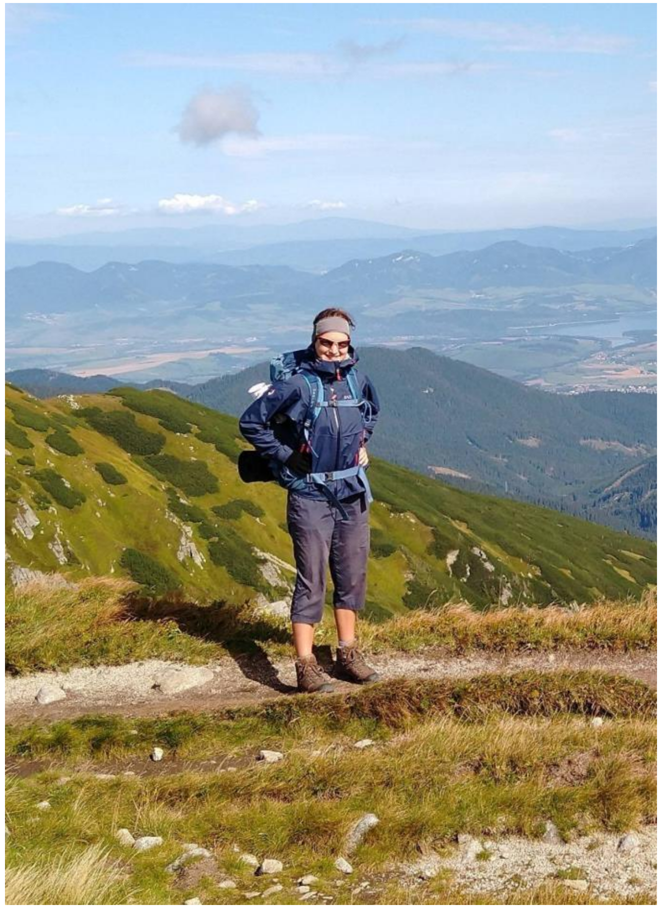
Zephyr™ Project
Developer Summit
June 8-10, 2021 • @ZephyrIoT

Micropython binding to LVGL in Zephyr OS

ZUZANA MIKLÁNKOVÁ

whoami

- student at FI MUNI (Brno, Czech Republic)
- junior C++ developer
- writing master thesis in cooperation with the NXP Semiconductors
  - thesis topic is Zephyr + Micropython + LVGL
- integration steps and measurements mentioned in this presentation are sourced from this thesis



Motivation

- GUI capable environment for embedded devices with a fast development cycle

- features combined from multiple projects:
  - HW support - Zephyr OS
  - interactive prompt - Micropython
  - GUI components - LVGL

Description of used projects

- Zephyr OS
  - modular RTOS for constrained devices
- Micropython (MPY)
  - implementation of Python3 for constrained devices
- LVGL (Light and Versatile Graphics Library)
  - open-source GUI library
- LV bindings
  - generates Micropython bindings to LVGL

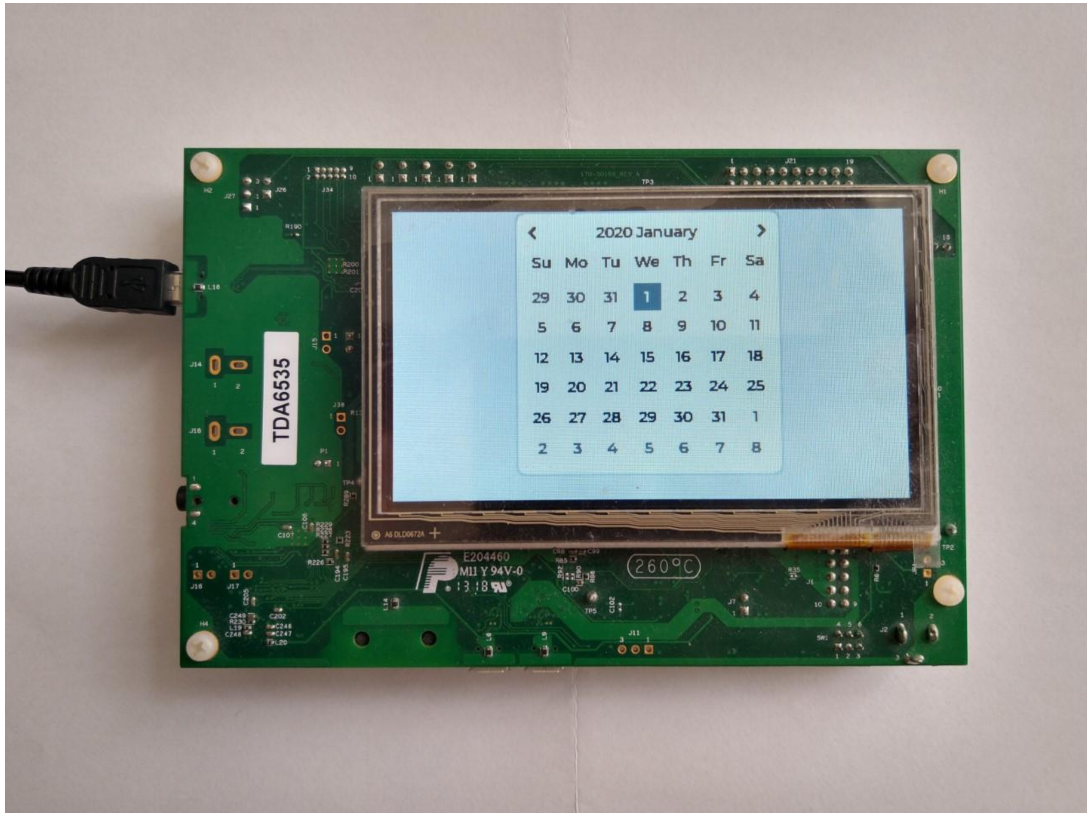
Zephyr™ Project
Developer Summit

Integration

Used SW & HW

- Zephyr OS
  - v2.5
- Micropython
  - v1.14
- LVGL
  - v7.6.1
- LV bindings

- used board
  - NXP i.MX RT1050 EVK (Arm® Cortex®-M7 core)
  - but the proposed solution should work on any board supported by the Zephyr OS (with display and input devices)



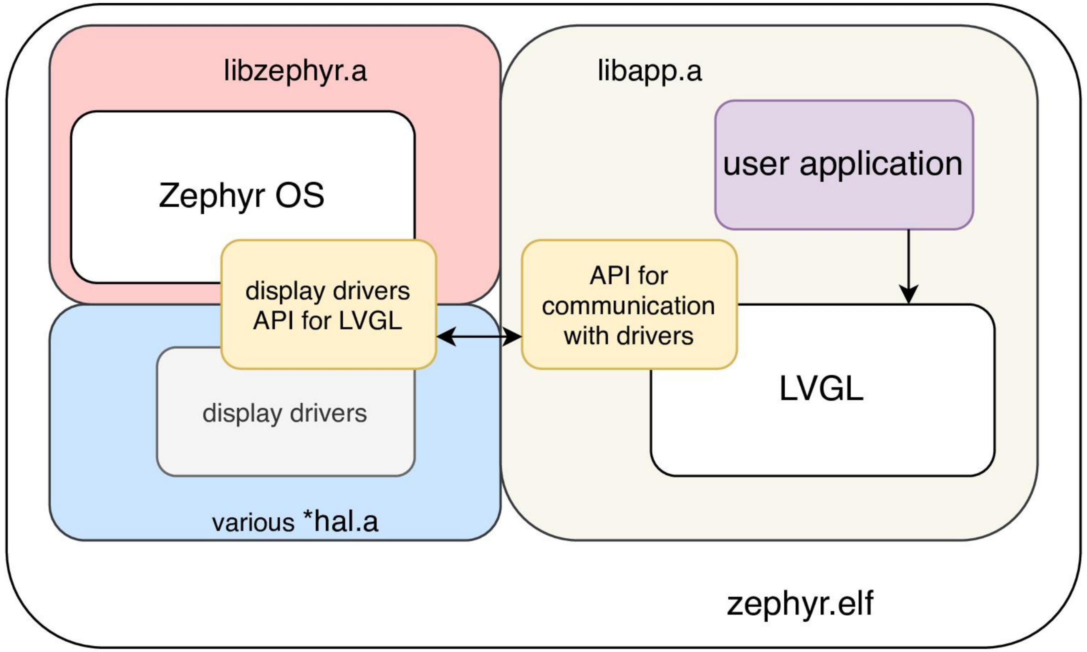
LVGL Integration (with any project)

Zephyr™ Project
Developer Summit

1. adjust LVGL definitions in the `lv_conf.h`:
   - e.g., `LV_HOR_RES_MAX, LV_VER_RES_MAX, LV_COLOR_DEPTH`
2. call `lv_init()` for initialization of the LVGL
3. display driver registration for LVGL
4. input driver registration for LVGL
5. call `lv_task_handler()` periodically every `n` milliseconds for LVGL to manage its internal tasks, such as read input task

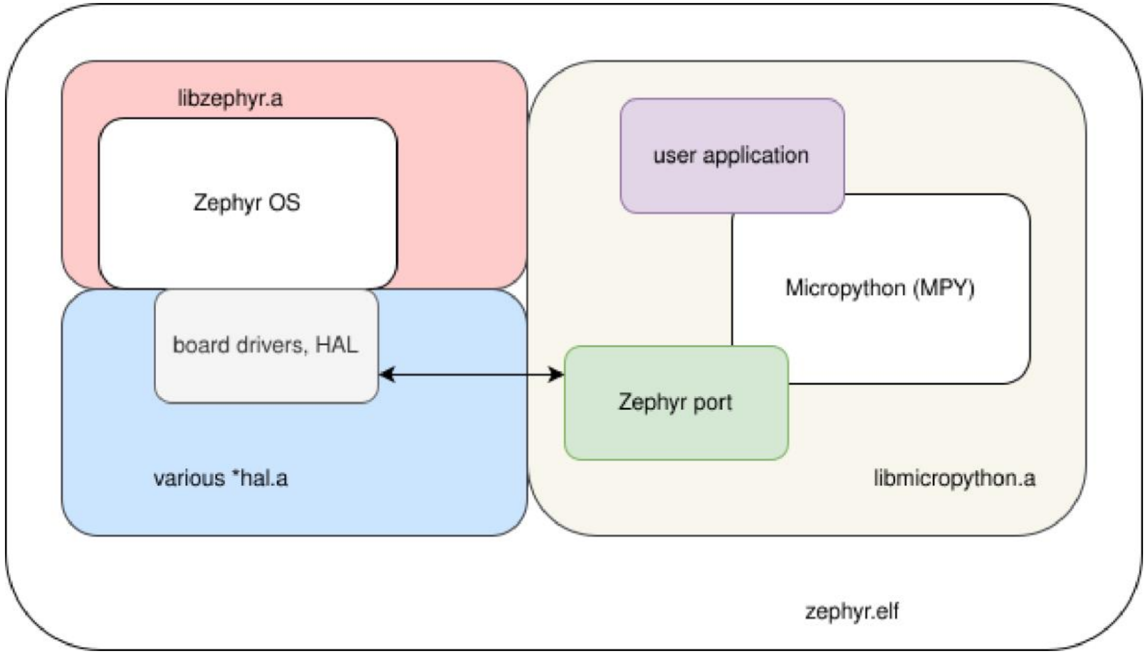
Zephyr OS with LVGL Integration

- Zephyr OS supports LVGL
  - drivers registration and initializations available in `$ZEPHYR_BASE/lib/gui/lvgl/` directory
  - LVGL source files included
- LVGL is initialized automatically during boot
  - when LVGL is enabled in Kconfig
  - implemented with the Zephyr's `SYS_INIT` macro
- Zephyr OS provides HW support
  - display driver
  - input driver
    - uses the Zephyr OS's kernel queues
    - queue defined with the `K_MSGQ_DEFINE` macro



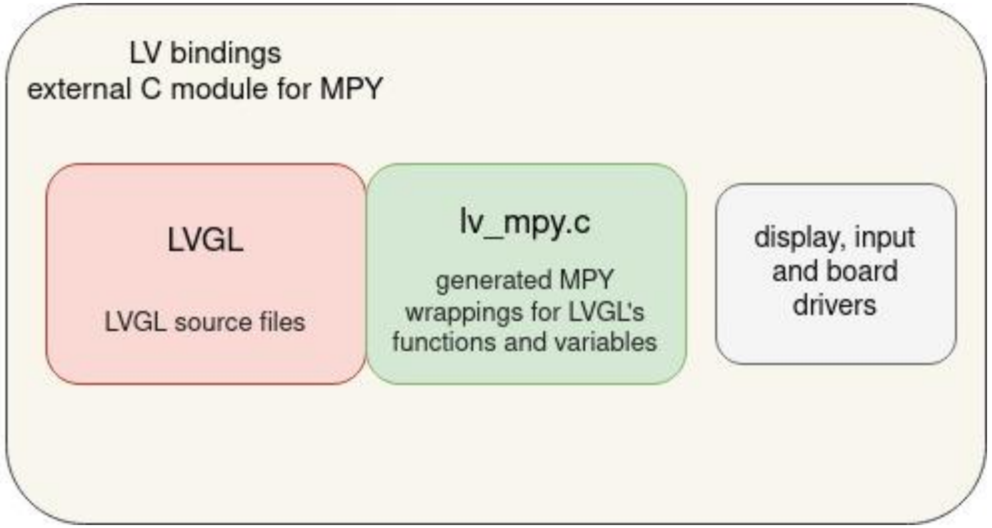
Micropython with Zephyr OS Integration

- the Micropython project contains the Zephyr port
- Micropython is built as a Zephyr application
- using Zephyr's kernel and HW support
- Micropython v1.14
  - Zephyr port is built with Makefile
  - for i.MXRT1050 make BOARD=mimxrt1050_evk
- Micropython v1.15
  - Zephyr port is built with CMake
  - more Zephyr-like - could be built with the West tool
  - west build -b mimxrt1050_evk micropython/ports/zephyr/
- Zephyr port is in a development phase, REPL is working



LV bindings with Micropython Integration

- automatically creates Micropython binding for the LVGL with Python script
- generated bindings are stored in file `lv_mpy.c`
  - integrated to the Micropython as an external C module
- based on `lv_conf.h`
- integrates own set of hardware drivers
  - mostly display and input
  - with proper LVGL drivers' registration
- contains own LVGL source files
- lv_micropython
  - forked Micropython repository with integrated bindings



Zephyr OS + MPY + LVGL integration strategy

Zephyr™ Project
Developer Summit

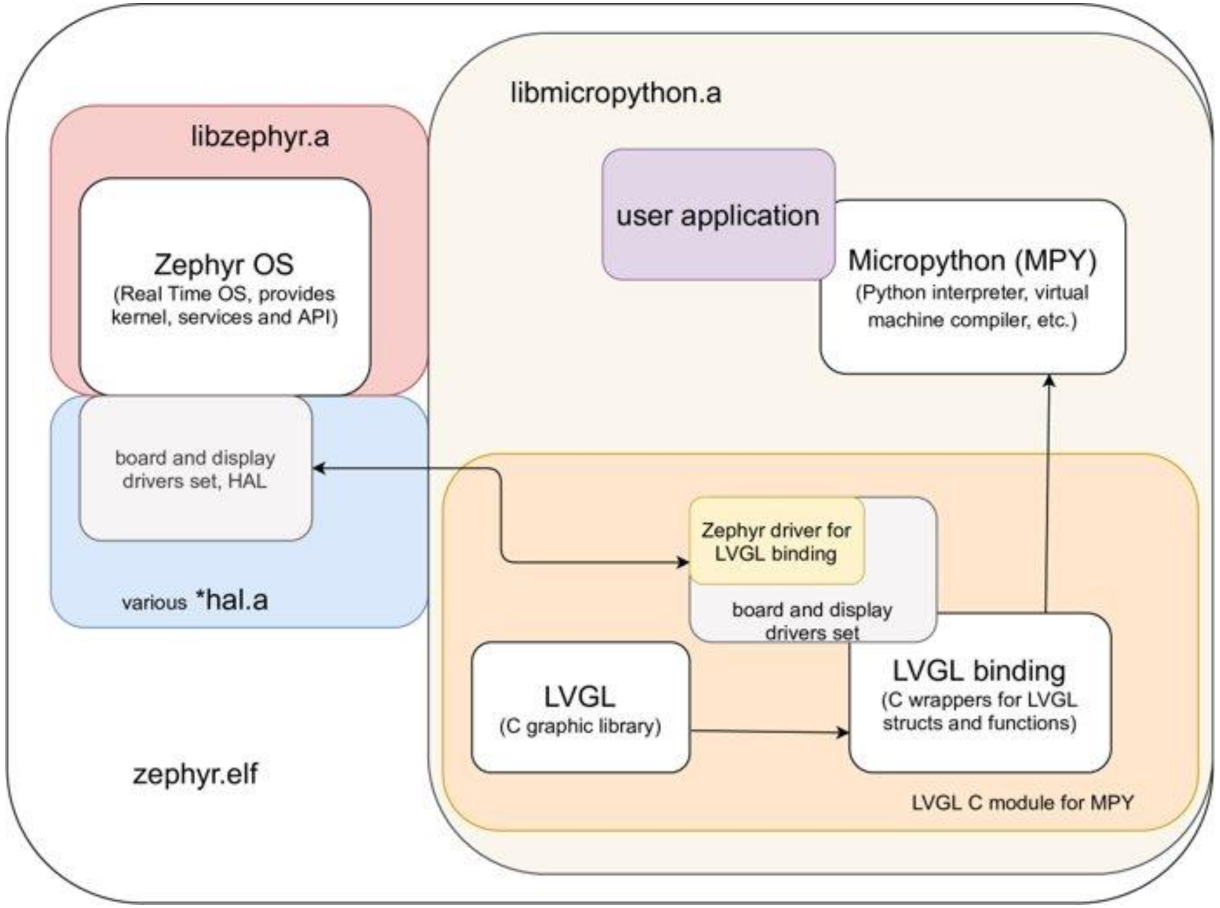
1. use LVGL sources from lv_bindings project
   - because LVGL is part of the Micropython application as an external C module
2. disable LVGL in Zephyr (with Kconfig options)
3. no code adjustments in Zephyr
4. add Zephyr driver in lv_bindings project
5. adapt Micropython build system
6. upstream the result if possible

Integration details

Zephyr™ Project
Developer Summit

- core of the driver registration and initialization for LVGL taken from the Zephyr codebase
  - **Zephyr "driver"** in lv_bindings project was created
- LVGL sources used from lv_bindings
  - checkouted on the version which is supported by the Zephyr OS
    - because of the driver abstraction layer
- Makefile & CMakeLists.txt of the Zephyr target in the Micropython
  - added include paths for needed header files
  - linked Zephyr kernel directly to the Micropython app – because of kernel queues
  - integrate binding generation, compile and link resulting files
  - compile and link the Zephyr driver from lv_bindings
- changes upstreamed
  - in lv_bindings_micropython
  - in lv_micropython

Integration details



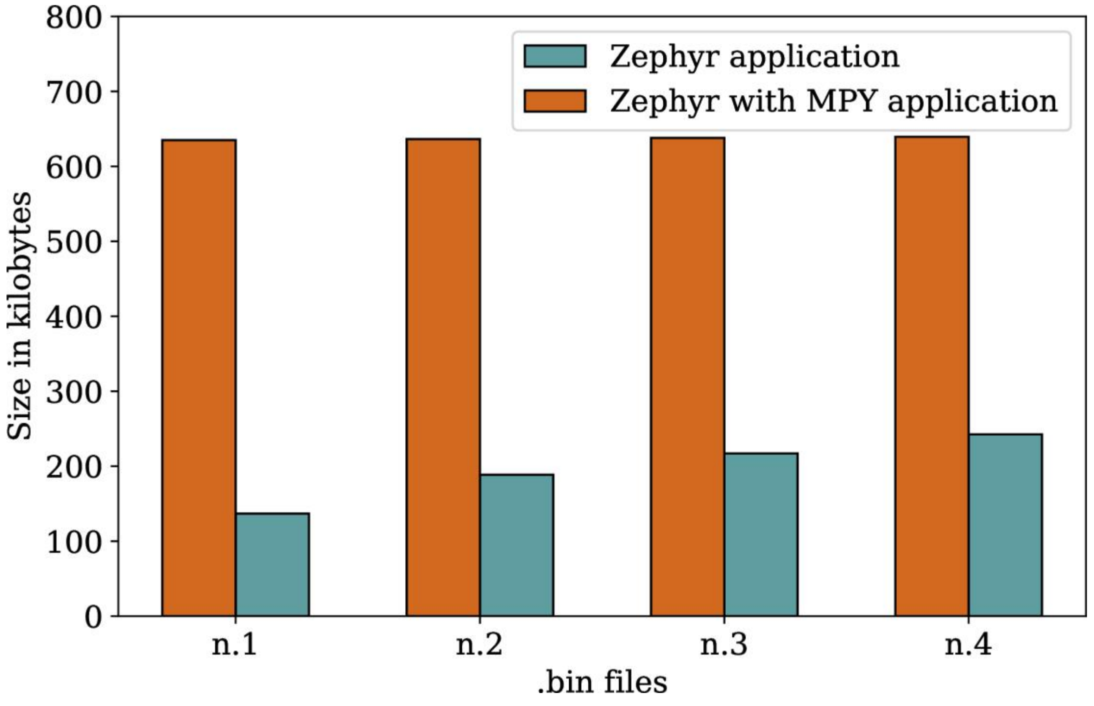
Profiling

Profiling

- applications with the same functionality but written in different languages have been compared
  - **Zephyr Application**
    - LVGL directly on the Zephyr OS
    - Applications are written in C language
  - **Zephyr with MPY Application**
    - LVGL as an external C module for the Micropython above the Zephyr OS
    - Applications are written in Micropython
- `lv_conf.h` set equally for both, except for memory management primitives
- Kconfig for both type of binaries also set (almost) the same
  - exceptions such as logging – Micropython v1.14 does not use Zephyr's logging system
- binaries have been compiled with GCC
  - size optimization enabled (Zephyr default)

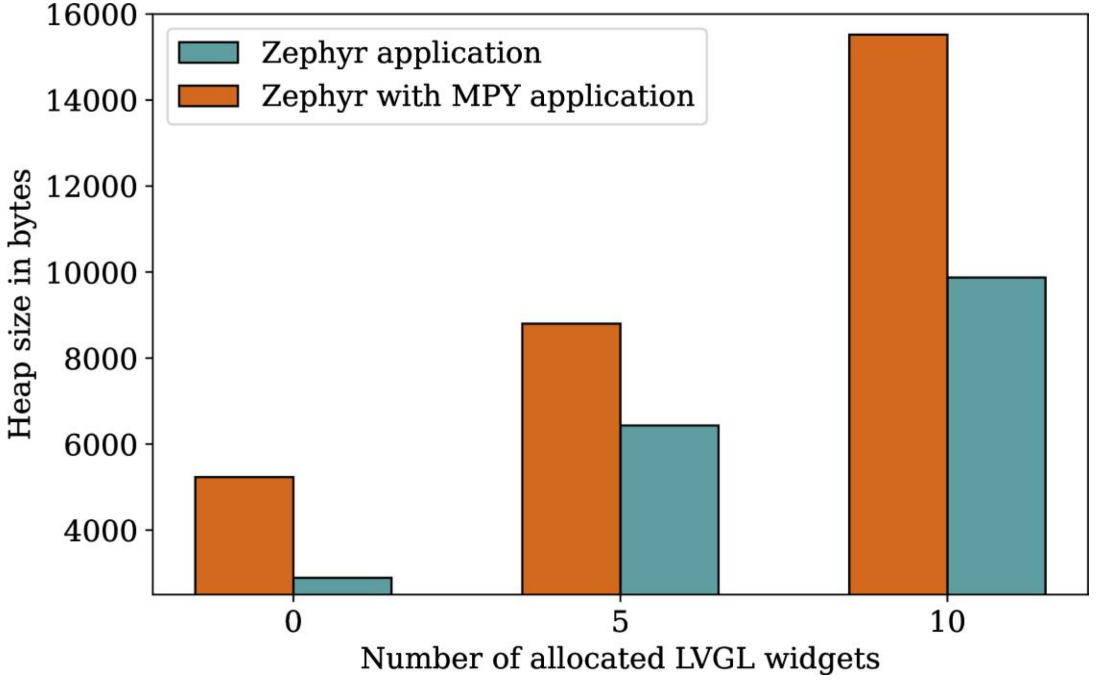
Profiling – binary sizes

- four demo applications with the following content:
    1. LVGL + drivers init
    2. 1.+ approx. third of LVGL widgets initialized
    3. 1.+ approx. two-thirds of LVGL widgets initialized
    4. 1+ all LVGL widgets initialized
- Zephyr with Micropython application is significantly bigger, but its size does not grow with added widgets
    - number of used LVGL widgets is not known during compile time because Micropython is interpreted language



Profiling – heap size

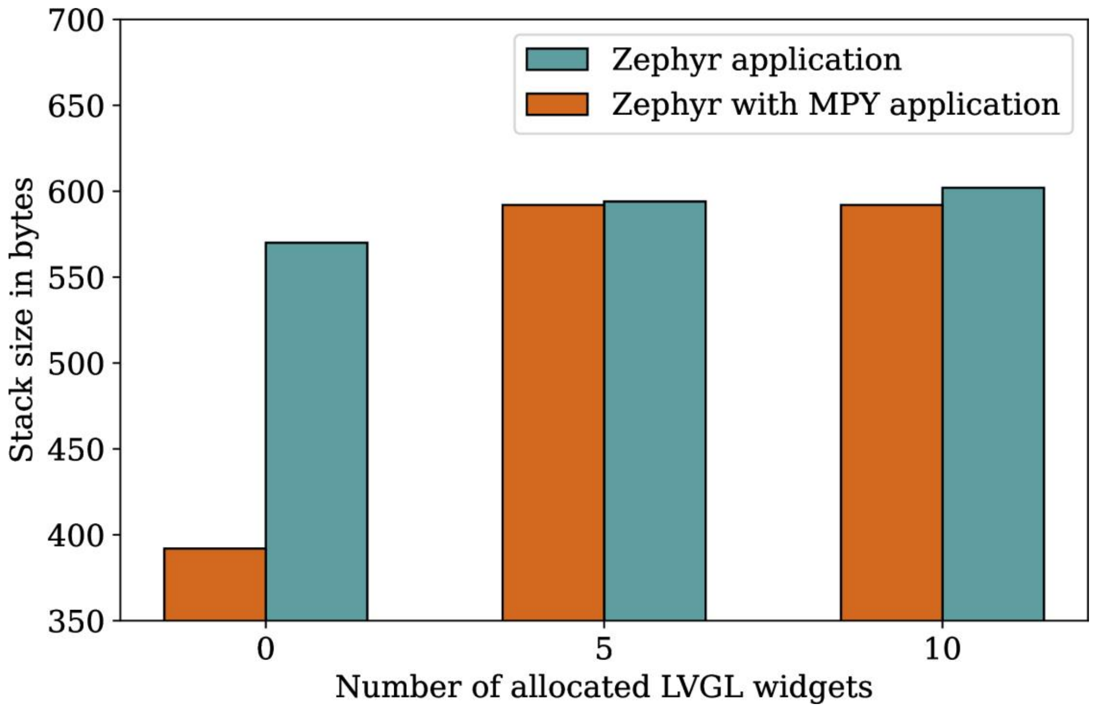
- LVGL stores its widgets on the heap
- two applications compared, heap size measured in the following states:
  1. after LVGL + drivers init
  2. after allocating 5 LVGL widgets
  3. after allocating 10 LVGL widgets



Profiling – stack size

- two application compared stack size measured in the following states:
  1. in the `main` function
     - contain LVGL + drivers init + calling other functions
  2. in function, where 5 widgets are allocated
  3. in function, where 10 widgets are allocated

- pointers are stored in the stack
  - all pointers located in an array



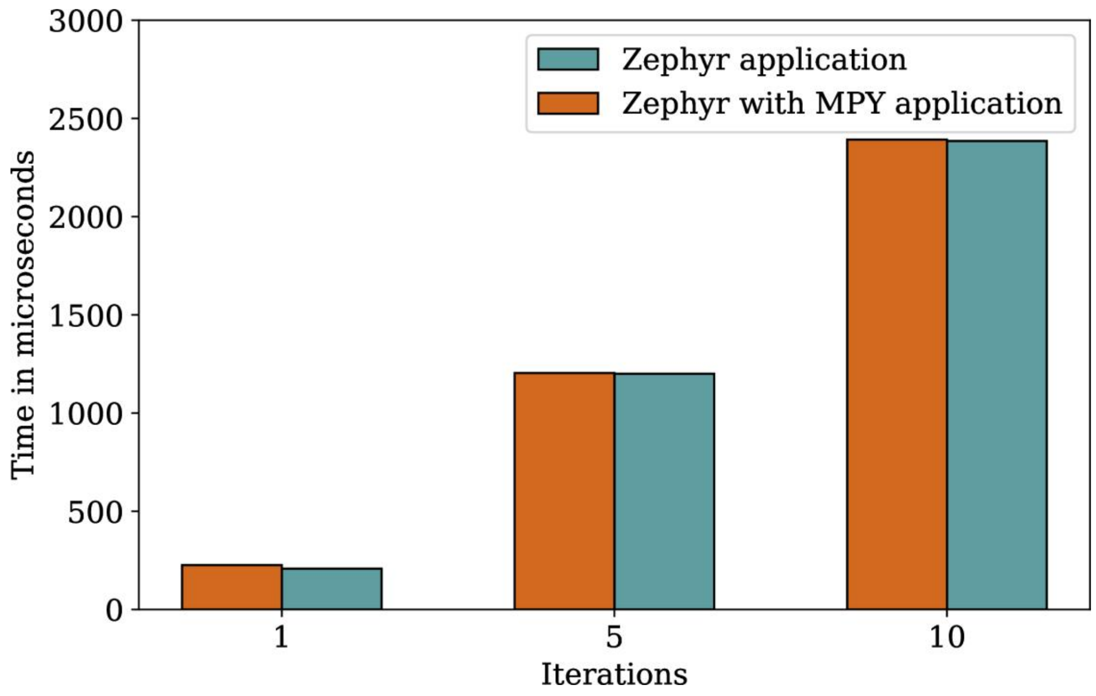
Profiling – input delay

- the first time mark is taken in input driver
- the second time mark is taken in LVGL callback function (for the button)
- **the result is biased with the measurement tool**
  - can be used only for comparison
  - time marking – changing the state of GPIO, the state is then analyzed with an external signal analyzer

Zephyr application average	Zephyr with MPY application average
21445 µs	22855 µs

Profiling – execution time

- demo applications contain initialization of two fully filled LVGL screens and switching between them
- time of screen redrawing is measured
  - switching from the first screen to the second and back
  1. in 1 iteration
  2. in 5 iterations
  3. in 10 iterations



Zephyr™ Project
Developer Summit

Conclusion

Conclusion

Zephyr™ Project
Developer Summit

- working solution was integrated
- changes were upstreamed
- profiling results:
  - the most relevant difference is in binary size
  - no other significant differences in the performance when the Micropython layer is / is not present

- **raw C is better for small binary aiming application**
- **Micropython is eligible for fast prototyping**

References

- https://github.com/zephyrproject-rtos/zephyr
- https://github.com/micropython/micropython
- https://github.com/lvgl/lv_micropython
- https://github.com/lvgl/lv_binding_micropython
- https://github.com/lvgl/lvgl
- https://is.muni.cz/th/pmpvp/

Zephyr™ Project
Developer Summit

Thank you for your attention!

Zephyr™ Project
Developer Summit
June 8-10, 2021 • @ZephyrIoT