

Wayland Desktop Environment

MA35D1/MA35H0

江天文

2024/03/02

Joy of innovation
nuvoTon

Content

Introduction to Wayland

LVGL with Wayland

The skeleton minimal LVGL with Wayland

Qt Creator remote debugging with LVGL on Wayland

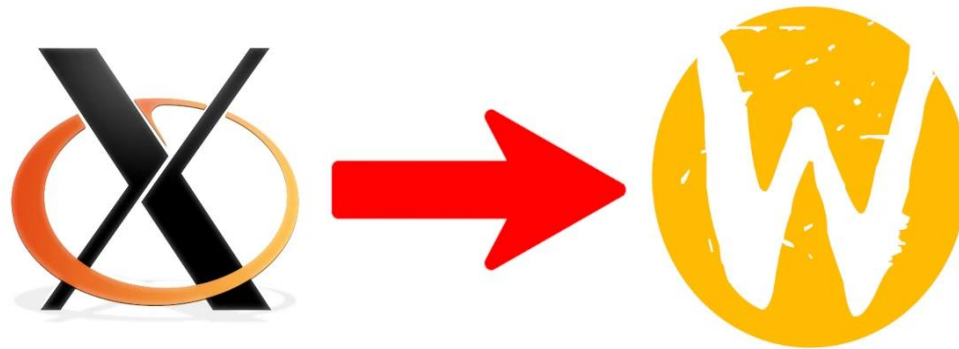
Qt5 with Wayland

Customizing Weston

Changing wallpapers

Adding launch icons

Setting idle lock time



Introduction to Wayland

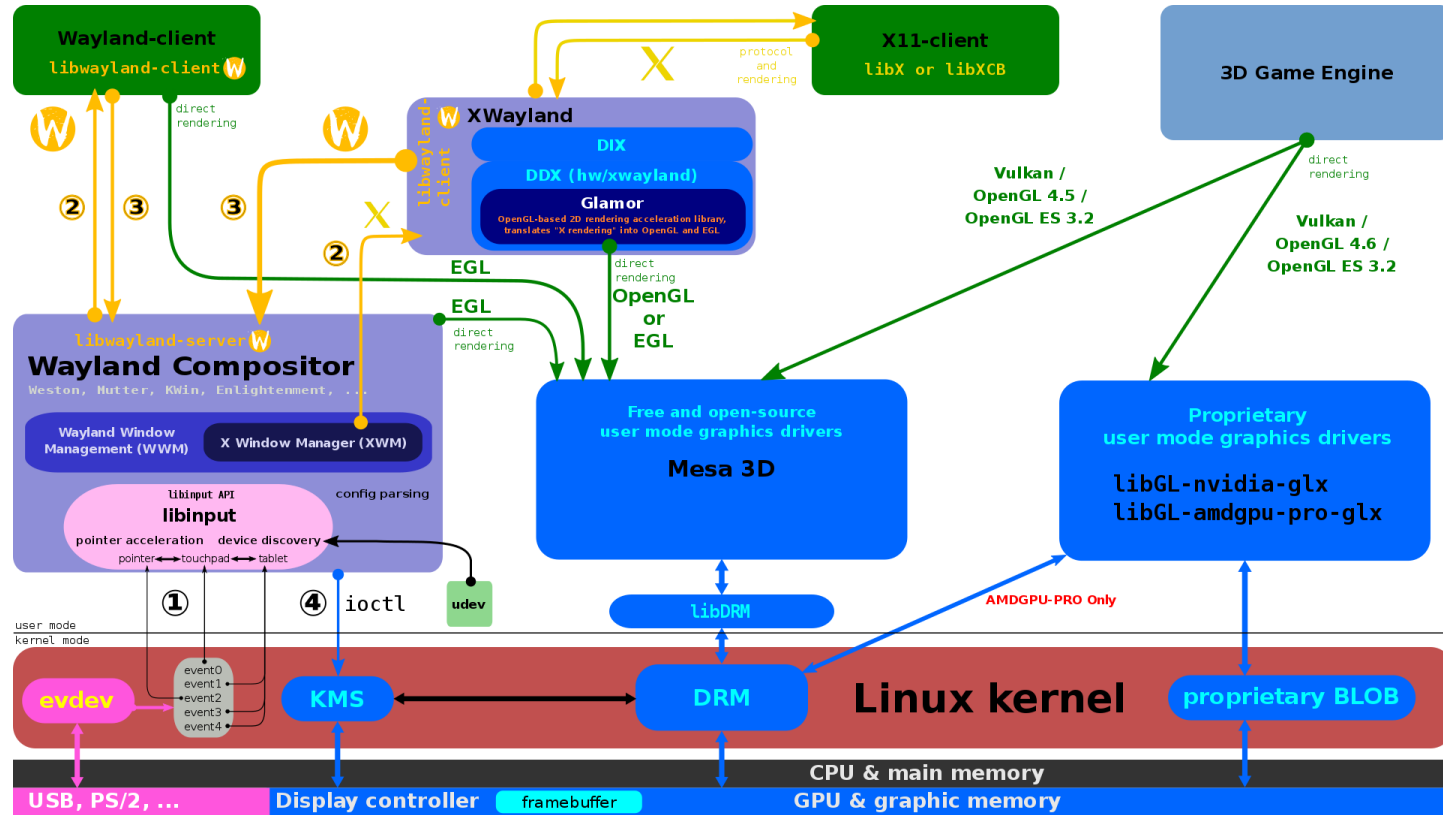
<https://wayland.freedesktop.org>

Wayland



Wayland is a **communication protocol** that specifies the communication between a **display server** and its client, as well as a C library implementation of that protocol. A display server using the Wayland protocol is called a **Wayland compositor**, because it additionally performs the task of a **compositing window manager**.
(Wikipedia)

Wayland architecture



Weston

Weston is the reference implementation of a Wayland compositor. When running on Linux, handling of the input hardware relies on **evdev**. Out of the box, Weston provides a very basic desktop, or a full-featured environment for non-desktop uses such as automotive, embedded, in-flight, industrial, kiosks, set-top boxes and TVs.

It contains a plug-in system of “shell” for common desktop features like docks and panels.

a plug-in Maynard for Weston



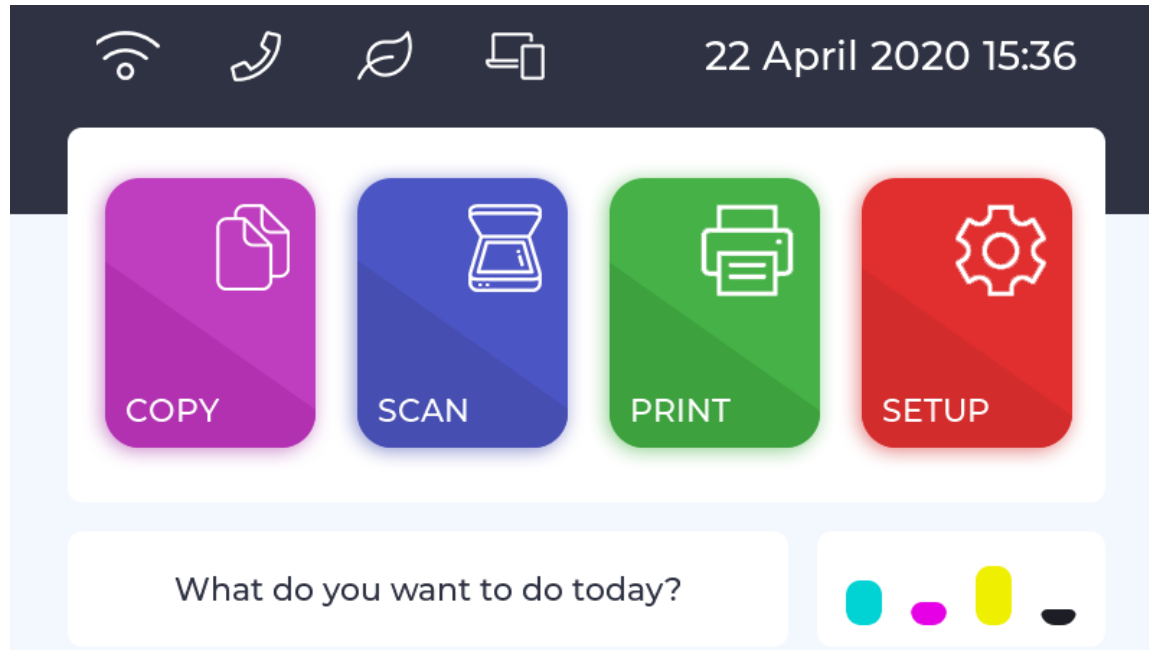
Extension protocols to the core protocol

■ XDG-Shell protocol

The traditional way to manipulate surface (maximize, minimize, fullscreen) is through Wayland client. XDG shell, on the contrary, is supposed to be provided by the compositor. The XDG (cross-desktop group) shell is a standard protocol extension for Wayland which describes the semantics for application windows.

■ IVI-Shell protocol

IVI-Shell is an extension to the Wayland core protocol, targeting in-vehicle infotainment (IVI) devices.



LVGL with Wayland

<https://lvgl.io>

Getting started with LVGL on Wayland

- Start from scratch by cloning the official Buildroot repository
\$ git clone https://github.com/OpenNuvoton/MA35D1_Buildroot.git
- Change directory to *the root of Buildroot* (`${BR2_DIR}`), and clone repository *ma35d1-portal* on branch *weston-10.0.5*.
\$ cd `${BR2_DIR}`
\$ git clone -b weston-10.0.5 <https://github.com/symfund/ma35d1-portal.git>
- Delete original packages, and replace with corresponding packages from *ma35d1-portal/package* on branch *weston-10.0.5*.
\$ cd `${BR2_DIR}/package`; rm -Rf cairo eudev libdrm libinput libpng
libxkbcommon mesa3d-headers mesa3d pixman wayland-protocols wayland-
utils wayland weston; cp -Rf ../ma35d1-portal/package/wayland wayland; ...

Configuring Buildroot to import packages

- Clone *buildroot-external* into the root of Buildroot

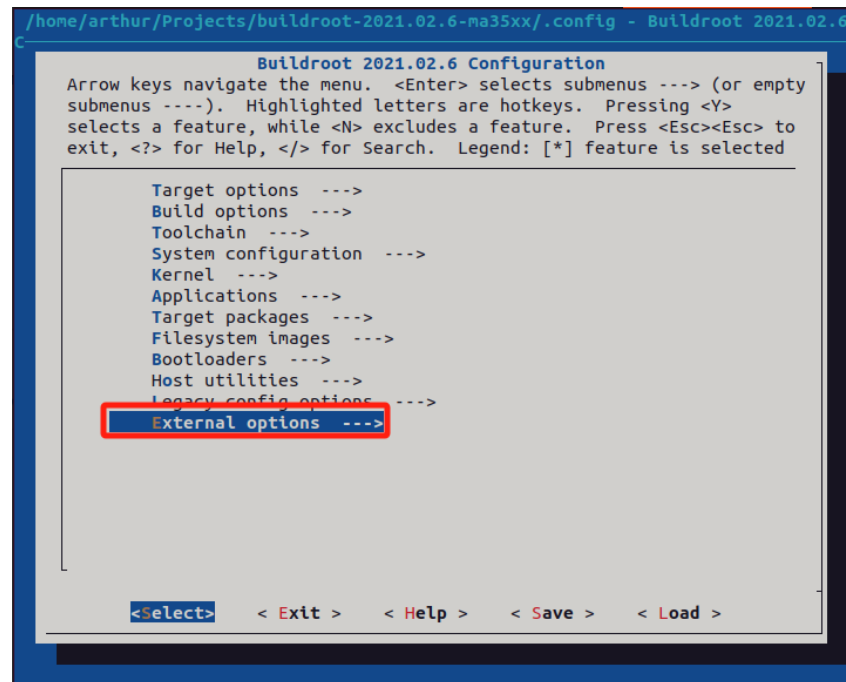
\$ git clone <https://github.com/symfund/buildroot-external.git>

- Load a specific board default configuration

\$ make numaker_som_ma35d16a81_defconfig

- Configure Buildroot

\$ make BR2_EXTERNAL=buildroot-external menuconfig



```
/home/arthur/Projects/buildroot-2021.02.6-ma35xx/.config - Buildroot 2021.02.6

Buildroot 2021.02.6 Configuration
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
submenus ----). Highlighted letters are hotkeys. Pressing <Y>
selects a feature, while <N> excludes a feature. Press <Esc><Esc> to
exit, <?> for Help, </> for Search. Legend: [*] feature is selected

Target options --->
Build options --->
Toolchain --->
System configuration --->
Kernel --->
Applications --->
Target packages --->
Filesystem images --->
Bootloaders --->
Host utilities --->
Legacy config options ---->
External options --->

<Select>  <Exit>  <Help>  <Save>  <Load>
```

Configuring Buildroot to manage /dev

- **/dev management** using *devtmpfs + eudev*

→ System configuration
/dev management

- () Static using device table
- () Dynamic using devtmpfs only
- () Dynamic using devtmpfs + mdev
- (X) Dynamic using devtmpfs + eudev

The screenshot shows the Buildroot configuration interface. At the top, a red box highlights the 'System configuration' option. Below it, the 'System configuration' menu is displayed. A red box highlights the option '/dev management (Dynamic using devtmpfs + eudev) --->'. The menu also shows other options like 'Root FS skeleton', 'System hostname', 'System banner', 'Passwords encoding', 'Exit system (BusyBox)', 'Path to the permission tables', 'support extended attributes in device tables', 'Use symlinks to /usr for /bin, /sbin and /lib', 'Enable root login with password', 'Root password', 'bin/sh (busybox' default shell)', 'Run a getty (login prompt) after boot', 'remount root filesystem read-write during boot', 'Network interface to configure through DHCP', 'Set the system's default PATH', 'Purge unwanted locales', and 'Locales to keep'. At the bottom, there are navigation buttons: '<Select>', '<Exit>', '<Help>', '<Save>', and '<Load>'.

```

Buildroot-2021.02.6-ma35xx/.config - Buildroot 2021.02.6
→ System configuration

System configuration
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
submenus ----). Highlighted letters are hotkeys. Pressing <Y>
selects a feature, while <N> excludes a feature. Press <Esc><Esc> to
exit, <?> for Help, </> for Search. Legend: [*] feature is selected

Root FS skeleton (default target skeleton) --->
(buildroot) System hostname
(Welcome to Buildroot) System banner
Passwords encoding (sha-256) --->
Exit system (BusyBox) --->
/dev management (Dynamic using devtmpfs + eudev) --->
(system/device_table.txt) Path to the permission tables
[ ] support extended attributes in device tables
[ ] Use symlinks to /usr for /bin, /sbin and /lib
[*] Enable root login with password
() Root password
/bin/sh (busybox' default shell) --->
[*] Run a getty (login prompt) after boot --->
[*] remount root filesystem read-write during boot
() Network interface to configure through DHCP
(/bin:/sbin:/usr/bin:/usr/sbin) Set the system's default PATH
[*] Purge unwanted locales
(C en_US) Locales to keep
k(+)

<Select>  <Exit>  <Help>  <Save>  <Load>
  
```

Configuring Buildroot to enable mesa3d

- Select *mesa3d*
 - Gallium swrast driver
 - OpenGL EGL
 - OpenGL ES

The screenshot shows the Buildroot configuration menu. At the top, a red box highlights the navigation path: "Target packages -> Graphic libraries and applications (graphic/text) -> mesa3d". Below this, a text box explains the navigation controls: "Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenu --->). Highlighted letters are hotkeys. Pressing <Y> selects a feature, while <N> excludes a feature. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] feature is selected [] feature is excluded".

The main menu lists various features under the heading "-- mesa3d". A red box highlights the selection of "Gallium swrast driver" with an asterisk [*]. Another red box highlights the selection of "OpenGL EGL" and "OpenGL ES" with asterisks [*].

At the bottom of the menu, there are navigation buttons: "<select>", "< Exit >", "< Help >", "< Save >", and "< Load >".

Configuring Buildroot to select free fonts

- Free fonts
 - mono fonts
 - sans fonts
 - serif fonts

```

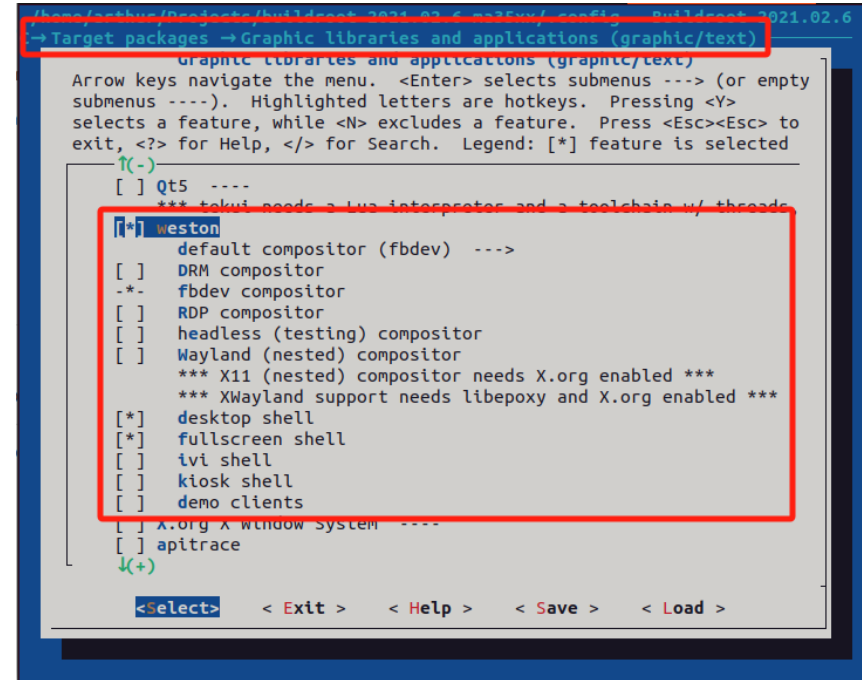
phoney@phoney:~/project/buildroot-2021.02.6/config$ Buildroot 2021.02.6 Configuration
->Target packages -> Fonts, cursors, icons, sounds and themes
    Fonts, cursors, icons, sounds and themes
    Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenu
    ----). Highlighted letters are hotkeys. Pressing <Y> selects a feature, while <N>
    excludes a feature. Press <Esc><Esc> to exit, <?> for Help, </> for Search.
    Legend: [*] feature is selected [ ] feature is excluded

    *** Cursors ***
    [ ] conix-cursors
    [ ] obsidian-cursors
    *** Fonts ***
    [ ] Bitstream Vera
    [ ] cantarell
    [ ] DejaVu fonts
    [ ] font-awesome
    [ ] ghostscript-fonts
    [ ] libxfont
    [*] Liberation (Free fonts)
    [*] mono fonts
    [*] sans fonts
    [*] serif fonts
    [ ] wqy-zenhei
    *** Icons ***
    [ ] google-material-design-icons
    [ ] hicolor icon theme
    *** Sounds ***
    [ ] sound-theme-borealis
    [ ] sound-theme-freedesktop
    *** Themes ***

    <select>  < Exit >  < Help >  < Save >  < Load >
    
```

Configuring Buildroot to enable weston

- Select **weston**
 - default compositor: fbdev
 - desktop shell
 - fullscreen shell



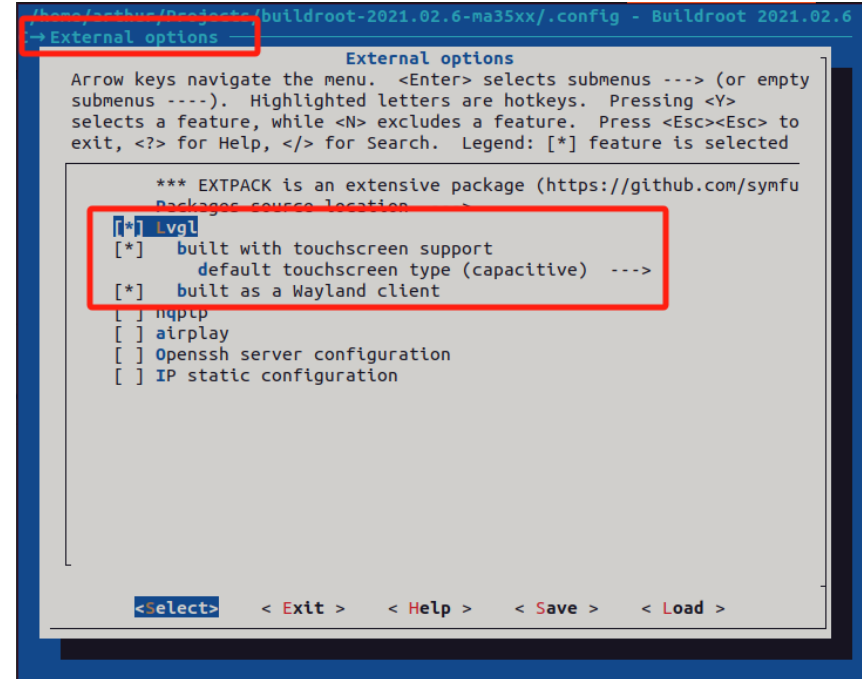
```
Buildroot 2021.02.6
→Target packages →Graphic libraries and applications (graphic/text)
Graphic libraries and applications (graphic/text)
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
submenus ----). Highlighted letters are hotkeys. Pressing <Y>
selects a feature, while <N> excludes a feature. Press <Esc><Esc> to
exit, <?> for Help, </> for Search. Legend: [*] feature is selected
↑(-)
[ ] Qt5 ----
*** tokui needs a lua interpreter and a toolchain w/ threads
[*] weston
    default compositor (fbdev) --->
    [ ] DRM compositor
    -*. fbdev compositor
    [ ] RDP compositor
    [ ] headless (testing) compositor
    [ ] Wayland (nested) compositor
    *** X11 (nested) compositor needs X.org enabled ***
    *** XWayland support needs libepoxy and X.org enabled ***
    [*] desktop shell
    [*] fullscreen shell
    [ ] ivi shell
    [ ] kiosk shell
    [ ] demo clients
[ ] X.org X Window System ----
[ ] apitrace
↓(+)
<Select> <Exit> <Help> <Save> <Load>
```

Configuring Buildroot to enable LVGL

- Select *lvgl*

Choose the right *default touchscreen type*, capacitive or resistive.

With Wayland enabled, LVGL should always be *built as a Wayland client*.



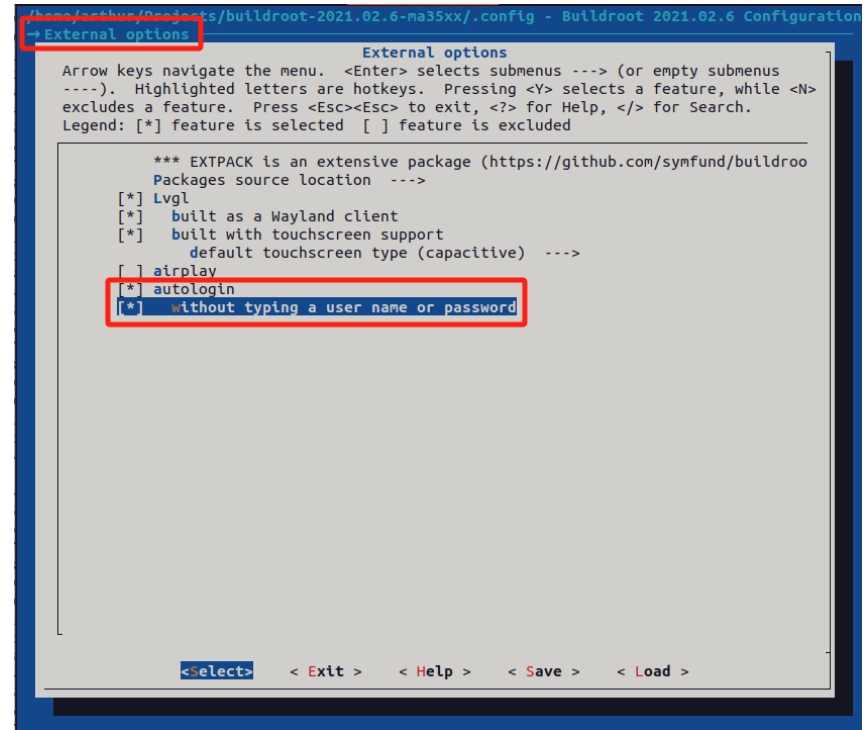
```
buildroot-2021.02.6-ma35xx/.config - Buildroot 2021.02.6
-> External options
External options
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
submenus ----). Highlighted letters are hotkeys. Pressing <Y>
selects a feature, while <N> excludes a feature. Press <Esc><Esc> to
exit, <?> for Help, </> for Search. Legend: [*] feature is selected

*** EXTPACK is an extensive package (https://github.com/symfu
Packages source location:
[*] lvgl
    [*] built with touchscreen support
        default touchscreen type (capacitive) --->
    [*] built as a Wayland client
    [ ] ntp
    [ ] airplay
    [ ] Openssh server configuration
    [ ] IP static configuration

<Select> <Exit> <Help> <Save> <Load>
```

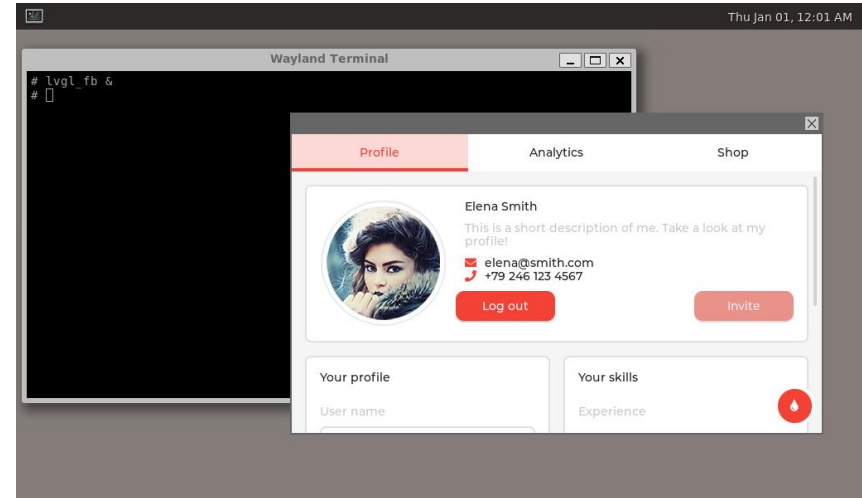
Auto login without typing user name

- Auto-login logs into Wayland/ Weston without typing a user name or password



Runing LVGL on Wayland

- Touch the terminal icon in the title bar to launch the **weston-terminal**.
- On the command line, type '**lvgl_fb &**' to run LVGL application.



The skeleton minimal LVGL with Wayland

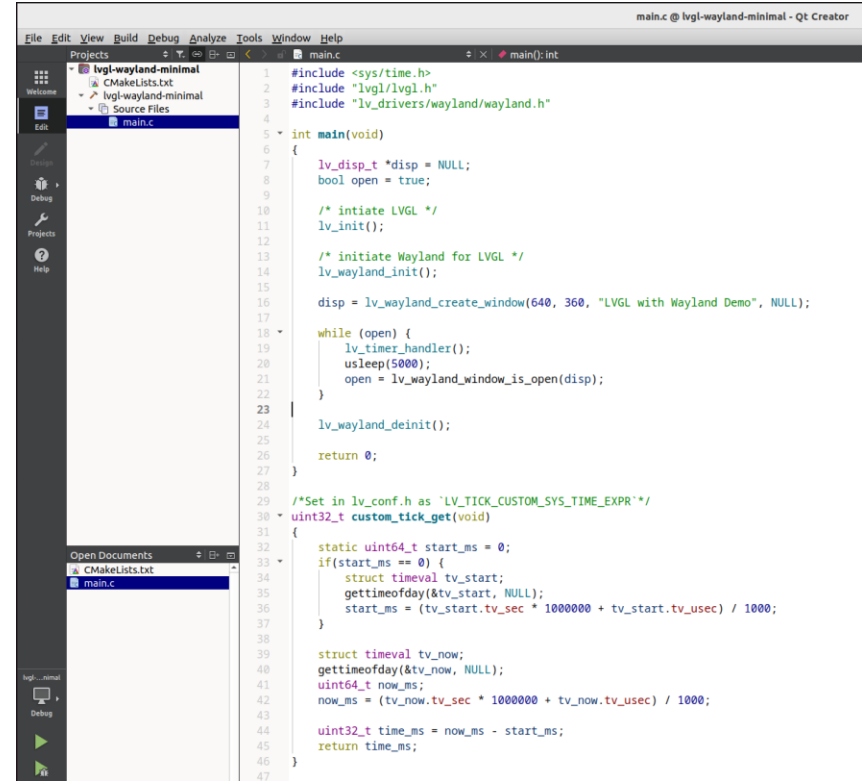
- The skeleton example

[ma35d1-portal/examples/wayland/lvgl/lvgl-wayland-minimal](#)

- Modify **CMakeLists.txt**, set variable **BR2_DIR** to the actual path of Buildroot. If Qt Creator Kits use SDK toolchain, just set **CMAKE_SYSROOT** to the path of SYSROOT.

`set(BR2_DIR "/path/to/buildroot")`

`set(CMAKE_SYSROOT "/path/to/sysroot")`

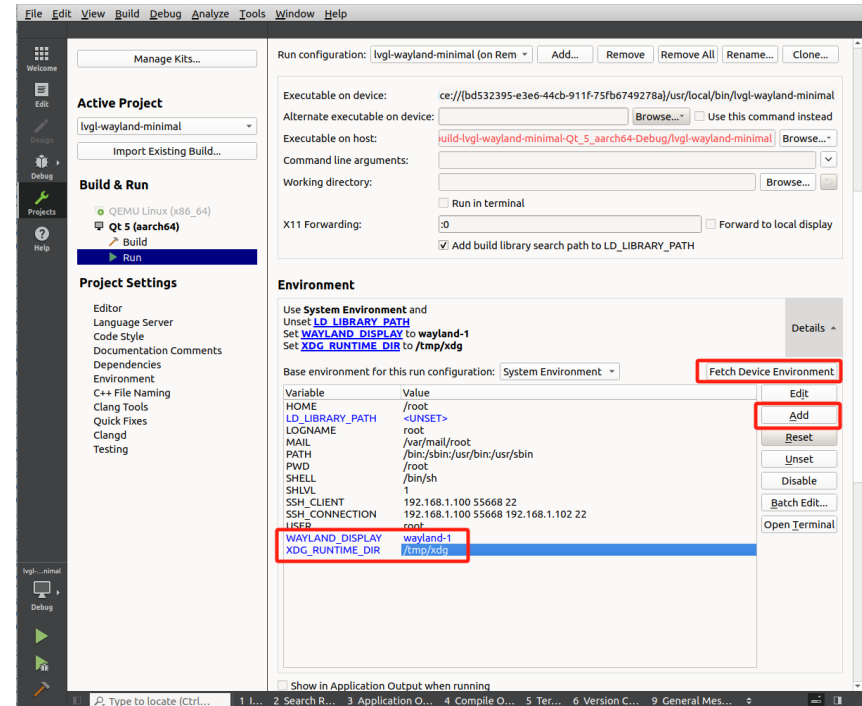


The screenshot shows the Qt Creator IDE with the 'lvgl-wayland-minimal' project open. The 'main.c' file is selected in the 'Source Files' pane. The code in the editor includes headers for `<sys/time.h>`, `"lvgl/lvgl.h"`, and `"lv_drivers/wayland/wayland.h"`. The `main` function initializes LVGL and Wayland, creates a window titled "LVGL with Wayland Demo", and enters a loop that calls `lv_timer_handler()` and `usleep(5000)` while the window is open. It also includes a custom tick function `custom_tick_get` that calculates time in milliseconds using `gettimeofday`.

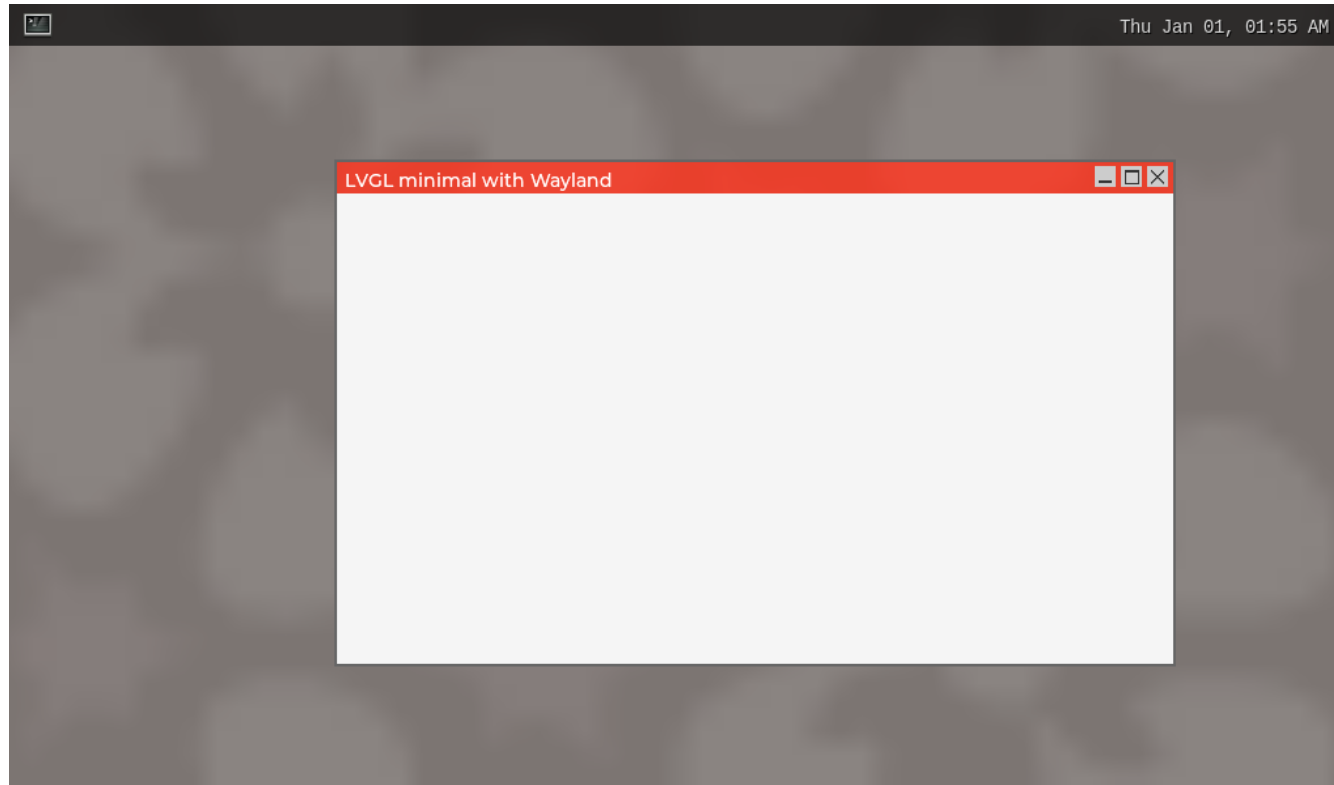
```
1 #include <sys/time.h>
2 #include "lvgl/lvgl.h"
3 #include "lv_drivers/wayland/wayland.h"
4
5 int main(void)
6 {
7     lv_disp_t *disp = NULL;
8     bool open = true;
9
10    /* initiate LVGL */
11    lv_init();
12
13    /* initiate Wayland for LVGL */
14    lv_wayland_init();
15
16    disp = lv_wayland_create_window(640, 360, "LVGL with Wayland Demo", NULL);
17
18    while (open) {
19        lv_timer_handler();
20        usleep(5000);
21        open = lv_wayland_window_is_open(disp);
22    }
23
24    lv_wayland_deinit();
25    return 0;
26
27
28
29 /*Set in lv_conf.h as 'LV_TICK_CUSTOM_SYS_TIME_EXPR'*/
30 uint32_t custom_tick_get(void)
31 {
32     static uint64_t start_ms = 0;
33     if(start_ms == 0) {
34         struct timeval tv_start;
35         gettimeofday(&tv_start, NULL);
36         start_ms = (tv_start.tv_sec * 1000000 + tv_start.tv_usec) / 1000;
37     }
38
39     struct timeval tv_now;
40     gettimeofday(&tv_now, NULL);
41     uint64_t now_ms;
42     now_ms = (tv_now.tv_sec * 1000000 + tv_now.tv_usec) / 1000;
43
44     uint32_t time_ms = now_ms - start_ms;
45     return time_ms;
46 }
```

Remote debugging with LVGL on Wayland

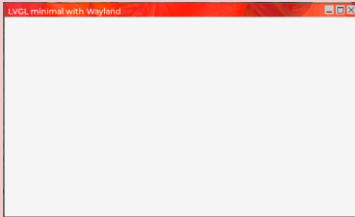
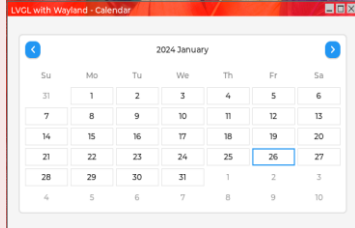

- Fetch device environment
- Add two variables
 - **XDG_RUNTIME_DIR** (/tmp/xdg)
 - **WAYLAND_DISPLAY** (wayland-1)



LVGL minimal with Wayland



LVGL examples with Wayland

Apps	Location (https://github.com/symfund/ma35d1-portal)	Thumbnails
Minimal	examples/wayland/lvgl/lvgl-wayland-minimal	 A screenshot of a window titled "LVGL minimal with Wayland" showing a plain white background.
Calendar	examples/wayland/lvgl/lvgl-wayland-calendar	 A screenshot of a window titled "LVGL with Wayland - Calendar" showing a calendar for January 2024. The calendar is a grid with days of the week (Su, Mo, Tu, We, Th, Fr, Sa) and dates. The date 26 is highlighted.
Widgets	examples/wayland/lvgl/lvgl-wayland-widgets-demo	 A screenshot of a window titled "LVGL widgets demo with Wayland" showing a user profile interface. It includes a profile picture, name "Elena Smith", email "elena@smith.com", phone number "+79 246 123 4567", and buttons for "Log out" and "Invite". There are also sections for "Your profile" and "Your skills".



Qt5 with Wayland

<https://www.qt.io>

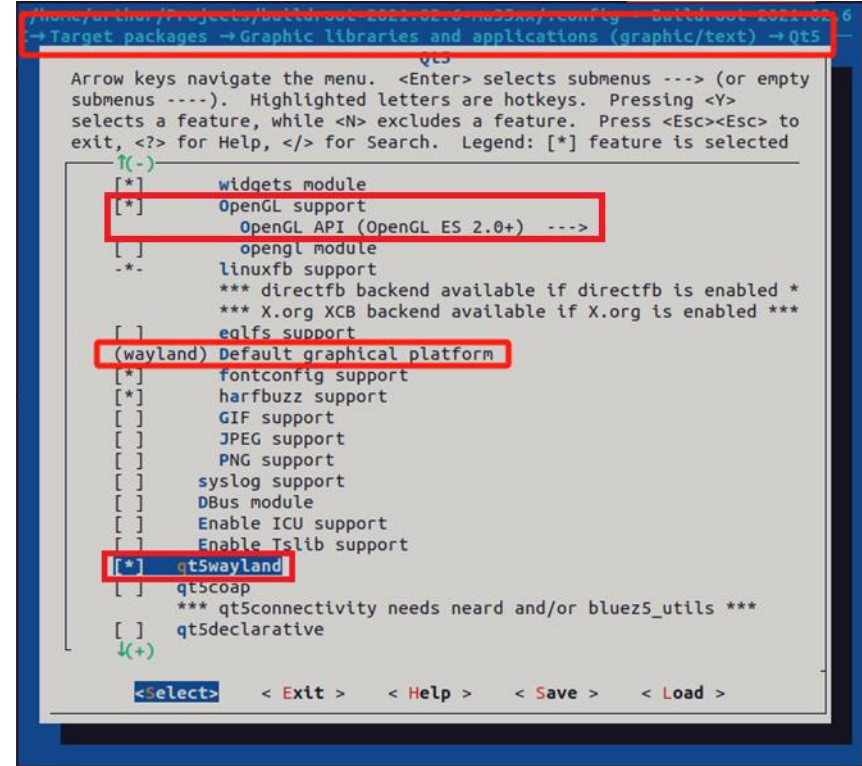
Configuring Buildroot to enable Qt5

- Select Qt5

Component `qt5wayland` requires *OpenGL support*. OpenGL API uses *OpenGL ES 2.0+*.

The *default graphical platform* is set to `wayland` or `wayland-egl`.

With wayland enabled, *qt5wayland* component should always be selected.

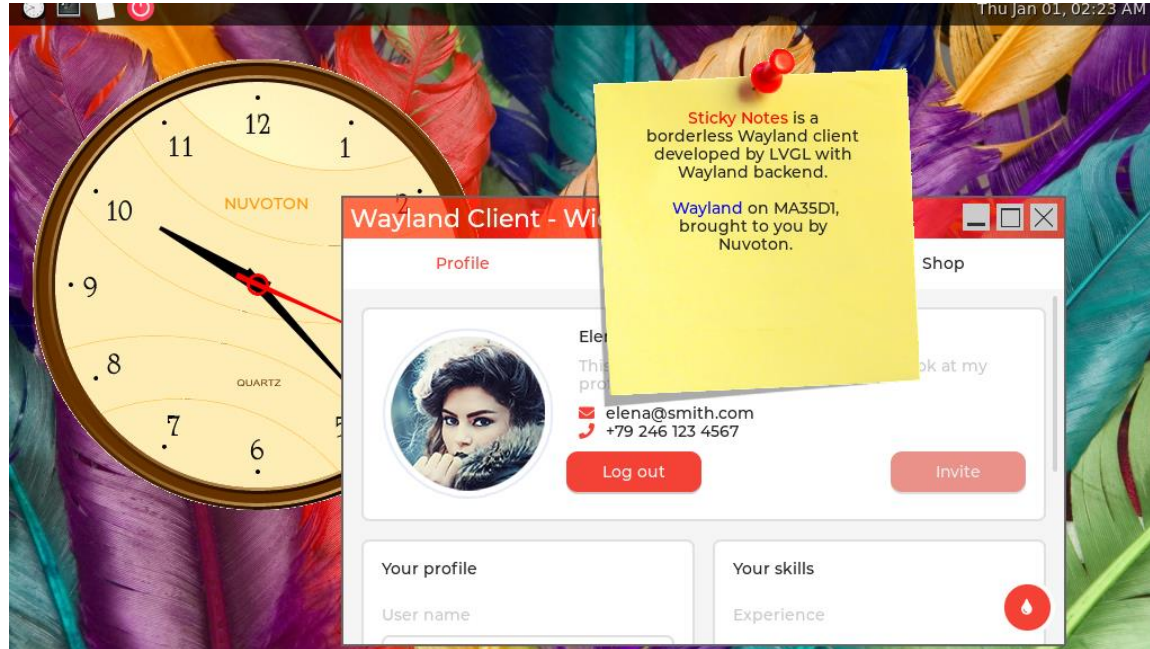


The screenshot shows the Buildroot configuration menu. At the top, a red box highlights the navigation path: "Target packages → Graphic libraries and applications (graphic/text) → Qt5". Below this, a text box explains the navigation: "Arrow keys navigate the menu. <Enter> selects submenus --- (or empty submenu ---). Highlighted letters are hotkeys. Pressing <Y> selects a feature, while <N> excludes a feature. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] feature is selected". The menu lists various features with checkboxes. A red box highlights "OpenGL support", which has a sub-menu "OpenGL API (OpenGL ES 2.0+) ---". Another red box highlights "(wayland) Default graphical platform". A third red box highlights "[*] qt5wayland", which is currently selected. At the bottom, navigation buttons are shown: "<Select>", "<Exit>", "<Help>", "<Save>", and "<Load>".

```
phone/other/projects/buildroot-2020.02.0-mass/config - Buildroot 2020.02.0 6
→ Target packages → Graphic libraries and applications (graphic/text) → Qt5
Qt5
Arrow keys navigate the menu. <Enter> selects submenus --- (or empty
submenu ---). Highlighted letters are hotkeys. Pressing <Y>
selects a feature, while <N> excludes a feature. Press <Esc><Esc> to
exit, <?> for Help, </> for Search. Legend: [*] feature is selected
[*] (-)
[*] widgets module
[*] OpenGL support
    OpenGL API (OpenGL ES 2.0+) ---
[ ] opengl module
[*] linuxfb support
    *** directfb backend available if directfb is enabled *
    *** X.org XCB backend available if X.org is enabled ***
[ ] eglfs support
(wayland) Default graphical platform
[*] fontconfig support
[*] harfbuzz support
[ ] GIF support
[ ] JPEG support
[ ] PNG support
[ ] syslog support
[ ] DBus module
[ ] Enable ICU support
[ ] Enable Islib support
[*] qt5wayland
[ ] qt5coap
    *** qt5connectivity needs neard and/or bluez5_utils ***
[ ] qt5declarative
k(+)
<Select> <Exit> <Help> <Save> <Load>
```

Qt5 Icons example on Wayland





Customizing Weston

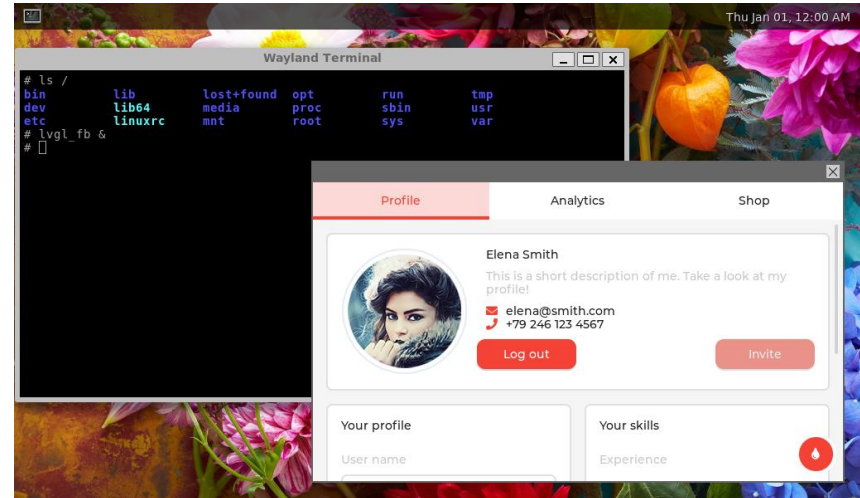
Changing wallpapers, adding launch icons, setting idle lock time

Changing wallpaper

- Prepare a wallpaper on target location
`/usr/share/weston/wallpaper.jpg`
- Edit **`/etc/xdg/weston/weston.ini`**, under section **'shell'**, modify the entry **'background-image'**

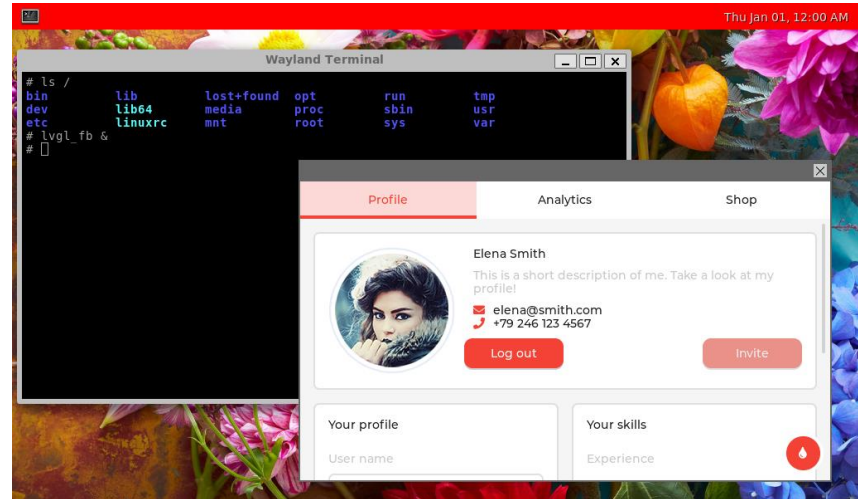
[shell]

`background-image=/usr/share/weston/wallpaper.jpg`



Modifying task panel color

- Edit `/etc/xdg/weston/weston.ini`,
under section **'shell'**, modify the
entry **'panel-color'**
[shell]
`panel-color=0xffff0000`
- Add translucent color
[shell]
`panel-color=0x12ff0000`



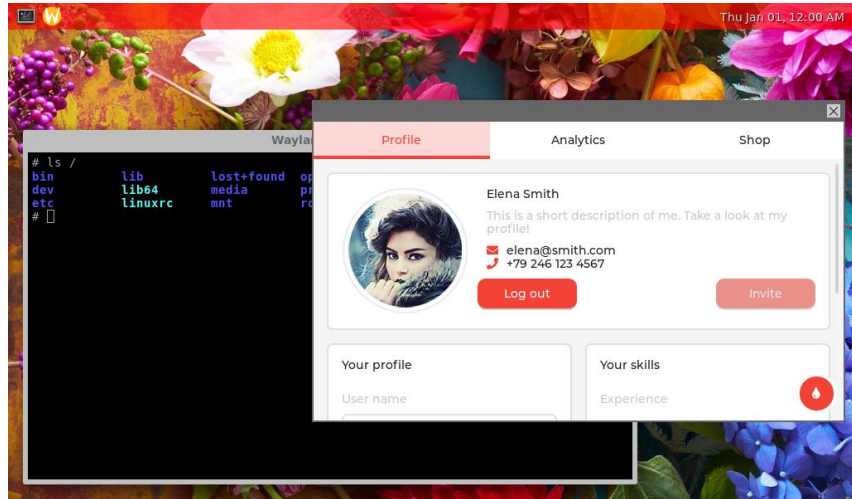
Adding launch icons in task panel

- Edit `/etc/xdg/weston/weston.ini`, add section **'launcher'**, under section **'launcher'**, append entries **'icon'** and **'path'**

[launcher]

icon=`/usr/share/weston/wayland24.png`

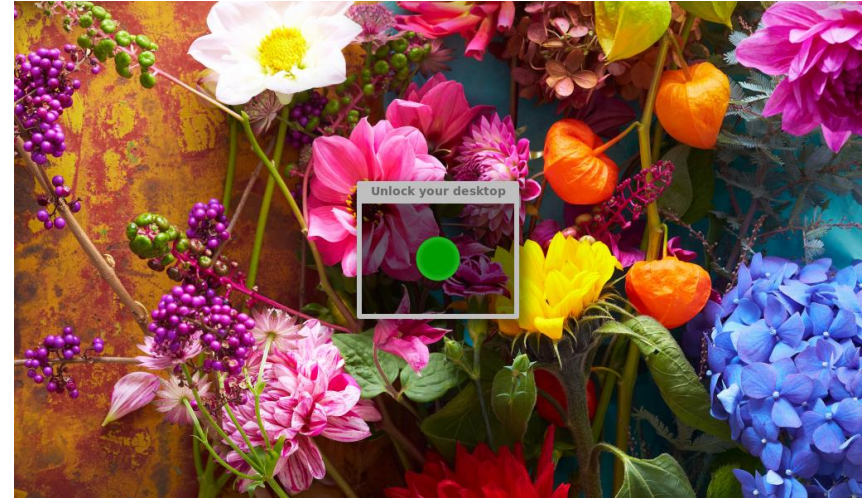
path=`/usr/bin/lvgl_fb`



Setting idle lock time

- Edit `/etc/xdg/weston/weston.ini`, under section **'core'**, modify the entry **'idle-time'** (seconds)

```
[core]  
idle-time=300
```



Irregular non-rectangle shaped window

- Some irregular non-rectangle shaped window applications like **Analog Clock** require LVGL to run with window decorations (title, border) disabled. This goal can be achieved by setting environment variable `LV_WAYLAND_DISABLE_WINDOWDECORATION` to **"1"** before launching the shaped window.

```
$ LV_WAYLAND_DISABLE_WINDOWDECORATION=1 lvgl-wayland-analog-clock &
```

