

System Requirements Specification

Cloud⁹

August 23, 2008

Contents

1. Version History	3
2. Introduction	5
2.1 Overall Description and Scope of Software	5
2.2 Purpose of this document	5
2.3 Definitions	6
2.4 References, Definitions and Abbreviations	7
2.5 Developer's Responsibilities	7
3. General Description	8
3.1 Product Perspective	8
3.2 Product Functions Overview	8
3.3 User Characteristics	8
3.4 General Constraints	9
3.5 High-level Software Architecture	9
3.5.1 Client-Server Architecture	10
3.5.2 Interrupt-driven Architecture	10
3.5.3 Event-driven Architecture	10
3.5.4 Layered Architecture	10
3.6 Technical Constraints	10
3.7 Assumptions and Dependencies	11
4. Functional Requirements	13
4.1 Interface Requirements	13
4.1.1 Robot movement information should be relayed to the operator	13
4.1.2 The robot should be able to initially be placed anywhere on the map.	14
4.1.3 Map format	14
4.1.4 Loading a map	15

4.1.5	Emergency stop	15
4.1.6	User-settable "no-go" zones	16
4.1.7	User-settable speed limits	17
4.1.8	Path determination	17
4.1.9	Path determination with user-selectable metrics	18
4.1.10	Manual robot controls	19
4.1.11	Initial scanning of the factory floor	19
4.1.12	Displaying the factory map	20
4.2	Robot Requirements	21
4.2.1	Speed	21
4.2.2	Line following	21
4.3	Robot Requirements	22
4.3.1	Display	22
4.4	Communications Requirements	23
4.4.1	Error Correction	23
4.4.2	Packet Loss	23
4.4.3	Loss of communication	24
4.5	System Requirements	25
4.5.1	Movement	25
4.5.2	Line Mapping	26
5.	Interaction Scenarios	27
5.1	Initial Positioning of the Robot	27
5.2	Initial Scanning of a New Factory	28
5.3	Moving the Robot Around a Factory	28
6.	Appendices	30
6.1	Appendix 1: XML DTD	30
6.2	Appendix 2: Cloud ⁹ Member Information	32
7.	Group Signing Sheet	38

Version History

Version	Changes	Author	Date
0.1	Initial layout	Andrew L	17/08/06
0.2	Added milestones to General Constraints	Andrew B	18/08/06
0.3	Added more Technical Constraints	Andrew B	18/08/06
0.4	Added more Assumptions/Dependencies	Andrew B	18/08/06
0.5	Created Interface Requirements	Andrew B	19/08/06
0.6	Created Robot Requirements	Dave	19/08/06
0.7	Added DTD into appendices	Andrew B	20/08/06
0.8	Created Communications Requirements	Andrew B	20/08/06
0.9	Added XML graphic	Andrew L	20/08/06
0.10	Added to Glossary	Andrew B	20/08/06
0.11	Updated User Characteristics	Andrew L	20/08/06
0.12	Moved a requirement into non-functional	Andrew B	20/08/06
0.13	Added a little more to Jin's CV	Andrew B	20/08/06
0.14	Sorted glossary alphabetically	Andrew B	20/08/06
0.15	Added software architecture section	Andrew B	20/08/06
0.16	Added signing sheet	Andrew B	20/08/06
0.17	Tweaking of robot requirements	Andrew B	20/08/06
0.18	Added scenario	Andrew L	21/08/06
1.0	External draft release	All	21/08/06
1.1	Rewrote/added requirements	Andrew B	22/10/06
1.2	Added two more interaction scenarios	Andrew B	22/10/06
2.0	Final release	All	22/10/06

- Layout document sections & chapters
- Include sample templates for information entry
- Added info to:
 - Introduction
 - General Description

- General Constraints
- Technical Constraints
- Assumptions and Dependencies
- Interface Requirements
- Robot Requirements
- Communications Requirements
- Appendices (Appendix 1: Map DTD)
- Appendices (Appendix 2: Group CVs)
- Group signing sheet
- Scenarios

Introduction

2.1 Overall Description and Scope of Software

This purpose of this project is to create the controlling software for a robot which will traverse a factory floor, automating delivery of components. This software is to be delivered as a working prototype on a fully functional, scale model of the robot and is to be tested on a paper based representation of the factory floor.

The software is to be split into 3 modules of development; a GUI, a communications protocol and robot executed control software.

- The software will include a GUI through which an operator can issue commands to the robot, set destinations and "no-go" zones from which the robot will move along predefined paths.
- A communication protocol will be developed to enable communication between the host computer and the robot via a wireless network.
- The software will also include control software which is to be executed by the robot's microprocessors.

2.2 Purpose of this document

This document formally states the requirements of the users, enumerates the requirements and provides metrics by which the suitability of proposed solutions can be measured.

This document forms the base of the contractual agreement between Cloud⁹ and the client. It provides a clear outline that defines the functionality of the software that is implemented with explicit client agreement.

2.3 Definitions

Throughout this document, various domain-specific terms, acronyms and abbreviations will be used. For clarity, these are accepted to have the meanings as listed below:

802.15: IEEE 802.15 standard for wireless communication. The Zigbee chips that will be used in the Robot System is based on a subset of the 802.15 standard (802.15.4).

ARQ: ARQ (Automatic Repeat-reQuest) is a system of communications protocols. It handles cases of lost and erroneous data.

Bootloading: Bootloading is a process whereby the operating system is started on a computer system.

C: C is a popular programming language that started development in the 1970s.

DTD: A DTD (Document Type Description) is a file that specifies certain attributes of a standardised XML file.

Flash: Flash is a type of re-writeable memory that holds its data when it loses power.

GUI: Graphical User Interface

IEEE: The IEEE (Institute of Electrical and Electronics Engineers) is a professional organisation that is also involved in creating standards across a broad range of electrical-based areas.

Java: Java is an object-orientated programming language created by Sun Microsystems.

LED: Light Emitting Diode

Make: Make is a software utility that aids in the conversion of files from one kind to another. For example, from source code into compiled machine code.

NGZ: A "no-go" zone (NGZ) is a section of the map that the robot is not allowed to enter. This is specified by the user on the controller software.

PIC: Programmable Integrated Circuit

RAM: RAM (Random Access Memory) is a type of memory commonly used for holding program data. It will lose its contents when the power source is removed.

robot: the physical Robot

RXTX: RXTX is a library providing serial and parallel communications support for Java.

SRS: System Requirements Specification

UART: A UART (Universal Asynchronous Receiver-Transmitter) device is used to translate between parallel and serial streams of data.

Unix: Unix is an operating system that started development in the 1960s. We consider operating systems such as Linux to be equivalents to Unix in terms of this project.

USB: A USB (Universal Serial Bus) interface is a standardised and common interface on many desktop computers and is used for a large range of devices.

XML: XML (eXtensible Markup Language) is a flexible general purpose language for specifying different types of data.

2.4 References, Definitions and Abbreviations

Throughout this document, various other documents and sources are referenced in abbreviated form. These sources are as follows:

SRS	This document	MC1	Meeting-Client-060808
CRSP(REF)	Correspondence (with Reference)	MC2	Meeting-Client-060815
		WS3	Week 3 Summary
		PD	Project Description

2.5 Developer's Responsibilities

Cloud⁹ is responsible for delivering all three parts of the software system as outlined in the description. The components of the software will be delivered as per the timeline outlined in the constraints section of the SRS. Cloud⁹ will also demonstrate suitability of the software and physical use thereof.

General Description

3.1 Product Perspective

This product does not interface with any existing software in the environment. There is one primary interface, a GUI on the host computer, for controlling the robot. The host program will also read and store XML formatted files on the file system on the host computer.

3.2 Product Functions Overview

Host Controller

1. Read/Store map data
2. Robot control
3. Auto-route robot along map
4. Set "no-go" zones in map
5. Load/Unload robot from trailer
6. Fine manual control of robot

3.3 User Characteristics

The expected users of the system will be nominated operators selected from the factory staff. Each operator will need training in the use of the GUI interface, including but not limited to, loading the factory map, initial positioning of the robot, positioning of the robot in the event of a power failure, manual control of the robot, setting tasks and no-go zones. An absolute minimum requirement for certification to operate the system is training in the safe operation of the robot.

The nominated operator is expected to have basic computer literacy prior to undertaking the training session.

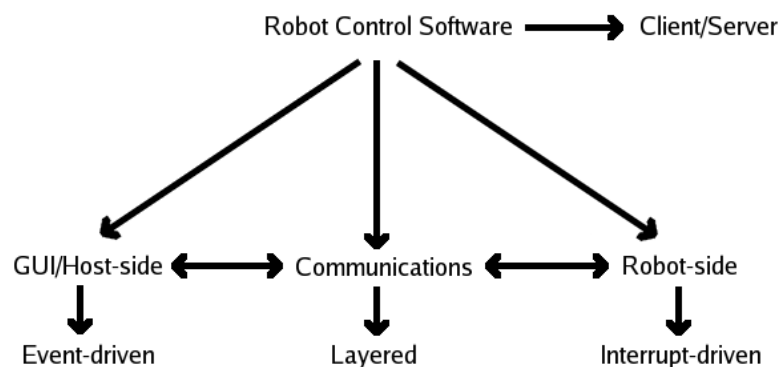
General factory staff will need to be trained in safe work practices in the vicinity of the robot. This remains the responsibility of the client.

3.4 General Constraints

- 21/08/2006: Draft of this document due
- 22/08/2006: First GUI milestone (mock-up)
- 22/08/2006: First radio milestone (simple communication)
- 5/09/2006: First robot milestone (LEDs)
- 12/09/2006: Second robot milestone (basic movement)
- 5/09/2006: Second GUI milestone (maps)
- 12/09/2006: Third GUI milestone (paths)
- 12/09/2006: Second radio milestone (ARQ)

3.5 High-level Software Architecture

The sub-systems that make up the robot system are identified below. We have also outlined the sub-system control for each module.



3.5.1 Client-Server Architecture

The Client-Server Architecture is used for the overall system because we have two sub-systems (the robot and the GUI controller) connected by a third sub-system (the communications system). The robot acts as the server in that moving is considered part of the service with the controller making requests of the robot sub-system and these requests are sent back and forth over the communications link.

3.5.2 Interrupt-driven Architecture

For the physical robot we have chosen to pursue an interrupt-driven architecture. This is because we consider the fast response to particular events to be critical. This may involve the user telling the robot to stop, change paths, etc. or if the robot itself recognises that it is off course.

3.5.3 Event-driven Architecture

The event-driven architecture was chosen for the GUI subsystem. This is because the GUI needs to listen to and respond to events that are caused by the user (eg. clicking on a button) and the robot itself (reports of speed, position, etc.).

3.5.4 Layered Architecture

The communications system will be broken down into two layers. One part will deal with the low-level process of transferring packets of data. The other layer will deal with the order these packets are sent/received in and the error correction resulting from transmission errors.

3.6 Technical Constraints

- Platform
 - The host machine must run a Java Virtual Machine.
 - The host machine must run the Unix operating system (or equivalent).
 - The host needs to read and write maps of the factory floor in the XML file format.
 - The host needs to have at least one USB port free and be able to recognise the CP2102 Single Chip USB-to-UART Bridge.

- Robot
 - 8K x 14 words of Flash (Program Memory)
 - 368 x 8 bytes of RAM (Data Memory)
 - It has a single dimensioned sensor array, consisting of 3 sensors.
 - Uses the PIC16F87X microcontroller.
 - Has two wheels, each equipped with a stepper motor.
 - Has a seven segment LED display.
- Communication
 - The communications system consists of two radios (one on the host machine and one on the robot) that implement the 802.15 standard.
 - The 802.15 support is provided by the MaxStream Xbee chipset.
 - The radios must communicate using 8.6 kilobits per second, framed as 8 data bits with no parity and no hardware flow control.
 - There will be errors in data received by either unit.

3.7 Assumptions and Dependencies

Assumptions

- The environment bounds for robot movement is planar and horizontal.
- There are no obstacles on the factory floor.
- The factory floor has clear line markings which are suitable for the robot to follow.
- The system has a colour monitor capable of a resolution of at least 800x600 pixels.
- There are no other robots of this type operating in the factory.

Dependencies

- The host machine will have a suitable environment to run the software.
- Java Development Kit will be installed and of a version not earlier than 1.5.0_06
- Version 3.80 of GNU Make should be installed on the system.

- The PIC C compiler needs to be installed on the system.
- The RXTX Java library must be installed on the system.
- When bootloading the robot the radios are placed next to each other so as to minimise potential transmission errors.

Functional Requirements

4.1 Interface Requirements

4.1.1 Robot movement information should be relayed to the operator

The interface should show suitable information about the current movement of the robot.

Justification:

The operator of the robot system should be kept informed of where the robot is in relation to both the operator and the factory itself.

Source:

WS3

Suitability Metric:

Once the map has been loaded into the interface and the robot starts moving the user should be able to see somewhere an indication of the robots travel. This should include where the robot is currently positioned on the map, its speed and its direction of movement.

Status:

Implemented.

4.1.2 The robot should be able to initially be placed anywhere on the map.

Once it has scanned the factory floor the robot must be able to start from anywhere on the map. It should be able to be positioned on the map by the GUI.

Justification:

If there is a power failure the robot should not have to be placed at its initial starting position in order to continue operation.

Source:

WS3.

Suitability Metric:

Assuming the robot already has a correct map of the factory floor, once powered on the interface should ask the operator to indicate the current position of the robot and the position in which it is currently facing.

Status:

Implemented.

4.1.3 Map format

Maps of the factory floor must be stored in an XML file format that implements the specified Document Type Definition (see Appendix 1).

Justification:

XML is a both human and machine-readable file format that can easily be used by other applications if the client chooses to do so.

Source:

WS3

Suitability Metric:

All map files should be valid XML and conform to the DTD. Files that don't conform will require fixing, either by the system automatically or by necessitating the robot remapping the factory floor.

Status:

Implemented.

4.1.4 Loading a map

The GUI should be able to load XML maps of the factory floor. These maps should then be displayed visually on the GUI.

Justification:

The user should not have to use the robot to scan the factory floor and create a map everytime the robot is needed to be used.

Source:

WS3

Suitability Metric:

A simple XML map file should be able to be verified visually as to whether it is correct or not by checking the XML code against what is displayed on the GUI. Further tests can be done by checking that a map, once saved, is identical to the map if the GUI is restarted and the saved map loaded.

Status:

Implemented.

4.1.5 Emergency stop

The user needs to be able to quickly stop the robot in an emergency.

Justification:

This may be necessitated by an emergency in the factory, created either by something external to the robot (ie. the actions of humans or other equipment) or the robot itself (an error).

Source:

PD, MC2

Suitability Metric:

Upon clicking on the stop button on the GUI the robot should stop after its current command has completed.

Status:

Implemented.

4.1.6 User-settable "no-go" zones

The user should be able to temporarily mark certain areas of the factory (via the map) as places where they do not wish the robot to go to. The robot cannot enter these no-go zones. No-go zones cannot be added if the robot is currently in that section of the map.

Justification:

There may be activities occurring on the factory floor where the robot could cause damage to itself or others if it travelled there.

Source:

PD

Suitability Metric:

Once the map has been loaded the user should be able to mark certain areas of the map as "no-go" zones (NGZs). This should be able to happen regardless of whether the robot is moving or not. Once a NGZ is defined the map on the controller should provide a visual indication of where the NGZ is.

Status:

Implemented.

4.1.7 User-settable speed limits

The user should be able to specify the maximum speed that the robot can travel over a specific part of the map.

Justification:

There may be occasional use of the area in question by personnel where it is guaranteed that if the robot is travelling slowly enough there will not be any foreseeable problem.

Source:

WS3

Suitability Metric:

The controller must allow the use to mark certain areas of the map with a speed limit. Once set it should provide a visual indication of the speed-limited parts of the map. When a robot is travelling in a speed-limited area that visual indication should be heightened.

Status:

Not implemented. Time constraints prevented this requirement from being fulfilled.

4.1.8 Path determination

The user should be able to select a start and destination point for the robot and the system will calculate a continuous path between the two points that does not involve entering a no-go zone.

Justification:

The robot needs to be told which particular movement instructions (go forward, turn left, turn right, etc.) it needs to perform in order to navigate the factory floor.

Source:

PD

Suitability Metric:

From within the GUI the user should be able to set a destination point that lies on one of the lines of the factory map. The GUI should then calculate a continuous path that does not violate the no-go zone requirement. This path should be displayed on the map and the GUI should then break this path down into movement commands to send to the robot.

Status:

Implemented.

4.1.9 Path determination with user-selectable metrics

The path determined by the GUI based upon the starting and destination points of the robot should be able to be optimised by following two metrics. The first should be to choose the shortest path by distance, the second should be to minimise the number of turns (one turn is defined as one 90 degree rotation) the robot takes in its route. The user should be given the choice of these two metrics when setting a destination point for the robot.

Justification:

The robot may need to transport delicate goods such as goods that may be in danger of slipping off if the robot makes too many turns - hence the user may wish to minimise the number of turns the robot makes. Alternatively, the user may want a robot that is not carrying anything to make its journey in the fastest time possible.

Source:

WS3

Suitability Metric:

The user should be able to indicate how important speed and number of turns is to them at any given moment. The routing of the robot should change accordingly.

Status:

Implemented. There may be some 'edge cases' where the an alternative route may be better than the route determined by the GUI but these cases occur infrequently.

4.1.10 Manual robot controls

The user should be able to control the movement of the robot without needing a map. This should be possible by using the GUI - allowing the user to rotate the robot clockwise or counter-clockwise and move forwards or backwards a small distance.

Justification:

Although the prototype robot is light the real robot is quite heavy and cannot be lifted easily. Manual controls would allow the user to quickly position the robot on the map for the map creation to begin.

Source:

WS3, MC1

Suitability Metric:

An interface on the controller should enable the user to move the robot in any direction (forwards, backwards, rotate clockwise, rotate counter-clockwise).

Status:

Implemented. The user is able to click on buttons corresponding to the manual movement action they wish the robot to perform. The robot will then move a small distance in that direction.

4.1.11 Initial scanning of the factory floor

When a robot arrives at the factory for the first time it needs to be able to 'scan' the factory floor for paths that it is allowed to move around. Once this is done the map should be saved in an appropriate XML document.

Justification:

Creating a map manually is tedious and time-consuming.

Source:

MC1, PD

Suitability Metric:

Upon being positioned along a line of the factory the robot should scan the full extents of the factory and communicate its findings back to the controller. The controller will tell the robot where to scan and in what order. The controller must take this data and create the required XML file.

Status:

Not implemented. Due to a lack of time this requirement was unable to be fulfilled. It is still possible to operate the robot if the map XML file is already written.

4.1.12 Displaying the factory map

The GUI should be able to display a visual representation of the map.

Justification:

The user needs a visual representation of the factory floor in order to be able to operate the rest of the GUI.

Source:

PD, MC1

Suitability Metric:

When a map is loaded by the user it should be displayed on the GUI. Things like intersections and no-go zones should be clear to the user.

Status:

Implemented.

4.2 Robot Requirements

4.2.1 Speed

The robot must obey any speed limits for the segment they are on or any overall speed limit imposed via the GUI.

Justification:

Individual parts of the factory floor may have speed limits. Also, the speed of the robot may want to be globally restricted by the user.

Source:

WS3

Suitability Metric:

When the robot told to move at said speed the robot should do so at the given speed. This can be verified by physically observing the distance the robot travels in a given time.

Status:

Not implemented. Time constraints led to this requirement not being fulfilled.

4.2.2 Line following

The robot must always maintain contact with the black lines of the factory floor.

Justification:

This is to allow automated movement of the robot and so that the robot and GUI can keep track of where the robot is.

Source:

PD

Suitability Metric:

If the robot is initially positioned on a black line and told to move forward it should move forward whilst adjusting its path continuously in order to follow that black line.

Status:

Implemented.

4.3 Robot Requirements

4.3.1 Display

The robot should display information on its LEDs to inform the user what it is doing.

Justification:

This will give warning to other people on factory floor as they will know in what direction the robot is moving or turning.

Source:

PD

Suitability Metric:

When the robot is moving forward, the LEDs display FWD. When the robot is rotating, the LEDs display ROT. etc.

Status:

Implemented to some degree. The LEDs are also used to show what the sensors are reading in for the purpose of calibrating the sensors.

4.4 Communications Requirements

4.4.1 Error Correction

The wireless communications link needs to correct errors that may arise during transmission.

Justification:

If other radio systems are operating within the vicinity this will dramatically increase the chance of errors. If erroneous signals are not identified it could lead to the robot performing incorrect instructions. The robots will be operating in a factory environment with considerable separation between the host and the robot, with potential for a large amount of interference.

Source:

Radio How-To (Kevin Macuinas)

Suitability Metric:

A signal that intentionally contains erroneous packets is sent from either the host to the robot or the robot to the host. The receiving party should be able to identify when this occurs. If the command passes error checks but ends up in a command that the robot does not recognise this should be identified as an error also.

Status:

Using two host-side radios this was implemented. Due to hardware issues with our supplier the communications link was changed to instead use a serial cable connected directly between the host and the robot. The result of this was that all the features of the radio interface were ported to use the serial link instead (including this one).

4.4.2 Packet Loss

The communications system needs to take into account the fact that some packets will never arrive at the receiver. An ARQ scheme (Automatic Repeat-reQuest) will need to be implemented to handle this problem.

Justification:

The wireless communications link is prone to packet loss as the distance between the two parties increases. The sender of the data needs an acknowledgement from the receiver that the data has been received otherwise it will not know if it should continue to send data.

Source:

Radio How-To (Kevin Macuinas)

Suitability Metric:

One radio should be able to send a numbered sequence of packets and, with help of the ARQ scheme, the same numbered sequence of packets should be able to be received at the other radio.

Status:

Using two host-side radios this was implemented. Due to hardware issues with our supplier the communications link was changed to instead use a serial cable connected directly between the host and the robot. The result of this was that all the features of the radio interface were ported to use the serial link instead (including this one).

4.4.3 Loss of communication

If the robot and the controller lose communications after a pre-set timeout period the robot needs to come to a halt. The controller should notify the user that there has been a loss of communications.

Justification:

A loss of communication could potentially be dangerous if the robot is travelling in the absence of human control.

Source:

Group Discussion

Suitability Metric:

A loss of communication can be artificially created by unplugging the host radio. After the pre-defined timeout period both the host and the robot should recognise this and as a consequence complete the above requirement.

Status:

Using two host-side radios this was implemented. Due to hardware issues with our supplier the communications link was changed to instead use a serial cable connected directly between the host and the robot. The result of this was that all the features of the radio interface were ported to use the serial link instead (including this one).

4.5 System Requirements

4.5.1 Movement

The robot must be able to move forward and backwards said distance. Rotate either left or right said number of degrees.

Justification:

The robot will need to be able to traverse the factory floor.

Source:

WS3

Suitability Metric:

When the robot is told to move forward/backwards a given distance it should do so. Likewise, if the robot is told to rotate a given number of degrees left/right the robot should rotate that given number of degrees.

Status:

Implemented.

4.5.2 Line Mapping

The system shall support auto-mapping of factoryfloors, such that the robot scans the lines on the floor and relays the information back to the host. The factory layout can then be stored in the XML map format by the GUI.

Justification:

This will allow the GUI to display the map for the user to see. It also avoids having to manually write an XML file to describe the factory floor.

Source:

WS3

Suitability Metric:

When asked by the user, the robot traverses the map and sends back information to the GUI, allowing the GUI to draw the map so that the user can see and, if they wish, to save the map.

Status:

Not implemented. Time constraints prevented this requirement from being fulfilled.

Interaction Scenarios

5.1 Initial Positioning of the Robot

Initial Assumptions

The user has opened the controlling software on the host computer and the robot is powered on. The user then loads a previously created map on the GUI.

Normal

The operator manually verifies the location of the robot and may use the manual-controls of the GUI to move the physical robot to the start point on the factory floor, which must be on a line. Then the user uses the manual positioning tool to indicate the position where the robot is on the map.

What can go wrong

The user may incorrectly position the physical robot so that it is not in what would be considered the start point.

The user may incorrectly enter the heading of the robot. In this case, the robot will only enter the auto-sensing phase and allow the user to rotate the map.

Other activities

The controlling software will not concurrently perform other activities.

System state on completion

The controlling software is running and connected to the robot. It allows manual control via the GUI and is waiting for confirmation to begin auto-sensing the factory floor.

5.2 Initial Scanning of a New Factory

Initial Assumptions

The user has opened the controlling software on the host computer and the robot is powered on. The user uses the manual movement function of the GUI to place the robot in one corner of the factory grid.

Normal

The user will active map scanning mode through the GUI. The robot will then take movements from the GUI, which will build the map into memory and display it when it is finished. The robot will report throughout when it reaches an intersection and tell the controller what kind of intersection it has reached (eg. crossroads, T-junction, etc.).

What can go wrong

The user may incorrectly position the physical robot so that it is not in a corner of the map.

If the floor of the factory is highly reflective or inconsistent with its line markings then the robot may incorrectly sense an intersection, leading to the robot arriving at the wrong position on the factory floor.

Other activities

The controlling software will not concurrently perform other activities.

System state on completion

The GUI controller will have a map file in its memory that it displays for the user. This map can be saved as an XML map file for later use.

5.3 Moving the Robot Around a Factory

Initial Assumptions

The user has opened the controlling software on the host computer and the robot is powered on. The user loads a pre-made map by using the GUI. The user uses the manual movement function of the GUI to place the robot on a line of the

factory floor and then uses the robot placement tool to indicate where on the map the robot is and in which direction it is facing.

Normal

The user uses the set destination tool to pick a spot on the map where they would like the robot to move to. The controller will not allow the user to set the destination point to be in a no-go zone. The user will then click the go to destination button upon which the GUI will tell the robot which movement commands it needs to execute in order to reach the destination.

What can go wrong

The user may incorrectly position the physical robot so that it is not on a line. Additionally, the user may not indicate the correct orientation (north, south, east or west) of the robot.

If the floor of the factory is highly reflective or inconsistent with its line markings then the robot may incorrectly sense an intersection, leading to the robot arriving at the wrong position on the factory floor.

Other activities

The controlling software will update on the map the position of the robot every time the robot reaches an intersection.

System state on completion

The robot will be at the desired destination on the factory floor and that same position will be reflected on the map display on the GUI.

Appendices

6.1 Appendix 1: XML DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- document type declaration for the SEP Factory Map -->
<!-- 1.0.2006 David Hemer -->
<<<<<< .mine
<!DOCTYPE FACTORY-MAP [
  <!ELEMENT FACTORY-MAP (ATTRIBUTES, AREA, ROBOT, (HUMAN)*)>
  <!ELEMENT HUMAN (POINT, ATTRIBUTES)>
  <!-- all humans xare uniquely identified -->
  <!ATTLIST HUMAN hid ID #REQUIRED>
  <!ELEMENT ROBOT (POINT, ATTRIBUTES)>

  <!ELEMENT AREA ((AISLE)*, (NOGOZONE)*, ATTRIBUTES)>

  <!-- General Attributes definition -->
  <!ELEMENT ATTRIBUTES (PAIR)*>
  <!ELEMENT PAIR (KEY, VALUE)>
  <!ELEMENT KEY (#PCDATA)>
  <!ELEMENT VALUE (#PCDATA)>

  <!-- Aisle definition -->
  <!ELEMENT AISLE (POINT, POINT)>

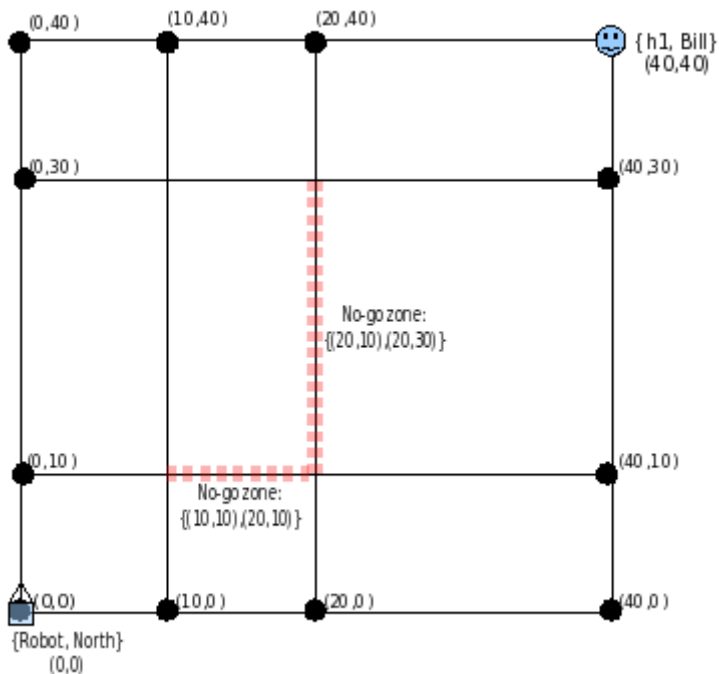
  <!-- No go zone definition -->
  <!ELEMENT NOGOZONE (POINT, POINT)>

  <!-- Point definition -->
  <!ELEMENT POINT (X, Y)>
  <!ELEMENT X (#PCDATA)>
  <!ELEMENT Y (#PCDATA)>
]>
<FACTORY-MAP>
  <ATTRIBUTES>
    <PAIR> <KEY>name</KEY> <VALUE>"Example Map"</VALUE> </PAIR>
    <PAIR> <KEY>units</KEY> <VALUE>"centimetres"</VALUE> </PAIR>
  </ATTRIBUTES>
  <AREA>
    <AISLE>
      <POINT> <X>0</X> <Y>0</Y> </POINT>
      <POINT> <X>0</X> <Y>40</Y> </POINT>
    </AISLE>
    <AISLE>
      <POINT> <X>10</X> <Y>0</Y> </POINT>
      <POINT> <X>10</X> <Y>40</Y> </POINT>
    </AISLE>
    <AISLE>
      <POINT> <X>20</X> <Y>0</Y> </POINT>
      <POINT> <X>20</X> <Y>40</Y> </POINT>
    </AISLE>
    <AISLE>
      <POINT> <X>40</X> <Y>0</Y> </POINT>
      <POINT> <X>40</X> <Y>40</Y> </POINT>
    </AISLE>
    <AISLE>
      <POINT> <X>0</X> <Y>0</Y> </POINT>
      <POINT> <X>40</X> <Y>0</Y> </POINT>
    </AISLE>
    <AISLE>
      <POINT> <X>0</X> <Y>10</Y> </POINT>
```

```

    <POINT> <X>40</X> <Y>10</Y> </POINT>
  </AISLE>
  <AISLE>
    <POINT> <X>0</X> <Y>30</Y> </POINT>
    <POINT> <X>40</X> <Y>30</Y> </POINT>
  </AISLE>
  <AISLE>
    <POINT> <X>0</X> <Y>40</Y> </POINT>
    <POINT> <X>40</X> <Y>40</Y> </POINT>
  </AISLE>
  <NOGOZONE>
    <POINT> <X>10</X> <Y>10</Y> </POINT>
    <POINT> <X>20</X> <Y>10</Y> </POINT>
  </NOGOZONE>
  <NOGOZONE>
    <POINT> <X>20</X> <Y>10</Y> </POINT>
    <POINT> <X>20</X> <Y>30</Y> </POINT>
  </NOGOZONE>
  <ATTRIBUTES>
    <PAIR> <KEY>name</KEY> <VALUE>"Factory"</VALUE> </PAIR>
  </ATTRIBUTES>
</AREA>
<ROBOT>
  <POINT> <X>0</X> <Y>10</Y> </POINT>
  <ATTRIBUTES>
    <PAIR> <KEY>name</KEY> <VALUE>"Robot"</VALUE> </PAIR>
    <PAIR> <KEY>direction</KEY> <VALUE>"North"</VALUE> </PAIR>
  </ATTRIBUTES>
</ROBOT>
<HUMAN hid="h1">
  <POINT> <X>40</X> <Y>40</Y> </POINT>
  <ATTRIBUTES>
    <PAIR> <KEY>name</KEY> <VALUE>"Bill"</VALUE> </PAIR>
  </ATTRIBUTES>
</HUMAN>
</FACTORY-MAP>
\end{document}

```



A visual representation of the above XML example.

6.2 Appendix 2: Cloud⁹ Member Information

Curriculum Vitae - Jin Rong

Group Role

Robot Developer

Education

Adelaide University 2005-present

Bachelor of Business Information Technology

Knowledge

Proficient in Java and Assembly, and has previously coded in VB, PHP, SQL, and HTML design.

Experience

Extensive background in business, resulting in effective management skills and organizational abilities. Strong background and experience in Java programming. Completed group projects in the business discipline in addition to projects in computer science.

Relevant Subjects

- Software Engineering and Project
- Introduction to Software Engineering
- Data Structures and Algorithms
- Computer Systems
- Computer Science I

Curriculum Vitae - Andrew Brock

Group Role

Communications Developer

Education

Scotch College, Adelaide 1993-2002

SACE Certificate 2002

University of Adelaide 2003-present

Bachelor of Engineering (Telecommunications) & Bachelor of Finance

Knowledge

A high level of knowledge of Java, driven by university and private projects. An intermediate level of experience in Python. Familiar with the Motorola 68000 and Xilinx Spartan architectures.

Experience

Completed a DSTO Summer Vacation Scholarship over 2005-2006. In 2005, tutored first year Computer Science students at the Computer Science Learning Centre and as a consequence gained skills in explaining complex topics to people not necessarily sharing a high level of technical competence. Completed group projects at university across a variety of disciplines including Electrical Engineering, Computer Science, Statistics and Commerce.

Relevant Subjects

- Software Engineering and Project
- Telecommunications System Modelling
- RF Engineering III
- Control III
- Embedded Computer Systems
- Computer Systems
- Digital Electronics
- Communications, Signals and Systems
- Introduction to Software Engineering

Curriculum Vitae - David Lucas

Group Role

Robot Developer

Education

Prince Alfred College, Adelaide 1998-2002

SACE Certificate 2002

University of Adelaide 2003-present

Bachelor of Engineering (Computer Systems) & Bachelor of Economics

Knowledge

A high degree of knowledge of Java through university study and private projects. A basic knowledge of C, C++ and Assembly through university study and side projects. Familiar with the Motorola 68000 and Xilinx Spartan architectures.

Experience

Completed group projects at university across a variety of disciplines including Electrical Engineering, Computer Science, Commerce and Economics.

Relevant Subjects

- Software Engineering and Project
- Real Time Systems IV
- RF Engineering III
- Control III
- Embedded Computer Systems
- Computer Systems
- Digital Electronics
- Communications, Signals and Systems
- Data Structures and Algorithms
- Computer Science I

Curriculum Vitae - Ian Young

Group Role

GUI/Host-Side Control Developer

Education

Grinnell College, Iowa, United States 2004-present

Bachelor of Computer Science

University of Adelaide 2006-present

Knowledge

Skilled with Java and C, and to a lesser extent with Scheme, PHP, and HTML/CSS design. Familiar with underlying data structures and low-level operations, as well as the principles of functional programming and object-oriented design.

Experience

Extensive group work incorporated into past classes, resulting in good people skills and organizational abilities. In 2006, undertook a two-month software design project as part of a three-man team, gaining valuable experience in all parts of the software design process.

Relevant Subjects

- Software Engineering and Project
- Data Representation, Memory Management, and Formal Method
- Data Structures, Algorithms, and Object Oriented Design
- Computer Science I

Curriculum Vitae - Andrew Limareff

Group Role

GUI/Host-Side Control Developer

Education

Charles Darwin University 2003-2004

Bachelor of Information Technology

Royal Melbourne Institute of Technology 2004-2005

Bachelor of Software Engineering

Adelaide University 2005-present

Bachelor of Computer Science (Software Engineering)

Knowledge

Highly proficient in Java and C#, versed in C, and has previously coded in assembly. Has a thorough working knowledge of Object Oriented and procedural program design.

Experience

Has completed numerous IT related group projects in the higher-education environment and electronics projects in the workforce. Apart from the various assignments undertaken for University studies, Andrew has written event-driven, GUI interfaced programs used to automate pen and paper roleplaying games, and ASP.NET web based scripts for not-for-profit organisations.

Relevant Subjects

- Software Engineering and Project
- Event Driven Computing
- Artificial Intelligence
- Data Structures and Algorithms
- Computer Systems
- Foundations of Data Communications
- IT Project Management

Curriculum Vitae - Joel Stanley

Group Role

Communications and Robot Software Developer

Education

Saint Josephs School, Port Lincoln 1994-2002

SACE Certificate 2002

University of Adelaide 2003-present

Bachelor of Engineering (Computer Systems) & Bachelor of Economics

Knowledge

An intermediate level of experience in Python, C and PHP. Familiar with the Motorola 68000, PIC16F819, Xilinx Spartan architectures, as well as Linux and VxWorks operating system kernels. A strong knowledge of Open Object programming methodologies through Java programming.

Experience

Developed various web based database systems, in using PHP, MySQL, javascript and HTML. Strong understanding of assembly programming and low level hardware design and operation, through practical experience. Have taught highschool students soldering and basic assembly programming. Open source software development experience.

Relevant Subjects

- Software Engineering and Project
- Real Time Systems
- RF Engineering III
- Control III
- Embedded Computer Systems
- Computer Systems
- Digital Electronics
- Communications, Signals and Systems
- Data Structures and Algorithms

Group Signing Sheet

David Lucas
1105029

Andrew Brock
1105664

Joel Stanley
1105806

Jin Rong
1121937

Andrew Limareff
1135144

Ian Young
1152777