

Real Programmers Debug With Fire Extinguishers

Software Controlling (Astronomy) hardware in the Real World[TM]

– Craig Kulesa –

GENERAL OUTLINE

- PHILOSOPHIES
- ARCHITECTURES
- PROTOCOLS (SPI, I2C, SERIAL, CAN, ETHERNET, ETC.)
- NETWORK SOCKETS, INDI AND OTHER GLUE LAYERS
- GUIS

PHILOSOPHY

- Simplicity is prerequisite for reliability. It is not a dispensable luxury but a quality that decides between success and failure.
- Use unix.
 - Everything looks like a file. hardware shows up as special nodes in /dev.
 - Write “programs” that do one thing and do it well.
 - Stitch them together to make the system do what you want.
 - Linux prevails in embedded systems but isn’t always the best fit.
- The C language is the lingua franca of embedded systems
 - The best match for speed, efficient use of resources, latency.
 - Python is becoming more prevalent when those things aren’t critical.

Open foundation: NetBSD



Linux is great, but NetBSD is also an excellent choice for developing embedded software and hardware.

Less chaotic development leads to more stable interfaces that are tested better. (documentation and developer skill can accumulate)

Less superfluous redesign – recognition that developer time is not '0-cost'.

Very portable by design rather than brute force (proper hardware abstraction, vs. patch on top of patch to make ARM look like a i386 to the kernel)

All features built from a single, unified source tree.

Cross-compilation toolchains and a single, efficient libc is an integral part of the build system. To build an ARM distribution from any POSIX system, you type one command. Doesn't matter what OS or what hardware -- I swear it's magical:

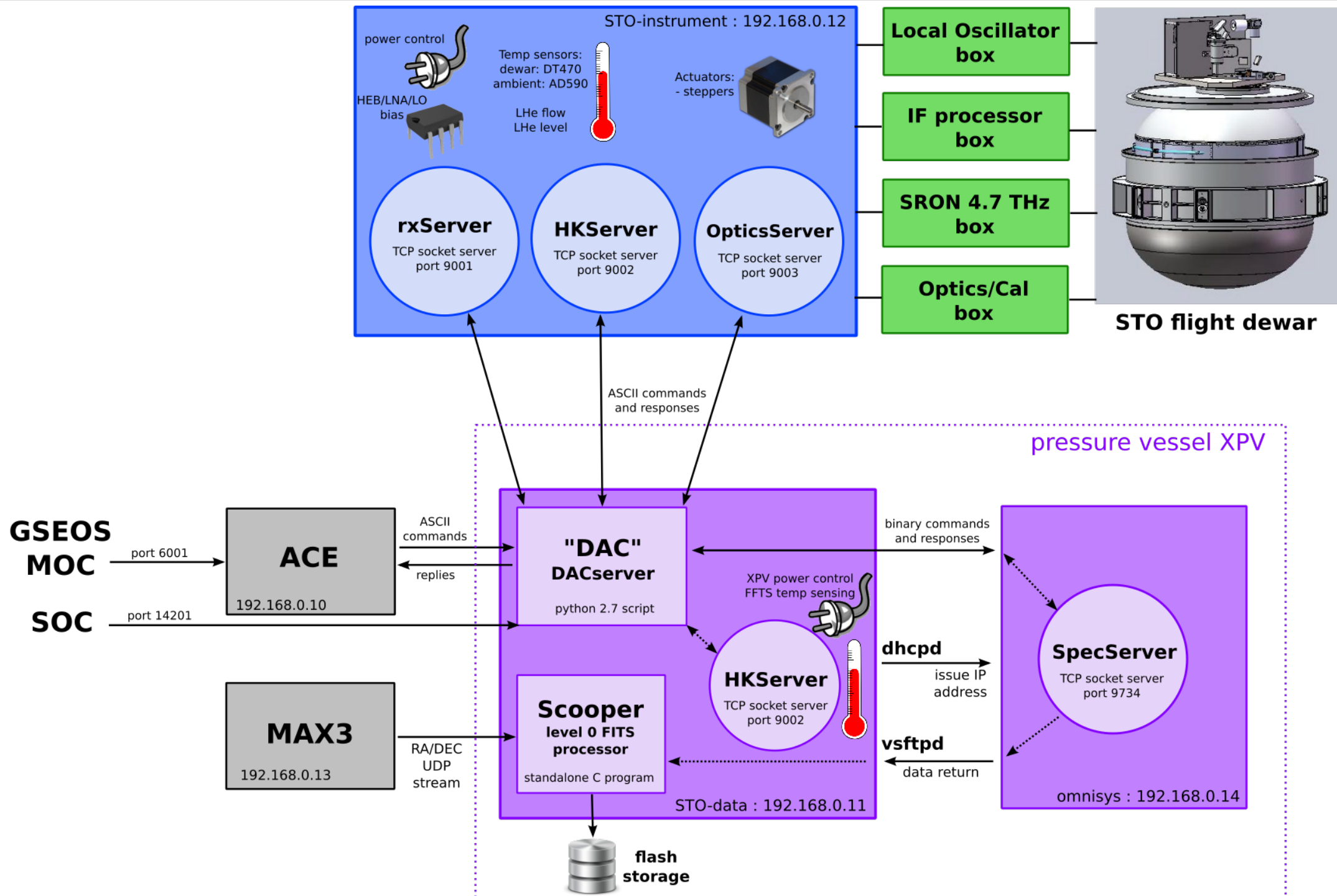
```
./build.sh -U -m evbarm tools distribution sets
```

Kernel debugging, dynamic tracing, kernel level core dumps has always been possible.

Very open BSD license.

ARCHITECTURE

- "Hardware: The Part of the System that can be Kicked"
 - Actually, the line between hardware and software is highly blurred
- Kernel-space vs Userspace
- Monolithic vs Distributed
- Hardware requirements
 - General purpose I/O pins (GPIO)
 - Ethernet
- Include a hardware-less "simulator" mode for standalone testing



PROTOCOLS : I. I2C AND SPI

- Simple 2-4 wire protocols to communicate with discrete electronics elements like DACs and ADCs, multiplexers, bus expanders, etc.
- I2C is 2-wire: a clock line and a bidirectional data line. A unique device address determines which device you are talking to.
- SPI is 4 wire: No addresses. You pull a Chip Select (CS) line to get a device's attention. Two unidirectional data lines handle transfers.

I2C

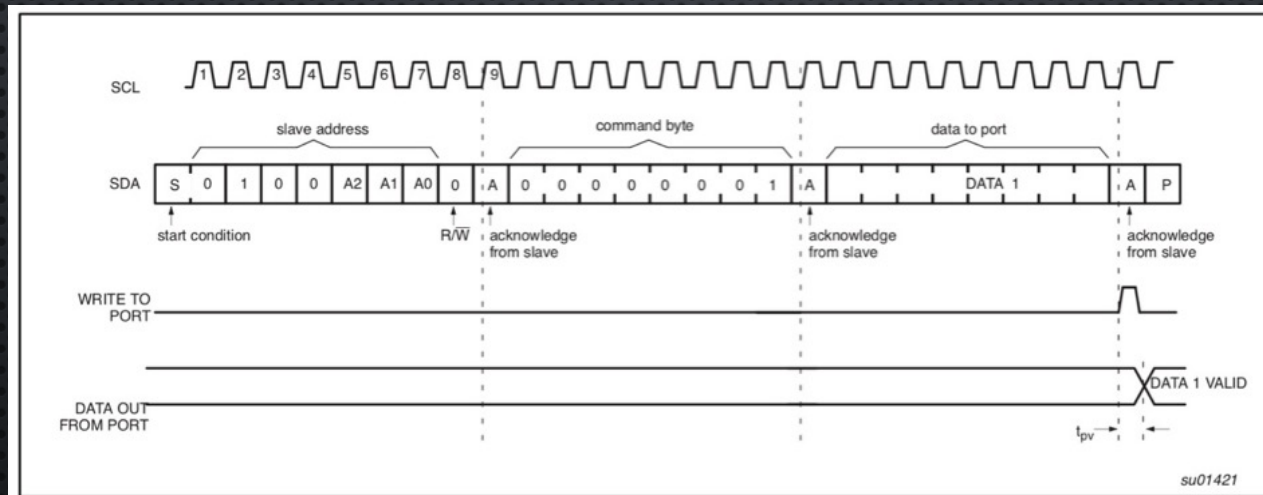


Figure 6. WRITE to output port register

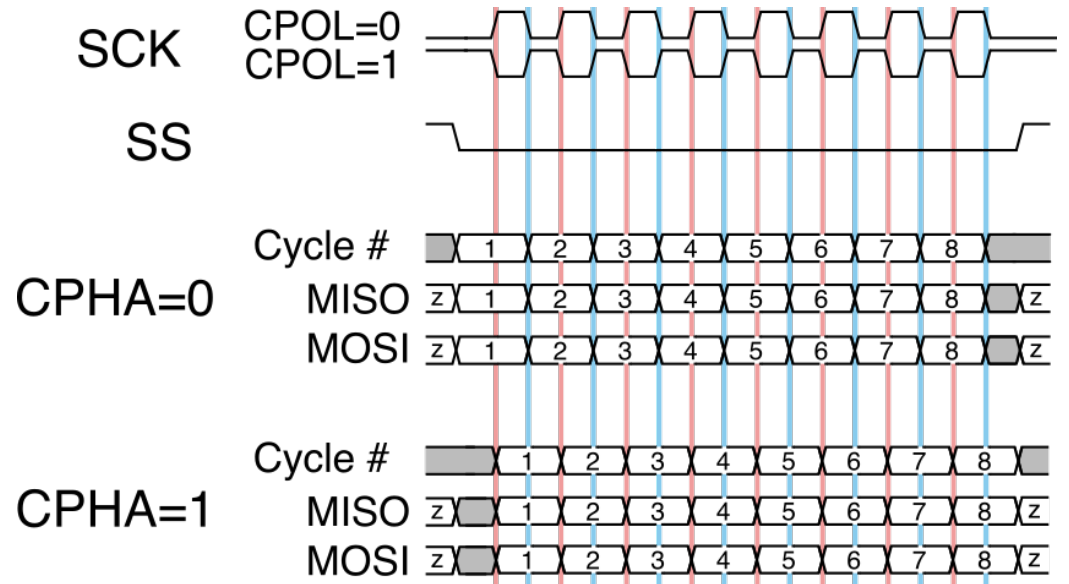
BARE WRAPPER TO SET A PIN STATE USING GPIO

```
void setI2C_scl(int val)
{
    struct gpio_req req;
    memset(&req, 0, sizeof(req));
    req.gp_pin = SCL; req.gp_value = val;
    ioctl(dio_fd, GPIOWRITE, &req);
}
```


BARE WRAPPER TO WRITE A I2C BYTE USING THE KERNEL

```
void i2c_write(int addr, int reg, int data)
{
    int file = open("/dev/i2c-0", O_RDWR);
    ioctl(file, I2C_SLAVE, addr);
    i2c_smbus_write_byte_data(file, reg, data);
    close(file);
}
```


HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC.)



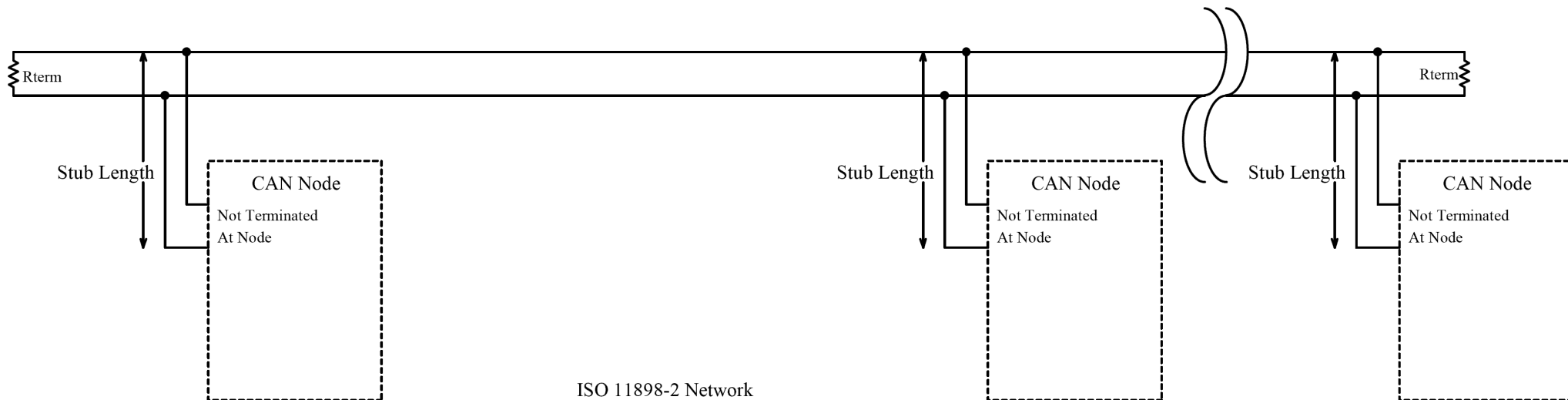
SPI modes

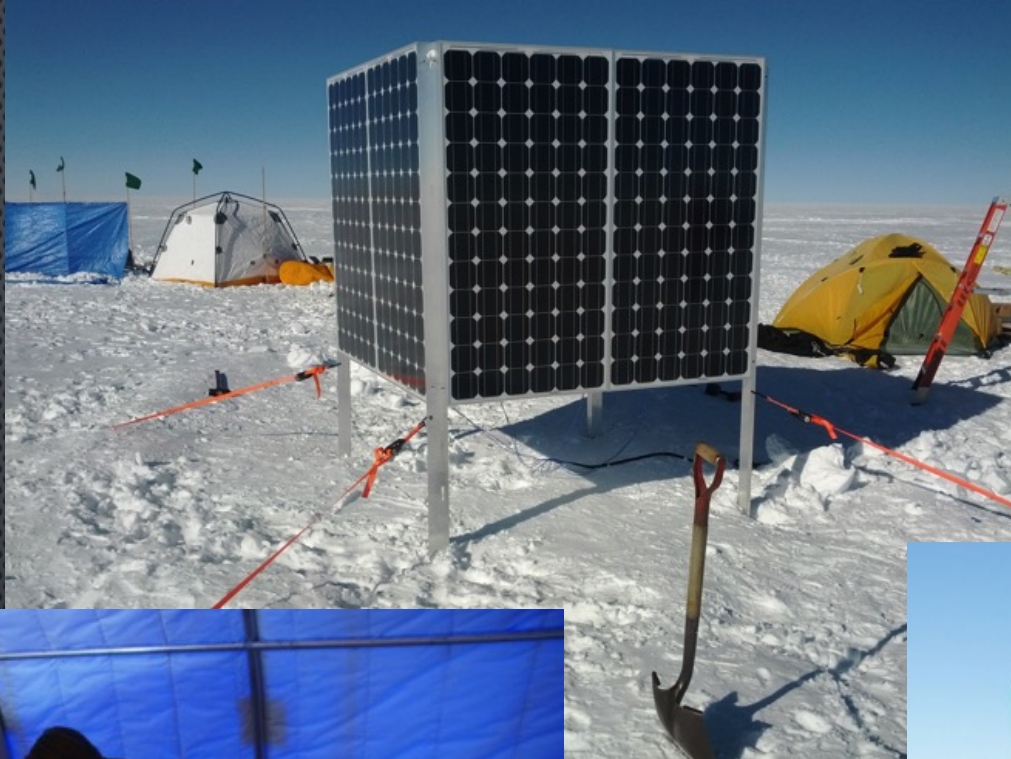
PROTOCOLS : 2. SERIAL (RS232, RS485, RS422)

- Standardized Point-to Point data transfer protocol. Historically used for line printers and modems, but now generically used for industrial automation systems, scientific lab equipment, managed network equipment, embedded computing.
- Easy to add to a standard laptop using USB-to-serial converter. Common interface is the canonical 9-pin D-Sub connector.
- Requires very little supporting software (screen, minicom, 'import serial')

PROTOCOLS : 3. CAN BUS

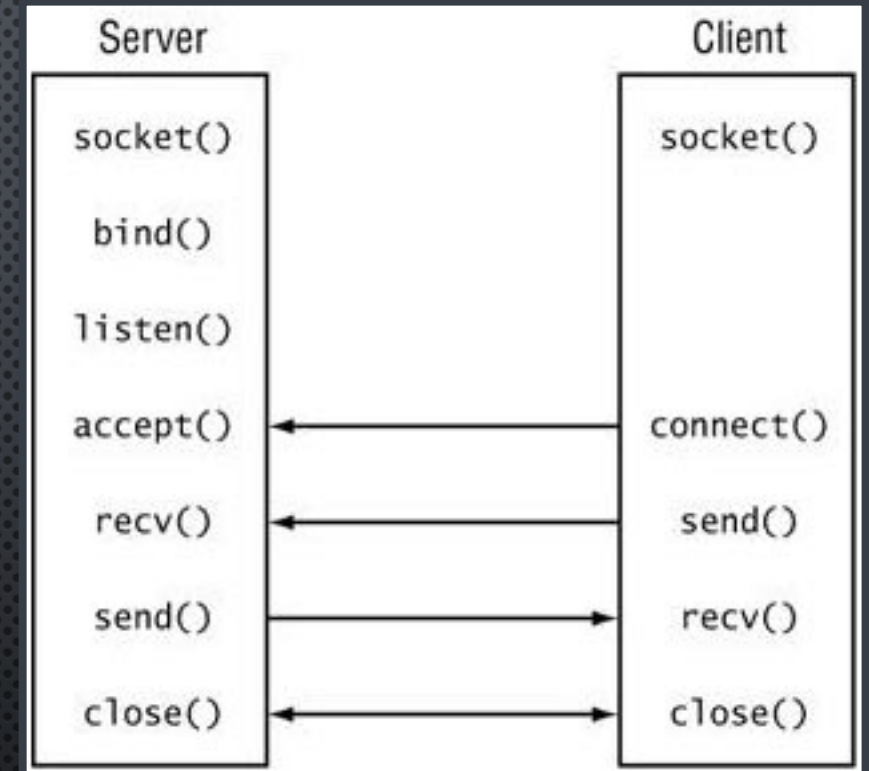
- Controller Area Network
- 2-wire robust vehicle bus standard used to connect many disparate (microcontroller) systems without a host controller or computer
- Your car runs on CAN!
- For observatories, power systems tend to use CAN (solar panel controllers, peak power trackers, generators, wavesculptors, wind turbines, etc...)





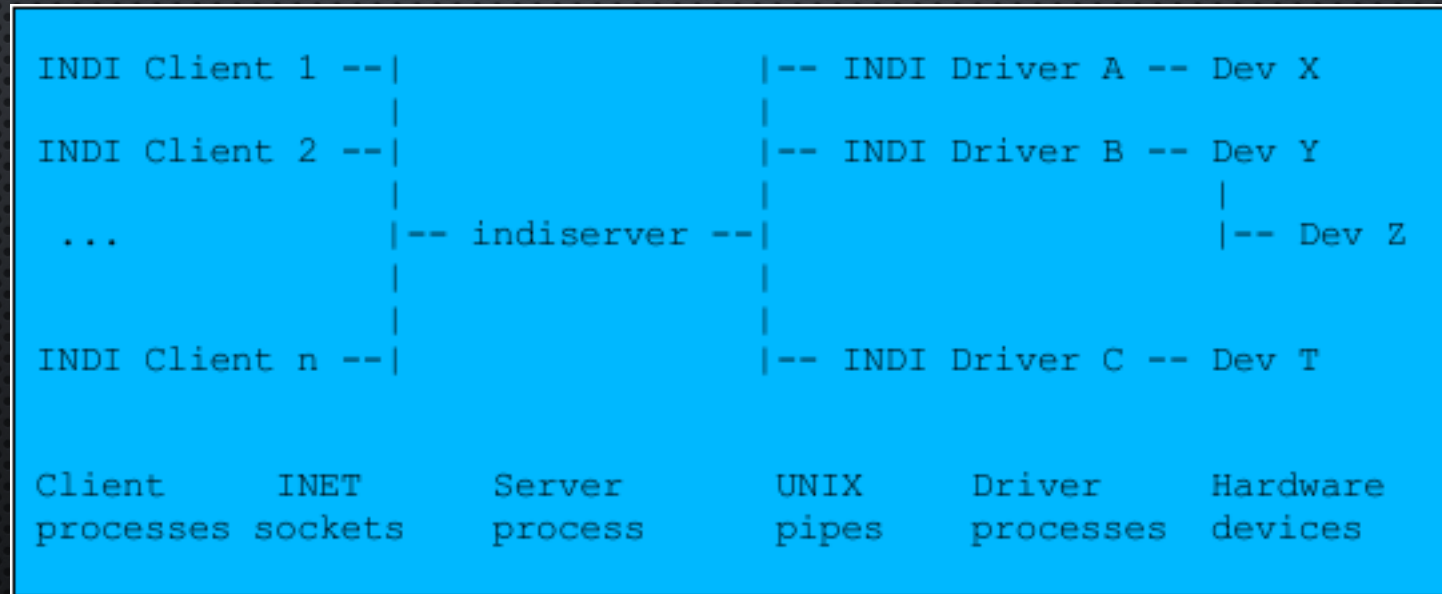
NETWORK SOCKET SERVERS

- Access to your hardware from anywhere (internet of things...)
- The entire Internet is just a pile of socket servers
 - Example of email sending, http server, etc.
- Example Python and C implementations for clients, servers
- Examples of use



OTHER GLUE LOGIC EXAMPLES: INDI


- **Instrument Neutral Distributed Interface**; see indilib.org
- INDI drivers advertise their capabilities by XML; an INDI Server collects them for presentation to clients
- GUIs for INDI servers can be **self-described!** This has to be seen to be believed.



KStars FITS Viewer — KStars

File Edit View Help

SH155_Light_SII_18.fits* Focus



1.569.00 X:475 Y:40 10% 3326x2504

Ekos — KStars

Setup CCD Focus Alignment Guide

CCD

Device: QSI CCD
 Exposure: 600.000 Binning: X: 1 Y: 1
 Frame: X: 0 Y: 0 W: 3326 H: 2504
 Type: Light
 Prefix: SH155 Type Filter Expose Duration
 Count: 16 Delay: 0
 Directory: /home/jasem/Pictures/PixInsight/SH155/SII
 Filters: -- ISO: --

Sequence Queue

	Status	Filter	Type	Bin	Exp
1	In progress	SII	Light	1x1	600

Filter

Device: QSI CCD
 Filter: SII

Auto dark subtract
 Add ISO 8601 time stamp
 Display in FITS Viewer
 Maximum Guiding Deviation 3.00 "
 Autofocus if HFR > 0.535 pixels
 Park When Complete

Progress

Expose: 271.00 seconds left
 Progress: 5 of 16 completed
 31%

2014-12-20T19:16:05 Capturing image...
 2014-12-20T19:16:05 Guiding deviation 2.19 is now lower than limit value of 3 arcsecs, resuming exposure.
 2014-12-20T19:16:01 Guiding deviation 3.14 exceeded limit value of 3 arcsecs, aborting exposure.
 2014-12-20T19:15:59 Capturing image...

Enable logging

2014-12-20T16:15:39: Focuser reached requested position.
 2014-12-20T16:15:37: Focuser is moving to position 24496
 2014-12-20T16:15:29: Focuser reached requested position.