Using UV To Manage a ESP8266 MicroPython Project

Background

> The ESP8266 is a low-cost Wi-Fi microcontroller, with built-in TCP/IP networking software, and microcontroller capability, produced by Espressif Systems in Shanghai, China.
>
> The chip was popularized in the English-speaking maker community in August 2014 via the ESP-01 module, made by a third-party manufacturer Ai-Thinker. This small module allows microcontrollers to connect to a Wi-Fi network and make simple TCP/IP connections using Hayes-style commands. However, at first, there was almost no English-language documentation on the chip and the commands it accepted. The very low price and the fact that there were very few external components on the module, which suggested that it could eventually be very inexpensive in volume, attracted many hackers to explore the module, the chip, and the software on it, as well as to translate the Chinese documentation.
>
> from Wikipedia

I'm using the ESP32 Basic Starter Kit, which contains an ESP8266 module with the following specifications:

Model	ESP8266MOD
Vendor	AI-THINKER
ISM	2.4 GHz
PA	+25 dBm
Wireless	802.11b/g/n

Project Setup

Adapted from here: https://docs.micropython.org/en/latest/esp8266/tutorial/intro.html

Make sure your ESP8266 is connected.

Initialize Project, Install ESP Tools

Create a project directory and cd into it:

```
mkdir esp_test

cd esp_test
```

Initialize the project:

```
uv init
```

The esptool utility is used to communicate with the ROM bootloader in Expressif chips. Add the esptool package:

```
uv add esptool
```

Show available commands:

```
uv tool run --from esptool esptool.py
```

Commands:

Command Name	Description
load_ram	Download an image to RAM and execute
dump_mem	Dump arbitrary memory to disk
read_mem	Read arbitrary memory location
write_mem	Read-modify-write to arbitrary memory location
write_flash	Write a binary blob to flash
run	Run application code in flash
image_info	Dump headers from a binary file (bootloader or application)
make_image	Create an application image from binary files
elf2image	Create an application image from ELF file
read_mac	Read MAC address from OTP ROM
chip_id	Read Chip ID from OTP ROM
flash_id	Read SPI flash manufacturer and device ID
read_flash_status	Read SPI flash status register
write_flash_status	Write SPI flash status register
read_flash	Read SPI flash content
verify_flash	Verify a binary blob against flash
erase_flash	Perform Chip Erase on SPI flash
erase_region	Erase a region of the flash
read_flash_sfdp	Read SPI flash SFDP (Serial Flash Discoverable Parameters)
merge_bin	Merge multiple raw binary files into a single file for later flashing
get_security_info	Get some security-related data
version	Print esptool version

Get chip information:

```
uv tool run --from esptool esptool.py chip_id
```

Output:

```
esptool.py v4.8.1
Found 33 serial ports
Serial port /dev/ttyUSB0
Connecting....
```

```
Detecting chip type... Unsupported detection protocol, switching and trying
again...
Connecting....
Detecting chip type... ESP8266
Chip is ESP8266EX
Features: WiFi
Crystal is 26MHz
MAC: //(redacted)//
Uploading stub...
Running stub...
Stub running...
Chip ID: //(redacted)//
Hard resetting via RTS pin...
```

Install MicroPython

> MicroPython is a software implementation of a programming language largely compatible with Python 3, written in C, that is optimized to run on a microcontroller.
>
> MicroPython consists of a Python compiler to bytecode and a runtime interpreter of that bytecode. The user is presented with an interactive prompt (the REPL) to execute supported commands immediately. Included are a selection of core Python libraries; MicroPython includes modules which give the programmer access to low-level hardware.
>
> from Wikipedia

Erase flash:

```
uv tool run --from esptool esptool.py erase_flash
```

Get the MicroPython firmware:

```
wget https://micropython.org/resources/firmware/ESP8266_GENERIC-20241129-
v1.24.1.bin
```

Flash the firmware to the ESP8266:

```
uv tool run --from esptool esptool.py --baud 460800 write_flash --
flash_size=detect 0 ESP8266_GENERIC-20241129-v1.24.1.bin
```

Test MicroPython

Install picocom (if needed) and connect to REPL:

```
sudo apt install picocom
```

```
picocom /dev/ttyUSB0 -b115200
```

Test the REPL:

```
>>> print('hello esp8266!')

hello esp8266!
```

Upload and Run a Python Script

Adapted from here: https://problemsolvingwithpython.com/12-MicroPython/12.06-Uploading-Code/

> MicroPython Tool (ampy) is a utility to interact with a CircuitPython or MicroPython board over a serial connection.
>
> Ampy is meant to be a simple command line tool to manipulate files and run code on a CircuitPython or MicroPython board over its serial connection. With ampy you can send files from your computer to the board's file system, download files from a board to your computer, and even send a Python script to a board to be executed.
>
> from PyPI

Install ampy:

```
uv add adafruit-ampy
```

Upload a script, then list the file contents of the microcontroller to confirm it's there:

```
uv tool run --from adafruit-ampy ampy -p /dev/ttyUSB0 put hello.py

uv tool run --from adafruit-ampy ampy -p /dev/ttyUSB0 ls
```

Output:

```
/boot.py
/hello.py
```

Run the uploaded script:

```
uv tool run --from adafruit-ampy ampy -p /dev/ttyUSB0 run hello.py
```

Output:

```
Hello from esp-test!
```

Links

Description	URL
ESP32 Technical Reference	https://www.espressif.com/sites/default/files/documentation/esp8266-technical_reference_en.pdf
ESP8266MOD Datasheet PDF – Wi-Fi Module – Espressif	https://www.datasheetcafe.com/esp8266mod-datasheet-wi-fi-module/
MicroPython Downloads	https://micropython.org/download/

python, embedded and iot