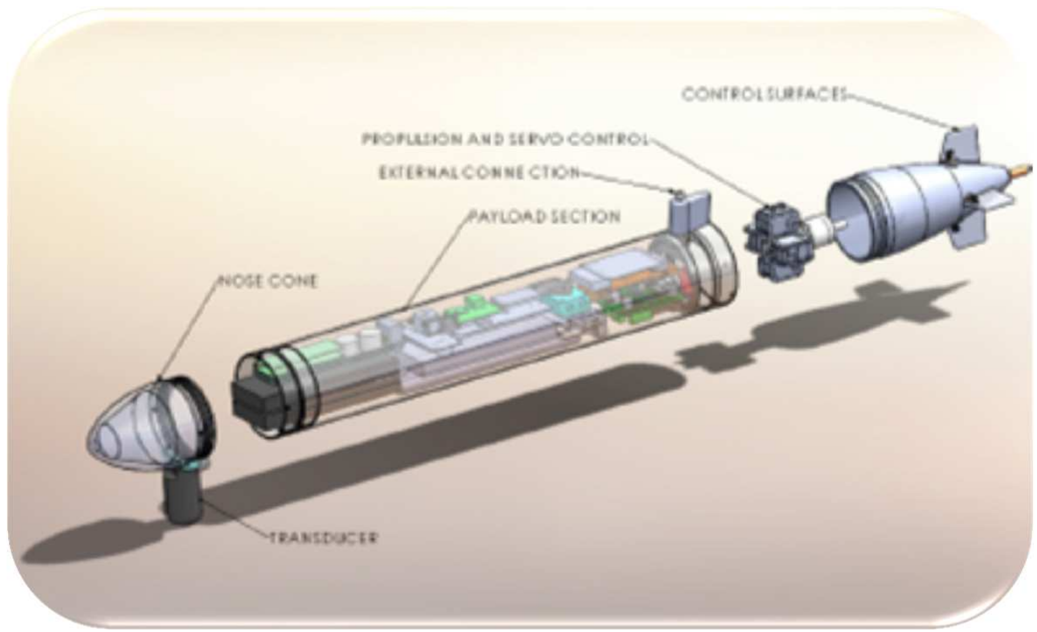
An Implementation of ROS on the Yellowfin Autonomous Underwater Vehicle (AUV)



[1]

Sam Shue

The Yellowfin AUV

- Designed for Multi-Robot cooperation projects
- Can be refitted with sensors for underwater sampling and mine detection
- Uses 2 Microcontrollers – one for high level operations and one for sensing and control
- Weighs less than 17 lbs.

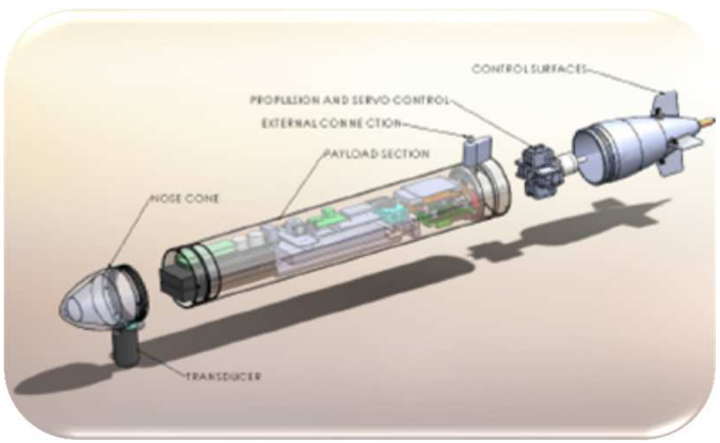


Figure 1: The Yellowfin [1]

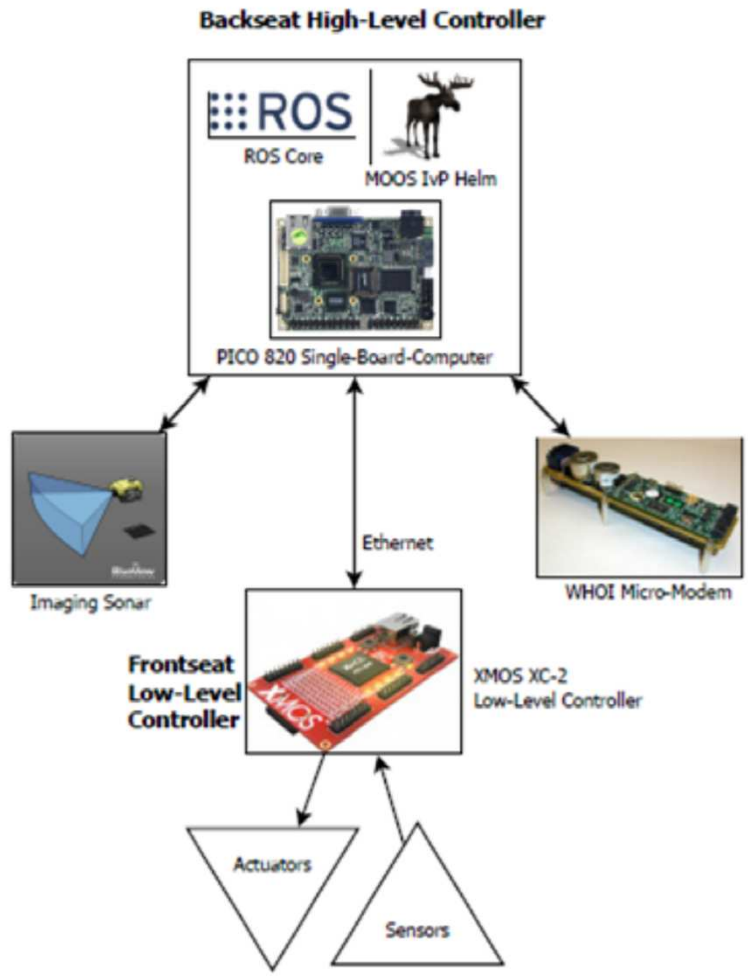


Figure 2: System Architecture [1]

Microcontrollers and Sensors

- XMOS XC-2,
  - Four-core XS1-G4 Microcontroller
  - Ethernet Controller
- WHOI MicroModem
  - Low-speed Communications
- Pico-ITX Single-Board-Computer
  - Intel Atom Z510
  - 2 GB RAM
  - Ubuntu
- Imaging Sonar

XMOS XC-2

- Quad Core CPU, which each core can run up to 8 threads
- The Robot's Control loops are implemented on the microcontroller to reduce latency
- Reads sensor data and then transmits the data back to the single board computer through ethernet
- Interfaced with the IMU (Intertial Measurement Unit), Leak Sensor, Pressure Sensor, and Digital Compass

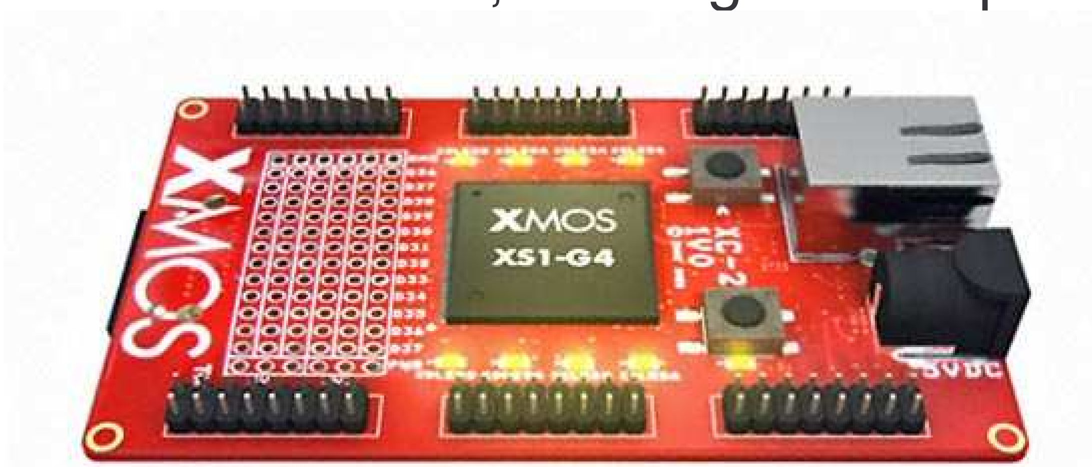
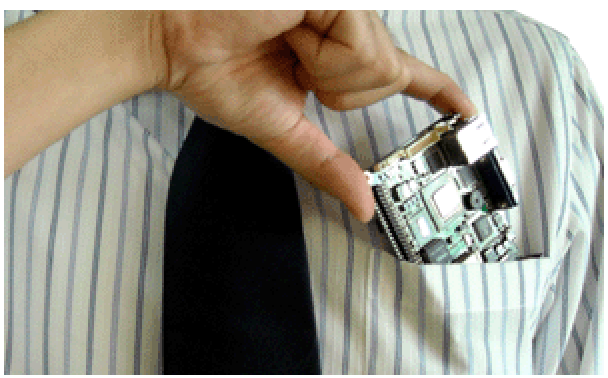


Figure 3: The XMOS XC-2 Board [4]

Pico-ITX Single Board Computer

- Intel® Atom™ Z510/ Z530 Pico-ITX SBC
- VGA/LVDS
- SATA
- LAN
- SDIO (Secure Digital Input Output, used to boot Ubuntu)



[5]

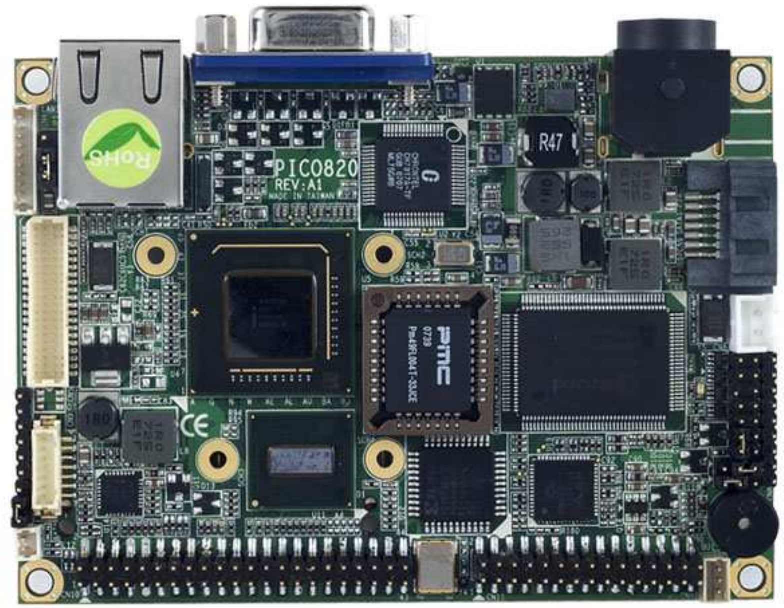


Figure 7: The Pico-ITX SBC [5]

ROS (Robotic Operating System)

- ROS is a software framework for developing robotic software
- It provides numerous libraries for sensors, robotic drive bases, path planning and localization algorithms, and more
- ROS allows multiple software to communicate with each other through a publishing and subscription method
- ROS is designed to be hardware “agnostic,” allowing each algorithm and hardware driver to be completely modular, so essentially any code can run on any robot

MOOS (Mission Oriented Operating Suite)

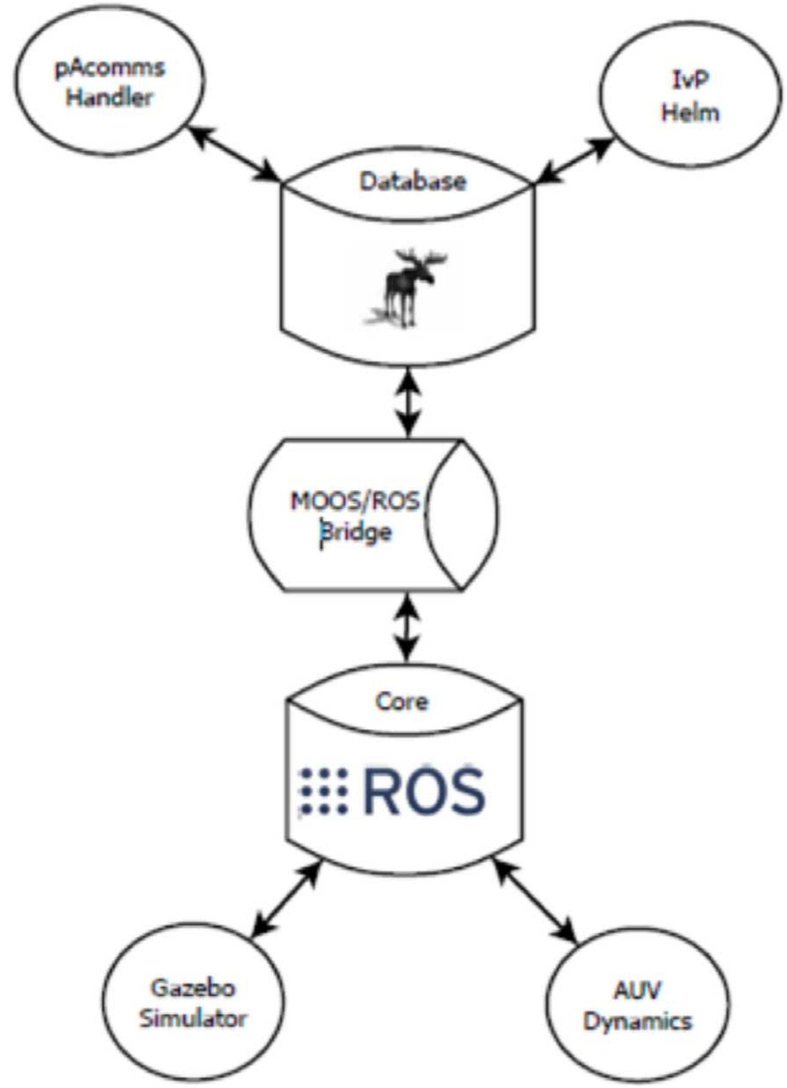
- Another Publish and Subscribe messaging system similar to ROS
- While MOOS has historically been popular within the underwater robotics community, the Robot Operating System(ROS), has been widely accepted among the academic ground and aerial robotics community.
- Unlike ROS, MOOS's publishing and subscribe system is not peer-to-peer, but instead all data goes through the central MOOS database

Integrating ROS and MOOS

- A bridge node joins ROS and MOOS

- Designed so information could be relayed without changing the bridge code itself

- Reads an xml file that specifies the messages that will be used on startup



Auctioneer and Bidder Task Planning

- A task is presented to the network of UAV
- Each robot places a "bid" to the "Auctioneer," which is the source of the task
- After all the bids are in, the robot with the lowest bid is assigned the task
- A bid's cost is determined by the robot's distance away from the area of interest
- This model allows for easy scalability of the network

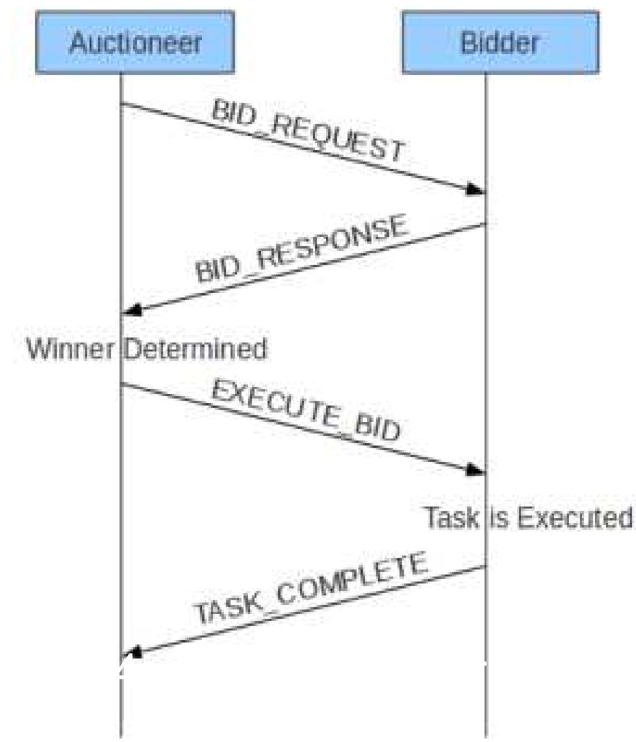


Figure 5: The Auctioneer Communication Protocol [1]

AUV Equations of State

- Six Degrees of Freedom described by 12 state variables
- Fossen's AUV Model $\eta_1 = [x, y, z]^T; \quad \eta_2 = [\phi, \theta, \psi]^T;$
  - x, y, and z variables represent the vehicle's position
  - Φ,Ө, and ψ represent the vehicle's orientation relative the earth's fixed reference frame

$$\nu_1 = [u, v, w]^T; \qquad \nu_2 = [p, q, r]^T$$

  - The u, v, and w represent the vehicle's linear velocities and p, q, and r represent the rotational velocities with respect to the fixed reference frame

$$\tau_1 = [X, Y, Z]^T; \qquad \tau_2 = [K, M, N]^T$$

  - The tau variables represent external forces on the vehicle

AUV Equations of State (continued)

- The position of the vehicle is described by Euler Angles
- The following rotation matrix gives the pose of the robot in Euler angles:

$$J_1(\eta_2) = \begin{bmatrix} c\psi c\theta & -s\psi c\phi + c\psi s\theta s\phi & s\psi s\phi + c\psi c\phi s\theta \\ s\psi c\theta & c\psi c\phi + s\phi s\theta s\psi & -c\psi s\phi + s\theta s\psi c\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix}$$

$$s\cdot \equiv \sin(\cdot) \quad c\cdot \equiv \cos(\cdot) \quad t\cdot \equiv \tan(\cdot)$$

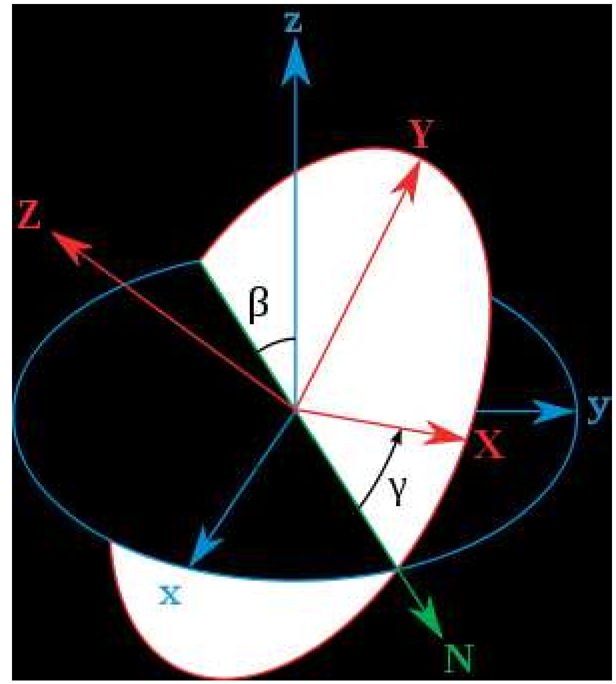


Figure 8: Euler Angles [6]

- The following relationship computes the translational velocities:

$$\nu_1 = J_1^{-1}(\eta_2)\dot{\eta}_1 .$$

AUV Equations of State (continued)

- The Updated Pose can be computed with:

$$J_2(\eta_2) = \begin{bmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\phi & -s\phi \\ 0 & s\phi/c\theta & c\phi/c\theta \end{bmatrix}$$

- With the relationship:

$$\nu_2 = J_2^{-1}(\eta_2)\dot{\eta}_2.$$

- AUV dynamics can be compactly expressed as:

$$M\dot{\nu} + C(\nu)\nu + D(\nu)\nu + g(\eta) = \tau.$$

M represents the vehicle's inertia with added mass, C() is the matrix that includes the Coriolis* and centripetal terms, D() is the damping matrix, g() is the vector of gravitational forces and moments, and is the vector of control inputs.

* the **Coriolis effect** is a deflection of moving objects when they are viewed in a rotating reference frame [6]

NASA WorldWind

- Open Source Java Application for plotting on a map
- ROSJava used to interface with WorldWind
- WorldWind displayed the locations of the robots and paths on a map GUI

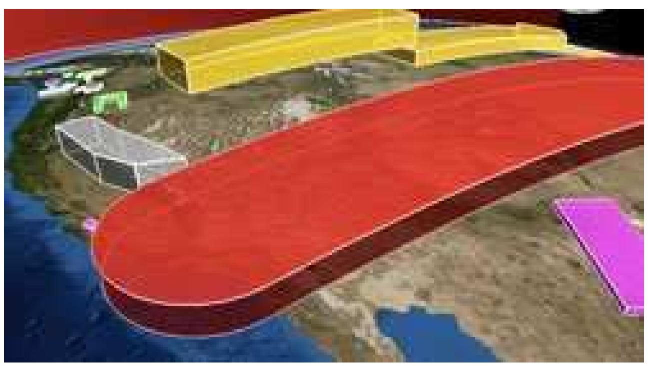


Figure 6: NASA Worldwind [3]

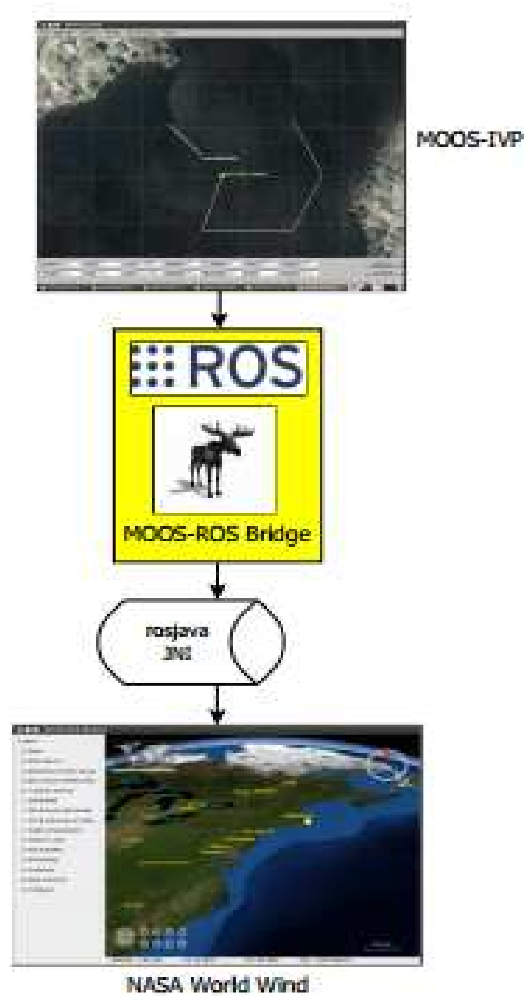


Figure 7: MOOS To WorldWind [1]

Conclusion

- Integration of ROS/MOOS node was successful
- Using both ROS and MOOS allows further development using libraries and features on both systems
- Path planning and mission distribution could be improved in future work, by allowing the system to determine whether an auctioneer method would work or not

References

[1] DeMarco, K.; West, M.E.; Collins, T.R.; , "An implementation of ROS on the Yellowfin autonomous underwater vehicle (AUV)," *OCEANS 2011* , vol., no., pp.1-7, 19-22 Sept. 2011
[2] ROS.org
[3] http://worldwind.arc.nasa.gov/java/
[4] http://www.nlvocables.com/blog/?p=207
[5] http://www.axiomtek.com/products/ViewProduct.asp?view=680
[6] http://en.wikipedia.org/wiki/Euler_angles