Project Summary

Intelligent robots have not quite made it to the front lines of security and reconnaissance. A more prevalent approach is to remotely tele-operate a robot from a safe distance to investigate and, if needed, disarm a threat. However as research continues in intelligent algorithms, we expect that operators will become a peer of the robot, allowing for a higher robot to human ratio. Unfortunately as algorithms become more intelligent, the practice of operators making in field hardware modifications to a robot will become an issue. Updates to the hardware will need to be reflected in intelligent algorithms either autonomously or by highly-skilled operators.

This proposal is about defining a specification that completely describes robot hardware devices in a way that enables three goals: 1) provide enough information for simulation and visualization of hardware and controllers, 2) declaratively specify the mechanism (both syntax and semantics) for requesting data and actuation, and 3) inform users of standard message types that can be obtained from the hardware to facilitate connection to existing frameworks. RDIS (Robot Device Interface Specification) is a domain specific language that currently supports the syntax and semantics of differential drive robots and automates the creation of ROS drivers and command line programs. We will expand RDIS to accommodate a larger set of kinematic designs, sensors, execution semantics and enabled tools and frameworks to move towards a specification that can be used to communicate hardware configuration changes to intelligent algorithms or operators.

Intellectual Merit. The intellectual merit of the proposed work centers on creating a domain-specific language to leverage the services provided on hardware devices by embedded firmware. RDIS provides a mechanism for centralizing the knowledge of the syntax (how to access the service) and semantics (what is the meaning of the input/output of the service) to the embedded system itself through a fully descriptive, declarative language. This aspect of the proposed project is transformative: it could potentially affect the creation of new hardware platforms, reduce the complexity of device drivers (thus encourage new devices) and lower the learning curve to working with platforms. To accomplish a specification that is generalizable, this project will expand a preliminary domain model that includes base concepts (linkages, joints and transmission) and account for simulation parameters (collision, appearance, dynamics, etc) to a more general set of popular platforms and kinematic linkages. Although some frameworks use declarative specifications for hardware, none are complete or meet the needs of both simulation and capturing the interface syntax and semantics. It is expected that manufacturers will choose RDIS to enable their devices once there are more RDIS-enabled environments available. RDIS also has applications to reconfigurable hardware platforms. Research methods that learn the current configuration based on embedded sensors would use the RDIS to communicate changes to physical configuration to support in-field modifications.

Broader Impacts. The broader impacts of the proposed work can be categorized as either enabling more efficient controller design or enabling new research. First, RDIS will allow for discovery of hardware services without needing to create point solutions that map each device to each framework. Second, the creation of an RDIS file for a custom platform will ease integration into existing tools and frameworks. Third, RDIS is the mechanism for allowing platforms to communicate hardware changes, allowing for the operator or intelligent algorithms to accommodate that change. These results will be disseminated not only through high-quality publication venues but this project also enables collaboration with industry partners with similar concerns (NASA and RTP-Huntsville AL). This project will train researchers adept at both software engineering and robot device interfaces as well as incorporate undergraduate researchers through REUs and paid research positions.

Keywords: Domain Specific languages, Robotics, Robot Architectures and Frameworks

Summary Technical Details of Breakthrough Project

RDIS is the main innovation within the proposed research. Technical choices about what and how RDIS will evolve will be based on the analysis of current software tools and hardware platforms as well as in conjunction with stakeholders (framework developers and platform/device manufacturers). RDIS will be supported as an embedded or externally obtainable hardware description language that meets requirements of both manufacturers (concise, complete) and framework developers (sufficiently expressive and generalizable without redundancy).

RDIS is currently based on the declarative language JSON, which is attractive to manufacturers unwilling to work with proprietary standards. Currently JSON-parsers, lexers and interpreters are built based on industry standard tools (ANTLR, StringTemplate). However, it is expected that interpreters will be written in an appropriate language and toolset for the wide-range of support needed (IDE plug ins, etc). In this project, we will further improve the JSON implementation to support the complete set of features enumerated thereafter. We will also investigate a new domain-specific language that sits on top of JSON, that is optimally fit and maximally constrain domain-experts to build RDIS specifications. A model transformation (compilation) to JSON will ensure a full integration.

RDIS supports seven concepts, each of which will be expanded to support additional platforms. Also new concepts and language constructs will be added to support development and simulation tools and runtime execution.

1. Connections will be expands to include bus, memory and port-based transports as well as more of the standard transports (TCP/IP, Bluetooth, I2C).

2. Also better granularity of thread control will be introduced including multithreaded models that allow different frequencies for request/reply or publish/subscribe services.

3. State variables currently support two modes: constants and asynchronous primitives. Scripting will updated to support data translation between primitives and state variables.

4. Primitives and message formats will be modified as needed to support new platforms. However, it is thought that the current format string design is sufficient for formatting both encoded and raw byte strings. Primitives will be modified to work with the expanded role of state variables.

5. Interfaces will also take advantage of the new embedded data translation scripting language.

6. Domain concepts will be expanded to better generalize actuators by incorporating a link/joint model with a novel transmission component that links to primitives. Also generalized models of sensors (odometry, GPS or landmark identification, etc) will be added to complete the specification enough to support most common mobile sensor suites. We will add a complete visual and collision model as well as relevant dynamics suitable for use in existing simulation environments.

7. Error handling semantics within the runtime environment will be added to give manufacturers a default approach to handling errors and to customize error handling based on the transport or runtime environment.

8. Management of sensor and actuator error models consistent with existing frameworks will allow for manufacturer provided error models to be propagated to the framework or controller. An example would be a transformation of encoder error to pose error. Although all sources of error (systematic and non-systematic) cannot be accounted for in this approach, it is currently better than existing approaches that abstract out specific device errors.