Architectural Design Specification

Electric-Eye

Nichelous Herndon
Adis Kovacevic
Zack Davis
Juan Sifuentes
Rommel Alonzo
## TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Figures</td>
<td>5</td>
</tr>
<tr>
<td>List of Tables</td>
<td>6</td>
</tr>
<tr>
<td>Document Revision History</td>
<td>7</td>
</tr>
<tr>
<td>1. Introduction</td>
<td>8</td>
</tr>
<tr>
<td>1.1 Overview</td>
<td>8</td>
</tr>
<tr>
<td>1.2 Product Concept</td>
<td>8</td>
</tr>
<tr>
<td>1.3 Project Scope</td>
<td>9</td>
</tr>
<tr>
<td>2. Meta-Architecture</td>
<td>10</td>
</tr>
<tr>
<td>2.1 Architectural Vision</td>
<td>10</td>
</tr>
<tr>
<td>2.2 Guiding Assumptions</td>
<td>10</td>
</tr>
<tr>
<td>2.3 Guiding Principles</td>
<td>10</td>
</tr>
<tr>
<td>2.4 Tradeoffs</td>
<td>11</td>
</tr>
<tr>
<td>3. Architecture Overview</td>
<td>12</td>
</tr>
<tr>
<td>3.1 Block Diagram</td>
<td>12</td>
</tr>
<tr>
<td>3.2 Input/output (I/O) Interface Layer</td>
<td>12</td>
</tr>
<tr>
<td>3.3 Data Processing Layer</td>
<td>13</td>
</tr>
<tr>
<td>3.4 Communication Layer</td>
<td>13</td>
</tr>
<tr>
<td>4. I/O Interface Layer</td>
<td>15</td>
</tr>
<tr>
<td>4.1 Description</td>
<td>15</td>
</tr>
<tr>
<td>4.2 Purpose</td>
<td>15</td>
</tr>
<tr>
<td>4.3 Function</td>
<td>15</td>
</tr>
<tr>
<td>4.4 Dependencies</td>
<td>16</td>
</tr>
</tbody>
</table>
8.1 Overview
8.2 Inter-Subsystem Data Element Descriptions
8.3 Producer-Consumer Relationship

9. Requirements Mapping
9.1 Overview

10. Testing Considerations
10.1 Overview
10.2 Testing Approach
10.3 I/O Interface Layer
10.4 Data Processing Layer
10.5 Communication Layer

11. Glossary
11.1 Robot Operating System (ROS)
11.2 Point Cloud Library (PCL)
11.3 Transformation Matrix (6DOF)
11.4 RGB-D Camera
11.5 OpenNI
11.6 Point Cloud
11.7 Point Cloud Frame
11.8 Mapping
11.9 Localization
11.10 ROS Node
List of Figures

Figure 1-1: Product Concept .............................................................................................................. 8
Figure 3-1: Electric-Eye Block Diagram.......................................................................................... 12
Figure 4-1: I/O Interface Layer with Subsystems ............................................................................. 15
Figure 5-1: Data Processing Layer with Subsystems........................................................................ 19
Figure 6-1: Communication Layer with Subsystems ........................................................................ 25
Figure 8-1: Inter-Subsystem Data Flow .......................................................................................... 30
List of Tables

Table 4-1: I/O Interface Layer Interfaces ................................................................. 16
Table 4-2: RGB-D Camera Interpreter Subsystem Interfaces ................................ 17
Table 4-3: Signal Interpreter Subsystem Interfaces .................................................. 17
Table 5-1: Data Processing Layer Interfaces ............................................................ 20
Table 5-2: Controller Subsystem Interfaces .............................................................. 21
Table 5-3: Localization Subsystem Interfaces ........................................................... 22
Table 5-4: Mapping Subsystem Interfaces ................................................................. 23
Table 5-5: Store Frame Subsystem Interfaces ............................................................ 23
Table 5-6: Camera Handler Subsystem Interfaces ...................................................... 24
Table 6-1: Communication Layer Interfaces ............................................................. 26
Table 6-2: Pack Transformation Matrix Subsystem Interfaces ................................... 27
Table 6-3: Command Interpreter Subsystem Interfaces ............................................ 27
Table 8-1: Inter-Subsystem Data Element Descriptions .......................................... 32
Table 8-2: Producer-Consumer Relationship ............................................................ 33
Table 9-1: Requirements Mapping ............................................................................ 34
## Document Revision History

<table>
<thead>
<tr>
<th>Revision Number</th>
<th>Revision Date</th>
<th>Description</th>
<th>Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>12/28/2012</td>
<td>First Draft</td>
<td>Compiling all individual sections</td>
</tr>
<tr>
<td>0.2</td>
<td>1/4/2013</td>
<td>Final Draft</td>
<td>Making necessary changes to first draft</td>
</tr>
<tr>
<td>0.9</td>
<td>1/16/2013</td>
<td>Final Draft</td>
<td>Formatting final draft and making small changes</td>
</tr>
<tr>
<td>1.0</td>
<td>1/21/2013</td>
<td>Final Draft</td>
<td>Making final changes to final draft, suggested by team members and our peer review team</td>
</tr>
<tr>
<td>1.1</td>
<td>1/28/2013</td>
<td>Baseline Revisions</td>
<td>Making changes after Gate Review Presentation</td>
</tr>
<tr>
<td>2.0</td>
<td>1/31/2013</td>
<td>Baseline Revisions</td>
<td>Making changes as suggested by Professor O'Dell</td>
</tr>
</tbody>
</table>
1. Introduction

1.1 Overview

The Architectural Design Specification will cover the architectural design of the Electric-Eye and include details of the product concept, scope, and key requirements.

1.2 Product Concept

The Electric-Eye is a module that will first scan and map a room using the point cloud library in 3D, then perform localization when mounted onto a robot. Users of this product will be able to integrate this product into their robots that use the robot operating system (ROS) and use the features provided by the module. This will provide an easy to use plug and play system which robot designers can easily integrate into their projects.

![Figure 1-1: Product Concept](image-url)
1.3 Project Scope

The Electric-Eye will be limited to a standard sized room. It should be an easy to use system where the module shall assist the robot by giving it precise and formatted data. The key requirements will represent the most important functionalities that the Electric-Eye needs to make it successful.

The Electric-Eye must be able to scan an average size indoor room, map a world model based on the frames stored, and localize a robot based on its location in the world map. All of these operations must be processed within a reasonable time, and it is only allowed to be in one mode at a given period. The product must also be ROS compatible to the extent that it can send the transformation matrix and receive simple standard commands from ROS. What the robot does with the information provided by the product is up to the robot designer.

For packaging requirements, the Electric-Eye has to be physically mountable to a robot (which depends on the user). It must also have a single case housing to provide minimal interference with the robot. Since safety is a great concern, the Electric-Eye must have no sharp edges and be able to prevent fires from overheating and power surges.

Other requirements that must be noted are that the Electric-Eye has to be tested to ensure functionality and be able to operate under small vibration such as a robot operating on rugged terrain. Since it will draw power from a robot, it must not draw more than 150 watts of power, and be able to have an on board storage of at least 8 GB. To finalize the product, we will include documentation in American English, and will not provide further maintenance after May 2013.
2. Meta-Architecture

2.1 Architectural Vision

The Electric-Eye is a module that will use a RGB-D camera to read in frames from the surrounding environment. The frames will then be processed and converted into a map that the system will use to perform localization operations with. Once localization is completed, a transformation matrix will be sent to ROS. Given the functionality of the Electric-Eye along with our guiding principles, we have decided to split the architecture into 3 layers: I/O Interface, Data Processing, and Communication. Each layer will be independent of the other layers which will make the system easier to change, should change be needed in a certain layer.

2.2 Guiding Assumptions

When designing the architecture for the Electric-Eye, we came up with some assumptions that impact the design of the product. The following are the guiding assumptions for the Electric-Eye:

- Power will be provided by the host robot
- The module will be used indoors.
- The module will be used in a well-lit area
- Mapping operations will be performed before localization operations

2.3 Guiding Principles

Guiding principles are rules that we have discussed as a team and decided on. These principles will influence the design of the system which will help us ensure that the system is working properly and will also ensure that we meet our customer’s requirements.

2.3.1 Modularity

Components of the system shall work independently and shall also work together as a whole.

2.3.2 Component Recovery

System components shall recover from software failures experienced within their own layer subsystems. Software failure will be limited to one layer. In other words, an error will not extend beyond the layer it originated from. The error will be contained, and the layer will do appropriate exception handling.
2.3.3 Extensibility
The system shall be designed so that the signal interpreter subsystem will have the ability to handle more than 3 switch positions if a user decides to further extend the modes of the Electric-Eye.

2.3.4 External Compatibility
The system must be able to integrate with ROS and any RGB-D camera that is OpenNI compatible.

2.3.5 Software Efficiency
The system software components shall not overwhelm storage and/or memory resources.

2.4 Tradeoffs
The first tradeoff that we had when designing the systems architecture involved the on-board storage component. Originally we wanted to have storage as its own separate layer that would handle the storing of frames and would also retrieve data and send it to the data processing layer. In the end, we decided to keep the system simple and make the storage component a part of the data processing layer. The decision to make it only part of the data processing layer still keeps our system modular and keeps our layers independent of each other because the data processing layer is the only layer that accesses the on-board storage.

Another tradeoff involved which camera we would be using for the module. Initially the project was going to use the Kinect camera to read in frames to the module. Since we are using the PCL library to handle our mapping and localization algorithms we decided that we could use any camera that is OpenNI compatible to read in frames to the module. Designing the system to use any OpenNI compatible card will allow more flexibility to the users of the module allowing them to use cameras other than the Kinect.
3. Architecture Overview

3.1 Block Diagram

The Electric-Eye will have 3 main layers to its architecture. The layers are: I/O Interface, Data Processing, and Communication. The figure below is the block diagram for the Electric-Eye.

![Electric-Eye Block Diagram]

3.2 Input/output (I/O) Interface Layer

3.2.1 Description

This layer will communicate with the camera, convert camera input to point cloud format, and interpret the switch's position. Any information coming in from the camera or mode switch will be interpreted in this layer and sent to the Data Processing Layer.
3.2.2 Services Provided
- Sends point cloud data to the Data Processing Layer
- Sends commands to the camera
- Sends switch mode to Data Processing Layer

3.2.3 Services Expected
- Raw data from the camera
- Raw data from the switch
- Camera command from the Data Processing Layer

3.3 Data Processing Layer

3.3.1 Description
This layer will perform read/write operations to the on-board storage component along with mapping and localization operations. All major processing will be handled in this layer, specifically the world map and transformation matrix.

3.3.2 Services Provided
- Commands to the I/O Interface Layer for camera control
- Transformation matrix to the Communication Layer

3.3.3 Services Expected
- Switch mode from the I/O Interface Layer
- Point cloud frame from the I/O Interface Layer
- ROS sleep command from the Communication Layer

3.4 Communication Layer

3.4.1 Description
This layer will provide the necessary communication between the module and ROS. In this layer, messages will be packed into ROS format and transmitted to ROS. This layer will also receive commands from ROS, when a message is received it will be unpacked, interpreted, and sent to the Data Processing Layer.
3.4.2 Services Provided

- Packs a transformation matrix and sends it to ROS
- Unpacks and interprets ROS messages and sends them to the Data Processing Layer

3.4.3 Services Expected

- A transformation matrix is expected from the Data Processing Layer
- Commands from ROS
4. I/O Interface Layer

4.1 Description

The I/O interface layer is responsible for converting any raw information signals into data that the system can use. This module has only two subsystems: RGB-D camera interpreter and Signal Interpreter.

4.2 Purpose

The purpose of this layer is to provide an interface between the other systems and any hardware required to operate the Electric-eye.

4.3 Function

When gathering depth information from an RGB-D camera, the RGB-D camera interpreter subsystem will convert this information into a Point Cloud. Equally, the Signal Interpreter subsystem is meant to convert an analogue signal coming from a switch and convert it into a unique digital signal. Each unique digital signal will correspond to the state that switch is in. Each of these subsystems will asynchronously send their formatted data into the Data Processing Layer.
4.4 Dependencies

The I/O interface layer depends on receiving information from two sources. The first source is from an OpenNI compatible RGB-D camera meant to provide this layer with depth information of its surroundings. The second source is from a switch which is meant to represent the current state of the Electric-Eye.

4.5 Interfaces

<table>
<thead>
<tr>
<th>Type</th>
<th>Source</th>
<th>Destination</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal</td>
<td>RGB-D Camera Interpreter</td>
<td>Data Processing Layer</td>
<td>Point Cloud</td>
</tr>
<tr>
<td>Internal</td>
<td>Signal Interpreter</td>
<td>Data Processing Layer</td>
<td>Integer Value</td>
</tr>
<tr>
<td>External</td>
<td>RGB-D Camera</td>
<td>RGB-D Camera Interpreter</td>
<td>Unformatted Depth Information</td>
</tr>
<tr>
<td>External</td>
<td>Switch</td>
<td>Signal Interpreter</td>
<td>Analogue Signal</td>
</tr>
</tbody>
</table>

Table 4.1: I/O Interface Layer Interfaces

4.6 Subsystems

4.6.1 RGB-D Camera Interpreter

4.6.1.1 PURPOSE

Provide an abstraction interface between other systems and a RGB-D camera.

4.6.1.2 FUNCTION

Initialize and read in depth information coming from an OpenNI compatible RGB-D camera. Also, as information is being read, the information will be converted into usable data.

4.6.1.3 DEPENDENCIES

This subsystem depends on an OpenNI compatible RGB-D camera.
4.6.1.4 INTERFACES

<table>
<thead>
<tr>
<th>Type</th>
<th>Source</th>
<th>Destination</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal</td>
<td>RGB-D Camera Interpreter</td>
<td>Data Processing Layer</td>
<td>Point Cloud</td>
</tr>
<tr>
<td>External</td>
<td>RGB-D Camera</td>
<td>RGB-D Camera Interpreter</td>
<td>Unformatted Depth Information</td>
</tr>
</tbody>
</table>

Table 4-2: RGB-D Camera Interpreter Subsystem Interfaces

4.6.1.5 PROCESSING

The subsystem will convert depth information into Point Cloud data object.

4.6.1.6 DATA

The data produced will be depth information of the module’s current surroundings.

4.6.2 Signal Interpreter

4.6.2.1 PURPOSE

Provide an abstraction interface between other systems and the switch.

4.6.2.2 FUNCTION

This subsystem is meant to take in an analogue signal that is controlled by switch and convert it into a corresponding digital signal. This signal is then sent to the Data Processing Layer.

4.6.2.3 DEPENDENCIES

This subsystem is dependent on a hardware switch.

4.6.2.4 INTERFACES

<table>
<thead>
<tr>
<th>Type</th>
<th>Source</th>
<th>Destination</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal</td>
<td>Signal Interpreter</td>
<td>Data Processing Layer</td>
<td>Integer Value</td>
</tr>
<tr>
<td>External</td>
<td>Switch</td>
<td>Signal Interpreter</td>
<td>Analogue Switch</td>
</tr>
</tbody>
</table>

Table 4-3: Signal Interpreter Subsystem Interfaces
4.6.2.5  PROCESSING

Convert analogue signal into an integer value.

4.6.2.6  DATA

Data will consist of an analogue switch signal.
5. Data Processing Layer

5.1 Description

The data processing layer will sort through any data that is given or is holding (in internal storage) and provide a necessary output to the appropriate systems.

5.2 Purpose

The purpose of this layer is to perform mapping, localization, and storing of frames to on-board storage.

5.3 Function

This layer has two functions. The first is to provide mapping functionality when given depth information of its environment and create a world map of it. The second is to provide localization information by using a stored world map and current depth information from the camera as references.

5.4 Dependencies

The Data Processing Layer requires depth information when requested from the I/O interface layer as well as state information from the switch.
5.5 Interfaces

<table>
<thead>
<tr>
<th>Type</th>
<th>Source</th>
<th>Destination</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>External</td>
<td>I/O Interface Layer</td>
<td>Camera Handler</td>
<td>Point Cloud</td>
</tr>
<tr>
<td>External</td>
<td>I/O Interface Layer</td>
<td>Controller</td>
<td>Integer Value</td>
</tr>
<tr>
<td>External</td>
<td>Localization</td>
<td>Communication Layer</td>
<td>6DOF Frame</td>
</tr>
<tr>
<td>External</td>
<td>Command Interpreter</td>
<td>Controller</td>
<td>Command</td>
</tr>
<tr>
<td>Internal</td>
<td>Controller</td>
<td>Localization</td>
<td>Status Signal</td>
</tr>
<tr>
<td>Internal</td>
<td>Controller</td>
<td>Mapping</td>
<td>Status Signal</td>
</tr>
<tr>
<td>Internal</td>
<td>Controller</td>
<td>Store Frame</td>
<td>Status Signal</td>
</tr>
<tr>
<td>Internal</td>
<td>Camera Handler</td>
<td>Localization</td>
<td>Point Cloud</td>
</tr>
<tr>
<td>Internal</td>
<td>Camera Handler</td>
<td>Store Frame</td>
<td>Point Cloud</td>
</tr>
<tr>
<td>External</td>
<td>Store Frame</td>
<td>Internal Storage</td>
<td>Point Cloud</td>
</tr>
<tr>
<td>External</td>
<td>Internal Storage</td>
<td>Localization</td>
<td>Point Cloud</td>
</tr>
</tbody>
</table>

Table 5.1: Data Processing Layer Interfaces

5.6 Subsystems

5.6.1 Controller

5.6.1.1 PURPOSE

The purpose of this subsystem is to help control which other subsystems are supposed to be operating. By sending control signals, the controller subsystem can turn on/off any other subsystem as required.
5.6.1.2 FUNCTION
This subsystem will be able to send control signals to other subsystems to turn them on/off. Also from the communication layer, it must be able to receive a control state signal to effectively place the RGB-D camera in sleep mode.

5.6.1.3 DEPENDENCIES
This subsystem is dependent on receiving signals from both the I/O and Communication Layer to determine what state the other subsystems need to be in.

5.6.1.4 INTERFACES

<table>
<thead>
<tr>
<th>Type</th>
<th>Source</th>
<th>Destination</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>External</td>
<td>I/O Interface Layer</td>
<td>Controller</td>
<td>Integer Value</td>
</tr>
<tr>
<td>External</td>
<td>Command Interpreter</td>
<td>Controller</td>
<td>Command</td>
</tr>
<tr>
<td>Internal</td>
<td>Controller</td>
<td>Localization</td>
<td>Status Signal</td>
</tr>
<tr>
<td>Internal</td>
<td>Controller</td>
<td>Mapping</td>
<td>Status Signal</td>
</tr>
<tr>
<td>Internal</td>
<td>Controller</td>
<td>Store Frame</td>
<td>Status Signal</td>
</tr>
</tbody>
</table>

Table 5-2: Controller Subsystem Interfaces

5.6.1.5 PROCESSING
The purpose of this subsystem is to send control information to respective subsystems.

5.6.1.6 DATA
Receives control and binary state number from other external systems and also send respective control signals to other subsystems.

5.6.2 Localization

5.6.2.1 PURPOSE
The purpose of this subsystem is to compute localization information.

5.6.2.2 FUNCTION
By receiving current depth information from the Camera Handler subsystem, the subsystem will compute localization information using a stored world map.
5.6.2.3 DEPENDENCIES
This subsystem depends on receiving control information from the Controller subsystem and from receiving current depth information from the Camera Handler. Also, the subsystem gains access to the world map by reading it in from its internal storage drive.

5.6.2.4 INTERFACES

<table>
<thead>
<tr>
<th>Type</th>
<th>Source</th>
<th>Destination</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>External</td>
<td>Localization</td>
<td>Communication Layer</td>
<td>6DOF Frame</td>
</tr>
<tr>
<td>Internal</td>
<td>Controller</td>
<td>Localization</td>
<td>Status Signal</td>
</tr>
<tr>
<td>Internal</td>
<td>Camera Handler</td>
<td>Localization</td>
<td>Point Cloud</td>
</tr>
<tr>
<td>External</td>
<td>Internal Storage</td>
<td>Localization</td>
<td>Point Cloud</td>
</tr>
</tbody>
</table>

Table 5-3: Localization Subsystem Interfaces

5.6.2.5 PROCESSING
This subsystem computes a 6DOF frame and sends that frame to the Communication Layer.

5.6.2.6 DATA
The subsystem receives current depth and state information and a world map, computes a 6DOF frame and sends it to the Communication Layer.

5.6.3 Mapping
5.6.3.1 PURPOSE
The purpose of this subsystem is to create a world map and store it onto the on-board storage.

5.6.3.2 FUNCTION
Receives depth frames from internal storage and concatenates them together to create a world map. Once done, the subsystem will store the completed world map.

5.6.3.3 DEPENDENCIES
This subsystem needs access to the Electric-Eye's internal storage to read in depth frames and to store the world map.
5.6.3.4  INTERFACES

<table>
<thead>
<tr>
<th>Type</th>
<th>Source</th>
<th>Destination</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal</td>
<td>Controller</td>
<td>Mapping</td>
<td>Status Signal</td>
</tr>
</tbody>
</table>

Table 5.4: Mapping Subsystem Interfaces

5.6.3.5  PROCESSING

The mapping subsystem will use mapping algorithms to concatenate depth frames to create a world map.

5.6.3.6  DATA

Data will consist of depth frames and a world map.

5.6.4  Store Frames

5.6.4.1  PURPOSE

The purpose of this subsystem is to store depth frames into the Electric-Eye's internal storage.

5.6.4.2  FUNCTION

This subsystem will store depth frames.

5.6.4.3  DEPENDENCIES

This subsystem will need the Camera Handler subsystem to transfer current depth frames and needs access to the Electric-Eye's internal storage.

5.6.4.4  INTERFACES

<table>
<thead>
<tr>
<th>Type</th>
<th>Source</th>
<th>Destination</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal</td>
<td>Controller</td>
<td>Store Frame</td>
<td>Status Signal</td>
</tr>
<tr>
<td>Internal</td>
<td>Camera Handler</td>
<td>Store Frame</td>
<td>Point Cloud</td>
</tr>
<tr>
<td>External</td>
<td>Store Frame</td>
<td>Internal Storage</td>
<td>Point Cloud</td>
</tr>
</tbody>
</table>

Table 5.5: Store Frame Subsystem Interfaces

5.6.4.5  PROCESSING

Moves formatted depth frames into storage

5.6.4.6  DATA

Data will consist of processed depth frames.
5.6.5 Camera Handler

5.6.5.1 PURPOSE

The purpose of the Camera Controller subsystem is to be a waypoint for other subsystem in the Data Processing Layer to control the RGB-D camera.

5.6.5.2 FUNCTION

Send control information and receive depth frames from I/O Interface Layer.

5.6.5.3 DEPENDENCIES

N/A

5.6.5.4 INTERFACES

<table>
<thead>
<tr>
<th>Type</th>
<th>Source</th>
<th>Destination</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>External</td>
<td>I/O Interface Layer</td>
<td>Camera Handler</td>
<td>Point Cloud</td>
</tr>
<tr>
<td>Internal</td>
<td>Camera Handler</td>
<td>Localization</td>
<td>Point Cloud</td>
</tr>
<tr>
<td>Internal</td>
<td>Camera Handler</td>
<td>Store Frame</td>
<td>Point Cloud</td>
</tr>
</tbody>
</table>

*Table 5-6: Camera Handler Subsystem Interfaces*

5.6.5.5 PROCESSING

N/A

5.6.5.6 DATA

Data will consist of processed depth frames.
6. Communication Layer

6.1 Description

The Communication layer is responsible for handling all ROS communication for the Electric-Eye. It will receive data that is outgoing and pack it into a ROS compatible message. It also receives ROS messages, unpacks them, filters the message for appropriate commands, and sends them out to the Data Processing Layer.

6.2 Purpose

The purpose of this layer is to centralize all ROS communications and allow for an interface with the module and ROS.

6.3 Function

The Communication Layer will provide the Electric-Eye with the ROS commands that are coming in, filtering out unimplemented commands. It will also send out the modules transformation matrix to external ROS.

6.4 Dependencies

The Communication Layer is dependent on the Localization subsystem in the Data Processing Layer.
6.5 Interfaces

<table>
<thead>
<tr>
<th>Type</th>
<th>Source</th>
<th>Destination</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>External</td>
<td>Data Processing Layer</td>
<td>Pack Transformation Matrix</td>
<td>6DOF Frame</td>
</tr>
<tr>
<td>External</td>
<td>Pack Transformation Matrix</td>
<td>ROS</td>
<td>ROS Packed 6DOF Frame</td>
</tr>
<tr>
<td>External</td>
<td>Command Interpreter</td>
<td>Controller</td>
<td>Integer Value</td>
</tr>
<tr>
<td>External</td>
<td>ROS</td>
<td>Command Interpreter</td>
<td>ROS Packed Message</td>
</tr>
</tbody>
</table>

Table 6-1: Communication Layer Interfaces

6.6 Processing

The Communication Layer will process the transformation matrix into a ROS message. It will also process incoming ROS messages, filter the messages to ensure that it is a recognized command, and send them out to the Data Processing Layer.

6.7 Data

Data will consist of ROS messages.

6.8 Subsystems

6.8.1 Pack Transformation Matrix

6.8.1.1 PURPOSE

The purpose of the Pack Transformation Matrix subsystem is to parse the transformation matrix into a ROS message and send that message out of the module.

6.8.1.2 FUNCTION

This subsystem will convert a transformation matrix to a ROS compatible format.

6.8.1.3 DEPENDENCIES

This subsystem is dependent on the Data Processing Layer’s Localization subsystem
6.8.1.4 INTERFACES

<table>
<thead>
<tr>
<th>Type</th>
<th>Source</th>
<th>Destination</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>External</td>
<td>Pack Transformation Matrix</td>
<td>ROS</td>
<td>ROS Packed 6DOF Frame</td>
</tr>
<tr>
<td>External</td>
<td>Data Processing Layer</td>
<td>Pack Transformation Matrix</td>
<td>6DOF Frame</td>
</tr>
</tbody>
</table>

Table 6-2: Pack Transformation Matrix Subsystem Interfaces

6.8.1.5 PROCESSING

Process Transformation matrix to a ROS message.

6.8.1.6 DATA

Data consists of a ROS message.

6.8.2 Command Interpreter

6.8.2.1 PURPOSE

The purpose of the Command Interpreter is to parse incoming message and unpack them, then send the message to the Data Processing Layer’s Controller subsystem. Filtering will be done in this subsystem to ensure that any commands or data that ROS has sent to the Electric-Eye is a supported command.

6.8.2.2 FUNCTION

Unpack ROS messages and send them up to the Data Processing Layer.

6.8.2.3 DEPENDENCIES

This subsystem is dependent on a ROS message.

6.8.2.4 INTERFACES

<table>
<thead>
<tr>
<th>Type</th>
<th>Source</th>
<th>Destination</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>External</td>
<td>Command Interpreter</td>
<td>Data Processing Layer</td>
<td>Integer Value</td>
</tr>
<tr>
<td>External</td>
<td>ROS</td>
<td>Command Interpreter</td>
<td>ROS Packed Integer</td>
</tr>
</tbody>
</table>

Table 6-3: Command Interpreter Subsystem Interfaces
6.8.2.5  PROCESSING

Process messages from ROS.

6.8.2.6  DATA

ROS message
7. Operating System Dependencies

7.1 Overview

Each layer will be developed in C++ with a Linux-based PC. The executable file will be transferred to the Electric-Eye module that will also run a Linux-based operating system.

7.2 I/O Interface Layer

The I/O Interface Layer will be dependent on PCL libraries at runtime. The Linux operating system will utilize these Point-Cloud libraries and the drivers provided by OpenNI to read in frames from the RGB-D camera during runtime. This layer will also utilize the CUDA GPGPU framework which is required by PCL to operate properly. The CUDA framework allows the GPU on the NVIDIA graphics card to work in parallel with the CPU on the motherboard, thus allowing faster and more efficient processing when reading in frames. Additional services needed by the Linux operating system include: running the program at startup and handling input from the number pad switch.

7.3 Data Processing Layer

The Data Processing Layer will utilize Point-Cloud and boost libraries at runtime. The Linux operating system will provide services in this layer that monitor the system’s memory and hard drive usage. This layer will also store and retrieve PCL frames which will require the Linux operating system to perform the appropriate actions.

7.4 Communication Layer

The Communication Layer will utilize the Linux operating system networking stack to establish a connection to the ROS node via Ethernet. There will be no dependencies on any Point-Cloud or boost libraries in this layer.
8. **Inter-Subsystem Data Flow**

![Diagram](image)

**Figure 8-1: Inter-Subsystem Data Flow**

- **Camera**
- **Switch**
- **RGB-D Camera Interpreter**
- **I/O Interface Layer**
- **Signal Interpreter**
- **Store Frames**
- **Data Processing Layer**
- **Controller**
- **Data Storage**
- **Data Processing**
- **Localization**
- **Mapping**
- **Communication Layer**
- **Pack Transformation Matrix**
- **Command Interpreter**
- **ROS Packet**
8.1 Overview

This section describes how data flows between each subsystem in each layer. All data elements will be labeled with a single letter prefix to describe what layer it is in; a number within the layer to create a distinction between data elements; and the letter “a” or “b” to show direction in the case of a multi-directional data flow. If the data element is labeled with an "I", it is in the I/O Interface Layer. If the data element is labeled with a "D", it is in the Data Processing Layer. If the data element is labeled with a "C", it is in the Communication Layer.

8.2 Inter-Subsystem Data Element Descriptions

The Inter-Subsystem Data Element Descriptions section describes the data flow to and or from subsystems of the Electric-Eye module that are shown in Figure 8-1.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Data Element</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>I/O Interface</td>
<td>I1a</td>
<td>Sends commands to camera (sleep/wake/send frames)</td>
</tr>
<tr>
<td>I/O Interface</td>
<td>I1b</td>
<td>Raw data from camera</td>
</tr>
<tr>
<td>I/O Interface</td>
<td>I2</td>
<td>Raw data from switch</td>
</tr>
<tr>
<td>I/O Interface</td>
<td>I3</td>
<td>Sends point cloud data to the Data Processing Layer</td>
</tr>
<tr>
<td>I/O Interface</td>
<td>I4</td>
<td>Sends switch mode to Data Processing Layer (read/map/localize)</td>
</tr>
<tr>
<td>Data Processing</td>
<td>D1</td>
<td>Camera command from Data Processing Layer (sleep/wake/send frames)</td>
</tr>
<tr>
<td>Data Processing</td>
<td>D2a</td>
<td>Point cloud data formatted into a frame</td>
</tr>
<tr>
<td>Data Processing</td>
<td>D2b</td>
<td>Signal that the process of reading in camera frames should begin</td>
</tr>
<tr>
<td>Data Processing</td>
<td>D3a</td>
<td>Point cloud data formatted into a frame</td>
</tr>
<tr>
<td>Data Processing</td>
<td>D3b</td>
<td>Sends a request for a frame to use in the localization process or sends sleep command</td>
</tr>
<tr>
<td>Data Processing</td>
<td>D4</td>
<td>Stores frame in data storage</td>
</tr>
<tr>
<td>Data Processing</td>
<td>D5</td>
<td>Sends program control in the form of the formatted switch value 1</td>
</tr>
<tr>
<td>Data Processing</td>
<td>D6</td>
<td>Sends program control in the form of the formatted switch value 2</td>
</tr>
<tr>
<td>Data Processing</td>
<td>D7</td>
<td>Sends program control in the form of the formatted switch value 3 or the sleep command value 4 for wake up or 5 to go to sleep</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Data Processing</td>
<td>D8a</td>
<td>Sends either a frame back to the data storage when finished with it in the mapping process or the finished map</td>
</tr>
<tr>
<td>Data Processing</td>
<td>D8b</td>
<td>Retrieves a frame for the mapping process</td>
</tr>
<tr>
<td>Data Processing</td>
<td>D9a</td>
<td>Sends back the map for storage when done using it for localizing</td>
</tr>
<tr>
<td>Data Processing</td>
<td>D9b</td>
<td>Retrieves the map for the localization process</td>
</tr>
<tr>
<td>Data Processing</td>
<td>D10</td>
<td>Sends transformation matrix to Communication Layer</td>
</tr>
<tr>
<td>Communication</td>
<td>C1</td>
<td>Interpreted ROS sleep message sent to Data Processing Layer</td>
</tr>
<tr>
<td>Communication</td>
<td>C2</td>
<td>Packages a transformation matrix and sends it to ROS</td>
</tr>
<tr>
<td>Communication</td>
<td>C3</td>
<td>Sleep command from ROS</td>
</tr>
</tbody>
</table>

**Table 8-1: Inter-Subsystem Data Element Descriptions**

### 8.3 Producer-Consumer Relationship

The producer is the subsystem that generates the data element. The consumer is the subsystem that receives the data element. The interactions are shown in the intersection of the producer row and the consumer column. The intersecting cell is labeled with the data element name given, and described, in Table 8-1.
# Architectural Design Specification

## Table 8-2: Producer-Consumer Relationship

<table>
<thead>
<tr>
<th>Producer-Consumer Relationship</th>
<th>Consumer Subsystem</th>
<th>I/O Interface</th>
<th>Data Processing</th>
<th>Communication</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>I/O Interface</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Camera</td>
<td>-</td>
<td>I1b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RGB-D Camera Interpreter</td>
<td>I1a</td>
<td>-</td>
<td></td>
<td>I3</td>
</tr>
<tr>
<td>Switch</td>
<td>-</td>
<td>I2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Signal Interpreter</td>
<td>-</td>
<td>-</td>
<td></td>
<td>I4</td>
</tr>
<tr>
<td><strong>Data Processing</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Store Frames</td>
<td>-</td>
<td>D4</td>
<td>D2b</td>
<td></td>
</tr>
<tr>
<td>Data Storage</td>
<td>-</td>
<td>-</td>
<td>D8b</td>
<td>D9b</td>
</tr>
<tr>
<td>Camera Handler</td>
<td>D1</td>
<td>D2a</td>
<td>-</td>
<td>D3a</td>
</tr>
<tr>
<td>Controller</td>
<td>D5</td>
<td>-</td>
<td>D6</td>
<td>D7</td>
</tr>
<tr>
<td>Mapping</td>
<td>D8a</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Localization</td>
<td>D9a</td>
<td>D3b</td>
<td>-</td>
<td>D10</td>
</tr>
<tr>
<td><strong>Communication</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pack Transformation Matrix</td>
<td>-</td>
<td>C2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Command Interpreter</td>
<td>-</td>
<td>C1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROS</td>
<td>C3</td>
<td>-</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Legend:**
- **Producer-Consumer Relationship:** Describes the flow of data and control signals in the system.
- **I/O Interface:** Includes input (I) and output (O) interfaces.
- **Data Processing:** Includes processing stages such as frame storage and data handling.
- **Communication:** Includes communication and control signals.
9. Requirements Mapping

9.1 Overview

Requirements mapping will be used in order to ensure that all key requirements are satisfied in the Architectural Design. The figure below displays every key requirement and which architectural layer that it is associated with. High level requirements that are part of packaging are not included.

<table>
<thead>
<tr>
<th>Requirement Number</th>
<th>Requirement Name</th>
<th>I/O Interface Layer</th>
<th>Data Processing Layer</th>
<th>Communication Layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>ROS Compatible</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>3.2</td>
<td>Mapping</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>3.3</td>
<td>Localization</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>3.7</td>
<td>RGB-D Camera</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>3.8</td>
<td>On-Board Storage</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>8.3</td>
<td>Mode Switch</td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

Table 9-1: Requirements Mapping
10. Testing Considerations

10.1 Overview

The system architecture will be tested by team Power Vision to verify that the Electric-Eye fulfills all of the requirements laid out in the System Requirements Specification and this Architectural Design Specification. Each component and subsystem is designed to work independently and this independence allows each to be tested individually.

10.2 Testing Approach

Team Power Vision will test all of the subsystem components independently to verify correct output before integration into the rest of the component. After all subsystems of a component are complete, that component will be tested as a whole for correct results. This process will be repeated for each component in the architecture.

Once all components are created, Power Vision will integrate and perform integration testing to ensure all components still work as expected after being integrated into one system.

Team Power Vision will check to make sure all outputs are valid and the data flow is as expected by the architecture.

10.3 I/O Interface Layer

The I/O Interface Layer must be able to gather input from the hardware devices such as the RGB-D camera and the physical switch. Team Power Vision will ensure that this layer is receiving the signals from the hardware correctly.

10.3.1 RGB-D Camera Interpreter

Verify that the interpreter is able to receive the output from the physical RGB-D camera and parse it correctly.

10.3.2 Signal Interpreter

Verify that the interpreter is able to distinguish the position of the physical switch and send out the correct output.
10.4 Data Processing Layer

This layer must be able to parse RGB-D camera data into a useful map and localize the Electric-Eye’s position within the map. Based on the switch’s position, this layer will either store frames, map, or localize the Electric-Eye. It will also respond to ROS requests brought up by the Communication Layer.

10.4.1 Camera Handler

Verify that the Camera Handler is receiving the frames and sending them out to the correct subsystem.

10.4.2 Store Frames

Verify that Store Frames is receiving frames from the Camera Handler and sending them to the hard disk. Also verify that it responds to activation and shutdown from the Controller subsystem.

10.4.3 Mapping

Verify that the Mapping subsystem is pulling frames from the hard disk and is converting them to a 3-D map, and storing that map back to the hard disk. Power Vision will also verify that the Mapping subsystem also responds to activation and shutdown from the Controller subsystem.

10.4.4 Localization

Verify that Localization is pulling the map from the hard disk, using the map to localize the Electric-Eye, and sending the transformation matrix to the Communication Layer. Power Vision will also verify that the Localization subsystem is responding to activation and shutdown from the Controller subsystem.

10.4.5 Controller

Verify the controller is receiving the switch position from the Signal Interpreter and is activating and shutting down the corresponding subsystem.

10.5 Communication Layer

This layer must be able to pack and unpack messages to and from ROS, passing them on to the Data Processing Layer or sending them out of the module.

10.5.1 Pack Transformation Matrix

Verify that the Pack Transformation Matrix subsystem can receive the transformation matrix from the Data Processing Layer and send it out as a ROS message out of the module.
10.5.2 Command Interpreter

Verify that the Command Interpreter can receive ROS packets, unpack them and send the command up to the Data Processing Layer.
11. Glossary

11.1 Robot Operating System (ROS)

The Robot Operating System, or ROS, is an operating system that can be installed on top of Linux or Microsoft Windows. This operating system provides libraries and tools to help software developers create robot applications. Some of the features that ROS offers are hardware abstraction, device drivers, message-passing and more. More information about ROS can be found at: www.ros.org

11.2 Point Cloud Library (PCL)

Point Cloud Library, or PCL, is a large scale open project for 2D/3D image and point cloud processing. PCL contains many algorithms that are useful for 3D image processing such as noise filtering. PCL is open source software that is free for commercial and research use. More information about PCL can be found at: www.pointclouds.org

11.3 Transformation Matrix (6DOF)

The transformation matrix, or 6DOF, is a term used throughout the document. The transformation matrix is a set of data that will be sent to ROS from the Electric-Eye which will include the pitch, roll, yaw, x, y, and z data of the current location that the robot is located in respect to the camera.

11.4 RGB-D Camera

A special camera that reads frames like a normal camera but can represent the depth of the image using color. (Red means near while blue means far). Thus this type of camera can be very useful in helping robot navigate. More information can be found at: http://www.hizook.com/blog/2010/03/28/low-cost-depth-cameras-aka-ranging-cameras-or-rgb-d-cameras-emerge-2010

11.5 OpenNI

OpenNI is an open source software development kit used for the development of 3D sensing libraries and applications. More information can be found at: http://www.openni.org/
11.6 **Point Cloud**

Point Cloud is a set of vertices in 3D space that include x, y, and z coordinates. It is used by 3D sensing cameras to capture and store 3D images. More information can be found at: http://en.wikipedia.org/wiki/Point_cloud

11.7 **Point Cloud Frame**

A camera screen shot that holds point cloud data of an area.

11.8 **Mapping**

Mapping is the process of collecting data of one’s environment and using it to construct a type of map that can be used for localization. More information can be found at: http://en.wikipedia.org/wiki/Robot_localization

11.9 **Localization**

Localization is the process of determining one’s position and direction in a given environment. More information can be found at: http://en.wikipedia.org/wiki/Navigation

11.10 **ROS Node**

A node is processes that are used to help control robot functionality by communicating with other nodes. More information can be found at: http://www.ros.org/wiki/Nodes