|  |  |
| --- | --- |
| **Logo_FPT_University_doc** | **MINISTRY OF EDUCATION AND TRAINING** |

|  |
| --- |
| **FPT UNIVERSITY** |
| Capstone Project Document |
| SCARA ROBOT |
|  |
| |  |  | | --- | --- | | **Group 5** | | | **Group Members** | Phan Anh Dũng Cường – Leader  Nguyễn Minh Quân – Member  Cao Đình Nguyên Khoa – Member | | **Supervisor** | Trần Khánh Ninh | | **Ext Supervisor** |  | | **Capstone Project code** | APOD | |
|  |

|  |
| --- |
| - Ho Chi Minh City, September/2013 - |

# PART B – SOFTWARE PROJECT MANAGEMENT PLAN

## PROBLEM DEFINITION

### NAME OF THIS CAPSTONE PROJECT

* + Project name: **Lynxmotion A-Pod  robot controller**
  + Project code: **APOD**
  + Product type: **Embedded robot controller**

### PROJECT OVERVIEW

#### The Systems

A-Pod Robot is a automatic/ manual controlling system. Assembly with 6 leg (3 DOF), A mandible and Tail. It has total 25 servos to serve moving, rotating, gripping. There are three basic board for controlling (A SSC32 servos controller, STM32F4 Discovery using ARM, BOT BOARD II using BASIC ATOM PRO 28)\*

Each of board will be define and description later.





#### The Purpose

The main purpose of A-Pod Robot is to be used at home with variable simple tasks. The Project ‘s scope is to control A-Pod from a distance via a HCI program on PC (or mobile devices). A-Pod can perform task without the present of human and report back working information when needed.

#### Boundaries of the System

This project focus on programming on microprocessor unit (MCU) to developing, control the A-Pod robot. The BASIC is let him move forward, backward, right, left. Thus, we concern on control over Bluetooth. Through Camera put on robot control it/ or automatic function, connect with Sensor to discover obstructions.

Be sure to have :

- The legal hold , conversion optimization

- Ability to work flexible

- The structure must comply with the principle of modularity

#### Development Environment

**Hardware environment:**

* 1. Lyxnmotion APOD
  2. IP camera, Bluetooth device, Distance sensor.
  3. 4 laptops with appropriate configuration for embedded development.

**Software environment:**

1. Developing environment ARV studio 4 for main board programming.
2. Developing environment Visual studio 2012 with C# language for user interface programming.

## PROJECT ORGANIZATION

### SOFTWARE PROCESS MODEL

The process model used for developing this project is Spiral Model.

The spiral model is a software development process combining elements of both design and prototyping-in-stages, in an effort to combine advantages of top-down and bottom-up concepts. Also known as the spiral lifecycle model (or spiral development), it is a systems development method (SDM) used in information technology (IT). This model of development combines the features of the prototyping and the waterfall model. The spiral model is intended for large, expensive and complicated projects.



### ROLES AND RESPONSIBILITIES

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Full name | Team Role | Scrum Team Role | Responsibilities |
| 1 | Trần Khánh Ninh | Supervisor | Stake Holder | * Define business * Support in technical issues |
| 1 | Phan Anh Dũng Cường | Team Leader | Product Owner, Scrum Master | * Prioritize work * Answer question and deliver direction * Facilitate productivity – maximize team performance * Complete all individual work |
| 3. | Nguyễn Minh Quân | Team Member | Scrum Team Member | * Commit individual product on time * Support each other to complete team work |
| 4 | Cao Đình Nguyên Khoa | Team Member | Scrum Team Member | * Commit individual product on time * Support each other to complete team work |

### TOOLS AND TECHNIQUES

#### For Development

**Hardware environment:**

* 1. Lynxmotion APOD
  2. IP camera, Bluetooth device, Distance sensor.
  3. 4 laptops with appropriate configuration for embedded development.

**Software environment:**

1. Developing environment ARV studio 4 for main board programming.
2. Developing environment Visual studio 2010 with C# language for user interface programming.
3. Developing environment Keilc studio for program ARM.

#### For Management

* Microsoft Project: Task tracking
* Tortoise SVN: Source version control
* Google Code: Connect, synchronize source code and documents
* Beyond Compare: Manage changes in source files

#### For Communication

* Gmail for daily report and meeting minute
* Skype chat system

## PROJECT MANAGEMENT PLAN

### TASKS

* + 1. Initiating and Planning

|  |  |
| --- | --- |
| Description | Registering project, kick-off meeting and planning |
| Output | Registered project, team spirit, overview plan |
| Deliverables | Draft project plan |
| Effort (man-month[[1]](#footnote-1)) | 0.8 mm |
| Dependencies and Constrains | N/A |
| Risks | Some members may be absent |

* + 1. Software Requirements Analysis

|  |  |
| --- | --- |
| Description | Analyzing software requirements based on available hexapod’s hardware to create software requirements specification document |
| Output | Software Requirement Specification document |
| Deliverables | SRS document file |
| Effort (man-month) | 1.5 mm |
| Dependencies and Constrains | N/A |
| Risks | - Lack of knowledge about hexapod’s hardware |

* + 1. Creating Software Design Description

|  |  |
| --- | --- |
| Description | Designing the controller for hexapod robot based on actual requirements |
| Output | Architecture design, circuits diagram, board diagram, algorithms and design specification |
| Deliverables | SDD document |
| Effort (man-month) | 3.0 mm |
| Dependencies and Constrains | Completion of SRS |
| Risks | - Choosing inappropriate algorithms and design patterns  - The hardware is hard to maintain  - Causing high coding efforts |

* + 1. Coding

|  |  |
| --- | --- |
| Description | Implementation phase |
| Output | Source code |
| Deliverables | Source code is successfully loaded into chip  Executable program and source code in Hexapod’s hardware and hexapod’s controller simulator. |
| Effort (man-month) | 8.5 mm |
| Dependencies and Constrains | Completion of SRS, SDD |
| Risks | - Lack of experience in ES developing  - Hardware‘s limitations and errors  - Working without the lab  - Lack of knowledge to resolve the circuit ‘s errors |

* + 1. Testing

|  |  |
| --- | --- |
| Description | Creating test case and execute test |
| Output | Test plan, test case document, test report, all tested modules and tested system |
| Deliverables | Test plan, test report |
| Effort (man-month) | 2.2 mm |
| Dependencies and Constrains | Completion of SRS, SDD, coding |
| Risks | * Lack of professional testers in team * Unit test may not be performed thoroughly causing spending many efforts in system test phase.   - Hardware ‘s limitations and errors  - ES testing is different with IS testing |

* + 1. Deployment

|  |  |
| --- | --- |
| Description | Deploying system include : hexapod robot and hexapod controller simulator |
| Output | Software packages, user manual |
| Deliverables | Software packages, user manual |
| Effort (man-month) | 0.3 mm |
| Dependencies and Constrains | Completion of all other tasks |
| Risks | - Hardware’s errors |

### DOCUMENTATION

As a SE, I expect to have all related documents to be delivered on time. Those documents are separated into 6 reports, include:

|  |  |  |
| --- | --- | --- |
| REPORT NO. | DESCRIPTION | DELIVERY DATE |
| 1 | Introduction | 2013.09.16 |
| 2 | Software Project Management Plan (SPM) | 2013.09.24 |
| Problem Definition |  |
| Project organization |  |
| Project management plan |  |
| Coding convention |  |
| 3 | Software Requirements Specifications (SRS) | 2013.10.05 |
| User Requirement Specification |  |
| System Requirement Specification (Specific Requirements) |  |
| External Interface Requirements |  |
| System Features |  |
| Use Case Diagram & Use Case Specification |  |
| Software System Attributes |  |
| Entity Relationship Diagram |  |
| 4 | Software Design Description (SDD) | 2013.10.19 |
| Design Overview |  |
| System Architectural Design |  |
| Component Diagram |  |
| Detailed Description of Components |  |
| Sequence Diagram |  |
| User Interface Design |  |
| Database Design |  |
| 5 | Software Test Documentation (STD) | 2013.11.16 |
| Test Plan |  |
| Test Cases |  |
| Checklists |  |
| 6 | Software User’s Manual (SUM) | 2013.12.08 |
| Installation Guide |  |
| User’s Guide |  |
| FINAL | Completed version of Report No.1 to Report No.6 – Soft copy |  |
| Completed version of Report No.1 to Report No.6 – Hard copy |  |
|  |  |  |
| IMPLEMENTATION The CDs contain all source code and deployable APOD at the end of project  Application must be developed and tested completely. Those feature need to be implemented: | | |  |
|  |  |  |
|  |  |  |
| MODULE | DESCRIPTION | DELIVERY DATE |
| Lyxnmotion APOD | | |
| 0 | Servo Controller (Legs controller) on SSC32 board |  |
| 1 | Communication module (using ARM + Bluetooth) |  |
| 3 | Distance sensor (HC-SR04 + Botboard II) |  |
| 3 | C# Control |  |
| UI. |  |
| APOD emulation. |  |
| 3 | Move definition motor values to positions |  |
| Final | CDs contain all source code and deployable APOD as final version |  |

### DELIVERABLES

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1. No | Deliverable | Delivery date | Delivery location | Note |
| 1 | Introduction Document | 2013.09.16 | FU – CMS | Report No.1 |
| 2 | Software Project Management Plan | 2013.09.24 | FU – CMS | Report No.2 |
| 3 | Software Requirements Specification | 2013.10.05 | FU – CMS | Report No.3 |
| 4 | Software Design Description | 2013.10.19 | FU – CMS | Report No.4 |
| 5 | Software Test Documentation | 2013.11.16 | FU – CMS | Report No.5 |
| 6 | Software User’s Manual | 2013.12.08 | FU – CMS | Report No.6 |
| 7 | Completed version of Report No.1 to Report No.6 – Soft copy |  | Classroom  FU – CMS | Final version |
| 8 | Completed version of Report No.1 to Report No.6 – Hard copy |  | Classroom | Final version |
| 9 | CD contains all Source code and installable system |  | Classroom | Final version |

\* Deliverables is *potentially shippable product,* which can be a part of documents or runnable software application which is implementation of some program features.

### ALL MEETING MINUTES

*<To be updated>*

## CODING CONVENTION

The following rules follow:

* The standard rules for developing application using C/C++ (<http://users.ece.cmu.edu/~eno/coding/CCodingStandard.html>)

However, we have some differences:

* 1. Naming convention

Variables

* Normal variables
  + Beginning with underscore or letter.
  + Variable names must be in lowercase, meaning words are separated by underscore symbol “\_”.

For example:  
 \_uint8 current\_position = 0;  
 \_uint16 cmd\_running = 0;

* Constant
  + Beginning with underscore or letter.
  + Variable names must be in UPPERCASE, meaning words are separated by underscore symbol “\_”.

For example:  
 \_uint8 MAX\_COUNTER = 0xFF;  
 float PI = 3.14;

* 1. Value types

- Integer Number

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Common Types** | **User-define Type** | **Bits** | **Min** | **Max** |
| **char** | \_int8 | 8 | -128 | 127 |
| **unsigned char** | \_uint8 | 8 | 0 | 255 |
| **short, int** | \_int16 | 16 | -32768 | 32767 |
| **unsigned short, unsigned int** | \_uint16 | 16 | 0 | 65535 |
| **Long** | \_int32 | 32 | -231 | -231 - 1 |
| **unsigned long** | \_uint32 | 32 | 0 | -232 - 1 |
| **long long** | \_int64 | 64 | -263 | -263 - 1 |
| **unsigned long long** | \_uint64 | 64 | 0 | -264 - 1 |

Table 4.2.1: Definition of value types

- Floating Point Number

* float with 32 bits
* double with 64 bits.

- Boolean Type

* true as 1, false as 0
  1. Layout
* Should have a space between symbols  
   For example:  
   new\_pos = ((cur\_pos – last\_pos) \* 2) – 1;
* Open brace should be in the same line, after a space “ “, close space should be in new line.
* All sub blocks should be braced even if it has only 1 statement.

For example:

if (new\_pos < 0) {

new\_pos = 0;

} else if (new\_pos > 360) {

new\_pos = 360;

cycle\_counter++;

}

* Do not parenthesize the unnecessarily.

For example:

return new\_pos;

instead of: return (new\_pos);

* If an expression gets too long to fit in a line, break it next to a binary operator. Put the operator at the beginning of the next line to emphasize that it is continued from the previous line. Don't add additional indenting to the continued line*.*

1. 1 man-month equals to 22 man-day [↑](#footnote-ref-1)