Development of Real Time Solar Spectrum Measurement Tool using Opto-electronic Sensors with help of Arduino/Raspberry Pi and Python GUI

submitted in partial fulfilment of the requirement of
the degree of

Bachelor of Engineering

by

Mihir Rahate	**54**
Madhura Shinde	**69**
Harsh Rajendra	**29**
Pratik Shetty	**68**

Under the supervison of

Prof. Lakshmi Iyer (DBIT)

Dr. Hemant Kumar Singh (IITB)

Department of Electronics and Telecommunication

Don Bosco Institute of Technology, Mumbai–400 070

University of Mumbai

2019-20

Don Bosco Institute of Technology
(Affiliated to the University of Mumbai)
Premier Automobiles Road, Kurla, Mumbai - 400070

Certificate

This is to certify that the project entitled **"Development of Real Time Solar Spectrum Measurement Tool using Opto-electronic Sensors with help of Arduino/Raspberry Pi and Python GUI."** is a bonafide work of

Mihir Rahate	**54**
Madhura Shinde	**69**
Harsh Rajendra	**29**
Pratik Shetty	**68**

submitted to the University of Mumbai in the partial fulfilment of the requirement for the award of the degree of **"Undergraduate"** in **"Bachelor of Engineering"**This approval does not necessarily endorse or accept every statement made, opinion expressed or conclusion drawn as recorded in the report. It only signifies the acceptance of the report for the purpose for which it has been submitted..

Prof. Lakshmi Iyer
Project Guide

Dr. Hemant Kumar Singh
Project Guide

Dr. Ashwini Kotrashetti
HOD, EXTC Department

Dr. Prasanna Nambiar
Principal

Don Bosco Institute of Technology
(Affiliated to the University of Mumbai)
Premier Automobiles Road, Kurla, Mumbai - 400070

Project Report Approval for B.E.

This project report entitled **'Development of Real Time Solar Spectrum Measurement Tool using Opto-Electronic Sensors with help of Arduino/Raspberry Pi and Python GUI'** by **Mihir Rahate, Harsh Rajendra, Madhura Shinde and Pratik Shetty** is approved for the degree of **Bachelor of Engineering in Electronics & Telecommunication.**

Examiners

1. _____

2. _____

Date:
Place: Mumbai

Declaration

We declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, We have adequately cited and referenced the original sources. We also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea / data / fact / source in my submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

_____ _____

Mihir Rahate **Madhura Shinde**

_____ _____

Harsh Rajendra **Pratik Shetty**

Date:
Place: Mumbai

Acknowledgement

A project is a teamwork which involves the contribution of many people. We would like to thank everyone who have contributed by taking interest in our work and motivating us all the way through. Sincere thanks to our project guide Prof. Lakshmi Iyer, for motivating, co-operating and guiding us throughout the project work, with her effective skills and huge knowledge base. We would like to thank Dr. Ashwini Kotrashetti, Head of Department and Prof. Khan Naheed Anjum, Project Coordinator for their continuous valuable guidance, support, suggestions and their precious time in every possible way throughout the project activity. We would like to thank Dr. Hemant Kumar Sigh, our external project guide, for his continued guidance and support in helping us out with the troubleshooting of errors.

Also we are very grateful to the python community forums.

Date :
Place : Mumbai

Abstract

The Sun provides a broad range of energy, primarily concentrated around the visible and infrared regions. This energy is an important feature of the back- ground conditions that led to the evolution of life forms on our Earth, and continue to support this life. The measure of solar radiation spectrum is needed for identification and evaluation of solar energy resources. This project proposes a methodology to plot the solar irradiance at different wavelengths using certain transmittance equations and The National Solar Radiation Data Base (NSRDB) data. The composition of the Solar irradiance will be studied. Terrestrial spectrum between 300nm and 4000nm with a resolution of approximately 10nm to 20nm will be stored in a CSV file which will be created by python code itself. By making use of python which is an Open source tool, GUI will be created and the Solar spectrum will be plotted on the monitor.

List of Abbreviations

λ	Wavelength
$I_{T\lambda}$	Spectral global irradiance on horizontal surface
$I_{S\lambda}$	Total Scattered Irradiance
$I_{r\lambda}$	Rayleigh Scattering Component
$I_{a\lambda}$	Aerosol Scattering Component
$I_{g\lambda}$	Component for multiple reflections between ground and air
$I_{D\lambda}$	Direct Irradiance
D	Earth-Sun Distance
a	Day Angle
$T_{R\lambda}$	Transmittance after Rayleigh Scattering
M	Relative Air Mass
M'	Pressure corrected air mass
$T_{A\lambda}$	Transmittance after Aerosol Scaterring and Absorption
$T_{W\lambda}$	Transmittance after Water Absorption
$a_{W\lambda}$	Water Vapor Absorption Coefficient
$T_{O\lambda}$	Transmittance after Ozone Absorption
O_3	Ozone Amount(atm-cm)
M_o	Ozone Mass
h_o	Height of maximum ozone concentration
$T_{U\lambda}$	Transmittance after uniformly mixed gases absorption
$T_{AA\lambda}$	Transmittance after Aerosol Absorption
$T_{AS\lambda}$	Transmittance after Aerosol Scattering
$R_{G\lambda}$	Ground albedo as a function of wavelength
$R_{S\lambda}$	Sky Reflectivity

List of Figures

Contents

Chapter 1

Introduction

1.1 Background

Primary importance has been given to the development of alternate renewable energy sources due to the increasing demand of energy and depleting fossil fuels. The best alternative that has emerged is solar energy not only due to its sustainability but also because it is environment friendly and an abundant energy source. The applications of solar energy are endless such as agriculture, electricity generation, heating and cooking etc. Thus, solar energy harvesting and utilization has been assigned a higher priority by many countries.

1.2 Motivation

Intensive knowledge of the solar radiation incident on the earth's surface is of utmost importance for engineers, scientists and architects associated with the design of solar energy systems.In this project we developed a model to plot solar irradiance. This plot of irradiance will help with the design of solar energy systems, one can analyse and understand the solar composition of that particular area.

1.3 Problem Statement

To plot the solar irradiance at different wavelengths using transmittance equations and National Solar Radiation Data Base (NSRDB) data between 300 nm and 4000 nm with a resolution of approximately 10 nm to 20 nm and creating a GUI using Python.

1.4 Objectives of the Project

1. To learn Raspberry Pi, Arduino and Python software and their interfacing with each other
2. To study different transmittance equations for Rayleigh scattering, Aerosol scattering and absorption, Water Vapour absorption and Ozone and uniformly mixed gas absorption
3. To develop a model using equations for Global Horizontal Irradiance (GHI) in order to plot the solar irradiance
4. To ensure that the model developed can calculate and plot Global Horizontal Irradiance on any day of the year at any local time
5. To plot the recorded Global Horizontal Irradiance from the NSRDB (The National Solar Radiation Data Base) data

Chapter 2

Literature Survey

The following papers were referred while doing this project:

[1] Richard E. Bird and Carol Riordan," Simple Solar Spectral Model for Direct and Diffuse Irradiance on Horizontal and Tilted Planes at the Earth's Surface for Cloudless Atmosphere", Solar Energy Research Institute, Golden, CO, 80401, 26 July 1985.

This paper provides us a model which has some advancement in the algorithm to calculate direct normal and diffuse horizontal spectral irradiance at the earth's surface for the condition of cloudless sky. For calculation of direct normal irradiance factors such as distance between earth-sun, water vapour transmittance, absorption coefficients, ozone mass were considered. For diffuse irradiance on a horizontal surface, components like Rayleigh scattering, aerosol scattering and reflection of irradiance between ground and air and for incluned surfaces three algorithms by Hay Davies, Temps and Coulson, Klucher were used.

[2] Shanmuga Priya Selvanathan , Norman Carl Freudenberg, and Arunabh Borkataky, "Solar Radiation Measurement Using Raspberry Pi and it's Modelling using Artificial Neural Networks", MATEC Web of Conferences 77, ICMMR, 2016

This paper provides us details of solar radiation measurement using a Artificial Neural Network that is used for estimation of non-linear function, model prediction, etc. It also has information of a data logger which is instrument that records as well as reports the environmental changes over

time like humidity, tempertaure, solar radiation, etc.

[3] Christian A. Gueymard, "Direct solar transmittance and irradiance predictions with broadband models. Part I: detailed theoretical performance assessment", Solar Consulting Services, 174 Bluebird Lane, Bailey, CO 80421, USA, 13 May 2013.

This paper basically deals with the research on performance of broadband direct irradiance predictions using solar radiation models. For this research they have examined nineteen models. Errors were found in the modelling of different transmittance functions from different models and they tried to resolve those errors in most of the models. While examining these models additional two models were also developed which are good in predicting the clear sky DNI only from the sun's position and few atmospheric equations. They made a spectral radioactive code (SMARTS) which generated reference transmittance and DNI values. After this each model was tested in various atmospheric conditions using their equations and then were compared to the reference value generated by the SMARTS code. On the basis of their analysis they recommended four models(CPCR2,MLWT2,REST and YANG).

[4] Geoffrey Toon, "The Solar Spectrum: an Atmospheric Remote Sensing Perspective", Noble Seminar, University of Toronto, Oct 21, 2013.

This paper performs a through investigation to accurately quantify the composition of the Earth's atmosphere in order to understand the solar spectrum. To understand solar spectrum, they started analyzing the low-resolution solar spectrum and high-resolution solar spectrum. In this they have discussed some concepts which will be helping in determining solar spectrum or developing solar model.

Chapter 3

Methodology

3.1 Spectral Composition

Solar irradiance is the intensity with which radiation from the sun enters Earth's atmosphere. A relatable method to think about solar irradiance is by considering the difference between a 20-watt light bulb and a 100-watt light bulb. Both the bulbs produce visible light in the same wavelengths, but the brightness and intensity are very different. Solar irradiance is defined as the amount of radiant flux on an area, and is measured in watts per meter squared (W/m) [4].

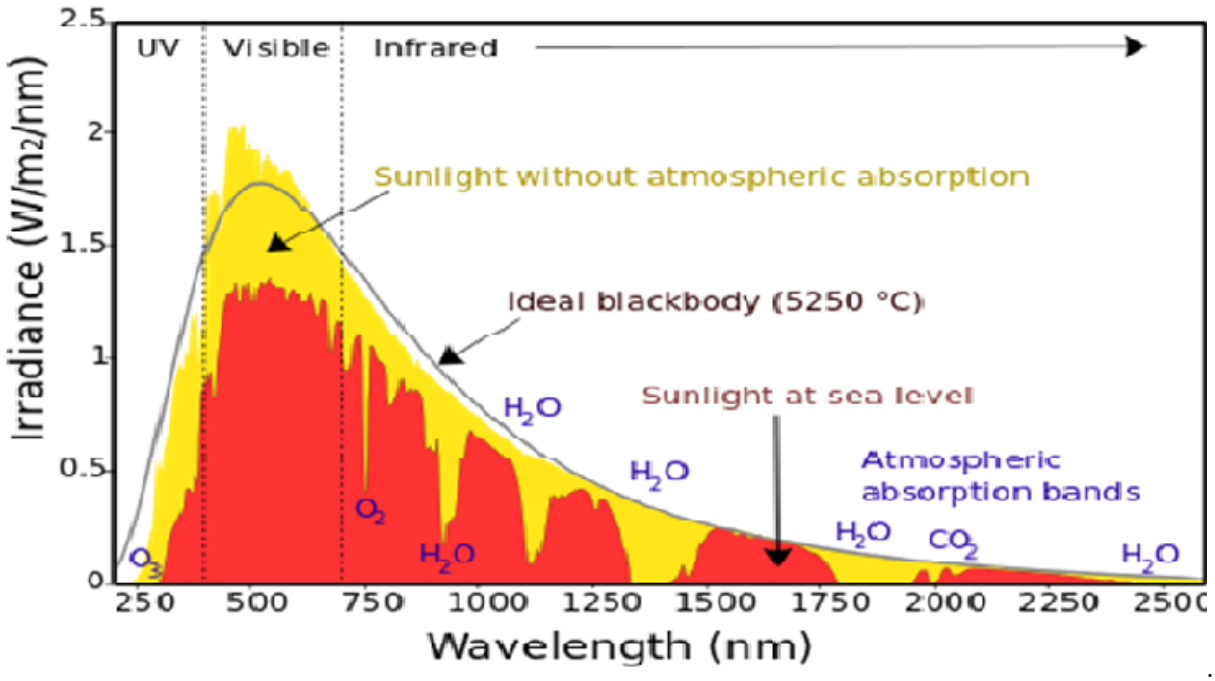

Figure 3.1: Plot section of GUI

The solar irradiance that is received by a particular location or body of water depends on various factors such as the elevation above sea level,

the angle of the sun and scattering elements such as clouds. The lower the angle of the sun, the larger amount of ozone light has to pass through it. Ozone has the ability to absorb UV light and can reduce radiation intensity. The angle of the sun is dependents on various parameters such as latitude, time of year, and time of day. The distance that radiation has to travel will be at its lowest value when the sun is exactly overhead. Other factors such as cloud coverage and air pollution can also reduce the amount of radiation that reaches Earth's surface. Clouds and aerosols in the atmosphere are responsible for scattering and absorbing all radiation bands. The fig. 3 above represents the solar spectrum composition which is as follows: 6 – 7 percent ultraviolet light; around 42 percent visible light and 51 percent near infrared.

3.2 Transmittance Equations

In order to plot the solar irradiance, a model is developed using equations for global horizontal irradiance. Richard E Bird and Carol Riordan [1] presents different equations that can be used for the model. The model is developed using Python. The model consists of transmittance equation for direct and diffuse irradiance on a horizontal surface.

The Spectral Global Irradiance on a horizontal surface is given by

$$I_{T\lambda} = \mathrm{I}_{D\lambda}\mathrm{Dcos(Z)} + \mathrm{I}_{s\lambda} \quad -----(1)$$

The diffused irradiance on a horizontal surface is divided into three components: 1)Rayleigh Scattering($\mathrm{I}_{r\lambda}$), 2)Aerosol scattering component ($\mathrm{I}_{a\lambda}$) and 3)the component that accounts for multiple reflection of irradiance between the ground and the air($\mathrm{I}_{g\lambda}$).
The total scattered irradiance $\mathrm{I}_{s\lambda}$ is given given by the sum,

$$I_{s\lambda} = \mathrm{I}_{r\lambda} + \mathrm{I}_{a\lambda} + \mathrm{I}_{g\lambda} \quad -----(2)$$

Expression for all these components are as follows:

$$I_{r\lambda} = \mathrm{H}_{o\lambda}\mathrm{Dcos(Z)T}_{O\lambda}\mathrm{T}_{U\lambda}\mathrm{T}_{W\lambda}\mathrm{T}_{AA\lambda}(1 - \mathrm{T}_{R\lambda}{}^{0.95})0.5 \quad -----(3)$$

$$I_{a\lambda} = \mathrm{H}_{o\lambda}\mathrm{Dcos(Z)T}_{O\lambda}\mathrm{T}_{U\lambda}\mathrm{T}_{W\lambda}\mathrm{T}_{AA\lambda}\mathrm{T}_{R\lambda}{}^{1.5}(1 - T_{AS\lambda})\mathrm{F}_S \quad -----(4)$$

Where the transmittance terms $\mathrm{T}_{AA\lambda}$ and $\mathrm{T}_{AS\lambda}$ are for aerosol absorption and aerosol scattering.

$$I_{g\lambda} = (\ \mathrm{I}_{D\lambda}\mathrm{cos(Z)} + \mathrm{I}_{r\lambda} + \mathrm{I}_{a\lambda}\ )\mathrm{R}_{S\lambda}\mathrm{R}_{G\lambda}\ /(1 - \mathrm{R}_{S\lambda}\mathrm{R}_{G\lambda}) \quad -----(5)$$

The parameter $\mathrm{R}_{G\lambda}$ is the ground albedo as a function of wavelength, $\mathrm{R}_{S\lambda}$ is the sky reflectivity.

$$R_{S\lambda} = \mathrm{T'}_{O\lambda}\mathrm{T'}_{W\lambda}\mathrm{T'}_{AA\lambda}\ [0.5(1 - \mathrm{T'}_{R\lambda}) + (1 - \mathrm{F'}_S)T'_{R\lambda}(1 - \mathrm{T}_{AS\lambda})] \text{——(6)}$$

In the next set of equations the parameters used in each of these three equations 1.1,1.2,1.3 is defined

$$T_{AS\lambda} = \exp(\text{-}\omega_\lambda\tau_\lambda\mathrm{M}) \quad -----(7)$$

$$AFS = ALG[1.459 + ALG(0.1595 + ALG0.4129)] \quad -----(8)$$

$$BFS = ALG[0.0783 + ALG * (-0.3824 - ALG0.5874)] \quad -----(9)$$

$$ALG = ln(1 - cos\theta) \quad -----(10)$$

$$F_S = 1 - 0.5exp[AFS + BFS/1.8)/1.8] \quad -----(11)$$

$$\omega_\lambda = \omega_0.4exp{-}\omega\text{'}[\ln(\lambda/0.4)]^2] \quad -----(12)$$

$\mathrm{I}_{D\lambda}$ is the direct irradiance on a surface to the direction of the sun at ground level for wavelength λ given by

$$I_{D\lambda} = \mathrm{H}_{O\lambda}\mathrm{DT}_R\mathrm{T}_{A\lambda}\mathrm{T}_{W\lambda}\mathrm{T}_{O\lambda T}U_\lambda \quad -----(13)$$

where,
D is the Earth-Sun distance factor which is given by,
D = 1.00011 + 0.034221 cos(a) + 0.00128 sin(a) + 0.000719 cos(2a) + 0.000077 sin(2a)

The day angle 'a' in radians is represented by
a = 2pi(d - 1)/365, where d is the day number of a year(1-365)

The Transmittance parameters used in the equation 3,4,5,13 are as fol-

lows with their equations:

1. Rayleigh Scattering:

$$T_{R\lambda} = \text{exp-M'}/[\lambda^4(115.6406 - 1.335/\lambda^2)] \quad -----(14)$$

$$M = [cosZ + 0.15(93.885 - Z)^{-1.253}]^{-1} \quad -----(15)$$

Z is solar zenith angle. The pressure corrected air mass is M' = MP/P$_O$, $where P_O$ = 1013 MB and P is measured surface in MB.

2. Aerosol Scattering and Absorption:

$$T_{A\lambda} = \text{exp(-(beta)}_n\lambda_n^{alpha}M) \quad -----(16)$$

$$\tau_{beta\lambda} = \text{beta}_n\lambda_n^{-alpha} \quad -----(17)$$

We have considered the value of alpha as 1.140

3. Water Vapor Absorption:

$$T_{W\lambda} = \text{exp[-0.2385a}_{W\lambda}\text{WM}/(1 + 20.07\text{a}_{W\lambda}\text{WM})^{0.45}] \quad -----(18)$$

W = perceptible water vapor(cm) in a vertical path.

$a_{W\lambda}$ = Water vapor absorption coefficient as a function of wavelength

4. Ozone and uniformly mixed gas absorption:

$$T_{O\lambda} = \text{exp(-a}_{O\lambda}\text{O}^3M_O) \quad -----(19)$$

$$M_O = (1 + h_O/6370)/[cos^2(Z) + 2h_O/6370]^{0.5} \quad -----(20)$$

h_O is the height of maximum ozone concentration, which is approximately 22 Km. The ozone height varies with latitude and time of year. Expression for uniformly mixed gas absorption:

$$T_{U\lambda} = \text{exp[-1.41a}_{u\lambda}\text{M'}/(1 + 118.93\text{a}_{u\lambda}\text{M'})^{0.45}] \quad -----(21)$$

Where $a_{u\lambda}$ is the combination of an absorption coefficient and gaseous amount.

Certain parameters such as day number, hour, longitude, latitude will be taken from the user as input.

3.3 NSRDB Data

The National Solar Radiation Data Base (NSRDB), consists of solar radiation and meteorological data over the United States and regions of the surrounding countries. It is a publicly open dataset.
Meteorological Data can be downloaded from this website for a particular area over the last 10 years. The NSRDB spectrum between 300 nm and 1800 nm with a resolution of 10nm to 20nm is stored in a CSV file.
In order to check the accuracy of the equations, we can compare the graph from equation with graph from NSRDB data.

Chapter 4

Development of Graphical User Interface (GUI)

4.1 Process Flow for GUI Development

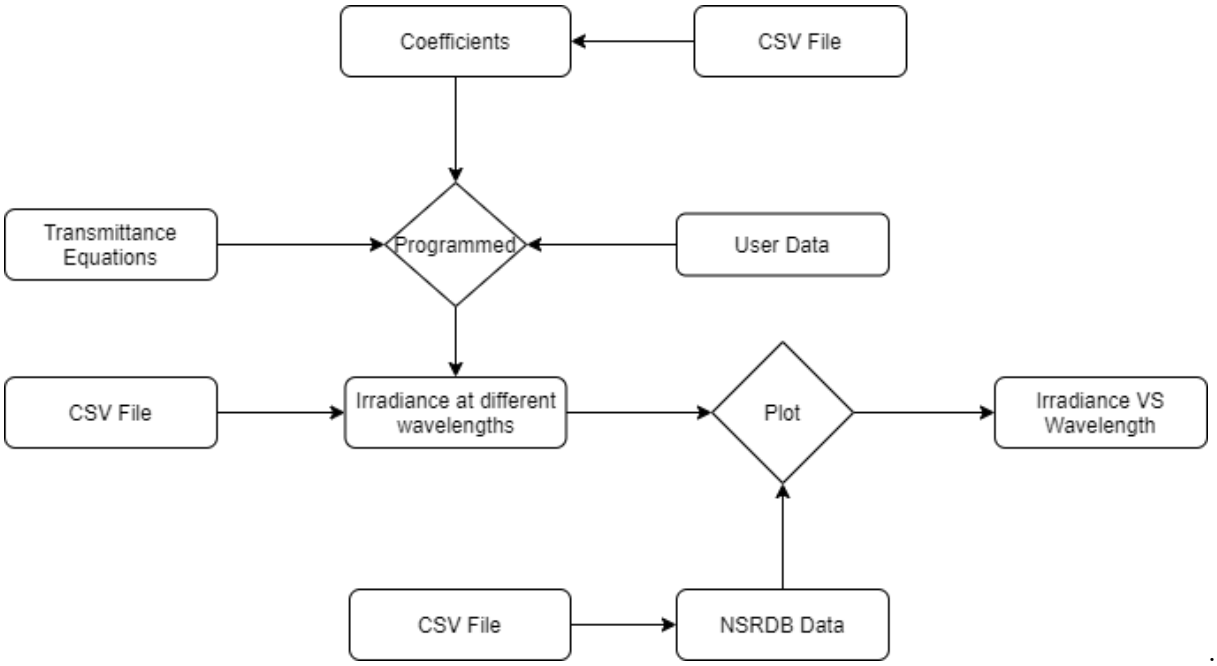

Figure 4.1: GUI development Flowchart

The flow diagram shown in Figure 4.1 presents the flow of the GUI. The coefficients used for the equations are stored in a csv file. The coefficients vary as per the wavelengths and so they are stored against the corresponding wavelength. Based on the user data obtained, the system is programmed to calculate the global horizontal irradiance(GHI) using the transmittance equations and coefficients stored in the csv file. The calculated GHI at different wavelength is then stored in a csv file with its corresponding wavelength. The NSRDB data containing the recorded

GHI at different wavelengths is also stores in the csv file. In order to plot the graph, the data from the csv file containing the NSRDB data or GHI calculated using equations is extracted. The final irradiance vs wavelength plot is obtained.

4.2 Solar Spectrum Simulator GUI

The final GUI of solar spectrum simulator is shown in Figure 4.2. In this GUI we can see there two sections for plotting graph "plot 1" and "plot 2", the GUI is designed in such a way that user can choose any plot. It can also select two plots simultaneously using "Plot 1" and "Plot 2".

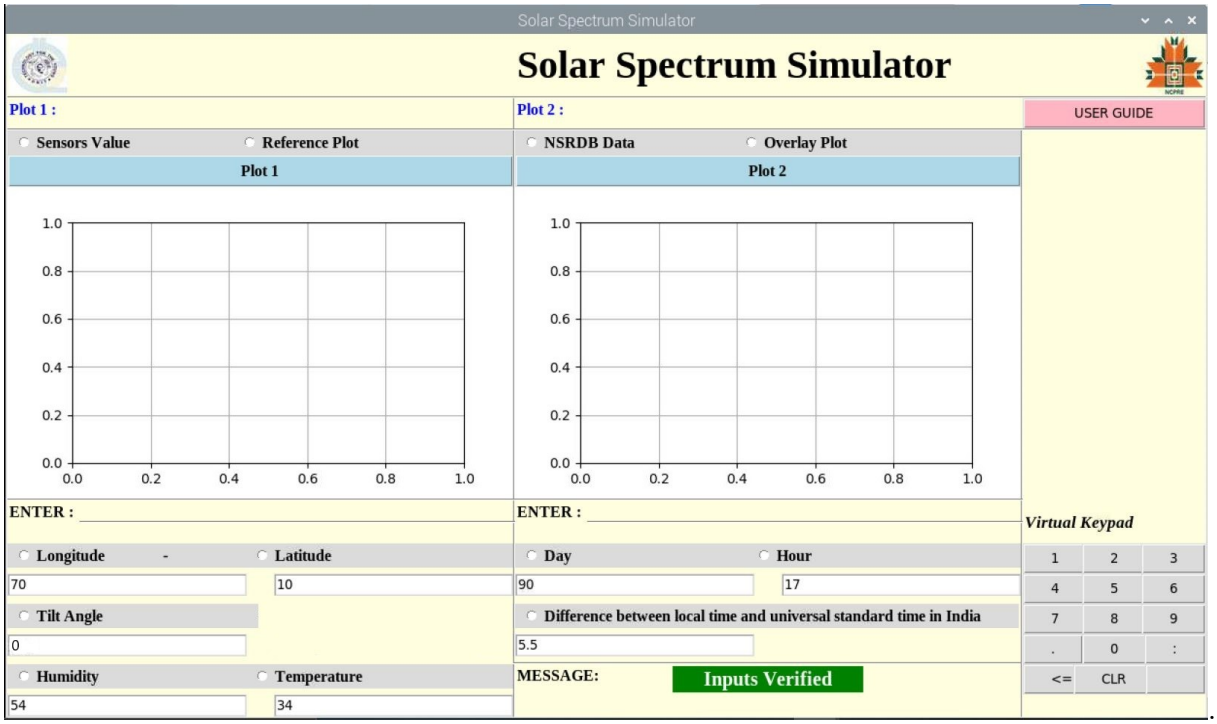

Figure 4.2: Plot section of GUI

Once the "Plot 1" or "Plot 2" button is pressed, the plot of that selected type would be displayed. In order to obtain the equations plot, the user has to enter certain inputs i.e. longitude, latitude, day number, hour, tilt angle, zenith angle and difference between local time and universal standard time in India as shown in Figure 4.3. Programming has been done in such a way that the global horizontal irradiance calculated from the equations is stored in their respective CSV file and can be used for data comparison and analysis with the NSRDB data. The data is saved in CSV

file. A new CSV file is created by the coded program for storing the mentioned values as the day or hour changes.

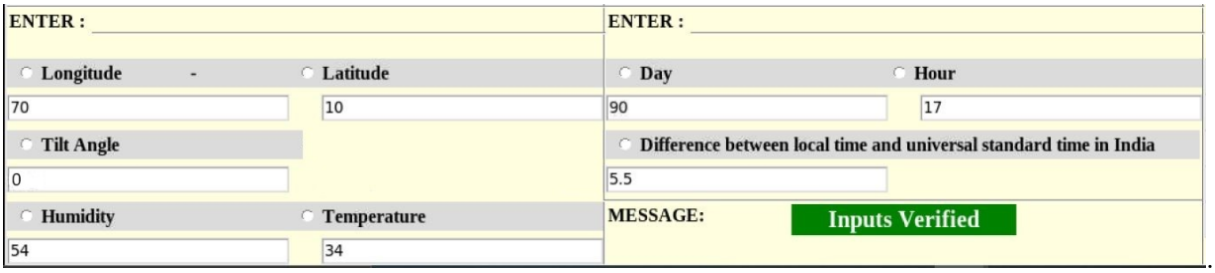

Figure 4.3: Inputs to be entered by the user

4.3 Plot 1 and Plot 2 Section of Solar Spectrum Simulator

As shown in Figure 4.2, there are two plot sections. Each plot section has two options provided for plotting graph. The plot 1 is used for plotting graph from sensors and equations(reference plot). While plot 2 is used for plotting NSRDB plot and overlay graph of reference plot and NSRDB plot. Each Plot section is explained in brief below.

- Sensors Value
  - To plot the graph using sensor values, select Sensors Value. This will be further explored in future for plotting live graph.
- Reference Plot
  - The Reference Plot is the plot obtained from the equations. In order to obtain the equations plot, the user has to enter certain inputs i.e. longitude, latitude, day number, hour, tilt angle, humidity, temperature and difference between local time and universal standard time in India.
- NSRDB Data
  - The NSRDB plot will plot the NSRDB data containing GHI values. In order to plot this graph, user has to enter difference between local time and universal standard time in India, the day

number and the hour of the day at which you want to plot the graph.

- Overlay Plot
  - The overlay plot will plot the reference plot and the NSRDB plot on the same canvas.

.

4.4 User Guide Section

User Guide is provided, which once clicked, will open a user manual having detailed instructions that would guide the user on how to use the system as shown in Figure 4.4.

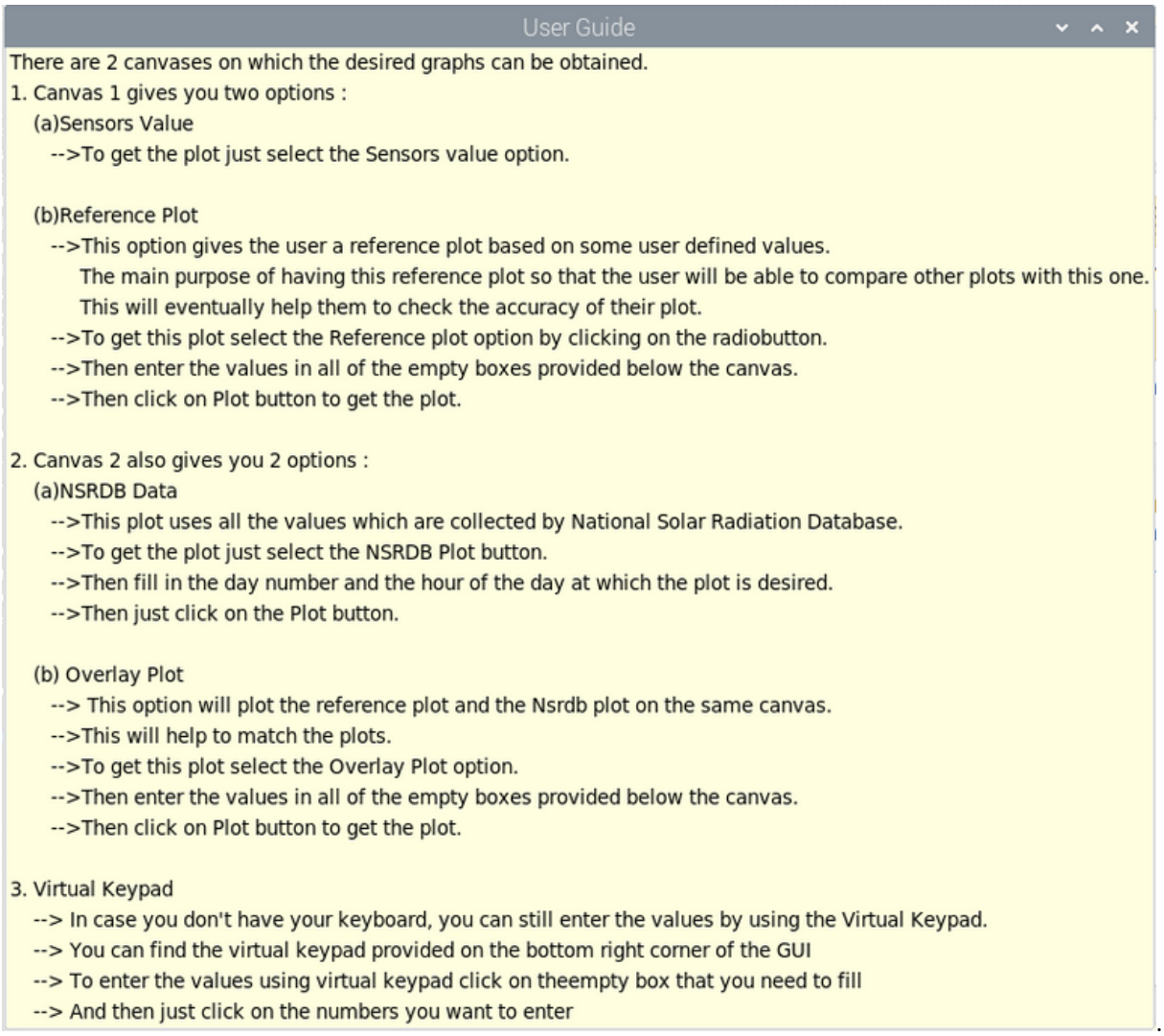

Figure 4.4: User Guide

Chapter 5

Experimental Results

5.1 NSRDB Database

The National Solar Radiation Data Base (NSRDB), consisting of solar radiation and meteorological data over regions of the countries , is a publicly open data set. From this website we can download meteorological data of a particular area for past 10 years. The NSRDB spectrum between 300 nm and 1800 nm with a resolution of 10nm to 20nm is stored in a .csv file format. In order to check the accuracy of the equations, we can compare the graph from equation with graph from NSRDB data.
Figure 5.1 shows the CSV file containing GHI values from NSRDB data.

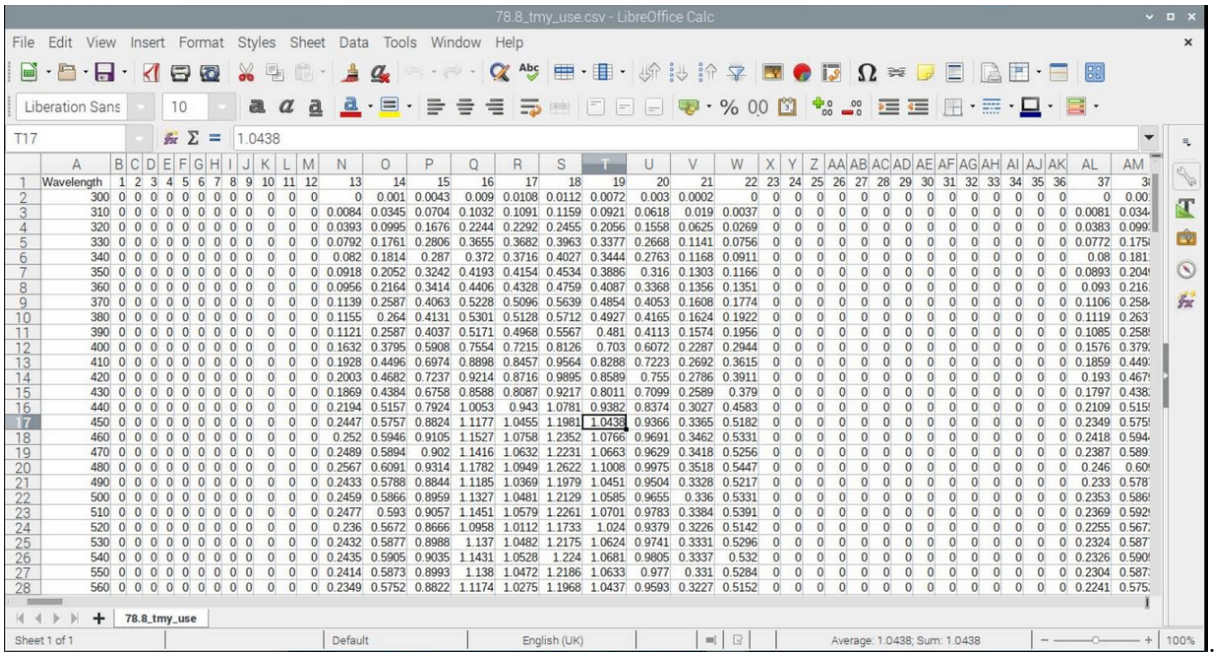

Figure 5.1: CSV file containing GHI values from NSRDB data

5.2 Plot using Transmittance Equations

The equations are used to obtain GHI at any hour of the day and any day of the year. The interface is made in a way that the user can enter the day number and hour of the day. Also, certain constant coefficients required for the equations are stored in a .csv file format. These coefficients are extracted from the csv file for wavelengths from 300nm to 4000nm with a resolution of 10nm to 20nm. Other inputs required such as latitude, longitude, tilt angle and difference between local time and universal standard time in India can be entered by the user. Once the inputs are entered and "Plot 1" button is pressed as shown in Figure 5.2, the plot of GHI values obtained from equations will be displayed as shown in Figure 5.2. The GHI values at different wavelengths are stored in a CSV file.

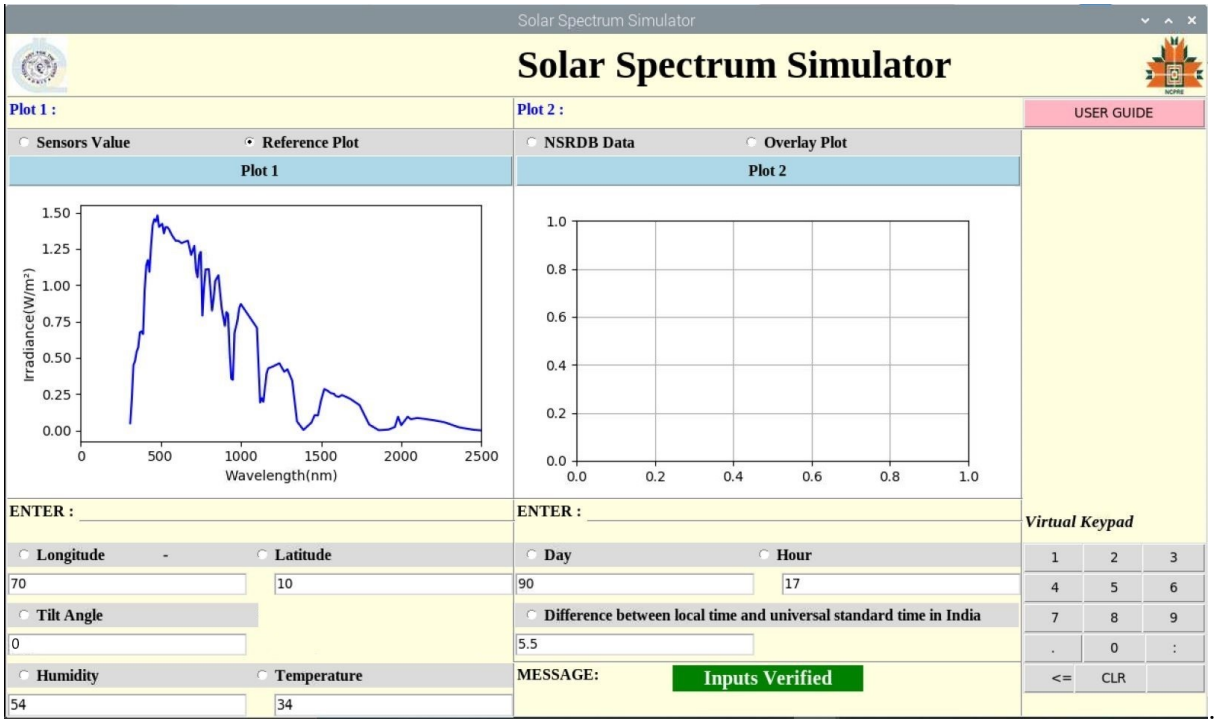

Figure 5.2: Plot of GHI obtained from Equations

5.3 Plot Using NSRDB data

To plot the graph from NSRDB data, the user just need to enter the day number and hour for which the user wants to plot the GHI. Once the "Plot 1" button is pressed, the plot of GHI from the NSRDB data will be displayed as shown in Figure 5.3.

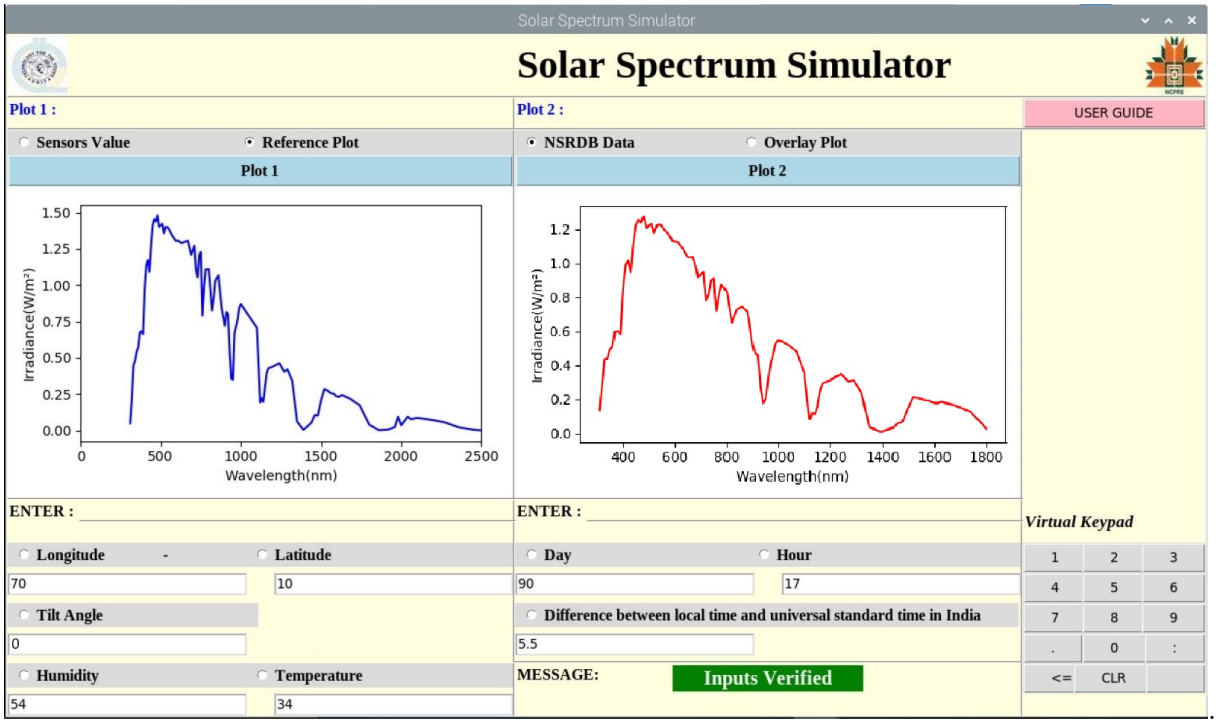

Figure 5.3: Plot of GHI obtained from NSRDB data

5.4 Overlay Plot

This type of plot is to provide a comparative analysis between NSRDB plot and Equations Plot. The NSRDB data consists of global horizontal irradiance for wavelengths between 300nm and 1800 nm with a resolution of 10nm to 20nm. The equations are used to obtain GHI at any hour of the day and any day of the year. The user can select the day number and the hour of the day for which the user wants a combined graph. Other inputs required such as humidity, temperature, latitude, longitude, tilt angle and difference between local time and universal standard time in India can be entered by the user. These inputs can also be entered using the virtual keypad.

Once the inputs are entered and "Plot 2" button is pressed, the plot of GHI values obtained from equations and GHI values from NSRDB data will be displayed as shown in Figure 5.4 This graph provides the accuracy of the GHI obtained from the equations as compared to the officially recorded GHI.The GUI is also capable of plotting two graphs simultaneously. The

Figure 5.4 shows the plot of GHI obtained from equations and NSRDB data.

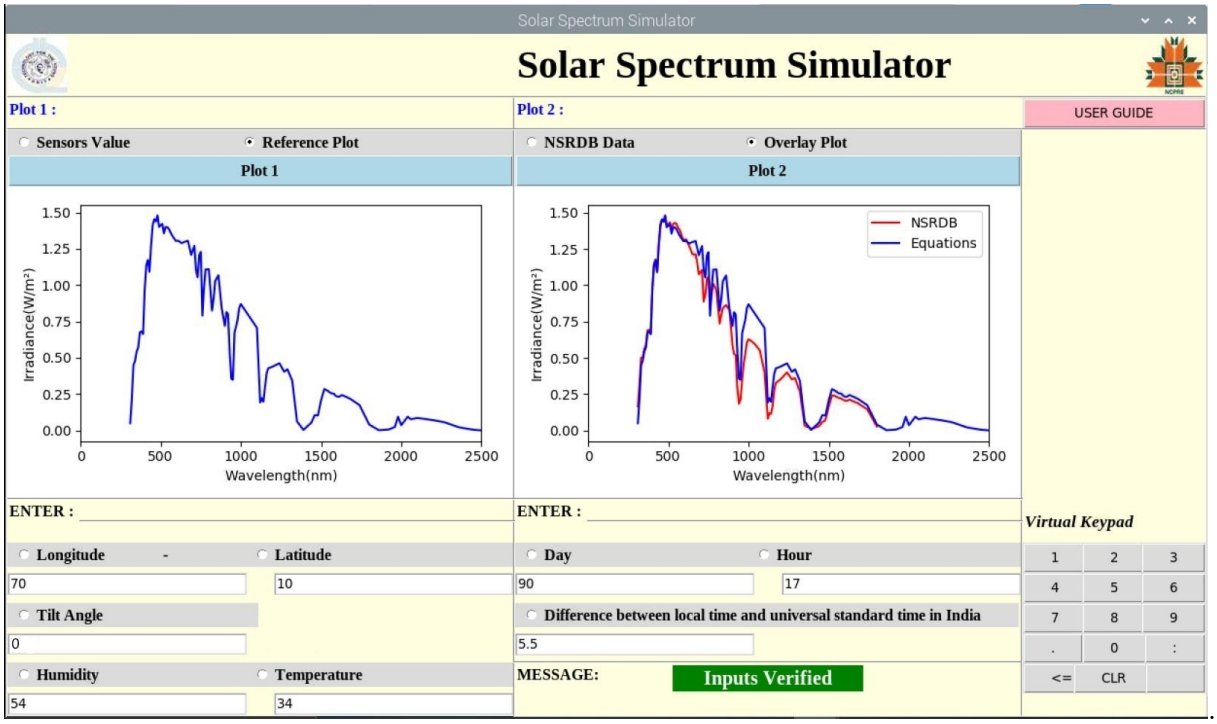

Figure 5.4: Overlay plot of GHI obtained from equations and NSRDB data

Chapter 6

Conclusion

Solar irradiance at different wavelengths was plotted by using the transmittance equations and NSRDB (National Solar Radiation Database). A GUI was created by using python coding. The data plotted was between wavelength 300nm to 4000nm having a approximate resolution of 10nm to 20nm. It will help us to plot the GHI for different days of a year depending on the type of plot selected by the user. The current simulator can provide the overlap of the two plots on any given day. In summary, the Solar Spectrum Simulator is suitably created GUI using Python Tkinter module and python programming for data handling and plotting at the backend. The Simulator can further be explored in future to extract live data from sensors and by using Arduino and Raspberry pi.

References

1. Richard E. Bird and Carol Riordan, "Simple Solar Spectral Model for Direct and Diffuse Irradiance on and Titled Planes at the Earth's Surface for Cloudless Atmospheres", Solar Energy Research Institute, Golden, CO 80401, 26 July 1985.

2. Shanmuga Priya Selvanthan, Norman Carl Freudenberg and Arunabh Borkataky , "Solar Radiation Measurement Using Raspberry Pi and its Modelling Using Artificial Neural Networks", ICMMR, 2016.

3. Christian A. Gueymard, "Direct Solar Transmittance and Irradiance Predictions With Broadband Models. Part I: Detailed Theoretical Performance Assesment", Solar Consulting Services, 174 Bluebird Lane, Bailey, CO 80421, USA, 13 May 2003.

4. Geoffrey Toon, "The Solar Spectrum: An Atmospheric Remote Sensing Perspective", Noble Seminar, University of Toronto, Oct 21, 2013.

5. Silicon Labs, "Si1145/46/47, Peoximity/UV/Ambient Light Sensor IC With I2C Interface", December 2013.

6. AMS, "AS7262 6-channel Visible Spectral ID Device with Electronic Shutter and Smart Interface", March 2017.

7. AMS, "AS7263 6-channel NIR Spectral ID Device with Electronic Shutter and Smart Interface", Nov 2016.

8. Vishay Semiconductors, "VEML6075, UVA and UVB Light Sensor with I2C Interface", 84304 Datasheet, Nov 2016.

APPENDIX

Appendix A

Source Code

```
from tkinter import *
import tkinter as tk
import matplotlib.pyplot as plt
from PIL import Image, ImageTk
import serial
from datetime import datetime, date
global file
global a
global b
global ser

a=date.today()
print (a)
while True:
   import math
   from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
   global temperature

   b=date.today()

   if a==b :

      print ("continue")

      global temperature
      global fig
      fig = plt.figure(figsize=(5.2,3.2))

      running = None  # Global flag

      root = Tk()
      root.title("Solar Spectrum Simulator")

      root.configure(bg='light yellow')

      ws = root.winfo_screenwidth() # width of the screen
      hs = root.winfo_screenheight()

      def welcome():
          print("Welcome")

#####################################

      def start():
         """Enable scanning by setting the global flag to True."""
         global running
         running = True
         root.after(1000, scanning)

      def stop():
         """Stop scanning by setting the global flag to False."""
```

```
52          global running
53          running = False
54
55      def equations():
56
57          import math
58          import numpy as np
59          import matplotlib.pyplot as plt
60          import csv
61          import pandas as pd
62
63          day1 = int(day.get());
64
65          tilt1 = float(tilt.get())
66
67          delta1 = float(delta.get());
68
69          hour1 = int(hour.get())
70
71          longitude1 = float(longitude.get());
72
73          latitude1 = float(latitude.get())
74
75          humidity1 = float(humidity.get())
76
77          temperature1 = float(temperature.get())
78
79          temp=(2*(math.pi)*(day1-1))/365;
80
81  ####### extraction of Zenith #######
82          f1= (int(day.get()) - 1) * 24
83          f2= (int(hour.get())) + f1+5
84          print(f2)
85          f3 = str(f2)
86
87          data = pd.read_csv('zenith_use.csv')
88
89          # calling .iat[] method
90          zenith1 = data.iat[f2,1]
91
92          # display
93          print(zenith1)
94
95          LSTM=15*delta1;
96          print('LSTM=',LSTM);
97          B=23.45*math.sin((360/365)*(284+day1))
98          EoT=9.87*math.sin(2*B)-7.53*math.cos(B)-1.5*math.sin(B)
99          print('Eot=',EoT)
100
101         TC=4*(longitude1-LSTM)+EoT;
102         print('TC=',TC);
```

```
103
104         LST=hour1+(TC/60)
105         print('LST=',LST)
106         HRA=15*(LST-12)
107         if HRA<0:
108             HRA=HRA+360
109             if HRA>360:
110                 HRA=HRA-360
111         print('HRA=',HRA)
112         ang_delta=23.45*math.sin(B)
113
114         alpha=math.asin((math.sin(ang_delta)*math.sin(latitude1))+
115           (math.cos(ang_delta)*math.cos(latitude1)*math.cos(HRA)))
116         azimuth=math.acos((math.sin(ang_delta)*math.cos(latitude1))
117         -(math.cos(ang_delta)*math.sin(latitude1)*math.cos(HRA))
118         /math.cos(alpha))
119         print('azimuth=',azimuth)
120
121         M=(math.cos(30.8)+(0.15)*(93.885-30.8)**(-1.253))**(-1);
122         print('M=',M);
123
124         ho=22000;           ##height of maximum ozone concentration
125
126         #ozone mass
127         Mo = (1+(ho/6370)/((math.cos(zenith1)*math.cos(zenith1))
128             +(2*ho/6370)))**0.5;
129
130         #ozone amount
131         O3=0.344
132
133         P=1013;             #measure surface pressure
134         Po=1013;
135
136         ##earth-sun distance factor
137         D=1.00011+0.034221*math.cos(temp)+0.00128*math.sin(temp)
138         +0.000719*math.cos(2*temp)+0.000077*math.sin(2*temp);
139         print('D=',D);
140
141         ALG= math.log(1-0.65)
142         AFS=ALG*(1.459+ALG*(0.1595+ALG*0.4129))
143         BFS=ALG*(0.0783+ALG*(-0.3824-ALG*0.5874))
144         Fs=1*0.5*math.exp((AFS+BFS*math.cos(zenith1))
145                 *math.cos(zenith1))
146         print('Fs=',Fs);
147
148         Td=temperature1-((100-humidity1)/5);
149         W=math.exp(0.981+(0.0341*Td));
150
151         au_file = csv.DictReader(open("coefficients.csv"))
152         file= open('transmittance' + '.csv', 'w')
153         file.write("wavelength"+","+"Tu"+","+"To"+","+"Tr"+","
```

```
            +"Ta"+","+"Tw"+","+"I"+","+"omega\n")
    for col in au_file:

        #gaseous absorption coefficient
        au= float(col["au"])

        #ozone absorption coefficient
        ao=float(col["ao"])

        #water vapor absorption coefficient
        aw=float(col["aw"])

        #extraterrestrial spectrum
        Ho=float(col["Ho"])

        w=int(col["wavelength"])

        ##uniformly mixed gas absorption
        Tu=math.exp((-1.41*au*M)/(1+118.93*au*M)**0.45);

        ##ozone absorption
        To=math.exp(-ao*O3*Mo);

        ##Rayleigh scattering
        Tr=(math.exp(-M/((w**4)*(115.6406-(1.335/w**2)))));

        ##aerosol scattering
        Ta=(math.exp(0.155*w**(-1.140)*M));

        ##water absorption
        Tw=math.exp((-0.2385*aw*1.42*M)/(1+20.07*aw*1.42*M)**0.43);

        ##extraterrestrial irradiance*earth-sun distance
        I= Ho*D;

        ##wavelength variation factor
        omega=0.945*(math.exp(-0.095*(math.log(w/0.4))**2))

        file= open('transmittance' + '.csv','a')
        file.write(str(w)+","+str(Tu)+","+str(To)+","+str(Tr)
              +","+str(Ta)+","+str(Tw)+","+str(I)+","+str(omega)+"\n" )
        file.flush()
        file.close()
    file.close()
    final_file = csv.DictReader(open("transmittance.csv"))
    file= open('irradiances' + '.csv', 'w')
    file.write("wavelength"+","+"I"+","+"Id"+","+"Ir"+","
          +"Ia"+","+"Ig"+","+"Is\n")
    for col in final_file:
        Tu=float(col["Tu"])
        To=float(col["To"])
```

```
205             Tw=float(col["Tw"])
206             Ta=float(col["Ta"])
207             Tr=float(col["Tr"])
208             I=float(col["I"])
209             w=int(col["wavelength"])
210             o=float(col["omega"])
211
212             ##direct irradiance
213             Id=Tu*To*Ta*Tr*Tw*I;
214
215             ##rayleigh scattering component
216             Ir=I*math.cos(zenith1)*Tu*To*Ta*Tw*(1-Tr)*0.5
217
218             ##aerosol scattering component
219             Ia=I*math.cos(zenith1)*Tu*To*Tr*Tw*(1-Ta)*o*Fs
220
221             ##component that accounts for multiple reflections
222             betwwen ground and air
223             Ig=(Tu*To*Ta*Tr*Tw*I*math.cos(zenith1))+
224             (I*math.cos(zenith1)*Tu*To*Ta*Tw*(1-Tr)*0.5)+
225             (I*math.cos(zenith1)*Tu*To*Tr*Tw*(1-Ta)*o*Fs)
226
227             ##total scattered irradiance
228             Is=((I*math.cos(zenith1)*Tu*To*Ta*Tw*(1-Tr)*0.5)+
229                (I*math.cos(zenith1)*Tu*To*Tr*Tw*(1-Ta)*o*Fs)+
230                ((Tu*To*Ta*Tr*Tw*I*math.cos(zenith1))+
231                 (I*math.cos(zenith1)*Tu*To*Ta*Tw*(1-Tr)*0.5)+
232                 (I*math.cos(zenith1)*Tu*To*Tr*Tw*(1-Ta)*o*Fs)))/1000
233
234             file=open('irradiances' + '.csv','a')
235             file.write(str(w)+","+str(I)+","+str(Id)+","+str(Ir)+","
236                    +str(Ia)+","+str(Ig)+","+str(Is)+"\n")
237             file.flush()
238             file.close()
239         file.close()
240         last_file = csv.DictReader(open("irradiances.csv"))
241         file= open('total' + '.csv', 'w')
242         file.write("wavelength"+","+"It\n")
243         for col in last_file:
244             Ii=float(col["I"])
245             Id=float(col["Id"])
246             Is=float(col["Is"])
247             w=int(col["wavelength"])
248
249     ##spectral global irradiance
250             It = abs(((Id*math.cos(zenith1))+Is)/1000)
251
252             file=open('total' + '.csv','a')
253             file.write(str(w)+","+str(It)+"\n")
254             file.flush()
255             file.close()
```

```
256         file.close()
257
258     def sensors():
259         import csv
260         from datetime import datetime
261         import pandas as pd
262         import matplotlib.pyplot as plt
263
264         ser = serial.Serial('/dev/ttyACM0',baudrate=9600,timeout=10
265                     ,parity=serial.PARITY_NONE,
266                     stopbits=serial.STOPBITS_ONE,
267                     bytesize=serial.EIGHTBITS
268                     )
269         print(ser)
270         file = open('hi' + '.csv', 'a')
271
272         temp = ser.readline()
273         temp = ser.readline()
274         temp = ser.readline()
275         print(temp)
276
277         file= open('hi' + '.csv', 'w')
278         file.write("Wavelength"+","+"Irradiance"+"\n")
279         file.write(str(temp,"utf-8"))
280         file.flush()
281
282         temp = ser.readline()
283
284         file.write(str(temp,"utf-8"))
285         file.flush()
286         file.close()
287
288     def nsrdb():
289
290         import csv
291         from datetime import datetime
292         import pandas as pd
293         import matplotlib.pyplot as plt
294         global t2
295         day1 = int(day.get())
296         hour1 = int(hour.get())
297         t1= (int(day.get()) - 1) * 24
298         t2= (int(hour.get())) + t1
299         t3 = str(t2)
300         print(t3)
301         if day1 < 366 and hour1 < 25:
302             labelText7=StringVar(master=root)
303             labelText7.set("                    ")
304             labelText7.set("    Inputs Verified    ")
305             labelDir7=Label(root, textvariable=labelText7,fg="white",
306                     bg='green',font= "Times 15 bold", height=1)
```

```
307             labelDir7.grid(row =15, column=2)
308             labelText7=StringVar(master=root)
309
310         else :
311             labelText7=StringVar(master=root)
312             labelText7.set("Error : Invalid Inputs ")
313             labelDir7=Label(root, textvariable=labelText7,fg="white",
314                     bg='red',font= "Times 15 bold", height=1)
315             labelDir7.grid(row =15, column=2)
316             labelText7=StringVar(master=root)
317
318     def gettext1():
319
320         import pandas as pd
321         import time
322         from time import sleep
323         from datetime import datetime
324         from matplotlib import pyplot
325         from matplotlib import pyplot as plt
326
327         if var.get()==2:
328
329             equations()
330             df = pd.read_csv('irradiances.csv', skiprows = [1])
331             y2=df['Is']
332             x2=df['wavelength']
333
334         if var.get()==1:
335
336             sensors()
337             df = pd.read_csv('hi.csv', skiprows = [1])
338             y1=df['Irradiance']
339             x1=df['Wavelength']
340
341
342         global fig
343         global samples
344         fig.clf()
345         canvas = FigureCanvasTkAgg(fig, master=root)
346         plot_widget = canvas.get_tk_widget()
347         plt.xlabel('Wavelength(nm)')
348         plt.ylabel('Irradiance(W/m²)')
349         if var.get()==1:
350           plt.plot(x1,y1,'r')
351         elif var.get()==2:
352           plt.plot(x2,y2,'b')
353           plt.xlim((0,2500))
354         elif var.get()==3:
355           plt.plot(x1,y1,'r',label="NSRDB")
356           plt.plot(x2,y2,'b',label="Equations")
357           plt.legend()
```

```
358         elif var.get()==4:
359           plt.ylabel('Temperature Sensor (Fahrenheit)')
360 
361         plot_widget.grid(column=0, row=7, rowspan=1,columnspan=1)
362         plt.tight_layout()
363         fig.canvas.draw()
364 
365       def gettext2():
366 
367         import pandas as pd
368 
369         if var3.get()==1:
370 
371           nsrdb()
372           df = pd.read_csv('78.8_tmy_use.csv', skiprows = [1])
373           y1=df[str(t2)]
374           x1=df['Wavelength']
375 
376         if var3.get()==2:
377 
378           nsrdb()
379           df = pd.read_csv('78.8_tmy_use.csv', skiprows = [1])
380           y1=df[str(t2)]
381           x1=df['Wavelength']
382 
383           equations()
384           ff = pd.read_csv('irradiances.csv', skiprows = [1])
385           y2=ff['Is']
386           x2=ff['wavelength']
387 
388         global fig
389         global samples
390         fig.clf()
391         canvas = FigureCanvasTkAgg(fig, master=root)
392         plot_widget = canvas.get_tk_widget()
393         plt.xlabel('Wavelength(nm)')
394         plt.ylabel('Irradiance(W/m²)')
395         if var3.get()==1:
396           plt.plot(x1,y1,'r')
397         elif var3.get()==3:
398           plt.plot(x2,y2,'b')
399         elif var3.get()==2:
400           plt.plot(x1,y1,'r',label="NSRDB")
401           plt.plot(x2,y2,'b',label="Equations")
402           plt.xlim((0,2500))
403           plt.legend()
404         elif var3.get()==4:
405           plt.ylabel('None')
406 
407         plot_widget.grid(column=2, row=7, rowspan=1,columnspan=1)
408         plt.tight_layout()
```

```
409
410
411 #.............HEADING AND LOGO.................#
412
413
414     labelText71=StringVar(master=root)
415     labelText71.set("Solar Spectrum Simulator")
416     labelDir71=Label(root, textvariable=labelText71,font=
417               "Times 30 bold", height=1,bg='light yellow')
418     labelDir71.grid(sticky=tk.W,row=0,column=2)
419
420     labelText72=StringVar(master=root)
421     labelText72.set("      ")
422     labelDir72=Label(root, textvariable=labelText72,font=
423               "Times 30 bold", height=1,bg='light yellow')
424     labelDir72.grid(sticky=tk.E,row=0,column=4)
425
426     im = Image.open("dbitlogo.png")
427     imag = im.resize((60,60), Image.ANTIALIAS)
428     img = ImageTk.PhotoImage(imag, master=root)
429     label11 = Label(root, image=img,bg='light yellow')
430     label11.grid(row =0, column=0,sticky=tk.N+tk.W,
431               padx=1,pady=1)
432
433     im2 = Image.open("ncprelogo.png")
434     imag2 = im2.resize((60,60), Image.ANTIALIAS)
435     img2 = ImageTk.PhotoImage(imag2,master=root)
436     label13 = Label(root, image=img2,bg='light yellow')
437     label13.grid(row =0, column=4,sticky=tk.N+tk.E,
438               padx=1,pady=1)
439
440 #.............BOUNDARY.................#
441     import tkinter.ttk
442
443     tkinter.ttk.Separator(root, orient=VERTICAL)
444     .grid(column=1, row=1, rowspan=16, sticky=tk.E+tk.N+tk.S)
445     tkinter.ttk.Separator(root, orient=VERTICAL)
446     .grid(column=3, row=1, rowspan=16, sticky=tk.E+tk.N+tk.S)
447     tkinter.ttk.Separator(root, orient=HORIZONTAL)
448     .grid(column=0, row=1, columnspan=10, sticky='ew')
449     tkinter.ttk.Separator(root, orient=HORIZONTAL)
450     .grid(column=0, row=3, columnspan=10, sticky='ew')
451     tkinter.ttk.Separator(root, orient=HORIZONTAL)
452     .grid(column=0, row=9, columnspan=4, sticky='ew')
453     tkinter.ttk.Separator(root, orient=HORIZONTAL)
454     .grid(column=0, row=10, columnspan=4, sticky='ew')
455     tkinter.ttk.Separator(root, orient=HORIZONTAL)
456     .grid(column=1, row=14, columnspan=2, sticky='ews')
457
458 #.............LIVE PLOTTING.....................#
459     labelText2=StringVar(master=root)
```

```
460     labelText2.set("Plot 1 :")
461     labelDir1=Label(root, textvariable=labelText2,fg="blue",
462             bg='light yellow',font= "Times 12 bold", height=1)
463     labelDir1.grid(row =2, column=0 ,sticky=tk.N+tk.W)
464
465     labelText2=StringVar(master=root)
466     labelText2.set("Plot 2 :")
467     labelDir1=Label(root, textvariable=labelText2,fg="blue",
468             bg='light yellow',font= "Times 12 bold", height=1)
469     labelDir1.grid(row =2, column=2 ,sticky=tk.N+tk.W)
470
471     var = IntVar(master=root)
472     R1 = Radiobutton(root, text="Sensors Value
473             ",variable=var, value=1,height=1)
474     R1.grid(row=5, column=0,sticky=tk.N+tk.W)
475
476     R2 = Radiobutton(root, text="Reference Plot
477             ", variable=var, value=2,height=1)
478     R2.grid(row=5, column=0,sticky=tk.N+tk.E)
479
480     fig.clf()
481     plt.grid(True)
482     canvas = FigureCanvasTkAgg(fig, master=root)
483     plot_widget = canvas.get_tk_widget()
484     plot_widget.grid(row=7, column=0,sticky=S+W)
485     fig.canvas.draw()
486
487
488 #.....SOLAR PANEL APPLICATION PART.....#
489
490 #.....SAVED DATA PLOTTING.....#
491
492     var3 = IntVar(master=root)
493     R6 = Radiobutton(root, text="NSRDB Data
494             ",variable=var3, value=1,height=1)
495     R6.grid(row=5, column=2,sticky=tk.N+tk.W)
496
497     R7 = Radiobutton(root, text="Overlay Plot
498             ", variable=var3, value=2,height=1)
499     R7.grid(row=5, column=2,sticky=tk.N+tk.E)
500     fig.clf()
501     plt.grid(True)
502     canvas = FigureCanvasTkAgg(fig, master=root)
503     plot_widget = canvas.get_tk_widget()
504     plot_widget.grid(column=2, row=7, rowspan=1,columnspan=1)
505     fig.canvas.draw()
506
507 #.....ENTRY WIDGET.....#
508
509     labelText7=StringVar(master=root)
510     labelText7.set("ENTER : ")
```

```
511     labelDir7=Label(root, textvariable=labelText7,fg="black",
512             bg='light yellow',font= "Times 10 bold", height=1)
513     labelDir7.grid(row =10, column=2,sticky=tk.N+tk.W)
514
515     var12 = tk.IntVar(master=root)
516
517     R22 = Radiobutton(root, text="Day
518             ",variable=var12, value=7,height=1)
519             .grid(row=11, column=2,sticky=tk.N+tk.W)
520     day = tk.Entry(root,width=30)
521     day.grid(row=12, column=2, sticky=tk.W+tk.N)
522
523     R23 = Radiobutton(root, text="Hour
524             ",variable=var12, value=8,height=1)
525             .grid(row=11, column=2,sticky=tk.N+tk.E)
526     hour = tk.Entry(root,width=30)
527     hour.grid(row=12, column=2,sticky=tk.N+tk.E)
528
529     R24 = Radiobutton(root, text="Difference between local
530             time and universal standard time in India",
531             variable=var12, value=9,height=1)
532             .grid(row=13, column=2,sticky=tk.N+tk.W)
533     delta = tk.Entry(root,width=30)
534     delta.grid(row=14, column=2,sticky=tk.N+tk.W)
535
536     R16 = Radiobutton(root, text="Longitude          -
537             ",variable=var12, value=1,height=1)
538             .grid(row=11, column=0,sticky=tk.N+tk.W)
539     longitude = tk.Entry(root,width=30)
540     longitude.grid(row=12, column=0, sticky=tk.N+tk.W)
541
542     R17 = Radiobutton(root, text="Latitude
543             ",variable=var12, value=2,height=1)
544             .grid(row=11, column=0,sticky=tk.N+tk.E)
545     latitude = tk.Entry(root,width=30)
546     latitude.grid(row=12, column=0,sticky=tk.N+tk.E)
547
548     R18 = Radiobutton(root, text="Tilt Angle
549             ",variable=var12, value=3,height=1)
550             .grid(row=13, column=0,sticky=tk.N+tk.W)
551     tilt = tk.Entry(root,width=30)
552     tilt.grid(row=14, column=0, sticky=tk.N+tk.W)
553
554     R20 = Radiobutton(root, text="Humidity
555             ",variable=var12, value=5,height=1)
556             .grid(row=15, column=0,sticky=tk.N+tk.W)
557     humidity = tk.Entry(root,width=30)
558     humidity.grid(row=16, column=0, sticky=tk.N+tk.W)
559
560     R21 = Radiobutton(root, text="Temperature
561             ",variable=var12, value=6,height=1)
```

```
562                 .grid(row=15, column=0,sticky=tk.N+tk.E)
563         temperature = tk.Entry(root,width=30)
564         temperature.grid(row=16, column=0,sticky=tk.N+tk.E)
565
566         labelText9=StringVar(master=root)
567         labelText9.set("ENTER : ")
568         labelDir9=Label(root, textvariable=labelText9,fg="black",
569                 bg='light yellow',font= "Times 10 bold", height=1)
570         labelDir9.grid(row =10, column=0,sticky=tk.N+tk.W)
571         labelText9=StringVar(master=root)
572
573         labelText3=StringVar(master=root)
574         labelText3.set("MESSAGE:")
575         labelDir2=Label(root, textvariable=labelText3, height=1,
576                 bg='light yellow')
577         labelDir2.grid(row =15, column=2 ,sticky=tk.N+tk.W)
578
579         labelText3=StringVar(master=root)
580         labelText3.set("Virtual Keypad")
581         labelDir=Label(root, textvariable=labelText3,font=
582                 "Times 12 bold italic", height=2,bg='light yellow')
583         labelDir.grid(row=10,column=4, sticky=tk.S+tk.W)
584
585         c = Button(root,text='Plot 1',bg='light blue',command=gettext1)
586         c.grid(sticky=tk.N+tk.S+tk.W+tk.E,column=0,row=6)
587
588         cc = Button(root,text='Plot 2',bg='light blue',command=gettext2)
589         cc.grid(sticky=tk.N+tk.S+tk.W+tk.E,column=2,row=6)
590
591 #------------------popup window------------------#
592
593         def clickforhelp():
594           global serialno
595           serialno=0
596           toplevel = Toplevel(bg="light yellow")
597           toplevel.title("User Guide")
598
599           label2 = Label(toplevel, text="There are 2 canvases on
600                 which the desired graphs can be obtained.", height=0,
601                 bg="light yellow")
602           label2.grid(sticky=tk.W,column=0,row=2)
603           label2 = Label(toplevel, text="1. Canvas 1 gives
604                   you two options :",
605                   height=0,bg="light yellow")
606           label2.grid(sticky=tk.W,column=0,row=4)
607           label2 = Label(toplevel, text="    (a)Sensors Value", height=0,
608                   bg="light yellow")
609           label2.grid(sticky=tk.W,column=0,row=5)
610           label2 = Label(toplevel, text="
611             -->To get the plot just select the Sensors value option.  ",
612                   height=0,bg="light yellow")
```

```
613         label2.grid(sticky=tk.W,column=0,row=6)
614         label2 = Label(toplevel, text="
615         -->And click on the button below ", height=0,bg="light yellow")
616         label3 = Label(toplevel, text="      spacing      ",
617                 bg="light yellow",fg="light yellow")
618         label3.grid(sticky=tk.W,row=8,column=0)
619         label2 = Label(toplevel, text="    (b)Reference Plot    ",
620                 height=0,bg="light yellow")
621         label2.grid(sticky=tk.W,column=0,row=9)
622         label2 = Label(toplevel, text="
623         -->This option gives the user a reference plot based on some
624                 user defined values.",
625                 height=0,bg="light yellow")
626         label2.grid(sticky=tk.W,column=0,row=10)
627         label3 = Label(toplevel, text="
628         The main purpose of having this reference plot so that
629             the user will
630             be able to compare other plots with this one.",
631                 bg="light yellow")
632         label3.grid(sticky=tk.W,row=11,column=0)
633         label2 = Label(toplevel, text="
634             This will eventually help them to check the accuracy of their plot.",
635                 height=0,bg="light yellow")
636         label2.grid(sticky=tk.W,column=0,row=12)
637         label2 = Label(toplevel, text="
638           -->To get this plot select the Reference plot option by clicking on
639             the radiobutton.", height=0,bg="light yellow")
640         label2.grid(sticky=tk.W,column=0,row=13)
641         label2 = Label(toplevel, text="
642         -->Then enter the values in all of the empty boxes provided
643                 below the canvas.", height=0,bg="light yellow")
644         label2.grid(sticky=tk.W,column=0,row=14)
645         label2 = Label(toplevel, text="
646         -->Then click on Plot button to get the plot.", height=0,
647                 bg="light yellow")
648         label2.grid(sticky=tk.W,column=0,row=15)
649         label3 = Label(toplevel, text="      spacing      ",
650                 bg="light yellow",fg="light yellow")
651         label3.grid(sticky=tk.W,row=16,column=0)
652         label2 = Label(toplevel, text="2. Canvas 2 also gives you 2 options :",
653                 height=0,bg="light yellow")
654         label2.grid(sticky=tk.W,column=0,row=17)
655         label2 = Label(toplevel, text="    (a)NSRDB Data",
656                 height=0,bg="light yellow")
657         label2.grid(sticky=tk.W,column=0,row=18)
658         label2 = Label(toplevel, text="
659           -->This plot uses all the values which are collected by National
660                 Solar Radiation Database.", height=0,bg="light yellow")
661         label2.grid(sticky=tk.W,column=0,row=19)
662         label2 = Label(toplevel, text="
663         -->To get the plot just select the NSRDB Plot button.",
```

```
664           height=0,bg="light yellow")
665         label2.grid(sticky=tk.W,column=0,row=20)
666         label2 = Label(toplevel, text="
667           -->Then fill in the day number and the hour of the day at which the
668             plot is desired.", height=0,bg="light yellow")
669         label2.grid(sticky=tk.W,column=0,row=21)
670         label2 = Label(toplevel, text="
671         -->Then just click on the Plot button.", height=0,bg="light yellow")
672         label2.grid(sticky=tk.W,column=0,row=22)
673         label3 = Label(toplevel, text="      spacing     ",
674             bg="light yellow",fg="light yellow")
675         label3.grid(sticky=tk.W,row=23,column=0)
676         label2 = Label(toplevel, text="    (b) Overlay Plot ",
677             height=0,bg="light yellow")
678         label2.grid(sticky=tk.W,column=0,row=24)
679         label2 = Label(toplevel, text="
680         --> This option will plot the reference plot and the Nsrdb
681             plot on the same canvas.",
682                 height=0,bg="light yellow")
683         label2.grid(sticky=tk.W,column=0,row=25)
684         label2 = Label(toplevel, text="
685           -->This will help to match the plots.", height=0,
686                 bg="light yellow")
687         label2.grid(sticky=tk.W,column=0,row=26)
688         label2 = Label(toplevel, text="
689         -->To get this plot select the Overlay Plot option.", height=0,
690                 bg="light yellow")
691         label2.grid(sticky=tk.W,column=0,row=27)
692         label2 = Label(toplevel, text="
693         -->Then enter the values in all of the empty boxes
694                 provided below the canvas.",
695             height=0,bg="light yellow")
696         label2.grid(sticky=tk.W,column=0,row=28)
697         label2 = Label(toplevel, text="
698         -->Then click on Plot button to get the plot.", height=0,
699                 bg="light yellow")
700         label2.grid(sticky=tk.W,column=0,row=29)
701         label3 = Label(toplevel, text="      spacing     ",
702                 bg="light yellow",
703                   fg="light yellow")
704         label3.grid(sticky=tk.W,row=30,column=0)
705         label2 = Label(toplevel, text="3. Virtual Keypad ", height=0,
706             bg="light yellow")
707         label2.grid(sticky=tk.W,column=0,row=31)
708         label2 = Label(toplevel, text="
709          --> In case you don't have your keyboard,
710             you can still enter the values by
711         using the Virtual Keypad. ", height=0,bg="light yellow")
712         label2.grid(sticky=tk.W,column=0,row=32)
713         label2 = Label(toplevel, text="
714           |--> You can find the virtual keypad provided
```

```
715         on the bottom right corner of the GUI", height=0,
716                 bg="light yellow")
717         label2.grid(sticky=tk.W,column=0,row=33)
718         label2 = Label(toplevel, text="
719             --> To enter the values using virtual
720         keypad click on theempty box that you need to fill",
721                 height=0,
722           bg="light yellow")
723         label2.grid(sticky=tk.W,column=0,row=34)
724         label2 = Label(toplevel, text="
725         --> And then just click on the numbers you want to enter",
726         height=0,bg="light yellow")
727         label2.grid(sticky=tk.W,column=0,row=35)
728
729       button1 = Button(root, text="USER GUIDE", width=20,
730       command=clickforhelp,bg="light pink")
731       button1.grid(column=4,row=2,sticky=tk.E)
732
733       def clicktoseelivevalue():
734         toplevel1 = Toplevel(bg="light yellow")
735         toplevel1.title("Live Data Analysis")
736         label1 = Label(toplevel1, font= "Times 14 bold italic",
737           height=3,text="Live Data Values for all sensors:",
738                 bg="light yellow")
739         label1.grid(row=0,column=1)
740         label3 = Label(toplevel1, text="     spacing     ",
741           bg="light yellow",fg="light yellow")
742         label3.grid(sticky=tk.W,row=1,column=0)
743         label2 = Label(toplevel1, text="Device Temperature:",
744         font= "Times 12 bold",bg="light green")
745         label2.grid(sticky=tk.E,row=1,column=0)
746         label3 = Label(toplevel1, text="     spacing     ",
747         bg="light yellow",fg="light yellow")
748         label3.grid(sticky=tk.W,row=2,column=0)
749         label2 = Label(toplevel1, text="Temperature sensor1:",
750         font= "Times 12 bold",bg="light green")
751         label2.grid(sticky=tk.E,row=3,column=0)
752         label3 = Label(toplevel1, text="     spacing     ",
753         bg="light yellow",fg="light yellow")
754         label3.grid(sticky=tk.W,row=4,column=0)
755         label2 = Label(toplevel1, text="Temperature sensor2:",
756         font= "Times 12 bold",bg="light green")
757         label2.grid(sticky=tk.E,row=5,column=0)
758         label3 = Label(toplevel1, text="     spacing     ",
759         bg="light yellow",fg="light yellow")
760         label3.grid(sticky=tk.W,row=6,column=0)
761         label2 = Label(toplevel1, text="Temperature sensor3:",
762         font= "Times 12 bold",bg="light green")
763         label2.grid(sticky=tk.E,row=7,column=0)
764         lab = Label(toplevel1,bg='light yellow',font= "Times 14 bold")
765         lab.grid(row=1,column=1)
```

```
766         lab1 = Label(toplevel1,bg='light yellow',font= "Times 14 bold")
767         lab1.grid(row=3,column=1)
768
769       button2 = Button(root, text="Live Data Analysis",
770       command=clicktoseelivevalue,bg="light pink")
771       button2.grid(column=0,row=2,sticky=tk.E)
772
773 #.....VIRTUAL KEYPAD.....#
774
775  def code1(value):
776         global pin
777
778         if value == '<=':
779           # remove last number from `pin`
780           pin = pin[:-1]
781           # remove all from `entry` and put new `pin`
782           if var12.get()==1:
783             longitude.delete('0', 'end')
784             longitude.insert('end', pin)
785           if var12.get()==2:
786             latitude.delete('0', 'end')
787             latitude.insert('end', pin)
788           if var12.get()==3:
789             tilt.delete('0', 'end')
790             tilt.insert('end', pin)
791           if var12.get()==4:
792             zenith.delete('0', 'end')
793             zenith.insert('end', pin)
794           if var12.get()==5:
795             humidity.delete('0', 'end')
796             humidity.insert('end', pin)
797           if var12.get()==6:
798             temperature.delete('0', 'end')
799             temperature.insert('end', pin)
800           if var12.get()==7:
801             day.delete('0', 'end')
802             day.insert('end', pin)
803           if var12.get()==8:
804             hour.delete('0', 'end')
805             hour.insert('end', pin)
806           if var12.get()==9:
807             delta.delete('0', 'end')
808             delta.insert('end', pin)
809
810         elif value == 'CLR':
811           # clear `pin`
812           pin = ''
813           # clear `entry`
814
815           if var12.get()==1:
816             longitude.delete('0', 'end')
```

```
817             if var12.get()==2:
818               latitude.delete('0', 'end')
819             if var12.get()==3:
820               tilt.delete('0', 'end')
821             if var12.get()==4:
822               zenith.delete('0', 'end')
823             if var12.get()==5:
824               humidity.delete('0', 'end')
825             if var12.get()==6:
826               temperature.delete('0', 'end')
827             if var12.get()==7:
828               day.delete('0', 'end')
829             if var12.get()==8:
830               hour.delete('0', 'end')
831             if var12.get()==9:
832               delta.delete('0', 'end')
833
834
835           else:
836             # add number to pin
837             pin += value
838             # add number to `entry`
839
840             if var12.get()==1:
841               longitude.insert('end', value)
842             if var12.get()==2:
843               latitude.insert('end', value)
844             if var12.get()==3:
845               tilt.insert('end', value)
846             if var12.get()==4:
847               zenith.insert('end', value)
848             if var12.get()==5:
849               humidity.insert('end', value)
850             if var12.get()==6:
851               temperature.insert('end', value)
852             if var12.get()==7:
853               day.insert('end', value)
854             if var12.get()==8:
855               hour.insert('end', value)
856             if var12.get()==9:
857               delta.insert('end', value)
```

```
858
859       keys = [
860          ['1', '2', '3'],
861          ['4', '5', '6'],
862          ['7', '8', '9'],
863          ['-', '0', ':'],
864          ['<=', 'CLR',' ']
865       ]
866       pin = '' # empty string
867
868       # create buttons using `keys`
869       for y, row in enumerate(keys, 1):
870          for x, key in enumerate(row):
871                b1 = tk.Button(root, text=key,
872                command=lambda val=key:code1(val))
873                if x==0:
874                   b1.grid(row=y+10,sticky=tk.W, column=4, ipadx=15)
875                if x==1:
876                   b1.grid(row=y+10, column=4, ipadx=15)
877                if x==2:
878                   b1.grid(row=y+10,sticky=tk.E, column=4, ipadx=15)
879
880       root.mainloop()
881
882    else:
883       print ("change")
884
885       a=b
886
887 root.mainloop()
```

```
917       c.grid(sticky=tk.N+tk.S+tk.W+tk.E,column=0,row=6)
918
919       cc = Button(root,text='Plot 2',bg='light blue',command=gettext2)
920       cc.grid(sticky=tk.N+tk.S+tk.W+tk.E,column=2,row=6)
921      #  c.size(height=50, width=50)
922  #------------------popup window------------------#
923
924       def clickforhelp():
925         global serialno
926         serialno=0
927         toplevel = Toplevel(bg="light yellow")
928         toplevel.title("User Guide")
929         #label1 = Label(toplevel, text="USER GUIDE:", height=0,bg="light pink")
930         #label1.grid(sticky=tk.W+tk.S)
931         #label3 = Label(toplevel, text="      spacing     ",bg="light yellow",
932         fg="light yellow")
933         #label3.grid(sticky=tk.W,row=1,column=0)
934
935         label2 = Label(toplevel, text=
936        "There are 2 canvases on which the desired graphs can be obtained.",
937         height=0,bg="light yellow")
938         label2.grid(sticky=tk.W,column=0,row=2)
939         #label3 = Label(toplevel, text="      spacing     ",bg="light yellow",
940         fg="light yellow")
941         #label3.grid(sticky=tk.W,row=3,column=0)
942         label2 = Label(toplevel, text="1. Canvas 1 gives you two options :",
943        height=0,bg="light yellow")
944         label2.grid(sticky=tk.W,column=0,row=4)
945         label2 = Label(toplevel, text="    (a)Sensors Value", height=0,
946        bg="light yellow")
947         label2.grid(sticky=tk.W,column=0,row=5)
948         label2 = Label(toplevel, text=
949        "-->To get the plot just select the Sensors value option.  ",
950         height=0,bg="light yellow")
951         label2.grid(sticky=tk.W,column=0,row=6)
952         label2 = Label(toplevel, text="      -->And click on the button below ",
953         height=0,bg="light yellow")
954  ##         label2.grid(sticky=tk.W,column=0,row=7)
955         label3 = Label(toplevel, text="      spacing     ",bg="light yellow",
956        fg="light yellow")
957         label3.grid(sticky=tk.W,row=8,column=0)
958         label2 = Label(toplevel, text="    (b)Reference Plot    ", height=0,
959        bg="light yellow")
960         label2.grid(sticky=tk.W,column=0,row=9)
961         label2 = Label(toplevel, text="
962         -->This option gives the user a reference plot based on some user defined values.",
963         height=0,bg="light yellow")
964         label2.grid(sticky=tk.W,column=0,row=10)
965         label3 = Label(toplevel, text=
966        "The main purpose of having this reference plot so that the
967        user will be able to compare other plots with this one.",
```

```
967         user will be able to compare other plots with this one.",
968         bg="light yellow")
969         label3.grid(sticky=tk.W,row=11,column=0)
970         label2 = Label(toplevel, text=
971         " This will eventually help them to check the accuracy of their plot.",
972         height=0,bg="light yellow")
973         label2.grid(sticky=tk.W,column=0,row=12)
974         label2 = Label(toplevel, text=
975         " -->To get this plot select the Reference plot option by clicking on
976         the radiobutton.",
977         height=0,bg="light yellow")
978         label2.grid(sticky=tk.W,column=0,row=13)
979         label2 = Label(toplevel, text=
980         "      -->Then enter the values in all of the empty boxes provided
981             below the canvas.",
982         height=0,bg="light yellow")
983         label2.grid(sticky=tk.W,column=0,row=14)
984         label2 = Label(toplevel, text="      -->Then click on Plot button
985                 to get the plot.",
986         height=0,bg="light yellow")
987         label2.grid(sticky=tk.W,column=0,row=15)
988         label3 = Label(toplevel, text="      spacing      ",
989         bg="light yellow",fg="light yellow")
990         label3.grid(sticky=tk.W,row=16,column=0)
991         label2 = Label(toplevel, text="2. Canvas 2 also gives you 2 options :",
992         height=0,bg="light yellow")
993         label2.grid(sticky=tk.W,column=0,row=17)
994         label2 = Label(toplevel, text="    (a)NSRDB Data", height=0,
995         bg="light yellow")
996         label2.grid(sticky=tk.W,column=0,row=18)
997         label2 = Label(toplevel, text=
998         "-->This plot uses all the values which are collected by National
999                 Solar Radiation Database.",
1000        height=0,bg="light yellow")
1001        label2.grid(sticky=tk.W,column=0,row=19)
1002        label2 = Label(toplevel, text=
1003        " -->To get the plot just select the NSRDB Plot button.",
1004        height=0,bg="light yellow")
1005        label2.grid(sticky=tk.W,column=0,row=20)
1006        label2 = Label(toplevel, text=
1007        "-->Then fill in the day number and the hour of the day at
1008                which the plot is desired.",
1009        height=0,bg="light yellow")
1010        label2.grid(sticky=tk.W,column=0,row=21)
1011        label2 = Label(toplevel, text=
1012        "      -->Then just click on the Plot button.",
1013        height=0,bg="light yellow")
1014        label2.grid(sticky=tk.W,column=0,row=22)
1015        #label2 = Label(toplevel, text=
1016        "-->If all information is mentioned properly then the graph will be seen",
1017        bg="light yellow")
```

```
1017        bg="light yellow")
1018        #label2.grid(sticky=tk.W,row=22,column=0)
1019        label3 = Label(toplevel, text="     spacing     ",
1020        bg="light yellow",fg="light yellow")
1021        label3.grid(sticky=tk.W,row=23,column=0)
1022        label2 = Label(toplevel, text="   (b) Overlay Plot ",
1023        height=0,bg="light yellow")
1024        label2.grid(sticky=tk.W,column=0,row=24)
1025        label2 = Label(toplevel, text=
1026        " --> This option will plot the reference plot and the Nsrdb
1027                plot on the same canvas.",
1028        height=0,bg="light yellow")
1029        label2.grid(sticky=tk.W,column=0,row=25)
1030        label2 = Label(toplevel, text=
1031       "     -->This will help to match the plots.",
1032        height=0,bg="light yellow")
1033        label2.grid(sticky=tk.W,column=0,row=26)
1034        label2 = Label(toplevel, text=
1035       " -->To get this plot select the Overlay Plot option.",
1036        height=0,bg="light yellow")
1037        label2.grid(sticky=tk.W,column=0,row=27)
1038        label2 = Label(toplevel, text=
1039        "-->Then enter the values in all of the empty boxes
1040                provided below the canvas.",
1041        height=0,bg="light yellow")
1042        label2.grid(sticky=tk.W,column=0,row=28)
1043        label2 = Label(toplevel, text=
1044        "     -->Then click on Plot button to get the plot.",
1045        height=0,bg="light yellow")
1046        label2.grid(sticky=tk.W,column=0,row=29)
1047        label3 = Label(toplevel, text="     spacing     ",
1048        bg="light yellow",fg="light yellow")
1049        label3.grid(sticky=tk.W,row=30,column=0)
1050        label2 = Label(toplevel, text="3. Virtual Keypad ",
1051        height=0,bg="light yellow")
1052        label2.grid(sticky=tk.W,column=0,row=31)
1053        label2 = Label(toplevel, text=
1054        " --> In case you don't have your keyboard, you can still
1055                enter the values by using the Virtual Keypad. ",
1056        height=0,bg="light yellow")
1057        label2.grid(sticky=tk.W,column=0,row=32)
1058        label2 = Label(toplevel, text=
1059        " --> You can find the virtual keypad provided on the
1060                bottom right corner of the GUI",
1061        height=0,bg="light yellow")
1062        label2.grid(sticky=tk.W,column=0,row=33)
1063        label2 = Label(toplevel, text=
1064        "--> To enter the values using virtual keypad click on
1065                the empty box that you need to fill",
1066        height=0,bg="light yellow")
1067        label2.grid(sticky=tk.W,column=0,row=34)
```

```
1068        label2 = Label(toplevel, text=
1069        "--> And then just click on the numbers you want to enter",
1070        height=0,bg="light yellow")
1071        label2.grid(sticky=tk.W,column=0,row=35)
1072        #label3 = Label(toplevel, text="     spacing     ",
1073        bg="light yellow",fg="light yellow")
1074        #label3.grid(sticky=tk.W,row=33,column=0)
1075
1076      button1 = Button(root, text="USER GUIDE", width=20,
1077       command=clickforhelp,bg="light pink")
1078      button1.grid(column=4,row=2,sticky=tk.E)
1079
1080      #button3 = Button(root, text="Virtual Keypad Guide",
1081      width=20, command=clickforhelp,bg="light pink")
1082      #button3.grid(column=4,row=2,sticky=tk.E)
1083
1084      def clicktoseelivevalue():
1085        toplevel1 = Toplevel(bg="light yellow")
1086        toplevel1.title("Live Data Analysis")
1087        label1 = Label(toplevel1, font= "Times 14 bold italic",
1088        height=3,text="Live Data Values for all sensors:",bg="light yellow")
1089        label1.grid(row=0,column=1)
1090        label3 = Label(toplevel1, text="     spacing     ",
1091        bg="light yellow",fg="light yellow")
1092        label3.grid(sticky=tk.W,row=1,column=0)
1093        label2 = Label(toplevel1, text="Device Temperature:",
1094       font= "Times 12 bold",bg="light green")
1095        label2.grid(sticky=tk.E,row=1,column=0)
1096        label3 = Label(toplevel1, text="     spacing     ",
1097        bg="light yellow",fg="light yellow")
1098        label3.grid(sticky=tk.W,row=2,column=0)
1099        label2 = Label(toplevel1, text="Temperature sensor1:",
1100        font= "Times 12 bold",bg="light green")
1101        label2.grid(sticky=tk.E,row=3,column=0)
1102        label3 = Label(toplevel1, text="     spacing     ",
1103        bg="light yellow",fg="light yellow")
1104        label3.grid(sticky=tk.W,row=4,column=0)
1105        label2 = Label(toplevel1, text="Temperature sensor2:",
1106        font= "Times 12 bold",bg="light green")
1107        label2.grid(sticky=tk.E,row=5,column=0)
1108        label3 = Label(toplevel1, text="     spacing     ",
1109        bg="light yellow",fg="light yellow")
1110        label3.grid(sticky=tk.W,row=6,column=0)
1111        label2 = Label(toplevel1, text="Temperature sensor3:",
1112        font= "Times 12 bold",bg="light green")
1113        label2.grid(sticky=tk.E,row=7,column=0)
1114        lab = Label(toplevel1,bg='light yellow',font= "Times 14 bold")
1115        lab.grid(row=1,column=1)
1116        lab1 = Label(toplevel1,bg='light yellow',font= "Times 14 bold")
1117        lab1.grid(row=3,column=1)
1118        lab2 = Label(toplevel1,bg='light yellow',font= "Times 14 bold")
```

```
1119        lab2.grid(row=5,column=1)
1120        lab3 = Label(toplevel1,bg='light yellow',font= "Times 14 bold")
1121        lab3.grid(row=7,column=1)
1122        label3 = Label(toplevel1, text="     spacing     ",bg="light yellow",
1123        fg="light yellow")
1124        label3.grid(sticky=tk.W,row=8,column=0)
1125        label3 = Label(toplevel1, text="     spacing     ",bg="light yellow",
1126        fg="light yellow")
1127        label3.grid(sticky=tk.W,row=9,column=0)
1128 ##         canvas = FigureCanvasTkAgg(fig, master=toplevel1)
1129 ##         plot_widget = canvas.get_tk_widget()
1130 ##         plot_widget.grid(row=10, columnspan=2,column=3,sticky=S+E)
1131 ##         fig.canvas.draw()
1132
1133        label2 = Label(toplevel1, text="Device Temperature:",
1134        font= "Times 12 bold",bg="light green")
1135        label2.grid(sticky=tk.W,row=1,column=2)
1136        label3 = Label(toplevel1, text="     spacing     ",bg="light yellow",
1137        fg="light yellow")
1138        label3.grid(sticky=tk.W,row=2,column=2)
1139        label2 = Label(toplevel1, text="Temparature sensor1:",
1140        font= "Times 12 bold",bg="light green")
1141        label2.grid(sticky=tk.W,row=3,column=2)
1142        label3 = Label(toplevel1, text="     spacing     ",
1143        bg="light yellow",fg="light yellow")
1144        label3.grid(sticky=tk.W,row=4,column=2)
1145        label2 = Label(toplevel1, text="Temperature sensor2:",
1146        font= "Times 12 bold",bg="light green")
1147        label2.grid(sticky=tk.W,row=5,column=2)
1148        label3 = Label(toplevel1, text="     spacing     ",
1149        bg="light yellow",fg="light yellow")
1150        label3.grid(sticky=tk.W,row=6,column=2)
1151        label2 = Label(toplevel1, text="Temperature sensor3:",
1152        font= "Times 12 bold",bg="light green")
1153        label2.grid(sticky=tk.W,row=7,column=2)
1154
1155        label2 = Label(toplevel1, text="Minimum value",
1156       font= "Times 12 bold",bg="light green")
1157        label2.grid(sticky=tk.W,row=0,column=3)
1158
1159        label2 = Label(toplevel1, text="Maximum value",
1160        font= "Times 12 bold",bg="light green")
1161        label2.grid(sticky=tk.E,row=0,column=4)
1162
1163
1164 ##         label2 = Label(toplevel1, text="Mean value",
1165        font= "Times 12 bold",bg="light green")
1166 ##         label2.grid(sticky=tk.E,row=0,column=5)
1167
1168        lab4 = Label(toplevel1,bg='light yellow',font= "Times 14 bold")
1169        lab4.grid(row=1,column=3)
```

```
1170        lab5 = Label(toplevel1,bg='light yellow',font= "Times 14 bold")
1171        lab5.grid(row=3,column=3)
1172        lab6 = Label(toplevel1,bg='light yellow',font= "Times 14 bold")
1173        lab6.grid(row=5,column=3)
1174        lab7 = Label(toplevel1,bg='light yellow',font= "Times 14 bold")
1175        lab7.grid(row=7,column=3)
1176        lab8 = Label(toplevel1,bg='light yellow',font= "Times 14 bold")
1177        lab8.grid(row=1,column=4)
1178        lab9 = Label(toplevel1,bg='light yellow',font= "Times 14 bold")
1179        lab9.grid(row=3,column=4)
1180        lab10 = Label(toplevel1,bg='light yellow',font= "Times 14 bold")
1181        lab10.grid(row=5,column=4)
1182        lab11 = Label(toplevel1,bg='light yellow',font= "Times 14 bold")
1183        lab11.grid(row=7,column=4)
1184 ##        lab12 = Label(toplevel1,bg='light yellow',font= "Times 14 bold")
1185 ##        lab12.grid(row=1,column=5)
1186 ##        lab13 = Label(toplevel1,bg='light yellow',font= "Times 14 bold")
1187 ##        lab13.grid(row=3,column=5)
1188 ##        lab14 = Label(toplevel1,bg='light yellow',font= "Times 14 bold")
1189 ##        lab14.grid(row=5,column=5)
1190 ##        lab15 = Label(toplevel1,bg='light yellow',font= "Times 14 bold")
1191 ##        lab15.grid(row=7,column=5)
1192        lab7.config(text="-127.00")
1193        lab11.config(text="-127.00")
1194
1195        global o
1196        global g
1197        global w
1198        global h
1199        o=np.zeros(4,dtype=np.float32)
1200        g=np.zeros(4,dtype=np.float32)
1201        w=np.zeros(4,dtype=np.float32)
1202        h=np.zeros(4,dtype=np.float32)
1203        global max1
1204        global max2
1205        global max3
1206        global min1
1207        global min2
1208        global min3
1209        max1=0
1210        min1=0
1211        max2=0
1212        min2=0
1213        max3=max2+20.05
1214        min3=min2+20.05
1215        ad=0
1216        bc=0
1217        ef=0
1218        gh=0
1219        def livevalues():
1220          global max1
```

```
        global min1
        global max2
        global min2
        global max3
        global min3
        global serialno
        ser = serial.Serial('/dev/ttyACM0',baudrate=9600,timeout=10
                    ,parity=serial.PARITY_NONE,
                     stopbits=serial.STOPBITS_ONE,
                     bytesize=serial.EIGHTBITS
                     )
        cpua = CPUTemperature()
        tempa=cpua.temperature
        lab.config(text=tempa)
        tempvalue=ser.readline()
        y=float(tempvalue)
        x=float(tempvalue)+20.05
        z=-127.00
        if serialno<4:
          g[serialno]=y
          h[serialno]=x
          w[serialno]=z
          o[serialno]=tempa
          ad=max1
          ef=max2
          serialno=serialno+1
          lab2.config(text=x)
          lab1.config(text=y)
          lab3.config(text=z)
          t = np.arange(0, serialno)
          if ad<tempa:
            max1=tempa
            lab8.config(text=max1)
          else:
            max1=max1
            lab8.config(text=max1)
          if min1==0:
            min1=tempa
          cd=min1
          if cd>tempa:
            min1=tempa
            lab4.config(text=min1)
          else:
            min1=min1
            lab4.config(text=min1)
          if ef<y:
            max2=y
            lab9.config(text=max2)
          else:
            max2=max2
            lab9.config(text=max2)
```

```
1272            if min2==0:
1273              min2=y
1274            gh=min2
1275            if gh>y:
1276              min2=y
1277              lab5.config(text=min2)
1278            else:
1279              min2=min2
1280              lab5.config(text=min2)
1281            min3=min2+20.05
1282            lab6.config(text=min3)
1283            max3=max2+20.05
1284            lab10.config(text=max3)
1285            print(g)
1286            print(h)
1287            print(w)
1288            print(o)
1289            print(t)
1290
1291 ####             canvas = FigureCanvasTkAgg(fig, master=toplevel1)
1292 ####             plt.grid(True)
1293 ####             plt.xlabel('Time (s)')
1294 ####             plt.ylabel('Temperature Sensor (Celsius)')
1295 ####             plt.plot(t, tempa, 'r--')#, t, y, 'bs', t, x, 'g^', t, z, 'yo')
1296 ####             fig.clf()
1297 ####             plot_widget.grid(row=10, columnspan=2,column=0,
1298              sticky=S+E)
1299 ####             fig.canvas.draw()
1300 ####             plt.tight_layout()
1301 ####             plt.show()
1302           label3 = Label(toplevel1, text="      spacing      ",
1303            bg="light yellow",fg="light yellow")
1304           label3.grid(sticky=tk.W,row=9,column=0)
1305           label2 = Label(toplevel1, text="NOTE:",
1306            font= "Times 12 bold",bg="light pink")
1307           label2.grid(sticky=tk.W,row=10,column=0)
1308           label3 = Label(toplevel1, text="      spacing      ",
1309            bg="light yellow",fg="light yellow")
1310           label3.grid(sticky=tk.W,row=11,column=0)
1311           label3 = Label(toplevel1, text="      spacing      ",
1312            bg="light yellow",fg="light yellow")
1313           label3.grid(sticky=tk.W,row=12,column=0)
1314           label2 = Label(toplevel1, text="1) The device temperature
1315                    value should be always less than 70°C.",
1316          font= "Times 12 bold",bg="light yellow")
1317           label2.grid(sticky=tk.W,row=13,column=1)
1318           label3 = Label(toplevel1, text="      spacing      ",
1319           bg="light yellow",fg="light yellow")
1320           label3.grid(sticky=tk.W,row=14,column=0)
1321           label2 = Label(toplevel1, text=
1322            "2) The STC temperature value is 25°C.",
```

```
1322            "2) The STC temperature value is 25°C.",
1323           font= "Times 12 bold",bg="light yellow")
1324           label2.grid(sticky=tk.W,row=15,column=1)
1325           label3 = Label(toplevel1, text="      spacing     ",
1326            bg="light yellow",
1327          fg="light yellow")
1328           label3.grid(sticky=tk.W,row=16,column=0)
1329           label2 = Label(toplevel1, text="3) The temperature sensor
1330            DS18B20 work for the range of -55°C to +125°C.",
1331           font= "Times 12 bold",bg="light yellow")
1332           label2.grid(sticky=tk.W,row=17,column=1)
1333           label3 = Label(toplevel1, text="      spacing     ",
1334           bg="light yellow",
1335           fg="light yellow")
1336           label3.grid(sticky=tk.W,row=18,column=0)
1337           label2 = Label(toplevel1, text=
1338            "4) The sensitivity of the DS18B20 is 0.1°C.",
1339           font= "Times 12 bold",bg="light yellow")
1340           label2.grid(sticky=tk.W,row=19,column=1)
1341
1342
1343
1344           if serialno==4:
1345             serialno=0
1346           root.after(1000,livevalues)
1347         livevalues()
1348
1349       button2 = Button(root, text="Live Data Analysis",
1350        command=clicktoseelivevalue,bg="light pink")
1351       button2.grid(column=0,row=2,sticky=tk.E)
1352
1353
1354       def code1(value):
1355         global pin
1356
1357         if value == '<=':
1358           # remove last number from `pin`
1359           pin = pin[:-1]
1360           # remove all from `entry` and put new `pin`
1361           if var12.get()==1:
1362             longitude.delete('0', 'end')
1363             longitude.insert('end', pin)
1364           if var12.get()==2:
1365             latitude.delete('0', 'end')
1366             latitude.insert('end', pin)
1367           if var12.get()==3:
1368             tilt.delete('0', 'end')
1369             tilt.insert('end', pin)
1370           if var12.get()==4:
1371             zenith.delete('0', 'end')
1372             zenith.insert('end', pin)
```

```
1373            if var12.get()==5:
1374              humidity.delete('0', 'end')
1375              humidity.insert('end', pin)
1376            if var12.get()==6:
1377              temperature.delete('0', 'end')
1378              temperature.insert('end', pin)
1379            if var12.get()==7:
1380              day.delete('0', 'end')
1381              day.insert('end', pin)
1382            if var12.get()==8:
1383              hour.delete('0', 'end')
1384              hour.insert('end', pin)
1385            if var12.get()==9:
1386              delta.delete('0', 'end')
1387              delta.insert('end', pin)
1388
1389          elif value == 'CLR':
1390            # clear `pin`
1391            pin = ''
1392            # clear `entry`
1393
1394            if var12.get()==1:
1395              longitude.delete('0', 'end')
1396            if var12.get()==2:
1397              latitude.delete('0', 'end')
1398            if var12.get()==3:
1399              tilt.delete('0', 'end')
1400            if var12.get()==4:
1401              zenith.delete('0', 'end')
1402            if var12.get()==5:
1403              humidity.delete('0', 'end')
1404            if var12.get()==6:
1405              temperature.delete('0', 'end')
1406            if var12.get()==7:
1407              day.delete('0', 'end')
1408            if var12.get()==8:
1409              hour.delete('0', 'end')
1410            if var12.get()==9:
1411              delta.delete('0', 'end')
1412
1413
1414          else:
1415            # add number to pin
1416            pin += value
1417            # add number to `entry`
1418
1419            if var12.get()==1:
1420              longitude.insert('end', value)
1421            if var12.get()==2:
1422              latitude.insert('end', value)
1423            if var12.get()==3:
```

```
1424            tilt.insert('end', value)
1425          if var12.get()==4:
1426            zenith.insert('end', value)
1427          if var12.get()==5:
1428            humidity.insert('end', value)
1429          if var12.get()==6:
1430            temperature.insert('end', value)
1431          if var12.get()==7:
1432            day.insert('end', value)
1433          if var12.get()==8:
1434            hour.insert('end', value)
1435          if var12.get()==9:
1436            delta.insert('end', value)
1437
1438      keys = [
1439        ['1', '2', '3'],
1440        ['4', '5', '6'],
1441        ['7', '8', '9'],
1442        ['-', '0', ':'],
1443        ['<=', 'CLR',' ']
1444      ]
1445      pin = '' # empty string
1446
1447      # create buttons using `keys`
1448      for y, row in enumerate(keys, 1):
1449        for x, key in enumerate(row):
1450            b1 = tk.Button(root, text=key,
1451                 command=lambda val=key:code1(val))
1452            if x==0:
1453              b1.grid(row=y+10,sticky=tk.W, column=4, ipadx=15)
1454            if x==1:
1455              b1.grid(row=y+10, column=4, ipadx=15)
1456            if x==2:
1457              b1.grid(row=y+10,sticky=tk.E, column=4, ipadx=15)
1458
1459      root.mainloop()
1460
1461
1462      #root.after(1000, scanning)  # After 1 second, call scanning
1463
1464
1465    else:
1466      print ("change")
1467      # file= open('A'+str(a) + '.csv', 'a')
1468      a=b
1469 ###
1470
1471
1472 root.mainloop()
1473
```