1

SPECIFICATION

Code mobility in Networked Embedded System

Release 0.0.1

Group 4

November 19, 2012

Igor Pelesić, Matrikelnumber 0006828
Konstantin Selyunin, Matrikelnumber 1228206

Contents

2 Introduction

3

3.1 Purpose

The specification defines the goals that should meet the project "Code mobility in Networked Embedded system". It is written by project members listed on the title page to precisely identify the goals to meet at the end of the project.

3.2 Scope

3.3 Definitions, Acronyms, and Abbreviations

3.4 Background

3.5 References

3.6 Overview

4 General Description

This section of the SRS should describe the general factors that affect 'the product and its requirements. It should be made clear that this section does not state specific requirements; it only makes those requirements easier to understand.

4.1 Product Perspective

This subsection of the SRS puts the product into perspective with other related products or projects. (See the IEEE Guide to SRS for more details).

4.2 Product Functions

This subsection of the SRS should provide a summary of the functions that the software will perform.

4.3 User Characteristics

This subsection of the SRS should describe those general characteristics of the eventual users of the product that will affect the specific requirements. (See the IEEE Guide to SRS for more details).

4.4 General Constraints

This subsection of the SRS should provide a general description of any other items that will limit the developer's options for designing the system. (See the IEEE Guide to SRS for a partial list of possible general constraints).

4.5 Assumptions and Dependencies

This subsection of the SRS should list each of the factors that affect the requirements stated in the SRS. These factors are not design constraints on the software but are, rather, any changes to them that can affect the requirements in the SRS. For example, an assumption might be that a specific operating system will be available on the hardware designated for the software product. If, in fact, the operating system is not available, the SRS would then have to change accordingly.

5 Requirements

Define requirements for the project "Code mobility in Networked Embedded system"

1 **User roles**

1.1 Application Developers (Tasks: Create control application in agent language, debug, test, prepare deployment packages)

1.2 Application Consumers (Tasks: Deploy control application on target system, fill valuable bug reports)

1.3 Plattform Developers (Tasks: Maintenance, Extensions, Porting to another target board)

1.4 Maybe: Application Designers (Tasks: Design control application)

Global Requirements:

2 **Application development requirements:**

2.1 The App Developer should be enabled to instantiate up to 4 agents on a single node, which are running concurrently.

2.2 The App Developer should be allowed to configure the execution scheduling of the agents via a prioriatization of the agents.

2.3 The platform should provide a simple agent programming language to the App Developer in which the agents of an application can be developed.

2.4 The agent language should provide the App Developer with the possibility to reproduce its code on another node or on another board.

2.5 The agent language should provide the App Developer with the possibility to communicate with another agent on the same board.

2.6 The agent language should provide the App Developer with the possibility to access the node hardware.

2.7 The agent language should provide the App Developer with the possibility to implement loops.

2.8 The agent language should provide the App Developer with the possibility to compare variables.

2.9 The agent language should provide the App Developer with the possibility to perform addition, subtraction, multiplication and division on variables.

2.10 The agent language should provide the App Developer with the possibility to perform delays in the execution of code.

2.11 The platform should allow debugging of agents executions.

2.12 The platform should provide means for the creation of easily installable deployment packages.

3 **Application Consumers requirements:**

3.1 The platform should provide means to deploy the agent software on the target boards easily.

3.2 A tracing mechanism should be provided in order to ease the process of fault detection and to allow valuable bug descriptions.

4 **Non-functional requirements**

4.1 The platform should be open to extensions i.e adding new hardware. (Non-functional)

4.2 The agent language should be extendable.(Non-functional)

4.3 Scalability

4.4 Documentation

4.5 A platform tracing mechanism should be provided which allows for more efficient bugfixing.

5 **What do the Application Designers need:**

5.1 A description of the platform possibilities and limitations should be provided.

5.2 The platform should provide means for reducing the overall complexity of a system, by allowing encapsulation of different tasks.

5.3 The platform should provide configurable inter agent communication facilities.

5.4 The platform should provide means to enable standby scenarios by allowing dynamical code reproduction.

5.5 The platform should provide means for strong mobility, where an agent and its execution state are transferred to a new node or board and the execution on the new destination is started from the memorized state.

5.6 A description of a platform should provide a list of all available services

5.1 MESS)

What do the Application Testers need: 1) ?? To test control applications you need access to sensor data ?? Hence a test mode should be supported where sensor data are presented to the tester.

Functional requirements

Non-functional requirements

Specification (based on this functional and non-functional requirements)

User defines agents in a simple programming language (called "Agent"). "Agents" program is interpreted on PC to bytecode.

NES board support four platforms (nodes) for agents interactions and execution and 1 platform (node) for board-to-board communication.

Project outline

This will be the largest and most important section of the SRS. The customer requirements will be embodied within Section 2, but this section will give the D-requirements that are used to guide the project's software design, implementation, and testing.

Each requirement in this section should be: • Correct • Traceable (both forward and backward to prior/future artifacts) • Unambiguous • Verifiable (i.e., testable) • Prioritized (with respect to importance and/or stability) • Complete • Consistent • Uniquely identifiable (usually via numbering like 3.4.5.6)

Attention should be paid to the carefuly organize the requirements presented in this section so that they may easily accessed and understood. Furthermore, this SRS is not the software design document, therefore one should avoid the tendency to over-constrain (and therefore design) the software project within this SRS.

5.2 External Interface Requirements

5.2.1 User Interfaces

5.2.2 Hardware Interfaces

5.2.3 Software Interfaces

5.2.4 Communications Interfaces

5.3 Functional Requirements

This section describes specific features of the software project. If desired, some requirements may be specified in the use-case format and listed in the Use Cases Section.

5.3.1 Functional Requirement or Feature 1

Introduction

Inputs

Processing

Outputs

Error Handling

5.3.2 Functional Requirement or Feature 2

. . .

5.4 Use Cases

5.4.1 Use Case #1

5.4.2 Use Case #2

. . .

5.5 Classes / Objects

5.5.1 Class / Object #1

Attributes

Functions

¡Reference to functional requirements and/or use cases¿

5.5.2 Class / Object #2

. . .

5.6 Non-Functional Requirements

Non-functional requirements may exist for the following attributes. Often these requirements must be achieved at a system-wide level rather than at a unit level. State the requirements in the following sections in measurable terms (e.g., 95

5.6.1 Performance

5.6.2 Reliability

5.6.3 Availability

5.6.4 Security

5.6.5 Maintainability

5.6.6 Portability

5.7 Inverse Requirements

State any *useful* inverse requirements.

5.8 Design Constraints

Specify design constrains imposed by other standards, company policies, hardware limitation, etc. that will impact this software project.

5.9 Logical Database Requirements

Will a database be used? If so, what logical requirements exist for data formats, storage capabilities, data retention, data integrity, etc.

5.10 Other Requirements

Catchall section for any additional requirements.

6 Analysis Models

List all analysis models used in developing specific requirements previously given in this SRS. Each model should include an introduction and a narrative description. Furthermore, each model should be traceable the SRS's requirements.

6.1 Sequence Diagrams

6.2 Data Flow Diagrams (DFD)

6.3 State-Transition Diagrams (STD)

7 Change Management Process

Identify and describe the process that will be used to update the SRS, as needed, when project scope or requirements change. Who can submit changes and by what means, and how will these changes be approved.

8 Implementation timetable

Compiler for agent language:

- Agent language lexical analysis
- Agent language syntax analysis
- Agent language semantic analysis

Platform for mobile agents:

- Agent structure representation
- Implementation of Agent structure
- Implementation of Agent functions
- Implementation of drivers
- Implementation of scheduler
- Implementation of communication protocols

9 Agent language

Lexical elements	Agent program includes the following lexical elements
keyword	'function' | 'var' | 'function' | 'var' | 'int' | 'char' | 'boolean' | 'true' | 'false' | 'this' | 'if' | 'else' | 'while' | 'return' | 'move' | 'clone' | 'message' | 'to' | 'wrtarget' |
symbol	| '{' | '}' | '(' | ')' | '[' | ']' | '.' | ',' | ';' | '+' | '-' | '*' | '/' | '&' | ' | ' | '¡' | '¿' | '=' | '~'
intConstant	a number in range $0 \ldots 2^{16}$
stringConstant	' " ' a sequence of Unicode characters including in double quotes ' " '
identifier	A sequence of letters, digits, and underscore('_') starting with a letter
boardIdentifier	Board1 | Board2
platformIdentifier	boardIdentifier.(Platform1 | Platform2 | Platform3 | Platform4)
serviceIdentifier	boardIdenttifier.platformIdentifier.(serviceA | serviceB | serviceC | serviceD | serviceE | serviceF1 | serviceF2 | serviceG)
agentIdentifier	indentifier
Program structure	Structure of Agent program

10 Communication Architecture

The communication architecture is designed to support communication between nodes on the same development board as well as between boards.

10.1 Hardware

The communication on the board is carried out over two serial bus channels. One of them is to be used for a distributed control application running on nodes 0-3. Another bus is dedicated for code mobility between nodes 0-4.

Access to the bus is controlled by separate UART modules on each micro-controller. The bit rate is constrained by the maximum value of 2 Mbps according to the manual.

Node 4 functions as a gateway to another board. It is a bridge between the local and the wireless zigbee network.

10.2 Distributed Control

Time-triggered protocols are customary for distributed control applications. This approach is suitable for low data volumes and data subject to real time constraints and regular sending intervals. In a time triggered scheme each node has a separate slot for writing to the bus. Meanwhile, other nodes can read the bus in the same slot or process a computation task.

10.3 Code Mobility

Code mobility between nodes includes local mobility on the same board and remote mobility between different boards. Executable agents generally have larger volume than control data. Sending at regular time intervals is not assumed, thus communication is aperiodic. A simple protocol based on message acknowledgment can be used.

There are two use cases: a) local mobility: destination is one of the nodes 0-3. b) remote mobility: destination is the gateway node 4. The gateway is to contain a zigbee stack implementation to enable access to the personal area network.

10.4 Addressing Scheme

Simple local addressing requires unique identifiers for each node. The requirement is that this be compatible with the time-triggered protocol implementation. For remote communication, board addresses have to be compatible with the configuration of the zigbee network. Since, each node will have a static number of agent execution environments, the address has to contain its identifier as as well.

11 Appendix 1

12 Appendix 2

12.1 Standards

12.2 Training