The Joint Architecture For Unmanned Systems: A Subsystem of the Wunderbot 4

Jeremy Crouse Computer Engineering
Elizabethtown College
crousej@etown.edu

Abstract—**The Wunderbot 4 is going to be competing in the Intelligent Ground Vehicle Competition (IGVC) and as part of the competition we, the team, must make changes to the current system. We are in the process of rewriting the entire program that runs the Wunderbot in order to make future implementation easier and current implementation of new subsystems understandable. As part of the team I am in charge of programming the Joint Architecture for Unmanned Systems (JAUS) protocol used by the Department of Defense (DoD) which is a challenge in the upcoming competition.**

***Index Terms*—**
OCU – Operator Control Unit
JAUS – Joint Architecture for Unmanned Systems
IGVC – Intelligent Ground Vehicle Competition
WGS – World Geodetic System
UDP – User Datagram Protocol
ASCII – American Standard Code for Information Interchange
GUI – Graphical User Interface
VI – Virtual Instrument

I.INTRODUCTION

The Intelligent Ground Vehicle Competition is a yearly competition where universities and colleges come together to compete and show their achievements in the field of robotics. There are four main competitions that take place at the IGVC: Design, Navigation, Autonomous and JAUS challenges. In the past the Wunderbot has not competed in the the JAUS part of the competition since it was optional and no one on the team had time to work programming that needed done to compete in this area. Needing someone to work on this project for the competition and being new to the team at the time I chose to undertake this project.

II.RESEARCH

The Joint Architecture for Unmanned Systems is an architecture specified for use in research, development and acquisition for unmanned systems. The JAUS protocol is an emerging universal command language that all robots under the Department of Defense will use so that different organizations don't have to upgrade to the most current hardware and software technology in order for older versions of autonomous vehicles to continue interoperability with newer autonomous vehicles. This is strictly a message set that was developed by the Department of Defense to allow for interoperability.

The JAUS architecture has to support all classes of unmanned system, rapid technology insertion, interoperable operator control unit, interchangeable/interoperable payloads, and interoperable unmanned systems. According to the IEEE interoperability is defined as the ability of two or more subsystems to exchange information and to use the information that has been exchanged [1]. Classes of unmanned systems can range from air, ground, water and size (e.g. throwable to ship-sized) should not to be an issue when it comes to interoperability.

Interoperable operator control units (OCUs) can range in size with each having different levels of functionality and JAUS should allow for the interoperability of the operator control units. As concepts are developed and technology improves the ability of payload capabilities, organizations must have the ability to update the payload without redeveloping the whole unmanned system and JAUS has to allow technical advancements while not imposing specific hardware or software implementations.

Since some missions are not possible with just one unmanned vehicle JAUS must allow for communication between multiple unmanned systems no matter what platform they are running. JAUS must also be independent of technology, should not allocate functions to any particular element of the system, should not define or measure system performance, should not infringe upon intellectual property and data rights, should facilitate diverse operational procedures should focus on open standards and architectures and should be flexible enough to support a wide range of possible missions and be independent of operator use.

JAUS was also made to be independent of vehicle platform, mission, computer resource, technology and operator use. Vehicle platform independence means that no assumptions are made about about underlying vehicle since unmanned systems include a wide spectrum of platforms.

Mission Independence mans that JAUS must isolate mission specific functions because of there bing so many different types of uses and missions that unmanned systems can be used for. This allows for JAUS to be independent of any specific mission or set of missions that may be built into the unmanned system.

Computer resource independence means that JAUS must maintain computer hardware independence in order to be applicable to all unmanned systems. Technology independence means that JAUS cannot be built around a specific technology solution or else it may eliminate a superior solution. Operator use independence is the last requirement of JAUS which means that it must be independent of any uses the operator will have for the unmanned system. JAUS is not supposed to restrain the user in determining the best approach to accomplishing a mission.

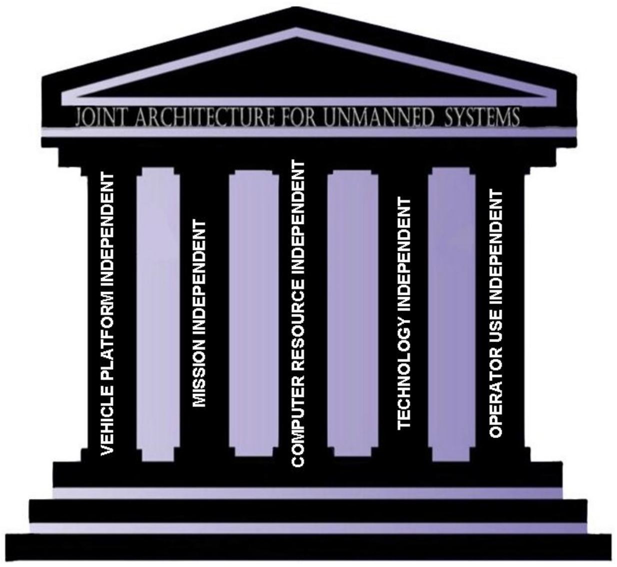

JOINT ARCHITECTURE FOR UNMANNED SYSTEMS
VEHICLE PLATFORM INDEPENDENT
MISSION INDEPENDENT
COMPUTER RESOURCE INDEPENDENT
TECHNOLOGY INDEPENDENT
OPERATOR USE INDEPENDENT

Figure 1: JAUS Independence requirements [3]

There are three different levels of compliance for joint architecture for unmanned systems. The levels are inter-subsystem, inter-nodal, and inter-component. According to the JAUS hierarchy a subsystem is a logical stand-alone operational entity such as an OCU or unmanned vehicle. Within a subsystem the computer processors are roughly defined as nodes and software processes executing within a node is defined as its components.

Level one compliance, inter-subsystem, covers the requirements of communication between subsystems and its purpose is to support the interoperability of the subsystems. Such subsystems that are included in level one compliance are robot to robot, robot to controller and controller to controller. Level two compliance deals with the requirements between nodes and its purpose is to support the interoperability of the nodes.

The nodes involved with this level of compliance are payload to payload and payload to on-board controller just to name a couple. Level three compliance addresses the requirements between the components. Since JAUS is still fairly new this level of compliance is still not defined within the JAUS standards and is not testable at this time.

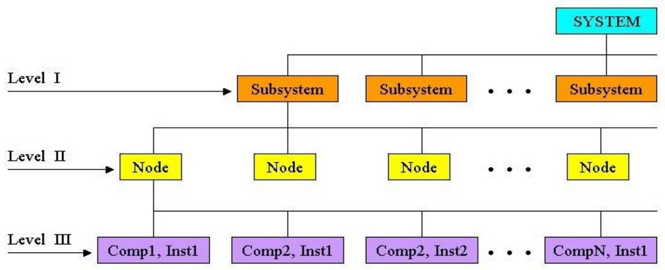


Figure 2: JAUS Levels of Compliance [4]

In the JAUS portion of the IGVC competition we will be using two of the many components that JAUS has implemented into its system. We will be sent information for the Primitive Drive (ID 33 given by JAUS) which is the component that performs basic driving and all platform related mobility functions.

This also includes other devices such as an engine and lights if present. Since JAUS has to promote interoperability the particular platform of wheels (e.g. tracked or not) and power plant (e.g. gasoline, diesel or battery) is not an issue.

The other component that we will be using in the JAUS challenge is the Global Pose Sensor (ID 38 given by JAUS). This component determines the global position and orientation of the platform and the reports the given latitude, longitude, and elevation in accordance with the WGS 84 standard. All this information is being sent via wireless Ethernet to an access point on the robot.

The packet structure that the Department of Defense has chosen to use is the User Datagram Protocol (UDP) protocol and as stated by JAUS compliance only one JAUS message is alloted per UDP packet sent.

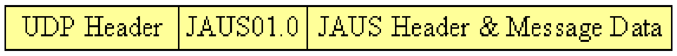
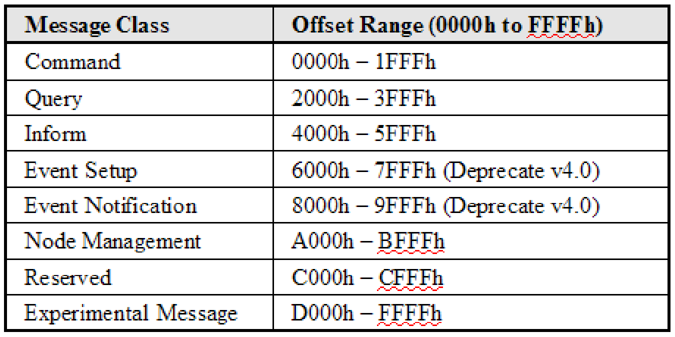


Figure 3: Basic UDP setup with JAUS incorporated

JAUS defines six different classes of messages at the component level. This segmentation is used as a method of organizing the messages and also allows message command codes to be masked prior to evaluation to help with the message transaction process.



Message Class	Offset Range (0000h to FFFFh)
Command	0000h – 1FFFh
Query	2000h – 3FFFh
Inform	4000h – 5FFFh
Event Setup	6000h – 7FFFh (Deprecate v4.0)
Event Notification	8000h – 9FFFh (Deprecate v4.0)
Node Management	A000h – BFFFh
Reserved	C000h – CFFFh
Experimental Message	D000h – FFFFh

Figure 4: Segmentation of Command Codes by class [6]

The command message class is used to effect system mode changes, actuation control, alter the state of a component or subsystem or to initiate some other type of action. The query message class is used to solicit information form another component. The inform message class allows components to transmit information to each other (e.g. status reports, geographic position, state information). The event setup message class is used to setup the parameters for an event notification message and to have a component start the monitoring for the event trigger.

The event notification message class communicates the occurrence of the event (e.g. engine over temperature, oil pressure, etc). The node management message class is only used by the node management task and is used for node specific communications (e.g. configuration information, component registration). Lastly the experimental message class is used to provide a mechanism for experimentation with new messages that have not been defined in the JAUS reference architecture. This class is mainly there for the expansion of the current message set.

All JAUS messages are required to have a header and data fields. JAUS is setup in this format because it is common to all messages and it allows JAUS to employ an embedded protocol which means that certain fields within the header provide information on how to handle the message. Each field in the message header is interpreted as an unsigned integer value and the header contains information regarding the properties, data size, handling requirements, data encoding and decoding and message routing.

Field #	Field Description	Type	Size (Bytes)
1	Message Properties	Unsigned Short	2
2	Command Code	Unsigned Short	2
3	Destination Instance ID	Byte	1
4	Destination Component ID	Byte	1
5	Destination Node ID	Byte	1
6	Destination Subsystem ID	Byte	1
7	Source Instance ID	Byte	1
8	Source Component ID	Byte	1
9	Source Node ID	Byte	1
10	Source Subsystem ID	Byte	1
11	Data Control (bytes)	Unsigned Short	2
12	Sequence Number	Unsigned Short	2
	Total Bytes		**16**

Figure 5: JAUS message header data format in bytes [6]

The message properties outlined in field one of the message header data are split into six distinct bit-fields. The message priority supports values ranging from 0 to 15. The default priority is what is being sent to us in the IGVC JAUS challenge.

The acknowledge/negative acknowledge behavior is set to none for the IGVC JAUS competition. Property bit six is the service connection indicator which is always set to zero unless a service connection is needed. The experimental bit should always be set to JAUS and the version bits get set to two for the JAUS challenge. The rest of the message properties are not used at this time and are being used for future expansion of the JAUS protocol and are set to zero.

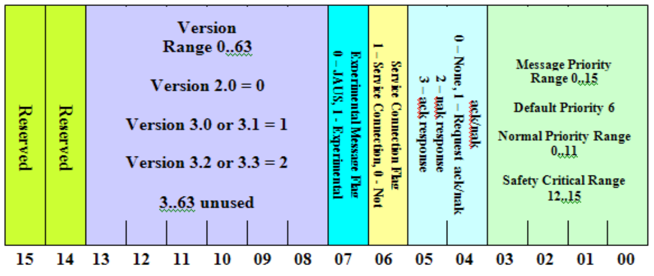


Figure 6: Message Property detailed structure [6]

The header defines the command code in field two which is a two byte numeric value that specifies the type of message and size of its required data and it encoding and decoding characteristics. The source and destination ids that are defined in fields three through ten identify where the message data is coming from (source) and where the message data should be sent to (destination).

These two fields deal with routing commands through the unmanned system. The data control field from bits zero to eleven are for the size of data per transaction and data flag bits from twelve to fifteen are used to control single/multi packet transactions. The last field is the sequence number which is used to serialize messages.

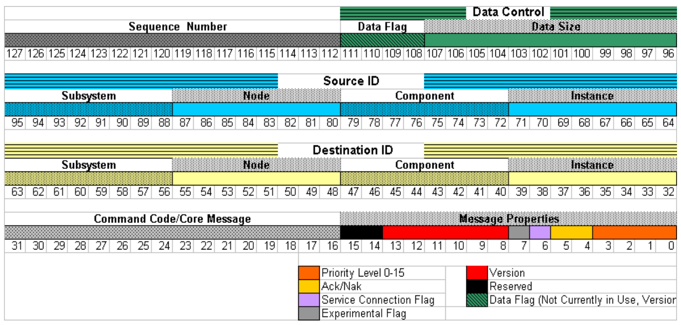


Figure 7: JAUS message header detailed structure [6]

III.JAUS & IGVC

A.Learning Process

After the 2006 IGVC competition JAUS research was started in order to participate in the 2008 JAUS challenge. Since the team did not participate in the 2006 JAUS challenge we had to start from the beginning. Reading through the reference architecture and doing some addition research on line was the beginning of the JAUS process.

After reading through the reference architecture and looking though the header structure coding was started to separate the difference parts of the structure. In order to have the Joint Architecture for Unmanned Systems ready for competition I need to program three different commands to pass the challenge, implement the LabVIEW code into our autonomous vehicle, and be able to receive the commands through wireless communication and have the commands carried out.

B.Implementation

With this implementation on the Wunderbot 4 can read and execute the JAUS message commands from the operator control unit through the 802.11g data link. During the JAUS challenge our system will be set to monitor for JAUS messages and check for incoming JAUS commands. At this level of implementation these messages will start the unmanned system moving forward in autonomous mode, stop the unmanned system from moving in autonomous mode, and activate a warning device (sound file output to speaker).

C.Challenges Encountered

When this project was first started i did not have any test software except what i had written to test how the header was being separated and i did not know LabVIEW that well. Reading though all the documentation keep the project at a standstill until i could figure out how to decipher what i needed from it.

There were various coding issues that were encountered during this project with the way that the header file was being split up and deciphered. Until the JAUS Compliance Suite was received i had no way to test our software to make sure if it was ready for competition. i could not getting the compliance suite to work for a while until i received an email about a bug in the way I installed it.

IV.JAUS on Wunderbot 4

The router on the Wunderbot 4 is programmed to only allow messages from the JAUS OCU IP address (192.168.128.1) and the JAUS router on the Wunderbot is set to the IP address of 192.168.128.253. The router on the Wunderbot also checks for the JAUS port (3794) and does not except messages from any other port and the frequency channel is set to 802.11g.

In order to change data on the router you must plug the Ethernet cable into the router then to your computer and open the Network Connections window, once in there right click on the Local Area Connection and click on Properties. Then scroll down to Internet Protocol (TCP/IP) and click the Properties button.

When the window appears click the Use the following IP address and enter the IP address (192.168.128.2) and Subnet Mask (255.255.255.0) and then open an Internet Explorer window. Type the IP address (192.168.128.253) into the address bar and then a GUI will show up where you enter the user name (*Admin*), password (*admin*), and check the “I agree to the terms and conditions below” then click the sign in button. This should not need done unless a new router is obtained.

In the JAUS software I first open the UDP Ethernet connection to listen for JAUS messages coming in through that port and IP address. When a message is received I check for the UDP header information that is eight bytes containing ASCII equivalent of “JAUS01.0” then parse that out of the UDP header.

I then double check to make sure the incoming IP address and port are correct for me to be able to receive data and carry out the commands. Next in my code I parse out the message properties and output them to the front panel of my LabVIEW code. The next piece of information that is in the JAUS header is the command code followed by the destination id and the source id and the data control and sequence number. For the competition there is no data control information being sent or sequence number.

After the command code is parsed out of the byte array I send it to the JAUS Command VI to carry out the command. There will be a new portion of code to accommodate the new part of the challenge where the destination id will be parsed out and need to be sent to a new set of commands to get information from the GPS. Screen shots of router setup and code are in Appendix A.

V.Social Impact

With the development of unmanned/autonomous vehicles and technology there have been many social and ethical impacts that we have had to face. Should robots and unmanned systems do our thinking and decision making for us removing the human error or should we keep the human in the equation.

Also the interoperability of these systems has an impact to us. If an unmanned/autonomous search team is sent out to find you lost in the woods would not you want them to be able to communicate no matter what platform they are built on so they could work together to find you faster.

These unmanned systems reduce exposure of humans to harmful environments, preform tasks not possible for humans and provide cost effective solutions to repetitive tasks. A large number of unmanned system products are being introduced to the market and many of these systems are characterized as task dependent and non-interoperable.

That is where JAUS comes into play because it makes these non-interoperable systems interoperable no matter the platform and it also supports the rapid and cost-effective development of unmanned systems.

Appendix

A.Appendix A: Router Setup Screen shots

B.Appendix B: Software Screen shots

C.Appendix C: JAUS Documentation

References

[1] Institute of Electrical and Electronics Engineers. *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries.* New York, NY: 1990.

[2] Pedersen, Jorgen. A Practical View and Future Look at JAUS. May 2006.

[3] The Joint Architecture for Unmanned Systems, Compliance Specification (CS), Version 1.2, 25 October 2006.

[4] The Joint Architecture for Unmanned Systems, Domain Model (DM), Volume I, Version 3.2, 10 March 2005.

[5] The Joint Architecture for Unmanned Systems, Reference Architecture Specification (RA), Volume II, Part 1 Architecture Framework, Version 3.3, 27 June 2007.

[6] The Joint Architecture for Unmanned Systems, Reference Architecture Specification (RA), Volume II, Part 2 Message Definition, Version 3.3, 27 June 2007.

[7] The Joint Architecture for Unmanned Systems, Reference Architecture Specification (RA), Volume II, Part 3 Architecture Framework, Version 3.3, 27 June 2007.