Using OSHW and OSS for building your custom hardware platform

Yet another talk about Olimex Lime2 hardware

Priit Laes @plaes

AboutPAF.com / K-Space.ee

February 2, 2020

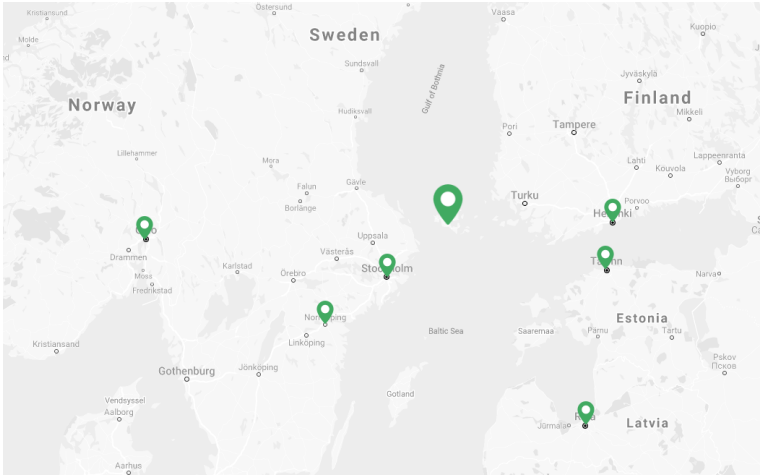
FOSDEM'20

AboutPAF.com

Ålands Penningautomatförening

AboutPAF.com

Ålands Penningautomatförening aka **PAF**



Offices:

- **Mariehamn** (Åland)
- Helsinki (FI)
- Stockholm (SE)
- Norrköping (SE)
- **Tallinn** (EE)
- Oslo (NO)
- Riga (LV)
- Madrid (SE)

AboutPAF.com

Money gambling operator, established in 1966.

AboutPAF.com

Money gambling operator, established in 1966.

1973 - **sole rights** to operate onboard Åland registered car ferries

AboutPAF.com

Money gambling operator, established in 1966.

1973 - **sole rights** to operate onboard Åland registered car ferries

1999 - online pioneers - one of the first online casinos - **PAF.com**

AboutPAF.com

Money gambling operator, established in 1966.

1973 - **sole rights** to operate onboard Åland registered car ferries

1999 - online pioneers - one of the first online casinos - **PAF.com**

Guinness World records
2012 - The largest jackpot payout in a non-pooled online slot machine game – 8,636,042 euros.
2013 - The largest jackpot payout in an online slot machine game – 17,861,813 euros.

AboutPAF.com

Money gambling operator, established in 1966.

1973 - **sole rights** to operate onboard Åland registered car ferries

1999 - online pioneers - one of the first online casinos - **PAF.com**

Guinness World records
2012 - The largest jackpot payout in a non-pooled online slot machine game – 8,636,042 euros.
2013 - The largest jackpot payout in an online slot machine game – 17,861,813 euros.

Global leader in responsible gaming

Why build your own hardware?

1973 - **sole rights** to operate onboard Åland registered car ferries

Why build your own hardware?

1973 - **sole rights** to operate onboard Åland registered car ferries
2005 - casinos on ferries in different jurisdictions on Baltic and Northern Seas

Why build your own hardware?

1973 - **sole rights** to operate onboard Åland registered car ferries
2005 - casinos on ferries in different jurisdictions on Baltic and Northern Seas

So far it has been quite a straightforward operation - buy slot machines, make sure the software has certificates required for target market and then just operate them...

Why build your own hardware?

1973 - **sole rights** to operate onboard Åland registered car ferries
2005 - casinos on ferries in different jurisdictions on Baltic and Northern Seas

So far it has been quite a straightforward operation - buy slot machines, make sure the software has certificates required for target market and then just operate them...

2008 - Estonia enacts new gambling law

Why build your own hardware?

1973 - **sole rights** to operate onboard Åland registered car ferries
2005 - casinos on ferries in different jurisdictions on Baltic and Northern Seas

So far it has been quite a straightforward operation - buy slot machines, make sure the software has certificates required for target market and then just operate them...

2008 - Estonia enacts new gambling law

- player protection (identify player and check against online database)

Why build your own hardware?

1973 - **sole rights** to operate onboard Åland registered car ferries
2005 - casinos on ferries in different jurisdictions on Baltic and Northern Seas

So far it has been quite a straightforward operation - buy slot machines, make sure the software has certificates required for target market and then just operate them...

2008 - Estonia enacts new gambling law

- player protection (identify player and check against online database)
- daily automated electronic reporting for each slot machine and table

Why build your own hardware?

1973 - **sole rights** to operate onboard Åland registered car ferries
2005 - casinos on ferries in different jurisdictions on Baltic and Northern Seas

So far it has been quite a straightforward operation - buy slot machines, make sure the software has certificates required for target market and then just operate them...

2008 - Estonia enacts new gambling law

- player protection (identify player and check against online database)
- daily automated electronic reporting for each slot machine and table

There's nothing like that in the market...

Why build your own hardware?

1973 - **sole rights** to operate onboard Åland registered car ferries
2005 - casinos on ferries in different jurisdictions on Baltic and Northern Seas

So far it has been quite a straightforward operation - buy slot machines, make sure the software has certificates required for target market and then just operate them...

2008 - Estonia enacts new gambling law

- player protection (identify player and check against online database)
- daily automated electronic reporting for each slot machine and table

There's nothing like that in the market...
...also operating on ships - limited network connectivity

Buy the solution

A solution was designed by third party based on Intel Atom based industrial PCs

Buy the solution

A solution was designed by third party based on Intel Atom based industrial PCs

Figure: Gen. 1 of TCD Hardware (2009?)

Buy the solution

A solution was designed by third party based on Intel Atom based industrial PCs

Figure: Gen. 1 of TCD Hardware (2009?)

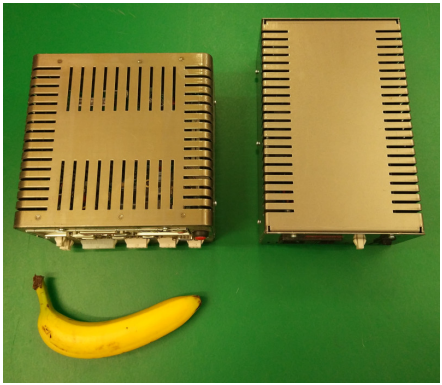


Figure: Gen. 2 of TCD Hardware (2015?)

Auditing the existing solution

Hired by PAF in 2018 to take a look of existing hardware-software solution (also including the whole automated meter reporting backend) and improve it.

Auditing the existing solution

Hired by PAF in 2018 to take a look of existing hardware-software solution (also including the whole automated meter reporting backend) and improve it.

- Embedded software actually written in Java, storing data in local MySQL server and communicating remotely using HTTP

Auditing the existing solution

Hired by PAF in 2018 to take a look of existing hardware-software solution (also including the whole automated meter reporting backend) and improve it.

- Embedded software actually written in Java, storing data in local MySQL server and communicating remotely using HTTP
- Also has graphical interface acting on events and showing animations (1024x800 gifs)

Auditing the existing solution

Hired by PAF in 2018 to take a look of existing hardware-software solution (also including the whole automated meter reporting backend) and improve it.

- Embedded software actually written in Java, storing data in local MySQL server and communicating remotely using HTTP
- Also has graphical interface acting on events and showing animations (1024x800 gifs)
- Boxes still running Ubuntu 12.04

Auditing the existing solution

Hired by PAF in 2018 to take a look of existing hardware-software solution (also including the whole automated meter reporting backend) and improve it.

- Embedded software actually written in Java, storing data in local MySQL server and communicating remotely using HTTP
- Also has graphical interface acting on events and showing animations (1024x800 gifs)
- Boxes still running Ubuntu 12.04
- Intel Atom D2500 CPU

Auditing the existing solution

Hired by PAF in 2018 to take a look of existing hardware-software solution (also including the whole automated meter reporting backend) and improve it.

- Embedded software actually written in Java, storing data in local MySQL server and communicating remotely using HTTP
- Also has graphical interface acting on events and showing animations (1024x800 gifs)
- Boxes still running Ubuntu 12.04
- Intel Atom D2500 CPU

Initial plan: reuse existing hardware. Started with a clean install of latest Ubuntu LTS.

Auditing the existing solution

Hired by PAF in 2018 to take a look of existing hardware-software solution (also including the whole automated meter reporting backend) and improve it.

- Embedded software actually written in Java, storing data in local MySQL server and communicating remotely using HTTP
- Also has graphical interface acting on events and showing animations (1024x800 gifs)
- Boxes still running Ubuntu 12.04
- Intel Atom D2500 CPU

Initial plan: reuse existing hardware. Started with a clean install of latest Ubuntu LTS.
Result: Ubuntu installed, but no graphics driver for **Intel GMA 3600**.

Auditing the existing solution

Hired by PAF in 2018 to take a look of existing hardware-software solution (also including the whole automated meter reporting backend) and improve it.

- Embedded software actually written in Java, storing data in local MySQL server and communicating remotely using HTTP
- Also has graphical interface acting on events and showing animations (1024x800 gifs)
- Boxes still running Ubuntu 12.04
- Intel Atom D2500 CPU

Initial plan: reuse existing hardware. Started with a clean install of latest Ubuntu LTS.
Result: Ubuntu installed, but no graphics driver for **Intel GMA 3600**.

Imagination Technologies PowerVR SGX 545

Auditing the existing solution

Hired by PAF in 2018 to take a look of existing hardware-software solution (also including the whole automated meter reporting backend) and improve it.

- Embedded software actually written in Java, storing data in local MySQL server and communicating remotely using HTTP
- Also has graphical interface acting on events and showing animations (1024x800 gifs)
- Boxes still running Ubuntu 12.04
- Intel Atom D2500 CPU

Initial plan: reuse existing hardware. Started with a clean install of latest Ubuntu LTS.
Result: Ubuntu installed, but no graphics driver for **Intel GMA 3600**.

Imagination Technologies PowerVR SGX 545

Nope - Not gonna happen.. :(

Let's go shopping

Our requirements for "slot" computer:

- Small form factor
- HDMI output
- USB
- Bunch of GPIOs
- Ethernet
- Proper storage (eMMC)
- Mainline u-boot / Linux
- Long term availability
- Cheap

Let's go shopping

Our requirements for "slot" computer:

- Small form factor
- HDMI output
- USB
- Bunch of GPIOs
- Ethernet
- Proper storage (eMMC)
- Mainline u-boot / Linux
- Long term availability
- Cheap

Designing a shield for Lime2

- Install KiCAD

Designing a shield for Lime2

- Install KiCAD
- Draw schematic and select components

Designing a shield for Lime2

- Install KiCAD
- Draw schematic and select components
- Design board

Designing a shield for Lime2

- Install KiCAD
- Draw schematic and select components
- Design board
- Use `A20_OLinuXino_Lime2_SHIELD_TEMPLATE` from github:Olimex/OLINUXINO

Designing a shield for Lime2

- Install KiCAD
- Draw schematic and select components
- Design board
- Use `A20_OLinuXino_Lime2_SHIELD_TEMPLATE` from github:Olimex/OLINUXINO
- ... No mounting holes?

Designing a shield for Lime2

- Install KiCAD
- Draw schematic and select components
- Design board
- Use `A20_OLinuXino_Lime2_SHIELD_TEMPLATE` from github:Olimex/OLINUXINO
- ... No mounting holes?
- No problem, let's open up the board file instead

Designing a shield for Lime2

- Install KiCAD
- Draw schematic and select components
- Design board
- Use `A20_OLinuXino_Lime2_SHIELD_TEMPLATE` from github:Olimex/OLINUXINO
- ... No mounting holes?
- No problem, let's open up the board file instead... Wait.. It's Eagle?

Designing a shield for Lime2

- Install KiCAD
- Draw schematic and select components
- Design board
- Use `A20_OLinuXino_Lime2_SHIELD_TEMPLATE` from github:Olimex/OLINUXINO
- ... No mounting holes?
- No problem, let's open up the board file instead... Wait.. It's Eagle?
- No problem, let's just use KiCAD's Eagle import

Designing a shield for Lime2

- Install KiCAD
- Draw schematic and select components
- Design board
- Use `A20_OLinuXino_Lime2_SHIELD_TEMPLATE` from github:Olimex/OLINUXINO
- ... No mounting holes?
- No problem, let's open up the board file instead... Wait.. It's Eagle?
- No problem, let's just use KiCAD's Eagle import... Wait.. It's old binary Eagle format?

Designing a shield for Lime2

- Install KiCAD
- Draw schematic and select components
- Design board
- Use `A20_OLinuXino_Lime2_SHIELD_TEMPLATE` from github:Olimex/OLINUXINO
- ... No mounting holes?
- No problem, let's open up the board file instead... Wait.. It's Eagle?
- No problem, let's just use KiCAD's Eagle import... Wait.. It's old binary Eagle format?
- Install Eagle and figure out which layers to export...

Designing a shield for Lime2

- Install KiCAD
- Draw schematic and select components
- Design board
- Use `A20_OLinuXino_Lime2_SHIELD_TEMPLATE` from github:Olimex/OLINUXINO
- ... No mounting holes?
- No problem, let's open up the board file instead... Wait.. It's Eagle?
- No problem, let's just use KiCAD's Eagle import... Wait.. It's old binary Eagle format?
- Install Eagle and figure out which layers to export...
- In case anyone else needs that, the layers are: `Dimension, tKeepout, Drills, tDocu`

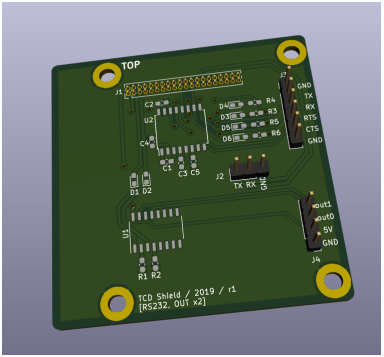
Designing a shield for Lime2

- Install KiCAD
- Draw schematic and select components
- Design board
- Use `A20_OLinuXino_Lime2_SHIELD_TEMPLATE` from github:Olimex/OLINUXINO
- ... No mounting holes?
- No problem, let's open up the board file instead... Wait.. It's Eagle?
- No problem, let's just use KiCAD's Eagle import... Wait.. It's old binary Eagle format?
- Install Eagle and figure out which layers to export...
- In case anyone else needs that, the layers are: `Dimension, tKeepout, Drills, tDocu`
- Import the `.dxf` into the template layout and position it

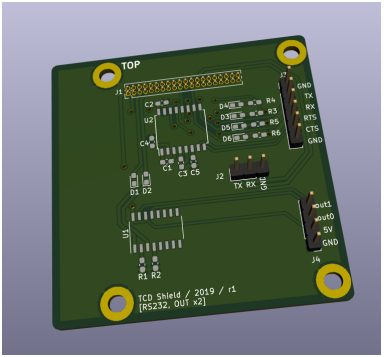
Designing a shield for Lime2

- Install KiCAD
- Draw schematic and select components
- Design board
- Use `A20_OLinuXino_Lime2_SHIELD_TEMPLATE` from github:Olimex/OLINUXINO
- ... No mounting holes?
- No problem, let's open up the board file instead... Wait.. It's Eagle?
- No problem, let's just use KiCAD's Eagle import... Wait.. It's old binary Eagle format?
- Install Eagle and figure out which layers to export...
- In case anyone else needs that, the layers are: `Dimension, tKeepout, Drills, tDocu`
- Import the `.dxf` into the template layout and position it
- Finish layout

Almost ready?

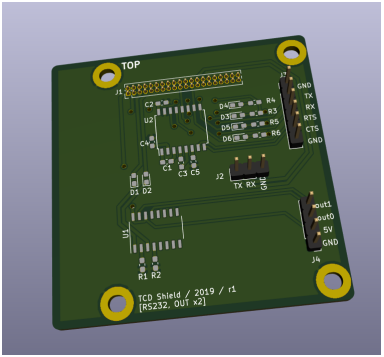


Almost ready?



Are we there yet?

Almost ready?



Are we there yet?

- Build (or order) the board
- Order components
- Solder it together
- Test it

K-Space.ee

Hackerspace in Tallinn, Estonia

Community driven co-working and meeting space.

K-Space.ee

Hackerspace in Tallinn, Estonia

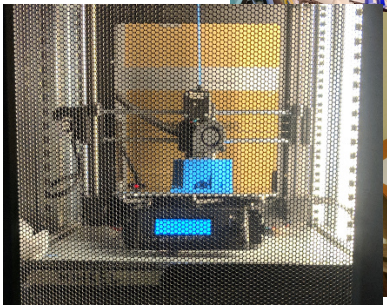
Community driven co-working and meeting space.

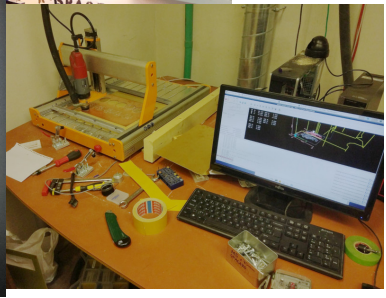
Basic fabrication capabilities

K-Space.ee

Hackerspace in Tallinn, Estonia

Community driven co-working and meeting space.





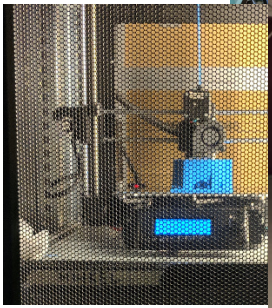
Basic fabrication capabilities

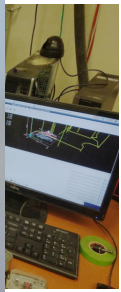
- 3D printer
- **CNC machine**
- Laser cutter
- **SMD reflow oven**

K-Space.ee

Hackerspace in Tallinn, Estonia

Community driven co-working and meeting space.





Basic fabrication capabilities

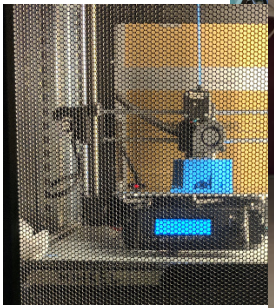
- 3D printer
- **CNC machine**
- Laser cutter
- **SMD reflow oven**

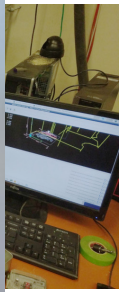
Our own server room with bunch of full-size racks.

K-Space.ee

Hackerspace in Tallinn, Estonia

Community driven co-working and meeting space.





Basic fabrication capabilities

- 3D printer
- **CNC machine**
- Laser cutter
- **SMD reflow oven**

Our own server room with bunch of full-size racks.

Also home to **Armbian.com** :)

Milling your own PCB?

Howto steps for the K-Space CNC in our wiki:

- Export front, back, drill and outline Gerbers from your favourite PCB design software
- Convert Gerbers to G-code using pcb2gcode command below (on next slide)
- Dump the files to the K-Space Nextcloud share
- Approach the CNC setup, in web browser open bookmarked link for the same share, download files
- Under supervision by **Lauri**, **Kaarel** or **Silver**: mill front, drill holes, flip, mill back, cut outline

Generating G-code using pcb2gcode (1)

KiCAD already generated our gerbers, so let's turn it something that CNC can eat..

Generating G-code using pcb2gcode (1)

KiCAD already generated our gerbers, so let's turn it something that CNC can eat..

```
git clone https://github.com/pcb2gcode/pcb2gcode/
cd pcb2gcode
# Pull usable version
git checkout eeee27db62b6b447f84d020cd80a65a81daa54b1
apt install libboost-all-dev libgtkmm-2.4-dev gerbv shtool autogen
autoreconf -fv
./configure --prefix=$HOME/opt
make -j4 && make install
```

Generating G-code using pcb2gcode (2)

```
pcb2gcode --vectorial \
  --software linuxcnc --zero-start --tile-x 3 --tile-y 2 \
  --front *-F.Cu.g* --front-output front.ngc \
  --back *-B.Cu.g* --back-output back.ngc \
  --drill *.drl --drill-output drill.ngc --drill-side back \
  --outline *-Edge.Cuts.g* --outline-output cutout.ngc \
  --metric --metricoutput --noconfigfile \
  --zsafe 1 --zchange 100 \
  --cut-feed 150 --cut-speed 6000 --cut-infeed 0.6 --zcut -1.5 \
  --zbridges -1 --bridges 3 --bridgesnum 4 --cutter-diameter 2 \
  --mill-feed 500 --mill-speed 6000 --zwork -0.2 --offset 0.2 \
  --drill-feed 500 --drill-speed 6000 --zdrill -3 \
  --spindown-time 2 --spinup-time 2
```

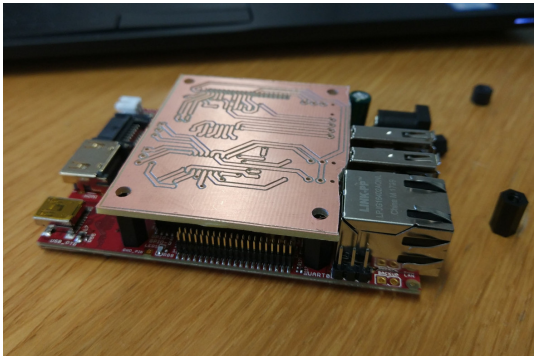


Figure: First fit!

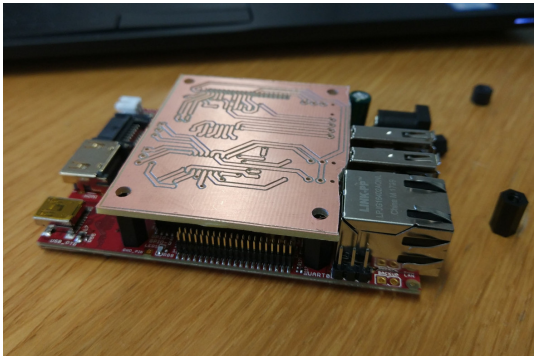


Figure: First fit!

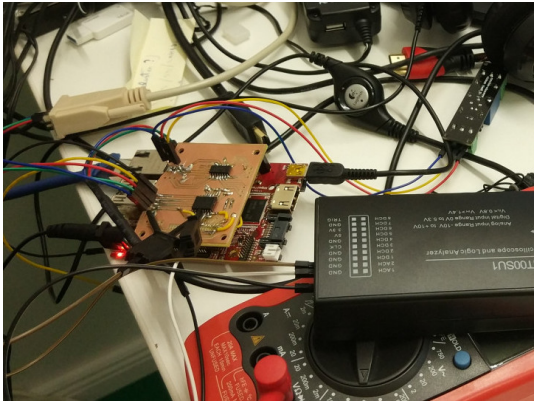


Figure: Why does it not work?!

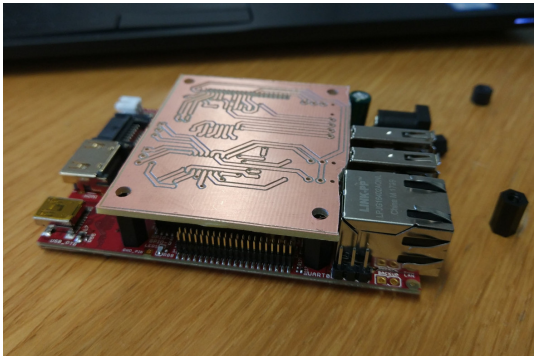


Figure: First fit!

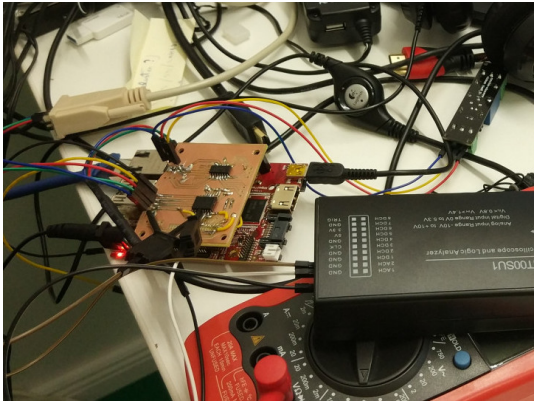
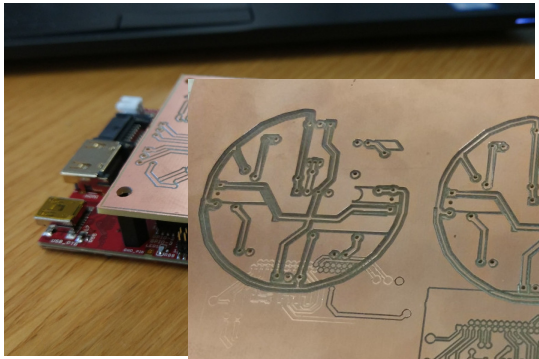
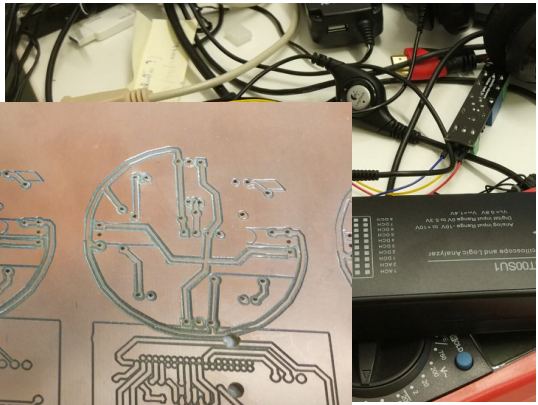


Figure: Why does it not work?!

Conclusion: CNC is too much work.. Let's try the fabs.





t work?!

PCBs from a fab house?

Buying locally vs from China?

PCBs from a fab house?

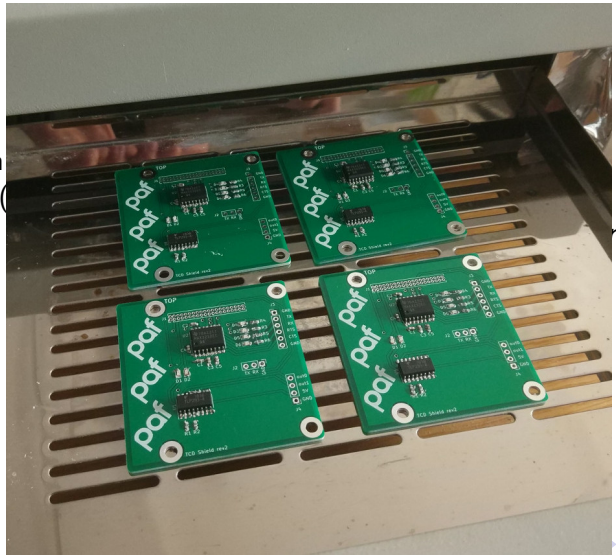
Buying locally vs from China?
3 weeks vs 1.5 weeks (with shipping from China).

PCBs from a fab house?

Buying locally vs from China?
3 weeks vs 1.5 weeks (with shipping from China).
Not going to mention the cost (even when using *"insert brandnames here"* for shipping.

PCBs from a fab house?

Buying locally vs from
3 weeks vs 1.5 weeks (
Not going to mention re" for shipping.



Software platform (1)

- Olimex Lime2-eMMC running Debian stable (aka Ticket Check Device)
  - Somewhat custom kernel and bootloader setup
  - We build our own `.deb` from mainline kernel (with some extra configuration options)
  - Our own software on devices is also packaged as `.deb` packages

Software platform (1)

- Olimex Lime2-eMMC running Debian stable (aka Ticket Check Device)
  - Somewhat custom kernel and bootloader setup
  - We build our own `.deb` from mainline kernel (with some extra configuration options)
  - Our own software on devices is also packaged as `.deb` packages
  - Applications written in Python: Gtk+ and Gstreamer

Software platform (1)

- Olimex Lime2-eMMC running Debian stable (aka Ticket Check Device)
  - Somewhat custom kernel and bootloader setup
  - We build our own `.deb` from mainline kernel (with some extra configuration options)
  - Our own software on devices is also packaged as `.deb` packages
  - Applications written in Python: Gtk+ and Gstreamer
  - Light media player on ships not under Estonian flag - no interaction with slot machine

Software platform (1)

- Olimex Lime2-eMMC running Debian stable (aka Ticket Check Device)
  - Somewhat custom kernel and bootloader setup
  - We build our own `.deb` from mainline kernel (with some extra configuration options)
  - Our own software on devices is also packaged as `.deb` packages
  - Applications written in Python: Gtk+ and Gstreamer
  - Light media player on ships not under Estonian flag - no interaction with slot machine
  - Ticket check application - interacting with slot machine (clicking a single relay)
    ...requires some udev GPIO magic

Software platform (1)

- Olimex Lime2-eMMC running Debian stable (aka Ticket Check Device)
  - Somewhat custom kernel and bootloader setup
  - We build our own `.deb` from mainline kernel (with some extra configuration options)
  - Our own software on devices is also packaged as `.deb` packages
  - Applications written in Python: Gtk+ and Gstreamer
  - Light media player on ships not under Estonian flag - no interaction with slot machine
  - Ticket check application - interacting with slot machine (clicking a single relay)
    ...requires some udev GPIO magic
  - Ticket check device for tables - Lime2 with I2C display for croupiers
    ...using devicetree overlays

Software platform (2)

- Central server on each ship (either VM or physical host)

Software platform (2)

- Central server on each ship (either VM or physical host)
  - Acts as a "gateway" with API translation layers - same gateway API for all the ships

Software platform (2)

- Central server on each ship (either VM or physical host)
  - Acts as a "gateway" with API translation layers - same gateway API for all the ships
  - Translates queries towards ship passenger database and compares results against block list

Software platform (2)

- Central server on each ship (either VM or physical host)
  - Acts as a "gateway" with API translation layers - same gateway API for all the ships
  - Translates queries towards ship passenger database and compares results against block list
  - Hosts a "hashed" version of gambling block list - no sensitive data on Lime2 devices

Software platform (2)

- Central server on each ship (either VM or physical host)
  - Acts as a "gateway" with API translation layers - same gateway API for all the ships
  - Translates queries towards ship passenger database and compares results against block list
  - Hosts a "hashed" version of gambling block list - no sensitive data on Lime2 devices
  - Central host to look up Lime2 devices on board (using Avahi for lookup)

Software platform (2)

- Central server on each ship (either VM or physical host)
  - Acts as a "gateway" with API translation layers - same gateway API for all the ships
  - Translates queries towards ship passenger database and compares results against block list
  - Hosts a "hashed" version of gambling block list - no sensitive data on Lime2 devices
  - Central host to look up Lime2 devices on board (using Avahi for lookup)
  - WIP: Hosts our own APT repository (Aptly)
  - WIP: Log collection and aggregation (rsyslog)
  - TODO: Proper monitoring... (MQTT maybe?)

Software platform (3)

Provisioning Lime2 devices in two minutes

- Prerequisites (`sunxi-fel` and `fastboot`):
  - `u-boot` image
  - Basic Debian image `325MB` with ssh keys and avahi service
  - Image for `ESP` partition containing boot scripts for u-boot

Software platform (3)

Provisioning Lime2 devices in two minutes

- Prerequisites (`sunxi-fel` and `fastboot`):
  - `u-boot` image
  - Basic Debian image `325MB` with ssh keys and avahi service
  - Image for `ESP` partition containing boot scripts for u-boot
- Preparation tasks (collect MAC address and format eMMC):
  - `sunxi-fel uboot $DATA/u-boot-sunxi-with-spl.bin write 0x43100000 $DATA/env.txt`
  - `echo $(fastboot getvar uboot:ethaddr 2>&1|head -n 1 |cut -f 3- -d ':' )`
  - `fastboot oem format && fastboot reboot`

Software platform (3)

Provisioning Lime2 devices in two minutes

- Prerequisites (`sunxi-fel` and `fastboot`):
  - `u-boot` image
  - Basic Debian image `325MB` with ssh keys and avahi service
  - Image for `ESP` partition containing boot scripts for u-boot
- Preparation tasks (collect MAC address and format eMMC):
  - `sunxi-fel uboot $DATA/u-boot-sunxi-with-spl.bin write 0x43100000 $DATA/env.txt`
  - `echo $(fastboot getvar uboot:ethaddr 2>&1|head -n 1 |cut -f 3- -d ':' )`
  - `fastboot oem format && fastboot reboot`
- And now the flashing process:
  - `sunxi-fel spiflash-write 0 $DATA/u-boot-sunxi-with-spl.bin`
  - `sunxi-fel uboot $DATA/u-boot-sunxi-with-spl.bin write 0x43100000 $DATA/env.txt`
  - `fastboot flash esp $DATA/esp.img`
  - `fastboot flash system $DATA/tcd-base-debian-buster.img`

Software platform (3)

Provisioning Lime2 devices in two minutes

- Prerequisites (`sunxi-fel` and `fastboot`):
  - `u-boot` image
  - Basic Debian image `325MB` with ssh keys and avahi service
  - Image for `ESP` partition containing boot scripts for u-boot
- Preparation tasks (collect MAC address and format eMMC):
  - `sunxi-fel uboot $DATA/u-boot-sunxi-with-spl.bin write 0x43100000 $DATA/env.txt`
  - `echo $(fastboot getvar uboot:ethaddr 2>&1|head -n 1 |cut -f 3- -d ':' )`
  - `fastboot oem format && fastboot reboot`
- And now the flashing process:
  - `sunxi-fel spiflash-write 0 $DATA/u-boot-sunxi-with-spl.bin`
  - `sunxi-fel uboot $DATA/u-boot-sunxi-with-spl.bin write 0x43100000 $DATA/env.txt`
  - `fastboot flash esp $DATA/esp.img`
  - `fastboot flash system $DATA/tcd-base-debian-buster.img`
- And now we can continue with `ansible-playbook`

Software platform (4)

Ansible playbooks for post-flash (we can run these in parallel):

Software platform (4)

Ansible playbooks for post-flash (we can run these in parallel):

- bootstrap
  - resize root file system
  - regenerate ssh host keys
  - "fix" the hostname

Software platform (4)

Ansible playbooks for post-flash (we can run these in parallel):

- bootstrap
  - resize root file system
  - regenerate ssh host keys
  - "fix" the hostname
- Run ship-specific setup:
  - Set up Xorg
  - Set up service user
  - Install required packages (our own TCD application)

Software platform (4)

Ansible playbooks for post-flash (we can run these in parallel):

- bootstrap
  - resize root file system
  - regenerate ssh host keys
  - "fix" the hostname
- Run ship-specific setup:
  - Set up Xorg
  - Set up service user
  - Install required packages (our own TCD application)
- Update the static DNS record in `/etc/hosts` with correct IP for ship gateway

Software platform (4)

Ansible playbooks for post-flash (we can run these in parallel):

- bootstrap
  - resize root file system
  - regenerate ssh host keys
  - "fix" the hostname
- Run ship-specific setup:
  - Set up Xorg
  - Set up service user
  - Install required packages (our own TCD application)
- Update the static DNS record in `/etc/hosts` with correct IP for ship gateway
- Update the avahi service txt-record with the correct slot location

Software platform (5)

How do we know the location of the devices?

Each device advertises its location using avahi service record:

```
$ cat /etc/avahi/services/paf-ssh.service
<?xml version="1.0" standalone='no'?><!--*-nxml-*-->
<!DOCTYPE service-group SYSTEM "avahi-service.dtd">
<service-group>
<name replace-wildcards="yes">PAF ssh %h</name>
  <service>
    <type>_ssh._tcp</type>
    <port>22</port>
    <txt-record>slot=unknown</txt-record>
  </service>
</service-group>
```

Software platform (5)

How do we know the location of the devices?

Each device advertises its location using avahi service record:

```
$ cat /etc/avahi/services/paf-ssh.service
<?xml version="1.0" standalone='no'?><!--*-nxml-*-->
<!DOCTYPE service-group SYSTEM "avahi-service.dtd">
<service-group>
<name replace-wildcards="yes">PAF ssh %h</name>
  <service>
    <type>_ssh._tcp</type>
    <port>22</port>
    <txt-record>slot=unknown</txt-record>
  </service>
</service-group>
```

```
$ avahi-browse -d local _ssh._tcp --resolve -t -p |grep slot
```

Bonus: Hardware picture

Figure: Gen. 1 of TCD Hardware (2009?)

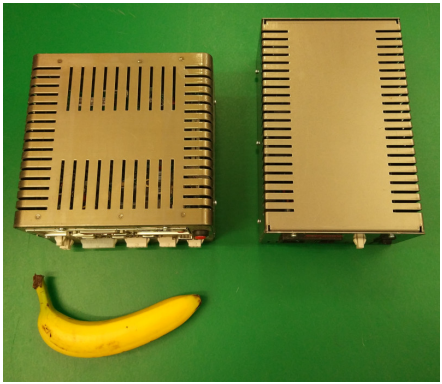


Figure: Gen. 2 of TCD Hardware (2015?)

Bonus: Hardware picture

Figure: Gen. 1 of TCD Hardware (2009?)

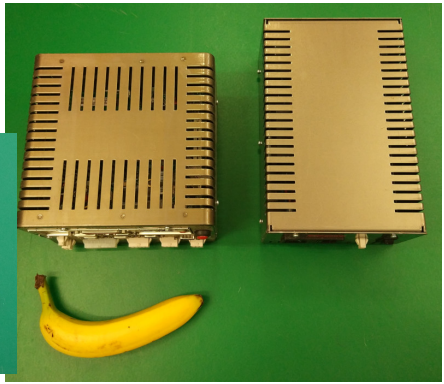


Figure: Gen. 2 of TCD Hardware (2015?)

PAF In-House TCD Hardware

Overview of problems encountered and solved

GPIO pins not working?

Symptom: GPIO pin not being able to push out `3.3V` to trigger relay.

GPIO pins not working?

Symptom: GPIO pin not being able to push out 3.3V to trigger relay.
Cause: Missing regulators for GPIO banks:

```
+++ b/arch/arm/boot/dts/sun7i-a20-olinuxino-lime2.dts
 &pio {
+       vcc-pa-supply = <&reg_vcc3v3>;
+       vcc-pc-supply = <&reg_vcc3v3>;
+       vcc-pe-supply = <&reg_ldo3>;
+       vcc-pf-supply = <&reg_vcc3v3>;
+       vcc-pg-supply = <&reg_ldo4>;
+
        led_pins_olinuxinolime: led-pins {
                pins = "PH2";
                function = "gpio_out";
```

Failure to boot from eMMC

Symptom: Occasional boot failures from eMMC

Failure to boot from eMMC

Symptom: Occasional boot failures from eMMC

Path to solution:

- Initial testing and setups were done on Olimex Lime2 `rev.G2` boards.

Failure to boot from eMMC

Symptom: Occasional boot failures from eMMC

Path to solution:

- Initial testing and setups were done on Olimex Lime2 `rev.G2` boards.
- Second batch for deployment (around 35 devices for prodution) were `rev.K`

Failure to boot from eMMC

Symptom: Occasional boot failures from eMMC

Path to solution:

- Initial testing and setups were done on Olimex Lime2 `rev.G2` boards.
- Second batch for deployment (around 35 devices for prodution) were `rev.K`
- We cut some corners and ship devices running Armbian from sdcard

Failure to boot from eMMC

Symptom: Occasional boot failures from eMMC

Path to solution:

- Initial testing and setups were done on Olimex Lime2 `rev.G2` boards.
- Second batch for deployment (around 35 devices for prodution) were `rev.K`
- We cut some corners and ship devices running Armbian from sdcard

Cause: eMMC chip was changed - Olimex agrees to fix boards

Failure to boot from eMMC

Symptom: Occasional boot failures from eMMC

Path to solution:

- Initial testing and setups were done on Olimex Lime2 `rev.G2` boards.
- Second batch for deployment (around 35 devices for prodution) were `rev.K`
- We cut some corners and ship devices running Armbian from sdcard

Cause: eMMC chip was changed - Olimex agrees to fix boards

- Problem: devices have been already deployed

Failure to boot from eMMC

Symptom: Occasional boot failures from eMMC

Path to solution:

- Initial testing and setups were done on Olimex Lime2 `rev.G2` boards.
- Second batch for deployment (around 35 devices for prodution) were `rev.K`
- We cut some corners and ship devices running Armbian from sdcard

Cause: eMMC chip was changed - Olimex agrees to fix boards

- Problem: devices have been already deployed
- Order second batch of devices - test and find more boot issues (1 out of 8 boots fails)

Failure to boot from eMMC

Symptom: Occasional boot failures from eMMC

Path to solution:

- Initial testing and setups were done on Olimex Lime2 `rev.G2` boards.
- Second batch for deployment (around 35 devices for prodution) were `rev.K`
- We cut some corners and ship devices running Armbian from sdcard

Cause: eMMC chip was changed - Olimex agrees to fix boards

- Problem: devices have been already deployed
- Order second batch of devices - test and find more boot issues (1 out of 8 boots fails)
- Olimex aknowledges the second issue, adds SPI eeprom and we send our boards back

Failure to boot from eMMC

Symptom: Occasional boot failures from eMMC

Path to solution:

- Initial testing and setups were done on Olimex Lime2 `rev.G2` boards.
- Second batch for deployment (around 35 devices for prodution) were `rev.K`
- We cut some corners and ship devices running Armbian from sdcard

Cause: eMMC chip was changed - Olimex agrees to fix boards

- Problem: devices have been already deployed
- Order second batch of devices - test and find more boot issues (1 out of 8 boots fails)
- Olimex aknowledges the second issue, adds SPI eeprom and we send our boards back
- Lots of device shuffling from our side

Device hangs after reboot in bootloader

Symptom: Device hangs in bootloader after reboot

Device hangs after reboot in bootloader

Symptom: Device hangs in bootloader after reboot
Cause: Wrongly sized capacitors in the power supply section.

Device hangs after reboot in bootloader

Symptom: Device hangs in bootloader after reboot
Cause: Wrongly sized capacitors in the power supply section.
Solution: Patchset from Olliver Schinagl (Ultimaker) that *had not been upstreamed yet*.

```
[PATCH v3 0/9] Stop AXP from crashing when enabling LDO3
...
The root cause is that some boards have too high capacitance on the LDO3
output port causing inrush currents exceeding the maximum of the AXP209.
...
```

Device hangs after reboot in bootloader

Symptom: Device hangs in bootloader after reboot
Cause: Wrongly sized capacitors in the power supply section.
Solution: Patchset from Olliver Schinagl (Ultimaker) that *had not been upstreamed yet*.

```
[PATCH v3 0/9] Stop AXP from crashing when enabling LDO3
...
The root cause is that some boards have too high capacitance on the LDO3
output port causing inrush currents exceeding the maximum of the AXP209.
...
```

Please submit your stuff upstream!

Display issues (1)

Symptom: HDMI display not working in u-boot

Display issues (1)

Symptom: HDMI display not working in u-boot
Cause: Display's HDMI hotplug detect pin not connected properly.

Display issues (1)

Symptom: HDMI display not working in u-boot
Cause: Display's HDMI hotplug detect pin not connected properly.
Workaround: Force display always on from kernel commandline and hardcode EDID data.

Display issues (1)

Symptom: HDMI display not working in u-boot
Cause: Display's HDMI hotplug detect pin not connected properly.
Workaround: Force display always on from kernel commandline and hardcode EDID data.
Solution: Always poll DDC bus for EDID data.

Display issues (1)

Symptom: HDMI display not working in u-boot
Cause: Display's HDMI hotplug detect pin not connected properly.
Workaround: Force display always on from kernel commandline and hardcode EDID data.
Solution: Always poll DDC bus for EDID data.

u-boot: Patch accepted in together with some EDID relaxation checks.

Display issues (1)

Symptom: HDMI display not working in u-boot
Cause: Display's HDMI hotplug detect pin not connected properly.
Workaround: Force display always on from kernel commandline and hardcode EDID data.
Solution: Always poll DDC bus for EDID data.

u-boot: Patch accepted in together with some EDID relaxation checks.

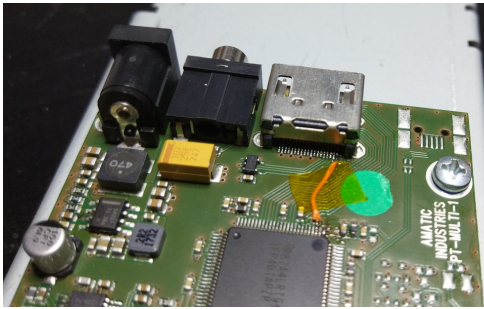
Linux kernel: Patch rejected with: *Fix your hardware!*

Display issues (1)

Symptom: HDMI display not working in u-boot
Cause: Display's HDMI hotplug detect pin not connected properly.
Workaround: Force display always on from kernel commandline and hardcode EDID data.
Solution: Always poll DDC bus for EDID data.

u-boot: Patch accepted in together with some EDID relaxation checks.

Linux kernel: Patch rejected with: *Fix your hardware!*



Display issues (2)

Symptom: Display does not turn on when kernel takes over from bootloader (1 in 8 boots)

Display issues (2)

Symptom: Display does not turn on when kernel takes over from bootloader (1 in 8 boots)
Workaround: Force display always on from kernel commandline.

Display issues (2)

Symptom: Display does not turn on when kernel takes over from bootloader (1 in 8 boots)
Workaround: Force display always on from kernel commandline.
Cause: HDMI TMDS clock turned on/off with each DDC probe.

Display issues (2)

Symptom: Display does not turn on when kernel takes over from bootloader (1 in 8 boots)
Workaround: Force display always on from kernel commandline.
Cause: HDMI TMDS clock turned on/off with each DDC probe.
Fix: Enable proper refcounting in the clock tree. (5e1bc251ce)

Display issues (2)

Symptom: Display does not turn on when kernel takes over from bootloader (1 in 8 boots)
Workaround: Force display always on from kernel commandline.
Cause: HDMI TMDS clock turned on/off with each DDC probe.
Fix: Enable proper refcounting in the clock tree. (5e1bc251ce)

```
+++ b/drivers/gpu/drm/sun4i/sun4i_hdmi_enc.c
@@ -92,6 +92,8 @@ static void sun4i_hdmi_disable(struct drm_encoder *encoder)
+ clk_disable_unprepare(hdmi->tmds_clk);
 }

@@ -102,6 +104,8 @@ static void sun4i_hdmi_enable(struct drm_encoder *encoder)

+ clk_prepare_enable(hdmi->tmds_clk);
```

Display issues (3)

Quirky display (640x480) with actually top 240 pixels are visible.

Problem: Need to identify quirky displays from the TCD application, so we can supply the correct sized media.

Display issues (3)

Quirky display (640x480) with actually top 240 pixels are visible.
Problem: Need to identify quirky displays from the TCD application, so we can supply the correct sized media.
Solution?: Use manufacturer information from Gtk+?

```
from gi.repository import Gdk
disp = Gdk.Display.get_default()
num = Gdk.Display.get_n_monitors(disp)
for m in range(0, num):
    monitor = Gdk.Display.get_monitor(disp, m)
    print (monitor.get_model())
    print (monitor.get_manufacturer())
```

Display issues (3)

Quirky display (640x480) with actually top 240 pixels are visible.
Problem: Need to identify quirky displays from the TCD application, so we can supply the correct sized media.
Solution?: Use manufacturer information from Gtk+?

```
from gi.repository import Gdk
disp = Gdk.Display.get_default()
num = Gdk.Display.get_n_monitors(disp)
for m in range(0, num):
    monitor = Gdk.Display.get_monitor(disp, m)
    print (monitor.get_model())
    print (monitor.get_manufacturer())
```

Bug?: Manufacturer fields always empty, and model contains output name (for example HDMI-1, DP-1-2, eDP-1, ...)

Display issues (3)

Quirky display (640x480) with actually top 240 pixels are visible.
Problem: Need to identify quirky displays from the TCD application, so we can supply the correct sized media.
Solution?: Use manufacturer information from Gtk+?

```
from gi.repository import Gdk
disp = Gdk.Display.get_default()
num = Gdk.Display.get_n_monitors(disp)
for m in range(0, num):
    monitor = Gdk.Display.get_monitor(disp, m)
    print (monitor.get_model())
    print (monitor.get_manufacturer())
```

Bug?: Manufacturer fields always empty, and model contains output name (for example HDMI-1, DP-1-2, eDP-1, ...)
Fix: `https://gitlab.gnome.org/GNOME/gtk/merge_requests/848`

Fun with /dev/random

Symptom: Reaaaallllyyy long boot time after switch to Debian 10.

Fun with /dev/random

Symptom: Reaaaallllyyy long boot time after switch to Debian 10.

Cause: Systemd which using `/dev/random` (?) is blocked due to lack of entropy.

IIRC it was actually caused by kernel itself changing the randomness behaviour.

Fun with /dev/random

Symptom: Reaaaallllyyy long boot time after switch to Debian 10.

Cause: Systemd which using `/dev/random` (?) is blocked due to lack of entropy.

IIRC it was actually caused by kernel itself changing the randomness behaviour.

Solution: `apt install haveged`

Networking troubles (1)

Symptom: Link does not come up when connected to certain switches

Networking troubles (1)

Symptom: Link does not come up when connected to certain switches
Cause: Buggy silicon in PHY chip

Networking troubles (1)

Symptom: Link does not come up when connected to certain switches
Cause: Buggy silicon in PHY chip
Solution: Enable correct PHY in defconfig (9567832aba7)

```
+++ b/arch/arm/configs/sunxi_defconfig
 CONFIG_STMMAC_ETH=y
 # CONFIG_NET_VENDOR_VIA is not set
 # CONFIG_NET_VENDOR_WIZNET is not set
+CONFIG_MICREL_PHY=y
 # CONFIG_WLAN is not set
 CONFIG_INPUT_EVDEV=y
 CONFIG_KEYBOARD_SUN4I_LRADC=y
```

Networking troubles (2)

Symptom: Unable to find devices from network using `avahi-browse` after the flashing step.

Networking troubles (2)

Symptom: Unable to find devices from network using `avahi-browse` after the flashing step.
Cause: DHCP pool full, new devices not getting IP anymore.

Networking troubles (2)

Symptom: Unable to find devices from network using `avahi-browse` after the flashing step.
Cause: DHCP pool full, new devices not getting IP anymore.

Easy, right :)

So long, and Thanks for All the Bugs!
Questions?