Software Architecture Summary

# Introduction

This document provides a high-level explanation for software developers of how the software works in the Cardiac Catheterization Testing Apparatus (CCTA). The software consists of two main parts: a MATLAB application and Arduino firmware. These two systems communicate with each other in real time to simulate and control physiological pressure and flow profiles.

# System Overview & Online Resources

* The **MATLAB application** is responsible for the user interface, handling serial communications to and from the Arduino, and data logging. It provides tools for the user to configure system settings, start or stop the test, and monitor sensor data.
  + The app was made using the MATLAB App Designer (documentation here: <https://www.mathworks.com/help/matlab/app-designer.html>).
* The **Arduino firmware** handles low-level interaction with the hardware. It reads data from pressure and flow sensors, receives commands from MATLAB, and drives the pump accordingly.
  + Arduino documentation is available here: <https://docs.arduino.cc/>

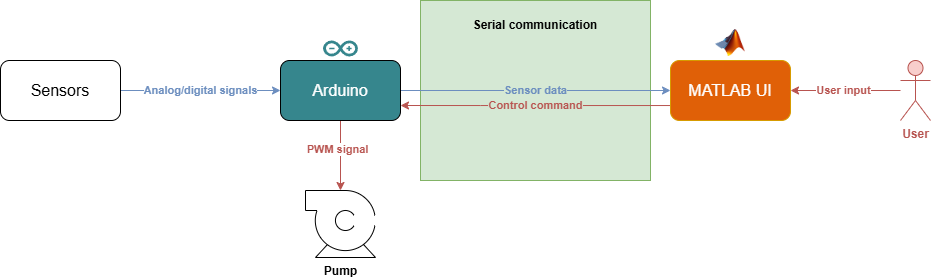
Both systems operate using a main loop structure:

* On the Arduino side, this is the loop() function — for the Arduino, this is where nearly all code executes, so is a good first checkpoint in case a developer wants to change anything.
* On the MATLAB side, this is the runMainLoop() function — for MATLAB, this is a major component but not the only one. There are many "callback functions" which are executed on certain UI actions, and these are disconnected from the main loop.

These functions coordinate the flow of data and commands between hardware and software.

# Code Flow and Communication Summary

## High-Level Data Flow



## Code Execution Path

1. **Arduino Data Collection**:
   * Arduino reads pressure and flow data from sensors
   * Data is packaged and sent to MATLAB over serial in each iteration of the Arduino main loop
2. **MATLAB Processing**:
   * Incoming serial data is parsed and displayed in the UI
   * Rolling averages and setpoint comparisons are performed
   * Any UI interactions are handled using appropriate callbacks (e.g. for clearing data, exporting data)
   * Depending on the pump control mode, the user can send appropriate serial commands are sent back over to the Arduino, containing either:
     + In **Manual Mode**, a direct PWM value
     + In **Auto Mode**, a PID control configuration (i.e. current value, setpoint, PID coefficients)
     + In **Pulsatile Mode**, BPM and amplitude values for sinusoidal oscillations
3. **Arduino Control System**:
   * Parses commands from MATLAB and updates pump behavior accordingly
   * Modes are handled differently:
     + **Manual Mode**: Direct PWM
     + **Auto Mode**: PID-based PWM
     + **Pulsatile Mode**: Sinusoidal signal based on BPM/amplitude

# Conclusion

This document outlines the main flow and structure of the system software, showing how the MATLAB UI and Arduino work together in a loop to simulate cardiac conditions. Most implementation details, including variable usage and edge case handling, are clearly documented in the code itself.

For future modifications, such as integrating new sensors or actuators, changes will be required both in the Arduino script (for signal processing and hardware control) and in the MATLAB application (for displaying values and sending commands).

# 