Project:	CS426 Spring 2018, Team #23: SkyWarden Senior Project, Aerial Drone Notification System (ADNS)
Team:	Rony Calderon, Bryan Kline, Jia Li, Robert Watkins
Subsystem:	Graphical User Interface Unit
File name:	Graphical User Interface Unit Documentation.pdf
Description:	Program documentation for Graphical User Interface Unit subsystem

Program Documentation Overview

Program Overview:	High level description of different modules and classes that make up the graphical user interface unit subsystem
Program Structure:	High level description of how different modules within the graphical user interface unit work with one another to pass data and information in and out of the subsystem to other subsystems
Dependencies:	Subsystem level description of the dependencies of the modules that make up the graphical user interface unit subsystem
Programming Units:	Detailed description of each module and class, including the methods used and their descriptions
Testing Modules:	Description of the testing modules included in the system

Program Overview

The *ADNS_Main_GUI* program is a modular software addition to the *Aerial Drone Notification System*. The main purpose of the *ADNS_Main_GUI* subsystem is to allow the user to efficiently interact with other components of *Aerial Drone Notification System*. Specifically, the *ADNS_Main_GUI* allows the user to set voltage and proximity thresholds, both upper and lower bound, that are crucial for *Aerial Drone Notification System* to determine when to alert the user visually and audibly. The *MyWindow* module, within the *ADNS_Main_GUI*, sends and receives voltage and proximity data from the *Ground Base Unit* subsystem and displays them in their respective text fields, which cannot be changed by the user. More importantly, the *MyWindow* module contains various testing submodules, such as *Add Alert*, *Remove Alert*, and *Reset Alert*, that allow user to mimic hardware soft and hard resets on the software side. A feature, not implemented, is *Plot Data*, which allows the user to see voltage and proximity data in real-time graphically. A crucial functionality of *MyWindow* module is the inclusion of modular menu system that allows the user access other modules such as the *QuaternionWindow* and *ThresholdWindow*. Furthermore, the menu system provides fault-tolerance by allowing the user to save voltage and proximity data at a point in time into individual log-files.

The *QuaternionWindow* module can be accessed via menu option on *MyWindow* module. The primary feature of the *QuaternionWindow* is to display quaternion data, consist of x, y, z, roll, yaw, and pitch values, of the twelve ToF sensors, which are part of the *On-Board* unit of *Aerial Drone Notification System*, in specific text fields. The *QuaternionWindow* communicate with the *Ground Base Unit* subsystem to obtain that information via an easily accessible *test.txt* configuration used specifically by the *ROS* core module within the *Ground Base Unit* subsystem. Additionally, the *QuaternionWindow* module's text fields are mutable, unlike the *MyWindow* module, which allow the user to change quaternion information at any point in time and resonate them to rest of the subsystems in *Aerial Drone Notification System*. A feature, not implemented, is a restriction mechanism that only allows the user to set the quaternion information once, at the start up of *Aerial Drone Notification System*, to ensure data consistency and durability within *Aerial Drone Notification System*.

The *ThresholdWindow* module, like the *QuaternionWindow* module, can also be easily accessed via menu options in the *MyWindow* module. The primary purpose of the *ThresholdWindow* module is to allow the user to manually enter lower and upper bounds for voltage and proximity respectively. The user can do so by manually typing decimal values in the text fields provided in the *ThresholdWindow* module. Additionally, the *ThresholdWindow* has a warning to where if the incoming voltage and proximity data exceeded the bounds set by the user, then the *ThresholdWindow* issues a visual alert. Furthermore, the *ThresholdWindow* also provide a file-saving feature to where the user can choose to save their inputted voltage and proximity threshold values into individual log-files.

Program Structure

To start the *Graphical User Interface Unit* subsystem, the *Ground Base Unit* can call the *windowProcess* function, which serves as a *main loop* for the entire *ADNS_Main_GUI* program, which in turn enumerates other modules within *ADNS_Main_GUI*. The *MyWindow* module has an *update* function, which interfaces with the *Ground Base Unit* through Python Pipes to send and receive voltage and proximity data utilizing the multi-threading mechanism. The *update* function also continuously checks if the value of the voltage and proximity exceeds the upper and lower bound values set by the user in the *ThresholdWindow* module for voltage and proximity data. If the thresholds are exceeded, the *MyWindow* triggers either the *voltageFlag* or *proximityFlag*, which are global variables shared with the *ThresholdWindow* to indicate that a visual alert needs to be triggered, in which case the rgba value for the entire window interface changes from ‘green’ to ‘red’ to indicate that an alarm is triggered. The user can reset the alarm through clicking the reset button in the *MyWindow* module, in which case the signal generated from the user’s action will change the Boolean value of either the *voltagFlag* or *proximityFlag*, in which the resonation to the *ThresholdWindow* module will systematically shutdown the entire alert system for *Aerial Drone Notification System*.

The *QuaternionWindow* module interacts with *ROS core* within the *Ground Base Unit* to efficiently configure quaternion data for ToF sensors, part of the *On-Board Unit*. The *QuaternionWindow* does so by reading and writing to modular configuration file called *test.txt*, in which the *ROS core* also utilize the configuration file to enumerate an automation process that determines the rotational and transformational information regarding all twelve ToF sensors on-board the *Matrice 100* drone. To simply the reading and writing process to the configuration file, the *QuaternionWindow* simplifies the parsing process from user input and file output via regular expression that significantly reduce running time, which ensures data consistency by reducing latency.

Dependencies

ADNS_Main_GUI.py:

> gi, re, datetime, gi.repository.Gtk, gi.repository.GObject, gi.repository.Gdk, multiprocessing.Process, multiprocessing.Pipe, threading.Thread, time.sleep, and pathlib.Path

Programming Units

ADNS_Main_GUI.py:

Class: MyWindow

Functions:

Name:	__init__
Description:	Free function used to initialize an instance of the MyWindow object. Additionally, the function populates the window object with various window interfaces such as buttons, text-fields, and labels. Essentially allowing the user to see various interfacial attributes on screen
Parameters:	None
Return:	None

Name:	updateText
Description:	Free function used to initialize a multi-thread process that continuously sends and receives voltage and proximity data from the Python Pipe interface with the Raspberry Pi. Then it will update text-fields with voltage and proximity data it received
Parameters:	None
Return:	None

Name:	set_thresholds
Description:	Free function used to create a ThresholdWindow object. Also dictates whether to display the ThresholdWindow object. The function is tied to the menu option within the MyWindow object
Parameters:	None
Return:	None

Name:	set_quaternion
Description:	Free function used to create an instance of QuaternionWindow object. Also dictates whether to display the QuaternionWindow object. The function is tied to the menu option within the MyWindow Object
Parameters:	None
Return:	None

Name:	save_value
Description:	Free function used to create log files for voltage and proximity read from the Python Pipe. The function utilizes the time library native to the Python programming language
Parameters:	None
Return:	None

Name:	shutdown_alerts
Description:	Free function, used in conjunction with the ThresholdWindow, initiates a hard reset for voltage and proximity alarm by resetting the rgba information for all windows and resets the Boolean value for voltage and proximity flags
Parameters:	widget
Return:	None

Name:	on_button1_click
Description:	Free function used to initiate a false alarm regardless of the voltage and proximity value coming in from the Python Pipe from the Ground Base Unit. Used primarily for testing of the warning feature
Parameters:	widget
Return:	None

Name:	on_button2_click
Description:	Free function that ensures that multithreading process can initiate the abstracted plotting feature for voltage, which is not implemented currently due to graphing difficulties and time constraints
Parameters:	widget
Return:	None

Name:	on_button5_click
Description:	Free function used to turn off a false alarm specifically for the voltage regardless of the voltage value coming in from the Python Pipe from the Ground Base Unit. Used primarily for testing of the warning feature
Parameters:	widget
Return:	None

Name:	on_button6_click
Description:	Free function that ensures that multithreading process can initiate the abstracted plotting feature for proximity, which is not implemented currently due to graphing difficulties and time constraints
Parameters:	widget
Return:	None

Name:	on_button7_click
Description:	Free function used to turn off a false alarm specifically for the voltage regardless of the voltage value coming in from the Python Pipe from the Ground Base Unit. Used primarily for testing of the warning feature
Parameters:	widget
Return:	None

Name: on_button8_click
Description: Free function used to initiate a false alarm specifically for the voltage regardless of the voltage value coming in from the Python Pipe from the Ground Base Unit. Used primarily for testing of the warning feature
Parameters: widget
Return: None

Name: check_flags
Description: Free function used to check if the voltageFlag and proximityFlag are active, which activates or deactivates the warning feature within the GUI
Parameters: None
Return: None

Name: setUpSendPipe
Description: Free function that set up Python Pipe to the Ground Base Unit that sends voltage and proximity data to the Ground Base Unit
Parameters: child
Return: None

Name: setUpReceivePipe
Description: Free function that set up Python Pipe to the Ground Base Unit that receives voltage and proximity data from the Ground Base Unit
Parameters: child
Return: None

Class: ThresholdWindow

Functions:

Name: __init__
Description: Free function used to create an instance of the ThresholdWindow object. Additionally, it populates the window object with various interfacial attributes such as buttons, text-fields, and labels
Parameters: None
Return: None

Name: on_button1_clicked
Description: Free function used to set voltage threshold for incoming voltage data from the Pipe from the Ground Base Unit. VoltageFlag is a global variable used in conjunction with the MyWindow object
Parameters: widget
Return: None

Name:	changeColor
Description:	Free function used to change the rgba value that dictates the color, either green or red, for all windows to simulate the alert visually from the software side
Parameters:	None
Return:	None

Name:	issueAlert
Description:	Free function that checks if the voltageFlag or the proximity from the MyObject is set. If so, the warning feature within the ThresholdWindow is instantiated
Parameters:	None
Return:	None

Name:	on_button3_clicked
Description:	Free function used to set voltage threshold for incoming proximity data from the Pipe from the Ground Base Unit. ProximityFlag is a global variable used in conjunction with the MyWindow object Parameters: widget
Return:	None

Name:	on_button4_clicked
Description:	Free function used to set default voltage and proximity threshold values if the user is undecided regarding the thresholds. The thresholds is based on the Matrice 100 drone model.
Parameters:	widget
Return:	None

Name:	on_button5_clicked
Description:	Free function used to save the threshold values into log files. Interface with the time library native to the Python programming language to accomplish this task
Parameters:	widget
Return:	None

Name:	on_button6_clicked
Description:	Free function used to destroy the instance of ThresholdWindow object. Utilizes the Gtk library and its built-in destroy function to accomplish this task
Parameters:	widget
Return:	None

Class: QuaternionWindow

Functions:

Name:	__init__
Description:	Free function used to create an instance of the QuaternionWindow object. Additionally, it populates the window object with various interfacial attributes such as buttons, text-fields, and labels
Parameters:	None
Return:	None

Name:	read
Description:	Free function used to read in quaternion data, consist of x, y, z, roll, pitch, and yaw values from the configuration file *test.txt* shared with the ROS core from Ground Base Unit. The parsing and reading process is optimized with regular expression parsing to reduce run-time and latency
Parameters:	None
Return:	None

Name:	write
Description:	Free function used to write quaternion data, consist of x, y, z, roll, pitch, and yaw values to the configuration file *test.txt* shared with the ROS core from Ground Base Unit
Parameters:	None
Return:	None

Testing Modules

The testing process for *Graphical User Interface Unit* is done in conjunction with the *Ground Base Unit*. To effectively test every functionality for the *MyWindow*, *ThresholdWindow*, and *QuaternionWindow* modules, the voltage and proximity data coming through the Python Pipe interface form the *Ground Base Unit* is a necessity. Thus, to abstract the testing process from the *Ground Base Unit*, the *Graphical User Interface* has an *add alert* and a *remove alert* feature, which operates independently of the *Ground Base Unit*. The *add alert* and *remove alert* testing modules loads default voltage and proximity values into the *Graphical User Interface Unit*. Then, these voltage and proximity data is send to *ThresholdWindow* so that the warning feature can be adequately tested for the entire window interface. Additionally, the *read.check* interface loaded by default in Python ensures that the configuration file *test.txt* to be used by the *ROS core* within the *Ground Base Unit* always exist. Moreover, several *Ground Base Unit* test modules also checks the existence of the configuration file for quaternion data. Thus, the testing process is also abstracted from the *Graphical User Interface Unit* for the *Ground Base Unit* conversely. Furthermore, the entire *update* module interfacing between the *Graphical User Interface Unit* and the *Ground Base Unit* can be omitted to efficiently abstract the testing process even more for *Graphical User Interface Unit*.