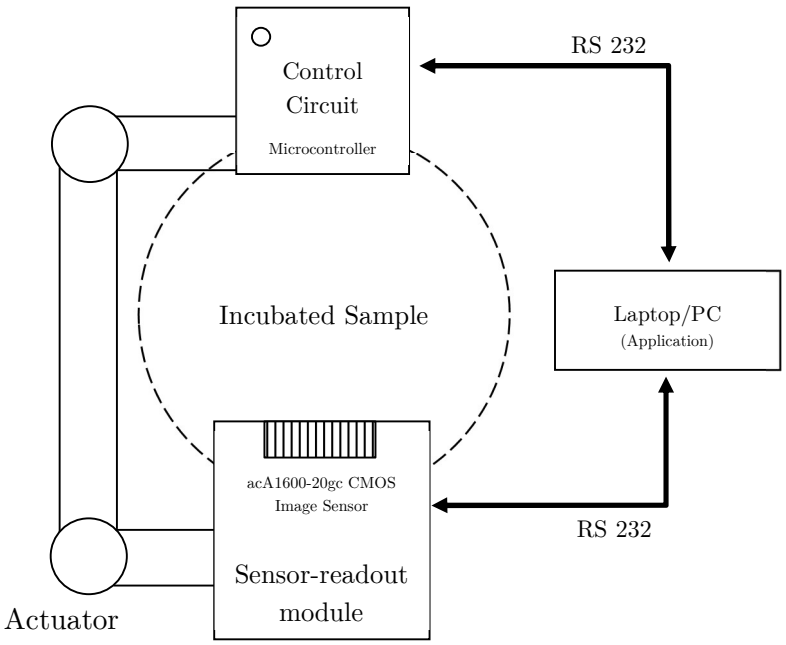
Visual Sepsis Detector

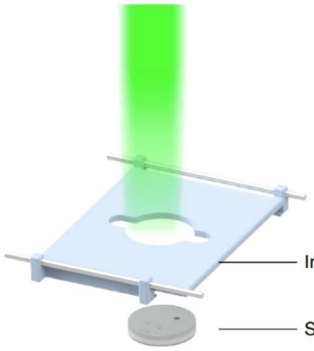
Microscope Computer System

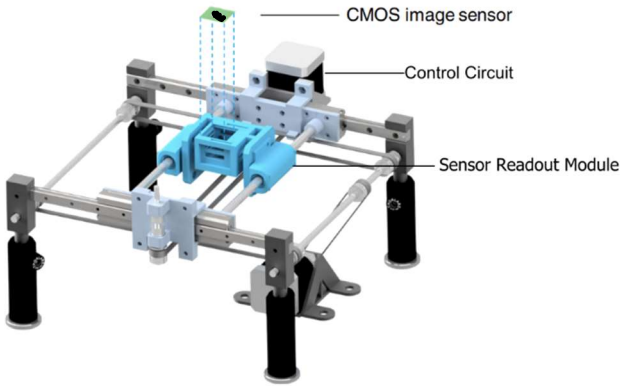
Our microscope has the following components:

- Sensor-readout module (Electrical team).
    - acA1600-20gc CMOS image sensor with a 4x objective lens attached.
    - Microcontroller PIC18F4450 for reading image sensor data.
- Translation stage that can move the camera around with stepper motors and two sets of pulleys. Movement powered by two-stepper motors, one for each direction with a controller chip. May need to use 3D printing (Mechanical team).
- RS 232 USB communication between sensor-readout module, translation stage and external PC (Electrical & Mechanical team).
- Control circuit (Electrical team)
    - Houses the central microcontroller: Raspberry Pi 4 Model B.
    - Stepper motor driver carrier to control translation stage.
    - Transistor-based digital switch (SUP75P03-07) to control the image sensor.
- Microscope incubator housed by a 3D-printed frame (Mechanical team).
- External Laptop/PC running application (Software team).
    - Image processing & image stitching.
    - Deep learning model.

This is a bench-microscope with a single image sensor with a microcontroller connected to it. It will use one-dimensional (1D) array CCD/CMOS image sensor inside a sensor readout module. The two-axis translation stage allows us to change where the microscope is looking at. We will use acA1600-20gc from Basler as our image sensor, which has 1624 × 1234 pixels resolution meaning each pixel is around 4.4 μm × 4.4 μm.

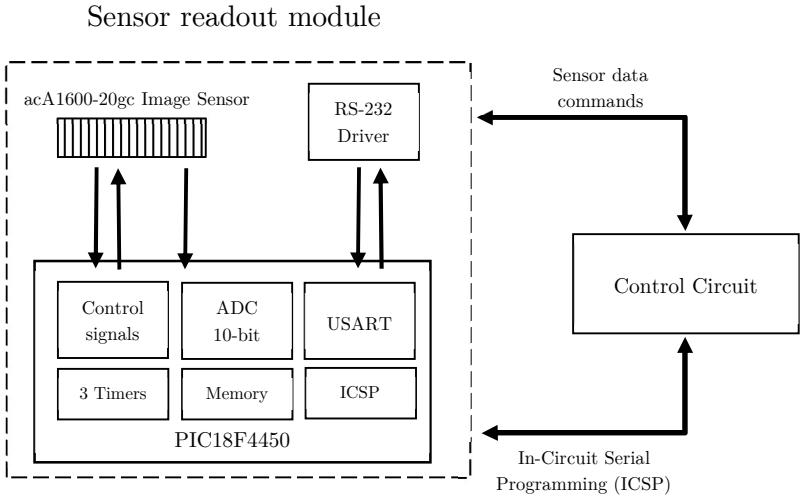






Sensor readout module

An SRM is responsible for reading out the signals from all the pixels composing each row and each column of the array to assemble an image from the photodiode charge accumulation data. The signals from all the pixels must be accurately detected and measured. We will use MATLAB's Instrument Control toolbox to handle the RS232 communications. We will use a PIC18F4450 microcontroller inside the sensor readout module. Our image sensor is a cheap one and does not have exactly have full computer-like capabilities that a regular full camera might have.



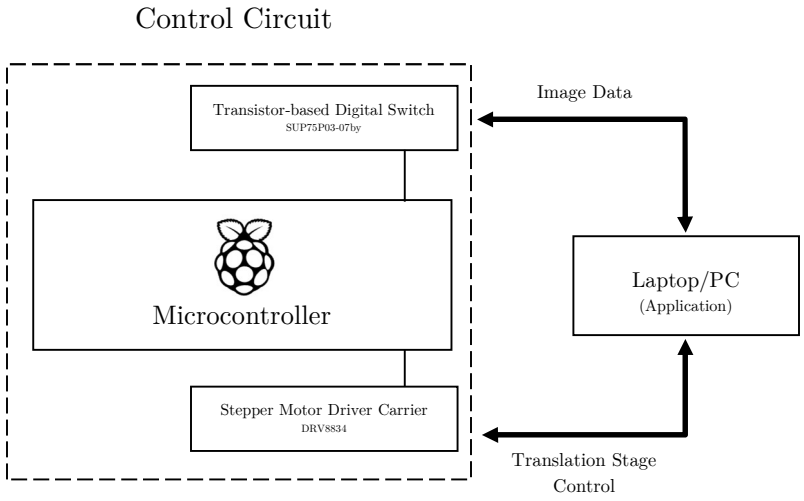
ICSP (In-Circuit Serial Programming) connection allows us to program the microcontroller to allow control of the image sensor from the Control Circuit outside.

The RS-232 is a serial port used to communicate between the sensor readout module and the PC. Uses a baud rate of 59200 bps. Timeslot width (time to transfer one unit of data) is 900 µs. We will say $T_{\text{cycle}} = 900$ µs. Therefore, the acquisition of all 1024 by 1024 sensor pixels takes approximately one second. We'll try RS-232 but it's highly subject to change.

There are some areas that the image sensor covers that are not necessary. We can use the ICSP functionality to specify a region of interest (ROI). ROI is the number of pixels that are being recorded. With a lower ROI, the acquisition rate is increased. We should be able to control the ROI through the external PC application.

Control Circuit

The control circuit will be used to get data from the image sensor to the Python application and control the translation stage. It will house the microcontroller (a good choice would be an Arduino Micro but to be decided by the software team), control chips for the motor drive (an example would be DRV8834 from Pololu) and a digital switch for controlling the CMOS image sensor connection (we will use SUP75P03-07 Vishay Siliconix).



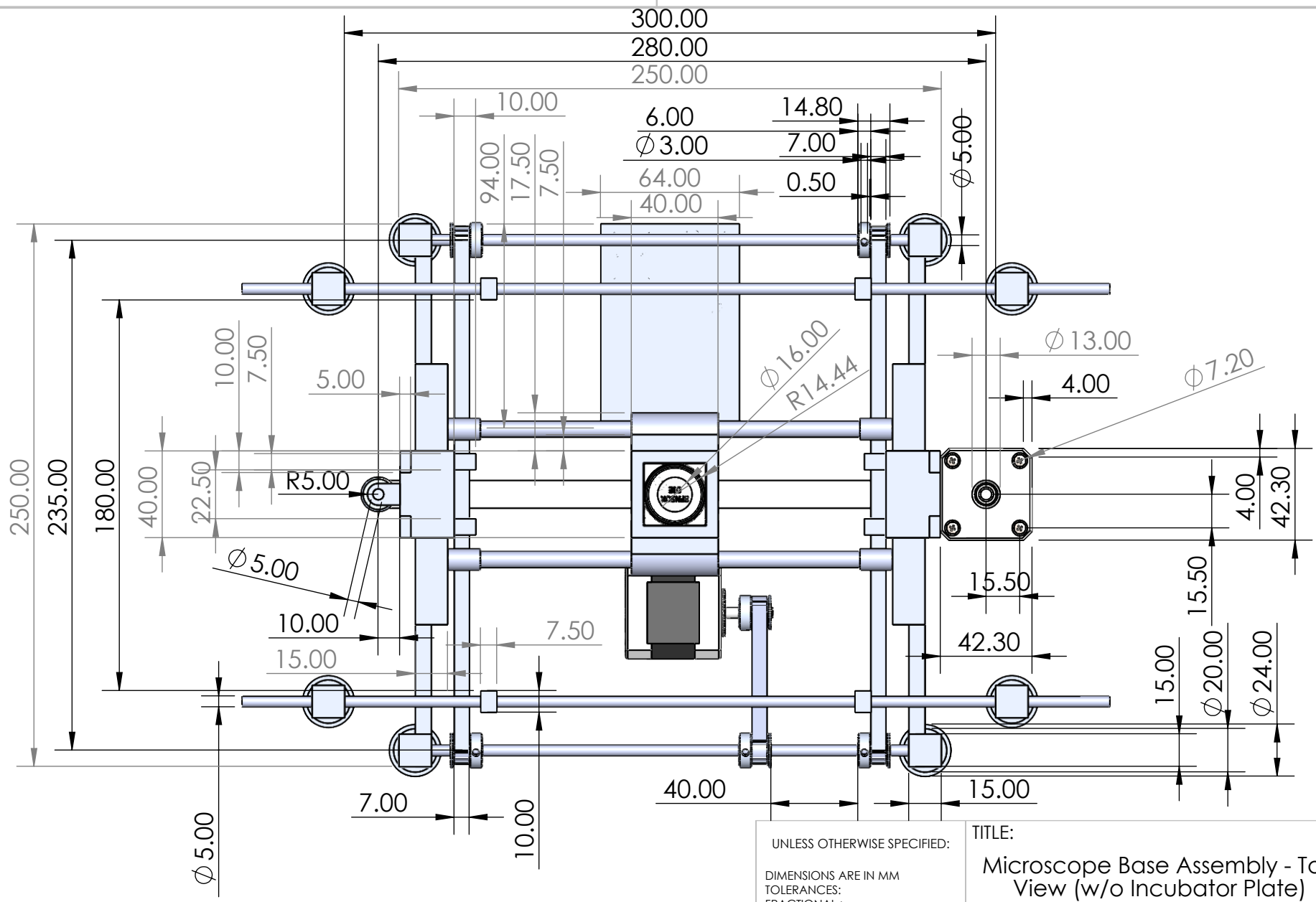
Translation Stage (by Mechanical team)

We will use custom actuators powered by two-stepper motors that move the microscope lens around the sample. This approach can make our device cheaper while still generating a high–resolution image. We call this XY axis area the translation stage. Both electrical and mechanical engineers may need to work together to set up the translation stage.

Incubator (by Mechanical team)

The incubator will be within the translation stage. It keeps the agar plate sealed away from outside interference/particles. In a UCLA study of an automated microscope, they used a special microscope incubator (Tokai Hit Stage Top® Incubator) housed by a 3D-printed frame [1]. The frame appears to be around 15-20 mm thick, which is as thick as the agar plate.

The agar plate that the lens will rotate around is 60 mm in diameter and around 15 mm thick. The area of the translation stage may depend on the actuators/movement component used but should be approximately 180 mm by 180 mm. This size is not specifically required for anything, but the agar plate must be 60 mm in diameter.



UNLESS OTHERWISE SPECIFIED:

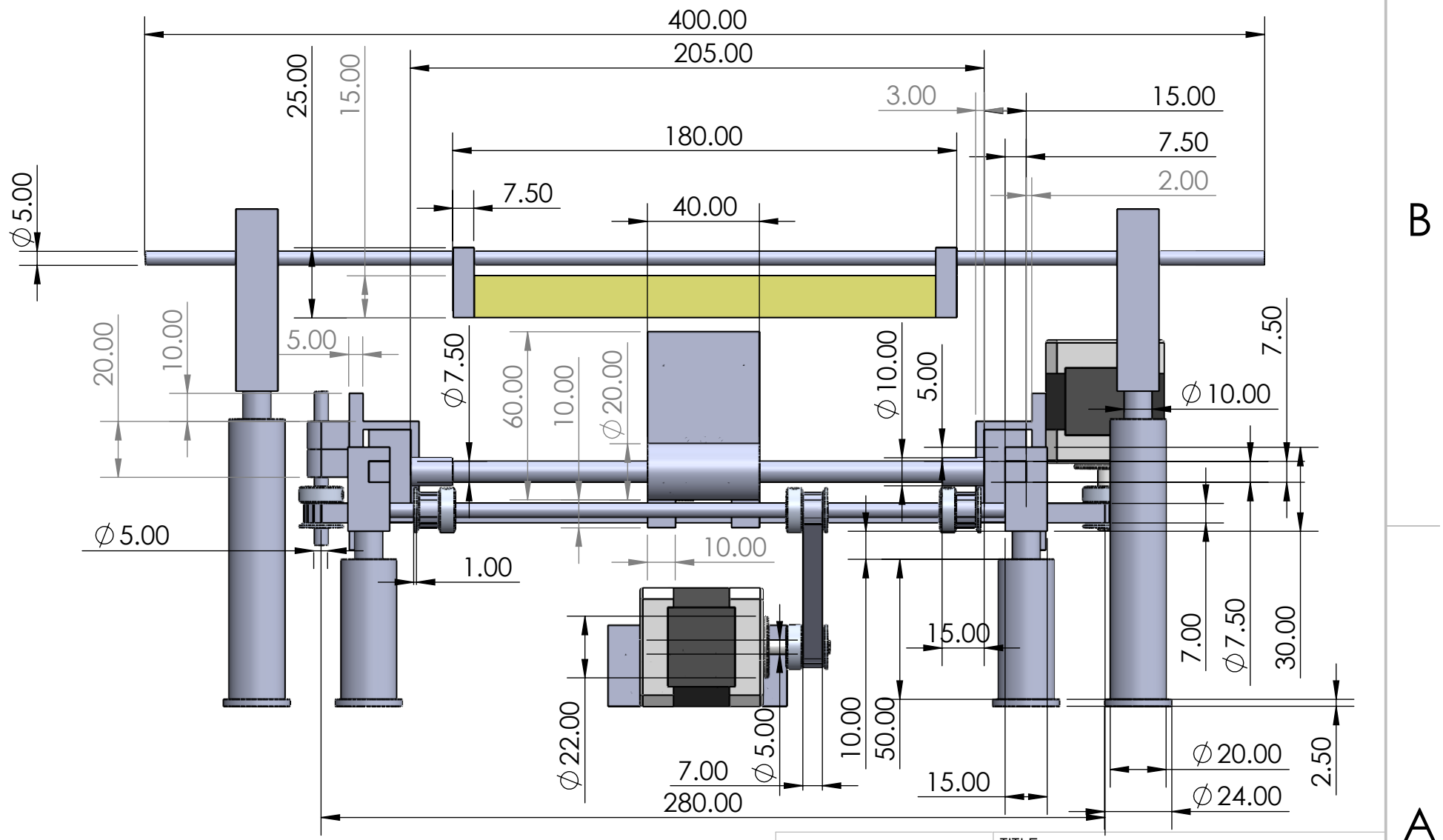
DIMENSIONS ARE IN MM
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±

COMMENTS:

TITLE:

Microscope Base Assembly - Top View (w/o Incubator Plate)

SIZE		NAME	DATE	REV
A	DRAWN	CG	08/17/23	A
SCALE: 2/5				SHEET 1 OF 5



UNLESS OTHERWISE SPECIFIED:

DIMENSIONS ARE IN MM
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±

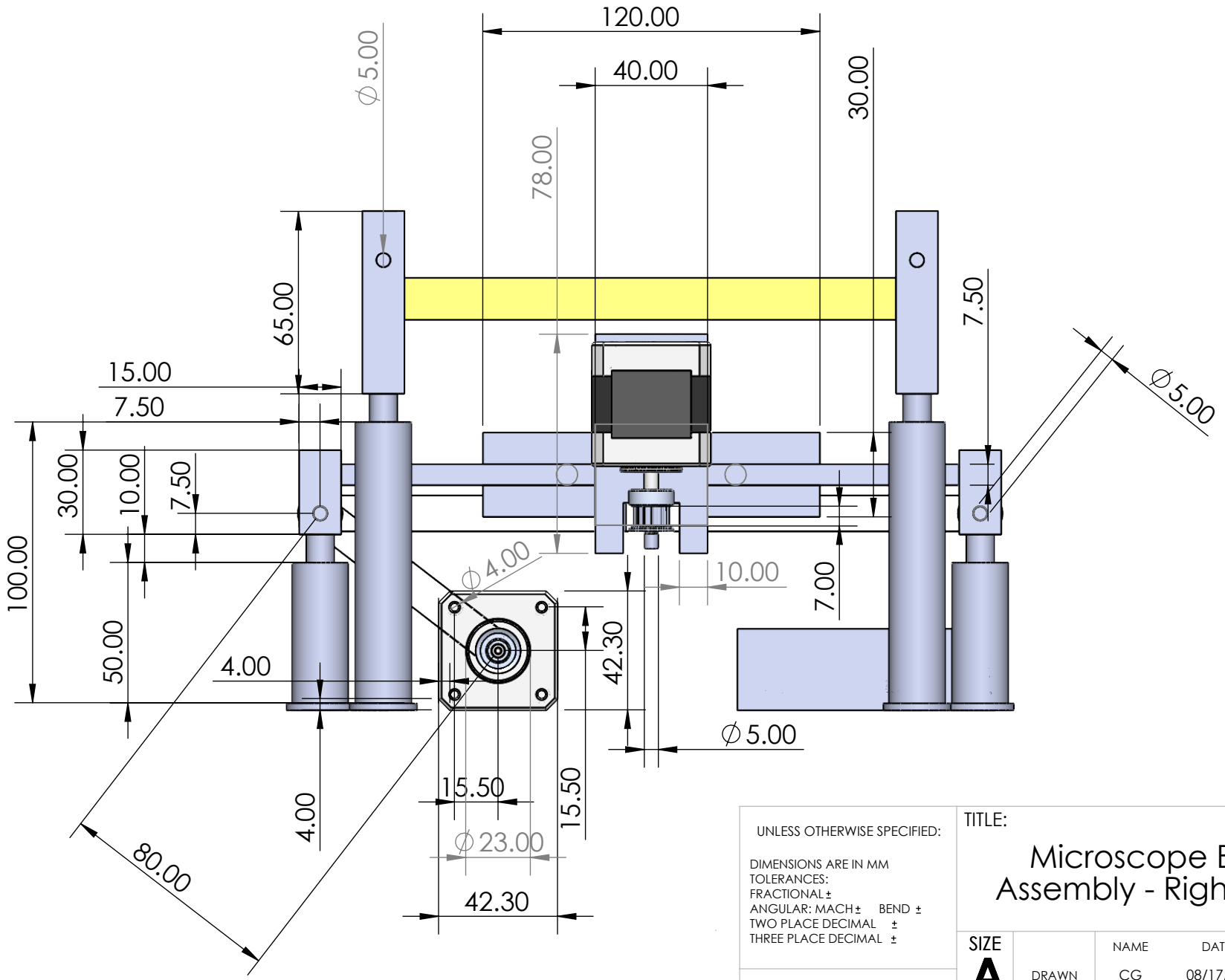
COMMENTS:

TITLE:

Microscope Base Assembly - Front View

SIZE		NAME	DATE	REV
A	DRAWN	CG	08/17/23	A

SCALE: 1:2 SHEET 2 OF 5



UNLESS OTHERWISE SPECIFIED:

DIMENSIONS ARE IN MM
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±

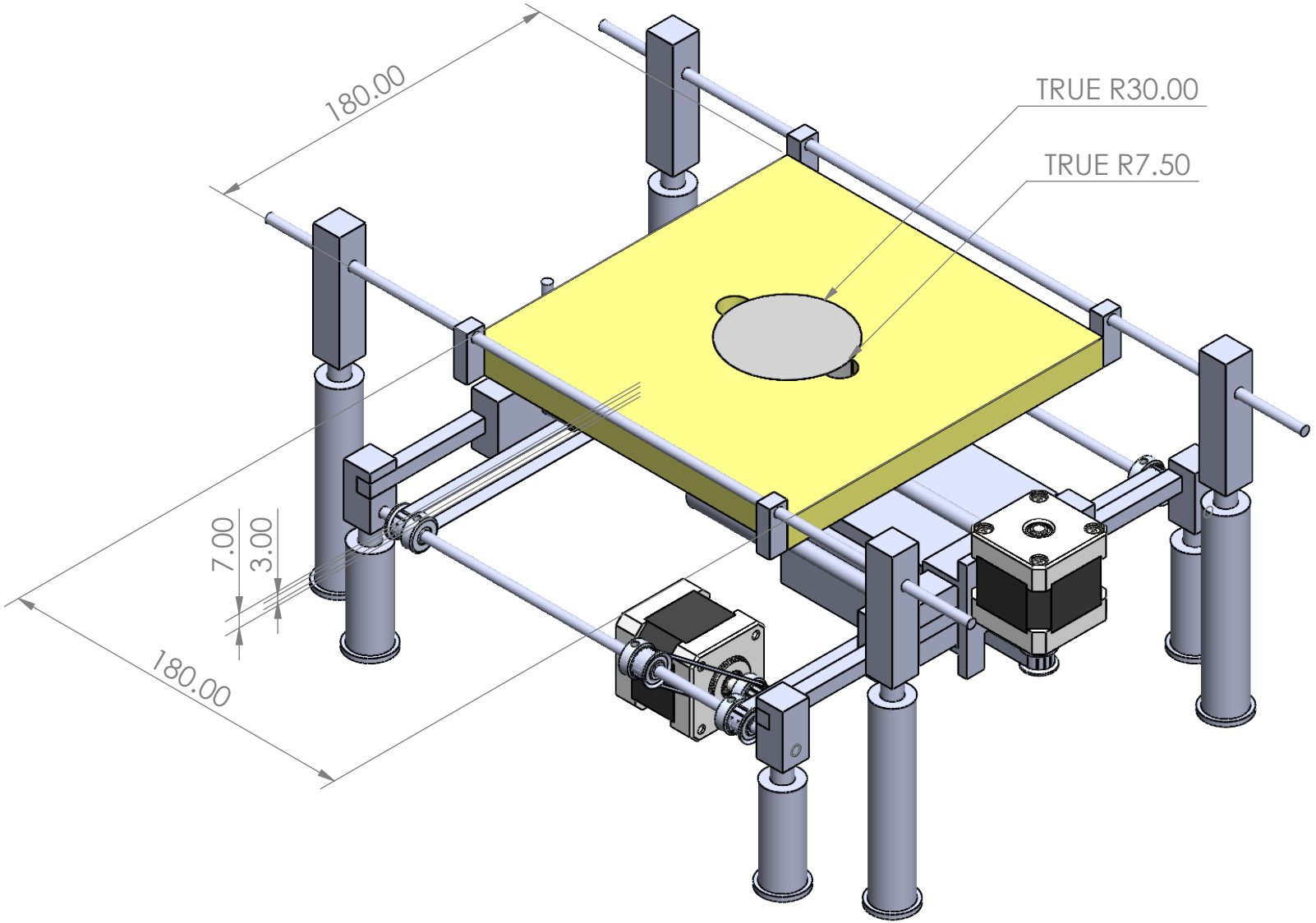
COMMENTS:

TITLE:

Microscope Base Assembly - Right View

SIZE		NAME	DATE	REV
A	DRAWN	CG	08/17/23	A

SCALE: 1:2 SHEET 3 OF 5



UNLESS OTHERWISE SPECIFIED:

DIMENSIONS ARE IN MM
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±

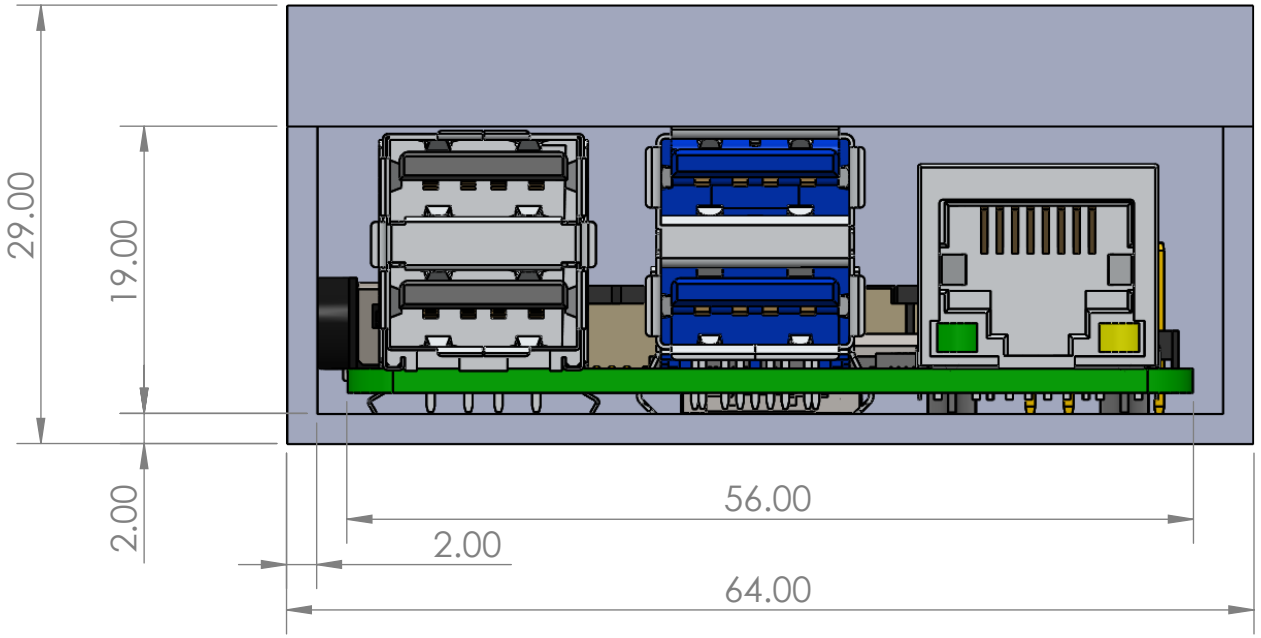
COMMENTS:

TITLE:

Microscope Base Assembly - Iso View

SIZE		NAME	DATE	REV
A	DRAWN	CG	08/17/23	A

SCALE: 2/5 SHEET 4 OF 5



UNLESS OTHERWISE SPECIFIED:

DIMENSIONS ARE IN MM
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±

COMMENTS:

TITLE:

Control Circuit + Casing Only

SIZE		NAME	DATE	REV
A	DRAWN	CG	08/17/23	A

SCALE: 1/1 SHEET 5 OF 5

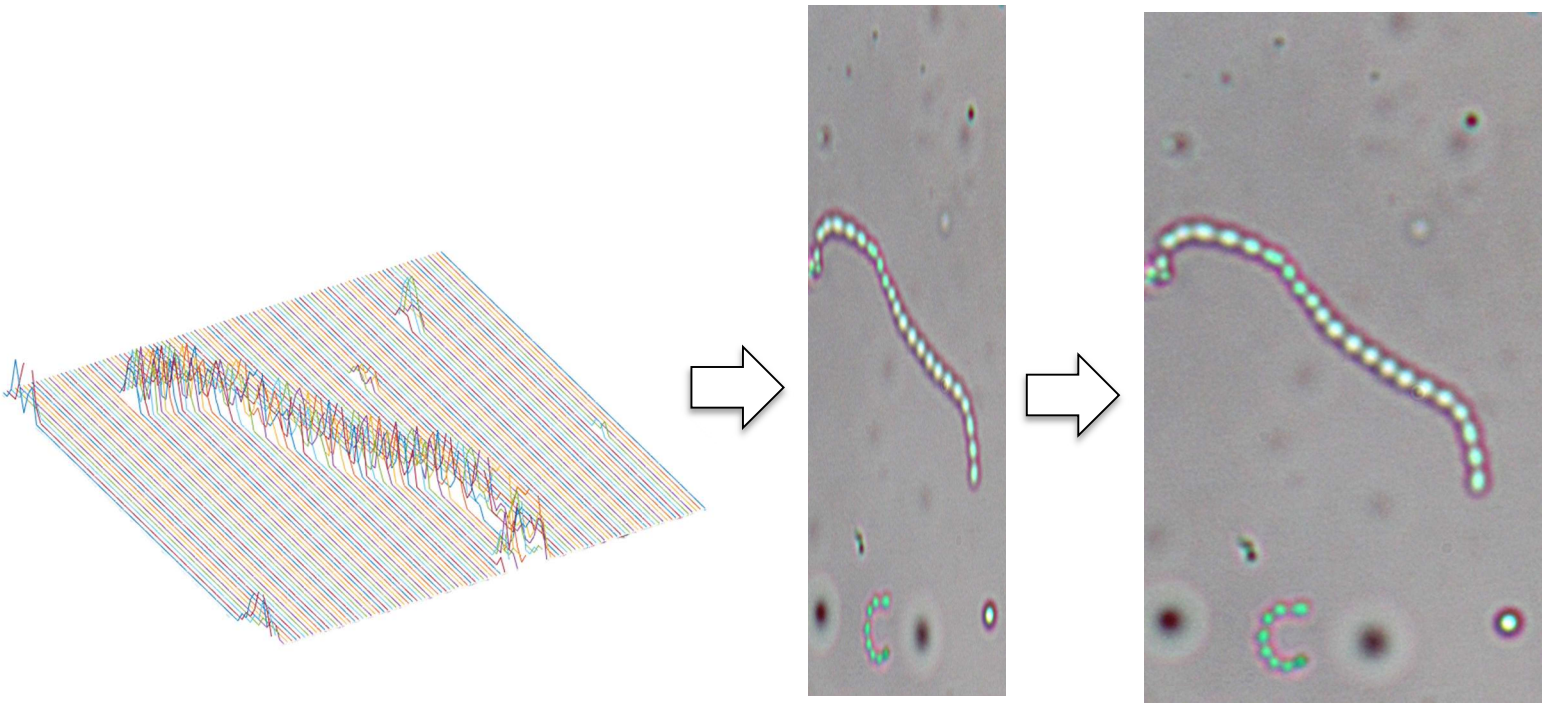
Translation Stage (by Electrical team)

Communications settings for the translation stage through the RS232 will use 9600 baud, no hand shaking, no parity, and one stop bit. Instructions will consist of a group of 6 bytes.

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Unit #	Command #	Data (LSB)	Data	Data	Data (MSB)

Bytes 3, 4, 5 and 6 are data in long integer, 2's complement format with the least significant byte transmitted first. How the data bytes are interpreted depends on the command byte (Byte 2). Essentially each byte will receive data from a single pixel and use it to build a coherent 2D image.

In our application, we will have several samples of different microorganisms/bacteria. They will be 3D wave objects of certain bacteria which will look like (2) in the figure below. The device will process the image (3) into what we see on (4). (1) reflects the data from one sensor readout. What we're seeing here is what the sensor accepts as input and therefore what we need to feed into it to get an image.



(a) the image sensor creates a 3D graph out of a series of 2D graphs that each of its photon cells detects. (Oversimplified)

(b) 3D graph gets converted into an unprocessed 2D image through image processing.

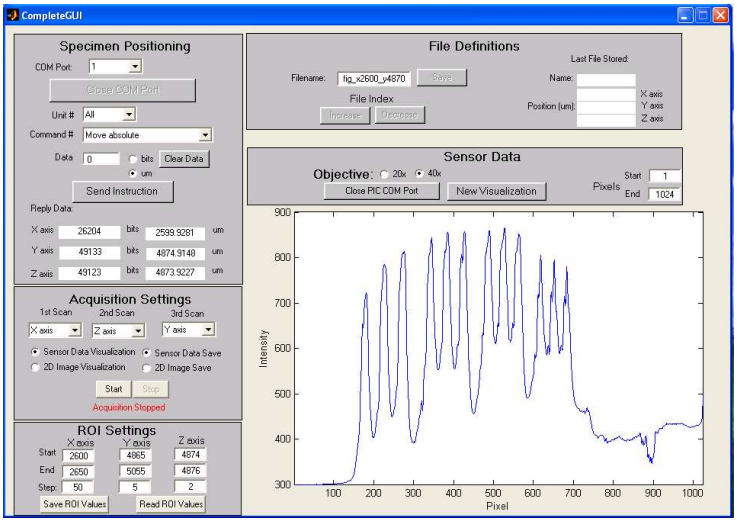
(c) We can use more image processing to turn it into a coherent image. From there we would feed it to the ML model.

Proposed GUI

Our GUI that allows us to see what the device is recording. This GUI will also allow us to control the translation stage. In general, our GUI will:

- Control the translation stage with actuators.
    - User can control the order of scanning in the three axes.
    - Use can control the region-of-interest (ROI) in the three axes.
    - Sends commands to the actuators through the RS232 serial port.
- Sensor data readout
    - Send a command to the sensor readout module to signal acquisition start through the RS232 serial port.
    - Receive the sensor data through the RS232.
- Image
    - Real-time visualization of each sensor readout, i.e., 1D image that shows sensor raw data.
    - Visualization of each 2D image as soon as the acquisition of the set of sensor data is completed.
- Data-files creation
    - Store data files (.csv, Excel, image files) in computer hard disk.
- Deep learning/Computer vision component
    - Functionality that begins the thorough scanning of a sample and applies the computer vision model. It will allow us to see the results of the model.
- Allow alternating between Simulation Mode and Device Mode

Any GUIs we make for the sake of simulation will then be easily used to control the actual fabricated device. The figure below is an example of such a GUI:



Simulation Mode and Device Mode

We will have two modes that we can switch in between through the GUI:

1) Simulation mode – the GUI will control/monitor a simulated 3D device on Simulink.
2) Device mode – the GUI will control/monitor actual fabricated device.

We can use both modes to develop our device.

As a sidenote, make sure MATLAB and Simulink are used purely for the sake of simulation. They should not be deployed on the active device. We will be using Python for the deployed device.

Image Processing & Image Stitching

https://www.nature.com/articles/s41377-020-00358-9 is an excellent resource for the image processing and analysis required for this device. See the "Image processing and analysis" section. Essentially the translation stage moves the image sensor around the sample and records multiple images and then stitches them together into one coherent image. The following is an oversimplification of what they've done:

Five major image processing steps were used for the classification and counting of colonies.
1. Image stitching to obtain the image of the entire plate area.
2. Colony candidate selection by differential analysis
3. Deep neural network-enabled detection of growing bacterial colonies
4. Colony counting
5. Calculating of the imaging throughput

Our image sensor is 1624 px $\times$ 1234 px. Each pixel is 4.4 μm $\times$ 4.4 μm.
That means each image that the sensor takes covers 7145.6 μm $\times$ 5429.6 μm.
Meaning a single image of the sensor covers 7.15 mm $\times$ 5.43 mm of the 60 mm agar plate.
We will need approximately ~80 pictures to cover the entire agar plate.

Image Quality Assessment

There are two approaches to assess the quality of an image, quantitative and qualitative. The quantitative approach involves using two special functions called the Modulation Transfer Function (MTF) which measures contrast (clarity) and the Point Spread Function (PSF) which measures resolution. MTF is most useful for microscopy applications:

$$M = \frac{I_{\max} - I_{\min}}{I_{\max} + I_{\min}}$$
$I_{\max}$ = maximum intensity value
$I_{\min}$ = minimum intensity value

Or

$$MTF(\xi) = \frac{M_{\text{output}}(\xi)}{M_{\text{sinusoidal}}(\xi)}$$

There are many existing Python libraries/functions that already implement MTF.

Preparation of samples

Preparation and experiments may best be carried out by health sciences, medical, microbiology, and/or molecular biology members of the team, especially if ever dealing with infectious samples and blood samples. Here is a sample preparation for an experiment that will test out the device for monitoring E. coli.

Bacterial Culture: Use E. coli Castellani and Chalmers as the culture organism.
Preparation of Agar Plates: Mix CHROMagar™ ECC (8.2 g) with 250 mL of reagent-grade water using a magnetic stirrer bar2. Heat the mixture to 100 °C on a hot plate while stirring regularly. After cooling the mixture to ~50 °C, dispense 10 mL of the mixture into Petri dishes45.
Inoculation: Prepare a bacterial suspension in a phosphate-buffered solution (PBS). The concentration of the suspension is measured using a spectrophotometer and then diluted in PBS to a final concentration of 1–200 CFU per 0.1 mL [1].

Bacterial Growth

For the success of the device, it will be necessary to simulate the potential 24-hour period that it will take for bacteria growth or search for a difficult to find microbes. Within our Simulink simulation, we can use preexisting MATLAB libraries and tools to simulate bacterial growth [4]. As time goes on, the device will have to perform multiple scans of the agar plate to see if any microbe is in sight or if any colony is beginning to emerge, and to differentiate mere dust particles. Such a model for bacterial growth assumes that the population grows at a rate proportional to the current population size. The differential equation is given by:

$$\frac{dN}{dt} = rN\left(1 - \frac{N}{K}\right)$$

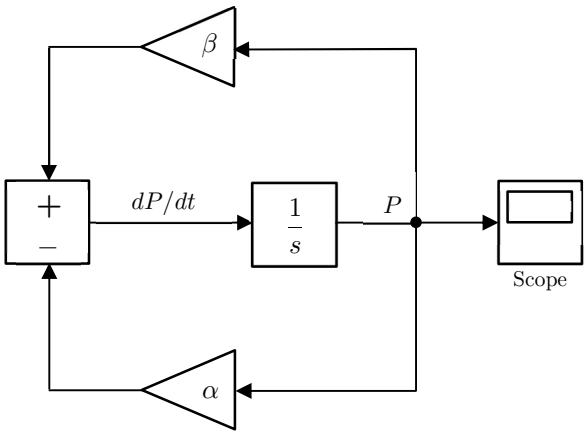
where N = population size
r = rate of increase (birth rate – death rate)
K = carrying capacity

A different differential equation we may also consider using is one that considers births and deaths separately:

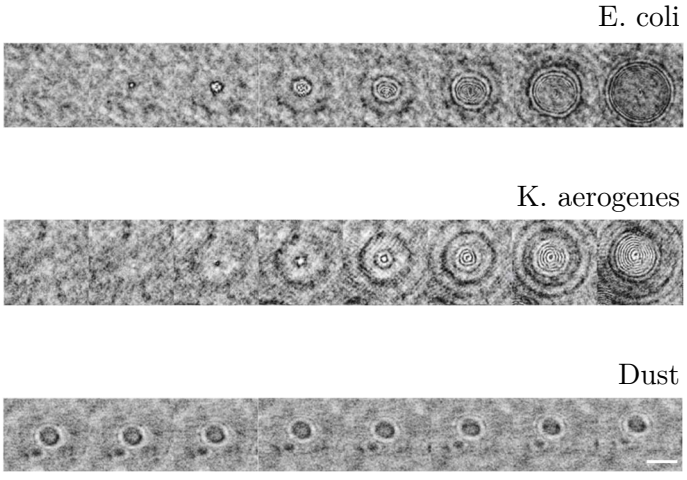
$$\frac{dP}{dt} = \beta P - \alpha P$$

where P = population size
β = birth rate
α = death rate

A proposed Simulink model for the latter option will use integrator blocks and gain blocks which will then place a starting microbe or a colony within a randomized location on the plate:



Proposed Simulink block diagram model for bacterial growth, α and β values to be determined for specific bacteria (i.e. E. coli)

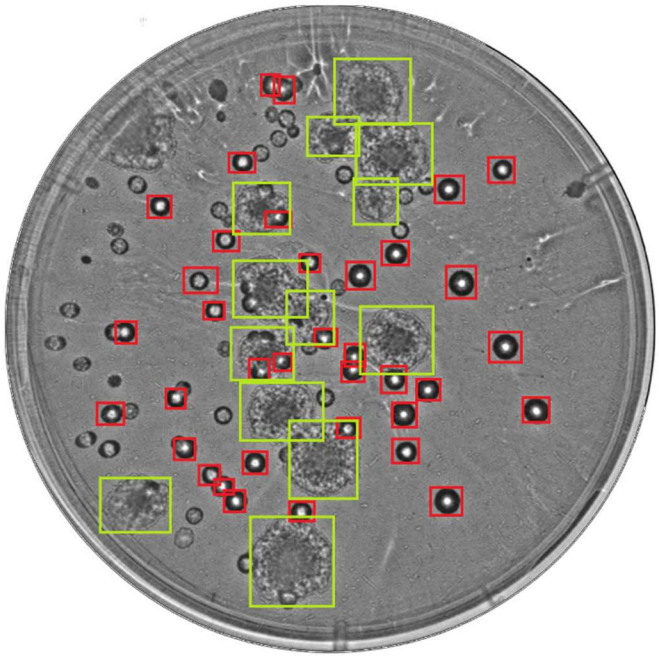


(1)-(2) growth of E. coli and K. aerogenes colonies over a span of an hour to be simulated in Simulink. (3) dust particles may easily be mistaken as a colony but could corrected through time analysis by ML model.

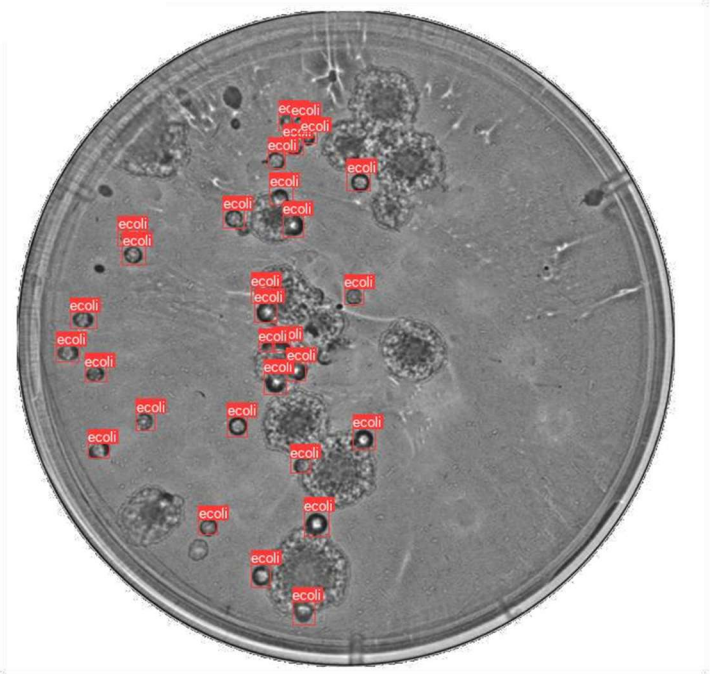
Computer Vision Classification Model

We've had success so far using the Python YOLOv5 (You Only Look Once) computer vision modules for sample images of E. coli and K. aerogenes. We were able to secure sample training data from a UCLA study authors who had similar microscope properties to ours [1]. The model so far can differentiate, segment, and classify E coli. and K. aerogenes colonies using a small collection of images. Most of which also get modified via rotations and alterations to create a larger quantity of training data. On the left sample at the bottom there is an original image and on the right there is an altered image who had some colonies removed and some added. Currently little data exists of samples who have been zoomed in and had image stitching performed to it. Such image data has extremely large resolutions, are a very large file size and are difficult to store. We would need to gather our own images with this device soon so that we may move from the colony level to the microbe level.

The method we've been using so far is known as object recognition and comes with its own functions in YOLOv5 and we have been incredibly successful with it so far. Originally we had tried classifying everything on the plate but we've moved to focus exclusively on E coli., as it is the most important strain to study specifically for bacterial infections.



First iteration of the model



Most recent iteration, as of 9/9/23; it is better capable of finding E coli. stuck inside other particles, very important for blood samples.

Plan of Action

Our main goal is to:

- Simulate the entire device in Simulink with all realistic electrical, mechanical, software and biological components.
- Fabricate the device/order necessary parts.

Here is a proposed roadmap that we will follow for now:

<table>
<tr><th>Electrical team</th><th>Mechanical team</th><th>Software team</th></tr>
<tr><td>1) Work on the sensor-readout module and control circuit in Simulink.
The sensor-readout module is connected to the control circuit via USB or RS-232 (can be either).
The control circuit is connected to the external PC via USB or RS-232.
The sensor readout module contains the image sensor which sends its data to the control circuit.
Control circuit will only contain the microcontroller and a digital switch for now.</td><td>1) Work on the CAD design of the translation stage that incubates the agar plate. Transfer SOLIDWORKS/Inventor model into the Simulink project via SimMechanics plugin.
It has a base with legs.
Has an incubator that covers a 60-mm diameter agar plate that is 15 mm thick.
Area of the stage should be approximately 180 mm by 180 mm, which may depend on actuators.
Create the frame that will house the sensor readout module. It will fit our image sensor which is 7.2 mm $\times$ 5.4 mm, but account for up to three 2-3 mm thick wires that will need to come out of it.</td><td>1) Figure out a workflow/code for training the model and then transferring the model to a microcontroller (decide on which microcontroller to use and let the Electrical team know).
Decide on a microcontroller and let the electrical team know the microcontroller model.
Decide on programming language, ML framework (likely Python + libraries).
Begin a codebase with a simple foundation for performing computer vision with deep neural networks.</td></tr>
<tr><td colspan="3">Integrate everything together by project lead.</td></tr>
<tr><td>2) Begin work on the control circuit and its communication between the sensor-readout module, translation stage and external PC.</td><td>2) Simulate the translation stage; allowing the image sensor to move around the sample. Work closely with the electrical team.</td><td>2) Create the GUI. Will allow us to control the simulated device and ultimately the fabricated device. See “Proposed GUI”.</td></tr>
<tr><td colspan="3">Integrate everything together by project lead; add MATLAB + Simulink unit tests.</td></tr>
<tr><td>3) Order necessary parts and fabricate the device. Improve if failing unit tests.</td><td>3) Order necessary parts and fabricate the device. Improve if failing unit tests.</td><td>3) Work on the image processing and image stitching.</td></tr>
<tr><td></td><td></td><td>4) Start intensive data collection process and reiterating the model.</td></tr>
</table>

Deviations from original design

It is completely fine to divert from the original design and components and to use components or a different design that you deem a better fit for the purpose of our device. Even if it diverts from the specifications and the output we're expecting, please do it if you feel it's a better choice. Let us know about it through Discord or ClickUp.

Possible Resources

Zaber (https://www.zaber.com/) is a Vancouver/UBC-based engineering company that makes XY and XYZ movement-based actuators as well as entire automated scanning microscopes such as the ones we're making here. It may be worthwhile to reach out to them and ask for any recommendations on what components we should use or if they have any discounts for student projects and/or research projects. Here are some interesting components that Zaber provides:

- Motorized Microscope stages (https://www.zaber.com/products/scanning-microscope-stages)
- Simple linear stages (https://www.zaber.com/products/linear-stages)
- Automated microscope: this one is very expensive, one of our goals is to make sepsis detection as cheap as possible such that it's favourable to use in medical/hospital settings. But it does have a very interesting design (see picture below) that we could take inspiration from for future reiterations of the device (https://www.zaber.com/products/microscopes).

References

[1] H. Wang et al., "Early detection and classification of live bacteria using time-lapse coherent imaging and Deep Learning," Light: Science & Applications, vol. 9, no. 1, 2020. doi:10.1038/s41377-020-00358-9

[2] M. P. Macedo, "A MATLAB-based microscope," MATLAB Applications for the Practical Engineer, 2014. doi:10.5772/58532

[3] T.-C. Poon and T. Kim, Engineering Optics with MATLAB. World Scientific, 2018.

[4] Roman, "Simulation tool for continuous microbial cultivation," MathWorks, https://www.mathworks.com/matlabcentral/fileexchange/43583-simulation-tool-for-continuous-microbial-cultivation (accessed Aug. 31, 2023).