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
 - o 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

Store in csv - choose which array

to store based on user input

PySerial RX (Thread) Connect to correct serial port and establish connection Reads all and clears buffer from PySerial internal buffer into its own internal **UART Input** 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 Data Manager (On/Off Thread) Signal, input buffer Initialization – use Serial to connect to PySerial RX, open reset connection to all others and Keep monitoring user open user select line input Receive information from Signal (from PySerial) User inputs number -Unpack identifier and emit to different threads using signal depending on identifier part of dictionary as a Store current identifier to check for reset command line If reset incoming, call reset from data manager User can specify what User input emitted to storage and plotter to specify what to plot/store – all data still sent lines to store AND plot Raw Data Processor (Class) Initialization - connect to Data Manager message and reset signal, receive R,A,W identifier data and precalculate all covariances, setup open Signal Apply Kalman Filter to new data Apply Kalman Smoother to ALL data Emit data Storage (Class) GUI Plotter (Class) Initialization – Connect to Raw Application Layer Data Processor (Class) Data Signal, Data manager reset Initialization - Create Window Initialization - Create Window and User input plot and data buffer, connect

plot and data buffer, connect

to Raw Data Signal and Data

manager reset

to Raw Data Signal, Data

data in buffer

manager reset and User input

Update the plot according to

User Command Line

Structure:

user_input = command/option

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 (if it already exists don't do anything)

- plot raw Adds plot of all raw data (including norm)
- plot filter Adds plot of all filtered data (including norm)
- plot smooth Adds plot of all smoothed data (including norm)
- plot x
- plot y
- plot z
- plot norm Adds plot of all norms
- plot acc Adds all acceleration plots
- plot ang Adds all angular velocity plots
- plot vel Adds all velocity plots
- plot jerk Adds all jerk plots
- plot [index] e.g. plot 1,2 gives temp and acceleration x

Command: Remove

- remove raw Removes plot of all raw data (including norm)
- remove filter Removes plot of all filtered data (including norm)
- remove smooth Removes plot of all smoothed data (including norm)
- remove norm Removes plot of all norms
- remove acc Removes all acceleration plots
- remove ang Removes all angular velocity plots
- remove vel Removes all velocity plots
- remove jerk Removes all jerk plots
- remove [index] e.g. remove 1,2 removes temp and acceleration x