# Automated wet-etch system with near-UV/visible light spectrum-based endpoint detection

**Electrical and Computer Systems Engineering**
**ECE4095- Project B**
**FINAL YEAR PROJECT REPORT**
**S2, 2021**

**Date: OCTOBER 22**
**Student: Chew Lik Siang**
**ID: 29035058**
**Supervisor/s: Dr. Ajay Achath Mohanan**

MONASH University

# Abstract

Microfabrication of surface acoustic wave (SAW) devices usually undergoes conventional procedures such as photolithography, development, and etching. Recently, research interests have emerged in fabricating flexible SAW devices as an attempt to incorporate into consumer wearables. Hence, the “Si-SU-8-Al-ZnO-Al” device is currently investigated heavily in the lab. Due to the high transparency of ZnO and the double Al layers in the multi-layered structure, the etching endpoint determination for fabricating electrodes is extremely difficult in human vision. This often leads to minor fabrication anomalies which result in fabrication failure and thus, an etching endpoint algorithm should ensure optimum termination accuracy. In the present work, both near-UV and visible spectrum light are investigated to illuminate the “LiNbO3-Al” and the “Si-SU-8-Al-ZnO-Al” substrate. During the etching process, images of “LiNbO3-Al” are taken during the etching process to quantify the etching progress whereas the “Si-SU-8-Al-ZnO-Al” substrate is analysed stationarily due to on-campus lab access limitation. Image processing techniques and pixel-based analysis is performed to extract pixel changes and with that, an inflection point in the derivative datapoints is used to compute the endpoint. A prototype is also developed to automate the post-development procedures to eliminate possible interactions between near-UV and humans. The feature dimensions measured using optical microscope and the frequency response of the device confirm the successful fabrication using the proposed prototype. No short is detected from the fabricated devices and with the accuracy of $\pm 0.86\ \mu m$ features sizes, the prototype is proven to be successful. Moreover, observable differences in the near-UV imaging chamber for transparent multi-layered structures (Si-SU-8-Al-ZnO-Al) is presented in the result section, proving the potential feasibility of near-UV spectrum in the prototype. Future research on the topics includes the near-UV exposure implications on hard bake photoresists and more variety of fabrication conditions should be carried out to verify the robustness of the prototype.

# Acknowledgement

Throughout the journey of the final year project (FYP), I have received help and support from several parties.

First, I would like to thank my supervisor of this project, Dr. Ajay Achath Mohanan for his continuous guidance and detailed explanation throughout the project. He is always willing to explain the working principles of MEMS technologies to the fundamental level to give me a clear picture on what I am working on. Despite his busy schedule, there will always be enough time for discussion, and he has never failed to ensure I truly understand the concept before ending the meeting.

Other than my supervisor, I would like to thank 2 of my close friends, Yip Wai Pong and Jolene Ong Su Lynn for their unconditional support throughout the project. As I am fairly new to 3D CADding software, I have received a tremendous amount of help and guidance in the 3D prototyping stage from Yip Wai Pong to realise the completion of my final prototype. Also, I am very grateful to Jolene as she proposed to provide me 3D printing services without any further questions and conditions. They are the prime reasons that my physical prototyping project is feasible in this pandemic period where on-campus lab access is limited.

Lastly, I am also thankful to both Howgen Pratam Kesuma and Amardeep Singh Dhillon, who are both current PhD students that are involved in the MEMS research field. I would like to express my gratitude to them for agreeing to supervise me whenever I needed to collect some data from the actual fabrication process. They are willing to provide me with lab assistance despite their busy and hectic schedules.

Without any one of them, the completion of this project would not be successful, and it would never be possible to achieve this level of execution.

# Table of Contents

## Contents

# List of Figures

# List of Tables

# Nomenclature

| | |
|---|---|
| *SF* | Savitzky-Golay filter |
| *SAW* | *Surface acoustic wave* |
| *DIPE* | *Derivative & inflection point-based endpoint* |
| *SWE* | *Sliding window endpoint* |
| *Al* | *Aluminium* |
| *ZnO* | *Zinc oxide* |
| *HMI* | *Human machine interface* |

# Chapter 1

# 1. Introduction

## 1.1 Background and significance

SAW devices are well-established technologies in various industries due to its compactness, low fabrication cost etc. Lately, many interests in retaining the advantages of SAW devices and yet introducing flexibility have emerged in the research field [3-6]. This form factor allows SAW device integration into consumer wearables. To obtain the desired IDTs, fabrication processes including photolithography and etching procedure are the conventional techniques.

Recently, the "Si-SU-8-Al-ZnO-Al" flexible device is investigated heavily in the lab and the etching endpoint determination has always been difficult due to the ZnO's high transparency in human vision [2, 7] and the double Al layers. The huge difference of ZnO and Al optical properties in the near-UV range [7, 8] has inspired the idea of replacing human vision with near-UV imaging. Hence, this project aims to prototype a fully automated wet-etch system with near-UV/visible light spectrum-based endpoint detection system. The significance of this project is to eliminate human participation in UV-exposed fabrication environments due to the health considerations [9, 10]. In fact, any arbitrary light spectrum can be easily integrated, and the automation will be a convenience in fabricating devices in the future.

## 1.2 Literature Review

### 1.2.1 Surface Acoustic Wave (SAW) Devices

SAW devices could easily be found in various industries ranging from conventional electronic components manufacturing industry to highly specific biochemistry industry. Depending on the pattern of interdigital transducers (IDTs) on the SAW devices, it can be classified into 2 major types [11], which are the SAW resonator setup or the SAW delay line setup. In the case of SAW resonators, the IDTs can be found in between a pair of resonators to create a resonant cavity for the surface wave. Whereas, for SAW delay line setup, reflectors will not be used.

In the medical industry, due to the miniature size of the SAW devices, they have been implemented as implantable wireless SAW sensors for pulmonary artery pressure (PAP) monitors [12]. The echo signal from the sensors proves to have a 200 Hz/mmHg sensitivity in measuring PAP without any signal processing. In the telecommunication industries, SAW devices are well known components for functions such as bandpass filters[13], delay lines in circuitry as well as many other building blocks in the telecommunications industry.

## 1.2.2 Flexible SAW devices

Lately, instead of fabricating SAW devices on rigid substrates [14, 15], fabrication of SAW devices on flexible substrates has been researched due to its high flexibility. This characteristic is highly desired in consumer wearables where the device undergoes bending and slight deformation in daily use. In previous work, flexible SAW devices have been fabricated using several methods such as (1) polymer-based [3] , (2) metal foil-based [6, 16], Ultra-thin-chip (UTC) silicon [4] and flexible glass-based substrate [5]. The fabricated samples of each method are tabulated in Table 1. In fact, lately, the multi-layered sandwich "Al/ZnO/Al/SU-8/ultra-thin Si/SU-8" flexible SAW devices have been investigated heavily in the Micro & nano device lab. The reason for the materials choices and the design structures are summarized in Table 2. It is worth mentioning that the "Al/ZnO/Al/SU-8/ultra-thin Si/SU-8" device consists of a transparent ZnO layer sandwiched in between the double Al layers.

**Table 1: Figures of fabricated samples from each literature.**

| Author | Fabrication methods | Figures list |
|---|---|---|
| **Jin, et al. [3]** | Polymer-based | Figure 1 |
| **Tao, et al. [6], [16]** | Metal foil-based | Figure 2 |
| **Zhang, et al. [4]** | UTC silicon and polymer sandwich structures | Figure 3 |
| **Chen, et al. [5]** | UTC flexible glass-based | Figure 6 |

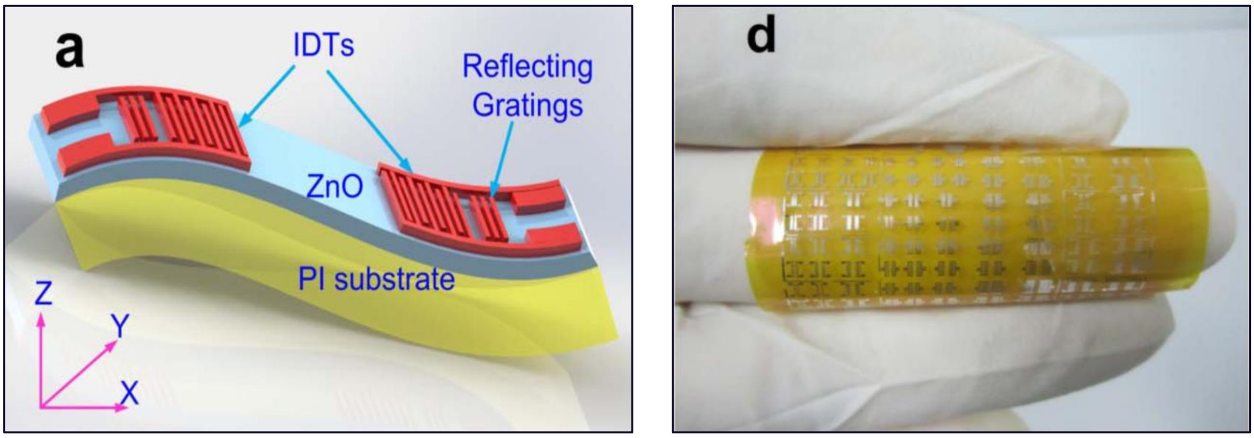



**Figure 1: The schematic of semi-transparent and flexible SAW devices on the ZnO/polymer substrate (a) and the actual product (b) [3].**

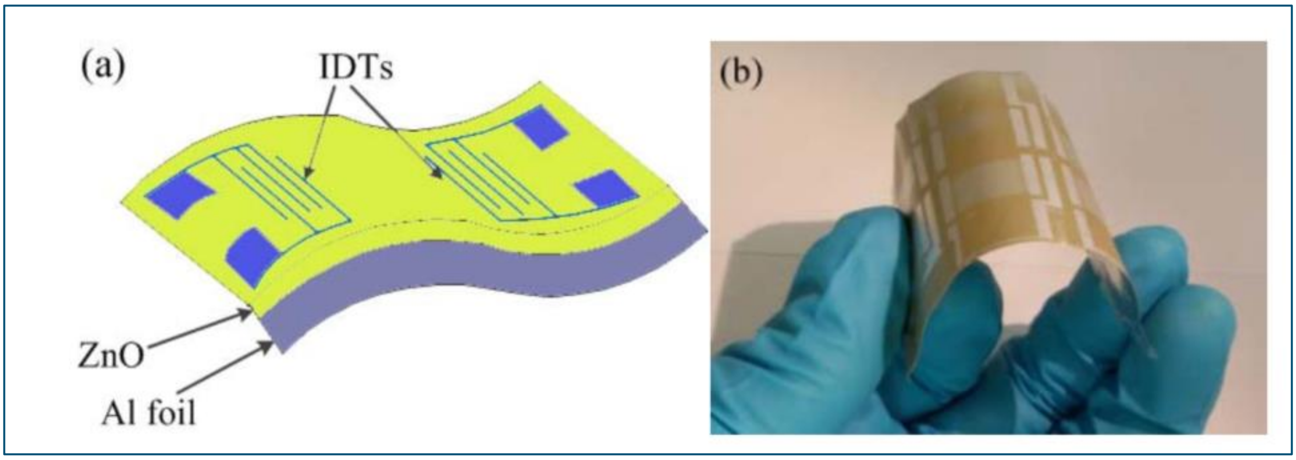



**Figure 2: a) Schematic illustration of the designed ZnO/Al SAW device. b) The actual fabricated ZnO/Al SAW device [6, 16].**

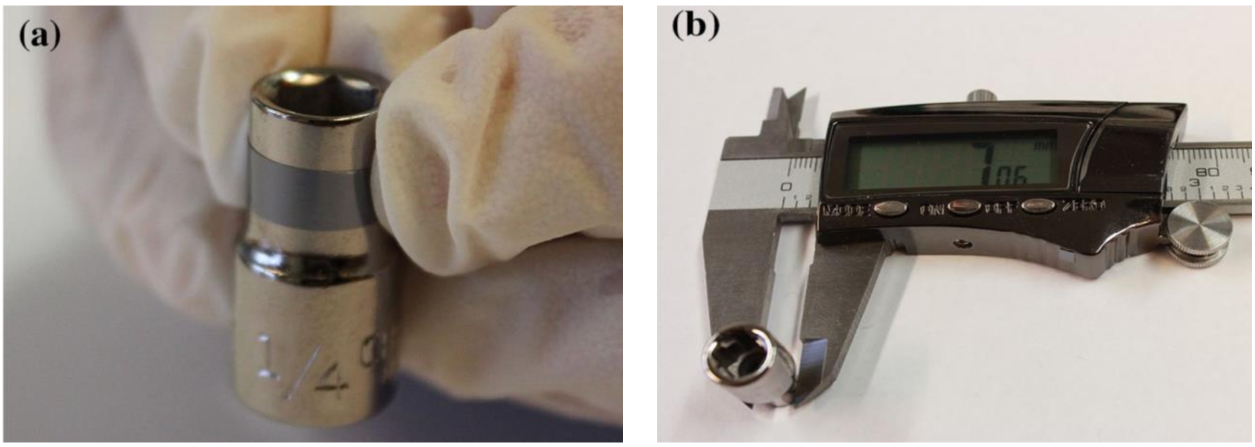


**Figure 3: Sandwiched silicon strip winded round cylinder (a) and the diameter of the cylinder (b) [4].**

**Table 2: The design aspects and considerations of " Al/ZnO/Al/SU-8/ultra-thin Si/SU-8"**

| Design aspects | Description | Purposes |
| --- | --- | --- |
| **SU-8** | Polymer | • High flexibility and mechanical stability of polymer provides excellent bendability to the device [3].<br>• Polymer dissipates acoustic wave energies into the substrate which often leads to signal attenuation [6]. However, this is controlled by optimizing the thickness. |
| **Al** | metal & acoustically hard substrate | • Minimizes the mismatch between crystal structures and the difference of thermal expansion coefficients of SU-8 and ZnO [6].<br>• Mitigates the build-up of film stress and promotes adhesion of ZnO crystals [6]. |
| **UTC silicon** | Thin substrate | • Maximum beam deflection is inversely proportional to the cube of the chip thickness [4].<br>• Initially brittle silicon is thinned down using back-side-etching to provide slight bendability [4]. |
| **SU-8 sandwiched UTC** | Polymer sandwich-like structure | • The brittle UTC silicon is placed in the "neutral stress and strain plane" [4].<br>• Minimal stress or strain is experienced by the UTC during bending motion, hence the flexibility is improved. |

However, due to the optical properties of certain materials such as ZnO and ITO, the fabricated devices can be invisible when seen with bare eyes as shown in Figure 4 to Figure 6. Jiang, et al. [1] utilizes the transparent property of ZnO thin film in fabricating transparent capacitive-type fingerprint sensor using ZnO thin-film transistors. This invention is aimed to provide an alternative of on-display transparent electronics to replace the conventional non-transparent silicon-based fingerprint sensors that limits the screen-to-body ratio of mobile devices. Furthermore, in [2], the transparent properties of indium tin oxide (ITO) and ZnO with the combination of transparent Ag nanowire electrode have also been utilised to form a multiaxial and transparent strain sensors (96% transparency at wavelength of 550 nm) for motion and emotion monitoring.

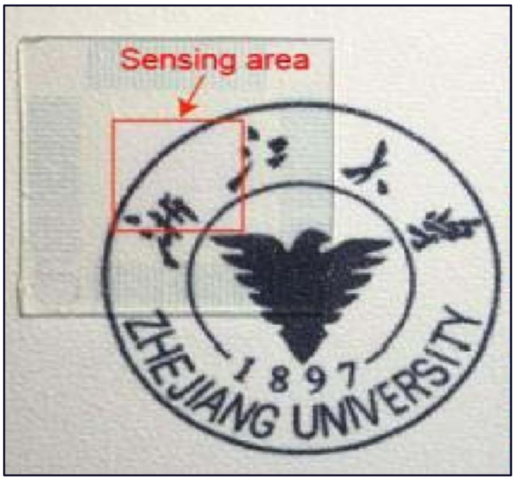



**Figure 5: Photo of the transparent fingerprint sensing panel [1].**

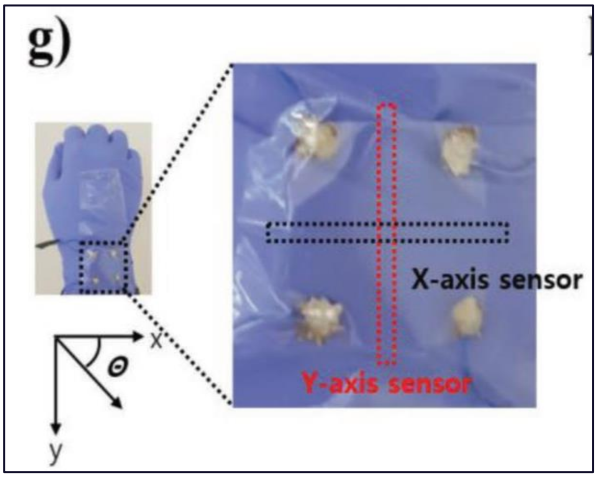



**Figure 4: Images of all-transparent multiaxial wearable strain sensors attached to a human wrist [2].**

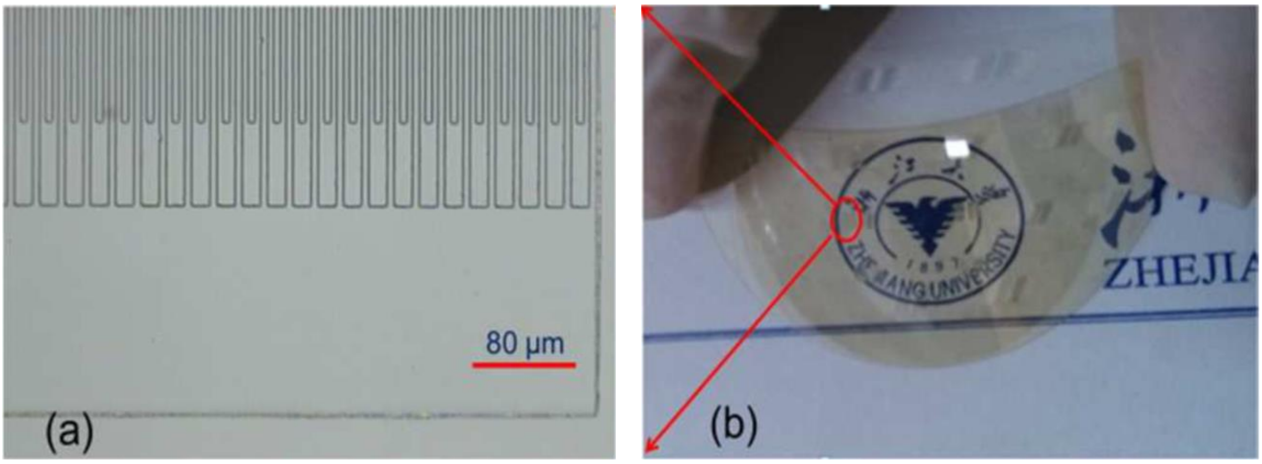



**Figure 6: A microscope photograph of ITO IDTs (a) and photograph of the ITO/ZnO on the flexible glass wafer with fabricated SAW devices (b) [5].**

### 1.2.3 Optical properties of material and the difficulties in fabrication

However, as shown in Figure 7, due to the high transparency (close to 100% transmittance) of the sensing material layer (e.g., pure ZnO) in the visible light range, it has caused difficulties in ensuring the shaping of the transparent thin film structure is in the correct dimension during fabrication.

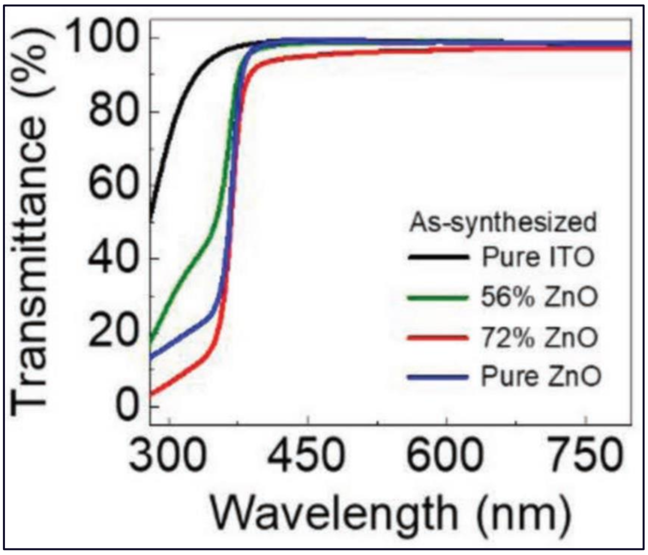



**Figure 7: UV-vis transmittance of ITO and ZnO thin films [2].**

In [2, 5, 6], lift-off is used to fabricate the transparent sensing layer to tackle this problem. However, lift-off process does possess some potential shortcomings as follows:

**Table 3: Potential shortcomings of lift-off process.**

| Process | Potential shortcomings |
|---|---|
| **Lift-off** | 1. Unexpected change of properties such as film resistance resulted from long rinsing time in developer (e.g., acetone) and redeposition of residual photoresist onto sensing material during rinsing process [17]. |
| | 2. Uncontrolled random breaking of photoresist as shown in Figure 8, causing low patterning quality in small features region [18]. |

| | |
|---|---|
| | 3. Thermal softening (rounding) of resist film during metal coating process (e.g., evaporation, sputtering) as shown in Figure 9, causing the resist features to be fully covered by deposited metal. Lift-off process becomes worsen or impossible to be done afterwards [19]. |

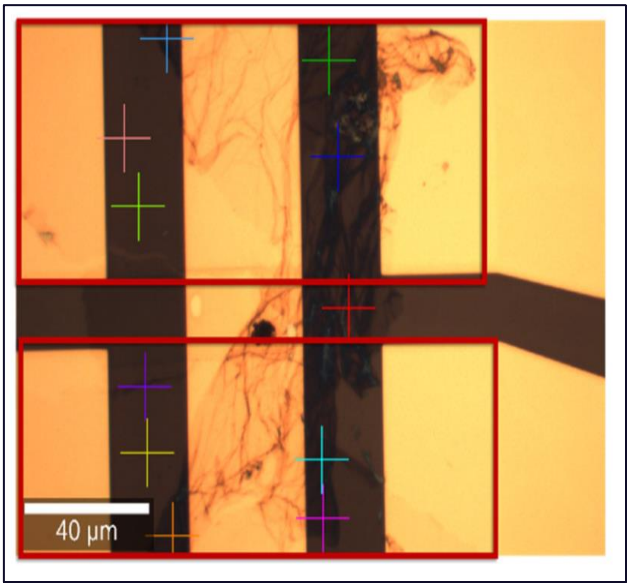



**Figure 8: Microscope images of pre-patterned gold contacts samples after lift-off graphene patterning. The size and density of the features influences the efficiency in controlling the breaking step [18].**

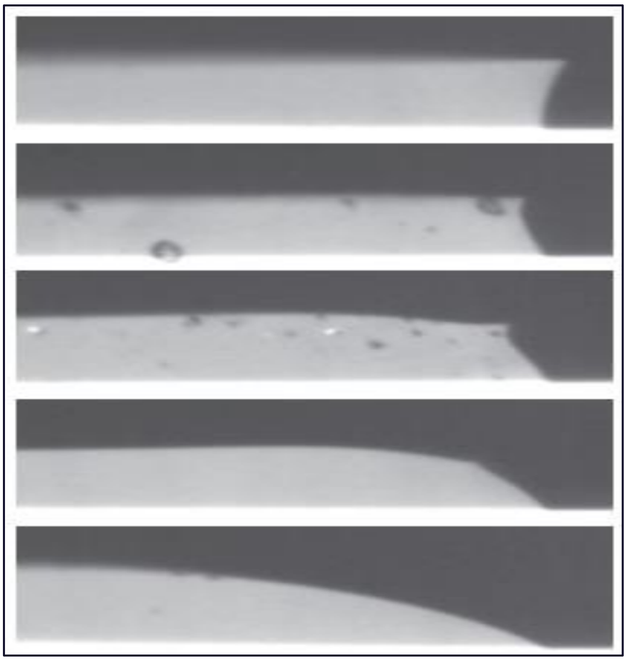


**Figure 9: Thermal softening of resist structure as the temperature rises above the softening point of resist (from top to bottom) [20].**

Due to the aforementioned factors, wet etching which photoresist processing is performed on a previously applied coating does not suffer from the issues, hence it is still a reasonable alternative of electrode patterning. However, due to the high transparency of these sensing materials, the etching endpoint of such material through human observation will be difficult. This will be further discussed in the later section.

### 1.2.4 Overview of existing optical etching endpoint detection technologies

During the etching process, it is vital to determine the stages and be able to identify the etching endpoint. Implications of failure to do so are summarized in Table 4. These scenarios are visualised in Figure 10.

**Table 4: Consequences of etching-caused anomalies.**

| Actions | Consequences |
|---|---|
| **Stopping etching too early** | Leads to under etching and short-circuits formation in the device |
| **Stopping etching too late** | Leads to over etching and over erosion of features. Fabricated features might be thinner than the designed measurement. |

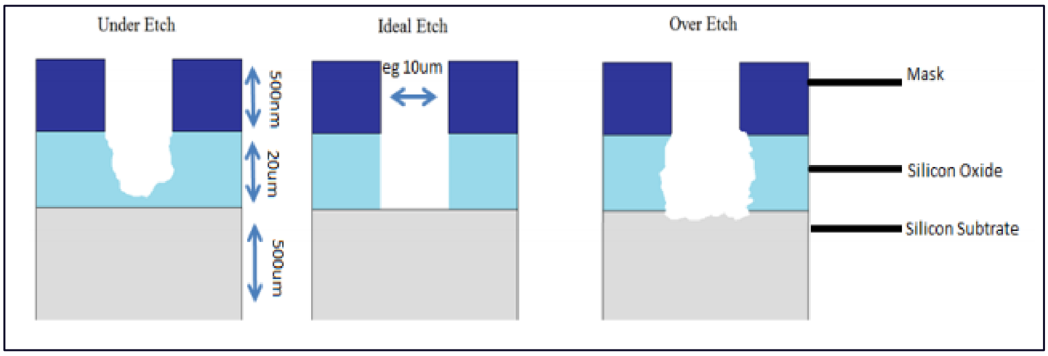



**Figure 10: Showing different etching stages. The materials used in this image are purely illustrative and can be replaced with respective substrate and patterned material.**

In the case of designing SAW devices, metallization ratios of the features are normally set to ½ for the maximum wave coupling strength. Any defect during the fabrication process will cause the actual metallization ratio to be away from this designated ratio. This will cause a significant impact on the device behaviour such as the IDT static capacitance and the wave coupling strength as the actual features developed on the device are not matching the designed ratio [21].

Existing technologies of optical wet etching endpoint detection will be discussed. There are mainly 3 working principles for in-situ etching endpoint detection which are based on (1) reflectance intensity measurement, (2) optical reflectance spectroscopy and (3) laser interferometry.

In [22, 23], laser interferometry has been applied by using a coherent source laser beam pointing towards the etching material surface. Looking at Figure 11, at the surface boundary, a portion of light will be reflected whereas a portion will be refracted into the second layer and be reflected at the following boundaries. With the configuration shown in Figure 12, the interference between these 2 sources of reflections will be recorded by the recorder and will result in either constructive or destructive interference depending on the changes of the first layer thickness. The real-time thickness changes resulting from etching can be calculated with the period between the successive maxima and minima from the intensity recorded as shown in Figure 13. Figure 14 shows the endpoint can also be the point when a flat signal is observed as the etch targeted material is fully etched out, exposing the cladding layer underneath. However, this technique has its limitations as tabulated in Table 5.

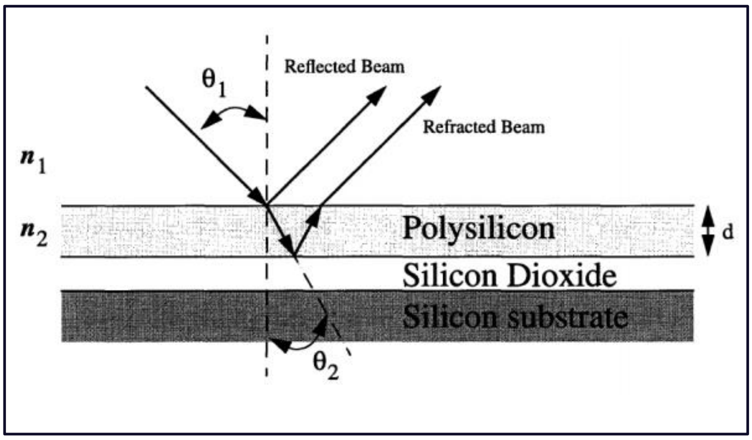



**Figure 11: Showing the path traveled by light (reflected & transmitted)[22].**

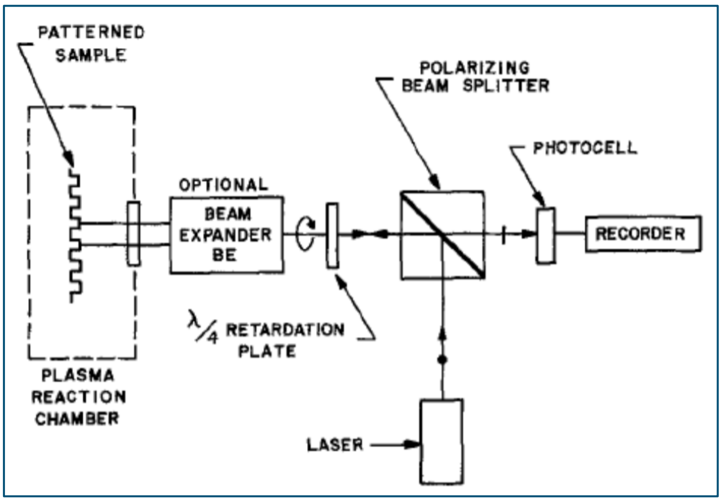



**Figure 12: Setup of single beam interferometer [22].**

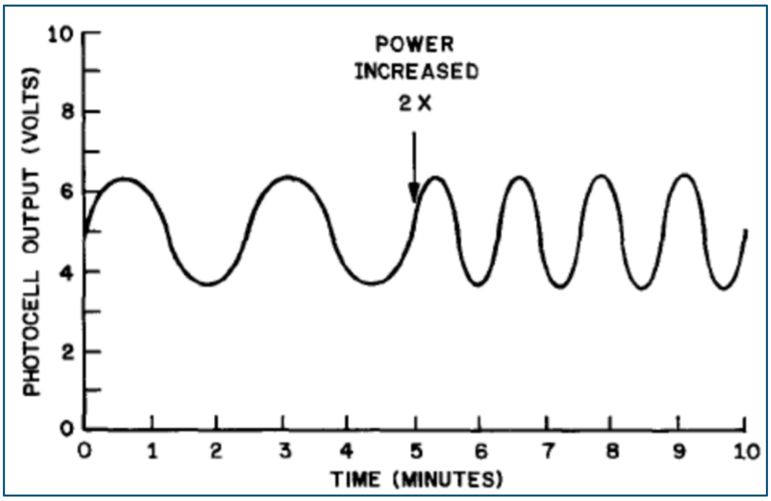



**Figure 13: Response from photocell as first layer becomes thinner during etching [22].**

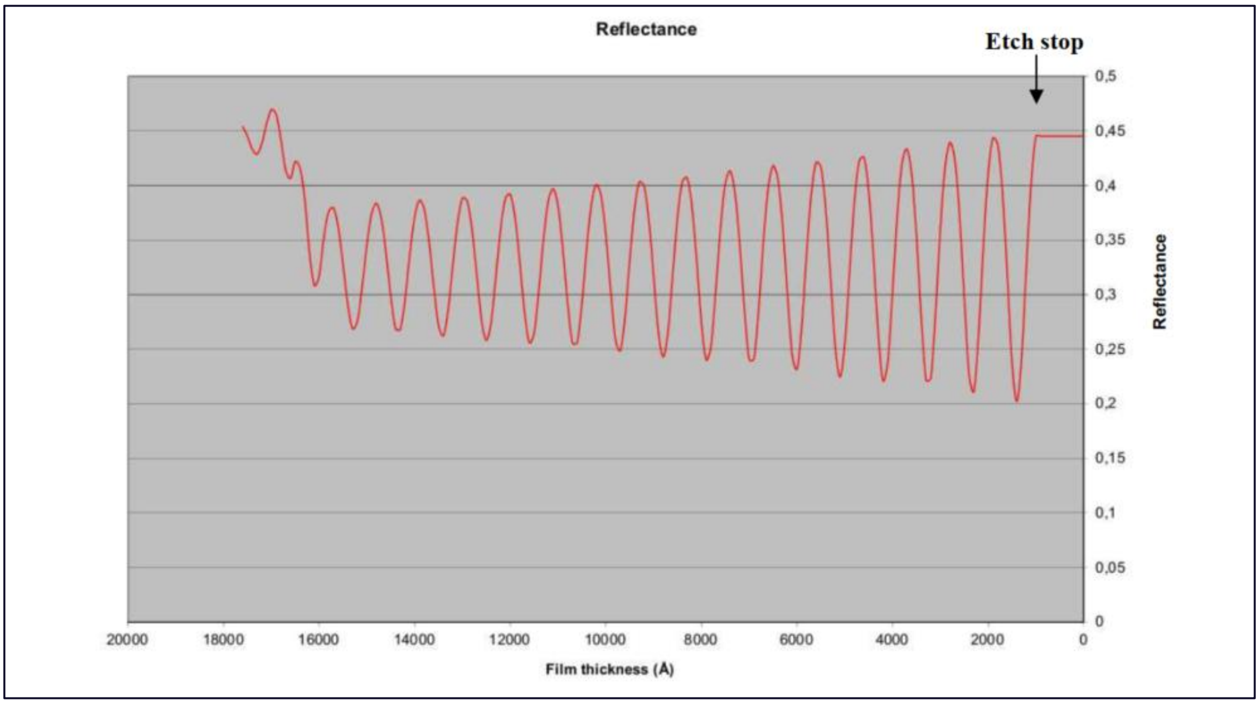



**Figure 14: Simulation of laser reflectance during etching process. Flat signal is observed as the GaSb layer is exposed after the cladding layer is fully etched [23].**

**Table 5: Laser interferometry's limitations**

| Technologies | Limitations |
|---|---|
| **laser interferometry** | 1. Not applicable for non-transparent etch target material as no light is able to pass through. |
| | 2. Only provide localised point monitoring. Endpoint determination will be greatly affected by uneven etch condition across the surface. |
| | 3. Require complex algorithms for multi-layer device etching process monitoring as the interference will be contributed by more than 1 boundary. |
| | 4. Less applicable for wet etching due to the presence of interference from the liquid etchant. |

Contrarily, with the setup of Figure 15, Furuya, et al. [24] discussed the use of optical reflectance spectroscopy in the thinning process of hydrogenated amorphous silicon (a-Si:H) film through etching. The etch rate is determined by fitting the measured reflection spectrum from the polychromator to the simulated spectrum of the material obtained from optical thin film theory as shown in Figure 16. However, accurate refractive index data of the material to be etched is needed to synthesize the simulated reflection spectrum as reference. However, this technique has limitations as stated in Table 6.

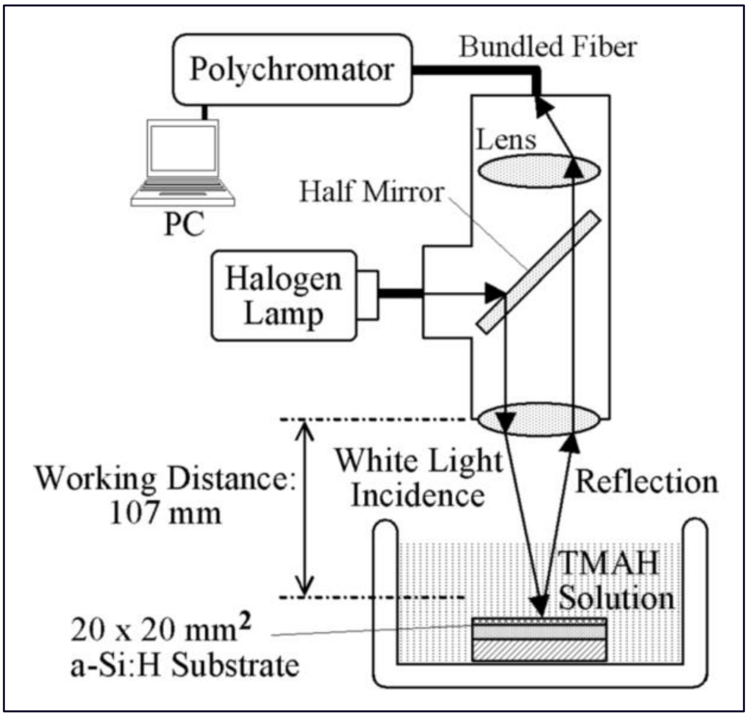



**Figure 15: Schematic diagram of the wet-etching apparatus [24].**

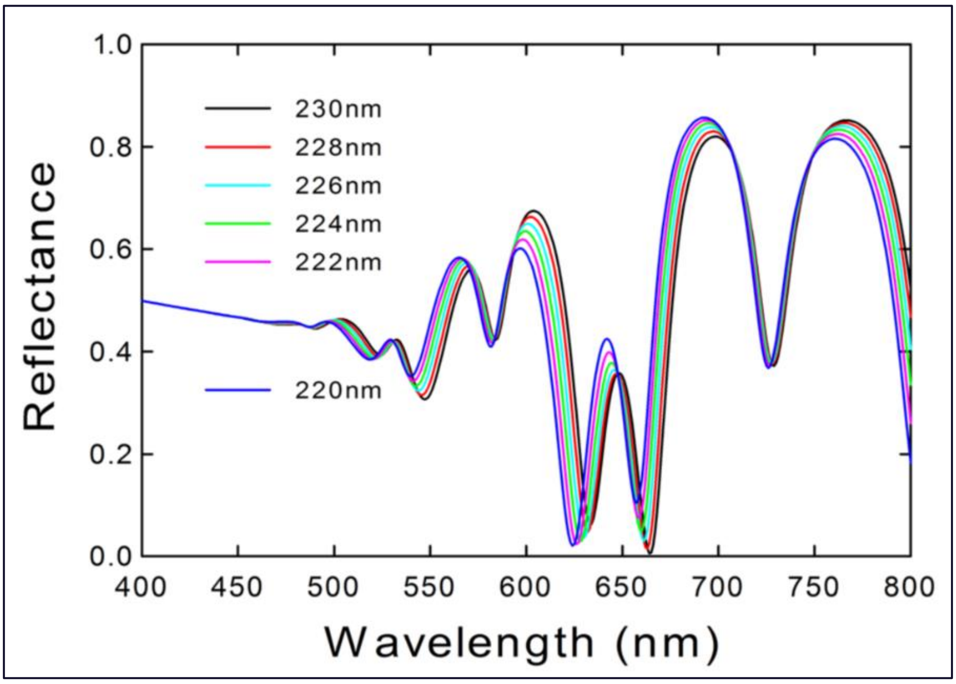



**Figure 16: Simulation of the reflection spectra as the thickness of the a-Si:H film varies between 230 and 220 nm. This will be fitted against the measured reflectance for thickness measurement [24].**

**Table 6: Optical reflectance spectroscopy's limitations**

| Technologies | Limitations |
|---|---|
| **Optical reflectance spectroscopy** | 1. Requires exact refractive index of the material to simulate the theoretical reflection spectrum for each thickness as reference. |
| | 2. Long computational time (~2s) for each iteration as the data of the whole spectrum needs to be collected and processed each cycle. This might affect the endpoint determination accuracy if the etching process is fast. |
| | 3. Requires specialised and expensive optics. |
| | 4. Complex software algorithm needed to compare the measured reflectance spectrum with the simulated one for thickness measurement. |

Furthermore, [25, 26] demonstrated the reflectance intensity measurement-based endpoint determination. Golzarian [25] explains that the reflected light intensity from the device surface will vary throughout the etching process due to the difference between the top layer and the exposed layer in several factors, (1) roughness, (2) reflectivity, and (3) refractive index. As shown in Figure 17 to Figure 19, both the setup from [25] and [26] have a similar overall structure. The main differences are tabulated in Table 7.

**Table 7: Comparison table for [25] and [26] setup.**

| | **Golzarian [25]** (Figure 17 & Figure 18) | **Suhard, et al. [26]** (Figure 19) |
|---|---|---|
| **Light sources** | Tungsten or mercury lamp with multiple emission peaks over the wavelength. | White light |
| **Optical measurement tools** | Monochrometer and photomultiplier tube. | Charge-coupled device (CCD) camera |

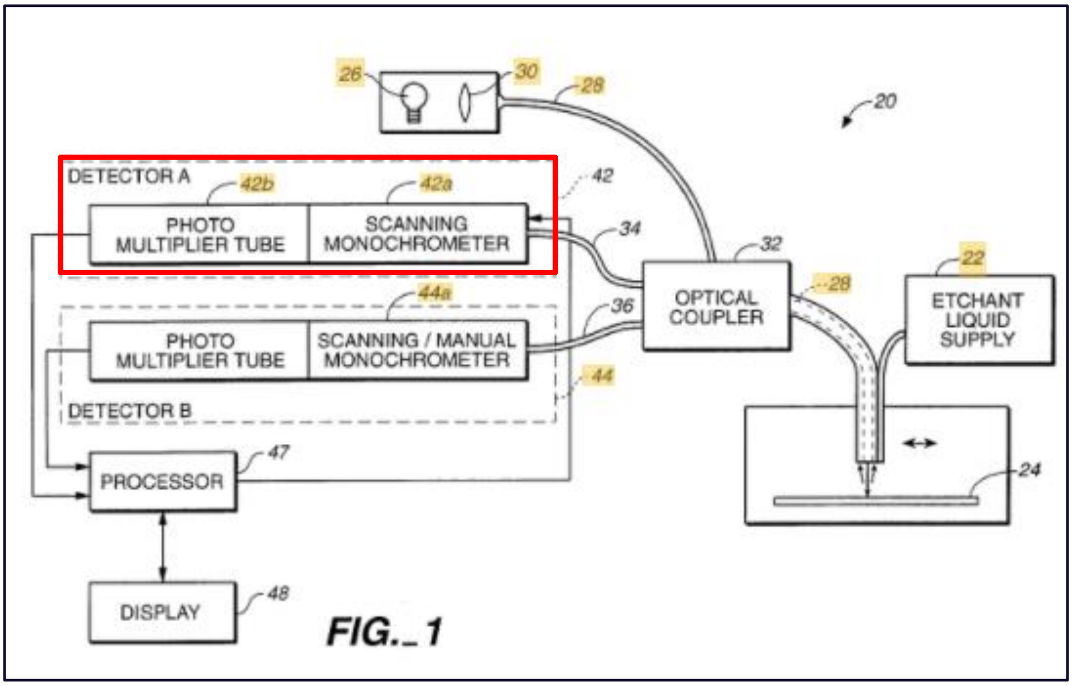



**Figure 17: Detector (red) comprises photomultiplier tube and monochromator. Output is fed into the processor to be further analyzed [25].**

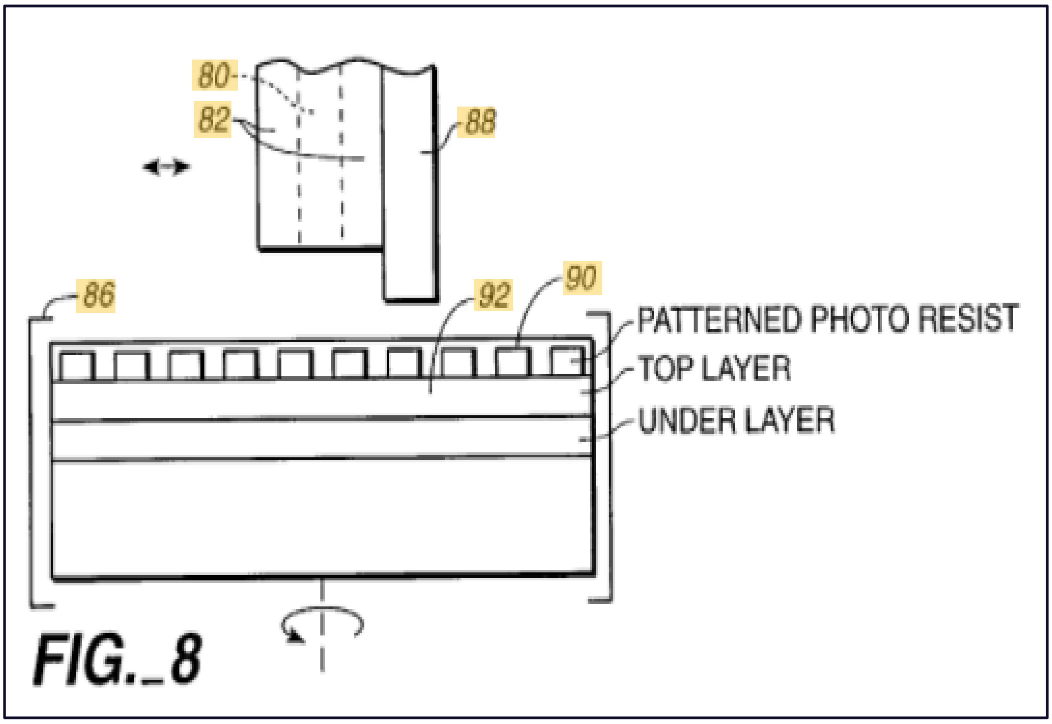



**Figure 18: The probe tip consists of directed light sources and light receiver from optical fibers pointed towards the device [25].**

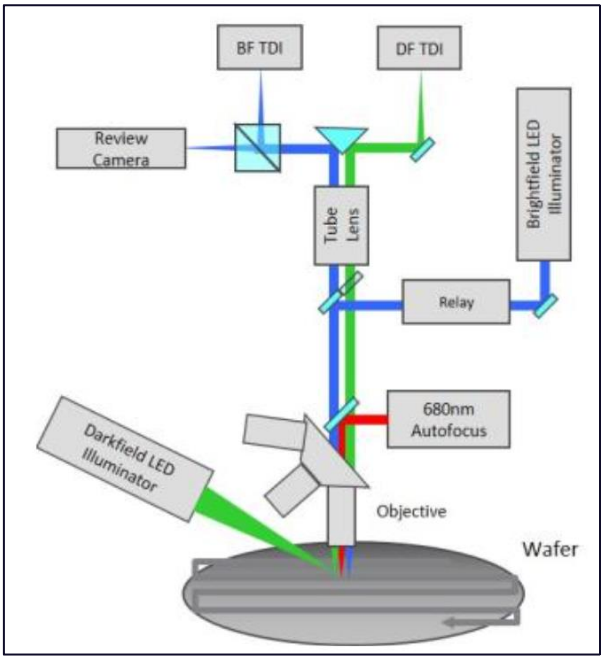



**Figure 19: An overview of the etching monitoring imaging chamber [26].**

Looking at Figure 20 and Figure 21, the reflected light intensity trend during the etching process from both authors is similar with the only difference of the trend direction. This is due to the difference of the under and top layer material used for the fabrication in the individual studies. Likewise, there is an obvious slop change detected from the light intensity measurement during the etching process from both results. The author in [27] also demonstrated the trend of the light intensity graph can be differentiated to obtain the inflection point (marked as “41” in Figure 22) which correspond to the highest etch rate point to compute the endpoint.

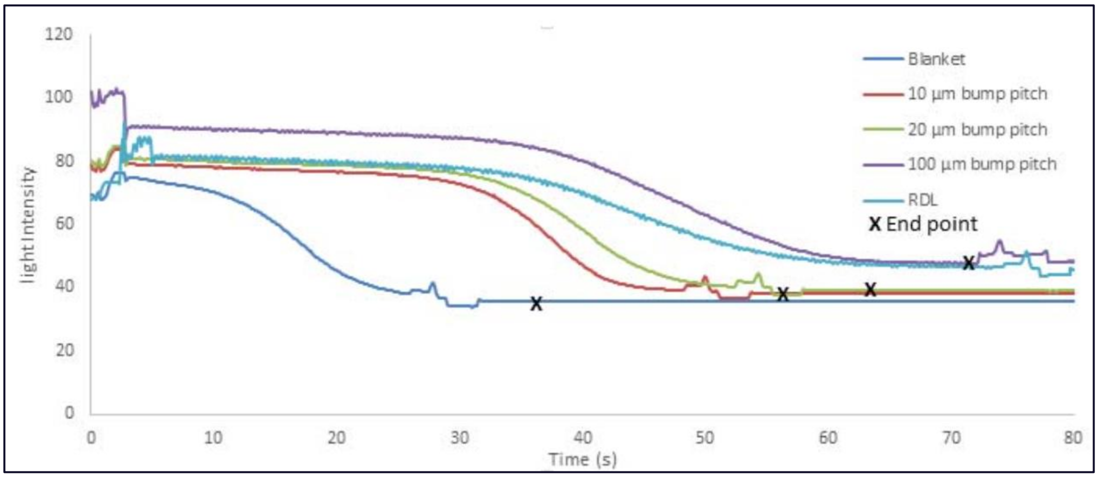



**Figure 20: The light intensity against time graph of different fabricated structures with endpoints marked [26].**

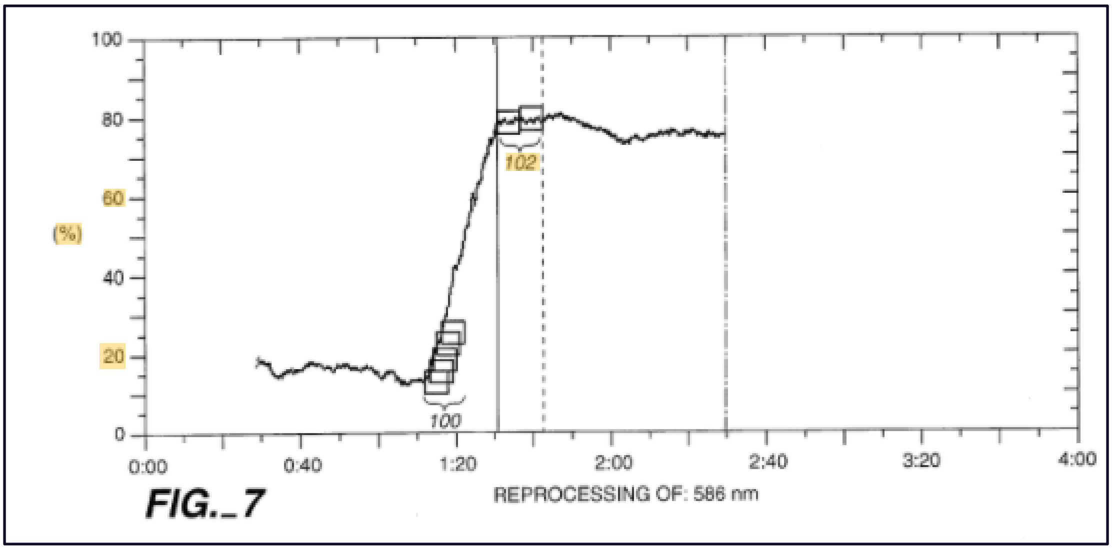



**Figure 21: Plot of intensity of reflected light over time [25].**

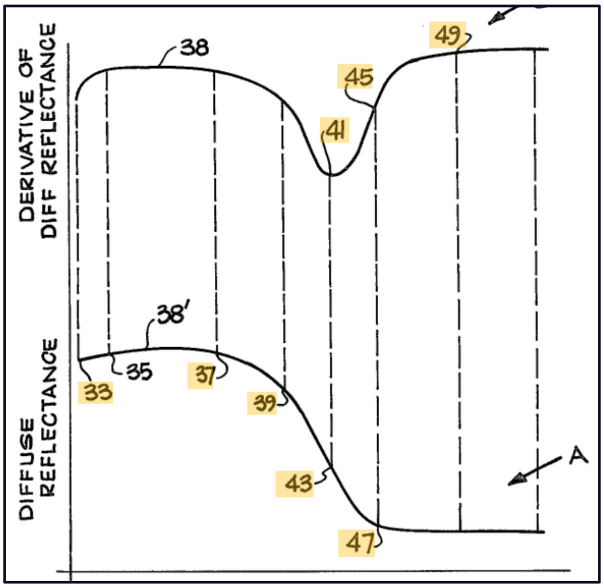



**Figure 22: The theoretical reflectance intensity graph (bottom) and the derivative (top) [27].**

### 1.2.5 Summary of reviewed optical etching endpoint technologies

**Table 8: Summary of the existing optical etching endpoint determination technologies**

| **Author** | | **Golzarian [25]** | **Suhard, et al. [26]** | **Pugh [22]** | **Furuya, et al. [24]** | **Tran, et al. [23]** |
|---|---|---|---|---|---|---|
| **Physical sampling method** | Reflectance intensity measurement | ✓ | ✓ | × | × | × |
| | Optical reflectance spectroscopy | × | × | × | ✓ | × |
| | Laser interferometry | × | × | ✓ | × | ✓ |
| **Characteristics** | Thickness | × | × | ✓ | ✓ | ✓ |

| | | | | | | |
|---|---|---|---|---|---|---|
| | Monitor | | | | | |
| | Wide area monitoring | ✓ | ✓ | × | ✓ | × |
| | Algorithm flexibility | ✓ | ✓ | × | × | × |
| **Research gap** | Image processing | × | × | × | × | × |
| | Require special optics | ✓ | ✓ | × | ✓ | ✓ |
| | Ease of handling | × | × | × | × | × |
| **✓ denotes covered aspect** <br> **× denotes uncovered aspect** | | | | | | |

Table 8 summarizes the etching endpoint technologies reviewed. Comparing the research gaps, most of the technologies require special and expensive optical equipment and none of them utilizes digital software or image processing methods to enhance the measurement. Also, etching endpoint for transparent material in the visible light range is rarely discussed.

## 1.3 Principles and theory behind project

The endpoint detection in this project is based on the principle of reflectance intensity measurement. As explained previously, some thin film materials have high transmittance in the visible light range, hence 2 types of light sources which are RGB LEDs, and near-UV LEDs will be experimented as illuminators to obtain changing parameters during etching. Table 9 tabulated the optical properties of ZnO and Al in the near-UV range and the potential application. The fabrication device surface is expected to change resulted from the etching process, thus changes in the reflectance intensity captured by the conventional CCD camera will be observed. In this work, image processing to minimize measurement noise from the conventional camera will be investigated and the pixel components (R, G, B) from the image will be recorded as a measure of the reflectance intensity.

**Table 9: ZnO and Al's near-UV range optical properties.**

| Material | Optical properties in near-UV range | Potential application |
|---|---|---|
| **ZnO** | As shown in Figure 23,<br>• High absorbance [28]<br>• Low transmittance [28] | Huge difference in measured reflectivity intensity under near-UV spectrum when the ZnO layer is exposed after the top Al layer is fully etched. |
| **Al** | As shown in Figure 24,<br>• High reflectance [8] | |

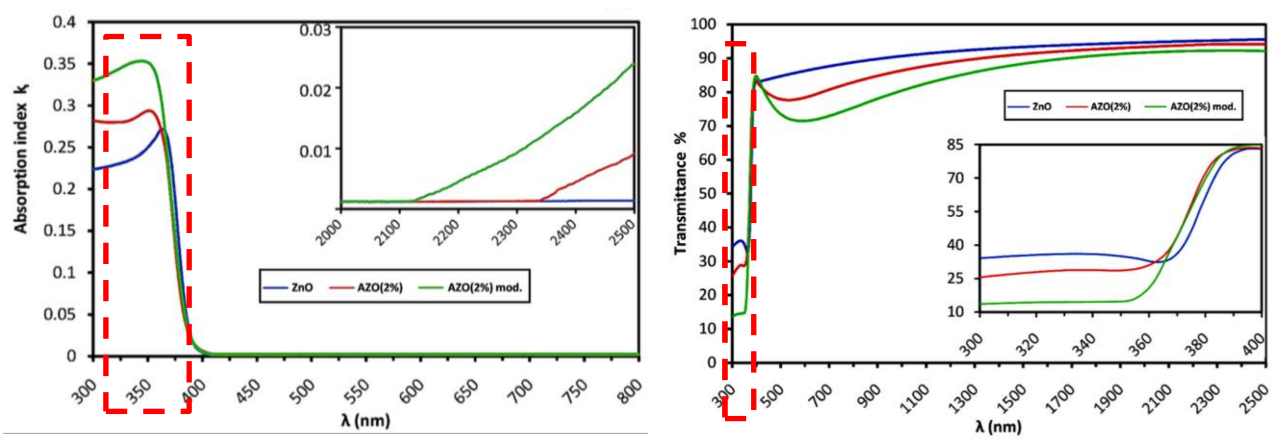



**Figure 23: The absorption index (left) and transmittance index (right) of ZnO (blue)[28].**

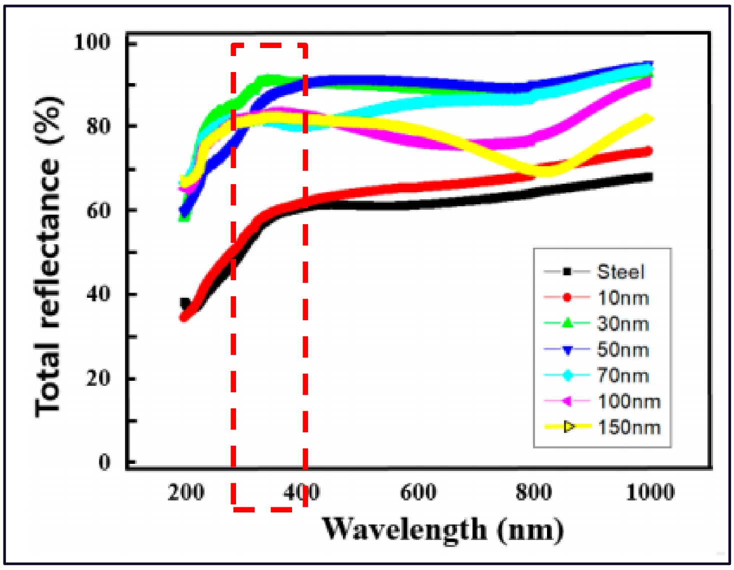



**Figure 24:Total reflection of Al film of different thickness across different wavelengths [8].**

## 1.4 Problem Statement

Lately, due to the aforementioned benefits of flexible SAW devices, the "Si-SU-8-Al-ZnO-Al" devices have been investigated in the lab. Due to the high transparency of ZnO layer in the visible light region as shown in Figure 4 to Figure 6, the ZnO layer which is sandwiched between the double Al layers is invisible. Etching process of patterning the Al electrode on such devices requires the user to observe the visual difference on the device to determine the etching endpoint. The transparent properties of ZnO and the double Al layers have made determining the etching endpoint for the top Al electrode to be difficult and often lead to either under etching or over etching. Although in [2, 5, 6], lift-off has been demonstrated to fabricate such devices, shortcomings of lift-off as stated in section 1.2.3 have made the etching process to still be a considerable solution. Interestingly, the difference of ZnO and Al optical properties in the near UV region has inspired the idea of applying near-UV spectrum to monitor the etching process. Considering the harmful effects of UV spectrum on humans such as skin photoaging [9] and skin DNA damage [10], the fabrication environment could be infeasible for human participation. Therefore, this led to the need of automating the etching and post-etching procedures to eliminate the interaction between UV and humans. In fact, the prototype can be easily integrated with arbitrary spectrum of light to monitor the process and yet the automation can still be a convenience for the researchers in fabricating devices in the future. Furthermore, from the reviewed journals, automated etching endpoint technologies typically require expensive investment on special optics equipment and require complex handling techniques. Moreover, minimal to no software enhancement on the reflection intensity measurement has been carried out in the previous work. These stated problems and research gaps will be investigated in the project.

## 1.5 Objectives

**Automated wet-etch system with near-UV/visible light spectrum-based endpoint detection**

1. To construct a visible/ultraviolet light-based imaging chamber for periodically monitoring the measurable changing parameter of the wet etching of aluminium thin film patterns on SAW devices.

2. To develop a software algorithm capable of analysing the trend of the visual changing parameters (e.g., RGB pixels) during the wet etching of aluminium thin film patterns on SAW devices for near-real time endpoint determination.

3. To construct a hardware prototype for automated handling of the SAW device fabrication process from post-photoresist-development stage until completion of device fabrication.

4. To integrate the imaging chamber constructed in (1) with the automated hardware prototype constructed in (2) to develop an automated wet-etch system for fabrication of SAW devices.

## 1.6 Layout of the report

The outline of the report is as follows. Section 2 explains the adopted methodology and methods in this project to achieve the objectives. Section 3 presents the results and provides a conclusion from the project. Lastly, Section 4 points out the limitation of the present work and proposes suggestions for future research.

# Chapter 2

# 2. Methodology and methods

The overall structure of the prototype can be broken down into electronic systems, material and mechanical systems and software systems. Each system is further elaborated in Figure 26 to Figure 28. The requirements and specifications are discussed based on this hierarchical system structure. The high-level operation overview is shown in Figure 29.

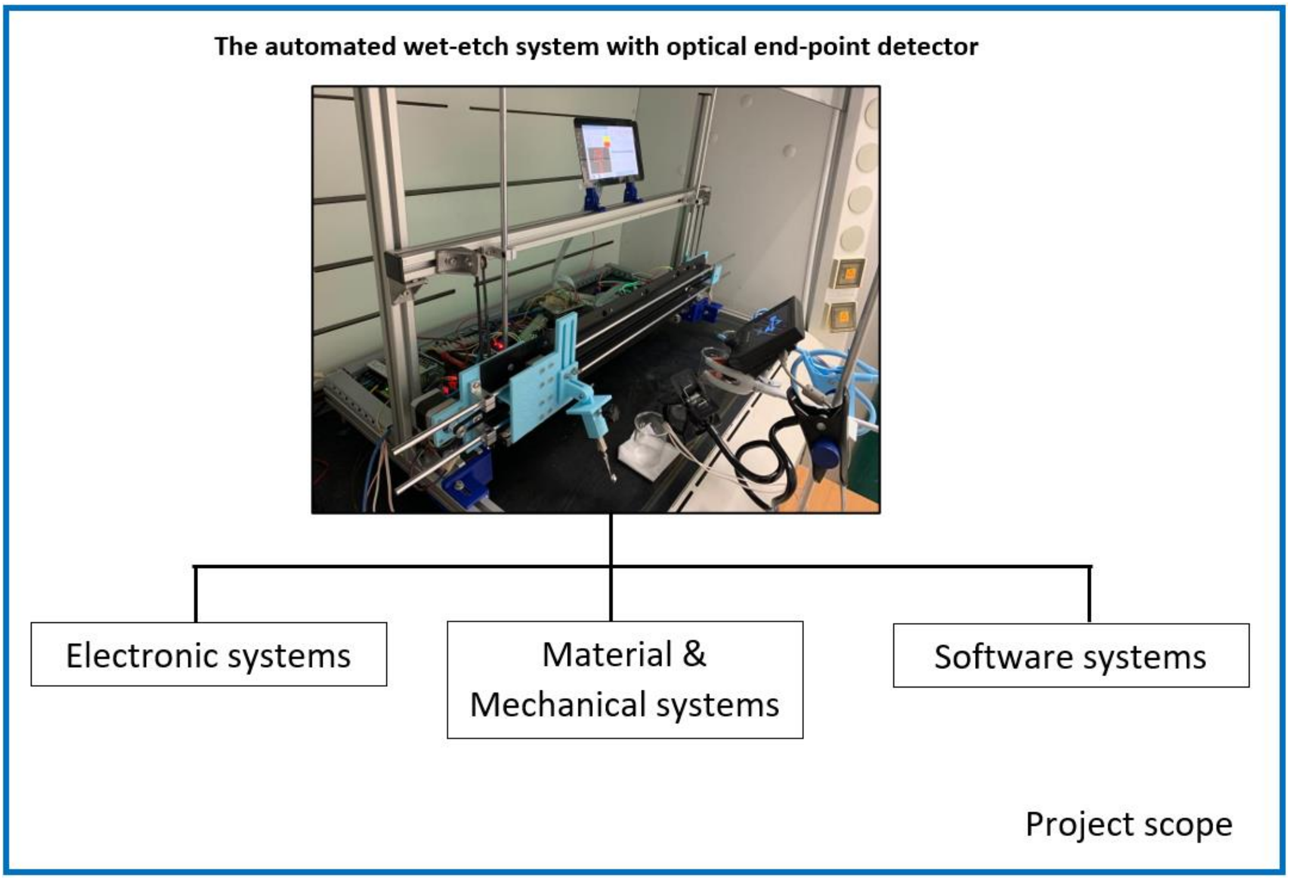



**Figure 25: Overall project's structure**

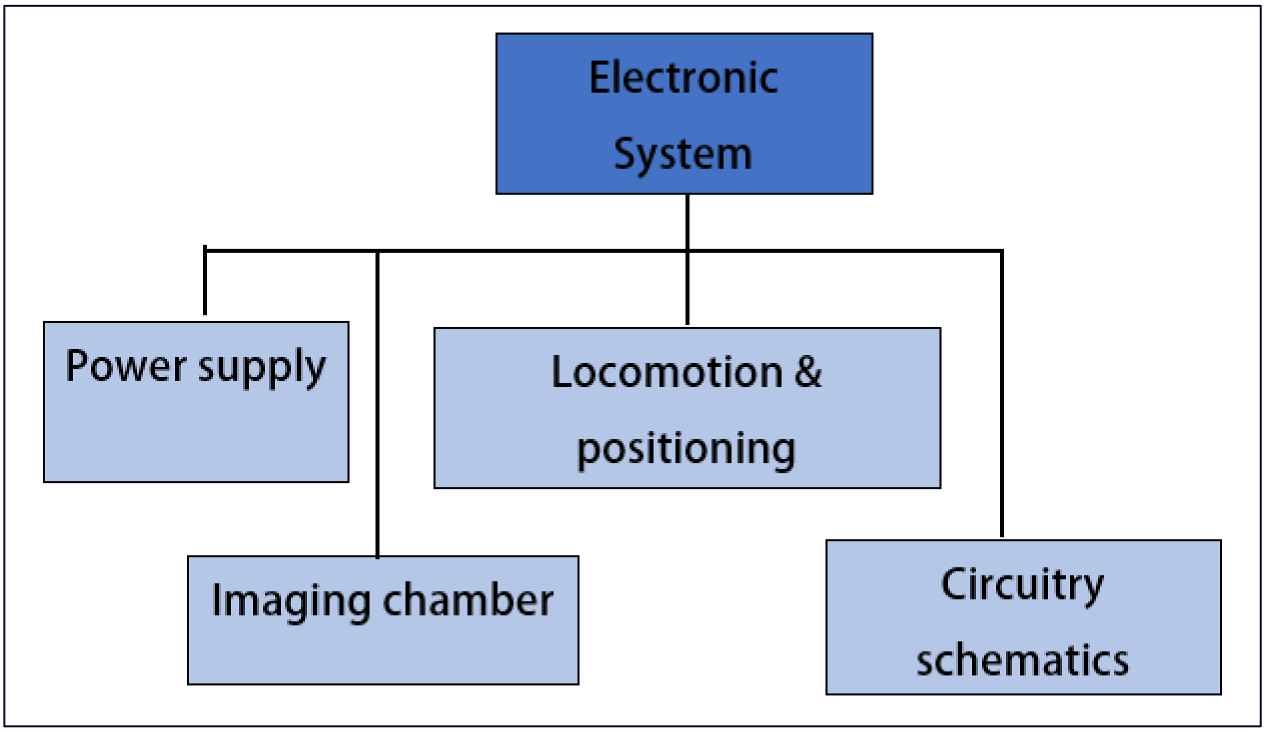



**Figure 26: Subcomponents of electronic system**

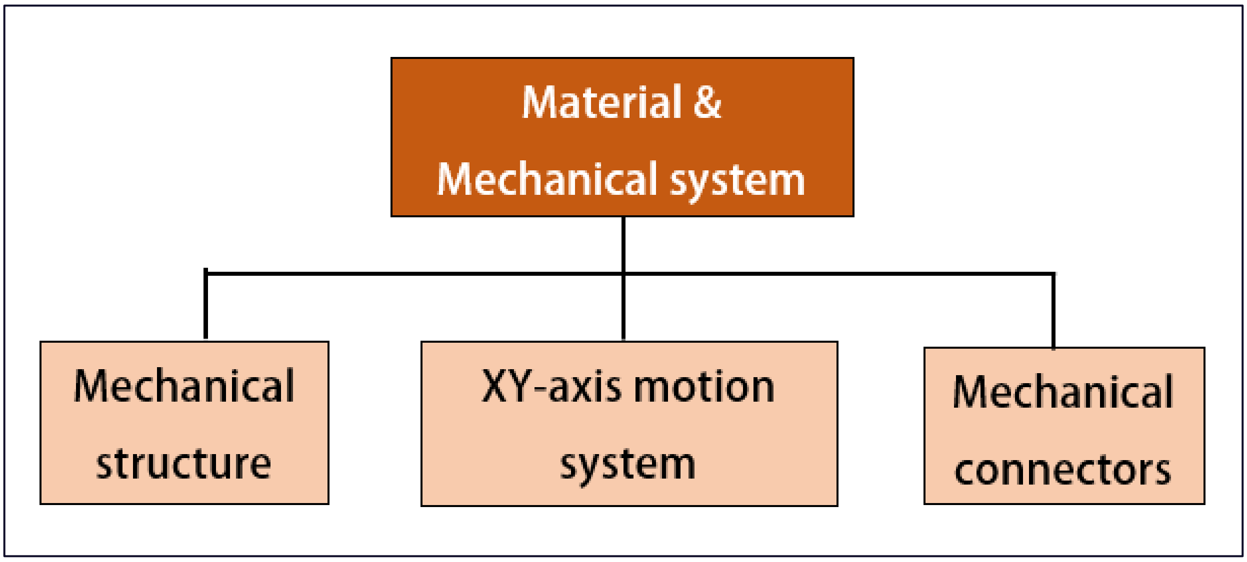



**Figure 27: Subcomponents of material & mechanical system**

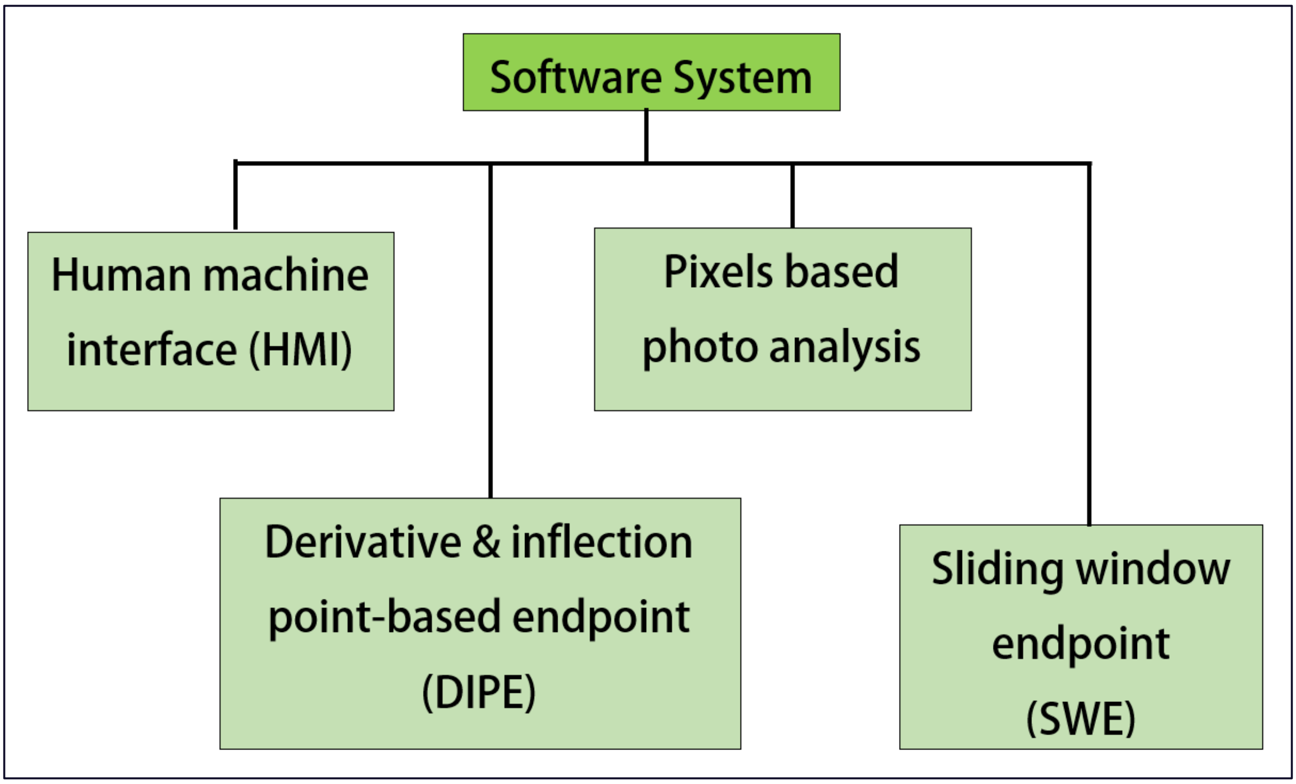



**Figure 28: Subcomponents of software system**

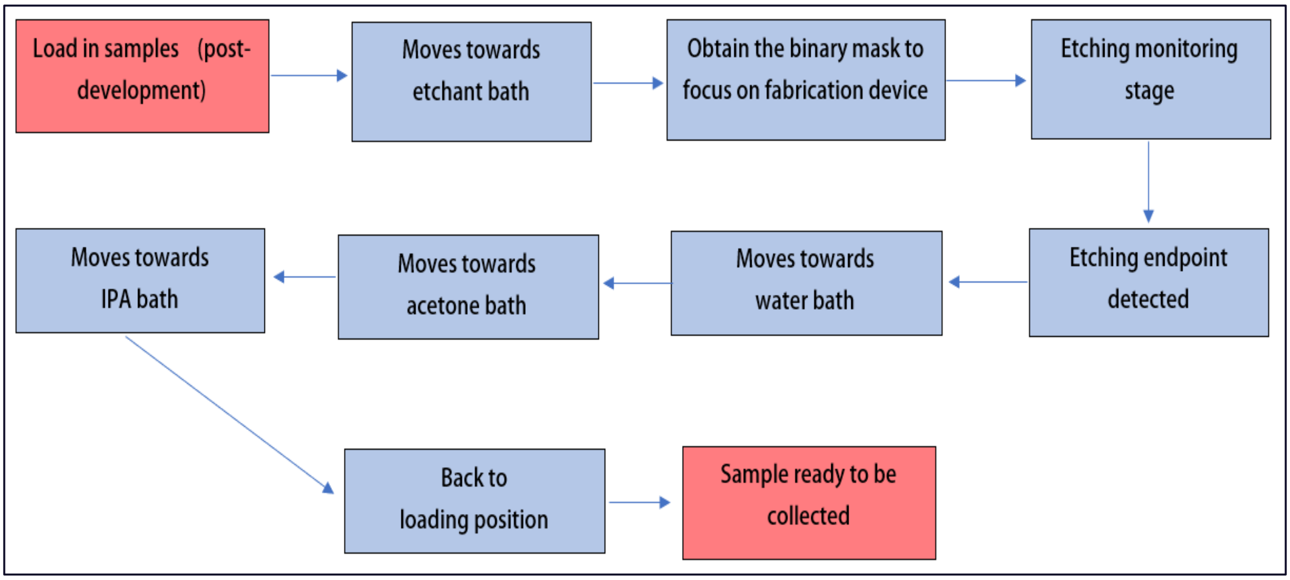



**Figure 29: General stages of the prototype.**

# 2.1 Requirements

**Table 10: Design requirements for the prototype**

| Features | Requirement | Description | Components involved | | |
|---|---|---|---|---|---|
| | | | Mechanical | Electronic | Software |
| Light spectrum flexibility | high | Ability to use different light spectrum to illuminate the substrates to obtain measurable parameters for different material | ✕ | ✓ | ✕ |
| Automation & locomotion | high | Ability to automate the movement between each station for post-development fabrication. | ✓ | ✓ | ✕ |
| Endpoint determination accuracy | high | Ability to provide accurate near real time wet etching endpoint determination that is tolerant to slight changes of etching conditions (e.g., etchant temperature, electrode, and substrate material). | ✕ | ✕ | ✓ |
| Reproducibility | high | Able to produce reproducible endpoint determination with reasonable accuracy. | ✕ | ✓ | ✓ |
| Manual overwriting flexibility | optional | Flexibility of manual operation overwriting mode of the automated wet-etch system. | ✕ | ✕ | ✓ |

| | | | | | |
|---|---|---|---|---|---|
| Transparent device endpoint detection capability | optional | Ability to determine the wet etching endpoint for visibly transparent fabrication devices. | × | ✓ | ✓ |

## 2.2 Design Specification

To meet the requirements, several comparisons between viable options for each subsection are performed and tabulated in the following tables.

**Table 11: Comparison table for microprocessor choices**

| | **Raspberry Pi 4B** | **Arduino Mega board** | **Arduino Uno WiFi Rev2.0 board** |
|---|---|---|---|
| **Board type** | Microprocessor (Single board computer) | Microcontroller | Microcontroller |
| **Peripheral connections** | • GPIO ports<br>• HDMI ports<br>• USB 2.0, USB 3.0 & USB C<br>• Ethernet port<br>• flexible flat cable (FFC) connector | • GPIO ports<br>• UARTs<br>• ICSP header | • GPIO ports<br>• ICSP header |
| **In-built internet connectivity** | • Wi-Fi<br>• Ethernet | Unavailable | Wi-Fi |
| **Availability of essential function libraries (e.g. OpenCV & Scipy)** | Available | Unavailable | Unavailable |
| **Community support** | Available | Available | Available |

| | | | |
|---|---|---|---|
| **Power consumption** | • 1 Core: 4.5 W<br>• 2 Core: 5.0 W<br>• 3 Core: 5.4 W<br>• 4 Core: 6.0 W<br>. | 0.66 W (with 9V supply) | 0.8856 W (with 9V supply) |
| **Clock speed** | 1.2 GHz | 16 MHz | 16 MHz |
| **Unit price** | RM 239.00 (4Gb RAM) | RM 179.00 | RM 239.00 |

Based on Table 11, Raspberry Pi 4B is selected as it is equipped with essential peripheral connections and the Python coding environment on the board provides convenient function libraries that are essential in this project. The power consumption is not a determining factor as this prototype will be a stationary device and be powered by the wall socket.

**Table 12: Comparison table for illumination system choices**

| | NeoPixels RGB LEDs | 10 W 395nm LEDs module | Self-made LEDs array from scratch |
|---|---|---|---|
| **Ease of accessibility** | Commercially available | Commercially available | Requires manual completion process (soldering and circuitry) |
| **Controllability** | Python function library available to control individual pixels | Using relay | Using relay |
| **Light spectrum** | Each pixel comprised of 3 primary colours (R, G, B) to output 16777216 possible colours. | 395 nm | Single colour |

| | | | |
|---|---|---|---|
| **Unit Price** | RM 11.90 | RM 24.00 | RM 0.10 (each) |

Besides, Neopixels RGB LEDs is selected over self-made LEDs array due to the ease of accessibility, ease of controllability, and the capability to output a wide range of colours. The compactness form factor eliminates the need of soldering and allows plug-and-play operation. Besides, the 10 W 395 nm LEDs module is also selected to provide near-UV spectrum light sources to examine the potential of a near-UV imaging chamber for transparent multi-layered substrate (Al/ZnO/Al/SU-8/ultra-thin Si/SU-8).

**Table 13: Comparison table for reflective intensity measurement systems choices**

| | **Digital microscope with CCD camera** | **Oriel 77341 Photomultiplier tube (185-870 nm)** |
|---|---|---|
| **Ease of integration** | Easy | Complex |
| **Expensive optics requirement (lenses & fibre optics)** | No | Yes |
| **Signal type** | Image | Voltage |
| **Unit Price** | RM 241.10 | RM 8985.41 |

From Table 13, the compact form factor and the ease of integration of commercial digital microscopes with CCD camera eliminates the need of expensive optics to construct the prototype. It is also chosen as it fits the purpose of measuring reflective intensity and able to meet the budget limitation.

Table 14: Comparison table for locomotion system choices

| | NEMA 17HS4401 Bipolar Stepper Motor | Standard 3.2kg.cm Plastic Gears Analog Servo | 12V 75RPM 5.5kgfcm Brushed DC Geared Motor |
|---|---|---|---|
| **Motion accuracy** | 1.8 degree (full step mode) | 1 degree | N.A. |
| **Operating voltage** | 12V | 4.8 V to 6 V | 12 V |
| **Operating current** | 1.7A per phase | • 0.78A (at 4.8V)<br>• 0.85A (at 6V) | < 1.2A |
| **Rotation type** | Continuous | 180 degrees (not suitable for continuous rotation motion) | Continuous (56 RPM) |
| **Holding torque** | 0.4 Nm | • 0.294 Nm (at 4.8V)<br>• 0.3138 Nm (at 6V)<br>• Might suffer jitter issue due to closed-loop feed back | 0.539 Nm |
| **Unit price** | RM 39.00 | RM 28.00 | RM 55.00 |

From Table 14, stepper motor that has reasonable holding torque is chosen as this project performs mostly pick-and-place operation in which holding and low speed rotation are executed. Besides, the incremental motion of the stepper motor could eliminate the need for closed-loop controller, allowing simple motion architecture.

**Table 15: Comparison table of material for overall mechanical structure**

| | Aluminium extrusion (30 x 30 mm) | MDF board |
|---|---|---|
| **Strength** | High | Moderate |
| **Ease of integration** | • commercially available accessories such as L-brackets and T-slot screws<br>• Ready-made groove on material for easy integration | Requires manual completion process (e.g., cutting and drilling) |
| **Unit price** | RM 0.28/cm | N.A. |

Based on Table 15, aluminium extrusion is chosen due to its ease of accessibility as it is commercially available for different dimensions. Most importantly, the accessories of connectors are widely available and hence eliminate the inconvenience of manual postprocessing.

## 2.2.1 Material and mechanical design

### 2.2.1.1 XY-axis motion system

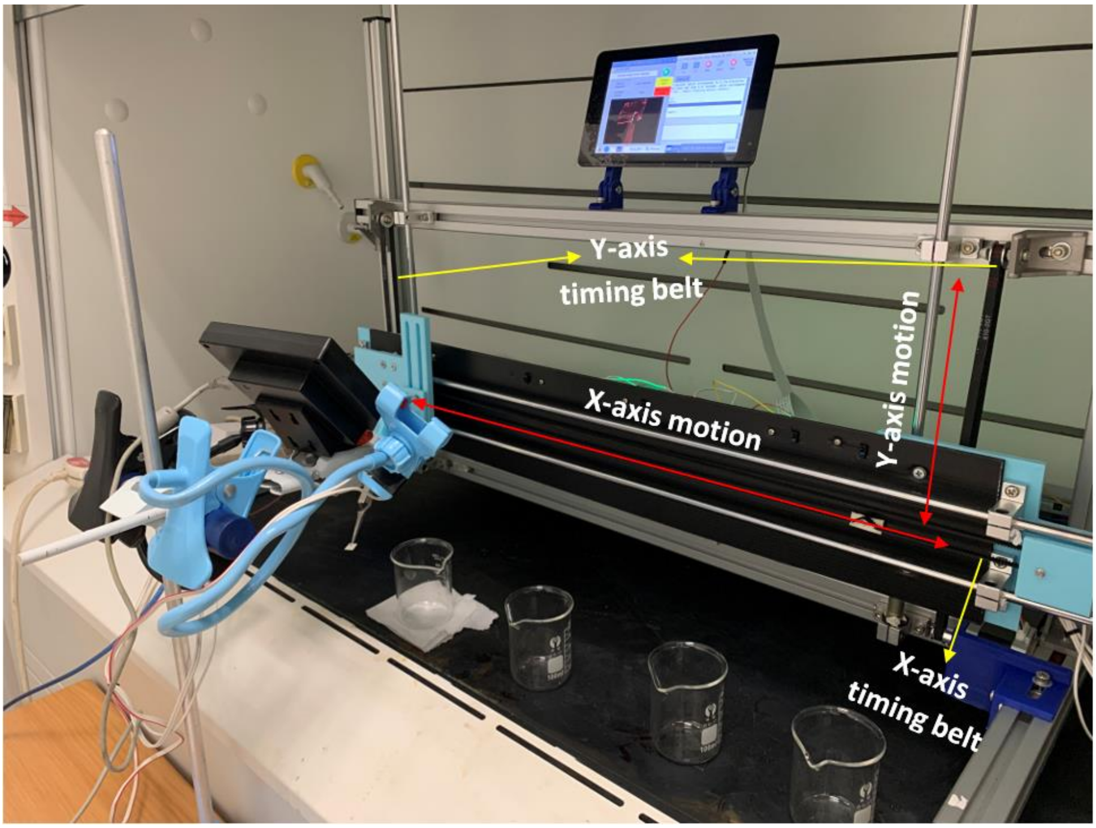



**Figure 30: The x-y axis motion system.**

Figure 30 shows the structure of the automated wet etcher with optical endpoint detection system. To automate the movement between each station for post-development fabrication, a XY-axis motion system has been chosen which the tweezer holder will move across all the beakers arranged in a line. The consideration for each aspect is summarized in Table 16.

**Table 16: Subcomponents of the motion system.**

| Aspect | Components used | Details |
|---|---|---|
| **Mechanical structure** | square aluminium extrusions of 30-30cm dimension | Individual pieces are connected by L-brackets shown in Figure 33. |

| | | |
|---|---|---|
| **Rail guide** | 8 mm diameter aluminium rod. | Connected by slider block and to guide tweezer holder motion as shown in Figure 34. |
| **Driving power** | NEMA 17HS4401 Bipolar Stepper Motor | 2 units on the vertical axis and 1 unit on the horizontal axis. |
| **Rotational to linear motion conversion** | • Pulley<br>• Timing belt<br>• Timing belt clamp | To convert the rotational motor motion to linear motion. The construction is shown in Figure 35. |

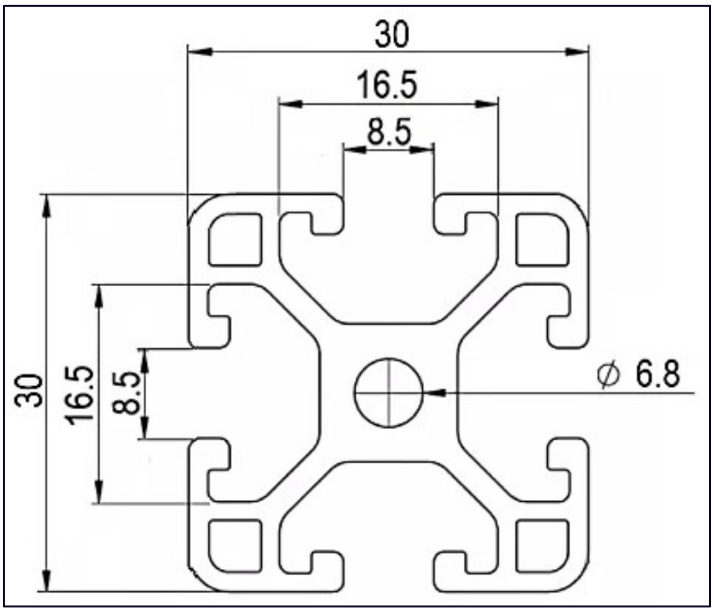



**Figure 31: Dimension of the 30-30 aluminum extrusion.**

**Figure 32: Size 30-30 L-bracket**

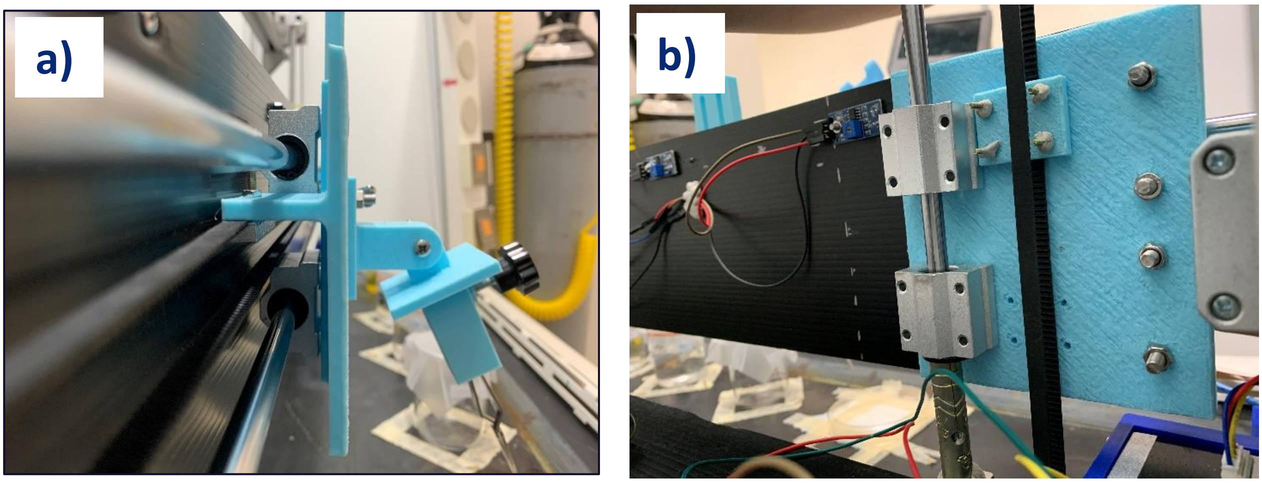



**Figure 33: Sliding blocks attached to the aluminium rods on the horizontal axis (a) and on the vertical axis (b) forming the XY-axis motion system.**

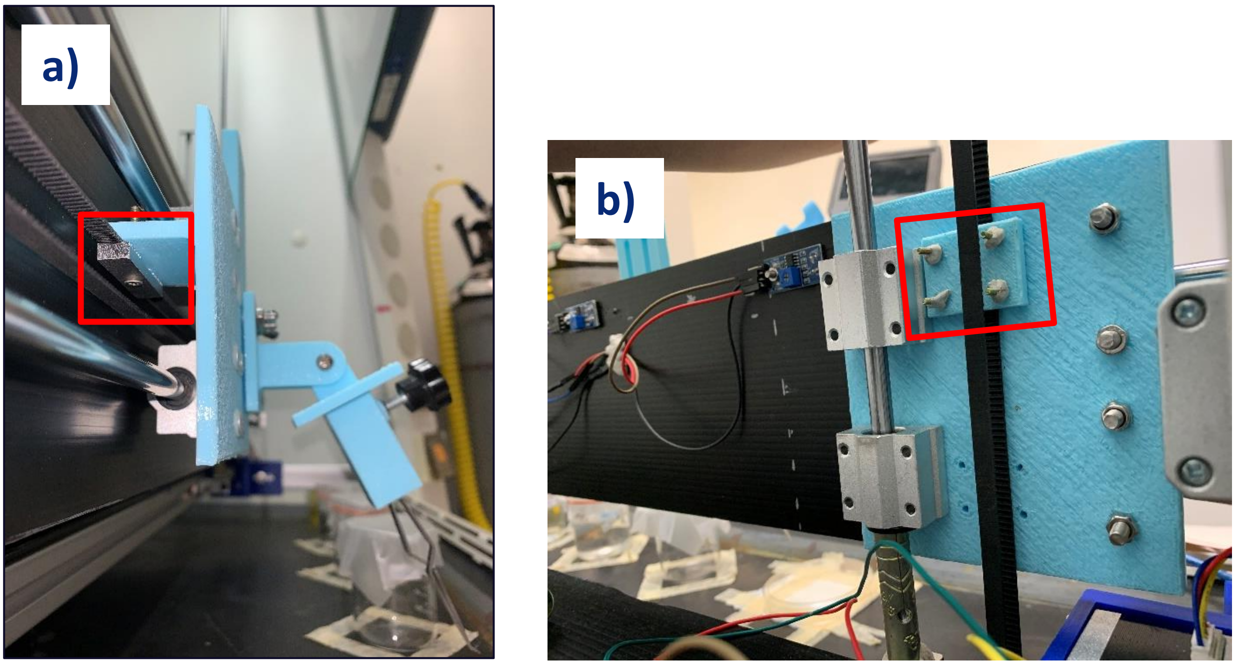



**Figure 34: Timing belt attached to the timing belt clamps on the horizontal (a) and vertical axis (b).**

### *2.2.1.2 Mechanical connecting components*

Looking at Figure 35, the XY-axis motion subsystems will be integrated together with carefully designed connectors. From Figure 36, the shaft support brackets are bolted onto the aluminium extrusions (vertical axis) and onto 3D printed plates (horizontal axis) providing stable support for the aluminium sliding rod. The slider block is bolted onto the 3D printed platform as shown in Figure 37, allowing the tweezer holder to be attached.

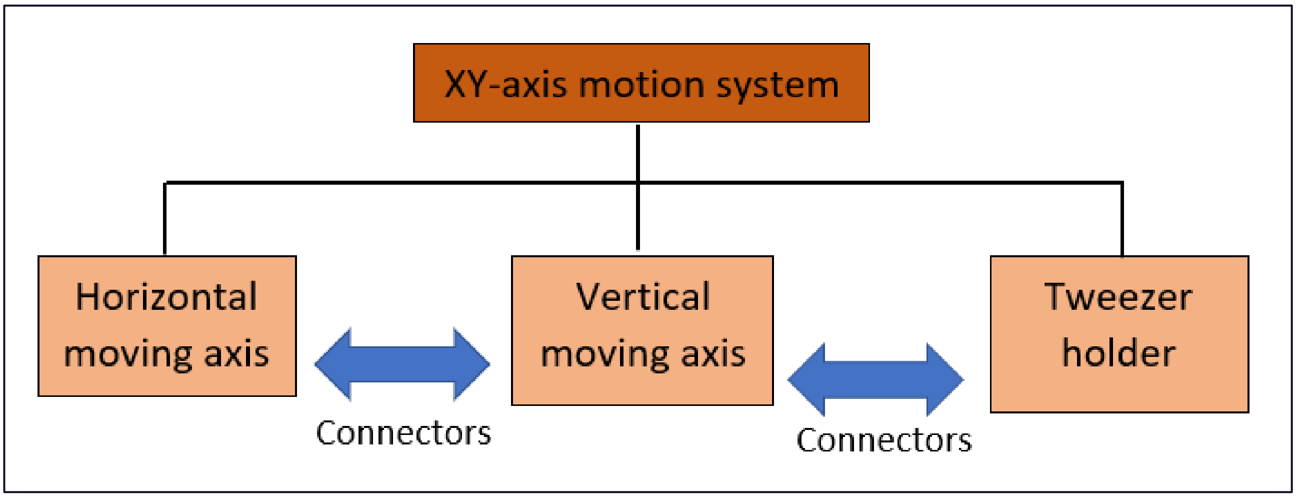



**Figure 35: Subcomponents of XY-axis motion system**

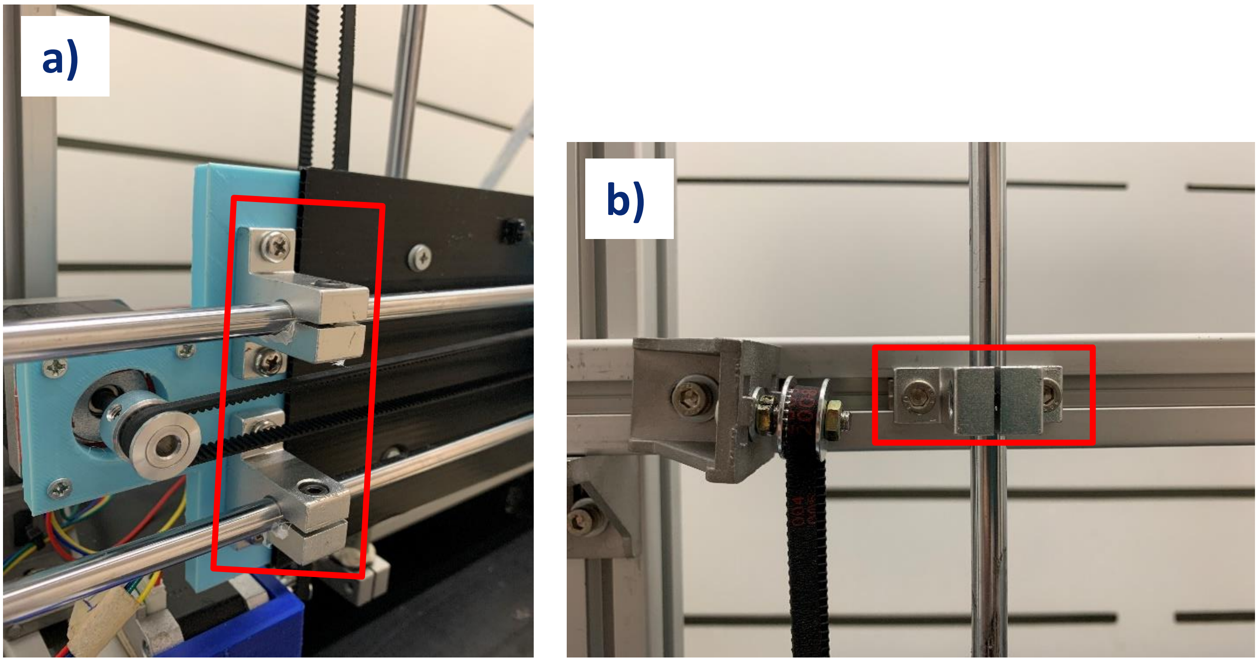



**Figure 36: Shaft support bracket bolted on the 3D printed plate for horizontal axis (a) and onto aluminum extrusions for vertical axis (b).**

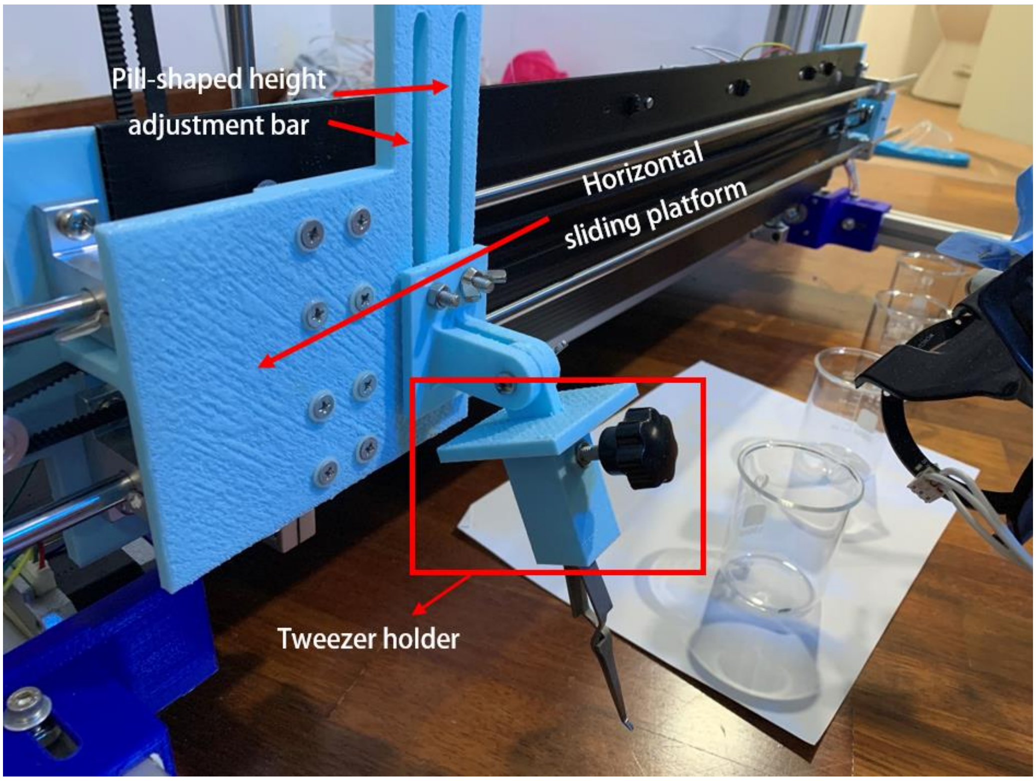



**Figure 37: The tweezer holder on the horizontal sliding plate.**

To connect the user-loaded, an adjustable tweezer holder shown in Figure 38 is designed. With the adjustment knobs, the height of the holder is adjustable to accommodate tweezers with different lengths as well as changeable angle to provide the clearest camera view.

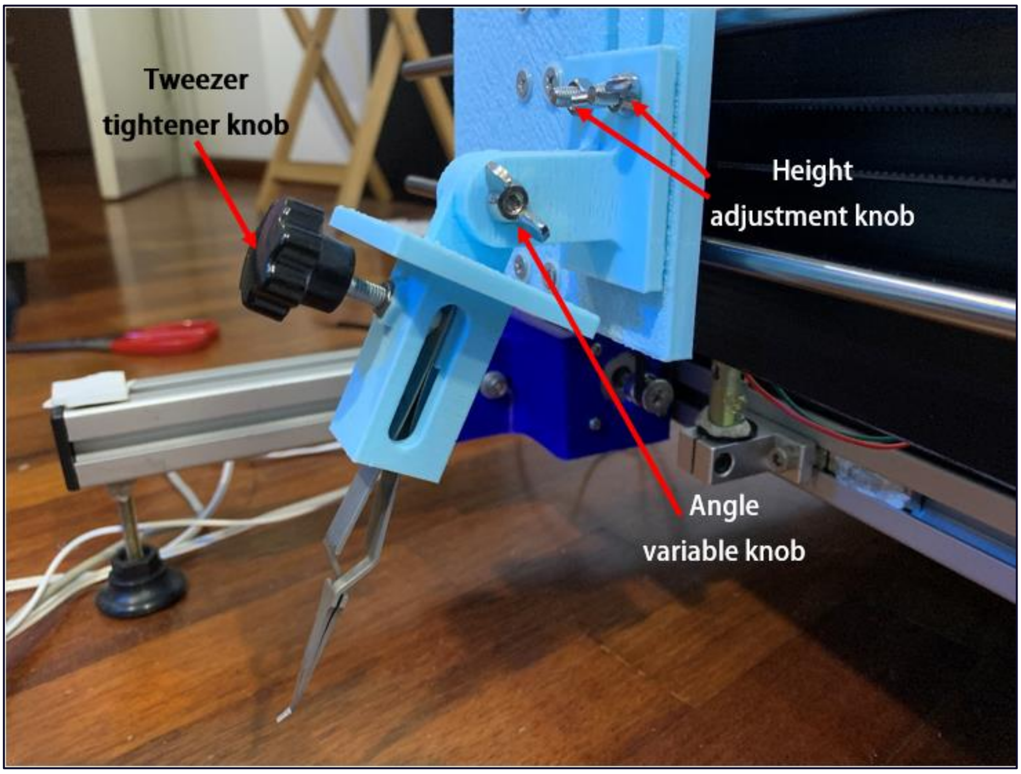



**Figure 38: The tweezer holder with adjusting knobs.**

### 2.2.2 Electronic design

The overall integration of the electronic is summarized in Figure 39. The blue arrows represent the signal flow from each component. Each of the subsystems is discussed in detail in the individual subsections.

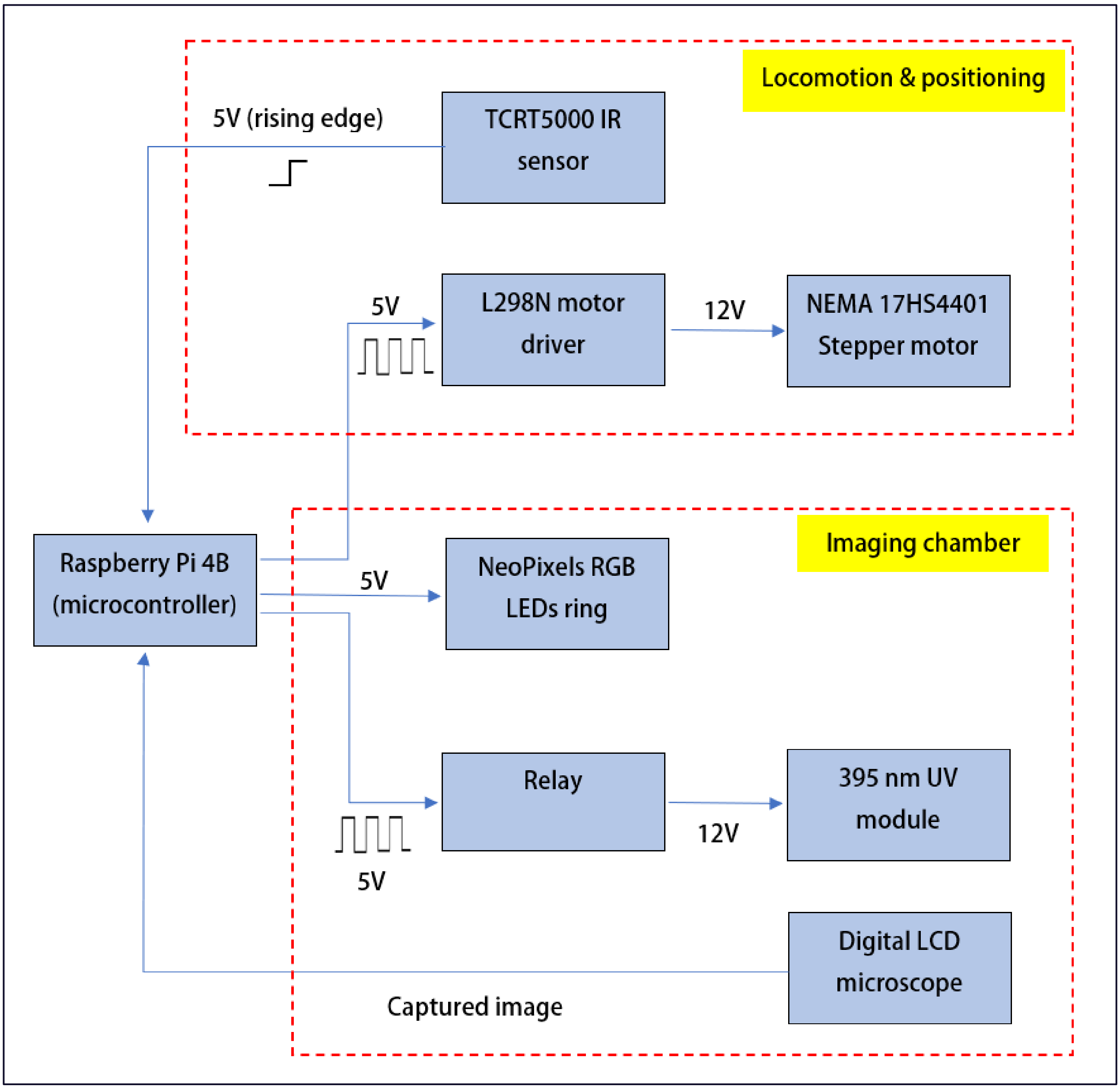



**Figure 39: System block diagram of the overall electronic systems.**

### *2.2.2.1 Power supply*

To power up the system, Meanwell AC/DC Power Supply (PSU) as shown in Figure 40 is chosen as the highest rating components are 12V rated as tabulated in Table 17. The current rating of the PSU is the main consideration factor. The current requirement is solely calculated based on the motor's current rating as it consumes the largest amount of current. Based on the datasheet of NEMA 17HS4401 stepper motor, at least 5.1 A is required assuming all 3 motors operated concurrently.

$$1.7A\ (rated\ current/phase) \times 3\ motors = 5.1\ A$$

A LM2596 DC-DC step down voltage regulator is used to lower down the voltage to 5V to power up lower voltage rating components.

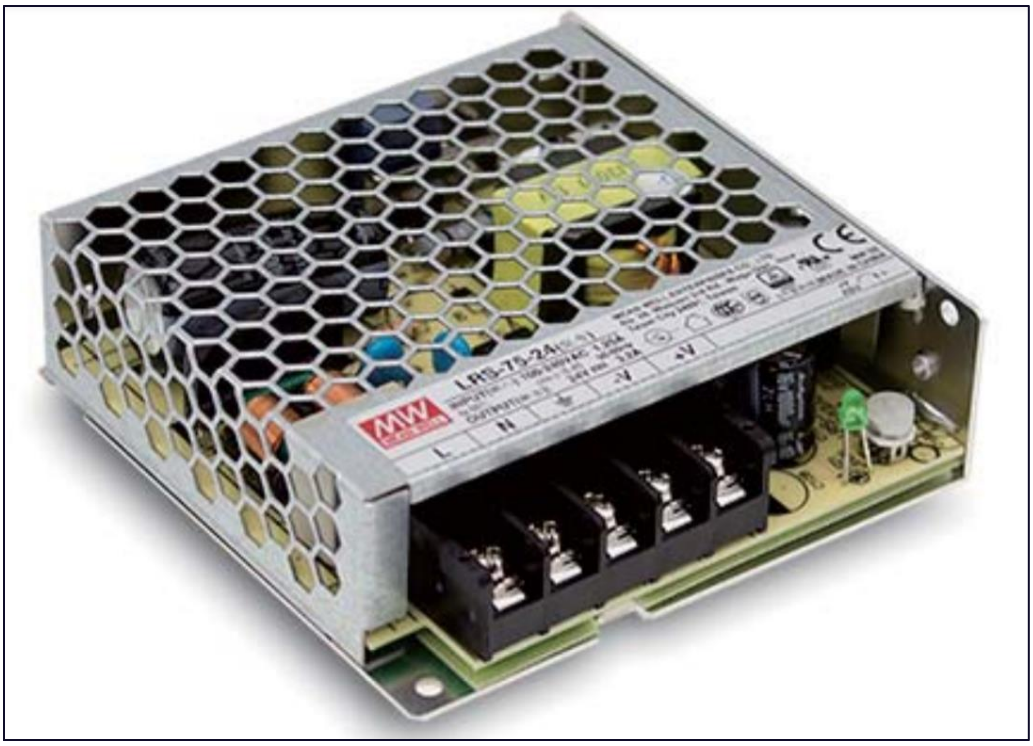


**Figure 40: Meanwell AC/DC Power Supply (PSU), 1 Output, 72 W, 12 V, 6 A**

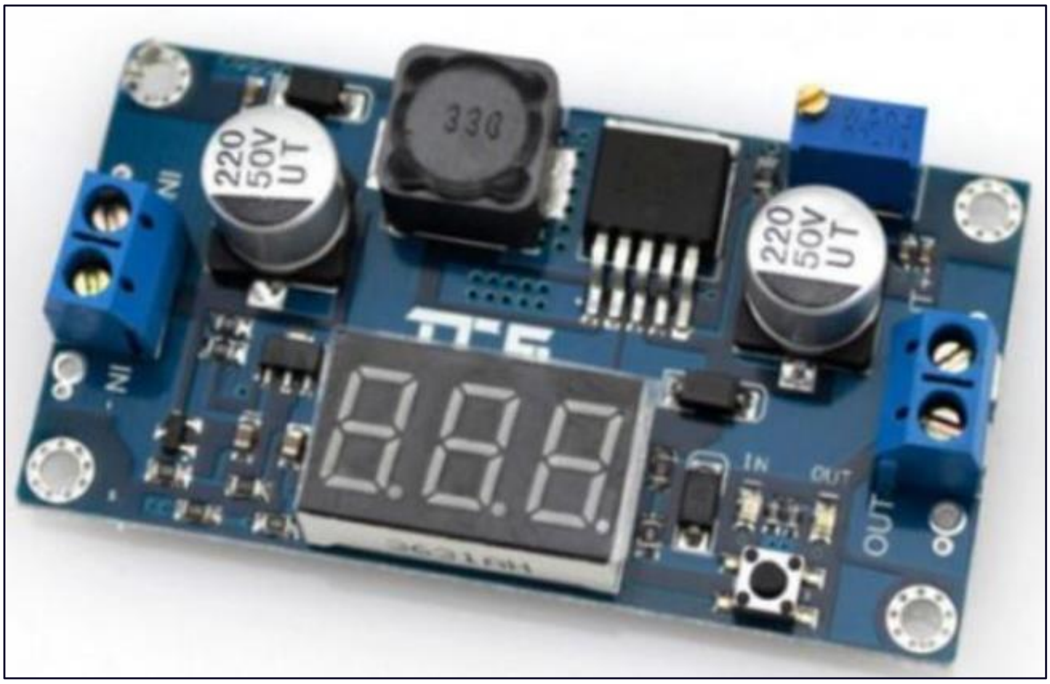



**Figure 41: Step down voltage regulator.**

**Table 17: Component's voltage rating**

| Components | Voltage rating (V) |
|---|---|
| **NEMA 17HS4401 Bipolar Stepper Motor** | 12 |
| **UV LEDs square module** | 11 |
| **Neopixels RGB LEDs ring** | 5 |
| **5V 1 channel relay** | 5 |
| **TCRT5000 Infrared Module** | 5 |

### *2.2.2.2 Locomotion and positioning*

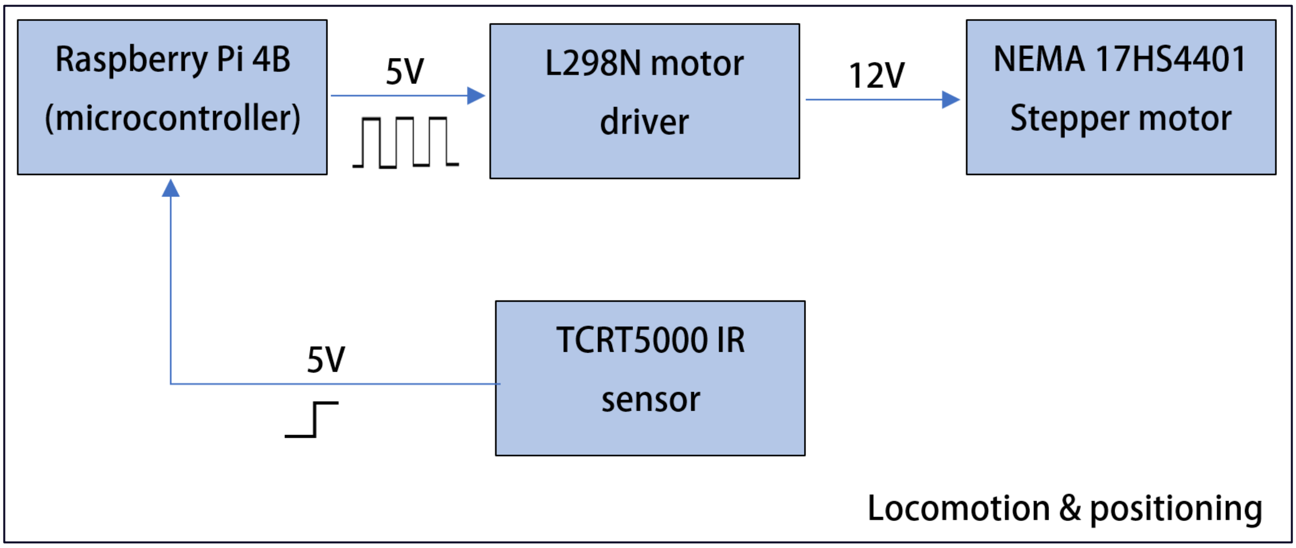



**Figure 42: System block diagram of the locomotion and positioning subsystems.**

To automate the motion, the combination of stepper motors and IR sensor is used. The motor driver shown in Figure 45 will be receiving the stepping command from the microcontroller to operate the NEMA 17HS4401 stepper motor. Half stepping which the drive alternates between 2 phases on and 1 phase on with the state diagram shown in Figure 46 will be applied to provide a smoother linear motion.

Technically, the positioning of stepper motor motion can be open-loop which a fixed count is given for each station. However, in this project, TCRT5000 Infrared Modules are positioned at the station as shown in Figure 47 to ensure accurate positioning for the tweezer to enter the beaker.

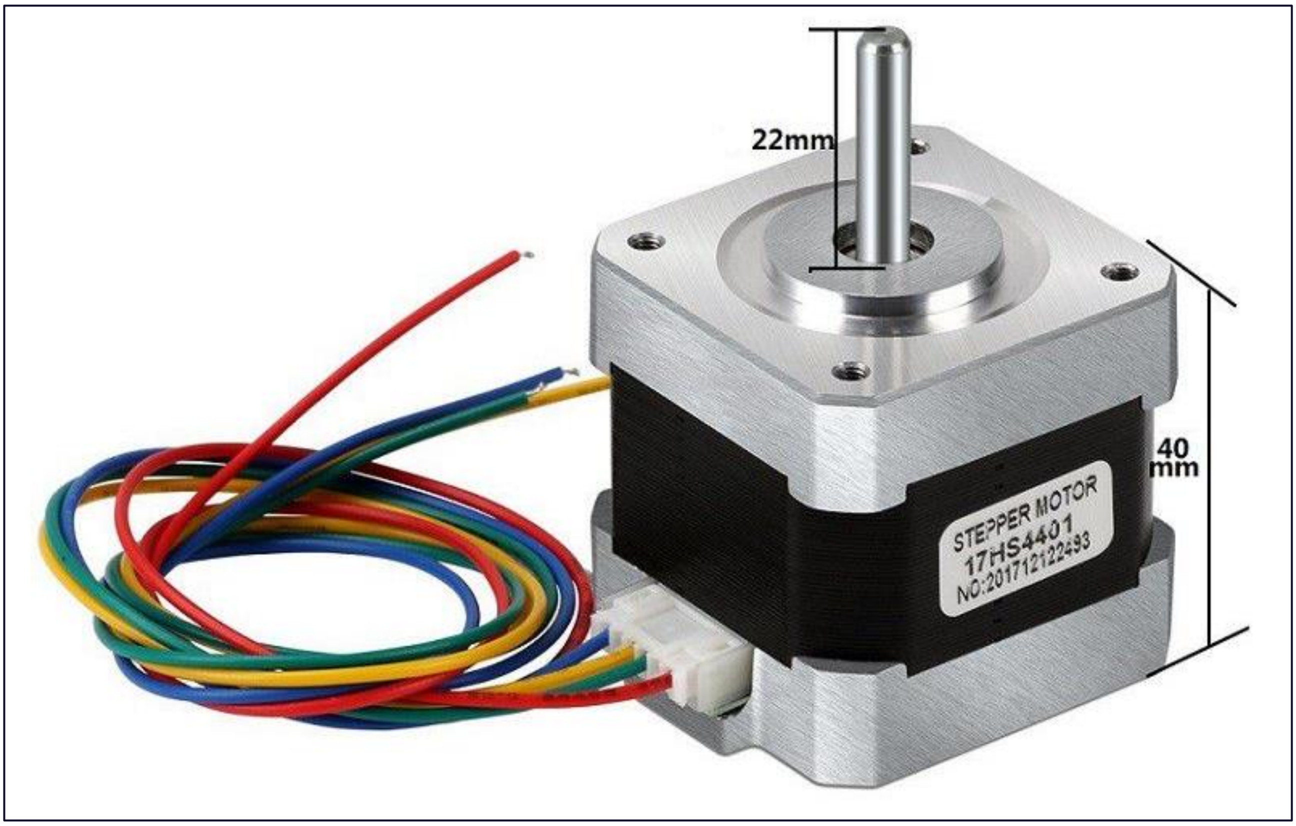



**Figure 43: NEMA 17HS4401 stepper motor**

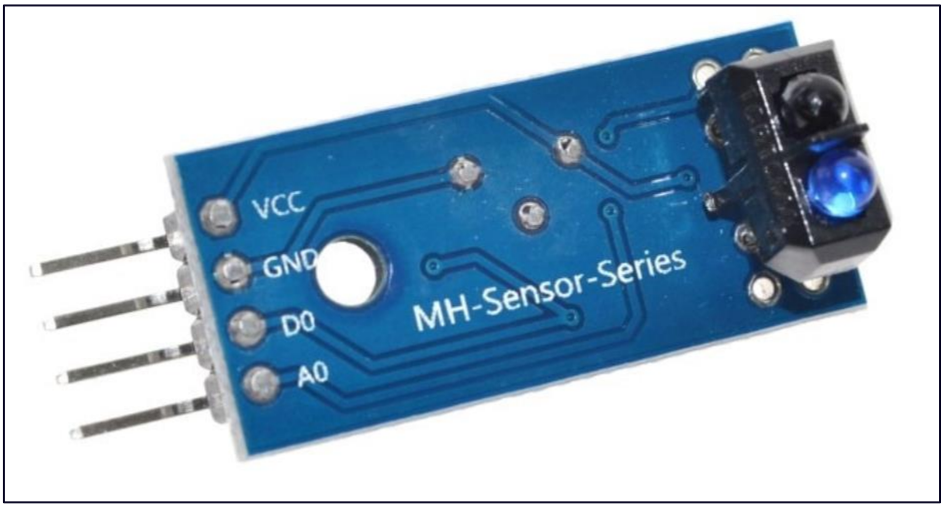



**Figure 44: TCRT5000 Infrared Module.**

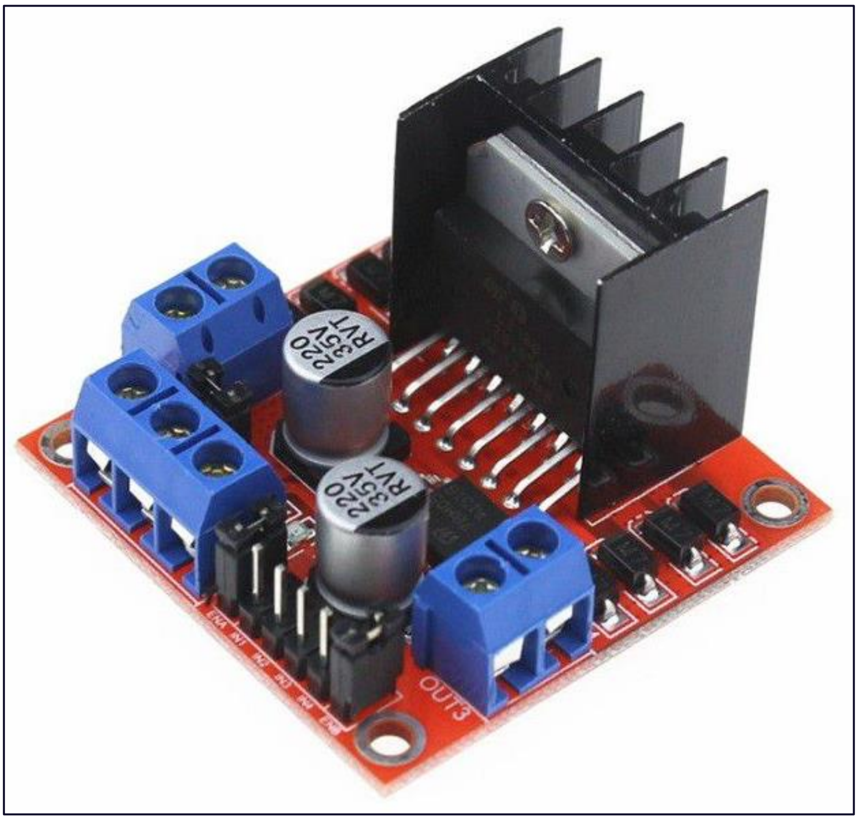
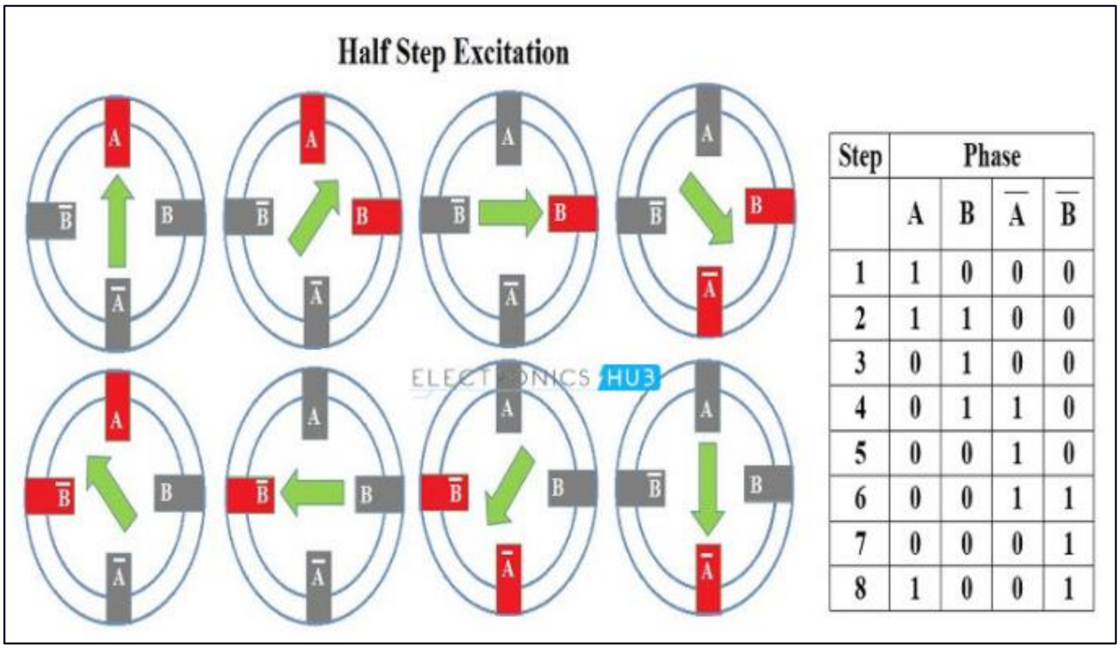



**Figure 45: L298N motor driver.**




| Step | Phase | | | |
|---|---|---|---|---|
| | A | B | $\bar{A}$ | $\bar{B}$ |
| 1 | 1 | 0 | 0 | 0 |
| 2 | 1 | 1 | 0 | 0 |
| 3 | 0 | 1 | 0 | 0 |
| 4 | 0 | 1 | 1 | 0 |
| 5 | 0 | 0 | 1 | 0 |
| 6 | 0 | 0 | 1 | 1 |
| 7 | 0 | 0 | 0 | 1 |
| 8 | 1 | 0 | 0 | 1 |

**Figure 46: State diagram for half stepping operation.**

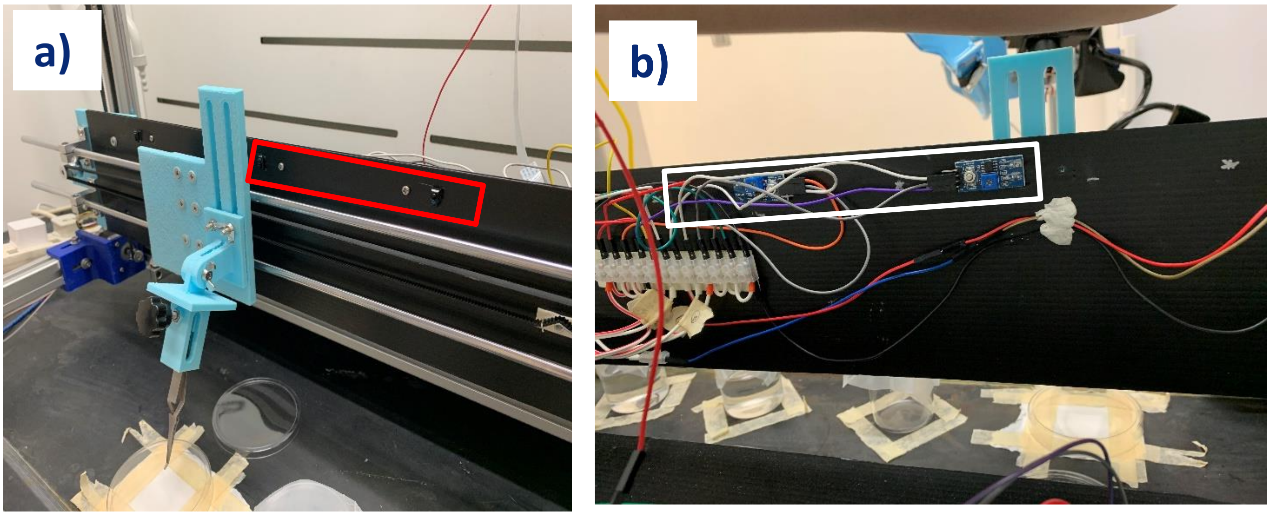



**Figure 47: The front view (a) and the rear view (b) of the IR sensors positioning in the horizontal motion axis.**

### *2.2.2.3 Imaging chamber*

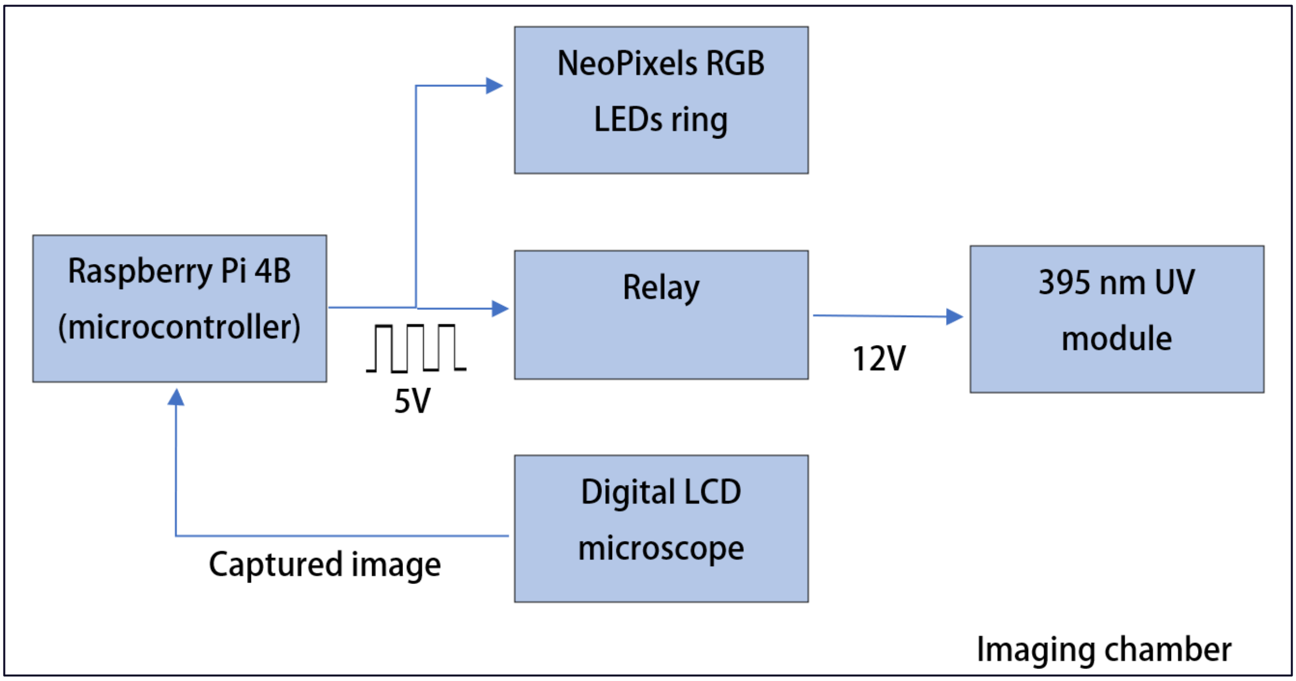



**Figure 48: System block diagram of the imaging chamber subsystems.**

To have the flexibility to illuminate with different light spectrum to obtain measurable parameters change, 2 different light sources which are NeoPixels RGB LEDs ring and 395nm UV module have been integrated in the system with the connections shown in Table 18. Besides, a HD 3.6MP camera equipped digital microscope is utilized and the captured image will be processed by the microcontroller.

Both the light sources and digital microscope will be directed towards the fabrication device as shown in Figure 50 and Figure 51.

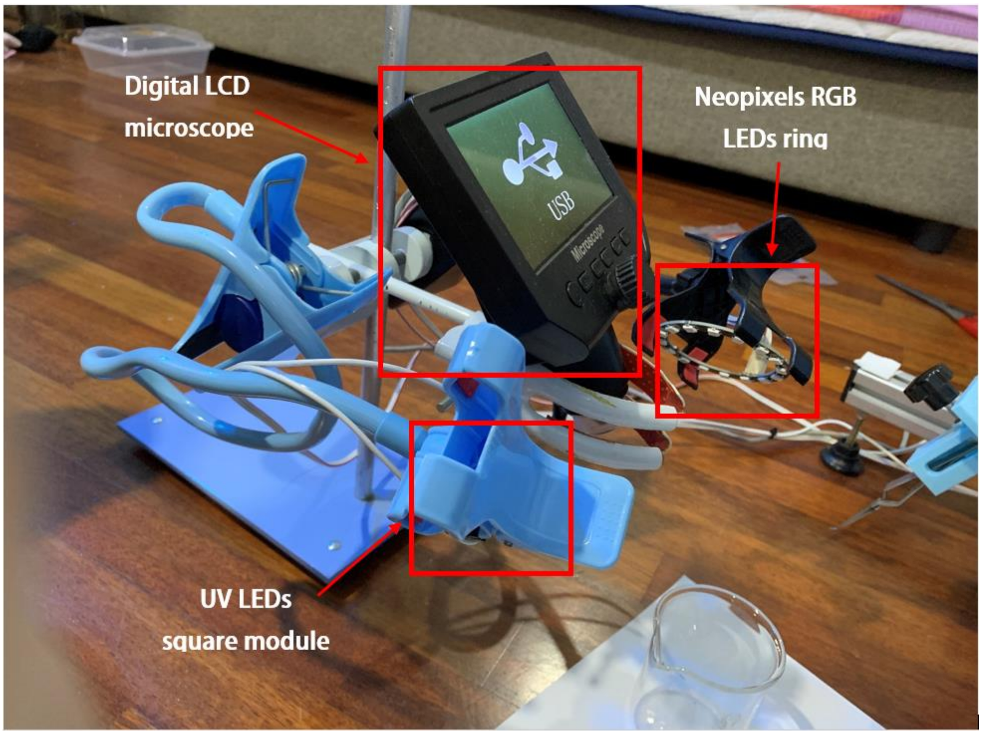



**Figure 49: Overview of imaging chamber (top right view).**

**Table 18: Imaging chamber's components and connection type.**

| Components | Signal Connection type |
|---|---|
| **Neopixels RGB LEDs ring** | single-wire control protocol |
| **UV LED square module** | Relay |
| **Digital LCD microscope** | USB Type-A to USB Type-A mini |

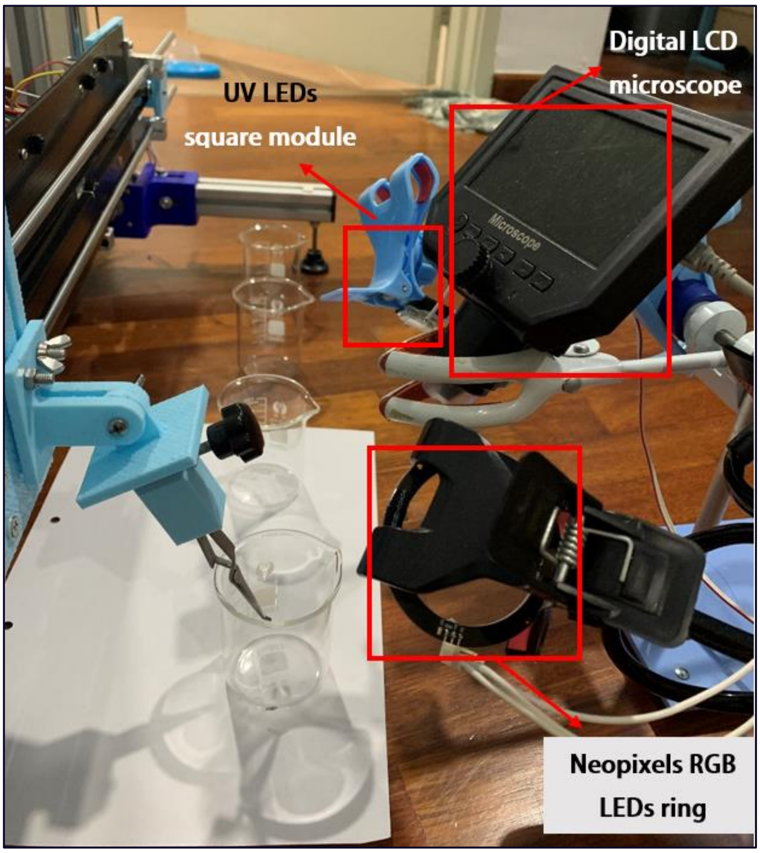



**Figure 50: Overview of imaging chamber (top left view).**

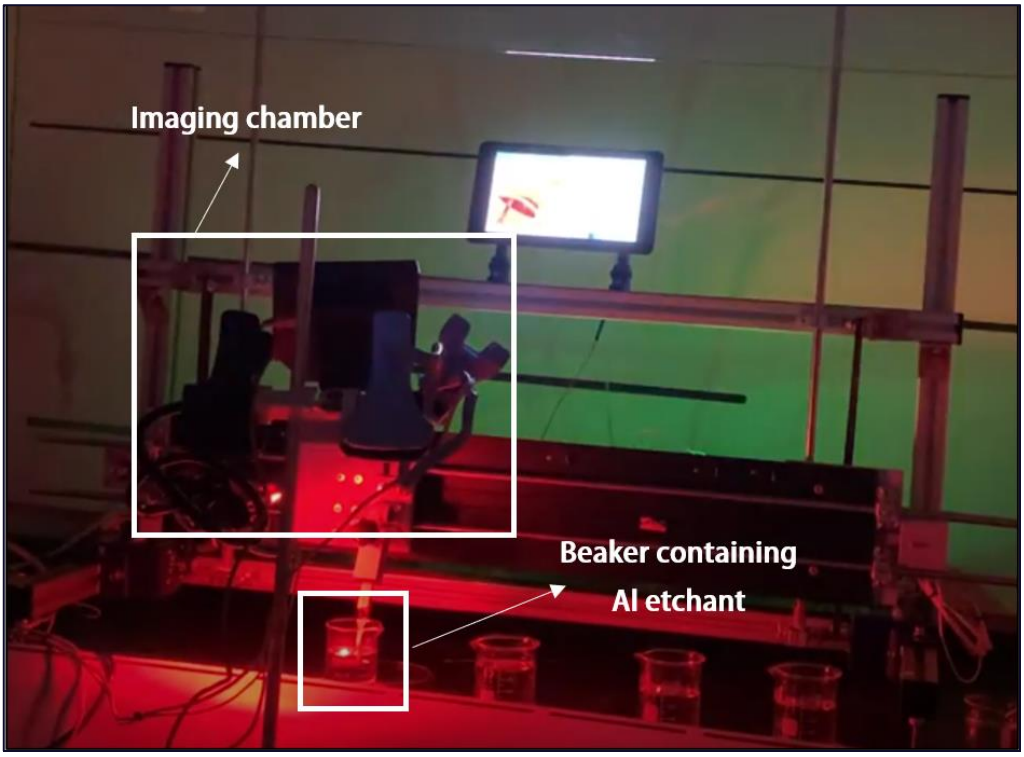



**Figure 51: Imaging chamber (with red light) in the prototype during operation.**

## 2.2.2.4 Circuitry schematics

The detailed schematics of the components' connections are summarized in Figure 52.

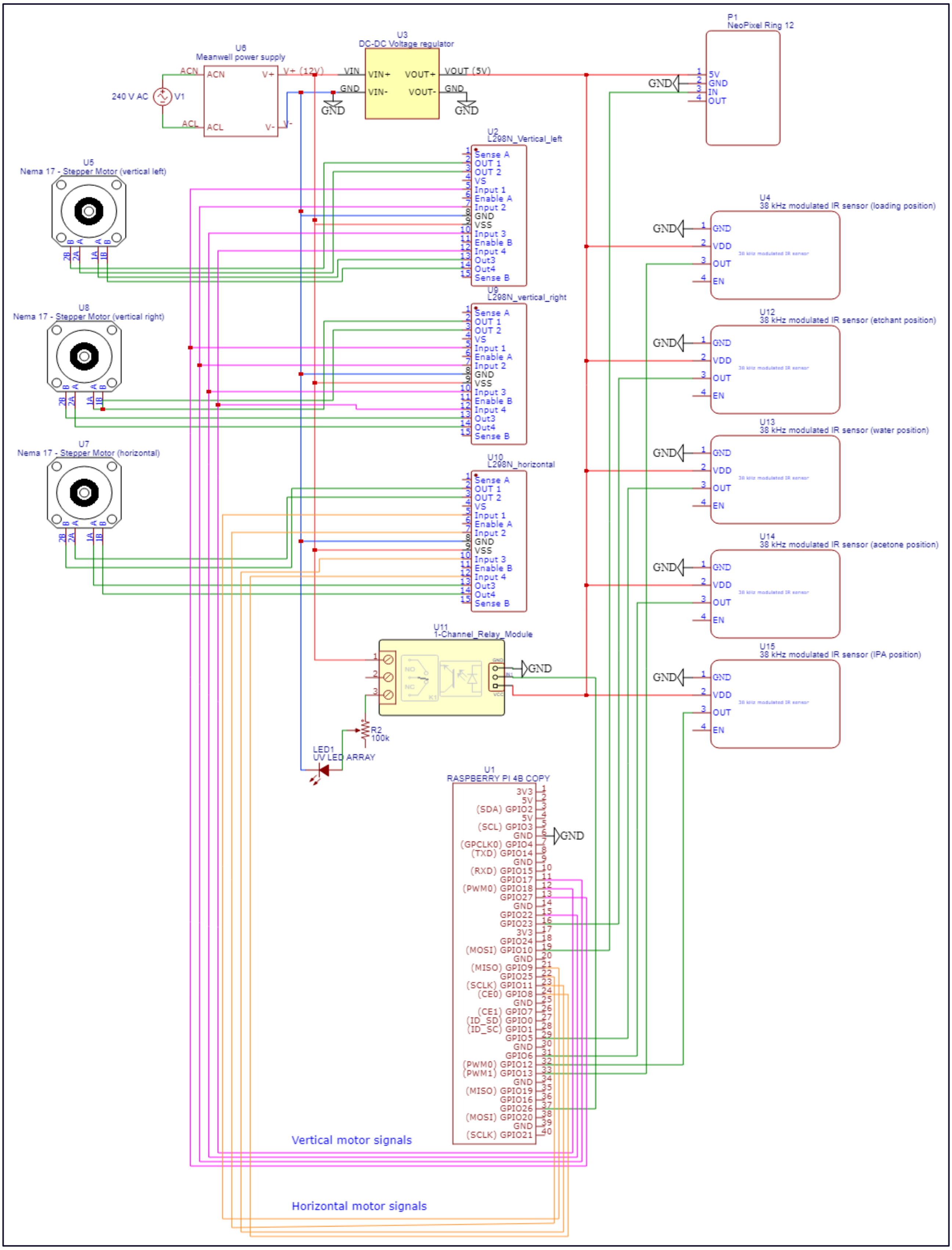



**Figure 52: Circuit schematics of the system**

## 2.2.3 Software algorithm design

### 2.2.3.1 Human Machine Interface (HMI)

To ease the operation, HMI as shown in Figure 53 is created with Python Tcl/Tk GUI toolkit. The respective function of each widget is explained in detail in Table 19. During the masking stage, a HSV adjustment threshold window will appear as shown in Figure 54 to enable the adjustment for suitable masking threshold. The actual presentation of the HMI on the prototype is shown in Figure 55.

One thing worth noticing is that the “Auto endpoint/Manual endpoint” button allows the user to manually toggle between fully automated and semi-automated endpoint detection mode. In the semi-automated mode, the user is given the flexibility to overwrite endpoint determination as shown in Figure 56 by clicking the “green arrow”.

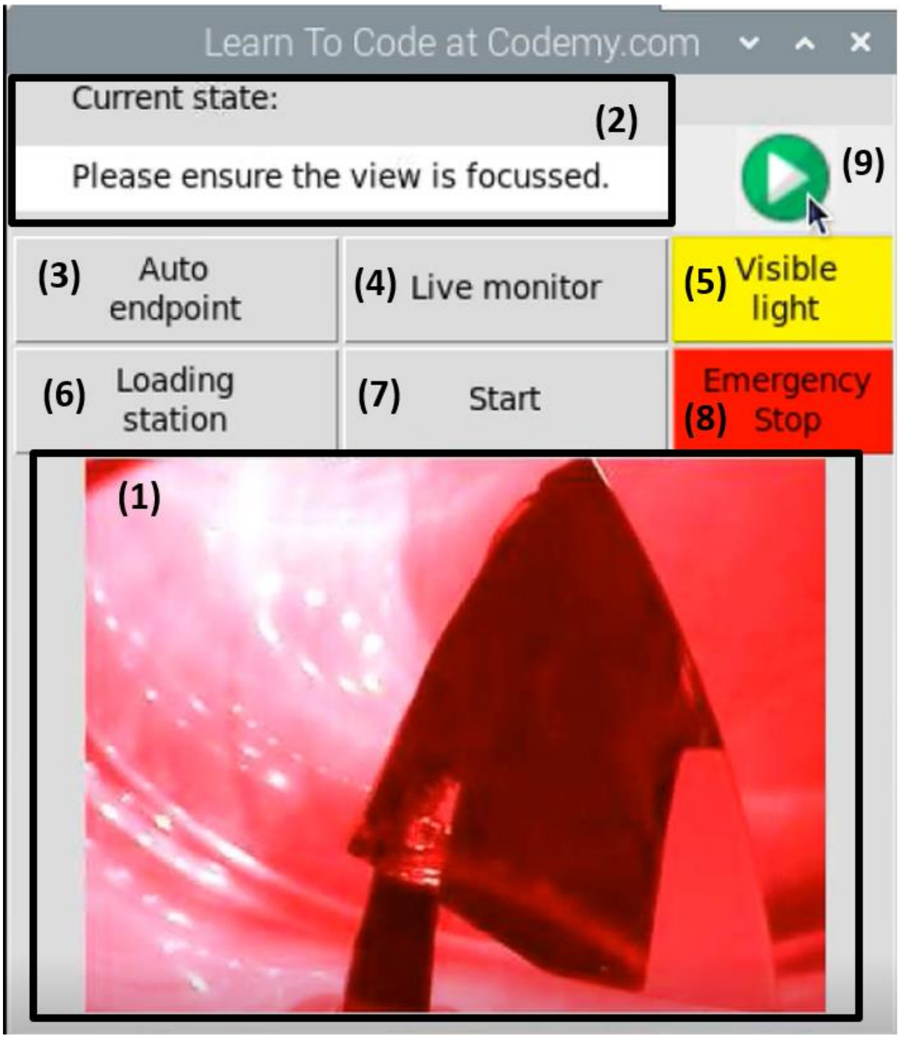



**Figure 53: Overview of the HMI with the widgets labelled.**

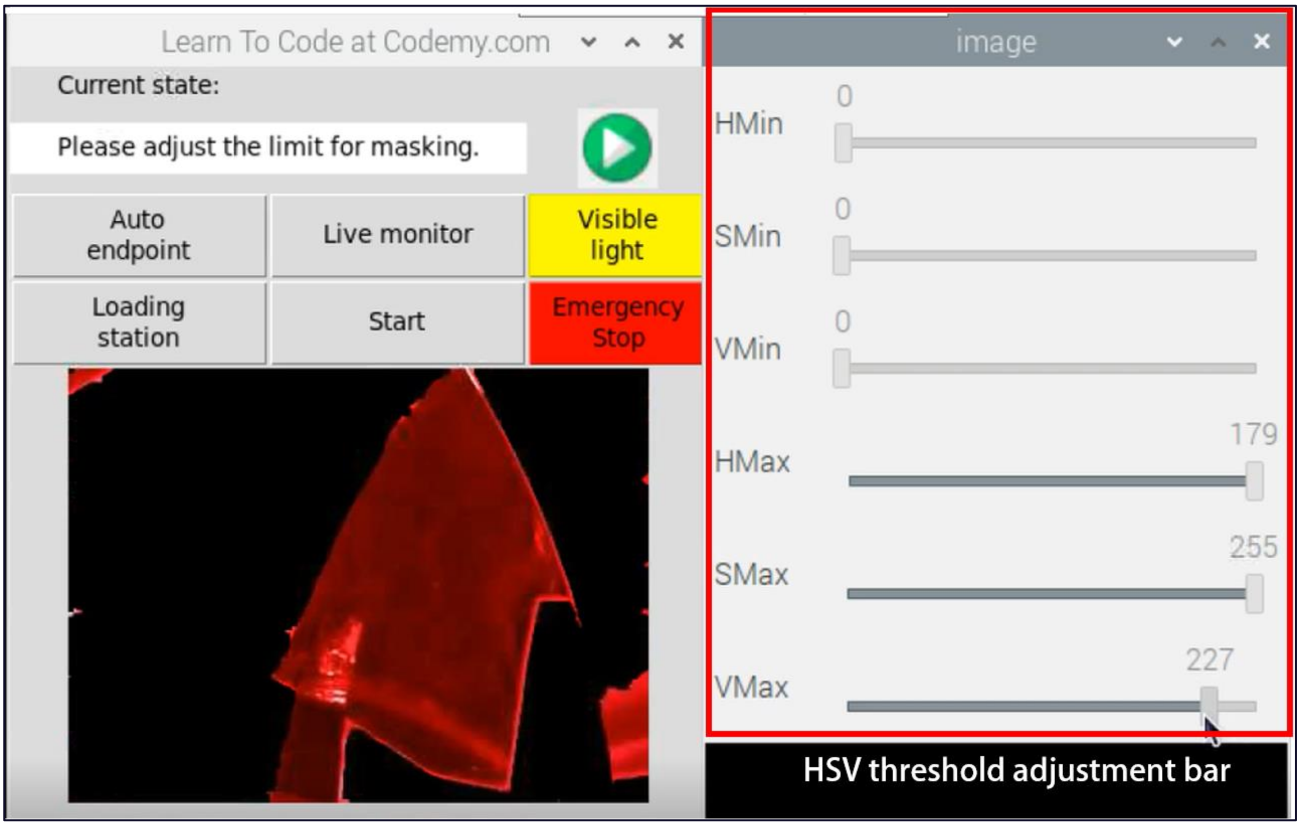



**Figure 54: HMI with the HSV threshold adjustment window pop up.**

**Table 19: HMI's widgets and the functions.**

| No. | Widget | Function |
|---|---|---|
| 1 | Image displaying window | Display the image captured from digital microscope |
| 2 | Current state displaying window | Display the current operation mode |
| 3 | Auto endpoint/ manual endpoint | Toggle between fully automated mode and semi-automated mode |
| 4 | Live monitor/post analysis | Toggle between live monitoring mode and post analysis mode |
| 5 | Visible light/ UV light | Toggle the light source type for monitoring process |

| | | |
|---|---|---|
| **6** | Loading station | Move the tweezer holder back to loading position |
| **7** | Start | Start the etching process. Tweezer holder moves towards first etchant bath. |
| **8** | Emergency stop | Stop the current operation and moves the tweezer holder back to loading station. |
| **9** | Green arrow | Confirm button for (1) view focussing, (2) HSV masking adjustment and (3) endpoint overwrite. |

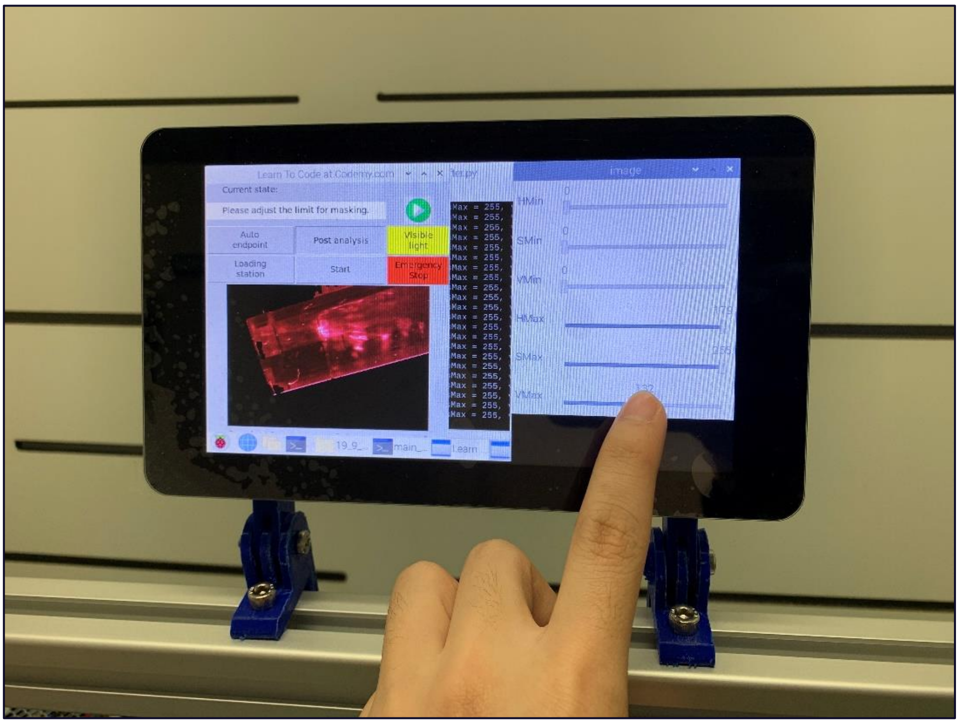



**Figure 55: HMI on the Raspberry Pi 7 Inch Touch Screen Display.**

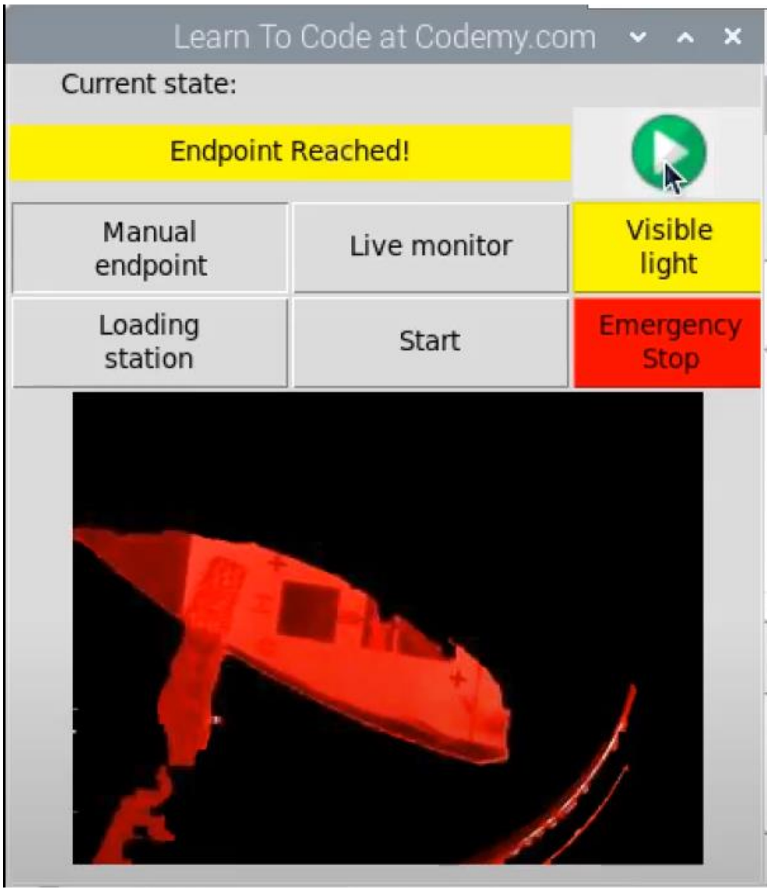



**Figure 56: Endpoint reached flag appearing on the "current state display".**

### 2.2.3.2Pixels based photo analysis

To measure the reflective intensity changes throughout the etching process, a pixels-based photo analysis is carried out. Since red light illumination is used in this prototype, the red pixel of the image is extracted as a measurement for etching monitoring. The camera's vision field covers more than the fabrication device area, hence any changes in the background will affect the pixels-based analysis. To negate this noise, a Hue, Saturation and Value (HSV) colour space representation thresholding as shown in Figure 57 is used to separate the background and the fabrication device. Interestingly, HSV model is chosen over the RGB model to allow user to adjust the masking threshold as HSV model corresponds closely to the human's visual colour perceptions [29].

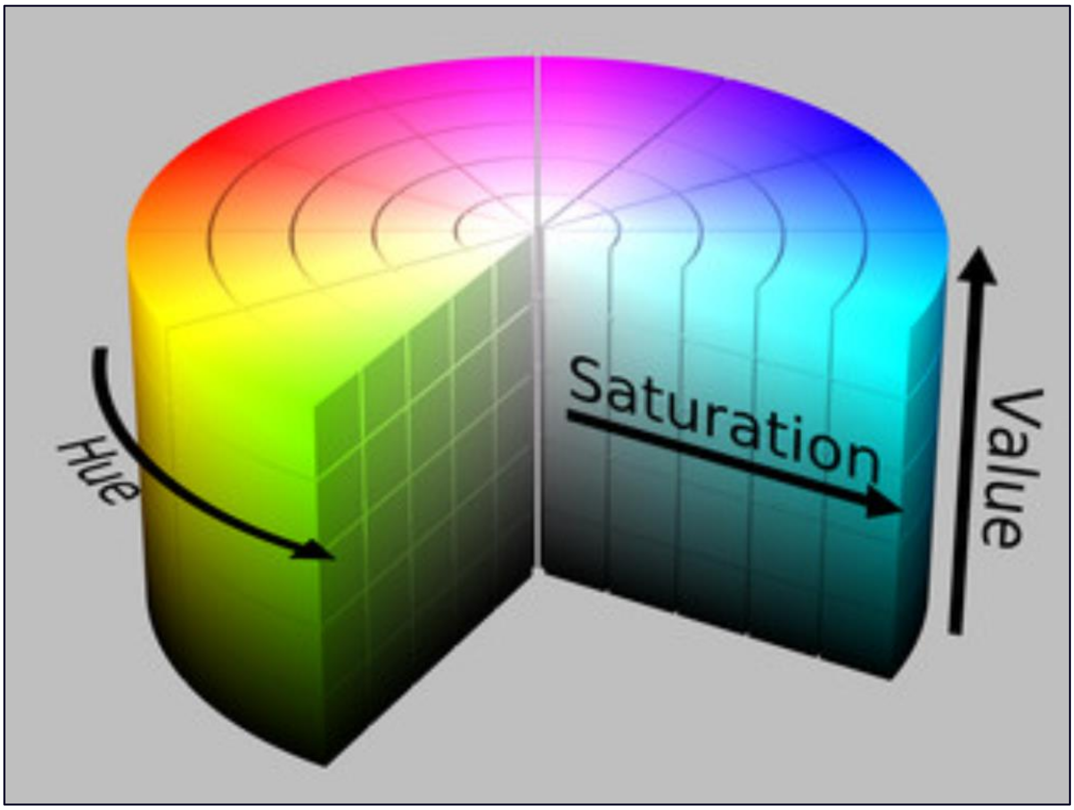



**Figure 57: HSV colour space representation model. [30]**

The detailed operation flowchart of obtaining the mask is shown in Figure 58 and the obtained binary mask is shown in Figure 60. Morphological functions from OpenCV library are used to smoothen the small holes in the raw binary mask obtained from the HSV thresholding operation. The comparison results of before and after the masking operation on the fabrication device are shown in Figure 61 and Figure 62, showing an improvement of locating the device in the image. This binary mask is saved and applied to the images throughout the etching monitoring process.

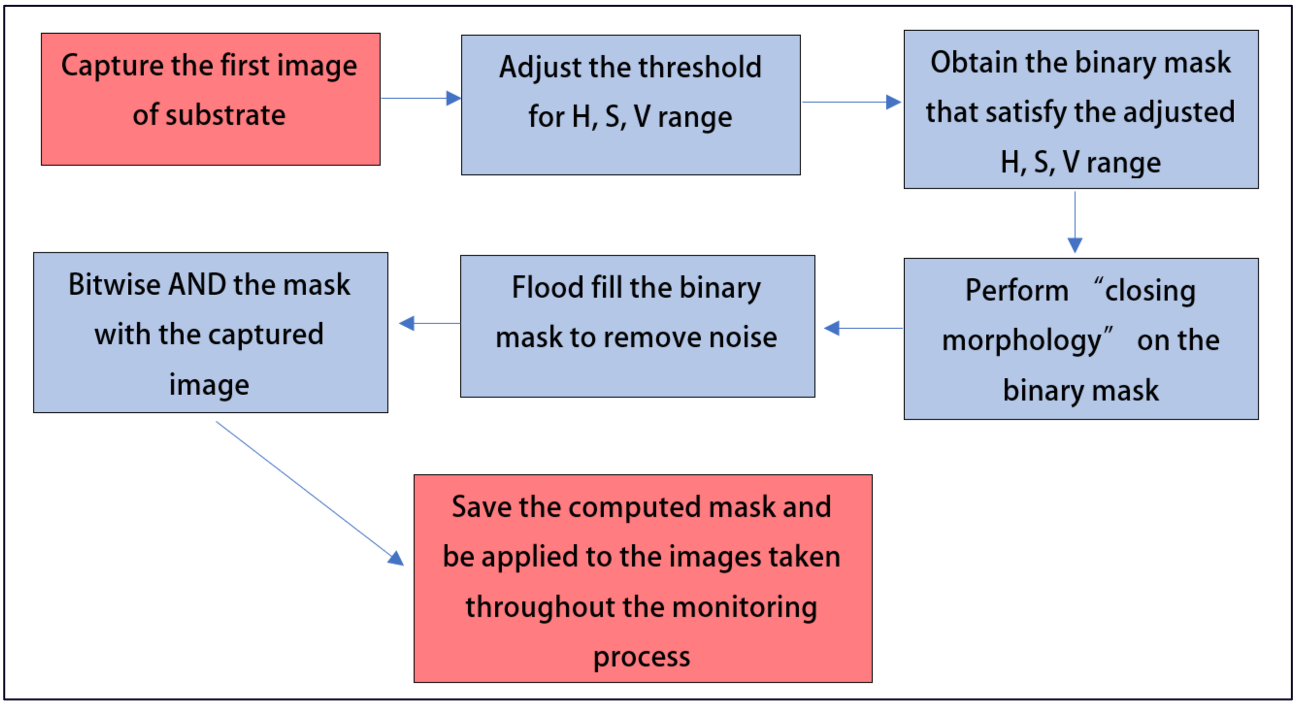



**Figure 58: Image masking flowchart.**

**Figure 59: The result of closing operation. (left) before and (right) after the operation.[31]**

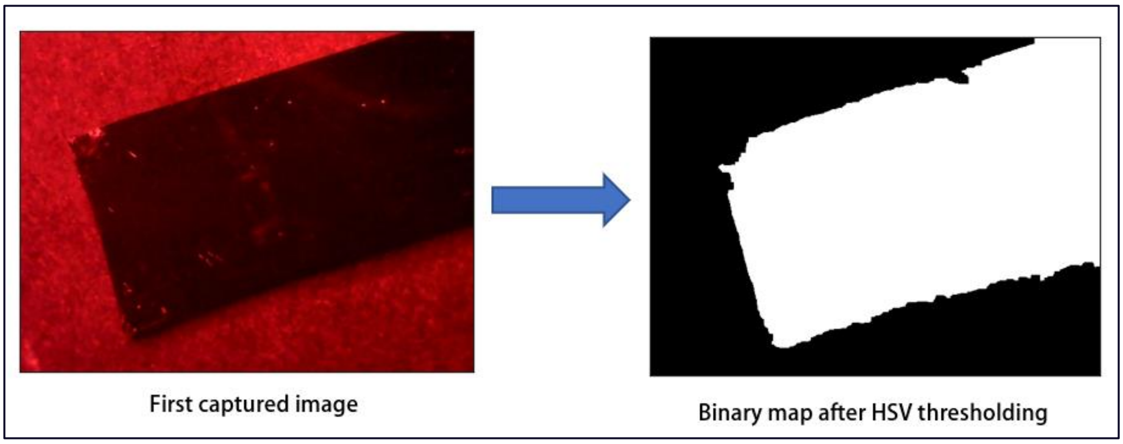



**Figure 60: Obtaining binary map (mask) from fabrication device image.**

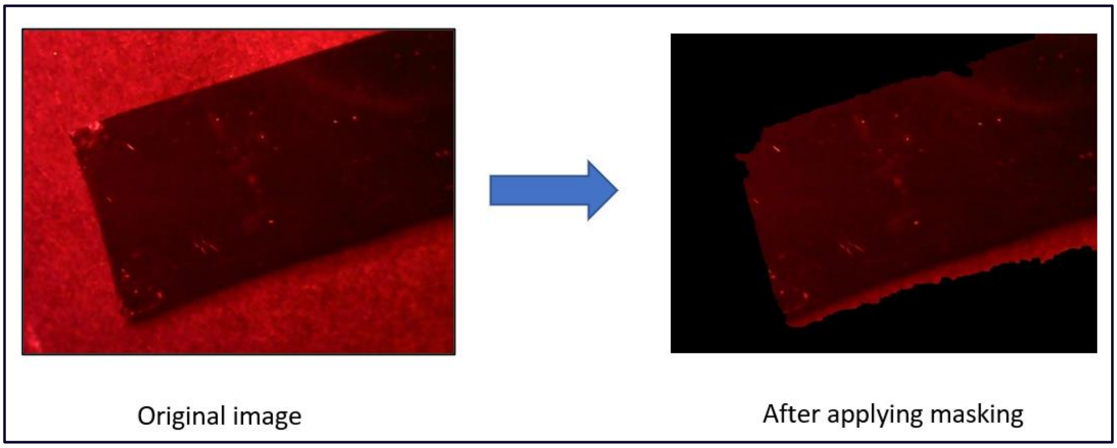



**Figure 61: Comparison between pre-masking and post-masking device image before etching (under red light).**

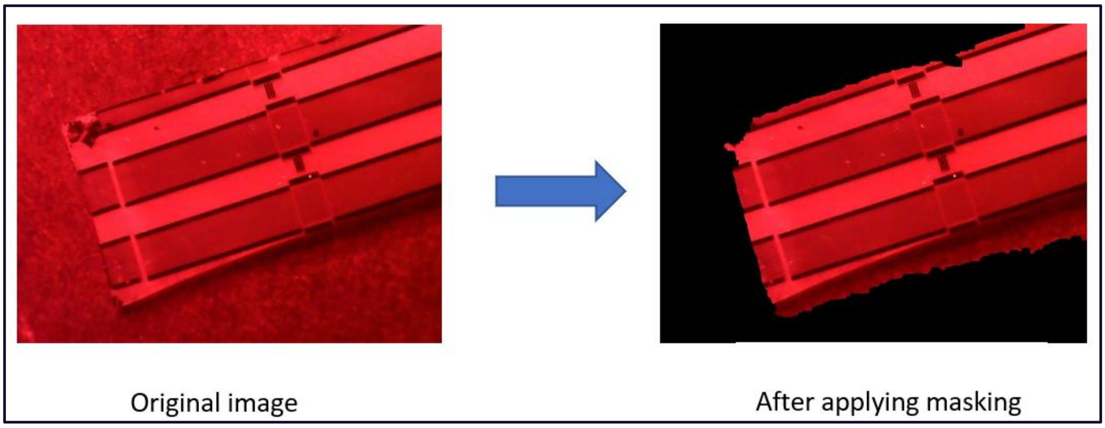



**Figure 62: Comparison between pre-masking and post-masking device image after etching (under red light).**

### *2.2.3.3 Sliding window endpoint (SWE)*

The etching endpoint can be determined by executing time-based red pixels datapoints analysis. This method uses window sliding method to detect sudden rising trend as shown in Figure 63 to compute the endpoint. Several parameters are required to manually pre-set as shown in Table 20. The detailed operation is summarized in Figure 64.

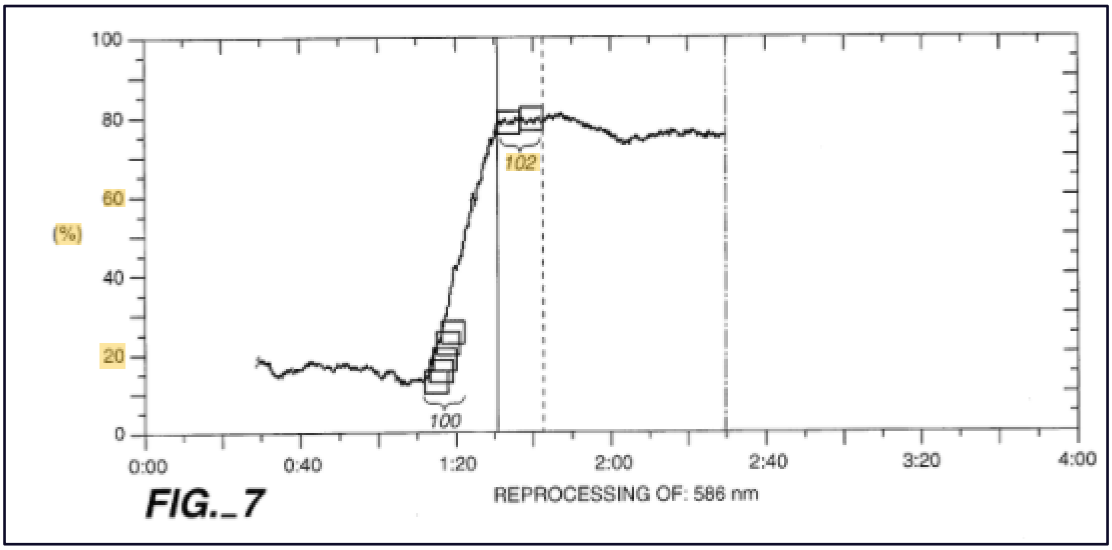



**Figure 63: Plot of reflected intensity over time with the analysis window getting passed vertically at sudden rise. [25]**

**Table 20: Considerations for the predefined parameters.**

| Parameter | Considerations | Chosen value |
|---|---|---|
| **Dimension of sliding window** | • Too small : window limit gets exceeded vertically easily, making it vulnerable to noise spike<br>• Too big : window limit hardly gets exceeded vertically. Might cause endpoint to be missed completely | • Window height = 10<br>• Window length = 3 |
| **Confidence threshold** | • Too little : false endpoint alarm triggered by noise spike from captured image<br>• Too large : threshold could not be reached and cause endpoint to be missed completely | Confidence threshold = 6 |

| | | |
|---|---|---|
| **Offset for endpoint determination** | • Too short : early endpoint determination which leads to under etching<br>• Too long : delayed endpoint determination which leads to over etching. | Offset = 1.1 |

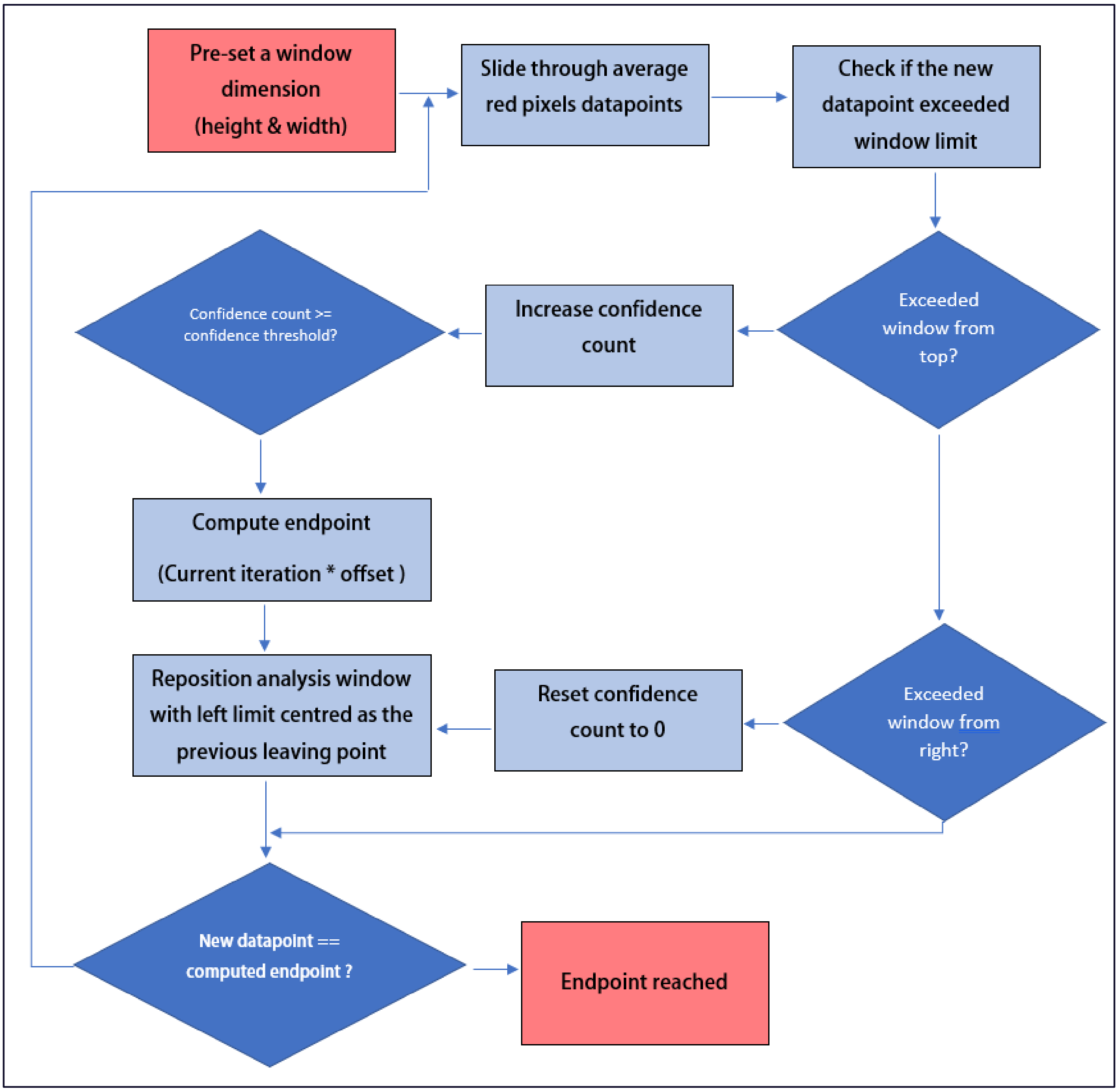



**Figure 64: Window sliding-based endpoint algorithm flowchart**

### 2.2.3.4 Derivative & inflection point-based endpoint (DIPE)

Inspired by [27], the derivative of the red pixels datapoints is computed using the polynomial Savitzky-Golay (Savgol) filter from the Scipy's signal processing library. As compared to conventional derivative operation, the derivative from Savgol filter is able to preserve the high frequency signal and reduce the impact of sudden spike noise [32]. From Figure 65, the desired etching endpoint (point 49) happens when the measured reflective intensity comes to a steady state after an abrupt change (point 43). This relation is well captured by the inflection point in the derivative datapoints (point 41). Hence, the endpoint is computed as the inflection point's index multiplied with an empirically derived offset. The detailed operation is summarized in Figure 66.

The predefined parameters required for this algorithm are tabulated in Table 21 and are obtained empirically by analysing previously captured data.

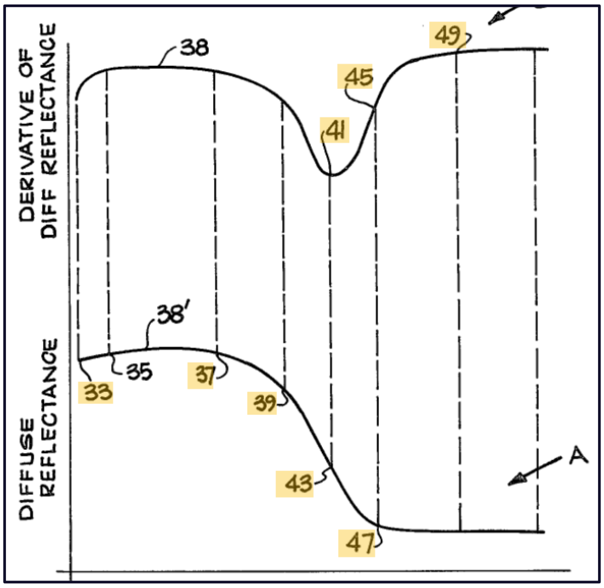



**Figure 65: The theoretical reflectance intensity graph (bottom) and the derivative (top). [27]**

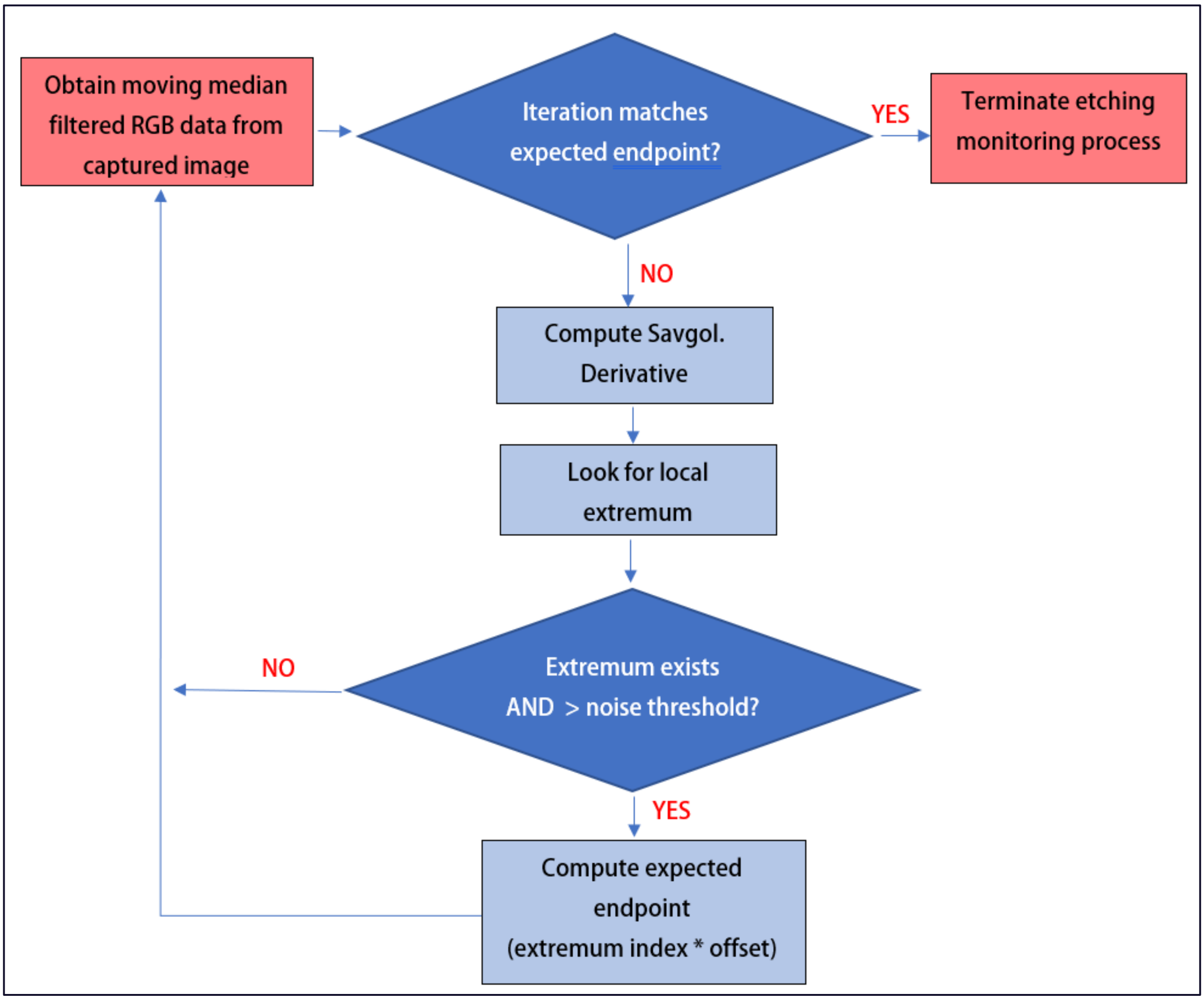



**Figure 66: DIPE algorithm flowchart**

**Table 21: Considerations for the predefined parameters.**

| Parameters | Consideration | Chosen value |
|---|---|---|
| **Inflection point threshold** | • Too low : false endpoint alarm triggered by noise spike from captured image<br>• Too high : threshold could not be reached and cause endpoint to be missed completely | Inflection point threshold = 1 |

| | | |
|---|---|---|
| **Offset amount from inflection point** | • Too short : early endpoint determination which leads to under etching<br>• Too long : delayed endpoint determination which leads to over etching. | Offset = 1.3 |

### 2.2.3.5 Automated etching endpoint determination algorithm summary

The software algorithm for the automated etching endpoint determination is summarized in Figure **67**. Once the endpoint is detected, the fabrication device will undergo the post-etching process until final completion.

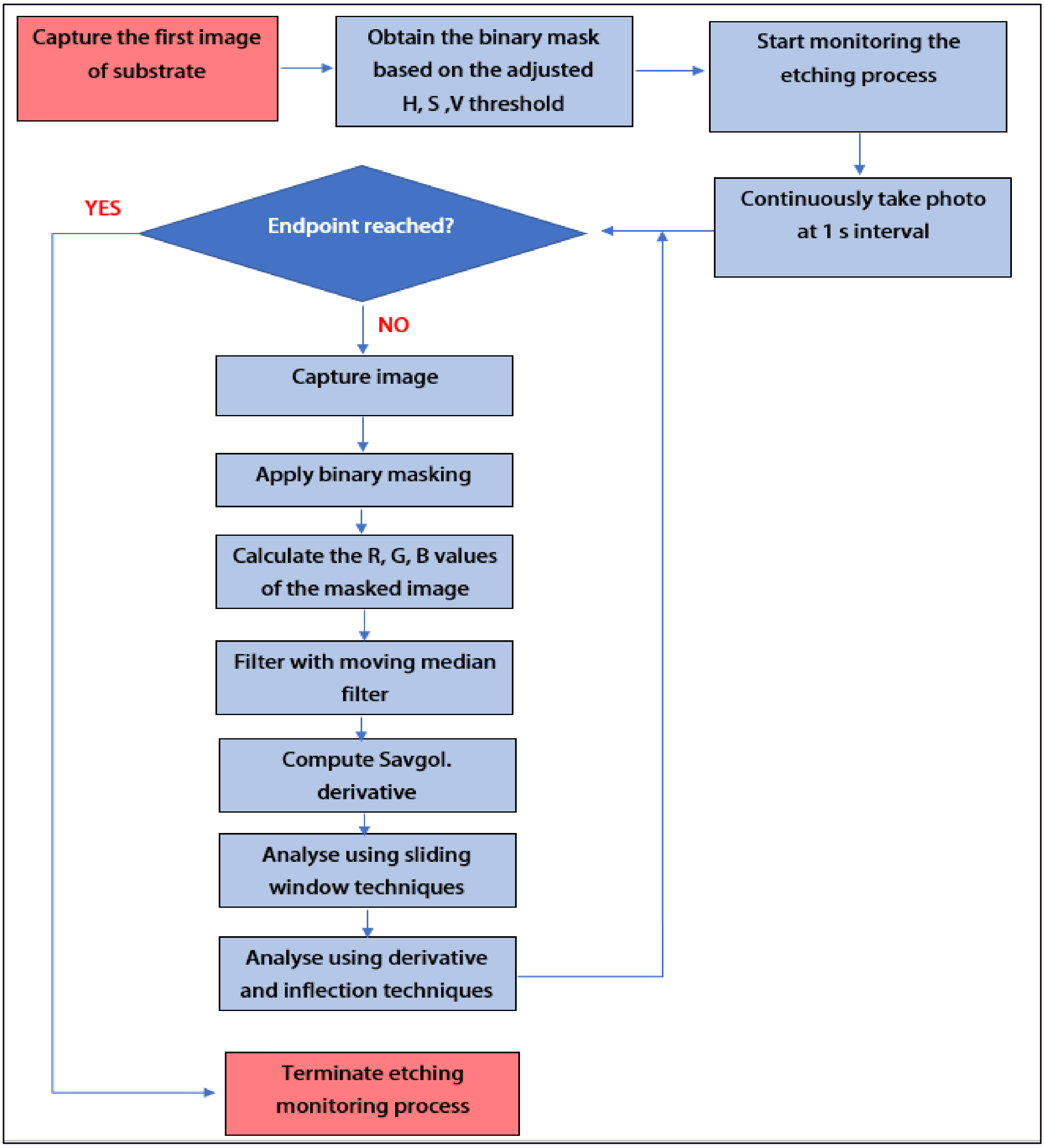



**Figure 67: High level flowcharts of the automated etching endpoint determination algorithm**

# Chapter 3

# 3. Results and Conclusions

The prototype is tested with the fabrication of LiNbO3/Al substrate. In this setup, the aluminium etchant used to etch and pattern the aluminium electrode is $HNO_3/H_3PO_4/CH_3COOH/H_2O$. Afterwards, post-etching processes such as dipping into deionised water, acetone and IPA will be carried out in an automated fashion. Besides, the feasibility of applying near-UV spectrum in etching process monitoring for multi-layered substrates is conducted by performing static analysis on the Al/ZnO/Al/SU-8/ultra-thin Si/SU-8 device due to the on-campus lab access limitation.

## 3.1 The measurements of the fabricated device

The prototype has successfully automated the fabrication of the LiNbO3/Al one port SAW resonator device with feature size of 7.5 $\mu m$ from the etching stage with automated endpoint detection to the post-development procedures. Figure 68 to Figure 71 show the feature sizes measured using the microscope. The average fabricated feature size is 8.3656 $\mu m$ as shown in Table 22 and the error comparing to the mask set is only 11.5407 %. This proves that the etching endpoint is reasonably accurate.

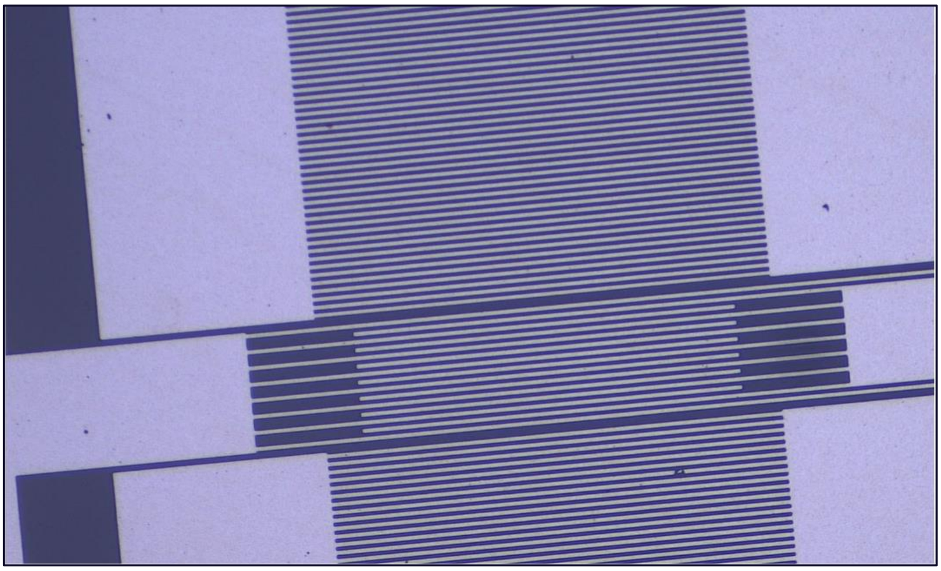


**Figure 68: The zoomed image (5x) of the fabricated LiNbO3/Al device.**

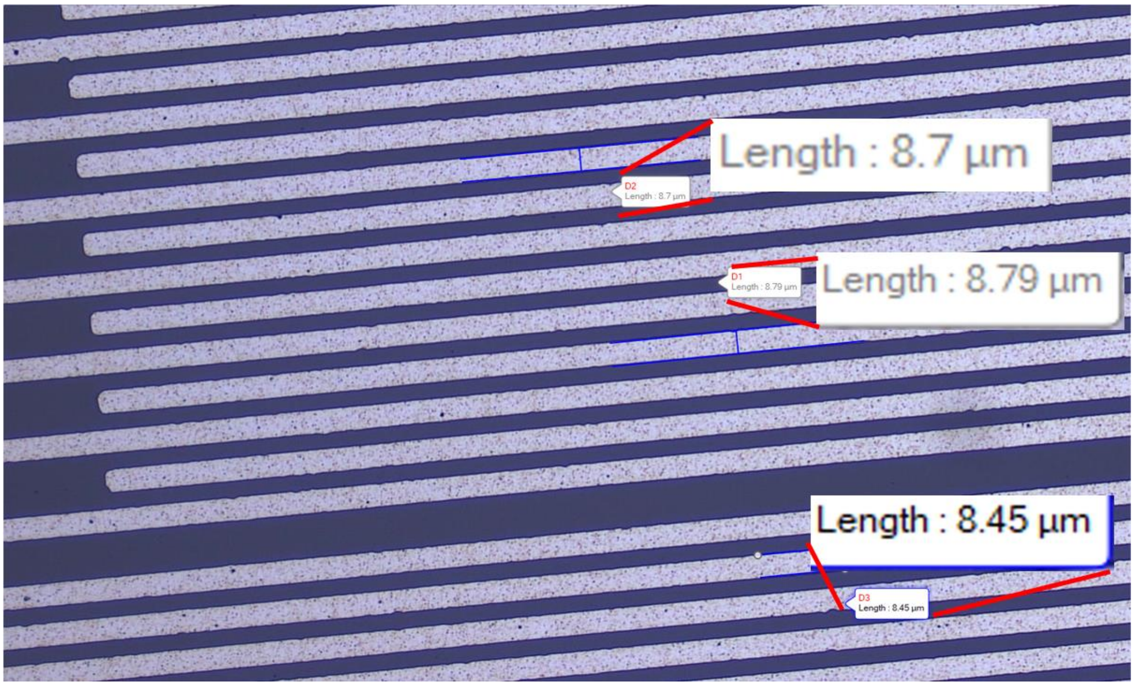



**Figure 69: The (20x) zoomed image on the IDT region of the device and the measurements.**

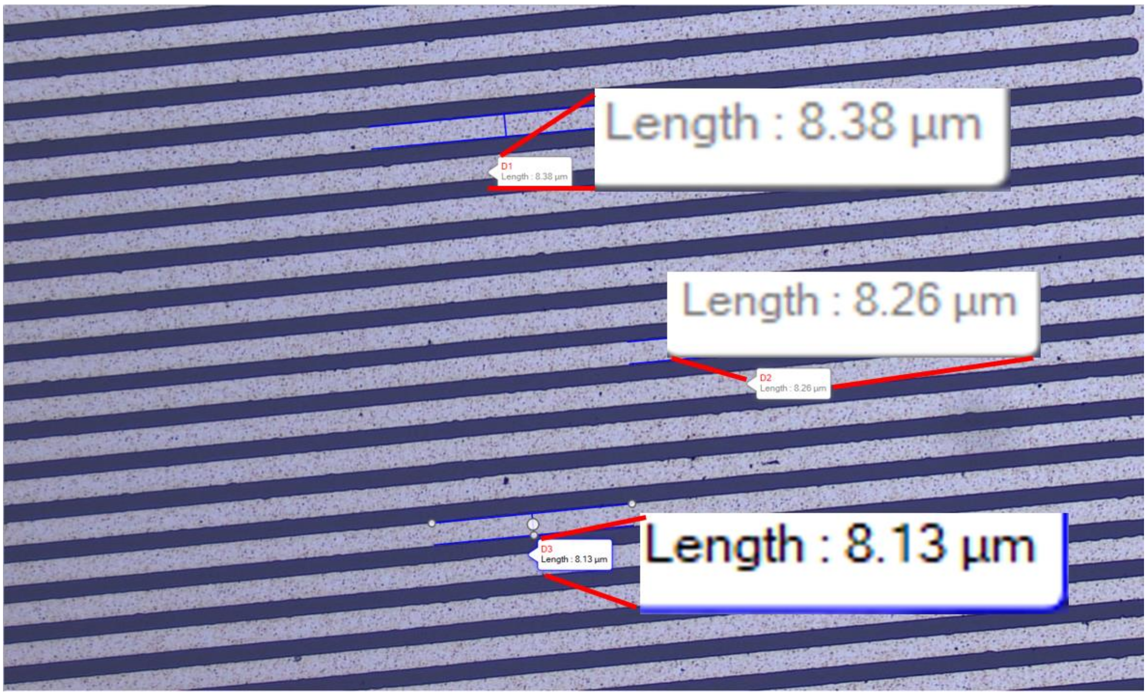



**Figure 70: The (20x) zoomed image on the top reflector region of the device and the measurements.**

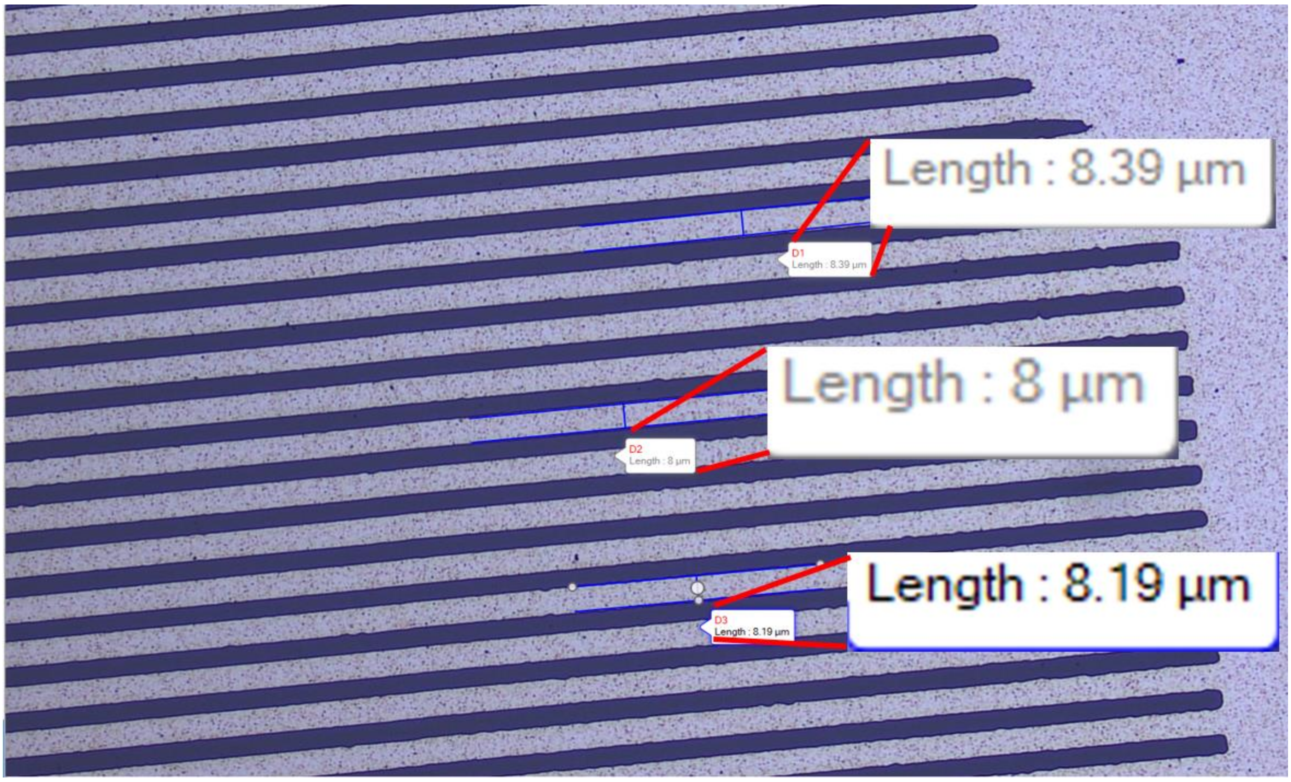



**Figure 71: The (20x) zoomed image on the bottom reflector region of the device and the measurements.**

**Table 22: Measurements of the fabricated device feature size and the error.**

| Device's region | Measurement ($\mu m$) |
|---|---|
| **Interdigital transducer (IDT)** | 8.7 |
| | 8.79 |
| | 8.45 |
| **Top reflector** | 8.38 |
| | 8.26 |
| | 8.13 |
| **Bottom reflector** | 8.39 |
| | 8.00 |
| | 8.19 |

| Average width ($\mu m$) | 8.3656 |
|---|---|
| Percentage of error (%) | $\frac{8.3656-7.5}{7.5} \times 100 = \mathbf{11.5407\%}$ |

To prove the features are etched completely, the resistance of the device is measured using a multimeter across IDTs bond pads as shown in Figure 72. Large resistance is observed, and it proves that no short circuit exists on the fabricated device.

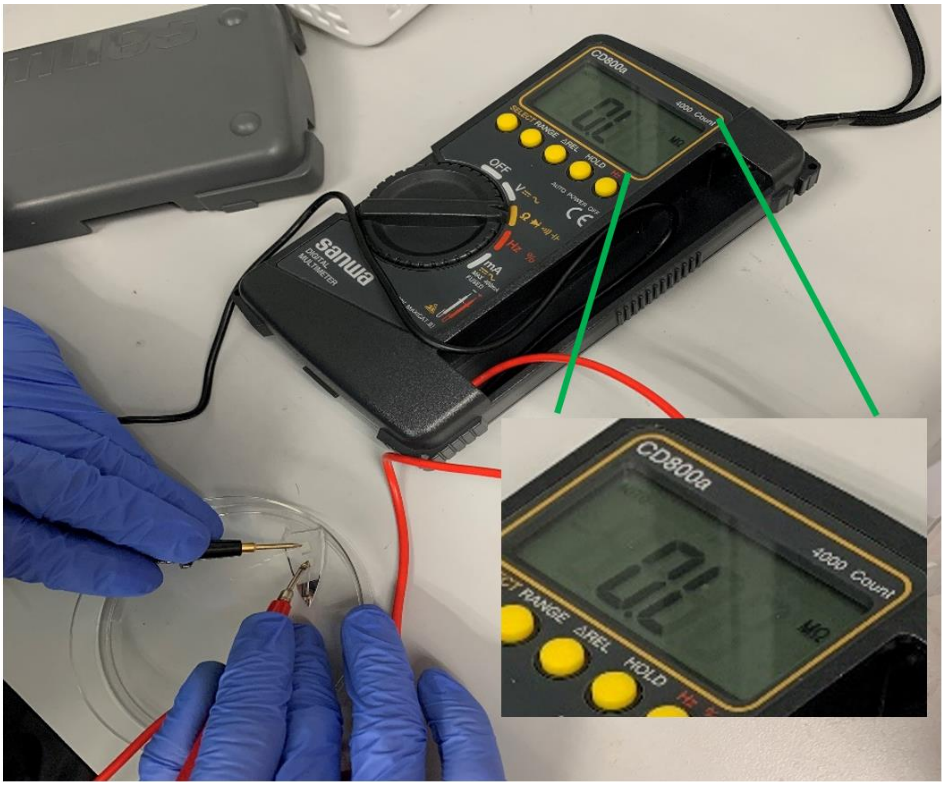



**Figure 72: Large resistance measured across the bond pads of the fabricated device using multimeter.**

## 3.2 The response of the fabricated device

Figure 73 shows the response measurement setup. The S11 plot in Figure 74 shows a dip at 129.428 MHz with magnitude of -5.7148 dB showing that the fabricated device's resonant frequency is very close to the designed value, 130 MHz. This confirms the etching endpoint determination from the prototype is optimum as the response matches the designed parameters quite well.

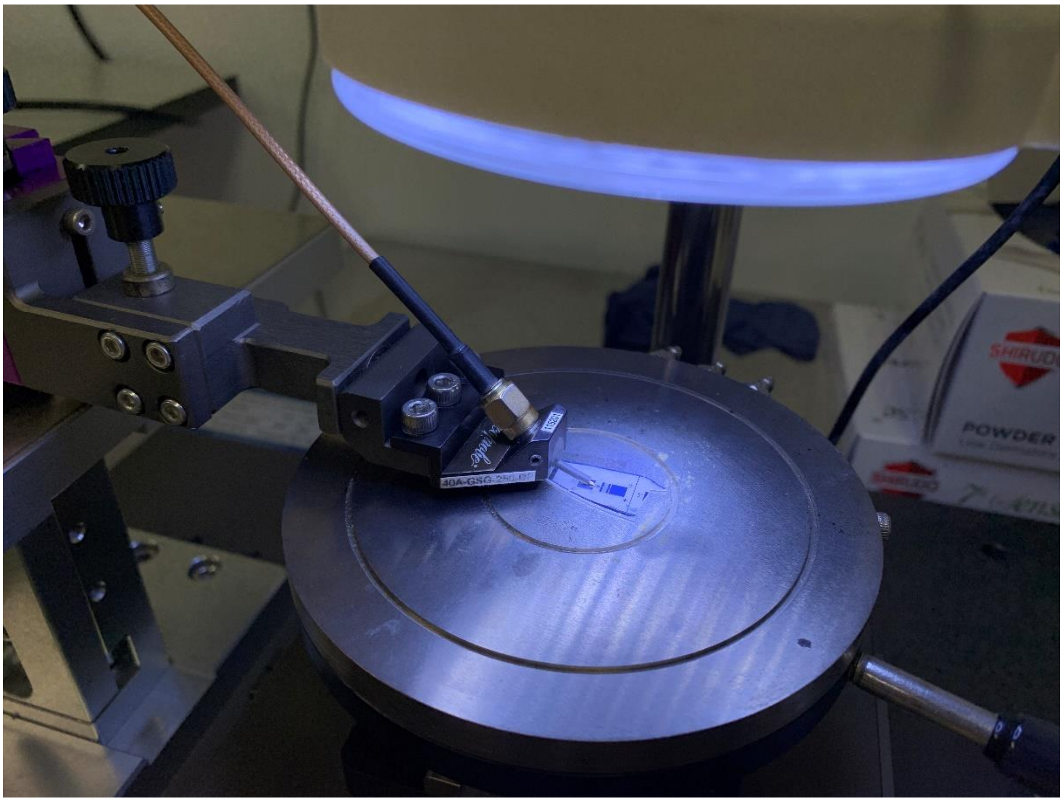



**Figure 73: Device response is measured by the microprobe tip connected to the network analyser.**

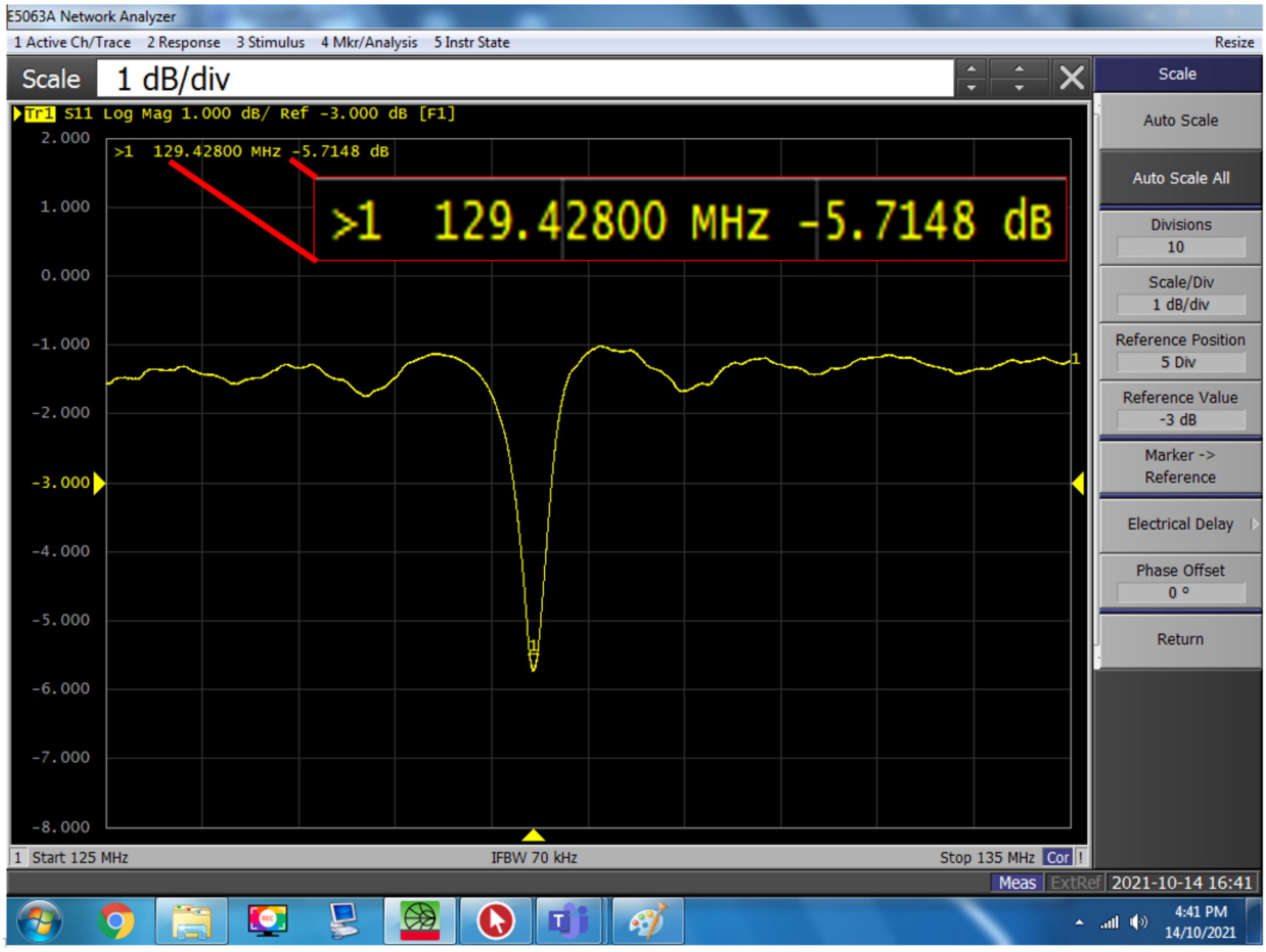



**Figure 74: The S11 plot of the fabricated one-port SAW resonator device.**

## 3.3 The photo pixels analysis and endpoint determination

Figure 75 to Figure 77 show that there is an obvious visual difference from the images captured by the prototype under red light throughout the process. With these images, the mask is applied and the average red pixels datapoints are computed and smoothen out by a moving median filter. The average red pixels datapoints are shown in Figure 78. Concurrently, the derivatives of these red pixel datapoints are also computed using normal derivative (numpy) and Savgol derivative as shown in Figure 79. Noticing that the Savgol derivative is much smoother than the normal derivative and this is particularly important for discrete datapoints computation, which sudden step noise would result in enormous magnitude in derivative. The endpoint is computed using both DIPE and SWE.

For DIPE, as shown in Figure 80, the Savgol derivative inflection point at iteration 98 exceeded the threshold and with that, the expected endpoint is computed by multiplying the inflection point's index with 1.3 offset, resulting in the expected endpoint to be at iteration 127.

Comparatively, from Figure 78, the SWE has missed the endpoint completely as the confidence threshold is not reached in this scenario. This shows that DIPE outperforms SWE in terms of robustness in endpoint determination.

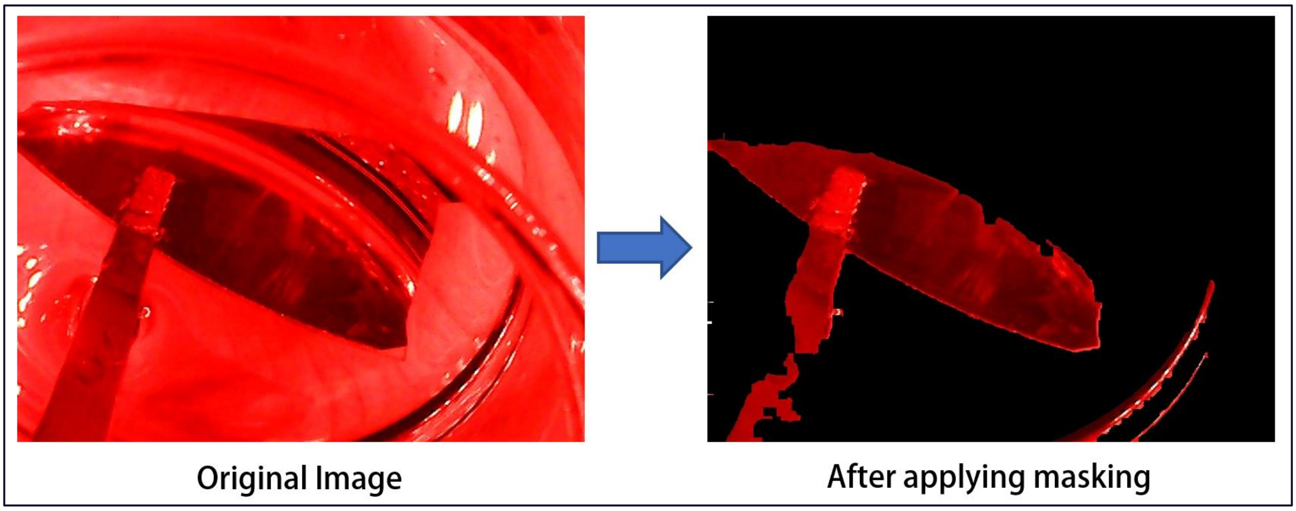



**Figure 75: Comparison between pre-masking and post-masking device image before etching.**

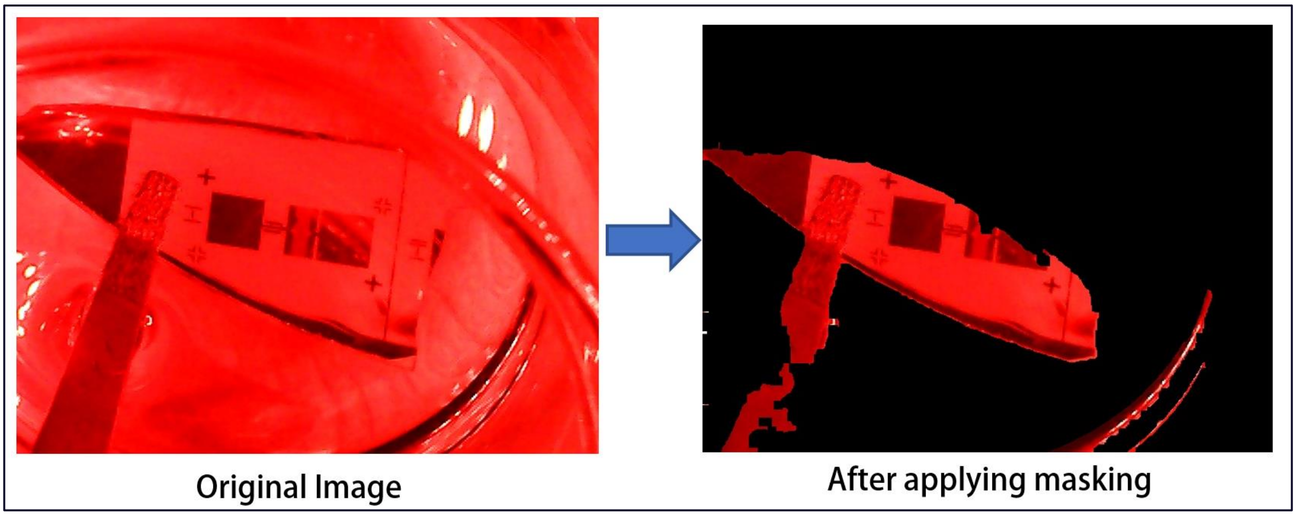



**Figure 76: Comparison between pre-masking and post-masking device image after etching.**

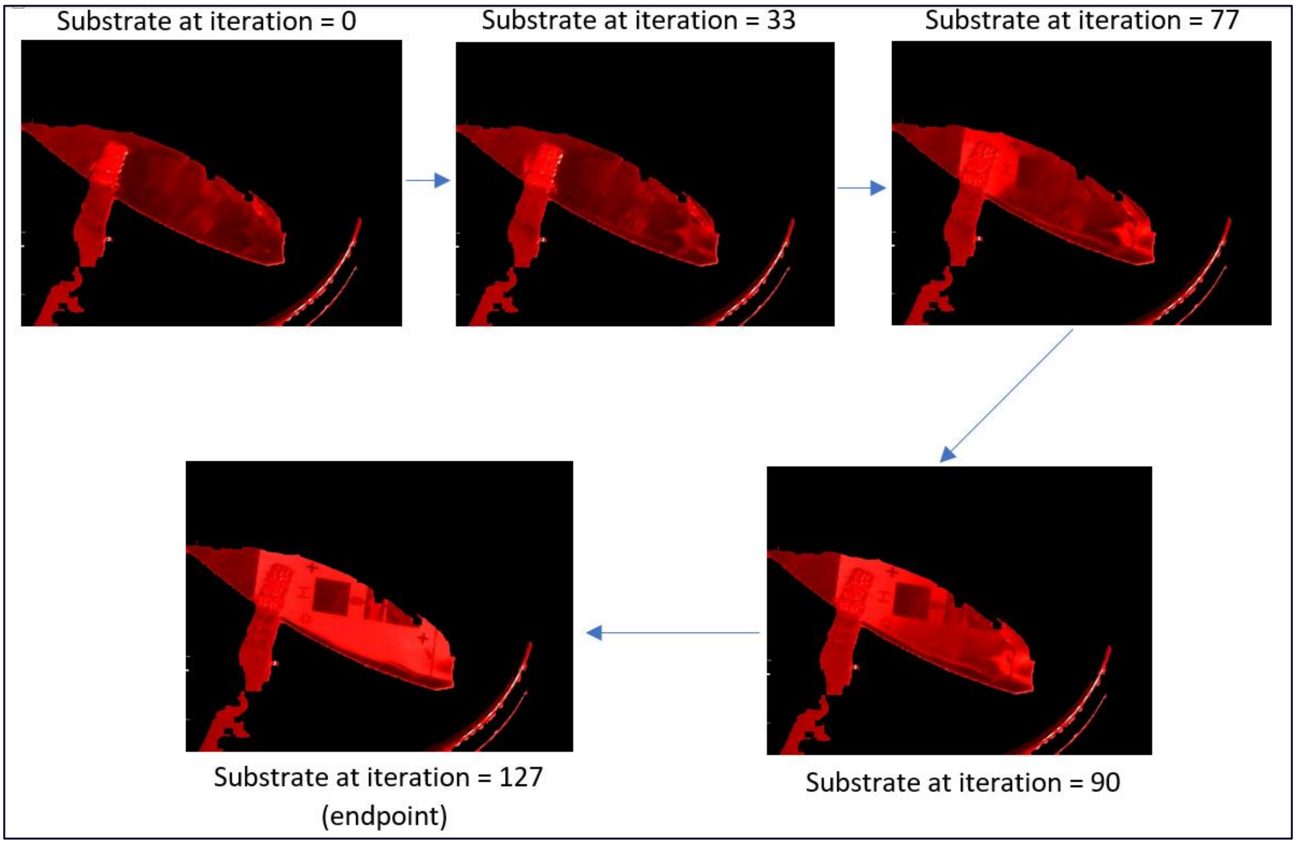



**Figure 77: Images of "LiNbO3-Al" substrates at different iterations.**

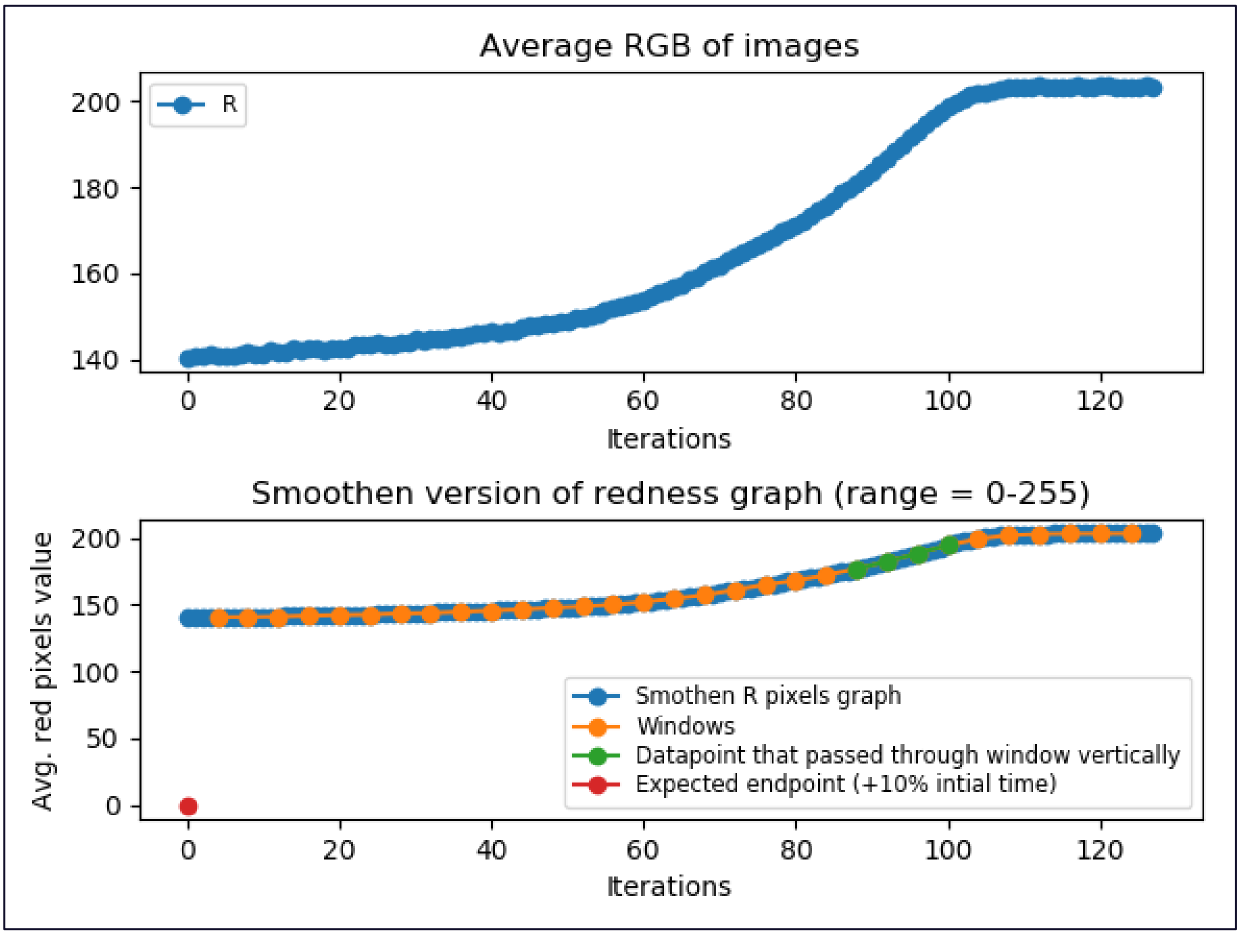



**Figure 78: Showing the moving median filtered red pixels datapoints (top) and the endpoint determination using SWE (bottom).**

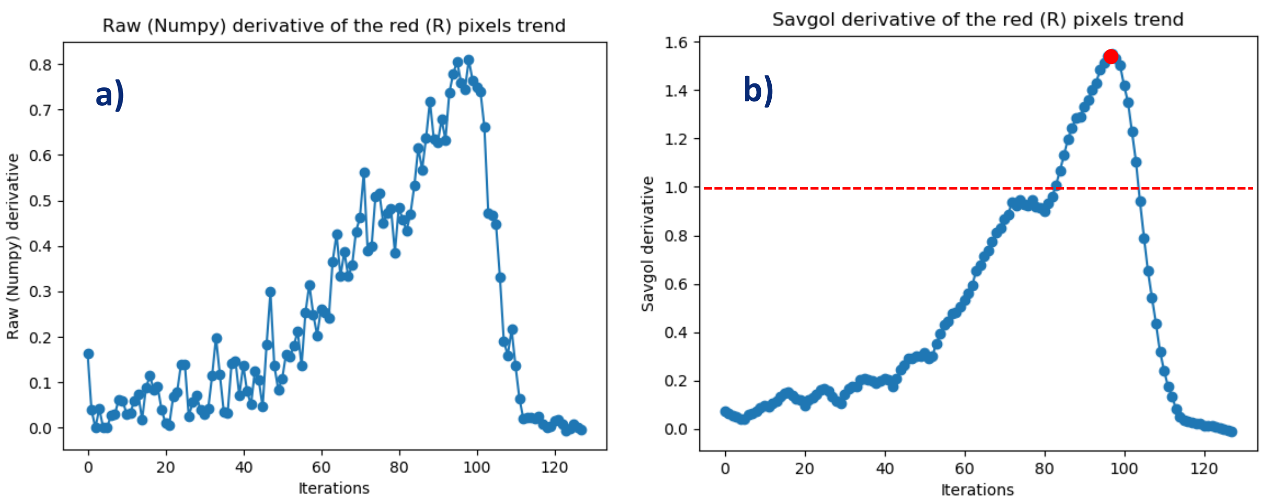



**Figure 79: The raw (a) and Savgol derivative (b) of the red pixels datapoints. The inflection point threshold is represented as the horizontal red line and the red marker is the inflection point.**

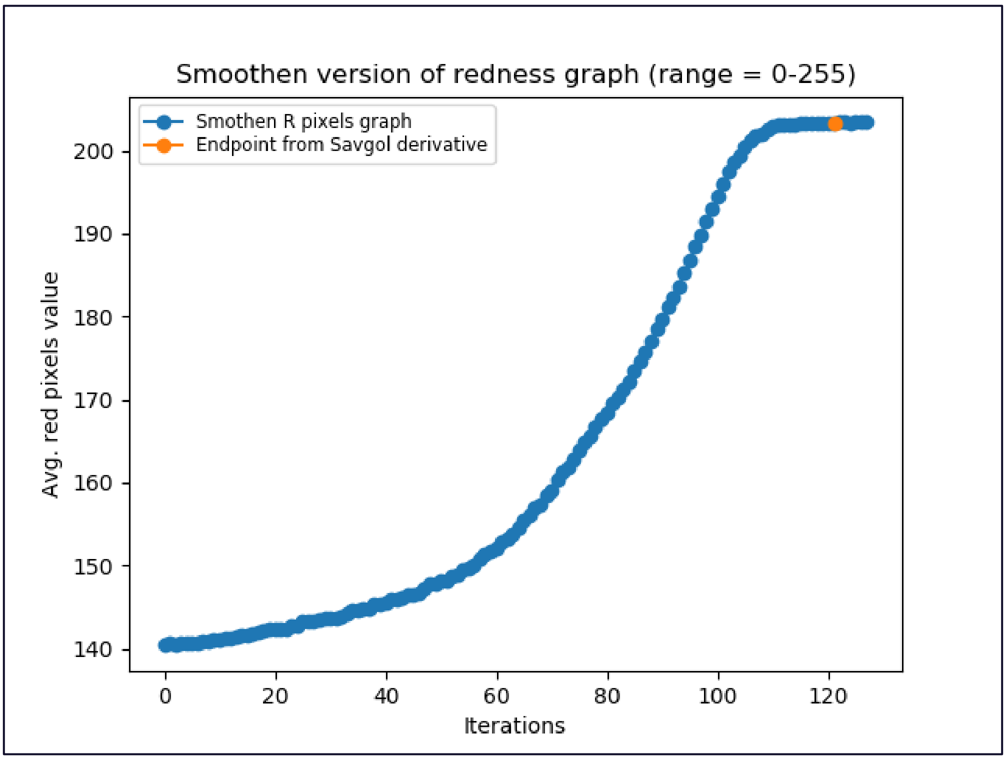



**Figure 80: The endpoint (orange) determined by DIPE.**

## 3.4 Near-UV spectrum imaging chamber approach on transparent multi-layered device

Due to lab access limitation, the fully fabricated Si-SU-8-Al-ZnO-Al device is examined stationarily under visible and near-UV to examine the potential of applying the same methodology with arbitrary spectrum of light for fabrication of transparent multi-layered devices. Figure 81 shows the Al electrodes on the device are almost invisible due to the double Al layers and the high transparency of ZnO in visible range. A static analysis is carried out on the images under visible and near-UV range by sliding a cropping window from the non-Al region (ZnO region) to the Al region that contains IDT as shown in Figure 82. As expected, Figure 83(b) shows that under near-UV spectrum, the pixels form reflective intensity measured shows a significant difference between the ZnO region and the Al electrode region. This observation matches the theory as ZnO has high absorbance as compared to Al that has high reflectivity in the near-UV range, resulting in brighter image at Al as compared to ZnO. Comparatively, the analysis in visible range as shown in Figure 83 provides no info in differentiating the Al region and the ZnO region. This is because ZnO is highly transparent and the double Al structure makes the reflected images captured to show no difference .

Based on this clear difference of near-UV imaging between the Al and ZnO region on the Si-SU-8-Al-ZnO-Al device, this proves the feasibility of integrating a near-UV imaging chamber into the current prototype with the same endpoint detection algorithm.

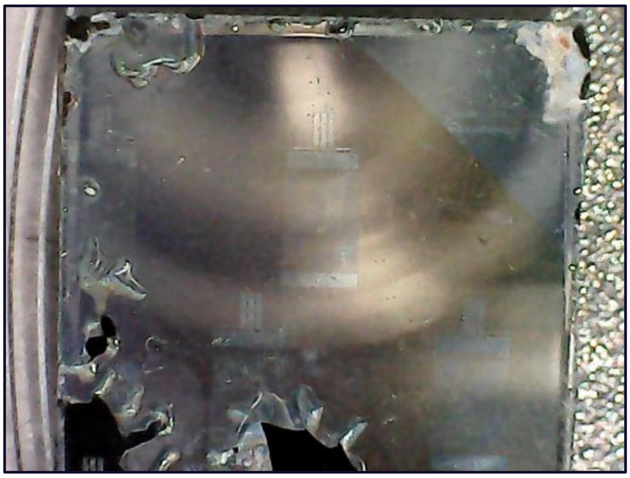


**Figure 81: Image of "Si-SU-8-Al-ZnO-Al" substrate under white light.**

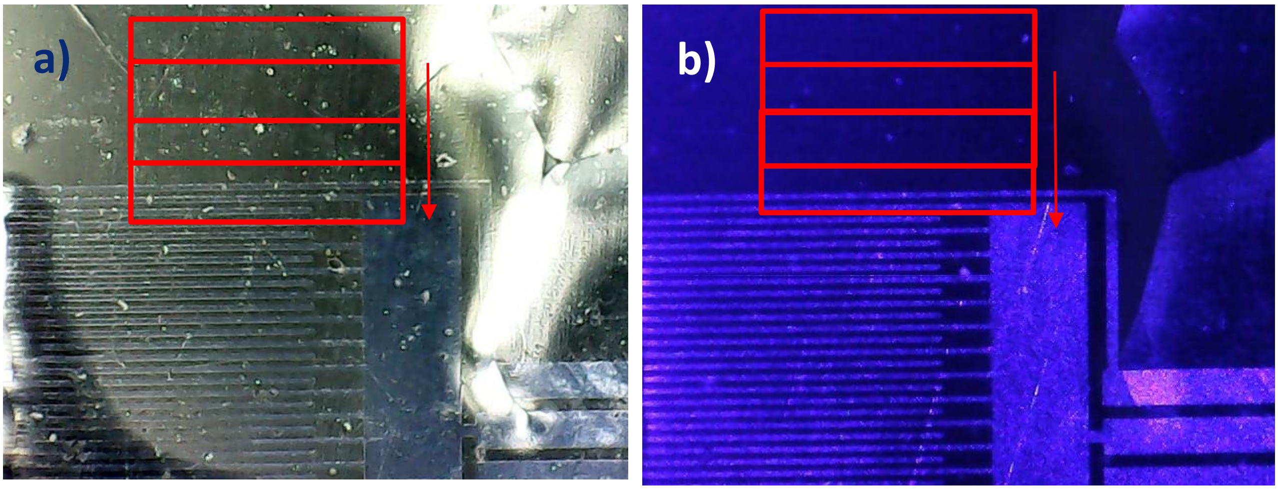
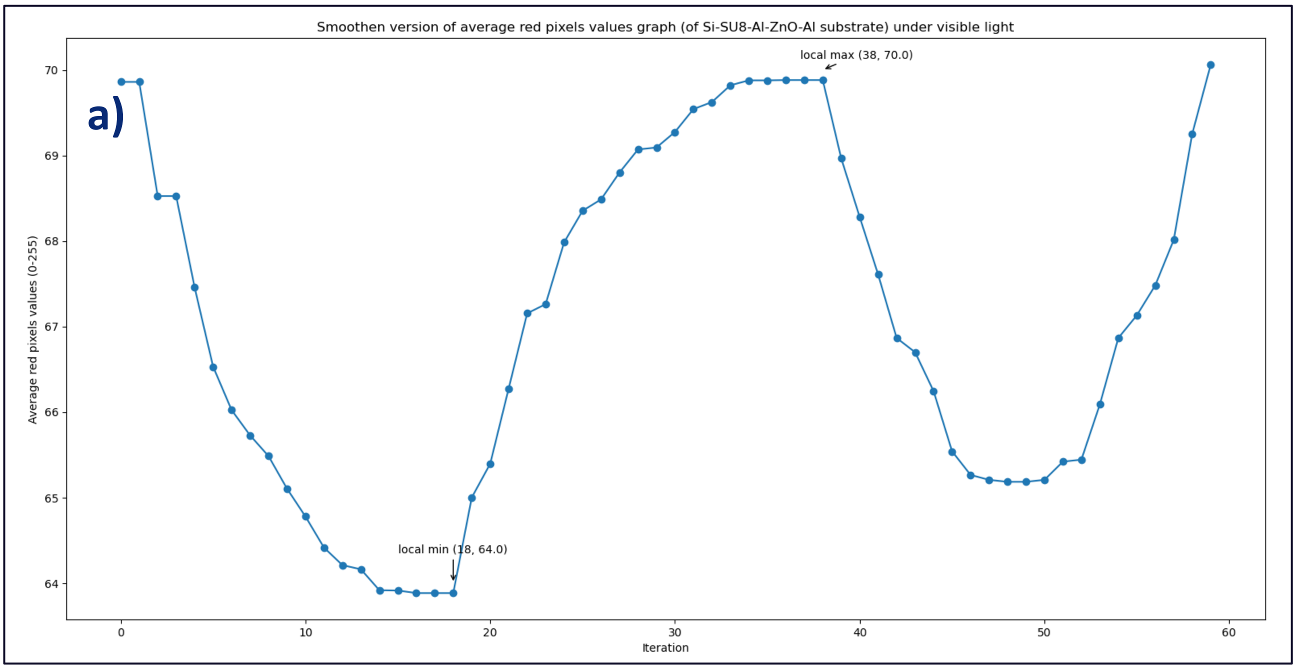



**Figure 82: Static analysis of substrate Si-SU-8-Al-ZnO-Al under visible (a) and near-UV (b).**



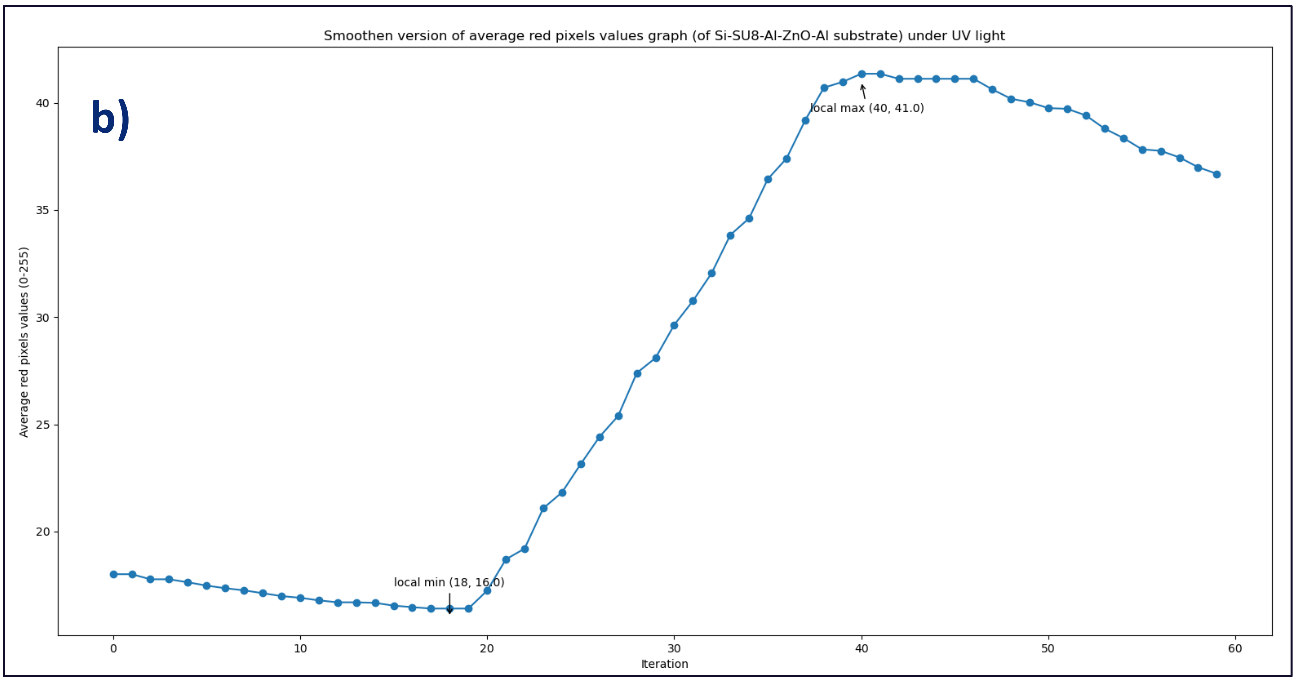




**Figure 83: The red pixels datapoints from the cropped images of Si-SU-8-Al-ZnO-Al under white light (a) and near-UV (b) obtained by sliding the window from top to bottom.**

However, in the etching of IDT fabrication, thin photoresists (Pr) layer would be covering the Al region. Hence, the following static analysis of comparing the visual difference of ZnO with and without photoresists is carried out as shown in Figure 84. With that, the average pixel value from the cropping window for each substrate is computed as shown in Figure 85. Considering, the possible range of red pixels is 0-255, the difference of 4.49 between “ZnO” and “ZnO-Pr” substrates shows that the visual difference under near-UV range is not significantly affected by the thin Pr layer. This observation further proves the feasibility of near-UV imaging chamber in fabricating the multi-layered device.

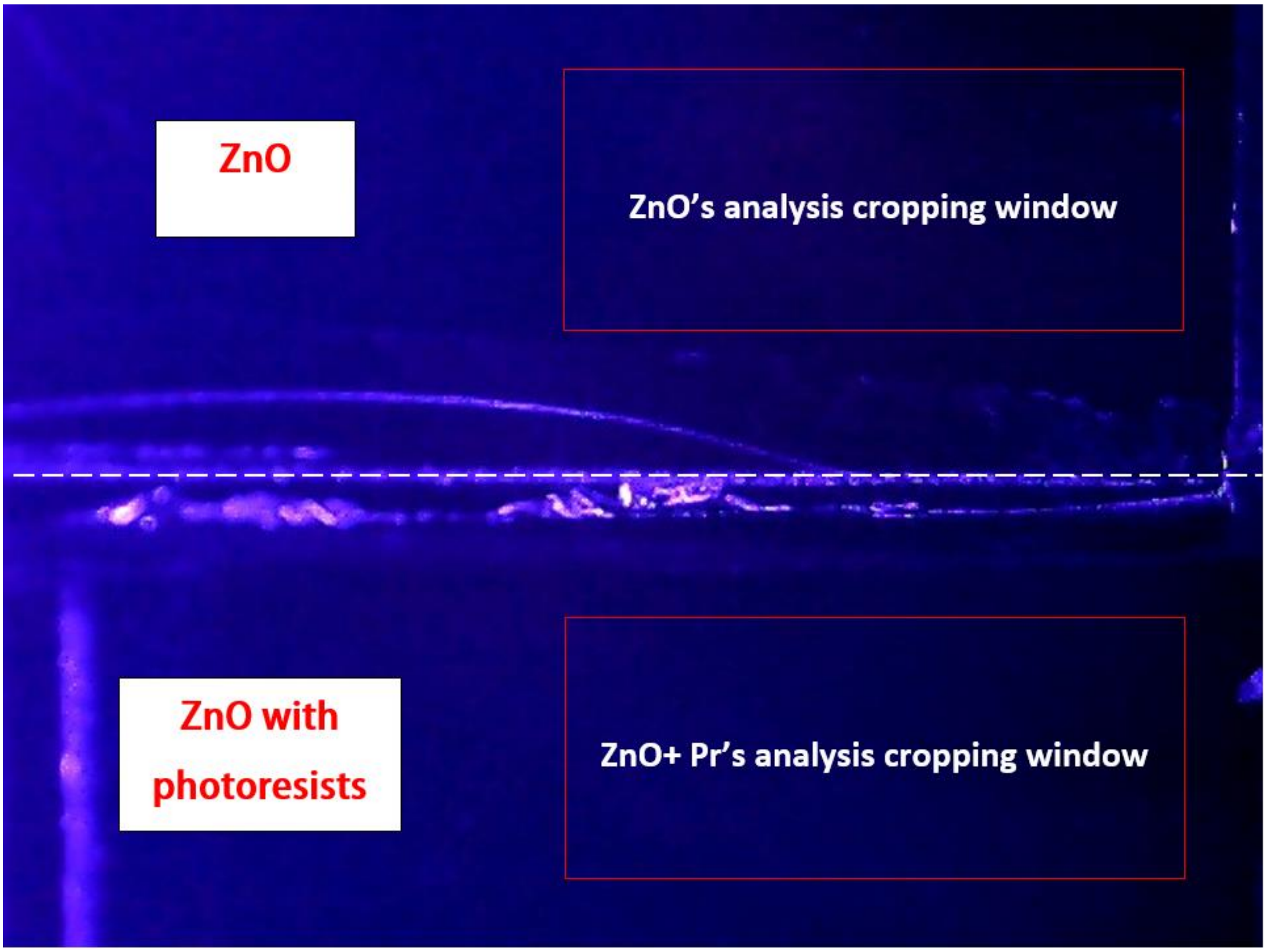



**Figure 84: Static analysis comparing "ZnO" and "ZnO-Pr" under near-UV.**

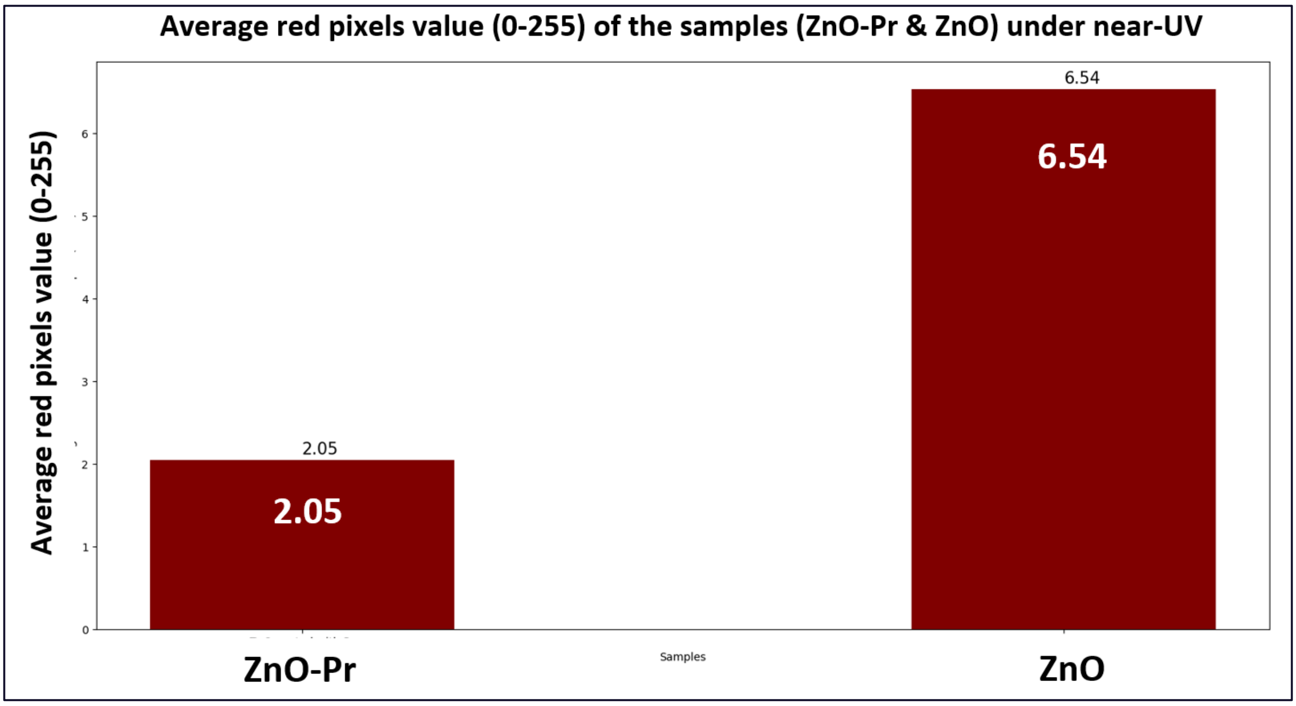



**Figure 85: Average pixels comparison (ZnO & ZnO-Pr)**

## 3.5 The ZnO-safe etchant and the implications

To verify the robustness of the algorithm, same fabrication steps have been taken on the LiNbO3/Al substrate with a slight change of the etchant to $KOH/K_3Fe(CN)_6/H_20$ etchant. This testcase is inspired by the need of ZnO-safe etchant to form the Al electrodes while not damaging the ZnO layer of the Si-SU-8-Al-ZnO-Al device. As shown in Figure 86 and Figure 87, the captured visual changes have a similar trend as before however the rate of the etching reaction is significantly lower. This low reaction rate is perfectly reflected on the derivative of the red pixels datapoints as shown in Figure 88. In comparison, the peak magnitude derivative in the case of $HNO_3/H_3PO_4/CH_3COOH/H_2O$ is 1.6 as shown in Figure 79 (b) whereas in this case, the derivative is only 0.3. This difference of derivative magnitude resulting from weaker etchant might signify the need of recalibration of the endpoint threshold for different etching recipes.

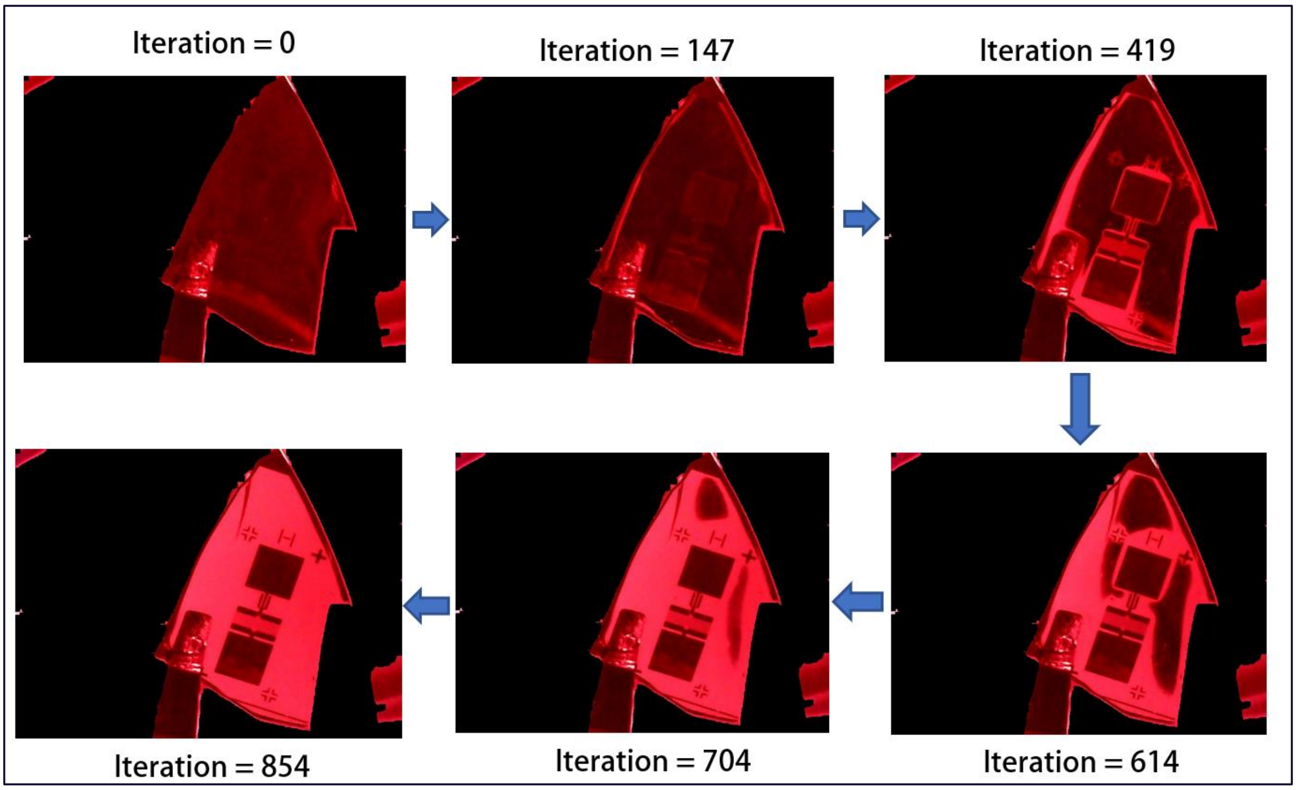



**Figure 86: Images of "LiNbO3-Al" at different iterations with the ZnO-safe etchant.**

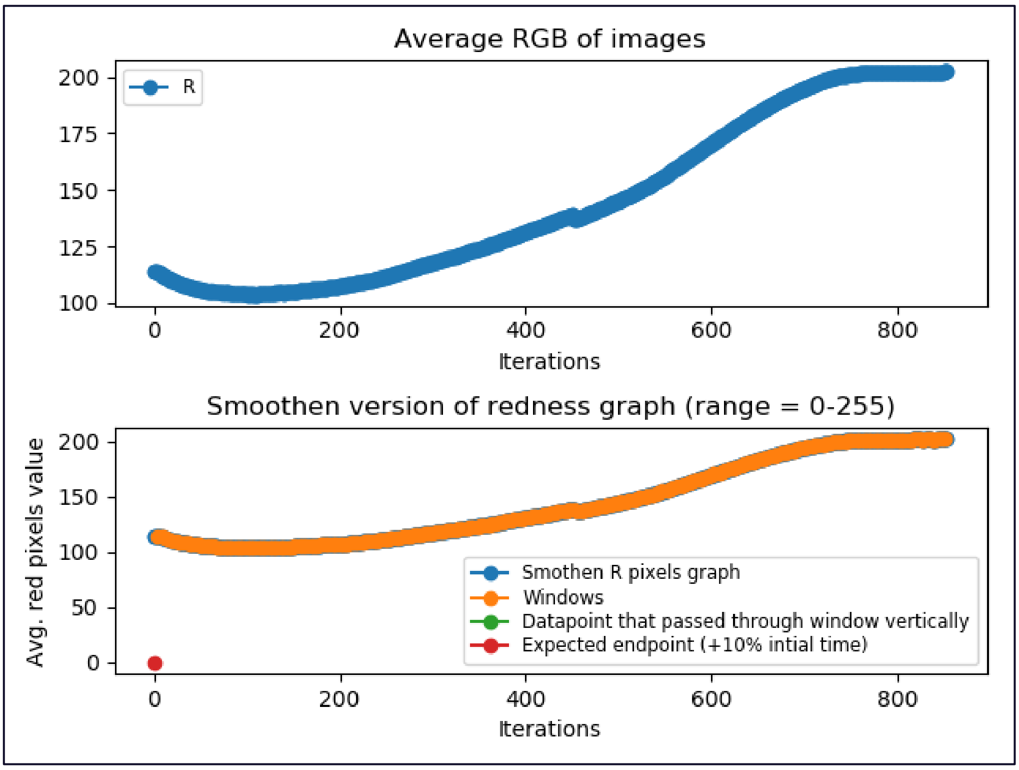



**Figure 87: Showing the moving median filtered red pixels datapoints (top) and the endpoint determination using SWE (bottom).**

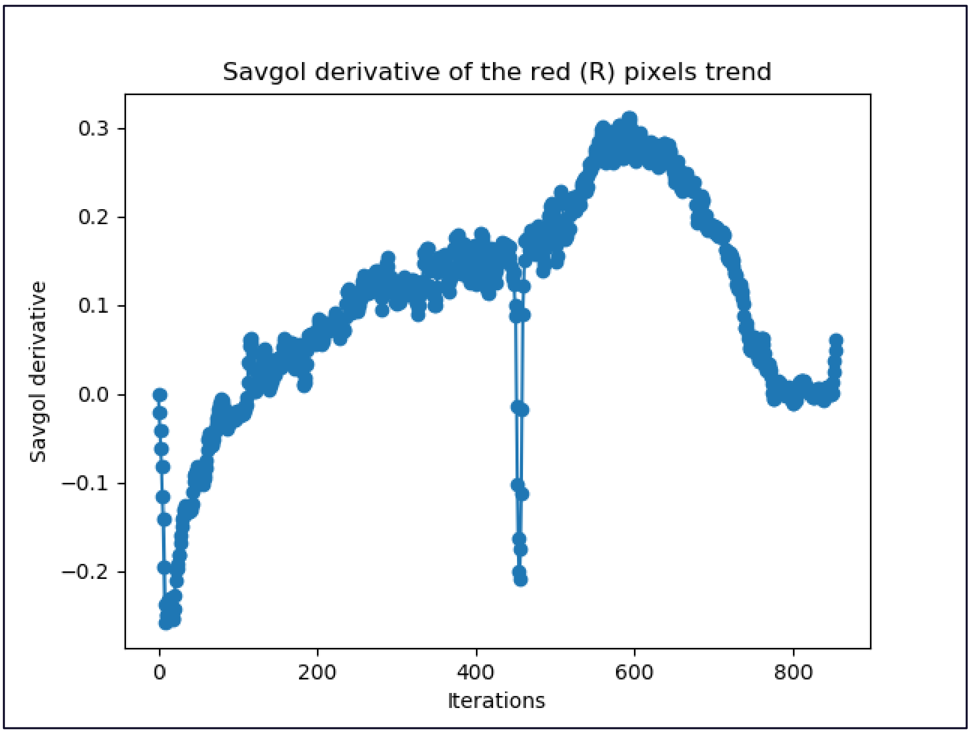



**Figure 88: The Savgol derivative of the red pixels datapoints.**

# 3.6 Conclusion

The automated endpoint determined by DIPE of the prototype has proven to be reasonably accurate as the fabricated feature dimension shows only 11.5407% of error and the resonant frequency of the device is 129.428 MHz which is considerably close to the designed value. Also, the significant measurable difference between the Al and ZnO region under near-UV illumination has shown the potential of applying a near-UV imaging chamber for transparent multi-layered structure fabrication monitoring. The effect of weaker etchant is also well-captured by this algorithm, showing the possibility of requiring slight threshold calibration for different etching solutions to achieve the accurate endpoint detection.

# Chapter 4

# 4. Limitations and future research

1. Near-UV spectrum imaging chamber is only tested with static analysis on fully fabricated devices. The practicality should be further verified in the actual substrate etching monitoring process.

2. Use of near-UV spectrum is investigated in differentiating ZnO and Al for etching monitoring. The potential implications of near-UV exposure on hardbake photoresist should be researched.

3. Endpoint determination algorithm is constructed based on a limited number of samples and less variation. More test scenarios such as varying substrate material, etchant and etchant temperature should be carried out to verify the robustness.

4. The current circuitry is constructed on terminal blocks and could be subjected to loose connections after time. Circuit fabrication on PCB for proper prototyping should be performed.

Word count: 6564

# References


[1] A. Jiang *et al.*, "Transparent Capacitive-Type Fingerprint Sensing Based on Zinc Oxide Thin-Film Transistors," *IEEE Electron Device Letters,* vol. 40, no. 3, pp. 403-406, 2019, doi: 10.1109/LED.2019.2895830. [Q1]

[2] W. S. Lee *et al.*, "Multiaxial and Transparent Strain Sensors Based on Synergetically Reinforced and Orthogonally Cracked Hetero-Nanocrystal Solids," *Advanced functional materials,* vol. 29, no. 4, pp. 1806714-n/a, 2019, doi: 10.1002/adfm.201806714. [Q1]

[3] H. Jin *et al.*, "Flexible surface acoustic wave resonators built on disposable plastic film for electronics and lab-on-a-chip applications," *Scientific Reports,* vol. 3, 2013. [Q1]

[4] Y.-h. Zhang, S. A. Campbell, L. Zhang, and S. Karthikeyan, "Sandwich structure based on back-side etching silicon (100) wafers for flexible electronic technology," *Microsystem Technologies,* vol. 23, no. 3, pp. 739-743, 2017/03/01 2017, doi: 10.1007/s00542-015-2737-7. [Q3]

[5] J. Chen *et al.*, "Bendable transparent ZnO thin film surface acoustic wave strain sensors on ultra-thin flexible glass substrates," *Journal of Materials Chemistry C,* 10.1039/C4TC01307G vol. 2, no. 43, pp. 9109-9114, 2014, doi: 10.1039/C4TC01307G. [Q1]

[6] R. Tao *et al.*, "Thin film flexible/bendable acoustic wave devices: Evolution, hybridization and decoupling of multiple acoustic wave modes," *Surface and Coatings Technology,* vol. 357, pp. 587-594, 2019/01/15/ 2019, doi: https://doi.org/10.1016/j.surfcoat.2018.10.042.

[7] S. Paul, M. F. Hossain, M. H. Islam, M. A. Raihan, and S. Chakladar, "Optical properties of ZnO thin films prepared by automatic sol-gel method," in *2013 International Conference on Electrical Information and Communication Technology (EICT)*, 13-15 Feb. 2014 2014, pp. 1-4, doi: 10.1109/EICT.2014.6777906.

[8] H. Khachatryan, S.-N. Lee, K.-B. Kim, and M. Kim, "Deposition of Al Thin Film on Steel Substrate: The Role of Thickness on Crystallization and Grain Growth," *Metals,* vol. 9, no. 1, 2019, doi: 10.3390/met9010012. [Q2]

[9] C.-C. E. Lan, Y.-T. Hung, A.-H. Fang, and W. Ching-Shuang, "Effects of irradiance on UVA-induced skin aging," *J Dermatol Sci,* vol. 94, no. 1, pp. 220-228, 2019, doi: 10.1016/j.jdermsci.2019.03.005. [Q1]

[10] Q. Gueranger *et al.*, "Protein Oxidation and DNA Repair Inhibition by 6-Thioguanine and UVA Radiation," *J Invest Dermatol,* vol. 134, no. 5, pp. 1408-1417, 2014, doi: 10.1038/jid.2013.509. [Q1]

[11] F. Di Pietrantonio, D. Cannatà, and M. Benetti, "Chapter 8 - Biosensor technologies based on nanomaterials," in *Functional Nanostructured Interfaces for Environmental and Biomedical Applications*, V. Dinca and M. P. Suchea Eds.: Elsevier, 2019, pp. 181-242.

[12] L. Zou, C. McLeod, and M. R. Bahmanyar, "Wireless Interrogation of Implantable SAW Sensors," *IEEE Trans Biomed Eng,* vol. 67, no. 5, pp. 1409-1417, 2020, doi: 10.1109/TBME.2019.2937224. [Q2]

[13] D. Psychogiou and R. Gomez-Garcia, "Symmetrical Quasi-Reflectionless SAW-Based Bandpass Filters With Tunable Bandwidth," *IEEE microwave and wireless components letters,* vol. 29, no. 7, pp. 447-449, 2019, doi: 10.1109/LMWC.2019.2918413. [Q2]

[14] X. Y. Du *et al.*, "Surface acoustic wave induced streaming and pumping in 128° Y-cut LiNbO3for microfluidic applications," *Journal of Micromechanics and Microengineering,* vol. 19, no. 3, p. 035016, 2009/02/18 2009, doi: 10.1088/0960-1317/19/3/035016.

[15] J. Zhou *et al.*, "Characterisation of aluminium nitride films and surface acoustic wave devices for microfluidic applications," *Sensors and Actuators B: Chemical,* vol. 202, pp. 984-992, 2014/10/31/ 2014, doi: https://doi.org/10.1016/j.snb.2014.05.066. [Q1]
[16] Y. Liu *et al.*, "Annealing Effect on Structural, Functional, and Device Properties of Flexible ZnO Acoustic Wave Sensors Based on Commercially Available Al Foil," *IEEE Transactions on Electron Devices,* Article vol. 63, no. 11, pp. 4535-4541, 2016, Art no. 7588108, doi: 10.1109/TED.2016.2610466. [Q2]
[17] M. P. Aleksandrova, G. D. Kolev, I. N. Cholakova, and G. H. Dobrikov, "Photolithography versus lift off process for patterning of sputtered indium tin oxide for flexible displays," 2012.
[18] P. D. Cabral *et al.*, "Clean-Room Lithographical Processes for the Fabrication of Graphene Biosensors," *Materials,* vol. 13, no. 24, p. 5728, 2020. [Online]. Available: https://www.mdpi.com/1996-1944/13/24/5728. [Q1]
[19] MicroChemicals, "Lift-off Processes with Photoresists," *Basics of Micorstructuring,* 2009. [Online]. Available: https://research.engineering.ucdavis.edu/cnm2/wp-content/uploads/sites/11/2014/07/lift_off_photoresist.pdf.
[20] MicroChemicals, "Hardbake, Reflow and DUV hardening," *Basics of Microstructuring*. [Online]. Available: https://www.microchemicals.com/technical_information/photoresist_hardbake_reflow_uv_hardening.pdf.
[21] D. Morgan and E. G. S. Paige, "5 - Non-reflective transducers," in *Surface Acoustic Wave Filters (Second edition)*, D. Morgan and E. G. S. Paige Eds. Oxford: Academic Press, 2007, pp. 114-156.
[22] C. J. Pugh, "End Point Detection in Reactive Ion Etching," 2013.
[23] N. T. Tran, M. Breivik, S. K. Patra, and B. O. Fimland, "High precision AlGaAsSb ridge-waveguide etching by in situ reflectance monitored ICP-RIE," in *SEMICONDUCTOR LASERS AND LASER DYNAMICS VI*, vol. 9134, K. Panajotov, M. Sciamanna, A. Valle, and R. Michalzik Eds., 2014.
[24] K. Furuya *et al.*, "Nanometer-scale thickness control of amorphous silicon using isotropic wet-etching and low loss wire waveguide fabrication with the etched material," *Applied Physics Letters,* vol. 100, no. 25, p. 251108, 2012, doi: 10.1063/1.4729416. [Q2]
[25] R. Golzarian, "Liquid etch endpoint detection and process metrology," ed: Google Patents, 2002.
[26] S. Suhard, A. Iwasaki, M. Liebens, K. Stiers, J. Slabbekoorn, and F. Holsteyns, "Development of an all-in one wet single wafer process for 3D-SIC bump integration and its monitoring," in *2016 IEEE 18th Electronics Packaging Technology Conference (EPTC)*, 30 Nov.-3 Dec. 2016 2016, pp. 107-110, doi: 10.1109/EPTC.2016.7861453.
[27] J. T. Christol and J. S. Burchard, "End point detection in etching wafers and the like," ed: Google Patents, 1982.
[28] M. H. Nateq and R. Ceccato, "Enhanced Sol–Gel Route to Obtain a Highly Transparent and Conductive Aluminum-Doped Zinc Oxide Thin Film," *Materials,* vol. 12, no. 11, p. 1744, 2019. [Online]. Available: https://www.mdpi.com/1996-1944/12/11/1744. [Q1]
[29] K. Hyun-Chul, H. Hyo-Nyoung, B. Hee-Chul, K. Min-Gi, S. Ji-Yeon, and K. Young-Kuk, "HSV Color-Space-Based Automated Object Localization for Robot Grasping without Prior Knowledge," *Applied sciences,* vol. 11, no. 7593, p. 7593, 2021, doi: 10.3390/app11167593. [Q2]
[30] Docs.opencv.org. "OpenCV: Thresholding Operations using inRange." OpenCV. https://docs.opencv.org/3.4.15/da/d97/tutorial_threshold_inRange.html (accessed.

[31] "OpenCV: Morphological Transformations." https://docs.opencv.org/master/d9/d61/tutorial_py_morphological_ops.html (accessed 27 May 2021.
[32] R. Jardim and F. Morgado-Dias, "Savitzky–Golay filtering as image noise reduction with sharp color reset," *Microprocessors and Microsystems,* vol. 74, p. 103006, 2020/04/01/ 2020, doi: https://doi.org/10.1016/j.micpro.2020.103006. [Q3]

# Appendices

# Appendix A: Significant contribution

- Designed and prototyped a low cost automated wet-etch system with near-UV/visible light spectrum-based endpoint detection with commercially available digital microscope and LEDs array.

- Implemented image processing techniques in enhancing the measurement readings of the reflected intensity in order to improve the etching endpoint determination accuracy.

- Designed and implemented a software algorithm that is capable of analysing the reflected intensity trend to determine the etching endpoint.

- Collected and evaluated the functionality of the prototype with experimental data such as fabricated device's features dimensions and frequency response.

- Discovered the optical properties difference of ZnO and Al in the near-UV spectrum which can be potentially useful in visually differentiating these 2 materials for etching monitoring of transparent flexible SAW devices.

# Appendix B: Risk Analysis Document

## 1. Activity or Task Based Risk Assessment [Ref Number: 27790]

| | | | |
|---|---|---|---|
| **Name** | **FYP - Automated wet-etch system with infrared light based end-point detector towards fabrication of microdevices on flexible** | **Current Rating** | **Residual Rating** |
| **Location** | **Sunway - Jalan Lagoon Selatan - Building 05 (SW05) - Level 06 - M5601, Sunway - Jalan Lagoon Selatan - Building 05 (SW05) - Level 06 - M5602, Sunway - Jalan Lagoon Selatan - Building 05 (SW05) - Level 06 - M5603** | Medium | Medium |

| Business Unit | Last Review Date | Risk Owner |
|---|---|---|
| Malaysia Sch of Engineering | 23/03/2021 | LIK SIANG CHEW |

| Risk Assessment Team | Risk Approver |
|---|---|
| | Ajay Achath Mohanan |

**Additional Notes**

**Describe task / use**

The project involves the development of an automated system for wet etching of Al thin film deposited on ZnO/Al/SU-8/Si(100) substrates with end-point monitoring using an infrared sensor.

Monash University

Reports identifying people are confidential documents.
Statistical information shall only be used for internal reporting purposes.

Page 1 of 9

powered by riskware.com.au

# 1. Activity or Task Based Risk Assessment [Ref Number: 27790]

## Risk Factors

| Risk Factor | 6.1 - Short term contact with substances |
|---|---|

**Description**

The use of corrosive acid such as Nitric acid as an etchant to etch the thin aluminum layer from the substrate.

- 1.0 - Fixed plant (e.g. cool rooms, fume cupboards, safety showers, boilers, lathes, lifts, gas mains, PET scanners) -- No
- 2.0 - Transport and mobile plant (e.g. motor vehicles, forklifts, walky stackers, trolleys and wheelbarrows) -- No
- 3.0 - Powered equipment, tools and appliances (e.g. computers, workshop equipment, kitchen equipment, gas cylinders) -- No
- 4.0 - Non-powered handtools and equipment (e.g. furniture and fittings, ladders, handtools, packing equipment, glassware) -- No
- 5.0 - Chemical materials (e.g. dangerous goods, hazardous substances, poisons and drugs) -- Yes
- 6.0 - Materials and substances (not otherwise selected from category 5.0) -- No
- 6.3 - Fire and smoke -- No
- 7.1 - Outdoor working environment (e.g. carparks, walkways, outdoor stairs) -- No
- 7.2 - Indoor working environment (e.g. internal rooms, floor surfaces, stairwells) -- No
- 8.5 - Biological materials (e.g. animals, non-living animal materials, microorganisms) -- No
- 8.4 - Personal impairment and/or interaction (e.g. pre-existing medical condition, assisting a patient) -- No
- 9.1 - Psychological (e.g. stressful situations) -- No

| Medium | Medium | | |
|---|---|---|---|
| **Existing Controls** | **Proposed Controls** | | |
| • 4 - Engineering control measure:<br>The use of acid should be carried out in the fume chamber in order to prevent any unwanted acid particles from being inhaled.<br>Also, careful handling of acids in lab grade beaker ensures n unwanted reaction from happening at all times. | Description | Responsibility | Target Date |
| | **No Control:** The existing control method works fine. | | |

MONASH University

# 1. Activity or Task Based Risk Assessment [Ref Number: 27790]

| Risk Factor | 6.1 - Short term contact with substances |
|---|---|

**Description**

Acetone, methanol and photoresists are used in the process of fabricating substrates.

- 1.0 - Fixed plant (e.g. cool rooms, fume cupboards, safety showers, boilers, lathes, lifts, gas mains, PET scanners) -- No
- 2.0 - Transport and mobile plant (e.g. motor vehicles, forklifts, walky stackers, trolleys and wheelbarrows) -- No
- 3.0 - Powered equipment, tools and appliances (e.g. computers, workshop equipment, kitchen equipment, gas cylinders) -- No
- 4.0 - Non-powered handtools and equipment (e.g. furniture and fittings, ladders, handtools, packing equipment, glassware) -- No
- 5.0 - Chemical materials (e.g. dangerous goods, hazardous substances, poisons and drugs) -- Yes
- 6.0 - Materials and substances (not otherwise selected from category 5.0) -- No
- 6.3 - Fire and smoke -- No
- 7.1 - Outdoor working environment (e.g. carparks, walkways, outdoor stairs) -- No
- 7.2 - Indoor working environment (e.g. internal rooms, floor surfaces, stairwells) -- No
- 8.5 - Biological materials (e.g. animals, non-living animal materials, microorganisms) -- No
- 8.4 - Personal impairment and/or interaction (e.g. pre-existing medical condition, assisting a patient) -- No
- 9.1 - Psychological (e.g. stressful situations) -- No

| Medium | Medium | | |
|---|---|---|---|
| **Existing Controls** | **Proposed Controls** | | |
| • 4 - Engineering control measure: The use of these harmful substances will be carried out in the fume chamber in order to prevent any unwanted particles from being inhaled. Also, careful handling of these chemicals in lab grade beaker ensures unwanted reactions from happening at all times.<br>• 6 - Personal Protective Equipment: Put on personal protective equipment such as PPE suits, goggles, disposable gloves and hair nets during the handling process of these chemicals. | Description | Responsibility | Target Date |
| | **No Control:** The existing control method works fine. | | |

# 1. Activity or Task Based Risk Assessment [Ref Number: 27790]

| Risk Factor | 5.5 - Non-ionising radiation |
| --- | --- |

**Description**

Ultraviolet light used in the photolithography of the photoresist.

- 1.0 - Fixed plant (e.g. cool rooms, fume cupboards, safety showers, boilers, lathes, lifts, gas mains, PET scanners) -- No
- 2.0 - Transport and mobile plant (e.g. motor vehicles, forklifts, walky stackers, trolleys and wheelbarrows) -- No
- 3.0 - Powered equipment, tools and appliances (e.g. computers, workshop equipment, kitchen equipment, gas cylinders) -- Yes
- 4.0 - Non-powered handtools and equipment (e.g. furniture and fittings, ladders, handtools, packing equipment, glassware) -- No
- 5.0 - Chemical materials (e.g. dangerous goods, hazardous substances, poisons and drugs) -- No
- 6.0 - Materials and substances (not otherwise selected from category 5.0) -- No
- 6.3 - Fire and smoke -- No
- 7.1 - Outdoor working environment (e.g. carparks, walkways, outdoor stairs) -- No
- 7.2 - Indoor working environment (e.g. internal rooms, floor surfaces, stairwells) -- No
- 8.5 - Biological materials (e.g. animals, non-living animal materials, microorganisms) -- No
- 8.4 - Personal impairment and/or interaction (e.g. pre-existing medical condition, assisting a patient) -- No
- 9.1 - Psychological (e.g. stressful situations) -- No

| Medium | Medium | | |
|---|---|---|---|
| **Existing Controls** | **Proposed Controls** | | |
| • 6 - Personal Protective Equipment: Wear UV filtering goggles to prevent the UV radiation from reaching our eyes. | Description | Responsibility | Target Date |
| | **No Control:** The existing control method works fine. | | |

MONASH University

# 1. Activity or Task Based Risk Assessment [Ref Number: 27790]

## Appendix

| Risk Matrix Level | |
|---|---|
| Negligible | No additional control measures required |
| Low | Manage by routine procedures at local management level |
| Medium | Management responsibility must be specified and response procedures monitored |
| High | Senior management attention needed and management responsibility specified |
| Extreme | Immediate action required and must be managed by senior management with a detailed plan |

# 1. Activity or Task Based Risk Assessment [Ref Number: 27790]

| Risk Assessment Reviews | | |
|---|---|---|
| **Date of Review** | **Review Team** | **Summary of Review** |
| 23/03/2021 | | Clone |

# Appendix C: Ethics Compliance Form

**Ethical Compliance Form for ECSE FYP Projects**
**School of Engineering**
**Monash University**

This form should be used in conjunction with your Design document and should be attached to it as an appendix.

Prior to conducting your project, you and your supervisor will have discussed the ethical implications of your research. Monash ethical guidelines are comprehensively detailed in the Monash Research Office website, http://www.monash.edu.au/researchoffice/ethics.php. Some of FYP projects may have ethical concerns related to research integrity, research data management and human ethics. Students should identify any ethical implications related to the project and should complete relevant forms if these ethical issues are relevant.

Are any of the above ethical concerns related to your project? (yes/no)
If yes to the previous question, have you completed the relevant clearance forms? (yes/no)
(All forms could be downloaded from above link)

**Project title:** Automated wet-etch system with visible/ultra-violet light-based end-point detector
**Student's Name: Chew Lik Siang**
**Student's ID Number: 29035058**
**Student's Signature:** Chew
**Date: 28/5/2021**

# Appendix D: Environmental Guidelines Form

**Environmental Compliance Form for ECSE FYP Projects**

**School of Engineering**

**Monash University**

This form should be used in conjunction with your Design document and should be attached to it as an appendix.

Prior to conducting your project, you and your supervisor will have discussed the environmental impacts of your project. Monash Malaysia environmental policies are detailed in the OHSE (Occupational Health, Safety and Environment) website, http://www.monash.edu.my/ohse/. Some other potential sources are listed in the table on the next page. Check the appropriate box for any environmental hazard or impact associated with the project.

Are any of environmental aspects related to your project? (yes/no)

If yes to the previous question, have you completed the relevant clearance forms? (yes/no)

(Please seek your supervisor's help to get relevant forms)

Project title: **Automated wet-etch system with visible/ultra-violet light-based end-point detector**

Student's Name: **Chew Lik Siang**

Student's Registration Number: **29035058**

Student's Signature: *Chew*

Date: **28/5/2021**

| ENVIRONMENTAL ASPECT/IMPACT | Yes | ENVIRONMENTAL ASPECT/IMPACT | Yes |
|---|---|---|---|
| **1. Air emissions:** Will the project generate emissions from combustion, dust, greenhouse gases, ozone-depleting substances, or chemical gases? | ✔ | **10. Construction, Renovation, and Demolition By-Products:** Will any project activities generate construction debris or soil stockpiles by clearing or excavation, or disturb lead- or asbestos-containing materials? | |
| **2. Chemical Use, Storage, and Inventory:** Will the project require lab chemicals, fuel, oils, coolants, cleaners, or solvents? | ✔ | **11. Industrial and Hazardous Waste Generation, Management, Storage, Transport and Disposal:** Will any unused or spent chemicals, fuel, oils, solvents, PCBs, lead, asbestos, or other hazmat require handling as waste? | ✔ |
| **3. Soil and Groundwater Contamination:** Could project activities impact soil and groundwater in any way? | | **12. Biohazards:** Will the project use or generate biological materials, such as microorganisms? | |
| **4. Discharge to Wastewater Systems:** Will the project discharge any material to the sanitary sewer? | | **13. Nanotechnology:** Will the project use or generate any nanosubstances? | |
| **5. Surface and Stormwater Contamination:** Could material from the project contaminate stormwater or be discharged to the storm drain system? | | **14. Noise:** Could the project generate noise that would impact personnel or wildlife nearby? | |
| **6. Radioactive Materials Reduction and Radioactive Mixed-Waste Generation, Management, Storage, Transportation and Disposal:** Will any radiological waste be generated by the project? | | 15. **Other?** | |
| **7. Environmental Radiation and Radioactivity:** Could any project activities generate and/or release radioactivity? | | | |
| **8. Use, Reuse, and Recycling:** Are any project activities designed to minimize generation of waste through reuse, recycling, and environmentally preferable purchasing, such as purchasing recycled-content materials? | | | |
| **9. Conservation of Resources:** (Are any project activities designed to conserve natural resources such as water, energy, fuel, etc.? | | | |

# Appendix E: Overall operation codes

```
#!/usr/bin/python3.7

from PIL import Image as Image_PIL
from PIL import ImageTk
import numpy as np
import matplotlib.pyplot as plt
import cv2
import time
from subprocess import call
import board
import neopixel
from collections import deque
from bisect import insort, bisect_left
from itertools import islice
import os
from datetime import datetime
import scipy
from scipy.signal import savgol_filter, argrelmax, argrelextrema
import scipy.signal as ss
import motor_function as motor
import state as sm
import RPi.GPIO as GPIO
from tkinter import *
import threading
import ctypes
# import multiprocessing
threads = 0

pixels = neopixel.NeoPixel(board.D10, 16)
is_post_analysis = False
is_uv = False
last_iteration = 119
is_endpoint_overwrite = False
is_image_confirm = False
selection_array = list()
imgtk_frame = 0

uv_exposure_duration = 0

current_directory = 0
original_dir = 0
processed_dir = 0

def windowing_technique(moving_median_redness):
     # attempt to do windowing technique
    window_length = 3
    window_height = 10
    confidence_threshold = 6
    window_item = list()
    window_sample = list()
    window_count = 0;

    # point that passed through window vertically
    window_exceed_limit_item = list()
    window_exceed_limit_sample = list()
```

```
    endpoint_item = 0
    endpoint_sample = 0
    endpoint_offset_point = None
    final_endpoint_item = 0
    final_endpoint_sample = 0

    confidence_lvl = 0;

    end_point_reached_flag = 0

    for sample , item in enumerate(moving_median_redness):

        if sample == 0:
            # creating window
            left = sample
            right = sample + window_length
            up = window_height/2 + item
            down = item - window_height/2

        if (item > up or item < down or sample > right):

            # if the new data point passes through the window from top
            if (item > up):
                window_exceed_limit_item.append(item)
                window_exceed_limit_sample.append(sample)
                confidence_lvl += 1

                # if it passes through top window 3 consecutive times, it's
actual etching point
                if confidence_lvl >= confidence_threshold:
                    endpoint_item = item
                    endpoint_sample = sample
                    endpoint_offset_point = round(sample * 0.10) + sample
                    #print(f"endpoint point is : {endpoint_offset_point}")

            else:
                # else, it is just a noise and we need to collect again
                confidence_lvl = 0

            # create new window if new data point passes through window from
sides
            left = sample
            right = sample + window_length
            up = window_height/2 + item
            down = item - window_height
            window_count += 1

            window_item.append(item)
            window_sample.append(sample)

        # determining the the final endpoint (from last vertical crossed
sample
        # + 15% offset of previous period)
        if (sample == endpoint_offset_point):
            final_endpoint_item = item
            final_endpoint_sample = sample
            #print(f"END POINT REACHED.")
            end_point_reached_flag = 1
```

```
    return window_item, window_sample, window_exceed_limit_item,
window_exceed_limit_sample, final_endpoint_item, final_endpoint_sample,
end_point_reached_flag

def running_median_insort(seq, window_size):
    """Contributed by Peter Otten"""
    seq = iter(seq)
    d = deque()
    s = []
    result = []
    for item in islice(seq, window_size):
        d.append(item)
        insort(s, item)
        result.append(s[len(d)//2])
    m = window_size // 2
    for item in seq:
        old = d.popleft()
        d.append(item)
        del s[bisect_left(s, old)]
        insort(s, item)
        result.append(s[m])
    return result


def nothing(x):
    pass

def _fill_image(img, connectivity):
    """Fills all holes in connected components in a binary image.

    Parameters
    ------
    img : numpy array
        binary image to fill

    Returns
    ------
    filled : numpy array
        The filled image
    """
    # Copy the image with an extra border
    h, w = img.shape[:2]
    img_border = np.zeros((h + 2, w + 2), np.uint8)
    img_border[1:-1, 1:-1] = img

    floodfilled = img_border.copy()
    mask = np.zeros((h + 4, w + 4), np.uint8)
    cv2.floodFill(floodfilled, mask, (0, 0), 255, flags=connectivity)
    floodfill_inv = cv2.bitwise_not(floodfilled)
    filled = img_border | floodfill_inv
    filled = filled[1:-1, 1:-1]
    return filled

def colour_calculator_np(image, non_zero, non_zero_pixels_only = True):

    # converting into numpy version
    pix = np.array(image)
```

```
    if non_zero_pixels_only  == False:
        non_zero = pix.size/3

    # check for redness (R)
    #redness = pix[:,:,0] - (pix[:,:,1]+pix[:,:,2])/2
    redness = pix[:,:,2]#+pix[:,:,1] +pix[:,:,2] # only taking the red
pixels value
#     print(non_zero)
    redness[redness<0] = 0
    avg_redness = np.sum(redness)/non_zero


    # check for greenness (G)
    greenness = pix[:,:,1]# - (pix[:,:,0]+pix[:,:,2])/2
    greenness[greenness<0] = 0
    avg_greenness = np.sum(greenness)/non_zero

    # check for blueness (B)
    blueness = pix[:,:,2]# - (pix[:,:,0]+pix[:,:,1])/2
    blueness[blueness<0] = 0
    avg_blueness = np.sum(blueness)/non_zero

    return avg_redness,avg_greenness,avg_blueness

def masking_preview(image_dir):

    global is_image_confirm, imgtk_frame

    # Load in image
    image = cv2.imread(image_dir)

    # Create a window
    cv2.namedWindow('image')
#     cv2.namedWindow('picture_output')

    # create trackbars for color change
    cv2.createTrackbar('HMin','image',0,179,nothing) # Hue is from 0-179 for
Opencv
    cv2.createTrackbar('SMin','image',0,255,nothing)
    cv2.createTrackbar('VMin','image',0,255,nothing)
    cv2.createTrackbar('HMax','image',0,179,nothing)
    cv2.createTrackbar('SMax','image',0,255,nothing)
    cv2.createTrackbar('VMax','image',0,255,nothing)
#     cv2.createTrackbar('Colour','image',0,1,nothing)

    # Set default value for MAX HSV trackbars.
    cv2.setTrackbarPos('HMax', 'image', 179)
    cv2.setTrackbarPos('SMax', 'image', 255)
    cv2.setTrackbarPos('VMax', 'image', 255)
#     cv2.setTrackbarPos('Colour', 'image', 0)

    # Initialize to check if HSV min/max value changes
    hMin = sMin = vMin = hMax = sMax = vMax = 0
    phMin = psMin = pvMin = phMax = psMax = pvMax = colour = 0

    wait_time = 33

    while(1):
```

```
        # get current positions of all trackbars
        hMin = cv2.getTrackbarPos('HMin','image')
        sMin = cv2.getTrackbarPos('SMin','image')
        vMin = cv2.getTrackbarPos('VMin','image')

        hMax = cv2.getTrackbarPos('HMax','image')
        sMax = cv2.getTrackbarPos('SMax','image')
        vMax = cv2.getTrackbarPos('VMax','image')
#         colour = cv2.getTrackbarPos('Colour','image')

        # Set minimum and max HSV values to display
        lower = np.array([hMin, sMin, vMin])
        upper = np.array([hMax, sMax, vMax])


        # Print if there is a change in HSV value
        if( (phMin != hMin) | (psMin != sMin) | (pvMin != vMin) | (phMax !=
hMax) | (psMax != sMax) | (pvMax != vMax) ):
            print("(hMin = %d , sMin = %d, vMin = %d), (hMax = %d , sMax
= %d, vMax = %d)" % (hMin , sMin , vMin, hMax, sMax , vMax))
            phMin = hMin
            psMin = sMin
            pvMin = vMin
            phMax = hMax
            psMax = sMax
            pvMax = vMax
            # Create HSV Image and threshold into a range.
            hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
            mask = cv2.inRange(hsv, lower, upper)

            # perform "closing" morphology on the mask
            kernel = np.ones((10,10),np.uint8)
            mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel)
            mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel)
            mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel)
            # cv2.imwrite(f"MASK_without_flood.jpg", mask)

            # perform filling on the mask
            mask = _fill_image(mask,8)
            # cv2.imwrite(f"MASK.jpg", mask)
            # mask = cv2.bitwise_not(mask)

            # flood fill again
            mask = cv2.bitwise_not(mask)
            mask = _fill_image(mask,8)
            mask = cv2.bitwise_not(mask)

            non_zero = np.count_nonzero(mask)

            output = cv2.bitwise_and(image,image, mask= mask)
            cv2.imwrite(f"MASK_2nd_flood.jpg", mask)

            output_smaller = cv2.resize(output,(320,240))

            imgtk_raw =
Image_PIL.fromarray(cv2.cvtColor(output_smaller,cv2.COLOR_BGR2RGB))
            imgtk_frame = ImageTk.PhotoImage(image = imgtk_raw)
            imgtk_frame_window = label_image.create_image(0, 0, anchor=NW,
image=imgtk_frame)
```

```
        # Wait longer to prevent freeze for videos.
        if is_image_confirm == True or cv2.waitKey(wait_time) & 0xFF ==
ord('q'):
#              pixels.fill((0,0,0)) #yellow
            is_image_confirm = False
            break

    cv2.destroyAllWindows()

    return mask,non_zero



def focus_preview():
    global is_image_confirm

    cap = cv2.VideoCapture(0)

    if not cap.isOpened():
        raise IOError("Cannot open webcam")

#     print("wait 5 seconds")
#     time.sleep(5)
    pixels.fill((255,0,0)) #yellow 247, 202, 24
    time.sleep(0.1)
    motor.relax_motion(27, 17, 22, 18)
    time.sleep(0.1)
    motor.relax_motion(9, 25, 11, 8, vertical = False)
    counter = 0
    while True:
        counter = (counter + 1)%1000
        if counter == 0:
            try:
                label_image.delete(focus_preview_window)
                frame = None
                frame_smaller = None
                imgtk_raw  = None
                imgtk = None

            except:
                pass
    #          pixels = neopixel.NeoPixel(board.D10, 16)
    #          pixels.fill((255,0,0)) #yellow 247, 202, 24
        ret,frame = cap.read()
    #     frame = cv2.resize(frame,None, fx = 0.5, fy = 0.5,
interpolation=cv2.INTER_AREA)
        frame_smaller = cv2.resize(frame,(320,240))
#         cv2.imshow('Input',frame_smaller)
        imgtk_raw =
Image_PIL.fromarray(cv2.cvtColor(frame_smaller,cv2.COLOR_BGR2RGB))
        imgtk = ImageTk.PhotoImage(image = imgtk_raw)
#         label_image.imgtk = imgtk
#         label_image.configure(image=imgtk)
        focus_preview_window = label_image.create_image(0, 0, anchor=NW,
image=imgtk)
        if is_image_confirm == True:
#              pixels.fill((0,0,0)) #yellow
            is_image_confirm = False
```

```
            time.sleep(0.1)
            motor.relax_motion(27, 17, 22, 18)
            time.sleep(0.1)
            motor.relax_motion(9, 25, 11, 8, vertical = False)
            cap.release()
            cv2.destroyAllWindows()
            break

def emergency_stop():
    global threads

    print("STOPPPPPPPPP")
    my_label.config(text="Emergency stop")
    motor.stop_flag = 1
    threads.raise_exception()
    print("Done raising flag")
#     threads.join()
    print("Emergency settled")
    sm.initialisation()
    sm.emergency_stop_motion()# this function should be changed to able to
keep track current vertical count. Able to stop half way.

def loading_pos():
    # motor movement
    print("hello")
    sm.initialisation()
    sm.original_pos_check()
    my_label.config(text="Back to loading position")

class thread_with_exception(threading.Thread):
    def __init__(self, name):
        threading.Thread.__init__(self)
        self.name = name

    def run(self):

        # target function of the thread class
        try:
            global current_directory, original_dir, processed_dir,
uv_exposure_duration
            global is_endpoint_overwrite, last_iteration, is_uv,
is_image_confirm

            savgol_endpoint = 0
            savgol_endpoint_value = 0

            # create a new directory whenever a new operation started
            # new directory = 'current date' 'current time'
            current_directory, original_dir, processed_dir, plot_dir =
motor.create_new_directory()
            try:
                sm.initialisation()
            except:
                my_label.config(text="Back to loading position")

            sm.station_1()
            my_label.config(text="Reaches station 1")
            if is_post_analysis != True:
                my_label.config(text="Please ensure the view is focussed.")
```

```
                # First to preview the image to help focus
                # Press "q" to exit
                focus_preview()

            max_iter = 5
            hue_handler = list()
            saturation_handler = list()
            value_handler = list()
            redness_avg = list()
            greenness_avg = list()
            blueness_avg = list()


            # capture picture from "webcam"
            if is_post_analysis != True:
                snapshot_string = original_dir + '/' + "snapshot.jpg"
                cap = cv2.VideoCapture(0)
                ret,frame = cap.read()
                cap.release()
                cv2.imwrite(snapshot_string, frame)

            else:
                sample_no = var.get()
                if (var.get() == 1):
                    directory = os.path.join(current_directory, r'sample_1')
                    last_iteration = 119

                elif (var.get() == 2):
                    directory = os.path.join(current_directory, r'sample_2')
                    last_iteration = 186

                elif (var.get() == 3):
                    directory = os.path.join(current_directory, r'sample_3')
                    last_iteration = 187

                elif (var.get() == 4):
                    directory = os.path.join(current_directory, r'sample_4')
                    last_iteration = 109

                else:
                    directory = os.path.join(current_directory,
r'sample_6/Original')#'sample_1')
                    last_iteration = 658

#                 directory = os.path.join(current_directory, sample_str)
                snapshot_string = directory + '/' + "snapshot.jpg"#
directory of the downloaded image

            # enpoint flag
            end_point_reached_flag = 0

            my_label.config(text="Please adjust the limit for masking.")
            mask,non_zero = masking_preview(snapshot_string)
            my_label.config(text="Etching endpoint monitoring ongoing.")

            start_time = time.time()

            i = 0
```

```
            while (1):
                iteration_starttime = time.time()
                if is_post_analysis != True:
                    # Live analysis
                    # capture picture from "webcam"
                    snapshot_string = original_dir + '/' + "microscope_" +
str(i) + ".jpg"
                    # call(["fswebcam","-r", "1280x700","--no-banner",
snapshot_string])

                    cap = cv2.VideoCapture(0)
#                     if is_uv == True:
#                         pixels.fill((0,0,0)) #turn off led
#                         motor.uv_led_on();
#                         uv_start_time = time.time()
#                         time.sleep(0.05)

                    ret,frame_1 = cap.read()

#                     if is_uv == True:
#                         motor.uv_led_off();
#                         uv_end_time = time.time()
#                         uv_exposure_duration = (uv_end_time -
uv_start_time) + uv_exposure_duration

                    if is_uv == True and i%2 == 0:
                        motor.uv_led_on();
                        uv_start_time = time.time()
                        time.sleep(0.15)
                        motor.uv_led_off();
                        uv_end_time = time.time()
                        uv_exposure_duration = (uv_end_time - uv_start_time)
+ uv_exposure_duration

                    cap.release()
                    cv2.imwrite(snapshot_string, frame_1)


                else:
                    # Post analysis
                    snapshot_string = directory + '/' + "microscope_" +
str(i) + ".jpg" # directory of the downloaded image with counter

                # this is to open precaptured image
                image = cv2.imread(snapshot_string)

                # mask the image to extract chips region and save
                image = cv2.bitwise_and(image, image, mask = mask)
                processed_string = processed_dir + '/' + "sample_masked_" +
str(i) + ".jpg"
                cv2.imwrite(processed_string, image)

                # this is to open precaptured image
                masked_image = image
                masked_image_smaller = cv2.resize(masked_image,(320,240))

                if end_point_reached_flag == 1:
                    font                   = cv2.FONT_HERSHEY_SIMPLEX
                    bottomLeftCornerOfText = (0,200)
```

```
                    fontScale              = 1
                    fontColor              = (255,0,0)
                    lineType               = 2

                    masked_image_smaller =
cv2.putText(masked_image_smaller,'##END POINT!##', bottomLeftCornerOfText,
font, fontScale,fontColor,lineType)

        #         cv2.imshow('picture_output',masked_image_smaller )

                imgtk_raw =
Image_PIL.fromarray(cv2.cvtColor(masked_image_smaller ,cv2.COLOR_BGR2RGB))
                imgtk_endpoint = ImageTk.PhotoImage(image = imgtk_raw)
#                 label_image.configure(image=imgtk_endpoint)
                label_image.create_image(0, 0, anchor=NW,
image=imgtk_endpoint)

                # It converts the BGR color space of image to HSV color
space
                hsv = cv2.cvtColor(masked_image, cv2.COLOR_BGR2HSV)

                np_hue = np.resize(hsv[:,:,0],(1,-1))
                hue_avg = ((np.mean(np_hue))/180)
                hue_handler.append(hue_avg)
                # print(hue_avg)

                np_saturation = np.resize(hsv[:,:,1],(1,-1))
                saturation_avg = np.mean(np_saturation)/255
                saturation_handler.append(saturation_avg)

                np_value = np.resize(hsv[:,:,2],(1,-1))
                value_avg = np.mean(np_value)/255
                value_handler.append(value_avg)

                # calculate redness, greenness, blueness (RGB)
                redness, greenness, blueness =
colour_calculator_np(masked_image,non_zero)
                redness_avg.append(redness)
            #     greenness_avg.append(greenness)
            #     blueness_avg.append(blueness)

            #     if(len(redness_avg)>1):
            #         redness_gradient_numpy =
np.gradient(np.array(redness_avg),2)

                if(len(redness_avg)>=7):

                    median_window_size = 7
                    # eleminate noise using moving median
                    moving_median_redness =
running_median_insort(redness_avg, median_window_size)

                    # window method to analyse filtered redness graph
                    window_item, window_sample, window_exceed_limit_item,
window_exceed_limit_sample, final_endpoint_item, final_endpoint_sample,
end_point_reached_flag = windowing_technique(moving_median_redness)

                    redness_gradient_numpy =
np.gradient(np.array(moving_median_redness),2)
```

```
                # inflection computation for endpoint deterination
                if(len(redness_avg)>=11):
                    div = ss.savgol_filter(moving_median_redness,9, 2, deriv
= 1)

                    maxima = argrelmax(div)

                    if len(maxima[0]) != 0:
                        maxima_value = max(div[maxima[0]])
                        location = maxima[0][np.argmax(div[maxima[0]])]

                        # setting a threshold for derevative inflection
determination
                        if maxima_value >= 0.3: #5

                            # predicting the endpoint from inflection point
                            predicted_endpoint = location*1.4 # should still
be around 1.25
                            end_string = "Change detected!" + "End in " +
str(round(predicted_endpoint-i))
                            my_label.config(text=end_string)

                            if i >= predicted_endpoint:
                                print("HELLO!!!!!!!!!!!!!!!!!!!!")
                                my_label.config(text="Endpoint Reached!")
                                my_label.config(bg = 'yellow')
                                savgol_endpoint = predicted_endpoint
                                savgol_endpoint_value =
moving_median_redness[int(savgol_endpoint)]

                                if is_endpoint_overwrite:
                                    # manual
                                    pass
                                else:
                                    # auto termination
                                    pixels.fill((0,0,0)) #turn off led
                                    my_label.config(bg = 'white')
                                    cv2.destroyAllWindows()
                                    break



                i = i + 1
                if is_image_confirm:
                    is_image_confirm = False
                    pixels.fill((0,0,0)) #turn off led
                    my_label.config(bg = 'white')
                    cv2.destroyAllWindows()
                    break

                if is_post_analysis == True:
                    if i == last_iteration:
                        break

                iteration_endtime = time.time()
                # add in delay to make each iteration exactly 1 s each
                if is_post_analysis == False:
                    try:
```

```
                        time.sleep(1-(iteration_endtime -
iteration_starttime))

                    except:
                        pass

            end_time = time.time()
            print("%d iterations took %.2f seconds, which corresponds
to %.2fs/iteration" % (i, end_time - start_time, (end_time - start_time)/i))
            print("Samples exposed to UV %d times and took %.2f seconds,
which corresponds to %.2fs/iteration" % (i, uv_exposure_duration,
uv_exposure_duration/i))
        # #np.savez('redness_data_sample_5.npz',
sample_1=moving_median_redness)
        #
            # motor move to the next station
            my_label.config(text="Water bath")
            sm.station_2()
            time.sleep(10)
            my_label.config(text="Acetone bath")
            sm.station_3()
            time.sleep(10)
            my_label.config(text="IPA bath")
            sm.station_4()
            time.sleep(10)
            my_label.config(text="Back to loading position")
            sm.back_to_loading_pos()
            my_label.config(text="Sample ready to be collected")

            x = np.arange(0,int(len(hue_handler)),1)
            smoothed_red = savgol_filter(moving_median_redness, 9, 3)

            fig = plt.figure()
#             ax1 = fig.add_subplot(311)
            ax2 = fig.add_subplot(211)
            ax3 = fig.add_subplot(212)

#             ax1.plot(x, hue_handler, marker='o')
#             ax1.plot(x, saturation_handler, marker='o')
#             ax1.plot(x, value_handler, marker='o')

            ax2.plot(x, redness_avg, marker= 'o')
            # ax2.plot(x, greenness_avg, marker= 'o')
            # ax2.plot(x, blueness_avg, marker= 'o')

            x_moving_avg = np.arange(0,int(len(moving_median_redness)),1)
            ax3.plot(x_moving_avg,moving_median_redness, marker= 'o')
            ax3.plot(window_sample,window_item, marker= 'o')
            ax3.plot(window_exceed_limit_sample,window_exceed_limit_item,
marker= 'o')
            # ax3.plot(endpoint_sample, endpoint_item, marker= 'o')
            ax3.plot(final_endpoint_sample, final_endpoint_item, marker=
'o')

#             ax1.set_title("HSV parameters of images")
#             ax1.set_xlabel('Iterations')
#             ax1.legend(['hue','saturation','lightness'],
fontsize='small',labelspacing=0.2)
```

```
            ax2.set_title("Average RGB of images")
            ax2.set_xlabel('Iterations')
            ax2.legend(['R','G','B'], fontsize='small',labelspacing=0.2)

            ax3.set_title("Smoothen version of redness graph (range = 0-
255)")
            ax3.set_xlabel('Iterations')
            ax3.set_ylabel('Avg. red pixels value')
            ax3.legend(['Smothen R pixels graph','Windows','Datapoint that
passed through window vertically', "Expected endpoint (+10% intial time)"],
fontsize='small',labelspacing=0.2)

            fig.tight_layout(pad = 1.0)

            plot_string = plot_dir + '/' +
"Red_pixels_trend_smoothed_and_raw.png"
            fig.savefig(plot_string)

            fig2 = plt.figure()
            ax = fig2.add_subplot(111)
            ax.plot(x_moving_avg ,redness_gradient_numpy, marker='o')
            ax.set_title("Raw (Numpy) derivative of the red (R) pixels
trend")
            ax.set_xlabel('Iterations')
            ax.set_ylabel('Raw (Numpy) derivative')
            plot_string = plot_dir + '/' + "numpy_derivative.png"
            fig2.savefig(plot_string)


            fig3 = plt.figure()
            ax = fig3.add_subplot(111)
            ax.plot(x_moving_avg ,div, marker='o')
            ax.set_title("Savgol derivative of the red (R) pixels trend")
            ax.set_xlabel('Iterations')
            ax.set_ylabel('Savgol derivative')
            plot_string = plot_dir + '/' + "savgol_derivative.png"
            fig3.savefig(plot_string)

            fig4 = plt.figure()
            ax = fig4.add_subplot(111)
            ax.plot(x_moving_avg,moving_median_redness, marker='o')
            ax.plot(savgol_endpoint ,savgol_endpoint_value, marker='o')
            ax.set_title("Smoothen version of redness graph (range = 0-
255)")
            ax.set_xlabel('Iterations')
            ax.set_ylabel('Avg. red pixels value')
            ax.legend(['Smothen R pixels graph', 'Endpoint from Savgol
derivative'], fontsize='small',labelspacing=0.2)
            plot_string = plot_dir + '/' + "savgol_derivative_endpoint.png"
            fig4.savefig(plot_string)


            plt.show()
        finally:
            print('ended')

    def get_id(self):

        # returns id of the respective thread
```

```
        if hasattr(self, '_thread_id'):
            return self._thread_id
        for id, thread in threading._active.items():
            if thread is self:
                return id

    def raise_exception(self):
        thread_id = self.get_id()
        res = ctypes.pythonapi.PyThreadState_SetAsyncExc(thread_id,
              ctypes.py_object(SystemExit))
        if res > 1:
            ctypes.pythonapi.PyThreadState_SetAsyncExc(thread_id, 0)
            print('Exception raise failure')

def thread_creation():
    global threads
    t1 = thread_with_exception('Thread 1')
    t1.start()

    threads = t1

def toggle():
    global is_post_analysis

    if toggle_btn.config('relief')[-1] == 'sunken':
        toggle_btn.config(relief="raised")
        toggle_btn.config(text="Live monitor")
        is_post_analysis = False

        for i in selection_array:
            i.grid_remove()

        selection_array.clear()


    else:
        toggle_btn.config(relief="sunken")
        toggle_btn.config(text="Post analysis")
        is_post_analysis = True
        sample_choice = Label(text = "Sample no?")
        sample_choice.grid(row = 6, column = 0)

        R1 = Radiobutton(root, text="1", variable=var, value=1,
                         command=sel)
        R1.grid(row = 6, column = 1)

        R2 = Radiobutton(root, text="2", variable=var, value=2,
                         command=sel)
        R2.grid(row = 6, column = 2)

        R3 = Radiobutton(root, text="3", variable=var, value=3,
                         command=sel)
        R3.grid(row = 7, column = 1)

        R4 = Radiobutton(root, text="4", variable=var, value=4,
                         command=sel)
        R4.grid(row = 7, column = 2)

        selection_array.extend((sample_choice, R1, R2, R3, R4))
```

```
def toggle_uv():
    global is_uv

    if toggle_btn_uv.config('relief')[-1] == 'sunken':
        toggle_btn_uv.config(relief="raised", bg = 'yellow')
        toggle_btn_uv.config(text="Visible\nlight")
        is_uv = False

    else:
        toggle_btn_uv.config(relief="sunken", bg = 'violet')
        toggle_btn_uv.config(text="UV\nlight")
        is_uv= True


def toggle_endpoint():
    global is_endpoint_overwrite

    if toggle_btn_endpoint.config('relief')[-1] == 'sunken':
        toggle_btn_endpoint.config(relief="raised")
        toggle_btn_endpoint.config(text="Auto\nendpoint")
        is_endpoint_overwrite = False

    else:
        toggle_btn_endpoint.config(relief="sunken")
        toggle_btn_endpoint.config(text="Manual\nendpoint")
        is_endpoint_overwrite = True

def confirm_image():
    global is_image_confirm

    is_image_confirm = True

def sel():
    selection = "You selected the option " + str(var.get())
#    label.config(text = selection)
    print(str(var.get()))

root = Tk()

root.title("Learn To Code at Codemy.com")
root.geometry("380x480")

current_display = Label(root, text="Current state:", anchor=N)
current_display.grid(row = 0, column = 0)

my_label = Label(root, width=35, text="Welcome to wet etching process
bench!")
my_label.config(bg = 'white')
my_label.grid(row = 1, column = 0, columnspan = 2, pady = 10, sticky =
'nsew')

next_btn_img = PhotoImage(file = "green_button.PNG")
# Create A Button
next_btn = Button(root, image = next_btn_img , bd = 0, command =
confirm_image)
next_btn.grid(row = 1, column = 2, sticky = 'nsew')

my_button1 = Button(root, text="Loading\nstation", command=
loading_pos)#lambda: threading.Thread(target=loading_pos).start())
```

```
my_button1.grid(row = 3, column = 0, sticky = 'nsew')

toggle_btn_endpoint = Button(root, text="Auto\nendpoint", width = 9,
relief="raised", command=toggle_endpoint)
toggle_btn_endpoint.grid(row = 2, column = 0, sticky = 'nsew')

# toggle_btn = tk.Button(text="Toggle", width=12, relief="raised")
toggle_btn = Button(root, text="Live monitor", width = 9, relief="raised",
command=toggle)
toggle_btn.grid(row = 2, column = 1, sticky = 'nsew')

toggle_btn_uv = Button(root, text="Visible\nlight", width = 9,
relief="raised", bg = 'yellow', command=toggle_uv)
toggle_btn_uv.grid(row = 2, column = 2, sticky = 'nsew')

# t1 = threading.Thread(target=start)
# thread_with_exception
# t1 = thread_with_exception('Thread 1')

my_button2 = Button(root, text="Start", command= lambda:
thread_creation())#multiprocessing.Process(target=start).start())#threading.
Thread(target=start).start())
my_button2.grid(row = 3, column = 1, sticky = 'nsew')#.pack(padx=50, pady =
5, side=LEFT, expand='yes', anchor = N)

my_button3 = Button(root, text="Emergency\nStop", bg='red', command= lambda:
threading.Thread(target=emergency_stop).start())#emergency_stop)#lambda:
threading.Thread(target=emergency_stop).start())
my_button3.grid(row = 3, column = 2, sticky = 'nsew')#.pack(padx=50, pady =
5, side=RIGHT, expand='yes', anchor = N)

label_image = Canvas(root, width=320, height = 240)
label_image.grid(row = 5, column = 0, columnspan = 3)

var = IntVar()

root.mainloop()
```

# Appendix F: Codes of basic function blocks

```
import RPi.GPIO as GPIO
import time
import board
from datetime import datetime
import os

# for stepper motor
GPIO.cleanup()
```

```
# Sensor_flag
sensor_1_flag = 0
sensor_2_flag = 0
sensor_3_flag = 0
sensor_4_flag = 0
sensor_5_flag = 0
stop_flag = 0

# movement counter
vertical_counter = 0
horizontal_counter = 0
sign = 1
vertical_current_pos = 0

# Pin Assignment
out1 = 27 #pin 13
out2 = 17 #pin 11
out3 = 22 #pin 15
out4 = 18 #pin 12
uv_led = 26 #pin37

out1a = 9 #pin 21
out2a = 25 #pin 22
out3a = 11 #pin 23
out4a = 8 #pin 24

sensor1 = 23 #pin16
sensor2 = 5 #pin29
sensor3 = 6 #pin31
sensor4 = 12 #pin32
sensor5 = 13 #pin33

# setup the allocated pin and set it to be input/ouput
GPIO.setmode(GPIO.BCM)
GPIO.setup(out1,GPIO.OUT)
GPIO.setup(out2,GPIO.OUT)
GPIO.setup(out3,GPIO.OUT)
GPIO.setup(out4,GPIO.OUT)
GPIO.setup(uv_led,GPIO.OUT)

GPIO.setup(out1a,GPIO.OUT)
GPIO.setup(out2a,GPIO.OUT)
GPIO.setup(out3a,GPIO.OUT)
GPIO.setup(out4a,GPIO.OUT)

GPIO.setup(sensor1, GPIO.IN)  # Set our input pin to be an input
GPIO.setup(sensor2, GPIO.IN)  # Set our input pin to be an input
GPIO.setup(sensor3, GPIO.IN)  # Set our input pin to be an input
GPIO.setup(sensor4, GPIO.IN)  # Set our input pin to be an input
GPIO.setup(sensor5, GPIO.IN)  # Set our input pin to be an input

GPIO.output(out1,GPIO.LOW)
GPIO.output(out2,GPIO.LOW)
GPIO.output(out3,GPIO.LOW)
GPIO.output(out4,GPIO.LOW)
GPIO.output(uv_led,GPIO.HIGH)
GPIO.output(out1a,GPIO.LOW)
GPIO.output(out2a,GPIO.LOW)
GPIO.output(out3a,GPIO.LOW)
```

```
GPIO.output(out4a,GPIO.LOW)


def horizontal_right_motion(p1, p2, p3, p4):
    """Horizontal stepper motor moves to the right.

    Parameters:
    p1 (int): Stepper motor signal pin1
    p2 (int): Stepper motor signal pin2
    p3 (int): Stepper motor signal pin3
    p4 (int): Stepper motor signal pin4

   """

    global horizontal_counter

    if horizontal_counter==0:
        GPIO.output(p1,GPIO.HIGH)
        GPIO.output(p2,GPIO.LOW)
        GPIO.output(p3,GPIO.LOW)
        GPIO.output(p4,GPIO.LOW)
        time.sleep(0.001)

    elif horizontal_counter==1:
        GPIO.output(p1,GPIO.HIGH)
        GPIO.output(p2,GPIO.HIGH)
        GPIO.output(p3,GPIO.LOW)
        GPIO.output(p4,GPIO.LOW)
        time.sleep(0.001)

    elif horizontal_counter==2:
        GPIO.output(p1,GPIO.LOW)
        GPIO.output(p2,GPIO.HIGH)
        GPIO.output(p3,GPIO.LOW)
        GPIO.output(p4,GPIO.LOW)
        time.sleep(0.001)

    elif horizontal_counter==3:
        GPIO.output(p1,GPIO.LOW)
        GPIO.output(p2,GPIO.HIGH)
        GPIO.output(p3,GPIO.HIGH)
        GPIO.output(p4,GPIO.LOW)
        time.sleep(0.001)

    elif horizontal_counter==4:
        GPIO.output(p1,GPIO.LOW)
        GPIO.output(p2,GPIO.LOW)
        GPIO.output(p3,GPIO.HIGH)
        GPIO.output(p4,GPIO.LOW)
        time.sleep(0.001)

    elif horizontal_counter==5:
        GPIO.output(p1,GPIO.LOW)
        GPIO.output(p2,GPIO.LOW)
        GPIO.output(p3,GPIO.HIGH)
        GPIO.output(p4,GPIO.HIGH)
        time.sleep(0.001)

    elif horizontal_counter==6:
```

```
        GPIO.output(p1,GPIO.LOW)
        GPIO.output(p2,GPIO.LOW)
        GPIO.output(p3,GPIO.LOW)
        GPIO.output(p4,GPIO.HIGH)
        time.sleep(0.001)

    elif horizontal_counter==7:
        GPIO.output(p1,GPIO.HIGH)
        GPIO.output(p2,GPIO.LOW)
        GPIO.output(p3,GPIO.LOW)
        GPIO.output(p4,GPIO.HIGH)
        time.sleep(0.001)

    if horizontal_counter==0:
        horizontal_counter=7
    else:
        horizontal_counter=horizontal_counter-1

def horizontal_left_motion (p1, p2, p3, p4):
    """Horizontal stepper motor moves to the left.

    Parameters:
    p1 (int): Stepper motor signal pin1
    p2 (int): Stepper motor signal pin2
    p3 (int): Stepper motor signal pin3
    p4 (int): Stepper motor signal pin4

   """
    global horizontal_counter

    if horizontal_counter==0:
        GPIO.output(p1,GPIO.HIGH)
        GPIO.output(p2,GPIO.LOW)
        GPIO.output(p3,GPIO.LOW)
        GPIO.output(p4,GPIO.LOW)
        time.sleep(0.001)
    #time.sleep(1)
    elif horizontal_counter==1:
        GPIO.output(p1,GPIO.HIGH)
        GPIO.output(p2,GPIO.HIGH)
        GPIO.output(p3,GPIO.LOW)
        GPIO.output(p4,GPIO.LOW)
        time.sleep(0.001)
    #time.sleep(1)
    elif horizontal_counter==2:
        GPIO.output(p1,GPIO.LOW)
        GPIO.output(p2,GPIO.HIGH)
        GPIO.output(p3,GPIO.LOW)
        GPIO.output(p4,GPIO.LOW)
        time.sleep(0.001)

    elif horizontal_counter==3:
        GPIO.output(p1,GPIO.LOW)
        GPIO.output(p2,GPIO.HIGH)
        GPIO.output(p3,GPIO.HIGH)
        GPIO.output(p4,GPIO.LOW)
        time.sleep(0.001)

    elif horizontal_counter==4:
```

```
        GPIO.output(p1,GPIO.LOW)
        GPIO.output(p2,GPIO.LOW)
        GPIO.output(p3,GPIO.HIGH)
        GPIO.output(p4,GPIO.LOW)
        time.sleep(0.001)

    elif horizontal_counter==5:
        GPIO.output(p1,GPIO.LOW)
        GPIO.output(p2,GPIO.LOW)
        GPIO.output(p3,GPIO.HIGH)
        GPIO.output(p4,GPIO.HIGH)
        time.sleep(0.001)

    elif horizontal_counter==6:
        GPIO.output(p1,GPIO.LOW)
        GPIO.output(p2,GPIO.LOW)
        GPIO.output(p3,GPIO.LOW)
        GPIO.output(p4,GPIO.HIGH)
        time.sleep(0.001)

    elif horizontal_counter==7:
        GPIO.output(p1,GPIO.HIGH)
        GPIO.output(p2,GPIO.LOW)
        GPIO.output(p3,GPIO.LOW)
        GPIO.output(p4,GPIO.HIGH)
        time.sleep(0.001)

    if horizontal_counter==7:
        horizontal_counter=0
    else:
        horizontal_counter=horizontal_counter+1

def vertical_down_motion(p1, p2, p3, p4, magnitude):
    """Vertical stepper motor moves downwards by a certain count.

    Parameters:
    p1 (int): Stepper motor signal pin1
    p2 (int): Stepper motor signal pin2
    p3 (int): Stepper motor signal pin3
    p4 (int): Stepper motor signal pin4
    magnitude (int): Stepper motor total move down count

   """
    global vertical_counter, vertical_current_pos

    for stepmotor_turns in range (magnitude):
#         delay = (magnitude - stepmotor_turns)/magnitude*0.01
#
#         if delay<0.005:
#             delay = 0.005
        delay = 0.005

        if vertical_counter==0:
            GPIO.output(p1,GPIO.HIGH)
            GPIO.output(p2,GPIO.LOW)
            GPIO.output(p3,GPIO.LOW)
            GPIO.output(p4,GPIO.LOW)
            time.sleep(delay)
        #time.sleep(1)
```

```
        elif vertical_counter==1:
            GPIO.output(p1,GPIO.HIGH)
            GPIO.output(p2,GPIO.HIGH)
            GPIO.output(p3,GPIO.LOW)
            GPIO.output(p4,GPIO.LOW)
            time.sleep(delay)
        #time.sleep(1)
        elif vertical_counter==2:
            GPIO.output(p1,GPIO.LOW)
            GPIO.output(p2,GPIO.HIGH)
            GPIO.output(p3,GPIO.LOW)
            GPIO.output(p4,GPIO.LOW)
            time.sleep(delay)

        elif vertical_counter==3:
            GPIO.output(p1,GPIO.LOW)
            GPIO.output(p2,GPIO.HIGH)
            GPIO.output(p3,GPIO.HIGH)
            GPIO.output(p4,GPIO.LOW)
            time.sleep(delay)

        elif vertical_counter==4:
            GPIO.output(p1,GPIO.LOW)
            GPIO.output(p2,GPIO.LOW)
            GPIO.output(p3,GPIO.HIGH)
            GPIO.output(p4,GPIO.LOW)
            time.sleep(delay)

        elif vertical_counter==5:
            GPIO.output(p1,GPIO.LOW)
            GPIO.output(p2,GPIO.LOW)
            GPIO.output(p3,GPIO.HIGH)
            GPIO.output(p4,GPIO.HIGH)
            time.sleep(delay)

        elif vertical_counter==6:
            GPIO.output(p1,GPIO.LOW)
            GPIO.output(p2,GPIO.LOW)
            GPIO.output(p3,GPIO.LOW)
            GPIO.output(p4,GPIO.HIGH)
            time.sleep(delay)

        elif vertical_counter==7:
            GPIO.output(p1,GPIO.HIGH)
            GPIO.output(p2,GPIO.LOW)
            GPIO.output(p3,GPIO.LOW)
            GPIO.output(p4,GPIO.HIGH)
            time.sleep(delay)

        if vertical_counter==7:
            vertical_counter=0
        else:
            vertical_counter=vertical_counter+1

        vertical_current_pos = vertical_current_pos - 1

def vertical_up_motion(p1, p2, p3, p4, magnitude):
    """Vertical stepper motor moves upwards by a certain count.
```

```
    Parameters:
    p1 (int): Stepper motor signal pin1
    p2 (int): Stepper motor signal pin2
    p3 (int): Stepper motor signal pin3
    p4 (int): Stepper motor signal pin4
    magnitude (int): Stepper motor total move up count

   """
    global vertical_counter, vertical_current_pos

    for stepmotor_turns in range (magnitude):

        delay = 0.003

        if vertical_counter==0:
            GPIO.output(p1,GPIO.HIGH)
            GPIO.output(p2,GPIO.LOW)
            GPIO.output(p3,GPIO.LOW)
            GPIO.output(p4,GPIO.LOW)
            time.sleep(delay)

        elif vertical_counter==1:
            GPIO.output(p1,GPIO.HIGH)
            GPIO.output(p2,GPIO.HIGH)
            GPIO.output(p3,GPIO.LOW)
            GPIO.output(p4,GPIO.LOW)
            time.sleep(delay)

        elif vertical_counter==2:
            GPIO.output(p1,GPIO.LOW)
            GPIO.output(p2,GPIO.HIGH)
            GPIO.output(p3,GPIO.LOW)
            GPIO.output(p4,GPIO.LOW)
            time.sleep(delay)

        elif vertical_counter==3:
            GPIO.output(p1,GPIO.LOW)
            GPIO.output(p2,GPIO.HIGH)
            GPIO.output(p3,GPIO.HIGH)
            GPIO.output(p4,GPIO.LOW)
            time.sleep(delay)

        elif vertical_counter==4:
            GPIO.output(p1,GPIO.LOW)
            GPIO.output(p2,GPIO.LOW)
            GPIO.output(p3,GPIO.HIGH)
            GPIO.output(p4,GPIO.LOW)
            time.sleep(delay)

        elif vertical_counter==5:
            GPIO.output(p1,GPIO.LOW)
            GPIO.output(p2,GPIO.LOW)
            GPIO.output(p3,GPIO.HIGH)
            GPIO.output(p4,GPIO.HIGH)
            time.sleep(delay)

        elif vertical_counter==6:
            GPIO.output(p1,GPIO.LOW)
            GPIO.output(p2,GPIO.LOW)
```

```
            GPIO.output(p3,GPIO.LOW)
            GPIO.output(p4,GPIO.HIGH)
            time.sleep(delay)

        elif vertical_counter==7:
            GPIO.output(p1,GPIO.HIGH)
            GPIO.output(p2,GPIO.LOW)
            GPIO.output(p3,GPIO.LOW)
            GPIO.output(p4,GPIO.HIGH)
            time.sleep(delay)

        if vertical_counter==0:
            vertical_counter=7
        else:
            vertical_counter=vertical_counter-1

        vertical_current_pos = vertical_current_pos + 1

# def vertical_up_motion(p1, p2, p3, p4):
#     """Stepper motor reverse motion. Counter decrease by 1.
#
#     Parameters:
#     i (int): Current motion state
#
#     Returns:
#     i (int): Next motion state
#
#    """
#     global vertical_counter
#     if vertical_counter==0:
#         GPIO.output(p1,GPIO.HIGH)
#         GPIO.output(p2,GPIO.LOW)
#         GPIO.output(p3,GPIO.LOW)
#         GPIO.output(p4,GPIO.LOW)
#         time.sleep(0.003)
#
#     elif vertical_counter==1:
#         GPIO.output(p1,GPIO.HIGH)
#         GPIO.output(p2,GPIO.HIGH)
#         GPIO.output(p3,GPIO.LOW)
#         GPIO.output(p4,GPIO.LOW)
#         time.sleep(0.003)
#
#     elif vertical_counter==2:
#         GPIO.output(p1,GPIO.LOW)
#         GPIO.output(p2,GPIO.HIGH)
#         GPIO.output(p3,GPIO.LOW)
#         GPIO.output(p4,GPIO.LOW)
#         time.sleep(0.003)
#
#     elif vertical_counter==3:
#         GPIO.output(p1,GPIO.LOW)
#         GPIO.output(p2,GPIO.HIGH)
#         GPIO.output(p3,GPIO.HIGH)
#         GPIO.output(p4,GPIO.LOW)
#         time.sleep(0.003)
#
#     elif vertical_counter==4:
#         GPIO.output(p1,GPIO.LOW)
```

```
#         GPIO.output(p2,GPIO.LOW)
#         GPIO.output(p3,GPIO.HIGH)
#         GPIO.output(p4,GPIO.LOW)
#         time.sleep(0.003)
#
#     elif vertical_counter==5:
#         GPIO.output(p1,GPIO.LOW)
#         GPIO.output(p2,GPIO.LOW)
#         GPIO.output(p3,GPIO.HIGH)
#         GPIO.output(p4,GPIO.HIGH)
#         time.sleep(0.003)
#
#     elif vertical_counter==6:
#         GPIO.output(p1,GPIO.LOW)
#         GPIO.output(p2,GPIO.LOW)
#         GPIO.output(p3,GPIO.LOW)
#         GPIO.output(p4,GPIO.HIGH)
#         time.sleep(0.003)
#
#     elif vertical_counter==7:
#         GPIO.output(p1,GPIO.HIGH)
#         GPIO.output(p2,GPIO.LOW)
#         GPIO.output(p3,GPIO.LOW)
#         GPIO.output(p4,GPIO.HIGH)
#         time.sleep(0.003)
#
#     if vertical_counter==0:
#         vertical_counter=7
#     else:
#         vertical_counter=vertical_counter-1


def relax_motion(p1, p2, p3, p4, vertical = True):

    """Stepper motor off mode. Motor not energised.

    Parameters:
    p1 (int): Stepper motor signal pin1
    p2 (int): Stepper motor signal pin2
    p3 (int): Stepper motor signal pin3
    p4 (int): Stepper motor signal pin4
    vertical (boolean): True: vertical or False: horizontal

   """
    if vertical == True:
        time.sleep(0.1)
    GPIO.output(p1,GPIO.LOW)
    GPIO.output(p2,GPIO.LOW)
    GPIO.output(p3,GPIO.LOW)
    GPIO.output(p4,GPIO.LOW)

    if vertical == True:
        time.sleep(0.1)

def hold_motion():
    """Stepper motor holding mode. Motor maintains current position."""

    global vertical_counter
```

```
    vertical_counter = vertical_counter - vertical_counter%2

    if vertical_counter==0:
        GPIO.output(out1,GPIO.HIGH)
        GPIO.output(out2,GPIO.LOW)
        GPIO.output(out3,GPIO.LOW)
        GPIO.output(out4,GPIO.LOW)

    elif vertical_counter==1:
        GPIO.output(out1,GPIO.HIGH)
        GPIO.output(out2,GPIO.HIGH)
        GPIO.output(out3,GPIO.LOW)
        GPIO.output(out4,GPIO.LOW)

    elif vertical_counter==2:
        GPIO.output(out1,GPIO.LOW)
        GPIO.output(out2,GPIO.HIGH)
        GPIO.output(out3,GPIO.LOW)
        GPIO.output(out4,GPIO.LOW)

    elif vertical_counter==3:
        GPIO.output(out1,GPIO.LOW)
        GPIO.output(out2,GPIO.HIGH)
        GPIO.output(out3,GPIO.HIGH)
        GPIO.output(out4,GPIO.LOW)

    elif vertical_counter==4:
        GPIO.output(out1,GPIO.LOW)
        GPIO.output(out2,GPIO.LOW)
        GPIO.output(out3,GPIO.HIGH)
        GPIO.output(out4,GPIO.LOW)

    elif vertical_counter==5:
        GPIO.output(out1,GPIO.LOW)
        GPIO.output(out2,GPIO.LOW)
        GPIO.output(out3,GPIO.HIGH)
        GPIO.output(out4,GPIO.HIGH)

    elif vertical_counter==6:
        GPIO.output(out1,GPIO.LOW)
        GPIO.output(out2,GPIO.LOW)
        GPIO.output(out3,GPIO.LOW)
        GPIO.output(out4,GPIO.HIGH)

    elif vertical_counter==7:
        GPIO.output(out1,GPIO.HIGH)
        GPIO.output(out2,GPIO.LOW)
        GPIO.output(out3,GPIO.LOW)
        GPIO.output(out4,GPIO.HIGH)

def horizontal_shake(p1, p2, p3, p4, magnitude, shake_time = 10):
    """Horizontal stepper motor shaking mode.Can input the shake movement
magnitude and shake times.

    Parameters:
    p1 (int): Stepper motor signal pin1
    p2 (int): Stepper motor signal pin2
    p3 (int): Stepper motor signal pin3
    p4 (int): Stepper motor signal pin4
```

```
    magnitude (int): Shaking magnitude
    shake_time (int): Shake motion total repetition times

   """

    global horizontal_counter

    for shake_counter in range (shake_time):
        for stepmotor_turns in range (magnitude):
            horizontal_right_motion(p1, p2, p3, p4)

        relax_motion(p1, p2, p3, p4, vertical = False)
        time.sleep(0.5)

        for stepmotor_turns in range (magnitude):
            horizontal_left_motion(p1, p2, p3, p4)

        relax_motion(p1, p2, p3, p4, vertical = False)

        time.sleep(0.5)

def horizontal_right_motion_count(p1, p2, p3, p4, magnitude):
    """Horizontal stepper motor moves to the right for a certain counts.

    Parameters:
    p1 (int): Stepper motor signal pin1
    p2 (int): Stepper motor signal pin2
    p3 (int): Stepper motor signal pin3
    p4 (int): Stepper motor signal pin4
    magnitude (int): Stepper motor total move right count

   """

    global horizontal_counter

    for stepmotor_turns in range (magnitude):
        horizontal_right_motion(p1, p2, p3, p4)

    relax_motion(p1, p2, p3, p4, vertical = False)

def horizontal_left_motion_count(p1, p2, p3, p4, magnitude):
    """Horizontal stepper motor moves to the left for a certain counts.

    Parameters:
    p1 (int): Stepper motor signal pin1
    p2 (int): Stepper motor signal pin2
    p3 (int): Stepper motor signal pin3
    p4 (int): Stepper motor signal pin4
    magnitude (int): Stepper motor total move left count

   """
    global horizontal_counter

    for stepmotor_turns in range (magnitude):
        horizontal_left_motion(p1, p2, p3, p4)

    relax_motion(p1, p2, p3, p4, vertical = False)

def sensor_flag_reset():
```

```
    """Reset all the sensor flags. All flag values will be set to 0."""

    global sensor_1_flag, sensor_2_flag, sensor_3_flag, sensor_4_flag,
sensor_5_flag

    sensor_1_flag = 0
    sensor_2_flag = 0
    sensor_3_flag = 0
    sensor_4_flag = 0
    sensor_5_flag = 0

def create_new_directory():
    """Create new file directory whenever a new operation started.

    Returns:
    current_dir (str): directory of the current execution file
    original_dir (str): directory for raw image
    processed_dir (str): directory for framed image

   """

    now = datetime.now()
    # dd/mm/YY H:M:S
    dt_string = now.strftime("%d-%m-%Y %H:%M")
    current_dir = os.getcwd()
    dir_counter = 0

    while True:
        final_directory = os.path.join(current_dir, dt_string)
        if dir_counter != 0:
            final_directory = final_directory + '(' + str(dir_counter) + ')'

        if not os.path.exists(final_directory):
            os.makedirs(final_directory)
            original_dir = os.path.join(final_directory, r'Original')
            processed_dir = os.path.join(final_directory, r'Framed')
            plot_dir = os.path.join(final_directory, r'Plots')
            os.makedirs(original_dir)
            os.makedirs(processed_dir)
            os.makedirs(plot_dir)
            break
        else:
            dir_counter+=1

    return current_dir, original_dir, processed_dir, plot_dir

def uv_led_on():
    """Turn on the 11V powered UV LED."""

    GPIO.output(uv_led,GPIO.LOW)

def uv_led_off():
    """Turn off the 11V powered UV LED."""

    GPIO.output(uv_led,GPIO.HIGH)

def sensor_1_func(channel):
    """Sensor 1 callback function. Will set the sensor flag to 1."""
```

```
    global sensor_1_flag
    sensor_1_flag = 1
    print("\rsensor_1", end='');
#     GPIO.remove_event_detect(sensor1)

def sensor_2_func(channel):
    """Sensor 2 callback function. Will set the sensor flag to 1."""

    global sensor_2_flag
    sensor_2_flag = 1
    print("\rsensor_2", end='');
#     GPIO.remove_event_detect(sensor2)

def sensor_3_func(channel):
    """Sensor 3 callback function. Will set the sensor flag to 1."""

    global sensor_3_flag
    sensor_3_flag = 1
    print("\rsensor_3", end='');
#     GPIO.remove_event_detect(sensor3)

def sensor_4_func(channel):
    """Sensor 4 callback function. Will set the sensor flag to 1."""

    global sensor_4_flag
    sensor_4_flag = 1
    print("\rsensor_4", end='');
#     GPIO.remove_event_detect(sensor4)

def sensor_5_func(channel):
    """Sensor 5 callback function. Will set the sensor flag to 1."""

    global sensor_5_flag
    sensor_5_flag = 1
    print("\rsensor_5", end='');
#     GPIO.remove_event_detect(sensor5)
```

# Appendix G: Codes for state machine of the operation

```
import motor_function as motor
```

```
import RPi.GPIO as GPIO
import time


def initialisation():
    """Setup all the edge detection pin. """

    motor.sensor_flag_reset()

    GPIO.remove_event_detect(motor.sensor5)
    GPIO.remove_event_detect(motor.sensor1)
    GPIO.remove_event_detect(motor.sensor2)
    GPIO.remove_event_detect(motor.sensor3)
    GPIO.remove_event_detect(motor.sensor4)

    GPIO.add_event_detect(motor.sensor5, GPIO.FALLING,
callback=motor.sensor_5_func, bouncetime=1000)
    GPIO.add_event_detect(motor.sensor1, GPIO.FALLING,
callback=motor.sensor_1_func, bouncetime=2000)
    GPIO.add_event_detect(motor.sensor2, GPIO.FALLING,
callback=motor.sensor_2_func, bouncetime=2000)
    GPIO.add_event_detect(motor.sensor3, GPIO.FALLING,
callback=motor.sensor_3_func, bouncetime=2000)
    GPIO.add_event_detect(motor.sensor4, GPIO.FALLING,
callback=motor.sensor_4_func, bouncetime=2000)

    # flag = 0
    # time.sleep(7)
    print("\nInitializing motor variables...")

def remove_event():
    """Remove the event detection. Prepare for next iteration. """

    GPIO.remove_event_detect(motor.sensor5)
    GPIO.remove_event_detect(motor.sensor1)
    GPIO.remove_event_detect(motor.sensor2)
    GPIO.remove_event_detect(motor.sensor3)
    GPIO.remove_event_detect(motor.sensor4)

    # flag = 0
    # time.sleep(7)
    print("\nEvent detection removed. Ready for next iteration.")

def original_pos_check():
    """Check whether or not the handle is initially located at the loading
position."""

    print("\nChecking handle current position...")

    if GPIO.input(motor.sensor5):
        print("\nMoving back to the loading station...")

        motor.vertical_up_motion(27, 17, 22, 18, 1200)

        while True:
            if motor.sensor_5_flag != 1:
        #          motor_counter2 = motor.reverse_motion(motor_counter2, 9,
25, 11, 8) # horizontal right motion
                motor.horizontal_left_motion(9, 25, 11, 8)
```

```
            else:
#                 print("hello")
                motor.relax_motion(9, 25, 11, 8, vertical = False)
                motor.vertical_down_motion(27, 17, 22, 18, 1200)
                break

    motor.relax_motion(27, 17, 22, 18)
    motor.sensor_flag_reset()

def station_1():
    """Moves towards station 1 - aluminum etchant. """

    print("\nMoving to first station... ")
    motor.vertical_up_motion(27, 17, 22, 18, 1200)

    motor.hold_motion()

    while True:
        if motor.sensor_1_flag != 1:
    #         motor_counter2 = motor.reverse_motion(motor_counter2, 9, 25,
11, 8) # horizontal right motion
            motor.horizontal_right_motion(9, 25, 11, 8)

        else:
            print("\nstation_1")
            motor.relax_motion(9, 25, 11, 8, vertical = False)
            break


    motor.vertical_down_motion(27, 17, 22, 18, 1200)
    time.sleep(0.1)
    motor.relax_motion(27, 17, 22, 18)
    time.sleep(0.1)
    motor.relax_motion(9, 25, 11, 8, vertical = False)
#
#
# time.sleep(10)
#
# # go to next location
#
def station_2():
    """Moves towards station 2 - water bath. """

    motor.sensor_flag_reset()

    motor.vertical_up_motion(27, 17, 22, 18, 1200)

    motor.hold_motion()

    while True:
        if motor.sensor_2_flag != 1:
    #         motor_counter2 = motor.reverse_motion(motor_counter2, 9, 25,
11, 8) # horizontal right motion
            motor.horizontal_right_motion(9, 25, 11, 8)

        else:
            motor.relax_motion(9, 25, 11, 8, vertical = False)
            print("\nstation_2")
```

```
            break

    motor.vertical_down_motion(27, 17, 22, 18, 1200)
    motor.relax_motion(27, 17, 22, 18)
    motor.horizontal_shake(9, 25, 11, 8, 100, 12)
#
#
# time.sleep(20)
# # go to next location
#
def station_3():
    """Moves towards station 3 - Acetone bath. """
    motor.sensor_flag_reset()
    motor.vertical_up_motion(27, 17, 22, 18, 1200)

    motor.hold_motion()

    while True:
        if motor.sensor_3_flag != 1:
    #         motor_counter2 = motor.reverse_motion(motor_counter2, 9, 25,
11, 8) # horizontal right motion
            motor.horizontal_right_motion(9, 25, 11, 8)

        else:
            motor.relax_motion(9, 25, 11, 8, vertical = False)
            print("\nstation_3")
            break

    motor.vertical_down_motion(27, 17, 22, 18, 1200)

    motor.relax_motion(27, 17, 22, 18)

    motor.horizontal_shake(9, 25, 11, 8, 100, 12)
#
# time.sleep(20)
# # go to next location
#

def station_4():
    """Moves towards station 4 - IPA bath. """
    motor.sensor_flag_reset()
    motor.vertical_up_motion(27, 17, 22, 18, 1200)

    motor.hold_motion()

    while True:
        if motor.sensor_4_flag != 1:
            motor.horizontal_right_motion(9, 25, 11, 8)

        else:
            motor.relax_motion(9, 25, 11, 8, vertical = False)
            print("\nstation_4")
            break

    motor.horizontal_right_motion_count(9, 25, 11, 8, 600)

    motor.vertical_down_motion(27, 17, 22, 18, 1200)

    motor.relax_motion(27, 17, 22, 18)
```

```
    motor.relax_motion(9, 25, 11, 8, vertical = False)

    motor.horizontal_shake(9, 25, 11, 8, 100, 12)
#
# time.sleep(20)
#

def back_to_loading_pos():
    """Moves towards original loading station for recollection. """

    motor.vertical_up_motion(27, 17, 22, 18, 1200)

    motor.hold_motion()
    motor.sensor_flag_reset()

    while True:
        if motor.sensor_5_flag != 1:
    #         motor_counter2 = motor.reverse_motion(motor_counter2, 9, 25,
11, 8) # horizontal right motion
            motor.horizontal_left_motion(9, 25, 11, 8)

        else:
            print("")
            motor.relax_motion(9, 25, 11, 8, vertical = False)
            break


    motor.vertical_down_motion(27, 17, 22, 18, 1200)

    motor.relax_motion(27, 17, 22, 18)
    motor.relax_motion(9, 25, 11, 8, vertical = False)

def emergency_stop_motion():

    motor.vertical_up_motion(27, 17, 22, 18, 1200 -
motor.vertical_current_pos)
    motor.hold_motion()
    motor.sensor_flag_reset()

    while True:
        if motor.sensor_5_flag != 1:
    #         motor_counter2 = motor.reverse_motion(motor_counter2, 9, 25,
11, 8) # horizontal right motion
            motor.horizontal_left_motion(9, 25, 11, 8)

        else:
            print("")
            motor.relax_motion(9, 25, 11, 8, vertical = False)
            break


    motor.vertical_down_motion(27, 17, 22, 18, 1200)

    motor.relax_motion(27, 17, 22, 18)
    motor.relax_motion(9, 25, 11, 8, vertical = False)
```

# Appendix H: Design Proposal