Rows[r].Cells[0 + 1];
        }
    }
    else
    {
        PoCloud1.Enabled = true;
        PoCloud2.Enabled = false;
        PoCloud3.Enabled = false;
        PoCloud4.Enabled = false;
        PoCloud5.Enabled = false;
        PoC2Type.Enabled = true;
```
2. (e). **File creation and saving**

Data collected through various windows needs to be saved in text document, using stream writer and save file dialog. The file location may be chosen at the time saving by the user. Sample coding for saving the file as follows.

```csharp
private void dataGridView2.RowsAdded(object sender, DataGridViewRowsAddedEventArgs e)
{
    for (int i = 0; i < e.RowCount; i++)
    {
        if (e.RowIndex == 1)
        {
            dataGridView2.Rows[e.RowIndex - 1].Cells[0].Value = (e.RowIndex - 1) + ".5".ToString();
        }
        else
        {
            dataGridView2.Rows[e.RowIndex - 1].Cells[0].Value = (e.RowIndex - 1) + ".0".ToString();
        }
    }
}

private void Save_Click(object sender, EventArgs e)
{
    string saved_File = "";
    saveFD.InitialDirectory = "C:";
    saveFD.Title = "Save a Text File";
    saveFD.FileName = "";
    saveFD.Filter = "Text Files|*.PBL| Text Files|*.txt";
    if (saveFD.ShowDialog() == DialogResult.Cancel)
    {
        MessageBox.Show("Operation Cancelled");
    }
    else
    {
        saved_File = saveFD.FileName;
        objWriter = new System.IO.StreamWriter(saveFD.FileName);
        //Station Information
        objWriter.WriteLine(SName.Text);
        objWriter.WriteLine(Index.Text);
        objWriter.WriteLine(SHeight.Text);
        if (BalloonAttachType.Text.Contains("F")
        {
            objWriter.WriteLine(dataGridView2.Columns[0].HeaderText + "\t" + dataGridView2.Columns[1].HeaderText + "\t" + dataGridView2.Columns[2].HeaderText + "\t" + dataGridView2.Columns[3].HeaderText);
            int rowcount = dataGridView2.Rows.Count;
            for (int i = 0; i < rowcount - 1; i++)
            {
                objWriter.WriteLine(dataGridView2.Rows[i].Cells[0].Value.ToString() + "\t" + dataGridView2.Rows[i].Cells[1].Value.ToString() + "\t" + dataGridView2.Rows[i].Cells[2].Value.ToString() + "\t" + dataGridView2.Rows[i].Cells[3].Value.ToString());
            }
        }
        objWriter.Close();
    }
```
MessageBox.Show("Data Successfully Exported");
}
}

3. Result and discussion

Newly developed software has tested at various stages of the development and with various degrees of rigour for fulfilling the computation software formats. This software has successfully created an input file, and tested with SAMEER Pibal computation software. Day and night ascent real time data are tested for more than three months at Chennai. It is observed that it is very user friendly than the HHDL and successfully creating an input file. Copy of the software has given to M.C. Hyderabad (meteorological observatory) for their use since HHDL is out of service. This software will be a good standby/alternate for the HHDL in case of failure. No limitations are found for archival/storage of the input files and processed files because it is stored directly in the system. This software is portable to any computer system with windows 7 or windows XP with visual studio updated versions or any higher versions of windows OS. Fig. 1 to Fig. 5 explains the steps to use this software. Simply run it for entering the Pibal input data and creating the file. This will solve the problem of reverting back to manual computation incase of failure of HHDL Hardware at all PB Observatories.

Acknowledgment

We are thankful to Dr. R. Suresh, Director in-charge of AMO Chennai for his helpful suggestions and encouragement in developing and testing the software. Also we thank all the staffs of observatory, Meenambakkam for extending their help for testing this software using the PB real time data’s.

Reference

SAMEER make Pibal Hand Held Data Logger for IMD User manual

www.homeandlearn.co.uk/csharp/