DAATA Documentation

GT Off-Road Racing | Data Acquisition

Andrew Hellrigel

07/11/2021

Table of Contents

[1.0 Overview 3](#_Toc76865613)

[1.1 Introduction 3](#_Toc76865614)

[1.2 High Level File Structure 3](#_Toc76865615)

[1.2.1 DaataLogging 3](#_Toc76865616)

[1.2.2 DataAcquisition 3](#_Toc76865617)

[1.2.3 MainWindow 3](#_Toc76865618)

[1.2.4 Layouts 4](#_Toc76865619)

[1.2.5 Utilities 4](#_Toc76865620)

[2.0 Logging/Debugging 5](#_Toc76865621)

[2.1 Overview of Logging 5](#_Toc76865622)

[2.2 Using a Logger Object 5](#_Toc76865623)

[2.2.1 Creating a logger object 5](#_Toc76865624)

[2.2.2 Changing the level of a logger objects output 5](#_Toc76865625)

[2.3.3 Creating log messages 5](#_Toc76865626)

[2.3 When to use Log Messages 6](#_Toc76865627)

[2.3.1 DEBUG Messages 6](#_Toc76865628)

[2.3.2 INFO Messages 6](#_Toc76865629)

[2.3.3 WARNING Messages 6](#_Toc76865630)

[2.3.4 ERROR Messages 6](#_Toc76865631)

[2.3.5 CRITICAL Messages 6](#_Toc76865632)

[2.4 When to use Print Statements 6](#_Toc76865633)

[3.0 The “Data” Object 7](#_Toc76865634)

[3.1 Overview of the Data Object 7](#_Toc76865635)

[3.2 What are “Sensors” 7](#_Toc76865636)

[3.3 How to Access the Data Object 7](#_Toc76865637)

[3.4 How the Data Object is Fast and Protected 7](#_Toc76865638)

[3.5 Storing Information in the Data Object 7](#_Toc76865639)

[3.6 Retrieving Information from the Data Object 7](#_Toc76865640)

[4.0 The Main Window 8](#_Toc76865641)

[4.1 Overview of the Main Window 8](#_Toc76865642)

[5.0 Layouts 9](#_Toc76865643)

[5.1 Overview of Layouts 9](#_Toc76865644)

[6.0 Importing Data and the Communication Utility 10](#_Toc76865645)

[6.1 Overview of Importing Data 10](#_Toc76865646)

[7.0 Utilities 11](#_Toc76865647)

[7.1 Data Export Utility 11](#_Toc76865648)

[7.2 Plotting Utility 11](#_Toc76865649)

[8.0 Revision History 12](#_Toc76865650)

# 1.0 Overview

## 1.1 Introduction

DAATA (Data Acquisition and Telemetry Application) is the application that has been developed by GTOR from scratch that we use for running essentially all the testing and data collection that we do. It also performs live data visualization and diagnostics since it can communicate with the systems on our car wirelessly.

DAATA was developed because there was a need for a unification of the way that tests are run within our team and with how data is managed. In the past, when a test needed to be run, Arduino code was developed that would communicate with all the sensors needed for the test, and the data collected from those sensors was stored onto a micro-SD card. A MATLAB script was then developed to parse the binary data on the SD card and turn it into an excel file so that the person that needed the data could analyze it. This caused problems because for one, there was not a good way of knowing that all of the sensors used for the test were working properly until when the data was being analyzed afterwards, potentially wasting hours of testing time. There was also an issue because if the needs of the test changed at all, the entire system would need to be changed. The Arduino code would need to be updated to support the addition or removal of a sensor, and then the way that the data was written to the SD card would need to be changed to support the extra or lesser amounts of data. Finally, the MATLAB script would need to be changed to parse the data correctly. This was not scalable in any way and was very error prone due to the tediousness in making changes to the way the test was run.

The advantage of the custom software that we have made is that it is smart and can automatically configure how it stores data based on what sensors it detects are connected to it. This can be done thanks to the communication protocol that our team developed specifically for this purpose. Since the actual details of this smart communication protocol are somewhat complex, refer to the separate documentation on our Comms Utility for a more detailed description.

## 1.2 High Level File Structure

The structure of DAATA is separated into five main folders.

### 1.2.1 DaataLogging

The DaataLogging folder is where our logging objects are created and managed. This folder doesn’t actually have any software in it that helps the application function, it is solely for managing logger objects to be used for debugging and software development.

### 1.2.2 DataAcquisition

The DataAcquisition folder is where most of the “back-end” data manipulation occurs. Everything in this folder runs in a separate thread that is launched from the Main Window. It manages communication with an external microcontroller, and it also stores the data object that manages all collected sensor data.

### 1.2.3 MainWindow

The MainWindow folder can be thought of somewhat as a container that houses everything else that goes on in the code. It launches the DataAcquisition thread, it stores all the layouts and runs an update timer that manages when the information within each of the layouts gets updated.

### 1.2.4 Layouts

The Layouts folder has a subfolder within it for each layout within the application. Within each subfolder is all of the code that creates the graphic elements for each layout and also describes the function of each graphic element and what part of the application those elements control (such as buttons, checkboxes, etc.).

### 1.2.5 Utilities

The utilities folder contains miscellaneous sections of code that are used mostly by layouts. Everything that is within one of the subfolders in the Layouts folder must be unique to that layout, however, everything in the Utilities folder can be used by multiple layouts. Examples include the plotting utility, the data export utility, pop-up boxes, etc.

# 2.0 Logging/Debugging

## 2.1 Overview of Logging

This section is included first because, no matter where you are working in the code, the logger and debug tools used in the code will be necessary elements that you will interact with to solve problems and to develop clean code that allows other people to understand what is happening.

Most people that are familiar with Python or really any coding language in general will most likely be familiar with print statements as they are the most basic form of debugging tool available for software developers. However, with a more complex codebase and with more contributors working on a single project, a more sophisticated debugging tool is required. This is where the use of a logger comes in.

In short, a logger is basically a print statement that has much more control over how a message gets displayed on the debug terminal. The main advantage is that a logger object can be created for distinct sections of the code so that when you are focused on one aspect of the code, other log messages can be disabled so that the debug terminal doesn’t get cluttered with useless information. The other advantage is that log messages can have levels of importance associated with them which dictate whether or not the message gets displayed or not. There are five levels of log messages: DEBUG, INFO, WARNING, ERROR, and CRITICAL.

Say you are working on performance improvements for the graphing utility within python. You likely don’t care about what is happening in the section of the code that handles the communication with a connected microcontroller (the DataImport section). The logger object that is used in this section of the code can be set to only display log messages that are at a level of WARNING or higher that way you can still make sure that section of the code is working properly, but you also don’t get all the regular log messages from that section of the code that might be more important to someone working on performance improvements for the microcontroller communication.

Hopefully, the scenario gives an appreciation for the need for isolated but efficient ways of getting the necessary debug information in the debug terminal. In the following sections I will cover how to actually use the logger, general rules of thumb for when and where to use log messages and what level messages should be set to, and also when to use generic print statements over a log message.

## 2.2 Using a Logger Object

### 2.2.1 Creating a logger object

### 2.2.2 Changing the level of a logger objects output

### 2.3.3 Creating log messages

## 2.3 When to use Log Messages

### 2.3.1 DEBUG Messages

### 2.3.2 INFO Messages

### 2.3.3 WARNING Messages

### 2.3.4 ERROR Messages

### 2.3.5 CRITICAL Messages

## 2.4 When to use Print Statements

# 3.0 The “Data” Object

## 3.1 Overview of the Data Object

Since the main job of DAATA is to manage collected data and either store it for post processing or to provide a visual representation of the data, the actual management of the data is a very important part of the application.

For the data management to be successful it needs to accomplish a few things.

1. The storage and retrieval of information must be fast enough to support the rate of data collection that we need.
2. The storage and retrieval of information must be protected to avoid any data corruption
3. The storage method of data must be adaptable to support the various amounts of sensors that we use
4. The retrieval method of data must be smart enough so that the application can know what sensor data is available at any point in time to be used

For each of these points I will go into detail about the method in which they are accomplished and why that is an appropriate method for us (albeit none of it is perfect and I’m sure there are better ways of doing everything that we did, so keep in mind that any of this can and should change if we need it to be better in some way).

All data that is stored in the application is done so through the data object. It is not dissimilar to a python dictionary with a bunch of extra features that make the storage and retrieval of information smarter and easier to use (actually, at its core it is just a dictionary of Sensor objects with each Sensor object storing a list of the data it contains).

## 3.2 What are “Sensors”

## 3.3 How to Access the Data Object

## 3.4 How the Data Object is Fast and Protected

## 3.5 Storing Information in the Data Object

## 3.6 Retrieving Information from the Data Object

# 4.0 The Main Window

## 4.1 Overview of the Main Window

# 5.0 Layouts

## 5.1 Overview of Layouts

# 6.0 Importing Data and the Communication Utility

## 6.1 Overview of Importing Data

# 7.0 Utilities

## 7.1 Data Export Utility

## 7.2 Plotting Utility

# 8.0 Revision History

7/11/2021 (Andrew Hellrigel) – Created first revision of document