Internship DNANudge Product Design Specification

Product Design Specification

- We want a live plotting program which records and stores data through the Python console, where the user is able to choose whether to store and/or to plot the data.
- The user should also be able to choose what data is being plotted at any time during operation – no need to choose what to store (if store mode then store everything)
- A restart and calibration sequence should be able to be triggered at any time from the microcontroller reset button, where all current saved data is cleared/a new set of data is stored elsewhere so that the machine does not store 2 sets of data in the same file, causing confusion

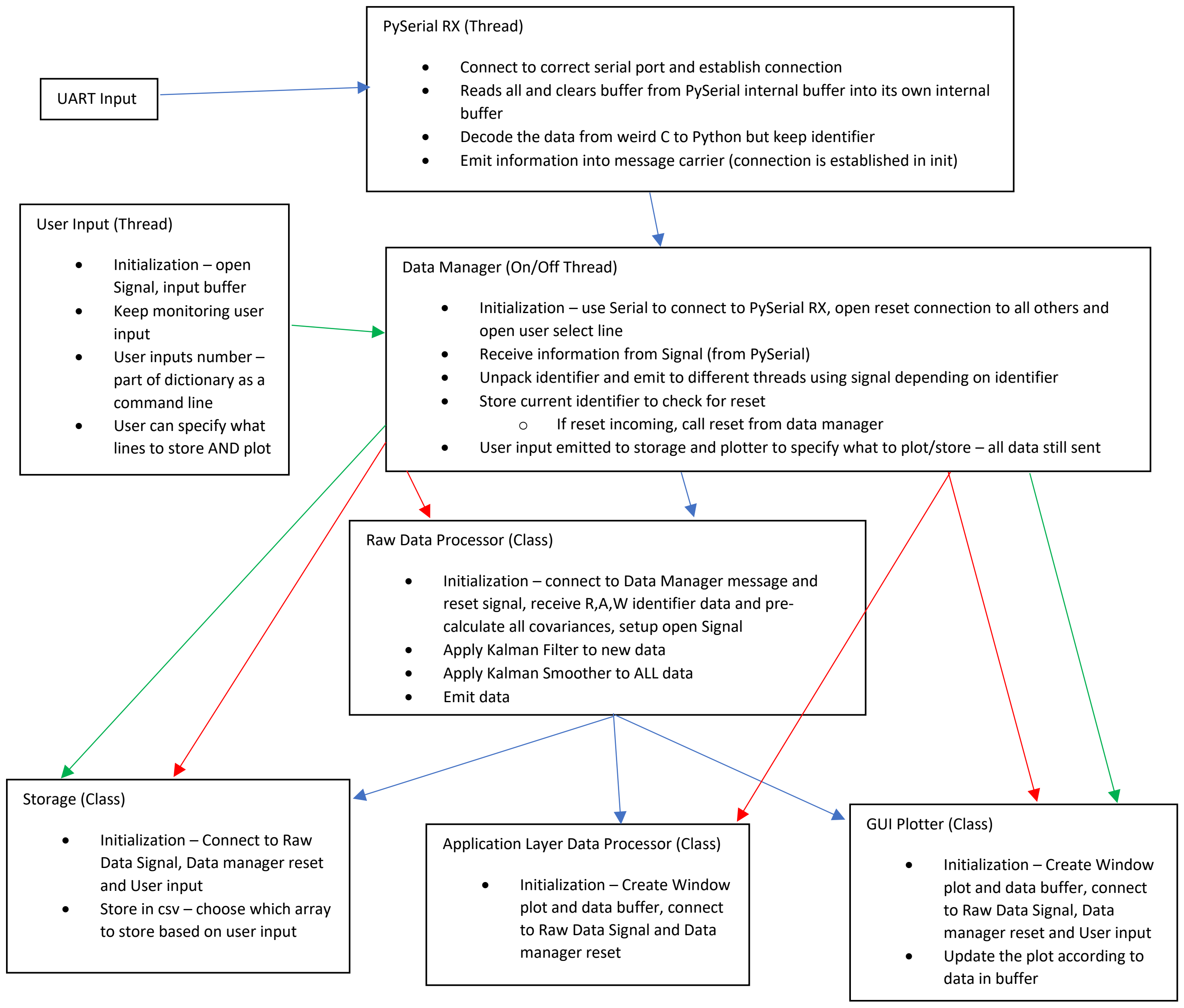
Main components

- A main program which initializes all of the threads and calls the looping functions
- Inputs:
    - A UART connection to transfer data from the microcontroller to the computer, either by serial connection or BLE
    - A constant user input from the Python console
- Outputs:
    - A fast, sensitive live plot of whatever variables the user wants (must support at least 9 live plots without lagging at 20Hz)
    - Data storage for each new reset – Python makes a new text file inside a repository upon startup/reset

Threads/Program Architecture

Main Program

- Initializing variables, call UART and plot functions (NOT loop in thread)
- Initialize User Input
- Initialize UART (sets up signal connect – leave open for Data Manager to pick up)
- Initialize Data Manager (Chooses to connect to UART through Signal into raw message buffer and initialize Signal for each identifier)
- Initialize Raw Data Processor
- Initialize Application Layer Data Processor
- Initialize data storage



PySerial RX (Thread)

- Connect to correct serial port and establish connection
- Reads all and clears buffer from PySerial internal buffer into its own internal buffer
- Decode the data from weird C to Python but keep identifier
- Emit information into message carrier (connection is established in init)

User Input (Thread)

- Initialization – open Signal, input buffer
- Keep monitoring user input
- User inputs number – part of dictionary as a command line
- User can specify what lines to store AND plot

Data Manager (On/Off Thread)

- Initialization – use Serial to connect to PySerial RX, open reset connection to all others and open user select line
- Receive information from Signal (from PySerial)
- Unpack identifier and emit to different threads using signal depending on identifier
- Store current identifier to check for reset
    - If reset incoming, call reset from data manager
- User input emitted to storage and plotter to specify what to plot/store – all data still sent

Raw Data Processor (Class)

- Initialization – connect to Data Manager message and reset signal, receive R,A,W identifier data and pre-calculate all covariances, setup open Signal
- Apply Kalman Filter to new data
- Apply Kalman Smoother to ALL data
- Emit data

Storage (Class)

- Initialization – Connect to Raw Data Signal, Data manager reset and User input
- Store in csv – choose which array to store based on user input

Application Layer Data Processor (Class)

- Initialization – Create Window plot and data buffer, connect to Raw Data Signal and Data manager reset

GUI Plotter (Class)

- Initialization – Create Window plot and data buffer, connect to Raw Data Signal, Data manager reset and User input
- Update the plot according to data in buffer

User Command Line

Command: Store

- store start – starts storing data in a new CSV file – if used again before stop it executes stop and starts new
- store stop – stops data storage

Command: Connection

- connection setport ___ - Sets port to a new port and restarts the whole system
- connection closeport – closes the port and stops all function

Command: Plot

Plot is a very adaptable command line command – it allows the user to add to the plot whatever they want using words. For example:

$$\text{plot smooth acc norm}$$

plots a smoothed magnitude of acceleration. If we then type:

$$\text{plot filter ang}$$

it will add to the plot all filtered data which is angular velocity.

This means that the user can type these combinations in any order, with as little or many following arguments as the user wants.

The groups available are:

- raw, filter, smooth, temp, all
- x, y, z , norm
- vel, acc, ang, jerk

If the user enters an error, it will keep the existing plot. If the user enters something already existing, it will also keep the same plot.

Command: Remove

Does the exact same thing as plot but removes the thing you enter.

Command: Terminate

Terminates the program and closes all threads

Future Improvements

- The program still does not detect an absence of data, e.g. wired connection error in the MCU. The next step for robustness is to improve on this and communicate with the MCU via UART to resend and re-initialize.
- The covariance matrix check does not check for negative semidefiniteness, so that can be implemented
- The plotter does not support multiple plots/windows yet, so this should be implemented in the future
- The application processor is not coded yet, so this could be used for anything, e.g. status recognition, pedometer, machine learning, etc.
- The modules have not been made to fully accept other sources of data, e.g. UV sensor yet, so the code needs to be a bit easier to expand to other sensors
- The plotting speed and threading structure can definitely be optimized in order to make the program run smoother
- Other forms of processing, e.g. Particle filtering or EKF models (non-linear filtering) can still be implemented, and the user may choose to apply this through the command line
- A help command has not been implemented, so that can definitely be added
- The embedded part is not fully commented, and no documentation has been made for it.