Opentrons Art

Overview | Objective

In this two-day lab, you'll program the Opentrons OT-2 pipetting robot to create stunning, glowing designs by depositing genetically engineered *E. coli* onto black agar plates. These bacteria express fluorescent proteins in vibrant colors, forming "bio-art" that comes to life under UV light. It's your chance to turn cutting-edge biotech, with a peek into automation included, into a canvas for creativity, combining science, automation, and art into a single exciting project!

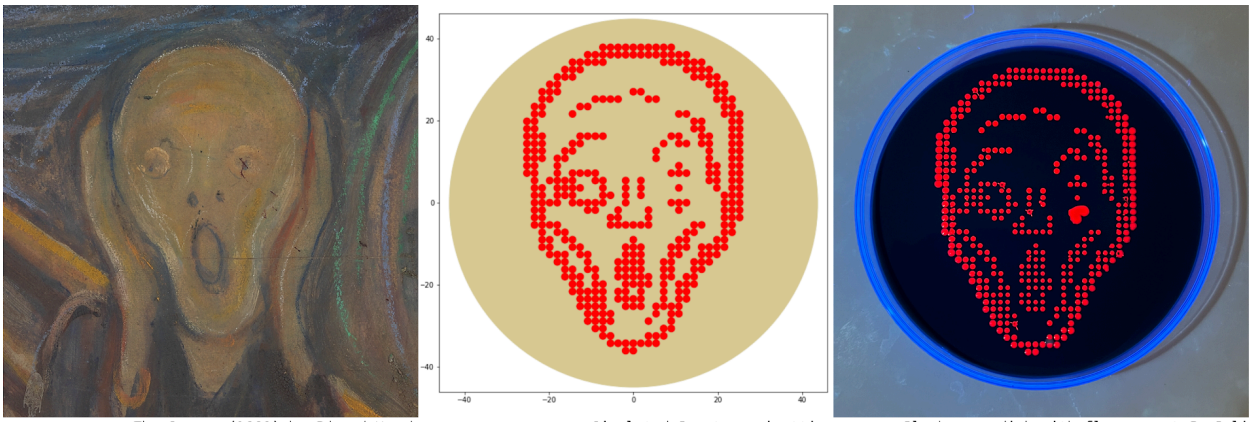
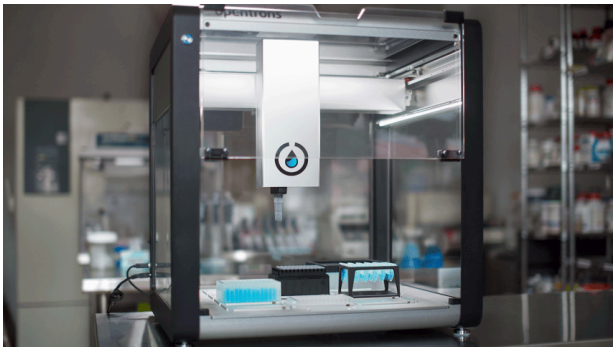


Figure: Daniel Jacobs's (a) inspiration, (b) simulation, and (c) UV-illuminated petri dish

Overview | Concepts Learned & Skills Gained

This week, you will be working with the Opentrons OT-2, a liquid handling robot that's used in various life-science laboratories on a variety of applications, and learn how to incorporate automation into synthetic biology research. This lab will require you to code a script to run on the Opentrons.d You will also learn how to create agar plates, a basic tool in molecular and synthetic biology research.



Learn more from the makers themselves: Introducing the OT-2 Pipetting Robot

Pre-Lab | Reading

Before diving into programming the Opentrons OT-2, it's important to understand the step-by-step workflow that transforms a simple idea into a precise robotic procedure. The following sections will guide you through the foundational concepts, from writing a paper protocol to running a simulated protocol, ensuring you are prepared to create your glowing bio-art masterpiece.

"Central Dogma" of Opentrons:

Paper Protocol → Opentrons Protocol → Compiled Protocol

Think of this process as the **central dogma**! Let's go through each of these one by one:

Paper Protocol

These protocols are basically what you have been using in the lab. They are instructions written in plain language. Let's describe an example protocol:

1. Set up a 12-well 22 mL reservoir with four colors of liquid (Green, Blue, Red, & Yellow)
2. Pipette 100 uL of Green into well A1
3. Pipette 100 uL of Blue in well A2

This is a good first step to describing our experiment as it gives us an opportunity to think through our logic. Now we must translate the steps into a form that works with the Opentrons robot.

Opentrons Protocol

Often creating the Opentrons Protocol only requires writing down the Paper Protocol step-by-step using the terms the chosen framework understands. With a GUI framework this might be a series of clicks, with a Python framework this might be a series of simple function calls corresponding to commands directing the robot. For a more regular/generalizable series of commands, Python gives you the power to use the power of a programming language to automate the generation of some or all of these instructions.

There are several alternative frameworks for creating a protocol for Opentrons:

1. Opentrons Python API - a Python library provided by Opentrons which controls their OT-2 and Flex robots

2. PyLabRobot - a Python library which abstracts out the hardware and can run the same code on multiple backend robots
3. Opentrons Protocol Designer - a web-based visual interface for creating protocols, allowing you to enter each operation of the robot by hand in a GUI. Does not require knowledge of any programming language. In the past, students have experienced glitches giving unexpected results, but students have also produced working protocols.

For the purpose of this lab, we want to focus on using the Opentrons Python API in our Google Colab notebook. In that environment, you write Python code which sends commands to the Opentrons robot to control its movements in order to create a design of your choice.

While this may sound daunting to people not familiar with Python, the lab has several examples, and simple but attractive results can be achieved with just a few basic commands by using the examples as a guide. (If you find yourself stuck on expressing your ideas in Python, as always reach out to a TA for help! Also note that while Google Gemini (integrated with Colab) can produce Python code which runs in our simulator and on an Opentrons, sometimes the code it produces doesn't actually do what you ask it to; it may be able to give you some code fragments or example code patterns for you to write your own code based on, or it may be more trouble than it's worth…)

Our Colab contains a custom simulator which implements a small subset of what the Opentrons Python API provides; it's good enough for the purposes of this Lab, but if you're considering programming an Opentrons outside of this Lab, see the additional functionality as explained in the API Reference for ideas of what all you can do with the robot.

Compiled Protocol

The Opentrons App compiles the Opentrons Protocol (expressed in either Python code, or a JSON Protocol file exported by the Protocol Designer) into a series of commands controlling the robot hardware. On the computer connected to the Opentrons, we will use the Opentrons App to load our protocol file, validate the protocol, calibrate the correct tip offset, and execute the Compiled Protocol.

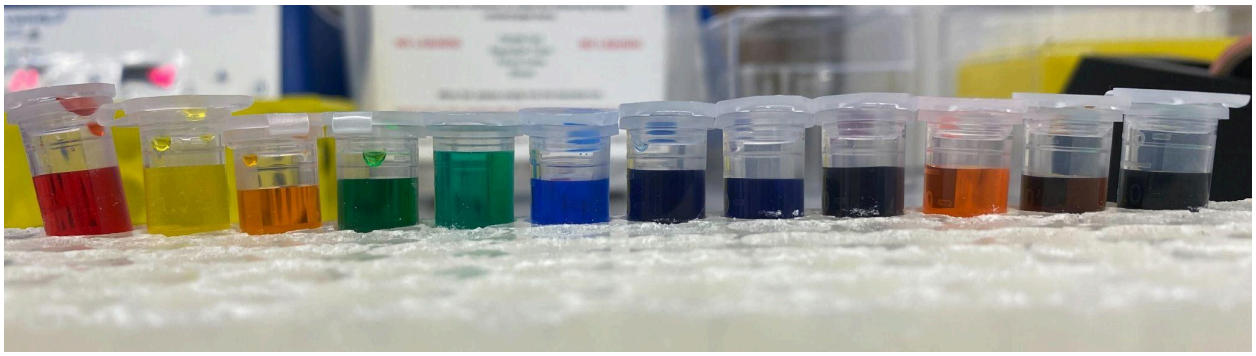
You can run the Opentrons App on any laptop or desktop (even if it doesn't have a robot attached) in order to validate your protocol ahead of time and fix any errors early. Instructions on this process are provided in the "Import a Protocol" section of the Flex robot manual, but should apply equally to the OT-2. This step is not a requirement for this lab.

Simulating your Protocol

Just like you wouldn't want to perform a new musical piece on stage without running through it first in your practice space, you don't want to run the protocol you've just coded up on a real robot using expensive reagents without running through it beforehand in the robot's "practice space" - a simulation. It's always a good idea to first run in simulation every protocol you write (probably many times!), reworking it until you're reasonably confident of a good outcome on the robot.

The Colab we are using for this Lab has its own custom simulation capabilities built in (just run both of your code blocks!). If you're using other frameworks, the Opentrons API can simulate either from the command line or interactively from a Jupyter notebook (such as a different Colab). PyLabRobot has its own simulator. The Protocol Designer GUI shows the final state of the deck visually and every step along the way, effectively simulating the entire protocol as you develop it.

No matter which Opentrons Protocol framework you are using, the protocol can be additionally validated by running the Opentrons App on any laptop or desktop (even without a robot) and loading the protocol as an final simulation/verification step. (Not a requirement for this lab.)



GFP and Friends

Green Fluorescent Protein (GFP) is, as its name suggests, a protein which glows green when illuminated with ultraviolet light. "Fluorescing" is absorbing light at one wavelength - in this case UV - and then re-emitting it at another (typically longer) wavelength - in this case green. GFP, the first such fluorescent protein, was originally isolated from jellyfish and the DNA sequence has since been mutated to produce a rainbow of colors (although to get red (RFP) with its long wavelength and distinct profile a different gene was isolated from coral).

GFP and the other colors are a handy and commonly used way to make the results of genetic engineering visible to the naked eye - as an indicator either that a particular metabolic pathway

is active, or even just that gene splicing was successful. The team that discovered and developed GFP was awarded the 2008 Nobel Prize in Chemistry.

For this lab, the R/G/B/C/YFP genes have been spliced into *E. coli* bacteria which also have genes for resistance to specific (old, some discontinued) antibiotics. The *E. coli* are grown on an agar growth medium which contains those antibiotics so that only these bacteria will grow. We mix charcoal powder into the agar to make it black so that the designs will be more visible.

Chatting with GFP

Large Language Model AI is everywhere these days, including inside your colab! Google's Gemini LLM is integrated into colab and is accessible by clicking the "Gemini" tab near the upper right of the screen. This can provide help by writing Python code which at least runs as legal Python (although making it produce code which has a particular result ***you*** desire can sometimes be quite a challenge…); it may also be helpful in debugging problems in your code.

You can use any LLMs to assist you in your work on this lab, though we ask that you at least try to write your own Python routine first by yourself. The submission form will ask you to describe if and how you used AI in your work on this lab.

The following "AI prompt" is an example of what LLMs can do in this situation - it was able to produce code that should run both in simulation and on the robot, though the design is probably not what a human would produce when given the same prompt. In our experience, Gemini is better at producing figures which are straightforward geometric/mathematical shapes, but its fidelity drops for other figures. Note that each phrase in the prompt was added successively to refine the result after it produced a result with some undesirable characteristic:

```
complete the run() function in the penultimate code block using
2-space indentation to draw a star with 20 points in each arm,
each arm a different color, centered at (0,0), with the farthest
point 35mm from center. use the labware already loaded.  use a
different pipette for every color.  only aspirate what you're
going to dispense.
```

(The first attempt was: "`complete the run() function to draw a star`")

Pre-Lab | Reminder

This lab has a **greater emphasis** on **Python** than other assignments do. If you are not familiar with Python, you can go through tutorials like LearnPython.org. Even if you don't have much experience programming Python, don't be daunted by this – Python is an accessible language with only a few basic principles, and as above you can use LLM assistants when you need help with the details.

YOU SHOULD START YOUR WORK ON THIS LAB RIGHT AWAY – you need to complete this lab on your own **WELL BEFORE** your designated lab time. It needs to be already finished and submitted a day before your robot slot! Unlike other Labs where your work is mainly done during the lab time, in this one the lab time is for you to see the results of your completed work! In some sense, the work here is *entirely* "Pre-Lab!"

2023 student Alice Cai's final plate, without (left) and with (right) UV illumination.

Protocol | Part 1: Fluorescent Bacteria & Black Agar Script

Time Estimate: 2 Hours

Requirements

This assignment is fairly open ended! We ask that you design and implement a custom Python procedure that deposits any pattern of your choice onto a petri dish. You will develop your code in a Colab with a robot simulator, and then submit it to be run on a real robot. In doing so you will learn the basics of programming a pipetting robot to follow a lab procedure, with an attractive result.

Details

As with any lab procedure run on a robot, this lab will involve programming the robot to pick up pipette tips, aspirate reagents from the correct sources, dispense them at the correct locations, and drop the used pipette tips (when done with that reagent) to be ready for the next action.

Those four fundamental actions correspond to these function calls; as also noted in the Colab's introduction, in a couple cases we have provided routines which should assist you in successfully completing this lab:

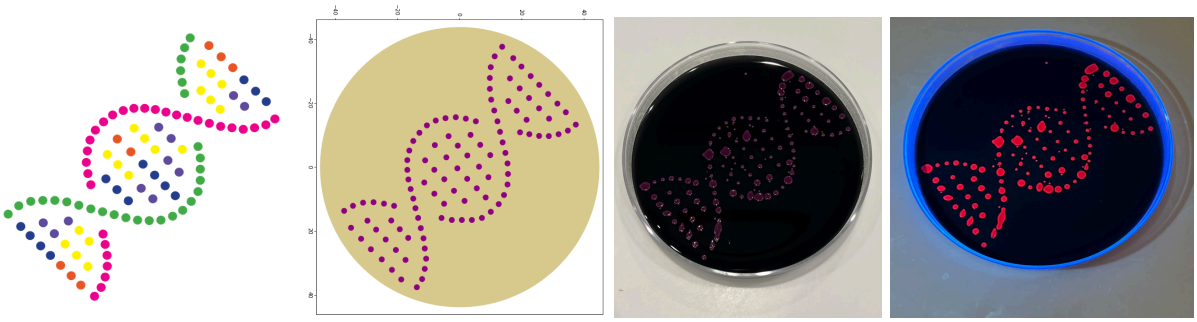
- `pick_up_tip()`
- `aspirate()`, with our `location_of_color()` to help find where we've put reagents
- `dispense()`, though we recommend our `dispanse_and_jog()` for this lab instead
- `drop_tip()`

Plus, you will likely need to use the `.move()` method on Locations you are generating along the way, starting with our provided `center_location`. Your moves will mainly or entirely be in the x & y directions; z is the vertical direction (up is positive z), but your dispensing should be done at z=0 (like `center_location` has) unless you're trying for a special effect by intentionally dispensing higher (but never go below z=0!).

Your robot run starts without any tips! Be sure to use any given tip on a single color only to avoid cross-contaminating your source wells. Finally, it's considered poor form to waste any resources unnecessarily – pipette tips (ideally just one per color), reagents (only aspirate what you're going to use!), etc. The simulator should catch many common errors and help you to produce a robot-ready submission.

Instructions

1. Open the HTGAA25 Opentrons Colab notebook and **make your own copy**. (And then do all your work in your copy, of course.)
2. Follow the instructions and examples in the notebook to write your Python routine which generates your design. Pay due attention to the section at top "Several important notes." Be sure to run the code blocks in the "Prerequisite Code" section at the start of each Colab session. Run your own two code blocks to simulate your results often, as you code each bit of your design!



2023 student Peggy Yin's (a) design, (b) simulation, (c) petri dish, (d) UV-illuminated

Protocol | Part 2: Submission and Running Your Protocol

Time Estimate: 2 Hours

To ensure you're ready for lab time, please follow these steps:

- **Read the pre-lab materials** to familiarize yourself with the concepts and workflow.
- **Create your protocol file** in the Colab notebook by writing and testing your Python routine.
- **Submit your completed protocol** to your TA

Students from MIT/Harvard and Global Committed Listeners will have their submissions run on available robots during the scheduled lab times!

Once complete, please sign up for a time slot with a robot (MIT/Harvard students should sign up for the robot at MIT during Lab hours), and submit your colab block via this Form. Have this submitted AT LEAST ONE DAY before your robot time slot!

If you are having any trouble with scripting, contact us as soon as possible. Do not wait until your scheduled session or you may not be able to complete the assignment.

Post Lab Questions | Mandatory for All Students

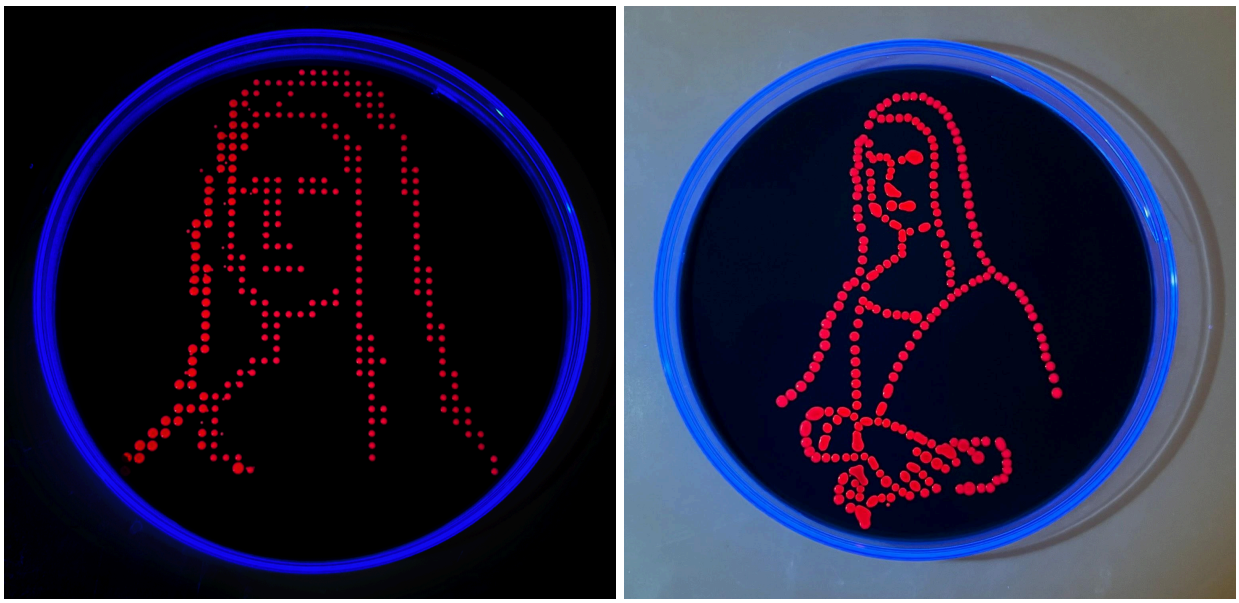
One of the great parts about having an automated robot is being able to precisely mix, deposit, and run reactions without much intervention. This year, a greater emphasis will be placed on utilizing the Opentrons to accelerate your final projects.

For this week, we'd like for you to do the following

(1) **Write a description** about what you intend to do with automation tools for your final project. You may include example pseudocode or Python scripts, procedures you may need to automate, 3D printed holders you may need, and more.
(2) Find and describe a published paper that utilizes the Opentrons or similar automation tools to achieve novel biological applications (eg automated PACE)

While your idea doesn't need to be set in stone, we would like to see core details of what you would automate. This is due right before class and does not need to be tested on the Opentrons for now.

Example: You are creating a custom fabric, and want to deposit art onto specific parts that need to be intertwined in odd ways. You can design a 3D printed holder to attach this fabric to it, and be able to deposit bio art on top. Check out the Opentrons 3D Printing Directory.



Mona "E.coLisas" by 2023 students Jeremy Wohlwend, Jocelyn Keyser (left) and Rodmehr Basidj (right)