



AUTOMOTIVE  
GRADE LINUX

ALL MEMBER MEETING



MAY 13 – 14, 2026

TOKYO, JAPAN

# Ultimate Unagi: Flutter App Development on Real Target Hardware

*Jan-Simon Möller, [jsmoeller@linuxfoundation.org](mailto:jsmoeller@linuxfoundation.org)*

# Intro



Jan-Simon Möller

Director of Release Management and Developer Community, AGL

CIAT/CICD, YP AB for AGL, GSOC@AGL

[jsmoeller@linuxfoundation.org](mailto:jsmoeller@linuxfoundation.org)

# Topics



- AGL and Flutter
- meta-flutter
  - workspace-tool
- Setting up your dev environment
  - Basic setup
  - Setup using AGL image
  - Setup using AGL image and VScode
- Attach to hardware!
- Q/A



MAY 13 – 14, 2026

TOKYO, JAPAN

# AGL and Flutter

- Automotive Grade Linux (AGL) transitioned to a Flutter-based UI for its Unified-Code-Base (UCB) demo.
- The UI was created by ICS and AGL within a short 3-4 month timeframe
- This talk will help you setup a powerful development environment





MAY 13 – 14, 2026

TOKYO, JAPAN

# meta-flutter



# meta-flutter



meta-flutter



Overview



Repositories

74



Projects



Packages



People



## meta-flutter

- meta-flutter is the home of the 'Embedded Flutter'
- It hosts 3 main projects:
  - meta-flutter/meta-flutter - the Yocto Project layer
  - meta-flutter/workspace-automation - setup tooling
  - meta-flutter/flutter-engine - port of the flutter engine
- <https://github.com/meta-flutter/>

# meta-flutter/meta-flutter - the YP layer



- AGL UCB is based on Yocto Project's LTS 'scarthgap'
- The 'unagi' branch of AGL uses meta-flutter 'scarthgap'
- Regular uprevs of Yocto Project and meta-flutter about every other month

meta-flutter / meta-flutter Public

<> Code Issues 33 Pull requests 3 Discussions Actions Projects Wiki Security

f8205b8 12 Branches 7 Tags

Go to file Code

jwinarske sdbus-cpp-examples recipe f8205b8 · last week 1,122 Commits

.github/workflows	el sobrante	last week
classes	all: export xdg_config_dir for depot tools	last year
conf	common.inc: fix DeprecationWarning from regexp	last week
dynamic-layers/clang-layer	el sobrante	last week
lib	'Flutter SDK 3.32.5	last week
meta-flutter-apps	el sobrante	last week
recipes-devtools	sdbus-cpp-examples recipe	last week

# meta-flutter/workspace-automation



The workspace-automation tool was created to help setup your local development environment.

Supported systems:

- Ubuntu 22.x (x86\_64)
- Fedora 37 (x86\_64)
- Mac M1/M2 (arm64)

We will focus here on Ubuntu 22.x for this talk ...

meta-flutter / workspace-automation Public

<> Code Issues 3 Pull requests 2 Actions Projects Security Insights

v2.0 12 Branches 1 Tag Go to file Code

jwinarske Merge pull request #149 from meta-flutter/jw/dart-analytics 43e6532 · 2 days ago 373 Commits

.github/workflows	Remove fedora-40 from fedora CI matrix	last month
configs	flutter-engine	4 days ago
patches	flutter-engine	4 days ago
.gitignore	Windows ARM64 support	7 months ago
CHANGELOG.md	CI matrix (#48)	10 months ago
LICENSE	Configs	2 years ago
README.md	Added ubuntu-legacy (20.04) to the build matrix	5 months ago



MAY 13 – 14, 2026

TOKYO, JAPAN

# Setting up your dev environment



# We will focus on a certain combination ...



- meta-flutter
  - revision `b6e338a8b1fc9f8907f2c9a167c972a1907ff2ea`
  - SDK 3.38.3 (this needs to match with the AGL branch used)
- workspace-automation
  - revision `ef660bd3c3f8da2da2139608336aa3a33f7fc2f4`
- AGL (unagi branch)
  - AGL-repo revision
- If you clone AGL-repo, there will be a matching workspace-automation in `./external/workspace-automation`

# for reference ... '--remote' feature



Use in-tree version in `./external/workspace-automation/`

Call:

```
./flutter_workspace.py \  
  --remote https://gerrit.automotivelinux.org/gerrit/AGL/workspace-config-agl \  
  --enable agl-ivi-demo-flutter-qemu-unagi
```

... wait ...

```
source setup_env.sh # all the magic ;)
```

```
[run-agl-ivi-demo-flutter-qemu]
```

```
cd <path_to_your_flutter_project>
```

```
flutter run [-d agl-qemu-unagi]
```

# 4 Example cases next

- Basic Setup (no AGL, 'Linux Desktop')
- Setup with AGL + Qemu
- AGL + QEMU + VScode
- AGL + Hardware (raspberrypi5) + VScode

# Basic Setup (no AGL, 'Linux Desktop')



- Call in new terminal:
  - `source setup_env.sh`
  - `cd <path_to_your_flutter_project>`
  - `flutter run -d linux`
- This will compile and run your flutter application on the Linux desktop (aka Ubuntu 22.x in our case).
- You can develop, debug, run all on your PC.
- No IDE
- No specifics of the target environment!

# flutter run -d linux

```
user@user-VirtualBox:~/user/git/workspace_automation_desktop/app/samples/simplistic_calculator$ flutter run -d linux
```

```
Resolving dependencies... (1.4s)
```

```
Downloading packages...
```

```
+ analysis_defaults 0.0.0 from path ../analysis_defaults
```

```
+ async 2.11.0 (2.13.0 available)
```

```
+ auto_size_text 3.0.0
```

```
+ boolean_selector 2.1.1 (2.1.2 available)
```

```
+ characters 1.3.0 (1.4.0 available)
```

```
+ clock 1.1.1 (1.1.2 available)
```

```
+ collection 1.19.0 (1.19.1 available)
```

```
+ cupertino_icons 1.0.8
```

```
+ fake_async 1.3.1 (1.3.3 available)
```

```
+ fluent_ui 4.11.1
```

```
+ fluentui_system_icons 1.1.273
```

```
+ flutter 0.0.0 from sdk flutter
```

```
+ flutter_layout_grid 2.0.7
```

```
+ flutter_lints 5.0.0
```

```
+ flutter_localizations 0.0.0 from sdk flutter
```

```
+ flutter_riverpod 2.6.1
```

```
+ flutter_test 0.0.0 from sdk flutter
```

```
+ intl 0.19.0 (0.20.2 available)
```

```
+ leak_tracker 10.0.7 (10.0.9 available)
```

```
+ leak_tracker_flutter_testing 3.0.8 (3.0.9 available)
```

```
+ leak_tracker_testing 3.0.1
```

```
+ lints 5.1.1
```

```
+ matcher 0.12.16+1 (0.12.17 available)
```

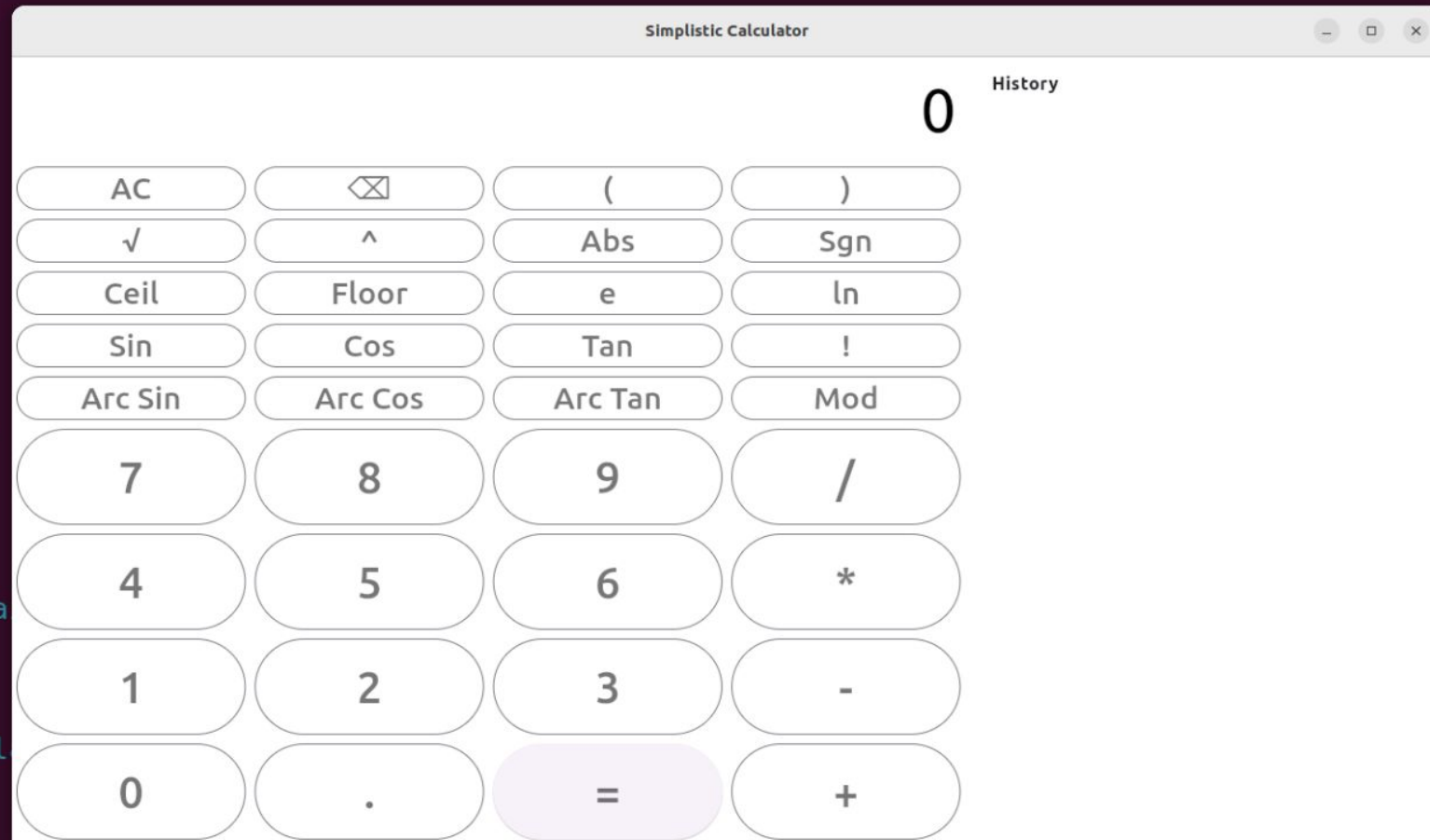
```
+ material_color_utilities 0.11.1 (0.12.0 available)
```

```
+ math_expressions 2.6.0 (2.7.0 available)
```

```
+ meta 1.15.0 (1.16.0 available)
```

```
+ path 1.9.0 (1.9.1 available)
```

```
+ quiver 3.2.2
```



# Pros and cons

- + simple and straightforward setup
- + native performance running the app
  
- no target environment specifics / apis available
- integration to target environment necessary later on
- no big difference to plain flutter development environment

# Setup with AGL unagi + QEMU

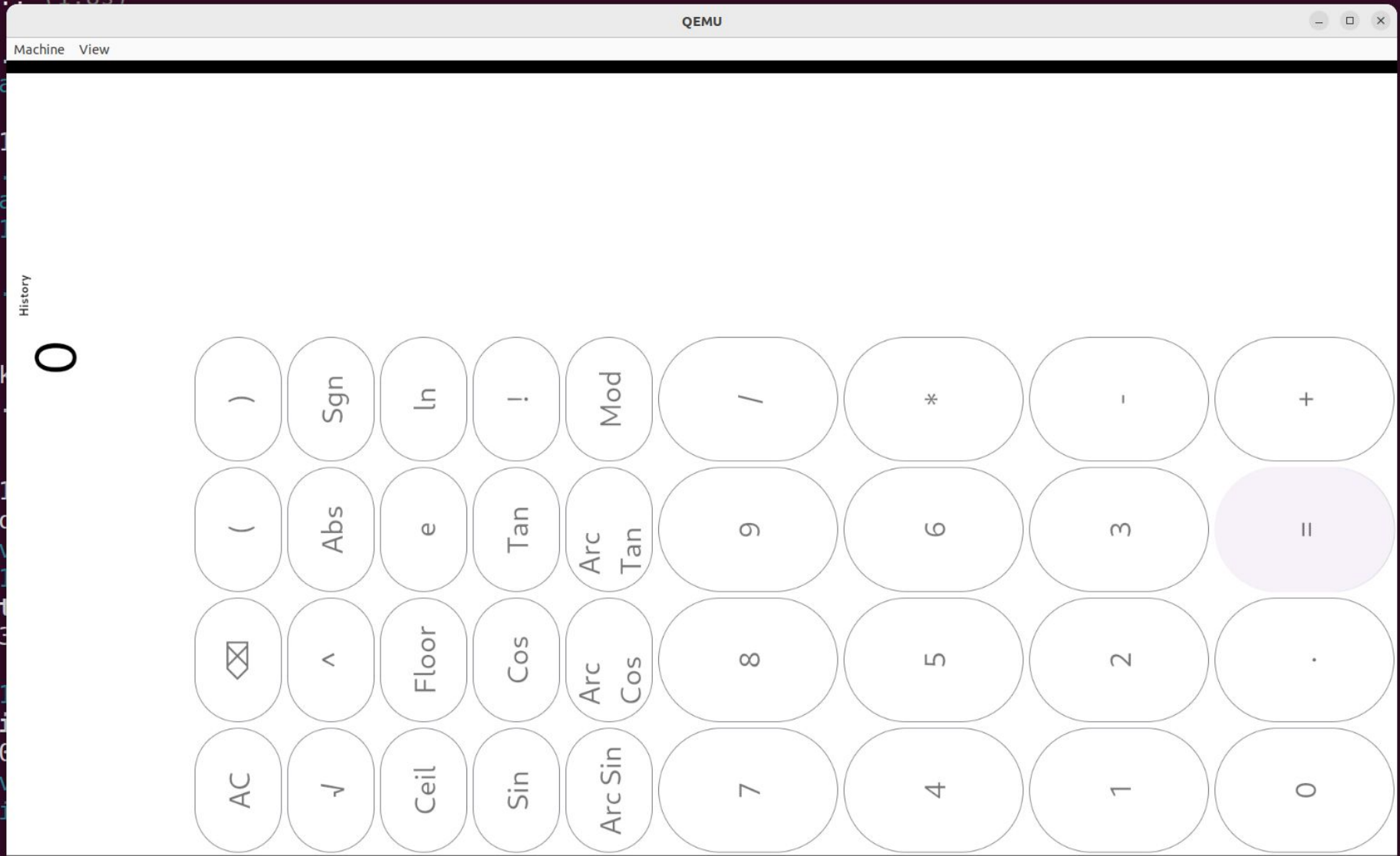


- Call in new terminal:
  - `source setup_env.sh`
  - **`run-agl-ivi-demo-flutter-qemu-unagi`**
  - `cd <path_to_your_flutter_project>`
    - `flutter doctor -v`  
should show: "AGL unagi-latest QEMU Image (mobile)"
  - `flutter run -d agl-qemu-unagi`
- This will compile and run your flutter application for use within a VM that is running in the background.
- You can develop, debug, run for the target environment.
- No IDE, yet.

# flutter run -d agl-qemu-unagi

```
user@user-VirtualBox:~/user/git/workspace_automation/app/samples/simplistic_calculator$ flutter run -d agl-qemu-salmon
```

```
Resolving dependencies... (1.8s)  
Downloading packages...  
+ analysis_defaults 0.0.0  
+ async 2.11.0 (2.13.0 available)  
+ auto_size_text 3.0.0  
+ boolean_selector 2.1.1  
+ characters 1.3.0 (1.4.0 available)  
+ clock 1.1.1 (1.1.2 available)  
+ collection 1.19.0 (1.19.1 available)  
+ cupertino_icons 1.0.8  
+ fake_async 1.3.1 (1.3.2 available)  
+ fluent_ui 4.11.1  
+ fluentui_system_icons 1.1.0  
+ flutter 0.0.0 from sdk  
+ flutter_layout_grid 2.0.0  
+ flutter_lints 5.0.0  
+ flutter_localizations 0.0.0 from sdk  
+ flutter_riverpod 2.6.1  
+ flutter_test 0.0.0 from sdk  
+ intl 0.19.0 (0.20.2 available)  
+ leak_tracker 10.0.7 (10.0.8 available)  
+ leak_tracker_flutter_testing 2.0.0  
+ leak_tracker_testing 3.0.1  
+ lints 5.1.1  
+ matcher 0.12.16+1 (0.12.17 available)  
+ material_color_utilities 0.1.0  
+ math_expressions 2.6.0  
+ meta 1.15.0 (1.16.0 available)  
+ path 1.9.0 (1.9.1 available)  
+ quiver 3.2.2
```



# Pros and cons

- + simple and straightforward setup
- + target environment specifics / apis
- + no separate integration step later on needed
  
- requires emulated target VM (qemu), thus slower

# AGL unagi + QEMU + VScode



- Call in new terminal:
  - `source setup_env.sh`
  - `run-agl-ivi-demo-flutter-qemu-unagi`
  - `cd <path_to_your_flutter_project>`
    - `flutter doctor -v`  
should show: "AGL unagi-latest QEMU Image (mobile)"
  - `code .`
    - select run config under flutter and choose "agl-unagi latest"
- This will compile and run your flutter application for use within a VM that is running in the background.
- You can develop, debug, run for the target environment.
- within IDE ;)

# vscode - run/debug w/ agl-qemu-unagi



The screenshot shows the Visual Studio Code interface with the following components:

- Run and Debug sidebar:** Shows the selected configuration as `simplistic_calculator (Flutter AGL salmon-latest QEMU Image)`. The `VARIABLES` section is currently empty.
- Code Editor:** Displays the `main.dart` file with the following code:

```
1 // Copyright 2014 The Flutter Authors. All rights reserved.
2 // Use of this source code is governed by a BSD-style license that can be
3 // found in the LICENSE file.
4
5 import 'dart:ui' as ui;
6
7 import 'package:flutter/material.dart';
8 import 'package:flutter/services.dart';
9 import 'package:flutter/widgets.dart';
10 import 'package:math_expressions/math_expressions.dart';
11 import 'package:math_expressions/math_expressions.dart';
12 import 'package:math_expressions/math_expressions.dart';
13 import 'package:math_expressions/math_expressions.dart';
14
15 void main() {
16   runApp(MyApp());
17 }
18
19 class MyApp extends StatelessWidget {
20   @override
21   Widget build(BuildContext context) {
22     return MaterialApp(
23       title: 'Simplistic Calculator',
24       theme: ThemeData(
25         primaryColor: Colors.purple,
26       ),
27       home: Calculator(),
28     );
29   }
30 }
31
32 @immutable
33 class CalculatorState {
34   const CalculatorState({
35     required this.buffer,
36     required this.calcHistory,
```
- QEMU Window:** Displays a calculator application with a white background and a grid of buttons. The display shows the number `0`. The buttons include `AC`, `√`, `Ceil`, `Sin`, `Arc Sin`, `7`, `4`, `1`, `0`, `⊗`, `^`, `Floor`, `Cos`, `Arc Cos`, `8`, `5`, `2`, `.`, `(`, `Abs`, `e`, `Tan`, `Arc Tan`, `9`, `6`, `3`, `=`, `)`, `Sgn`, `ln`, `!`, `Mod`, `/`, `*`, `-`, `+`.

# Pros and cons

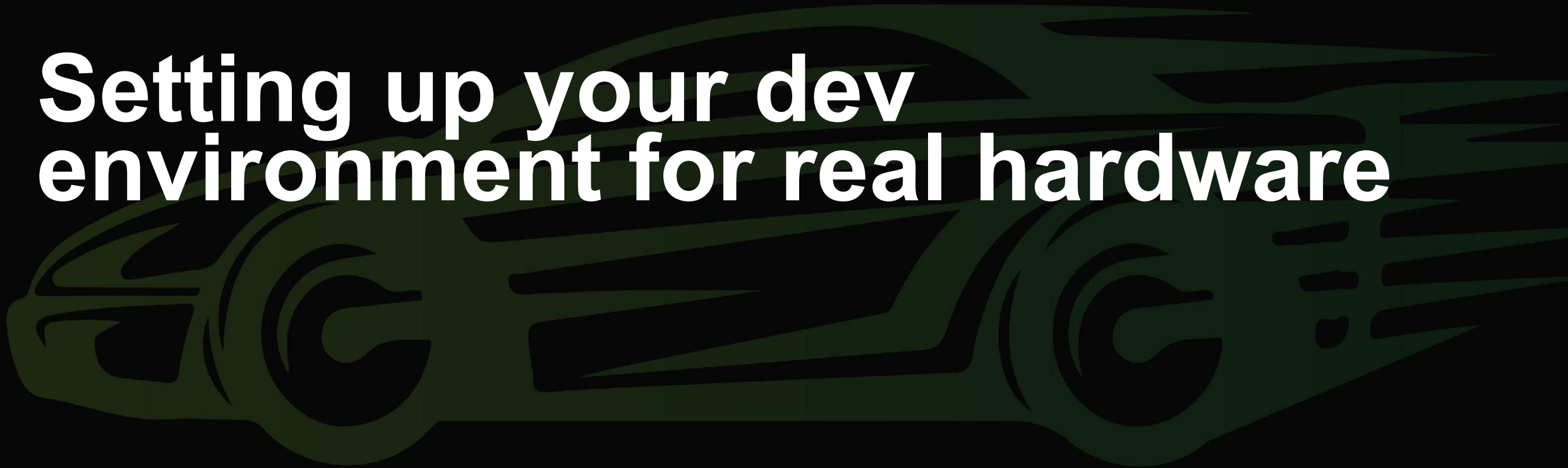
- + simple and straightforward setup
- + target environment specifics / apis
- + no separate integration step later on needed
- + IDE integration
- requires emulated target VM (qemu), thus a bit slower



MAY 13 – 14, 2026

TOKYO, JAPAN

# Setting up your dev environment for real hardware



# Prerequisites



- **IP address**
  - The board needs to be connected to the local network and have an IP address (or at least WIFI).
- **password-less ssh**
  - need to setup **agl-driver** user for *password-less ssh access*
    - e.g. `ssh root@IP && passwd -d agl-driver`
    - **WARNING**: of course this is insecure !
  - or deploy ssh keys !

# Adding a custom device !!



To add a custom device, there is "`flutter custom-devices add`":

```
Please enter the id you want to device to have. Must contain only alphanumeric or underscore characters.  
pi5-agl-unagi  
Invalid input. Please enter id:  
agl_pi5_unagi  
Please enter the label of the device, which is a slightly more verbose name for the device.  
agl-unagi-latest-pi5  
SDK name and version (example: Raspberry Pi 4 Model B+)  
pi5 unagi latest  
Should the device be enabled? [Y/n] (empty for default)  
y  
Please enter the hostname or IPv4/v6 address of the device. (example: raspberrypi)  
192.168.2.165  
Please enter the username used for sshing to the remote device.  
agl-driver  
Please enter the command executed on the remote device for starting the app. "/tmp/${appName}" is the  
path to the asset bundle. (example: flutter run -t /tmp/${appName})  
flutter-auto -t /tmp/${appName}  
Should the device use port forwarding? Using port forwarding is the default because it works in all  
cases, however if your remote device has a static IP address and you have a way of specifying the  
"-s-service-host=<ip>" engine option, you might prefer not using port forwarding. [Y/n]  
n  
Enter the command executed on the remote device for taking a screenshot. (example: fbgrab  
/tmp/screenshot.png && cat /tmp/screenshot.png | base64 | tr -d '\n\t', empty for no screenshotting  
support)  
Would you like to add the custom device to the config now? [Y/n] (empty for default)  
y
```

**NOT ENOUGH**

# Manual edits ... :(



The automated configuration is not perfect, yet.

We need to edit `.config/flutter/custom_devices.json` manually ...

# Adding a custom device - required !!



Edit .config/flutter/custom\_devices.json to add section

```
{
  "id": "agl-pi5-unagi",
  "label": "AGL unagi rpi5",
  "sdkNameAndVersion": "pi5-unagi latest",
  "platform": "linux-arm64",
  "enabled": true,
  "ping": [ "ping", "-c 1", "10.42.42.52" ],
  "postBuild": [
    "bash",
    "-c",
    "ssh -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -o BatchMode=yes -t agl-driver@10.42.42.52 systemctl stop
flutter-ics-homescreen && ssh -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -o BatchMode=yes -t root@10.42.42.52 passwd -d agl-driver"
  ],
  "install": [
    "bash",
    "-c",
    "ssh -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -o BatchMode=yes -t agl-driver@10.42.42.52 mkdir -p
/tmp/${appName}/data/flutter_assets /tmp/${appName}/lib && ssh -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -o BatchMode=yes -t
agl-driver@10.42.42.52 ln -sf /usr/share/flutter/*/debug/data/icudt.dat /tmp/${appName}/data/icudt.dat && ssh -o StrictHostKeyChecking=no -o
UserKnownHostsFile=/dev/null -o BatchMode=yes -t agl-driver@10.42.42.52 ln -sf /usr/share/flutter/*/debug/lib/libflutter_engine.so
/tmp/${appName}/lib/libflutter_engine.so && ssh -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null
/home/user/user/git/workspace_automation/.config/flutter/custom_devices.json -i vi-demo-flutter-qemu/config.toml agl-driver@10.42.42.52:/tmp/${appName}/ && scp -o
StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -r {local}/lib/* agl-driver@10.42.42.52:/tmp/${appName}/data/flutter_assets/"
  ],
  "uninstall": [
    "bash",
    "-c",
    "ssh -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -o BatchMode=yes -t agl-driver@10.42.42.52 rm -rf /tmp/${appName}"
  ],
  "runDebug": [
    "bash",
    "-c",
    "ssh -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -o BatchMode=yes agl-driver@10.42.42.52 flutter-auto -b /tmp/${appName}"
  ],
  "forwardPort": null,
  "forwardPortSuccessRegex": null,
  "screenshot": null
}
```

**ERROR PRONE!**

# Use workspace\_automation template



## Instead:

```
./flutter_workspace.py \  
--remote https://gerrit.automotivelinux.org/gerrit/AGL/workspace-config-agl \  
--enable agl-ivi-demo-flutter-pi5-unagi
```

- source setup\_env.sh
- cd <path\_to\_your\_flutter\_project>
  - flutter doctor -v  
should show: "AGL unagi-latest pi5 (mobile)"

```
[✓] Connected device (3 available) [64ms]  
• Linux (desktop) • linux • linux-x64 • Ubuntu 22.04.5 LTS 6.8.0-111-generic  
• Toyota homescreen (mobile) • desktop-homescreen • linux-x64 • homescreen x86_64  
• AGL unagi-latest pi5 (mobile) • agl-pi5-unagi • linux-arm64 • pi5-unagi latest
```

- flutter run -d agl-pi5-unagi

# Template:

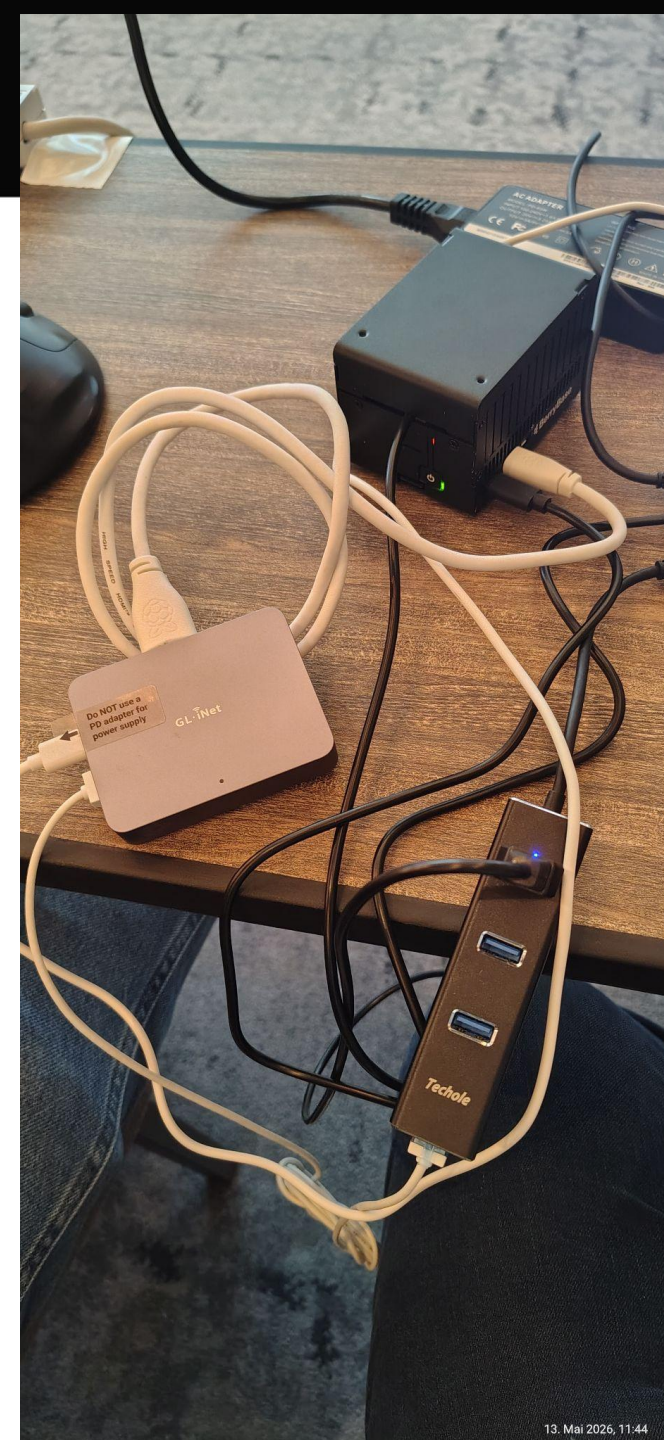


```
{
  "id": "agl-ivi-demo-flutter-pi5-unagi",
  "flutter_version": "3.38.3",
  "load": false,
  "supported_archs": [
    "aarch64",
    "arm64",
    "x86_64"
  ],
  "supported_host_types": [
    "darwin",
    "fedora",
    "ubuntu"
  ],
  "type": "remote",
  "env": {
    "DEVICE_HOSTNAME": "raspberrypi5.local",
    "COMPOSITOR_USER": "agl-driver",
    "SERVICE_NAME": "flutter-ics-homescreen",
    "SCP_PREFIX": "scp -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null",
    "SSH_PREFIX": "ssh -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -o BatchMode=yes",
    "PING_CMD": "ping -c 1 -w 400 ${DEVICE_HOSTNAME}",
    "STOP_SERVICE": "${SSH_PREFIX} -t root@${DEVICE_HOSTNAME} systemctl stop ${SERVICE_NAME}",
    "FLUTTER_AUTO_EXE": "flutter-auto -b /tmp/${appName}",
    "FLUTTER_DEBUG_PORT": "1234",
    "FLUTTER_OBSERVATORY_HOST": "raspberrypi5.local",
    "RELEASE_NAME": "unagi",
    "RELEASE_VERSION": "latest",
    "DELETE_USER_PWD": "${SSH_PREFIX} -t root@${DEVICE_HOSTNAME} passwd -d ${COMPOSITOR_USER}",
    "DELETE_APP_FOLDER": "${SSH_PREFIX} -t ${COMPOSITOR_USER}@${DEVICE_HOSTNAME} rm -rf /tmp/${appName}",
    "CREATE_BUNDLE_FOLDERS": "${SSH_PREFIX} -t ${COMPOSITOR_USER}@${DEVICE_HOSTNAME} mkdir -p /tmp/${appName}/data/flutter_assets /tmp/${appName}/lib",
    "CREATE_BUNDLE_ICUDTL_SYMLINK": "${SSH_PREFIX} -t ${COMPOSITOR_USER}@${DEVICE_HOSTNAME} ln -sf /usr/share/flutter/*/debug/data/icudtl.dat /tmp/${appName}/data/icudtl.dat",
    "CREATE_BUNDLE_ENGINE_SYMLINK": "${SSH_PREFIX} -t ${COMPOSITOR_USER}@${DEVICE_HOSTNAME} ln -sf /usr/share/flutter/*/debug/lib/libflutter_engine.so /tmp/${appName}/lib/libflutter_engine.so",
    "CREATE_BUNDLE_FOLDER": "${CREATE_BUNDLE_FOLDERS} && ${CREATE_BUNDLE_ICUDTL_SYMLINK} && ${CREATE_BUNDLE_ENGINE_SYMLINK}",
    "COPY_PLATFORM_ID_CONFIG_TO_DEVICE": "${SCP_PREFIX} ${PLATFORM_ID_DIR}/config.toml ${COMPOSITOR_USER}@${DEVICE_HOSTNAME}:/tmp/${appName}/",
    "COPY_ASSETS_TO_DEVICE": "${SCP_PREFIX} -r ${localPath}/* ${COMPOSITOR_USER}@${DEVICE_HOSTNAME}:/tmp/${appName}/data/flutter_assets/"
  },
  "runtime": {
    "config": {
      "global": {
        "cursor_theme": "DMZ-White",
        "app_id": "agl-pi5-unagi"
      },
      "view": {
        "window_type": "BG",
        "width": 1080,
        "height": 1920,
        "fullscreen": true
      },
      "window_activation_area": {
        "x": 0,
        "y": 0,
        "width": 1080,
        "height": 1920
      }
    }
  },
  "pre-requisites": {
    "arm64": {
      "darwin": {
        "cmds": [
          "bash -c \"arch -arm64 brew install xz qemu\""
        ]
      },
      "x86_64": {
        "darwin": {
          "cmds": [
            "bash -c \"arch -arm64 brew install xz qemu\""
          ]
        }
      }
    },
    "overwrite-existing": true,
    "custom-device": {
      "id": "agl-pi5-${RELEASE_NAME}",
      "label": "AGL ${RELEASE_NAME}-${RELEASE_VERSION} pi5",
      "sdkNameAndVersion": "pi5-${RELEASE_NAME} ${RELEASE_VERSION}",
      "platform": "linux-arm64",
      "enabled": true,
      "ping": "bash -c \"${PING_CMD}\"",
      "pingSuccessRegex": "1 received",
      "postBuild": "bash -c \"${STOP_SERVICE} && ${DELETE_USER_PWD}\"",
      "install": "bash -c \"${CREATE_BUNDLE_FOLDER} && ${COPY_PLATFORM_ID_CONFIG_TO_DEVICE} && ${COPY_ASSETS_TO_DEVICE}\"",
      "uninstall": "bash -c \"${DELETE_APP_FOLDER}\"",
      "runDebug": "bash -c \"${SSH_PREFIX} ${COMPOSITOR_USER}@${DEVICE_HOSTNAME} ${FLUTTER_AUTO_EXE}\"",
      "forwardPort": null,
      "forwardPortSuccessRegex": null,
      "screenshot": null
    }
  }
}
```

# Executing on the target ...

PI5 , USB gadget enabled for cdc-network

HDMI KVM device for capturing output + mouse input



# Executing on the target ...



DEMO:

```
cd flutter/examples/image_list/  
flutter run -d agl-pi5-unagi
```

# Summary



To enable a local hardware device with the meta-flutter workspace\_automation, we need to create/use a template file!

It contains the necessary abstractions to enable running your application on the device.

We do either hardcode an IP address or a hostname.

Todo: enable debug port connection

PATCHES FOR MORE DEVICES TEMPLATES WELCOME !!



MAY 13 – 14, 2026

TOKYO, JAPAN

Q/A

