Acoustic communication Task - Protocol

The acoustic communication between the AUV and a cooperating platform is required to accomplish our vision to have multiple interoperable AUVs working to accomplish a common goal in future SAUC-E competitions. This year we are introducing the concept for the first time with the simple messages passed between the two platforms.

Considering the small bandwidth and slow transmission rates of the acoustic communication an optimized communication protocol has to be determined. Common modems that are available to the competitors for SAUC-E ’12 are compact and can be added to a platform fairly easy. However, the 'Standard, Off the Shelf' Tritech Mini-Modems are only capable of 'point-to-point' communications, and can send/receive simple 'Raw' 4 byte packets. There is no addressing or error checking available with this protocol, and it is up to the users to include their own error checking capability.

The acoustic communication data payload is 4 bytes and 2 data values should be transmitted from the competitors AUV to the cooperative platform. First data value is the identifier (message coming from your AUV should be 0, with 1 bit resolution), then current AUV heading (10 bit resolution, range 0 to 360 degrees) and current AUV depth (4 bit resolution, range 0 to 10 meters), then if the light is OFF or ON (0 for OFF and 1 for ON, 1 bit resolution), and the rest you can use for various boolean info. For you sake the last byte(s) should be a CRC byte(s). See Table 1.

	Value	Range	Description
Data [1]	0	[0, 1]	Identifier
Data [2]	psi, depth	[0, 360], [0, 10]	Cur. Heading&Depth
Data [3]	ON / OFF	[0, 1]	Light on or off
Data [4]	CRC or boolean	you decide	Error correction

Table 1: Data[1] value and related message type – AUV to Collaborator

For the message that you should receive, the first data value is the identifier (message coming from the collaborator should be 1, with 1 bit resolution), then commanded AUV heading (10 bit resolution, range 0 to 360 degrees), and finally commanded AUV depth (4 bit resolution, range 0 to 10 meters). For you sake the last byte(s) should be a CRC byte(s). See Table 2.

	Value	Range	Description
Data [1]	1	[0, 1]	Identifier
Data [2]	psi_cmd	[0, 360]	Desired Heading
Data [3]	dpth_cmd	[0, 10]	Desired Depth
Data [4]	CRC or boolean	you decide	Error correction

Table 2: Data[1] value and related message type – Collaborator to AUV

NOTE: The competitors should provide the software that can be loaded on Windows or Linux laptop which will be loaded on the ASV or rubber boat computer so the judges can receive the AUV messages and send the heading commands ([0, 360]) upon the AUV has finished the circle search.

A judge will be close to the mid water target (close to the wall) and note if the light was on or off for the respective AUV heading. Another judge will receive your messages (and send commands) on a laptop via: 1) a wireless link from our ASV, 2) a modem on a pole and laptop on the rubber boat, 3) a modem on a cooperating AUV – this case we will evaluate the log. The main interest for the judges is that they can view the arriving messages, be able to send the heading and depth commands and compare with the observations.

According to Marcus Cardew, a former Research and Development Director, 'Raw' 4 byte packets are at the limits of receiver SNR and it is likely for bit errors to occur. The recommended 'Most Rugged' method of bit error detection is to use the last 2 bytes in a packet for a 16bit CRC, but any other choice is up to the user. CRC16 is one of the options which works very well, but the competitors can consider whether it makes sense to look deeper at the The Bose, Chaudhuri, and Hocquenghem (BCH) error correcting codes.

Namely, Jeff Neasham (Newcastle University), who is the guru of all things acoustic and communications, says the following: “If they only want about 12 bits of data then I think your suggestion of 2 data bytes + a CRC-16 is a good plan and easy to implement. The CRC-8, although I have used it, is not very robust and not really much better than a straight 8-bit parity sum. Better still might be to use your ~50% overhead for an error correction code such as a (31,16) BCH code which would give you 16 data bits + 15 parity bits, the ability to correct up to 3 bit errors and similar error detecting capability to a CRC-16. This could be really powerful and I think we should strongly consider this for future spread spectrum modems with short packets. Here is a reference although like everything in coding it is math heavy: http://www.aqdi.com/bch.pdf.”

Your software should handle the serial communication and message decoding. Binary data received from the modems should be encoded in a dictionary XML format. Be advised that the binary data from the modems is returned differently based on the modem type (applies to the competitors with other modems). We focus on the Tritech Micron modems that are used for AUV communication. Modem is abstracted through a serialization interface that allows decoding from binary data to a generic XML format. The modem class should (but not required) load one of the desired translation plugins based on the XML configuration. It should establish a serial communication with the device and collects the messages. When the required number of packets is accumulated the messages should decoded using your implementation. We suggest that the class is callback oriented and a callback is registered by your modem application. When a modem message is successfully received and decoded the callback should be triggered and the application should publish the new modem data to the some sort of display.

As a reference please see the following communication protocol:

http://acomms.whoi.edu/publications/CCL%20April%202005%20Public%20Release%201.0.pdf