2009 Linux Plumbers Conference Portland, Oregon

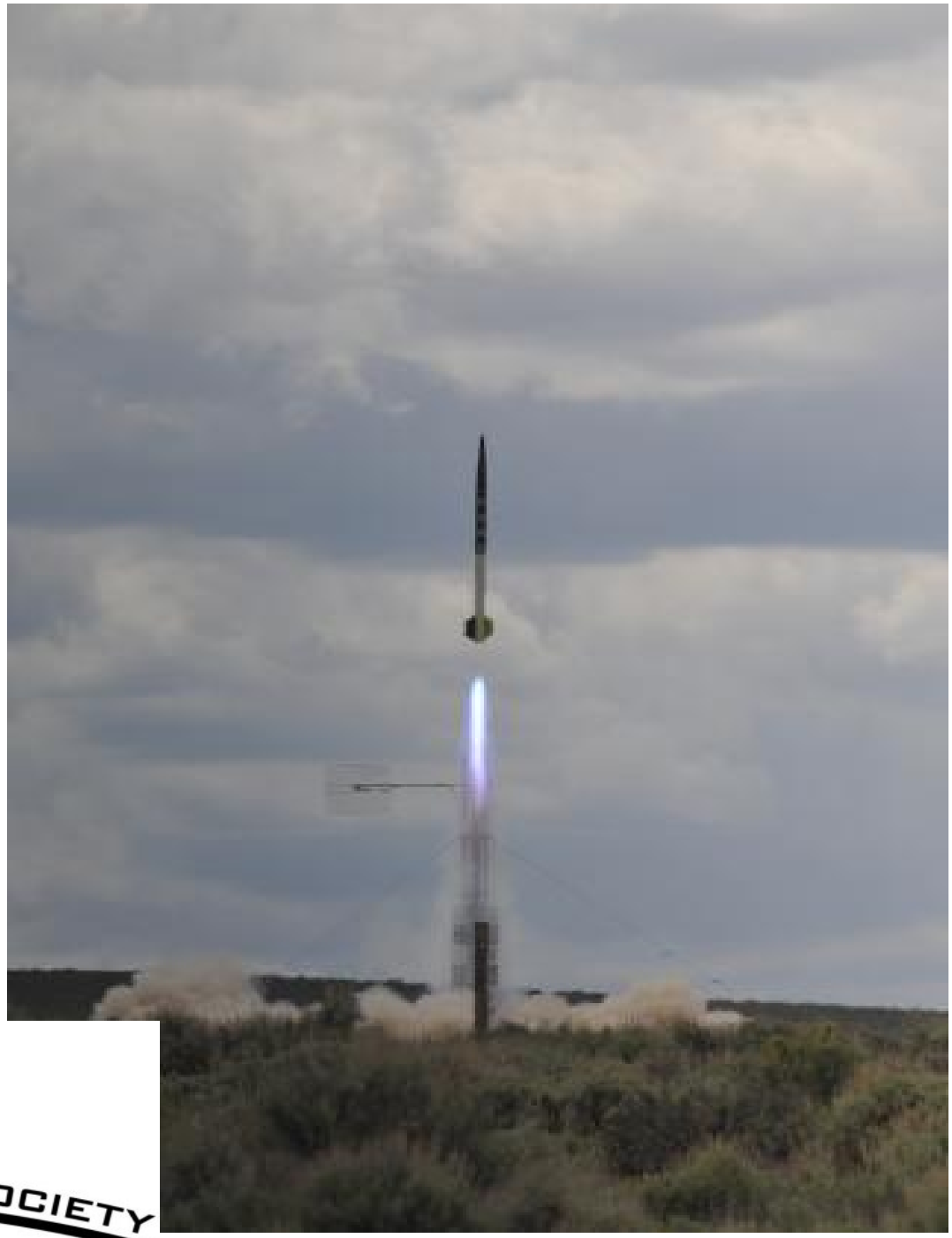
Challenges with Userspace USB Embedded Device Interfacing

Dave Camarillo
K Wilson

Background...

Portland State Aerospace Society http://psas.pdx.edu

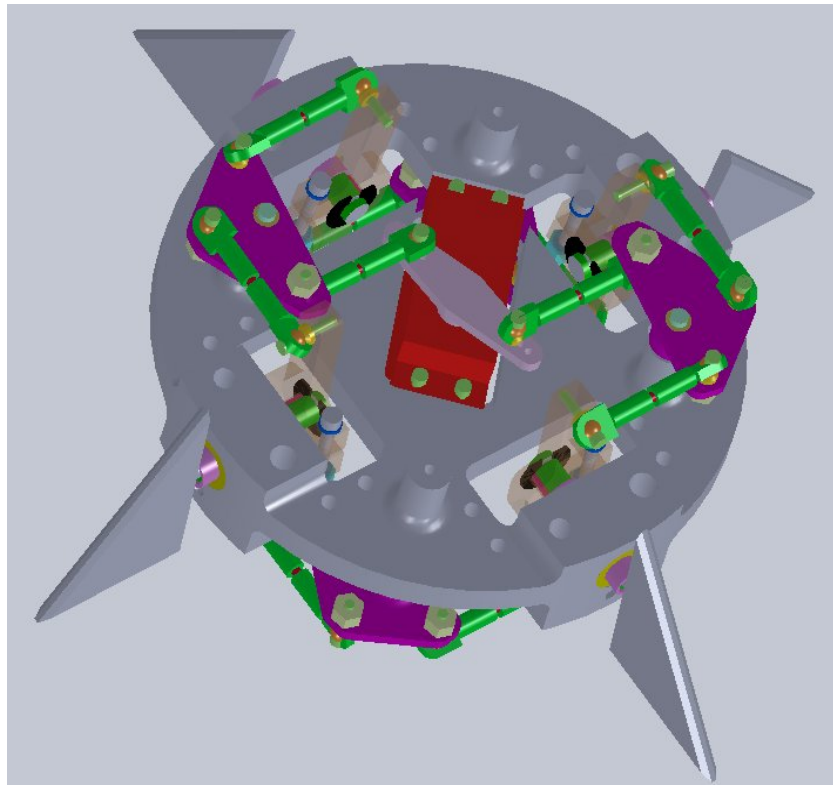
PSAS is a student aerospace engineering project at Portland State University. We're building ultra-low-cost, open hardware and open source rockets that feature perhaps the most sophisticated amateur rocket avionics systems out there today.



PORTLAND STATE AEROSPACE SOCIETY

What PSAS Wants to Do and Why

- Build avionics system on advanced amateur sounding rocket.
- WiFi at mach speeds
- Video downlink
- Gathering sensor data



Why?

- It's interesting
- It's exciting
- It's hard... if it were easy it would be boring...

Background...

Dave Camarillo

Dave is a professional software engineer with a background in embedded systems, medical device firmware, safety-critical industrial control systems, distributed high-availability clusters, large scale databases and cross-technology integration.

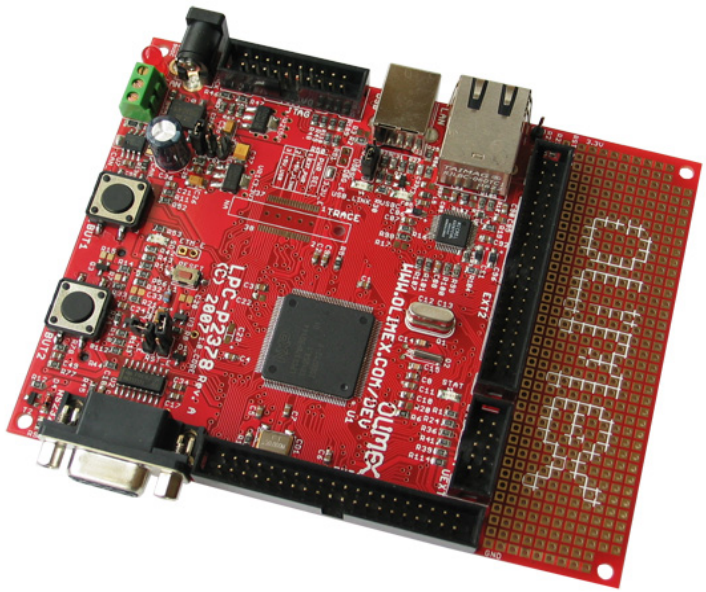
'K' Keith Wilson

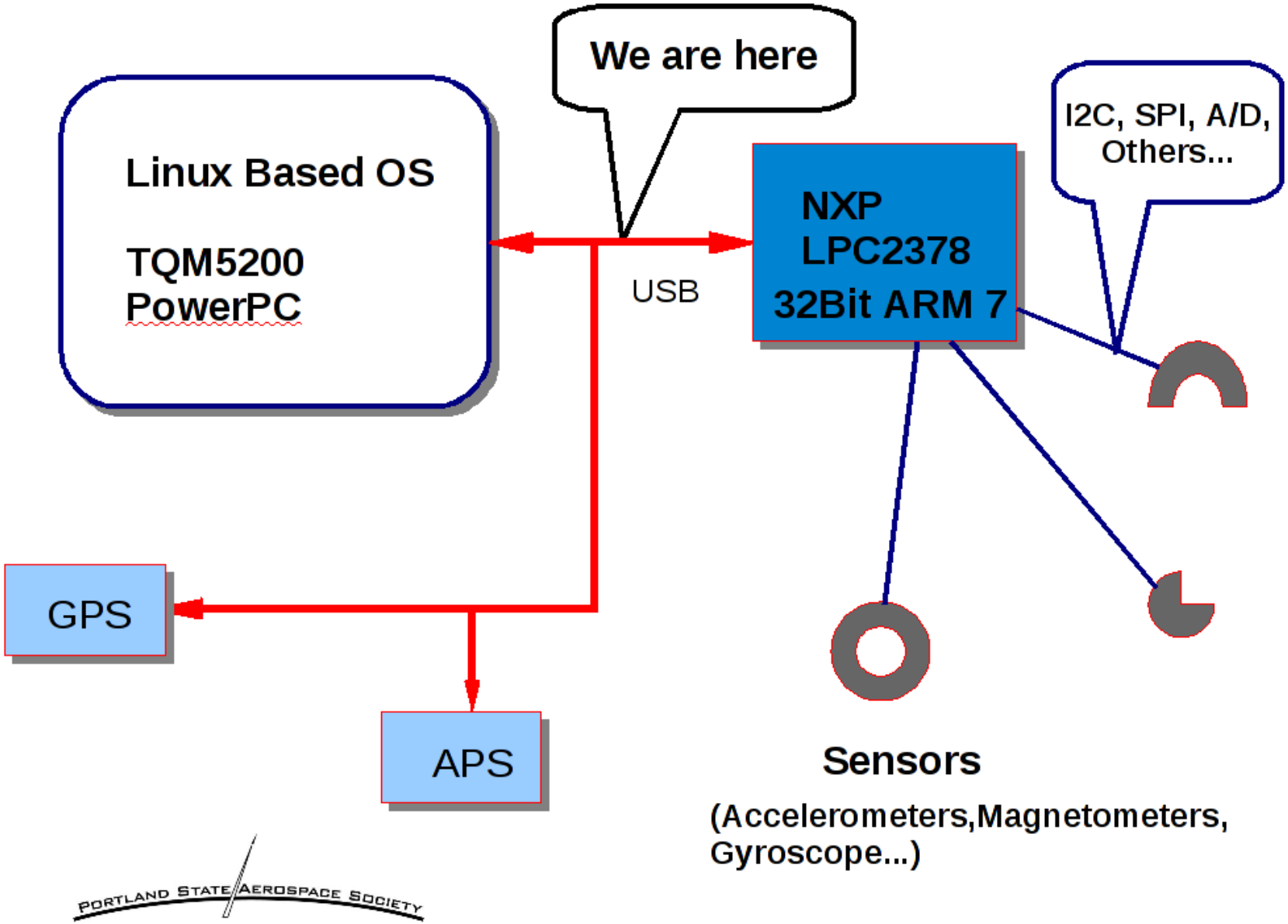
Keith is an MS Computer Science student at Portland State University with a previous background in VLSI/ASIC and microprocessor design.

Current Linux based flight computer
TQM5500 PowerPC

Connected with wires ('usb cable') to...

The NXP LPC Arm7 CPU



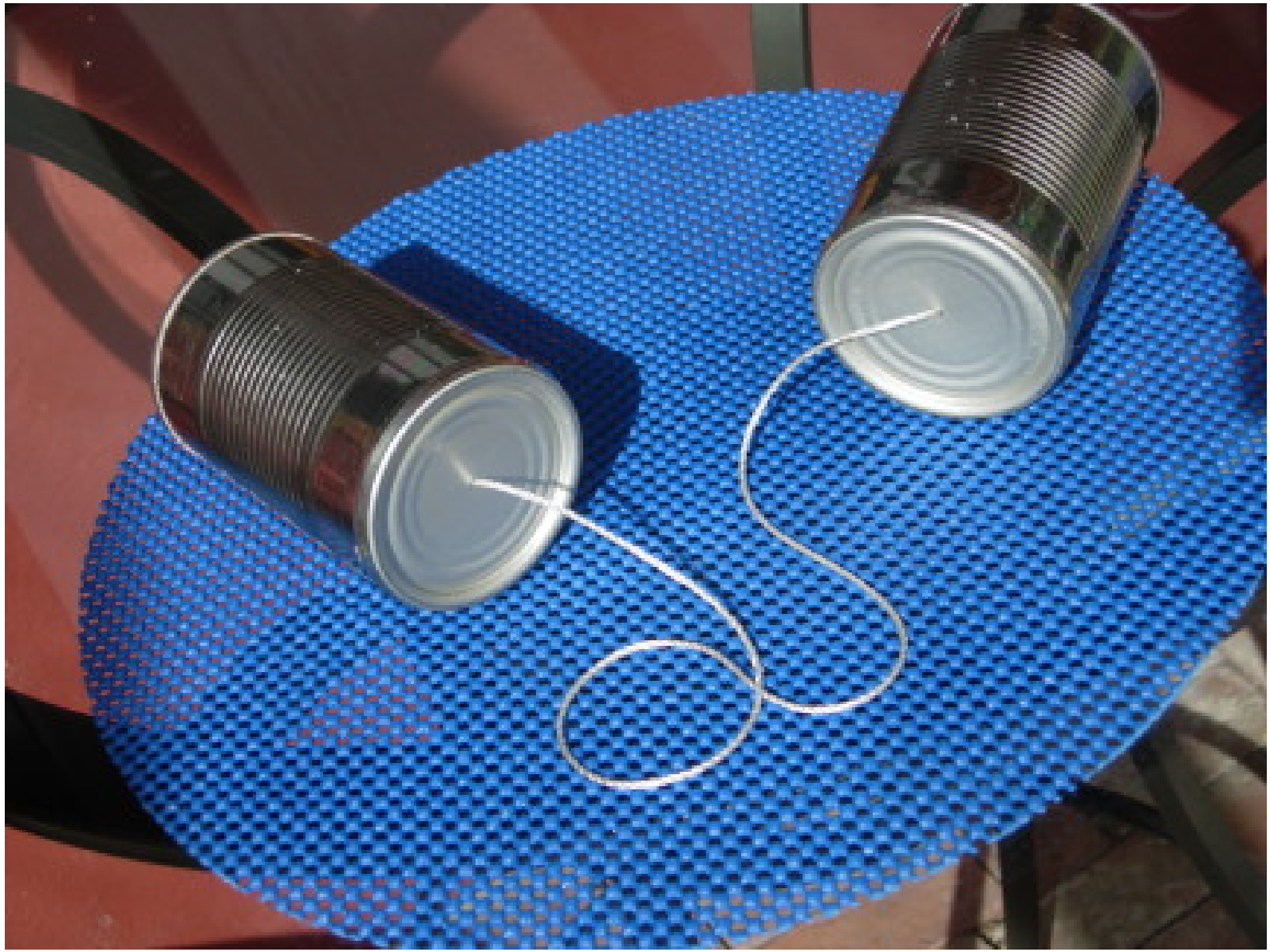

We are here
Linux Based OS
TQM5200
PowerPC
USB
NXP
LPC2378
32Bit ARM 7
I2C, SPI, A/D,
Others...
GPS
APS
Sensors
(Accelerometers,Magnetometers,
Gyroscope...)
PORTLAND STATE AEROSPACE SOCIETY

Problem Space Technical Requirements

- Guaranteed latency of data transfers
  - Old sensor data is useless when moving at mach speed
- High bandwidth communications
  - High sample rates * numerous sensors = lots of data
- Communications technology that is low cost and readily available
- Communications technology that is reasonable to interface with a Linux host operating system

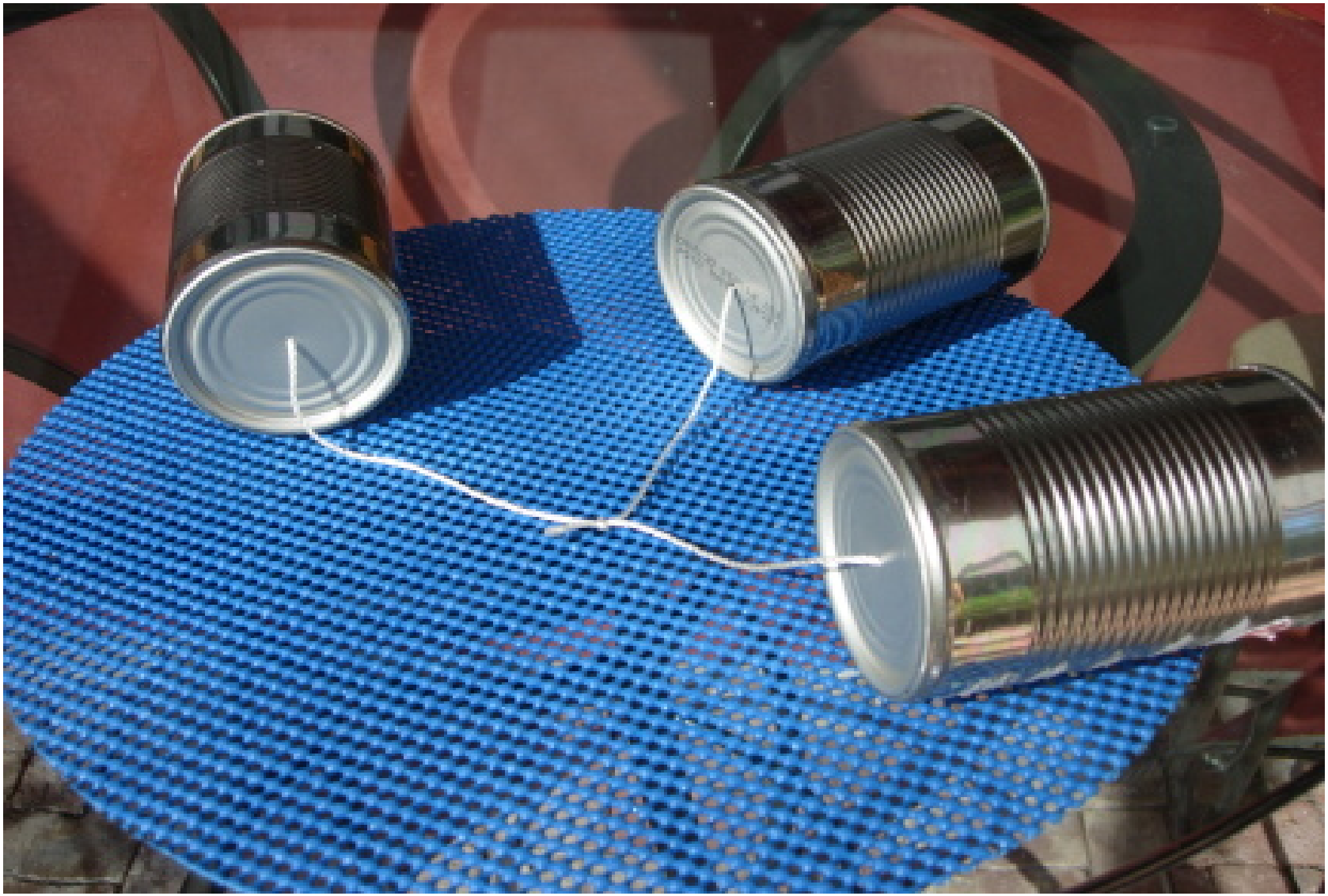
Latency V. Bandwidth OR "Please write again soon!"

Poor Bandwidth Serial Communication System



But:
Potentially Good Latency

What if...



More people want to join the conversation? Only one person can talk at the same time. Latency increases.

Possible solution: Divide time up into pieces and call them frames.

Negotiate how many words each person gets to say every frame.

A Little USB Background

Modes of Transfer

- Control Transfers: Used for device configuration

- Bulk Data Transfers: "Generated or consumed in relatively large and bursty quantities and have wide dynamic latitude in transmission constraints."

- Interrupt Data Transfers: "Used for timely but reliable delivery of data." Think keyboards.

- Isochronous Data Transfers

For Full Speed USB isochronous mode, each frame is 1ms, and a device can send a maximum of 1023 bytes during a frame.

How far does a rocket go at Mach 1.5...

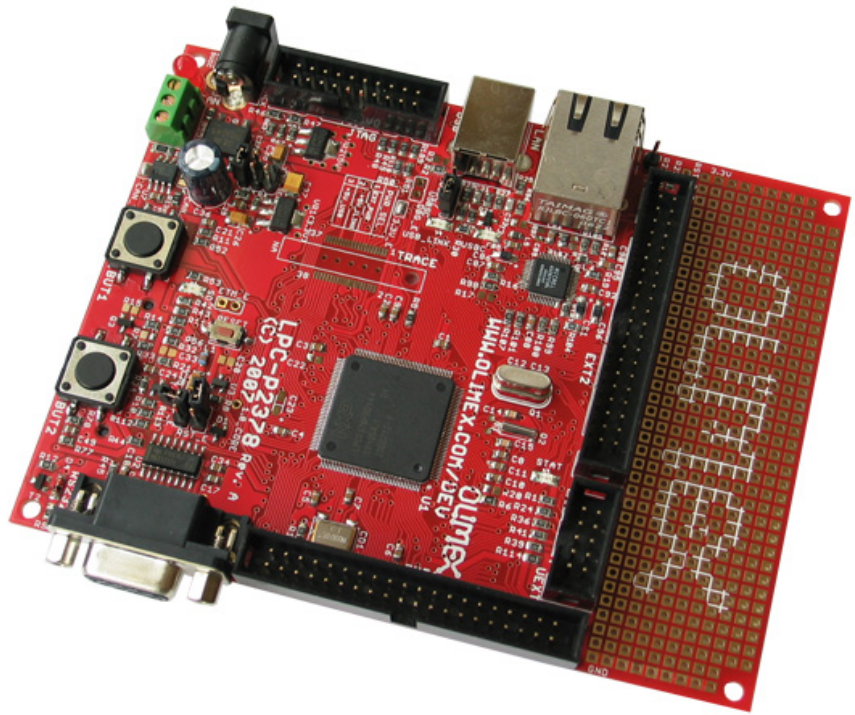
In 1 millisecond?
1.65 feet (0.5 meters)

In 1 Second?
1650 feet (500 meters)
(length of 5 soccer pitches)

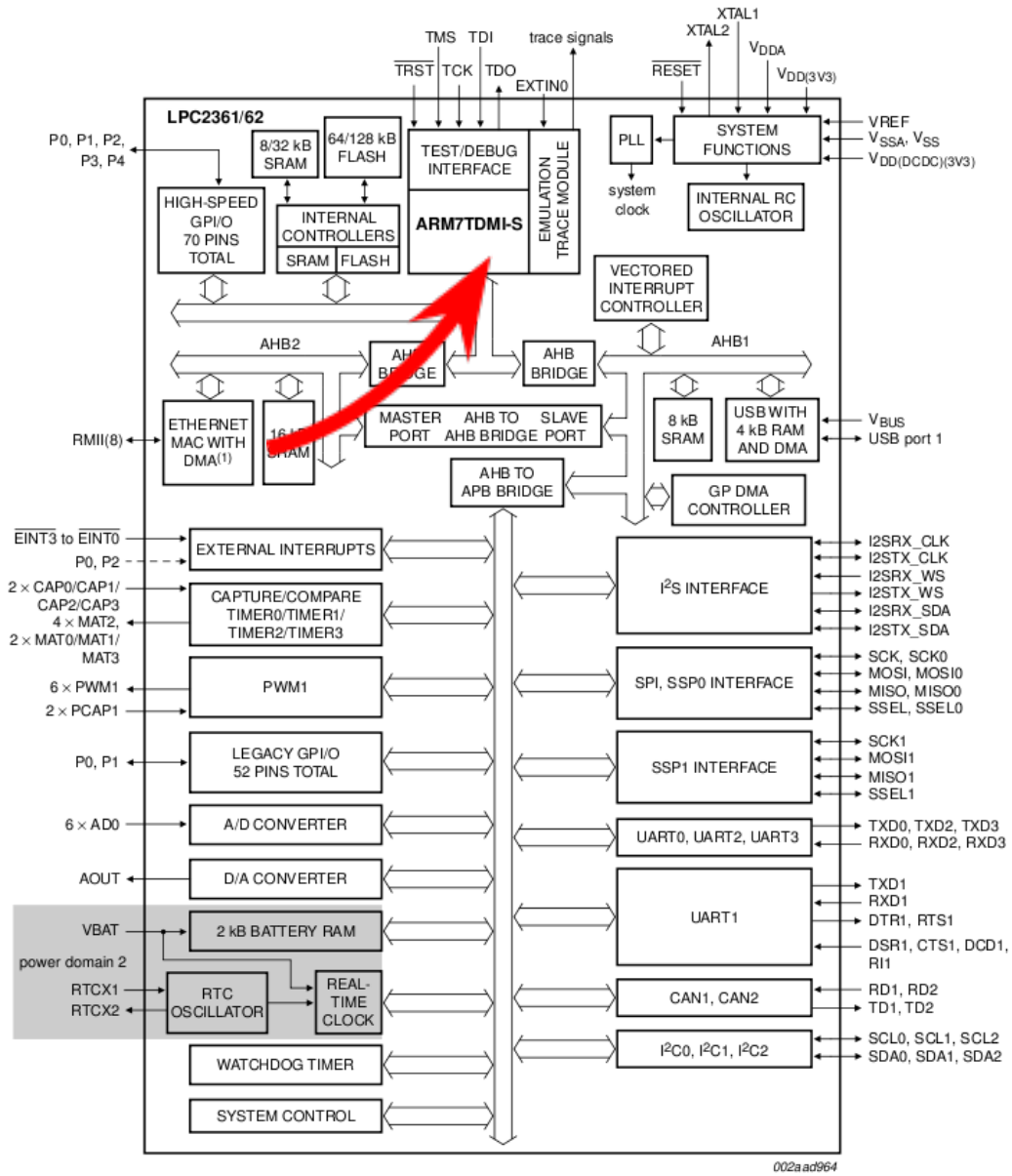
This is why a predictable latency is important in this application, more than the amount of data we can transmit between units.

The NXP LPC Arm7 CPU

- Arm core, up to 72mhz
- onboard USB, CAN, and many other perepherials
- gcc tool chain available
- openocd: jtag programming software, allows for gdb debugging, break points, flash memory manipulation



10. Block diagram



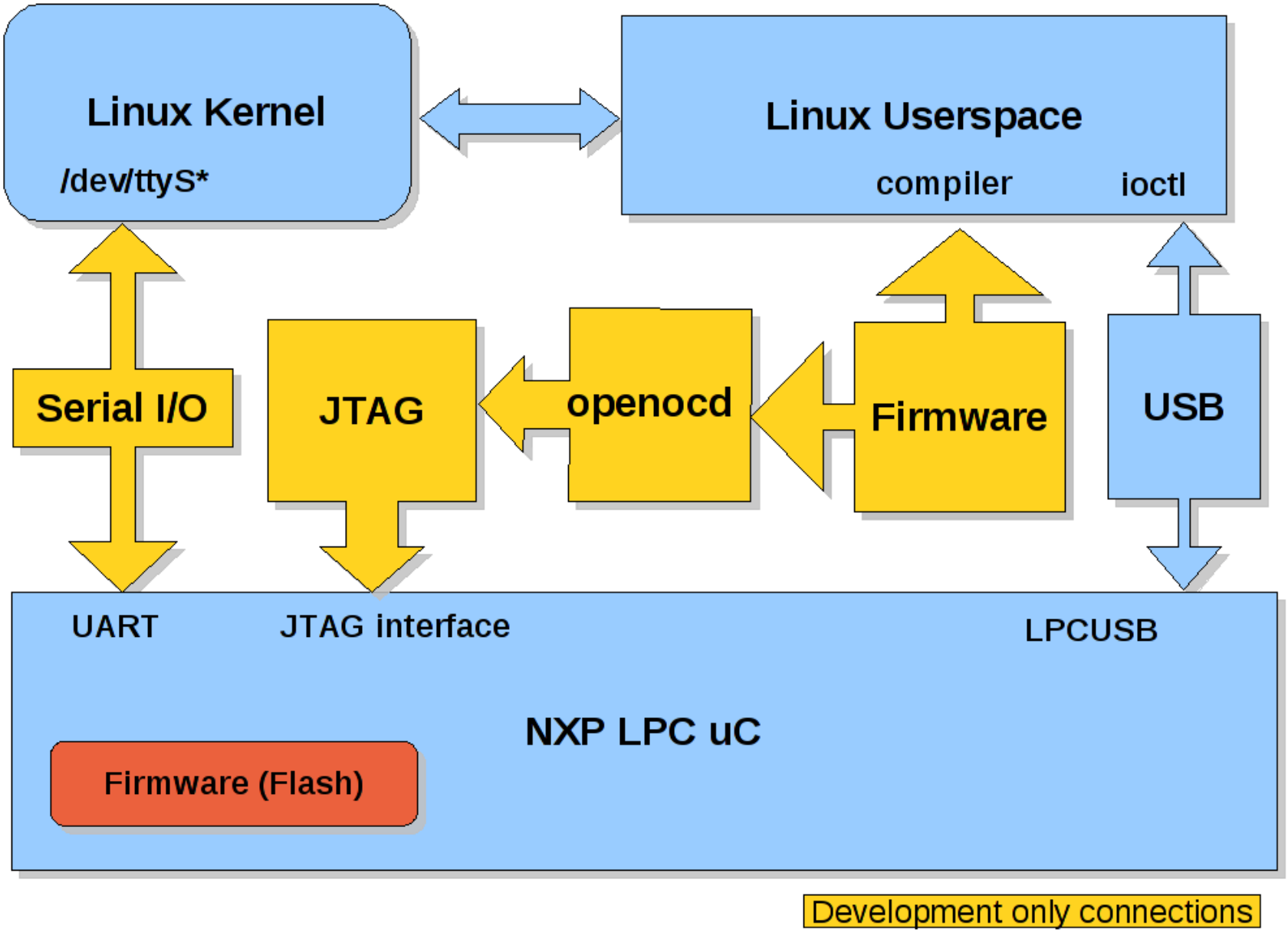
(1) LPC2362 only.

Fig 1. LPC2361/62 block diagram

Perspectives and Values...

- of a kernel developer
  - ideal, capable, flexible, correct kernel
- of an embedded systems developer
  - reducing complexity
- of a widget making company
  - profitability, time to market
- of a hobbyist
  - the hobby

System Development Diagram



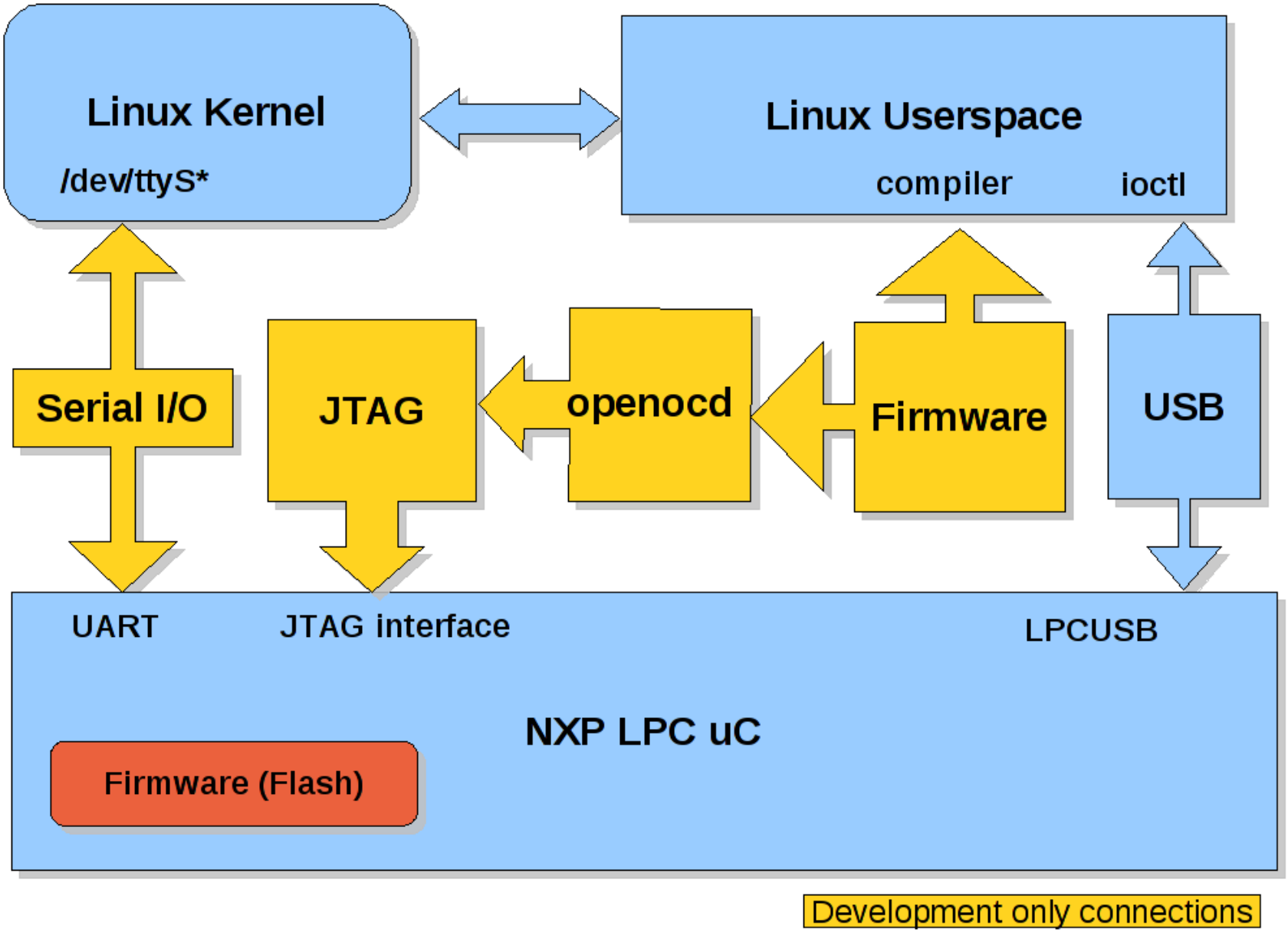
Kernel Space vs. Userspace Driver?

- Considerations:
  - Custom device interfacing - we're making dozens, at most, of these devices
  - Our group members have varying levels of development background, ranging from kernel development experts, to undergraduate CS students
  - We want it to be easy to test our devices an a variety of hosts, including our flight computer and individual's laptops or desktops.
  - Limited time

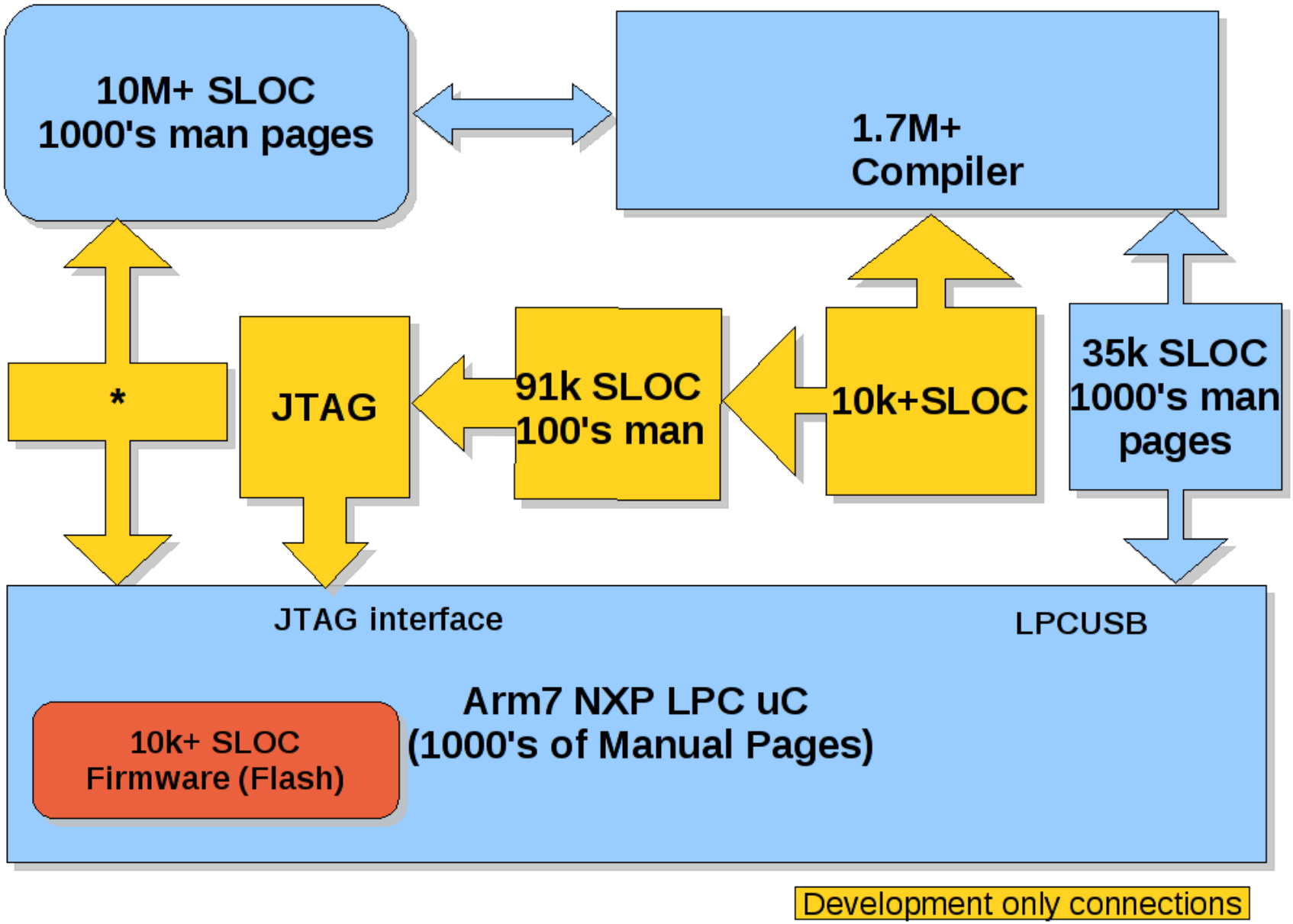
Kernel Space vs. Userspace Driver?

- There were a number of drawbacks to a kernel driver approach:
  - Learning curve for various members of the group regarding kernel driver development
  - Dangers associated with writing and testing custom kernel drivers, and not wanting to run such tests on a primary computer as we can trigger kernel panics.
- There were a number of benefits to a userspace driver approch:
  - Meets functional requirements of our project
  - Practical to test and debug by anyone in the group.

System Development Diagram



System Complexity in SLOC



Practicalities of Embeded Development

- Complex by it's very nature: if it was already built we'd by a COTS part for $5, not spend thousands on custom development
- Finicky hardware, behaves in mysterious ways, sometimes yields new errata, electrical problems with new circuit and board designs
- Development parts frequently don't work
- printf() is a luxury
- no virtual address space
- no segfaults, CPU crashes, hardware exceptions
- debugging USB ISR's is tricky - breakpoint it and the USB device drops off the bus

How People Learn...

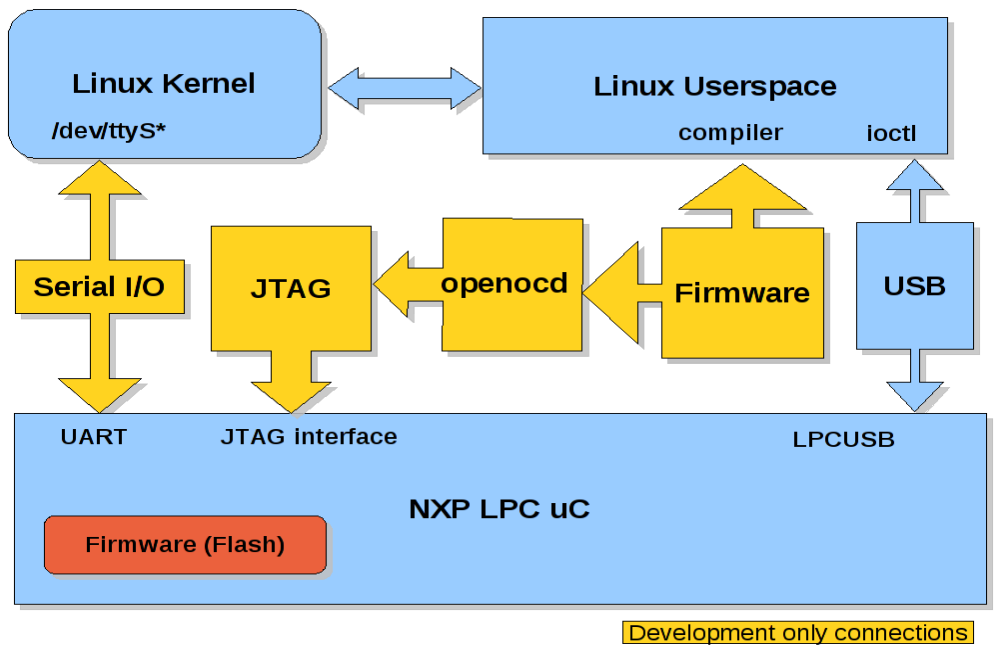
- "hello world"?
- "pthreads"?
- Taking on incrementally more complex problems
  - Novices are easily buried in details.
  - Experts work from abstractions, after recognizing the details.
- Making mistakes
- Reading
- Talking to others
- We learn quickest what we are most interested in.

Our Plan, Accounting for the Learning Curve....

- Get the sample LPCUSB Bulk ttyACM0 device running, as it's intended, with normal host drivers. (no variables)
- Write USBFS userspace code to communicate with Bulk ttyACM0 device (one variable)
- Modify LPCUSB firmware to do non-DMA isocronous transfers. Impliment USBFS userspace code to communicate with new firmware (two variables)
- Modify LPCUSB firmware to do DMA based isocronous transfers. Modify USBFS userspace code to handle additional speed. (the systems variable)

I wrote some code, it doesn't work - whats wrong?

- is it userspace code issue?
- is it incorrectly interfacing with the kernel?
- is it Arm7 USB perepherial misconfiguration issue?
- is it firmware code issue?
- is it a compiler bug?
- is it hardware issue?
- is it incorrect use of Arm7 CPU?



Debugging

Being able to eliminate possible sources of a problem is critical - divide and conquer

- usbmon was enormously helpful on the host side
- kernel logs were sometimes useful
- digging into the kernel source code was necessary for some types of problems
- indirect monitoring via UART was one of few semi-viable options for the Arm7
- Logic Analyzers Oscilloscopes and blinky lights are critical ('side effects')
- Verification of hypothesis

I Fixed the code, did I do it right?

- Inspection of reference code?
- Inspection of kernel source code?
- Never really convinced that what we've written is correct?

- Often don't know about a problem until it fails
- Having a knowledge of functions, structures, and knowledge about context and theory of operation of system
- How do judge when I've performed due diligence?
- ***All open questions....***

This is why we launch in the middle of the desert :)

Example: USBMON

Reference: ...kernel/Documentation/usb/usbmon.txt

May need to rebuild kernel with usbmon. (module is fine)
If/once installed, you should be able to view bus sockets.
ls /sys/kernel/debug/usbmon
0s 0u 1s 1t 1u 2s 2t 2u 3s 3t 3u 4s 4t 4u

Watch USB mouse data:
cat /sys/kernel/debug/usbmon/2u
f6527540 148092302 C Ii:2:003:1 0:8 5 = 0001fe00 00
f6527540 148092329 S Ii:2:003:1 -115:8 5 <
id timestamp event interrupt:bus2:add3:endp1 etc....
May be more or fewer entries to parse depending on transfer.

How does this apply to the big picture?

- If a piece of functionality is claimed to be supported, does it entail just the code, or is there more then code?
- To what extent are the perspectives and values of various different types of people or groups considered? and how well does a piece of functionality fit into the value system of the person or group in question?
- Developers form opinions about software interfaces based on their value systems, and if they are not matched the opinion can turn out poor.

References

- .../kernel/Documentation/usb
- Embedded device manual may have example usage code and discussion for device side implementation.
- libusb: http://www.libusb.org/
- lpcusb: http://sourceforge.net/projects/lpcusb/
  - even if you aren't using NXP LPC uC, you may find this helpful.
- Linux Journal #181, May 2009, "Linux-powered Amateur Rocket Goes USB", Sarah Sharp, http://m.linuxjournal.com/article/10421

Kernel 2.6.3x at Mach 1.5 soon....

October 3rd & 4th, 2009
Black Rock Desert, NV