Underwater ROV Control Package

P09201 | Project Review
Kate Gleason College of Engineering
Multidisciplinary Senior Design II

Brandt | Gregory | Lavrich | Letourneux | Savage | Stanton
Our Team

Jon Savage
GUI Design
EECC | 09

William Stanton
Hardware Design / Testing
EEEE | 09

Colby Gregory
Team Lead / Hardware
EEEE | 09

Florian Leterneux
Hardware / PCB Design
EEEB | 09

Dan Brandt
Software Interface
EECB | 09

Bryan Lavrich
Digital Design
EEEB | 09

William Stanton
Hardware Design / Testing
EEEE | 09

Colby Gregory
Team Lead / Hardware
EEEE | 09

Florian Leterneux
Hardware / PCB Design
EEEB | 09

Dan Brandt
Software Interface
EECB | 09

Bryan Lavrich
Digital Design
EEEB | 09
Project Description

In order to provide an end user with the means of controlling RIT's underwater ROV, this project must create a system that will allow the user to interface with the ROV in real time. An ancillary requirement for providing this system is that this project must also enable and streamline the communications between the current and future individual components of the ROV.
Project Scope

- Provide a user-friendly graphical interface for controlling the lighting and thrusters of the ROV.
- Define software communication protocols to be used by all current and future ROV modules.
- Define hardware communication interface to be used by all current and subsequent ROV modules.
- Design a bridge for easy networking of modules.
Customer Needs

- All components must fit in standard housing already developed
- Develop testing apparatus to show operation of software and interface
- Design should be modular, allow for adding of components easily
- Minimize power efficiency and heat dissipation
- Must be scalable, must be able to add additional bridges to the interface
- RS232 interface to surface
Customer Needs

- Must accept user input through a GUI
- Software should control light intensity, spectrum, thrust direction and speed
- Control both surrogate lights and thrusters simultaneously
- Open Source Software Implementation
- GUI incrementally adds modules as devices are attached
Customer Needs

- Design documentation is a must
- Both HW and SW should be able to control a slave bridge
- HW/SW Command prioritization
- Interfaces with other projects (to some extent)
- Anticipate the migration to land vehicles, not necessarily plug and play compatible
Our Concept

- Starts with a GUI and a user input
- Users command is then wrapped and sent down the communication and interfacing layers
- Command ends up at interface board (aka 'The Bridge'), routes to device
- This is a two way path, feedback is anticipated from devices
Our Concept (user)
Our Concept (data)
Our Concept - Bridge

- Our hardware interface between software and the device
- 1 Input - 10 Outputs, can be made up to 16
- Completely expandable to other bridges
- RS232 to surface
- EIA485 to devices and other bridges
- Uses a time sharing algorithm to allocate a fair amount of time to each module
- Programmed through JTAG
The Bridge

Power Circuit
Atmega328
Programming Header
RS232 Interface
Bridge Mode Selector
Daisy Chain
EIA485 (10x)
USART
Design Challenges

• PCB Layout issues
• Soldering (by hand) 0.2mm pin spacing
• Testing our hardware and software
• Reliability
Testing

- We designed a test fixture to emulate a thruster, light, or 'gamma device'.
- Software on the host computer facilitates the ability to switch between what the test fixture will look like to the bridge.
- Allows the comprehensive test of the interface between hardware and software subsystems.
Risk Assessment

- SW Risks
  - Communication problems
  - Coding standard violations
  - SW/HW integration
- HW Risks
  - Prototyping
  - Noise
  - PCB layout
  - Drive capacitance
Murphy’s Law

- Software components work excellent on their own, not necessarily together
- Test fixture glitches
- ‘Connectorizing’ the system
- GUI issues, crashing on device removal
- Two-way communication
- Flow control on the bridge
Beating Murphy's Law

- Debug-debug-debug!
- Capacitors seem to solve all glitching problems
- Generous donation from L3 GCS solved the connector issue
- More debugging and feature addition solved the GUI crash
- Implemented after test fixture reliability was proven
Current State of Design

- 1.75 Bridges, 2 Test benches
- PCB issue has been fixed in design software
- Software 90% implemented
- Still some bugs to work out
- All issues are documented and published on EDGE
Lessons Learned

- Hardware design process MUST be followed
- Design >> Prototype >> Implement >> Debug
- Hardware has to be finished before software can be fully tested
- Measure twice, cut once
- Talk to your customer, despite what Todd says
- All disciplines must remain in contact
Special Thanks

- Edward Hensel, PhD
- Wayne Walter, PhD
- Todd Fernandez
- Jeffrey G. Lonneville
- Ken Snyder
- James Stefano
- Daniel Phillips, PhD
- Dorin Patru, PhD
Q&A