ELCE 2025



Feasibility and Architecture of a Dual-Board Embedded System: Yocto & Zephyr Integration

Andrea Ricchi & Dario Binacchi

www.amarulasolutions.com

Slides under Creative Commons license BY-SA 3.0.

About Us

Amarula Solutions

info@amarulasolutions.com
www.amarulasolutions.com

- Software Consulting Across Europe
  - Offices in multiple European countries
  - Focused on mobile apps, cloud platforms, and embedded systems
- Full-Stack Embedded Expertise
  - From bootloaders & kernels to UI applications
  - Deep knowledge of Android & Linux OS
- Open Source at our Core
  - Active upstream contributions
  - Solutions built for security and long-term maintainability



About Us

Andrea Ricchi

andrea.ricchi@amarulasolutions.com

https://andrearicchi.github.io/

- C
- C++ & Qt
- Dart & Flutter
- Yocto & Buildroot





About Us

Dario Binacchi

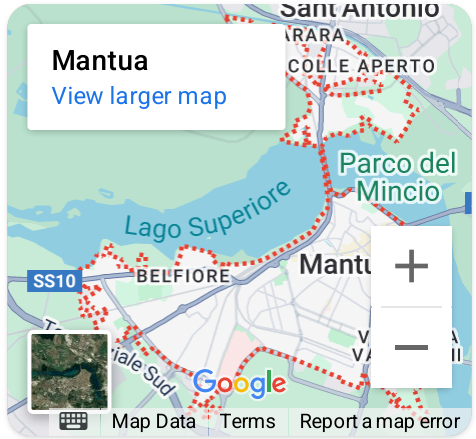
dario.binacchi@amarulasolutions.com

https://github.com/passgat

https://openhub.net/accounts/passgat

- Buildroot
- Yocto
- Linux
- U-Boot
- Zephyr





Agenda

- Hardware setup
- Yocto
- Flutter
- Zephyr
- LVGL
- Demo
- Q&A

Embedded World 2025

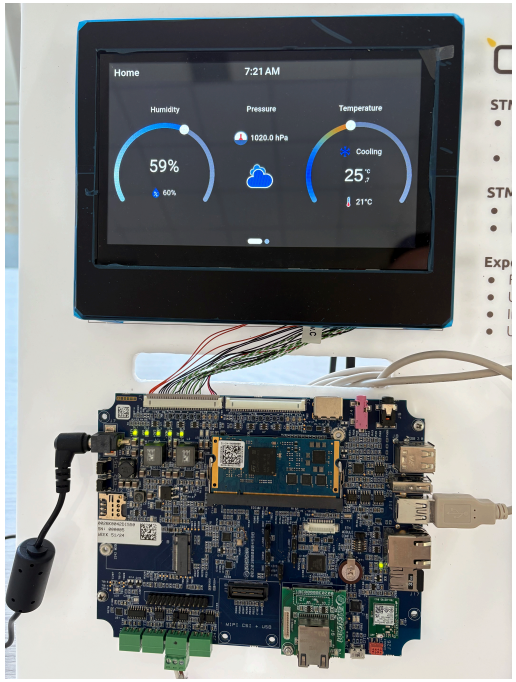
- **Embedded Linux build systems**
- **Graphical frameworks**
- **FOSS contributions**
- **Integration and adaption to new hardware**
- **ST authorised partner program**
- **Using upstream-oriented software to support CRA compliance**
- **Designing a reusable infrastructure**
- **Showcasing at Engicam booth**

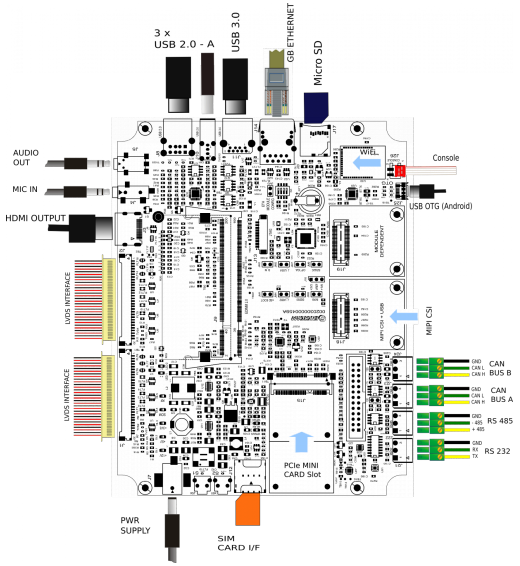
Output

- **Smart thermostat demo**
- **Main board**
  - EDIMM 2.0 EVK - STM32MP257 SOM
  - Yocto - scarthgap
  - Flutter - 3.32.2
- **Sensor board**
  - STM32F429I-DISC1 board
  - Temperature, pressure and humidity sensors
  - Zephyr RTOS - 4.0.0
  - LVGL UI - 8.4.0
- **Boards communication over CAN bus**

Hardware setup - Main board

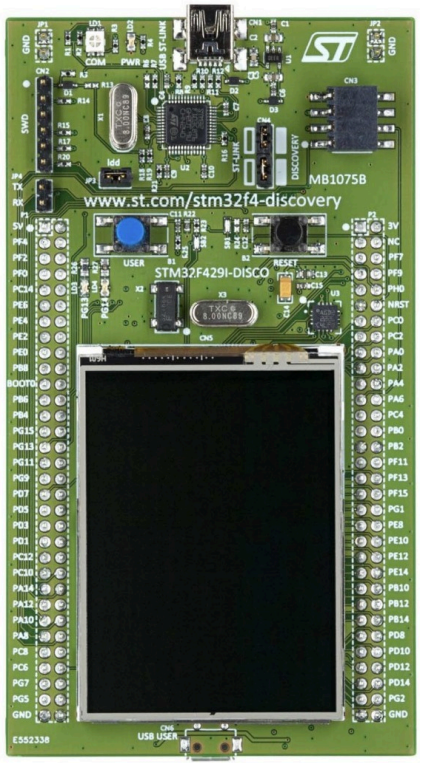
- **Engicam EDIMM 2.0 STARTER KIT**
- **STM32MP257 SOM**
  - 2 Cortex®-A35 up to 1.5 GHz
  - 1 Cortex®-M33 at 400 MHz
  - H.264 encoding/decoding
  - 3D GPU
    - OPENGL ES3.1
    - Vulkan 1.3
    - OpenCL 3.0
  - 3 CAN
- **RAM 2GB**
- **CAN bus connector**



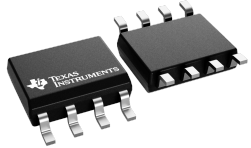


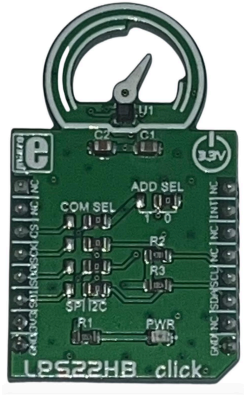
Hardware setup - Sensor board

- **STM32f429I-DISC1 board**
  - Arm® Cortex®-M4 MCU up to 180 Mhz
  - 2MB Flash
  - 8MB SDRAM
  - 2.4" QVGA TFT LCD
  - 2 CAN
- **LPS22HB - pressure**
- **SHT4x - temperature & humidity**
- **SN65HVD230 - CAN transceiver**



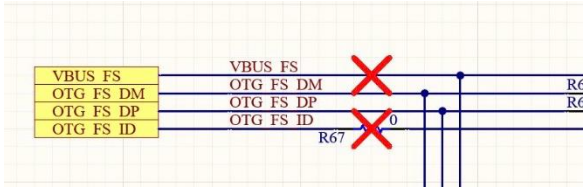


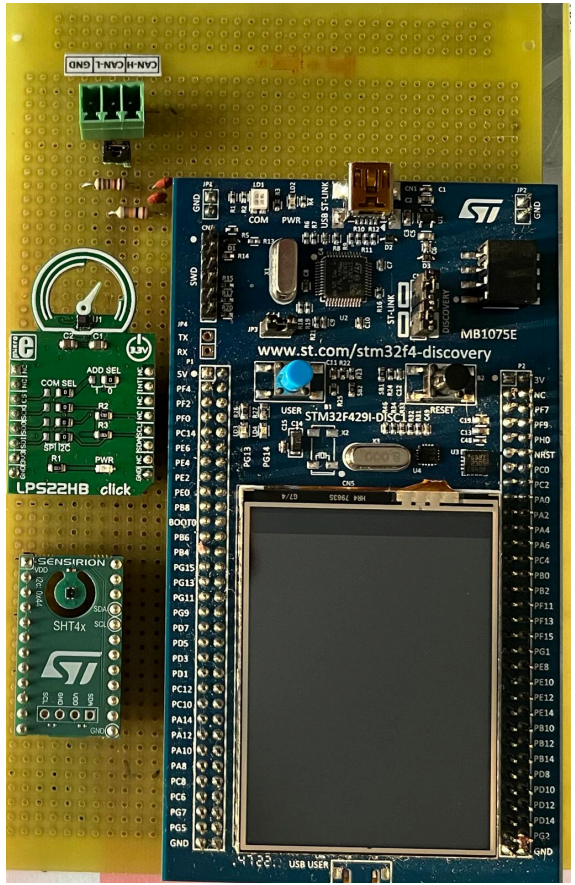


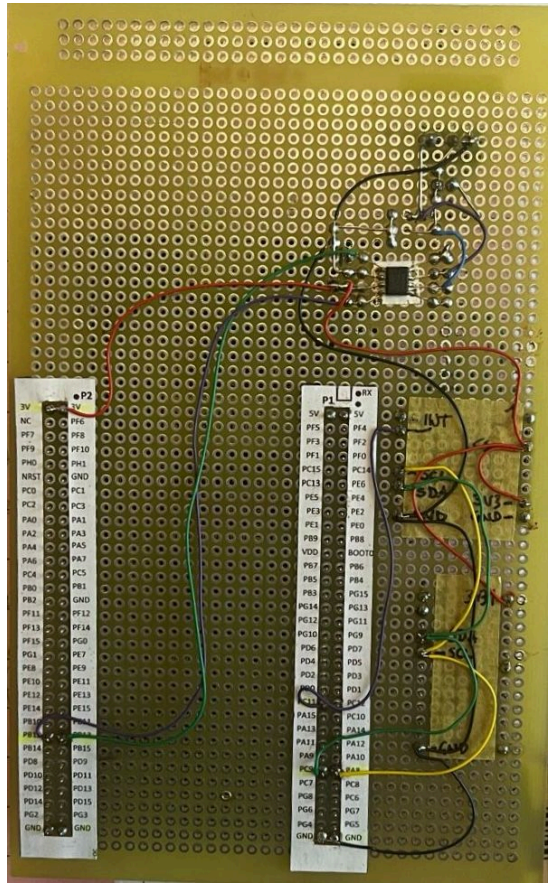


Hardware setup - Sensor board

- **Sensors wiring to I²C bus**
- **CAN bus wiring**
  - Fix USB OTG conflicts
  - Transceiver
  - Connector







Yocto



- Flexible
- Scalable
- Customizable
- Reproducible
- Package-based
- Toolchain control
- Community-driven

Yocto - Layers Setup

This configuration combines core OpenEmbedded layers with hardware-specific BSPs and additional functionality layers.

```
# conf/bblayers.conf

BBLAYERS ?= " \
    ${TOPDIR}/../work/openembedded-core/meta" \
    ${TOPDIR}/../work/meta-openembedded/meta-multimedia \
    ${TOPDIR}/../work/meta-openembedded/meta-gnome \
    ${TOPDIR}/../work/meta-openembedded/meta-oe \
    ${TOPDIR}/../work/meta-openembedded/meta-networking \
    ${TOPDIR}/../work/meta-openembedded/meta-webserver \
    ${TOPDIR}/../work/meta-openembedded/meta-python \
    ${TOPDIR}/../work/meta-st-openstlinux \
    ${TOPDIR}/../work/meta-st-stm32mp \
    ${TOPDIR}/../work/meta-st-stm32mp-addons \
    ${TOPDIR}/../work/meta-engicam-st \
    ${TOPDIR}/../work/meta-clang \
    ${TOPDIR}/../work/meta-flutter \
    ${TOPDIR}/../work/meta-amarula-demo
```

Yocto - Layers Setup

This configuration combines core OpenEmbedded layers with hardware-specific BSPs and additional functionality layers.

```
# conf/bblayers.conf

BBLAYERS ?= " \
    ${TOPDIR}/../work/openembedded-core/meta" \
    ${TOPDIR}/../work/meta-openembedded/meta-multimedia \
    ${TOPDIR}/../work/meta-openembedded/meta-gnome \
    ${TOPDIR}/../work/meta-openembedded/meta-oe \
    ${TOPDIR}/../work/meta-openembedded/meta-networking \
    ${TOPDIR}/../work/meta-openembedded/meta-webserver \
    ${TOPDIR}/../work/meta-openembedded/meta-python \
    ${TOPDIR}/../work/meta-st-openstlinux \
    ${TOPDIR}/../work/meta-st-stm32mp \
    ${TOPDIR}/../work/meta-st-stm32mp-addons \
    ${TOPDIR}/../work/meta-engicam-st \
    ${TOPDIR}/../work/meta-clang \
    ${TOPDIR}/../work/meta-flutter \
    ${TOPDIR}/../work/meta-amarula-demo
```

Yocto - Layers Setup

This configuration combines core OpenEmbedded layers with hardware-specific BSPs and additional functionality layers.

```
# conf/bblayers.conf

BBLAYERS ?= " \
    ${TOPDIR}/../work/openembedded-core/meta" \
    ${TOPDIR}/../work/meta-openembedded/meta-multimedia \
    ${TOPDIR}/../work/meta-openembedded/meta-gnome \
    ${TOPDIR}/../work/meta-openembedded/meta-oe \
    ${TOPDIR}/../work/meta-openembedded/meta-networking \
    ${TOPDIR}/../work/meta-openembedded/meta-webserver \
    ${TOPDIR}/../work/meta-openembedded/meta-python \
    ${TOPDIR}/../work/meta-st-openstlinux \
    ${TOPDIR}/../work/meta-st-stm32mp \
    ${TOPDIR}/../work/meta-st-stm32mp-addons \
    ${TOPDIR}/../work/meta-engicam-st \
    ${TOPDIR}/../work/meta-clang \
    ${TOPDIR}/../work/meta-flutter \
    ${TOPDIR}/../work/meta-amarula-demo
```

Yocto - Layers Setup

This configuration combines core OpenEmbedded layers with hardware-specific BSPs and additional functionality layers.

```
# conf/bblayers.conf

BBLAYERS ?= " \
    ${TOPDIR}/../work/openembedded-core/meta" \
    ${TOPDIR}/../work/meta-openembedded/meta-multimedia \
    ${TOPDIR}/../work/meta-openembedded/meta-gnome \
    ${TOPDIR}/../work/meta-openembedded/meta-oe \
    ${TOPDIR}/../work/meta-openembedded/meta-networking \
    ${TOPDIR}/../work/meta-openembedded/meta-webserver \
    ${TOPDIR}/../work/meta-openembedded/meta-python \
    ${TOPDIR}/../work/meta-st-openstlinux \
    ${TOPDIR}/../work/meta-st-stm32mp \
    ${TOPDIR}/../work/meta-st-stm32mp-addons \
    ${TOPDIR}/../work/meta-engicam-st \
    ${TOPDIR}/../work/meta-clang \
    ${TOPDIR}/../work/meta-flutter \
    ${TOPDIR}/../work/meta-amarula-demo
```

Yocto - Layers Setup

This configuration combines core OpenEmbedded layers with hardware-specific BSPs and additional functionality layers.

```
# conf/bblayers.conf

BBLAYERS ?= " \
    ${TOPDIR}/../work/openembedded-core/meta" \
    ${TOPDIR}/../work/meta-openembedded/meta-multimedia \
    ${TOPDIR}/../work/meta-openembedded/meta-gnome \
    ${TOPDIR}/../work/meta-openembedded/meta-oe \
    ${TOPDIR}/../work/meta-openembedded/meta-networking \
    ${TOPDIR}/../work/meta-openembedded/meta-webserver \
    ${TOPDIR}/../work/meta-openembedded/meta-python \
    ${TOPDIR}/../work/meta-st-openstlinux \
    ${TOPDIR}/../work/meta-st-stm32mp \
    ${TOPDIR}/../work/meta-st-stm32mp-addons \
    ${TOPDIR}/../work/meta-engicam-st \
    ${TOPDIR}/../work/meta-clang \
    ${TOPDIR}/../work/meta-flutter \
    ${TOPDIR}/../work/meta-amarula-demo
```

Yocto - Layers Setup

This configuration combines core OpenEmbedded layers with hardware-specific BSPs and additional functionality layers.

```
# conf/bblayers.conf

BBLAYERS ?= " \
    ${TOPDIR}/../work/openembedded-core/meta" \
    ${TOPDIR}/../work/meta-openembedded/meta-multimedia \
    ${TOPDIR}/../work/meta-openembedded/meta-gnome \
    ${TOPDIR}/../work/meta-openembedded/meta-oe \
    ${TOPDIR}/../work/meta-openembedded/meta-networking \
    ${TOPDIR}/../work/meta-openembedded/meta-webserver \
    ${TOPDIR}/../work/meta-openembedded/meta-python \
    ${TOPDIR}/../work/meta-st-openstlinux \
    ${TOPDIR}/../work/meta-st-stm32mp \
    ${TOPDIR}/../work/meta-st-stm32mp-addons \
    ${TOPDIR}/../work/meta-engicam-st \
    ${TOPDIR}/../work/meta-clang \
    ${TOPDIR}/../work/meta-flutter \
    ${TOPDIR}/../work/meta-amarula-demo
```

Yocto - Challanges

meta-flutter **scarthgap**

meta-engicam-st (engicam) **mickledore**

meta-engicam-st (amarula) **scarthgap**

Yocto - Configuration

This configuration is tailored for efficient, reproducible builds and integrates project-specific components like.

```
# conf/local.conf

MACHINE ??= "stm32mp25-icore"
DISTRO ??= "openstlinux-weston"
BBMULTICONFIG ?= ""

BB_NUMBER_THREADS ?= "12"
PARALLEL_MAKE ?= "-j 6"

# ST specific
EULA_FILE_ST = "1"
ACCEPT_EULA_stm32mp25-icore = "1"
GLIBC_GENERATE_LOCALES = "en_GB.UTF-8 en_US.UTF-8"
IMAGE_LINGUAS ?= "en-gb"
INHERIT += "devshell"
RM_OLD_IMAGE = "1"

# Force the usage of debian package
PACKAGE_CLASSES = "package_deb"

# Setup environment for builds binary reproducibility
REPRODUCIBLE_TIMESTAMP_ROOTFS = ""
```

Yocto - Configuration

This configuration is tailored for efficient, reproducible builds and integrates project-specific components like.

```
# conf/local.conf

# Setup eSDK
SDK_EXT_TYPE="minimal"
SDK_INCLUDE_TOOLCHAIN="1"

# Enable PR server to avoid version-going-backward issue
PRSERV_HOST = "localhost:0"

# Archiver
HOSTTOOLS += "xz truncate"

# Flutter
IMAGE_INSTALL:append = " \
    flutter-engine \
    flutter-wayland-client \
    ew-2025-flutter-demo-can-lib \
    ew-2025-flutter-demo \
    connman \
    connman-client \
"

TOOLCHAIN_HOST_TASK:append = " nativesdk-flutter-sdk"
```

Yocto - KAS

This tool provides an easy mechanism to set up bitbake-based projects.

- **Clone and checkout bitbake layers**
- **Create default bitbake settings (machine, arch)**
- **Launch minimal build environment, reducing risk of host contamination**
- **Initiate bitbake build process**

Yocto - KAS config

A single YAML file that defines both layers and build configuration, making the Yocto setup reproducible, portable, and easy to manage.

```
# .config.yaml

machine: stm32mp25-icore
distro: openstlinux-weston
target: st-image-weston

repos:
    bitbake:
        url: "https://git.openembedded.org/bitbake"
        branch: "2.10"
        layers:
            .: excluded

    openembedded-core:
        url: "https://git.openembedded.org/openembedded-core"
        commit: 236ac1b43308df722a78d3aa20aef065dfae5b2b
        layers:
            meta:

[ ... ]
```

Yocto - KAS config

A single YAML file that defines both layers and build configuration, making the Yocto setup reproducible, portable, and easy to manage.

```yaml
# .config.yaml

machine: stm32mp25-icore
distro: openstlinux-weston
target: st-image-weston

repos:
    bitbake:
        url: "https://git.openembedded.org/bitbake"
        branch: "2.10"
        layers:
            .: excluded

    openembedded-core:
        url: "https://git.openembedded.org/openembedded-core"
        commit: 236ac1b43308df722a78d3aa20aef065dfae5b2b
        layers:
            meta:

[ ... ]
```

Yocto - KAS config

A single YAML file that defines both layers and build configuration, making the Yocto setup reproducible, portable, and easy to manage.

```yaml
# .config.yaml

machine: stm32mp25-icore
distro: openstlinux-weston
target: st-image-weston

repos:
    bitbake:
        url: "https://git.openembedded.org/bitbake"
        branch: "2.10"
        layers:
            .: excluded

    openembedded-core:
        url: "https://git.openembedded.org/openembedded-core"
        commit: 236ac1b43308df722a78d3aa20aef065dfae5b2b
        layers:
            meta:

[ ... ]
```

Yocto - KAS config

A single YAML file that defines both layers and build configuration, making the Yocto setup reproducible, portable, and easy to manage.

```
local_conf_header:
    standard: |
        BB_NUMBER_THREADS ?= "12"
        PARALLEL_MAKE ?= "-j 6"
        EULA_FILE_ST = "1"
        ACCEPT_EULA_stm32mp25-icore = "1"
        GLIBC_GENERATE_LOCALES = "en_GB.UTF-8 en_US.UTF-8"
        IMAGE_LINGUAS ?= "en-gb"
        INHERIT += "devshell"
        RM_OLD_IMAGE = "1"
        PACKAGE_CLASSES = "package_deb"
        REPRODUCIBLE_TIMESTAMP_ROOTFS = ""
        SDK_EXT_TYPE="minimal"
        SDK_INCLUDE_TOOLCHAIN="1"
        PRSERV_HOST = "localhost:0"
        HOSTTOOLS += "xz truncate"
        IMAGE_INSTALL:append = " \
            flutter-engine \
            flutter-wayland-client \
            ew-2025-flutter-demo-can-lib \
            ew-2025-flutter-demo \
            connman \
            connman-client \
        "
        TOOLCHAIN_HOST_TASK:append = " nativesdk-flutter-sdk"
```

Yocto - KAS Usage

Run KAS on the host

```
$ kas checkout
$ kas build
```

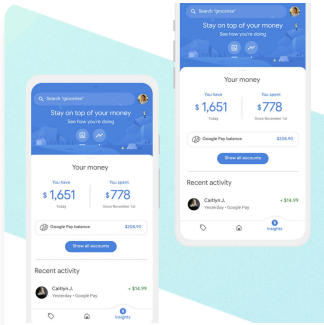
Run KAS inside a build container

```
$ kas-container checkout
$ kas-container build
```

Flutter

Flutter

Made by Google



Mobile



Web



Desktop



Embedded

Flutter - Power

This framework's developer experience and power are top-notch.

- **Easy to learn and use**
- **Hot reload**
- **Null safety and async/await support**
- **Natively compiled for great performance**
- **Cross-platform with consistent UI**
- **Excellent tooling for development and debugging**

Flutter - Ecosystem





- get
- fl_chart
- google_fonts
- animations
- flutekeyboard

Flutter - C++

Flutter apps can use the FFI library to call native C APIs and to read, write, allocate, and deallocate native memory.

```c
static double k_temperature = 1;

EXPORT double get_temperature() { return k_temperature; }
```

```dart
final lib = DynamicLibrary.open(libraryPath);

final getTemperature = lib
    .lookup<NativeFunction<DoubleFuncCpp>>('get_temperature')
    .asFunction<DoubleFunc>();

double _temperature = getTemperature();
```

Flutter - Yocto

Flutter SDK

Yocto meta-flutter

Add to local.conf file:

```
TOOLCHAIN_HOST_TASK:append = " nativesdk-flutter-sdk"
```

Then run:

```
bitbake <image name> -c populate_sdk
```

Flutter - Yocto

Flutter Engine

Yocto meta-flutter

The Flutter Engine is a low-level runtime that provides the core functionality for Flutter applications

Flutter - Yocto

Yocto meta-flutter

Flutter Embedders

An embedder is the platform-specific code that integrates the Flutter engine with the device's operating system, graphics stack, input system, and other low-level services.

- flutter-pi
- flutter-wayland-client
- ivi-homescreen

Flutter - Yocto

Yocto meta-flutter

Flutter App Recipe

```
SRCREV = "${AUTOREV}"
SRC_URI = " \
    git://github.com/amarula/ew_2025_flutter_demo.git;branch=main;protocol=https;destsuffix=git \
    file://ew-2025-flutter-demo \
    file://ew-2025-flutter-demo.service \
    "

S = "${WORKDIR}/git"

PUBSPEC_APPNAME = "ew_2025_flutter_demo"
FLUTTER_APPLICATION_INSTALL_SUFFIX = "ew_2025_flutter_demo"
FLUTTER_APPLICATION_PATH = ""

inherit flutter-app

inherit systemd
SYSTEMD_PACKAGES += " ${PN} "
SYSTEMD_SERVICE:${PN} = "ew-2025-flutter-demo.service"
SYSTEMD_AUTO_ENABLE:${PN} = "enable"
```

Zephyr



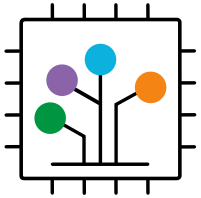
- **Open source RTOS**
- **Low memory footprint**
- **High portability - 500+ boards**
- **LTS support**
- **Linux like - Kconfig & Devicetree**
- **Ecosystem - not only the kernel**
  - Developing tools - west
  - Device drivers
  - Connectivity
  - Much more

Zephyr - Application

- **Build system**
  - Based on CMake
  - Application centric
- **Application directory**
  - CMakeLists.txt
    - Where to find application files - src/*.c
    - Link application to the Zephyr's build system
  - app.overlay
    - Devicetree changes
    - Apply to the Zephyr devicetree board
    - Turn STM32F429I-DISC1 into sensor board
  - prj.conf
    - Kconfig fragment
    - Merge with Zephyr configuration

```
<app>
├── CMakeLists.txt
├── app.overlay
├── prj.conf
└── src
    └── *.c
```







Zephyr - Application types

- **Repository application**
- **Workspace application**
  - Avoid unnecessary changes to Zephyr repo
  - Use a reference workspace application
- **Freestanding application**

```
# Zephyr workspace application for sensor board

$ source ~/zephyrproject/.venv/bin/activate
$ west init -m https://github.com/amarula/ew_lvgl_demo --mr main \
          ew_lvgl_demo_workspace
$ cd ew_lvgl_demo_workspace
$ west update
$ tree
├── ew_lvgl_demo
│   └── app
├── modules
│   ├── hal
│   │   └── stm32
│   └── lib
│       └── gui
│           └── lvgl
└── zephyr
```

```
# Zephyr repository application

zephyrproject/
└── zephyr/
    ├── ...
    └── samples/
        └── app/
```

```
# Zephyr workspace application

zephyrproject/
├── zephyr/
├── ...
└── applications/
    └── app/
```

```
# Zephyr freestanding application

<home>/
├── zephyrproject/
│   ├── .west/
│   │   └── config
│   ├── zephyr/
│   └── ...
│
└── app/
```

Zephyr - CMakeLists.txt

- **find_package()**
  - Pull in the Zephyr build system
    - Create a CMake target named app
  - $ENV{ZEPHYR_BASE} set by west build command

- **target_sources()**
  - Add application sources to the app target

```
# app/CMakeLists.txt

cmake_minimum_required(VERSION 3.13)

find_package(Zephyr REQUIRED HINTS $ENV{ZEPHYR_BASE})

project(Amarula-Demo VERSION 0.1.0 LANGUAGES C)

target_sources(app
               PRIVATE src/main.c
               src/lvgl_gui.h
               src/lvgl_gui.c
               resources/temperature.c
               resources/humidity.c
               resources/pressure.c
               resources/mono_font.c
               src/led.h
               src/led.c
               src/can.h
               src/can.c)
```

Zephyr - app.overlay

- **Extend the STM32F429I-DISC1 board**
- **Align with hardware changes**
- **Populate the i2c3 bus**
  - Temperature & humudity sensor @ 0x44
  - Pressure sensor @ 0x5c
- **Enable CAN controller**
  - Disable USB OTG - pins shared
  - No transceiver description

```
# app/boards/stm32f429i_disc1.overlay

&i2c3 {
    sht4x@44 {
        compatible = "sensirion,sht4x";
        status = "okay";
        reg = <0x44>;
        repeatability = <2>;
    };

    lps22hb@5c {
        compatible = "st,lps22hb-press";
        status = "okay";
        reg = <0x5c>;
    };
};

zephyr_udc0: &usbotg_hs {
    status = "disabled";
};

&can2 {
    pinctrl-0 = <&can2_rx_pb12 &can2_tx_pb13>;
    pinctrl-names = "default";
    status = "okay";
};
```

Zephyr - prj.conf

- I2C bus
- Sensor
- CAN bus
- LED
- Graphical library
- What about the drivers?
  - CAN_STM32_BXCAN
  - LPS22HB
  - SHT4X
  - ILI9341

```
# app/prj.conf

CONFIG_LOG=y
CONFIG_SHELL=y

CONFIG_I2C=y
CONFIG_SENSOR=y

CONFIG_CAN=y

CONFIG_LED=y

CONFIG_DISPLAY=y

CONFIG_LVGL=y
CONFIG_LV_USE_LABEL=y
CONFIG_LV_USE_BTN=y
CONFIG_LV_ ... =y
```

Zephyr - app/*.c

- **Clear API provided by Zephyr and its module**
- **A lot of application examples**
  - samples/drivers/display
  - samples/subsys/display/lvgl
  - samples/net/sockets/can
  - samples/sensor/lps22hb
  - samples/sensor/sht4x




```
$ source ~/zephyrproject/.venv/bin/activate
$ cd ~/zephyrproject/zephyr

$ west build --pristine -b stm32f429i_disc1 samples/drivers/display
$ west flash --runner openocd --config ${HOME}/zephyr-sdk-0.17.0/sysroots/x86_64-pokysdk-linux/usr/share/openocd/scripts/board/stm32f429disc1.cfg

$ west build --pristine -b stm32f429i_disc1 samples/subsys/display/lvgl
$ west flash --runner openocd --config ${HOME}/zephyr-sdk-0.17.0/sysroots/x86_64-pokysdk-linux/usr/share/openocd/scripts/board/stm32f429disc1.cfg

# change boards/st/stm32f429i_disc1/stm32f429i_disc1.dts to enable CAN2 controller
$ west build --pristine -b stm32f429i_disc1 samples/net/sockets/can/
$ west flash --runner openocd --config ${HOME}/zephyr-sdk-0.17.0/sysroots/x86_64-pokysdk-linux/usr/share/openocd/scripts/board/stm32f429disc1.cfg
```

Zephyr - FOSS contribution

- **Devicetree**
  - 2 bxCAN controllers on STM32F429 MCU
  - They share RAM (512 bytes) and filters
  - CAN1 primary controller
  - CAN2 secondary controller
  - Enable CAN1 RCC clock to use CAN2
  - Need to fix a regression by code refactoring
  - bxCAN internals? Don't miss Amarula's talk at FOSDEM 2024

```
/ {
    chosen {
        zephyr,canbus = &can2;
    };

    soc {
        can2: can@40006800 {
                clocks = <&rcc STM32_CLOCK_BUS_APB1 0x06000000>;
        };
    };
};
```

```
~/zephyrproject/zephyr$ git log --oneline --author="Dario Binacchi"
2c3294b079cf dts: arm: st: re-enable master can gating clock for can2
ed48de2583af boards: st: stm32f429i_disc1: doc: re-work flashing section
3f6978e913ff boards: st: stm32f429i_disc1: doc: support LTDC hardware
7bca40cceda4 boards: st: stm32f429i_disc1: doc: fix flashing info
```

- **Documentation**
  - Fix flashing section
  - LTDC hardware now supported

LVGL



Light and Versatile Graphics Library

- **Open source**
- **Written in C**
- **Low memory footprint**
- **Wide range of widgets**
- **Support input devices**
- **Zephyr packaged with LVGL**

LVGL - Zephyr integration

- **modules/lib/gui/lvgl/**
  - LVGL codebase
  - west update
  - Version in lvgl.h - 8.4.0

- **zephyr/modules/gui/lvgl**
  - Zephyr Kconfig options
  - Zephyr glue code
    - DT_CHOSEN(zephyr_display)
    - SYS_INIT(lvgl_init, APPLICATION, ...)

```
~zephyrprojects/
├── modules
│   └── lib
│       └── gui
│           └── lvgl
└── zephyr
    └── modules
        └── lvgl
```

```
int lvgl_init(void)
{
    for (int i = 0; i < DT_ZEPHYR_DISPLAYS_COUNT; i++) {
        if (!device_is_ready(display_dev[i])) {
            LOG_ERR("Display device %d is not ready", i);
            return -ENODEV;
        }
    }

    for (int i = 0; i < DT_ZEPHYR_DISPLAYS_COUNT; i++) {
        err = lvgl_allocate_rendering_buffers(lv_displays[i]);
    }

    err = lvgl_init_input_devices();
}
```

LVGL - Display Kconfig

- **Display subsystem**
- **Display driver**
  - ILI9341 from Ilitek

```
# app/prj.conf

CONFIG_DISPLAY=y
```

```
# drivers/display/Kconfig

if DISPLAY
source "drivers/display/Kconfig.ili9xxx"
endif # DISPLAY
```

```
# drivers/display/Kconfig.ili9xxx

config ILI9341
    bool "ILI9341 display driver"
    default y
    depends on DT_HAS_ILITEK_ILI9341_ENABLED
    ...
```

```
# boards/st/stm32f429i_disc1/stm32f429i_disc1.dts

ili9341: ili9341@0 {
     compatible = "ilitek,ili9341";
      ...
    }
```

```
# build/zephyr/.config

CONFIG_ILI9341=y
```

LVGL - Display DTS

- **zephyr,display property**
- **zephyr,lvgl-pointer-input compatible**
- **lvgl_init()**
  - Just before the application code (main)
  - Turn on display
  - Turn on input devices - touchscreen, mouse, ...
- **Hardware independent application code**

```
# boards/st/stm32f429i_disc1/stm32f429i_disc1.dts

chosen {
    ...
    zephyr,display = &ltdc;
    zephyr,touch = &stmpe811;
};

lvgl_pointer {
    compatible = "zephyr,lvgl-pointer-input";
    input = <&stmpe811>;
};
```

```
# modules/lvgl/lvgl.c

SYS_INIT(lvgl_init, APPLICATION, CONFIG_LV_Z_INIT_PRIORITY);
```

```
$ west build -t initlevels
EARLY ...
PRE_KERNEL_1 ...
PRE_KERNEL_2 ...
POST_KERNEL ...
 __init___device_dts_ord_99: ili9xxx_init(__device_dts_ord_99)
 __init___device_dts_ord_122: stm32_ltdc_init(__device_dts_ord_122)
 __init___device_dts_ord_16: stmpe811_init(__device_dts_ord_16)
APPLICATION
 __init_lvgl_init: lvgl_init(NULL)
SMP
```

LVGL - Library Kconfig

- **Enable LVGL**
- **Enable only the used widgets**
- **Fine-tune the application footprint**

```
# app/prj.cof

CONFIG_LVGL=y

CONFIG_LV_Z_MEM_POOL_SIZE=16384
CONFIG_LV_Z_SHELL=y
CONFIG_LV_MEM_CUSTOM=y

CONFIG_LV_USE_LABEL=y
CONFIG_LV_USE_BTN=y
CONFIG_LV_USE_ARC=y
CONFIG_LV_USE_IMG=y
CONFIG_LV_FONT_MONTSERRAT_16=y
CONFIG_LV_FONT_MONTSERRAT_32=y
```

LVGL - Application

- **In LVGL, everything is an object**
  - Create the object
  - Setup its properties
  - Place on the screen
- **Start from the basic data flow**
  1. Driver initialization - Zephyr
  2. Run lvgl_init() - Zephyr
  3. Create display and input devices - Zephyr
  4. **Create GUI**
  5. Read data from sensors
  6. Update widgets properties
  7. **Refresh the screen**

```
/* Create a white label, set its text and align it to the center */
lv_obj_t * label = lv_label_create(lv_screen_active());
lv_label_set_text(label, "Hello world");
lv_obj_set_style_text_color(lv_screen_active(), lv_color_hex(0xffffff),
                            LV_PART_MAIN);
lv_obj_align(label, LV_ALIGN_CENTER, 0, 0);
```

```
void main(void) {

     struct sensor_value temp;
     const struct device *const sht =
              DEVICE_DT_GET_ANY(sensirion_sht4x);

     gui_init();

     for (;;) {
          sensor_channel_get(sht, SENSOR_CHAN_AMBIENT_TEMP, &sv_temp);

          gui_set_temperature(temp);

          lv_timer_handler();
          k_sleep(K_MSEC(5));
     }
}
```

LVGL - Memory footprint

#	Applications	FLASH (2 MB)	RAM (192 KB)	SDRAM2 (8 MB)
1	samples/hello_world	16 KB (1%)	5 KB (2%)	0 KB (0%)
2	ew_demo (-DENABLE_GUI=OFF)	36 KB (2%)	9 KB (4%)	0 KB (0%)
3	samples/drivers/display	46 KB (2%)	26 KB (14%)	150 KB (2%)
4	samples/subsys/display/lvgl	251 KB (12%)	65 KB (33%)	150 KB (2%)
5	ew_demo	418 KB (20%)	63 KB (32%)	150 KB (2%)

- With GUI enabled: +380 KB FLASH, +54 KB RAM (5 vs 2)
- 150KB SDRAM2 used for framebuffer (3, 4, 5)
- Contained memory footprint, allowing for application extension (5)
  - < 20% of FLASH
  - ~33% of RAM
- Well suited for mid-range MCUs like the STM32F429

LVGL - Runtime memory profile

- **Zephyr shell - CONFIG_SHELL**
- **LVGL stats commands**
  - Enabled by CONFIG_LV_Z_SHELL
  - *lvgl stats memory*
- **LVGL monkey commands**
  - Enabled by CONFIG_LV_USE_MONKEY
  - *lvgl monkey create {pointer|keypad|button|encoder}*
  - *lvgl monkey set <index> {inactive|active}*
- **Useful for**
  - Tuning memory setup
  - Identify memory leaks
  - Test the UI

```
# app/prj.conf

CONFIG_SHELL=y
CONFIG_LV_Z_SHELL=y

CONFIG_LV_USE_MONKEY=y
```

```
# create a monkey for the touchscreen
uart:~$ lvgl monkey create pointer
id   device    active
0    pointer   yes

# get memory statistics
uart:~$ lvgl stats memory
Heap at 0×20001838 contains 2047 units in 11 buckets

  bucket#    min units        total      largest      largest
             threshold       chunks      (units)      (bytes)
  ------------------------------------------------------------
       0            1            5            1            4
       3            8            1            9           68
       4           16            1           25          196
       6           64            1           95          756
       9          512            1          554         4428

5468 free bytes, 9692 allocated bytes, overhead = 1220 bytes (7.4%)
```

Demo



Acknowledgements

This talk was made possible thanks to the help and encouragement of many people.

A special thanks to Alberto Bianchi for the designing and building the hardware, Michael Trimarchi for the Yocto operating system integration, and Eduardo Gonzalez for developing the application on the sensor board.

Thanks to Benjamin Cabé, Erwan Gouriou, Marwa, Henrik Brix Andersen, Mathieu CHOPLAIN and Abderrahmane Jarmouni for the review and acceptance of the Zephyr project patches.

Thanks to Alberto Panizzo, Michael Trimarchi and Vera Binacchi for the slides review and refinement.

Thanks to Amarula and the ELCE organisation for the assistance and support.

We are sorry if we forgot someone 😃.

Resources

Flutter

https://github.com/meta-flutter/meta-flutter
https://flutter.dev
https://pub.dev

Hands-on

https://github.com/amarula/ew_2025_flutter_demo
https://github.com/amarula/ew_2025_lvgl_demo
https://github.com/amarula/meta-engicam-st
https://github.com/amarula/meta-amarula-demo

Hardware

https://www.st.com/en/microcontrollers-microprocessors/stm32mp257.html
https://www.engicam.com/vis-prod/101366/EDIMM-2-0-STARTER-KIT-EDIMM-2-0-Form-Factor-Capacitive-Evaluation-Board
https://www.st.com/en/microcontrollers-microprocessors/stm32f429-439.html
https://www.st.com/en/evaluation-tools/32f429idiscovery.html
https://community.st.com/ysqtg83639/attachments/ysqtg83639/mcu-touch-gfx-gui-forum/1401/1/FRIDA_LCD_FRD240C48003-B.pdf
https://www.mikroe.com/lps22hb-click?srsltid=AfmBOoqLNTD-FiAuQrsgVKIiQ03Lkr36fZFyAKXXYfBffaFiuQxTN0Ok
https://sensirion.com/products/catalog/SENSEVAL-SHT4XV1
https://www.ti.com/lit/ds/symlink/sn65hvd230.pdf?ts=1752742041247&ref_url=https%253A%252F%252Fwww.google.com%252F
https://archive.fosdem.org/2024/schedule/event/fosdem-2024-2864-linux-can-upstreaming-on-mmu-less-systems

Resources

LVGL

https://docs.lvgl.io/master/index.html
https://docs.lvgl.io/master/intro/getting_started/learn_the_basics.html
https://docs.lvgl.io/master/details/integration/os/zephyr.html
https://docs.lvgl.io/master/intro/getting_started/learn_the_basics.html
https://www.youtube.com/watch?v=fRB9gn77XhE
https://www.youtube.com/watch?v=KfgIn6RdoYc

Slides

https://github.com/amarula/feasibility-architecture-dualboard-yocto-zephyr

Zephyr

https://www.zephyrproject.org
https://docs.zephyrproject.org/latest/develop/getting_started/index.html
https://docs.zephyrproject.org/latest/contribute/index.html
https://docs.zephyrproject.org/latest/boards/st/stm32f429i_disc1/doc/index.html
https://github.com/zephyrproject-rtos/example-application

Yocto

https://www.yoctoproject.org/
https://layers.openembedded.org/layerindex/branch/master/layers
https://github.com/engicam-stable/meta-engicam-st

Q&A

Thanks for your time

Question?
Comments?
Suggestions?

