



**Amsterdam University  
of Applied Sciences**

ROVER RESCUE SYSTEM

---

# Documentation

---

*Author:*

Christiaan van Arum

Raphaël Bunck

Nino van Galen

Martijn Vegter

*Student Number:*

500778983

500774349

500790589

500775388

January 7, 2019

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Installation Guide</b>	<b>3</b>
2.1	Pre-installation . . . . .	3
2.2	Headless setup procedure . . . . .	3
2.3	Installation . . . . .	4
2.4	Manual Installation . . . . .	4
<b>3</b>	<b>Usage Guide</b>	<b>5</b>
<b>4</b>	<b>Definition of Ready</b>	<b>6</b>
<b>5</b>	<b>Definition of Done</b>	<b>7</b>
5.1	Feature . . . . .	7
5.2	Sprint . . . . .	7
<b>6</b>	<b>Class Diagrams</b>	<b>8</b>
6.1	Project Overview . . . . .	8
6.2	Controller - Steam Controller . . . . .	9
6.3	Controller - DPad . . . . .	10
6.4	Controller - Dual Axis . . . . .	10
6.5	Controller - Simple Button . . . . .	11
6.6	Controller - Touch . . . . .	12
6.7	Controller - Single Axis . . . . .	12
6.8	Motor . . . . .	13
6.9	Nervi - Camera Mount . . . . .	14
6.10	Nervi - Rotary Encoder . . . . .	14
6.11	Nervi - Servo . . . . .	15
6.12	Nervi - Ultrasonic . . . . .	16

# 1 Introduction

In the (near) future robots will be more and more part of our daily life. Even more than they are already part of society today. Industrial robots are nowadays very common, the use of drones by military and civilians triggers discussion and science is creating robots to take care of people who need help. Functioning more or less autonomous requires that such machines have to be robust, take their own decisions and operate in a safe way.

Thus we created a robot that helps to rescue people trapped in a collapsed building.

The user is able to control the robot using the combination of a controller, specifically a Steam Controller, and a mobile phone as display used by the Google Cardboard. Of course, being a robot, the robot itself wants to do as much as possible. The degree of autonomy of the robot has been focused on assisting the user as much as possible.

## 2 Installation Guide

This chapter gives a step by step tutorial to get the RSS up and running. After completing the installation, continue reading the usage guide.

### 2.1 Pre-installation

Before installing the Rover software, make sure you have the following software up and running:

- A RaspberryPi B2 (RPI) with newest version of Raspbian.
- An Active internet connection on the RPI.
- An open SSH, see setup below.

### 2.2 Headless setup procedure

- Step 1: Create an empty file. You can use Notepad on Windows or TextEdit to do so by creating a new file. Just name the file `ssh`. Save that empty file and dump it into boot partition (microSD).
- Step 2: Create another file name **wpa\_supplicant.conf**. This time you need to write a few lines of text for this file. For this file, you need to use the **FULL VERSION** of `wpa_supplicant.conf`. Meaning you must have the 3 lines of data namely `country`, `ctrl_interface` and `update_config`.

```
}
country="Your country"
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1

network={
    ssid="your_real_wifi_ssid"
    scan_ssid=1
    psk="your_real_password"
    key_mgmt=WPA-PSK
}
```

## 2.3 Installation

To install Rover and its dependencies from your RPI shell prompt, use the following command:

```
$ wget -q https://git.io/fx0ko -O /tmp/rover && bash /tmp/rover
```

## 2.4 Manual Installation

Update your system package list:

```
$ sudo apt-get -update -y
```

Upgrade all you installed packages to their latest version:

```
$ sudo apt-get dist-upgrade -y
```

To download and install newest version of Node.js, use the following command:

```
$ curl -sL https://deb.nodesource.com/setup_8.x | sudo -E bash -
```

Now install it by running:

```
$ sudo apt-get install nodejs -y
```

To compile and install native addons from npm you also need to install build tools:

```
$ sudo apt-get install build-essential -y
```

To download the Rover code you need to install git:

```
$ sudo apt-get install git -y
```

Clone 'Scriptor' using git:

```
$ git clone https://github.com/RescueOnWheels/Scriptor.git
```

Clone 'Rover' using git:

```
$ git clone https://github.com/RescueOnWheels/Rover.git --recursive
```

Open the 'Scriptor' folder and start using the Rover:

```
$ cd Scriptor && ./start.sh all
```

### 3 Usage Guide

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

## 4 Definition of Ready

1. Must have clear description.
2. Must have a milestone.
3. Must have a estimate for the burndown chart.
4. Issues from the same type must be organised in epic's.
5. Relevant issues must be done before continuing.
6. Previous sprint issues must be done before new sprint issues. (Except icebox issues)
7. Issues can't be large and vague, should be split in multiple issues.

## 5 Definition of Done

### 5.1 Feature

1. DoD of each single User story, included in the Sprint are met
2. “To do’s“ are completed
3. All unit tests passing
4. Product backlog updated
5. Project deployed on the test environment identical to production platform
6. Tests on devices/browsers listed in documentation passed
7. Tests of backward compatibility passed
8. The performance tests passed
9. All bugs fixed
10. Sprint marked as ready for the production deployment by the Product Owner

### 5.2 Sprint

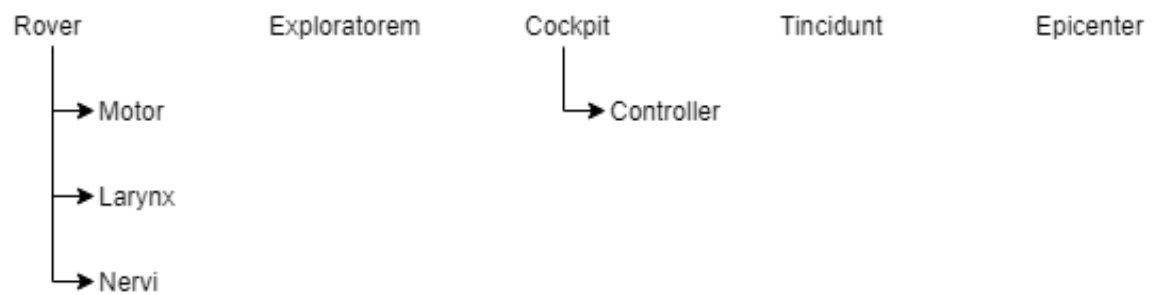
1. Code Complete
2. Environments are prepared for release
3. All unit & functional tests are green
4. All the acceptance criterias are met
5. QA is done & all issues resolved
6. All “To Do“ annotations must have been resolved
7. OK from the team: UX designer, developer, software architect, project manager, product owner, QA, etc.
8. Check that no unintegrated work in progress has been left in any development or staging environment.
9. Check that TDD and continuous integration is verified and working



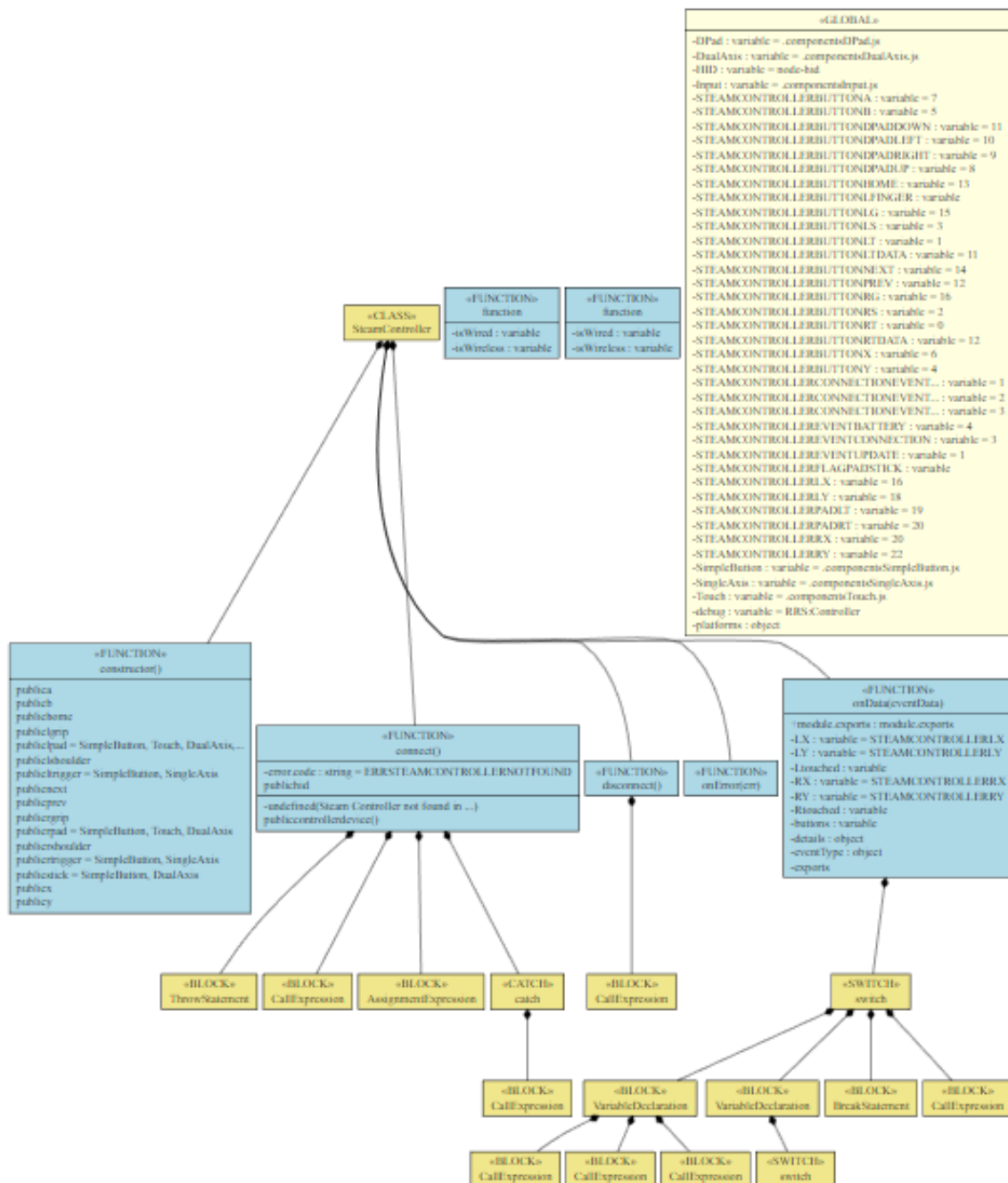
## 6 Class Diagrams

The class diagrams are a visual representation on how the RRS has been structured. The project overview given below shows which modules are submodules from the overhanging modules. In the UML diagrams below there has been given a precise and complete overview of the class diagrams per module. These diagrams have been made using a generating tool in a command line interface.

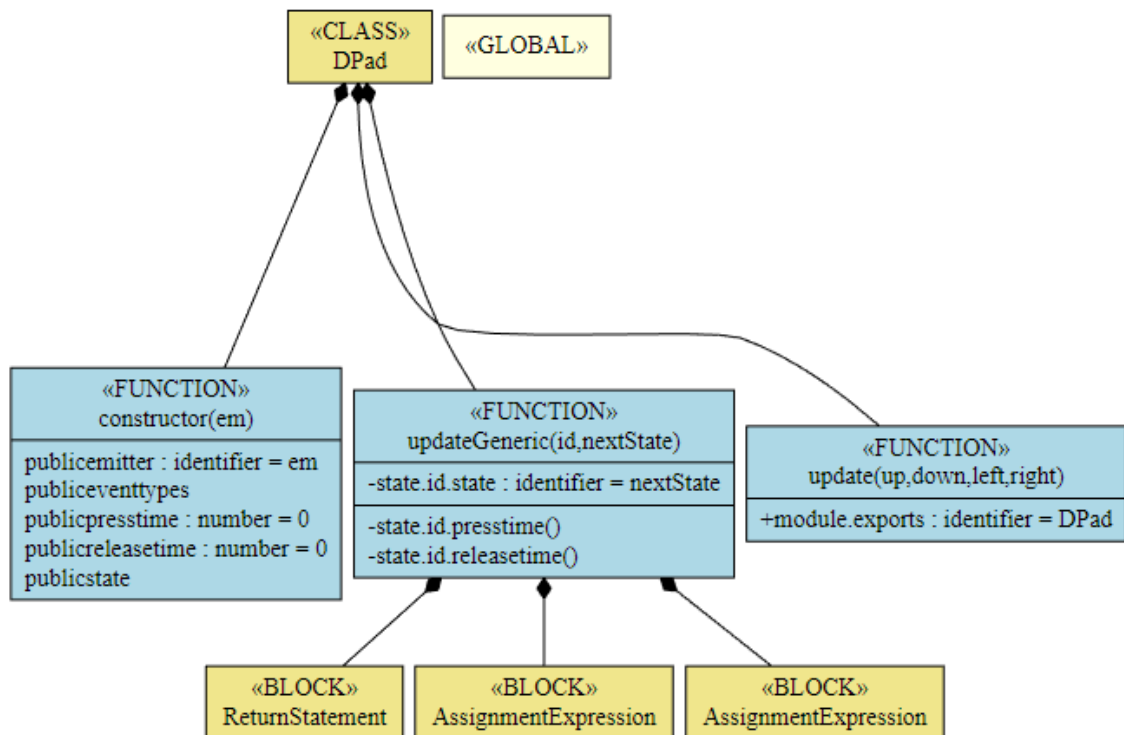
### 6.1 Project Overview



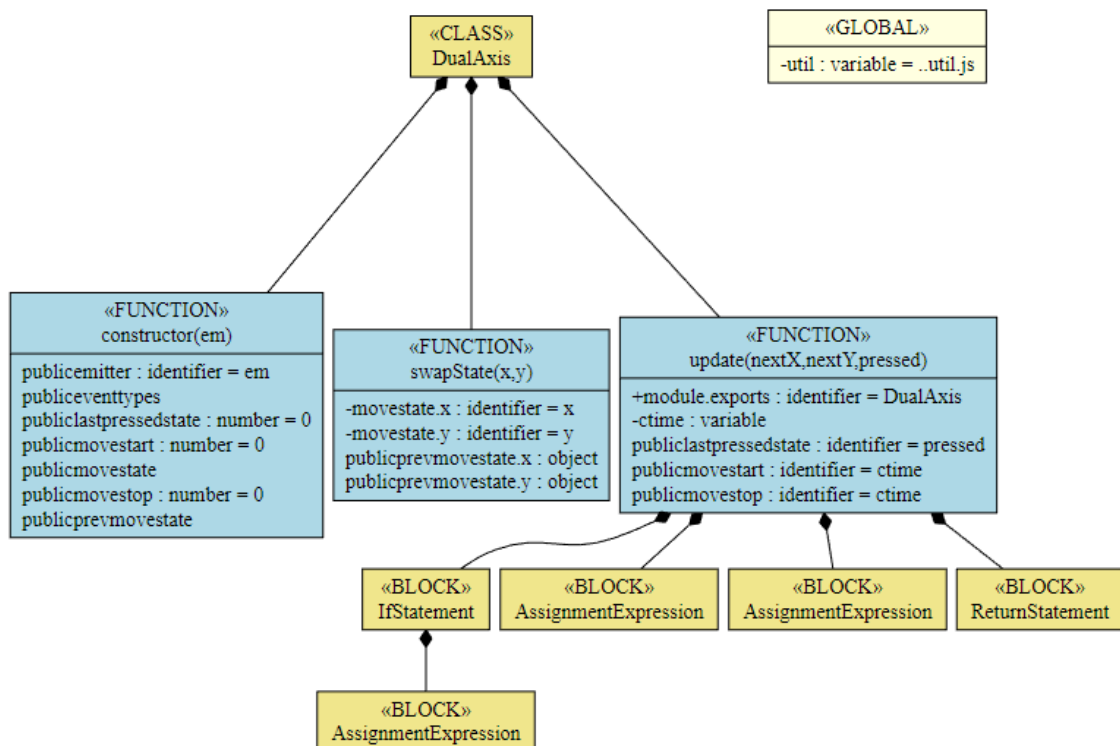
## 6.2 Controller - Steam Controller



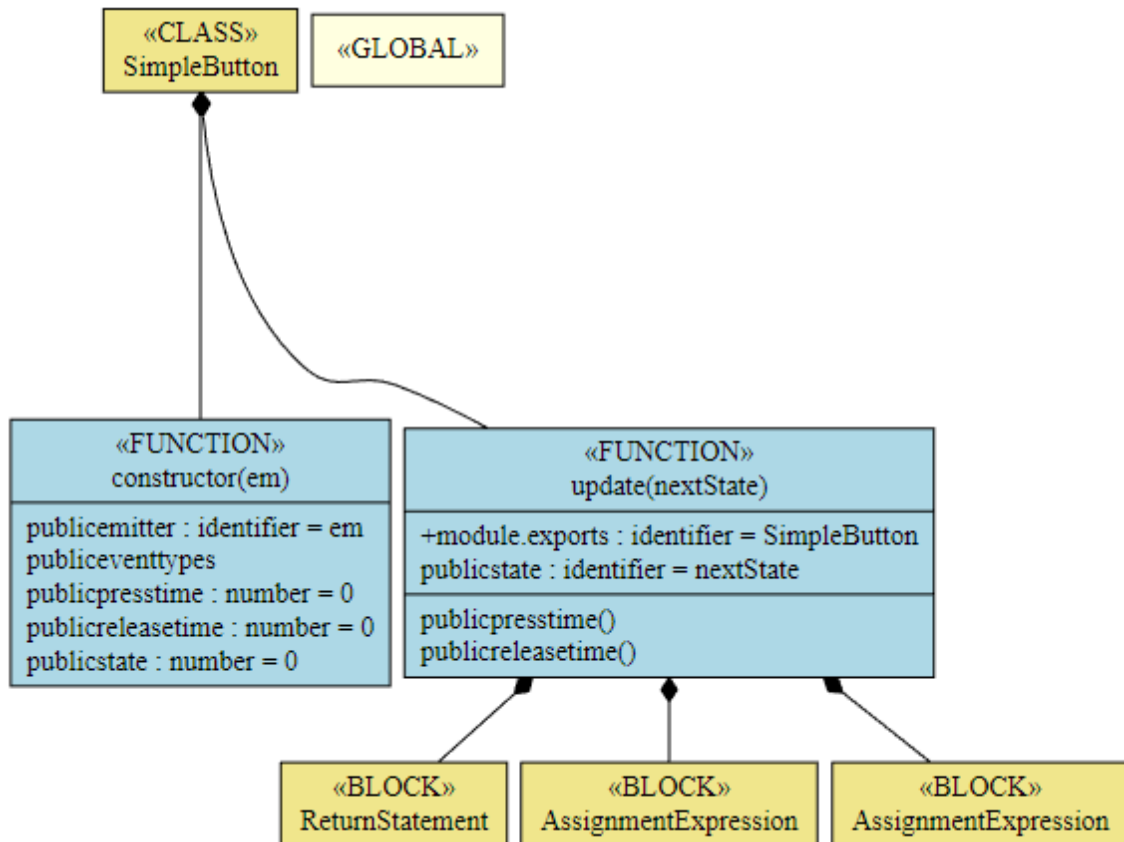
### 6.3 Controller - DPad



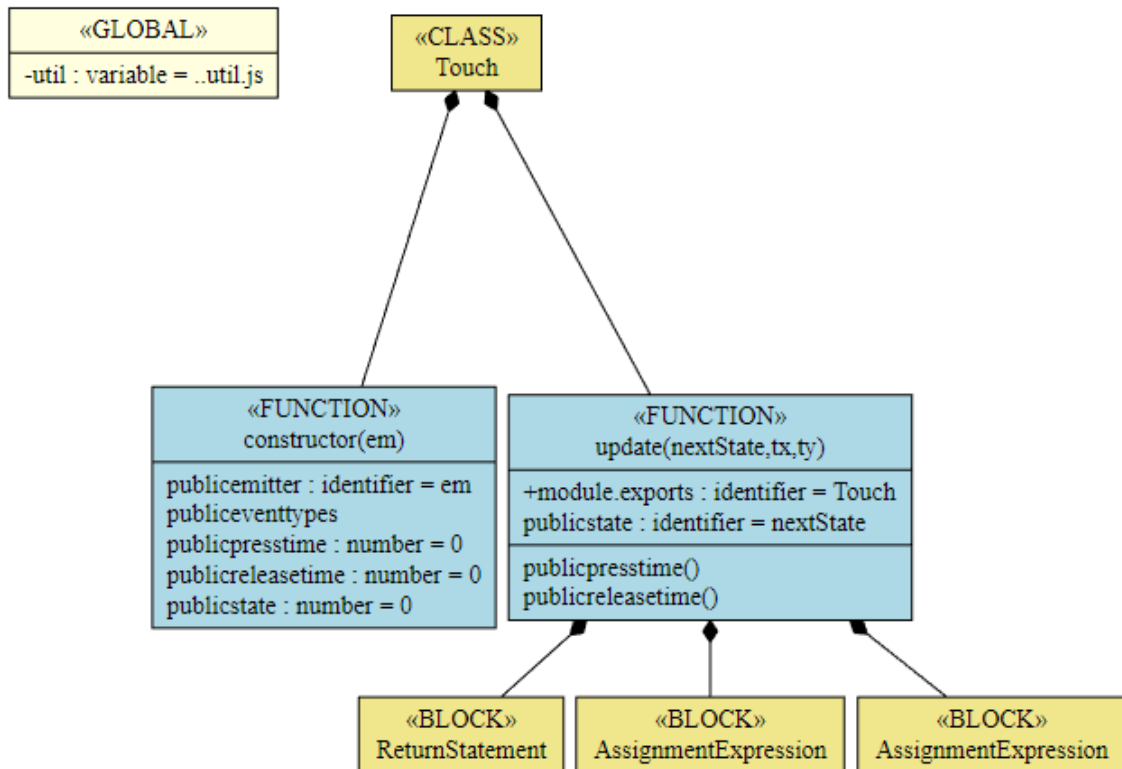
### 6.4 Controller - Dual Axis



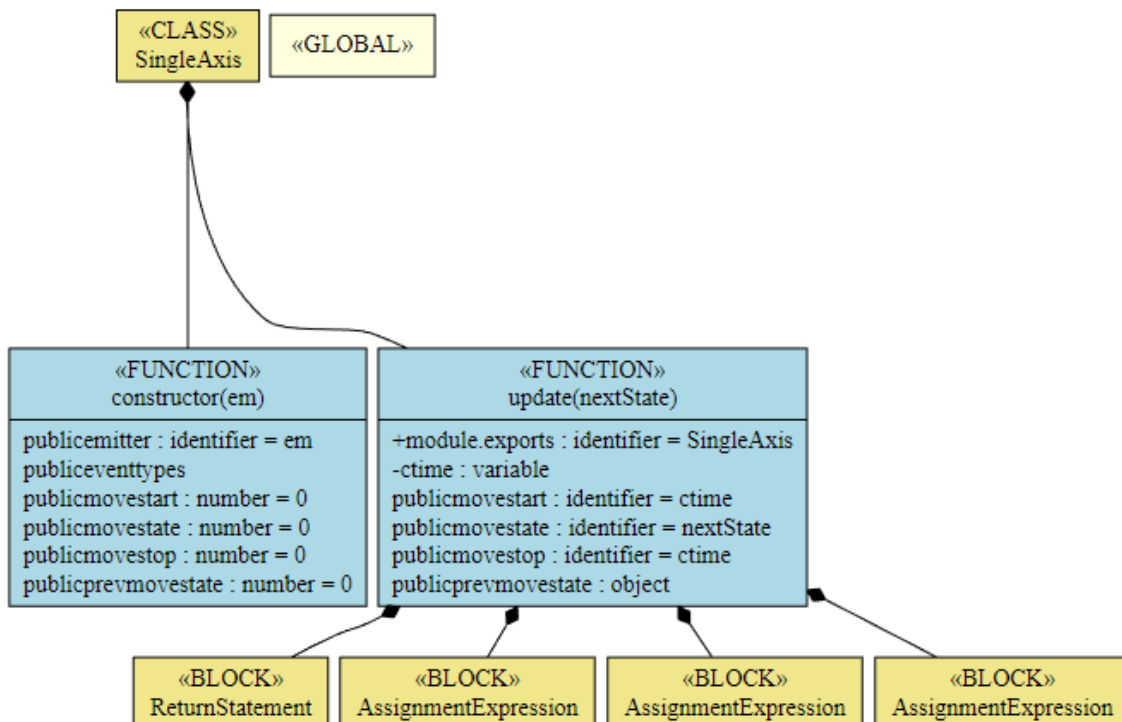
## 6.5 Controller - Simple Button



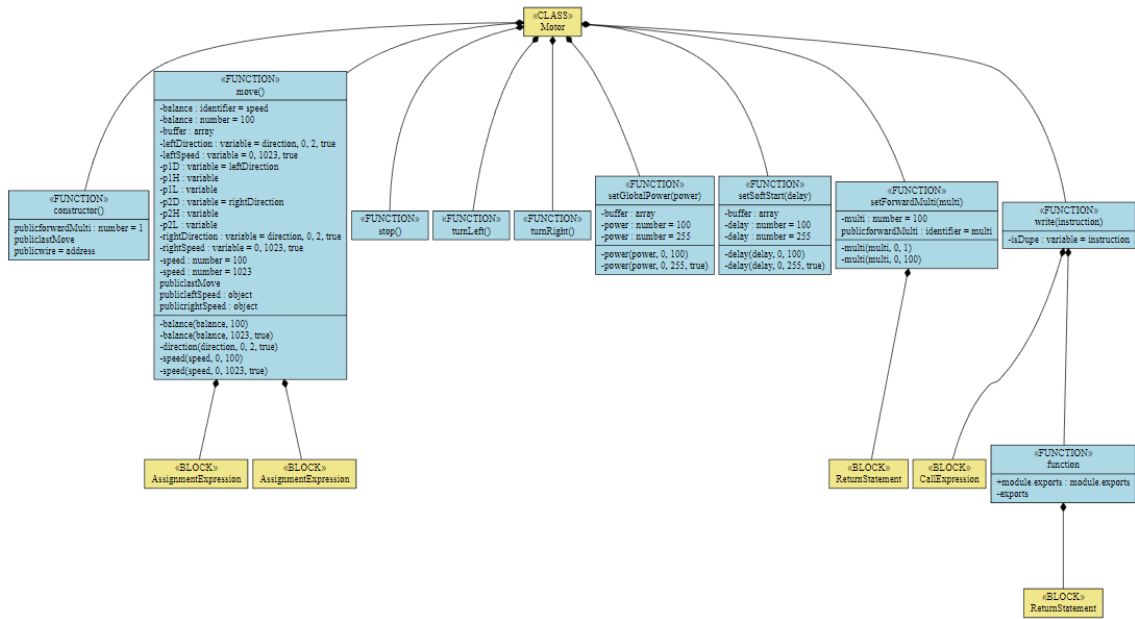
## 6.6 Controller - Touch



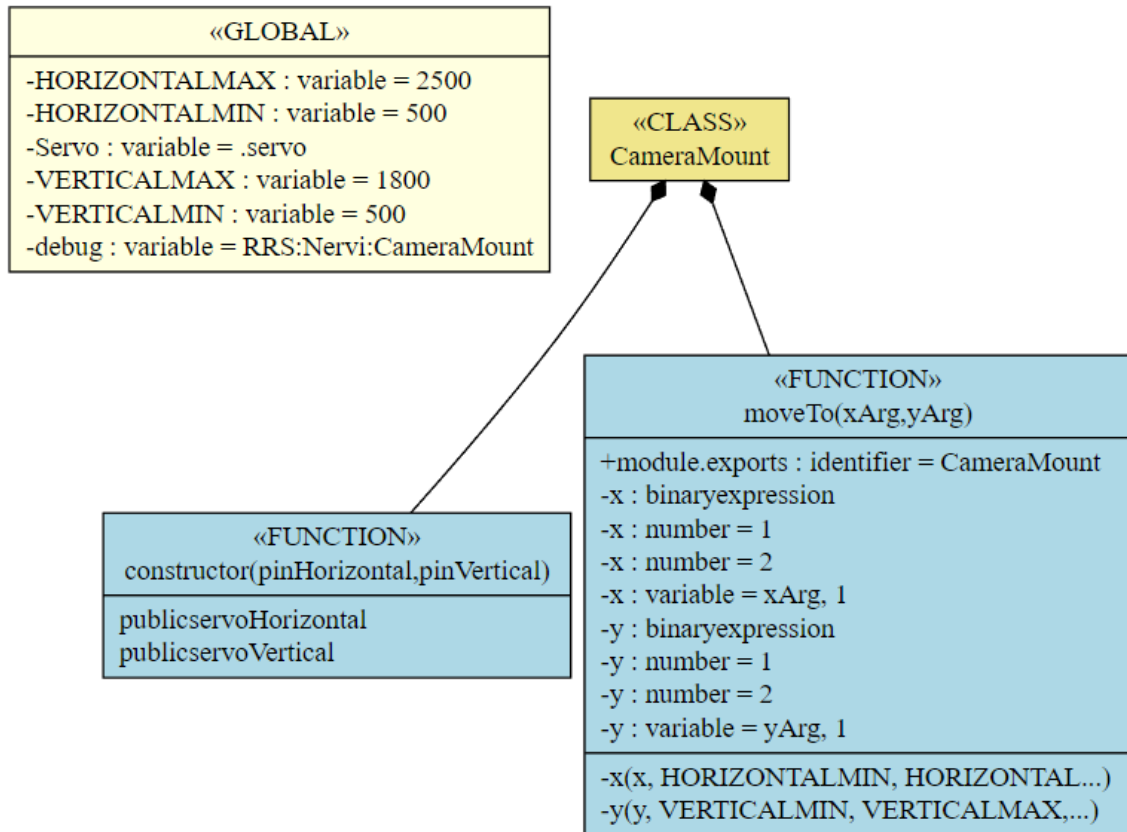
## 6.7 Controller - Single Axis



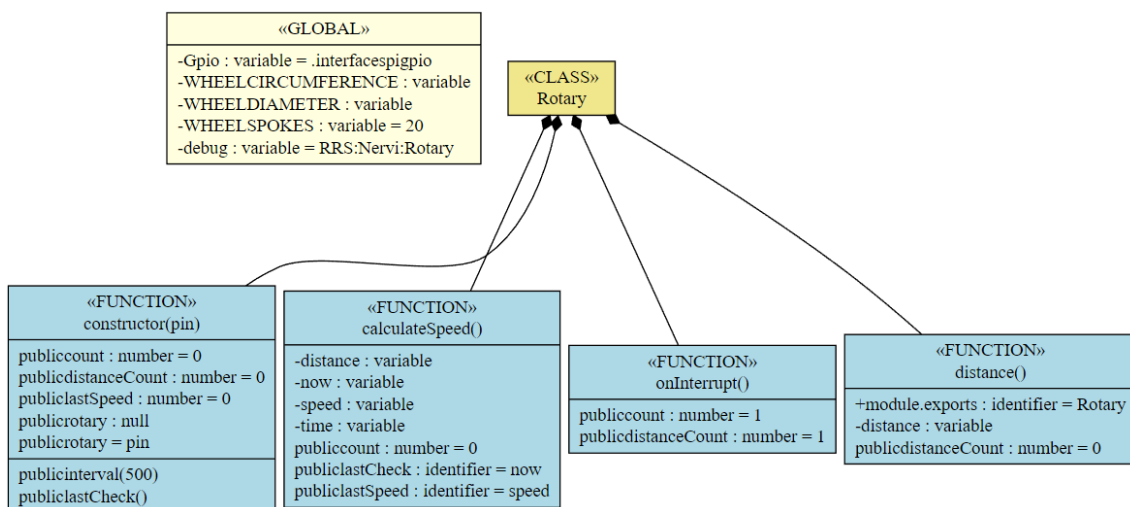
## 6.8 Motor



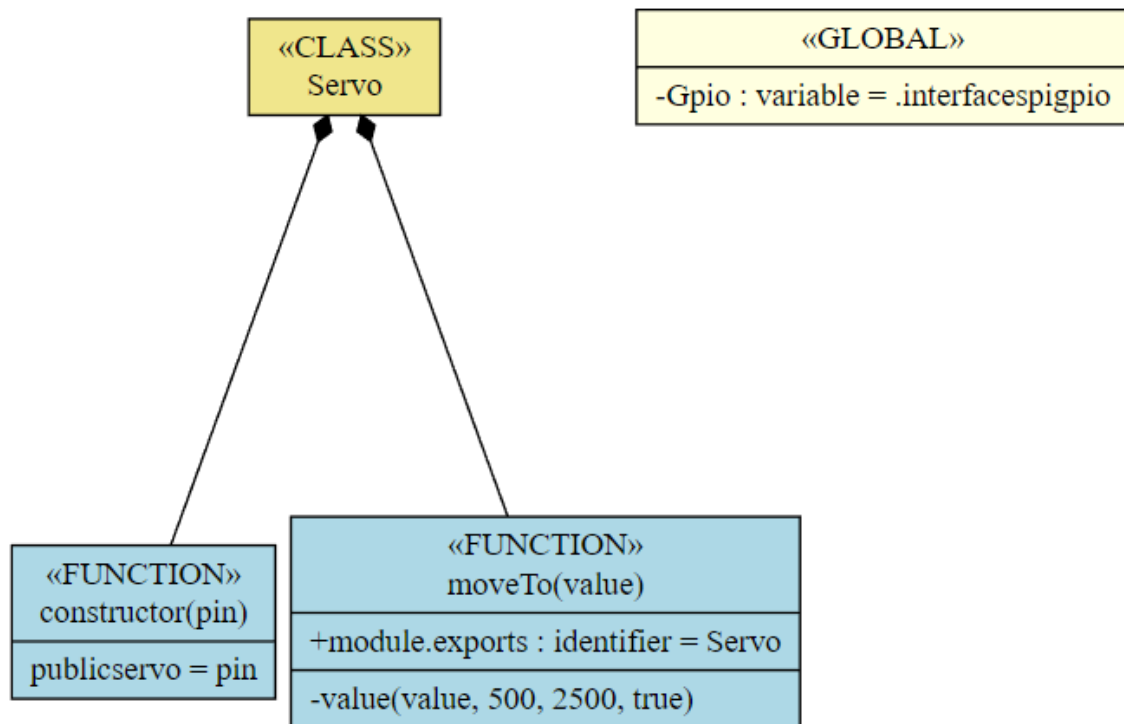
## 6.9 Nervi - Camera Mount



## 6.10 Nervi - Rotary Encoder



## 6.11 Nervi - Servo





## 6.12 Nervi - Ultrasonic

