Covmatic
Guide: setting up Covmatic – Part 2
Software

Covmatic consists both of hardware equipment (e.g. liquid handling robots, PCR machines, barcode scanners, etc.) and of a software system developed to control and manage the hardware. Each liquid handling robot and PCR machine constitutes a "Station" and is equipped with a laptop and a barcode scanner. From the software point of view, the system consists of a local user interface (enabling trained operators to track and manage the lab process) and of a cloud-based control software (coordinating all the Stations, and storing the digital records associated with each sample).

This Guide will briefly explain how the Covmatic software system works (including both its local and cloud-based components), and how to set it up in your laboratory.

Index:

1. System architecture: an overview

The Covmatic software system is based on two main components:

a. A web application (frontend + backend), and
b. A "local" server.

The "local" server is just a Flask application that controls all the automation procedures. The web application has a frontend, developed with Angular, that helps operators follow protocols and shows the results to managers. The backend, a Flask application, contains all the protocols, stores all the data generated locally (barcodes, automation results, PCR results, protocol executions) and ensures consistency in the operation of the system.

The system works like this:

- The frontend receives the protocols from the web server, and guides the operator through all the steps of the process.
- When the protocol asks for the execution of an automated procedure, the frontend calls the local web server on http://localhost:5001 with an end point that points to the correct automation instructions.
- The frontend saves in the backend the data generated locally and all other relevant information (operator identity, timestamp, etc.).

2. Web application

The web application has two software components:

- A web server (apache) that may implement security stuff (https, authentication) and stores the Angular application files.
- A REST server, developed in Flask, that stores all the system settings (protocols, machines, users) and implements a collection of APIs to check barcode consistency (e.g. the input of a station should be the output of a previous station), store locally generated data, process digital record instructions, and retrieve stored data.

The Web application or Interface, when opened, looks like the picture shown in Figure 1.

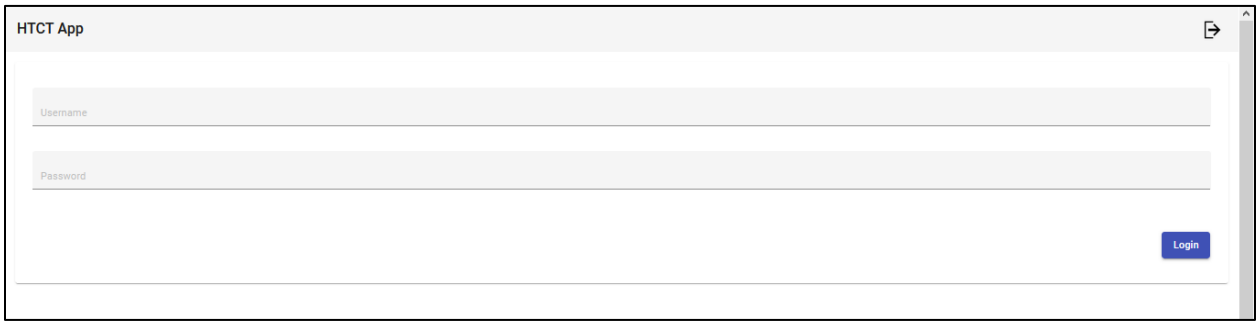


Figure 1: the login window of the Web application or Interface.

It is possible to create different accounts and with different levels of authentication. Figure 2 shows the menu that lets operator manage each machine. The machines are divided according to the type of the Station in which they operate (e.g. Station A, Station B, etc.). Inside each "box" there is the name of the machine and the checkbox related to the calibration of the machine. The system will ask about the calibration status of the machines daily, in order to remind the operators about which machine should be calibrated before starting the process.

Figure 2: the interface allowing operators to manage the hardware machines, and their calibration.

When an operator clicks on the box representing a particular machine, a very simple interface will open (Figure 3). Here the procedure that the operator should execute is shown, step by step. After executing each step, the operator must confirm its successful completion by pressing a button. When following these instructions, operators are not allowed to go back or modify the order of the tasks. This ensures that the prescribed process is correctly executed.

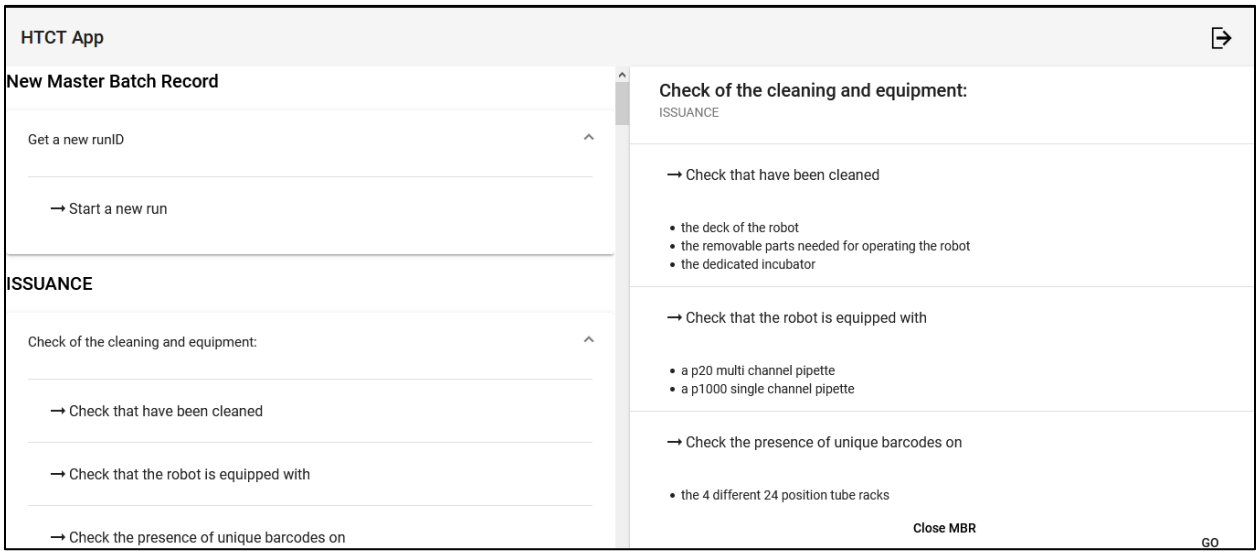


Figure 3: the interface guiding the operators through the series of tasks that must be executed to complete the testing process. On the left part of the screen there is the complete list of process steps for that machine; on the right part of the screen, the step being currently executed is shown in greater detail.

When the operator must enter a barcode in the system (Figure 4), the interface will show an empty field, waiting for the input of the barcode scanner. When the operator uses the barcode scanner, the empty field will be automatically completed by the system, and the operator will be able to advance to the next step.

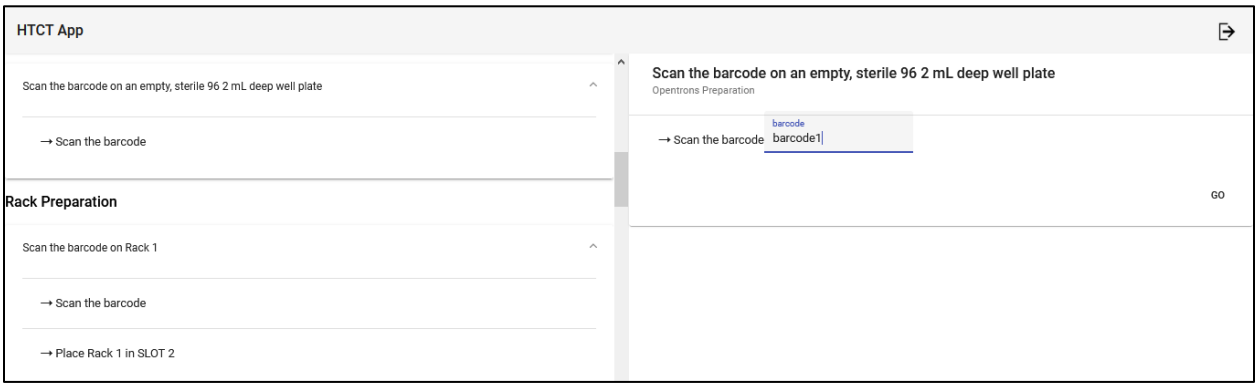


Figure 4: operator interface when a barcode must be entered in the system.

When the process needs the robotic equipment to perform a series of tasks (i.e. a protocol), the interface lets the operator initiate the execution of the protocol by pressing a button (Figure 5).

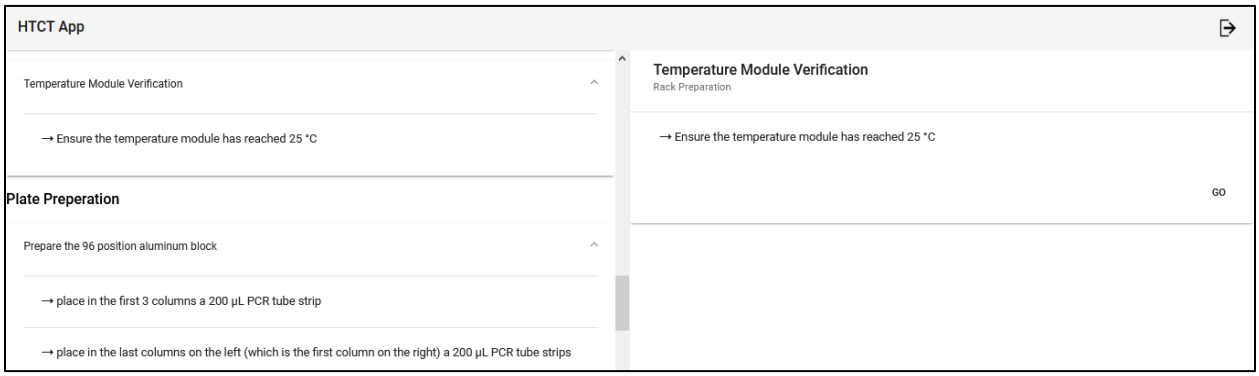


Figure 5: the interface lets operators run protocols on the machines.

The interface also lets operators visualize all the runs that have been executed. It also provides a complete summary (digital records) the process steps, including time stamps and the name of the operator (Figure 6).

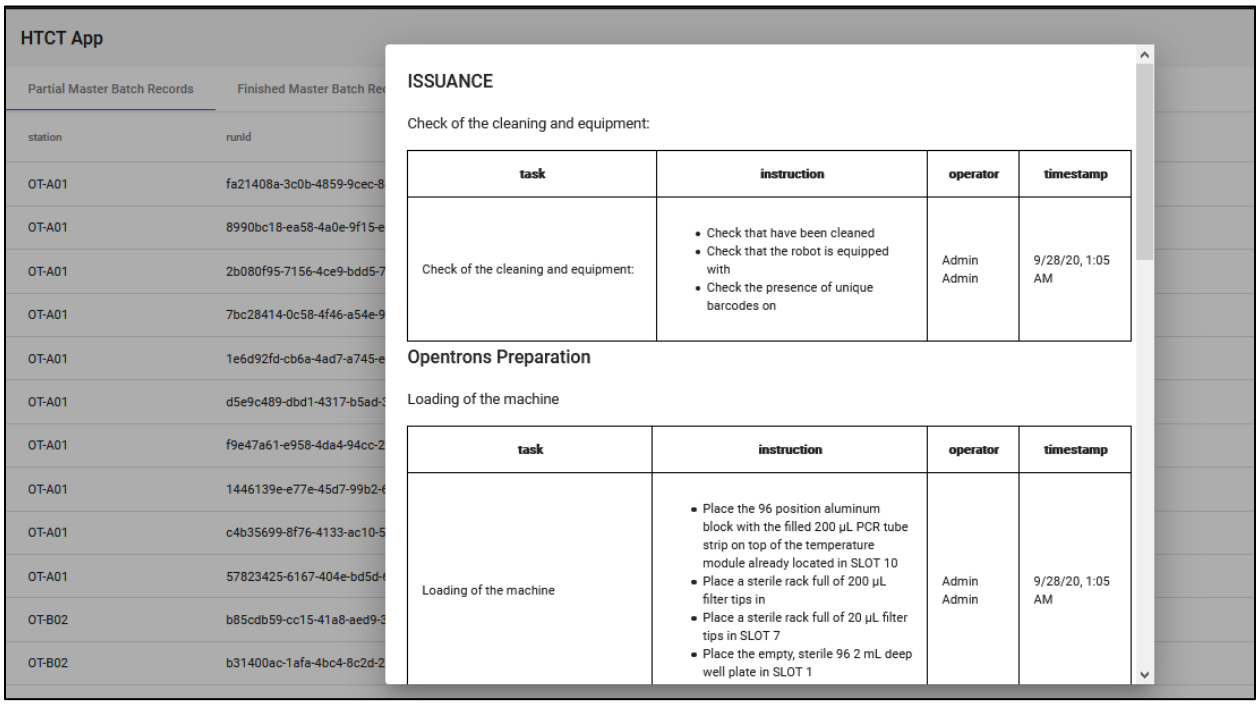
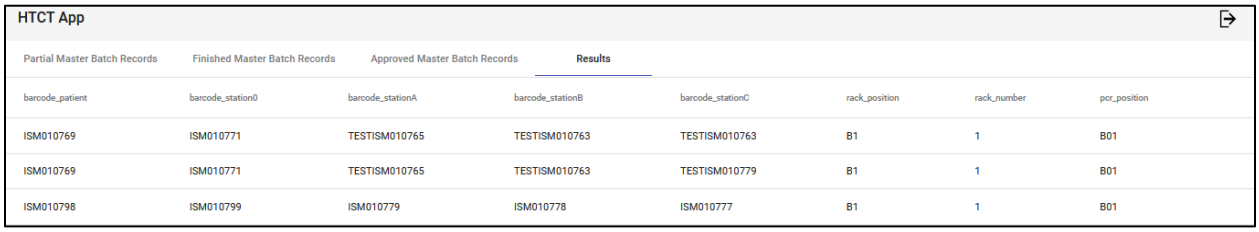


Figure 6: interface showing the digital record – a document summarizing all the steps that have been performed in the process (including time stamps and the operator's name).

Operators and lab managers can use the interface to track each patient sample within the system (Figure 7). Finally, the interface shows the individual test results for each patient (Figure 8). Individual patient results can be saved in a PDF file.

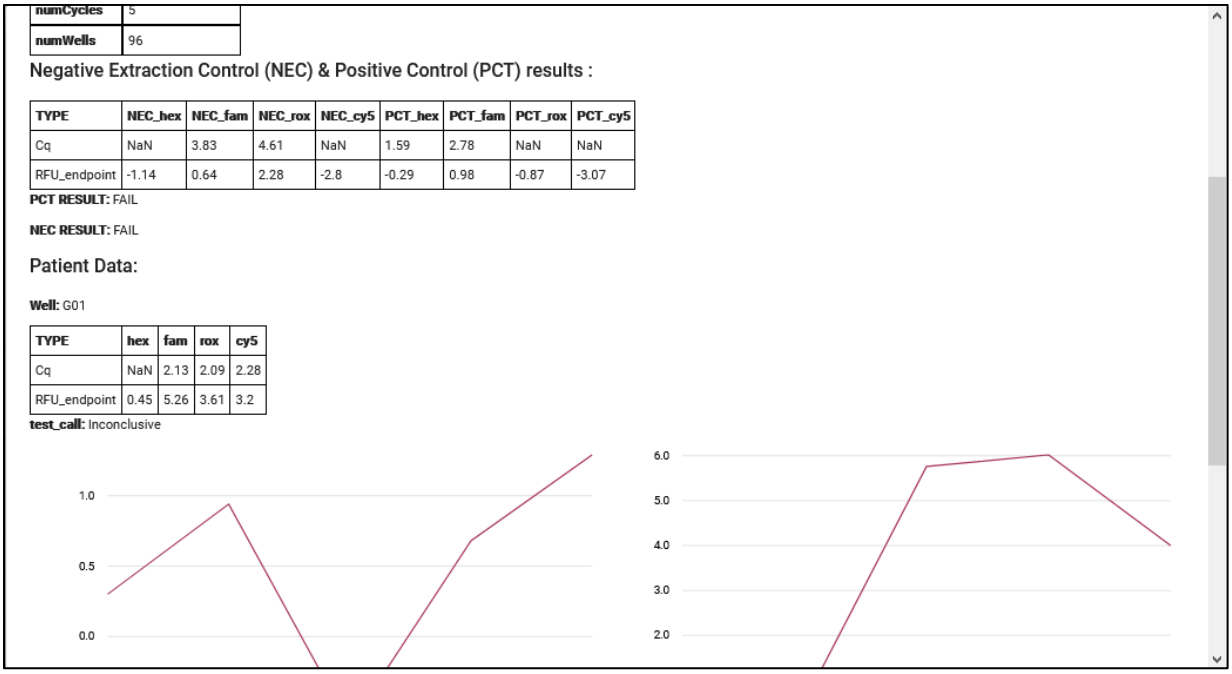


HTCT App

Partial Master Batch Records | Finished Master Batch Records | Approved Master Batch Records | **Results**

barcode_patient	barcode_station0	barcode_stationA	barcode_stationB	barcode_stationC	rack_position	rack_number	pcr_position
ISM010769	ISM010771	TESTISM010765	TESTISM010763	TESTISM010763	B1	1	B01
ISM010769	ISM010771	TESTISM010765	TESTISM010763	TESTISM010779	B1	1	B01
ISM010798	ISM010799	ISM010779	ISM010778	ISM010777	B1	1	B01

Figure 7: table showing the tracking information that allow operators and lab managers to trace the path of each patient sample through the system.



numCycles	5
numWells	96

Negative Extraction Control (NEC) & Positive Control (PCT) results :

TYPE	NEC_hex	NEC_fam	NEC_rox	NEC_cy5	PCT_hex	PCT_fam	PCT_rox	PCT_cy5
Cq	NaN	3.83	4.61	NaN	1.59	2.78	NaN	NaN
RFU_endpoint	-1.14	0.64	2.28	-2.8	-0.29	0.98	-0.87	-3.07

PCT RESULT: FAIL

NEC RESULT: FAIL

Patient Data:

Well: G01

TYPE	hex	fam	rox	cy5
Cq	NaN	2.13	2.09	2.28
RFU_endpoint	0.45	5.26	3.61	3.2

test_call: Inconclusive

Figure 8: when clicking on one patient, the interface shows the results of the PCR test. The curves and the data are sampled by the PCR machine. (Note: the results shown in this image are artificially generated as an example.)

3. Local interface

Note: the package of the Localwebserver can be installed directly from pypi with a: pip install covmatic-localwebserver (Repo: https://pypi.org/project/covmatic-localwebserver/).

The Local interface or LocalWebServer is a Python 3 program. Its function is to manage the execution of the protocols (i.e. sets of automation instructions) for the Opentrons liquid handling robots. The Local interface also manages the API of the PCR machine and sends process results and data to the Webserver.

This program allows us to work with a unique interface (the Web application), simplifying all the operations of the Covmatic system.

The program is a very simple backend made with a Python package called Flask. This program "reads" specific endpoints of the webservice and each endpoint is associated with a specific action that we need, e.g. executing a particular Opentrons' protocol or executing the API of the PCR machine.

To setup the local interface, start with getting ssh access to the Opentrons robots. After the installation of the robot, is necessary to generate and install the ssh key. This allows access to the robot and executes the protocols without using the Opentrons API.

The instructions to follow for installing the keys can be found at this address:
https://support.opentrons.com/en/articles/3203681-setting-up-ssh-access-to-your-ot-2

In order to simplify the operation of the system, our advice is to generate the ssh key once, as explained on the Opentrons guide, and install the same key on each robot. This will enable the use of the same key for all the robots.

LocalWebServer Program setup:

1. The LocalWebServer program can be downloaded from the GitHub at the following address: https://github.com/OpenSourceCovidTesting/localWebServer/tree/MVP
2. Place your ssh key files, i.e ot2_ssh_key.pub and ot2_ssh_key, inside the root folder of the LocalWebServer.
3. Inside the program "Task_runner.py" in the folder "services", modify the password of the ssh key. This password is declared in the variable called: "OT2_ROBOT_PASSWORD" (line 32). The password is the same that you have used for generating the ssh key.
4. Inside the program "Task_runner.py" in the folder "services", modify the path of the API of the PCR machine.
5. Set the environment variable called: "OT-2-IP" as the IP of your Opentrons robot connected to the laptop that you are using. This passage could be done on Windows in two ways:
    a. Open the Prompt of the Command and digit:
    setx OT-2-IP <IP-OF-THE-ROBOT>
    E.g. setx OT-2-IP 192.168.1.1
    b. Type in the Windows Search Bar: "Environment Variables". The menu shown in Figure 9 should appear.
    c. Click on the button that is encircled in Figure 9: "Environment Variables". Another window will open (Figure 10).

d. To create the environment variable, click on the red button (circled in Figure 10): “New”. This will create the environment variable (Figure 11).

e. Enter the IP as the value of the variable and the name: OT-2-IP.
Then press OK (Figure 11).

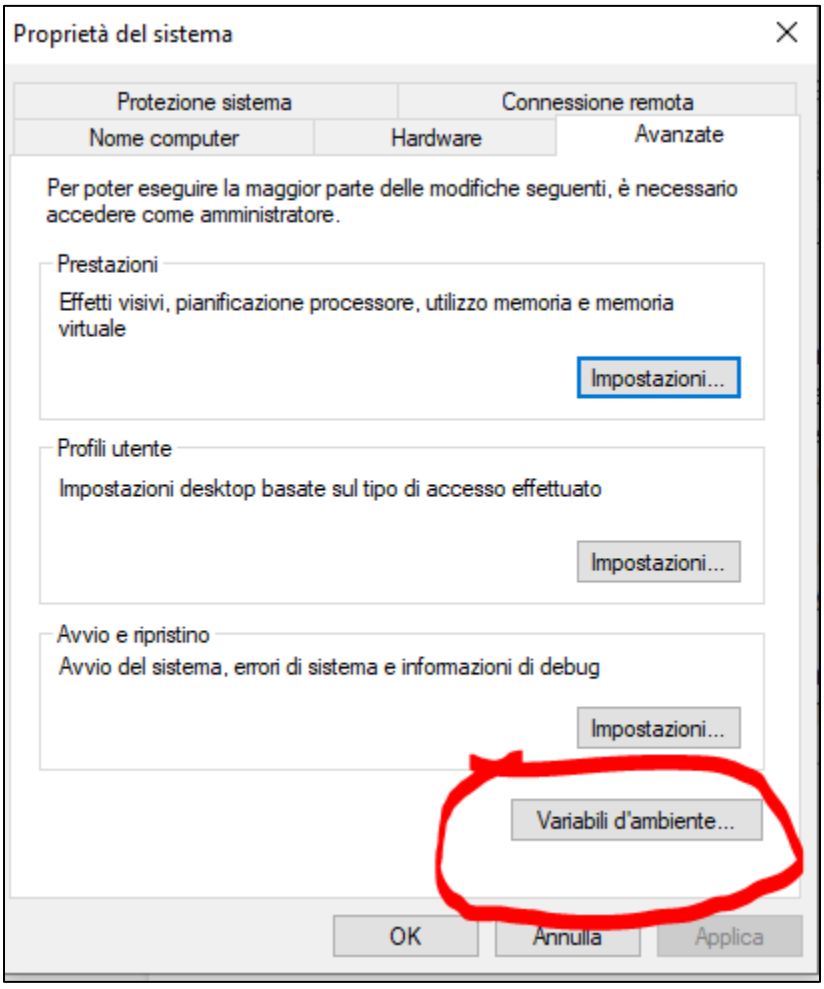


Figure 9

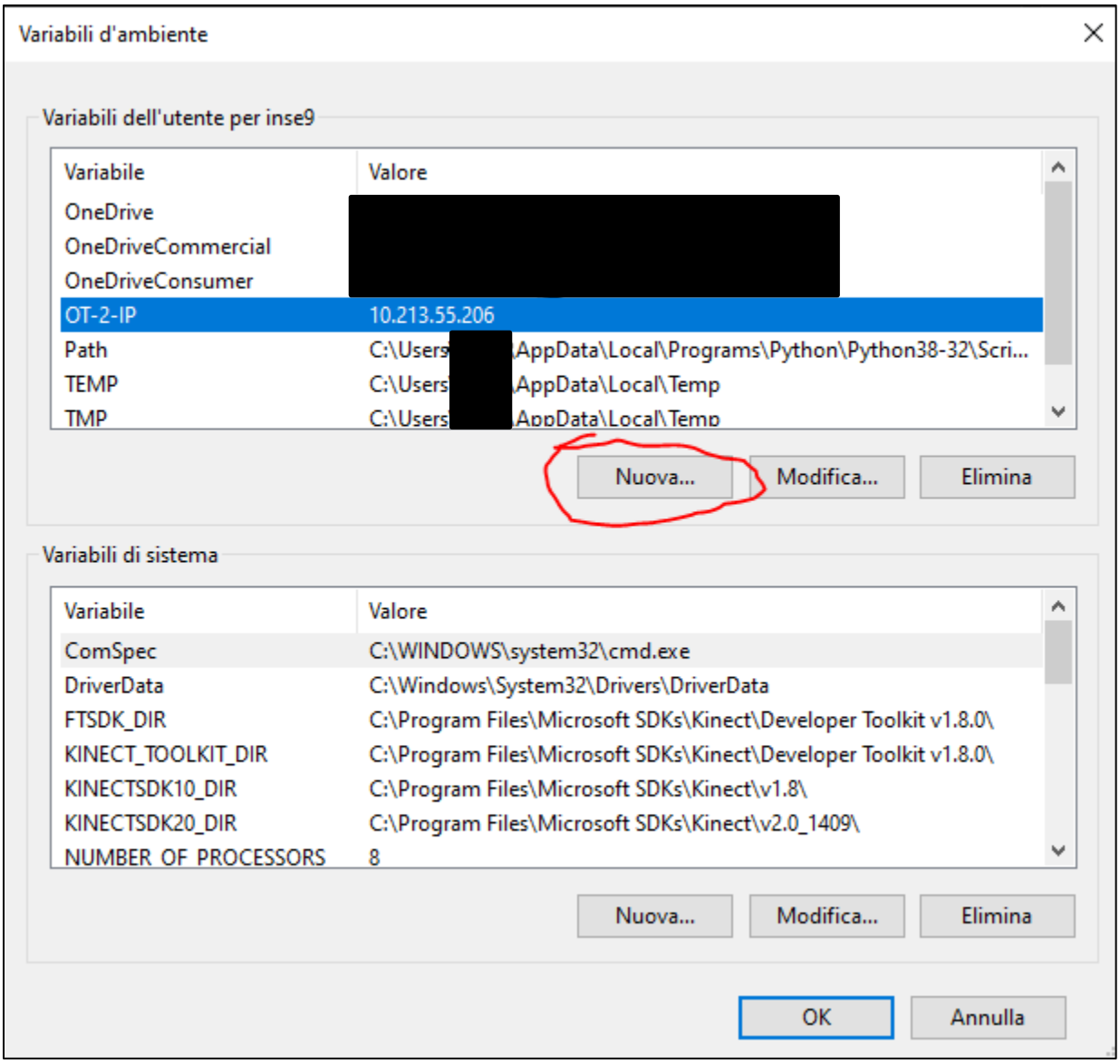


Figure 10

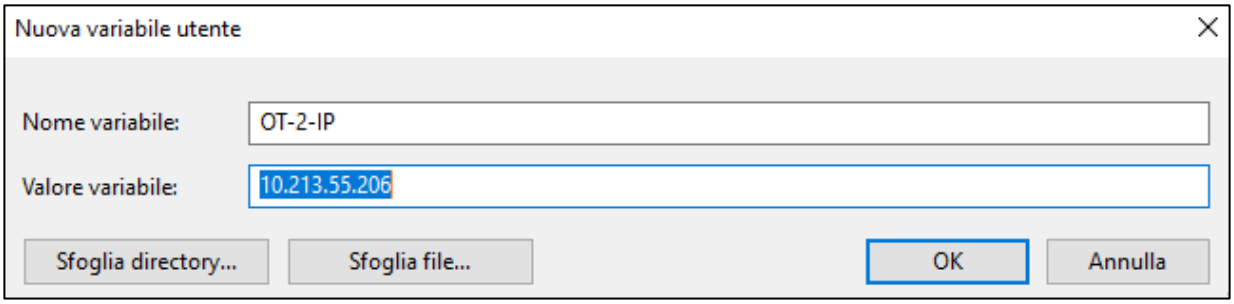


Figure 11

If using other OS, check how to change these variables in their documentation.

After you have followed these passages, you can execute the program using the "Prompt of Command" for Windows or the Terminal if you are using Linux or Mac OS, using the following command: "python app.py". This must be executed in the root folder of the program.

If you are using Windows, in the root folder there is the file: "LocalWebServer.bat". You can click on it to execute the program instead of using the terminal. You can also create a shortcut for it on your desktop, using drag and drop.

In the Readme of GitHub, you will find all the instructions and the details of the code, as well as troubleshooting recommendations.

Setting up the protocols inside the OT2 machines:

Before starting this process, it is necessary to copy inside the Opentrons OT2 liquid handling robot the protocols that are necessary for the process. The files that you need to copy depend on which Station you are working on. You can also decide to copy every protocol in each machine because each file has a different name and when you are launching the process with the Web Interface you will launch only the protocol dedicated to that specific station.

Copying of the files is quite easy and with the Jupyter interface.

1. Open the browser with a laptop connected to the robot or connected to the same network.
2. Type in the address bar: <IP-OF-THE-ROBOT>:48888. E.g.: 10.213.55.64:48888
3. You will see something like Figure 12.
   (Image courtesy: https://support.opentrons.com/en/articles/1795367-running-the-robot-using-jupyter-notebook)
4. Click on the "Upload" button on the top right.
5. Upload the protocol files.
6. Create a folder and name it "outputs" by clicking on the button "New" on the top right. The logs and the counter of the liquid handling tips that have been used will be saved in this folder.

Other information regarding Jupyter can be found on Opentrons' support pages, e.g.:
https://support.opentrons.com/en/articles/1795367-running-the-robot-using-jupyter-notebook

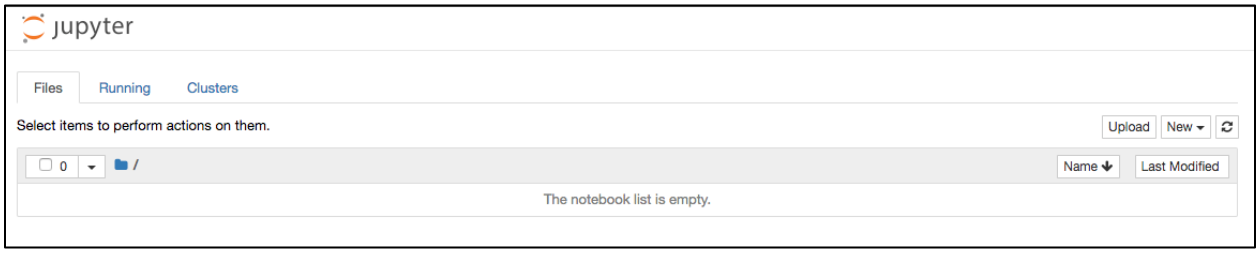


Figure 12

4. Protocols

The protocols are the instructions that run in the Opentrons OT-2 liquid handling robots.

The protocols for the Covmatic stations, and the instructions to use them, can be found in this repo: https://pypi.org/project/covmatic-stations/

5. PCR station API

The PCR station has several BioRad CFX96 C1000 and C1000 touch PCR machines, each machine connected to one unique workstation/computer. The PCR machines are controlled by a customized API that is included in the repository provided in the link below:
https://github.com/covmatic/PCR_station

First, you need to set up the Covmatic local webserver on your local machine/computer following the instructions provided in the "Local interface" section in this document. Then, to set up the PCR API on your local machine, follow the instructions in the document provided in the following link and refer to section *"2. How to Use?"*
https://github.com/covmatic/PCR_station/blob/main/PCR_Documentation/Covmatic-PCR_API%20documentation.pdf

You can use the PCR API instance both through the Covmatic webserver, or as a standalone application. Note that the PCR results will be saved on the Covmatic digital records ONLY if the API is launched through the Covmatic webserver following the steps provided in this document:
https://github.com/covmatic/PCR_station/blob/main/Station_Process_Flow/PCR_station_process_flow.pdf

A GOOD AND RECOMMENDED STARTING POINT IS TO REFER TO THE DOCUMENTATION
https://github.com/covmatic/PCR_station/blob/main/PCR_Documentation/Covmatic-PCR_API%20documentation.pdf

In brief, the PCR API does 3 things:

1. Allows the user to control the BioRad CFX96 PCR with C1000 Touch or C1000 thermal cycler
2. Interprets the results of PCR run and consolidates it into a single JSON file
3. Signals the Covmatic digital server that the json file is ready to be picked up

The Covmatic digital webserver then reconciles the contents of each well in the 96-well PCR plate to the patient from whom the sample was extracted. The code for this BioRad API is written in C# following the .NET framework. The API communicates with the BioRad PCR machine (via the BioRad application, CFX Manager/Maestro) by exchanging XML files (via USB serial connection) that follow a defined schema.

6. Containerized solution

To run a dockerized solution of the Web Application you can write a docker-compose.yml file and run it:

```
# docker-compose.yml
version: "3.8"
services:
  flaskwebserver:
    image: covmatic/webserver:latest
    volumes:
      - covid_data:/app/data

  ui:
    image: covmatic/userinterface:latest
    ports:
      - "6443:443"

volumes:
  covid_data:
```