Inspection Class ROVs

The Pickup Trucks of the Subsea Industry

Chris Parker, Lead Engineer

SEAMOR Marine, Ltd

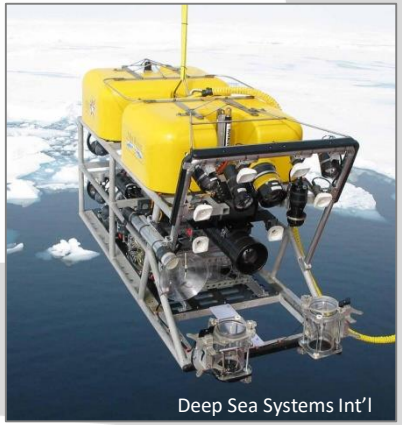
SEAMOR Marine

- Designer/manufacturer of inspection class ROVs
  - Located in Nanaimo, on the coast of Vancouver Island
  - In operation since 2006
  - Two standard vehicles: Chinook & Steelhead
  - A variety of accessories & custom ROV products

Inspection Class ROVs

- Strike a balance between large working class ROVs & small observation class ROVs
- Large enough to be interact with the environment, but not so large as to require significant infrastructure to operate
- Customizable with a wide range accessories

Deep Trekker, Inc


Deep Sea Systems Int'l

Inspection Class ROVs

What Customers Want to Buy:

- Vary widely with core industry
  - Aquaculture
  - Inspection
  - Security
  - Surveying
- Low cost
- Can change quickly
- Expandability

What Manufacturers Want to Sell:

- Standardized systems that can be mass produced
- Reduce per-system costs
- Minimize the number of SKUs
- Market agility

Building an Inspection Class ROV

- Over the past two years, SEAMOR Marine has carried out a complete overhaul of its inspection class ROVs

- Two models: Steelhead & Chinook

- Our work has been guided by a very simple idea: *Build an ROV system with the versatility & extensibility of a pickup truck*

Pickup Trucks

- Durable all-purpose vehicles
- Ubiquitous at jobsites around the world

Pickup Trucks

- Highly customizable & mass produced
- Wide range of 3rd party accessories available
  - Customers don’t have to commit to specific accessories or customizations at the time of their initial purchase
- Incredibly profitable

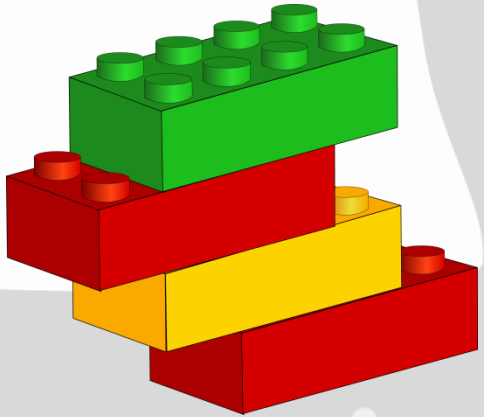



From Pickup Trucks to ROVs

- Pickup trucks enjoy universal commercial success because they are so customizable, and because they are an ideal starting point for a wide range of applications
- An inspection class ROV is a lot like a subsea pickup truck
  - How do we realize the versatility of a pickup truck in an inspection class ROV?

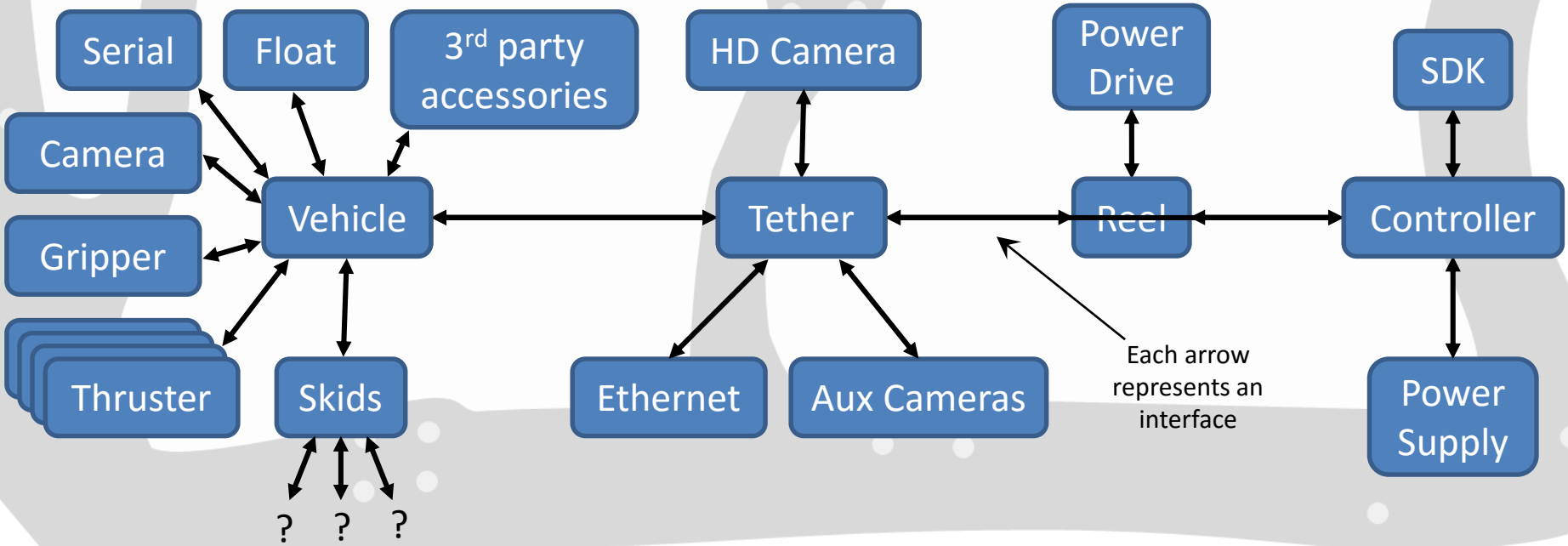
Design Philosophy

- Core idea: conceive of the ROV system as a set of interchangeable building blocks
- Fundamental problem: deciding where these interchangeable blocks meet each other, and how
  - This idea can be taken too far in either direction
- Build and test each block independently
  - Streamline production & increase reliability, reduce costs



Building Blocks & Interfaces

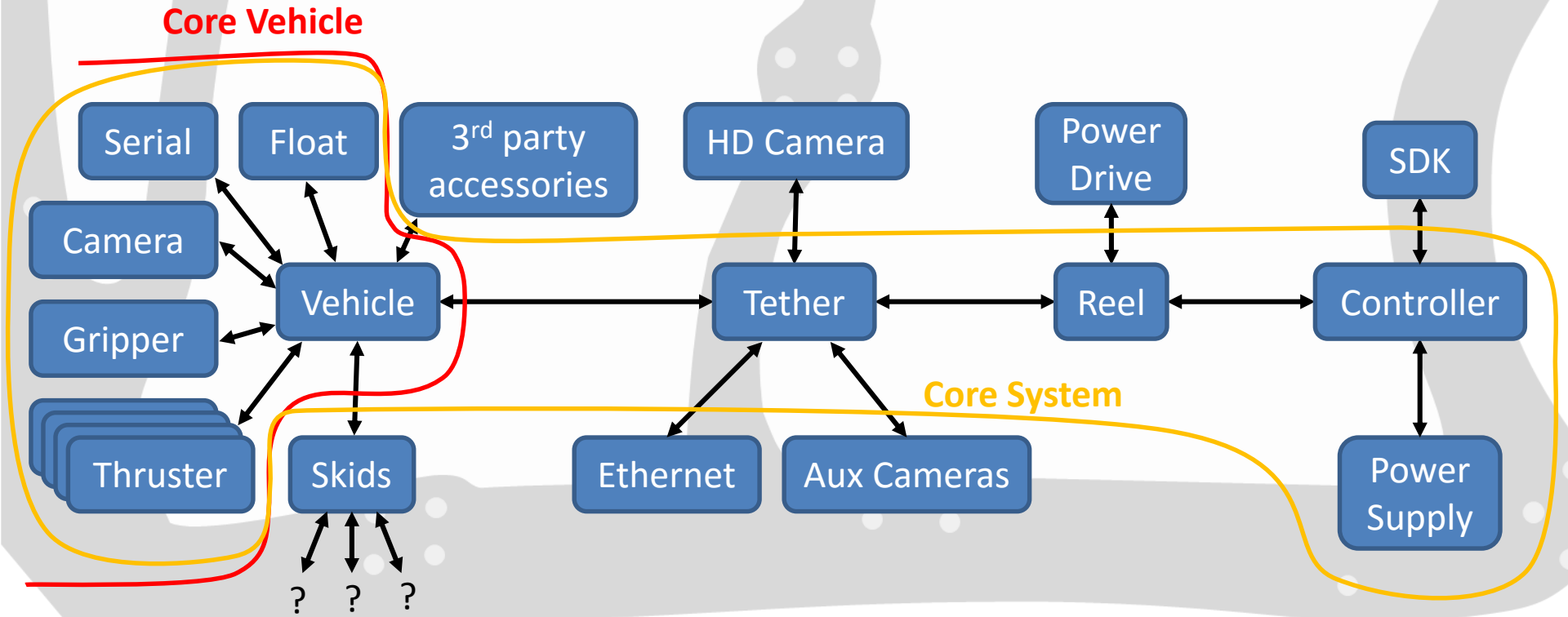
- Divide the entire ROV system into manageable components and standardize the interfaces between them. *Design to the interfaces*.



Setting Standard Interfaces

- Setting standards is *easy*, setting good standards is *hard*
- Good interfaces decouple blocks from each other
  - Enable designers to make local changes that don't necessitate system-level changes
    - Also protects manufacturers from supplier changes
  - Specifying standard interfaces is only a *means to an end*
- Following the pickup truck analogy, draw a clear line between the core vehicle, the core system, and its "accessories"
  - What is the base "package" that everyone will get?

Core Vehicle & System



Electrical Interfaces: Connectors

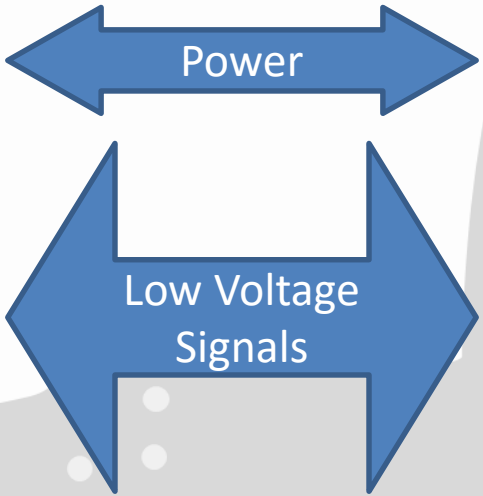
- Choose a few connector families and *stick with them*
- Unique connectors for each application is an unrealistic goal
  - Reuse a small pool of connectors intelligently
- Think in terms of *power* & *signals*
  - Put ground and power on the same pins every time a particular connector is used
  - Use same pins remaining pins for signals

Connector Example

- At SEAMOR, we use the same 8-pin whip for both our main camera canister and for Ethernet whips

Main Camera

Pin	Function
1	+48V
2	DC Ground
3	RS-485A
4	RS-485B
5	Video+
6	Video-
7	n/a
8	n/a



Ethernet

Pin	Function
1	+48V
2	DC Ground
3	TX+
4	TX-
5	RX+
6	RX-
7	n/a
8	n/a

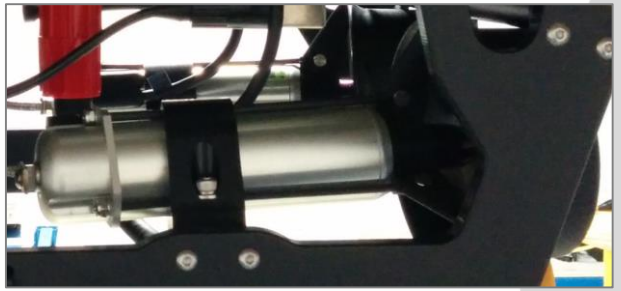
- Power and signals are in the same places for both applications
- Cross-plugging unlikely to cause damage

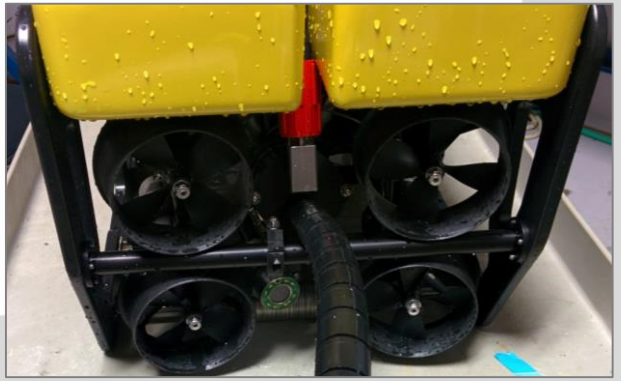
Mechanical Interface: Skids

- Pickup trucks have a standard bed sizes, ROVs have *skids*
  - Skids mount underneath the ROV to expand payload capacity
  - *Interchangeable*
  - Customize the ROV at run time
  - Can be added as needed

Thrusters

- ROV thruster design has converged to props in nozzles
- Two options for the motor: brushed & brushless
  - Brushed vs brushless is an interface decision!
  - At SEAMOR, we use Maxon brushed DC motors for almost all of our actuation
    - Reliable & robust
    - Simple 2-wire interface: fewer harness wires & no external controller required
    - Easy to test off of the vehicle (in the field)





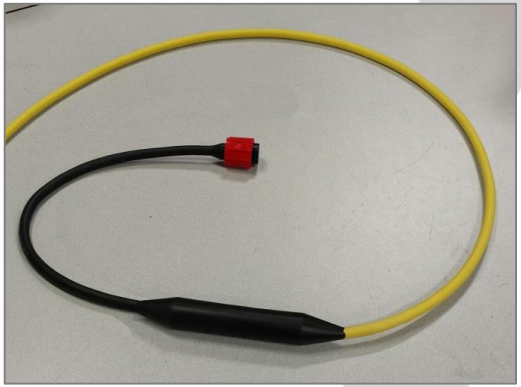
Operator Interface

- ROV features are useless unless the are accessible to the operator

- Many criteria to consider
  - Control fidelity
  - Ergonomics
  - Size & weight
  - Durability
  - Serviceability
  - Extensibility

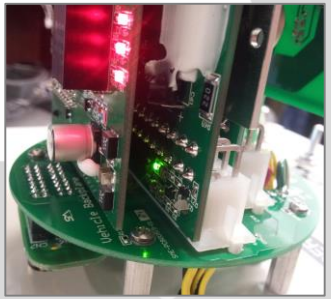
Tethers & Reels

- Deliver power and commands down to the vehicle; video and data back to the surface
  - No one tether/reel will meet every need, so plan for interchangeability
- Three interfaces to consider:
  1. Vehicle to tether
  2. Tether to reel
  3. Reel to controller
- Tether-based vehicle upgrades
  - Changing tethers should not affect the core vehicle



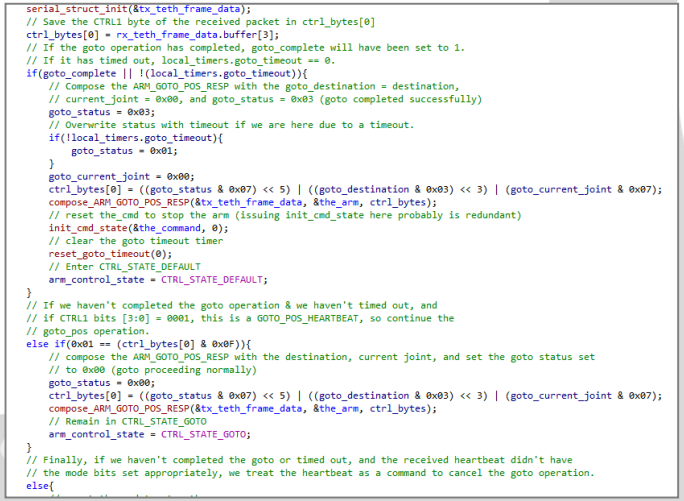
Testing & Diagnostics

- Include diagnostic systems to facilitate testing & repair of electronic building blocks
  - Significantly streamlines production
- Adding indication to PCBs is easy and inexpensive
- Empower customers & dealers to diagnose failures by building the tools right into the system
  - Failures are never good, but we should make recovering from them as painless as possible



Software/Firmware

- Customize the ROV system at *run time*, not *compile time*
  - One and only one active software release at a time
- Debug modes to speed development
- Modular code to promote maintainability
- Extensible communication
- Software Development Kits (SDK)
  - Sliding autonomy: ROV to AUV



```
serial_struct_init(&tx_teth_frame_data);
// Save the CTRL1 byte of the received packet in ctrl_bytes[0]
ctrl_bytes[0] = rx_teth_frame_data.buffer[3];
// If the goto operation has completed, goto_complete will have been set to 1.
// If it has timed out, local_timers.goto_timeout == 0.
if(goto_complete || !(local_timers.goto_timeout)){
    // Compose the ARM_GOTO_POS_RESP with the goto_destination = destination,
    // current_joint = 0x00, and goto_status = 0x03 (goto completed successfully)
    goto_status = 0x03;
    // Overwrite status with timeout if we are here due to a timeout.
    if(!local_timers.goto_timeout){
        goto_status = 0x01;
    }
    goto_current_joint = 0x00;
    ctrl_bytes[0] = ((goto_status & 0x07) << 5) | ((goto_destination & 0x03) << 3) | (goto_current_joint & 0x07);
    compose_ARM_GOTO_POS_RESP(&tx_teth_frame_data, &the_arm, ctrl_bytes);
    // reset the_cmd to stop the arm (issuing init_cmd_state here probably is redundant)
    init_cmd_state(&the_command, 0);
    // clear the goto timeout timer
    reset_goto_timeout(0);
    // Enter CTRL_STATE_DEFAULT
    arm_control_state = CTRL_STATE_DEFAULT;
}
// If we haven't completed the goto operation & we haven't timed out, and
// if CTRL1 bits [3:0] = 0001, this is a GOTO_POS_HEARTBEAT, so continue the
// goto_pos operation.
else if(0x01 == (ctrl_bytes[0] & 0x0F)){
    // compose the ARM_GOTO_POS_RESP with the destination, current joint, and set the goto status set
    // to 0x00 (goto proceeding normally)
    goto_status = 0x00;
    ctrl_bytes[0] = ((goto_status & 0x07) << 5) | ((goto_destination & 0x03) << 3) | (goto_current_joint & 0x07);
    compose_ARM_GOTO_POS_RESP(&tx_teth_frame_data, &the_arm, ctrl_bytes);
    // Remain in CTRL_STATE_GOTO
    arm_control_state = CTRL_STATE_GOTO;
}
// Finally, if we haven't completed the goto or timed out, and the received heartbeat didn't have
// the mode bits set appropriately, we treat the heartbeat as a command to cancel the goto operation.
else{
```

Summary

- Designing around interfaces requires significant up-front planning and work, but it quickly pays off
- Enables engineers to focus on the problem in front of them
  - Bounding a problem *always* simplifies the design process
- Well-designed interfaces facilitate IP reuse
  - Develop a library of reusable sub-components

Summary

- Turn-around time for custom systems has been greatly reduced
  - Truly custom systems are rare – just different combinations of standard parts
- A wide range of upgrades are now trivial
  - Equipment does not need to be shipped back for upgrades
- System reliability has increased significantly
  - Service also has streamlined
- Customers are attracted to the grow-as-you-go model of ROV ownership

Thank you