Flutter for Embedded Systems

05/27/2021

Hidenori Matsubayashi

Hidenori.Matsubayashi@sony.com
HidenoriMatsubayashi

R&D Center, Sony Group Corporation

Copyright 2021 Sony Group Corporation
Announcing Flutter at Google I/O 2021

Google I/O 2021 keynote: https://youtu.be/a5S3D0s7HeE?t=1778
Flutter for embedded use - Sony

More than a Google project
While Google continues to be the primary contributor to the Flutter project, we’re delighted to see the growth of the broader ecosystem around Flutter.

One area of particular growth over recent months has been the broadening of Flutter to an ever-growing number of platforms and operating systems. At Flutter Engage, we announced that Toyota is bringing Flutter to their next-generation vehicle instrument systems. And last month, Canonical shipped their first release of Ubuntu with integrated support for Flutter, with Snap integration and support for Wayland.

Two new partners demonstrate this ever-growing ecosystem. Samsung is porting Flutter to Tizen, with an open source repository that others can also contribute to. And Sony is leading the effort to deliver a solution for embedded Linux.

Announcing Flutter 2.2 at Google I/O 2021 | by Tim Sneath | Flutter | May, 2021 | Medium
Flutter for embedded use - Samsung

More than a Google project
While Google continues to be the primary contributor to the Flutter project, we're delighted to see the growth of the broader ecosystem around Flutter.

One area of particular growth over recent months has been the broadening of Flutter to an ever-growing number of platforms and operating systems. At Flutter Engage, we announced that Toyota is bringing Flutter to their next-generation vehicle infotainment systems. And last month, Canonical shipped their first release of Ubuntu with integrated support for Flutter, with Snap integration and support for Wayland.

Two new partners demonstrate this ever-growing ecosystem. Samsung is porting Flutter to Tizen, with an open source repository that others can also contribute to. And Sony is leading the effort to deliver a solution for embedded Linux.

Announcing Flutter 2.2 at Google I/O 2021 | by Tim Sneath | Flutter | May, 2021 | Medium

Tizen for Flutter:
https://github.com/flutter-tizen
Flutter for embedded use - Toyota

More than a Google project
While Google continues to be the primary contributor to the Flutter project, we’re delighted to see the growth of the broader ecosystem around Flutter.

AGL (Toyota’s in-vehicle use):
Keynote (Flutter Engage) – YouTube

One area of particular growth over recent months has been the broadening of Flutter to an ever growing number of platforms and operating systems. At Flutter Engage, we announced that [Toyota is bringing Flutter to their next generation vehicle infotainment systems](https://announcing-flutter-2-2-at-google-i-o-2021-by-tim-sneath-flutter-may-2021-medium). And last month, Canonical shipped their first release of [Ubuntu with integrated support for Flutter](https://announcing-flutter-2-2-at-google-i-o-2021-by-tim-sneath-flutter-may-2021-medium), with Snap integration and support for Wayland.

Two new partners demonstrate this ever-growing ecosystem. [Samsung is porting Flutter to Tizen](https://announcing-flutter-2-2-at-google-i-o-2021-by-tim-sneath-flutter-may-2021-medium), with an open source repository that others can also contribute to. And [Sony is leading the effort to deliver a solution for embedded Linux](https://announcing-flutter-2-2-at-google-i-o-2021-by-tim-sneath-flutter-may-2021-medium).
Agenda

1. **Overview of modern Linux graphics**
   - Window Manager: X11, Wayland
   - Graphics API: Open GL ES, EGL
   - Framebuffer API: GBM, EGLStream
   - Kernel module: DRM, KMS

2. **Flutter and its architectural overview**
   - Flutter overview
   - Flutter internal architecture
   - Arm64 for Flutter Linux support
   - Bindings to native code
   etc.

3. **Introduction to “Flutter for embedded Linux”**
   - Overview & Objective
   - Features
   - Future works
   etc.
Agenda

1. Overview of modern Linux graphics
   - Window Manager: X11, Wayland
   - Graphics API: Open GL ES, EGL
   - Framebuffer API: GBM, EGLStream
   - Kernel module: DRM, KMS

2. Flutter and its architectural overview
   - Flutter overview
   - Flutter internal architecture
   - Arm64 for Flutter Linux support
   - Bindings to native code
   etc.

3. Introduction to “Flutter for embedded Linux”
   - Overview & Objective
   - Features
   - Future works
   etc.
Overview of modern Linux graphics stack

User application: app1, app2, ...

UI toolkit: GTK, Qt, SDL, WebKit, Flutter, Unity, ...

Window Manager: X11, Wayland

Mesa / Vendor library:
- OpenGL
- OpenGL ES
- EGL
- EGLStream
- GBM
- libdrm_*
- libdrm

Input event:
- libinput
- libudev
- libEvdev

Linux Kernel:
- DRM
- KMS
- evdev

H/W:
- CPU (x64, Arm64, RISCV, etc)
- GPU
- Display Controller
<table>
<thead>
<tr>
<th>Category</th>
<th>Name</th>
<th>Software License</th>
<th>Main Maintainer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web-based</td>
<td>Electron</td>
<td>MIT License</td>
<td>GitHub</td>
</tr>
<tr>
<td></td>
<td>NW.js</td>
<td>MIT License</td>
<td>Intel</td>
</tr>
<tr>
<td></td>
<td>Chromium</td>
<td>BSD 3-Clause</td>
<td>Google</td>
</tr>
<tr>
<td></td>
<td>WebKit</td>
<td>LGPL, BSD</td>
<td>Apple</td>
</tr>
<tr>
<td></td>
<td>Gecko</td>
<td>Mozilla Public License 2.0</td>
<td>Mozilla</td>
</tr>
<tr>
<td>Desktop-based</td>
<td>GTK</td>
<td>LGPL v2.1+</td>
<td>GNOME</td>
</tr>
<tr>
<td></td>
<td>Qt</td>
<td>Commercial License (or GPL/LGPL v3.0)</td>
<td>Qt Company</td>
</tr>
<tr>
<td></td>
<td>Mono</td>
<td>MIT, BSD, GPL etc.</td>
<td>Microsoft (Xamarin)</td>
</tr>
<tr>
<td></td>
<td>SDL</td>
<td>zlib License</td>
<td>- (OSS Community)</td>
</tr>
<tr>
<td></td>
<td>Kivy</td>
<td>MIT License</td>
<td>- (OSS Community)</td>
</tr>
<tr>
<td></td>
<td>wxWidgets</td>
<td>wxWindows License</td>
<td>- (OSS Community)</td>
</tr>
<tr>
<td></td>
<td>openFrameworks</td>
<td>MIT License</td>
<td>- (OSS Community)</td>
</tr>
<tr>
<td>Mobile-based</td>
<td>Flutter</td>
<td>BSD 3-Clause</td>
<td>Google</td>
</tr>
<tr>
<td></td>
<td>Kotlin Multiplatform</td>
<td>Apache License 2.0</td>
<td>Jet Brains</td>
</tr>
<tr>
<td>Game-based</td>
<td>Unreal Engine</td>
<td>Commercial License (depends on sales)</td>
<td>Epic Games</td>
</tr>
<tr>
<td></td>
<td>Unity</td>
<td>Commercial License (depends on sales)</td>
<td>Unity</td>
</tr>
</tbody>
</table>
Window Manager: X11, Wayland

- **Window Manager**
  
  Window manager is system software that controls the placement and appearance of windows within a windowing system.

- **X11 (X Window System)**
  - [https://www.x.org/wiki/](https://www.x.org/wiki/)
  - X Window System (X11, or simply X) is a windowing system for bitmap displays, common on Unix-like operating systems.

- **Wayland**
  - [https://wayland.freedesktop.org/](https://wayland.freedesktop.org/)
  - Next-generation window manager that will replace X11.
  - Wayland is a communication protocol that specifies the communication between a display server and its clients.

[Diagram of Window Manager architecture]

Window Manager

- **Compositor (Display Server)**
  - Window Management
  - Window Composition
  - Transit Animation
  - Layer management
  - Multiple Display

- **Input Manager**
  - Fetch Event
  - Dispatch Event
  - Virtual Event

- **Shell (System UI)**
  - Look & feel
  - Window Decoration
  - Window Layout
  - Context Menu
  - Menu bar
  - App Launcher
Wayland

- Ubuntu 21 enabled Wayland graphics by default
- Wayland is a lightweight than X11
  - Wayland just requires far fewer libraries than X11
- Board Support Package (BSP)
  - Many SoC vendors support Wayland (Weston)

<table>
<thead>
<tr>
<th>Vendor</th>
<th>SoC/Board/Platform</th>
<th>Window Manager</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>NXP</td>
<td>i.MX 8M</td>
<td>×</td>
<td>X11 isn’t supported after i.MX 6</td>
</tr>
<tr>
<td>Xilinx</td>
<td>Zynq</td>
<td>○</td>
<td>Peta Linux supports Wayland from 2019.2</td>
</tr>
<tr>
<td>NVIDIA</td>
<td>Jetson</td>
<td>○</td>
<td>-</td>
</tr>
<tr>
<td>Qualcomm</td>
<td>RB5</td>
<td>○</td>
<td>SoC: SD865</td>
</tr>
<tr>
<td>Raspberry Pi4</td>
<td>Broadcom BCM2711</td>
<td>○</td>
<td>-</td>
</tr>
</tbody>
</table>

- ○ ••• Official Support
- △ ••• Unofficial (3rd party)
- × ••• Not supported
Graphics library

- **OpenGL, OpenGL ES**
  OpenGL (Open Graphics Library) is a cross-language, cross-platform application API for rendering 2D and 3D vector graphics

- **EGL**
  An interface between Khronos rendering APIs (such as OpenGL, OpenGL ES or OpenVG) and the underlying native platform windowing system

- **Generic Buffer Management (GBM)**
  - An abstraction of the graphics driver specific buffer management APIs
  - Allocating buffers for graphics rendering

- **EGLStream**
  - EGLStream is a mechanism that efficiently transfers a sequence of image frames from one API to another
  - Nvidia only supports EGLStream

Mesa / Vendor library

- **OpenGL**
- **OpenGL ES**

- **EGL**
- **GBM**
- **EGLStream**

- **Libdrm**
- **Libdrm-driver**

- **Kernel**
  - **DRM**
  - **KMS**

- **NVIDIA supports (Not use GBM)**
Kernel module: DRM / KMS

See: https://dri.freedesktop.org/docs/drm/gpu/index.html

- **Direct Rendering Manager (DRM)**
  - One of Linux kernel module (/dev/dri/card0, etc.)
  - Draw graphics to frame buffer directly
  - Authentication

- **Kernel Mode Setting (KMS)**
  - Display settings such as resolution and color depth
1. **Overview of modern Linux graphics**
   - Window Manager: X11, Wayland
   - Graphics API: Open GL ES, EGL
   - Framebuffer API: GBM, EGLStream
   - Kernel module: DRM, KMS

2. **Flutter and its architectural overview**
   - Flutter overview
   - Flutter internal architecture
   - Arm64 for Flutter Linux support
   - Bindings to native code
   etc.

3. **Introduction to “Flutter for embedded Linux”**
   - Overview & Objective
   - Features
   - Future works
   etc.
What is Flutter?

Flutter is Google’s UI toolkit for building beautiful, natively compiled applications.

- Released in 2017
- Original rendering engine using Skia
- Programming language: Dart
- Supported platforms from a single source code
  - Mobile (Android/iOS)
  - Web
  - Desktops (Linux/Windows/macOS): β version

✔ Official version will come within 2021?
Flutter Roadmap (History)

Google’s partners

- iOS
- Android
- Web
- Windows
- macOS
- Linux
- Embedded

2018/12 Stable version

Desktops for Flutter has started as a stand-alone project since Feb 2018
[google/flutter-desktop-embedding](https://google/flutter-desktop-embedding)

2021/3 Beta version

It has merged into [flutter/engine](https://flutter/engine) in 2019 or 2020

2021/3 Stable version

?
Pros & Cons of Flutter

- **Pros**
  - Easy to create beautiful UI
  - Create a prototype quickly using Adobe XD, etc.
  - A lot of 3rd libraries (plugins), documents, and information
  - Native compiled applications, strongly debug function like hot-reload
  - Cross-platform
  - Package management system and test tools (Dart SDK)
  - Support of major IDEs like VS Code
  - Active community
  - Embedded use cases
    - Custom embedder API-layer for specific platforms
    - Flutter Engine requires just fewer library dependencies
    - Easy access to other native codes and hardware resources
    - BSD 3-Clause licensed software

- **Cons**
  - Need to study Dart
  - Platform-specific features like media players need to be implemented natively
  - Lighter than WebView, but might be heavier than native apps (We need to benchmark)
Apps made with Flutter

https://flutter.dev/showcase
Flutter Widget

- In Flutter, Widget is the UI component to declare and construct UI

- See: Widget catalog - Flutter
Famous Flutter apps examples

- [https://github.com/flutter/gallery](https://github.com/flutter/gallery)
  Flutter Gallery is an official resource and example app to help developers evaluate and use Flutter

- [https://github.com/gskinnerTeam/flokk](https://github.com/gskinnerTeam/flokk)

- [https://github.com/gskinnerTeam/flutter-folio](https://github.com/gskinnerTeam/flutter-folio)

- [https://gallery.flutter.dev/#/](https://gallery.flutter.dev/#/)

- [https://flutter.gskinner.com/flokk/](https://flutter.gskinner.com/flokk/)
Flutter plugin

- A lot of official and 3rd party libraries (Flutter plugins) on pub.dev
- Flutter SDK provides a package management function to make developers install Flutter plugins easily
- Plugins
  Flutter is just the UI toolkit. It means not includes platform native features like a media player. You need to implement it yourself or use 3rd party one
Dart programming language

- Dart is being developed by Google from 2011 to be used to build server and desktop applications. Development started with the goal of replacing JavaScript.

- Dart 1.0 release: Nov 2013
  - Latest stable version: 2.13.0

- Null Safety support

- Dart language specification is quite close to JavaScript, and it is said it is a language with very low learning cost for developers who are familiar with JavaScript, Java, or C++

- Dart compiler
  - Dart to JavaScript
  - Dart to native (machine code JIT and AOT)

- Binding to native code / IPC
  - dart:ffi (FFI stands for foreign function interface)
  - Unix Domain Socket in dart:io
  - 3rd party library: gRPC, Dbus etc.
DartPad

- [https://dartpad.dev/flutter](https://dartpad.dev/flutter)

- DartPad is a free, open-source online editor to help developers learn about Dart and Flutter
Hot reload

- Hot reload helps developers quickly and easily create UIs, and fix bugs.
- Developers can inject updated source code files into the running Flutter apps without relaunching the app.

Excerpt from [https://github.com/flutter/flutter](https://github.com/flutter/flutter)
Flutter repo and development workflow

1. git clone or download from flutter.dev

2. Automatic download artifacts as needed by Flutter SDK

3. Create Flutter apps
4. Build Flutter apps

Source code on GitHub

- Flutter Engine
- Flutter SDK

Google Infra Server

- Build
- Test
- Deploy
Quick start on Ubuntu Linux Hosts

# Install Flutter SDK
$ git clone https://github.com/flutter/flutter
$ sudo mv flutter /opt/
$ export PATH=$PATH:/opt/flutter/bin

# Install dependent packages
$ sudo apt install clang curl pkg-config ninja-build cmake libgtk-3-dev libblkid-dev liblzma-dev unzip

# Enable Flutter desktop for Linux
$ flutter config --enable-linux-desktop

# Run Flutter sample app
$ flutter create sample
$ cd sample
$ flutter run –d linux
Flutter sample app & Widget tree

```dart
class _MyHomePageState extends State<_MyHomePage> {
  int _counter = 0;

  void _incrementCounter() {
    setState(() {
      _counter++;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(widget.title),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            Text('You have pushed the button this many times:'),
            Text('${_counter}',
              style: Theme.of(context).textTheme.display1,
            ),
            floatingActionButton: FloatingActionButton(
              onPressed: _incrementCounter,
              tooltip: 'Increment',
              child: Icon(Icons.add),
            ),
          ],
        ),
      ),
    );
  }
}
```
Flutter architecture overview

Flutter user apps (Dart)
- Written in Dart language
- Create UI by using widgets
- Works on Dart-VM

Framework (Dart)
- Material
- Cupertino
- Widgets
- Rendering
- Animation
- Painting
- Gestures
- Foundation

Engine (C/C++)
- Service Protocol
- Composition
- Platform Channels
- Dart Isolate Setup
- Rendering
- System Events
- Dart VM Management
- Frame Scheduling
- Asset Resolution
- Frame Pipelining
- Text Layout

3rd-party OSS
- dart-vm
- Skia
- Txt
- ...
Embedder API

- [Link](https://github.com/flutter/flutter/wiki/Custom-Flutter-Engine-Embedders)

- Flutter desktops (Linux / macOS / Windows) use this API
  - Android and iOS aren’t using it. The API were created for Flutter desktops?

- Header file
  - [Link](https://github.com/flutter/engine/blob/master/shell/platform/embedder/embedder.h)

```c
FLUTTER_EXPORT
FlutterEngineResult FlutterEngineRun(size_t version,
    const FlutterRendererConfig* config,
    const FlutterProjectArgs* args,
    void* user_data,
    FLUTTER_API_SYMBOL(FlutterEngine) * engine_out);
```
Arm64 Linux support (One of our contributions to Flutter)

What’s new in Flutter 2.2 | Flutter (medium.com)

- Currently, Flutter supports desktop for Arm64 Linux hosts. Also, Flutter SDK works on Arm64 Linux hosts
- Flutter works on general Arm64 devices such as Raspberry Pi4 and Jetson Nano

It’s exciting to see the Flutter community bringing Flutter to places that the team at Google could never have imagined. Keep up the good work, HidemoriMatsubayashi!
Binding to native code in Flutter

- **Binding to native code**
  - Communication APIs are provided by Flutter
    - ✓ Method Channel
    - ✓ Event Channel
    - ✓ Basic Message Channel
  - dart:ffi
    - ✓ Foreign Function Interface for interoperability with C programming language

- **Inter-process communication (IPC)**
  - Unix domain sockets in dart:io
  - 3rd party library
    - ✓ A native Dart client implementation of D-Bus: https://github.com/canonical/dbus.dart
    - ✓ grpc-dart: https://github.com/grpc/grpc-dart
      ...

Communication APIs are provided by Flutter

Flutter user apps (Dart) → Framework (Dart) → Engine (C/C++) → Embedder (Java, C++, Objective-C) → Platform (Native)

- Dart code
- Method Channel
- Event Channel
- Basic Message Channel
- Binary Messenger
- Platform Channel
- Native code
Flutter plugin: Unix domain socket support in grpc-dart

- grpc-dart is the gRPC library in Dart implementation
- We have contributed to add Unix Domain Socket support to grpc-dart

Use case
- IPC between dart or other language: Unix domain socket
- External communication between the device and other devices: HTTP/2

https://pub.dev/packages/grpc
Agenda

1. Overview of modern Linux graphics
   - Window Manager: X11, Wayland
   - Graphics API: Open GL ES, EGL
   - Framebuffer API: GBM, EGLStream
   - Kernel module: DRM, KMS

2. Flutter and its architectural overview
   - Flutter overview
   - Flutter internal architecture
   - Arm64 for Flutter Linux support
   - Bindings to native code
   etc.

3. Introduction to “Flutter for embedded Linux”
   - Overview & Objective
   - Features
   - Future works
   etc.
Flutter for Embedded Linux

Open-source: https://github.com/sony/flutter-embedded-linux

- **Overview**
  - Porting Flutter to embedded Linux systems

- **Objective**
  - Use Flutter in embedded systems

- **Non-goal**
  - Not intended to replace the existing Flutter desktop for Linux in desktop use cases
  - Objective is just for Embedded Systems use cases
Features

- Flutter embedder optimized for embedded systems
- Both x64 architecture and Arm64 architecture support
- Lightweight than Flutter desktop for Linux (without X11 and GTK/GDK)
- Wayland, DRM (GBM or EGLStream) backends support
- Single full-screen or flexible-screen
- Keyboard, touch, mouse, clipboard support
- API compatibility with Flutter desktop for Windows
  - External texture plugin (texture composition in Flutter) for media player, etc.
  - Based on Flutter desktop for Windows
    Initially, it was created with full scratch, but we changed from the middle etc.
Objective of open-source

➢ We are looking for partners to develop together
  • It's difficult to cover all embedded specifications for Sony alone
  • Google Flutter team is closely us

➢ Contribution
  • Welcome all your contribution and feedbacks
    ✓ If you want to send a PR, you need to accept our CLA
    ✓ CLA is still under construction...

➢ Target the mainline
  • Propose and contribute this software to the mainline of Flutter Engine repo, which means we would like to add embedded systems support into Flutter for all embedded developers in the future
Why is this embedder necessary instead of “Flutter desktop for Linux”?

Requirements for embedded systems are not equal desktop one.

Example:

<table>
<thead>
<tr>
<th>No</th>
<th>Item</th>
<th>Requirement</th>
<th>Desktop</th>
<th>Embedded</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Widgets (GTK) to create a desktop-like UI</td>
<td>✔</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Window manager: X11</td>
<td>✔</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Window manager: Wayland</td>
<td>✔</td>
<td>✔ (partially)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Multi window</td>
<td>✔</td>
<td>✔ (partially)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Graphics composition in app</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Keyboard input</td>
<td>✔</td>
<td>✔ (partially)</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Touch input</td>
<td>✔ (partially)</td>
<td>✔ (partially)</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Limited memory / storage capacity</td>
<td>-</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Limited CPU power</td>
<td>-</td>
<td>✔</td>
<td></td>
</tr>
</tbody>
</table>
Why is this embedder necessary instead of “Flutter desktop for Linux”?

For example, desktop apps require a widget like menubar (Need GTK)

[Image of a desktop app interface with options like Color, Counter, Reset, Increment, Decrement]

https://github.com/google/flutter-desktop-embedding
Why is this embedder necessary instead of “Flutter desktop for Linux”?

- Flutter desktop for Linux uses X11 and GTK/GDK

- Source files strongly dependent on GDK
  - We don’t want to install libraries that are not used as functions as much as possible
  - As a side note, we initially thought about sharing the source code with the desktop version but concluded that it was difficult

- X11 and GTK require a lot of dependent libraries (includes GPL/LGPL v3 licensed software)
  - e.g. (on Ubuntu 18.04)
    - Xserver-xorg: [https://packages.ubuntu.com/bionic/xserver-xorg](https://packages.ubuntu.com/bionic/xserver-xorg)
    - libgtk-3: [https://packages.ubuntu.com/bionic/libgtk-3-0](https://packages.ubuntu.com/bionic/libgtk-3-0)
Flutter desktop for Linux

- Flutter user apps (Dart)
- Framework (Dart)
- Engine (C/C++)

Common source files

Embedder for Linux desktop

flutter/engine/tree/master/shell/platform/linux

- GTK
- GDK
- X11
- DRM

Linux platform

- OpenGL ES
- EGL
- Wayland
Flutter for Embedded Linux

Flutter user apps (Dart)

Framework (Dart)

Engine (C/C++)

Embedder for Embedded Linux (C++)
sony/flutter-embedded-linux

Wayland or directly to DRM

Based Flutter desktop for Windows

Linux platform

Common source files

Embedded for Embedded Linux

OpenGL ES

EGL
Flutter artifacts

Flutter user apps (Dart)

Framework (Dart)

Engine (C/C++)

Embedder for Embedded Linux (C++)

libapp.so, which is built by Flutter SDK (dart-sdk)

libflutter_engine.so, which is built by GN/Ninja

Linux platform

Executable file (binary). Currently, we support to self-build and cross-build using CMake
Build Flutter for embedded Linux using Yocto

See: https://github.com/sony/flutter-embedded-linux/tree/master/meta-flutter

$ git clone https://github.com/sony/flutter-embedded-linux.git

# Add meta-flutter layer into your conf/bblayers.conf
$ bitbake-layers add-layer ../flutter-embedded-linux/meta-flutter

# Build flutter-embedded-linux with Wayland backend
$ bitbake flutter-wayland-client
Build Flutter for embedded Linux using CMake

See: https://github.com/sony/flutter-embedded-linux/tree/master/doc

$ git clone https://github.com/sony/flutter-embedded-linux.git

$ mkdir build && cd build

# Self-build on x64 or arm64
$ cmake -DUSER_PROJECT_PATH=examples/flutter-wayland-client ..
$ cmake --build .

# Cross-build on x64 for arm64
$ cmake -DUSER_PROJECT_PATH= examples/flutter-wayland-client -DCMAKE_TOOLCHAIN_FILE=<toolchain-template-file> ..
$ cmake --build .
Future works

- Embedder tasks
  - Platform Views (Texture composition in Flutter embedder) support
  - Multi / Dual-screen support
  - Vsync support
  - Add compiler switch to disable input function (Keyboard, touch, mouse) etc.

- Flutter plugins
  - Audio / Video Player
  - WebView
  - Path provider etc.

- Flutter SDK
  - Add / contribute custom-devices support to build and debug using Flutter SDK
Thank you for your time