INTEGRATING A MOBILE ROBOT WITH THE INTERNET OF THINGS

Samuel J. Barr
Appalachian State University
Department of Computer Science

December 20, 2019

Contents

1 Introduction

The Internet of Things (IoT) is a system where numerous addressable 'things' communicate with each other and over the internet using traditional or specialized network protocols [6]. Robotics have played a key role in our society with various applications such as industrial robots, military robots, home assistance robots, and many more [4]. Together, IoT and Robotics have formed a new research area called 'The Internet of Robotic Things' [8]. IoT's ability sense its environment and communicate with other things, along with the innate ability of robots to manipulate environments, allow for a system that can sense and manipulate its environment with much more accuracy and efficiency than traditional robotics.

This project focuses on using IoT infrastructure to visualize various sensors on a mobile robot on an online dashboard using a Raspberry Pi [2], Node-RED [1], and Thingsboard [3].

2 Background

2.1 VEX EDR Robotics Platform

This is a robotics education platform designed to help students learn how to engineer and program robots. Though it was designed for beginners, the system allows for third-party components and software to be loaded on for more experienced users.

The robotic system I have built this project is shown in Figure 1. The robot is driven by four independent motors with attached optical shaft encoders. The shaft encoders produce a value based upon how far a drive shaft has turned. These values are used to calculate the distance the robot has traveled and an angular position relative to a starting position. The system is also equipped with a LiDAR which produces a 360-point vector of distances to the objects closest to it.

The robot is programmed using the Purdue Robotics Operating System (PROS). This is a C/C++ API built for the VEX system by students at Purdue University.

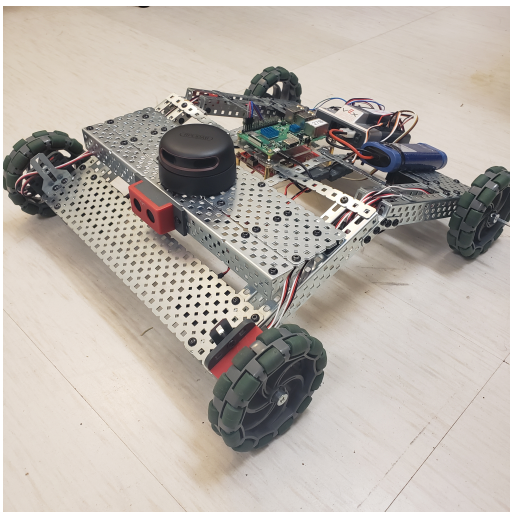

Figure 1: Robot

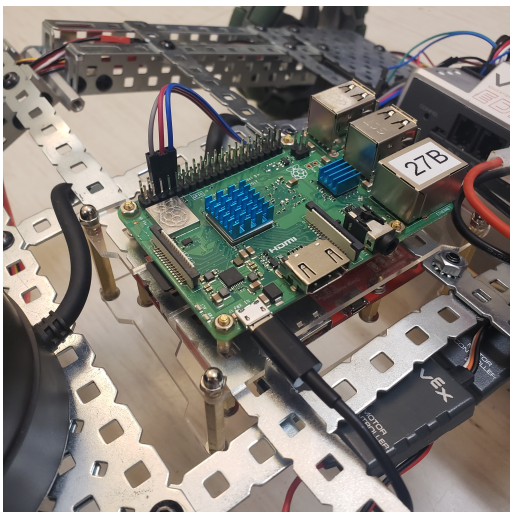

Figure 2: Raspberry Pi

2.2 Raspberry Pi

The Raspberry Pi is a cost-effective system-on-a-chip that is around the size of a credit card. Figure 2 shows how the Pi is integrated with the robot. This system uses a Raspberry Pi 3 B+ running the Raspbian Buster operating system. The Pi was chosen for its WiFi capabilities and its ability to use other programs and programming languages.

2.3 Node-RED

Node-RED is a programming tool that is used to connect hardware, APIs, and internet services together. It was built using Node.js, an event-based non-blocking system that makes it ideal for use on limited hardware such as Raspberry Pi's. Node-RED also comes with various nodes to manipulate, format, and publish data to online services using MQ Telemetry Transport (MQTT), a popular and lightweight transport layer protocol used for various IoT applications.

2.4 Thingsboard

Thingsboard is an open-source platform designed for data collection, manipulation, and visualization. This was chosen for ease-of-use, the ability to accept MQTT messages, and its library of well-designed widgets to create easy-to-read dashboards.

3 Related Works

3.1 IoRT for Warehouse Logistics

Karnouskos et al. created an IoRT system involving a mobile claw robot and a stationary claw arm, and a gesture control armband as an educational project and proof-of-concept for IoRT enabled warehouse robots [7]. Their system involved a series of multi-colored balls in an environment. The stationary claw arm was placed on the edge of the environment in reach of bins of the colored balls. A camera was placed above the environment facing down, and the mobile robot would roam the environment. First, a user would use a dashboard to select a layout of how they want the balls arranged in the environment. The stationary arm would pick up the colored balls from the bins and throw them into the environment, then the mobile robot will move them to the correct positions. The mobile robot could also clear the field by moving the balls to the stationary arm to be organized back into the bins. While this process is done autonomously, the gesture control armband allowed a user to manually control the stationary arm in case any balls are missed. The Figure 3 details how their architecture was designed.

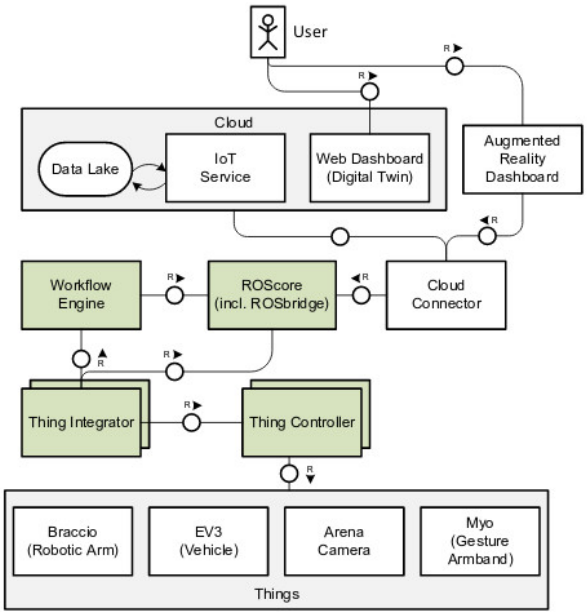

Figure 3: Architecture Design [7]

The middle-ware for this system was achieved using the Robotics Operating System (ROS). ROS is open-source, and has many implementations for various robotic things and modules that make it well-suited for any prototyping need. Each process in ROS is considered a 'ROSnode', where as 'ROScore' acts as a traffic controller for ROSnodes in a system. Researchers used the *Thing Controller* to handle low-level device control and the *Thing Integrator* to broadcast and receive messages from those physical devices. The *Workflow Engine* publishes tasks to be accomplished by the system, and monitors the system as a whole. To integrate with the cloud, researchers used *ROSbridge*, which is a JSON API to the ROS system. From there, *Cloud Connector* interfaces with ROSbridge to get data from ROScore and format it to MQTT protocol messages. Once this data was on the cloud, it could be analyzed, processed, and displayed to the user via dashboard.

3.2 Dashboard for a Robotic Platform

In [9], Yousif presents an IoRT platform for a remote control car for an academic project. The robot ran on ROS and generated data from an on-board gyroscope module while the platform handled storage, processing, and visualization of that data. This section will go over a summary of the technologies used and how they work. Figure 4 below shows how the system was designed.

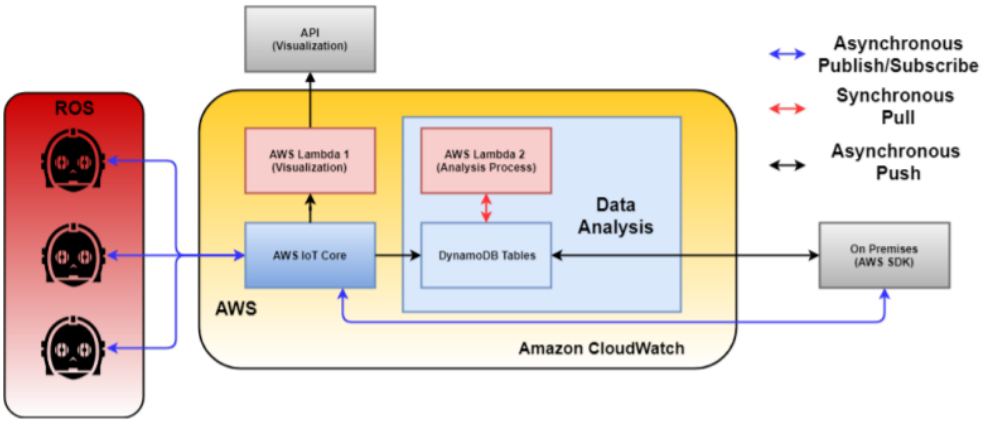

Figure 4: Architecture Design [9]

3.2.1 Robot Operating System

ROS operates in a publish-subscribe messaging pattern. Nodes in a ROS network share their information with other nodes using defined topics. In every ROS system, there exists a master node which initiates publish/subscribe communication. This allows other nodes to advertise the topics they are publishing or subscribing to. ROSbridge was used for this project to convert these messages to JSON format. The JSON payload is then sent through an MQTT-based module named MQTTbridge, which is built on top of ROSbridge. AWS IoT Device SDK was then used to establish a connection to the AWS IoT Core. Figure 5 shows how a message made its way through the system.

3.2.2 AWS IoT Core

The AWS IoT Core is a serverless cloud platform which manages secure bidirectional communication between devices. This platform uses the MQTT protocol, which supports the publish/subscribe messaging pattern. The platform also supports MQTT over WebSockets, which allows any service using WebSockets to connect to it.

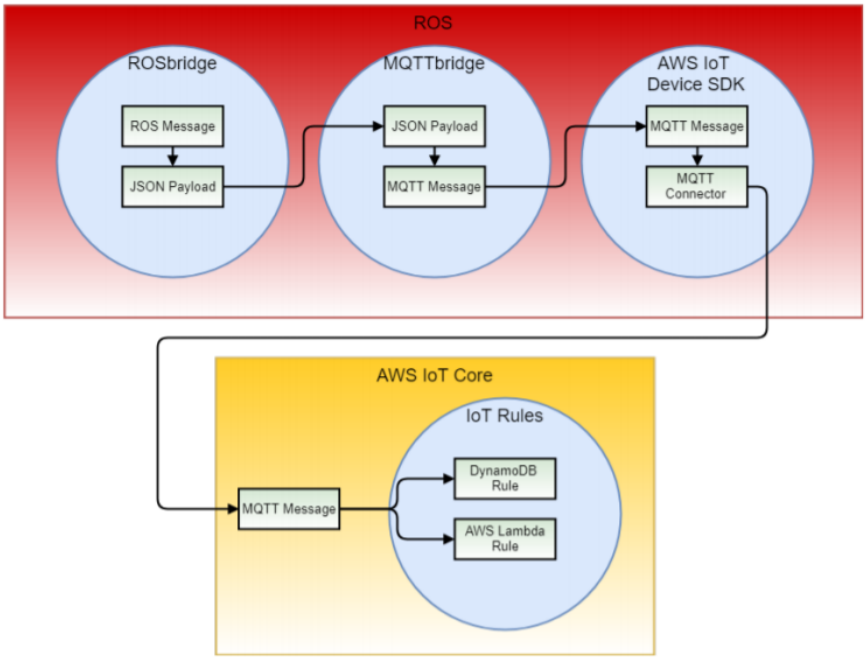

Figure 5: ROS message to AWS [9]

3.2.3 AWS IoT Rules

Rules are APIs for the IoT core which enable topics to interact with other AWS services. Yousif used these rules to send asynchronous messages of any topic to a Dynamo database or to invoke a lambda function.

3.2.4 Data Analysis

Once the data was in the cloud, Yousif performed 'data mining' on the data he received. Data mining is a multi-step process which involves removing inconsistent data, combining data sources, extracting/aggregating relevant data, and discover patterns in that data. Once the data was 'mined', Yousif used Microsoft Power BI to visualize the data.

4 Project

Figure 6 shows the architecture of the system. The remainder of this report will focus on how data moves from the robot to Thingsboard.

4.1 Robot

The data that is sent from the robot to Thingsboard are lidar data, angular position, distance traveled, and battery level. The robot is directed by the Vex *Cortex* microcontroller. A Raspberry Pi with a dedicated battery sits on top of the robot where both the Pi and the Cortex are connected by a wired UART serial connection. To transfer information between the two, I have developed a simple connectionless protocol. The robot will first send a start byte, then a defined "sensor id" byte. The sensor id is unique to each sensor/data the robot is sending. The robot will then send the payload and finally a stop byte to signal the end of the transmission.

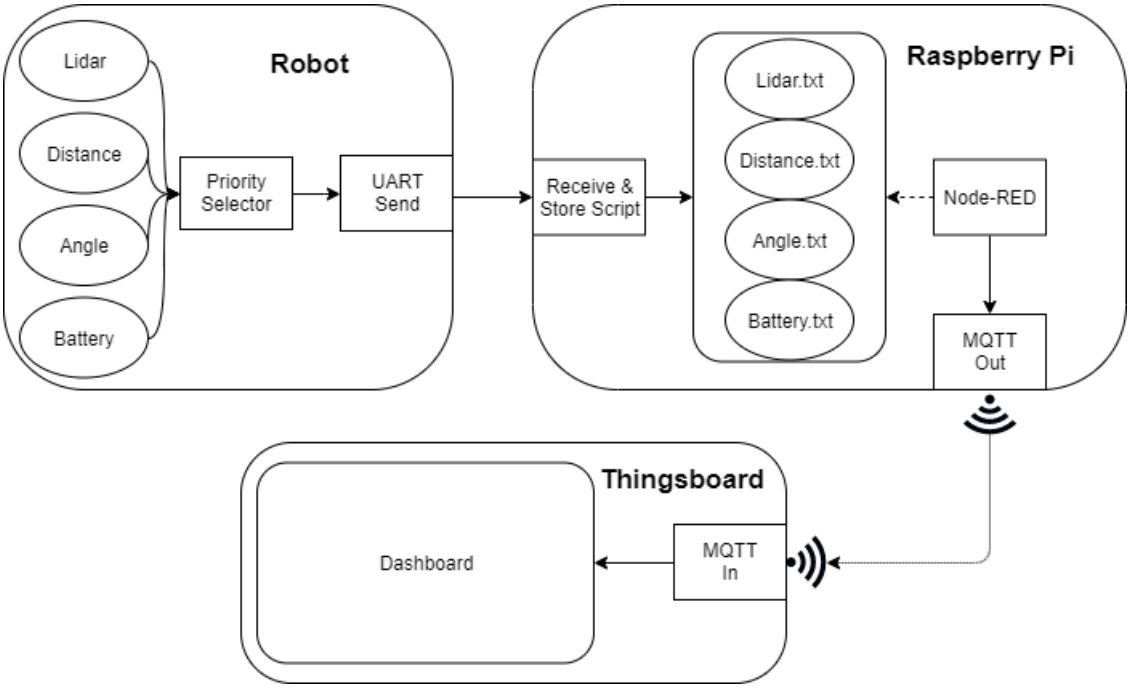

Figure 6: Project Design

Blasting the UART port with continuous data can result in dropped packets, and since sensor data can update at different intervals, a priority system was devised to send data at predetermined intervals. Existing in a separate thread on the Cortex, each sensor/data type has a priority value. That value corresponds to a millisecond value that dictates how long to wait in-between data transfers. For example, since battery level takes a considerable amount of time to update, it has the lowest priority, taking sixty seconds in between transfers. On the inverse, since lidar data updates quickly, it has the highest priority, transferring every second. The Cortex only sends eight lidar distance points at 0, 45, 90, 135, 180, 225, 270, and 315 degrees. This prevents flooding the UART port with too much data while also giving enough information to provide a general understanding of the robot's surroundings.

4.2 Raspberry Pi

A python script executes indefinitely when the Pi is powered on. This script takes in data sent from the robot, organizes that data, and saves it to labeled files. The script contains objects for each sensor/data type. When the robot sends a sensor id, the Pi will read it in, reference the specified object, and set its field to the received payload.

Saving the sensor/data objects is done en masse in a separate python thread. Every three seconds, the thread will save all updated objects into respective named text files in a designated directory. If a sensor object has not been updated since the last save, the script will ignore it. Since Node-RED is designed to trigger when a file updates, this is so that it will not trigger when there is no new data to report. Here is a link to my github where this code is hosted [5].

4.3 Node-RED

Figure 7 depicts the flow built for this project. The first node watches a directory for any changes. Once a file is changed, it reads in the contents of the file in the next node. When the script on the Pi updates a file, this sometimes causes Node-RED to trigger early and read in the file before the script has finished saving it. The *Payload Checker* mitigates this issue by verifying that the payload is not empty. The *Topic Director* then sends the file contents to the appropriate JSON format node based on the file name. Once the file contents are formatted, the MQTT Out node formats the JSON to MQTT and sends it to Thingsboard.

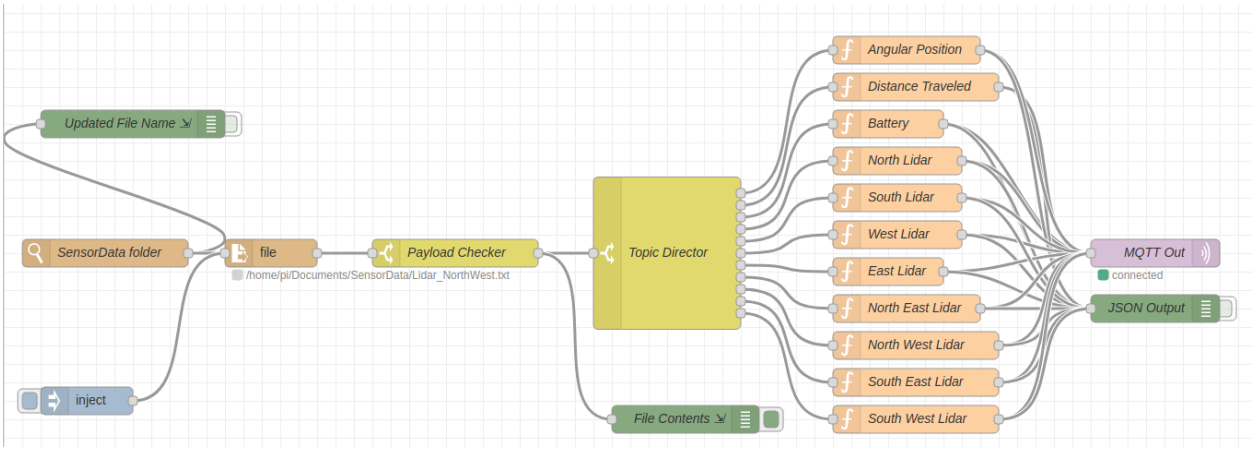

Figure 7: Node-RED Flow

4.4 Thingsboard

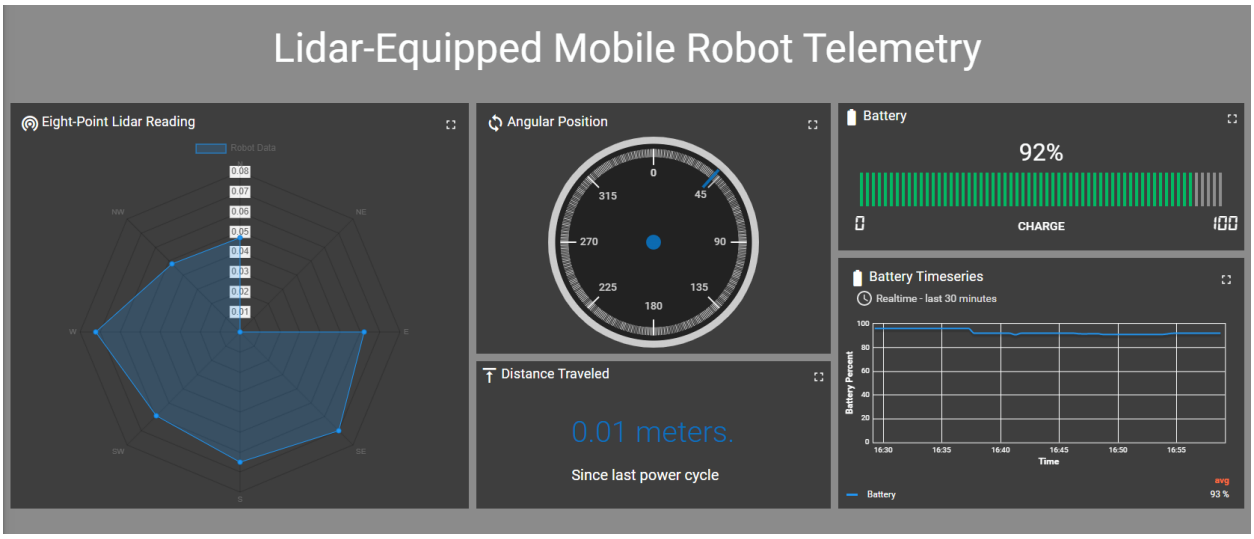

Figure 8: Thingsboard Dashboard

The dashboard contains a lidar reading that contains eight distances at major directional points relative to the robot. Figure 9 shows the environment the robot was in when the dashboard image was taken. A blue point represents the distance to the nearest object from the robot and the

shaded blue region in between those points represents open space. "Angular Position" starts at zero when the robot is powered on and represents the angular difference from the robot's start position. "Distance Traveled" displays how far the robot has traveled since it was powered on. Battery information is represented as a real time percentage bar and a time-series to display how much the battery has drained over a span of five minutes.

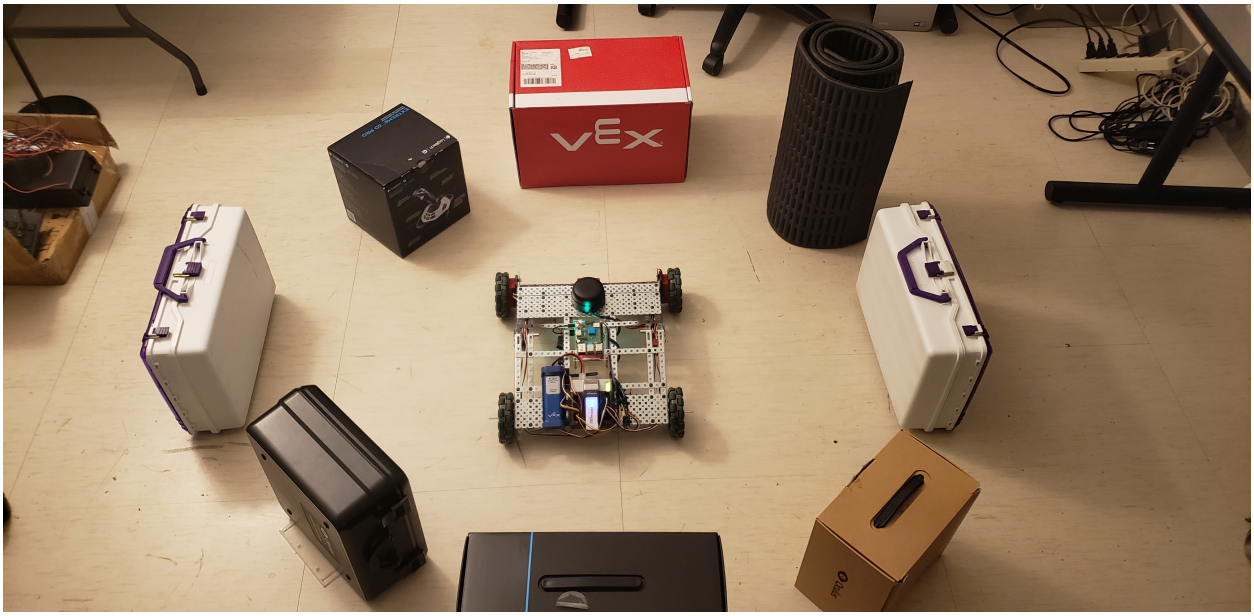

Figure 9: Robot Environment

Testing has shown to be quite responsive to any updates sent by the robot. The dashboard updates its widgets every second, which allows for some lidar points to update before the dashboard does. Some lidar points display as zero as shown for the North-East point in Figure 6. This is due to the lidar itself. If it obtains a bad reading for any distance, it marks that point as a zero. Tests have shown that no more than two points read as zero at a time and they are generally updated to a real value on the next dashboard update.

5 Conclusion

This project has served as a proof-of-concept for integrating a robot onto the IoT infrastructure. The only extra hardware needed is a Raspberry Pi with Wi-Fi capabilities to host Node-RED. It is possible to format data to MQTT on the robot and send it through a network medium without a Raspberry Pi, however, using a Pi cut down on development and allowed for faster completion of this project.

Future work on this project could entail sending data in the opposite direction, where user input on Thingsboard could manipulate the robot. This can be done using the current architecture, where Node-RED can accept data from Thingsboard, write it to a local file on the Pi, and the script would read it and send that request to the robot. Testing would have to be done, however, to determine the latency of the request and consider a more direct approach for time-sensitive applications. Thingsboard also features a "Rule Engine" in which rules can be created based on existing data to trigger alarms or actions for the robot to take. For example, many actions can be triggered based on the battery level. If the battery falls below a certain threshold, Thingsboard

could send a command to the robot and the Pi to slow transmission speeds. Transmission speeds could even be proportionate to the battery level as well. It could also affect how often the lidar scans the environment and how much power is delivered to the motors.

References

[1] Node-red - low-code programming for event-driven applications.

[2] Teach, learn, and make with raspberry pi – raspberry pi.

[3] Thingsboard - open-source iot platform.

[4] Afanasyev, I., Mazzara, M., Chakraborty, S., Zhuchkov, N., Maksatbek, A., Kassab, M., and Distefano, S. Towards the internet of robotic things: Analysis, architecture, components and challenges. *arXiv preprint arXiv:1907.03817* (2019).

[5] Barr, S. J. Iort project pi code, 2019.

[6] Initiative, I., et al. Towards a definition of the internet of things. *IEEE IoT Initiative white paper* (2015).

[7] Karnouskos, S., Gaertner, N., Verzano, N., Beck, F., Becker, A., Bianchino, S., Kuntze, D., Perez, M., Roy, R., Saelens, S., et al. Experiences in integrating internet of things and cloud services with the robot operating system. In *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)* (2017), IEEE, pp. 1084–1089.

[8] Simoens, P., Dragone, M., and Saffiotti, A. The internet of robotic things: A review of concept, added value and applications. *International Journal of Advanced Robotic Systems 15*, 1 (2018), 1729881418759424.

[9] Yousif, R. A practical approach of an internet of robotic things platform, 2018.