System Components:

- Nucleo_H723ZG Board
  - This has a maximum CPU clock frequency of 550 MHz, and 3 ADCs which are used to sample power supplies/plasma parameters
  - Power/communication is over the micro usb port labeled “power” on the silkscreen. The “user” usb is unused
  - Currently configured to communicate over UART at 6.875 Mbaud

STM32 Hardware Usage:

- ADC 1&2
  - These are highspeed (ideally) 16 bit ADCs that are used to sample bridge current, V_{L1}, V_{L2}, V_{S1}, V_{S2}
  - Theoretically should be able to sample well over 1 MSPS, but currently are only operating at 250 kSPS
  - See ST documentation for setting registers
  - Accessed through DMA and HAL provided functions
- ADC 3
  - Used for power supply/temp monitoring

Software Architecture:

The software flow and layout for the microcontroller can be a little confusing. The basic layout/low level access functions were laid out by a previous group.

- **Main files**
  - There are only two main “.c” files that run the system (A good future refactor would be to break out functionality into smaller files, ex: remoteControl.c, hbridge.c, ADC.c, etc)
    - Main.c
      - Mostly generated by CubeMX. Initializes hardware, starts PlasmaDriver.c
    - PlasmaDriver.c
      - Contains all of the actual functionality of the software
    - Other files
      - Generated by CubeMX. The .ioc file can be used to change clocks, register settings, ADC settings, UART etc.
- **High-level Flow**
  - Power supply checks
    - When the system is powered on, it initializes the hardware and checks for proper voltage on the low DC supplies (15V, 3.3V)
    - If successful, the 3.3V switch rail is switched to activate the opamp buffering the H-Bridge PWM
  - Text debug menu
    - A text over UART menu is immediately entered on startup
    - Provides direct low level access to H-Bridge settings etc

        - Accessible using any serial terminal set to the proper baud rate
    - Remote Control Mode
        - When interfacing via the remote control protocol, the host sends a '~' to put the system in the proper state. The remoteControl() function is entered
        - "\n" terminated strings are used to send commands. High level access is provided in the plasmaInterface.py class.
        - Table of supported commands provided at end of document
        - Operates in a state machine using the rc_state struct.

- **ADC Measurement Flow**
    - High level access to ADC measurements are provided by measureBridgePlasmaADC12 and measureVoltagesTemperaturesADC3()
        - Handle ADC start, and place the ADC data in the respective c struct member (within sADC)
        - ADC measurement is triggered via TIM1 (control signal B) to make sure the points of interest described in the frequency correction algorithm are captured.
    - doneMeasureingBridgePlasmaADCx()
        - Called after measureBridgePlasmaADC12() completes (i.e ADCs are done reading)
        - Sets the sADC.xxx_reading flag back to 0
    - convertADCxData()
        - Converts the raw ADC data into the real currents/voltages present at the bridge (or at the power supplies for ADC3)
        - Some of these conversions need to be recalculated, they are inaccurate.
    - printADCxData()
        - Prints ADCx data in a user readable format to UART
        - This is not used for data logging, printHbridgeDatalogging() is called to create a CSV format instead

- **High level Access Functions Provided in PlasmaDriver.c**
    - A large number of useful functions for controlling the system are provided. Most are fairly self-explanatory and are documented in the PlasmaDriver.c file.

- **Data logging**
    - When the system is used for remote control, the ADC12 datalog can be accessed via printHbridgeDatalogging()
    - Logged parameters
        - All bridge metrics (current, voltages, TIM1 status (control B))
        - Frequency correction points for debugging
            - This was added to aid in development of the frequency control algorithm. The log (or real-time plotting in the gui) can be used to

    - see exactly which points the algorithm select to use for calculating the frequency adjustment.
    - Our team ran out of time to fully use this functionality to diagnose issues with the frequency correction algorithm, but I suspect the current issue has to do with the wrong points being selected. The logging of these values should be extremely useful in modifying the algorithm.
- **Automatic Adjustment Calculations**
  - Both the frequency and voltage correction functions (freqCorrection() and voltageCorrection()) are documented within the code. The voltage correction is untested.
  - Voltage correction
    - Testing will need to be performed to determine the acceptable range of deadtime values. Initial testing suggested the range of 1%-20% would be the most useful for voltage control.

H-Bridge Parameters:
- The H-Bridge is driven by two complementary PWM signals (TIM1)
  - The generated AC frequency and voltage can be controlled from software by adjusting the timer settings
- Frequency: The frequency of the PWM signals directly controls the AC frequency
- Deadtime: This is somewhat related to pulse width. It is the measurement (in %) of the amount of time between PWM A and PWM B that the H-Bridge is off
  - Example: If deadtime is 0% (not permissible in real world), then there is no time where both sides of the H-Bridge are off
- Setting H-Bridge Parameters
  - The H-Bridge parameters are stored in a c struct within PlasmaDriver.c called sHbridge
  - The members of the struct are modified to the desired value, then the function programHbridge() is called to calculate the necessary TIM1 settings.

Questions/Contact: lukascrockett@u.boisestate.edu

Table 1: Remote Control Protocol

Statement Description	Statement Format	Reply Format	Notes
Initialize Communication	‘~’	‘~’ is sent on successful initialization	
Power Supply Query/Toggle	Query: “p?3.3”, “p?15”, “p?hv” Command: “p!lv”, “p!hv”	“on”, “off”	
Plasma Query/Toggle	Query: “s?” Command: “s!”	“on”, “off”	
Deadtime Query/Set	Query: “d?” Command: “d!xxx”	Query: “xx” where xx is the current deadtime Command: “ok”	
Voltage setpoint Query/Set	Query: “v?” Command: “v!xxx”	Query: “xxx” where xxx is the voltage setpoint Command: “ok”	Voltage is sent in units of Volts
Frequency setpoint Query/Set	Query: “f?” Command: “f!xxx”	Query: “xxx” where xxx is the frequency setpoint	Frequency is sent in units of Hz
ADC3 Query (Supplies/temps)	Query: ‘a’	Query: prints the current ADC3 readings in a CSV format	
Datalog Query	Query: “l?” Requests ADC1/2 data, “lh” Requests data header	Query: requested info printed in CSV format	Data header includes data position and units
Set Automatic Adjustments	Command: “mfx”, “mvx” where x is 1 or 0	Command: echoes ‘1’, or ‘0’	
Stop Plasma	Command: ‘q’	No reply	Plasma can also be stopped by toggling “s!”
Shut down system	Command ‘z’	No reply	Stops plasma (if running) and powers down the supplies.