An Open Source Network Protocol for Mobile "Internet of Things"

Christopher Ruediger and Dr. Rance Necaise,
Department of Computer Science, Randolph–Macon College, Ashland, VA

RANDOLPH-MACON COLLEGE

Problem and Motivation

The goal of this research project is to develop an open source network protocol that can be used to control, access, and manage robots and a collection of Internet of Things (IoT) devices that are deployed on a mobile platform. This protocol will not only provide for the communications between a robot containing a variety of IoT sensors and a controller, but it will also allow multiple users anywhere on the Internet to view the onboard camera. With the availability of an open source protocol, individuals and hobbyist will be able to easily construct, deploy, and control robots containing a variety of sensors and tools similar to those used in law enforcement, fire fighting, and the military for tasks that are too dangerous for humans.

Background

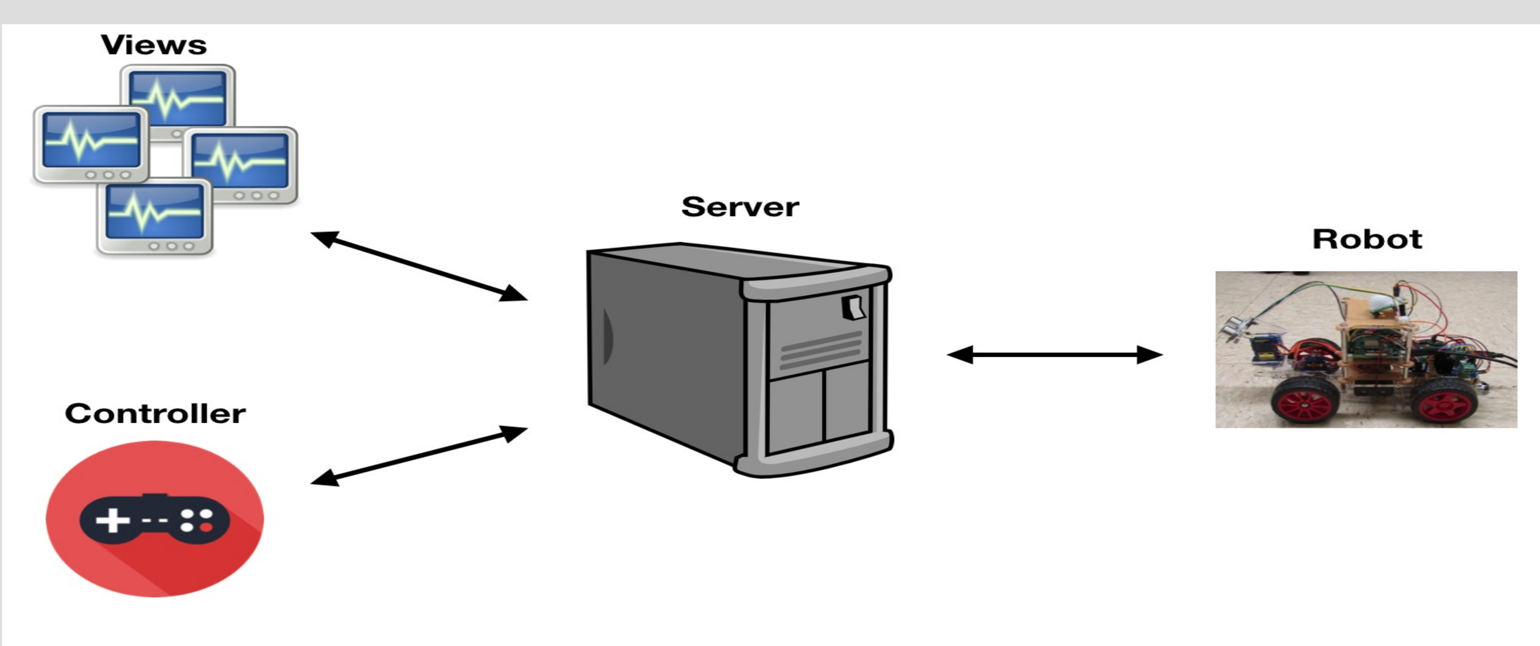
There are many occupations that require humans to put themselves in situations where they remain vulnerable to dangerous environments. Some of these occupations include firefighters, members of the military, and the police force. Today technology has advanced to a point at which robots have been designed to negate the risk of fatality in these dangerous circumstances. By replacing the humans with robots, the only risk is loss of hardware.

Within law enforcement, these robots are used to investigate bomb threats, to deescalate hostage situations, and to find suspects without putting officers in harm's way. With a human on the controls the tasks are still daunting, and the robots must be equipped with the necessary devices and sensors to be successful. An example is a robot that is equipped with front and rear cameras, infrared light sensors, and a microphone in order to locate criminals without putting officers in harm's way [7]. Military uses include robots for disarming and disposing explosive ordinance, also known as bomb robots. The user can manipulate the robot from a safe distance and use a series of cameras to utilize functions on the robot, like motorized arms and water jets to disable the ordinance. These robots have the capability to attach different tools for a diverse range of tasks. For example wire cutters would be needed to traverse through a wire fence [1]. Firefighters have the Thermite Robot that comes equipped with infrared and standard cameras to transmit images to the operator, and a water hose to fight the fires [2].

The Internet of Things (IoT) is a phrase that 87% of people haven't heard, but is considered the third wave of the information technology industry [5]. IoT is the idea of connected devices and sensors, which collect and transmit data over the Internet, supported by a wide network infrastructure. Components that are part of IoT are programmed as the client part of a Client/Server network system, each with their own communications protocol. The devices that are considered to be on the IoT all have Internet capability and most of the devices today rely on Wi-Fi for connection. Such devices could include IP cameras to monitor a space, or temperature and vapor sensors used to monitor its immediate environment.

Robots can be considered a mobile platform to house IoT devices such as their cameras, infrared sensors, and motored extensions. The robot itself now becomes an IoT device, with computers providing a hub for sensor monitoring, motor control, and allowing the devices to be Internet accessible. Today, most IoT devices, including robots, use proprietary protocols to facilitate the data communications between the device and a network hub [6]. These proprietary protocols prevent us from extending or modifying an IoT device to provide additional capabilities or controls for adapting their use to special applications. With the use of small Raspberry Pi's and a large collection of sensors, new IoT devices are easily constructed and integrated into an existing system that uses an open source protocol.

System

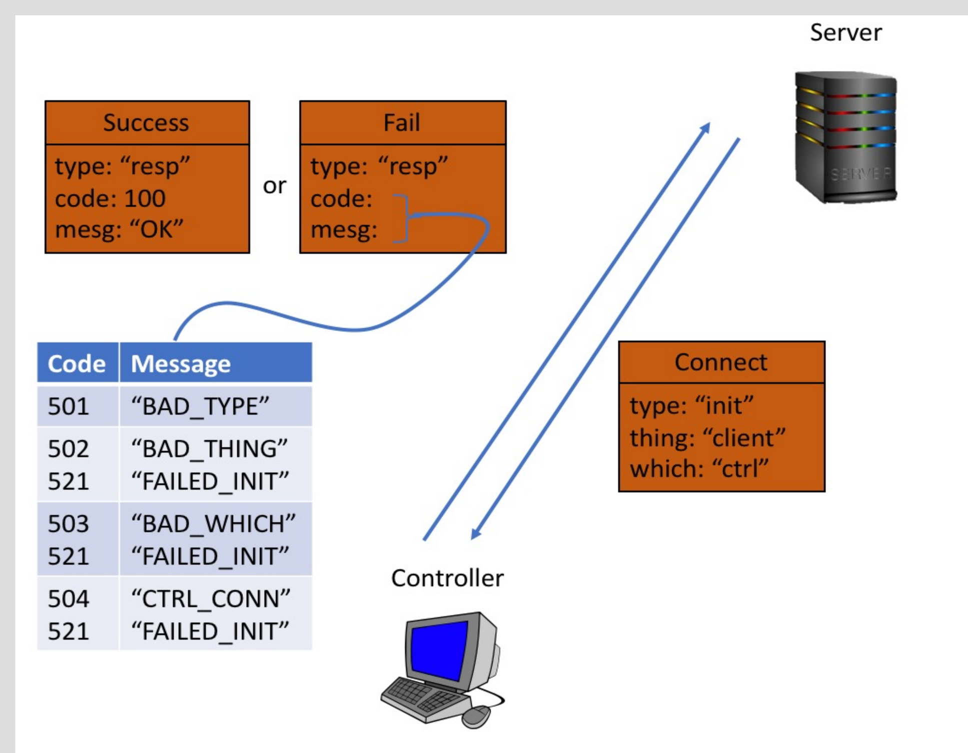


Protocol

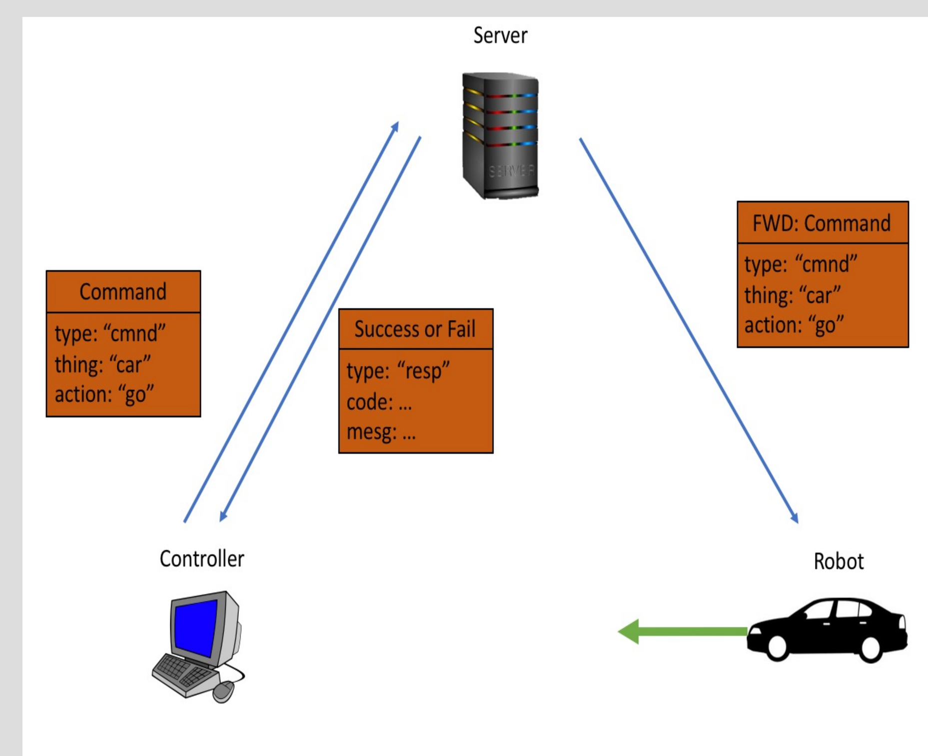
Outline of protocol packets

Used by:	Ctrl, View, Robot	Controller	Server, Robot	Server, Robot
Packet:	Initialization Packet	Command Packet	Response Packet	Alert Packet
Fulfilled:	*type:* "init" *thing:* "client" *which:* "ctrl", "view"	*type:* "cmnd" *thing:* "car", "cam", etc. *which:* some action *param*: ... ... - Optional	*type:* "init" *code:* "client" *msg:* "ctrl" *thing*: ... *state:* ... {} ... - Optional	*type:* "alert" *thing:* "robot" *Info:* ... {} ... - Optional
	type: "init" *thing:* "robot" *devices:* ["car", "cam", "motion"]			

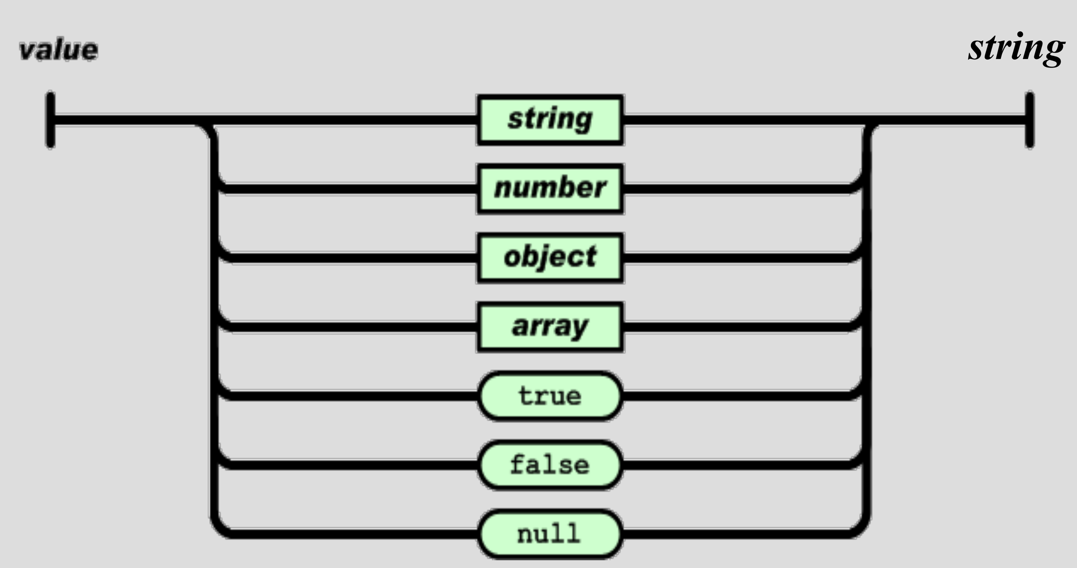
Sample: Initialization Sequence of Communications -



Sample: Command Sequence of Communications -



(JSON)- JavaScript Object Notation



Our protocol consists of a defined structure of organization for packet data, and the use of the JavaScript Object Notation in order to successfully communicate data within a given system.

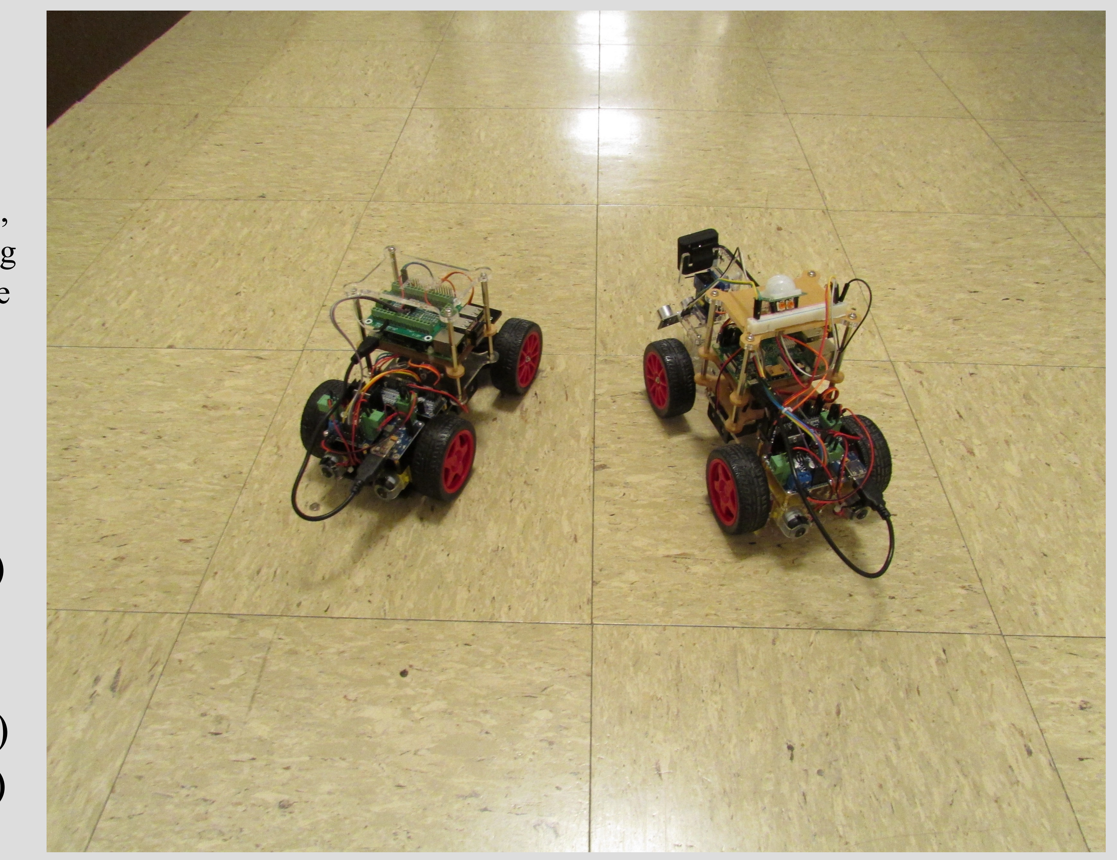
Response Codes

Code	Message
100	"OK"
500-509 501 502 503 504	***Unsupported*** "BAD_TYPE" "BAD_THING" "BAD_ACTION" "BAD_WHICH"
510-519 511 512 513	***Peripheral Already Connected*** "CTRL_CONN" "TO_MANY_VIEWS" "ROBO_CONN"
521	"FAILED_INIT" (failed initialization)
522	"NO_ROBO" (no robot connected)

Practical Implementation

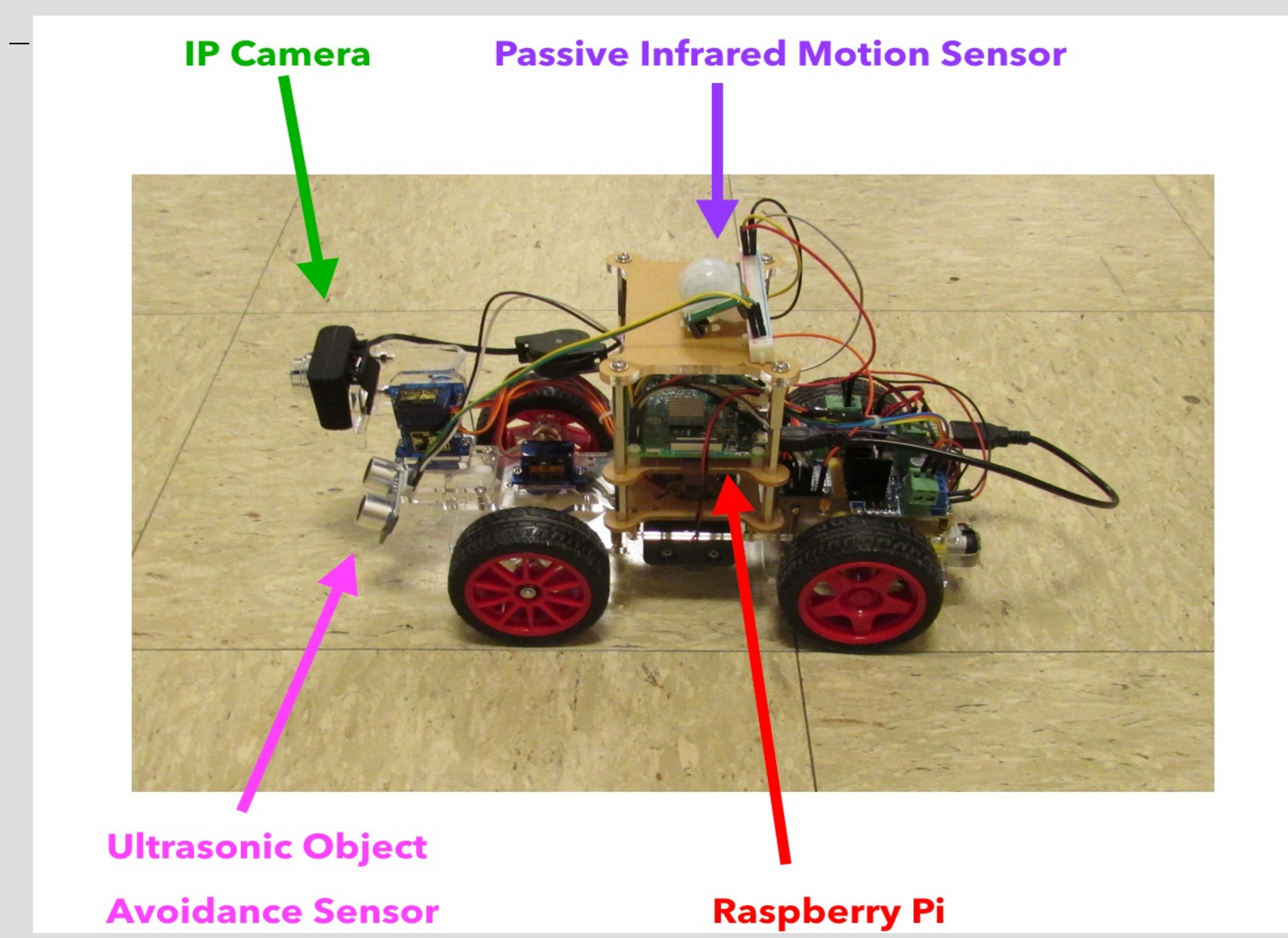
Specifically for our robot, we have used a raspberry pi to support an ultrasonic (object avoidance sensor), a PIR (motion sensor), and IP Camera. The ultrasonic device can give data regarding distance to objects and help the operator avoid damaging the robot. The Infrared sensor can be armed so that detected motion alerts the server. The IP Camera is our eyes on the ground if you will. The controller is the component which controls the robots movements and onboard sensors along with their supported functionality. The views do not have any control capabilities, however they do allow for other users to see state data and the video feed coming from the robot. The server, as I mentioned before, will direct the data traffic for communications between the systems components, which is accomplished by keeping track of pertinent information like role assignments and state data. The server provides most of the implementation for the protocol.

Server:	Linux (fedora)
Robot:	Raspberry Pi (Raspbian)
Client:	Linux (fedora)
Viewer:	Linux (fedora)



Robot Generation 1 and Generation 2

–



Robot and Labelled Sensors

Acknowledgements

Dr. Rance Necaise - Dr. John McManus -

Schapiro Undergraduate Research Fellowship -

Randolph Macon College: Computer Science Department