Programming hardware displays the easy way

Using MicroPython and LVGL

PyCon Italia 2025 – 30.05.2025

Marc-André Lemburg :: eGenix.com GmbH

Speaker Introduction

Marc-André Lemburg

- Studied Mathematics, Computer Science and Physics
- CEO eGenix.com GmbH
- Senior Solution Architect, Consulting CTO and Coach
- EuroPython Society Fellow and former Chair
- Python Software Foundation Fellow and former director
- Python Core Developer (Unicode, DB-API, platform module, ...)
- Co-founder Python Meeting Düsseldorf
- Based in Düsseldorf, Germany

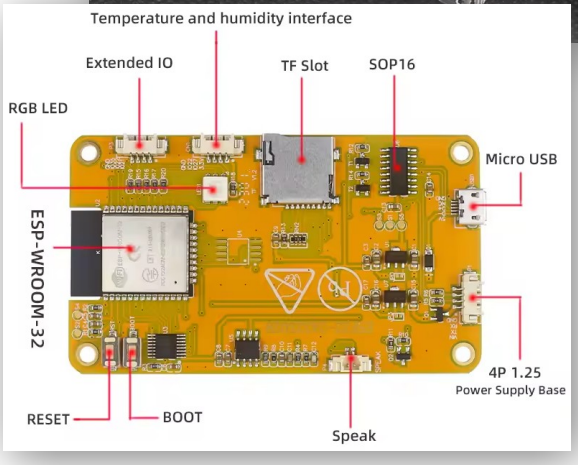
- More details and contact: https://malemburg.com/

Use case: Create a talk timer for conferences

- Use fixed or adhoc schedules (e.g. for lightning talks)
- Display remaining time to the speaker
- Alert speaker near end of talk
- Acquire exact time
  from the Internet
- Cheap and reliable device
- Should be able to work
  on batteries

Hardware: Cheap Yellow Display (CYD)

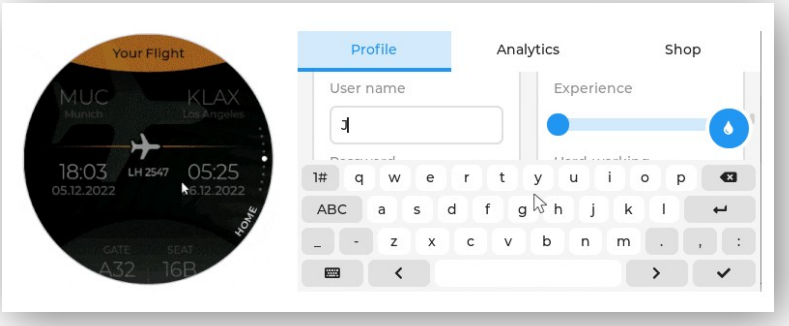
- ESP32 micro controller
  - Supports MicroPython (and CircuitPython)
  - 2.4GHz Wifi, Bluetooth, Dual Core, 240MHz
- 2.8″ TFT Display + Touchscreen
  - Connected via SPI to the ESP32
- All on one PCB, optionally with case
- Cheap: EUR 8-23
- Easy to order from many different sources



Software: Python + Graphics

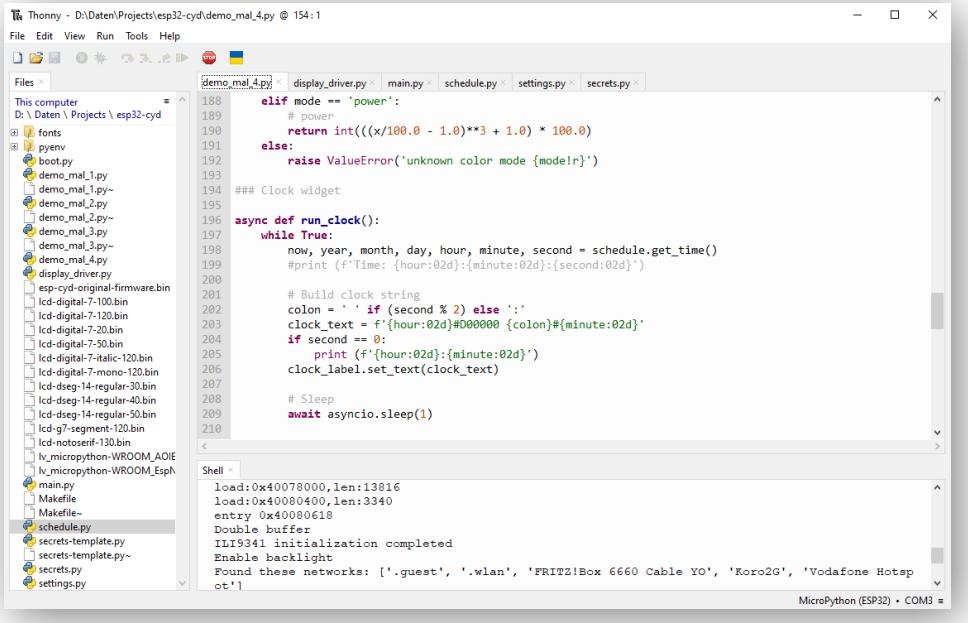
- MicroPython
  - Project by Damien George
  - Runs Python 3 on micro-controllers
  - Comes with interface support for e.g. SPI, Wifi, Bluetooth, etc.
  - Easy to flash to many different devices, e.g. the ESP32

- LVGL
  - Open source graphics library for micro controllers
  - MIT license
  - Professional looks and graphics
  - Comes with many powerful widgets
  - MicroPython port available



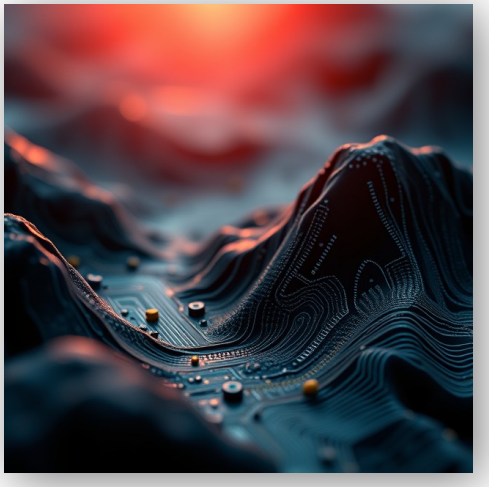
Suitable IDEs

- Thonny IDE
  - Not perfect, but best option
  - Device file browser,
    terminal console, code completion
- mu Editor
  - Nicer looks than Thonny
  - Lacks some tooling to provide
    a good dev cycle
- Visual Studio Code
  - Not well suited for the task
  - Setup too complex and error prone



Documentation ?

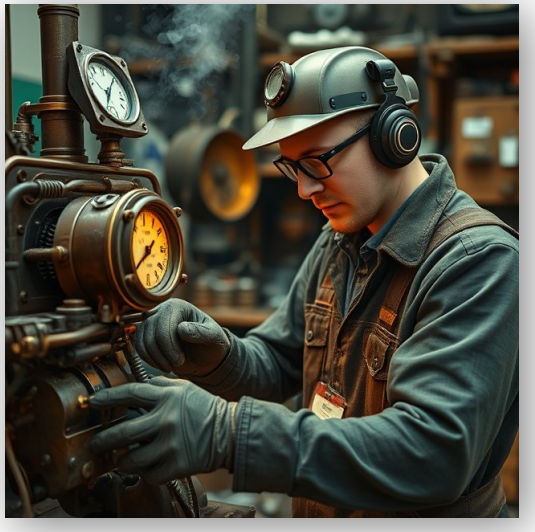
- MicroPython docs are excellent
- LVGL docs are excellent, but only for their C API
- MicroPython+LVGL docs don't really exist
  - Python interfaces don't always follow the LVGL C API
  - Some examples in the LVGL docs
- But: Lots of examples available on the net
- AI chats to the rescue...



AI generated :-)

Getting things working on the CYD...

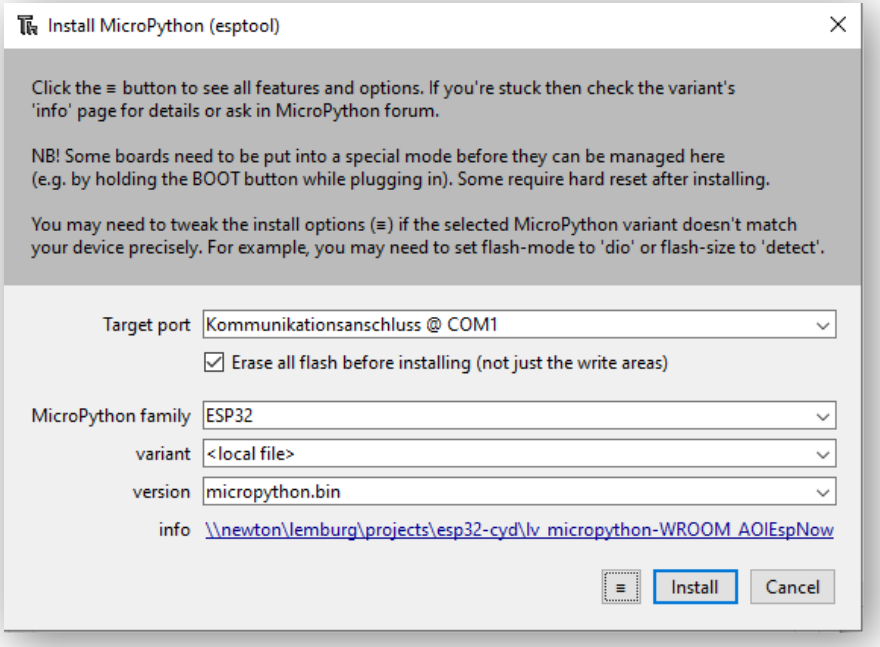
- MicroPython+LVGL: moderately easy
  - Search for pre-built firmware called “LV MicroPython”
  - Or build from source (a lot harder)
- Display: easy
  - The LVGL port comes with support
- Touchscreen: harder
  - But: others have already sorted this out
  - Search for “display_driver.py for CYD”
- Battery: easy
  - via Micro USB connector



AI generated :-)

Setup of the ESP32 micro controller

- Flash MicroPython+LVGL firmware on CYD
  - Use Thonny or esptool.py for this
  - Only needs to be done once
  - Be sure to flash the combined version of MicroPython + LVGL called *lv_micropython*
  - Prebuilt versions available from Stefan Scholz

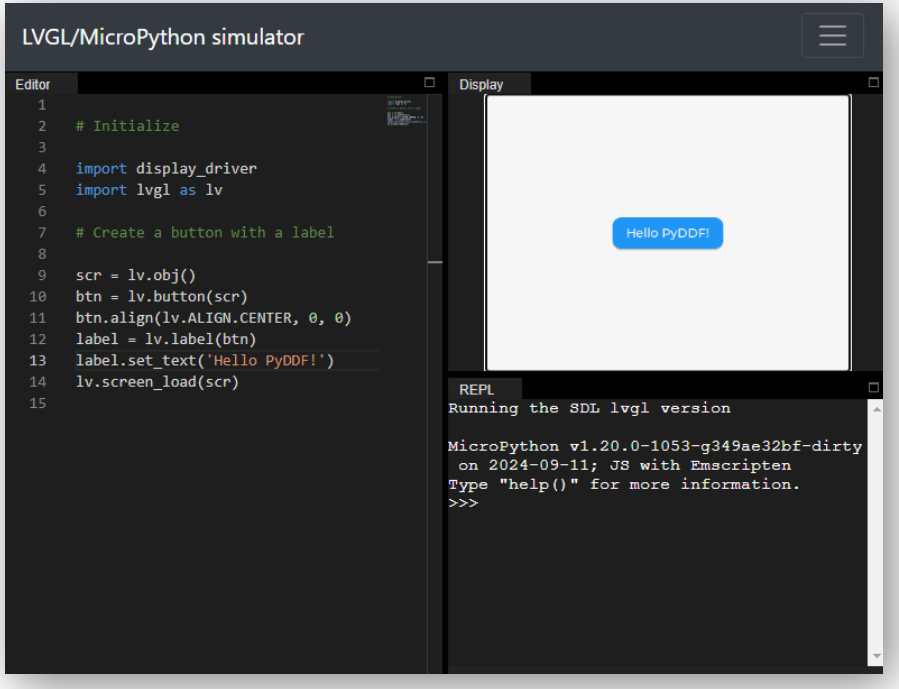


Development Workflow

- Create a main.py with your program
  - Add other modules as necessary
- Upload modules to the device via USB cable
  - Use Thonny for this
- Reset device to have main.py run
  - Can be done via console: *machine.reset()*
- Test and watch the console
  - Use Thonny
  - Or a terminal app connected to a USB serial adapter

Helpers

- MicroPython + LVGL Simulator
  - https://sim.lvgl.io/
  - Works in the Browser
  - Can be used to play around with the layout and controls
  - Does not support asyncio
- Font Converter
  - https://lvgl.io/tools/fontconverter
- Image Converter
  - https://lvgl.io/tools/imageconverter



Finding help

- Read the notes about the MicroPython bindings
  - https://docs.lvgl.io/master/details/integration/bindings/micropython.html
  - Includes many subtle details, which are hard to find otherwise
- Ask for help on the forum
  - https://forum.lvgl.io/c/micropython/8
- Ask your friendly AI chat agent
  - Perplexity works well for this

AI generated :-)

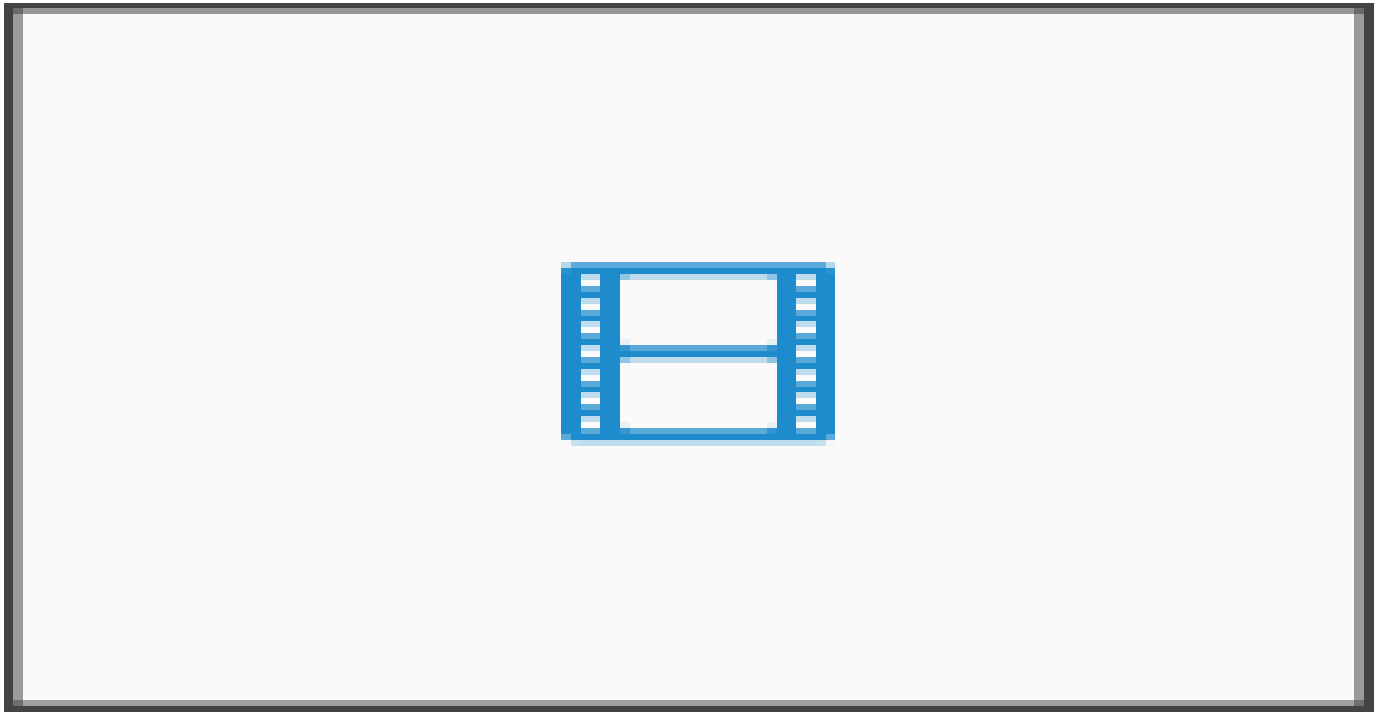
Some caveats

- Managing secrets is hard on micro controllers
  - Try to at least put secrets into separate modules, which are not added to the repo
  - Still: Anyone can extract the code from the controllers via USB cable
- Connecting to wifi sometimes takes longer than expected
  - 10 seconds or more are not unusual
- You often get weird memory or SPI errors while debugging
  - Simply reset the device to work around this

All that said:
This is great fun !!!

AI generated :-)

Let the fun begin...



Structuring apps for the CYD

- Keep UI elements defined as globals in a module
  - This makes sure they don't get garbage collected by accident
- Define a clear init process for the app
  - Esp. important when using async code
- Use many smaller modules to speed up development iterations
  - Uploads take time

AI generated :-)

Example UI code (excerpts only)

```
# Configure screen
screen = lv.scr_act()
screen.clean()
screen.set_style_bg_color(lv.color_white(), 0)

# Init label
init_label = lv.label(screen)
init_label.set_pos(10, 10)
init_label.set_style_text_font(
    lv.font_montserrat_16, 0)
init_label.set_text(
    f'{settings.APP_NAME}:\nInitializing')

# Progress bar
bar = lv.bar(screen)
bar.set_pos(280, 5)
bar.set_size(30, 230)
bar.set_range(0, 100)
bar.set_value(0, 0)

# ...
```

```
# Create a round button
gear_btn = lv.btn(lv.scr_act())
gear_btn.set_size(30, 30)
gear_btn.set_pos(240, 195)
gear_btn.add_flag(lv.obj.FLAG.CLICKABLE)
gear_btn.set_style_radius(30, 0)

# Add a gear icon to the button
gear_icon = lv.label(gear_btn)
gear_icon.set_style_text_font(
    lv.font_montserrat_16, 0)
gear_icon.set_text(lv.SYMBOL.SETTINGS)
gear_icon.center()

# Have the schedule_menu appear when the
# gear button gets clicked
def gear_btn_event_handler(e):
    code = e.get_code()
    if code == lv.EVENT.CLICKED:
        schedule_menu.clear_flag(
            lv.obj.FLAG.HIDDEN)
gear_btn.add_event_cb(
    gear_btn_event_handler, lv.EVENT.ALL, None)
```

Some additional hints to get you going

- Make sure to register file system drives
  - LVGL file loading doesn't work without them

```
import lvgl as lv
import fs_driver

fs_driver.fs_register(
    lv.fs_drv_t(), 'C')
```

- LVGL handlers will keep working even when the app is interrupted
  - They are implemented as C callbacks

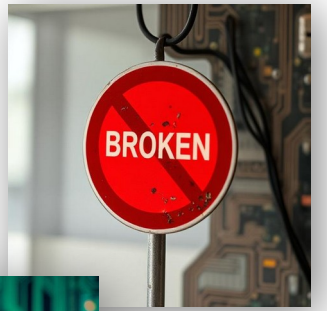
- Testing code works best by hard resetting the device
  - Helps avoid strange state issues caused by soft resets
  - Can be done in the terminal console:

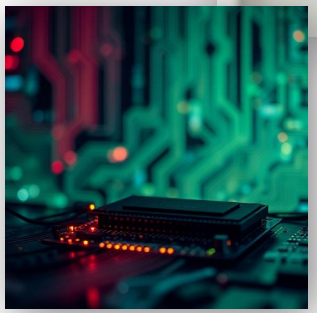
```
import machine; machine.reset()
```

Advanced Topics: Building from source

- Four parts to this
  - ESP IDF SDK for the ESP32
  - MicroPython
  - LVGL C library v8 and v9
  - LVGL wrapper for MicroPython

- Wrappers for LVGL:
  - lv_micropython
    - Integrates LVGL and MicroPython firmware
    - *Currently broken for ESP32 (as of May 2025)*
  - lvgl_micropython
    - New approach, still under heavy development
    - Much easier to get working
    - Requires rewriting display and touchscreen driver logic

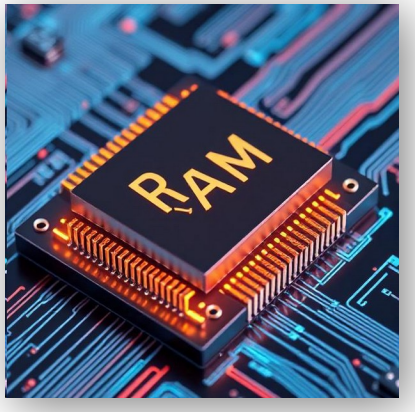




AI generated :-)

Advanced Topics: Careful RAM planning

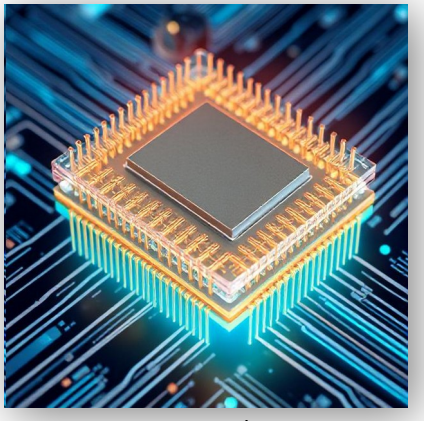
- Once running, MicroPython will have around 100kB RAM available for the apps on the CYD
- After loading display driver (with low mem settings) and WLAN driver, only 65kB RAM left
  - WLAN driver doesn't load with high mem display settings
- Loading images into RAM is not a good idea
  - Better have LVGL read these directly from the flash drive
  - It will then load these in batches



AI generated :-)

Advanced Topics: Have WLAN and display play nicely

- Display driver with low mem settings
  - use factor=8 and disable double buffering to have both work at the same time



AI generated :-)

- Or use a trick:
  - Run WLAN in a separate app to load initial data
  - Save data on flash drive
  - Soft reset the CYD using machine.reset() and load the actual app
  - Load init data from flash drive

- For the device clock, you can leave out the flash drive part
  - The clock keeps time across soft resets

Main takeaway: Never stop learning and trying out new things

Thank you for your attention !

Time for discussion

Contact

eGenix.com Software, Skills and Services GmbH
Marc-André Lemburg
Pastor-Löh-Str. 48
D-40764 Langenfeld
Germany

eMail: mal@egenix.com
Phone: +49 211 9304112
Fax: +49 211 3005250
Web: https://www.egenix.com/

LinkedIn:

References

- Several photos taken from Pixabay and Unsplash
- Some images were generated using DeepAI text2img
- Some screenshots taken from the mentioned websites
- All other graphics and photos are (c) eGenix.com or used with permission
- Details are available on request
- Logos are trademarks of their respective trademark holders