Overview of USAP Next Generation RVDAS

David Pablo Cohn / Valerie Warner
Scott Walker - Manager

Presented at RVTEC 2016 – 01 Nov 2016
Scripps Institution of Oceanography

An **architecture**, not a **system**

(systems change as requirements change; proper architecture lets you put together whatever system meets current requirements)

Read from serial port, prefix with timestamp and instrument id, write to file

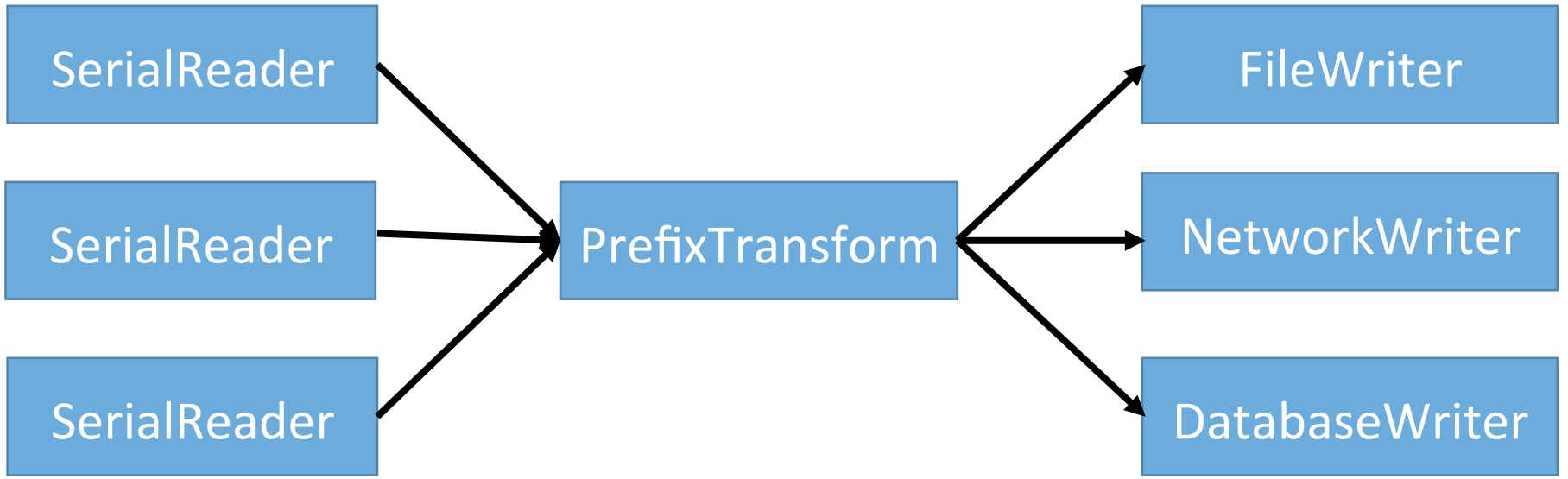
Everyone's needs are different **now**
Everyone's needs will be different in 5 years

Solution: small set of Lego-like components that can be easily "snapped together" to create what you need

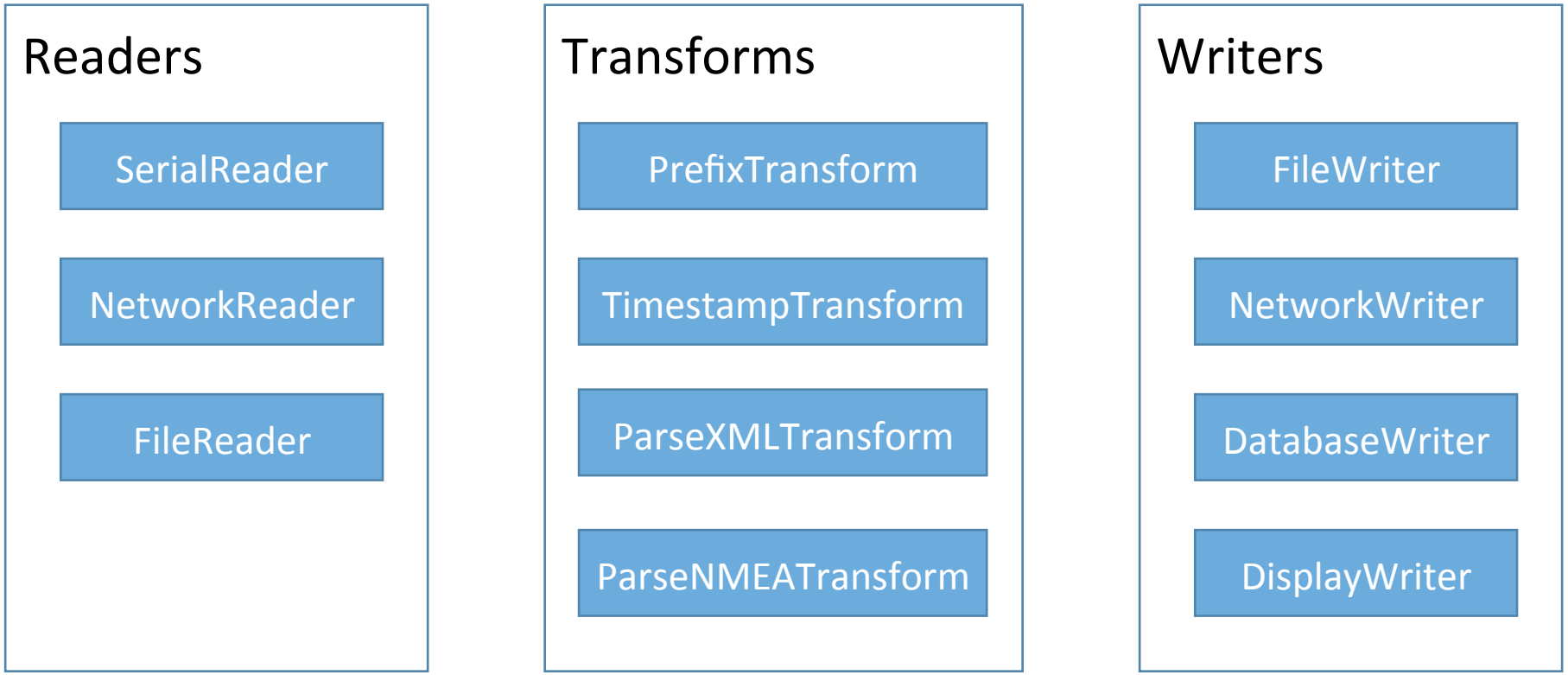
Read from serial port, prefix with timestamp and instrument id, write to file

Everyone's needs are different **now**
Everyone's needs will be different in 5 years

Solution: small set of Lego-like components that can be easily "snapped together" to create what you need



Readers, Transforms and Writers



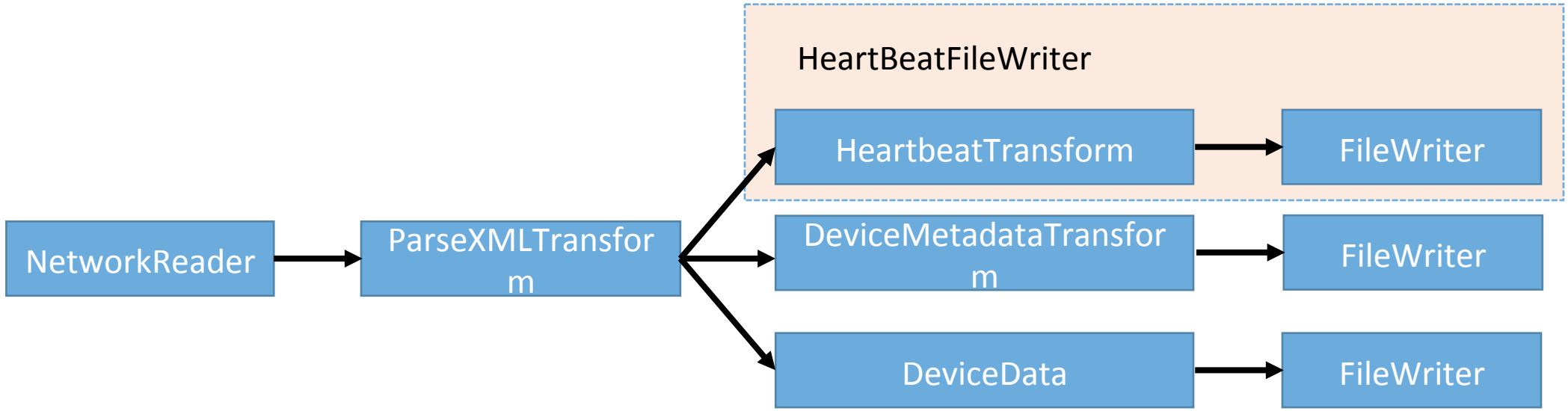
Dead Simple API

So it's easy to create your own Readers/Writers/Transforms as needed

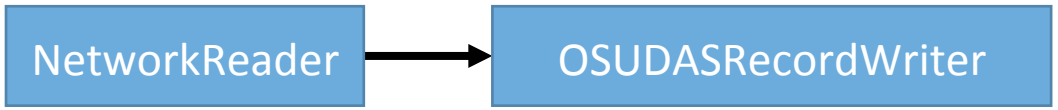
```
reader = SerialReader(instrument='Knudsen')
transform = PrefixTransform(prefix='knud',
                            add_timestamp=True)
writer = FileWriter(logfile='/var/logs/NBP1511/knud')
while True:
  in_record = reader.read_record()
  out_record = transform.transform(in_record)
  writer.write_record(out_record)
```

(We’re using Python, by the way)

Snap pieces together to build what you need



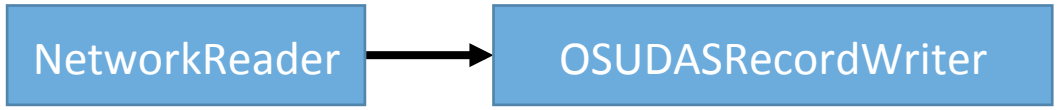
Add new modules as we get new requirements




```
class DASRecord(models.Model):
    record_type = models.CharField(max_length=15)
    version = models.FloatField(blank=True)

    board = models.ForeignKey('Board', blank=True, null=True)
    board_index = models.IntegerField(blank=True, null=True)

    device = models.ForeignKey('Device', blank=True, null=True)
    serial = models.ForeignKey('Serial', blank=True, null=True)
    data = models.ForeignKey('Data', blank=True, null=True)
```

```
class OSUDASRecordWriter:
    def __init__(self):
        pass

    def write_record(self, record):
        tree = xml_utils.parse_xml(record)
        record = DASRecord.from_xml(tree)
        record.save()
```

Add new modules as we get new requirements



**listener.py --read_network :6221 **
--write_osu_dasrecords

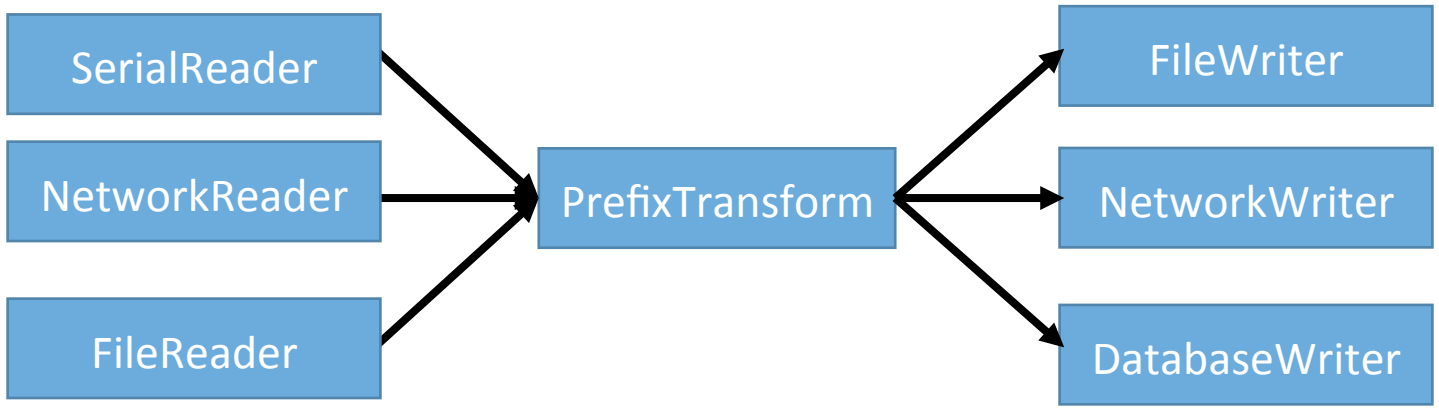
```
<?xml version="1.0" encoding="UTF-8"?>
<OSU_DAS_Record Type="Data" Version="2.0">

  <SUDS_received>
    <Timestamp TimeZone="UTC">
      <Date Format="YYYY-MM-
DD">2016-01-25</Date>
      <Time Format="HH:MM:SS">01:00:00</
Time>
      <Count Units="Seconds"
Epoch="1970-01-01 00:00:00
UTC">1453683600</Count>
    </Timestamp>
.....
```

02-04-A3-42-02-09 Serial	65937	echosounder_well Knudsen Chirp 3260	HalfDuplex 19200 Generic-Listen
02-04-A3-42-02-09 Serial	65938	echosounder_well Knudsen Chirp 3260	HalfDuplex 19200 Generic-Listen
2-04-A3-42-02-09 Serial	65939	echosounder_well Knudsen Chirp 3260	HalfDuplex 19200 Generic-Listen
2-04-A3-42-02-09 Serial	65940	echosounder_well Knudsen Chirp 3260	HalfDuplex 19200 Generic-Listen
2-04-A3-42-02-09 Serial	65942	echosounder_well Knudsen Chirp 3260	HalfDuplex 19200 Generic-Listen
2-04-A3-42- 06 Serial	106468	gyrocompass Sperry	HalfDuplex 4800 Generic-Listen
Serial	106469	gyrocompass Sperry	HalfDuplex 4800 Generic-Listen
2-04-A3-42-02-06 Serial			
2-04-A3-42-02-06 Serial			
2-04-A3-42-02-06 Serial			

ID	STATUS	LOCATION	MAC	IP
1	Good	echosounder_well	02-04-A3-42-02-09	10.240.158.
2	Good	gyrocompass	02-04-A3-42-02-06	10.240.158.
3	Good	gnss_bow_gps	02-04-A3-42-02-48	10.240.158.

Listeners – a snap-together tool

```
listener.py --read_serial Knudsen \
    --data_id knud \
    --logfile /data/logger/uw/NBP1511knud \
    --network :6221 \
    --database rvdas@bonnie:test
```



Chaining Listeners

```
listener.py --read_serial Knudsen \
    --data_id knud \
    --logfile /data/logger/uw/NBP1511knud \
    --network :6221
```

```
listener.py --read_logfile /data/logger/uw/NBP1511knud
    --database rvdas@bonnie:nbp1511
```

```
listener.py --read_network :6221 \
    --display
```

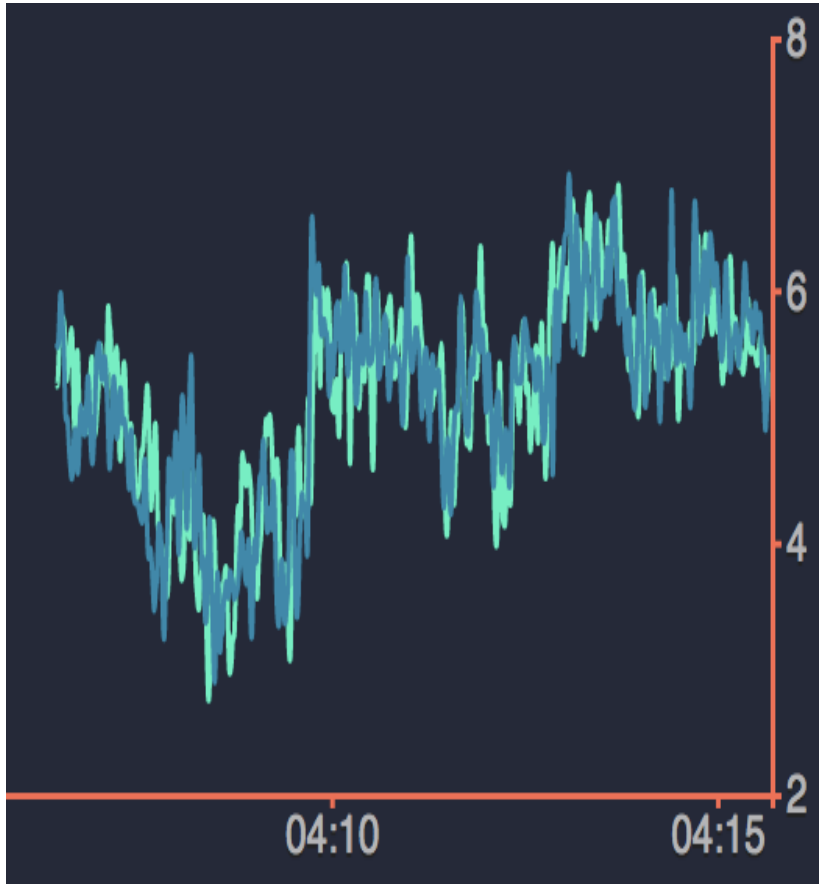
Displays: Browser-based “pull” architecture

DisplayWriter

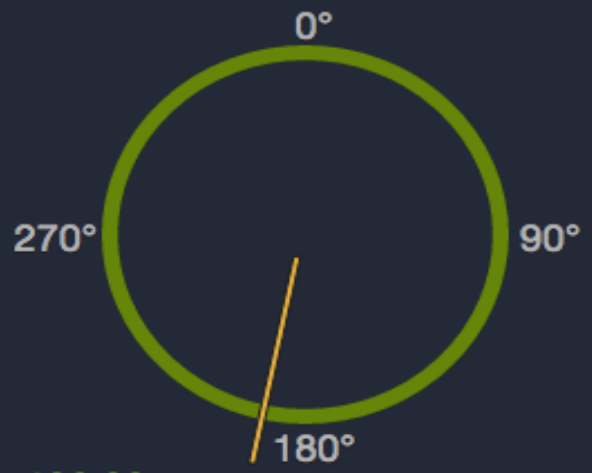
- Receives data records, parses out and caches values
- Serves XMLHttpRequests for updates (moving to websockets for efficiency)

Javascript DisplayWidget

- Renders dynamic Table/Dial/Chart
- Updates via websocket / XMLHttpRequests to DisplayWriter

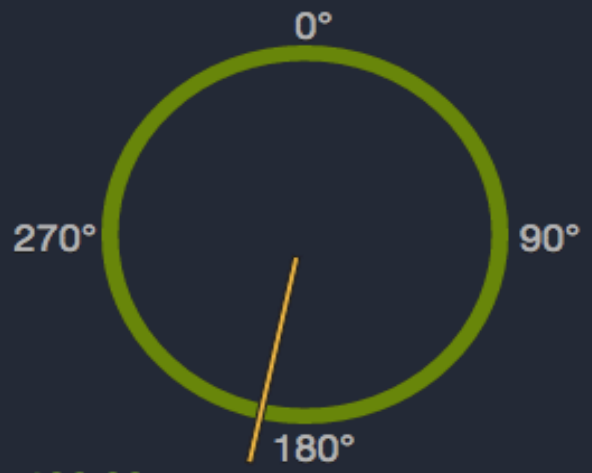


Port Relative Wind



192.00

Stbd Relative Wind



192.00

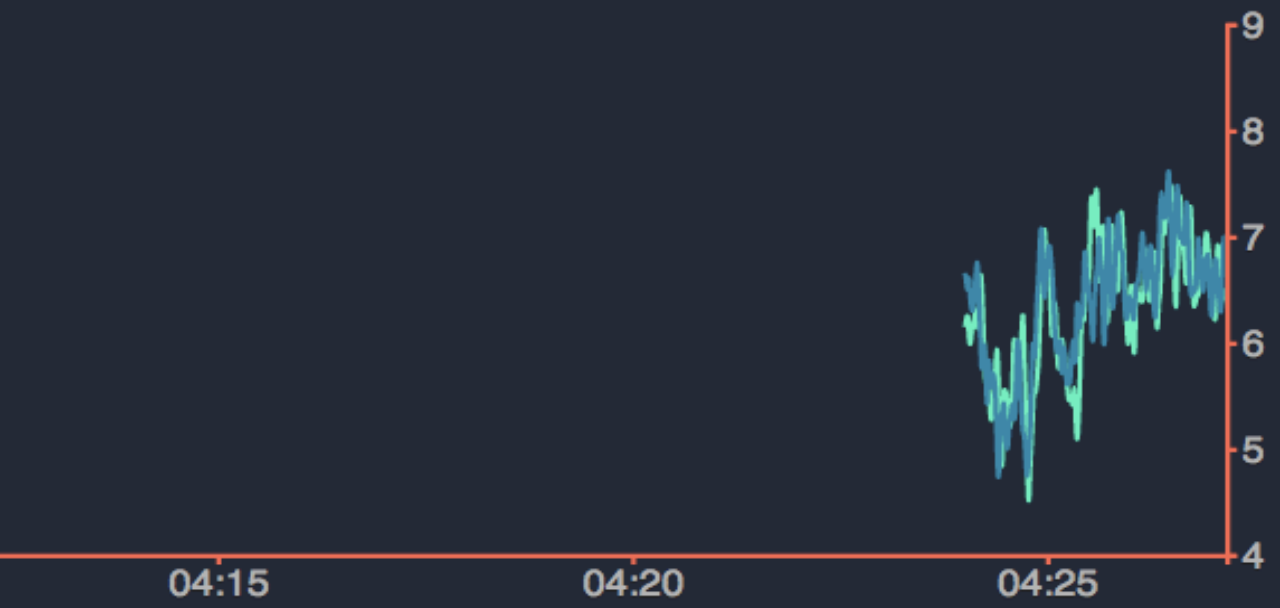
Relative Wind Direction

Port **190.00** deg

Starboard **192.00** deg

Port (Max) **6.56** m/s

Starboard (Max) **6.84** m/s



A bunch of data

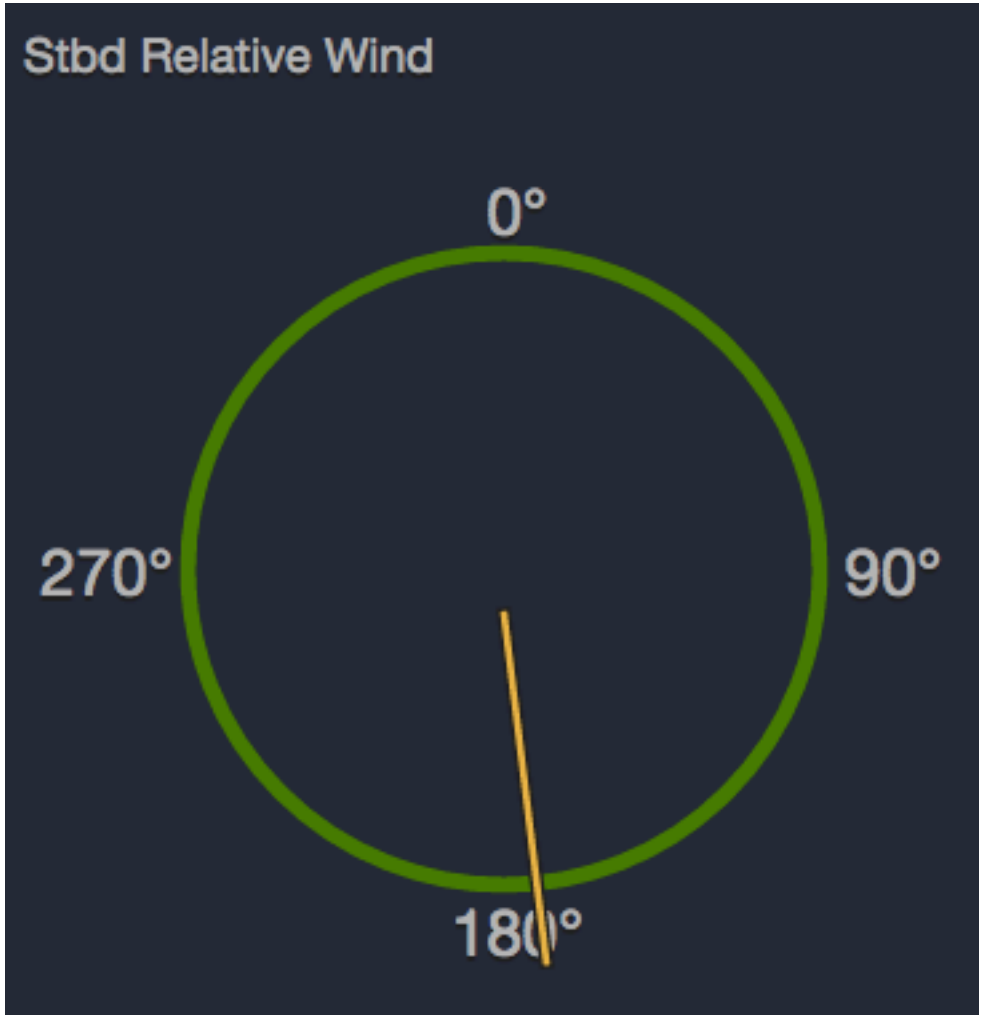
Remote Temp **6.61** °C
TSG1 Temperature **6.73** °C
TSG1 Conductivity **3.43**
TSG1 Salinity **34.14**
TSG2 Temperature **6.73** °C
TSG2 Conductivity **3.43**
TSG2 Salinity **34.17**
Gravity **26111.00**
Air Temperature **6.19**
Relative Humidity **79.60**
Barometer **1007**
PAR **74.11**
PSP **0.15**
Port Windspeed **6.56**
Port Wind Direction **190**
Stbd Windspeed **6.84**
Stbd Wind Direction **192**
HDAS Temp **12.81**
Fluorometry **222.07**
Transmissivity **4234.48**
Flow Meter 1 **16.00**
Flow Meter 2 **49.50**
Flow Meter 3 **35.00**
Flow Meter 4 **19.50**
Aquarium Room Flow **508.00**
Helo-deck Flow **565.50**

Slightly less data

GUV Ed0Gnd **0.0002**
GUV Ed0305 **0.0265**
GUV Ed0313 **0.2997**
GUV Ed0320 **0.6887**
GUV Ed0340 **1.9120**
GUV Ed0380 **2.5460**
GUV Ed0395 **2.4970**
GUV Ed0PAR **0.0058**
Air Temperature **6.2**
Latitude **5406.877696**
Longitude **8923.394238**
SOG **9.9**
Heading **0.0**
Heave **-0.1**
Pitch **-0.1**
Roll **-0.3**
pCO2 Pressure **1006.02**
pCO2 Flow Rate **49.36**
Multibeam Depth **0.00**

DisplayWidgets are configure-drag-drop

```
{
      'name': 'dial_wind',
      'widget': 'DASDialWidget',
      'angle': 'NBPSUSWindDir',
      'options': {
         'magnitude': 'NBPSUSWindSpd',
         'title': 'Stbd Relative Wind',
         'height': 300,
         'width': 300,
         'circle_color_range': ['green','orange','firebrick','purple'],
         'circle_color_domain': [2,15,30,45],
      }
   },
```



Control Architecture – a Database

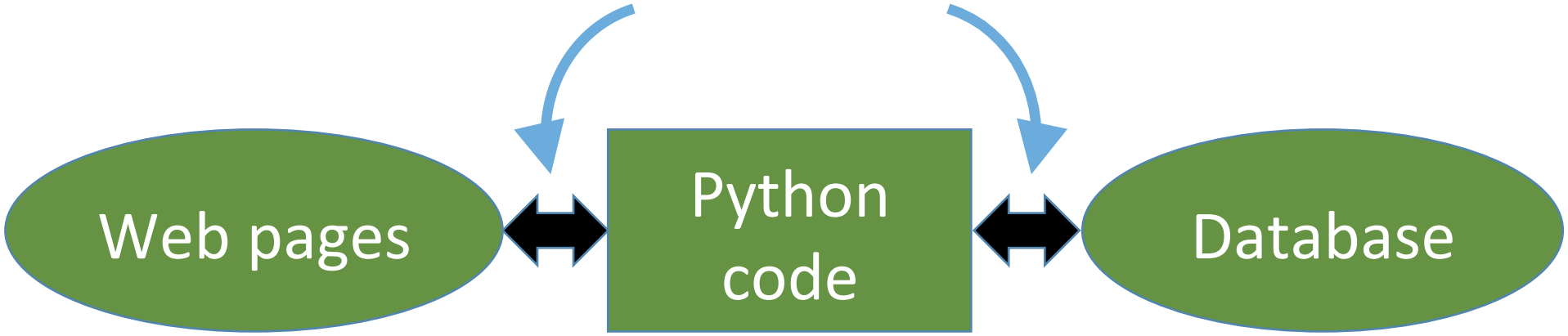
Action: --------- Go 0 of 21 selected

Id	Current state	Desired state	Sensor connection
PCOD	observed: PCOD running	desired: PCOD running	PCOD (serial)
adcp	observed: adcp not running	desired: adcp not running	adcp (serial /dev/ttyr05, 9600,
ctdd	observed: ctdd not running	desired: ctdd not running	ctdd (serial)
eng1	observed: eng1 running	desired: eng1 running	eng1 (serial)
flr1	observed: flr1 not running	desired: flr1 not running	flr1 (serial /dev/ttyr17, 19200,
gp02	observed: gp02 not running	desired: gp02 not running	gp02 (serial /dev/ttyr03, 4800,
grv1	observed: grv1 not running	desired: grv1 not running	grv1 (serial /dev/ttyr14, 9600,
gyr1	observed: gyr1 not running	desired: gyr1 not running	gyr1 (serial /dev/ttyr04, 4800,
hdas	observed: hdas not running	desired: hdas not running	hdas (serial /dev/ttyr27, 9600,

Control Architecture – a Database Approach

Django is "*a database-driven web development framework*"*

*translation: goop that lets us connect code to web pages and a database



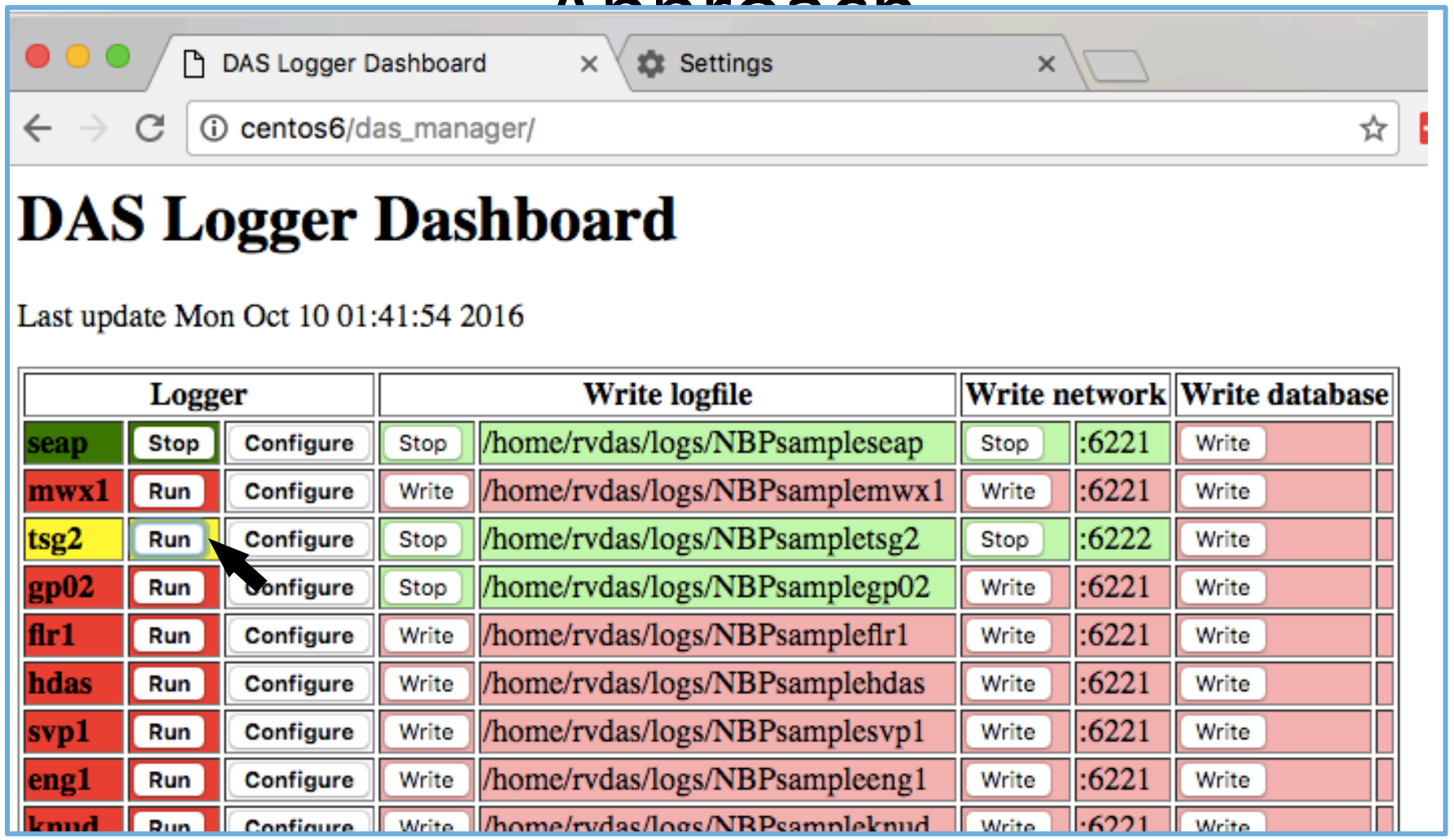
Control Architecture – a Database Approach

```
manage_loggers.py -loggers pguv,rtmp,s330,knud \
      --manage_interval 1
```

LoggerManager

- Reads desired state from database
- Checks observed state from system
- Starts/stops/modifies processes to reconcile

Control Architecture – a Database Approach

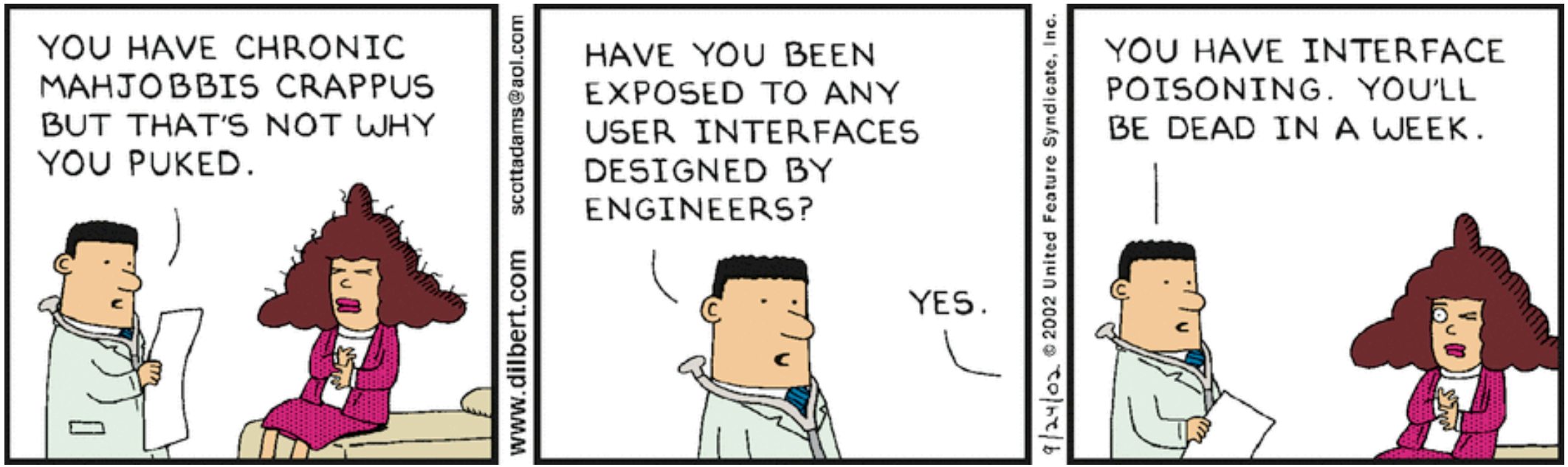


Control Architecture – a Database Approach

```
monitor_loggers.py \
    --sleep_seconds 5 \
    --stale_interval 60 \
    --email rvdas_alerts@nbp.usap.gov
```

LoggerMonitor - watches database, LoggerManager, filesystem and processes; raises alarm if things look stale

This is all early days – what's next?

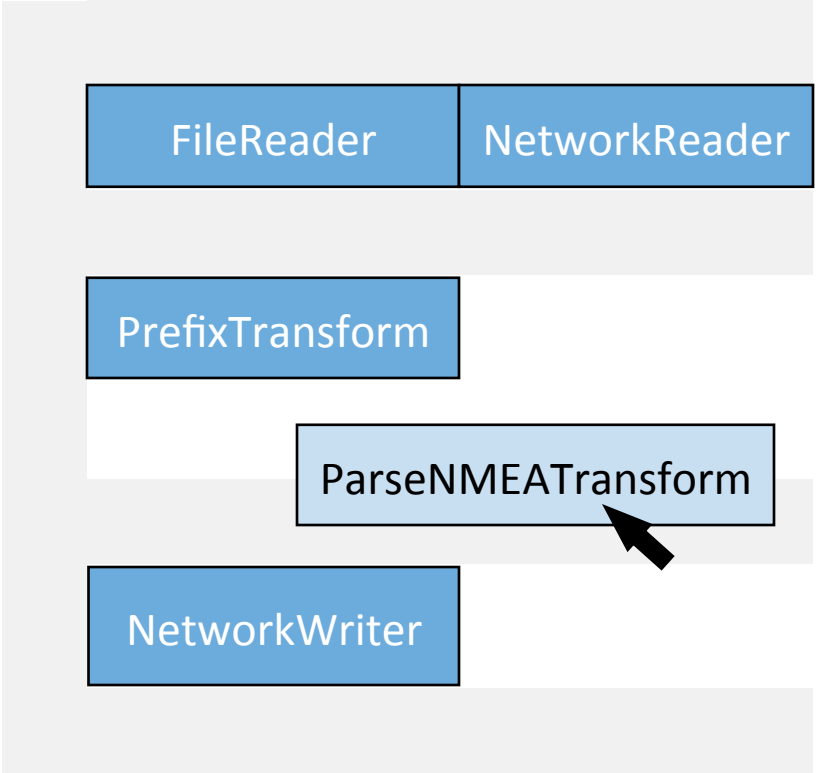


Better Control UX

This is all early days – what's next?

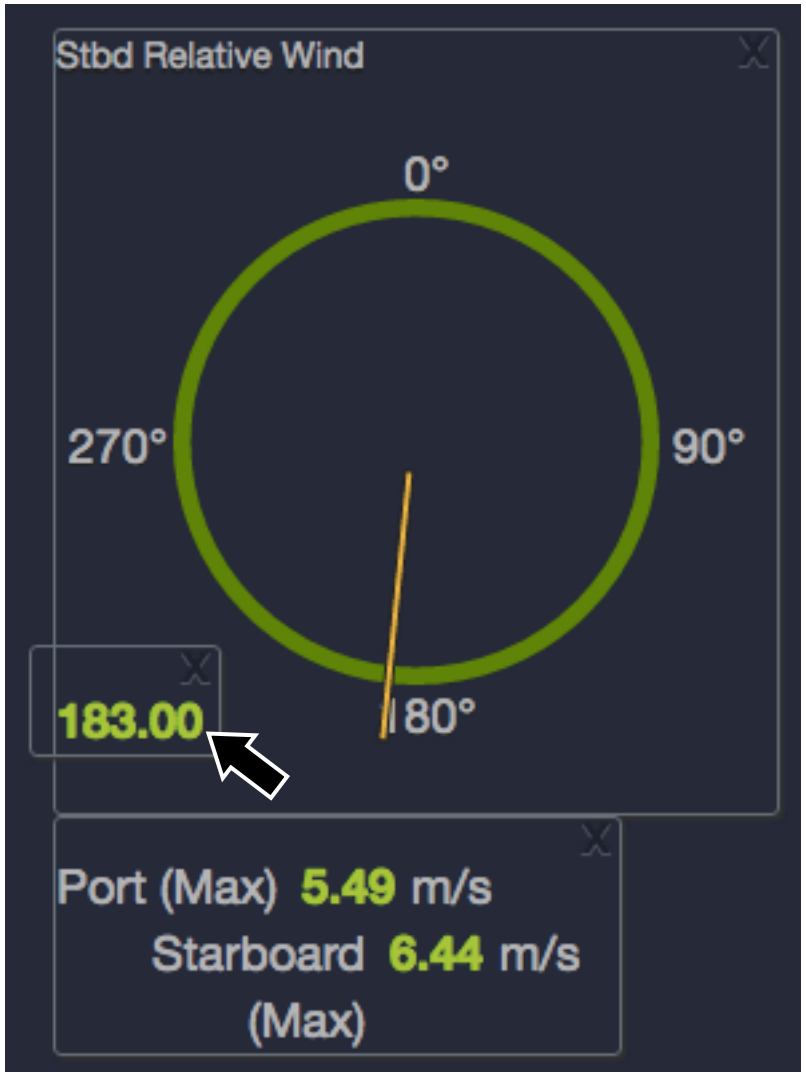
Better ways to compose Readers/Transforms/Writers

Maybe Scratch-like visual interface?



This is all early days – what's next?

Drag-and-drop creation of entire displays



This is all early days – what's next?

Beta deployment at sea

This is all early days – what's next?

Getting other teams and vessels involved

(come find us to talk)