How I built a Check-in Kiosk for UbuCon Korea using Ubuntu Frame, Flutter and Raspberry Pi

UbuCon Asia 2024
2024-09-01
Jaipur, India

Youngbin Han
Member @ Ubuntu LoCo Council
Organizer @ Ubuntu Korea Community

Youngbin Han

Mostly involved with Ubuntu Community

- Ubuntu Member - **ybhan@ubuntu.com**
- Member @ Ubuntu LoCo(Local Community) Council
- Organizer @ Ubuntu Korea Community (Korean LoCo)
- Organizer @ UbuCon Asia, UbuCon Korea, DebConf24

Work

- Software Engineer @ AhnLab CloudMate
  - (Cloud MSP company in Seoul)

Website: https://youngbin.xyz

Few things to note before start…

This talk isn’t about some kind of best practices

I’ll just talk about my own experience as a person new to Ubuntu Core, Ubuntu Frame and Flutter

And many of those experiences includes some kind of weird workaround. So… If you want to utilize some of my experiences, use it at your own risk :)

Why build check-in kiosk?

- Prevent missing check-in
- Automate check-in + other process(such as name tag printing)
- To handle check-in of registration from multiple platforms
- Event platform has no support for check-in app or it’s expensive paid add-on
- And… Just for fun!

Why I chose Ubuntu Frame & Ubuntu Core

What is Ubuntu Frame?

put your pictures up

- Simple display-server-shell using Mir
- One application, full screen
- Run as a daemon
- Easy to deploy to Ubuntu Core

Snap이란?

- 리눅스 시스템을 위한 차세대 패키지 관리 시스템
- 격리 된 앱 실행 환경 제공
- 보안 향상
- 의존성 해결 용이

By the way, What are Ubuntu Core and Ubuntu Frame?

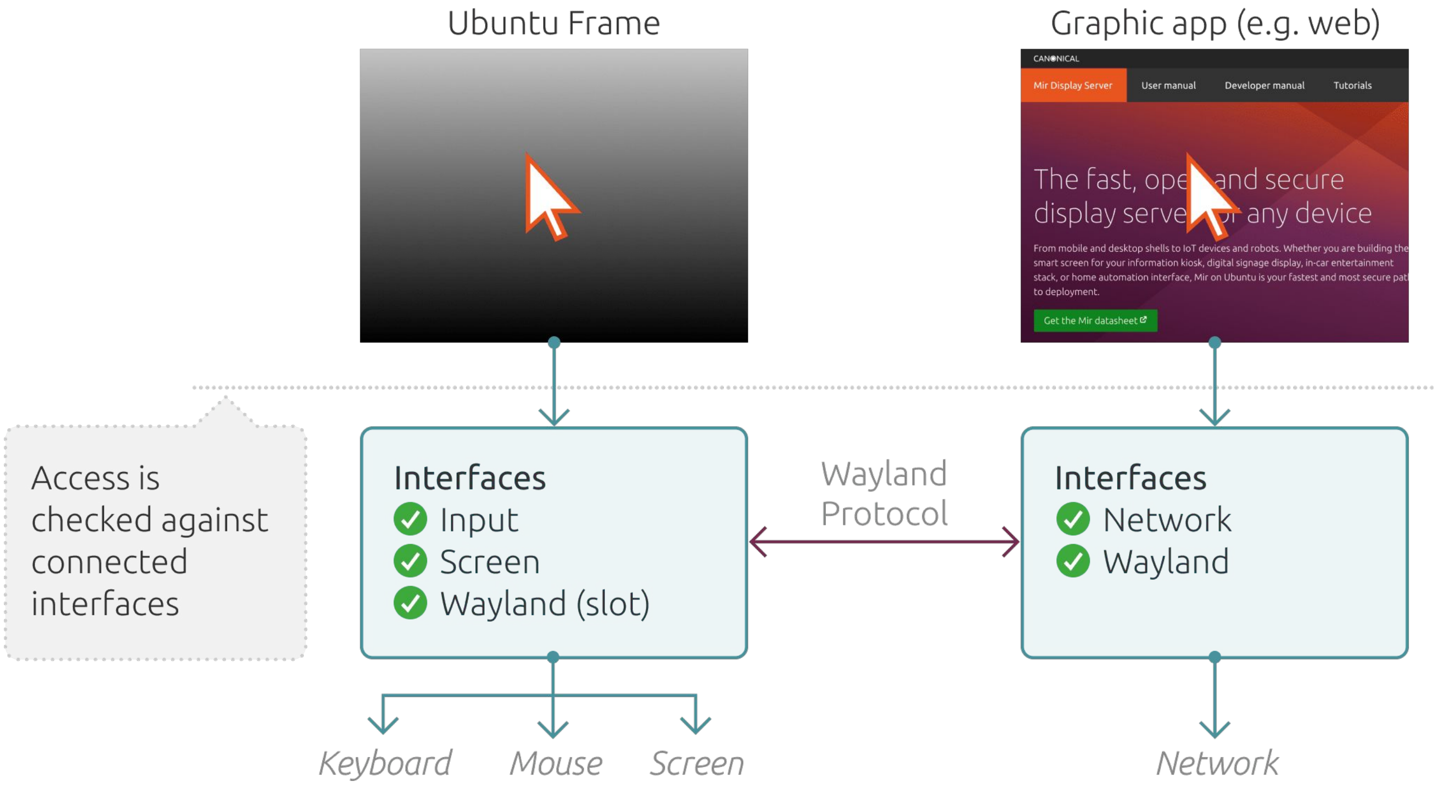
Ubuntu Core

- The OS Optimized for IoT, Edge and Embedded
- All packaged are managed with Snap

Ubuntu Frame

- Fullscreen display server for embedded graphical display such as kiosk and digital signage
- Built with Mir Display server - A Wayland compositor

Snap Confinement: Shell and App are confined separately



Ubuntu Frame

Graphic app (e.g. web)

The fast, ope and secure display serve r any device

From mobile and desktop shells to IoT devices and robots. Whether you are building the smart screen for your information kiosk, digital signage display, in-car entertainment stack, or home automation interface, Mir on Ubuntu is your fastest and most secure pat to deployment.

Access is checked against connected interfaces

Interfaces
- ✓ Input
- ✓ Screen
- ✓ Wayland (slot)

Wayland Protocol

Interfaces
- ✓ Network
- ✓ Wayland

Keyboard *Mouse* *Screen*

Network

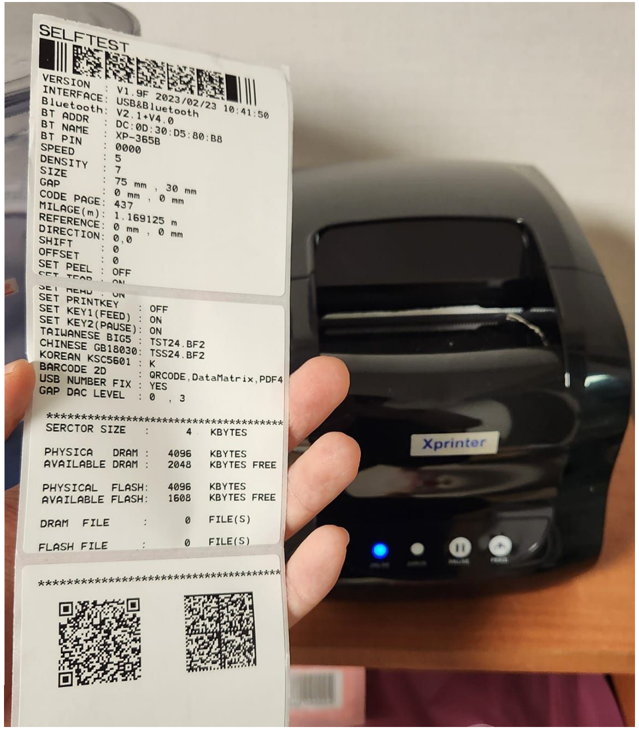
https://mir-server.io/ubuntu-frame

The original plan was to…

Build Kiosk with:

- Ubuntu Core, Ubuntu Frame
- Use existing webcam for scanning QR code
- Flutter with Yaru.dart
- VisionFive2 - A RISC-V SBC I just got from crowdfunding
- Cheap label printer with linux driver support

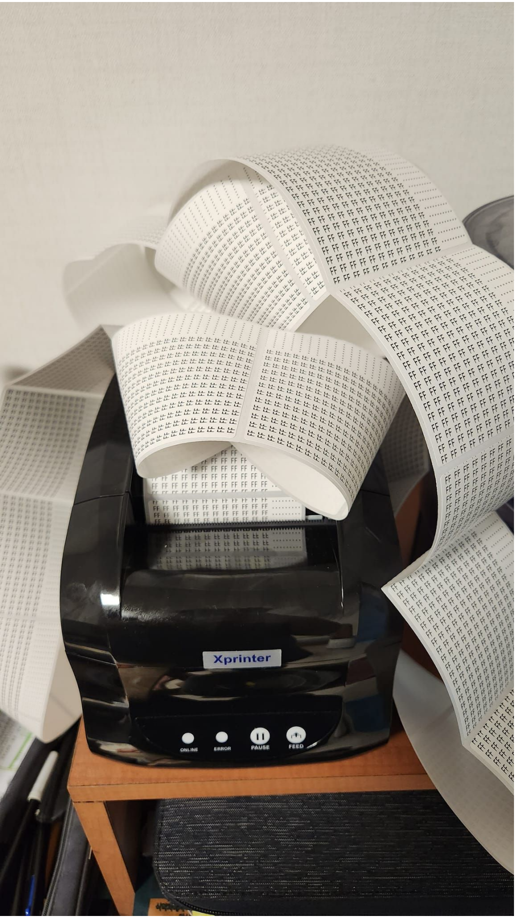
First, Buy a *cheap* label printer...?



```
SELFTEST
VERSION      : V1.9F 2023/02/23 10:41:50
INTERFACE: USB&Bluetooth
Bluetooth: V2.1+V4.0
BT ADDR      : DC:0D:30:D5:80:B8
BT NAME      : XP-365B
BT PIN       : 0000
SPEED        : 5
DENSITY      : 7
SIZE         : 75 mm , 30 mm
GAP          : 0 mm , 0 mm
CODE PAGE: 437
MILAGE(m): 1.169125 m
REFERENCE: 0 mm , 0 mm
DIRECTION: 0,0
SHIFT        : 0
OFFSET       : 0
SET PEEL : OFF
SET TEAR : ON
SET HEAD : ON
SET PRINTKEY       : OFF
SET KEY1(FEED)     : ON
SET KEY2(PAUSE): ON
TAIWANESE BIG5     : TST24.BF2
CHINESE GB18030: TSS24.BF2
KOREAN KSC5601     : K
BARCODE 2D         : QRCODE,DataMatrix,PDF4
USB NUMBER FIX     : YES
GAP DAC LEVEL      : 0  , 3

 ************************************
 SERCTOR SIZE    :      4   KBYTES

 PHYSICA   DRAM :   4096   KBYTES
 AVAILABLE DRAM :   2048   KBYTES FREE

 PHYSICAL  FLASH:   4096   KBYTES
 AVAILABLE FLASH:   1608   KBYTES FREE

 DRAM  FILE     :      0   FILE(S)

 FLASH FILE     :      0   FILE(S)

 ************************************
```

That “seem” to be working with linux

Xprinter XP-365B

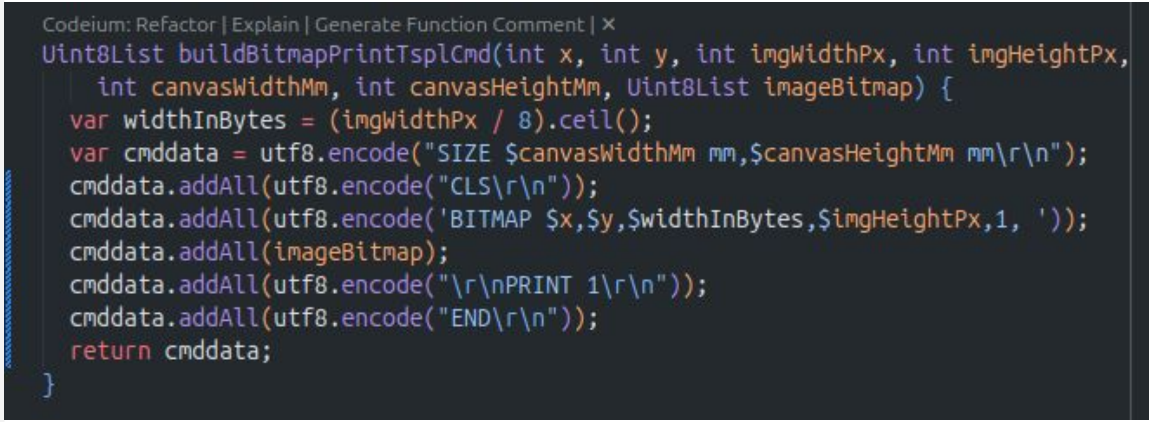


A bit of label printing test with some failures...

- Printer driver had linux support
  - But only for x86, not for arm64
  - And my SBC for Kiosk setup is either arm64 or riscv64...
- Tried to write and send TSPL command manually with libusb instead for printing labels
- <- Figuring out how other mobile label printer apps send TSPL commands to print labels by dumping data...
  - Oooooops...

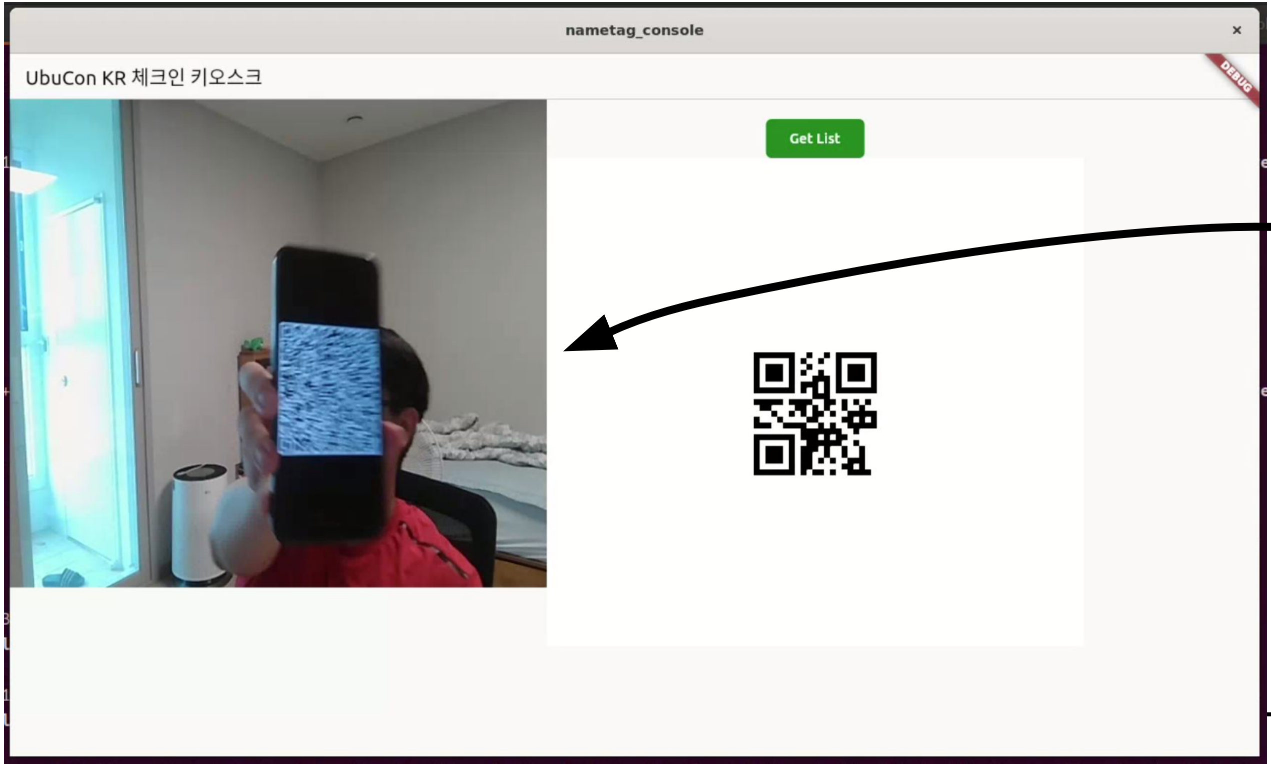
Working with TSPL commands

```
SIZE 70 mm,70 mm

CLS

BITMAP 0,50,68,500,1,

<BITMAP DATA>

PRINT

END
```



```
Uint8List buildBitmapPrintTsplCmd(int x, int y, int imgWidthPx, int imgHeightPx,
    int canvasWidthMm, int canvasHeightMm, Uint8List imageBitmap) {
  var widthInBytes = (imgWidthPx / 8).ceil();
  var cmddata = utf8.encode("SIZE $canvasWidthMm mm,$canvasHeightMm mm\r\n");
  cmddata.addAll(utf8.encode("CLS\r\n"));
  cmddata.addAll(utf8.encode('BITMAP $x,$y,$widthInBytes,$imgHeightPx,1, '));
  cmddata.addAll(imageBitmap);
  cmddata.addAll(utf8.encode("\r\nPRINT 1\r\n"));
  cmddata.addAll(utf8.encode("END\r\n"));
  return cmddata;
}
```

Write App with Flutter + Yaru.dart



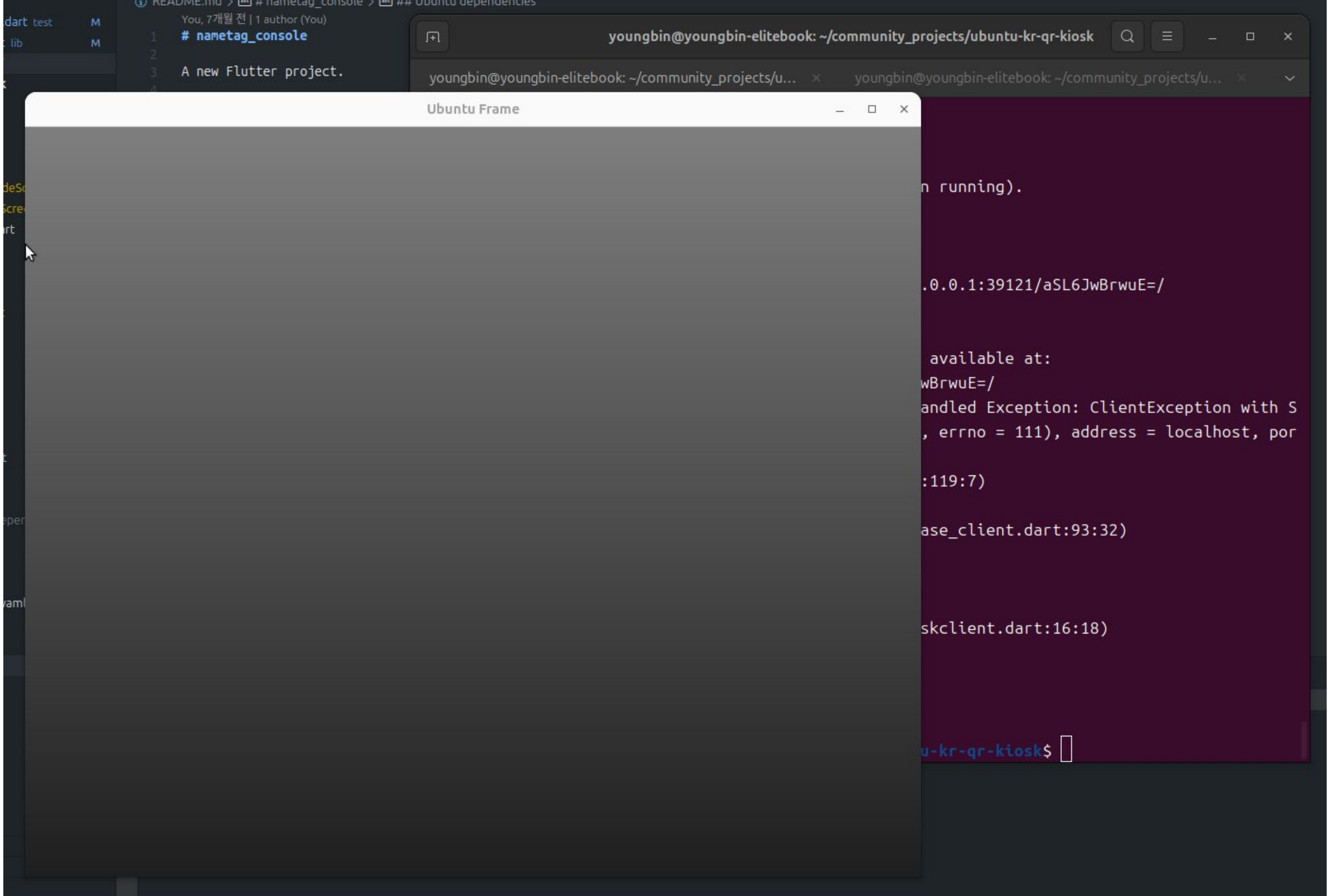
App for reading data from QR code

Used 3rd party gstreamer player plugin for webcam video input

Testing flutter app with Ubuntu Frame on your desktop

You can install ubuntu-frame snap then run it for testing your GUI app on Ubuntu Frame.

```
sudo snap install ubuntu-frame

WAYLAND_DISPLAY=wayland-99 ubuntu-frame

WAYLAND_DISPLAY=wayland-99 flutter run
```

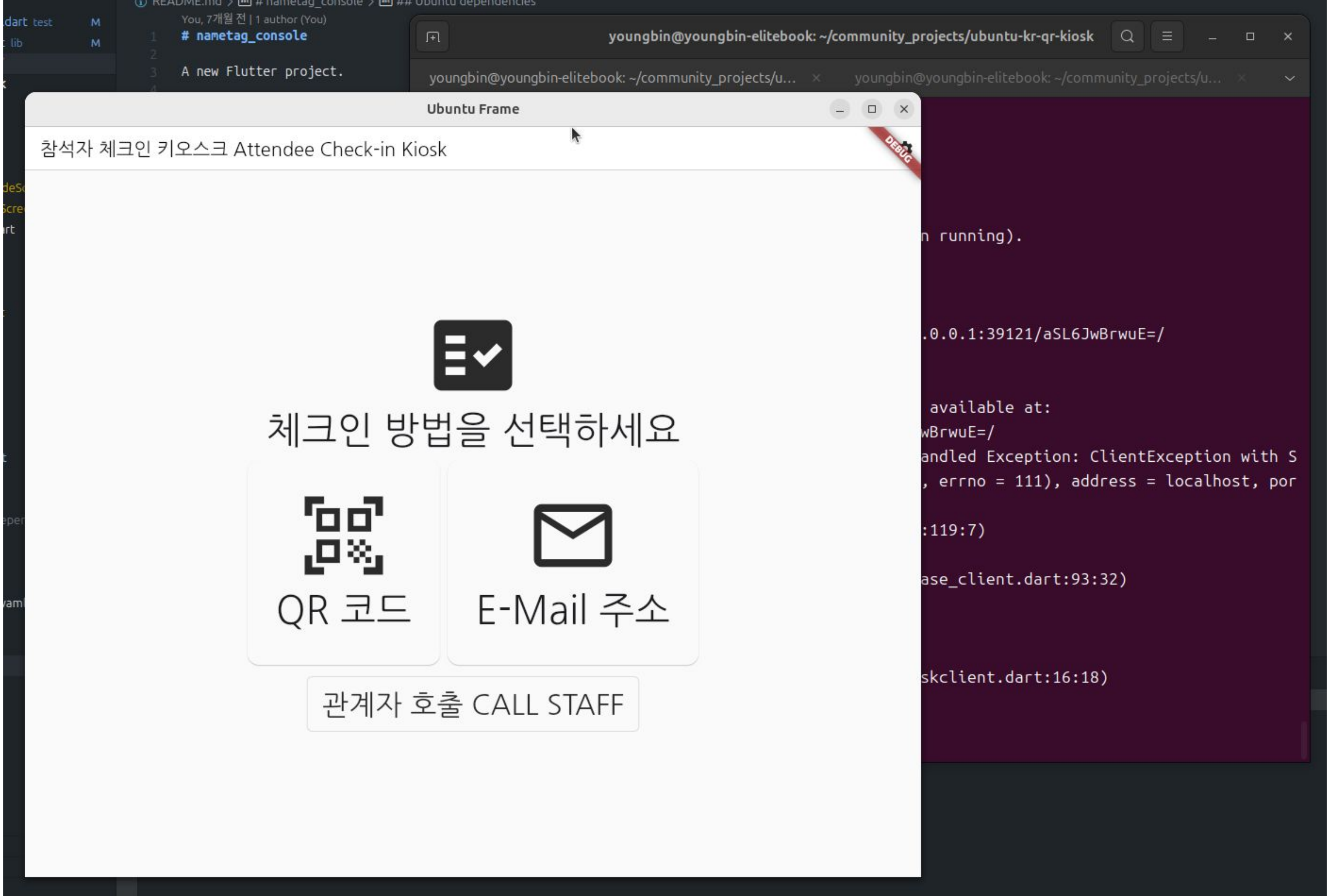

README.md > # nametag_console > ## Ubuntu dependencies
dart test M
lib M
Ubuntu Frame
youngbin@youngbin-elitebook: ~/community_projects/u...
youngbin@youngbin-elitebook: ~/community_projects/u...

You, 7개월 전 | 1 author (You)

```
# nametag_console

A new Flutter project.
```

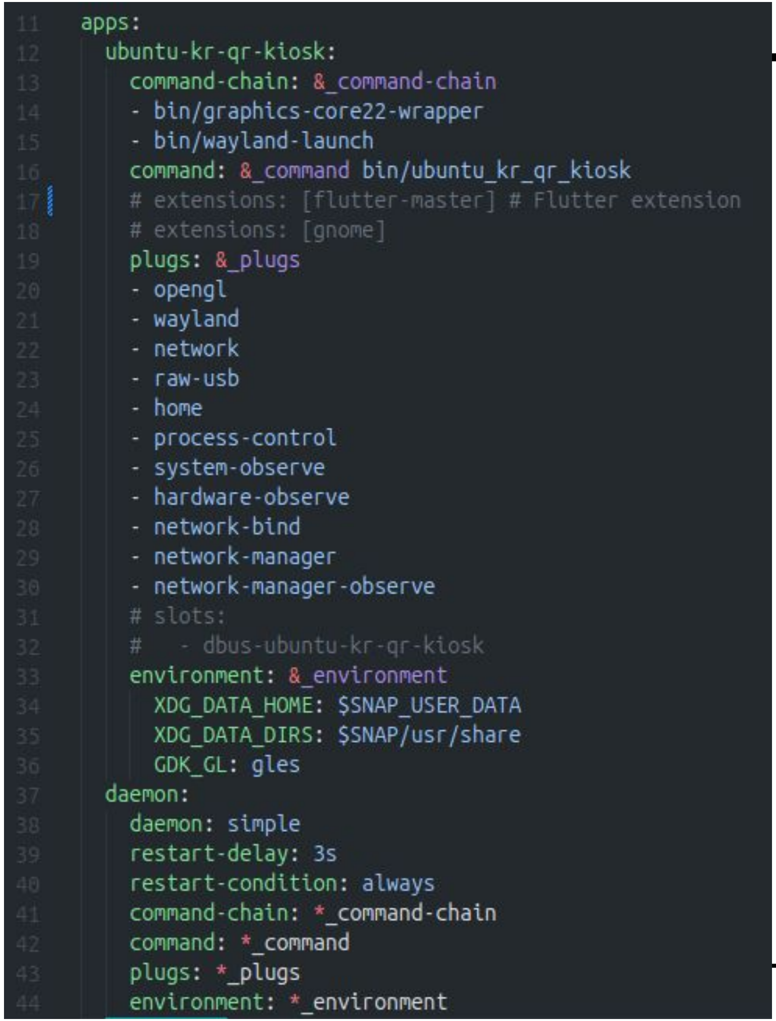
youngbin@youngbin-elitebook: ~/community_projects/ubuntu-kr-qr-kiosk

```
n running).

.0.0.1:39121/aSL6JwBrwuE=/

 available at:
wBrwuE=/
andled Exception: ClientException with S
, errno = 111), address = localhost, por

:119:7)

ase_client.dart:93:32)


skclient.dart:16:18)



u-kr-qr-kiosk$ 
```


README.md > # nametag_console > ## Ubuntu dependencies
dart test M
lib M
youngbin@youngbin-elitebook: ~/community_projects/ubuntu-kr-qr-kiosk
youngbin@youngbin-elitebook: ~/community_projects/u...
youngbin@youngbin-elitebook: ~/community_projects/u...
Ubuntu Frame
참석자 체크인 키오스크 Attendee Check-in Kiosk
DEBUG
체크인 방법을 선택하세요
QR 코드
E-Mail 주소
관계자 호출 CALL STAFF

```
You, 7개월 전 | 1 author (You)
# nametag_console

A new Flutter project.
```

```
n running).

.0.0.1:39121/aSL6JwBrwuE=/

 available at:
wBrwuE=/
andled Exception: ClientException with S
, errno = 111), address = localhost, por

:119:7)

ase_client.dart:93:32)

skclient.dart:16:18)
```



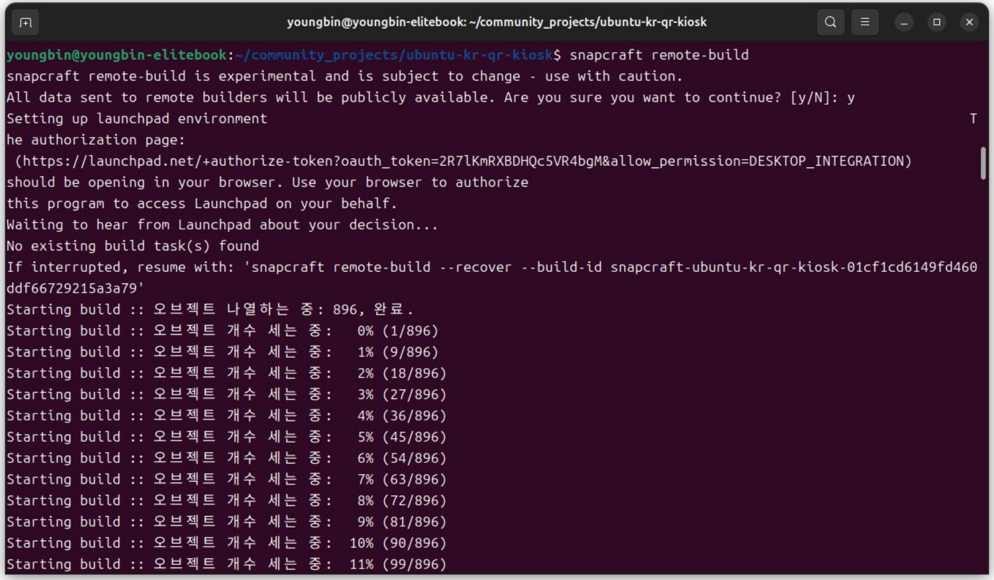
```
apps:
  ubuntu-kr-qr-kiosk:
    command-chain: &_command-chain
    - bin/graphics-core22-wrapper
    - bin/wayland-launch
    command: &_command bin/ubuntu_kr_qr_kiosk
    # extensions: [flutter-master] # Flutter extension
    # extensions: [gnome]
    plugs: &_plugs
    - opengl
    - wayland
    - network
    - raw-usb
    - home
    - process-control
    - system-observe
    - hardware-observe
    - network-bind
    - network-manager
    - network-manager-observe
    # slots:
    #   - dbus-ubuntu-kr-qr-kiosk
    environment: &_environment
      XDG_DATA_HOME: $SNAP_USER_DATA
      XDG_DATA_DIRS: $SNAP/usr/share
      GDK_GL: gles
  daemon:
    daemon: simple
    restart-delay: 3s
    restart-condition: always
    command-chain: *_command-chain
    command: *_command
    plugs: *_plugs
    environment: *_environment
```

Writing your snapcraft.yml

You can’t take advantage of snapcraft extensions for building desktop snap. But, You don’t need to start from scratch thanks to example project provided through documentation.

Your app will launch as daemon - So that it won’t block your command prompt and also automatically start on boot on Ubuntu Core environment.

Building Snap for your Raspberry Pi (or other SBCs)


youngbin@youngbin-elitebook: ~/community_projects/ubuntu-kr-qr-kiosk

```
youngbin@youngbin-elitebook:~/community_projects/ubuntu-kr-qr-kiosk$ snapcraft remote-build
snapcraft remote-build is experimental and is subject to change - use with caution.
All data sent to remote builders will be publicly available. Are you sure you want to continue? [y/N]: y
Setting up launchpad environment                                                                      T
he authorization page:
 (https://launchpad.net/+authorize-token?oauth_token=2R7lKmRXBDHQc5VR4bgM&allow_permission=DESKTOP_INTEGRATION)
should be opening in your browser. Use your browser to authorize
this program to access Launchpad on your behalf.
Waiting to hear from Launchpad about your decision...
No existing build task(s) found
If interrupted, resume with: 'snapcraft remote-build --recover --build-id snapcraft-ubuntu-kr-qr-kiosk-01cf1cd6149fd460
ddf66729215a3a79'
Starting build :: 오브젝트 나열하는 중: 896, 완료.
Starting build :: 오브젝트 개수 세는 중:   0% (1/896)
Starting build :: 오브젝트 개수 세는 중:   1% (9/896)
Starting build :: 오브젝트 개수 세는 중:   2% (18/896)
Starting build :: 오브젝트 개수 세는 중:   3% (27/896)
Starting build :: 오브젝트 개수 세는 중:   4% (36/896)
Starting build :: 오브젝트 개수 세는 중:   5% (45/896)
Starting build :: 오브젝트 개수 세는 중:   6% (54/896)
Starting build :: 오브젝트 개수 세는 중:   7% (63/896)
Starting build :: 오브젝트 개수 세는 중:   8% (72/896)
Starting build :: 오브젝트 개수 세는 중:   9% (81/896)
Starting build :: 오브젝트 개수 세는 중:  10% (90/896)
Starting build :: 오브젝트 개수 세는 중:  11% (99/896)
```

Using Remote build to leverage the Launchpad build farm

(Requires Launchpad.net account)

`snapcraft remote-build`

Build Snap with GitHub Actions

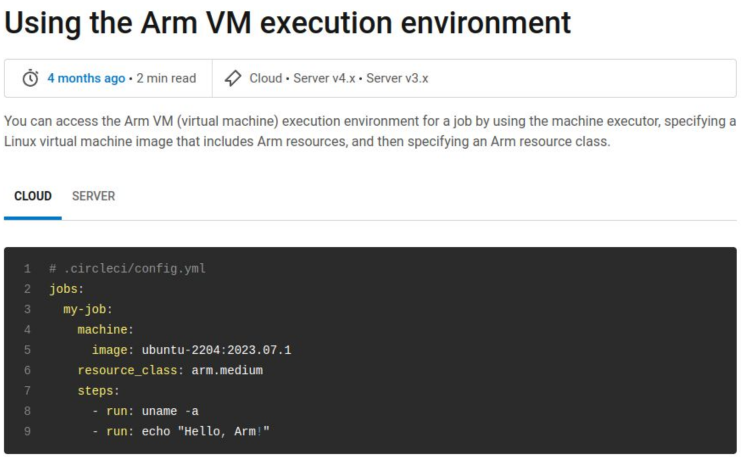
`snapcore/action-build` action

- If you're building for amd64 target
- https://github.com/snapcore/action-build

`diddlesnaps/snapcraft-multiarch-action` action

- If you want to build for multiple targets. such as amd64, arm64(for your RPi), armhf and more.
- https://github.com/diddlesnaps/snapcraft-multiarch-action

Build with Circle CI Native ARM Runner



Using the Arm VM execution environment

4 months ago • 2 min read | Cloud • Server v4.x • Server v3.x

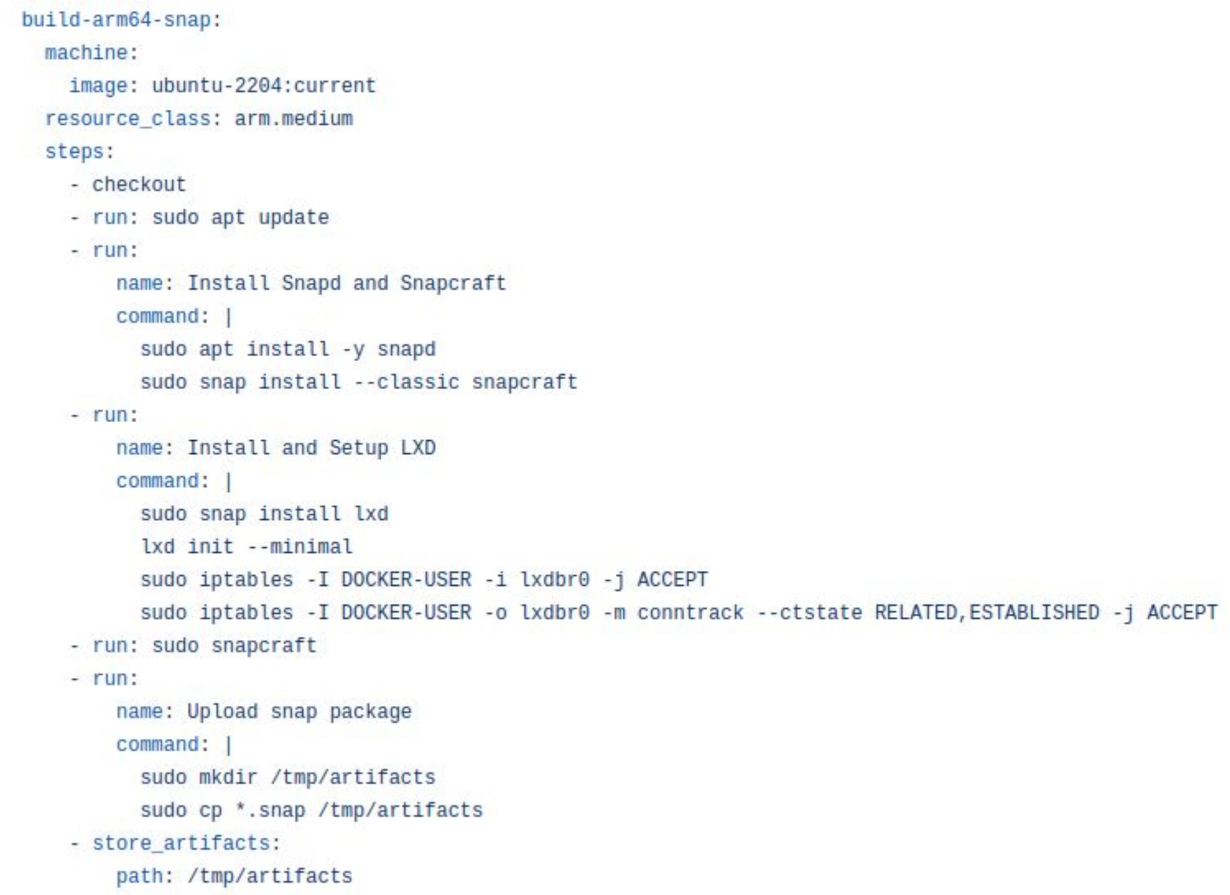
You can access the Arm VM (virtual machine) execution environment for a job by using the machine executor, specifying a Linux virtual machine image that includes Arm resources, and then specifying an Arm resource class.

```yaml
# .circleci/config.yml
jobs:
  my-job:
    machine:
      image: ubuntu-2204:2023.07.1
    resource_class: arm.medium
    steps:
      - run: uname -a
      - run: echo "Hello, Arm!"
```

Circle CI provides Linux(Ubuntu) Native ARM Runner for free, while GitHub Action not provides yet.

If you have spare ARM machine that you can use as self-hosted CI runner(for Circle CI, GitHub Actions or whatever), that would be also good idea.

Example Circle CI Config



```
build-arm64-snap:
  machine:
    image: ubuntu-2204:current
  resource_class: arm.medium
  steps:
    - checkout
    - run: sudo apt update
    - run:
        name: Install Snapd and Snapcraft
        command: |
          sudo apt install -y snapd
          sudo snap install --classic snapcraft
    - run:
        name: Install and Setup LXD
        command: |
          sudo snap install lxd
          lxd init --minimal
          sudo iptables -I DOCKER-USER -i lxdbr0 -j ACCEPT
          sudo iptables -I DOCKER-USER -o lxdbr0 -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
    - run: sudo snapcraft
    - run:
        name: Upload snap package
        command: |
          sudo mkdir /tmp/artifacts
          sudo cp *.snap /tmp/artifacts
    - store_artifacts:
        path: /tmp/artifacts
```

Setting up Ubuntu Frame and your app on your RPi with Ubuntu Core

Install and enable Ubuntu Frame daemon

```
sudo snap install ubuntu-frame
```

```
# Only if ubuntu-frame daemon not started automatically
```

```
sudo snap start ubuntu-frame
```

Setting up Ubuntu Frame and your app on your RPi with Ubuntu Core

Install kiosk app snap

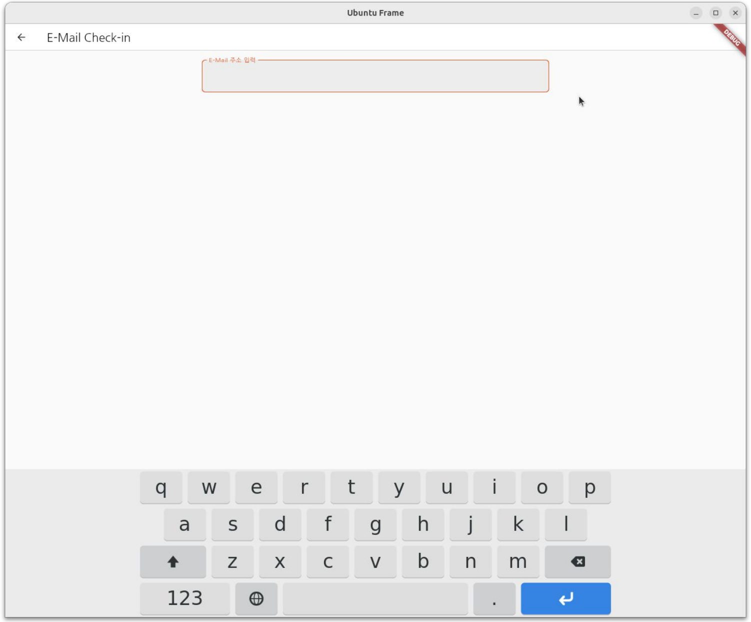
```
sudo snap install --dangerous ./<your_snap>.snap
```

Connect wayland interface then start your kiosk app daemon

```
sudo snap connect <your_snap>:wayland
```

```
sudo snap start <your_snap_daemon>
```

Ubuntu Frame On Screen Keyboard



Install OSK

```
sudo snap install ubuntu-frame-osk
```

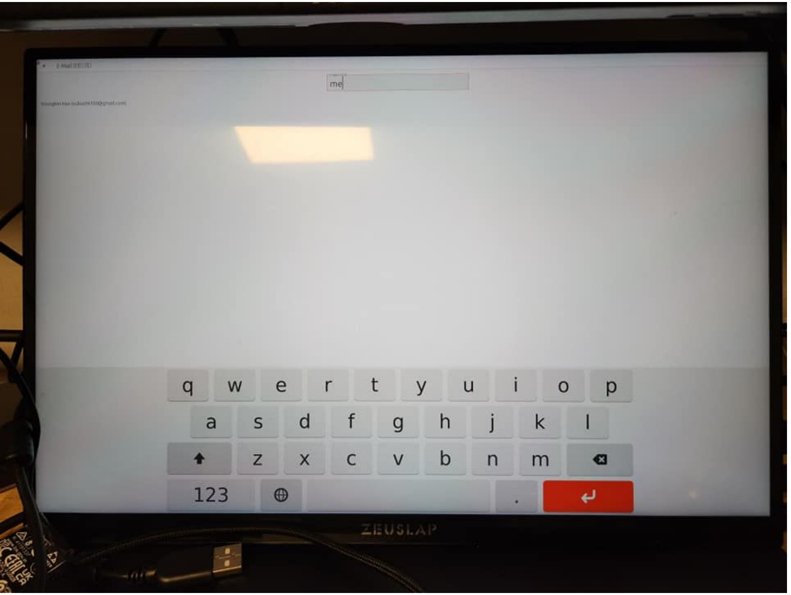
Set OSK Theme

```
sudo snap set ubuntu-frame-osk theme=light
```

Set Keyboard layout

```
sudo snap set ubuntu-frame-osk layout=us
```

Ubuntu Frame On Screen Keyboard



Install OSK

```
sudo snap install ubuntu-frame-osk
```

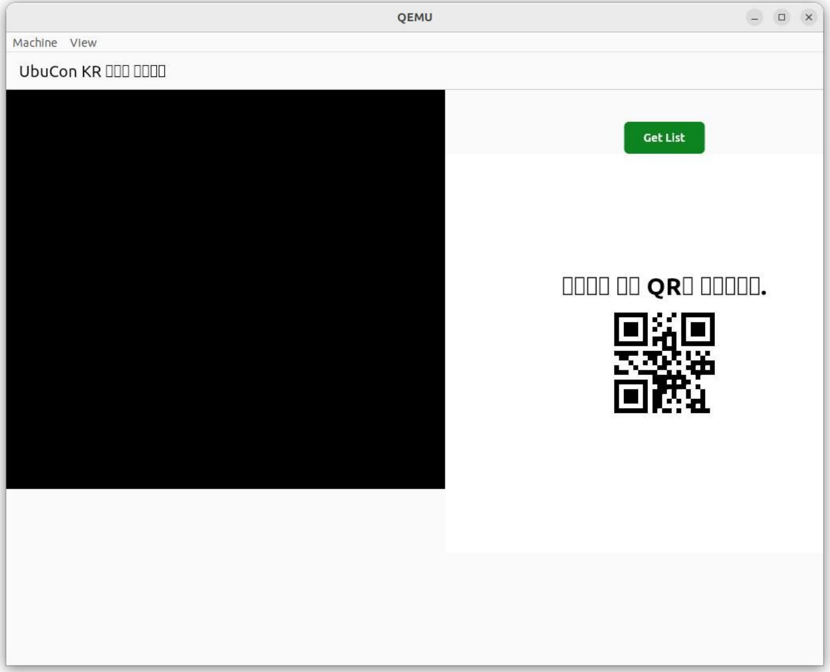
Set OSK Theme

```
sudo snap set ubuntu-frame-osk theme=light
```

Set Keyboard layout

```
sudo snap set ubuntu-frame-osk layout=us
```

Figuring out why it's not working...



UbuCon KR □□□ □□□□

□□□□ □□ QR□ □□□□□.

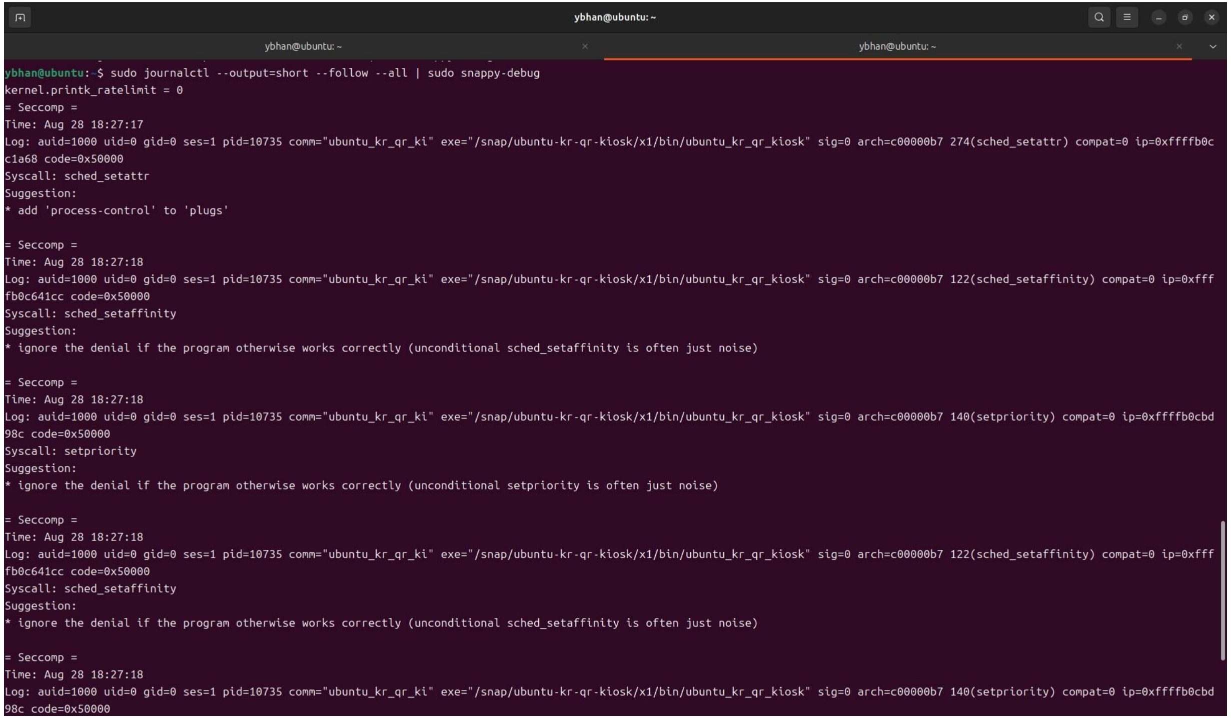
—

Debugging snap with snappy-debug

Useful for check if there’s any missing plugs for accessing resources in your snap.

```
sudo snap install
snappy-debug
```

```
sudo journalctl
--output=short --follow
--all | sudo snappy-debug
```

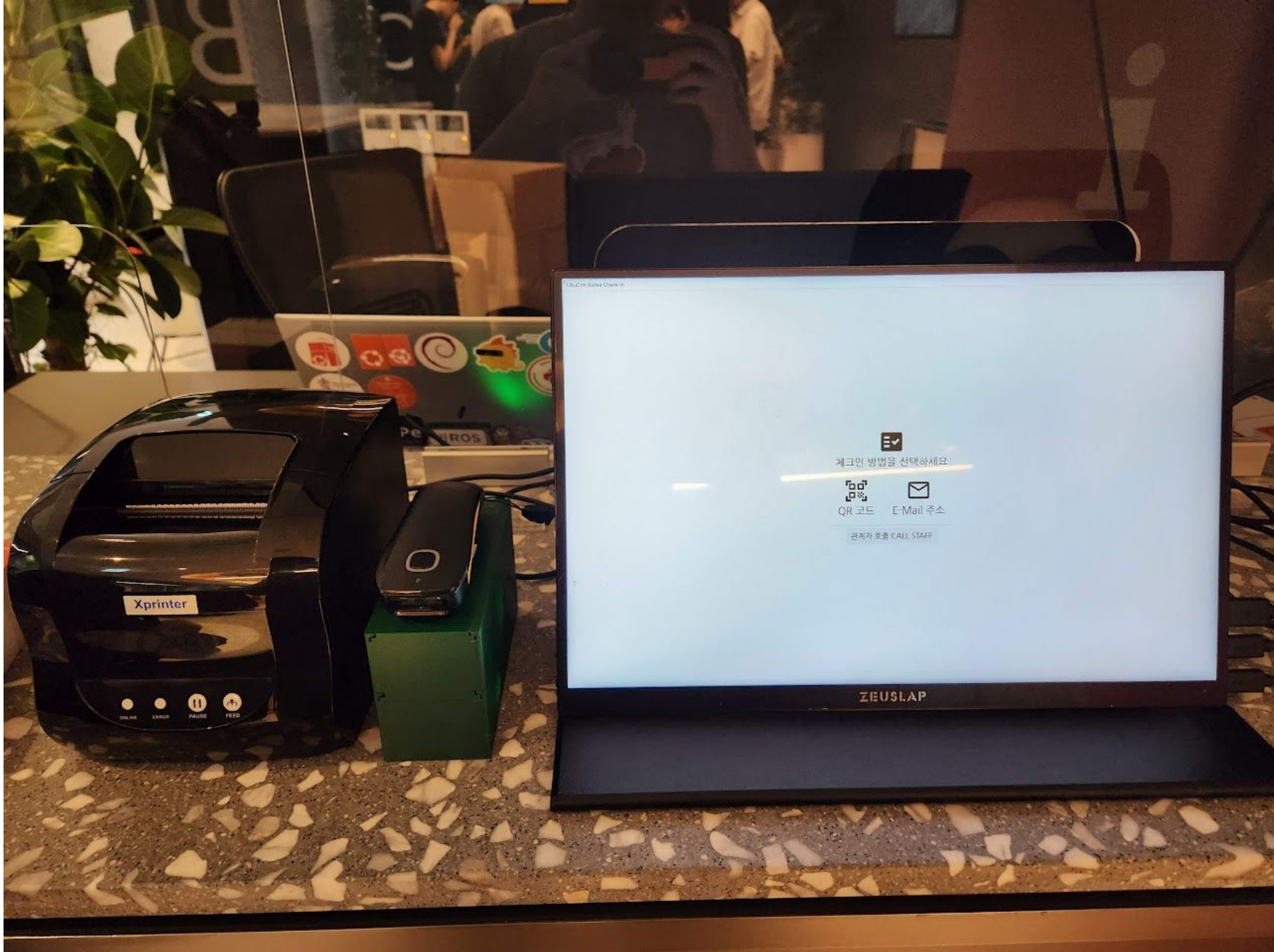


```
ybhan@ubuntu:~$ sudo journalctl --output=short --follow --all | sudo snappy-debug
kernel.printk_ratelimit = 0
= Seccomp =
Time: Aug 28 18:27:17
Log: auid=1000 uid=0 gid=0 ses=1 pid=10735 comm="ubuntu_kr_qr_ki" exe="/snap/ubuntu-kr-qr-kiosk/x1/bin/ubuntu_kr_qr_kiosk" sig=0 arch=c00000b7 274(sched_setattr) compat=0 ip=0xffffb0c
c1a68 code=0x50000
Syscall: sched_setattr
Suggestion:
* add 'process-control' to 'plugs'

= Seccomp =
Time: Aug 28 18:27:18
Log: auid=1000 uid=0 gid=0 ses=1 pid=10735 comm="ubuntu_kr_qr_ki" exe="/snap/ubuntu-kr-qr-kiosk/x1/bin/ubuntu_kr_qr_kiosk" sig=0 arch=c00000b7 122(sched_setaffinity) compat=0 ip=0xfff
fb0c641cc code=0x50000
Syscall: sched_setaffinity
Suggestion:
* ignore the denial if the program otherwise works correctly (unconditional sched_setaffinity is often just noise)

= Seccomp =
Time: Aug 28 18:27:18
Log: auid=1000 uid=0 gid=0 ses=1 pid=10735 comm="ubuntu_kr_qr_ki" exe="/snap/ubuntu-kr-qr-kiosk/x1/bin/ubuntu_kr_qr_kiosk" sig=0 arch=c00000b7 140(setpriority) compat=0 ip=0xffffb0cbd
98c code=0x50000
Syscall: setpriority
Suggestion:
* ignore the denial if the program otherwise works correctly (unconditional setpriority is often just noise)

= Seccomp =
Time: Aug 28 18:27:18
Log: auid=1000 uid=0 gid=0 ses=1 pid=10735 comm="ubuntu_kr_qr_ki" exe="/snap/ubuntu-kr-qr-kiosk/x1/bin/ubuntu_kr_qr_kiosk" sig=0 arch=c00000b7 122(sched_setaffinity) compat=0 ip=0xfff
fb0c641cc code=0x50000
Syscall: sched_setaffinity
Suggestion:
* ignore the denial if the program otherwise works correctly (unconditional sched_setaffinity is often just noise)

= Seccomp =
Time: Aug 28 18:27:18
Log: auid=1000 uid=0 gid=0 ses=1 pid=10735 comm="ubuntu_kr_qr_ki" exe="/snap/ubuntu-kr-qr-kiosk/x1/bin/ubuntu_kr_qr_kiosk" sig=0 arch=c00000b7 140(setpriority) compat=0 ip=0xffffb0cbd
98c code=0x50000
```

Things didn’t work while deadline coming in few days...

- Webcam view built with gstreamer
  - Got segment fault on RPi, Couldn’t figure out how to fix
  - Replaced with a simple text input + barcode scanner
- Flutter quick_usb plugin
  - Uses only x86 version of libusb embedded in their package making it not work on RPi.
  - Wrote a simple python http server with PyUSB as a replacement

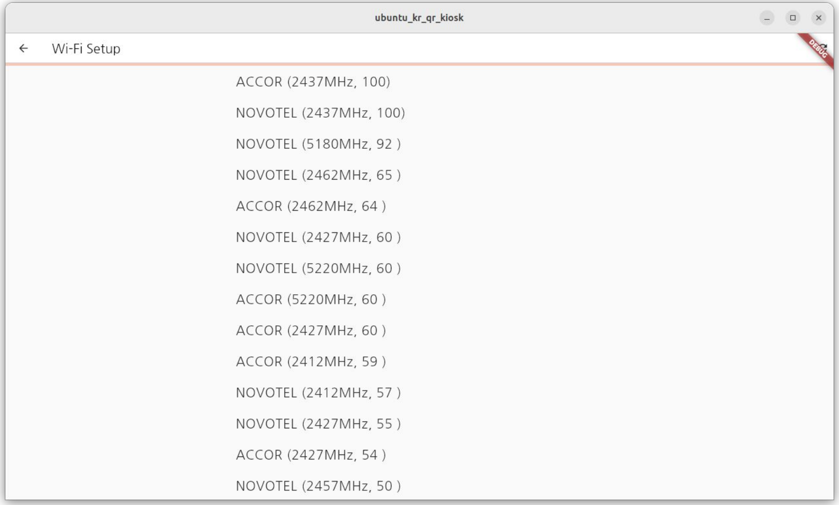
Setup Kiosk on-site!



Network connection issue on-site

- It’s not straightforward to setup network on-site :(
    - Venue setup was within few hours - Didn’t have enough time to connect to RPi remotely, setup and check network connection.
- Venue has no ethernet connection and has Wi-Fi with Captive Portal.
    - Ubuntu Core basically doesn’t have web browser that can deal with such things…
    - My workaround was to hook up RPi with my laptop then share network connection from my laptop. :(

What's improved this year (or working on to improve)



```
ACCOR (2437MHz, 100)
NOVOTEL (2437MHz, 100)
NOVOTEL (5180MHz, 92 )
NOVOTEL (2462MHz, 65 )
ACCOR (2462MHz, 64 )
NOVOTEL (2427MHz, 60 )
NOVOTEL (5220MHz, 60 )
ACCOR (5220MHz, 60 )
ACCOR (2427MHz, 60 )
ACCOR (2412MHz, 59 )
NOVOTEL (2412MHz, 57 )
NOVOTEL (2427MHz, 55 )
ACCOR (2427MHz, 54 )
NOVOTEL (2457MHz, 50 )
```

- Network connection setup UI
  - Uses nm package to interact with NetworkManager
- There are many more packages available for interacting with Linux system stack.
  - dbus, bluez, gsettings, lxd and more.

What’s improved this year (or working on to improve)

Trying out Lprint to handle label printing

- Lprint has support for Label printers with TSPL emulation.
- Printing Label with Lprint and XP-365B(Printer) didn’t work, improved printing server written in python instead. (Moved image conversion logic to the printing server)

What’s improved this year (or working on to improve)

Custom Ubuntu Core Image

- With Required Snaps already included: ubuntu-frame, ubuntu-frame-osk, mesa-2404, ubuntu-kr-qr-kiosk, network-manager
- Also with custom config and boot logo configured if possible (Custom Gadget Snap)
- Image built, but not boots. Things to fix in future

My thoughts on working with Ubuntu Frame, Ubuntu Core & Flutter

Working with Flutter on Linux is quite straightforward.

- Flutter SDK available as Snap, Flutter VSCode extension works of course.
- Many flutter packages already supports linux - But if you’re building flutter app for other then amd64(such as arm64), some plugins might not work. (Just like to quick_usb package you’ve seen today)

My thoughts on working with Ubuntu Frame, Ubuntu Core & Flutter

- Using Ubuntu Frame itself isn’t difficult.
  - It’s just a fullscreen wayland shell for displaying single app at a time. Touch input and OSK also just works.
- Building your Snap for Ubuntu Core & Ubuntu Frame would be a bit difficult if you’re trying for first time.
- Seems like there’s no easy network setup solution (something like Balena’s wifi-connect) for now. Would be nice if Ubuntu Core also have one for easy network setup - maybe someone can port?

More resources

Ubuntu Frame https://mir-server.io/ubuntu-frame

Ubuntu Core https://ubuntu.com/core

Flutter SDK Snap https://snapcraft.io/flutter

GitHub Repo
https://github.com/ubuntu-kr/ubuntu-kr-qr-kiosk

Let's see it in action

On-site demo video

https://youtu.be/Nd4mDMSv4po

Thank you!

To keep in touched with me,
visit https://youngbin.xyz
or contact me ybhan@ubuntu.com by email.