Zaxy

"gliderDos" for Slocum Backseat Control

Modular microcontroller based architecture for rapid sensor integration and flight autonomy

Justin Adam Scott Shapiro, jshapi@uw.edu

IUGC, Gotenburg, Sweden

20240614

Team Epsilon Fleet

- Gliders
  - APL/UW SeaGlider systems
    - 1 km SG
    - 6 km DG systems
  - TWR Slocum Gliders
    - 1xG1, 3xG2, 4xG3
    - 1km,350m,200m,50m,30m
    - Core platforms
      - uStructure sensing
      - Enhanced autonomy and sensor integration
- AUVs
  - IVER-900s
  - REMUS 1000
    - OTH C2 of long duration AUV
- My background
  - Previously at WHOI, TWR, Mote Marine, Georgia Tech, Rutgers University

Team Epsilon:
Lou St. Laurent
Harper Simmons
Christian Sarrason
Patricia Kassis
Shawn Albertson
Justin Shapiro

Overview

- 20000ft (6000m) view: Backseat Driver?
- Review Slocum as a platform for autonomy at sea
- Introduce Zaxy a scalable, modular, RTOS based ecosystem for managing sensor and autonomy integration
    - Probably in too much detail
- Current integrations and programs at sea

Why backseat drive

- OEMS have worked hard to make vehicles robust
  - Extensive, well tested codebases developed over time scales O(decade)
  - Robust, well monitored hardware
  - Managed power consumption
  - Data telemetry pathways
- Provide access for customers to add capabilities to the mature platform
  - Make changes to vehicle controls and mission underwater
  - Integrate new sensor technologies
  - Add novel capability while relying on the stability that we count on

Why backseat driving/sensing

- Critical growth sector for the community
- Glider applications roughly broken in half
  - Use of well proven technologies
    - Not to be underestimated, this is still a development task
    - Still under continuous development/improvement
    - Development of best practices
    - Understanding long term stability of sensor technologies
  - Integration of new capabilities
    - Historically has been in the hands of platform originators

Other backseats we've heard from at IUGC24

- ALSEAMAR – nvidia jetson/ROS
- Rich Camilli – Raspberry Pi/ROS
- Allsup, Wang – BeagleBone black
- All idle at over a 1W, strays from the original Stommel vision
  - ½kt ½W
  - Critical for endurance, survivability in the event of graceful degradation
  - More challenging, stepping from world of SBCs to that of uC
  - Use of ROS eases pain of entry, but requires hefty linux build and TCP/IP stack
    - Gliders have long avoided internal ethernet as it requires O(.5W) for each PHY
- Here I'll introduce a microcontroller based approach
  - Provide an abstraction layer in 10s of mW

What is backseat driving

All modern gliders are split into a multiprocessor architecture

Flight	Science
• Managing flight Dynamics • Navigation • Surfacing • Vehicle safety	• Manages science sensors • Updates science parameters for storage in archive

Sensor drivers in bold

What is backseat driving

"Backseat driver" provides at minimum an API to integrate an additional processor into the vehicle

Flight	Science	Backseat Controller
• Original Equipment Manufactuer (OEM) codebase • Managing flight Dynamics • Navigation • Surfacing • Vehicle safety	• OEM Codebase • Manages science sensors • Updates science parameters for storage in archive	• Customer Codebase • Access science/flight data • Store data for telemetry • Update flight controls according to algorithms coded by the USER on the Backseat

Backseat Control Implementation on the Slocum

Flight

- Managing flight Dynamics
- Navigation
- Surfacing
- Vehicle safety

Science

- CTD Proglet
- EXTCTL proglet

EXTCTL Proglet

- Establishes a Serial Message Passing Interface Between Science and an External Controller (EC)
- Flight/SCI Paramaters -> EC
- EC -> sci_generic_a … z
- EC-> Mission Parameters
- Manage file transfers
    - EC->Science
    - Science->EC
- Log messages to glider console log (MLG)

Focus of this talk

Flight	Science	External Controller
• Managing flight Dynamics • Navigation • Surfacing • Vehicle safety	• CTD Proglet • EXTCTL proglet	• gHandler • **Sensor Drivers** • **…** • **Autonomy Drivers**

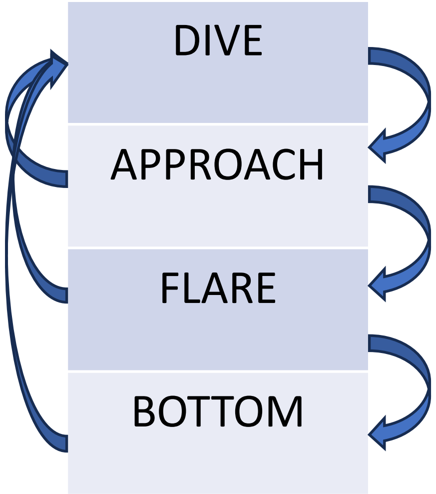
A motivating example: Consider you want to gently land a slocum glider on the seafloor and hold it in place

- Can try
  - Drift at depth
    - Aims for neutral buoyancy, swept away by current
  - Multiple yo behaviors
    - Each has to quit to hand off control, need to restart mission each segment
  - Fly with lots of drive, shallow angle
    - Hard to control where you're landing
  - Can fly with minimal drive
    - Drift away

If only there were some behavior …

It would just be a modification of a yo!

```
u_land_dive_cc -400
u_land_dive_pitch -0.5
u_land_approach_altitude 30
u_land_approach_cc -350
u_land_approach_pitch -0.3
u_land_flare_altitude 10
u_land_flare_cc -300
u_land_flare_pitch -0.216
u_land_bottom_altitude 3
u_land_bottom_depth_rate 0.04
u_land_bottom_cc -420
u_land_bottom_pitch -0.6
u_land_activation_depth 5
```



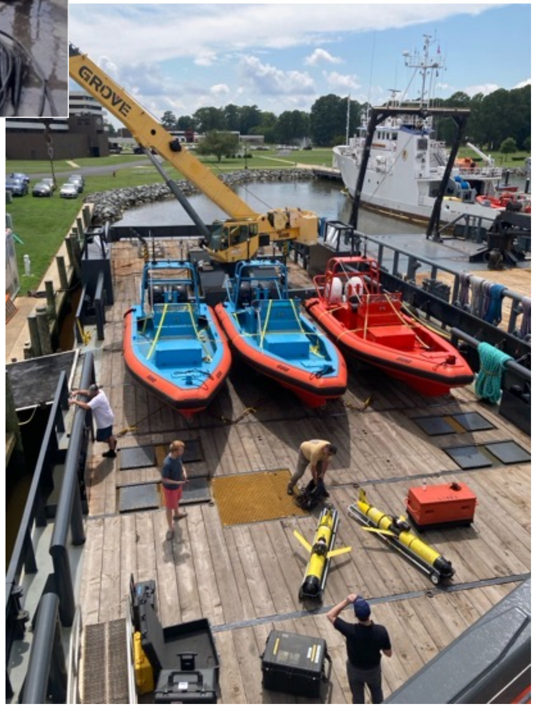

```
mp
u_mission_param_a
u_mission_param_b
u_mission_param_c
```

```
behavior_name=yo

<start:b_arg>
    b_arg: start_when(enum)       4   # updwn_idle
    b_arg: num_half_cycles_to_do(nodim) 2    # Number of dive/climbs to perform
                                 # <0 is infinite, i.e. never finishes

    # arguments for dive_to
    b_arg: d_target_depth(m)       195
    b_arg: d_target_altitude(m)    0

    b_arg: d_use_pitch(enum)       3     # servo on pitch
    b_arg: d_pitch_value(X)     1000001 #u_mission_param_b

    b_arg: d_use_thruster(enum)     0  # 1-%, 4-watts
    b_arg: d_thruster_value(X)     0

    b_arg: d_max_thermal_charge_time(s)  30.0
    b_arg: d_stop_when_stalled_for(sec) 42300 # 12 hours
    b_arg: d_stop_when_hover_for(sec)    42300 #! simple=False
    b_arg: d_bpump_value(X)     1000000 #u_mission_param_a
    # arguments for climb_to
    b_arg: c_target_depth(m)       3
    b_arg: c_target_altitude(m)   -1

    b_arg: c_use_pitch(enum)       3
    b_arg: c_pitch_value(X)      0.35 #climb steep
    b_arg: c_stop_when_stalled_for(sec) 30.0

    b_arg: c_use_thruster(enum)     0  # 1-%, 4-watts
    b_arg: c_thruster_value(X)   1000002
    b_arg: c_bpump_value(X)     300.0
    b_arg: end_action(enum) 2     # 0-quit, 2 resume
<end:b_arg>
```

Call it something like beh_land

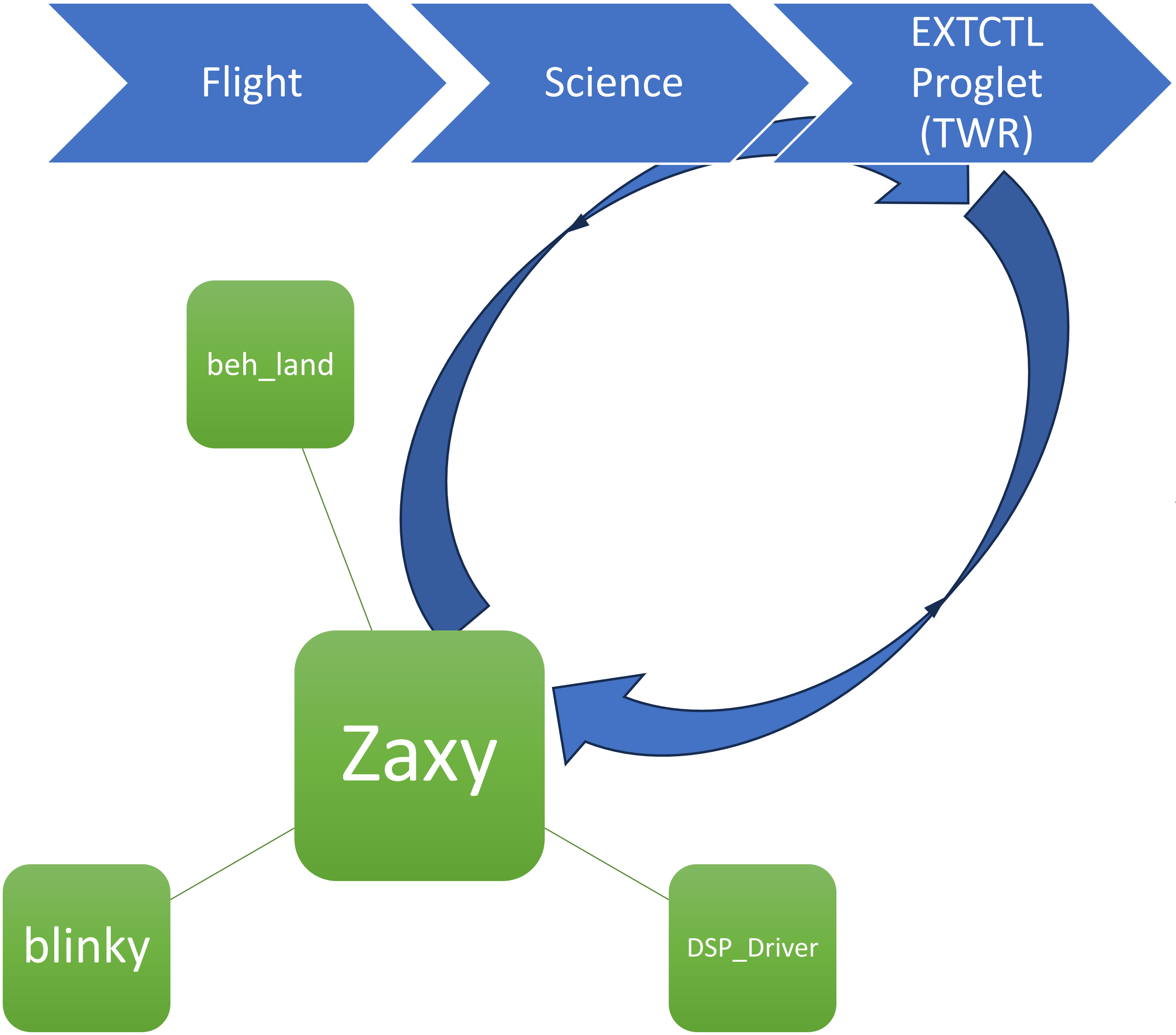
At the same time…

Call this oa_dsp_driver



Requirements
- 1+ Month endurance at 100% duty cycle sampling (unless at surface)
- ~4 months to first test
- ~5.5 months to have two platforms tested, shipped abroad for high priority demo

- Listen for fast boats
- 30+ channel hydrophone array
  - 0.86gb/hr
  - ~2.5W DAQ
- DSP
  - Fast boat classifier/tracker
    - Telemeter contact tacks
  - Monitor array(floating point)
    - Geometry
    - Acoustic statistics
  - ~1.5W DSP

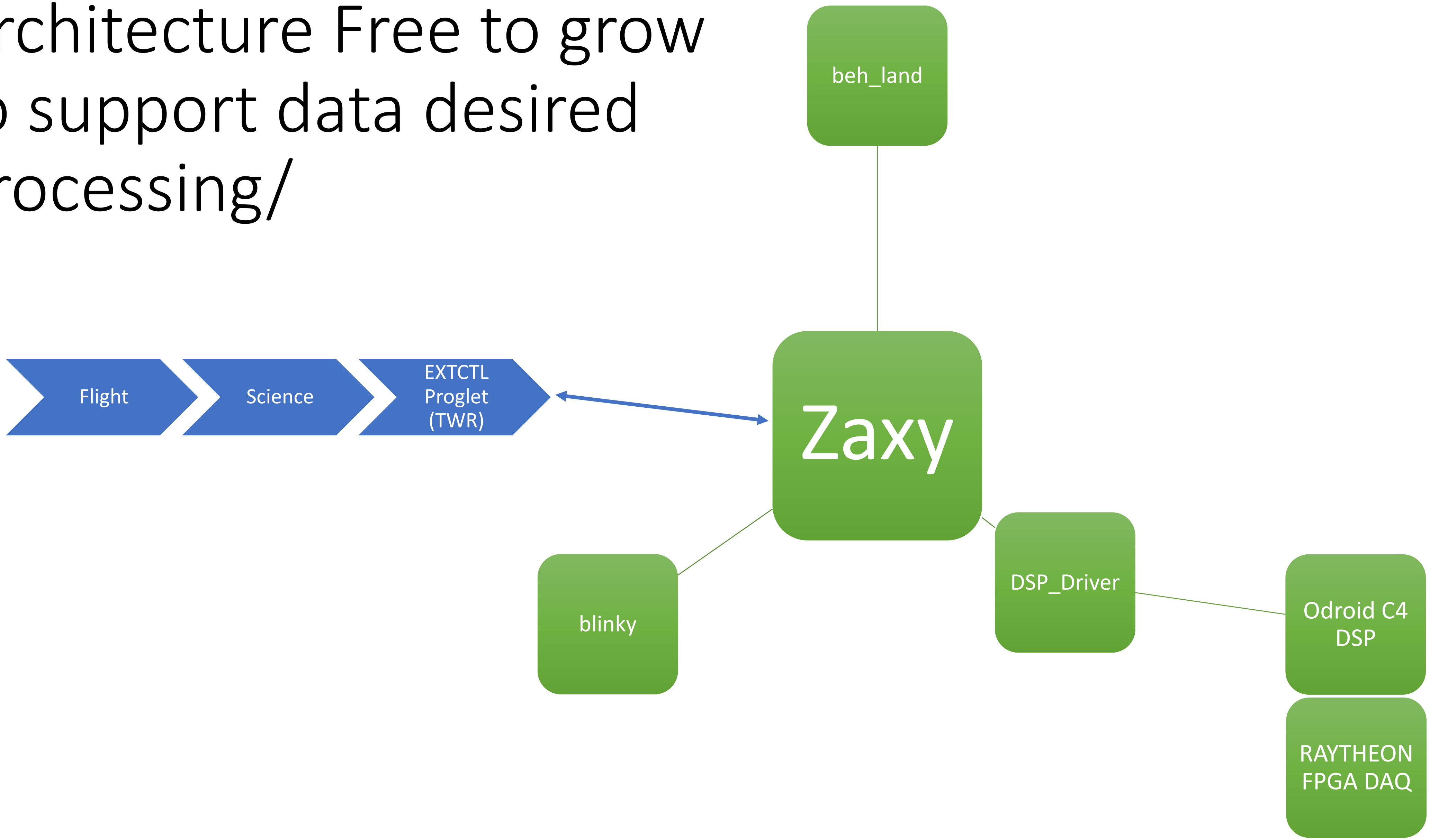


Zaxy
- “the gliderDos/sciDos” for the backseat
- Built on Nordic Semiconductor Zephyr
  - Real time, pre-emptive, multi-threaded
  - Can be compiled for most uC SOC’s
  - Currently using STM32L4/F4

As services
- Manages I/O with glider
- Manage file transfer with glider
- Maintain accurate clock/time sync with glider
- Provide file system for storage

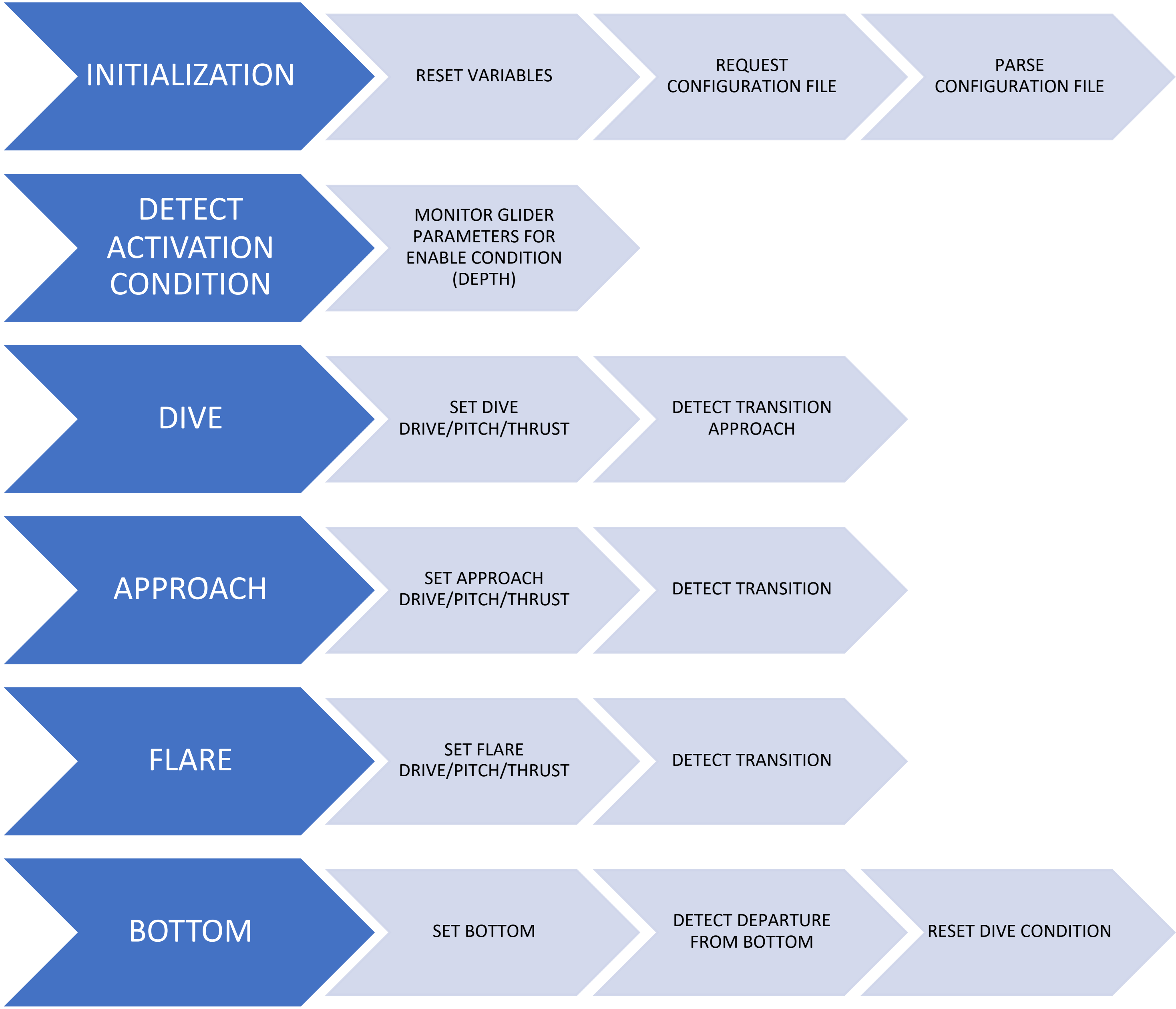
Zaxy abstracts away the EXTCTL protocol and hosts “drivers”
- Science
  - Newly integrated sensors
  - Derived data products
- Engineering
  - Sampling behaviors
  - Flight behaviors

Architecture Free to grow to support data desired processing/



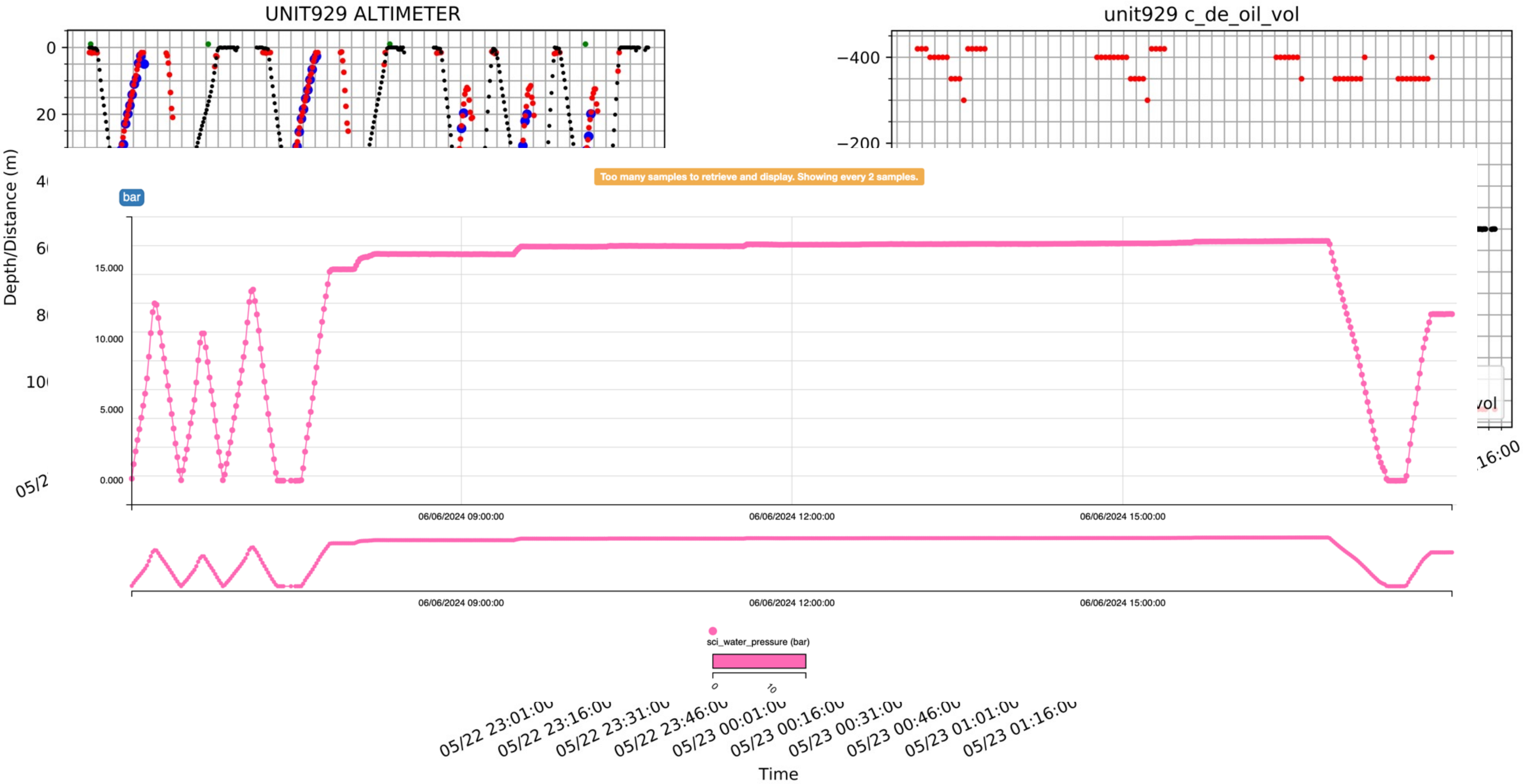
What is a “driver” typically responsible for

- Configuration
- Then its up to the application
- For the beh_land example

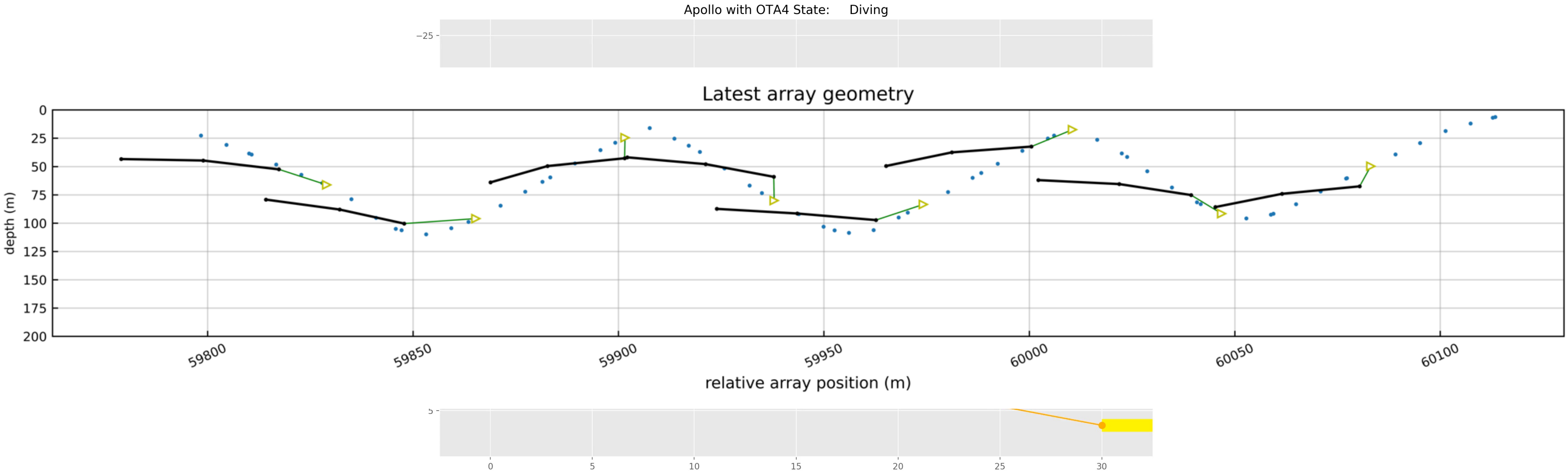


Data Offload

- In EXTCTL Two means for transport of data back for telemetry via satellite comms
  - Classic
    - 24 half float parameters can be published to extctl proglet
    - Packed into dbd/ebd/sbd/tbd
  - Custom
    - Periodically offload data file from external controller to science payload
    - Data must be converted to base64 in transit, so inherently binary or compressed data welcome
    - Limits:
      - Glider to backseat bandwidth
      - Telemetry plan
    - Zaxy manages files transfer protocol asynchronously

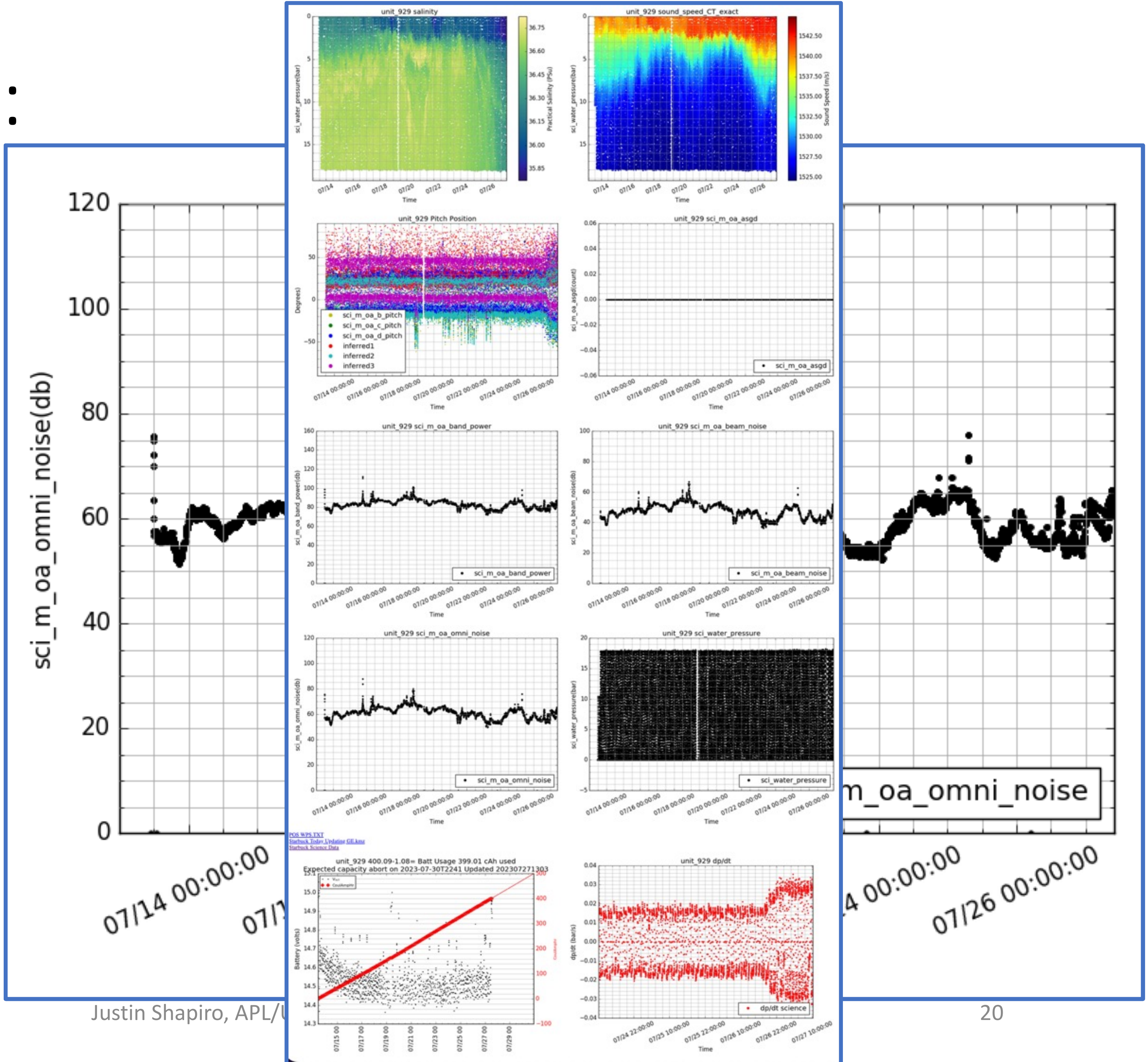

UNIT929 ALTIMETER
unit929 c_de_oil_vol
Depth/Distance (m)
Too many samples to retrieve and display. Showing every 2 samples.
bar
06/06/2024 09:00:00
06/06/2024 12:00:00
06/06/2024 15:00:00
sci_water_pressure (bar)
05/22 23:01:00
05/22 23:16:00
05/22 23:31:00
05/22 23:46:00
05/23 00:01:00
05/23 00:16:00
05/23 00:31:00
05/23 00:46:00
05/23 01:01:00
05/23 01:16:00
Time

Array geometry data

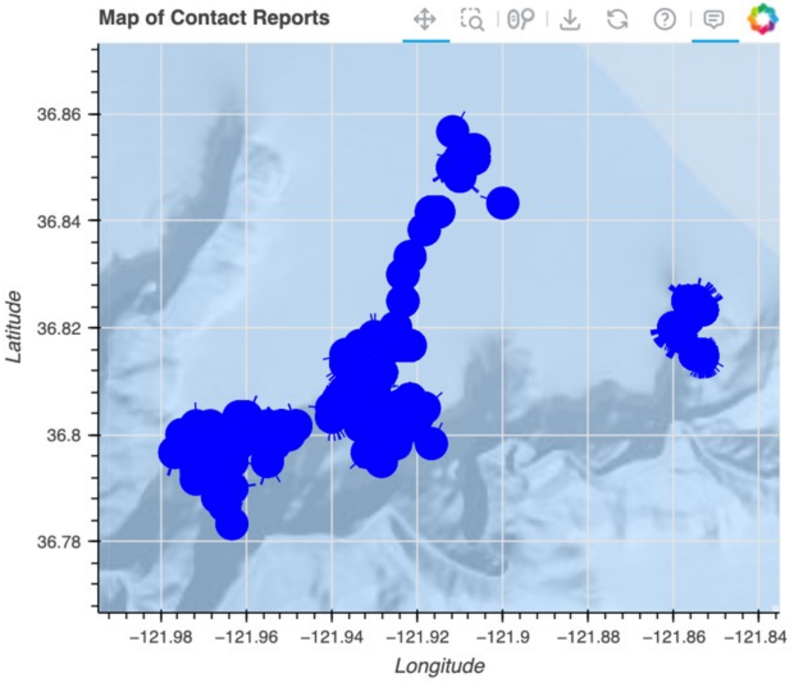


Array data continued:

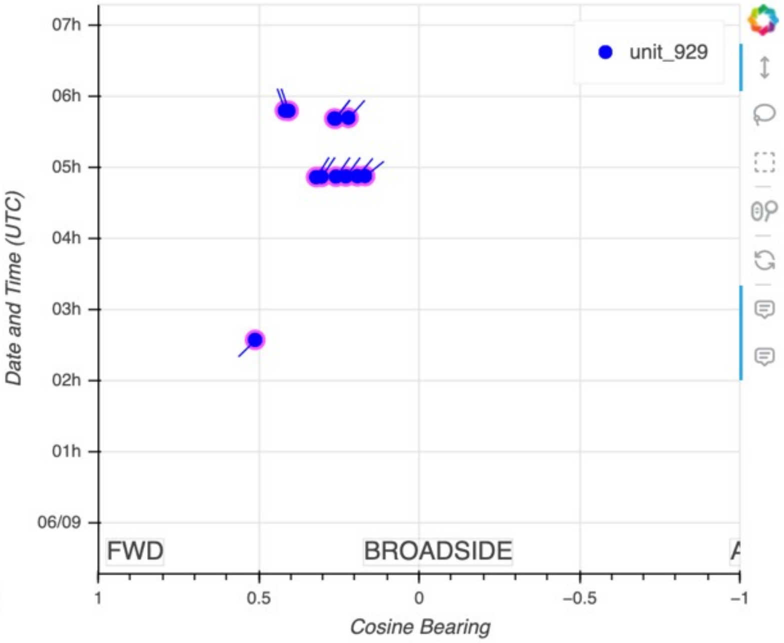
- Numeric data packed into glider data files
- Standard glider processing tools can visualize to introspect
    - Enable non-glider user partners to introspect the data



Dashboard Created At:
2024-06-09 00:17:32 US/Pacific
2024-06-09 07:17:32 UTC



Waterfall plot of Cosine Relative Bearing
Dots colored by glider



#	Time (UTC)	Glider	Latitude	Longitude	Depth (m)	Heading (de	Target Relat	True Bearin	True Bearin	Classifier Ty	Target Type	Score	Frequencies	Received Le
0	05/20/2024	unit_929	36.8150	-121.8550	11.10	97.3	54.2	0.0	0.0	2	10	3	188.216,35	78.7,86.8
1	05/20/2024	unit_929	36.8150	-121.8550	11.10	95.8	54.0	0.0	0.0	2	10	3	93.36,188.1	73.7,81.4,8
2	05/20/2024	unit_929	36.8150	-121.8550	20.10	94.1	50.8	0.0	0.0	2	10	3	93.576,349.	75.1,82.6
3	05/20/2024	unit_929	36.8150	-121.8550	20.10	93.5	53.9	0.0	0.0	2	10	3	93.319,187.	79.7,81.3,8
4	05/20/2024	unit_929	36.8150	-121.8550	20.10	93.7	75.0	0.0	0.0	2	10	3	92.439,188.	78.5,79.5,8
5	05/20/2024	unit_929	36.8150	-121.8550	20.10	92.9	56.8	0.0	0.0	2	10	3	93.549,349.	79,86.9
6	05/20/2024	unit_929	36.8150	-121.8550	20.10	91.2	75.1	0.0	0.0	2	10	3	93.591,188.	79.6,77.4,8
7	05/20/2024	unit_929	36.8150	-121.8550	28.50	90.8	77.1	0.0	0.0	2	10	3	93.66,188.2	80.5,81.2,8
8	05/20/2024	unit_929	36.8150	-121.8550	28.50	88.3	78.6	0.0	0.0	2	10	3	188.379,34	81.8,92.1
9	05/20/2024	unit_929	36.8150	-121.8550	28.50	87.5	54.3	0.0	0.0	2	10	3	93.154,188.	76.3,81.3,8
10	05/20/2024	unit_929	36.8150	-121.8550	28.50	86.3	56.7	0.0	0.0	2	10	3	92.923,188.	80,83.6,89.
11	05/20/2024	unit_929	36.8150	-121.8550	28.50	86.0	57.6	0.0	0.0	2	10	3	93.744,188.	80.5,84.5,8
12	05/20/2024	unit_929	36.8150	-121.8550	36.80	88.5	59.5	0.0	0.0	2	10	3	93.917,188.	82.4,89.7,9
13	05/20/2024	unit_929	36.8150	-121.8550	36.80	90.2	59.9	0.0	0.0	2	10	3	93.196,188.	82.7,89.6,9
14	05/20/2024	unit_929	36.8150	-121.8550	36.80	91.4	61.0	0.0	0.0	2	10	3	91.691,188.	85.2,87.6,9

Services are provided to enable clean, code relieved from the details of the glider interaction

For example,

Send data file from Zaxy's uSD card to the glider

```
ext_ft_parse_args_send(NULL, localFullPath,gliderPath, output_offset, blockLengthRemaining);
```

Request a file download from the glider

```
ext_ft_parse_args_request(&responseQ, localFullPath,gliderPath);
```

Send a character string to be stored on the glider console/MLG file

```
char msg[256] = {'\0'};
snprintf(msg,256,"[%s] set glide parameter for [%s] depth [%0.2f]",__func__,mode,b->bld.input[BL_GV_M_DEPTH]);
return ext_text_glider(msg);
```

Send updates buoyancy pump, pitch angle and thrust power, stage for transmission

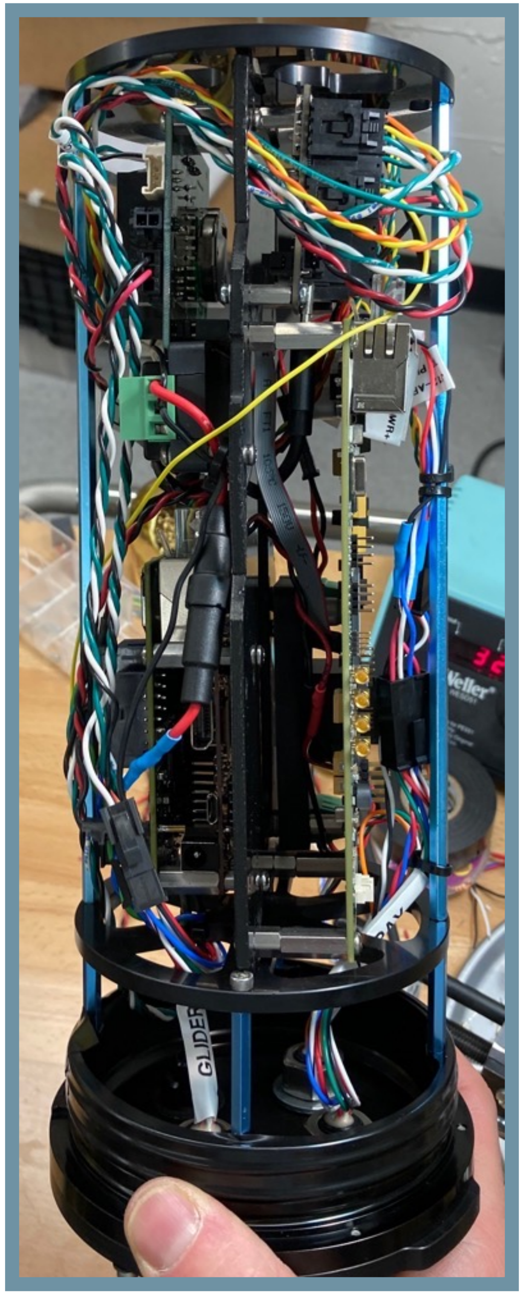
```
b->bld.mp[D_BPUMP_VALUE_A]=cc;
b->bld.mp[D_PITCH_VALUE_B]=pitch;
b->bld.mp[THRUST_VALUE_C]=thrust;

k_mutex_unlock(&land_mutex);
sync_land_to_gld_obj(obj);
```

This is not as simple as ROS

- Power is in milliwatts
- Zephyr ecosystem provides extensive libraries
    - Hardware support
        - Communications
        - Accelerometers
        - Modems
        - IMU
        - Environ
    - zscilib – uC optimized
        - Numerical analysis
        - Matrix Math
        - Statistics
        - Physics
        - Motion and Orientation
            - Transformations
            - Quaternion operations

Physical implementation

- Can be application specific
- Here due to DAQ/DSP size and rapid exchange between vehicles
- External dorsal mount 4” housing with SLS glass filled bracket
- Power electronics for VBAT/12V/24V
  - Enable/disable high power subsystems
- Diagnostic ports
- Nitrogen backfill port



Behind the scenes Zaxy manages a lot of book keeping… so that you don't have to…for beh_land

EXTCTL.ini

```
mp
u_mission_param_a
u_mission_param_b
os
sci_generic_a nodim
sci_generic_b nodim
sci_generic_c nodim
sci_generic_d nodim
sci_generic_e nodim
sci_generic_f nodim
sci_generic_g nodim
sci_generic_h nodim
sci_generic_i nodim
sci_generic_j nodim
sci_generic_k nodim
sci_generic_l nodim
sci_generic_m nodim
sci_generic_n nodim
sci_generic_o nodim
sci_generic_p nodim
sci_generic_q nodim
sci_generic_r nodim
sci_generic_s nodim
sci_generic_t nodim
sci_generic_u nodim
is
m_depth m
m_pitch rad
m_heading rad
m_lat lat
m_lon lon
m_altitude m
m_depth_rate_avg_final m/s
sci_m_present_time timestamp
```

0 1 2 3

23 22 28 29

EXTMAP.ini

```
mp
d_bpump_value_a
d_pitch_value_b
os
m_oa_year
m_oa_year_day
m_oa_day_minute
m_oa_a_depth
m_oa_a_pitch
m_oa_a_heading
m_oa_b_depth
m_oa_b_pitch
m_oa_b_heading
m_oa_c_depth
m_oa_c_pitch
m_oa_c_heading
m_oa_d_depth
m_oa_d_pitch
m_oa_d_heading
m_oa_dsp_uptime
m_oa_band_power
m_oa_omni_noise
m_oa_beam_noise
m_oa_asgd
m_oa_line_number
```

```
const char beh_land_input_strings[BEH_LAND_MAX_INPUT_FIELDS][BEH_LAND_FIELD_LEN]={
    "m_depth",
    "m_altitude",
    "m_depth_rate_avg_final"
};

const char beh_land_config_strings[BEH_LAND_MAX_CONFIG_FIELDS][BEH_LAND_FIELD_LEN]={
    "u_land_dive_cc",
    "u_land_dive_pitch",
    "u_land_dive_thrust",
    "u_land_approach_altitude",
    "u_land_approach_cc",
    "u_land_approach_pitch",
    "u_land_approach_thrust",
    "u_land_flare_altitude",
    "u_land_flare_cc",
    "u_land_flare_pitch",
    "u_land_flare_thrust",
    "u_land_bottom_altitude",
    "u_land_bottom_depth_rate",
    "u_land_bottom_cc",
    "u_land_bottom_pitch",
    "u_land_bottom_thrust",
    "u_land_activation_depth"
};

const char beh_land_mp_strings[BEH_LAND_MAX_MP_FIELDS][BEH_LAND_FIELD_LEN]={
    "d_bpump_value_a",
    "d_pitch_value_b",
    "thrust_value_c" //seemingly ok to use same value for D/C
};
```

Behind the scenes Zaxy manages a lot of book keeping… so that you don't have to…for array

EXTCTL.ini

```
mp
u_mission_param_a
u_mission_param_b
os
sci_generic_a nodim
sci_generic_b nodim
sci_generic_c nodim
sci_generic_d nodim
sci_generic_e nodim
sci_generic_f nodim
sci_generic_g nodim
sci_generic_h nodim
sci_generic_i nodim
sci_generic_j nodim
sci_generic_k nodim
sci_generic_l nodim
sci_generic_m nodim
sci_generic_n nodim
sci_generic_o nodim
sci_generic_p nodim
sci_generic_q nodim
sci_generic_r nodim
sci_generic_s nodim
sci_generic_t nodim
sci_generic_u nodim
is
m_depth m
m_pitch rad
m_heading rad
m_lat lat
m_lon lon
m_altitude m
m_depth_rate_avg_final m/s
sci_m_present_time timestamp
```

0 1 2 3 22 23 24 25 26 27 30

EXTMAP.ini

```
mp
d_bpump_value_a
d_pitch_value_b
os
m_oa_year
m_oa_year_day
m_oa_day_minute
m_oa_a_depth
m_oa_a_pitch
m_oa_a_heading
m_oa_b_depth
m_oa_b_pitch
m_oa_b_heading
m_oa_c_depth
m_oa_c_pitch
m_oa_c_heading
m_oa_d_depth
m_oa_d_pitch
m_oa_d_heading
m_oa_dsp_uptime
m_oa_band_power
m_oa_omni_noise
m_oa_beam_noise
m_oa_asgd
m_oa_line_number
```

```
const char oa_output_strings[OA_MAX_OUTPUT_FIELDS][OA_FIELD_LEN]={
    "m_oa_year", //1
    "m_oa_year_day",
    "m_oa_day_minute",
    "m_oa_a_depth",
    "m_oa_a_heading",
    "m_oa_a_pitch",
    "m_oa_b_depth",
    "m_oa_b_heading",
    "m_oa_b_pitch",
    "m_oa_c_depth",
    "m_oa_c_heading",
    "m_oa_c_pitch",
    "m_oa_d_depth",
    "m_oa_d_heading",
    "m_oa_d_pitch",
    "m_oa_dsp_uptime",
    "m_oa_band_power",
    "m_oa_omni_noise",
    "m_oa_beam_noise",
    "m_oa_asgd",
    "m_oa_line_number" //21
};

const char oa_input_strings[OA_MAX_INPUT_FIELDS][OA_FIELD_LEN]={
    "m_depth",
    "m_heading",
    "m_pitch",
    "m_lon",
    "m_lat",
    "sci_m_present_time",
    "c_logger_on"
};

const char oa_config_strings[OA_MAX_CONFIG_FIELDS][30]={
    "u_oa_boot_time",
    "u_oa_data_interval",
    "u_oa_shutdown_time",
    "u_oa_debug_level",
    "u_oa_pdspv_interval",
    "u_oa_enable_sample_depth",
    "u_oa_enable_logger_control",
    "u_oa_max_log_size",
    "u_oa_max_log_duration",
    "u_oa_transfer_empty_logs",
    "u_oa_transfer_compressed"
};
```

Thank you

- Questions

Only a rare breed loves hash tables

- This is taken care of by Zaxy core
- We'll see that sensor drivers are isolated from this

Developing a backseat driving application

- Define parameters taken from and sent to the glider
- Develop an algorithm that consists of
  - Initialization
    - Configuration, obtain config file etc
  - Behavior controller
    - Maps desired state onto glider behavior
  - Data offload to science

A motivating Example

Dr. Donglai Gong (VIMS) and Dr. Laur Ferris preparing to deploy glider Apollo near the Jan Mayen Ridge, Nov 21

- Sponsor was happy with demonstration of G2 Hybrid glider towing old oil filled array in high north
- Q3 FY22, funded a program to integrate modern array with G3 Hybrid glider

Sponsor Requirements

- 1+ Month endurance at 100% duty cycle sampling (unless at surface)
- ~4 months to first test
- ~5.5 months to have two platforms tested, shipped abroad for high priority demo

The problems:

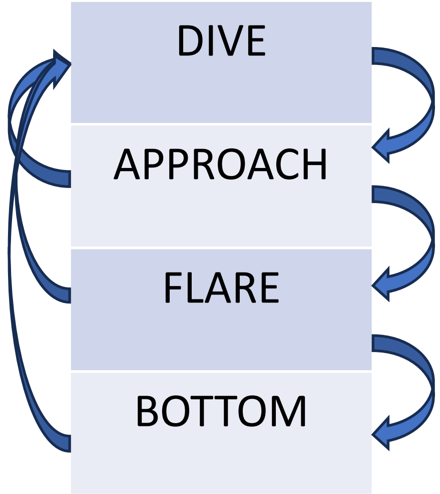
- Arrays not built yet
- Two DURIP G3 Hybrid gliders not yet available
- DSP software had not yet been completed
- No way to rapidly integrate Array and DSP

Raytheon

NPS

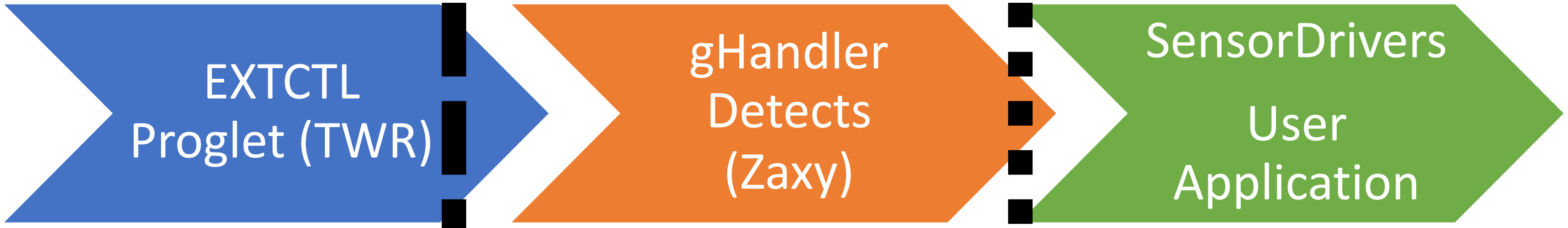
oasis A ThayerMahan Company

```
const char beh_land_input_strings[BEH_LAND_MAX_INPUT_FIELDS][BEH_LAND_FIELD_LEN]={
    "m_depth",
    "m_altitude",
    "m_depth_rate_avg_final"
};

const char beh_land_config_strings[BEH_LAND_MAX_CONFIG_FIELDS][BEH_LAND_FIELD_LEN]={
    "u_land_dive_cc",
    "u_land_dive_pitch",
    "u_land_dive_thrust",
    "u_land_approach_altitude",
    "u_land_approach_cc",
    "u_land_approach_pitch",
    "u_land_approach_thrust",
    "u_land_flare_altitude",
    "u_land_flare_cc",
    "u_land_flare_pitch",
    "u_land_flare_thrust",
    "u_land_bottom_altitude",
    "u_land_bottom_depth_rate",
    "u_land_bottom_cc",
    "u_land_bottom_pitch",
    "u_land_bottom_thrust",
    "u_land_activation_depth"
};
```




```
const char beh_land_mp_strings[BEH_LAND_MAX_MP_FIELDS][BEH_LAND_FIELD_LEN]={
    "d_bpump_value_a",
    "d_pitch_value_b",
    "thrust_value_c" //seemingly ok to use same value for D/C
};
```

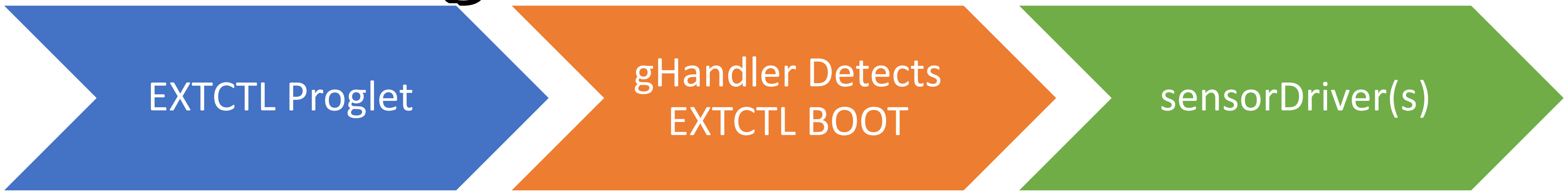
High Level Design



- Reads EXTCTL.INI at start
- Builds has table from
  - Input(IS)
  - Output(OS) sensors
  - Mission parameters (MP)

- Downloads EXTCTL.INI
- Download EXTMAP.INI
  - Equivalent to glider cache file
- Builds hash table from IS/OS/MP

High Level Design



- Run state machine which may
- request additional configuration files on boot as needed
- Monitor input struct
  - activation conditions
    - Start handling data
    - Power on a device
  - Processing
  - Set output parameters
  - Initiate calls to transfer files

Team E Slocum Sensor Integration Work

Towed Passive Acoustics:

Apollo, SN445:
- 1km Slocum G2
- 520cc pump, Hybrid
- Now with extended aft hull, extended energy bay

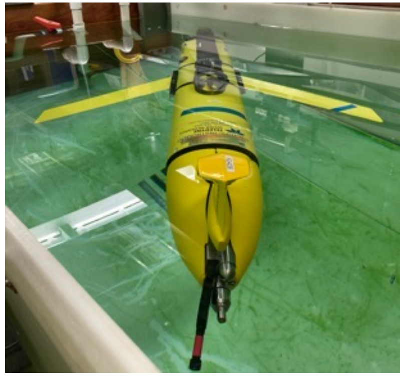
Payload:
- 40m passive acoustic array
- >30 nodes sampling >4kHz
- DSP processes array for ASGD, Transmission Loss, etc.
- Offloads reports to ebd/tbd

Freya, Seward, AK
2/22
Norse Prep

In Situ Microstructure Processing:

Boomer SN192, 1KHD, Extended Energy
Starbuck SN446, 350m, Hybrid
Freya (UAF) SN506, 1km Extended Energy

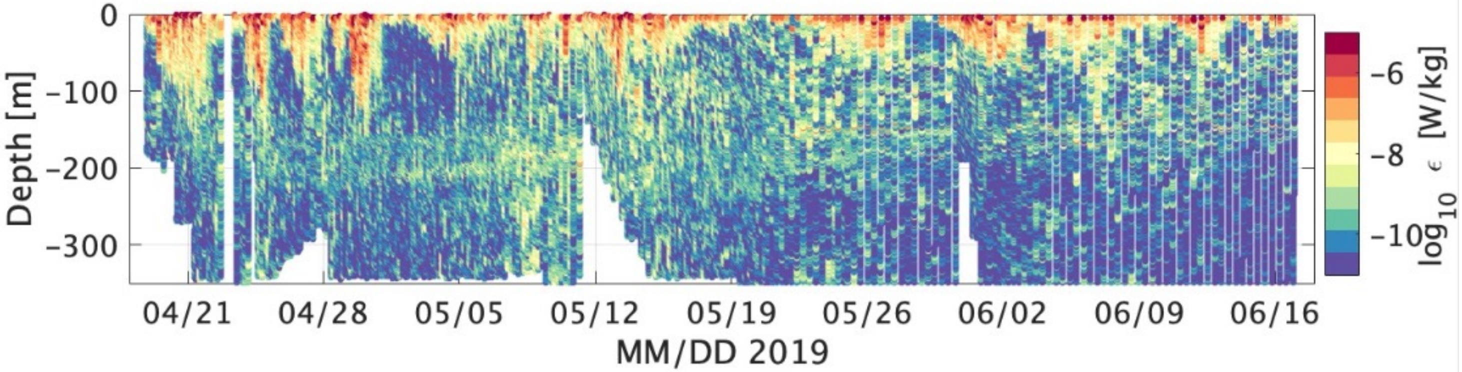
Payload:
- Rockland Scientific MR1000
  - 2 shear probe
  - 2 FP07
  - 1d velocimeter (Boomer only)
- Team E Processor (TP) (1 Ghz SOM, 2gb DDR3)
  - Pull data from MR
  - Filter, vibration removal
  - Calculate dissipation estimates
  - Pass dissipation estimates to science for telemetry



Starbuck, ARCTERX Prep, 2/22

Boomer, NISKINE Prep, 3/20

Apollo: NORSE Iceland 2021, Laur Ferris



Telemetered Dissipation Data, Apollo, NISKINE 2019

Applications

- Navigation
  - Integrate higher quality compass
  - Dynamical flight model
  - Improve estimate of Underwater position
- Improve DAC estimates
  - Onboard one-shot tidal model and path planner
- Sensor integration
  - Currently migrating acoustics and in situ turbulence processing integrations over
- Path planning using newly integrated sensors
  - Closed loop control based on environmental sensing