

High-throughput droplet-based microfluidics

Mohammed Amine Elidrissi

Tuan Hao Tran

Jin De Guillaume

Omar Mansour

Umaid Azad Malik

Visweshwaran Balasubramanian

ABSTRACT

Our goal with this capstone project is to improve the performance and user interface of the current microfluidic droplet sorting system used in the lab. Improving the sorting system is done by allowing the high throughput of droplets to feed the sorting channel while maintaining high sorting purity. We developed an automated system that sorts the electrodes based on the droplets size by designing the controller, and GUI (General User Interface). The main function of the controller is to actuate the electrodes for specified duration set by the user. The GUI is there to offer the user a real-time image through the microscope camera with real-time spectrometer response, control over the syringe pumps, and send parameters to control the sorting settings. The design process described in this report is separated into 3 main categories, the micro controller, the GUI, and the sorting algorithm. We managed to build a successfully working prototype that meets most of the criteria that we set ourselves in the first phase of the project.

Key Words: Microfluidics, Bioengineering, Controller

Phase 4 Report: Final Design

High-throughput droplet-based microfluidics

Team #17

A report

Presented to

**The Department of Electrical and Computer Engineering
Concordia University**

In Fulfillment

Of the requirements

Of ELEC/COEN 490

By

Mohammed Amine Elidrissi	ID: 40045069
Tuan Hao Tran	ID: 40079129
Jin De Guillaume	ID: 40095507
Omar Mansour	ID: 27749821
Umaid Azad Malik	ID: 27576110
Visweshwaran Balasubramanian	ID: 40071701

Project Supervisors:

Dr. Steve Shih

Fatemeh Ahmadi

Concordia University

March 2022

Table of Content

0. INTRODUCTION	7
0.1. Objective	7
0.2. Project Outlines	7
1. DESIGN	9
1.1. Controller Design	9
1.1.1. Controller Design Specification	9
1.1.2. Microcontroller choices	9
1.2. Graphical User Interface Design	11
1.2.1. Graphical User Interface Design Specification	11
1.2.2. Graphical User Interface Framework Selection	11
1.2.3. Configuration Page	11
1.2.4. Workspace directory and Sorting Control	12
1.2.5. Redock Camera and Configuration Control	12
1.2.6. Microcontroller Configuration	13
1.2.7. Flame Spectrometer Configuration	13
1.2.8. Droplet and Electrode Configuration	14
1.2.9. NEMESYS Pump Configuration	14
1.2.10. Camera Configuration	15
1.2.11. Spectrometer View Page	16
1.2.12. Detection Range Configuration	17
1.2.13. Pump control	17
1.2.14. Chip Viewer Page	17
1.3. Usage of the MicroSort User Interface	19
1.3.1. Setting up electrodes and starting sorting	19
1.4. Development	20
1.4.1. Controller development	20
1.4.2. Graphical User Interface Development	23
1.4.3. Configuration Page	23
1.4.4. Spectrometer View Page	25
1.4.5. Chip Viewer Page	26
1.4.6. Thread Workers	27
1.4.7. Microcontroller Communication in Graphical User Interface Process	28

1.4.8.	Yaml File Reader	28
1.4.9.	Sorting Process and Flame Data Processing Class	28
1.5.	Problems Encountered	29
1.5.1.	Controller	29
1.5.2.	Graphical User Interface	29
1.5.3.	Detection Function Execution Time	29
1.5.4.	Real Time Plotting	30
1.5.5.	Python Global Interpreter Lock	30
1.5.6.	Spectrometer View and Camera View Unresponsiveness	31
1.6.	Solution Proposed	31
1.6.1.	Controller	31
1.6.2.	Graphical User Interface	31
1.6.3.	Detection Function Execution Time Solution	31
1.6.4.	Real Time Plotting Solution	32
1.6.5.	Python Global Interpreter Lock Solution	32
1.6.6.	Spectrometer View and Camera View Unresponsiveness Solution	33
1.7.	Future Development	33
1.7.1.	Optocoupler Board	33
1.7.2.	Chip Viewer	35
1.7.3.	Camera Viewer	35
2.	RESULTS OBTAINED	36
2.1.	Timing Analysis	36
2.2.	Test Cases	40
3.	ELSEE ASPECTS	44

List of Figures

Figure 1 Configuration Page	12
Figure 2 Workspace Directory and Sorting Control button group	12
Figure 3 Redock Camera and Configuration Control button group	12
Figure 4 Flame Spectrometer Configuration panel	13
Figure 5 Droplet and Electrode Configuration panel	14
Figure 6 NEMESYS Pump Configuration panel	15
Figure 7 Camera Configuration panel	16
Figure 8 Spectrometer View page	16
Figure 9 Detection Range Configuration panel	17
Figure 10 Pump Control panel	17
Figure 11	18
Figure 12	18
Figure 13	19
Figure sorting14 signal received from host at t=0s	21
Figure 15 Detection thread puts event in event Queue	21
Figure 16 Sorting thread to turn on electrode	22
Figure 17 Sorting thread to turn off electrode	22
Figure 18 caption	23
Figure 19 caption	25
Figure 20 caption	26
Figure 21 caption	27
Figure 22 caption	30
Figure 23 caption	32
Figure 24 caption	33
Figure 25 caption	34
Figure 26 caption	34
Figure A27	36
Figure B28	37
Figure C29	37
Figure D30	38
Figure E31	38
Figure F32	39
Figure 33 High throughput 1600 droplets	39
Figure 34 Main Interface	47
Figure 35 Left Pane	48
Figure Full GUI	48
Figure Microscopic view of feeder, sorting and waste channels, along with the electrodes	49
Figure Configuration UI	50
Figure Spectrometer View UI	51
Figure 40 Sorting Control	52
Figure 41 Chip Viewer UI	52

List of Tables

Table 1 Controller Specification	9
Table 2 FPGA vs Microcontroller	10
Table 3 Graphical User Interface Specification	11
Table 4 Test Cases	41

1. INTRODUCTION

This report encloses the inspiration, design, and implementation of an automated microfluidic binary droplet sorter.

1.1. Objective

The goal of this project is to design and create a control board that can be connected to a microfluidic system to sort a stream of droplets based on a binary channel and a detection signal. The system would provide high efficiency with rapid droplet sorting actuation. The hardware will be supported by a user interface that allows the user to configure the sorting parameters and monitor various aspects of the operations. The designed system would be modular and interface with existing automation infrastructure.

1.2. Project Outlines

The project was planned to take place in four phases, efficiently splitting the work into bursts of tasks undertaken over two months each.

First, a controller platform was chosen from the requirements elicited during Phase 1. Once the controller was chosen, the design specification was updated with data from the controller's datasheets.

System implementation alternatives were evaluated in Phase 1. While choosing controllers in Phase 2, controller platforms alternatives were also being shortlisted for comparison. Controller platforms alternatives evaluation was split into two types, FPGA vs. microcontroller.

Detailed quantitative analysis and discussions of FPGA and micro-controller were carried out to determine that the micro-controller was the better. Then, various microcontrollers that fit our specifications were listed. Online benchmarks were then used to evaluate the better controller model.

The team then moved on to research and plan out the implementation by choosing communication protocols, a real-time operating system to sort droplets in real-time, GUI language, etc.

GUI design evaluation was next, with modifications being proposed to existing Droplet Sorter GUI. Wireframes of suggested GUI layout were brainstormed, and a final UI design was planned.

Phase 3 was planned for the design and development of the prototype.

Controller development was carried out on the teensy USB-based development board. The code is written in C++/Arduino. The sorting algorithm was developed as a Real Time Operating system. An RTOS system is required in this project to handle incoming detection signals and process them in a timely manner. The control algorithm was first implemented using the port of FreeRTOS for the Teensy 4.1 and then later, development was carried on

another RTOS platform called ChibiOS. This switch in the RTOS platform was due to some issues we encountered with FreeRTOS that will be discussed later in this report.

GUI implementation was developed in a Microsoft Windows environment using Python. The decision to continue development on Windows was to meet the stakeholder's requirement. PyQt5 is the framework we used to design the GUI. Further details are discussed in later sections of this paper.

Lastly, the testing and analysis of the system. Before individual modules were put together, the GUI was tested for itself, and tests were conducted to make sure the GUI was bug free. Once passed, the GUI was merged with different hardware components to test the overall system using an oscilloscope.

2. DESIGN

2.1. Controller Design

2.1.1. Controller Design Specification

The control board we decided to use for this project is the Teensy 4.1 microcontroller. It can operate under an input voltage of 3.3V to 5V through its VIN pin or 5V USB cable. The board features 55 digital I/O pins and a clock speed of 600 MHz and can be overclocked of up to 1GHz. The maximum current consumption for the board at max speed is at around 100mA. The controller will also be very portable due to the minimal dimension of only 36.8mm in length, 18.0mm in width and 4.6mm in height. At minimum droplet actuation our system will be able to sort at least 1000 droplets per second while the typical value should be in the range of 1200 droplets and a high of 1500 droplets. As most of the existing solutions have high sort purity percentage in the range of 95% or above, we plan to design our system within that range so it can produce a meaningful output for the users. The system we design should be able to control at least 110 electrodes while the typical number of electrodes should be 150 and a high of 200 electrodes.

Table 1 Controller Specification

#	Description/Parameter	Test Condition	Value			Unit
			Min.	Typ.	Max.	
1.	Droplet Actuation	Normal	1000	1200	1500	droplets/second
2.	Sort Purity	Normal	95	97	99	%
3.	Electrode Communication Path	Normal	110	150	200	paths
4.	Dimensions (L/W/H)			36.8 / 18.0 / 4.6		mm
5.	Input Voltage (DC): 1. USB Cable Power: 2. VIN pin	Normal	3.3	5	5	V V
6.	Current Consumption at 600MHz			100		mA
7.	Digital Input/Output			55		pins
8.	I/O Pins: 1. Voltage Output 2. Current Output 3. Voltage Input		0	3.3 10	3.3	V mA V
9.	Clock Speed	Normal	24	600	1000	MHz
10.	Operational Temperature		0		80	°C

11	Storage Temperature		-30		70	°C
12	Relative Humidity	Non-condensing			80	%

2.1.2. Microcontroller choices

Table 2 FPGA vs Microcontroller

Criteria	Importance Weight	FPGA Board		Microcontroller	
	0-1	Score (0-1)	Score*W	Score (0-1)	Score*W
Cost	0.75	0.25	0.1875	1	0.75
Portability	0.75	0.5	0.375	1	0.75
Processing Speed	1	1	1	0.8	0.8
Development Simplicity	0.75	0.25	0.1875	1	0.75
Power consumption	0.25	0.25	0.0625	1	0.25
Total:			1.8125		3.3

The decision matrix table above shows the difference in scoring between the two control board choices, microcontroller, and FPGA board. The low scoring of FPGA boards in terms of cost is due to its price being much more expensive than that of microcontroller. Since portability is one of the main criteria for the project, the scoring of the microcontroller is much higher than the FPGA board due to its small form factor in terms of size and weight. One could argue that FPGA should have higher portability scoring since it has hundreds of I/O pins that can replace the current optocoupler switching board giving it more advantages compared to microcontroller. However, the FPGA board still needs the switching board to switch on/off high voltage, and it is only replacing the existing port expander on the switching board which has minimal footprint on the PCB design, making it a lower scoring compared to the microcontroller. In terms of complexity, microcontrollers will have a higher scoring due to its simplicity in terms of the development languages and the number of resources available online compared to FPGA where one would need advanced knowledge in designing digital circuits, VHDL and signal processing techniques in low level systems. Other factors such as processing speed is also an important factor to consider, under those circumstances, FPGA scored higher compared to the microcontroller due to its extremely

fast processing speed through parallelization and versatile configurations. However, the same parallelization processing can be achieved using a PC with multithreading and multiprocessing and use the microcontroller as a switch since one of the requirements for the project is to have a graphical user interface interacting with the sorting system. As a result, with the processing work being offloaded on the PC side, the overall processing could potentially be as fast as an FPGA board. For the last factor, power consumption, microcontrollers usually consume much less power than FPGA due to its lower processing power. From the above scores, the microcontroller is clearly a better selection than the FPGA for the main reasons the microcontrollers usually cost much less than FPGA, higher portability, almost identical processing speed as well as the knowledge required around microcontrollers such as Arduino and C/C++ are much more commonly used and much simpler than the hardware description languages such as VHDL and Verilog used for FPGA board.

2.2. Graphical User Interface Design

2.2.1. Graphical User Interface Design Specification

Trying to offer maximum control over the sorting system to the user, the following specifications were identified. Additionally, the team strived to give the user the ability to view spectrometer output and microscope camera output through the interface itself.

Table 3 Graphical User Interface Specification

1	User should be able to make connection with the microfluidic sorting controller
2	User should be able to see all the interactable electrodes through the interface
3	User should be able to assign functionality to each/group of electrodes
4	User should be able to set the trigger threshold

Due to the considerable number of external variables in effect during sorting procedure like droplet size, speed and amount of fluorescence in the droplet to name a few, it would have been exceedingly difficult to implement an exhaustive mapping that sets sorting parameters based on droplet parameters. Hence, tuning of certain sorting parameters will have to be done by the user.

The following few sections will highlight the features of the final UI design and its usage.

2.2.2. Graphical User Interface Framework Selection

In phase 2, the design phase, we decided to go with developing our graphical user interface using Tkinter framework which is the framework that previous graphical user interface was developed on. However, due the abstract and cumbersome user interface design for complex graphical user interface, and a lack of designing tools, we decided to go with PyQt5 framework due to the complex designing tools that the framework provides, Qt Designer Tool, and lots of learning resources available online.

2.2.3. Configuration Page

In the configuration page, the user would be able to connect to the controller and change various parameters corresponding to the bench set up such as Flame Spectrometer integration time, droplet travel time in the fluid channel and the duration that the sorting electrode stays high.

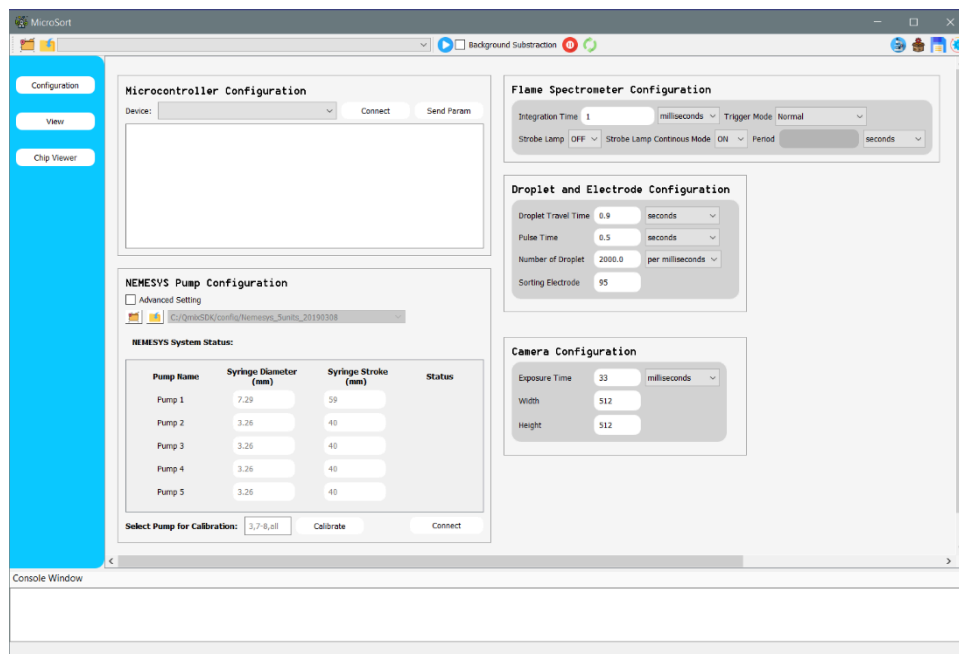


Figure 1 Configuration Page

Shown in **Figure 1** is the **Configuration window** with text fields and Buttons organized into panels. The menu bar is visible throughout the interface and contains two button groups.

2.2.4. Workspace directory and Sorting Control

The button group on the left of this group, shown in Fig. X can be used set the workspace directory. The blue 'Play' button starts sorting, while the red 'Pause' button stops sorting. Before changing parameters, it is advised to stop the sorting process and start it again after sending set parameters to the controller.

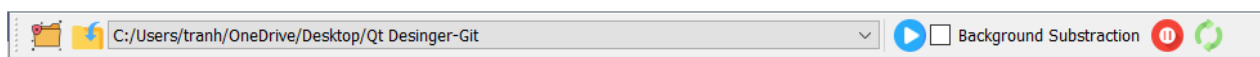


Figure 2 Workspace Directory and Sorting Control button group

The 'Background Subtraction' checkBox relates to the Spectrometer view and removes noise from the detection signal. This operation makes the detection signal from droplets to-be sorted, more apparent when there is luminescence from it.

2.2.5. Redock Camera and Configuration Control

This button group contains four buttons. The microscope camera view is rendered in a widget that can be dragged around, resized and docked to the user interface. The first button with the microscope icon re-docks this camera viewer widget to the interface.



Figure 3 Redock Camera and Configuration Control button group

The second button with the 'box and gear' icon opens a file explorer window and allows the user to load an existing configuration to avoid having to set all the parameters again. The 'Save As' button next to it saves the current configuration to a directory of choice using the file explorer.

The last button on the right gives the user an option to set the current configuration as default and loads said configuration whenever the application is launched.

Next, **the purpose and usage of the various panels for setting parameters are discussed below.**

2.2.6. Microcontroller Configuration

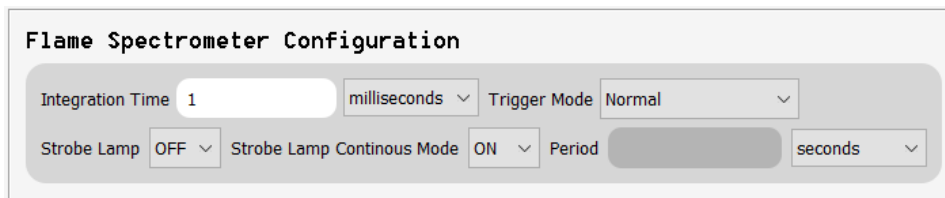
This panel allows the user to select the controller device from a list of external devices connected to the workstation. One can then connect to the microcontroller using the 'Connect' button.

The 'Send Param' button sends the current parameter values in the Droplet and Electrode Configuration panel to the controller. This operation is mandatory after setting new parameters as they will be used by the microcontroller to determine electrode actuation delay.

The textField below these buttons shows connection status, and current parameters when they are sent to the controller.

2.2.7. Flame Spectrometer Configuration

The parameters for the flame spectrometer are set from this panel. As can be seen below in Fig. X, there are a lot of parameters that affect the ability to sort droplets successfully.



Flame Spectrometer Configuration

Integration Time: 1 milliseconds Trigger Mode: Normal

Strobe Lamp: OFF Strobe Lamp Continuous Mode: ON Period: seconds

Figure 4 Flame Spectrometer Configuration panel

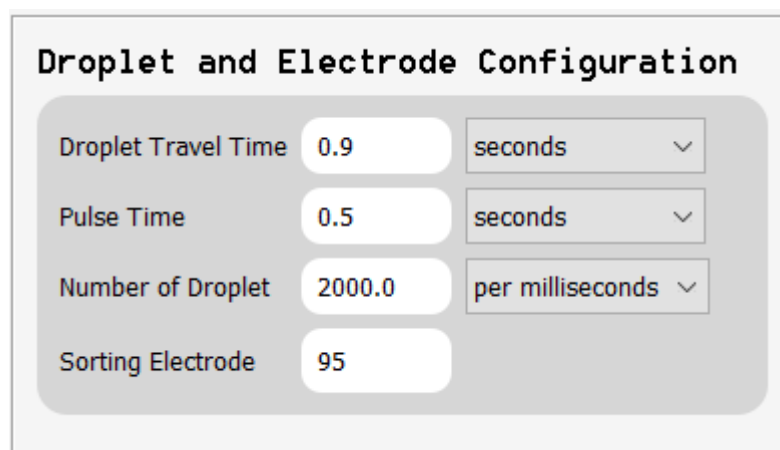
The 'Integration Time' parameter influences the intensity of the luminescence detected. Droplets with a lot of fluorescence would require less integration time as the light emitted is strong enough to produce a detection signal. The speed of the droplets was also found to be proportional to the integration time.

The flame spectrometer also comes with a strobe lamp that can be controlled from the UI itself. The time period of pulses can be set using the 'Period' textfield. The strobe lamp can also be used in 'Continuous Mode' without pulses using the 'Strobe Lamp Continuous Mode.'

For advanced control over data acquisition, the 'Trigger Mode' list allows the user to set data acquisition to only occur after an external event triggered by hardware voltage or synchronization events like button pushes. The trigger mode is by default set to 'NORMAL' where data acquisition is done continuously.

2.2.8. Droplet and Electrode Configuration

The microchannel and electrode activation parameters can be set from this panel. As shown below, there are four values that can be set.



Droplet and Electrode Configuration

Droplet Travel Time: 0.9 seconds

Pulse Time: 0.5 seconds

Number of Droplet: 2000.0 per milliseconds

Sorting Electrode: 95

Figure 5 Droplet and Electrode Configuration panel

The time that the droplet takes from the moment it passes the spectrometer probe till it reaches the sorting electrodes, defines the 'Droplet Travel Time'. This parameter affects when the sorting electrode is activated and hence must be accurately set to achieve highest efficiency of sorting.

The 'Pulse Time' sets how long the sorting electrode remains at high voltage. The pulse time was found to be proportional to the droplet size from trials. The voltage being sent to the electrode from the amplifier also influences the pulse time inversely.

The number of droplets that are generated by the microfluid channel would be set as the 'Number of Droplet' parameter. The parameter will be used to limit the number of electrode actuations in that timeframe.

The electrode number corresponding to the pin that the electrode connects to on the port expander, would be set as the 'Sorting Electrode.'



As supplementary features that allow for control over the full microfluid system, there are two extra panels for managing the NEMESYS pumps and the microscope camera.

2.2.9. NEMESYS Pump Configuration

The NEMESYS Pump system has 5 pumps which can each be configured using this panel. As changing the pump configuration can lead to adverse effects on the microfluid channel, editing these parameters is disabled by default. The 'Advanced Setting' checkBox must be selected to do so.

NEMESYS Pump Configuration

☐ Advanced Setting

  C:/QmixSDK/config/Nemesys_5units_20190308

NEMESYS System Status:

Pump Name	Syringe Diameter (mm)	Syringe Stroke (mm)	Status
Pump 1	7.29	59	
Pump 2	3.26	40	
Pump 3	3.26	40	
Pump 4	3.26	40	
Pump 5	3.26	40	

Select Pump for Calibration: 3,7-8,all Calibrate Connect

Figure 6 NEMESYS Pump Configuration panel

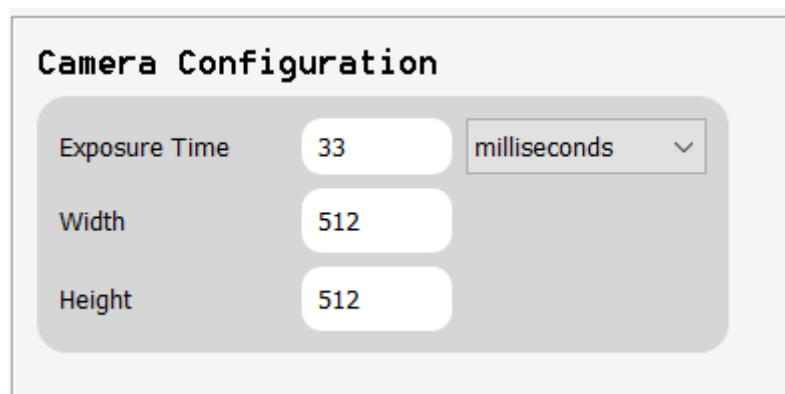
Initially, the user must connect to the pump system using the 'Connect' button. The user can also load prior configurations using the file browser provided.

The 'NEMESYS System Status' window displays the status of each pump. The user can change the syringe's swept volume using the 'Syringe Diameter' and 'Syringe Stroke' textFields.

The new settings are then applied for selected pumps using the 'Calibrate' button.

2.2.10. Camera Configuration

Lastly, through the Camera Configuration panel, the user can adjust the camera viewer window size using the 'Width' and 'Height' parameters in pixels. The user has control over the exposure time of the camera. Varying this parameter can give better image resolution and quality.



The image shows a 'Camera Configuration' panel with a light gray background. It contains three rows of controls: 'Exposure Time' with a text input field containing '33' and a dropdown menu set to 'milliseconds'; 'Width' with a text input field containing '512'; and 'Height' with a text input field containing '512'. The panel has a title bar at the top and a subtle shadow.

Figure 7 Camera Configuration panel

2.2.11. Spectrometer View Page

Clicking the 'View' button switches pages to the Spectrometer View Page, allowing the user to view the detection signal being output by the Flame spectrometer. The user must initially select the spectrometer to be used from a list of available devices. The user also configures the detection range that would cause the sorting electrode to activate.

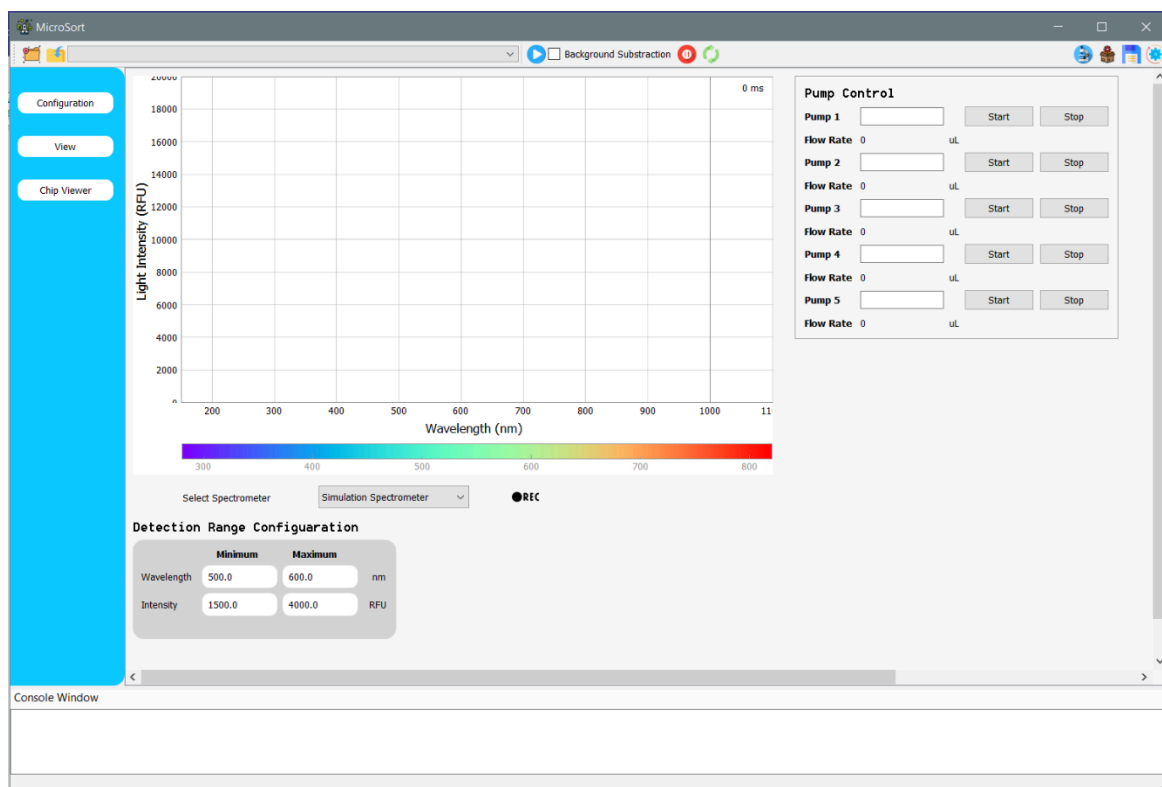


Figure 8 Spectrometer View page

Whenever sorting is started using the blue 'Play' button, the graph view will start showing spectrometer output as a function of light intensity in RFU (Relative Fluorescence Units) and the wavelength of light in nanometers.

There are two panels in this page which are discussed below.

2.2.12. Detection Range Configuration

The window of detection which triggers a droplet sorting event can be set from this panel. The minimum and maximum ranges of wavelength and intensity for sorting form a rectangular detection window.

Detection Range Configuration

	Minimum	Maximum	
Wavelength	500.0	600.0	nm
Intensity	1500.0	4000.0	RFU

Figure 9 Detection Range Configuration panel

2.2.13. Pump control

To avoid crowding the Configuration Page, pump flow control is provided on the Spectrometer view page. The user can set the flow rate of the desired pump in uL and start or stop the pump using the appropriate buttons provided next to each pump.

The Pump Control panel is a light gray rectangular box with the title "Pump Control" at the top left. It contains five identical rows, each representing a pump. Each row has a "Pump" label followed by a text input field. To the right of the input field are two buttons labeled "Start" and "Stop". Below each input field, the text "Flow Rate" is followed by the value "0" and the unit "uL".

Pump	Flow Rate	Start	Stop
Pump 1	0 uL	Start	Stop
Pump 2	0 uL	Start	Stop
Pump 3	0 uL	Start	Stop
Pump 4	0 uL	Start	Stop
Pump 5	0 uL	Start	Stop

Figure 10 Pump Control panel

2.2.14. Chip Viewer Page

The Chip Viewer page, shown in Fig. 11 allows users to manually control electrodes. Three buttons on the right allow users to clear all the electrodes on the scene, load an existing electrode configuration or save the current electrodes on the scene as a .yaml configuration file.

Furthermore, there are two textFields below the buttons. The electrode number corresponding to the sorting electrode is entered in the first textField. The second textField takes in the pulse time in uS; the duration for which the electrode is active. Once entered, the user simply clicks on 'Electrode,' which renders an electrode object and drags it into to scene on the left.

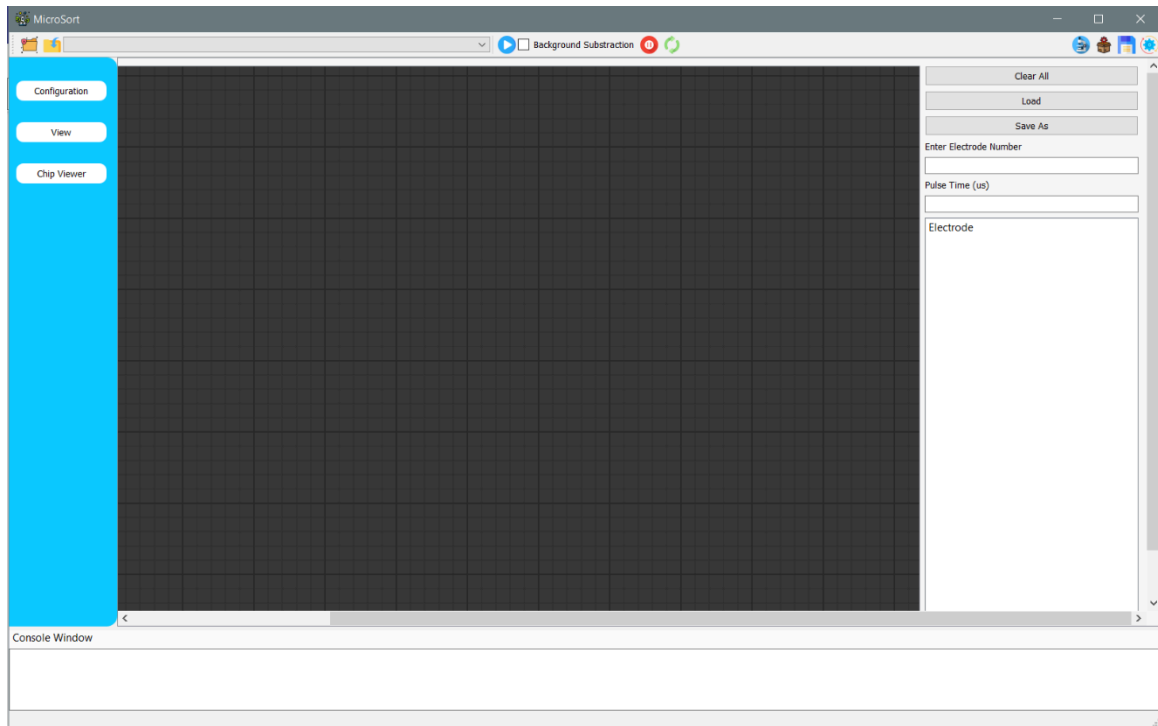


Figure 11

With the electrode on the scene, the user can click on the electrode object which will trigger corresponding electrode activation. This feature is intended to be used for testing and tuning of the sorting parameters.

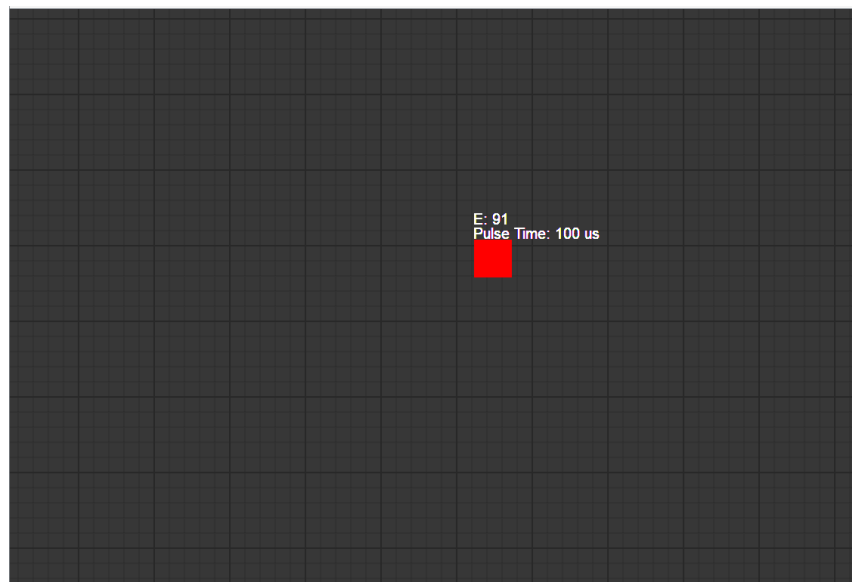


Figure 12

2.3. Usage of the MicroSort User Interface

2.3.1. Setting up electrodes and starting sorting

The general use case for a user setting up the system for sorting is shown below in Fig. 13

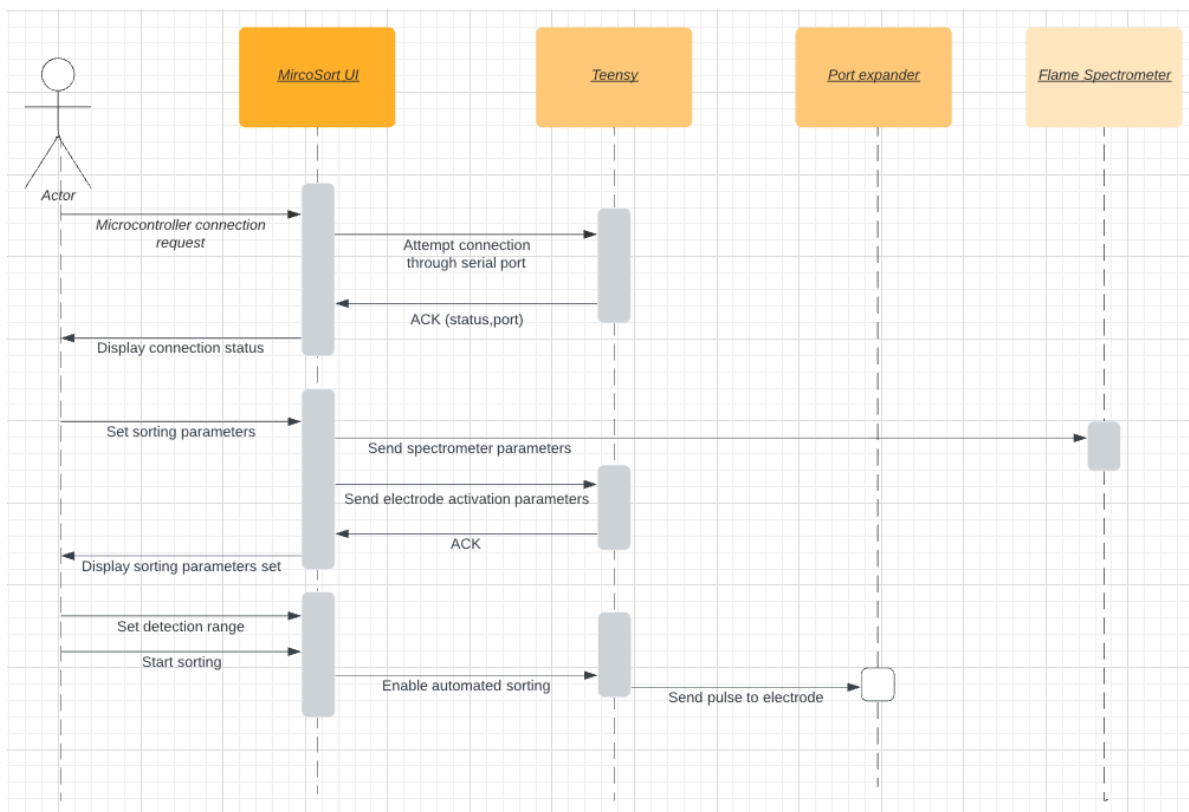


Figure 13

When the MicroSort UI is initialized, the user must first establish a connection with the sorting controller using the Microcontroller Configuration panel in the Configuration page. When connection is attempted, a message containing the connection status and port are displayed in the panel.

The user can then set various sorting parameters in accordance with their setup and then send the new values to the controller using the 'Send Param' button in the same page. Confirmation of the parameters being downloaded on the controller can be seen on the textfield below the 'Send Param' button.

The user must finally switch to the Spectrometer View page to input the detection range for which droplets are sorted. Setting the range completes the setup and automated sorting can be started by pressing the blue 'Play' button on the menu bar.

For the spectrometer view to work as intended, care should be taken to ensure that the spectrometer device is recognized by the user interface and is selected in the list below the detection signal graph view.

2.4. Development

This section will only discuss the main functionality and structure of several functions in the system. Detailed implementation can be found in the project source code.

2.4.1. Controller development

The controller in our project is responsible for sorting the droplet once a detection signal is received from the host computer. The two main functions of the controller are to listen for droplet sorting events and once an event is received the controller should be able to sort the droplet by actuating the electrode at the right time to sort only the droplet of interest out of the stream of droplets. The controller has also an auxiliary function that permits the user to apply sorting parameters to the controller through the GUI application. The parameters are also sent through serial, the controller parses the sent parameters and immediately applies them with minimal interruption to the sorting process.

To be able to continuously listen for droplet detection messages as well as sorting (turning on/off the electrodes in the right sequence at the right time), we must be able to run both functions in a non-blocking fashion, this means that both functions should continuously run. In a multicore system, each core would be responsible of running the detection thread and the sorting separately in parallel. In our case the best option is to use a Real Time Operating System with a cooperative schedule to run both functions in separate threads achieving near parallel processing.

The RTOS algorithm consists mainly of 3 main components. 2 threads, one thread to monitor the Serial bus and listens for any event that indicates the detection of a droplet, and a second thread responsible of sorting droplets once the thread is notified that a droplet is available for sorting. this thread will then actuate the electrode accordingly in a timely manner. The third main component is a Queue that serves as a link between the two threads. Once a droplet serial event is received, the detection threads notify the sorting thread of the availability of a droplet to be sorted through the queue. The detection thread enqueues a timestamped event in the queue signifying the presence of a droplet to be sorted. The sorting thread continuously monitors the presence of elements in the queue, once an element in the queue is available, the sorting thread dequeues the element and using the delay and electrode number parameters set by the user (travel delay, pulse delay, sorting electrode) as well as the timestamp of when the event was received, the thread will accordingly turn on or off the electrode.

Here is an example run on how the algorithm handles the sorting of droplets: for this example, the travel delay is chosen to be 3 seconds while the onTime (pulse) delay is chosen to be 2 sec. this translate to once a droplet is detected wait 3 seconds until the droplets arrives near the sorting channel and turn on the sorting electrode for 2 seconds.

The detection signal is sent through serial from the GUI application at $t=0s$, the detection thread is continuously monitoring the serial inputs and detect the serial event.

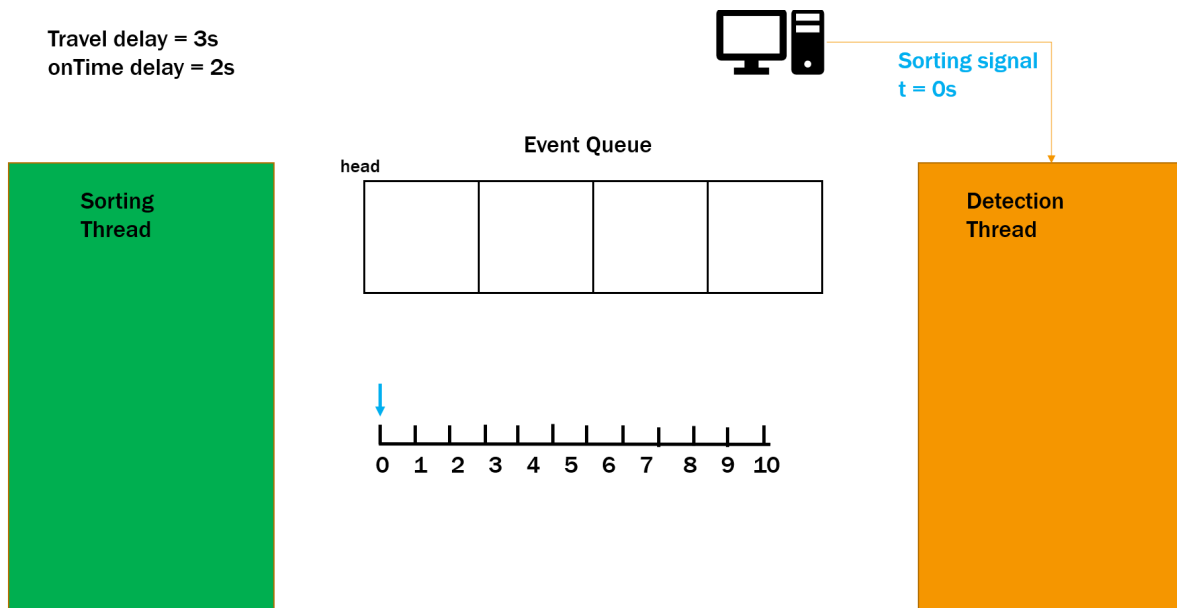


Figure sorting14 signal received from host at $t=0s$

The detection thread will create then two events to be put in the event queue, one ON event and one OFF event (the order in which these two event are enqueued is important, the ON event is first enqueued followed by the OFF event, this is because you first turn on the sorting electrode then turn it off and the queue is FIFO (first in first out) data structure), both event will be timestamped with the time at which the event was received from the host, in this case the timestamp is 0s.

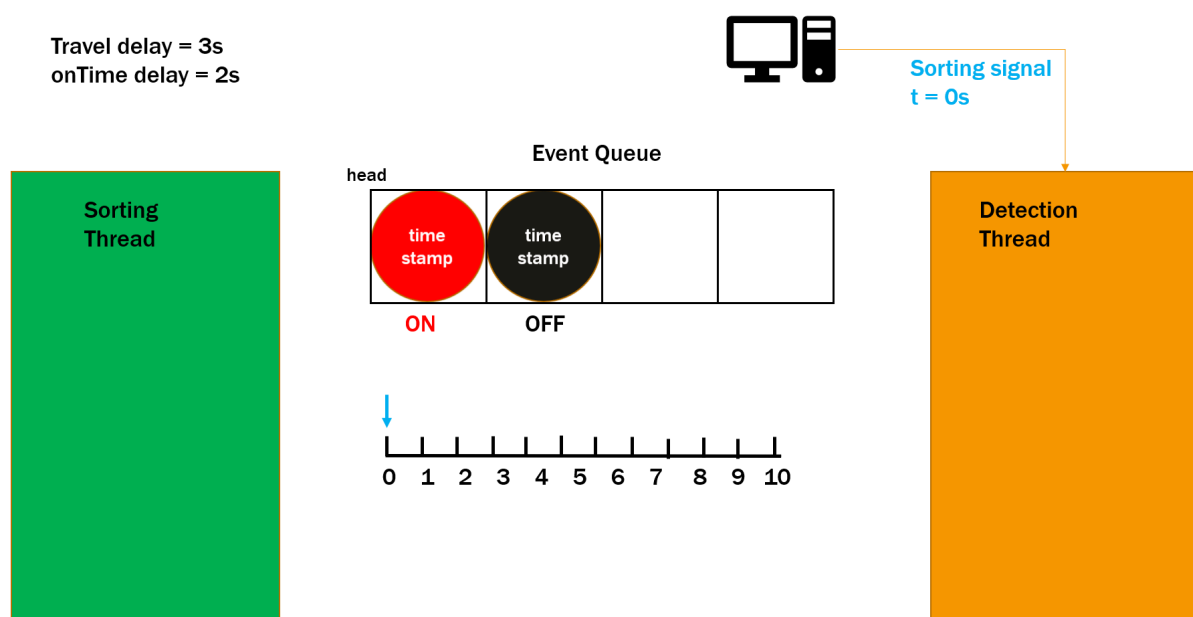


Figure 15 Detection thread puts event in event Queue

The sorting thread is continuously monitoring the event queue, so once event elements are in the queue, the sorting thread will dequeue the first available event. Since this is an ON event, the sorting thread will turn on the sorting electrode when the current time is equal to the timestamp time + the travel delay. The timestamp time represents the time at which the droplet was received.

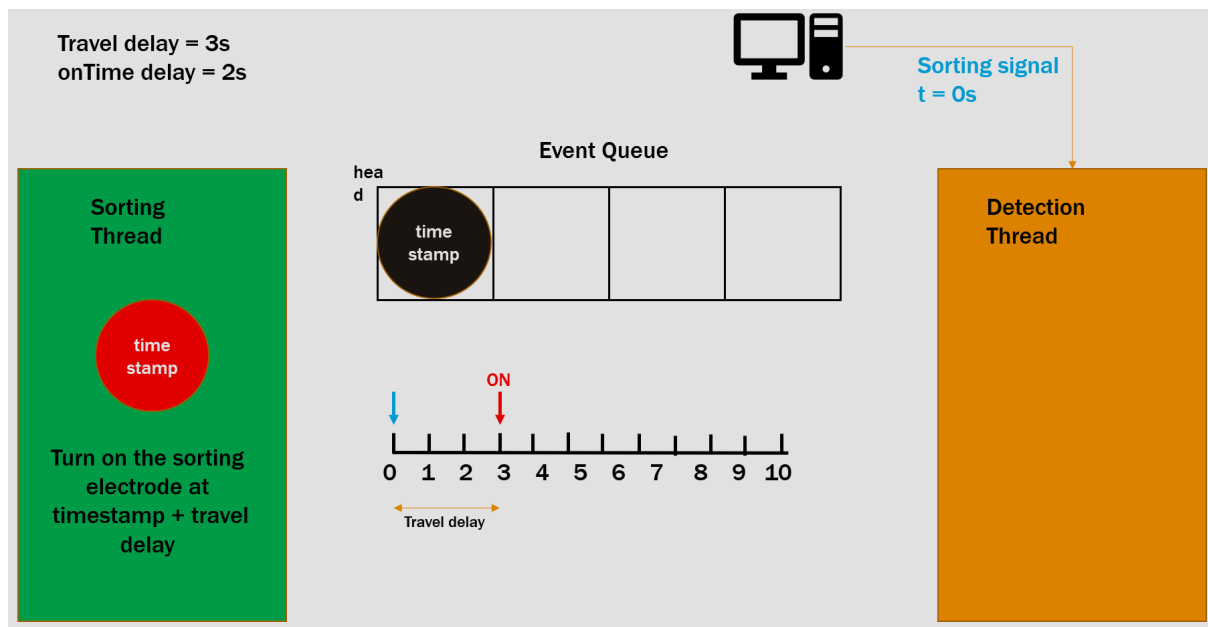


Figure 16 Sorting thread to turn on electrode

The sorting thread will then go ahead and dequeue the next available event in the queue which is the OFF event with the same timestamp ($t=0s$). In this case the sorting thread will turn off the electrode when time is equal to the timestamp + the travel delay + onTime delay, since the travel delay was already elapsed the electrode will be turned off after the onTime delay is elapsed, so the electrode was turned on after 3 seconds and stayed on for 3 seconds then turned off again.

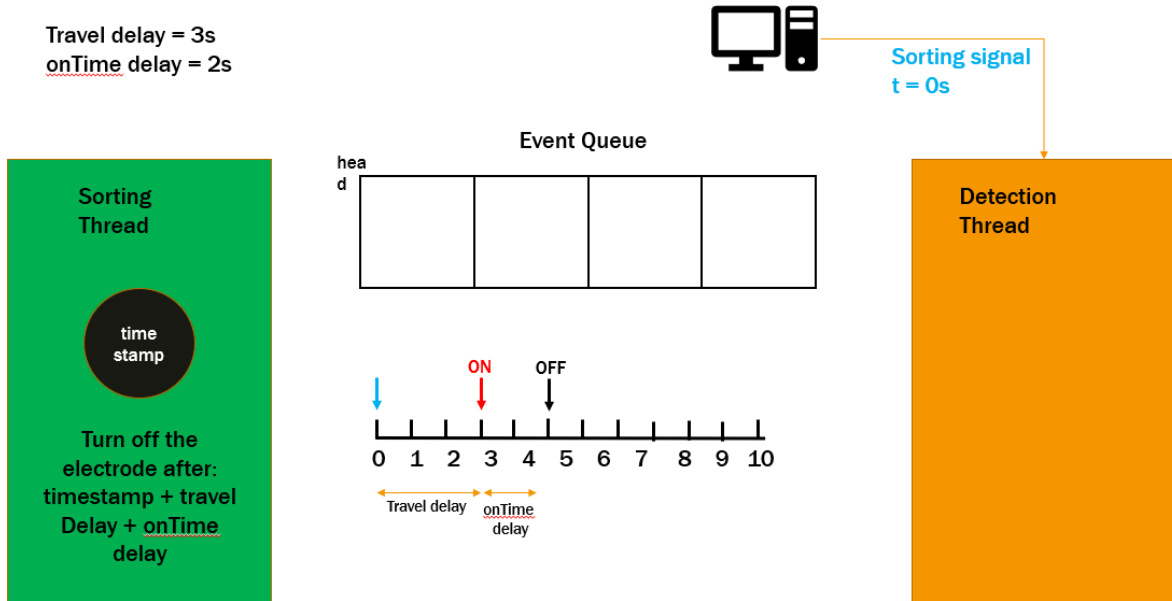


Figure 17 Sorting thread to turn off electrode

This algorithm is implemented using the Teensy 4.1 port of ChibiOS an RTOS platform very optimized for teensy products allowing to run the schedule in tickless mode and obtain time resolution down to the microsecond which is necessary when sorting very high throughput droplet streams.

2.4.2. Graphical User Interface Development

The Graphical User Interface was developed on the Windows operating system using the PyQt5 framework. Unlike the Tkinter which was first used in the project PyQt allows us to organize the UI scripts separately from the functionality. Because one of our goals was to make the software portable, we decided PyQt as our main framework. GUI as following. The figure below shows that an instance of `Ui_MainWindow` is instantiated in the APP classes and passed to each page along with the instance of the microcontroller. This coding style allows us to link the GUI elements to the functionality without polluting the class of each page, hence better maintained. As it can be seen the APP contains 3 pages namely `ConfigurationPage` where the user can set parameters for the system, `ViewPage` where the real-time data from the spectrometer is displayed on the chart and Image from the microscope, lastly `ChipViewPage` provides manual control of the electrodes to the user.

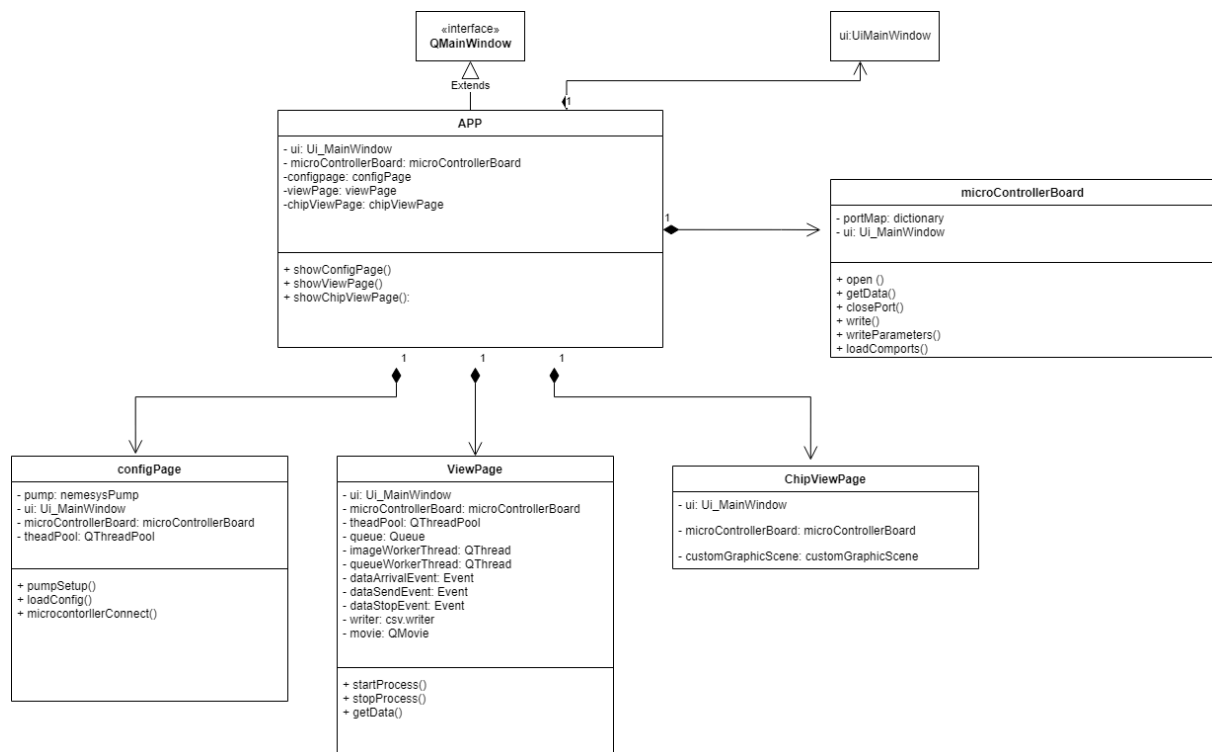


Figure 18 UI Architecture

2.4.3. Configuration Page

For the configuration page user interface elements, there are 5 configuration boxes namely are microcontroller configuration, flame spectrometer configuration, pump configuration, droplet configuration and camera configuration. The configuration page implementation was differed from our previous planned design in Phase 1 and Phase 2 due to that more features were added to Graphical User Interface. In the configuration page, the user will be able to establish communication with the microcontroller, the pump, and be able to set up/ tune all the parameters for the sorting system. The detail design of the configuration page is in the configPage.py file where all the user interface elements are linked to a specific function, as for the placement of the user interface elements, it is being stored in the GUI.py file where we generated from Qt Designer app using uiGenerate.py file which convert GUI.ui file to GUI.py file.

The detail implementations of the configuration page are stored in configPage class where in the first initialization of the object, it will take the user interface element object from the main window, the microcontroller object so it can establish communication with the microcontroller, and it also take the user interface elements initialized in the view page class for tool bar user interface elements access. Next, it will initialize the pump and thread pool object so it can be used later for pump functions manipulation. The class will then load the configuration file which is named config.yml stored in the default file path using the custom function called yamlFileReader where it will be able to set all the previously saved user input parameters. The class will also limit user to only be able to input number only in the

configuration parameter boxes using PyQt5 built in number validator function. The config page class also responsible for set up part of the user interface elements for the pump configuration boxes as the number of pumps will be dynamically loaded on startup based on the available pumps connected to the system. The microcontroller connect button clicked action will automatically connect to the microcontroller board using the microcontroller object previous passed. It will then send the parameters to the microcontroller namely are the droplet travel time, pulse time and the sorting electrode to the microcontroller. With the clicked of the send param button, it will send the current configuration parameters to the microcontroller. As for when the pump connect button is clicked, it will send all the user inputs parameters and the configuration file of the NEMESYS pumps to set up the connection with the all the pumps and will show the status of the bus if the connection is successful or failed.

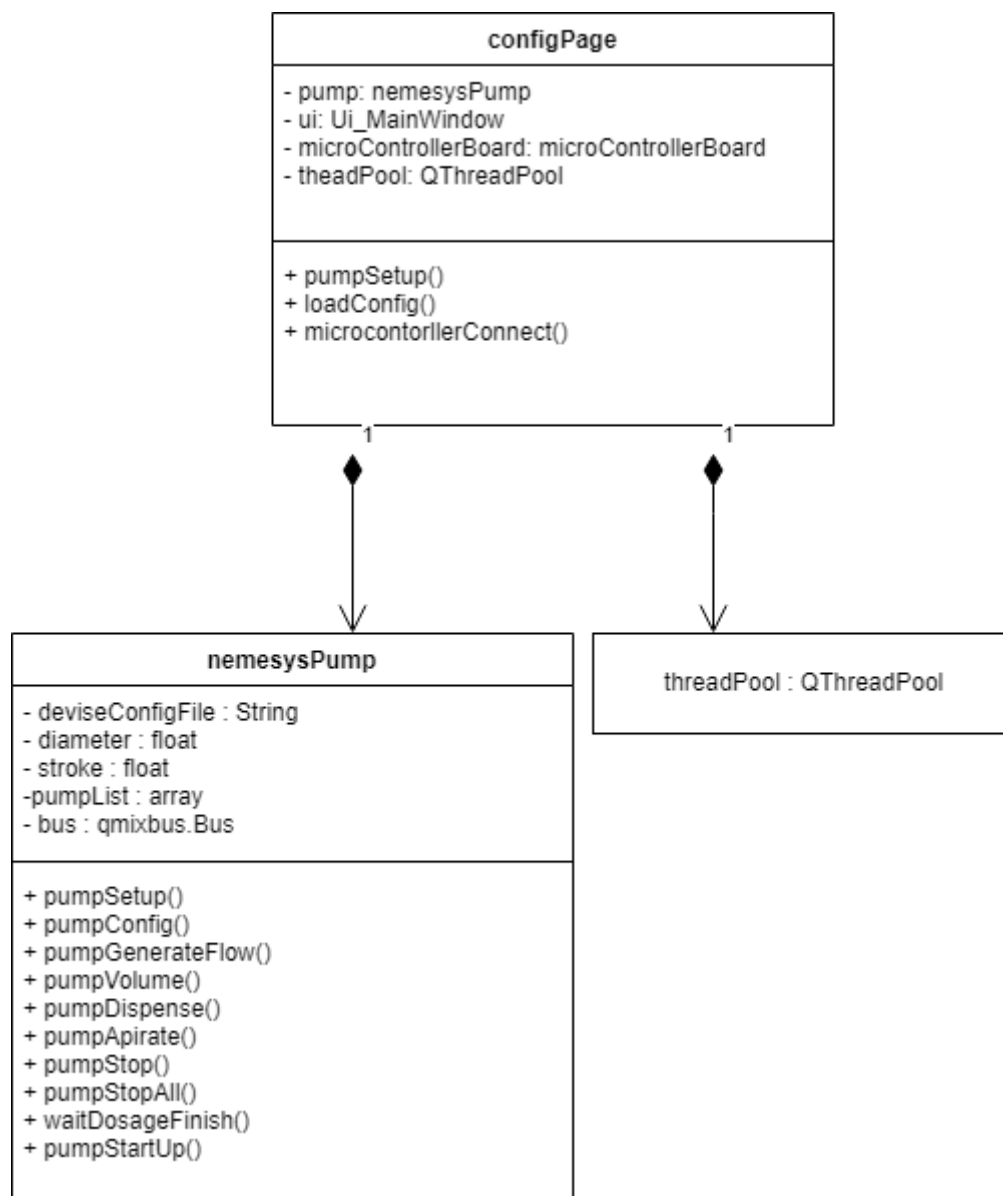


Figure 19 caption

2.4.4. Spectrometer View Page

With the spectrometer view page, there are three main elements in the page, first will be the real time spectrometer view where it will be used to display the data from the spectrometer, upper right corner of the spectrometer view will be the update rate of the plot. The second element in the spectrometer view page is the detection range configuration where user will be input the desired gates for detection. Last element in the spectrometer view will be the pump control where user will be able to control the 5 pumps available in the current system where the actions are start and stop and flow rate input and display. Similar to the config page user interface elements, the spectrometer view page user interface elements placement is stored in the GUI.py file. The detail design of the spectrometer view page is stored in the viewPage.py file.

At first, the view page class will take the user interface elements from the main window and the microcontroller board object. After that, the class will set up the tool bar user interface elements namely are the add, delete folder path button, folder path display, start sorting, stop sorting, connection list reloading, camera view startup, load configuration file, save current configuration as and save current configuration to the default config file. The set-up tool bar function will also link all the action to the corresponding action. Next, the class will set up a number validator for user input to the detection gates input and load the spectrometer list to the spectrometer option box. The class will also initialize a queue worker thread, required multiprocessing events to communicate with sorting process. When the start button is clicked on the toolbar, it will connect to start sorting function, at first, the function will check if the child process exists, if not it will check all the user input parameters making sure that it is not empty. After that, it will create a child process where the queue and previously created events will be passed to the child process for communication between the GUI process and the sorting process. Finally, the function will create a separate thread in the GUI to actively listening to data sent from other process so it can be plot in the spectrometer view. When the stop button is clicked, it will trigger a stop event to other process to stop the sorting and sending data to plot, if the other process did not stop after 1 second, the parent process will terminate it and set the process variable back to none. Another main function in the class is the camera view function where when the camera view button is clicked it will create a new subprocess and send all the required parameters for camera set up such as the window width, height, and camera exposure time. In the subprocess file, the startMonitor.py file, it will create the Hamamatsu camera instance using the received parameters and display the frame on a new window using OpenCV API.

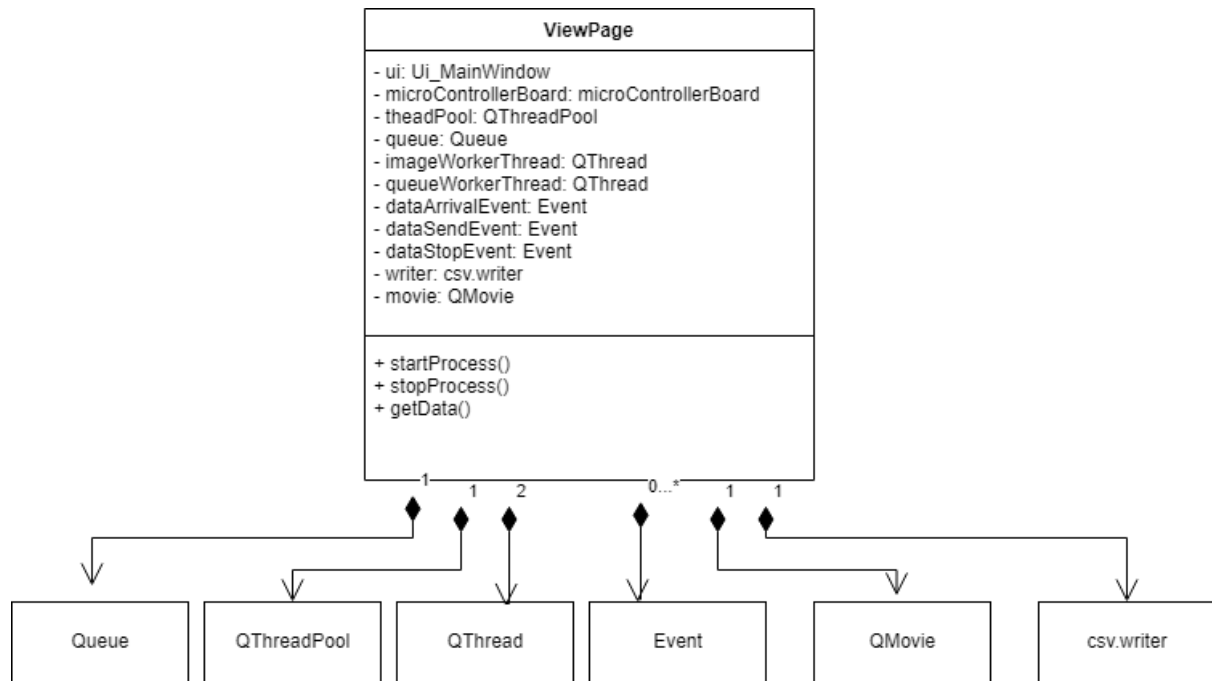


Figure 20 caption

2.4.5. Chip Viewer Page

The ChipViewPage used two custom classes namely CustomGraphicButton and CustomGraphicScene. CustomGraphicButton is extended from QGraphicsItems. Two methods are overwritten to add the custom action when the user click its instance. Backend API of the PyQt5 triggers an event based on the user's mouse action. The method mousePressedEvent is invoked and signal is sent to the controller to actuate the selected electrode for the preset time. CustomGraphicScene is extended from QGraphicsScene. Similar to the CustomGraphicButton class, some of the methods are overwrote to add custom actions. The method dropEvent is invoked when the user drags and drop a new electrode from the right side of the scene, it creates the instance of the CustomGraphicButton object. The dragMoveEvent method accept the drags and drop event of the previously created CustomButton object. Lastly ChipViewPage has few methods to help user save the current configuration of the electrodes in YAML file and load the electrodes from a previously saved YAML file.

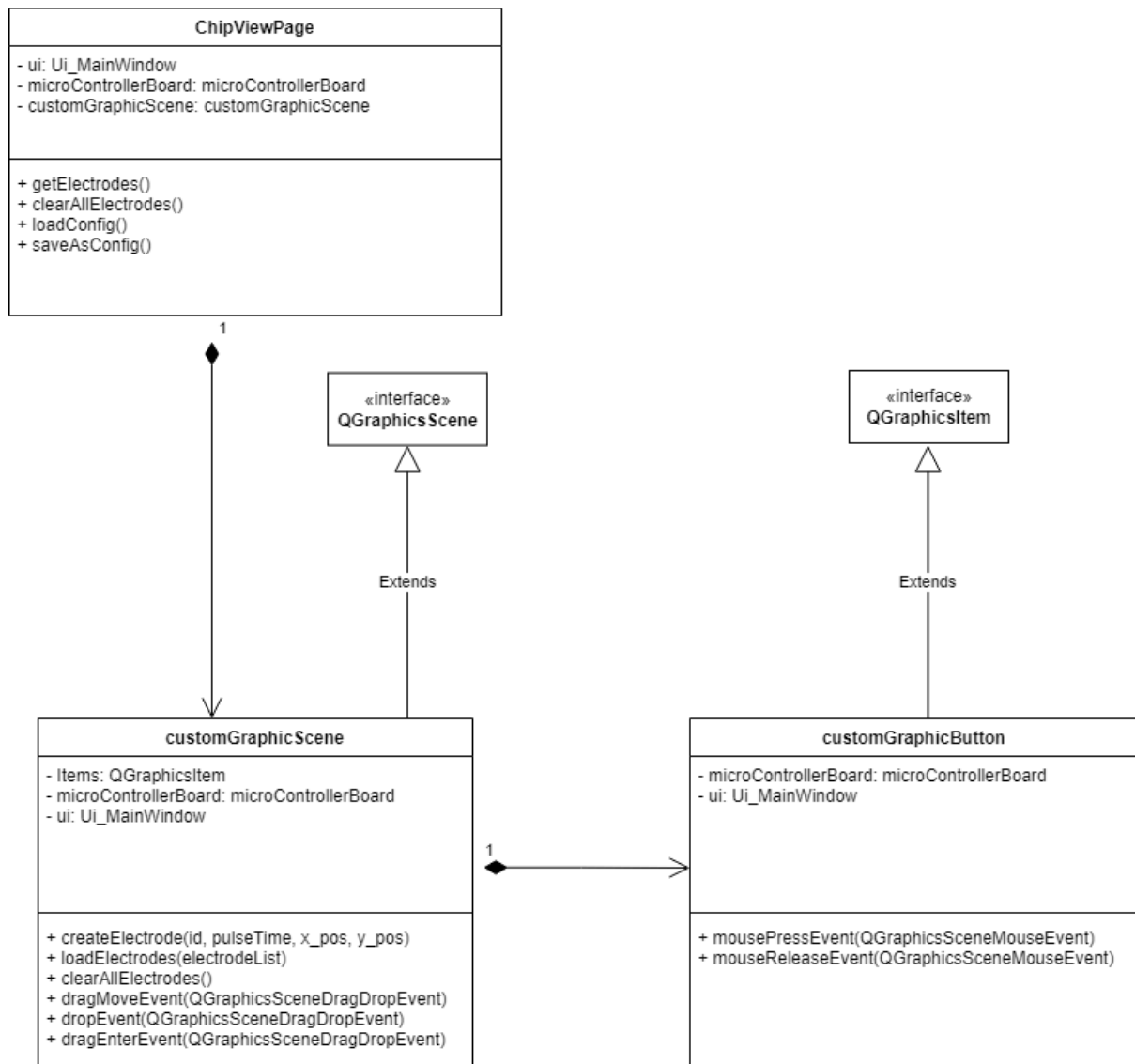


Figure 21 caption

2.4.6. Thread Workers

There are two main worker classes namely are the pump worker and queue worker. The pump worker main functionality is to execute pump functions so multiple pumps can run concurrently through `QThreadPool` that was implemented in the `configPage.py` file. As for the queue worker, it will actively be listening to the spectrometer data available in the queue so when if data is available, it can be sent to plot in the spectrometer view. The two workers implementation are stored in the `threadWorker.py` file.

2.4.7. Microcontroller Communication in Graphical User Interface Process

The communication between the Graphical User Interface process and the microcontroller is taken placed in the `microcontrollerCom.py` file. The implementation of the class is at first, the class will load the available comports that are connected to the system. After that, it will load the port mapping file that being stored in the developing folder where each port is map

corresponding to the chip design. There are 4 main functionalities being implemented in the class namely are the open selected comport, get data from serial port, close serial port and write parameter to the connected port. The functionality of opening the comport is that it will set up the connection with the selected comport, set the baud rate, open read and write channel and connect data handling to the get data function when data is available. The get data function is that it will first convert the received data to bytes type and decode it to utf8 format and append the converted data to the microcontroller console. For the close comport method, it will first check if there are any connections available, if yes, it will terminate the port and set the port variable to none. Finally, for the write param function, it will take in the droplet travel time, pulse time and electrode to be switched parameters, convert it to string and send it to the microcontroller board.

2.4.8. Yaml File Reader

The yamlFileRead.py file provides a way for the graphical user interface to load and save the configuration parameters from the yaml file. There are four main functions in the file which are the saveConfig, loadConfig, saveElectrodes and loadElectrodes. The first two functions will load and save all the parameter to and from the graphical user interface. The last two functions will be responsible for loading and save the electrode configuration in the chip viewer page.

2.4.9. Sorting Process and Flame Data Processing Class

The main processing of the spectrometer is being done in the sorting process where the implementation is being stored in the sortingProcess.py file. The process at first will take in the parameters sent from the queue and use it to initialize the data process class which being implemented in the getFlameData.py file. After initialization of the class, it submits the three main tasks to the thread pool that are the get spectrometer data, process spectrometer data and send data to plot functions. For the data acquisition the seabreeze API will provide us the intensity and wavelength. As for the data process of spectrometer, it's first subtracted the most recent data to the background noise previous acquired in the data process class initialization, after that, it will check if there are any values that is greater than the predefined gates, if so, it will send a trigger signal to the microcontroller board. The data process function is being optimized via Numba JIT decoration function and Numpy API. Finally for the send data to plot function, it will take the spectrometer data, concatenate the wavelength array and intensity array, and send it to queue for the GUI process to plot the data.

2.5. Problems Encountered

2.5.1. Controller

In our phase 2 report an alternative but similar RTOS sorting algorithm was proposed where, instead of having a queue-based system, each time a droplet is detected the detection thread would create a temporary thread to sort the droplet that gets deleted once the

sorting of the particular droplet is done. In this case each droplet will have its own thread spawned by the detection thread to actuate the electrode at the right time. This structure worked fine and got good results while testing, but a major problem soon arises where, the temporary task is done sorting and is deleted, the heap memory allocated to that task is not freed, so after roughly 1000 creation/deletion of tasks, the system would run out of heap memory.

At the same time, we ran into another problem concerning the frequency at which FreeRTOS runs its schedule. FreeRTOS is not capable of context switching of less than 100 microseconds, this limits the resolution at which threads can be executed for. In our case, we needed precision in the range of 20 to 30 microseconds. Also once the scheduler is running, the time function `millis()` and `micros()` available on the teensy become unusable.

2.5.2. Graphical User Interface

There are 4 problems we have faced during the implementation of the graphical user interface which are the execution time of the detection function, the real time plotting, Python Global Interpreter Lock and Spectrometer View and Camera View unresponsiveness.

2.5.3. Detection Function Execution Time

Python is known to be a slow interpreter language. The reason for that is because the code is compiled on the fly. It takes significantly longer time to execute the code compared to other language like C and C++. The execution time can also be influenced a lot depends on the computing environment. For example, if many processes are running on the user's PC, the Windows OS maybe preoccupied with other processes. Since our project requires real time response, between different component of the systems, it is very important for the detection task finished in a reasonable time frame based on the scheduled tasks table. Using the pure python implementation for the detection algorithm, the below figure shows the execution time of it over one million runs.

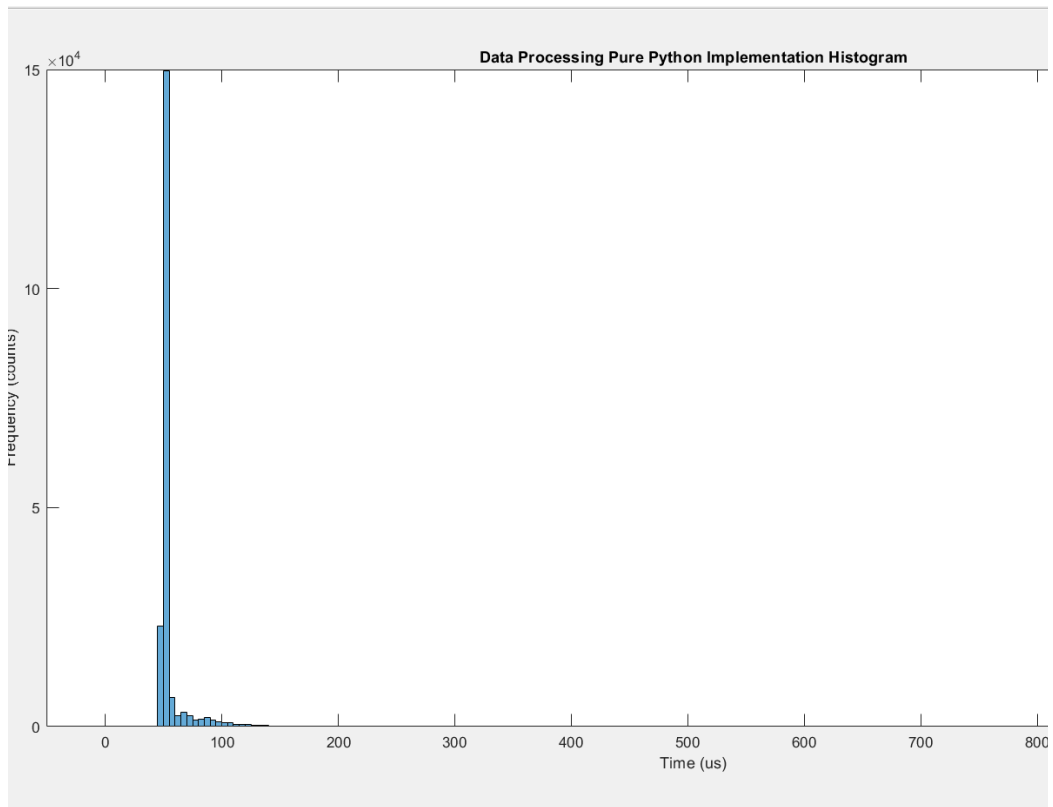


Figure 22 caption

The mean execution time of the function is to be around 55 microseconds, and the fluctuation of execution between run is very large around 500 microseconds. The percentage number of runs with execution time less than one hundred microseconds is to be around 87% which in term defined our sorting purity for the system as if an execution taking longer than the predefined range means a missing droplet. Hence, it is not fit for our design specification where we need a high sorting purity percentage.

2.5.4. Real Time Plotting

Another problem when implementing our design was that the delay of updating the spectrometer view. The fact that in our first implementation of the design, we used the built-in multiprocessing module of PyQt5 which is the QProcess function. It has a very slow communication speed between the two processes where the only mean of communication between the two process is via system standard output write where all the values must be in string format. The fact that we need to convert a large array with 4096 elements of float to string (sender side) and convert the string back to array of float (receiver side), it caused a huge delay when updating the plot, up to 2 seconds after the actual data received from the spectrometer. Hence, real time plot was not possible using the PyQt5 built in multiprocessing function.

2.5.5. Python Global Interpreter Lock

Python Global Interpreter Lock is a type of mutex (lock) where it only allows one thread to be run at a time to avoid the racing condition of values between threads. With that in mind, it is beating the purpose of concurrent run of tasks. Since with our application, we need to

execute multiple tasks in parallel namely are the data acquisition, sending the detection signal to the microcontroller and sending data for real time plot. It is important for us to be able to bypass the GIL to fully utilize the processing capability of the processor and maximizing sorting system throughput.

2.5.6. Spectrometer View and Camera View Unresponsiveness

While trying to embed the camera feature to the graphical user interface, we encountered the unresponsiveness when running the spectrometer view and camera view at the same time. The reason for that is due to two tasks were running in two threads and the amount of works among two threads are quite significant. It creates the unresponsiveness in the graphical interface itself which is very undesirable for our application where the responsiveness of the graphical user interface is critical for controlling the sorting system.

2.6. Solution Proposed

2.6.1. Controller

The solution we came up with is to switch to development in ChibiOS/RT. ChibiOS/RT is another real-time operating system for embedded application. It offers a preemptive and cooperative multi-tasking scheduler with priority levels and multi-threading primitives, including mutexes, condition variables and semaphores. ChibiOS is very optimized for the Teensy 4.1 and permits the use of the tickless mode. In this context switching is not done in a fixed interval but rather in a variable one depending on the different priorities and yield parameters of your different tasks. ChibiOS also provides timing function precise to the microsecond and also is compatible with the usage of Teensy's native timing functions `millis()` and `micros()`.

As for the algorithm, we changed how the sorting is handled by transitioning from a temporary thread system to an event queue system, where droplet events are queued and a sorting thread sorts the droplets based on the events in the queue, a full explanation of the current algorithm is available in the design section of this report.

2.6.2. Graphical User Interface

Each of the four problems encountered were solved using the methods highlighted below.

2.6.3. Detection Function Execution Time Solution

To combat the inconsistency run time of the detection function and to obtain a much higher sorting purity for our application, we came up with several solutions namely are moving the whole system to LINUX operating system, using C or C++ for the detection function or using Just In Time technology. With the simplicity of use and a very prominent solution, we went with using Numba, an API where it compiles python code into binary machine code for a much faster and consistency execution time. By combining Numba with Numpy which is an API with thin wrapper around optimized C functions for mathematical operations, the achieved execution speed was 10 faster than the pure python implementation. To further increase the stability of the execution time, we also change the sorting process priority to

high to prevent interruption from other process with the sorting process, giving a fully optimize detection function for our application.

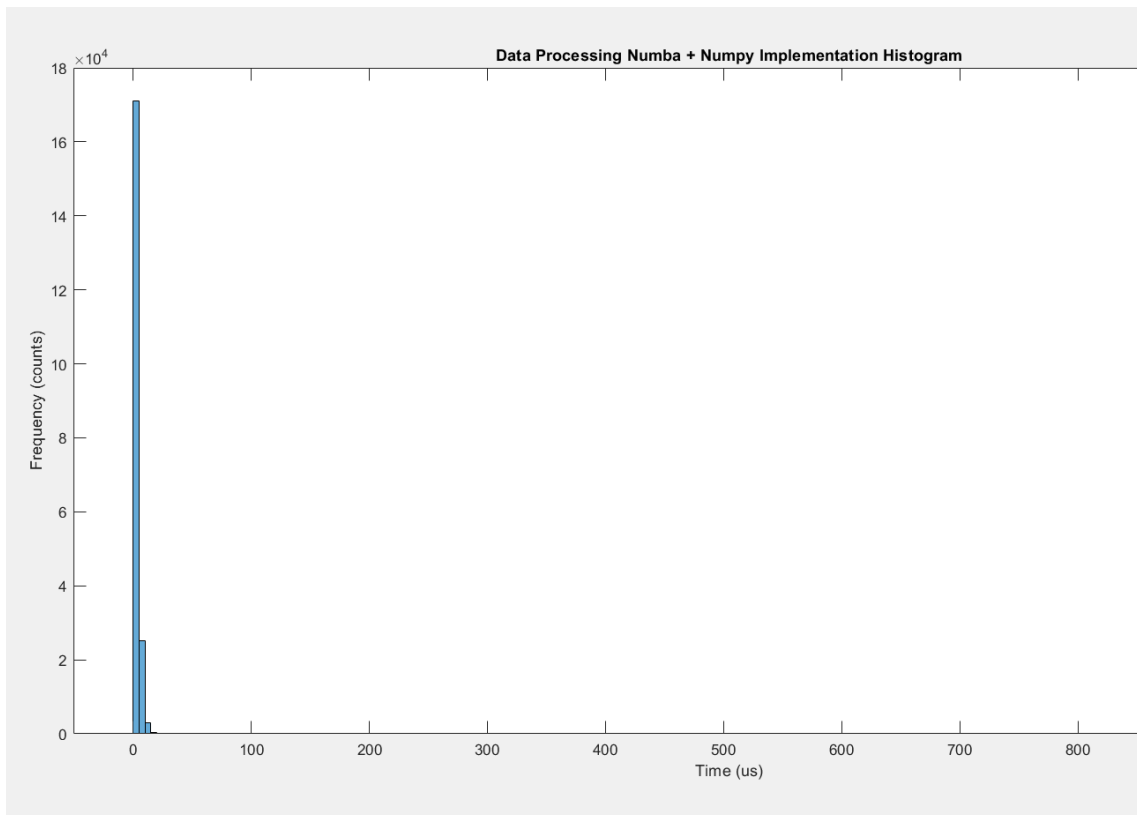


Figure 23 caption

The mean execution time of the detection using Numba and Numpy is to be around 4.9 microseconds which is a significant improvement over the previous implementation for the detection function. Not only that, but the variance between runs were also significantly less with only one to two microseconds difference between runs. With over a million run of the function, the percentage of number of runs that are less than 100 microseconds is to be around 99.9996% which is perfectly fit with our application where fast and consistence execution time of tasks are important.

2.6.4. Real Time Plotting Solution

The solution for fast update rate of the spectrometer view is to use the python multiprocessing module where the communication between processes can be done through queue. The fact that users can pass object element through the queue and require no conversion of array of float to string and convert the string back to array of float saves a significant amount of time. It makes the real time spectrometer view possible as data received from the spectrometer, array of 4096 elements can be sent directly to the graphical user interface process to be plot without any conversion of the data. Noticing that queue have a limit amount of memory size, managing the size of data to be send is also very

important as deadlock might occur during the sending and receiving data between processes.

2.6.5. Python Global Interpreter Lock Solution

With the implementation of Numba, we were also able to bypass the Global Interpreter Lock as the codes are being converted into binary code and the lock will also be released giving multitasking functionality in python possible. As for our case, the used of Numba for lock released were implemented for the detection function and data concatenation function which concatenate the data to be send for plotting. Hence the three threads, spectrometer data acquisition, detection function and process data to be sent for plot can be run concurrently.

2.6.6. Spectrometer View and Camera View Unresponsiveness Solution

The solution to solve the problem to offload the camera feature into a new process. We created another subprocess where it can be run in a separate process preserving the same performance of both the spectrometer and camera view without compromising the sorting performance which is hosted in another separate process.

2.7. Future Development

2.7.1. Optocoupler Board

One of the main bottlenecks for our current system is caused by the switching speed of the optocoupler board. With the current design using the MAX7300AAI+ port expander with the I2C write speed of 400 KHz, it takes 72 microseconds to turn on/off a pin and an addition of turn on/off delay of the photoMOS relay, AQW216H of around 250 microseconds making the total delay of around 320 microseconds per execution on or off. To reduce the delay of switching, we came up with a design where two modifications on the current design were made. The first is that the MAX7300AAI+ can be replaced by the MAX7301AAI+ version where it used the SPI communication protocol which have a write speed of 26MHz that is 65 times faster than the current chip giving the turn on/off a pin on the port expander to be around 1 microsecond. The second modification is that we add an Inrush Current Circuit before the input stage of the photoMOS relay where it can exploit the inrush current property to first increase the current passing through LED significantly and later reduce it to improve the life span of the LED. According to the datasheet of the AQW216H photoMOS shows in figure x, with higher current input to the LED will result in a much faster switch on/off time. Hence, with our modification, it will provide a faster turn on time and turn off time for the photoMOS relay. The circuit design is shown in figure x.

10. Turn on time vs. LED forward current characteristics

Measured portion: between terminals 5 and 6, 7 and 8;
Load voltage: Max. (DC); Continuous load current:
Max. (DC); Ambient temperature: 25°C

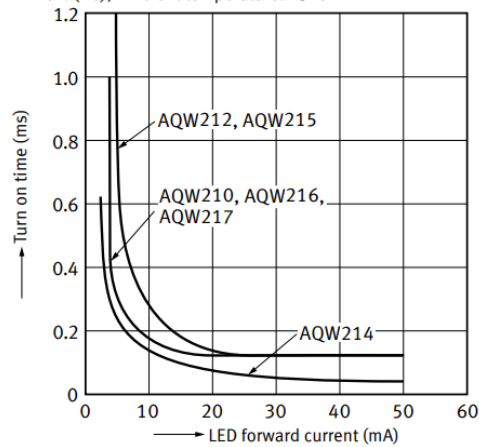


Figure 24 caption

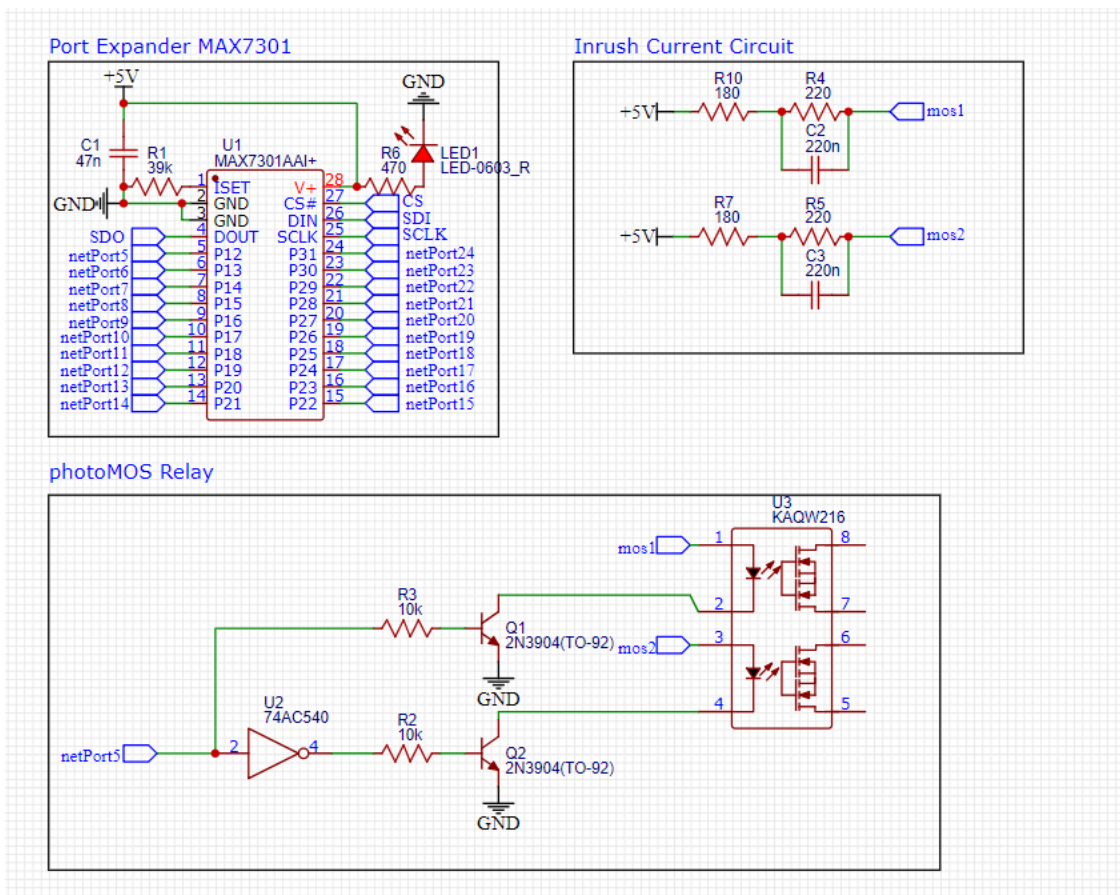


Figure 25 caption

By using PSPICE, the below graph shows the transient response of the inrush current circuit where at first the current going through the photoMOS's LED get a significant boost of up to 20ma and later reduced to 12.5 ma due to capacitor are fully charge up and the current path

has to pass the second resistor of 220 ohms which in term reduce the current flowing through the LED. The theoretical turn on time/ turn off time for our circuit based on the design and the data sheet provided by the manufacturer will be around 180 microseconds. As a result, the modified circuit will be able to decrease the total delay to be around 181 microseconds per execution on or off which is 1.8 times faster than the current design.

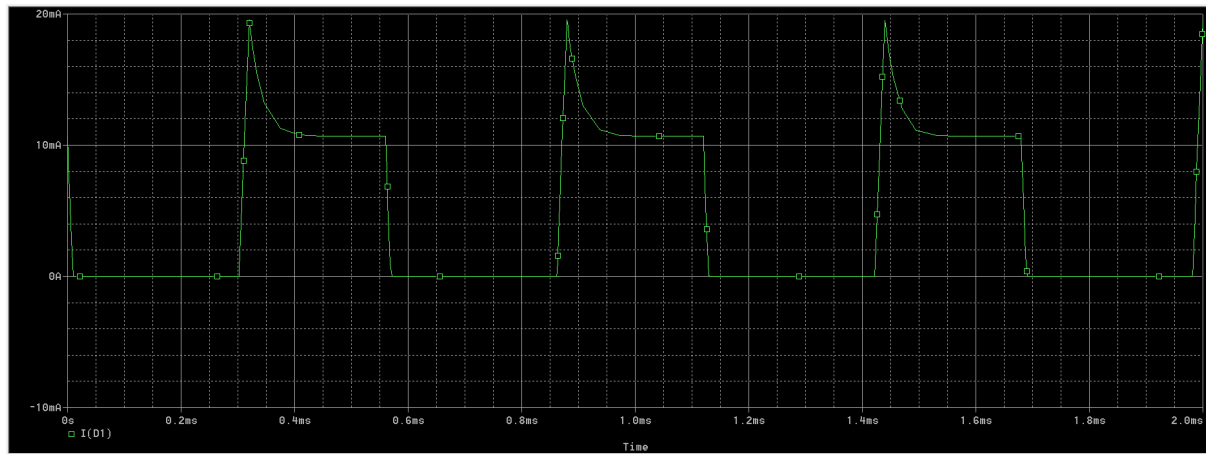


Figure 26 caption

2.7.2. Chip Viewer

With our implementation of the chip viewer feature, future work on it can allow various of droplet manipulation techniques to be implemented by controlling the sequence on time/off time of the electrode arrays. In addition, the save and load schematic feature will allow a flexible configuration of many different droplet manipulation functions in microfluidics technology.

2.7.3. Camera Viewer

The current implementation of the camera view is using OpenCV to display the frame captured from Hamamatsu Camera in a separate process. Future work on the camera feature can be expanded to many different areas by using the built-in functionality of OpenCV or in combination with many available image processing API for object detection, tracking and machine learning to be able to fully analyze droplet movement autonomously using the technologies. Noticing that, for high-speed droplet analysis, a need of high-speed camera is critical to capture the motion of the droplet.

3. RESULTS OBTAINED

3.1. Timing Analysis

Based on our project requirement of 1000 droplets or more we should expect to set the time between two consecutive electrodes' actuation signals as 1milli second or less. However, because our current spectrometer can spit out raw data approximately every 2.5 milli seconds, our sorting system can handle up to 400 droplets. The delta value of Figure A represents the time between the two consecutive signals measured in the oscilloscope.

Figure B and C represent the delay of the I2C signal. B shows that it takes approximately 350 microseconds for electrode to be turned off completely after the I2C signal is sent. C shows that time between the I2C signal is sent and Electrode to be turn on. It was recorded as 240 microseconds.

Figure D,E,F represents the results from the test when the integration time is set to be 600 microseconds. Figure D shows that I2C signals are sent precisely every 600 microseconds. Figure F shows that electrodes are fully on for precisely 600 microseconds of duration. Figure E shows that error of I2C is as small as 72 microseconds.

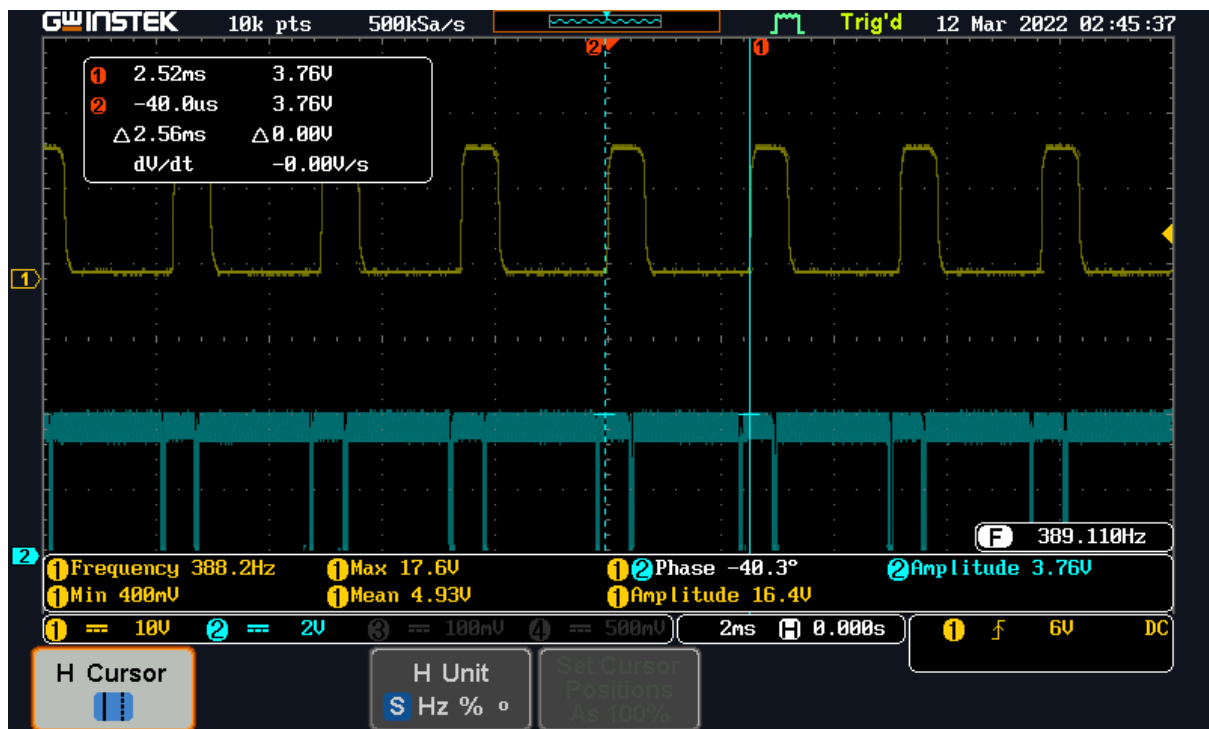


Figure A27

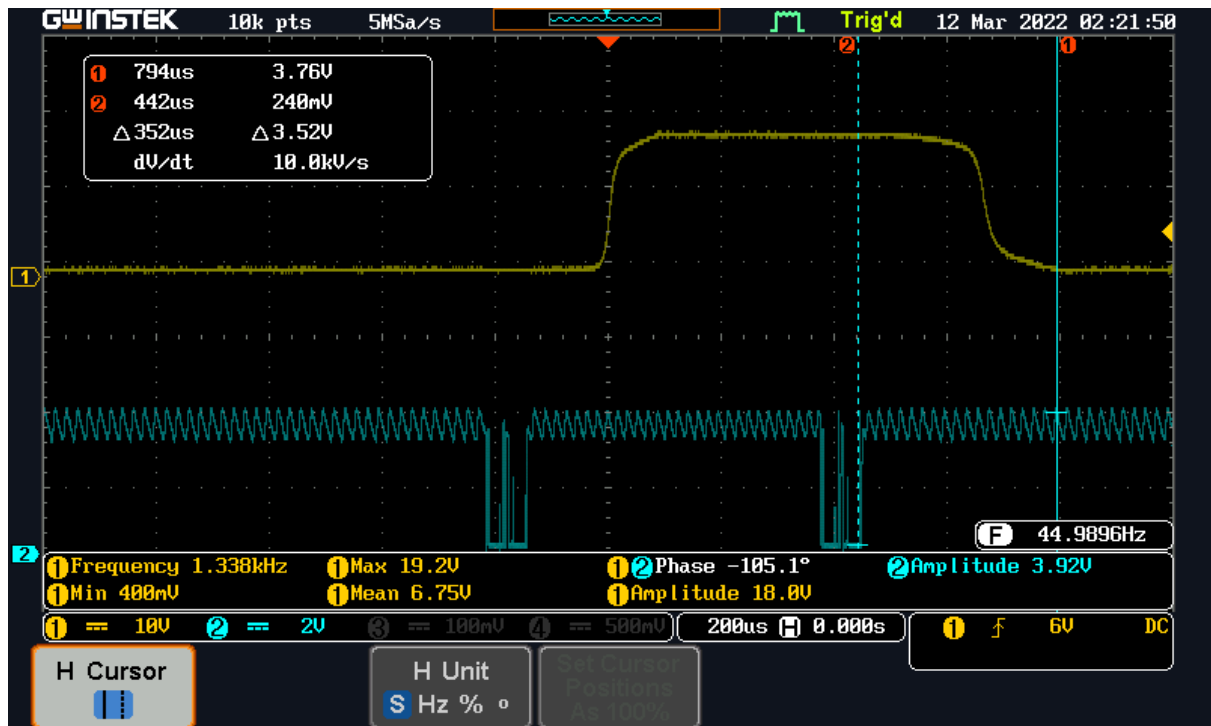


Figure B28

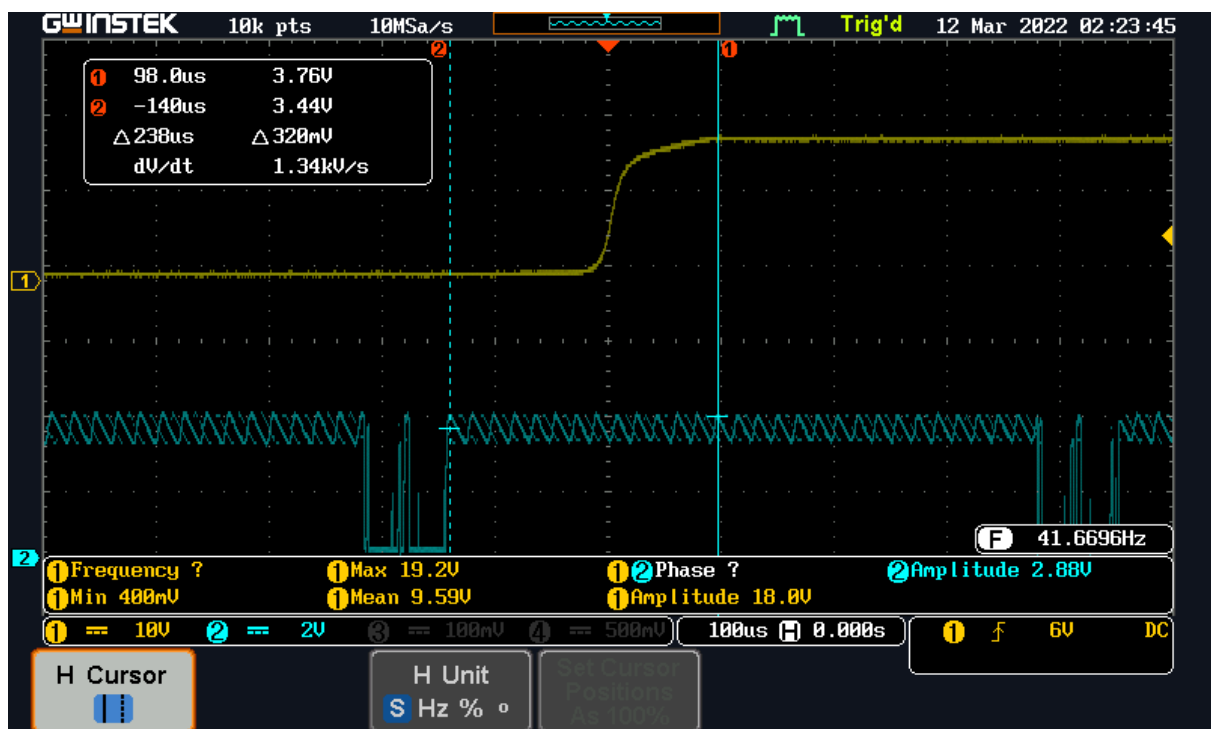


Figure C29

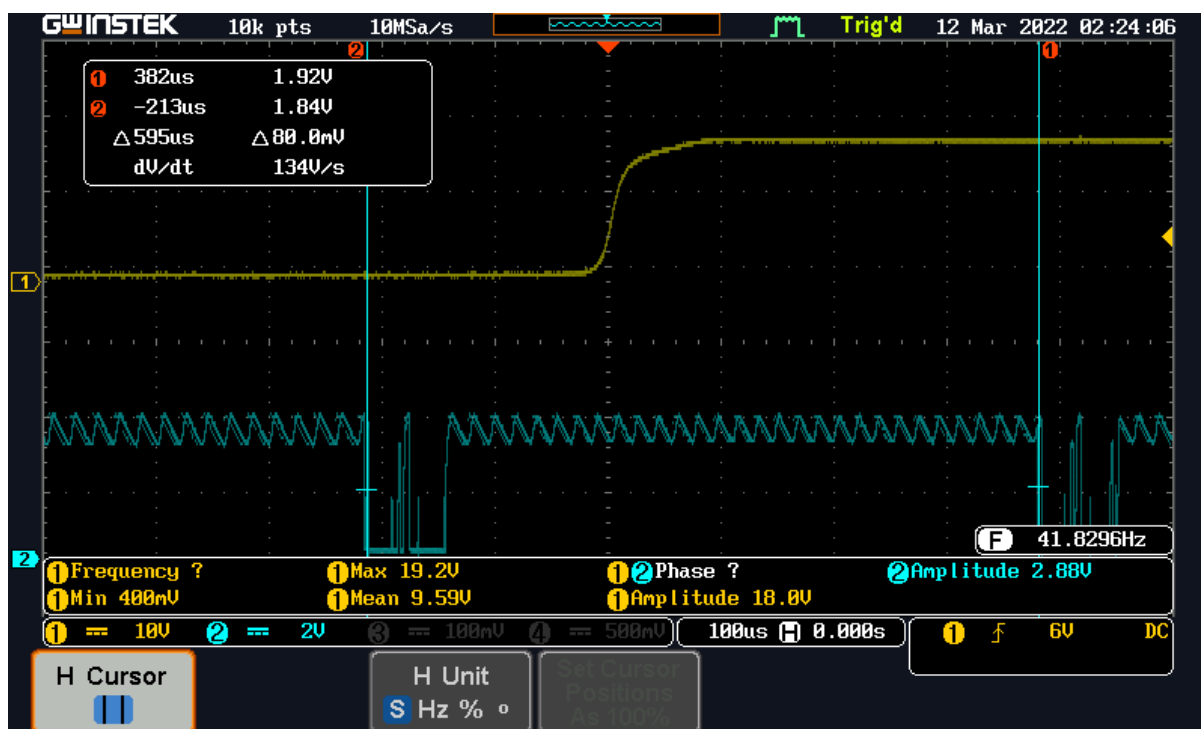


Figure D30

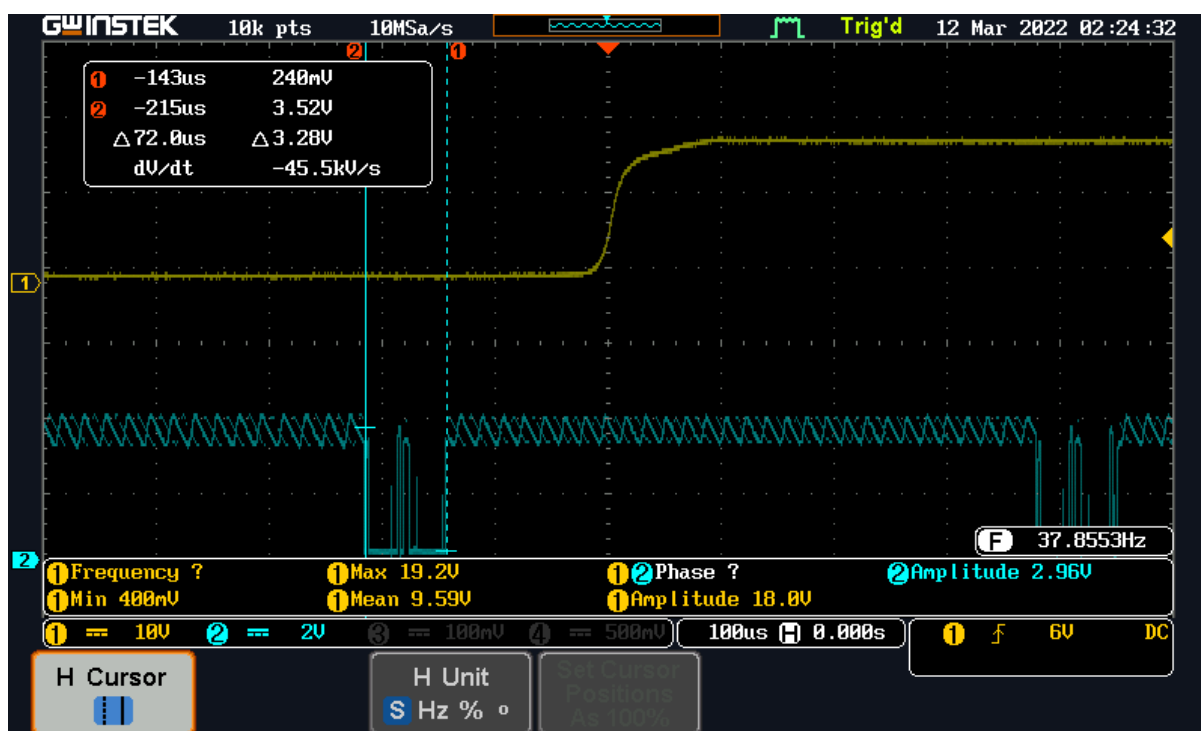


Figure E31

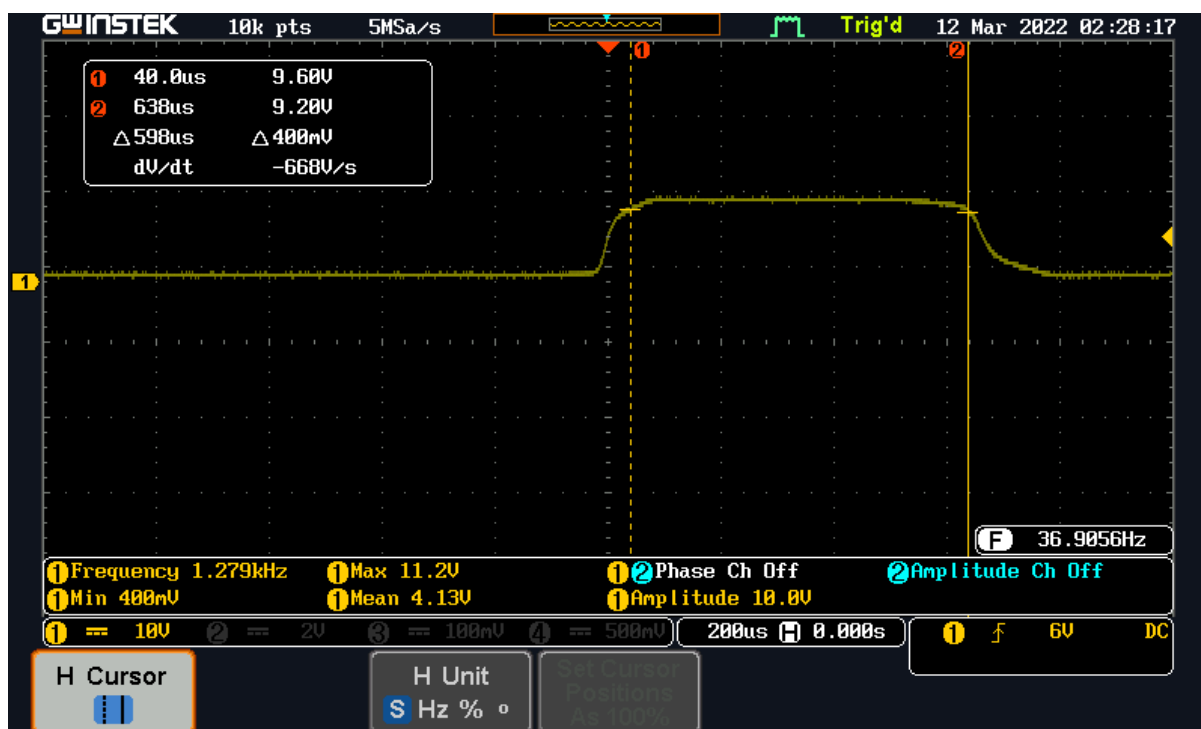


Figure F32

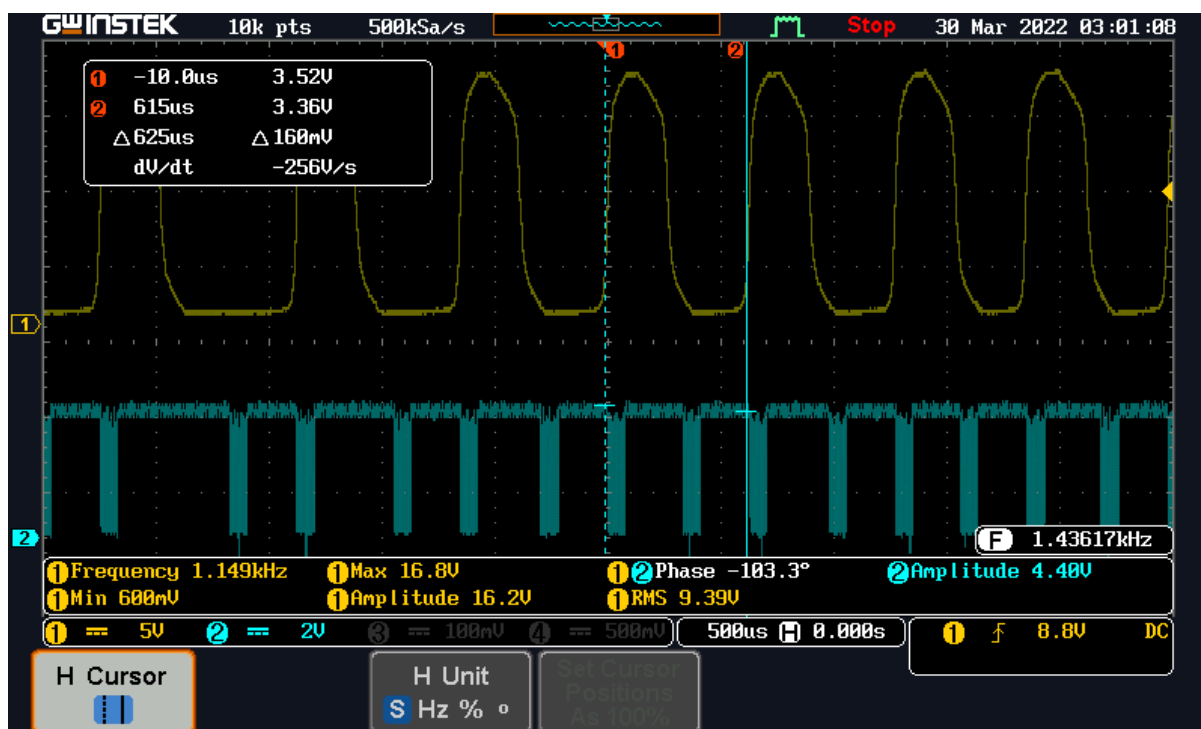


Figure 33 High throughput 1600 droplets

3.2. Test Cases

Table 4 Test Cases

ID	Responsible	Scope	Procedure	Input	Expected	Actual	Conclusion
1	Team	Check: Console message Equipment: PC, microcontroller	<ol style="list-style-type: none"> 1. Open up the GUI 2. Select The COM port 3. Send Sorting parameters 	*User can make a change to the default sorting parameters	User should be able to make connection with the microfluidic sorting controller	Successfully connected to the controller as we expected	Pass
2	Team	Check: Oscilloscope output Equipment: PC, microcontroller	<ol style="list-style-type: none"> 1. Open up the GUI 2. Open the ChipViewr page 3. Add some electrodes in the scene 4. Click to see if electrodes actuate in an oscilloscope 		User should be able to see all the interactable electrodes through the interface	Successfully controlled all electrode on the scene through the interface	Pass
3	Team		<ol style="list-style-type: none"> 1. Open up the GUI 	Pulse time and ID for	User should be able to assign functionality	Successfully assigned function	Pass

			<ol style="list-style-type: none"> 2. Open the ChipViewer page 3. Specify the pulse time and ID of the electrode 4. Add it to the scene 5. Click to check if the electrode is actuated for given period of time in an oscilloscope 	each electrode	to each/group of electrodes	ality to each electrodes	
4	Team		<ol style="list-style-type: none"> 1. Open up the GUI 2. Open the ChipViewer page 3. Input the Maximum and Minimum for the intensit 	Maximum and Minimum for the intensity and wavelength	User should be able to set the trigger threshold	Successfully set the trigger threshold to the specified value	pass

			<p>y and wavelen gth in the config page</p> <ol style="list-style-type: none"> 4. Start sorting 5. Verify the result by open the microso pe view in the GUI 				
5	Team		<ol style="list-style-type: none"> 1. Open up the GUI 2. Start storting 3. Check time betwee n two consecu tive sorting cycles 4. Calculat e the total output in 1 seconds 		Droplet Actuation > 1000	400	Fail *The test failed due to the spectro meter not able to spit out raw data less than 2.5milli seconds.
6	Team		<ol style="list-style-type: none"> 1. Prepare a script that generat e a random 		Droplet Actuation > 1000	1600	Pass

			<p>value in less than 1 milliseconds and send actuation signal to the microcontroller</p> <p>2. Run the script</p> <p>3. Check time between two consecutive sorting cycles</p> <p>4. Calculate the total output in 1 seconds</p>				
7	Team		<p>1. Open up the GUI</p> <p>2. Start sorting</p>		Sort Purity >95%	99.98%	Pass
8	Team		3.				

4. ELSEE ASPECTS

The objective of this capstone project is to improve the speed of sample preparation for microfluidic applications such as encapsulating cells in water droplets and sorting the droplets according to their content. We aim at allowing researchers to only sort droplet that contains some material of interest, such as droplets containing the desired cells that will be analyzed or genetically modified, at a faster rate making this automated process much less time consuming hence increasing the productivity and efficiency of many different lab processes, that today could take from days to months to complete on very large costly analysis machines. This makes the science community one of the main stakeholders of this project on a broad term. On the scale of our project, the principal stakeholders would be all the university's laboratories, including our supervisor's

Being able to select cell groups by sorting droplets on a microscopic scale at very high speeds will permit Dr. Steve Shih's lab to advance their research by reducing their scale of operation to a microfluidic system and the time required to finish research. Previous implemented solutions had performances 1000 times less than the system we designed, this shows how important and advanced the system we produced is and how much it would help advance the lab's work throughput.

As of now the ethics scope of our project is extremely limited inside laboratory settings but thinking ahead and imagining what these microfluidic technologies could become, we can identify benefits on diverse levels but also ethics concerns once these systems become commercially and publicly available. These benefits and concerns are not specific to a microfluidic sorter but are broader and touch on the advances bioengineering is rapidly making.

If our product gets used publicly then many stakeholders would emerge. For instance, if our product was used as part of a drug or therapeutic development, then our product would increase the speed of the process and increase efficiency in terms of material used, hence reducing the overall cost, and allowing the development of new drugs faster. This brings to light the social and economic aspects of our product as it would increase the quality of life for the public and increase profits for pharmaceuticals and research institutions. In addition, research companies and DNA analysis institutions such as 23andMe and others, would be allowed to prepare and analyze DNA samples on a much faster rate, allowing those institutions to recognize disease-prone genes faster and allowing potentially ill people to act quickly by either taking preventative measures or take therapeutics before the diseases can develop and spread in their body. On the other hand, allowing commercial institutions such as 23andMe, Ancestry and others to analyze the genes of people faster means that those institutions would be able to increase their database at a faster pace than governmental policies. This brings to question the ethical use of the database and the potential privacy breach of the people as those companies can act in ways that do not protect their clients' interests by selling their medical information to third parties like insurance companies which in return would discriminate their prices by charging higher fees to people with potential illness.

Our microfluidic droplet sorter can be mostly implemented in the biomedical field such as in the development of synthetic biotics and therapeutics which highlights the process of directed evolution. If the process were not handled properly, drug-resistant bacteria would appear causing environmental and medical issues.

For our prototype, there is not much we can do, whether in functionality or design architecture, to make the product more ELSEE compliant. But there exist some alternative options, such as communicating with governmental representatives to bring to their attention the possible unethical and malicious use by institutions with fast DNA and RNA sampling, analyzing, and editing. This would help enact policies that put strict and clear policies around them such as prohibiting the sale of medical data without the informed consent of the public, like in the European Union where the EU Data Protection Directive states that “the person from whom data is obtained should be informed of what will be done with this information and to whom it will be disclosed, Individuals can consent, withdraw or correct the data”[7]. In addition, campaigns can be made to the public by media outlets or social media to raise their awareness of the privacy concerns and the potential unethical use of their medical information, and to shed bright light on any medical privacy breach incident. Privacy breaches can also occur by unauthorized personnel such as hackers and one way to protect the public is by anonymizing or de-identifying the data so that neither hackers can blackmail individuals nor allowing insurance companies to discriminate their prices between their clients.

List of References

1. Fatemeh Ahmadi 1,2 Kenza Samlali 1,2 Philippe Q. N. Vo 1,2 Steve C.C. Shih*1-3, An integrated droplet-digital microfluidic system for on-demand droplet creation, mixing, incubation, and sorting, 2019, P11
2. S. S. Schütz, T. Beneyton, J.-C. Baret, and T. M. Schneider, "Rational design of a high-throughput droplet sorter," *Lab on a Chip*, 03-Jun-2019. [Online]. Available: <https://pubs.rsc.org/en/content/articlelanding/2019/lc/c9lc00149b#!divAbstract>. [Accessed: 25-Nov-2021].
3. "Teensy® 4.1 Development Board," *PJRC*. [Online]. Available: <https://www.pjrc.com/store/teensy41.html>. [Accessed: 25-Nov-2021].
4. Author Information ARTICLE SECTIONS Jump To Corresponding Author Petra S. Dittrich – Department of Biosystems Science and Engineering, C. Author, Petra S. Dittrich – Department of Biosystems Science and Engineering, E. , Authors, Todd A. Duncombe – Department of Biosystems Science and Engineering, Aaron Ponti – Department of Biosystems Science and Engineering, Florian P. Seebeck – NCCR Molecular Systems Engineering, Notes, and The authors declare no competing financial interest., "UV-vis spectra-activated droplet sorting for label-free chemical identification and collection of droplets," *ACS Publications*, 17-Sep-2021. [Online]. Available: https://pubs.acs.org/doi/10.1021/acs.analchem.1c02822?utm_source=pubs_content_marketing&utm_medium=twitter&utm_campaign=PUBS_0821_ECR_AC_ancham_AuthorTweets&src=PUBS_0821_ECR_AC_ancham_AuthorTweets&ref=pubs_content_marketing_twitter_PUBS_0821_ECR_AC_ancham_AuthorTweets&. [Accessed: 25-Nov-2021].
5. "2-Wire-Interfaced, 2.5V to 5.5V, 20-Port or 28-Port I/O Expander." Maxim Integrated, San Jose, 13-Oct-2014.
6. "AQW214 Panasonic Datasheet." Panasonic, Kadoma, Dec-2019.
7. European Union, "Data protection under GDPR", 26-March-2021. https://europa.eu/youreurope/business/dealing-with-customers/data-protection/data-protection-gdpr/index_en.htm

HIGH-THROUGHPUT DROPLET-BASED MICROFLUIDIC SORTER USER MANUAL

GUI Startup

Upon startup the interface will look as shown below:

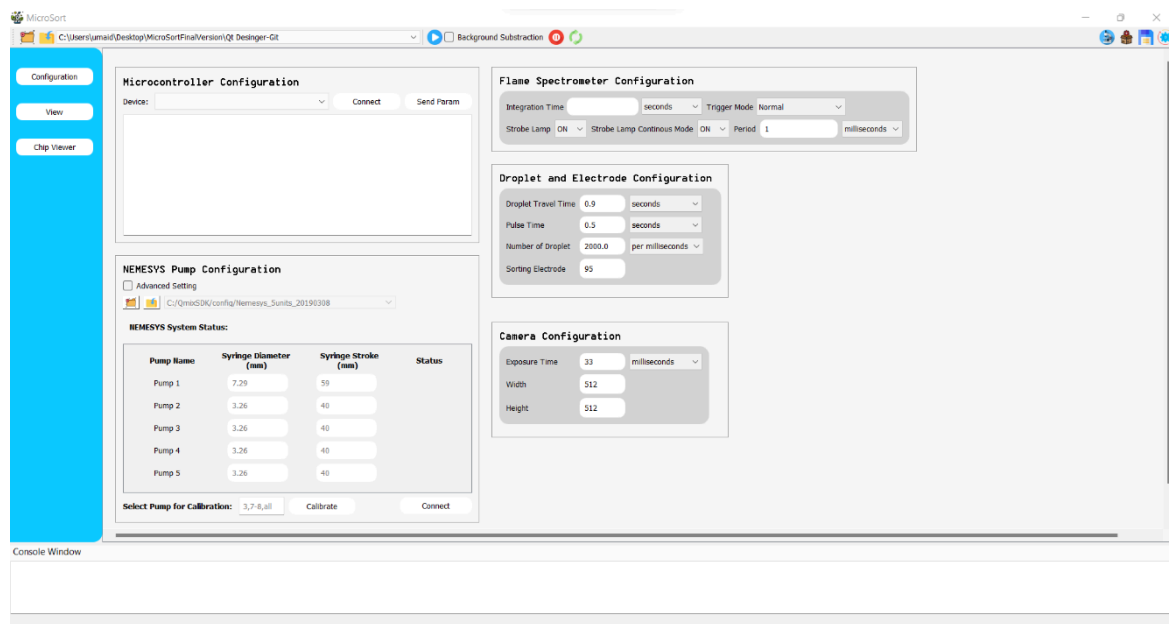


Figure 34 Main Interface

The left pane shows three buttons use for switching between the three pages; Configuration Page, View Page, and ChipViewer Page. The configuration page is displayed upon startup.

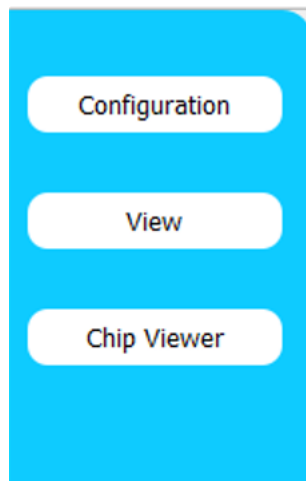


Figure 35 Left Pane

Camera Startup

The microscope camera can be activated by clicking the microscope icon button on the top right of the window:



When successfully loaded, it will display as shown below:

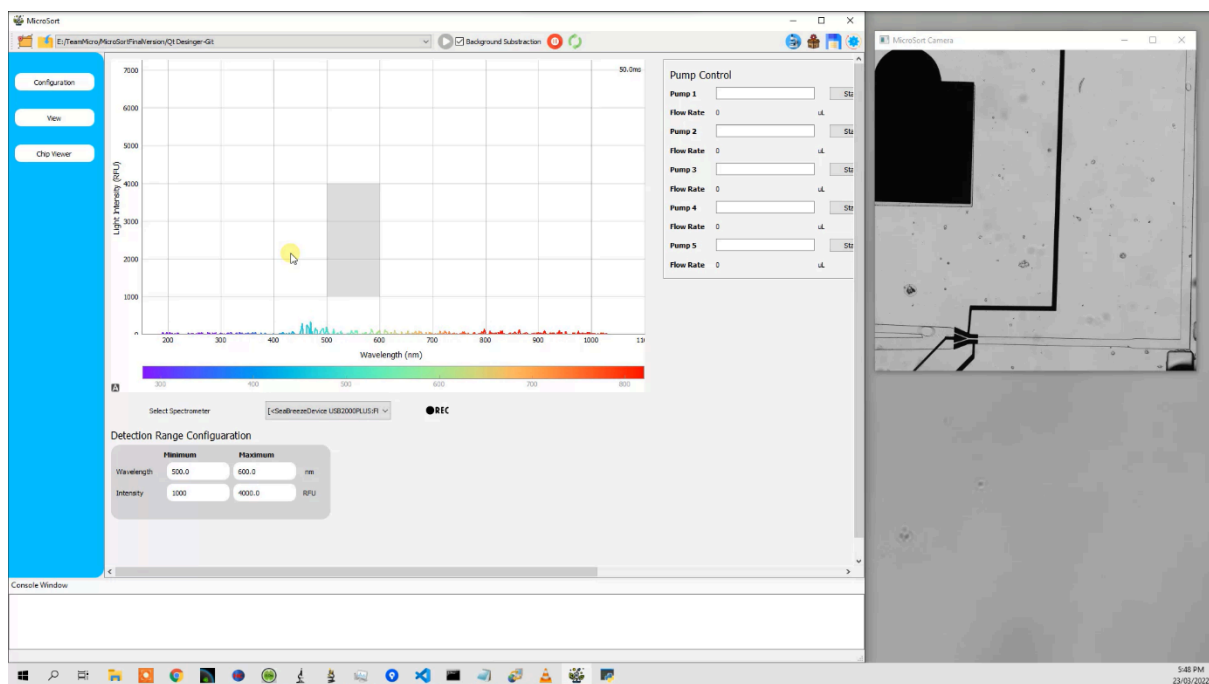
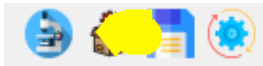


Figure 36 Full GUI



Figure Microscopic view of feeder, sorting and waste channels, along with the electrodes

Loading a Configuration



Configurations can be loaded by clicking the 'box and gear' icon. Clicking this button will open the file explorer where a configuration file can be loaded for quickly setting the previous parameters.

Saving a Configuration



Parameters set on the interface can be saved as a configuration YAML file by clicking the save icon. Clicking this button will open the file explorer to specify the file destination.

Setting the Default Configuration



Whenever the application is started, there is a default configuration file that will be loaded. These initial parameters can be updated by clicking this button.

Configuration Page

The screenshot displays a web-based configuration interface with five distinct panels. The 'Microcontroller Configuration' panel includes a 'Device' dropdown and 'Connect'/'Send Param' buttons. The 'Flame Spectrometer Configuration' panel features settings for 'Integration Time', 'Trigger Mode', 'Strobe Lamp', 'Strobe Lamp Continuous Mode', and 'Period'. The 'Droplet and Electrode Configuration' panel contains 'Droplet Travel Time', 'Pulse Time', 'Number of Droplet', and 'Sorting Electrode' settings. The 'NEMESYS Pump Configuration' panel includes an 'Advanced Setting' checkbox, a file path, and a table for pump status. The 'Camera Configuration' panel has 'Exposure Time', 'Width', and 'Height' settings.

Microcontroller Configuration

Device:

Flame Spectrometer Configuration

Integration Time seconds Trigger Mode Normal

Strobe Lamp Strobe Lamp Continuous Mode Period 1 milliseconds

Droplet and Electrode Configuration

Droplet Travel Time 0.9 seconds

Pulse Time 0.5 seconds

Number of Droplet 2000.0 per milliseconds

Sorting Electrode 95

NEMESYS Pump Configuration

☐ Advanced Setting

C:/QmixSDK/config/Nemesys_Sunits_20190308

NEMESYS System Status:

Pump Name	Syringe Diameter (mm)	Syringe Stroke (mm)	Status
Pump 1	<input type="text"/> 7.29	<input type="text"/> 59	
Pump 2	<input type="text"/> 3.26	<input type="text"/> 40	
Pump 3	<input type="text"/> 3.26	<input type="text"/> 40	
Pump 4	<input type="text"/> 3.26	<input type="text"/> 40	
Pump 5	<input type="text"/> 3.26	<input type="text"/> 40	

Select Pump for Calibration: 3,7-8,all

Camera Configuration

Exposure Time 33 milliseconds

Width 512

Height 512

Figure Configuration UI

The configuration page is sectioned off by five separate panels:

Microcontroller Configuration:

The dropdown button shows the list of controller devices to connect by selecting it and then clicking the 'Connect' button.

'Send Param' button sends the parameters written in the 'Droplet and Electrode Configuration' panel to the connected controller.

NEMESYS Pump Configuration:

By default, this panel is disabled. To enable, click the 'Advanced Setting' checkbox. The pumps must first be connected by clicking 'Connect' within the same panel. The syringe's swept volume can be set using the 'Syringe Diameter' and 'Syringe Stroke' fields. Once set, the 'Calibrate' button must be clicked to update the setting.

Flame Spectrometer Configuration:

Various parameters of the Flame Spectrometer can be controlled on this panel such as the turning ON/OFF the strobe lamp and the Continuous Mode as well. The period of the pulses can also be set in the 'Period' text field with units of seconds or milliseconds using the dropdown button.

'Integration Time' sets the intensity of luminescence for detection. Droplets with higher fluorescence do not require as much integration time over one

with lower fluorescence. Can be set in units of seconds or milliseconds using the dropdown button.

Control over the data acquisition can be set by the 'Trigger Mode' dropdown. By default, it is set to 'NORMAL'.

Droplet and Electrode Configuration:

'Droplet Travel Time' is defined as the time the droplets takes from the moment of spectrometer detection till reaching the sorting electrodes. Can be set in seconds or milliseconds.

'Pulse Time' is used to set how long the sorting electrode will remain active (in HIGH voltage). Pulse time is proportional to the voltage level. Can be set in seconds or milliseconds.

'Number of Droplet' is used to indicate to the microcontroller, the number of droplets begin generated so that it allows for limiting the number of electrode actuations in that timeframe.

'Sorting Electrode' is used for setting which corresponding electrode will be used for sorting the droplets

Camera Configuration:

This panel allows for adjusting the 'Exposure Time' in milliseconds.

The camera viewer window size is set by the 'Width' and 'Height' parameters in pixels.

Spectrometer View Page

Accessed by clicking 'View' on the left pane. The spectrometer must first be selected using the 'Select Spectrometer' dropdown menu. The pump control is still accessible in this page

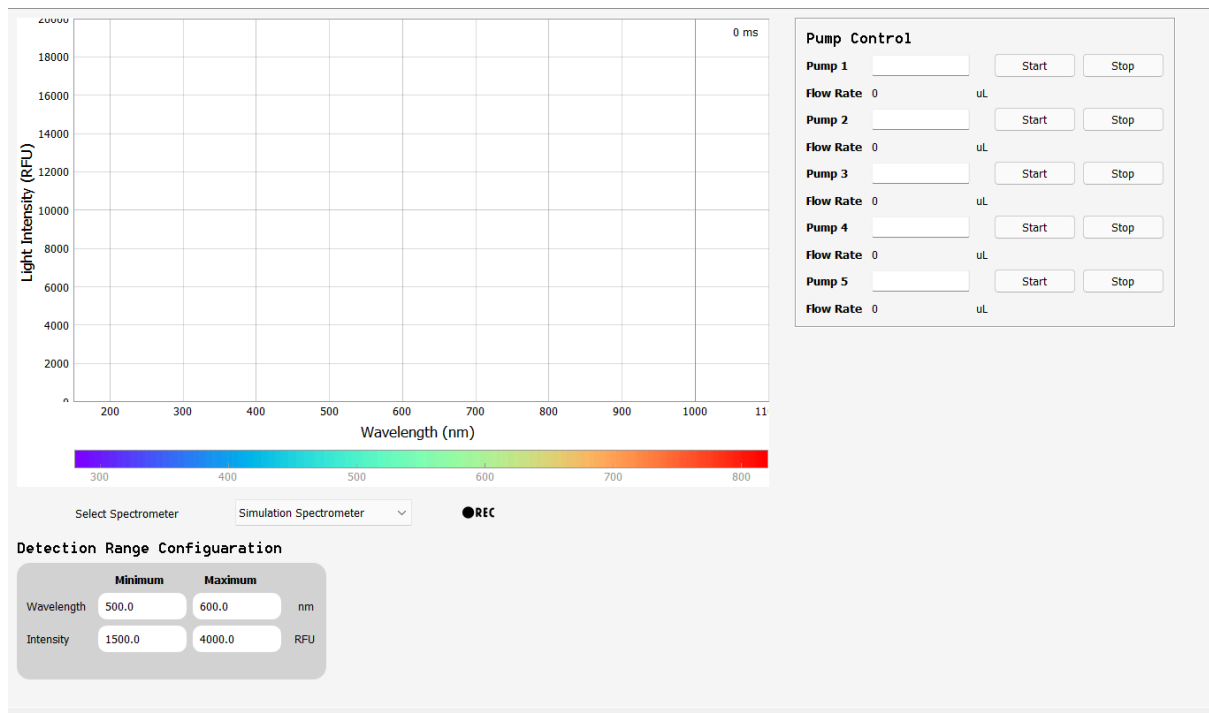


Figure 39 Spectrometer View UI

Detection Range Configuration:

The wavelength and intensity are set with a minimum and maximum for the range of detection of the droplets.

Clicking the 'Play' icon will start the sorting and the spectrometer graph view can be viewed as droplets are being detected.

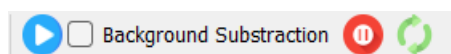


Figure 40 Sorting Control

Chip Viewer Page

Accessed by clicking 'Chip Viewer' on the left pane. The electrodes are dragged and dropped on the gray background. The 'Clear All' button removes all present electrodes shown on the UI. 'Load' allows saved electrode configurations to be loaded on the UI. 'Save As' is for saving the current list of electrodes displayed as a YAML file.

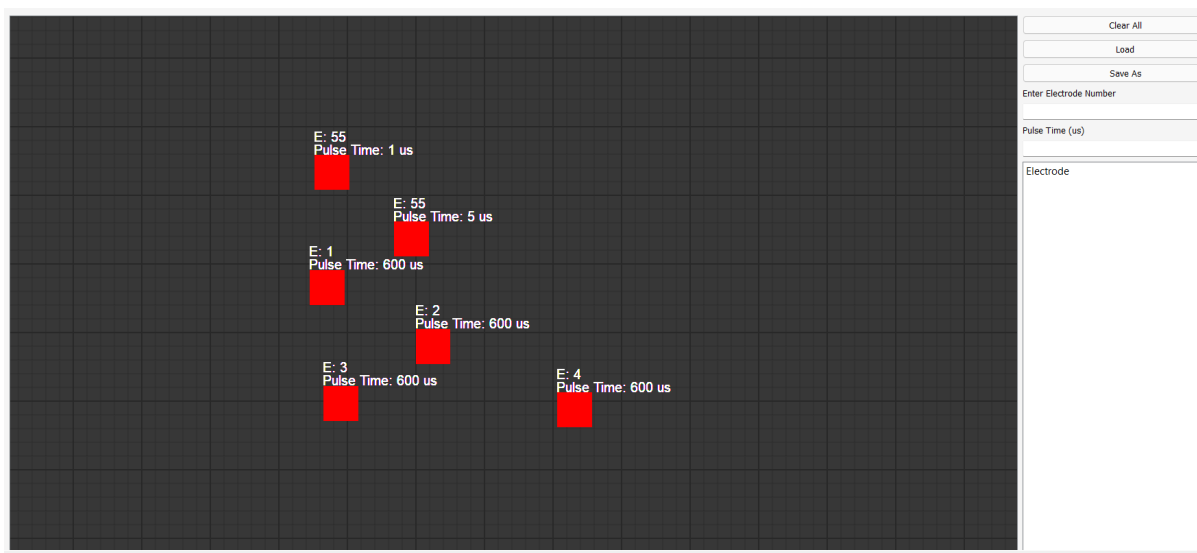


Figure 41 Chip Viewer UI