Generating a ROS/JAUS Bridge for an Autonomous Ground Vehicle

Patrick Morley
The University of Akron
pjm39@zips.uakron.edu

Alex Warren
University of Arizona
amwarren@email.arizona.edu

Ethan Rabb
Washington University in St. Louis
ethanrabb@gmail.com

Sean Whitsitt
University of Arizona
whitsitt@email.arizona.edu

Matt Bunting
University of Arizona
mosfet@email.arizona.edu

Jonathan Sprinkle
University of Arizona
sprinkle@ece.arizona.edu

National Science Foundation

In This Talk I Will Show

- A DSM solution for a ROS-JAUS bridge
  - Enables versatility and productivity
  - Allows developers to incorporate ROS-based sensors and actuators into JAUS-compliant systems
  - Yields correct-by-construction code

Cognitive and Autonomous Testing Vehicle

Autonomous Car Controlled by Embedded Controllers and a Master Computer Interacting on the OpenJAUS Platform

CAT Vehicle

- Modified Ford Escape Hybrid for autonomous driving
- JAUS–compliant system
- Communicates with on-board and external machines using TCP/IP
- Embedded systems only communicate with JAUS

Joint Architecture for Unmanned Systems (JAUS)

- Standard for autonomous systems in the defense industry
- Several components function under a *Node Manager*
- We used an implementation of the JAUS standard known as OpenJAUS

What Problem Did You Solve?

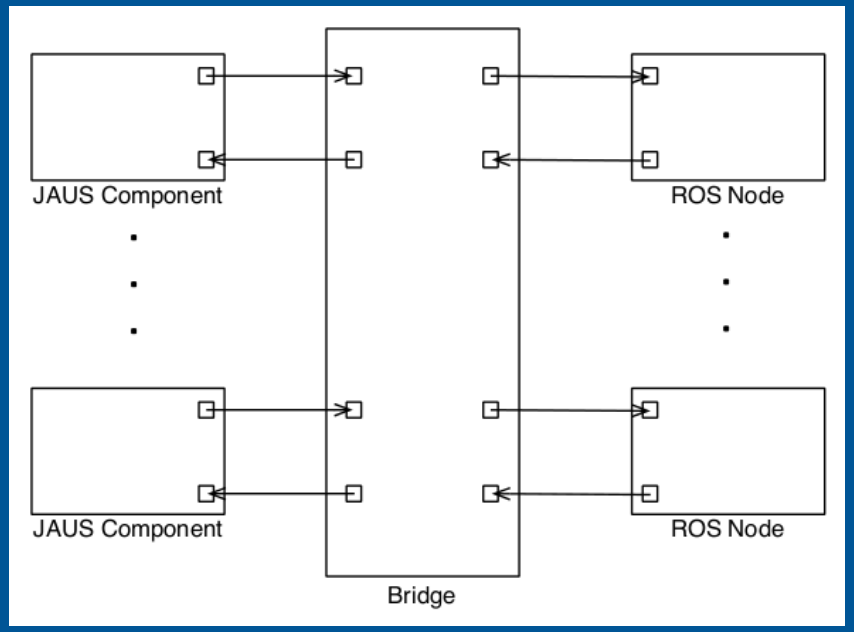
- The CAT Vehicle is a JAUS-compliant system
- Adding functionality to the vehicle is troublesome
  - Sensors and actuators do not come with JAUS drivers
  - A new driver must be written for each new sensor
- Open source operating systems such as ROS are easier to use
- For this reason, we developed a ROS/JAUS bridge to exchange commands and data to enable rapid prototyping

Who Cares?

- Bridging ROS and JAUS allows for researchers to focus on their work without wasting time with hardware compatibility
- Open source options are generally preferred by programmers
- Open source drivers can (and have been) viewed by several people
  - More likely to be safe
  - More likely to be robust

How Was This Problem Solved?

- A hybrid JAUS-ROS node is created. This is a JAUS-compliant node that includes the basic ROS libraries and can communicate with *roscore*.
- The hybrid node acts as a bridge and translates messages to and from JAUS components and ROS nodes.
- A textual modeling language is used to generate bridge nodes for future use.



So, What's the Big Deal?

- ROS to JAUS connections have been implemented before, but these connections rely heavily on human customization
- Our solution requires minimal details to generate a full bridge

```
struct [
      type name
]
array_type [s] name

MessageName {
      JAUS jausname (
            type membername
            type secondmembername
      )
      ROS rosname (
            type membername2
            type secondmembername2
      )
      JAUS->ROS (
            membername -> membername2
            secondmembername ->
secondmembername2

      ...
```

Innovation

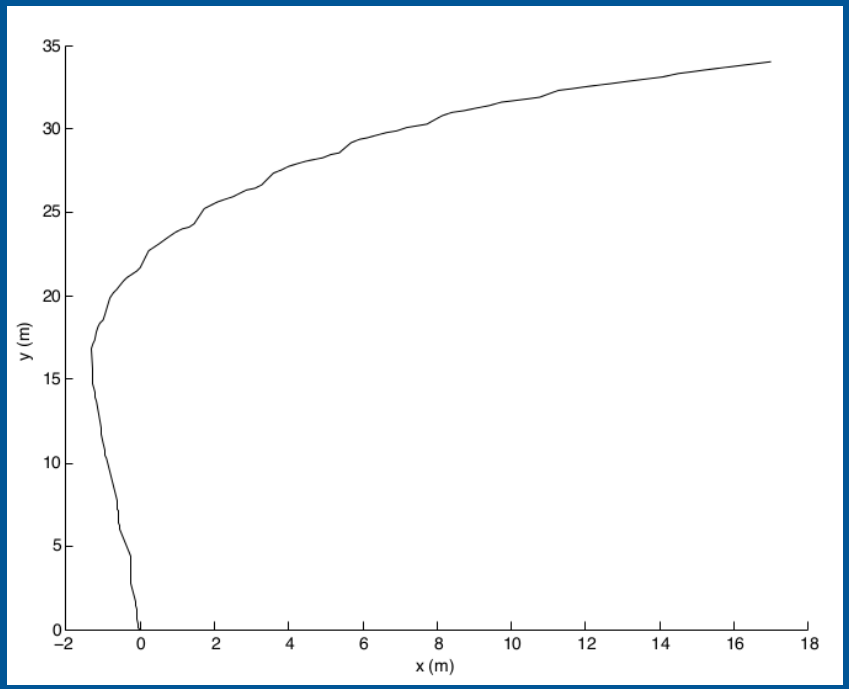
- Using this DSM solution, we were able to offload the car's steering to a ROS node. Passengers and spectators were unable to notice a difference in the steering mechanic.

Innovation

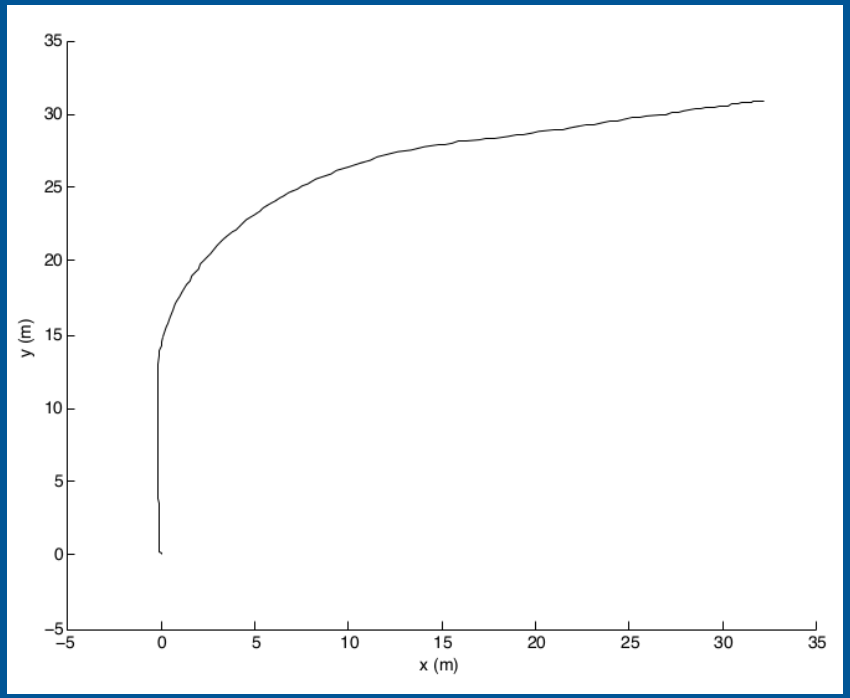
- With this DSM solution, any OpenJAUS system is now capable of using any component designed for usage with open-source robotics
- This modeling language generates the core components needed for ROS/JAUS interaction
- This provides a building block for large-scale communications between the two operating systems

How Do I Know This Solution Works?

Steering Computations on ROS



Steering Computations on JAUS



How Do I Know This Solution Works?

Summary

- We demonstrated an effective bridge between ROS and JAUS that allows exchange of commands and data
- This bridge allows all systems in OpenJAUS to communicate with all ROS nodes for passing data or commands
- The DSM language can be quickly imported to a graphical environment, such as *GME*

Summary

- JAUS-compliant systems can now have their simulation results externally verified by open-source platforms

- Mapping and computational procedures (such as *G-Mapping*) can be performed on a separate machine to reduce overhead on real-time systems