Start Here: the EE 105 Lab Kit

Every EE 105 lab runs from one Jupyter notebook in this folder. The notebook uploads the course firmware to the Arduino Nano 33 BLE Sense on the demo board, runs the measurements, plots the data, holds the analysis, and at the end writes one results file that you upload on the course portal. No Arduino IDE, no port names, no copying numbers out of a Serial Monitor. The textbook's lab sections (one per lab, in Chapters 0 to 5) are the lab manual: read the section before the lab. The guide "Getting Started with the Labs" in Chapter 0 of the textbook shows this setup, a lab session and the upload step by step, with pictures. Every code cell of a notebook is one step of that section, with the same label (A1, B2, …, Save); the cell above it shows the step's tag (RUN, DO, EDIT, ANSWER), a short line that says what the step asks, the book's picture of the action where it has one, and, where the step draws a result, an example of what it should look like (marked "Expected result"). The notebook opens with the lab's key takeaways, the board setup and a map of its Parts; a CHECK cell marks each checkpoint; and the notebook ends with the Questions cells and a checklist.

Setup (once per laptop, 15 to 30 minutes)

1. Unzip `ee105-labs.zip` somewhere you will find again, for example in Documents. Keep the folder together: the notebooks import the `ee105` toolkit from the folder above them. Do not run anything from inside the zip file.
2. Run the setup file for your computer. It needs an internet connection and about 5 GB of disk space (have 6 GB free: it checks for that before it installs conda), and no administrator password.
   - **Windows 10 or 11** (x64, or ARM64 with Windows 11): double-click `setup-windows.bat`. If Windows says "Windows protected your PC", click **More info**, then **Run anyway**.
   - **macOS 12 or newer** (Apple Silicon or Intel): **right-click** (or Control-click) `setup-mac.command` and choose **Open**, then **Open** again in the dialog. (A plain double-click the first time only says the file "cannot be opened because it is from an unidentified developer". On macOS 15 and newer, if the dialog has no Open button, click **Done**, open **System Settings > Privacy & Security**, scroll down to the message about `setup-mac.command`, click **Open Anyway**, and confirm.) After the first time, a double-click is enough.
3. The setup file finds conda (Miniforge, Miniconda or Anaconda) or installs Miniforge in your user folder, creates the Python environment `ee105` from `environment.yml` (Python 3.11 with NumPy, pandas, Matplotlib, SciPy, SymPy, pySerial, JupyterLab and PyTorch), installs the Arduino tools and the Nano 33 BLE board package, and opens JupyterLab in your web browser. It writes every step to `setup-log.txt` in this folder; if it stops, the last lines of that file say why.

Every lab

Run the setup file again (a double-click; it now only starts JupyterLab, in a few seconds). Keep its window open while you work. Open the lab's folder in the JupyterLab file list on the left, then its notebook, and run the cells from the top with **Shift+Enter**.

The first code cell connects to the board, uploads the course firmware if needed, and checks the

module selector S10: if the reading does not fit the position the lab needs, the cell prints a `[!!]` warning; check the selector and carry on. The lab's section in the book and the table below give the board setup (selector, jumpers, parts). If a cell stops with an error, the `[!!]` line under the error says what to do (most often: run the cells above first, or fill the lines marked (pencil)).

The selector S10 is the small rotary switch at the bottom center of the board; its four positions are printed around it: **Module 1**, **Module 2**, **Module 3** and **Camera**. Jumpers J7 and J8 are the two small upright headers just left of the Arduino, printed **Camera** at the top end and **Module 1** at the bottom end; both caps always sit on the same end. J5 is the small header lying on its side in the biosensing block, printed **P1-A-P2**.

Lab	Folder	Selector S10	Jumpers	Plugged in
0.1 Arduino and the demo board	`lab0.1-arduino-board`	Module 1	J7, J8 on the Module 1 end	nothing
0.2 Python bootcamp	`lab0.2-python-bootcamp`	no board		
1.1 Asymmetric touch panel	`lab1.1-touch-panel`	Module 1	J7, J8 on the Module 1 end	nothing
1.2 Circuit analysis with Python	`lab1.2-circuit-analysis-python`	Module 1	J7, J8 on the Module 1 end	two resistors in the divider header (values in the notebook), multimeter on the Network 2 test points
2.1 Diode and transistor I-V	`lab2.1-diode-transistor`	Module 2	no change	nothing
3.1 Thermistor temperature sensor	`lab3.1-thermistor`	Module 1	J7, J8 on the Module 1 end	10 kOhm NTC thermistor (beta 3950) in J1-J2 (top), 10 kOhm resistor in J3-J4 (bottom)
3.2 Pulse oximeter and PPG	`lab3.2-pulse-oximeter`	Module 3	J5 on P1-A	nothing
4.1 Neural networks	`lab4.1-neural-networks`	no board		
4.2 Camera classifier	`lab4.2-camera-classifier`	Camera	J7, J8 on the Camera end (back to the Module 1 end after the lab)	camera on J6
5.1 LiFi	`lab5.1-lifi`	Module 3 on both boards	J5 on A-P2 on both boards (one-board test: J5 on P1-A); afterwards back to P1-A	two boards with their right edges facing, the sender's TX pointing at the receiver's RX

Submitting a lab

The last cell of every notebook writes `lab<N.k>_submission.json` (for example `lab1.1_submission.json`) in the lab folder. Log in to the course portal, **ee105.org/portal**, with your student ID and portal

PIN, open the lab, and upload that one file. It is graded automatically and the grade is recorded. A run on the simulated board (`SIMULATE = True`) is recorded as a practice run with no points. You can upload again after a fix; keep your notebook too.

Practice without a board

Set `SIMULATE = True` in a lab's first code cell. The notebook then talks to a simulated demo board with realistic but made-up data.

What is in the kit

Folder or file	Contents
`setup-windows.bat`, `setup-mac.command`	setup (first run) and start of JupyterLab (later runs)
`environment.yml`	the Python packages of the `ee105` environment
`labN.k-.../`	the lab notebook, plus any data files or extra notebooks it uses
`ee105/`	the toolkit the notebooks use (`ee105.connect()`, `%%arduino`, live plots, simulator)
`firmware/ee105_board/`	the one Arduino sketch used by every demo-board lab (uploaded for you)
`firmware/ee105_camera/`	the camera sketch for Lab 4.2 (uploaded for you)
`data/mnist/`	the MNIST handwritten digits for Labs 4.1 and 4.2
`board-check.ipynb`	tests every circuit of a demo board in about three minutes

Commands you can type

The course firmware also answers short commands from a notebook cell, `board.monitor("circuits")`, or from any serial terminal at 115200 baud: `help`, `module`, `circuits`, `divider`, `temperature`, `diodes`, `transistors`, `transfer`, `light`, `sensors`, `communications send <text>`, `communications receive`, and `stop`.

The toolkit in five lines

```
import sys; sys.path.insert(0, "..")
import ee105
board = ee105.connect()                      # finds the board, uploads the course firmware if ne
board.check_module(1)                        # warns if the selector S10 does not read Module 1
df = board.read(pin=1, n=1000, interval_ms=1)   # 1000 samples of A1 as a table (t_s, raw, volts
```

A notebook cell that starts with `%%arduino name` is an Arduino sketch: running it compiles and uploads it. `ee105.monitor(5)` and `ee105.serial_plot(10)` replace the Serial Monitor and Plotter.

If something goes wrong

- **The setup file stops.** Read the last lines of `setup-log.txt`. Without internet it stops at once; connect and run it again (finished steps are skipped). A full disk needs about 5 GB freed.

- **No board found.** Use a USB data cable (a charge-only cable powers the board but carries no data) and plug straight into the laptop.
- **Upload fails.** Double-tap the white RESET button on the Nano (the orange LED pulses), then run the cell again.
- **Port busy** (`Could not open` the port). Close the Arduino IDE or any other notebook that is connected to the board.
- **`[!!] Warning: A7 reads ...`** The selector S10 is not on the position the lab needs. Turn it; if it already points there, carry on (the check is only a warning).
- **`[!!] A value in this cell is still ...`** A line marked (pencil) still holds `....` Fill it in, run that cell, then this one.
- **`[!!] ... is made by an earlier step`** The kernel restarted or a cell was skipped: run the cells above in order (Run > Run All Above Selected Cell).