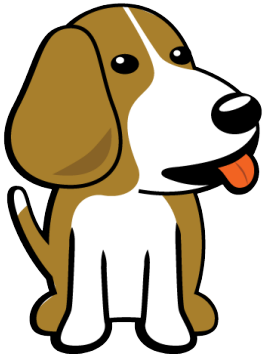
Slides: https://github.com/pdp7/talks/blob/master/rt-summit-2019.pdf

Real-time usage in the BeagleBoard.org community

Real Time Summit 2019 (Lyon)



Drew Fustini

BeagleBoard.org Foundation

drew@beagleboard.org

Twitter: @pdp7

beagleboard.org

CC BY SA

- Open Source Hardware designer at OSH Park
  - PCB manufacturing service in the USA
  - drew@oshpark.com / Twitter: @oshpark

- Member of Board of Directors of BeagleBoard.org Foundation
  - **drew@beagleboard.org**

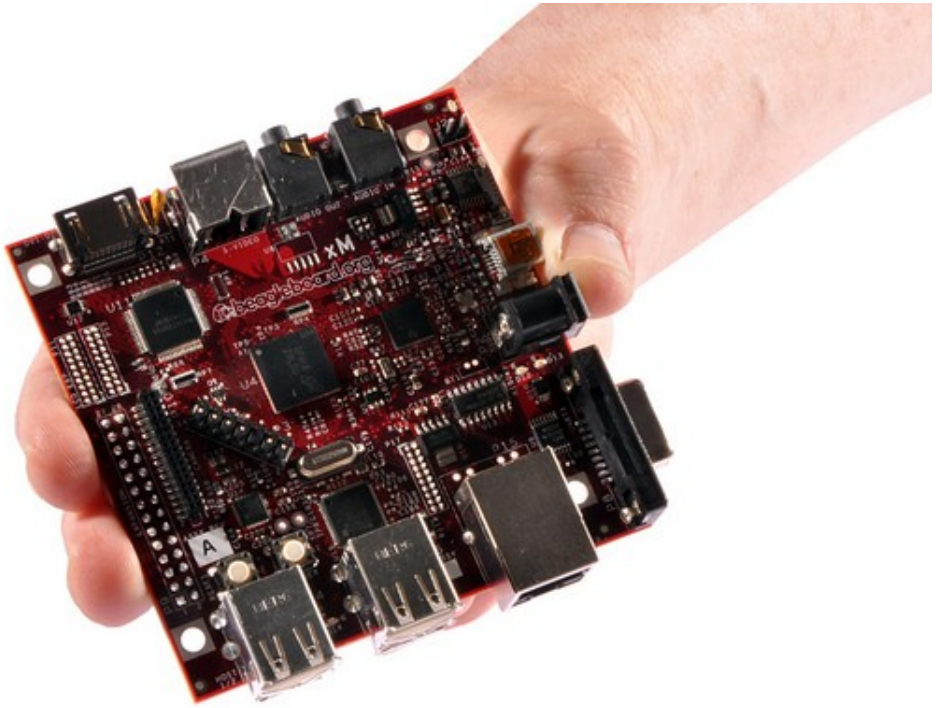
- Vice President of the Open Source Hardware Association (OSHWA)
  - serving as Vice President
  - **drew@pdp7.com**

beagleboard.org

- Open Source Hardware computing for Makers, Educators & Professionals
- Developed by BeagleBoard.org Foundation and BeagleBoard.org Community
- Manufacturers: element14, GHI, Seeed

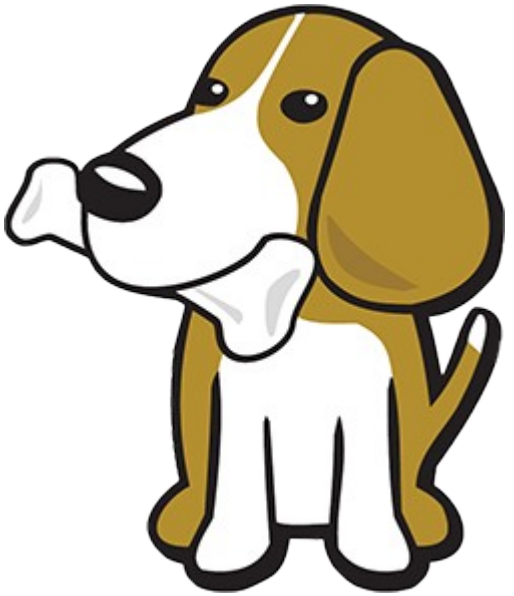
beagleboard.org

BeagleBoard.org released the first **BeagleBoard**, an affordable, open hardware ARM computer in **2008**



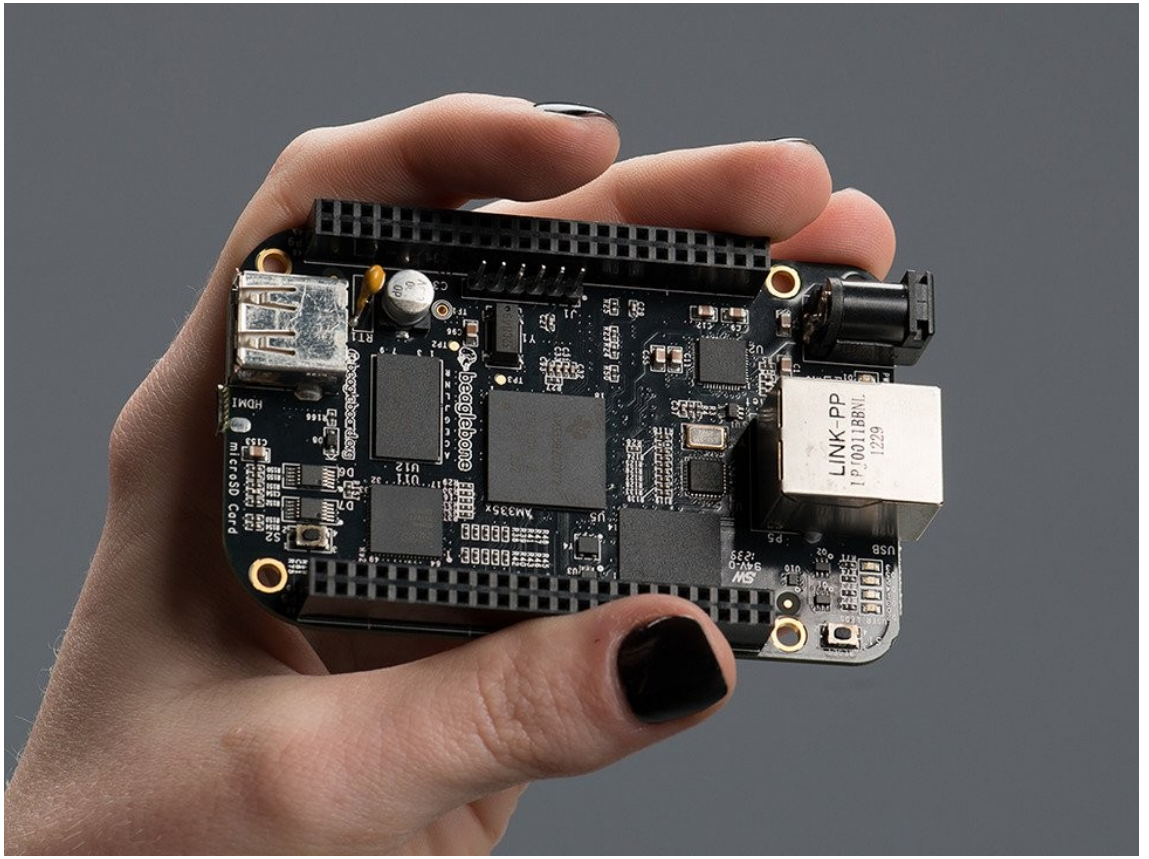
beagleboard.org

Maker focused, Altoids tin sized
BeagleBone introduced in **2011**

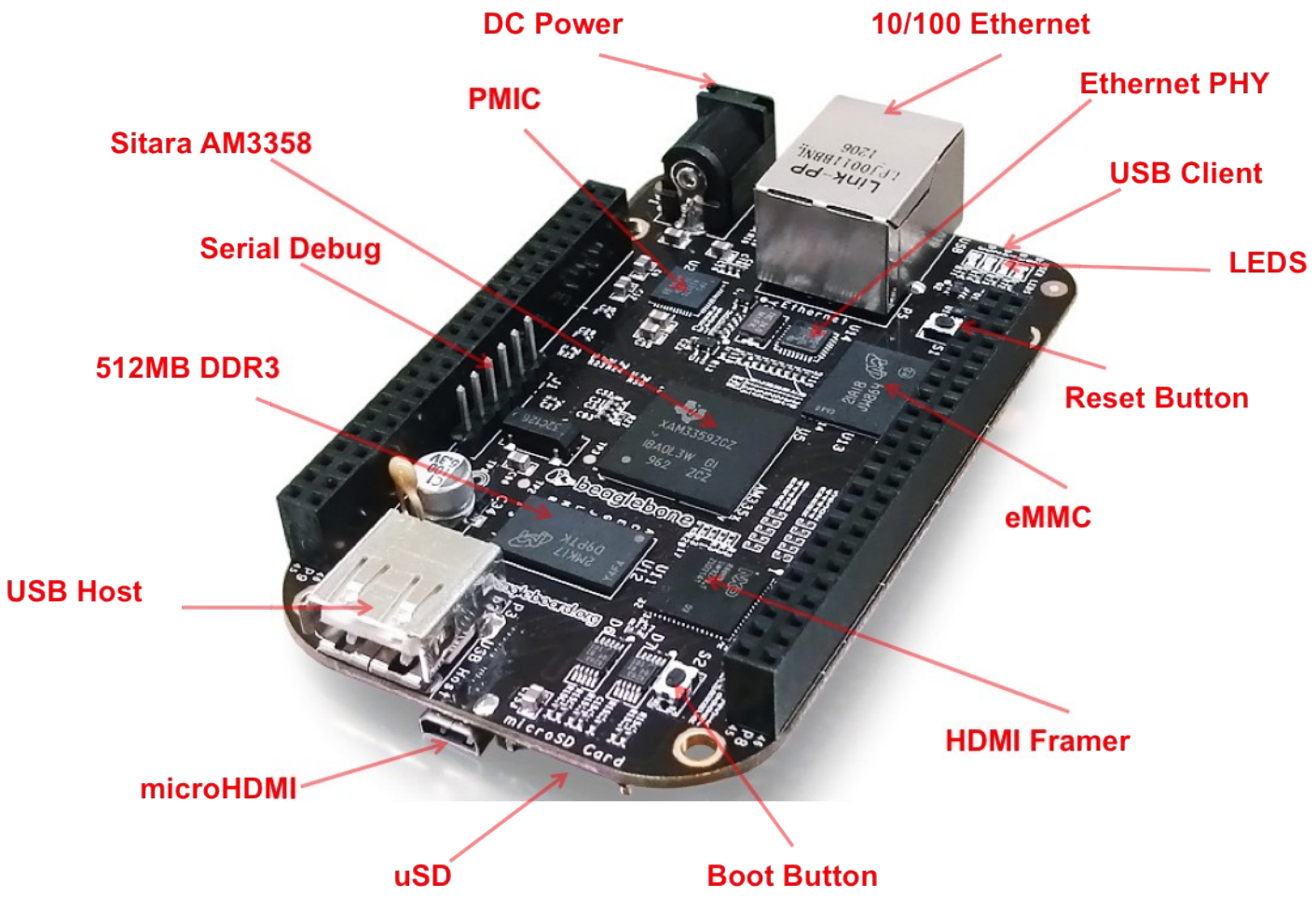


beagleboard.org

More affordable, more powerful
BeagleBone Black in **2013**



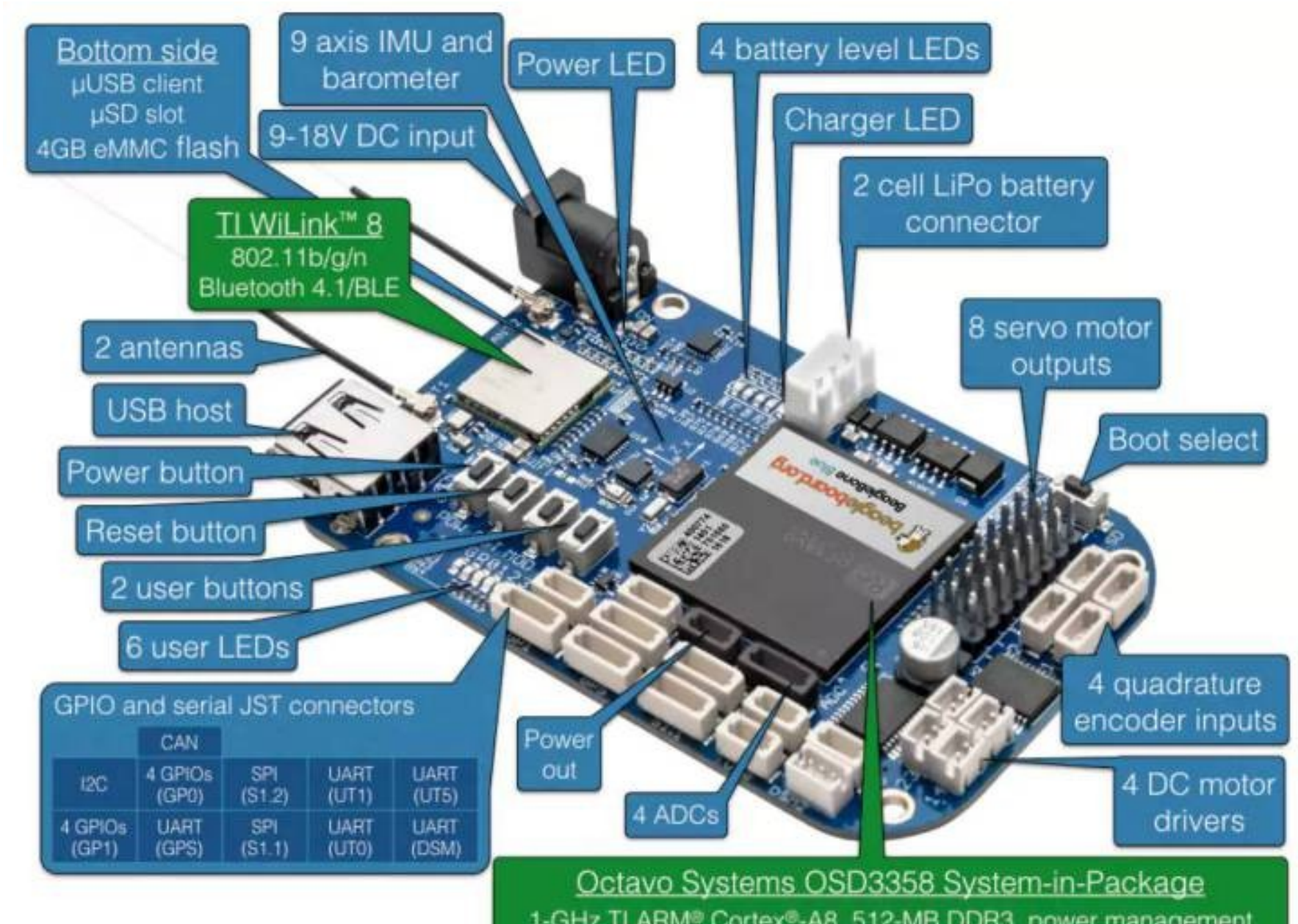
BeagleBone Black



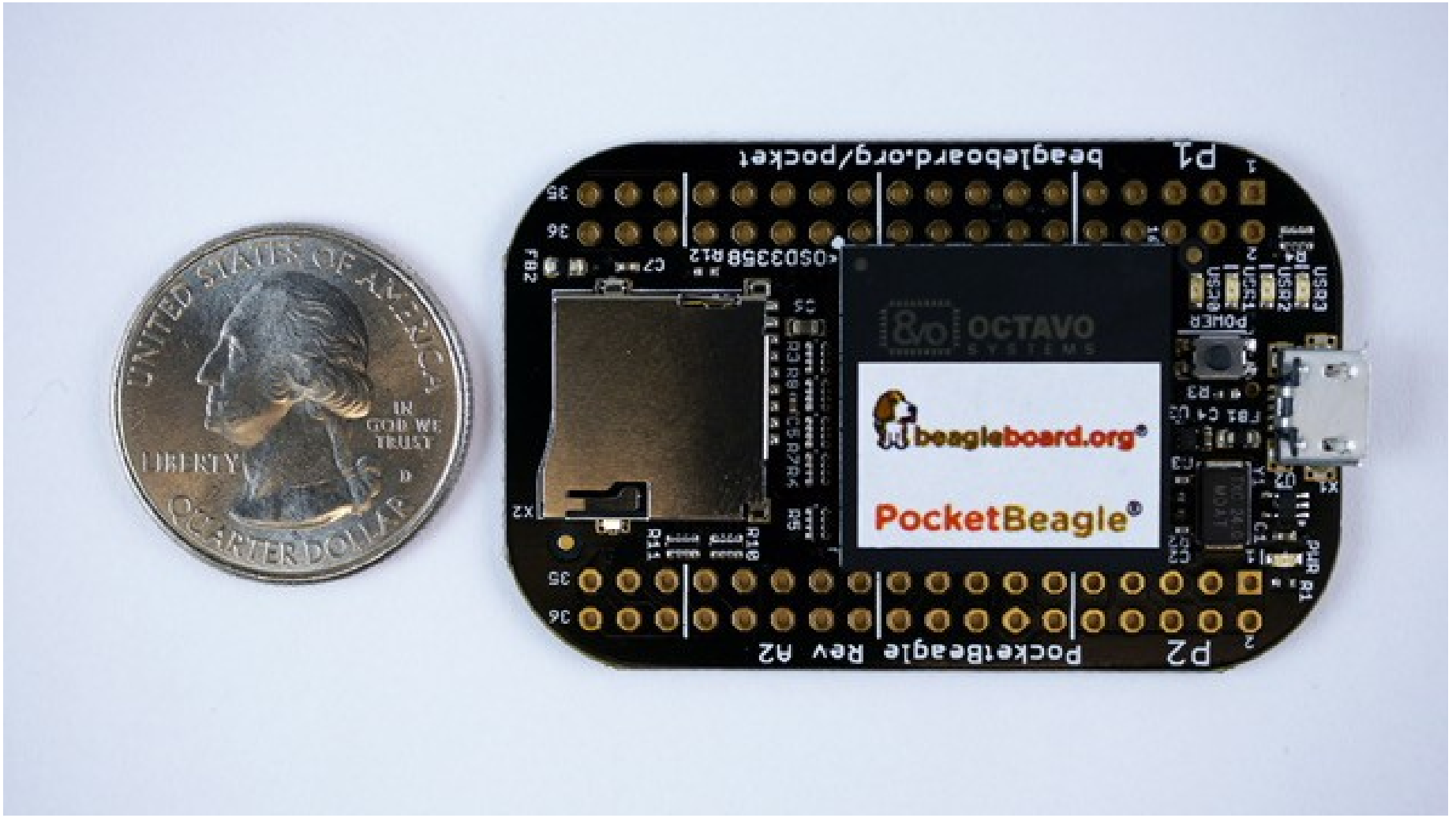
Open Source Hardware
BeagleBone derivatives

	Capes	HDMI	Flash	Special
BeagleBoard.org BeagleBone	Y	N	N	JTAG
BeagleBoard.org BeagleBone Black	Y	Y	Y	-
Arrow BeagleBone Black Industrial	Y	Y	Y	Industrial
Element14 BeagleBone Black Industrial	Y	Y	Y	Industrial
SeeedStudio BeagleBone Green	Y	N	Y	Grove
SanCloud BeagleBone Enhanced	Y	Y	Y	1GB, 1Gbit, wireless
BeagleBoard.org BeagleBone Blue	N	N	Y	Robotics
BeagleBoard.org BeagleBoard-X15	N	Y	N	Big jump in CPUs and I/O

BeagleBone Blue: complete Linux robotics controller. 4 layer PCB designed in EAGLE.



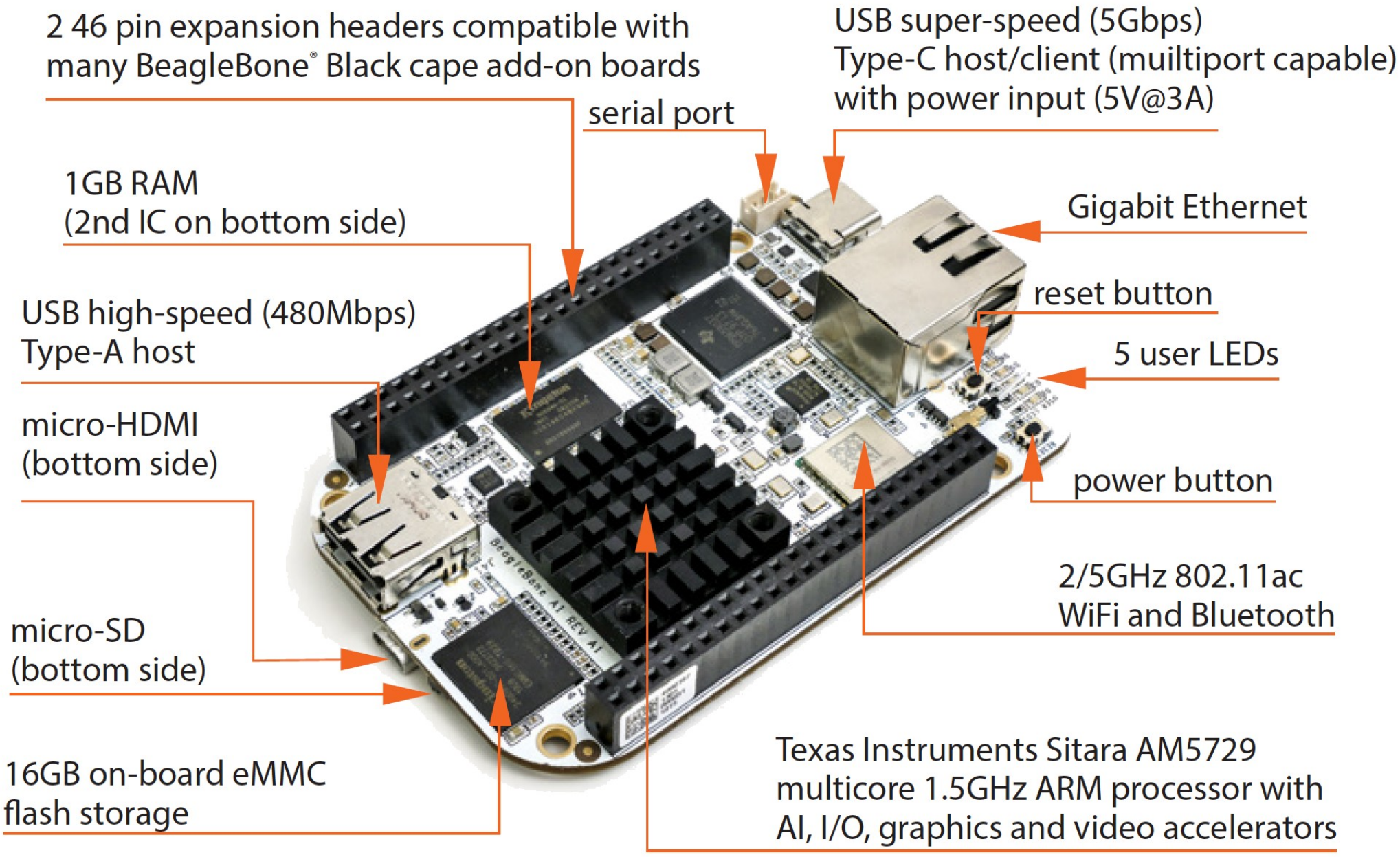
BeagleBoard.org PocketBeagle



Linux Foundation training

- PocketBeagle and TechLab board

BeagleBone AI: The Fast Track for Embedded Machine Learning


2 46 pin expansion headers compatible with many BeagleBone® Black cape add-on boards
serial port
USB super-speed (5Gbps) Type-C host/client (muiltiport capable) with power input (5V@3A)
1GB RAM (2nd IC on bottom side)
Gigabit Ethernet
reset button
USB high-speed (480Mbps) Type-A host
5 user LEDs
micro-HDMI (bottom side)
power button
2/5GHz 802.11ac WiFi and Bluetooth
micro-SD (bottom side)
16GB on-board eMMC flash storage
Texas Instruments Sitara AM5729 multicore 1.5GHz ARM processor with AI, I/O, graphics and video accelerators

BeagleBone AI

“TI C66x digital-signal-processor (DSP) cores and embedded-vision-engine (EVE) cores supported through an optimized TIDL machine learning OpenCL API with pre-installed tools. Focused on everyday automation in industrial, commercial and home applications.”

Feature highlights:

- BeagleBone Black mechanical and header compatibility
- TI AM5729 SoC: 2x A15 CPU, 2x C66 DSP, 4x M4 MCU, 4x PRU and 4x EVE
- 1GB RAM and 16GB on-board eMMC flash with high-speed interface
- USB type-C for power and superspeed dual-role controller; and USB type-A host
- Gigabit Ethernet, 2.4/5GHz WiFi, and Bluetooth
- microHDMI
- Zero-download out-of-box software experience

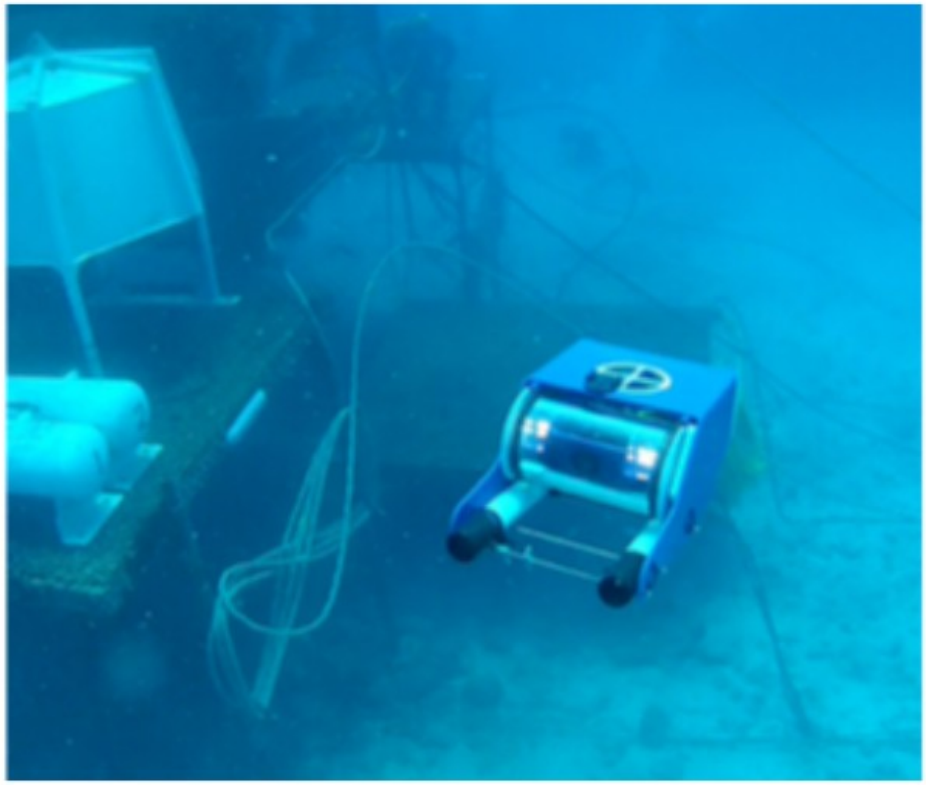
Why is BeagleBone perfect for bots?

- Lots of I/Os (65 digital, 7 analog inputs, 8 PWMs...)
- PRUs (2 32-bit RISC microcontrollers
- Fast (1GHz) super-scalar armv7a processor
- Linux makes networking easy
- Ready to use out-of-the-box

Examples usage

- Tight control loops
  - Driving motors in a mobile robot, CNC machine or 3D printer
- Custom protocols
  - WS28x LEDs, DMX512, ...
  - EtherCAT, ProfiBUS, ProfiNET, ...
- Soft peripherals
  - PWM, UART (LEGO), ...

OpenROV

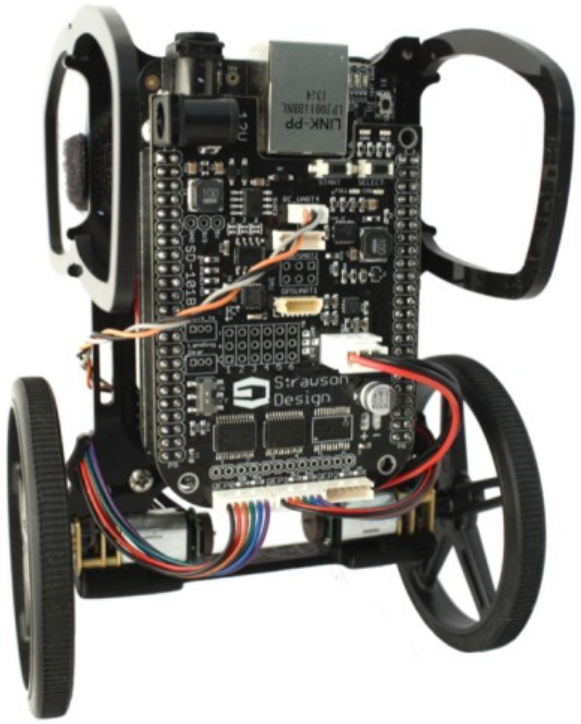


- Open-source underwater robot
- Community creating more accessible, affordable and awesome tools for underwater exploration
- Started by people wanting to explore an underwater cave
- Successfully Kickstarter'd

beagleboard.org

BeagleMIP

UCSD



- Self-Balancing robot powered by the BeagleBone Black and the Novus Robotics Cape
- Hackable Open Source Robotics Platform for Fun and Education
- Developed at the University of California, San Diego to Teach Advanced Digital Control Systems

beagleboard.org

BeagleQuad

UCSD

The new ***Novus Robotics Cape*** sends your ***BeagleBone Black*** projects to the sky!

beagleboard.org

TI Sitara AM3358 (BeagleBone White/Green/Blue, PocketBeagle)

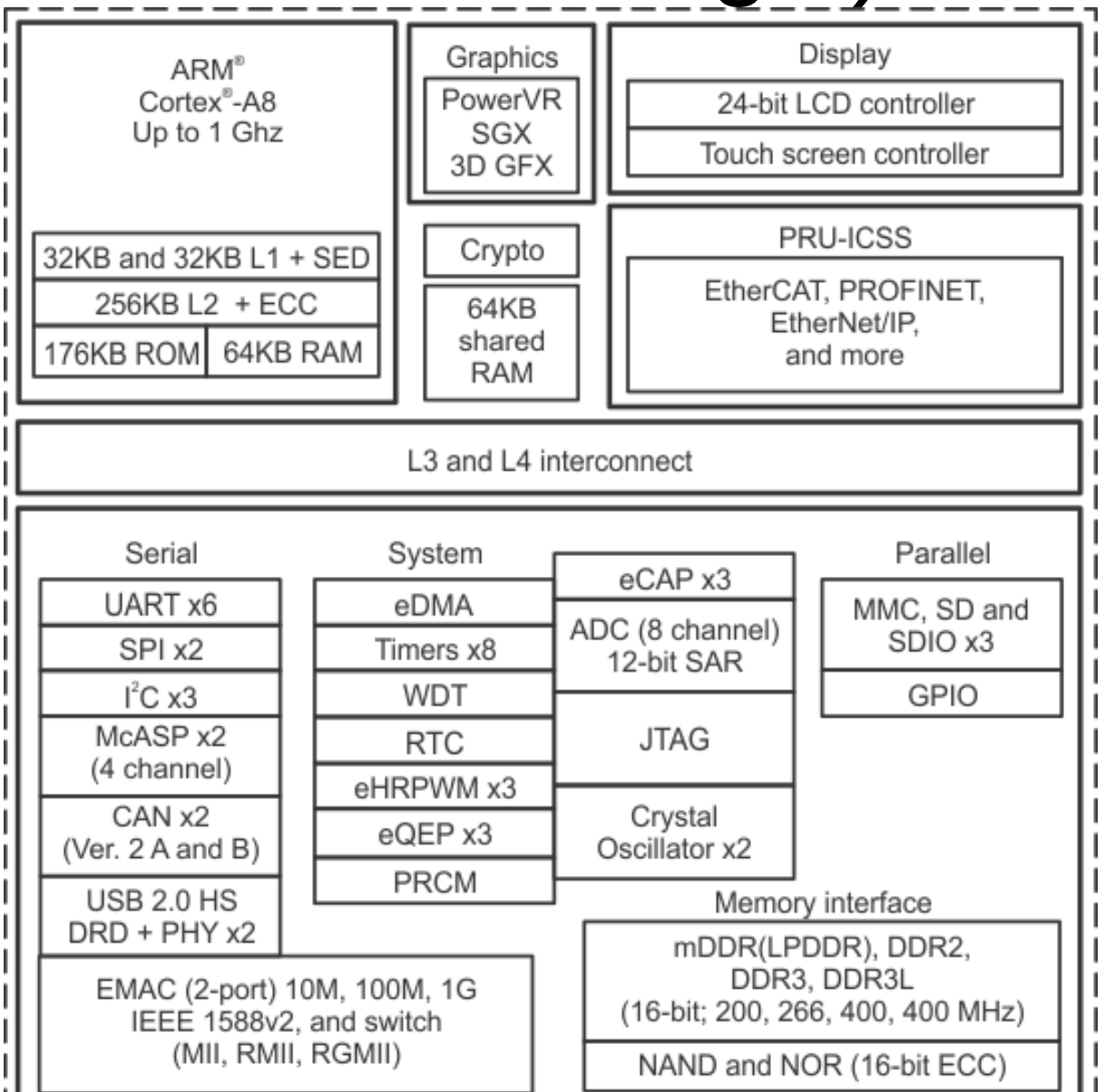
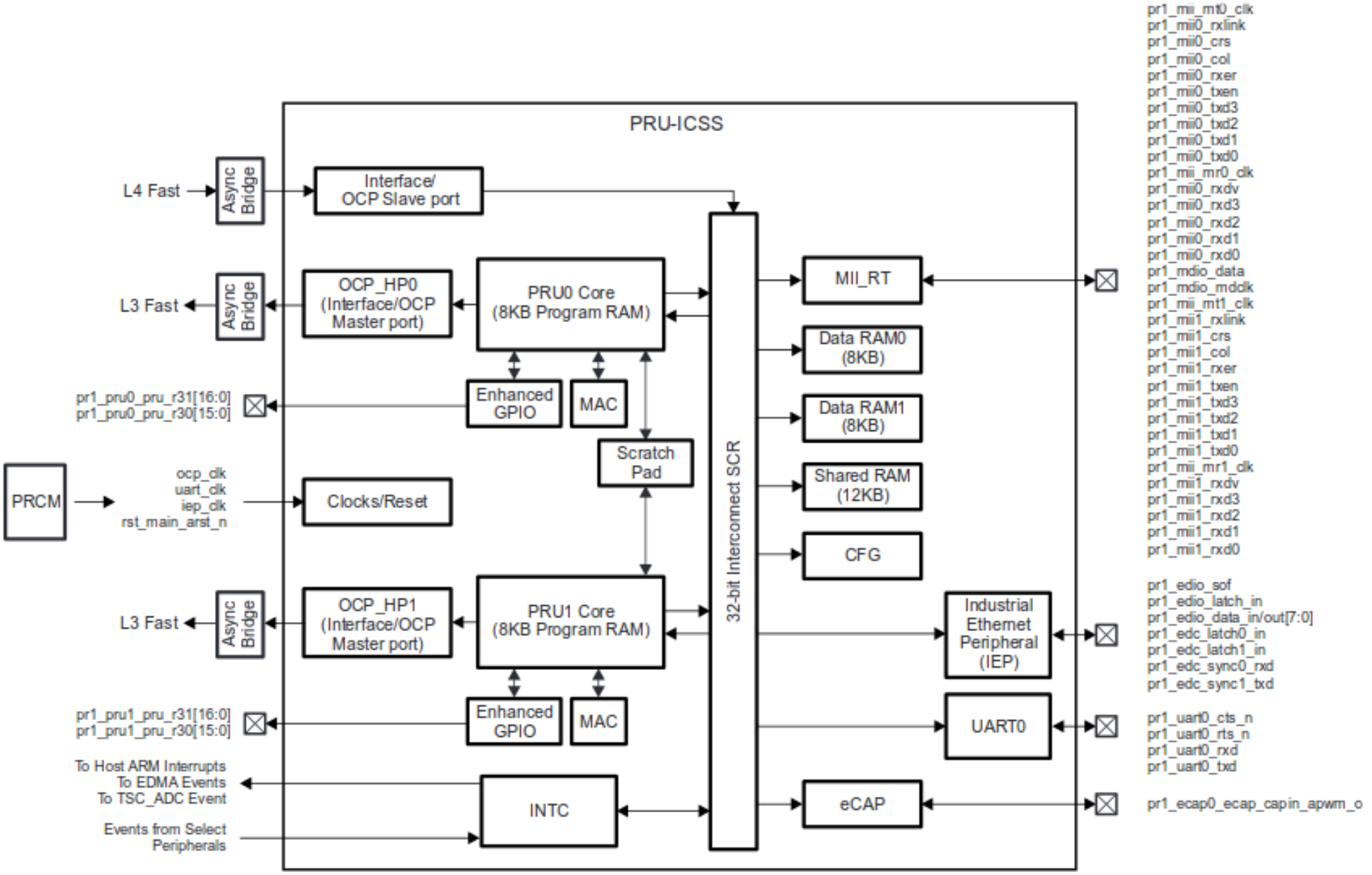


Figure 2. PRU-ICSS Integration



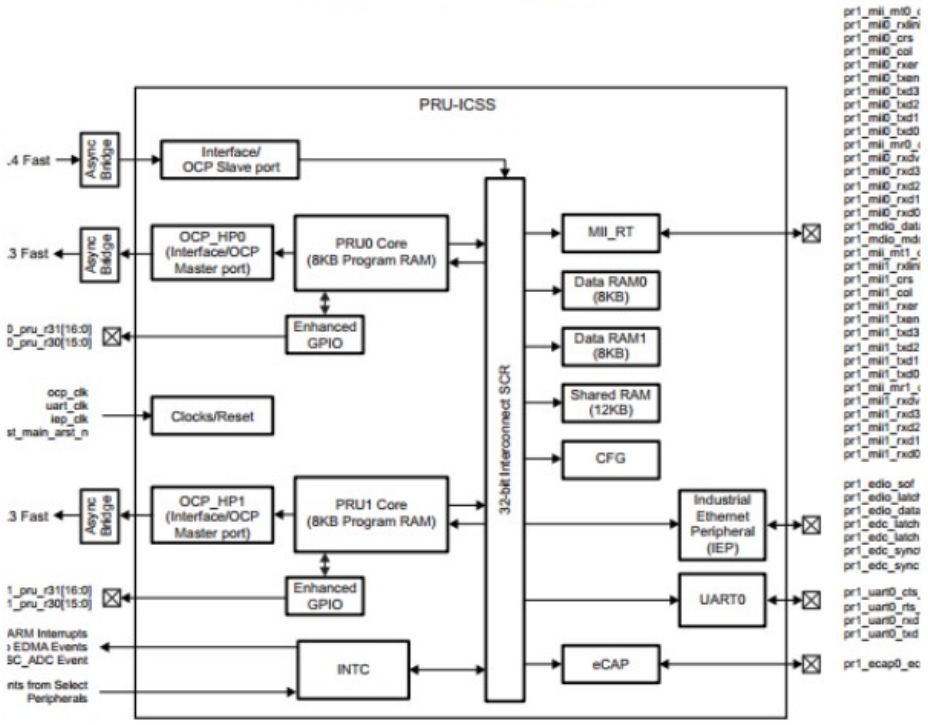
What are PRUs

- “Programmable Real-time Units”
- 32-bit RISC processors at 200MHz with single-cycle pin access for hard real-time
- Optimized for packet processing/switching and software implementations of peripherals
- Part of the PRU-ICSS, “Industrial Communications SubSystem”

PRUSS architecture details

- 2 cores at 200MHz each
- Memory
  - 8kB program each
  - 8kB data each
  - 12kB data shared
  - Access through L3 to external memory and peripherals

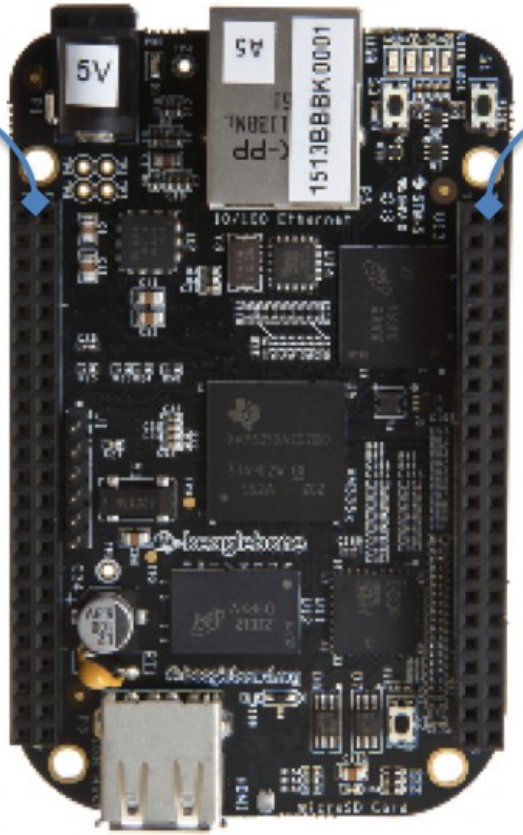
Figure 2. PRU-ICSS Integration



Cape Expansion Headers

P9

DGND	1	2	DGND
VDD_3V3	3	4	VDD_3V3
VDD_5V	5	6	VDD_5V
SYS_5V	7	8	SYS_5V
PWR_BUT	9	10	SYS_RESETn
UART4_RXD	11	12	GPIO_60
UART4_TXD	13	14	EHRPWM1A
GPIO_48	15	16	EHRPWM1B
SPI0_CS0	17	18	SPI0_D1
I2C2_SCL	19	20	I2C2_SDA
SPI0_D0	21	22	SPI0_SCLK
GPIO_49	23	24	UART1_TXD
GPIO_117	25	26	UART1_RXD
GPIO_115	27	28	SPI1_CS0
SPI1_D0	29	30	GPIO_112
SPI1_SCLK	31	32	VDD_ADC
AIN4	33	34	GNDA_ADC
AIN6	35	36	AIN5
AIN2	37	38	AIN3
AIN0	39	40	AIN1
GPIO_20	41	42	ECAPPWM0
DGND	43	44	DGND
DGND	45	46	DGND



LEGEND
Power/Ground/Reset
Available Digital
Available PWM
Shared I2C Bus
Reconfigurable Digital
Analog Inputs (1.8V)

P8

DGND	1	2	DGND
MMC1_DAT6	3	4	MMC1_DAT7
MMC1_DAT2	5	6	MMC1_DAT3
GPIO_66	7	8	GPIO_67
GPIO_69	9	10	GPIO_68
GPIO_45	11	12	GPIO_44
EHRPWM2B	13	14	GPIO_26
GPIO_47	15	16	GPIO_46
GPIO_27	17	18	GPIO_65
EHRPWM2A	19	20	MMC1_CMD
MMC1_CLK	21	22	MMC1_DAT5
MMC1_DAT4	23	24	MMC1_DAT1
MMC1_DAT0	25	26	GPIO_61
LCD_VSYNC	27	28	LCD_PCLK
LCD_HSYNC	29	30	LCD_AC_BIAS
LCD_DATA14	31	32	LCD_DATA15
LCD_DATA13	33	34	LCD_DATA11
LCD_DATA12	35	36	LCD_DATA10
LCD_DATA8	37	38	LCD_DATA9
LCD_DATA6	39	40	LCD_DATA7
LCD_DATA4	41	42	LCD_DATA5
LCD_DATA2	43	44	LCD_DATA3
LCD_DATA0	45	46	LCD_DATA1

25 PRU low-latency I/Os

P9

DGND	1	2	DGND
VDD_3V3	3	4	VDD_3V3
VDD_5V	5	6	VDD_5V
SYS_5V	7	8	SYS_5V
PWR_BUT	9	10	SYS_RESETN
GPIO_30	11	12	GPIO_60
GPIO_31	13	14	GPIO_50
GPIO_48	15	16	GPIO_51
GPIO_5	17	18	GPIO_4
I2C2_SCL	19	20	I2C2_SDA
GPIO_3	21	22	GPIO_2
GPIO_49	23	24	GPIO_15
PRU0_7	25	26	PRU1_16 IN
PRU0_5	27	28	PRU0_3
PRU0_1	29	30	PRU0_2
PRU0_0	31	32	VDD_ADC
AIN4	33	34	GNDA_ADC
AIN6	35	36	AIN5
AIN2	37	38	AIN3
AIN0	39	40	AIN1
PRU0_6	41	42	PRU0_4
DGND	43	44	DGND
DGND	45	46	DGND

P8

DGND	1	2	DGND
GPIO_38	3	4	GPIO_39
GPIO_34	5	6	GPIO_35
GPIO_66	7	8	GPIO_67
GPIO_69	9	10	GPIO_68
PRU0_15 OUT	11	12	PRU0_14 OUT
GPIO_23	13	14	GPIO_26
GPIO_47	15	16	GPIO_46
GPIO_27	17	18	GPIO_65
GPIO_22	19	20	PRU1_13
PRU1_12	21	22	GPIO_37
GPIO_36	23	24	GPIO_33
GPIO_32	25	26	GPIO_61
PRU1_8	27	28	PRU1_10
PRU1_9	29	30	PRU1_11
GPIO_10	31	32	GPIO_11
GPIO_9	33	34	GPIO_81
GPIO_8	35	36	GPIO_80
GPIO_78	37	38	GPIO_79
PRU1_6	39	40	PRU1_7
PRU1_4	41	42	PRU1_5
PRU1_2	43	44	PRU1_3
PRU1_0	45	46	PRU1_1

Accessing the other peripherals

- Yes, you can!
- The “L3” bus is exposed, so you can directly poke all of the peripheral registers
- Be careful! --- be sure the main CPU isn’t trying to access them at the same time, so you need to manually disable access to them on the main CPU

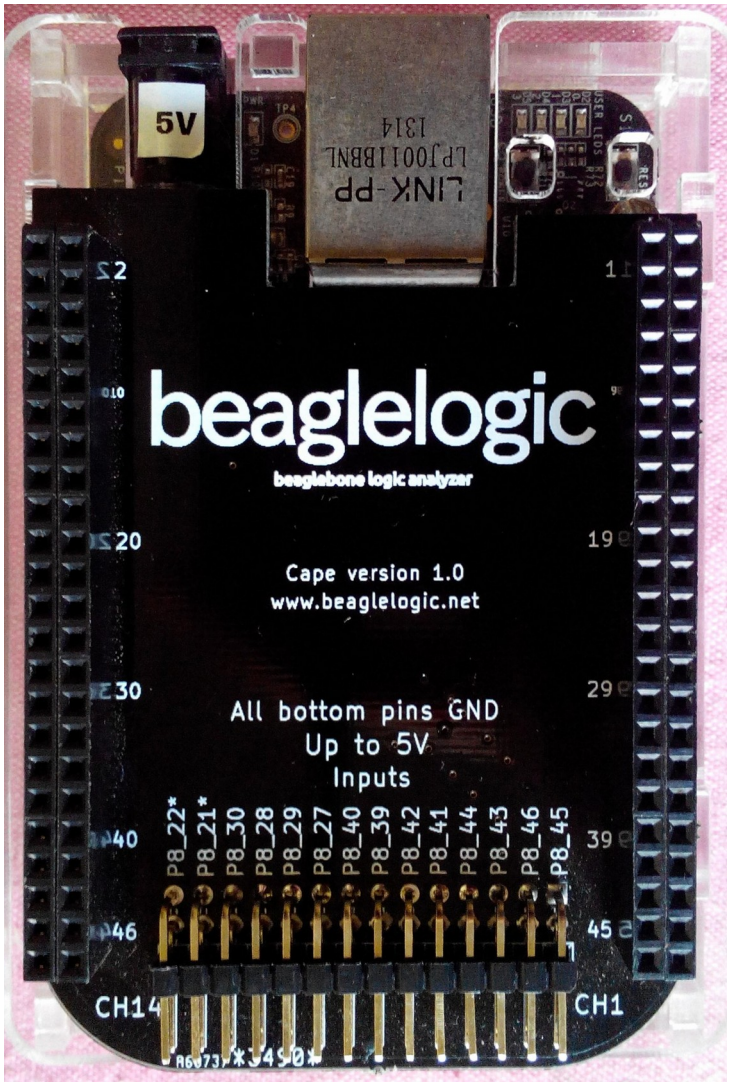
PRU Linux drivers

- uio_pruss – upstream
  - Memory mapped PRU control registers from userspace
  - Interfaces entirely in userspace library
- Various remote_proc implementations
  - “Proper” Linux abstraction of a processor
  - Lots of different “standard” interfaces

beaglelogic

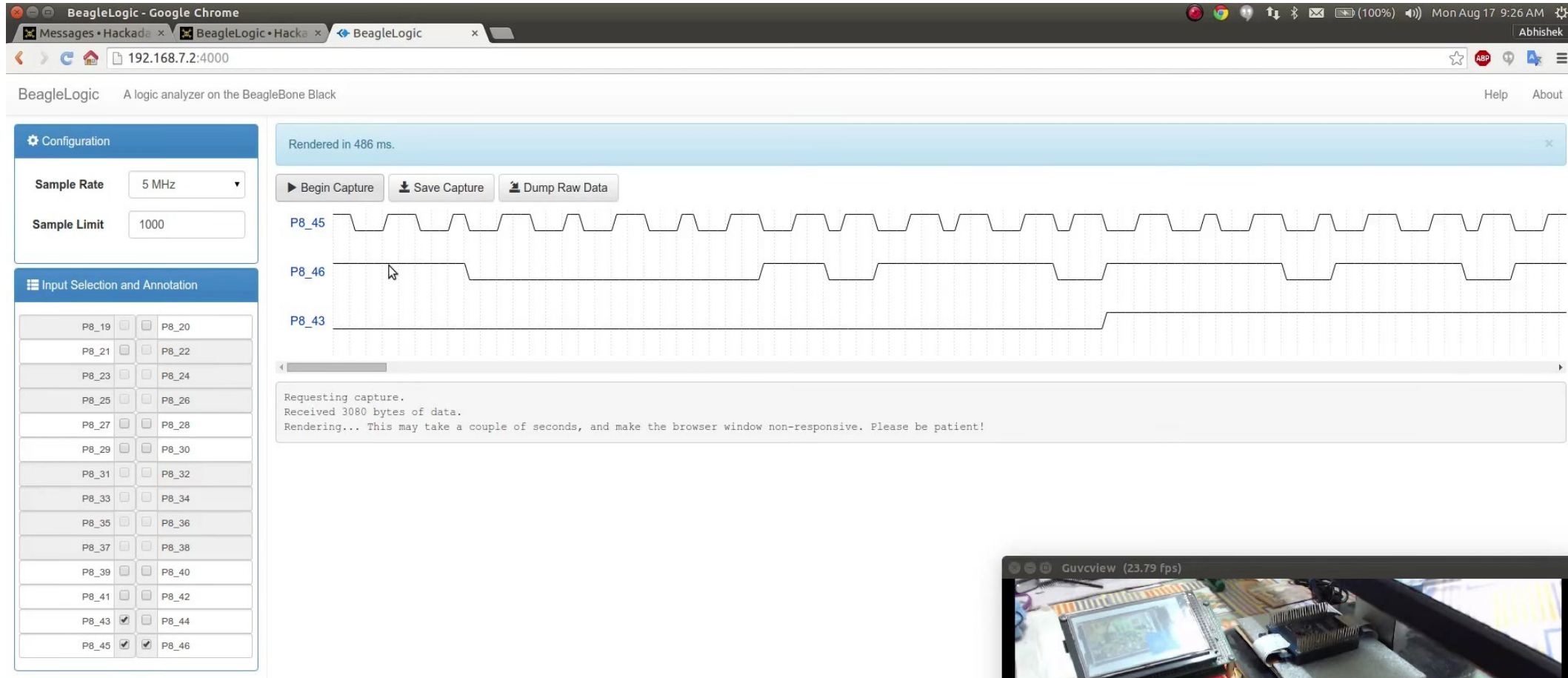
beaglebone logic analyzer

- **Kumar Abhishek** created BeagleLogic for GSoC 2014
- BeagleLogic turns BeagleBone into **Logic Analyzer**
- 14-channel, 100Msps
- Web browser user interface
- Video of final presentation



beaglelogic

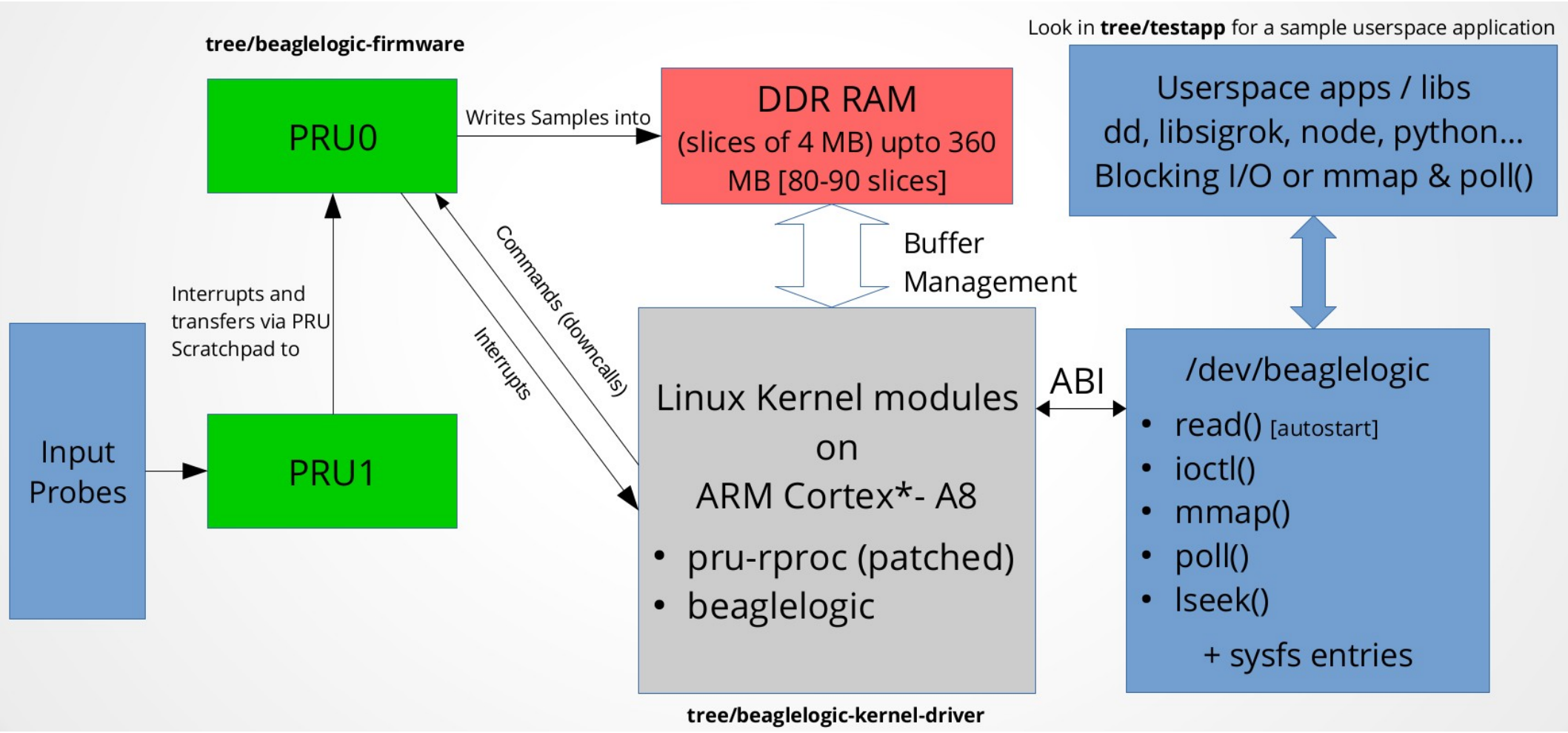
beaglebone logic analyzer



BeagleLogic - Logic Zero to One in 2 minutes

beaglelogic

beaglebone logic analyzer

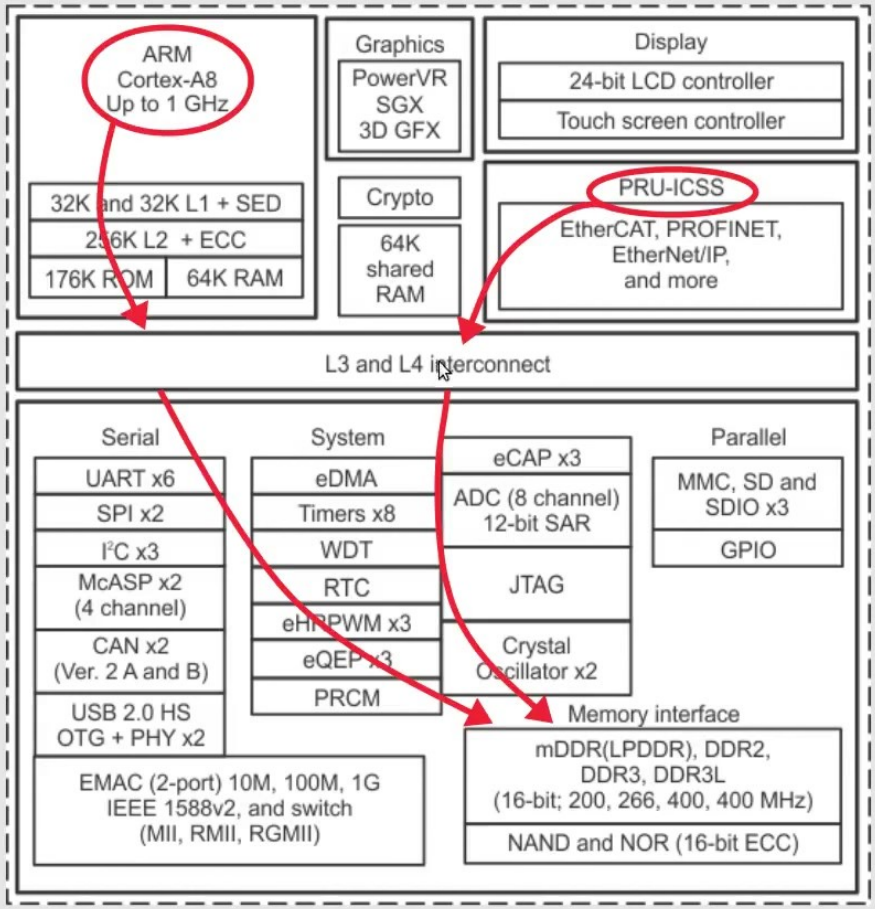


Source: https://hackaday.io/project/4395-beaglelogic

beaglelogic

beaglebone logic analyzer

The AM3358 SoC



PRU
[Programmable Real-Time Unit]:

Two Programmable 200 MHz Microcontrollers on chip for real-time tasks.

Share the interconnect which connects the ARM core to system memory

Can access DDR3 RAM independent of ARM Core!

beaglelogic

beaglebone logic analyzer

For Developers

- PRU Firmware
- Linux Kernel Driver [beaglelogic kernel module]
- Front-end /dev/beaglelogic **Character Device**
  - Appears as a standard file
  - open( ) to initialize
  - read( ) to sample, block-waits accordingly until data is available
  - ioctl( ) to configure sample rate and other settings
  - Non-blocking and Zero Copy I/O support via Memory Mapping [mmap ( )]
- NodeJS server, SocketIO link between Web interface & BeagleBone

BeaglePilot

- **Víctor Mayoral Vilches** for GSoC 2014
- Linux-based autopilot for flying robots based on BeagleBone
- Ported ArduPilot to Linux
- ROS integration
- Videos: Introduction & Final report
- GitHub: BeaglePilot

Latency	Task
100 ns	SPI bus transitions
1 us	PWM transitions, PPM-SUM input and SBUS
1 ms	IMU sensor input (gyros and accels)
20 ms	Barometer, compass, airspeed, sonar (I2C, SPI and analog).
200 ms	GPS

Table 1: Usual latencies required in an software autopilot

If these latencies are met, the autopilot will be able to fetch the sensor samples and respond in a good manner. Generally, for a capable autopilot system following sensors are necessary:

- 3-axis gyroscope
- 3-axis accelerometer
- 3-axis magnetometer
- barometer
- airspeed

BeaglePilot

- **Towards an Open Source Linux autopilot for drones**

"Linux can perfectly be used to meet the real-time requirements needed by an autopilot requiring only about 25% of the processor in BeagleBone Black."

Table 2: Kernel benchmarking results

Kernel type	Min (us)	Avg (us)	Max (us)
vanilla	14	19	193
PREEMPT	16	21	68
RT_PREEMPT	20	27	91
Xenomai	15	23	630

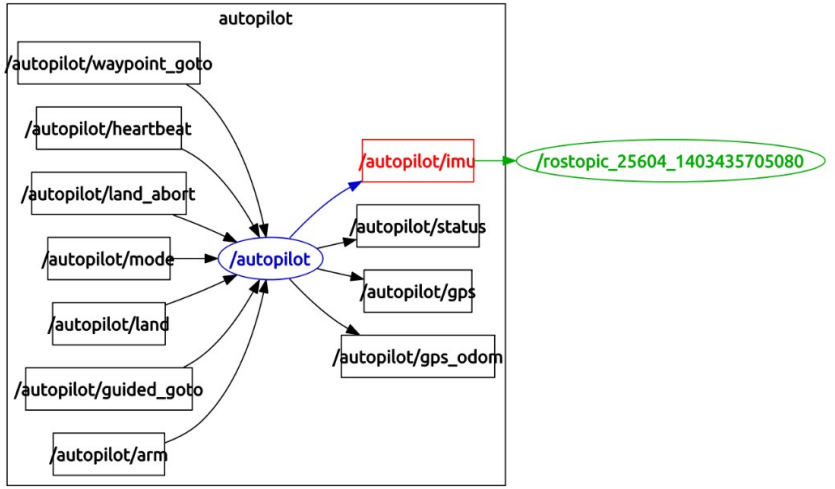
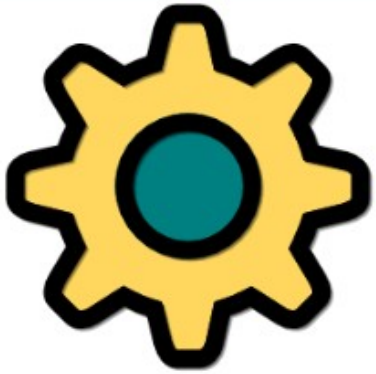


Figure 6: `autopilot_bridge` **topics and nodes pictured with** `rosgraph`

MachineKit.io (fork of LinuxCNC)



MACHINEKIT

Home About Community Documentation Blog GitHub Forum

Search ... Site Forum Commercial Support Edit this page

MOVES.

Using a wide variety of I/O driver components, effortlessly coordinate motion on your choice of ARM or x86 platforms and Xenomai or RT_PREEMPT real-time kernels.

CONTROLS.

Remote control your Machinekit project through C or Python APIs. Whether an Android GUI, a command line interface, or an interface into a higher-level system, Machinekit is flexible and simple to integrate.

THINGS.

Your imagination is the limit! Machinekit can control machine tools, quadrotors, robots, or your refrigerator.

MachineKit

- Machinekit will run on either Xenomai or PREEMPT_RT
- Current image for BeagleBone has kernel 4.19.72-bone-rt-r39
- “And the winner is: RT-PREEMPT”*
  - http://blog.machinekit.io/2015/11/and-winner-is-rt-preempt.html

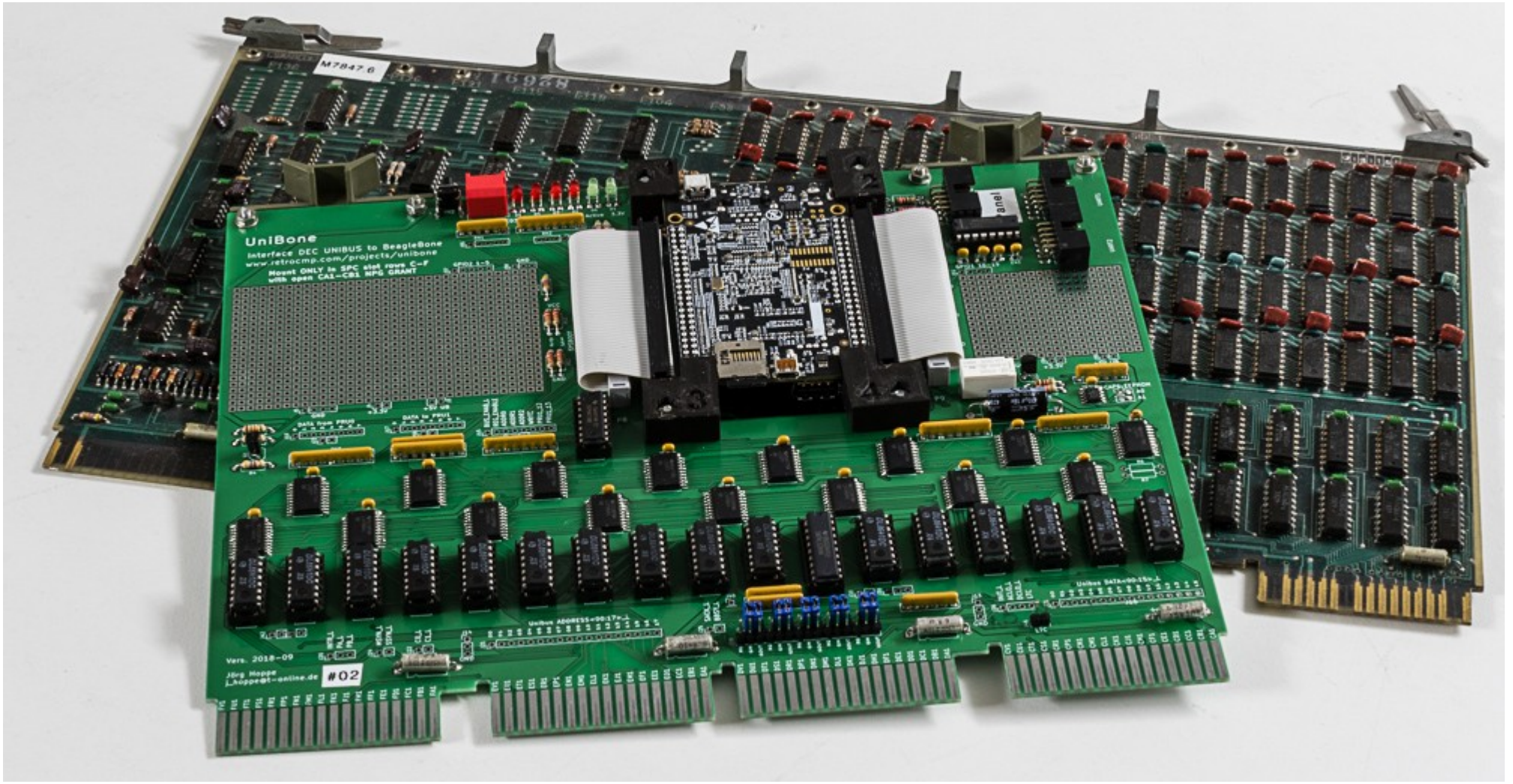
MachineKit

- Charles Steinkuehler implemented the LinuxCNC/MachineKit HAL on the BeagleBone
- The servo thread (motion planning) typically runs at 1 KHz (1 ms) on the ARM cores
- "Bit twiddling" (eg: step pulse generation) typically runs much faster, with a 3-5 uS thread period on the PRU cores

UniBone: PDP-11 card emulator

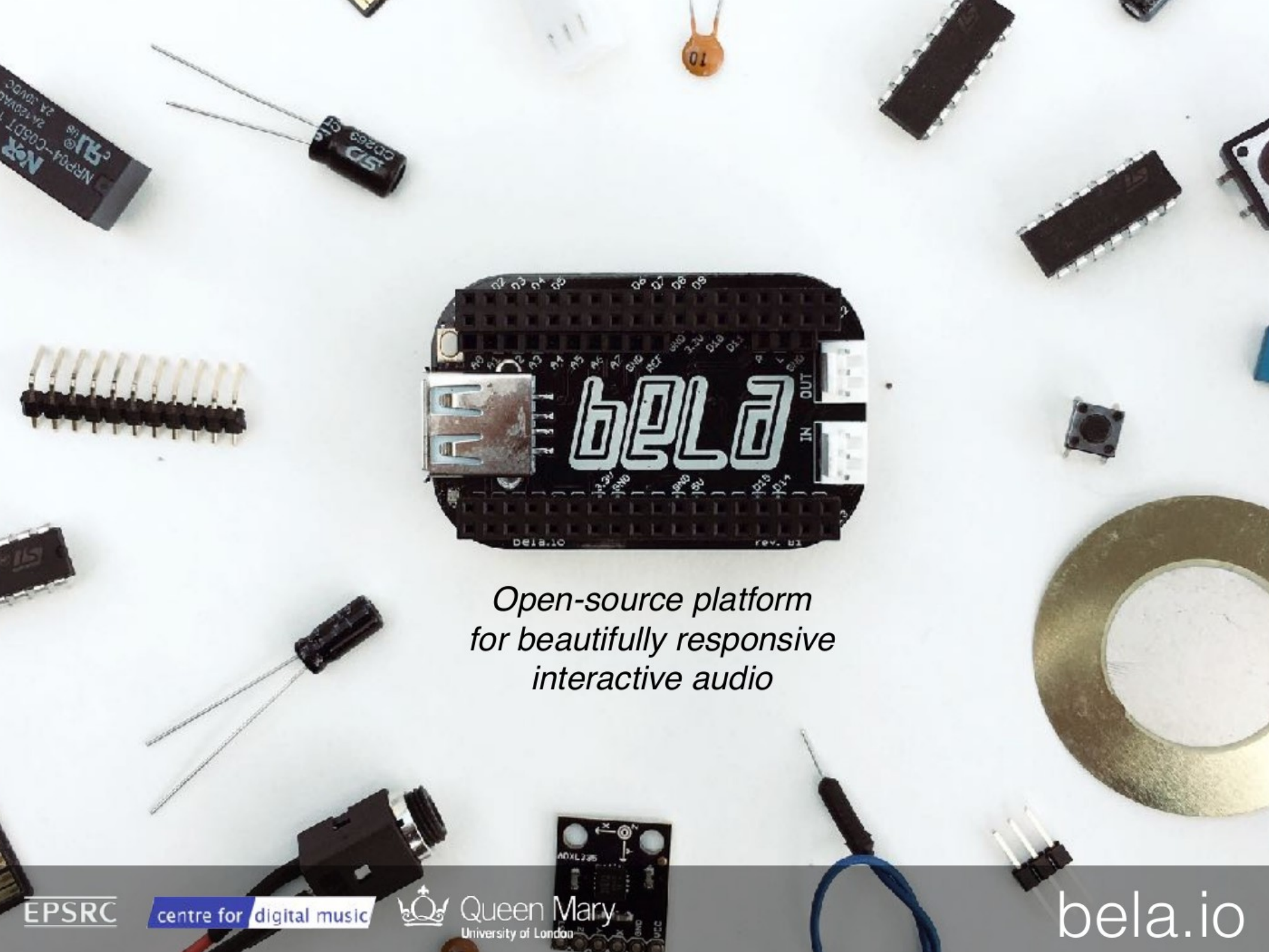
Jorge Hoppe

- http://retrocmp.com/projects/unibone



UniBone: PDP-11 card emulator

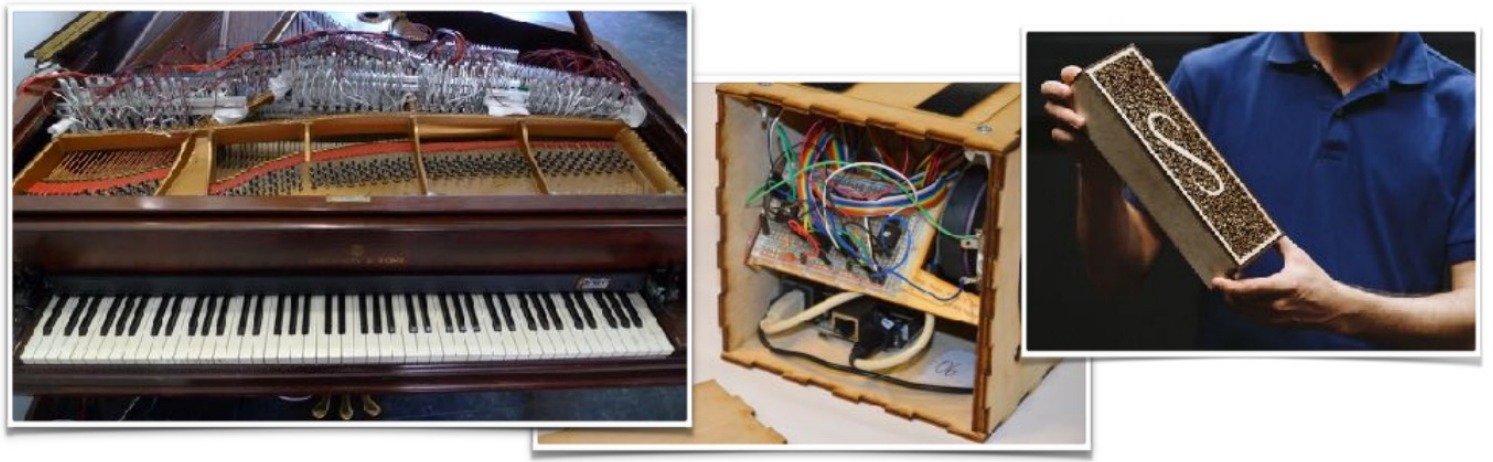
- Emulate PDP-11 disk controllers and disk
- Devices in separate pthreads. To verify these devices I let the PDP-11 CPU exercise them with DECs "XXDP" diagnostics.
- The emulator-threads must have realtime priority (SCHED_FIFO), else some diagnostics complain about violated timing constraints (obviously when other threads get preempted)
- When the PDP-11 CPU does an register access via UNIBUS to PRU to my devices, best case a delay of about 50 microseconds until the Linux threads get activated (via PRU interrupt)
- If not SCHED_FIFO, that delay can be 10-20 milliseconds.



Open-source platform
for beautifully responsive
interactive audio

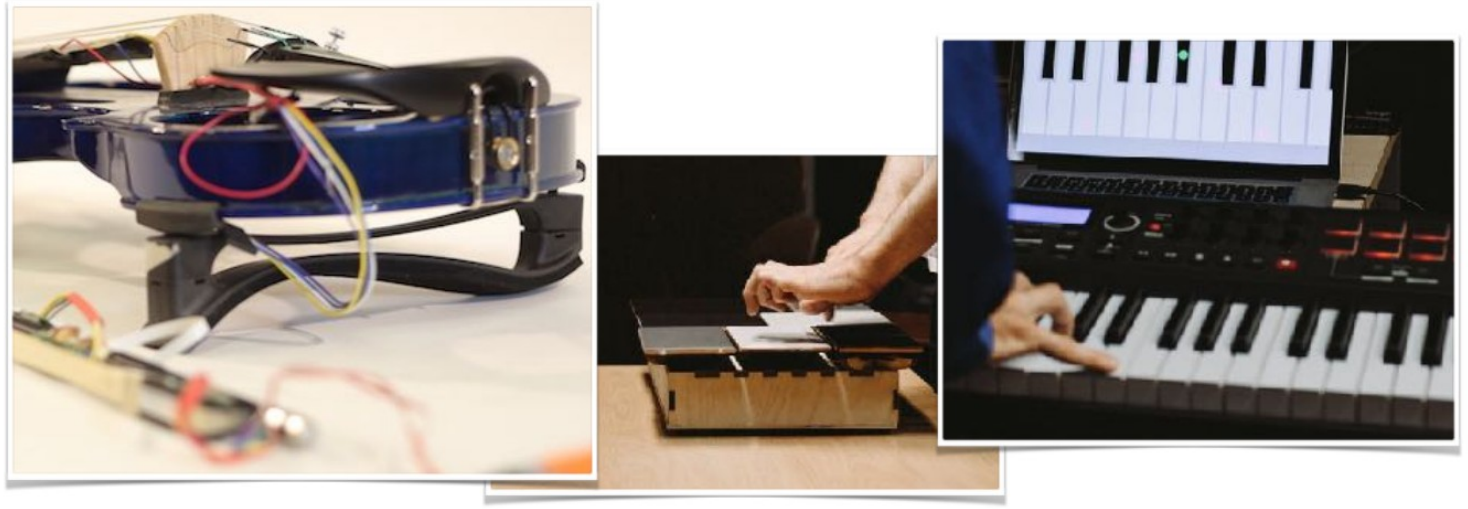
EPSRC centre for digital music Queen Mary University of London

bela.io

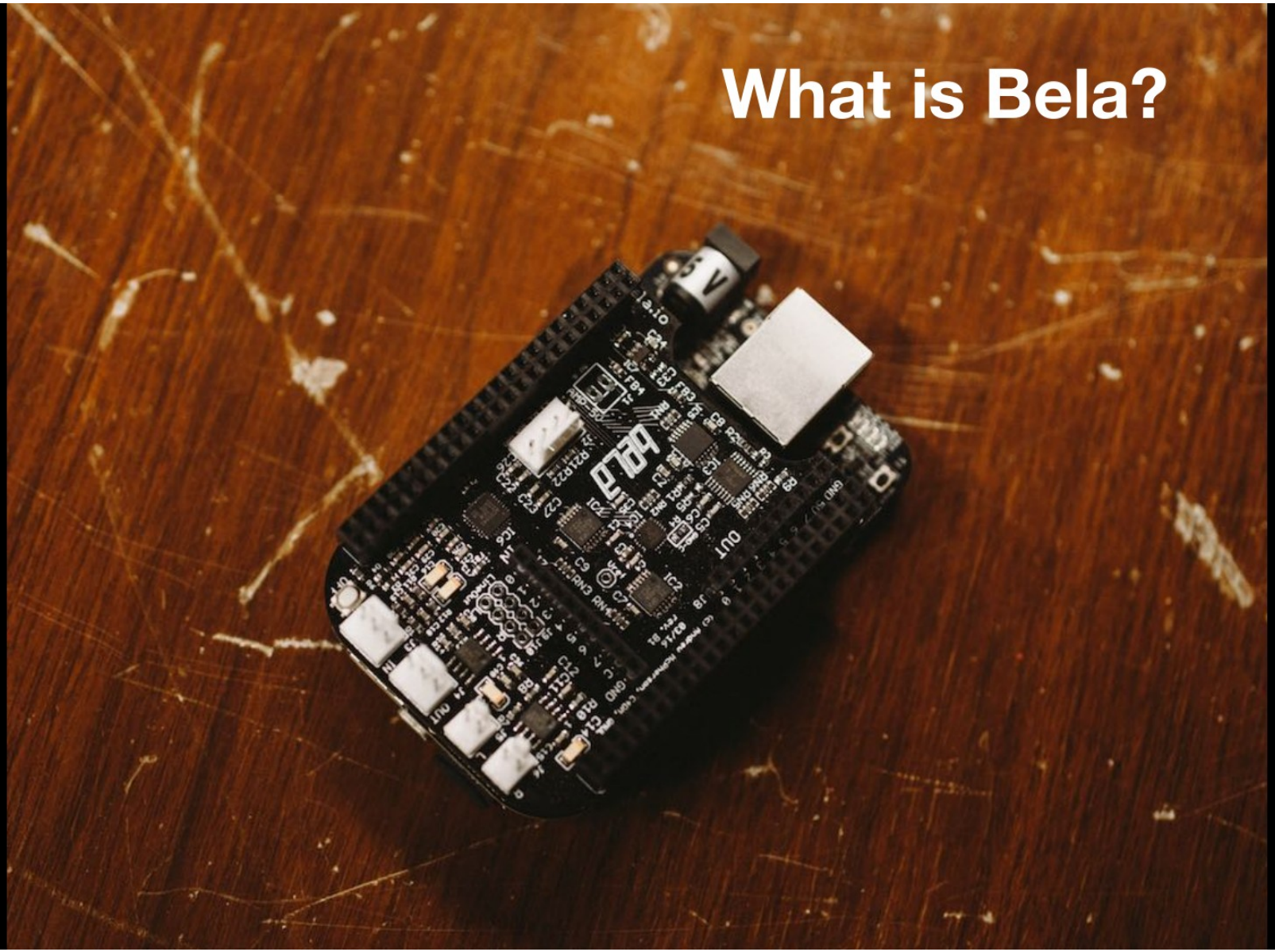


A project of the Augmented Instruments Laboratory:

http://instrumentslab.org

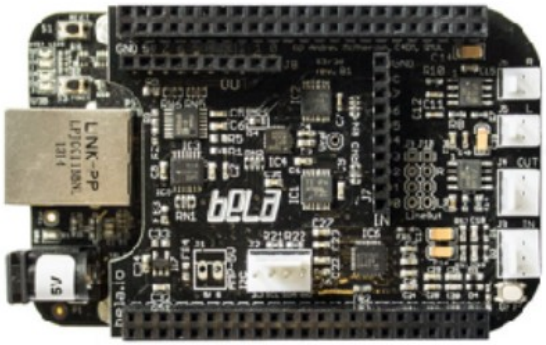


What is Bela?



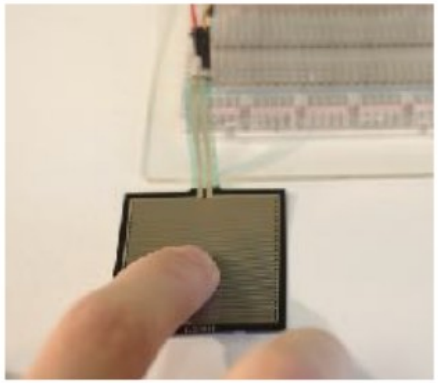
What is Bela?

Embedded computer designed for interactive audio



- Power of a single board computer
- Connectivity of a micro-controller
- Combines the benefits of both

New approach to high-bandwidth sensor processing



- Analog, digital I/O sampled at audio rate
- Ultra low action-sound latency
- Jitter-free alignment between audio and sensors

Open-source maker platform

- Open hardware and software
- Targeted at musicians, artists
- Online community resources: forum, wiki, blog.

Latency

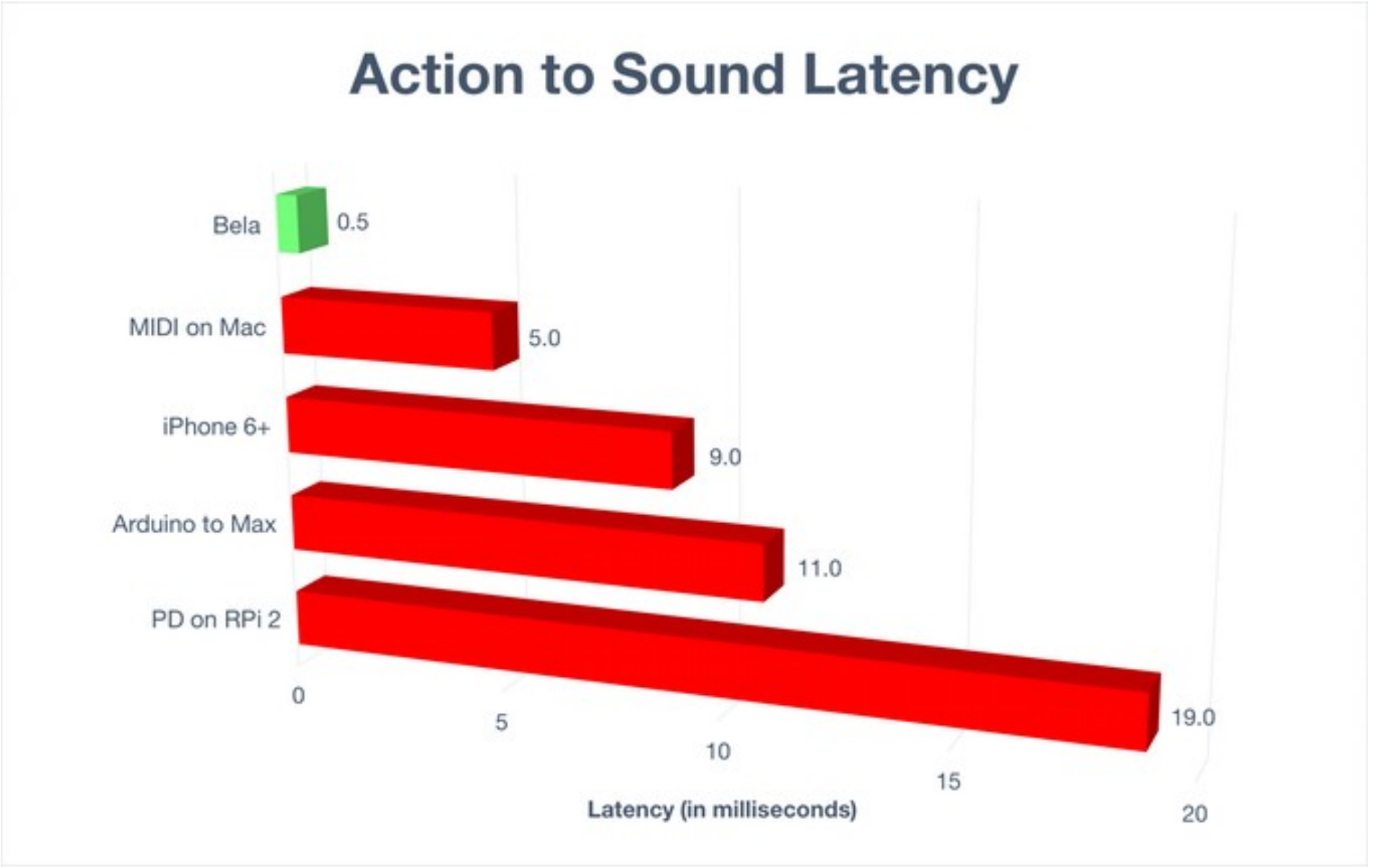
Does it matter? Yes!

- Long latency causes an audible delay
  - But even an imperceptible delay may make an instrument feel less responsive to play
- How low is “low enough”?

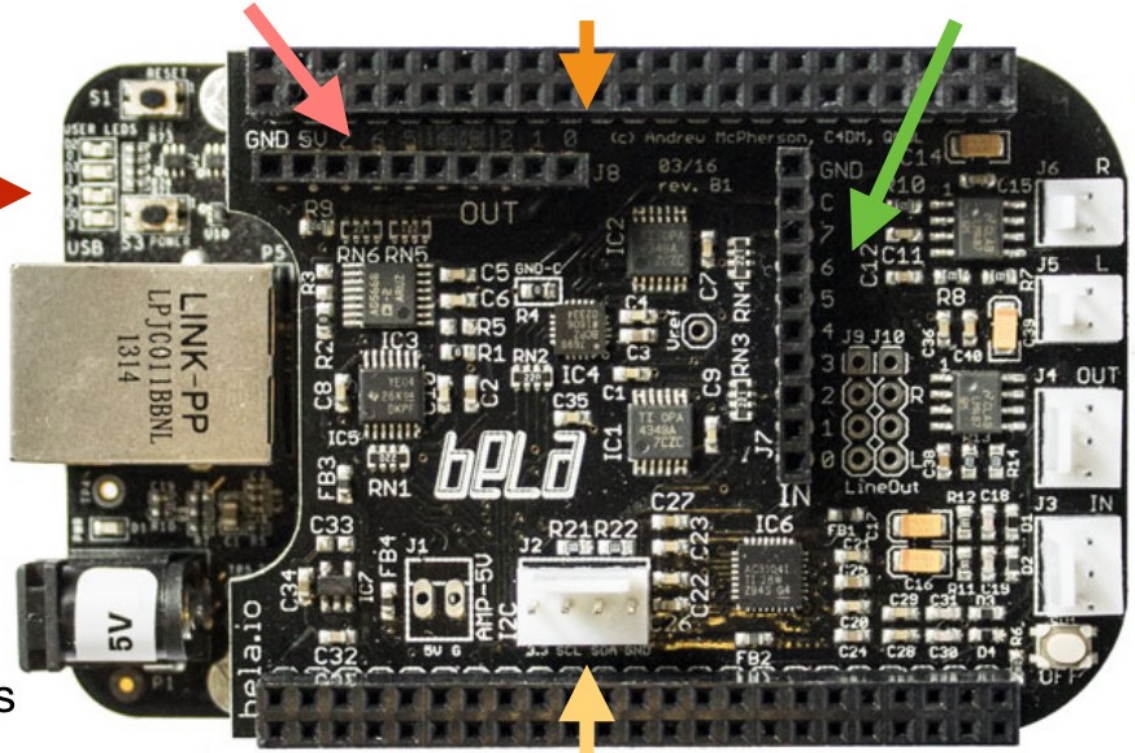
> “We place the acceptable upper bound on the computer’s audible reaction to gesture at 10 milliseconds. ... Low variation of latency is critical and we argue that the range of variation should not exceed 1 ms.”
>
> -- David Wessel & Matthew Wright, “Problems and Prospects for Intimate Musical Control of Computers”, Computer Music Journal, **2002**.

- Surprisingly few systems for building digital musical instruments meet this threshold!

A. McPherson, R. Jack and G. Moro. “Action-Sound Latency: Are Our Tools Fast Enough?” Proc. NIME, 2016.

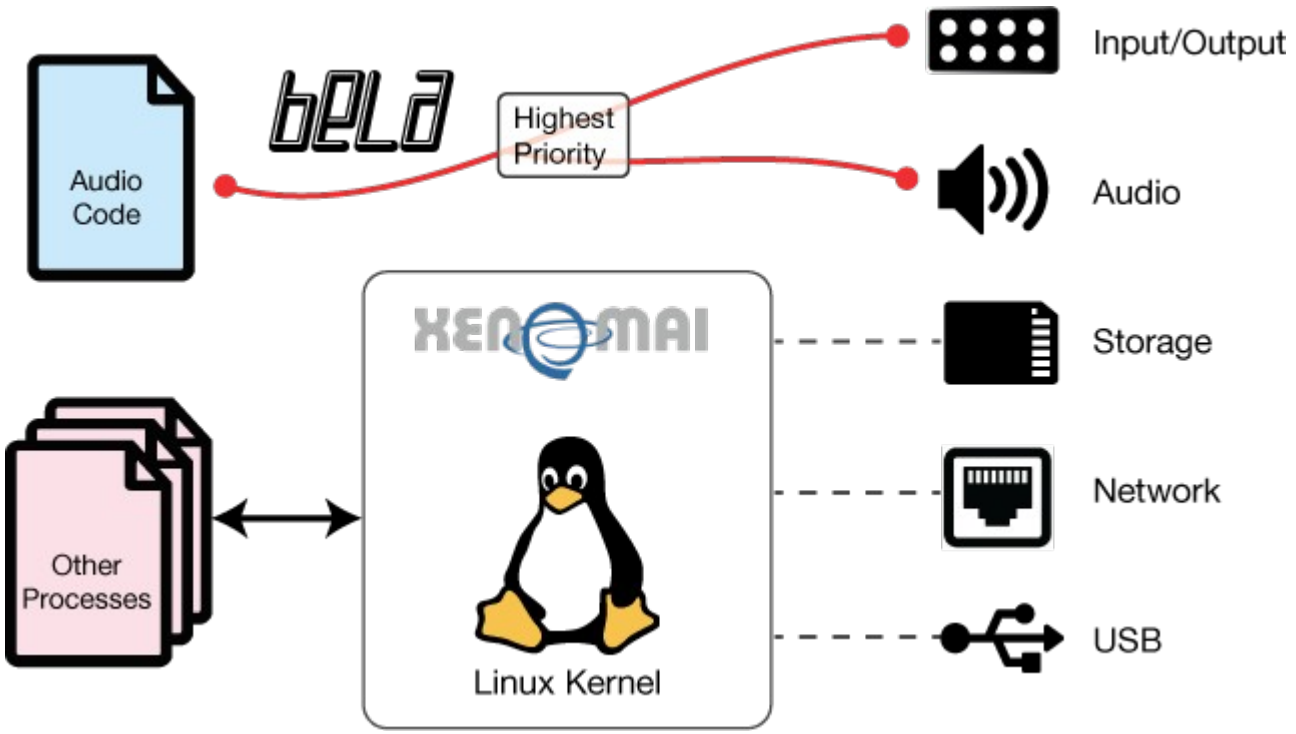

Action to Sound Latency
Bela
0.5
MIDI on Mac
5.0
iPhone 6+
9.0
Arduino to Max
11.0
PD on RPi 2
19.0
0
5
10
15
20
Latency (in milliseconds)

Bela hardware



Bela software

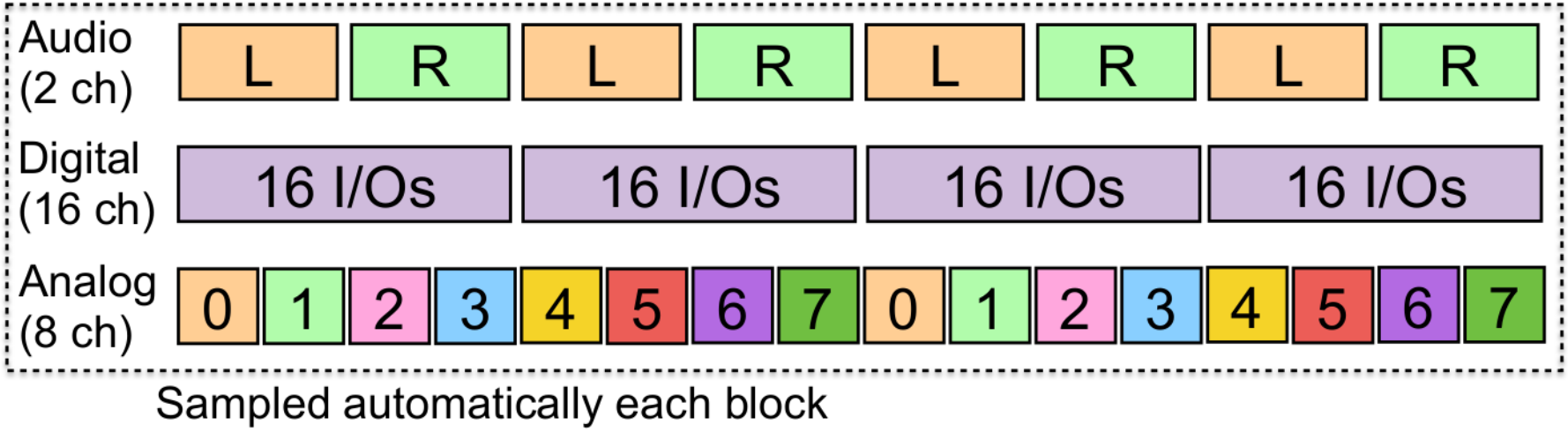
- Bela uses the Xenomai real-time Linux extensions to run audio code at higher priority than the entire OS



- Allows buffer sizes as small as 2 audio samples
- Digital and analog sensors sampled at audio rate, synchronously to audio clock for near-zero jitter

Bela features

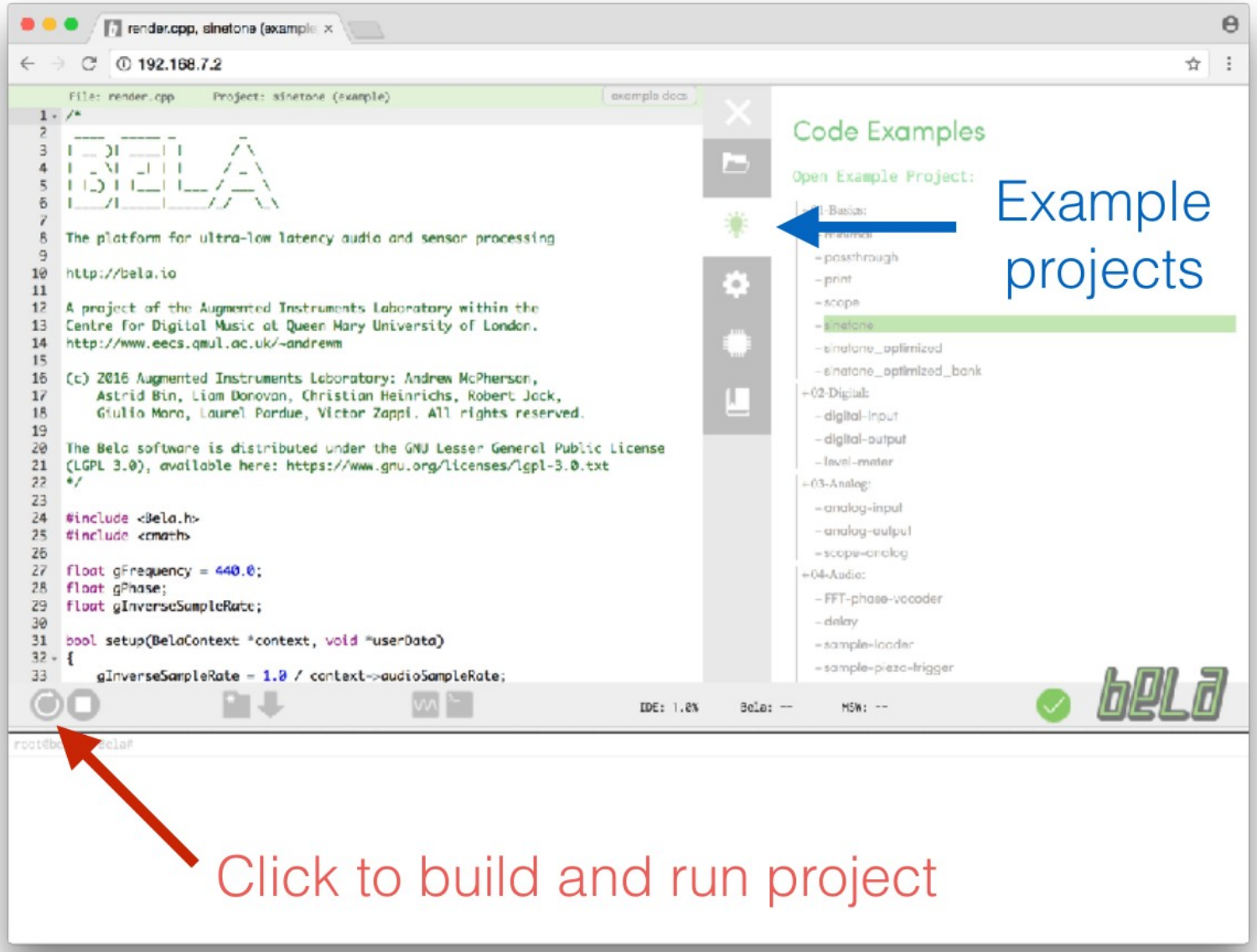
- High sensor bandwidth:
  - Digital I/Os sampled at 44.1kHz
  - Analog I/Os sampled at ≥ 22.05kHz
- Jitter-free alignment between sensors and audio
- Hard real-time audio/sensor performance but full Linux APIs still available
  - USB, network, filesystem, etc.



Sampled automatically each block

Bring up the Bela IDE:

http://bela.local/ (or 192.168.6.2 on Windows)



C++ API

- In **`render.cpp`**....
- Three main functions:
- `setup()`
  runs once at the beginning, before audio starts
  gives channel and sample rate info
- `render()`
  called repeatedly by Bela system ("callback")
  passes input and output buffers for audio and sensors
- `cleanup()`
  runs once at end
  release any resources you have used
- Code docs available in sidebar of IDE, or at docs.bela.io

Example: sinetone

```
#include <Bela.h>
#include <cmath>

float gPhase = 0.0; /* Phase of the oscillator (global variable) */

void render(BelaContext *context, void *userData)
{
    /* Iterate over the number of audio frames */
    for(unsigned int n = 0; n < context->audioFrames; n++) {
        /* Calculate the output sample based on the phase */
        float out = 0.8 * sinf(gPhase);

        /* Update the phase according to the frequency */
        gPhase += 2.0 * M_PI * gFrequency * gInverseSampleRate;
        if(gPhase > 2.0 * M_PI)
            gPhase -= 2.0 * M_PI;

        for(unsigned int channel = 0;
            channel < context->audioOutChannels; channel++) {
            /* Store the output in every audio channel */
            audioWrite(context, n, channel, out);
        }
    }
}
```

This runs once per block

This runs once per sample in the block (audioFrames gives the number)

This runs twice per frame, once for each channel

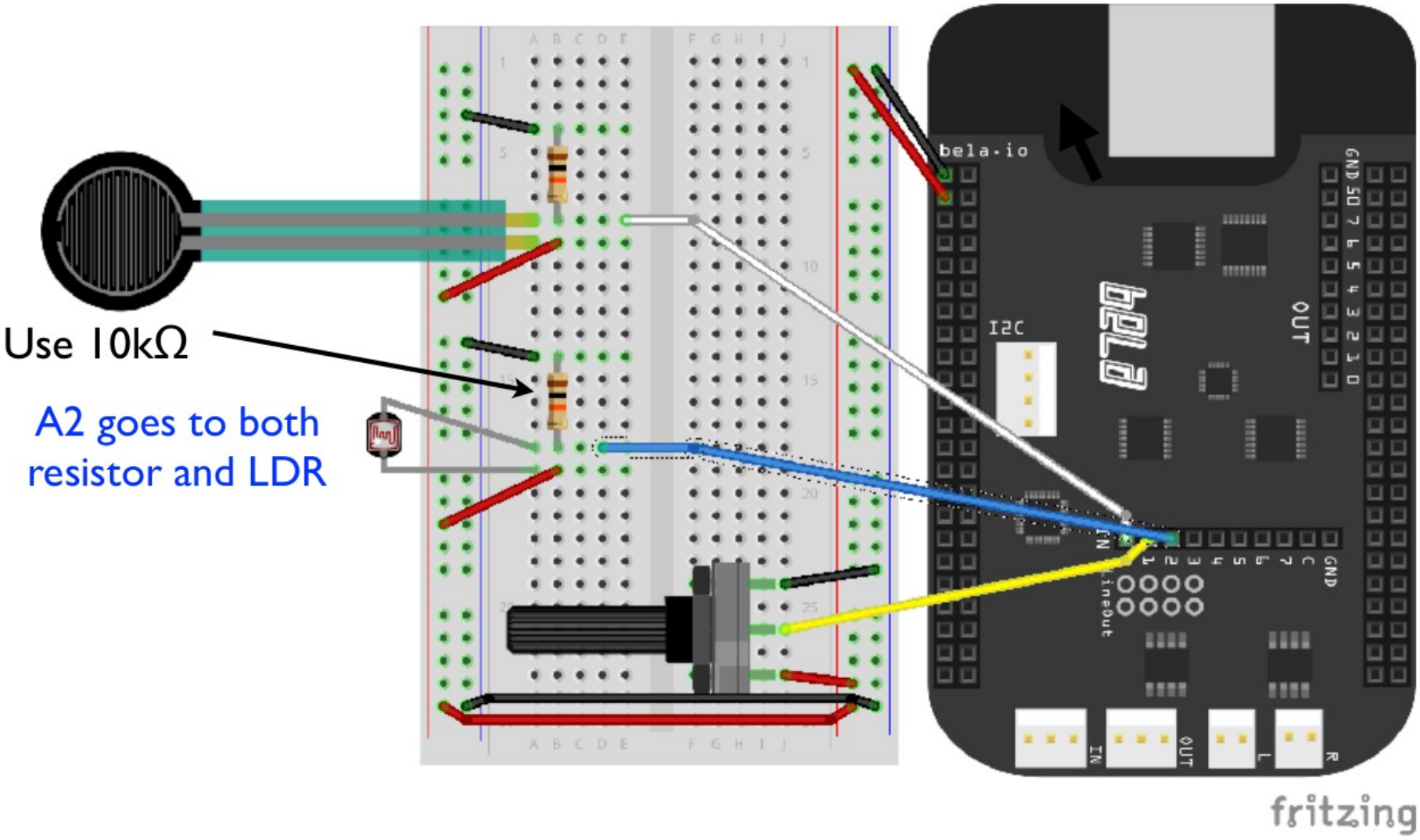
write to buffer of interleaved audio data, specifying a frame, a channel and a value

Queen Mary University of London

Light sensor

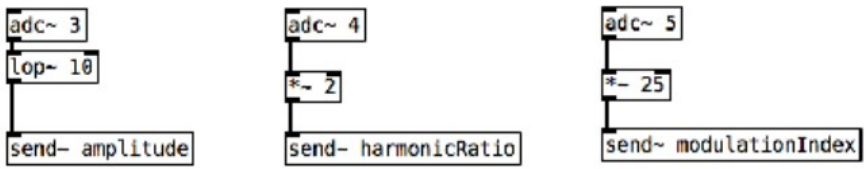
examples/00-Workshop/5-fm-synth

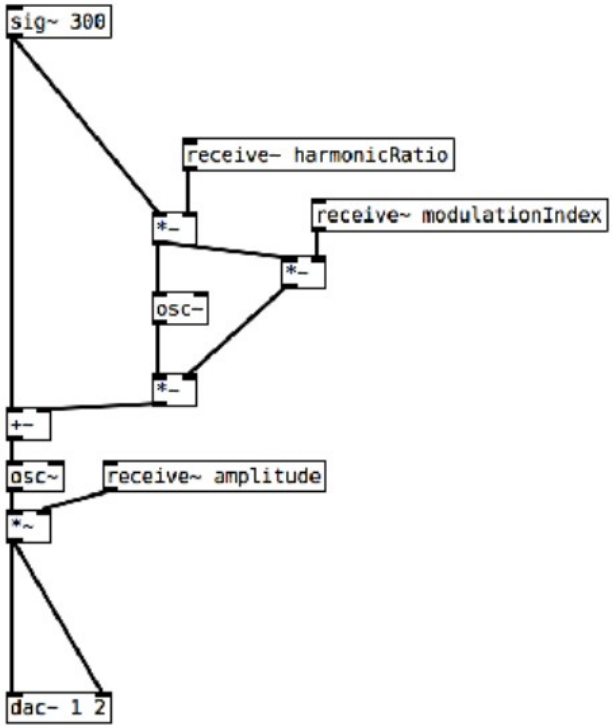
Add this to the existing components on the board



Light sensor

examples/00-Workshop/5-fm-synth



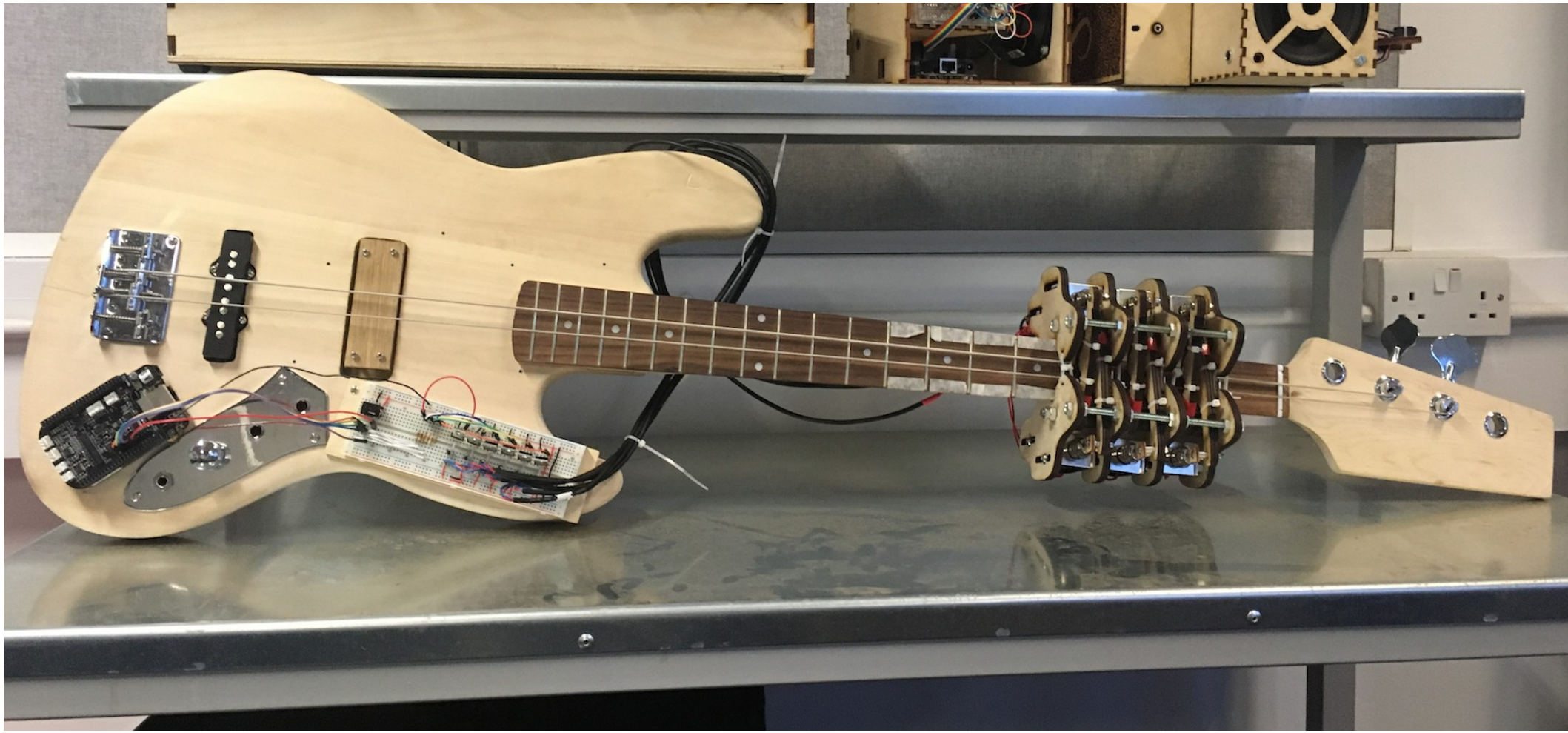


TASK 1: map the LDR to an appropriate range by using the scope

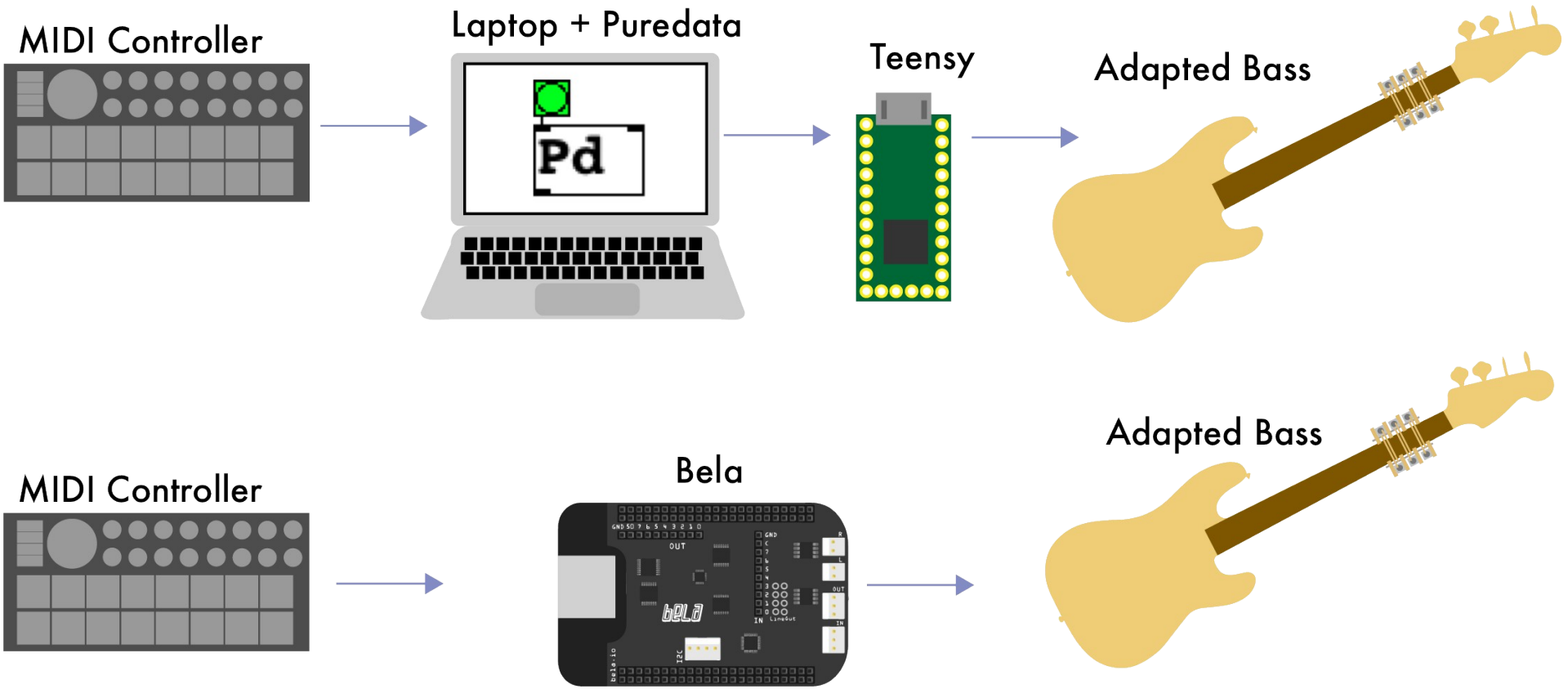
TASK 2: use the FSR to trigger an envelope once over a threshold

TASK 3: EXTRA try to connect simple-sequencer.

Bela one-handed bass



Bela one-handed bass




Adapted/One-Handed Bass Guitar (Mk2)
0:24 / 0:38
HD

Bela: real-time deadlines

- Xenomai 3.0.7 with 3.1 in current testing
- Kernel 4.4.113-ti-xenomai-r149, current testing 4.14.108-ti-xenomai-r122 (i-pipe bug just fixed)
- 2 or 4 samples per block. Sampling rate is 44100 Hz.
- The audio callback is called every 2 or 4 samples
- Periodic task that needs to wake up every 45us or 90us.
- If the thread wakeup latency is larger than that value, then the thread will not wake up on time to even run when it's meant
- If the thread takes a significant amount of that time to wake up then it will have very little CPU time to actually perform the computation it has to perform

Bela: Xenomai vs. PREEMPT_RT

- Giulio Moro from the Bela team has done some specific latency measurements
  - This was with 4.4 PREEMPT_RT versus 4.4 Xenomai co-kernel (i-pipe)
  - Xenomai i-pipe bug in 4.14 prevented moving to newer, fix found last week
  - Bela will soon be able to compare between 4.14 PREEMPT_RT and 4.14 Xenomai (i-pipe)
- “the thread wakeup latency is smaller on average and worst case for xenomai vs preempt rt, which means fewer time spent waiting for the audio thread to start executing and more time spent processing audio.”
- “average latency is as low or lower on RT, but that the ceiling is lower on Xenomai and therefore that is more dependable for real-time audio applications”
- “doubt RT would work reliably at 2 or 4 samples. That would mean having a repeatable thread wakeup latency below 45 and 90us respectively. While I think RT can achieve the latter fairly often, I don't think it can hit the former not even half of the time.”

Questions for PREEMPT_RT:

- What are the expectations could developers have for latency deadlines?
  - 100 uS? 50 uS? 10 uS?
  - Thread wakeup latency?
  - What are best practices for measuring their use case?
- Reasons to switch from Xenomai to straight RT?
- What PREEMPT_RT is new enough to see improvements?
  - 4.14? 4.19? 5.x?
- 1GHz Cortex A8 versus dual 1.5GHz Cortext A15 (BeagleBone AI)?