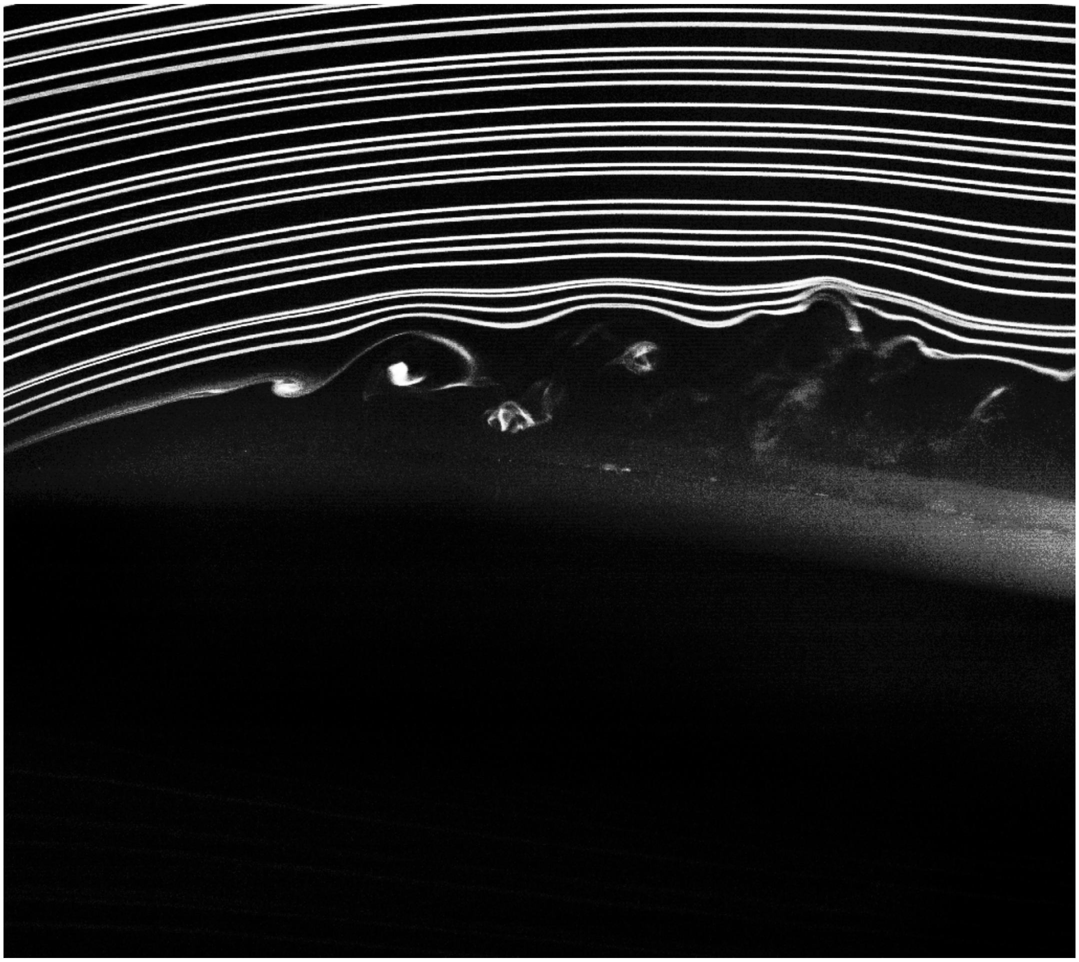
UTIAS SMOKE FLOW VISUALIZATION SETUP DOCUMENTATION



Author: Daniel Asadi
University of Toronto Division of Engineering Science
daniel.asadi@mail.utoronto.ca
+1 (416) 666-8615

Supervisors: Suraj Bansal
University of Toronto Institute for Aerospace Studies
suraj.bansal@mail.utoronto.ca
+1 (647) 331-4833

Philippe Lavoie
University of Toronto Institute for Aerospace Studies
lavoie@utias.utoronto.ca
+1 (416) 667-7716

REVISION HISTORY

REVISION	COMMENTS	AUTHOR	DATE
I	Initial release.	D. ASADI	28 AUG 2021
II	Reformated sections	D. ASADI	8 SEPT 2021

TABLE OF CONTENTS

1.0 INTRODUCTION

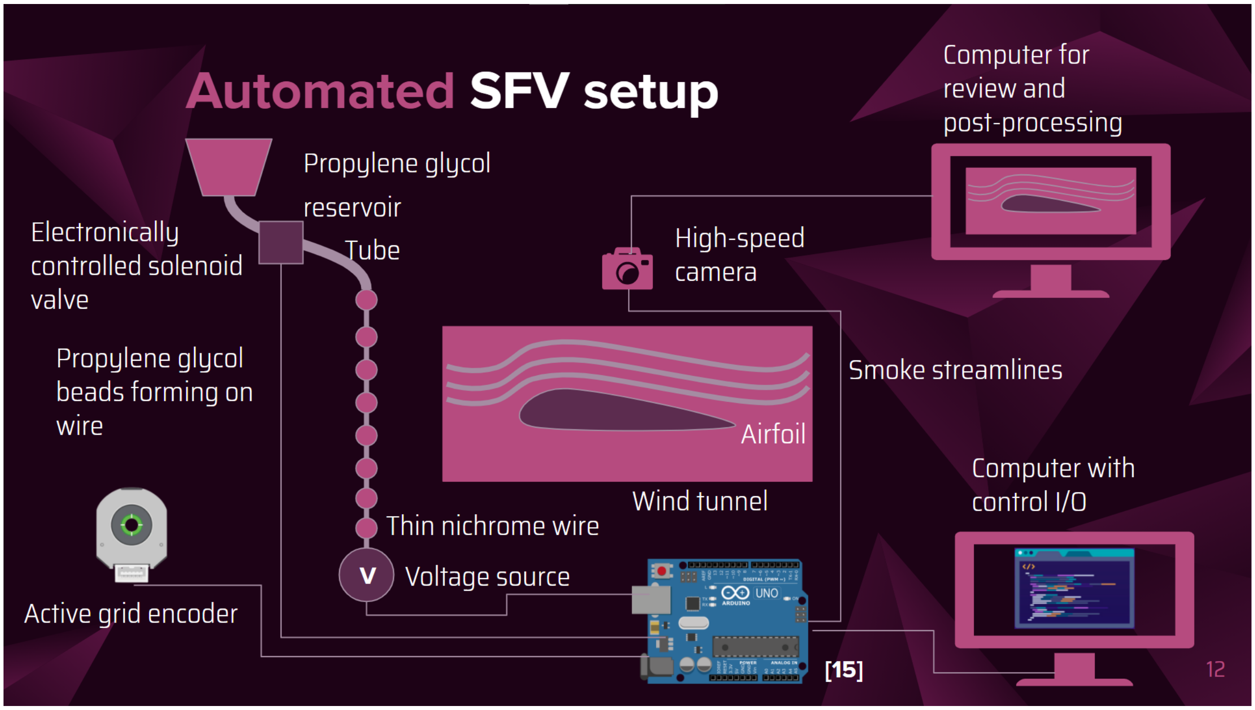
The UTIAS Smoke Flow Visualization (SFV) setup at the FCET lab wind tunnel enables a more efficient workflow with additional capabilities compared to the standard SFV setups. The SFV technique consists of a nichrome wire that is coated in propylene glycol and is heated to create lines of smoke through the wind tunnel. It allows for the visual identification of key features such as boundary layers, laminar or turbulent flow and vortex shedding. In the past, each instrument in the setup was controlled manually and was only effective for flow speeds under 5 m/s, therefore limiting the precision of equipment usage for unsteady conditions and the range of possible cases to study.

This setup has been improved by connecting to and electronically actuating all SFV I/O components - the valve that dispenses propylene glycol, the high-speed camera, the voltage source to heat the wire and an encoder to track the active grid phase angle - to an Arduino microcontroller and with the use of Arduino code, python code controlled by the user, and PFV4 software to view the images, the SFV process is much more efficient and automated. A future addition which work began on but was not completed is the implementation of capacitors to rapidly discharge high levels of current for usage above 5 m/s.

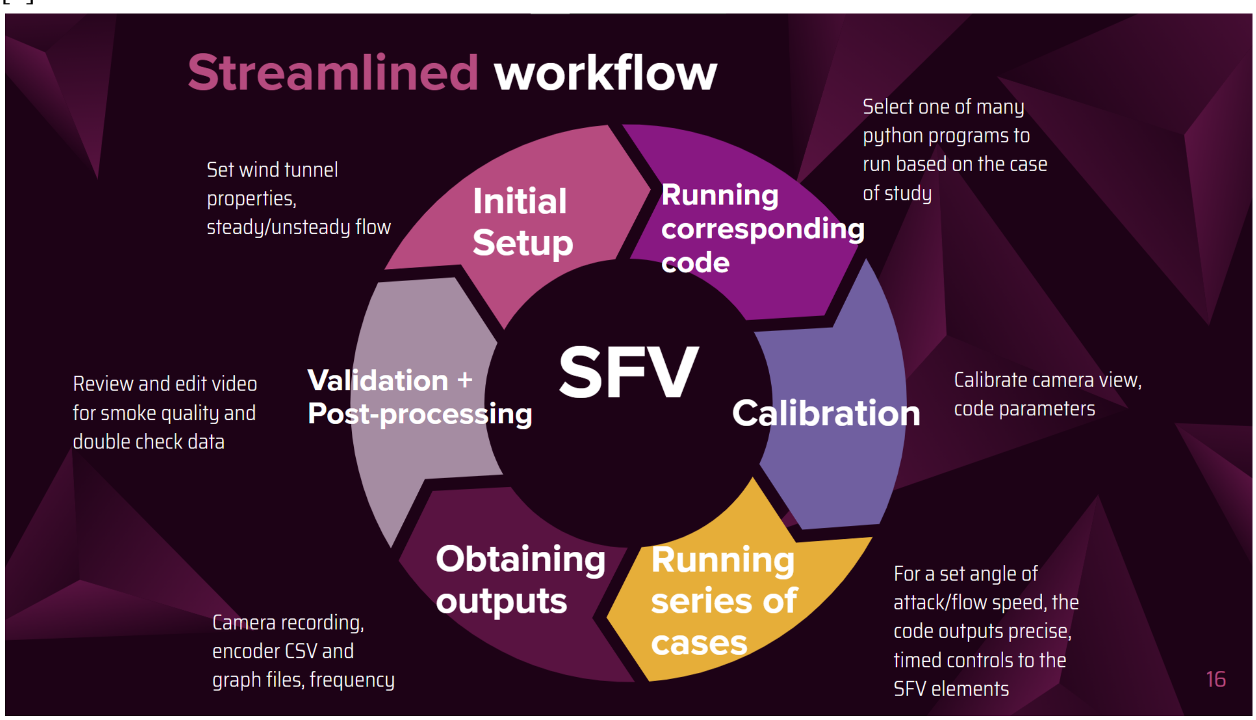
This documentation will serve as a guide on how to operate and maintain this SFV setup and inform where to find more resources for this project.

2.0 OVERVIEW OF THE SETUP

The following two images provide a visual overview of the SFV setup and the expected workflow.



[1]

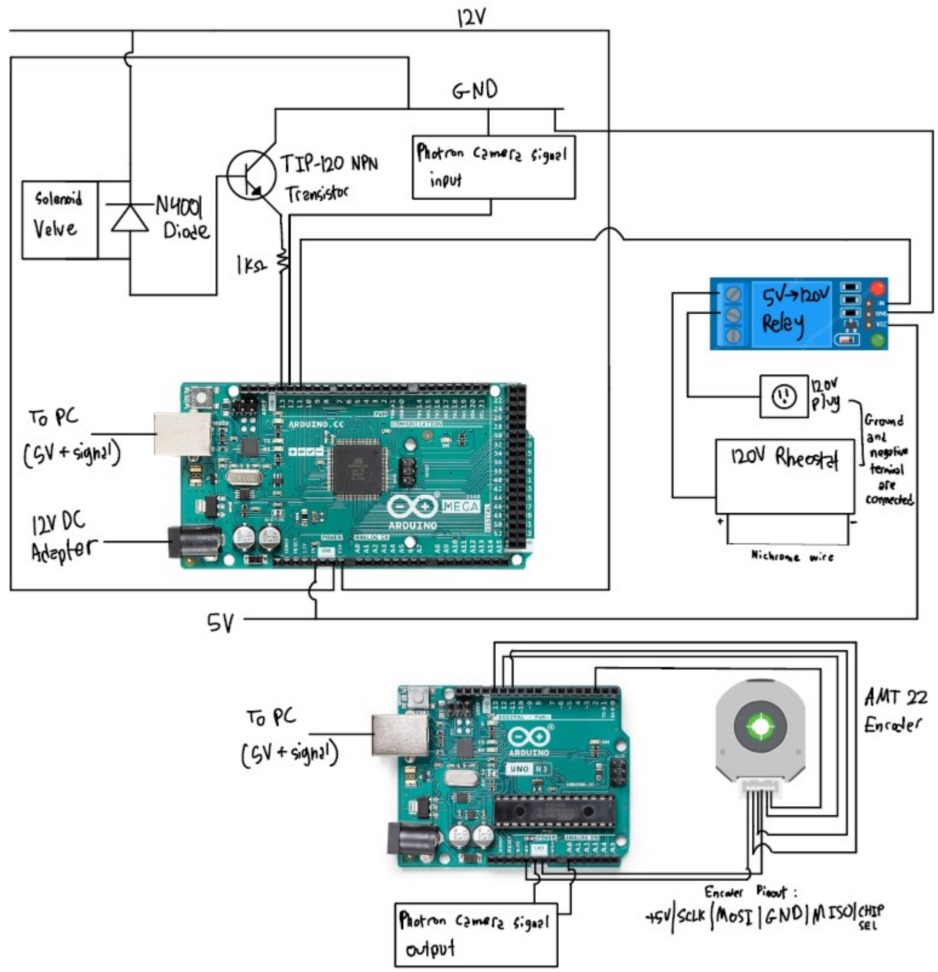


[1]

3.0 HARDWARE SETUP

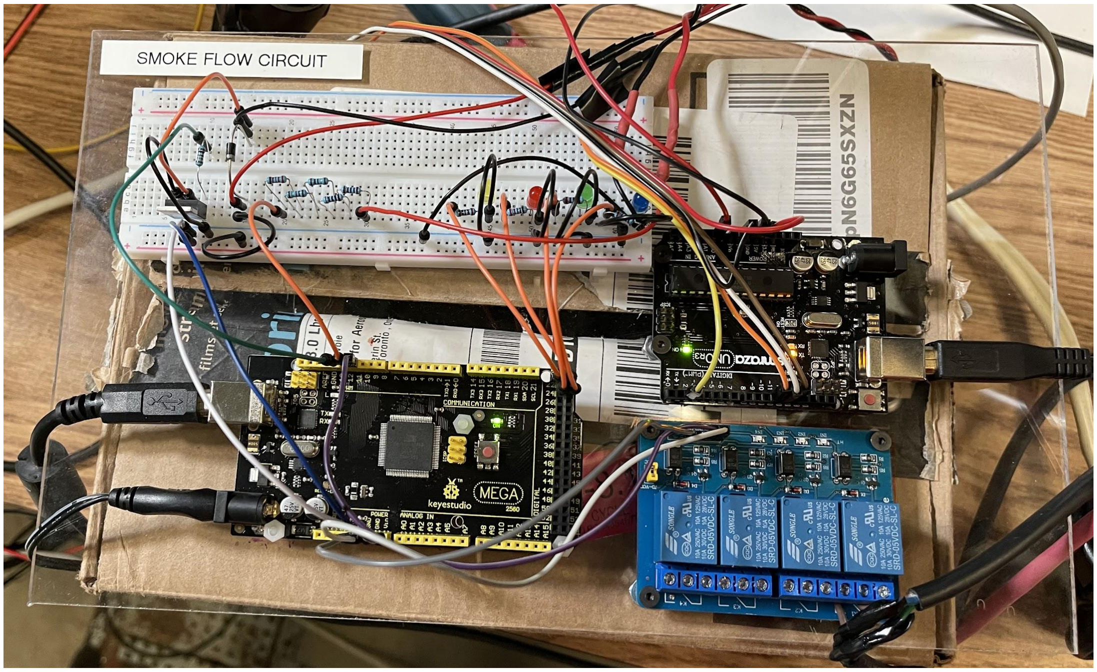
3.1 SFV Circuit Setup

At the core of the SFV circuit, there are 2 Arduino microcontrollers. The Arduino Mega controls the actuation of the SFV elements: the solenoid valve for the propylene glycol, the 120v voltage source through a 5v relay and the Photron camera through bnc connection. It is connected by usb to the computer with the python code and camera software and also is connected to a 12v DC adapter for power and instructions. The Arduino Uno is connected to the active grid encoder to constantly read the active grid data, in order to trigger the smoke at a specified phase angle and save CSVs, plots of this data and to verify the active grid frequency. It is also connected by usb to the computer. In the image below the circuit diagram is shown.



SFV circuit diagram [1]

Below is the circuit in the setup with a box beneath it for housing extra cables.



SFV Arduino circuit [1]

<u>3.2 Proplyene Glycol Reservoir Setup</u>

The image below shows the desired setup for the propylene glycol reservoir as well as the top attachment point for the nichrome wire. The rig is connected by two screws at the top of the wind tunnel test section, the solenoid valve is connected to the circuit and the reservoir is filled with propylene glycol. The tube is laid in the funnel. The nichrome wire is firmly wrapped around the spring and is routed through the funnel and the hole in the test section.

Propylene glycol resevoir [1]

3.3 Nichrome Wire Setup

The bottom end of the nichome wire is wrapped around an allan key and then pinned down by the two thin metal plates as shown below. The nichrome wire is then electrically connected to the voltage source at both ends.

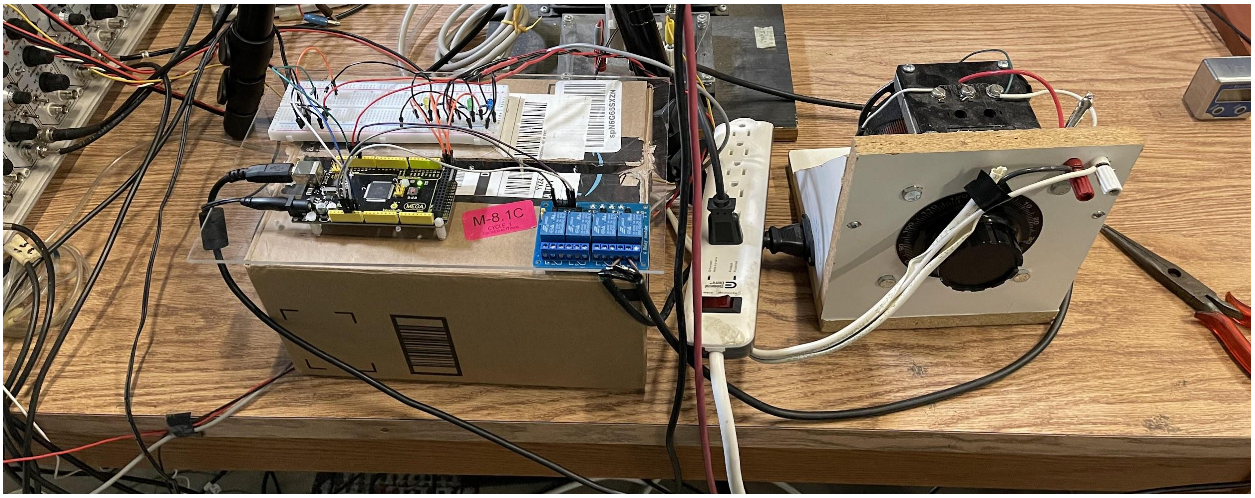
NOTE: In case of the nichrome wire being irregular and not smooth, a 2kg weight is used to straighten it out.

Nicrhome wire setup [1]

3.4 Voltage Source Setup

The voltage source was setup as seen in the image below. The 120V plug connected to the 5V relay is connected to the outlet, preferably an extension cord with an ON/OFF switch as an extra precaution. The other end of the plug connected to the 5V relay is connected to the voltage source. The positive/negative terminals are connected to the ends of the nichrome wire. After setup, the dial is set to 30V, but can be varied ±10V based on Reynolds Number.

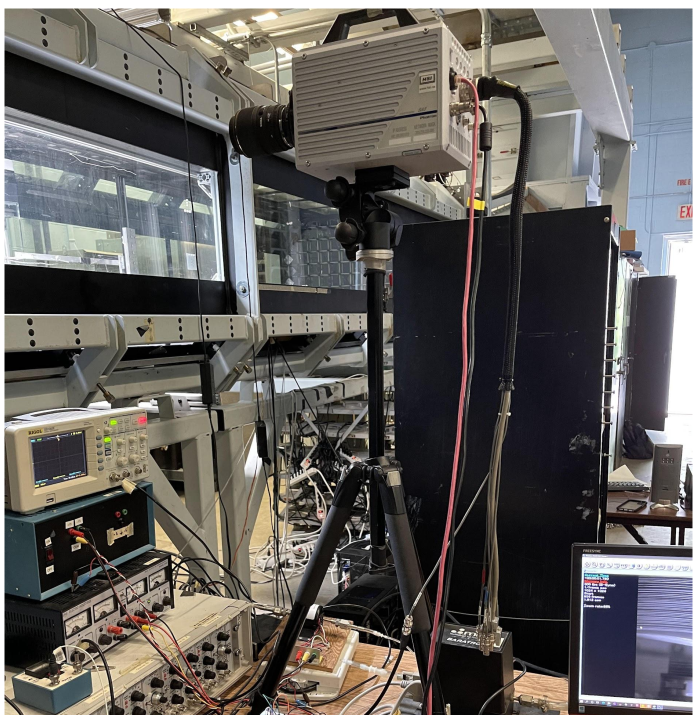
NOTE: Make sure the voltage source's dial is set to zero during setup and it is not powered on. This piece of equipment works with high voltages and is dangerous. By default the Arduino has a low signal for the 5V relay and thus the voltage source is always off except for when the code turns it on. This can be verified with a voltmeter. In addition make sure no exposed wires are present in the circuit which can cause a short circuit and damage components.



Voltage source next to Arduino circuit [1]

3.5 Photron Fastcam SA5 Setup

The Photoron Fastcam was setup by referencing its instruction manual, which is located in the Github info folder. The three main connections are the ethernet cable connected to the computer with the software (PFV4) to send images, a power cable, and a bundle of bnc cables. The circuit was connected to the "TRIG TTL IN" BNC cable. Finally the lens is mounted.

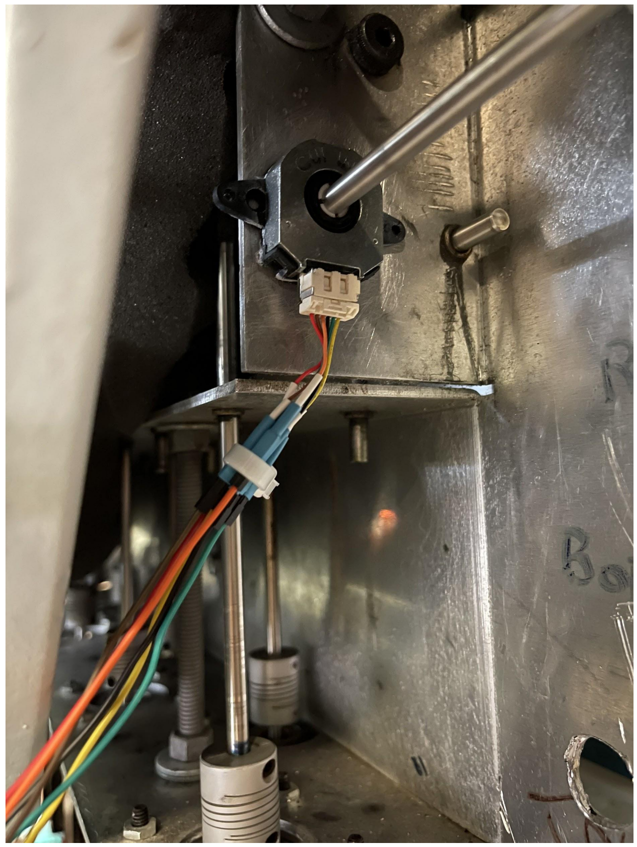


Photron Fastcam SA5 setup [1]

3.6 Encoder Setup

The encoder is already connected to the active grid, only the wiring needs to be connected with a 6 pin cable to the Arduino.

NOTE: In case of encoder displaying erroneous values, reset the wire connection and also reupload the arduino code.



Active grid encoder [1]

4.0 SOFTWARE SETUP

4.1 Github and VScode Setup

This project was published on Github. In the Github README there is an overview of the project and brief descriptions of each file and folder. In the "info" folder, PDFs can be found with more information and resources about this project. The python code was executed through the VScode IDE, where instructions for the python environment setup can be found in the README. The arduino code was run through the arduino app. Once VScode was setup, the project's Github repository was cloned, which can be found below. The main code folders which will be used are the following: steady, unsteady and arduino

GITHUB REPOSITORY: https://github.com/DanielAsadi/FCET_SmokeFlow

4.2 Python Code Setup

Reference the Github README.

4.3 Arduino Code Setup

The Arduino codes located in the "arduino" folder is already uploaded to each board, however in case of an issue they can be reuploaed respectively to each board.

4.4 PFV4 Setup

To work with the Photron camera, the software PFV4 was installed. After referencing the manual found in the Github info folder to connect to the camera, the following settings were adjusted for triggering to be done by the SFV circuit:

- ➢ Live mode after recording
- ➢ Configuration > Trigger > Direct Trigger
- ➢ I/O > TRIG TTL IN > TRIG POS

5.0 USING SFV

5.1 Steady State

To use the SFV setup for steady state cases, the ctrl_man.py and ctrl_seq.py codes are used which are located in the “steady” folder. The ctrl_man.py code allows for manual actuation of each SFV component through the toggling of the keys which are described when running the code. This mode is mostly used for camera calibration work as the wire heating and liquid dispensing can be left on, creating a continuous state of smoke. The ctrl_seq.py code runs an automated SFV sequence actuating each element at timed intervals, which is used to obtain the videos.

5.2 Unsteady State

For unsteady state usage (with the active grid), the ctrl_ag.py, ctrl_ag_cont.py, and ctrl_ag_cont.py codes are used which are located in the “unsteady” folder. The ctrl_ag.py sends a short burst of smoke at the 0 degree encoder angle (start of the phase). The ctrl_ag_cont.py sends a longer lasting stream of smoke at the 0 degree encoder angle (start of the phase), however with the drawback of less uniformity in the streamlines.

NOTE: After each active grid case, it is ideal for the encoder to be zeroed. This is done by re-uploading encUno arduino program twice, once with line 82 uncommented and once with line 82 commented before starting active grid.

6.0 REFERENCES

[1] D. Asadi, Suraj Bansal, “FCET SFV,” Github Repository, UofT Inst. for Aerospace Studies, University of Toronto, 2021. https://github.com/DanielAsadi/FCET_SmokeFlow