Mar 22, 2013

Team 19 Design Document

**Laboratory # 5: Design**

**Morgan, Laura**

**Miaw, Jireh**

**Hauser, Steven**

**Dworak, Catherine**

**Bertoglio, David**

***Work Product***

**Description of the design of the robot on-board software, including high level description, UML class and sequence diagrams, state diagram, concurrent structure, and class interfaces in Java**

***Document Revision Information***

**3/22/2013 – Design Document Created**

**Approval Sheet**

**All group members whose names are listed below approve of the document and contributed fairly.**

**Morgan, Laura**

**Miaw, Jireh**

**Hauser, Steven**

**Dworak, Catherine**

**Bertoglio, David**

**Pledge**

**On my honor, as a student, I have neither given nor received unauthorized aid on this assignment.**

**Morgan, Laura**

**Miaw, Jireh**

**Hauser, Steven**

**Dworak, Catherine**

**Bertoglio, David**

Contents

High-level system architecture 4

Activator 4

MessageHandler 4

Driver 4

Static structure 5

Object interaction structure 5

Concurrent structure 5

Class interfaces 5

# High-level system architecture

The robot on-board software will be object-oriented. It will consist of 3 classes, Activator, Driver, and MessageHandler. The Activator will contain instances of Driver and MessageHandler. Driver and MessageHandler will not be able to access each others’ fields and methods directly; any interaction between Driver and MessageHandler must go through the Activator class.

## Activator

The Activator class contains the main method. This class is the only one that deals with the Bluetooth connection. It will contain fields and methods to create the connection and check if the connection is there. It creates 3 threads: timer, read, and output. The timer thread is used to determine how much time has elapsed between sending the last message from the on-board system and receiving an acknowledgment from the base computer. The input and output threads are the channels to send and receive messages from the base computer.

The activator receives messages from the base computer, then sends them to the MessageHandler class for decoding, then channels the usable message to the Driver class to implement the required action.

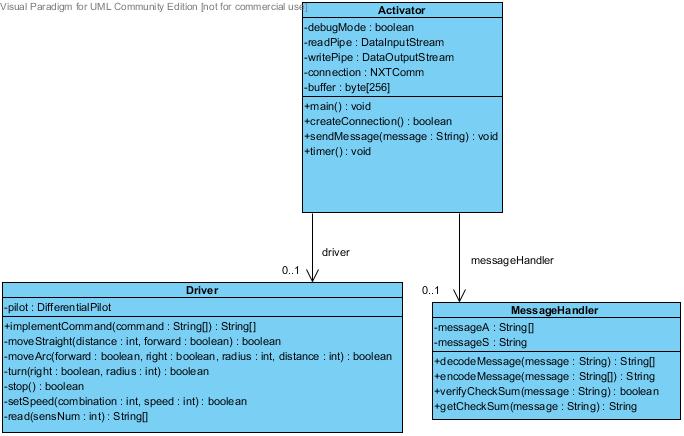
## MessageHandler

The MessageHandler class has one purpose: to deal with messages. It will be capable of decoding a message from the base station, validating the checksum, endcoding a new message to send to the base station, and creating a checksum for the new message. It will take messages in the format designated by the Communications Protocol and transform them into a format that the Driver can use to perform actions. On the reverse, it will take messages (acknowledgments or sensor data), and put them into the communications protocol format, so they can be sent over the Bluetooth channel from the Activator class. All encoded and decoded messages are passed back to the Activator class, and from there are sent to their final destination.

## Driver

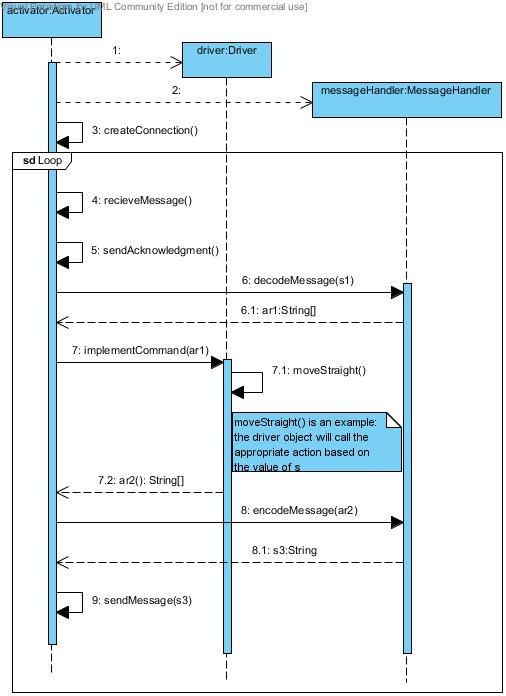
The Driver class is in charge of performing robot actions. It will contain an instance of the Differential Pilot Object from Lejos, which contains classes that control robot movement, such as setting the speed and rotating. The Driver class will contain a method for each action the robot should be able to perform: moveStraight, moveArc, turn, stop, setSpeed, read, and noOp. Additionally, it will have a method called implementAction, which will take in a decoded message and call the correct method to perform the required action.

# Static structure

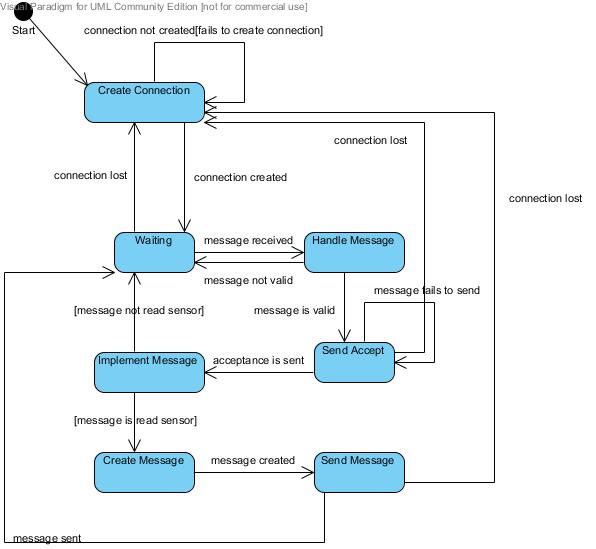


## **Object interaction structure**

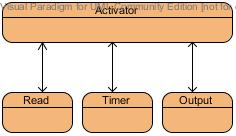
## UML Sequence Diagram



## Finite State Diagram



# Concurrent structure



# Class interfaces