Object Recognition and Path Smoothing Robot

Jake McKenzie
jake314@uw.edu

Alex Marlow
alexmarlow117@gmail.com

Ammon Dodson
ammon0@uw.edu

Prepared: 12/16/2018 Rev: 0.4

Revision History		
V0.1	10/18/2018	Initial Specification
V0.2.0	11/01/2018	Additional information on ROS
V0.2.1	11/01/2018	Added ethical concerns
V0.2.2	11/03/2018	Modelled the overall system architecture
V0.3	11/25/2018	Additions to the introduction. Rewrite of the requirements section with enumeration. Additions to the System Architecture and System Design section including hardware architecture and software architecture with higher level and lower level diagrams with text describing these diagrams and text for these sections generally.
V0.4	12/16/2018	Made the requirements table. Clarified some requirements. Improved page formatting. Added some to the design section. Use case diagram created for the introduction and general mistakes addressed.

Table of Contents

1. Introduction

This design specification describes the architectures and design framework that will aid and abet the production of the ORPS-Robot version 0.4 – The Object Recognition and Path Smoothing Robot. The ORPS-Robot will be a platform for validating the research of Michael McCourt and a scheme for exploring object recognition via OpenCV with Robot Operating System, which is a powerful framework for writing robot software. There will be a demonstration of Simultaneous Localization and Mapping (SLAM). Together this will demonstrate a "finder robot" with applications in search and rescue and threat detection.

Additionally, there will be beacon triangulation and/or GPS to fuse additional location information into the SLAM or finder functionality. Typically, feedback is used when some amount of uncertainty exists within a system and the environment for which that system operates. To implement feedback this requires analyzing the underlying dynamics of whatever system you plan to look at and the implementation of communications, computing and software to accomplish some task. The project aims at investigating the dynamics of a semi-autonomous robot with the Object Recognition and Path Smoothing Robot by delivering a platform that is some interplay of communications, computing and software to Michael McCourt to aid in his research in control.

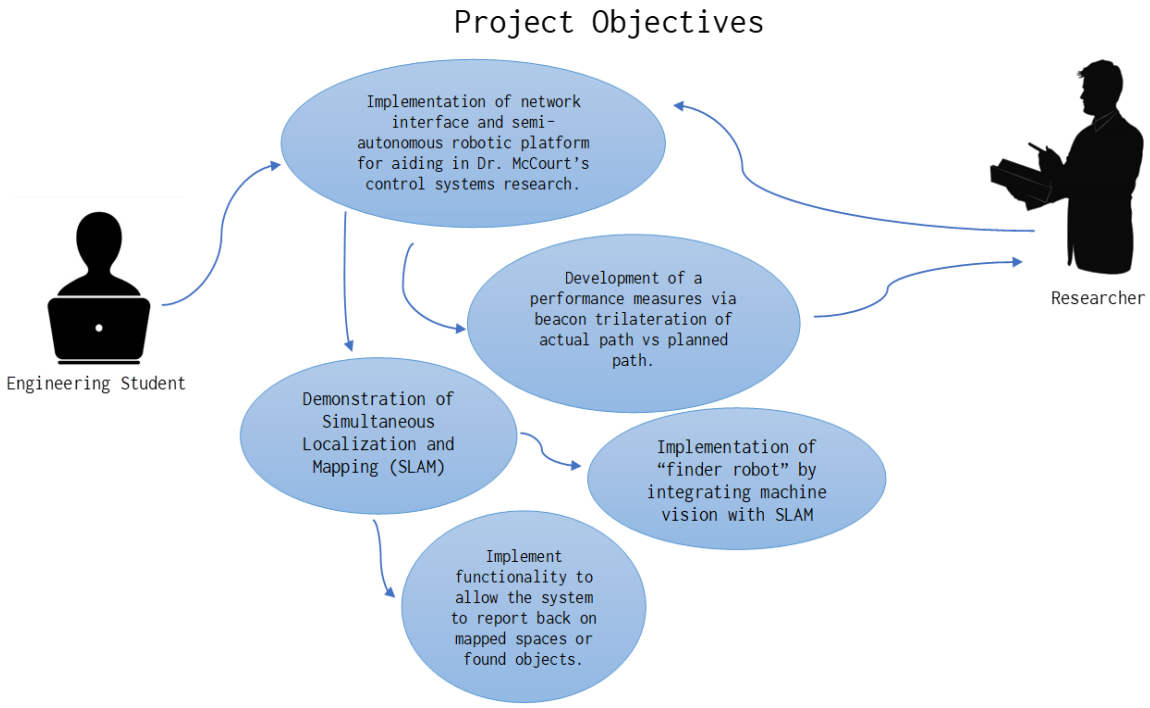

Figure 1: Use case diagram for the overall project

The networked control system to be explored in this project is commonly used in telerobotics. Telerobotics is still in the experimental phase of research and has been theorized for active shooter detection, a way for doctors to diagnose patients in disaster areas and telecommute from home.

1.1. Network Control

Network control systems have many useful applications; The typical use case involves using robots to interact with an environment that is too hazardous for a person. Any such Network control involves some delay of both outgoing control signals and incoming sensor data. In some situations, this delay may impair the intended function of the network control system.

Dr. McCourt has developed a set of filters intended to be placed in such a delayed, closed-loop control system. These filters apply a mathematical transformation on both incoming and outgoing loop signals such that communication delays are mitigated. This specification outlines the development of a controller-robot system intended to demonstrate the McCourt filter.

1.2. Autonomous Control

There may also be use cases for robots in hazardous environments where direct human control is impossible. Such a robot must be able to autonomously navigate and interact with an unknown space. SLAM is a fundamental technology for such autonomous activity, allowing the robot to navigate. Additionally, such an autonomous machine must be able to sense and recognize an objective before being able to interact with it. Computer vision is another fundamental technology for sensing a real-world environment. Sensing a condition is a necessary first step for being able to act based on the current environment.

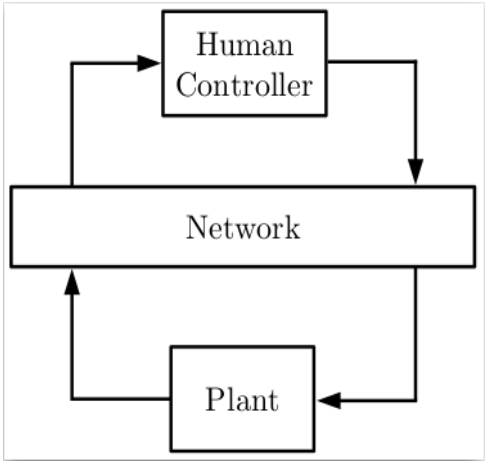

Figure 2: A typical network-control system

1.3. Scope

This specification covers the following:

- Any and all hardware modifications to the ORPS-Robot
- Software installed on the burger bot insofar as it deviates from the stock installation
- Base station control software setup insofar as it deviates from stock installation
- Any necessary techniques for integrating a Human Interface to the base station installation
- Any methods used to implement a communication delay between the robot and its base station
- Implementation of the McCourt input output transformation.
- An overview of implementing SLAM on the ORPS-Robot
- Integration of OpenCV into the ORPS-Robot

2. Requirements

In this section we will delve through the minimum requirements of this project to translate the needs of our client Michael McCourt into precise targets, establish metrics for a successful product and support design trade-off decisions. For this revision of the design specification we will articulate the marginally acceptable target values, without any of the ideal target values for our stretch goals.

Requirement Number	Requirement Criteria
R1	There must exist a mobile platform
R2	There must be a base station with a human interface device
R3	The base station must communicate with the mobile platform through a delayed link
R4	The mobile platform must be wirelessly controllable
R5	The mobile platform must be able to report its position in space
R6	There must be a means of recording and comparing the planned and actual path
R7	The mobile platform must be able to explore an unknown area
R8	The mobile platform must be able to map the unknown area
R9	The mobile platform must be able to identify and locate objects in the unknown area

2.1. Network Control

R1. There will be some ROS capable mobile platform

This robot must be ROS capable, mobile, and remotely controllable. The mobile platform must be capable of processing the R-transform for incoming and outgoing signals. All the hardware and software modules of the mobile platform must fit within the constraints of that platform, including such things as power, weight, and processing speed.

R2. There must be a base station interfaced with a USB game controller

The Human controller we have available currently is a USB game controller. This controller will have to be connected to some base station that can connect to the ORPS-Robot wirelessly. The base station is a laptop which will run the graphical user interface for the user, route communication traffic, perform transformations and public commands for the ORPS-Robot.

R3. The base station must communicate with the robot through a delayed link

In order to demonstrate the McCourt input-output transformation the robot must be remotely controllable. Control signals must be routed through a delayed communication medium. Ideally, or as a second stage, the human controller's feedback information should also be routed through the delayed medium.

R4. The Robot Should be wirelessly controllable

To fully demonstrate the usefulness of the McCourt transform we should model a real-world situation where the robot is out of sight and controllable solely from the base station.

R5. The robot must be able to report its position in space

To fully demonstrate the McCourt input-output transformation the full control loop should be routed through it as shown in Figure 2. This will require the position feedback to be in the form of a simple numerical array such as an x-y coordinate.

R6. There must be a means of recording and comparing the planned and actual path the robot follows

To demonstrate the McCourt input-output transformation we need to be able to compare the actual and planned path and have some measure of how closely they align. We should be able to make multiple test runs with and without the filter functioning and compare the course fidelity in aggregate.

2.2. Autonomous Control

R7. The mobile platform should be able to explore all accessible parts of a unknow area

If the robot is being used to search an area it will be important to ensure that the full area has been searched. Otherwise the system could produce a false negative. It is also important to consider the orientation and effective range of any sensors to ensure that they have made adequate coverage of the search area.

R8. The mobile platform should be able to create a map of the area

In order to be useful, The map should available on the base station as quickly as possible. This map should also include any found objects and links to sensor data for those objects. This will require some sort of performance measure to be defined for both the LiDAR and Marvelmind Beacons.

R9. The robot should be able to identify and locate objects through computer vision

The mobile platform should be able to be trained to identify specific objects in its environment. It should also be able to identify, locate, and report on the location or condition of these objects. Some sort of machine learning algorithm must be employed to detect objects in an environment in real time. This has not been well defined for our group yet and we don't even know if we're going to get to this portion of the project. This should be evident by the length of this section compared to the prior sections.

3. System Architecture

The main job of ORPS-Robot (Object Recognition and Path Smoothing) is to validate the research of Michael McCourt and a scheme for exploring object recognition via OpenCV with Robot Operating System (ROS). In this section we will develop our methodology for the overall system architecture by exploring the flow of data from each component in the overall system, by referring to the figure 2 on our right.

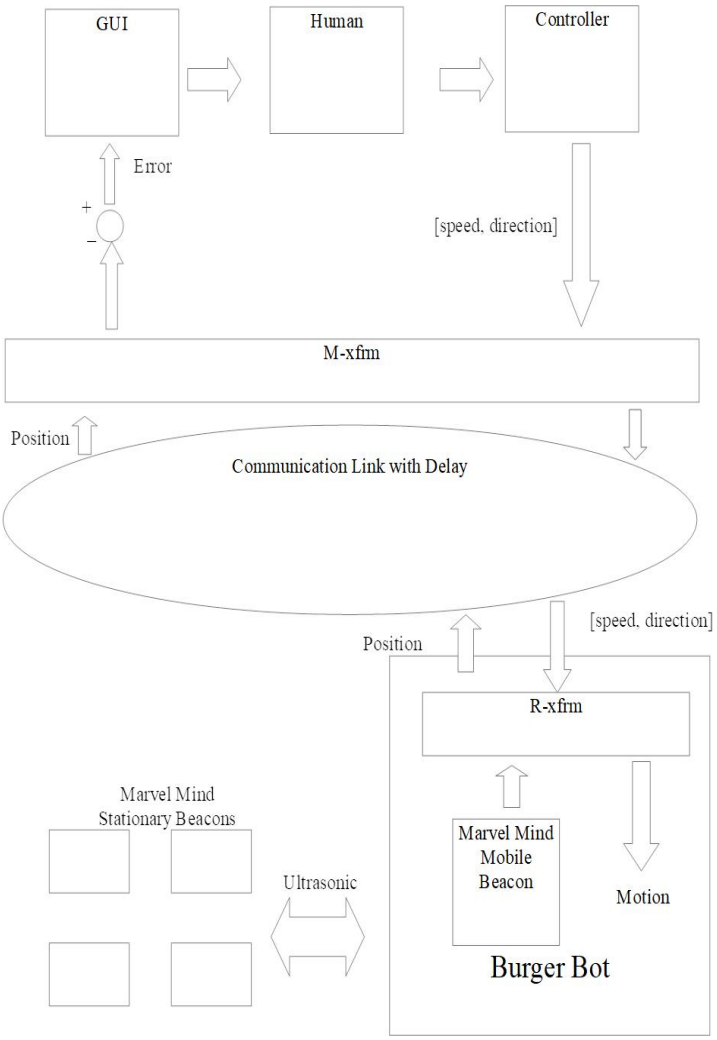

Figure 3: The System Architecture

First the user looks at the graphical user interface, which displays the reported position of the ORPS-Robot and the desired position based on some pre-programmed path. The user will attempt to move the robot towards the desired position via an Xbox 360 Controller. Movement commands, in the form of a vector will pass through the M-Transform before being transmitted over Wi-Fi to the ORPS-robot. Onboard the robot, the control vector will pass through the R-Transform on the Raspberry Pi 3 before being passed to the OpenCR as movement commands.

Marvelmind is an ultrasonic local positioning system. Several stationary beacons will be placed in the demonstration area. The ORPS-Robot will have the Marvelmind mobile beacon which will provide it with relative position information in the standard NMEA 0183 format. This data is then sent back to the basestation via Wi-Fi going first through the R-Transform then through the M-Transform where it is processed and presented to the user in the graphical user interface. This closes the loop. It is worth noting that no published ROS commands on the base station or the ORPS-Robot can be sent through the transform itself.

3.1. Hardware Architecture

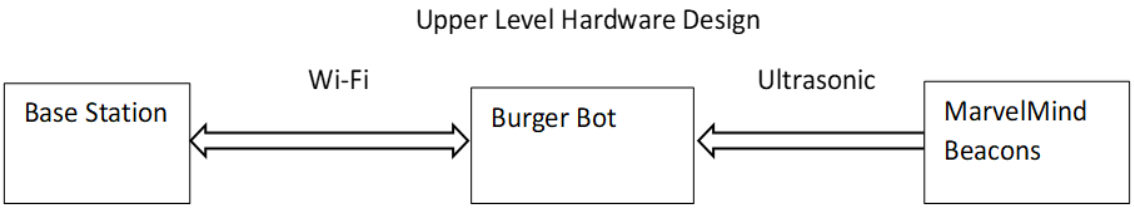

Figure 4: Overall Hardware Architecture

While hardware is not the focus of this project, the hardware architecture is an important component and each physical component will be introduced by their power requirements and/or communication protocols. The main hardware components are the Basestation, ORPS-Robot and Marvelmind Beacons. The basestation and ORPS-Robot will communicate via Wi-Fi while Marvelmind stationary beacons communicate amongst themselves via ultrasonic. A mobile Marvelmind beacon will communicate to the Raspberry Pi 3 via USB 2.0. The OpenCR microcontroller will communicate to the DYNAMIXAL Actuator system via serial signal via RS-485.

On the ORPS-Robot itself a Li-Po battery feeds the OpenCR microcontroller power which will then feed the Raspberry Pi 3 and Marvelmind mobile beacon. The OpenCR will be feeding power to the DYNAMIXAL Actuator system for movement.

The LDS-01 LiDAR will be mentioned briefly but we have not decided whether this is a crucial component of the architecture as it is becoming a stretch goal for the project, as such it is left out of the current set of diagrams.

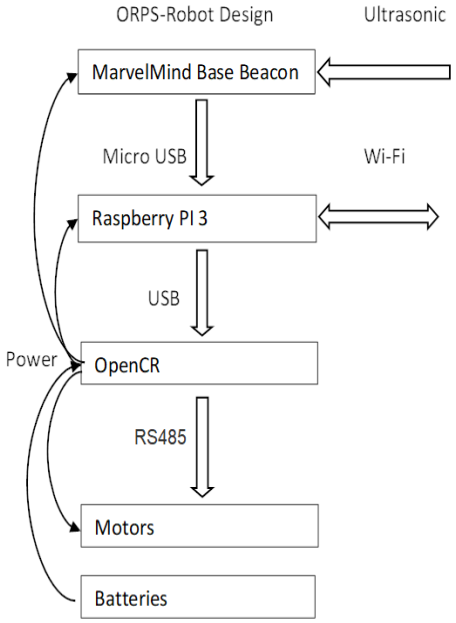

Figure 5: Hardware Architecture for the ORPS-Robot

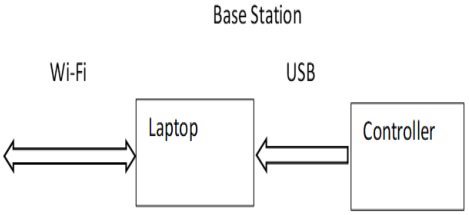

Figure 6: Hardware Architecture for Basestation

3.1.1. Communication Link

Most communications between devices in this project will be via either USB 2.0 or IEEE 802.11 Wi-Fi apart from Marvelmind which communicates via ultrasonic amongst themselves and USB 2.0 via the base station.

3.1.2. ORPS-Robot (TurtleBot3 Burger)

The ORPS-Robot is a collection of disparate components meant which combine to make a semi-autonomous robot meant for validating control systems research and to serve as a vessel for undergraduate education. The “Burger” design splits the robot into separate layers where each component is stored. The hardware in the ORPS-Robot is a literal stack with the LDS-01 LiDAR on the top of the robot, Raspberry Pi 3 on the middle stack and OpenCR microcontroller on the bottom stack

with the DYNAMIXAL Actuator system existing on the bottom with a single ball bearing for rotational movement.

3.1.4.1. Raspberry Pi 3 Model B

Raspberry Pi is a small computer which we are using as our main controller on the ORPS-Robot. It will be running Linux, and act as our point of contact to the ORPS-Robot. The Raspberry Pi 3 will be connected to the laptop via Wi-fi, and the OpenCR and Marvelmind via USB. The Raspberry Pi 3 will take signals from the laptop as well as the Marvel-Mind sensors, process them, then send them to the OpenCR board. The Pi is equipped with HDMI and USBs which we will use to program it.

3.1.4.2. OpenCR

OpenCR is a controller board used for controlling the different robotic components. This board is open source and made specifically for working with ROS systems. This board is attached to the battery, which it uses to supply power to all the other components on the robot. The OpenCR is attached to the Raspberry PI 3 via USB 2.0 and to the DYNAMIXAL Actuator system via RS-485. The OpenCR is programmable with the Arduino software development environment meaning that in addition to using ROS to control the ORPS-Robot, we have access to addition programming patterns using Arduino's C/C++ functions and libraries.

3.1.4.3. DYNAMIXAL Actuator system

The motors for the ORPS-Robot use DYNAMIXAL XL430-W250 motors. There is one motor for each of the two wheels. The motors are connected to the OpenCR on a TTL Multidrop Bus using a TTL Half Duplex Asynchronous Serial Communication protocol.

3.1.4.4. Li-Po Battery

The battery on the ORPS-Robot is a Li-Po Battery with a power output of 12V DC, 5A. This battery is connected to the OpenCR board where it is used to power the whole robot.

3.1.4.5. LDS-01 LiDAR

The LDS-01 is a 2D laser scanner capable of sensing $360°$ that collects a set of data around the ORPS-Robot to use for SLAM (Simultaneous Localization and Mapping) via a USB 2.0 connection on the Raspberry Pi 3. This portion was excluded from the figures as it is not a crucial part of the hardware architecture and may be excluded from the final revision.

3.2. Software Architecture

For this revision of the document we decided to meet the core requirements of the project. We will only talk about the software needed to meet the core of this project with the stretch goals to be talked about for a future revision.

There are two main software stacks, that of the basestation and ORPS-Robot. The basestation software will exist on a laptop, run a graphical user interface for the user controlling the robot to interact and control the robot via an Xbox 360 controller which will also be connected to this basestation. This data will be fed to a ROS receiver and be processed to be set to the ORPS-Robot. When the user makes some change in the controller input that is collected by the ROS receiver and fed through the M-transformation to be received by the Raspberry Pi 3 on the burgerbot via Wi-Fi. We have not worked out what this position data will look like, whether it be an actual video signal via a camera or the positioning data via the LiDAR but there will be some graphical user interface component to this project.

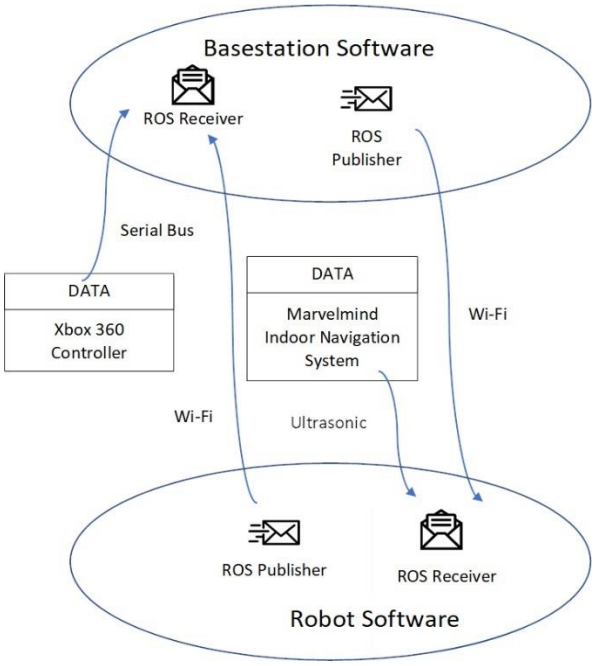

Figure 7: The overall software architecture with ROS publishers/receivers

On the basestation as mentioned previously we will be publishing commands to the burgerbot, including the movement commands via the controller but also doing error measurements between what the user commanded the robot to go and where it went. This is where the path smoothing part of the project comes into play. There will be some desired position and an actual position of the robot. The actual position will be what the Marvelmind ultrasonic indoor navigation system calculates and the desired position will be where the system tells the user to go. The aim is to minimize the error between the actual position and the desired position, even with latent inputs.

As for the software on the robot itself, it must be capable of receiving data from a Marvelmind sensor which will be connected to the robot itself via USB 2.0. This positioning data will be processed on the Marvelmind itself and will be published for further processing on the basestation. The robot software, which will be running on a Raspberry Pi 3 must be able to receive ROS commands from the basestation and publish the position data to the basestation. As the communication delay is too great to be doing this all on the basestation this means the system must be robust enough to be setup to run then process the information it receives on its own. This also means controlling the OpenCR, a microcontroller, to control the DYNAMIXEL actuator motors on the Burgerbot itself for movement.

4. System Design

For this project we need to produce a semiautonomous robot to aid in network control systems research. A network control system is a spatially distributed system wherein the control loop is closed through a communication network. This raises fundamental challenges and benefits to the overall

system design. By closing our loop over some communication network, we can do some processing on our laptop and some processing on the ORPS-Robot itself, allowing for a separation in system components, that in aggregate work better and more cheaply than standalone systems. The major challenges include unifying communication, estimation, control and *control over networks.* To send a signal over a digital network, the signal must be sampled, encoded in some digital format, transmitted over the network and finally the data must be decoded at the receiver side, which will induce large, potentially intermittent delays in the feedback loop. The variable network conditions such as congestion and channel quality may also induce variable delays between sampling and decoding at the receiver due to network access delays. Whenever you are dealing with feedback, reducing delay times are a primary for any linear time invariant system. It will be our job to reduce the delay and make it as stable as possible. The key role of this section will be to articulate a design pattern which will most accurately and acutely mitigate the limitations of network control systems.

4.1. Hardware Design

There are three levels to the ORPS robot. A bottom level where the chassis, motors, wheels and battery exist. A middle section where the OpenCR exists and a top section with Raspberry Pi. On top of the robot itself exists the LiDAR. The chassis are made of up waffle like plates, which allow for customization to our preferences, which we may very well take advantage of. We are unsure whether we will use the LiDAR or not on this system.

4.1.1. Objectives

The hardware design for this project is largely preset, coming from the manufacturer already assembled. In this respect our job is to finish this product, not design our own from scratch. There are tradeoffs and benefits to this approach. One tradeoff is that we have less flexibility to choose overcome challenges with hardware fixes, but the largest benefit is that it allows us to focus on delivering the product that our client wants most, which is a platform for research. The bottom line is to get this done as quickly and efficiently as possible, and that means taking what works off the shelf.

4.1.2. Constraints

I do not know what the constraints of our system are at this point or what should be in this section.

4.1.3. Composition

4.1.3.1. Xbox 360 Controller

The human interface device available is a commercial game controller. ROS has a node that can publish the current controller state as a topic. This controller will be connected to the laptop Base station using USB 2.0. The controller joysticks and triggers return an analog value to the laptop while the rest of the buttons work as a digital on and off switch.

4.1.3.2. Marvelmind Indoor Navigation System

Marvelmind indoor navigation system is designed for precise ($\pm 2cm$) location data meant for specifically off-the-shelf usage in autonomous robot, vehicle and copters. This product was chosen mainly due to its accuracy, which is very important for our application as it involves control systems based on position.

4.1.3.3. Raspberry Pi 3 Model B

Raspberry Pi is a small computer which we are using as our main controller on the ORPS-Robot. It will be running Libuntu, and act as our point of contact to the ORPS-Robot. The Raspberry was chosen for this project because of its small form factor, affordability, functionality, and because it was included in the burger bot package that was purchased.

If we need more computing power to run the image recognition stretch goal, we could switch out this board for something more powerful or outsource the processing to the laptop.

4.1.3.4. OpenCR

OpenCR is a controller board used for controlling the different robotic components. This board is open source and made specifically for working with ROS systems. This board is attached to the battery, which it uses to supply power to all the other components on the robot. The OpenCR is programmable with the Arduino software development environment meaning that in addition to using ROS to control the ORPS-Robot, we have access to addition programming patterns using Arduino's C/C++ functions and libraries. The OpenCR board was included in the Burger bot from ROBOTIS.

4.1.3.5. DYNAMIXEL Actuator System

The motors for the ORPS-Robot use DYNAMIXAL XL430-W250 motors. There is one motor for each of the two wheels. The motor system was included in the Burger bot from ROBOTIS.

4.1.3.6. Li-Po Battery

The battery on the ORPS-Robot is a Li-Po Battery with a power output of 12V DC, 5A. This battery is connected to the OpenCR board where it is used to power the whole robot. This is a 18,000 mAh battery. The battery was included in the ORPS-Robot.

4.1.3.7. LDS-01 LiDAR

The LDS-01 is a 2D laser scanner capable of sensing $360°$ that collects a set of data around the ORPS-Robot to use for SLAM (Simultaneous Localization and Mapping) via a USB 2.0 connection on the Raspberry Pi 3. This portion was excluded from the figures as it is not a crucial part of the hardware architecture and may be excluded from the final revision.

4.1.4. Interface

4.1.4.1. Xbox 360 Controller

The main controlling interface for the ORPS-Robot. This uses USB 2.0.

4.1.4.2. Marvelmind Indoor Navigation System

The Marvelmind sensors are attached to the Raspberry Pi via USB 2.0.

4.1.4.3. Raspberry Pi 3 Model B

The Raspberry Pi 3 will be connected to the laptop via Wi-fi, and the OpenCR and Marvelmind via USB. The Pi is equipped with HDMI and USBs which we will use to run the programs we create.

4.1.4.4. OpenCR

The OpenCR is connected to the Raspberry PI 3 via USB 2.0 and to the DYNAMIXAL Actuator system via RS-485. The OpenCR also has many other interfaces which we do not currently have any plans to use.

4.1.4.5. DYNAMIXAL Actuator System

The motors are connected to the OpenCR on a TTL Multidrop Bus using a TTL Half Duplex Asynchronous Serial Communication protocol.

4.1.4.6. Li-Po Battery

This battery is connected to the OpenCR using basic power connectors.

4.2. Software Design

This is mostly an embedded software project. There will be an embedded layer and host layer, where the embedded layer is split into three separate layers. For now, it will be described with a table. For future updates there will be a graphic that describes this. We need to meet as a group and hash out what this exactly looks like all together.

Inner Layer	Outer Layer	Software
User Interface	Host software	Graphical User Interface
User Interface	Host Software	User input script
User Interface	Host Software	Path Creation
Algorithm	Embedded software	Input-Output Transformation
Algorithm	Embedded Software	Communication Routing
Algorithm	Embedded Software	Mapping and Localization
Algorithm	Embedded Software	Task Definition
Platform	Embedded Software	Steering
Platform	Embedded Software	Sensor Fusion
Platform	Embedded Software	Image Recognition
Driver	Embedded Software	Marvelmind Sensor Interface
Driver	Embedded Software	Actuator Interface

4.2.1. Objectives

4.2.2. Constraints

4.2.3. Compositions

4.2.4. Uses and Interactions

4.2.5. Interface

4.2.6. Resources

4.2.7. Base Station Software

4.2.7.1. Robot Operating System (ROS)

ROS is a middleware library that creates interfaces for many common robot components and allows them to communicate in standardized ways. Each ROS process is called a node. Nodes communicate through topics or services. A service provides a classic server-client model where the client makes a request and the server responds. A topic implements a logical many-to-many data bus. Nodes can publish to or subscribe from any topic without knowing anything about other nodes on the topic.

4.2.7.2. Graphical User Interface (GUI)

When in remote control mode the robot can be controlled by a consumer game controller. Ideally the user will be presented with a graphic of a 2-dimensional field with a marker representing the robot's

current position and a fixed path to follow. The user will be tasked with attempting to make the robot follow the displayed path.

4.2.8. R-Transformation

The control signals from the Wi-Fi network will take the form of a 1x2 integer vector indicating speed and direction. As of this writing the actual math has not been explained to us for the R-Transformation other than to say this is more than a simple linear transformation, requiring more operations than the M-Transform. Our job is to test this functionality as a black box, not necessarily understand it.

4.2.9. M-Transformation

The control signals from the human controller will take the form of a 1x2 integer vector indicating speed and direction. As of this writing the actual math has not been explained to us for the M-Transformation other than to say this is a simple linear transformation. Our job is to test this functionality as a black box, not necessarily understand it.

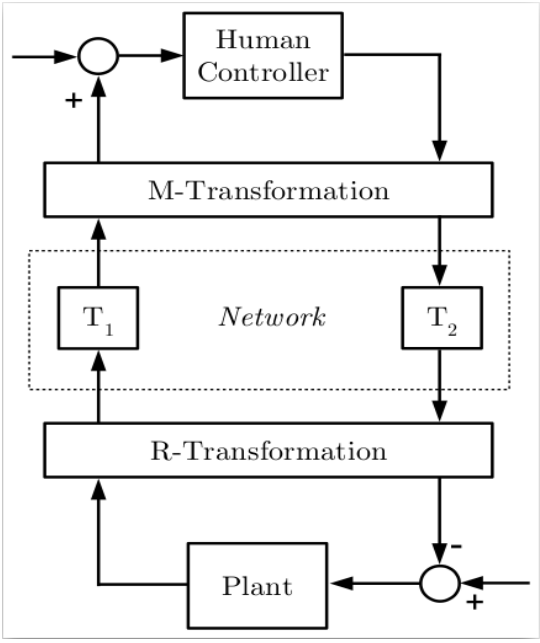

Figure 8: The McCourt input-output transformations

4.3. Human Interface Design

The ORPS-Robot must provide a graphical user interface that provides some prompt on how to control the robot via the controller or keyboard and how to read the commands. Good design is an act of communication. The graphical user interface must be cognizant of the type of user that will use this device and able to be operated by not only the creators of the ORPS-Robot but Michael McCourt and future researchers in this area.

5. Ethical Considerations

In this project, our specific goals do not raise many ethical concerns, however, there are still things that could happen from this project that should be considered.

Future use of McCourt's filter: The primary purpose of McCourt's algorithm is to help with the delay of human interacting in closed loop control systems. This algorithm could be used for military systems or other weapons systems. This is not directly our concern as McCourt's algorithms are not ours, but our work in enabling it could be considered helping. This concern is mainly dependent on the motives and intensions of the IP owner. With our countries current IP laws and our confidence in the owner of the IP we are not in any way currently worried about this ethical concern.

Complete testing: McCourt's filter could be use in the future in systems that could cause harm if they were to malfunction. Because our testing is a proof of concept, it is partly our responsibility to make sure this algorithm works in practice, so there is little chance of malfunctions in the future of this algorithms life.

Use of funds: This project is funded by Dr. McCourt who has funding from the University of Washington. We did not technically earn this money, so it is our responsibility to use this money efficiently, and to not cause money to be spent due to mistakes.

6. References

1. Turtlebot3 Manual: http://emanual.robotis.com/docs/en/platform/turtlebot3/overview/
2. Robot Operating System: http://www.ros.org/
3. Open Source Computer Vision Library (OpenCV): https://opencv.org/

7. Errata

None Currently