NICTA

seL4 & VM & CAmkES Tutorial

4 Feb 2015

Ihor Kuz

Australian Government
Department of Broadband, Communications and the Digital Economy
Australian Research Council

NICTA Funding and Supporting Members and Partners

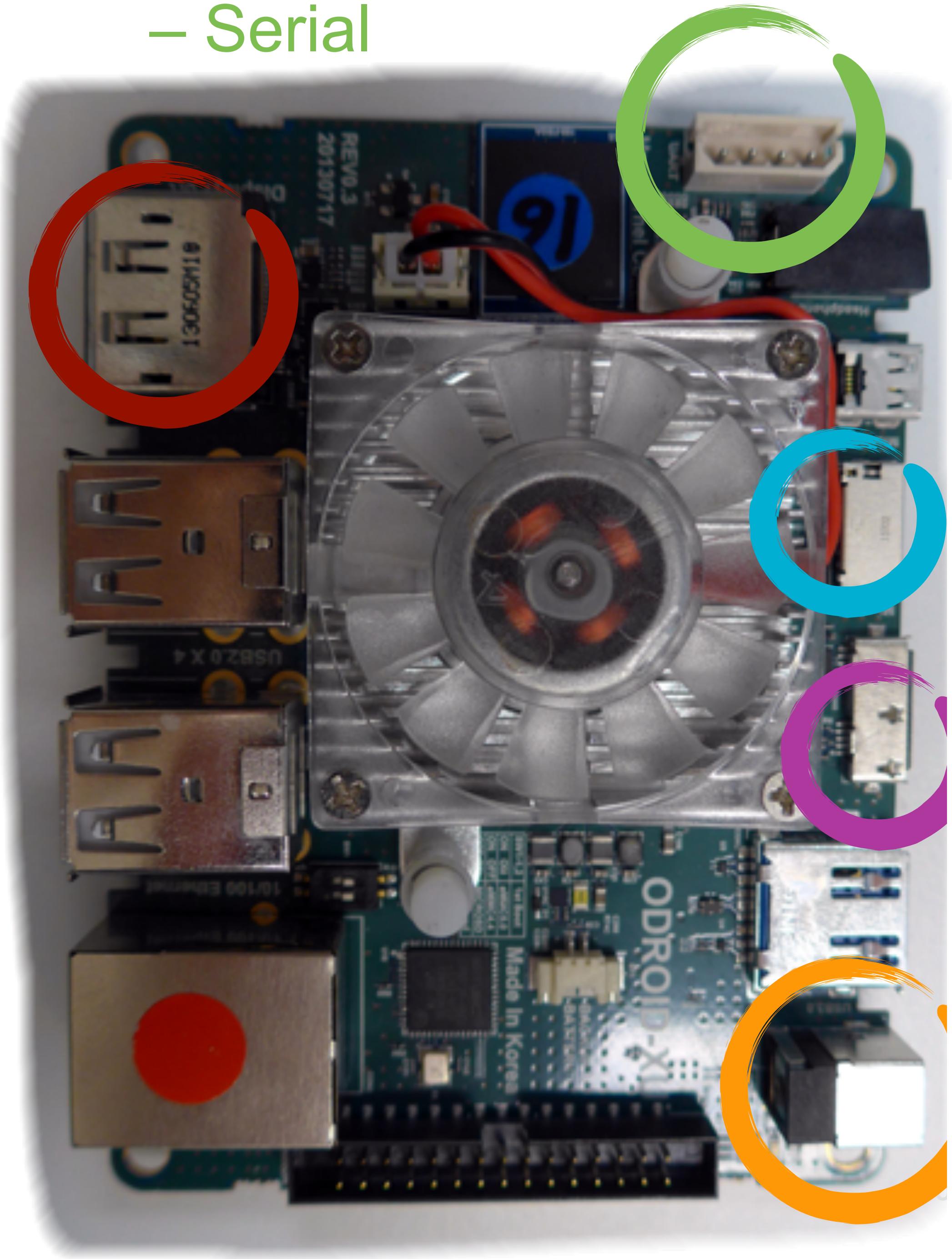
Australian National University
UNSW
NSW Trade & Investment
Victoria
The University of Melbourne
The University of Sydney
Queensland Government
Griffith University
QUT
The University of Queensland

Overview

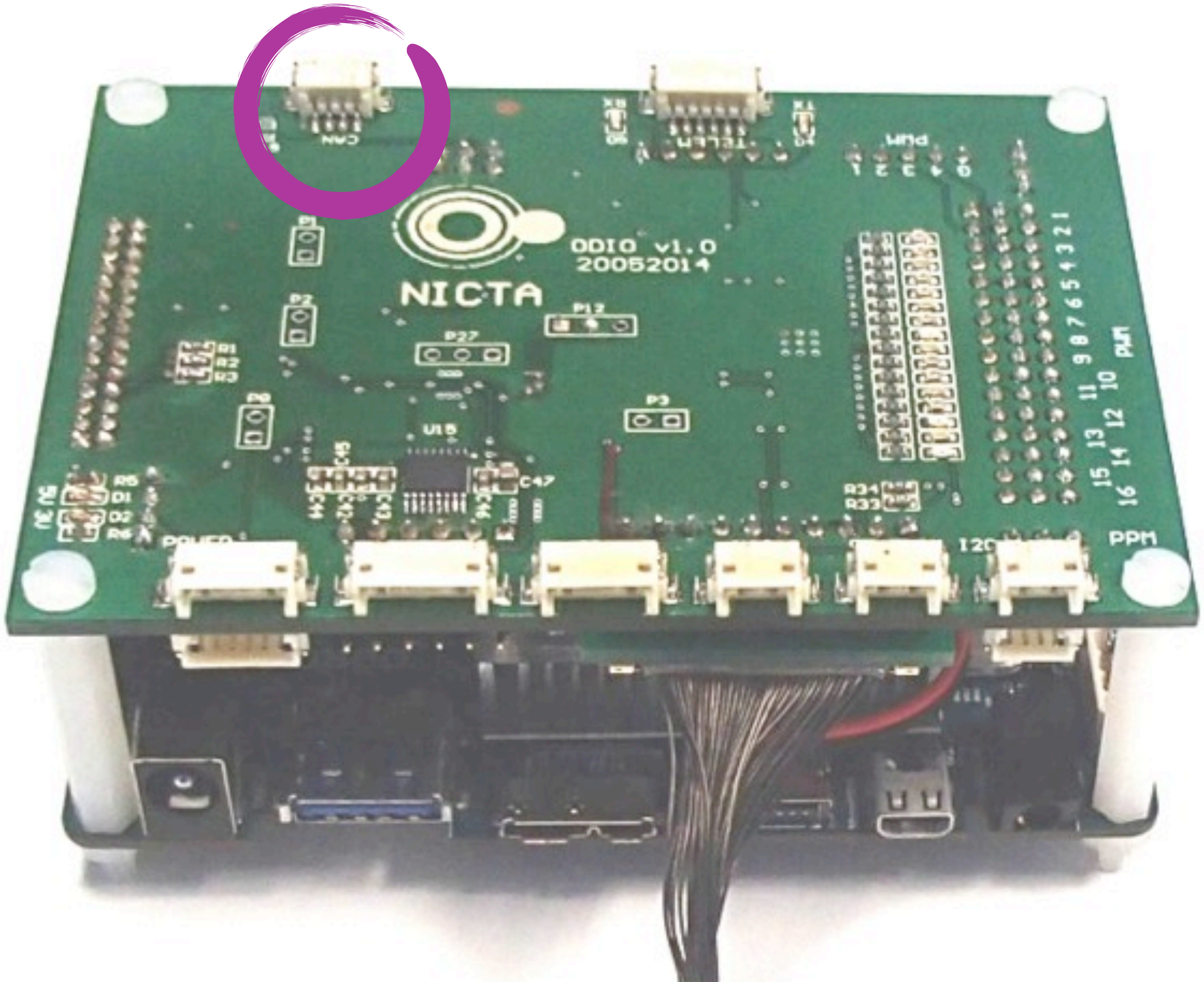
- Refresher
  - Hardware
  - Software
- seL4 & Linux VM
  - Basic
  - PIXY camera and networking
- Adding CAmkES components
  - PWM and LED
- CAN and fragmentation
  - Simple CAN with pixhawk
  - CAN fragmentation with Arduino

Hardware Setup

- Odroid-XU
  - Power
  - microUSB
  - Serial
  - SD
  - Ethernet



- Daughterboard
  - Odroid-XU connections
  - CAN



Hardware Setup

- Prerequisites
  - `sudo apt-get install minicom android-tools-fastboot u-boot-tools`
  - configure minicom
    - update `/etc/group`: add self to dialup
    - 115200 8N1 HW/SW flow control off, save as `odroid`
  - configure fastboot
    - `echo SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", MODE=="0666", GROUP=="users" | sudo tee /etc/udev/rules.d/40-odroidxu-fastboot.rules`
- Connect the cables
  - UART-USB to UART
  - micro USB to micro USB slot
- Start minicom
  - new window
  - `minicom odroid`
- Start the Odroid-XU

Software Prerequisites

- Linux (Ubuntu)
  - sudo apt-get install lib32z1 lib32ncurses5 lib32bz2-1.0
- Compiler

  -
    ```
    wget https://sourcery.mentor.com/public/gnu_toolchain/arm-none-eabi/arm-2013.11-24-arm-none-eabi-i686-pc-linux-gnu.tar.bz2
    ```
  - unpack into /opt/local

  -
    ```
    echo "export PATH=/opt/local/arm-2013.11/bin:\$PATH" >> ~/.bashrc
    ```
- Python

  -
    ```
    sudo apt-get instal python-pip python-tempita
    ```

  -
    ```
    sudo pip install --upgrade pip
    ```

  -
    ```
    sudo pip install jinja2 ply pyelftools
    ```
- Haskell

  -
    ```
    sudo apt-get install cabal-install
    ```

  -
    ```
    cabal update; cabal install MissingH data-ordlist split
    ```
- Qemu

  -
    ```
    sudo apt-get install qemu
    ```
- Misc

  -
    ```
    sudo apt-get install realpath libxml2-utils
    ```
- Git & repo

  -
    ```
    sudo apt-get git phablet-tools; git config --global user.email “<email>”
    ```

Getting and Building seL4 and CAmkES

NICTA

- Get project repository

```
mkdir camkes; cd camkes;
repo init -u http://github.com/seL4/camkes-manifest; repo sync
```

- Config

```
make arm_simple_defconfig; make silentoldconfig
```

- Build it!

```
make # make V=1
```

- Run it

```
qemu-system-arm -M kzm -nographic -kernel images/capdl-loader-experimental-image-arm-imx31
```

- Clean up

```
make clean # make clobber; make mrproper
```

CAmkES Linux VM: Building

- Get repo
  ```
  mkdir camkes-arm-vm; cd camkes-arm-vm
  ```
  ```
  repo init -u https://github.com/smaccm/camkes-arm-vm-manifest.git; repo sync
  ```
- Modify Linux device tree to access FS on SD
  ```
  sed -i 's/mmcblk0p2/mmcblk1p2/' apps/vm/linux/linux-dtb
  ```
- Build
  ```
  make vm_defconfig; make silentoldconfig; make
  ```
- Prepare to run on hardware
  ```
  mkimage -a 0x48000000 -e 0x48000000 -C none -A arm -T kernel -O qnx -d images/capdl-loader-experimental-image-arm-exynos5 odroid-image
  ```

CAmkES Linux VM: Running

- Prepare Linux root FS on SD
  - plug SD into dev machine
  - https://www.dropbox.com/s/hkduec0ezi7i2ux/smaccm_demo.img.gz
  - `gunzip -c smaccm_demo.img.gz | dd of=<device file, eg /dev/sdb>; sync`
  - mount 1st partition, modify boot.ini (remove `` `run verify` ``)
  - plug SD into odroid
- Run
  - `sudo fastboot boot odroid-image`
  - login odroid:odroid; su root:odroid
- Prepare Linux root FS on eMMC
  - use eMMC to SD converter, then as above
- Modify Linux device tree to access FS on eMMC
  - `sed -i 's/mmcblk1p2/mmcblk0p2/' apps/vm/linux/linux-dtb`

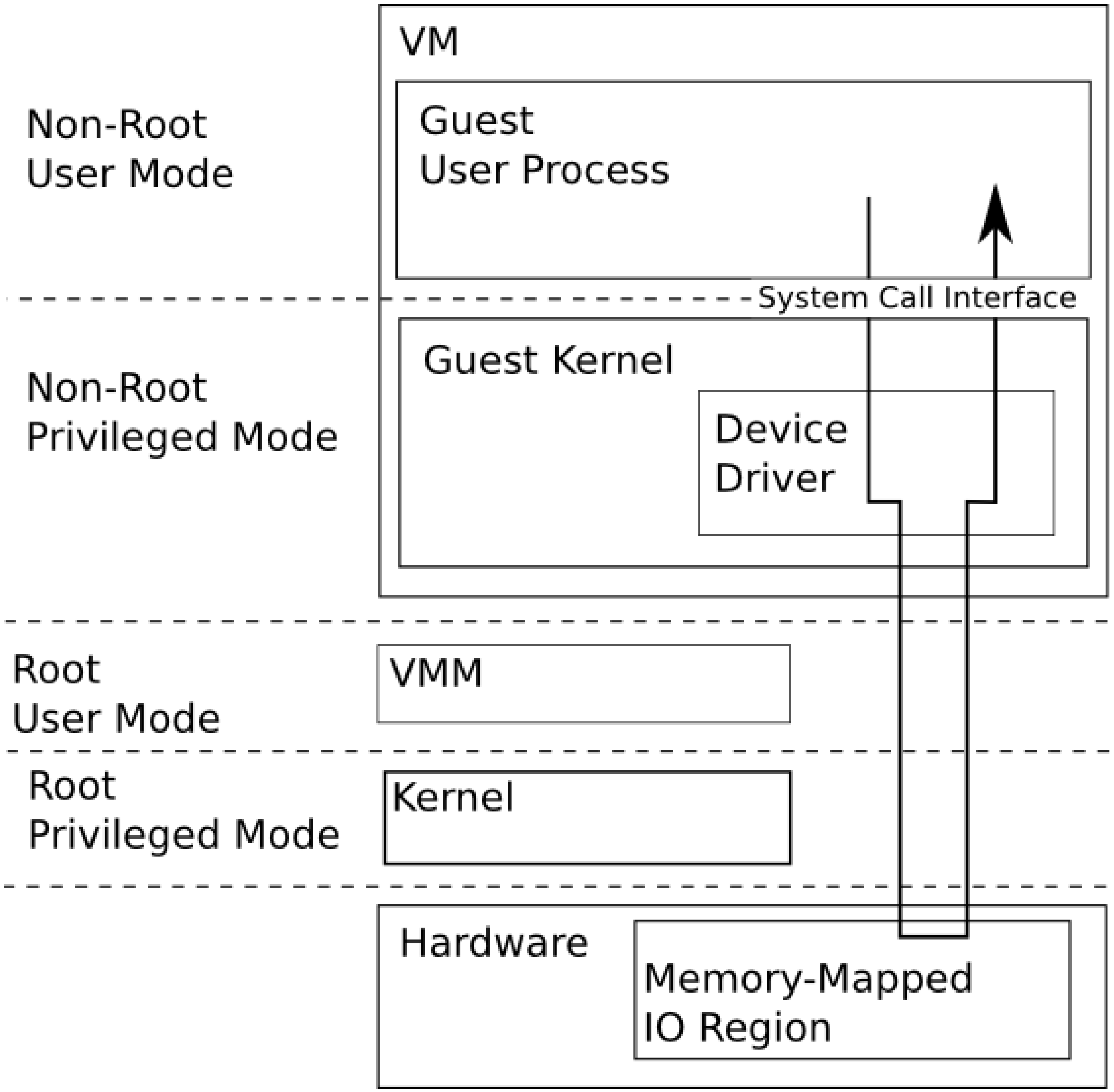
What’s in it?

- Look at apps/vm/vm.camkes
- vm : vmm & Linux vm
  - vmm: creates an inits vspace for VM, starts VM thread
    - loads kernel and device tree from archive in image. comes from apps/vm/linux/linux
    - kernel uses filesystem on SD (or eMMC)
  - vmm: catches and processes faults for VM
  - vmm: code in libs/libsel4arm-vmm
- configuration
  - vm.simple = true;
  - vm.mmio = "0x10486000:0x1000:12";
    - make caps for physical memory available to vm. For direct device access
    - vmm will map in appropriate location for vm (typically 1:1)
  - vm.irq = "27";
    - make irq available to vm
    - vmm will receive irqs, pass them through to vm
  - vm.cnode_size_bits = 18;
    - size of cnode given to vm
  - vm.untyped24_pool = 10; vm.asid_pool = true;

Linux with Pixy cam

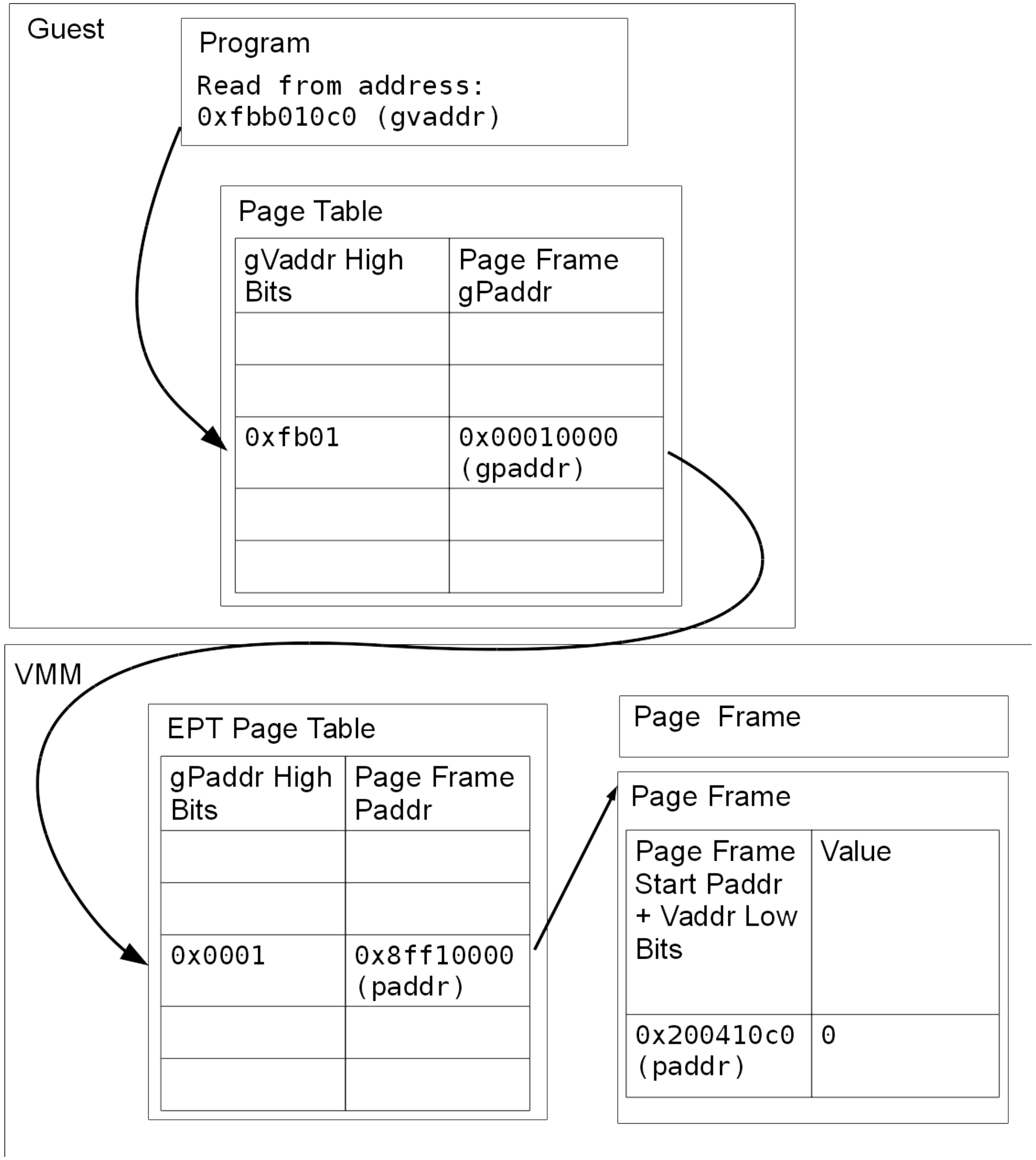
- Setup networking
  - odroid:
    - `nmcli dev disconnect iface eth0;`
    - `sudo ifconfig eth0 192.168.0.10`
  - client:
    - (set virtualbox nework to Bridge)
    - `sudo ifconfig eth0 192.168.0.30`
- Connect Pixy camera
- Run host software
  - software already on John's FS image :-)
  - `/root/camera_demo/demo`
- Run client software
  - prereqs: `sudo apt-get install java`
  - `git clone https://github.com/smaccm/camera_demo.git`
  - `cd camera_demo`
  - `java -jar SmaccmViewer.jar 192.168.0.10`

How Does Passthrough Work



Linux VM passthrough hardware access

- VM virtual address translation



Guest

Program

```
Read from address:
0xfbb010c0 (gvaddr)
```

Page Table

gVaddr High Bits	Page Frame gPaddr
0xfb01	0x00010000 (gpaddr)

VMM

EPT Page Table

gPaddr High Bits	Page Frame Paddr
0x0001	0x8ff10000 (paddr)

Page Frame

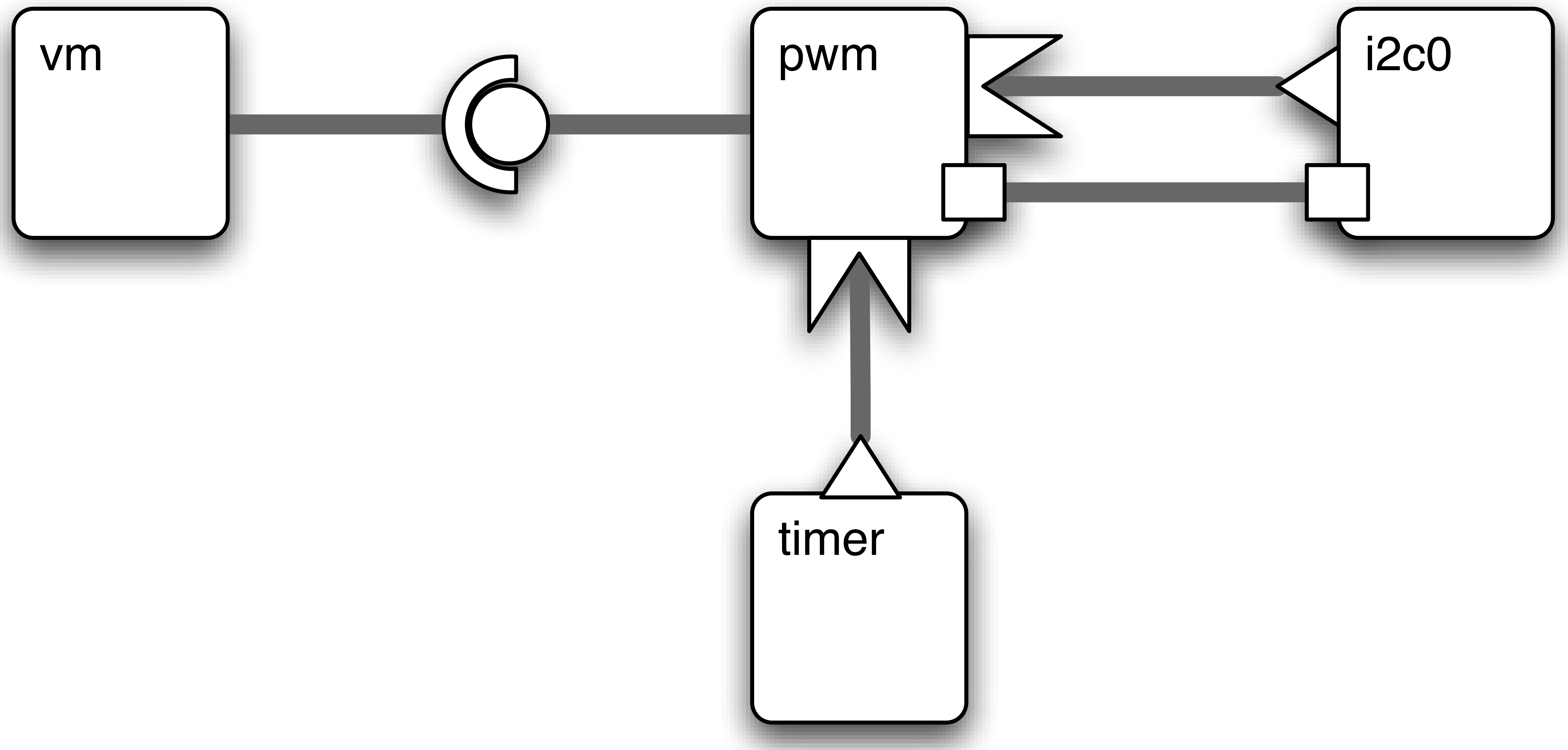
Page Frame

Page Frame Start Paddr + Vaddr Low Bits	Value
0x200410c0 (paddr)	0

Linux VM passthrough hardware access

- What does Linux access using passthrough:
  - currently: everything
- Why is passthrough a problem?
  - page granularity
    - can access other devices that are accessed on same page
  - DMA
    - if passthrough device uses DMA, then Linux can cause DMA
    - without IOMMU, Linux can cause device to write to any physical address
    - in particular, addresses not mapped to Linux

Linux to CAmkES communication



Linux to CAmkES communication

- PWM component
  - apps/vm/components/pwm
  - flash LEDs
  - provides pwm: vmsig(in int data), consumes timer event
  - i2c0: consume int
- VM component
  - uses pwm_inf
  - userland writes to unmapped memory to cause a fault
    - see led_flash.c
  - VMM catches fault, reads memory, calls pwm_vmsig(mem);
- Connect VM and PWM
  - regular seL4RPC connection
- In linux
  - `gcc leds_flash.c; ./a.out`

CAN example with pixhawk

- Prereqs:
  - gcc etc. gdb
  - haskell, cabal, etc.
- Get smaccmpilot
  - `git clone https://github.com/GaloisInc/smaccmpilot-build --recursive`
  - `cd smaccmpilot-stm32f4; git checkout 83c44b3`
  - `cd ivory; git checkout 2e9abf1`
- Build it, load it on pixhawk
- Get CAmkES code
  - `tar xf camkes-can-pixhawk.tgz`
- Build it, load it, run it
  - `make can_defconfig; make silentoldconfig; make`
- Connect pixhawk and odroid

CAN example with fragmentation

- Uses Arduino with CAN controller
- Get and build
  - `tar xf camkes-can-fragmentation.tgz`
- Fragmentation library
  - summer student project in progress (Rafael Belcher)
- canbus component:
  - creates long message
  - calls FRAG to send
  - receives reply fragments
  - calls RASS to reassemble
- Arduino
  - receives message fragment
  - ROT13's it
  - sends it back

TODO

- Include CAN with VM
  - must avoid busy looping
- We have 3DR radio working with VM
  - but encountered flakiness. Not sure why.
- seL4 device drivers for VM
  - already have: gic, uart, gpio, clock, etc.
  - need: USB!
    - seL4 USB driver in progress
- Linux to CAmkES communication
  - more sophisticated: interfaces, etc.

Backup Slides

Flashing U-Boot

- Prerequisites
  - minicom, fastboot, see previous slide
  - bl2: `odroid/smdk5410-spl.bin.signed`
  - u-boot: `odroid/u-boot.bin`
- Turn it on
  - in minicom window, make sure Odroid goes into fastboot
- Flash away
  - `sudo fastboot flash bl2 smdk5410-spl.bin.signed`
  - `sudo fastboot flash bootloader u-boot.bin`
  - `sudo fastboot reboot`
- Set fastboot as boot command
  - `setenv bootcmd fastboot`
  - `saveenv`