Usage

There are several IDEs (Integrated Development Environments) available for MicroPython programming, each with its own unique features and benefits. The most popular IDEs for MicroPython programming are:

- Thonny
- PyCharm
- Jupyter
- Mu is another multiplatform IDE for MicroPython programming that is designed for beginners. It has a built-in MicroPython REPL and includes a tool for flashing firmware onto your board. It runs on Windows, macOS, and Linux.
- Visual Studio Code is a popular and versatile IDE that supports multiple programming languages, including MicroPython. It provides many useful features such as syntax highlighting, code completion, and debugging. It runs on Windows, macOS, and Linux.

Thonny IDE

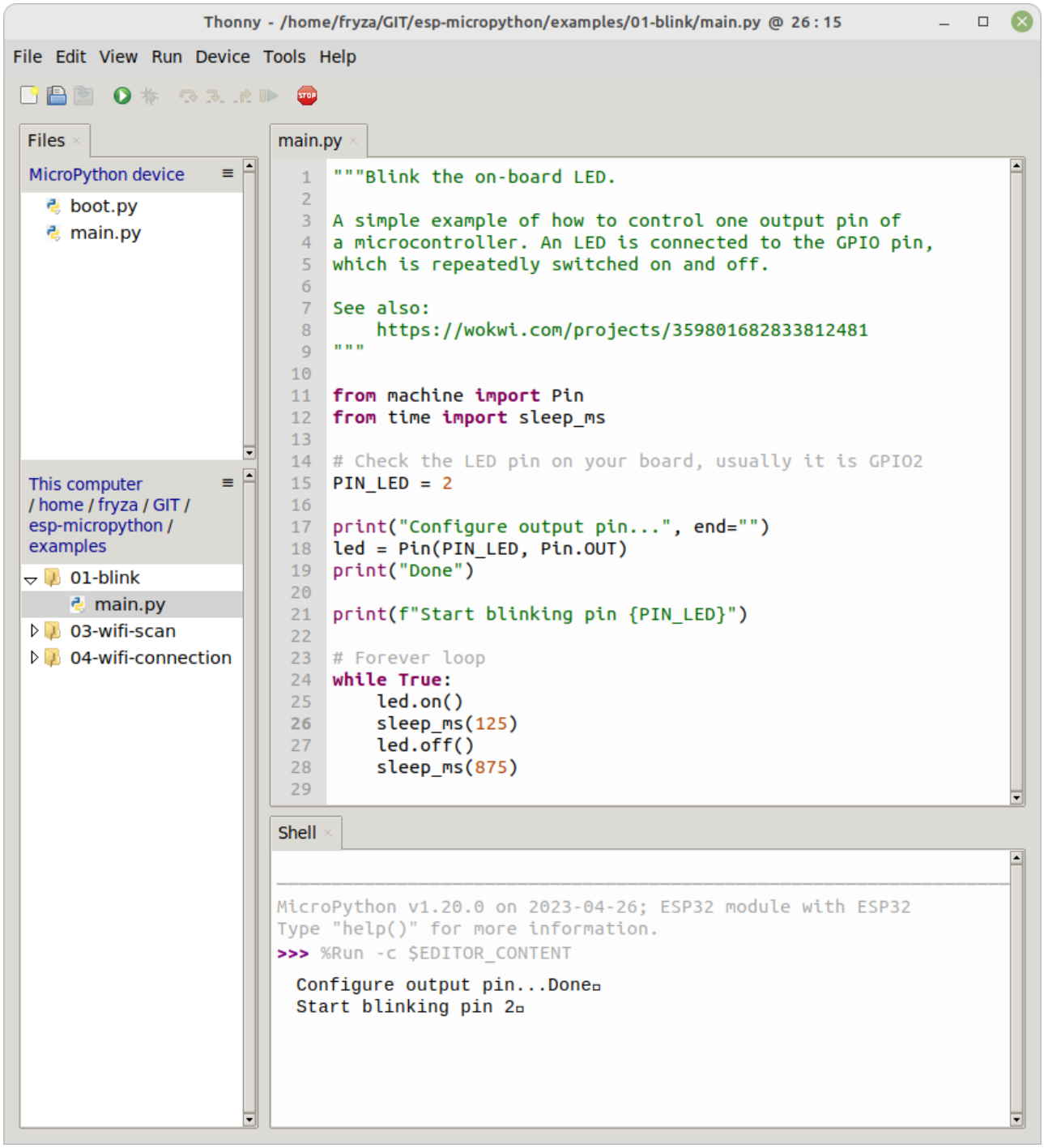
Thonny is a multiplatform IDE for MicroPython programming beginners. It has a simple and user-friendly interface and includes a MicroPython REPL (Read-Eval-Print Loop) that allows you to interact with the MicroPython interpreter and test your code in real-time.

1. Download and install Thonny IDE from webpage or directly in terminal:

   ```
   sudo apt install thonny
   ```

2. Run the Thonny and select on-board interpreter. Go to **Run > Select interpreter...** and select ESP32 or ESP8266. Test the board in **Shell**.

3. Copy/paste the example blink code and run the application by **Run > Run current script F5**.



PyCharm

PyCharm is a powerful IDE that provides advanced features such as code completion, debugging, and version control integration. PyCharm also includes a MicroPython debugger that allows you to step through your code and set breakpoints. PyCharm is a commercial product, but has a Community Edition free version. It runs on Windows, macOS, and Linux and it can be especially helpful for larger and more complex Python/MicroPython projects.

1. Download and install Community edition PyCharm.
2. Run the PyCharm and install MicroPython plugin for PyCharm. Go to **File > Settings... > Plugins > Marketplace**, search for `MicroPython` and install it. Restart the IDE.

3. Create a new project, name and locate it wherever you want. Connect your ESP32/ESP8266 board vie USB.

4. Go to **File > Settings... > Languages & Frameworks > MicroPython** and:

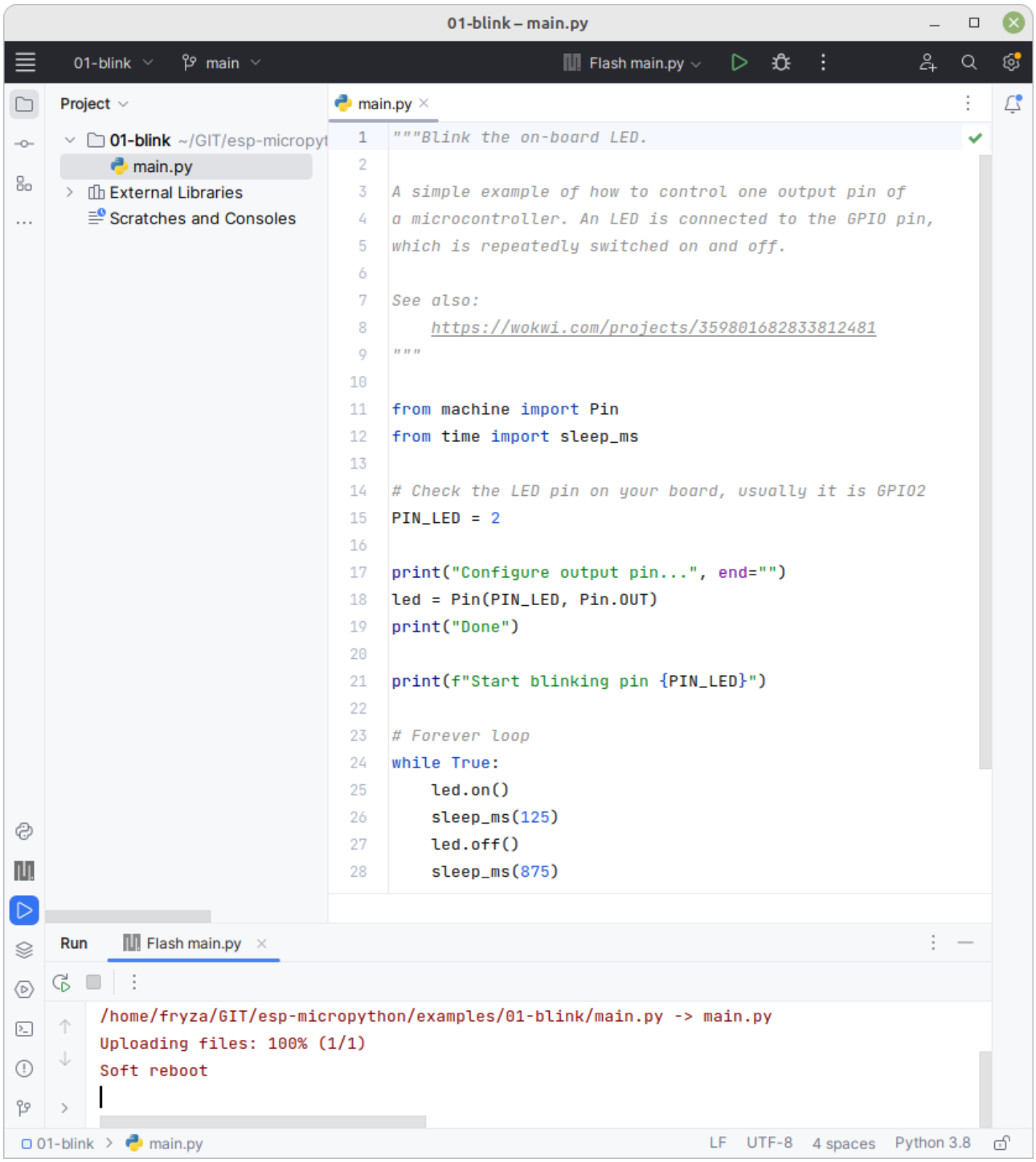
    - check `Enable MicroPython support`
    - select `ESP8266` device type (it works also for ESP32)
    - set `Device path` for your board, such as `/dev/ttyUSB0` in Linux
    - click on `OK` button

    Test REPL in **Tools > MicroPython > MicroPython REPL Alt+Shift+R**. Press on-board reset button if necesary.

    > **Note:** Sometimes, there is a useful function to clear all files store in device's memory. Select **Tools > MicroPython > Remove All Files from MicroPython Device**.

5. Add a file to the project. Select **File > New... > Python file** and name it `main.py`. The missing packages will be installed to work with the ESP32/8266. Copy/paste the example blink code to `main.py` file.

6. Upload a program. Right-click the `main.py` file in the project browser on the left side and select **Run 'Flash main.py'**.



> **Note:** Check MicroPython Tutorial for other simple examples and see description of machine module.

MicroPython in Jupyter

1. Open Terminal and install Jupyter notebook and/or lab:

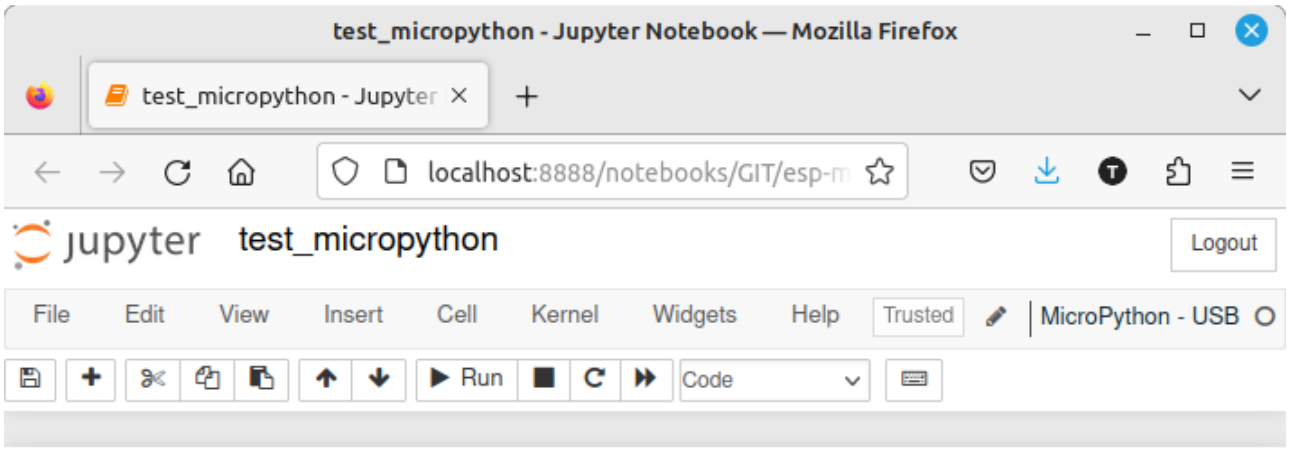
```
pip install notebook
pip install jupyterlab
```

2. Install MicroPython kernel:

```
pip install jupyter_micropython_kernel
python3 -m jupyter_micropython_kernel.install
```

3. Run Jupyter notebook and open example:

```
jupyter-notebook
```



References

1. Getting Started with the MicroPython in PyCharm for Raspberry Pi Pico
2. Jupyter MicroPython Kernel
3. MicroPython: Programming an ESP using Jupyter Notebook