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

|  |
| --- |
| **FPT UNIVERSITY** |
| Capstone Project Document |
| Designing and making  A Lynxmotion A-POD robot controller |
|  |
| |  |  | | --- | --- | | **Hexapod Team** | | | **Group Members** | * PhanAnhDũngCường * Nguyễn Minh Quân * Cao ĐìnhNguyênKhoa | | **Supervisor** | M.Si. TrầnKhánhNinh | | **Ext Supervisor** |  | | **Capstone Project code** | APOD | |
|  |

|  |
| --- |
| - Ho Chi Minh City, 09/2013 - |

# INTRODUCTION

## PROJECT INFORMATION

* Project name: **Lynxmotion A-Pod  robot controller**
* Project code: **APOD**
* Product type: **Embedded robot controller**
* Timeline: **from September 2013 to December 2013**

## TEAM MEMBER

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Name** | **Role** | **Contact** |
| **1** | PhanAnhDũngCường | Leader | * Phone: 0972708318 * Email: cuongpadse60774@fpt.edu.vn |
| **2** | Nguyễn Minh Quân | Member | * Phone: 0915699635 * Email: quannm60344@fpt.edu.vn |
| **3** | Cao ĐìnhNguyênKhoa | Member | * Phone: * Email: khoacnd60344@fpt.edu.vn |
|  |  |  |  |

Table A‑1

## APOD INTRODUCTION

### **HEXAPOD**

The trademarked name “hexapod” (by Geodetic Technology) was originally for Stewart platforms used in machine tools. However, the term is now used for 6-jack platforms outside of the machine tool area, since it simply means "six legs".

In the scope of this document, “hexapod” will represent a structure with six legs which can be used for variety of Robotics application.

Variety of robotics hexapod:



Figure A‑1

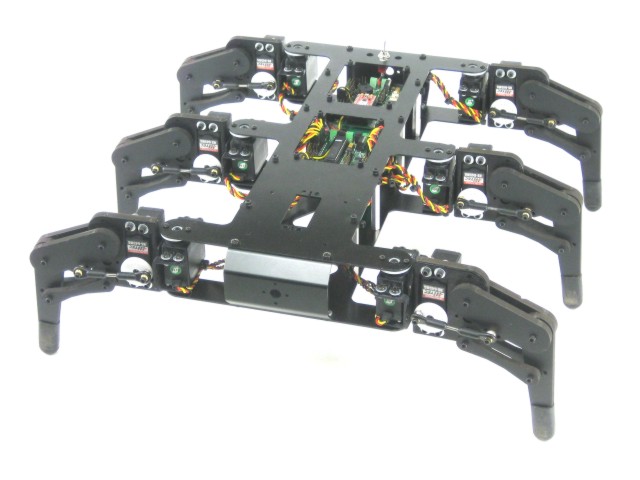


Figure A‑2

### **LYNXMOTION APOD**



Figure A‑3

* Insect inspired body.
* Assembly with 25 servos (motors) and PVC materials to form 6 legs, mandible grippers (with force sensor feature) and a tail.
* The three DOF (degree of freedom) leg design help the APOD moving in a variety of directions.
* Servos are directly controlled by SSC-32 board which receive command via RS232 interface from Bot board II / PC or any other devices communicate using RS232.

## EXISTING SOLUTIONS

* Flowbotics Studio: $39.9

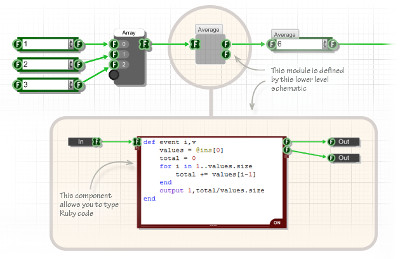


Figure A‑4

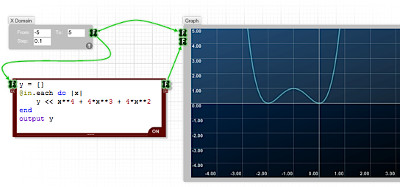


Figure A‑5

* Lynxmotion Visual Sequences: $39.94

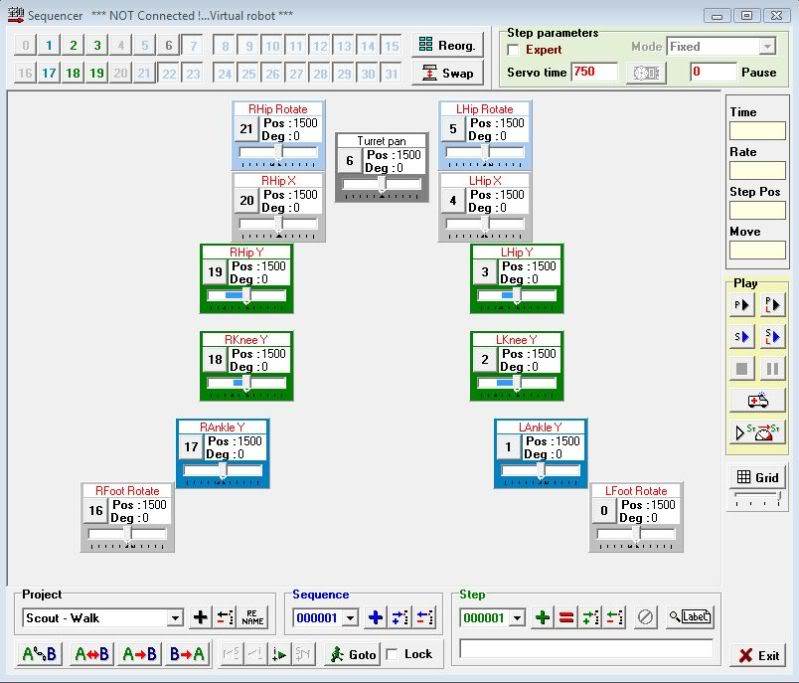


Figure A‑6

## PROJECT INTRODUCTION

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 obstacles.

Abstracts purpose: Design the controller board motor servo using microcontroller to control Lynxmotion A-Pod  robot  with available mechanical. Implement the hardware and software on PC to control Lynxmotion. The system requires Microcontroller ARM LPC,AT91SAM,PIC,AVR. Finding critical points and find best way algorithms.

## APPROACHES

* Building SSC32 firmware with ATMega328p microcontroller to control 25 servos.
* Using Bot Board II as universal sensor reader to serve automation purposes (additional sensors will be needed).
* Additional STM32F4 Discovery using ARM will be used as a central processing module which communicate with other device via Bluetooth.
* IP Camera to provide an insight view for user.
* A friendly HMI program will be provided for user to control the APOD with ease.

## OTHER

### **WHY WE NEED APOD ROBOT?**

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.

### **PRODUCT**

* HMI Program on PC (Possible extend: Android/IOS).
* Servo Control Module Software (SSC32 firmware).
* Sensor Service Module Software (Bot Board II firmware)
* Central Processing Module Software (STM32F4 Discovery).

# SOFTWARE PROJECT MANAGEMENT PLAN (SPMP)

## PROBLEM DEFINITION

### **NAME OF THIS CAPSTONE PROJECT**

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

### **PROBLEM ABSTRACT**

As mentioned above, this project will focus on implement a new control system for APOD using Bluetooth/wireless. The main purpose of new controller system is to make the APOD more flexible. In order to achieve that purpose, the final product should have those characteristics:

* + Friendly HMI interface.
  + Easy to use, easy to learn.
  + Different user experience: comfort and high interaction.
  + Adaptable to new mechanical constraint, prepared for complete automation.

### **PROBLEM OVERVIEW**

#### The Current System

Current APOD control system can be control using either USB-to-serial cable or an PS2 interface. The SSC32 will be responsible for direct controlling of motor servos.

* Advantages:
  + Direct command.
  + No delay due to no-subsystem needed (USB-to-serial control).
  + Simple architecture
* Limitation:
  + Very limited range of control due to “wire” problem
  + A considerable amount of wire/cable can be obstacles for APOD movement.
  + Domain or mechanical knowledge is required to perform.
  + On-sight performance: APOD must stay insight of user viewpoint.

#### The Propose System

The system consists of 4 module:

* + Servos Control Module (SSC-32): Directly control servo ‘s movement.
  + Sensor Service Module (Bot Board II): receive all kind of sensor signal and value.
  + Central Processing Module (STM32F4 Discovery) plus Bluetooth module
  + HMI software on PC

Main features will be provided:

* + Basic movement: forward, backward, turn left, turn right.
  + High interactive user interface.

#### Boundaries of The System

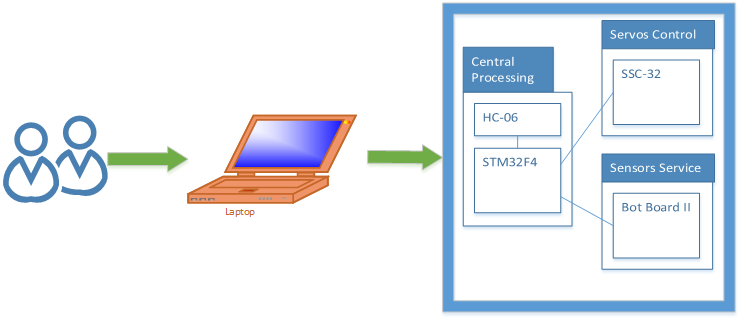


Figure B‑1

* + The APOD can perform only within Bluetooth or Wireless device ‘s range
  + Any interfere while receiving command (transmitting via Bluetooth) will cause the APOD to misbehave or unpredictable movement.

#### Developing Environment

##### Hardware environment:

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

##### Software environment:

* Developing environment ARV studio 4 for main board programming.
* 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.



Figure B‑2

### **ROLES AND RESPONSIBILITIES**

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Full name | Team Role | Responsibilities |
| 1 | TrầnKhánhNinh | Supervisor | Define business  Support in technical issues |
| 1 | PhanAnhDũngCường | Team Leader | Prioritize work  Answer question and deliver direction  Facilitate productivity – maximize team performance  Complete all individual work |
| 3. | Nguyễn Minh Quân | Team Member | Commit individual product on time  Support each other to complete team work |
| 4 | Cao ĐìnhNguyênKhoa | Team Member | Commit individual product on time  Support each other to complete team work |

Table B‑1

### **TOOL AND TECHNIQUES**

#### For Development

##### Hardware environment:

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

##### Software environment:

* Developing environment ARV studio 4 for main board programming.
* Developing environment Visual studio 2010 with C# language for user interface programming.
* Developing environment Keilcstudio 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 report
* Skype chat system
* CMS system

## PROJECT MANAGEMENT PLAN

### Task

#### 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 |

Table B‑2

#### Software Requirement 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 |

Table B‑3

#### 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 |

Table B‑4

#### Coding

|  |  |
| --- | --- |
| 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 |

Table B‑5

#### 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 |

Table B‑6

#### 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 |

Table B‑7

### Task sheet

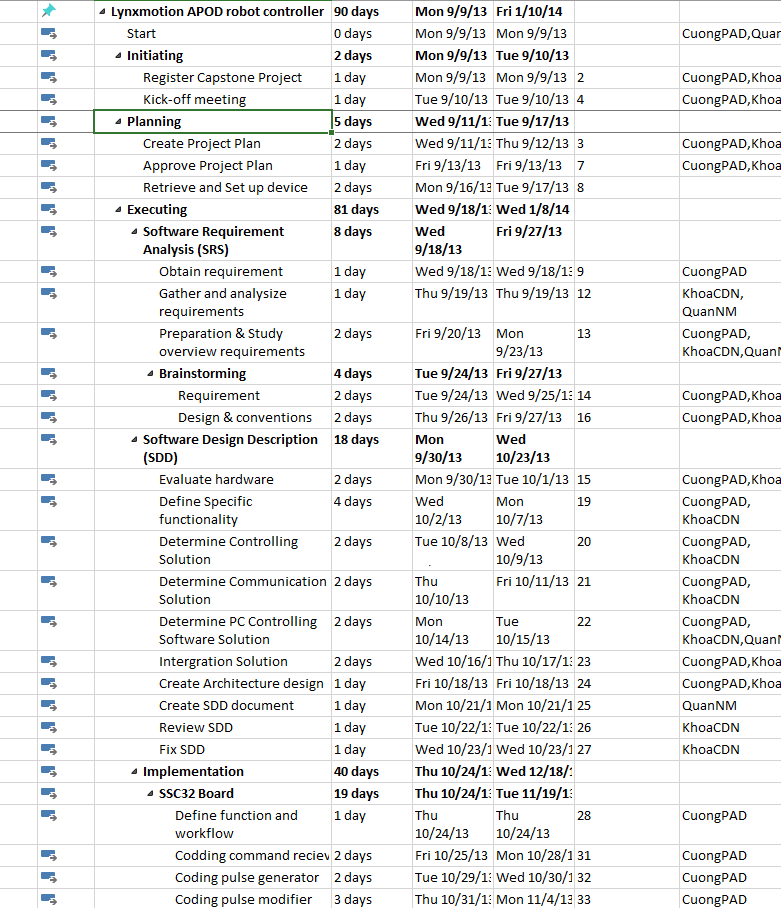


Figure 2‑B‑3



Figure 2‑2

### 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>)

# SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

## USER REQUIREMENT SPECIFICATIONS

* A-Pod should be able to walk freely in any direction.
* Controlling via a Serial connection like Bluetooth, or directly through a PS2 controller.
* A-Pod should be able to grab small things like a Coke.
* A-Pod should be able to recording video.
* A-Pod should be able to detect objects stand (within 30cm) in the ways and response back.
* Controller should be simple and easy.
* After receive movement signal, A-Pod should move within 2 seconds

## SYSTEM REQUIREMENT SPECIFICATIONS

### Hardware requirement

* 25 HS-645MG servo
* 1 SSC-32 servo controller
* 1 Bot Board II with Basic Atom Pro 28 microcontroller
* 1 HC-SR04 supersonic sensor
* 1 STM32F4 ARM
* Laptops with Bluetooth Device
* 1 PS2 controller
* Few serial cables, adapters , jumpers…

### Software interface

* Menu-driven design with : button, dialog box, screen.
* Button : movement button on the right, setting like PS2 controller.
* Dialog box : on the left, display A-Pod ‘s currently speed, object stand in way and distance.
* Screen : center of menu controller, display view from camera.

### External interface

* PS2 Controller

### Functional requirement

* Direction control : user can direct the A-Pod to turn left, right, go forward or backward.
* Camera view control : user can see the view sending by the A-Pod camera.
* Grab control : user can direct the A-Pod to grab things.
* Alarm control : response to user if any object stand in the way and its distance.
* Remote control : enable remote controlling, help the user control the A-Pod through a programmed Bluetooth device.

### Non-functional requirement

* The A-Pod should response and move after receiving order within 2 seconds.
* The longest range for leg movement should be 30 centimeter, 35 degree.
* Detected range should be larger than 2 meter.
* Raise alarm sound and open Alarm Dialog Box in user interface when distance between object and the A-Pod < 20 centimeter.
* View from camera should be refresh every 0.5 second.
* The A-Pod must be compatible with the following batteries and chargers:
  + NiCad & Ni-MH Universal Smart Charger (USC-02)
  + Volt Ni-MH 2800mAh Battery Pack
* All cable, jumper, adapter should be corrected connect

## USE-CASE DIAGRAM AND USE-CASE SPECIFICATIONS

### Use-case Diagram

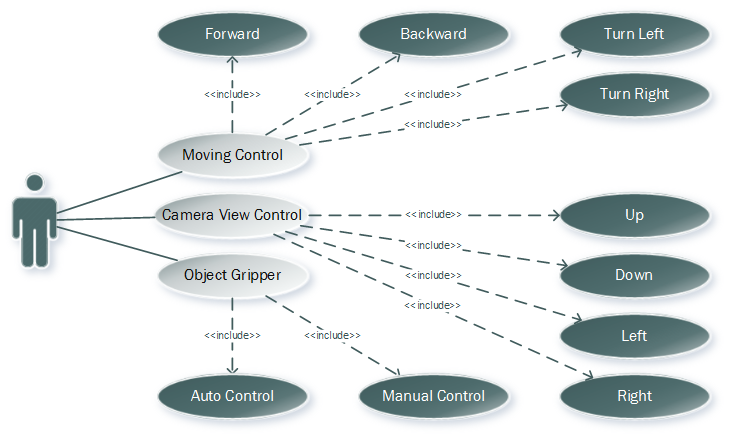
****

Figure C‑1

### Use-case Specifications

#### Direction control :

* Description : Using this control, User can direct the A-Pod to move as him/her want
* Flow of events:

|  |  |  |
| --- | --- | --- |
|  | User | System |
| Normal flow | 1.User click (Left/Right/Forward/Backward) button. | 1. Left side do not detect any obstacle.  2. The A-Pod move itself to the (Left/Right/Forward/Backward) side. |
| Alternative flow | 1.User click (Left/Right/Forward/Backward) button.  2.Dialog Box appear. User choose to “Continue” or “Stop” | 1. Left side detect an obstacle  2. Response back to the User.  3. The A-Pod move itself to the (Left/Right/Forward/Backward) side if the User choose “Continue”. Stay if the User choose “Stop” |

#### Detect object control:

* Description : this control detect any object appear in the way, calculate the distance between that object and response back.
* Flow of events:

|  |  |  |
| --- | --- | --- |
|  | User Control Menu | System |
| Normal flow | 1.Receive distance from system  2. A dialog box show up : “Warning, Object in the way within 20cm !!!” | 1.System detect an object and response its distance back to user  2. Distance <20 cm, raise alarm sound, send signal to User Control Menu |
| Alternative flow | 1. Distance : 500 cm ( no Object) | 1.System detect nothing |

#### View control:

* Description : this control display the View from the A-Pod camera.
* Flow of events:

|  |  |  |
| --- | --- | --- |
|  | User | System |
| Normal flow | 1.User click View button  2.Screen display the A-Pod’s camera view | 1. Camera start recording and transmitting data back to the User |
| Alternative flow |  |  |

## SOFTWARE SYSTEM ATTRIBUTES

### Reliability

* Easy to upgrade firmware.
* The hexapod controller can be replaced easily by loaded into chip if the controller has problem.
* The APOD system is guarantee by quality testing in:
  + Stability constraints.
  + Functionality.
  + Reliability.
* It’s mostly depending on hardware reliability.
* Small error margin when moving.

### Availability

* In case of electrical incident, the hexapod system will be shut down and reset automatically.
* Controller has the flexibility that allows changes in hardware design.
* Hexapod Controller can actuate the 6-legs forces to properly position the mobile plate given a desired trajectory.

### Security

* N/A

### Maintainability

* N/A

### Portability

* The hexapod controller is depend on hardware so that hexapod system do not have the portability attribute. However, the hexapod controller can easily be loaded into chips and use in different motors.
* The development environment and the language constructs used ensure portability as much as possible to avoid the limitation of software’s changes each time the hardware is upgraded or replaced.

### Performance

* Hexapod controller can control multi-motors in the same time.
* The current hardware can move correctly

## ENTITY RELATIONSHIP DIAGRAM

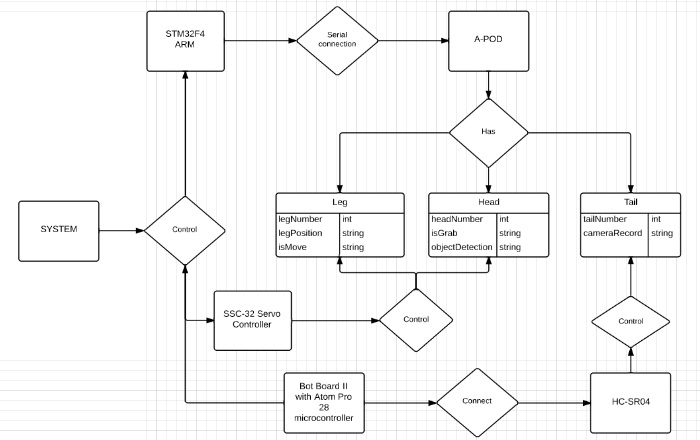


Figure C‑2

# SOFTWARE DESIGN DESCRIPTION (SDD)

## DESIGN OVERVIEW

### Introduction

This section is created to introduce and give a brief overview of the system design.

The followings information are given in this section:

* System Architecture Design : the system overall structure
* Components Description : detailed info of hardware components
* Class Diagram : entity & attributes
* Sequence Diagram : describe the flow of events
* User Interface Design

### Architecture

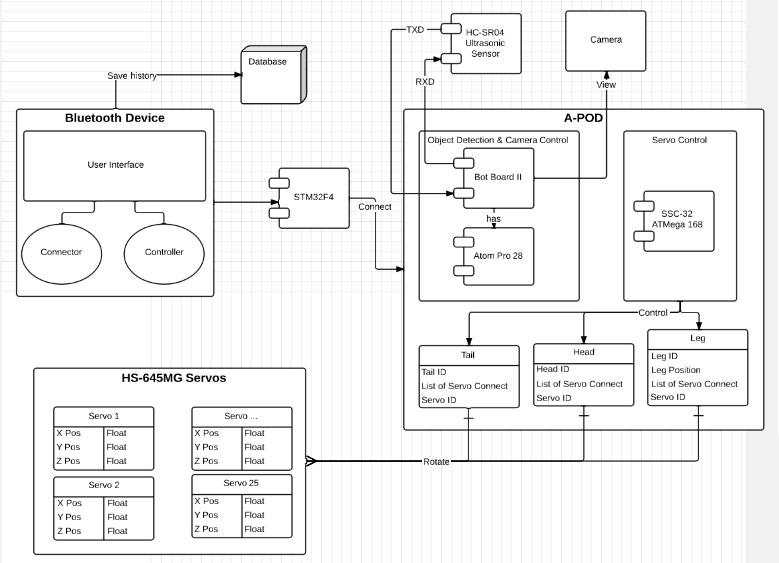


Figure D‑1

### User interface

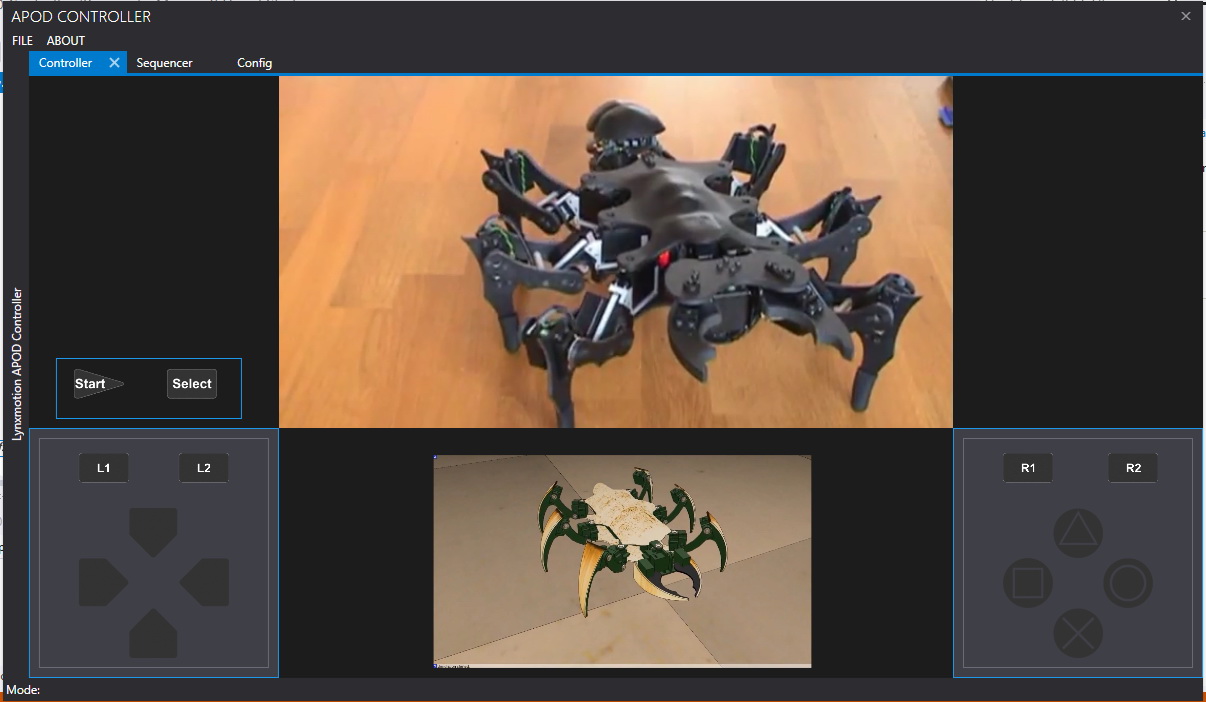


Figure D‑2

## COMPONENTS

### Central processing:

#### Bluetooth module (HC 05)

Communication module (using Bluetooth) to transmit and receive commands from Controller devices (PC)

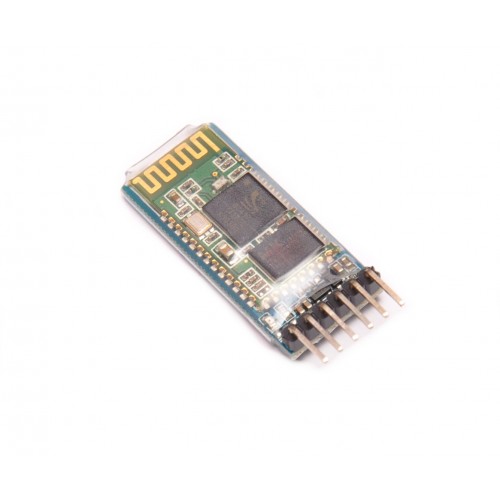


Figure D‑3

#### STM32F4 Discovery

The Center Processing module to control all APOD movements or reaction to a specific event (command or obstacle).

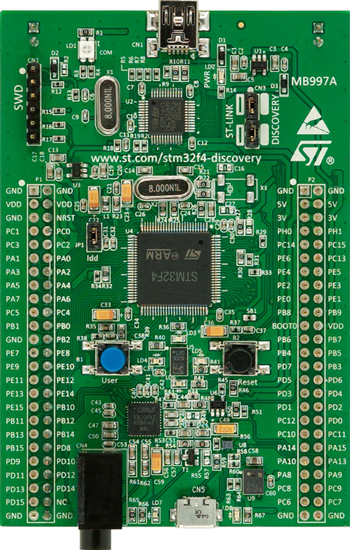


Figure D‑4

### Servos control:

#### HG-645MG Servo:

* Metal gear servo that provided the base of APOD movement.



Figure D‑5

#### SSC32 with ATMEGA328P:

* A servo controller : contains 32 pin channels of 1uS resolution servo control
* For further information, followings this link : [Lynxmotion SSC32](http://www.lynxmotion.com/images/html/build136.htm)

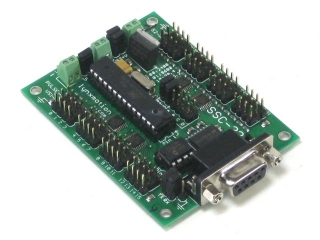


Figure D‑6

### Sensors service:

#### Ultrasonic ranging module (HC SR04)

* Ultrasonic module using for distance measurement, discovering obstacles.

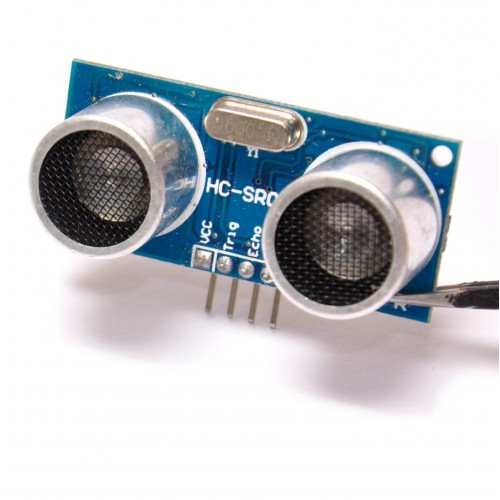


Figure D‑7

#### Bot Board II

* The Bot Board II is a carrier for the Basic Atom microcontrollers.
* For detail information, followings this link : [Bot Board II](http://www.lynxmotion.com/images/html/build151.htm)

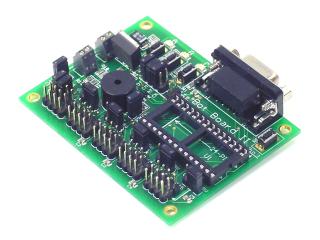


Figure D‑8

#### Basic Atom Pro 28

* Controller for Bot Board II.

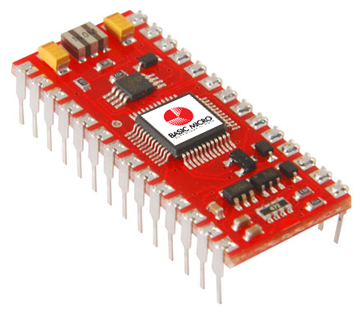


Figure D‑9

## ACTIVITY AND CLASS STRUCTURE

### Activity

#### HMI

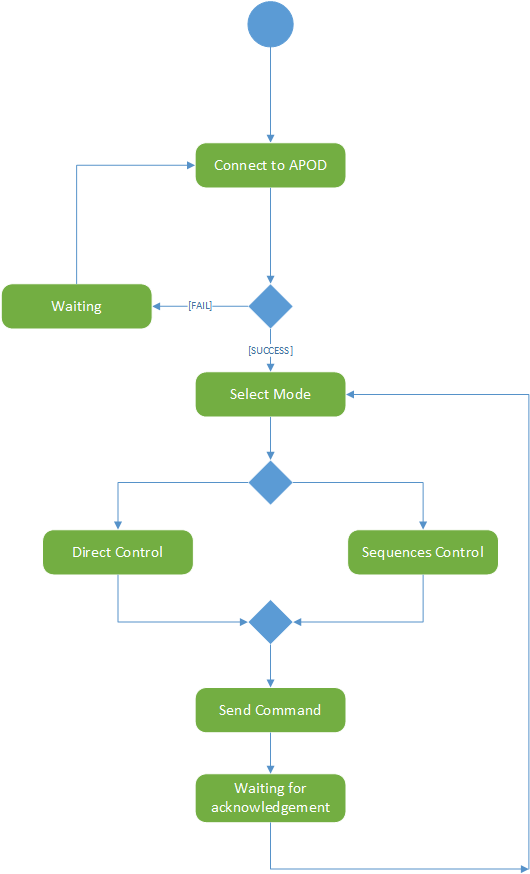


Figure D‑10

#### Embedded Controller

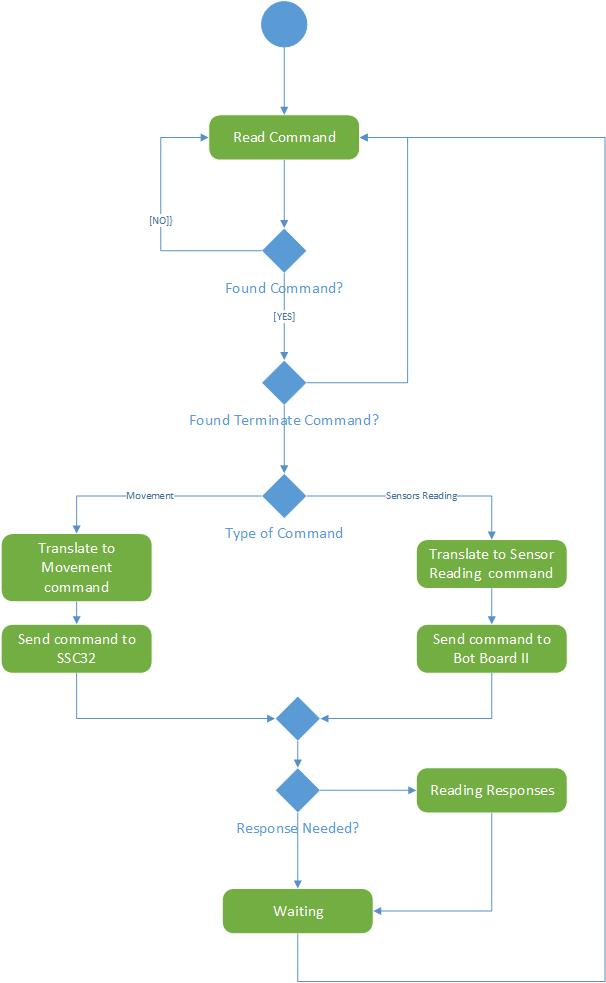


Figure D‑11

### Class Structure

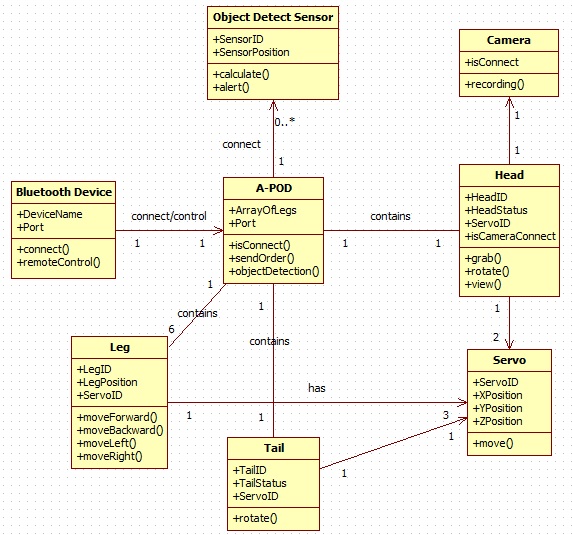
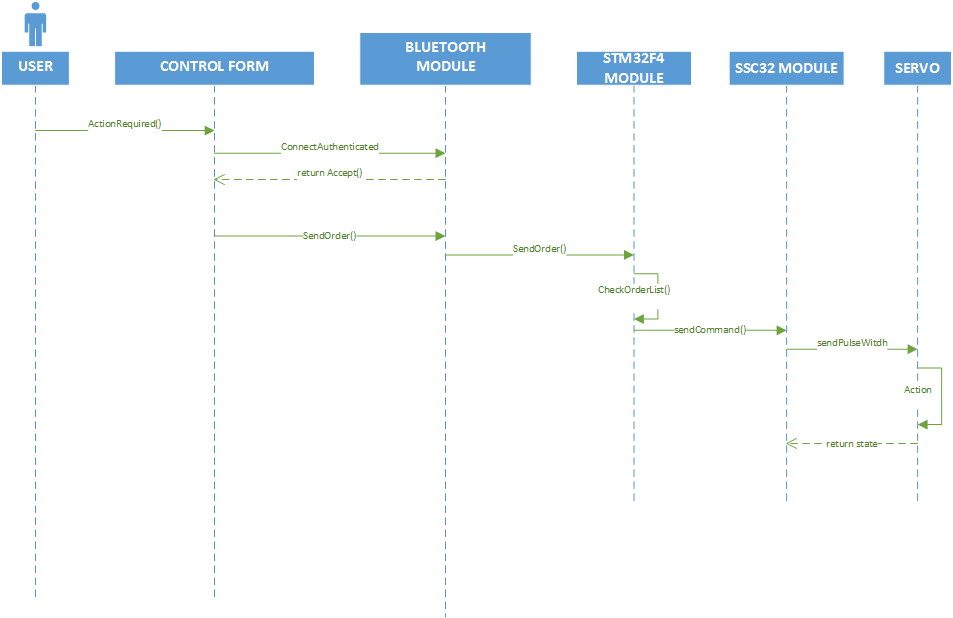


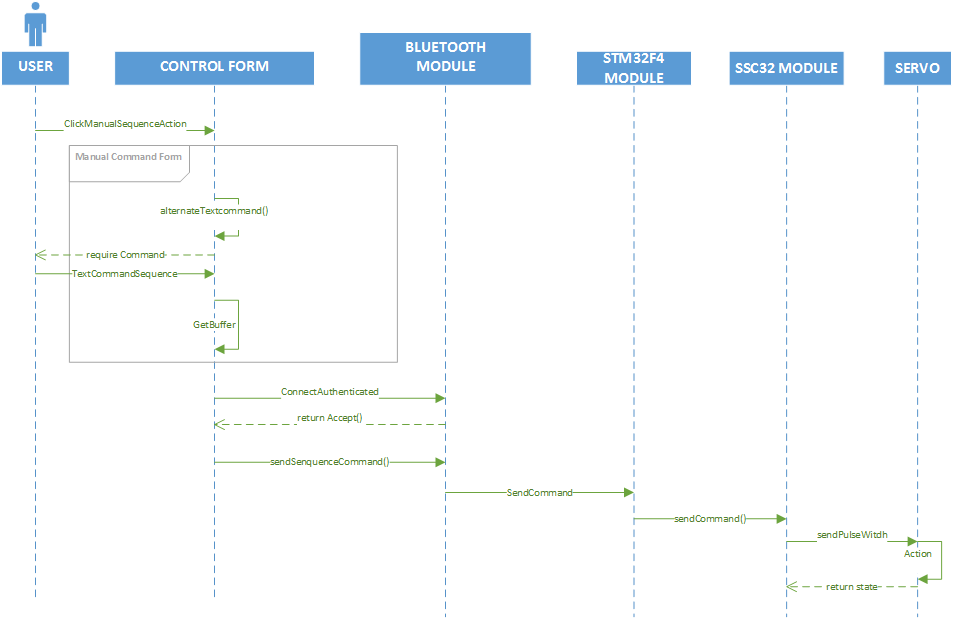
Figure D‑12

## SEQUENCE DIAGRAM



Direct control sequence diagram

Figure D‑13



Manual control sequence diagram

Figure D‑14

## USER INTERFACE DESIGN

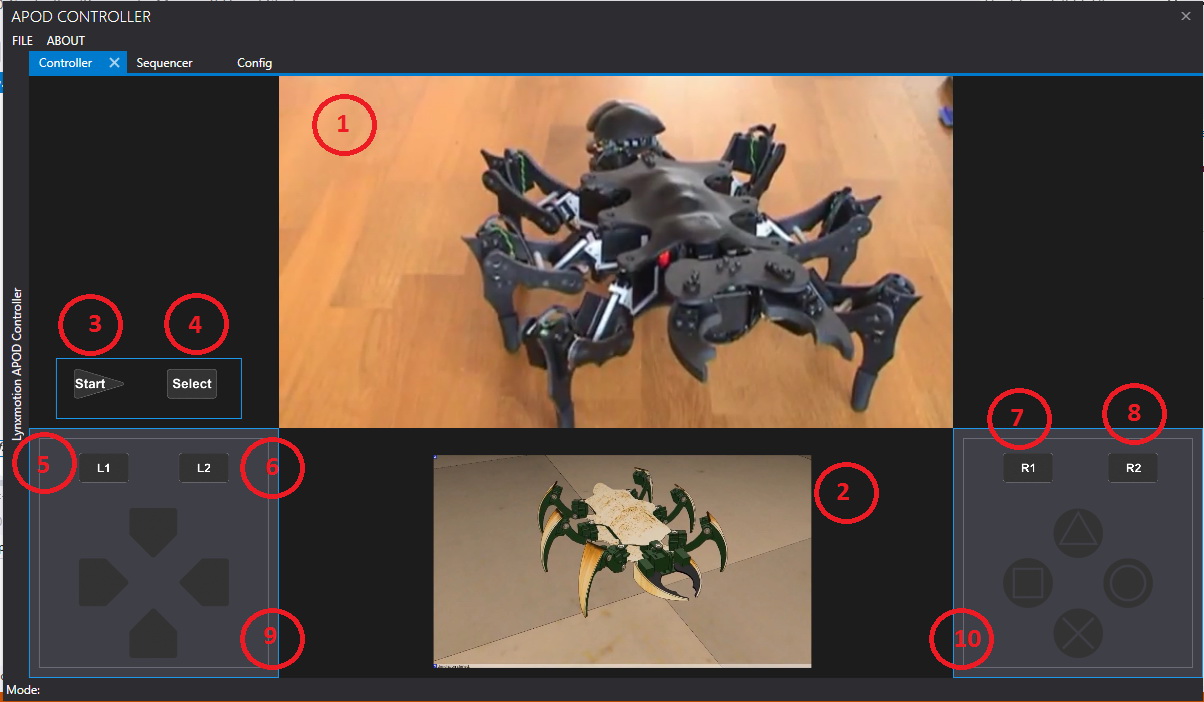


Figure D‑14

|  |  |  |
| --- | --- | --- |
| Item No. | Component Name | Description |
| 1 | vCamera | * Live Camera view from APOD’s camera |
| 2 | vSimulator | * Simulator view |
| 3 | btnStart | * Start button |
| 4 | btnSelect | * Select function button |
| 5 | btnL1 | * Parts bank selection 1 |
| 6 | btnL2 | * Parts bank selection 2 |
| 7 | btnR1 | * Action bank selection 1 |
| 8 | btnR2 | * Action bank selection 2 |
| 9 | pnlNavigation | * Navigation selection (4 direction for each part) |
| 10 | pnlAction | * Action selection (4 action for each action bank) |

Table D‑1

# SOFTWARE TEST DOCUMENTATION (STD)

## Introduction

### System Overview:

In this section, there has all necessary information about test plan, test process, test approach, test environment, test pass/fail criteria, test result and checklist to check when to test this system.

### Test Approach:

White box testing : Developers self-test on code in which function they developed (unit test).

Black box testing : Test on each function of each module in system to ensure each module execute the right function. Then do integration testing to test the function. Finally, system test to test the whole system.

Testing Goal : Discover bug , fix code, regression test and finally completed system.

## Test Plan

The purpose of this phase is to list and verify the functions in system that need to be tested and completed. In this process, we will check to ensure that function meet specification and requirement in system. Bugs ( errors ) will not occur after testing.

The next content will describe which function will be tested and which will not and plan for them.

### Feature to be tested :

|  |  |  |  |
| --- | --- | --- | --- |
| Feature to be tested | Test description | Test date | Responsibilities |
| Bluetooth Connection | Test port connection | 1/11/2013 | QuanNM |
| Test sending data | 4/11/2013 |
| Test receiving data | 4/11/2013 |
| Controller Interface | Test live controller tab | 1/11/2013 | CuongPAD |
| Test configuration tab | 3/11/2013 |
| Test sequence player tab | 6/11/2013 |
| A-POD | Test legs function | 6/11/2013 | QuanNM |
| Test Tripod Function | 9/11/2013 |
| Test head/tail function | 12/11/2013 |
| Test A-POD full body movements | 12/11/2013 |
| Servo Control | Test Pulse Width | 8/11/2013 | CuongPAD |
| Test Clock Gen | 8/11/2013 |
| Additional Device | Test camera recording | 10/11/2013 | CuongPAD |
| Test sensor signal | 12/11/2013 |
| System | Final system testing | 25/11/2013 | All team members |

### Test environment :

Operation System : Win 7/8.1

Installed Software : Microsoft Visual Studio 2012 ( newest version)

Real Device : A-Pod, Camera device, Bluetooth device, PS2 Controller

### Test pass/fail criteria :

Depend on significant the problem is. It affects a critical function or a peripheral one.

For system testing, the criteria are:

* 90% of the test cases must pass
* All test cases dealing with critical functionality must pass
* All medium and high severity defects must be fixed
* Test coverage must be at least 90%

## Test Case

### Test : Bluetooth Connection

#### Integration test: Functional test between PC and HC\_05 using Bluetooth connection

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| No. | Function / Feature | Pre-condition | Step of testing | Expected Result | Created by | Executed by | Result |
| UC1 | Connect | Active bluetooth on PC | 1. Click “Find” button  2. Search from devices list and select HC\_05 device.  3. Click OK. | 1. Refresh and update List of Bluetooth Devices nearby.  2. Detail information about HC\_05 Bluetooth Device show  3. Connected succesfully | KhoaCDN | QuanNM | Passed |
| UC2 | Send / Receive data | Devices connected | 1. Prepare a data package : 0xAA  2. Click “Send data” button | On HC\_05: Receiving data, reply back to PC  On PC monitor : Get data back : 0xAA | KhoaCDN | QuanNM | Passed |

#### Performance test : PC connect to HC\_05 using Bluetooth connection

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| No. | Function / Feature | Pre-condition | Step of testing | Expected Result | Created by | Executed by | Result |
| UC3 | Send/ Receive data | Devices connected | 1. Sending ten data packages | 9 of 10 package received successfully | KhoaCDN | QuanNM | Passed |
| UC4 | Send/ Receive data | Devices connected | 1. Sending 100 bytes data package | 100 bytes received successfully | KhoaCDN | QuanNM | Passed |
|  |  |  |  |  |  |  |  |

#### 

### Test : Controller Interface

#### Integration Test : Controller Main Window , Live Controller tab with Camera, PS2 Controller plugged-in.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| No. | Function / Feature | Pre-condition | Step of testing | Expected Result | Created by | Executed by | Result |
| UC5 | Main Window | None | 1. Start program | Controller Screen appear without bugs or crash.  Live Controller tab is selected by default.  Default control mode in Live controller tab is Direct Control.  Initial all Component. | KhoaCDN | CuongPAD | Passed |
| UC6 | Live Controller tab :  Movement () | 1.Start program | 1. Mouse click one of these button : Up – Down – Left- Right ( arrow), Square – Triangle – X – O, Start, Stop  2. Button release | Pressed button switch into red color  After release, button came back to initial state. | KhoaCDN | CuongPAD | Passed |
| UC7 | 1. Click “Direct Control” button  2. Re-peat UC2 step | Controller switch to Direct Control mode : control A-Pod by mouse clicking. | KhoaCDN | CuongPAD | Passed |
| UC8 | 1. Click “Keypad” button | Message appear : “No PS2 Controller plug-in” | KhoaCDN | CuongPAD | Passed |
| UC9 | 1.Start Program  2. Plug-in a PS2 Controller | 1. Click “Keypad” button | Controller switch into PS2 Mode  Text appear in corner : “PS2 mode” | KhoaCDN | CuongPAD | Passed |
| UC10 | 1.Start Program  2. Plug-in a PS2 Controller  3. PS2 Mode on | 1. Repeat UC2 | No button changed | KhoaCDN | CuongPAD | Passed |
| UC11 | 1. Press a PS2 controller button  2. Button release | Pressed button switch into red color  After release, button came back to initial state. | KhoaCDN | CuongPAD | Passed |
| UC12 | 1.Start Program  2. Plug-in a PS2 Controller  3. PS2 Mode on | 1.Press and hold two or more PS buttons.  2. Release one button.  3. Press another button while keep holding ones  4. Release all button | 1. All pressed button switch into red color  2. The released button come back to initial state.  3. New pressed button switch into red color  4. All release buttons come back to initial state. | KhoaCDN | CuongPAD | Passed |
| UC13 | 1.Start Program | 1. Click “Run Sequence” button | All buttons except “Start”, “Stop” changed into invisibled  Text appear in corner : “Auto mode” | KhoaCDN | CuongPAD | Passed |
| UC14 | 1.Start Program  2. Auto mode on  3. Sequence not setup in Sequence Player tab yet | 1. Click “Start” button | Message appear : “No sequence found ! Please setup the states in Sequence Player tab.” | KhoaCDN | CuongPAD | Passed |
| UC15 | Live Controller tab:  Camera Control() | 1.Start Program  2. Camera un-plug | 1. Click “Camera” button | Message appear : “No camera device found ! Please re-check your plug-in.” | KhoaCDN | CuongPAD | Passed |
| UC16 | 1. Start Program  2. Camera plug-in | 1. Click “Camera” button  2. Turn around camera | 1.Center monitor on. Live view from camera display on the monitor.  2. View update on monitor in within 1.5 seconds  3. Center monitor off. | KhoaCDN | CuongPAD | Passed |
| UC15 | Live Controller tab: Object Tracking() | 1.Start Program | 1. Click Object Tracking button | Text appear : Color Tracking – Form Tracking | KhoaCDN | CuongPAD | Passed |
| UC16 | 1. Start Program  2. Object Tracking clicked  3. Camera un-plug | 1. Select Color Tracking  2. Choose a color in Dialog then click OK | 1. Color dialog appear.  2. Chose color appear on the left corner. | KhoaCDN | CuongPAD | Passed |
| UC17 | 1. Start Program  2. Object Tracking clicked  3. Camera plug-in  4. Get a test object which has same color with the Chose color in front of the camera | 1. Select Color Tracking  2. Choose a color in Dialog then click OK  3. Repeat UC15 steps | 1. Color dialog appear.  2. Chose color appear on the left corner.  3. Center Monitor on. Camera view display on Monitor.  4. Color detected with square border surrounded | KhoaCDN | CuongPAD | Passed |
| UC18 | 1. Start Program  2. Object Tracking clicked  3. Camera un-plug | 1. Select Form Tracking  2. Choose a Form in Dialog then click OK | 1. Form dialog appear.  2. Chose Form appear on the left corner. | KhoaCDN | CuongPAD | Passed |
| UC19 | 1. Start Program  2. Object Tracking clicked  3. Camera plug-in  4. Get a test object which has same Form with the selected Object in front of the camera | 1. Select Form Tracking  2. Choose a Form in Dialog then click OK  3. Repeat UC15 steps | 1. Form dialog appear.  2. Chose Form appear on the left corner.  3. Center Monitor on. Camera view display on Monitor.  4. Form detected with square border surrounded | KhoaCDN | CuongPAD | Passed |
| UC20 | ? | ? | ? |  |  |  |  |

#### Unit Test : Sequence Player tab.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| No. | Function / Feature | Pre-condition | Step of testing | Expected Result | Created by | Executed by | Result |
| UC21 | Add state() | 1. Start Program  2. Switch to Sequence Player tab | 1. Click Add button  2. Input command  3. Click Ok button | 1. New state row created  2. State added | KhoaCDN | CuongPAD | Passed |
| UC22 | Remove state() | 1. Start Program  2. Switch to Sequence Player tab | 1. Choose a State row  2. Click remove button or (x) button  3. Confirm action | 1. Selected row highlight up.  2. Appear confirm Dialog  3. State removed | KhoaCDN | CuongPAD | Passed |
| UC23 | Export() | 1. Start Program  2. Switch to Sequence Player tab  3. No state added | 1. Click Export button | 1. Appear message: “ No state found ! Please check again!” | KhoaCDN | CuongPAD | Passed |
| UC24 | Export() | 1. Start Program  2. Switch to Sequence Player tab  3. State added | 1. Click Export button  2. Chose file location & name. Then click Save | 1. Save File Dialog appear  2. File saved successful, with file extension : “.xml” | KhoaCDN | CuongPAD | Passed |
| UC25 | Import() | 1. Start Program  2. Switch to Sequence Player tab  3. A test .xml file | 1. Click Import button  2. Chose test file. Then click Open | 1. Open File Dialog appear  2. File opened successfully. State added into rows | KhoaCDN | CuongPAD | Passed |
| UC26 | Import() | 1. Start Program  2. Switch to Sequence Player tab | 1. Click Import button  2. Chose a random xml file. | ???? |  |  |  |

#### Unit Test : Configuration tab.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| No. | Function / Feature | Pre-condition | Step of testing | Expected Result | Created by | Executed by | Result |
| UC27 | N/A |  |  |  |  |  |  |

### Test : A-POD (->\*Apod.c)

#### Function test : Leg, Neck, Mandible movement

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| No. | Function / Feature | Pre-condition | Step of testing | Expected Result | Created by | Executed by | Result |
| UC28 | Mandible() | Reset all mandible servos (S3,S19) to Init Position (1500) | 1. Add mandible (0,200)  2. Add mandible (1,200) | 1. Servo 3 increase position to 1700 (go to the right)  2. Servo 19 decrease position to 1300 ( go to the left) | KhoaCDN | QuanNM | Passed |
| UC29 | Neck\_Rotate() | Reset all neck servos (S13,S14,S29) to Init Position (1500) | 1. Add Neck\_Rotate(-200)  2. Add Neck\_Rotate(200) | 1. Servo 13 increase position to 1700 (rotate to the right)  2. Servo 13 decrease position to 1300 ( rotate to the left ) | KhoaCDN | QuanNM | Passed |
| UC30 | Neck\_Horizontal | Reset all neck servos (S13,S14,S29) to Init Position (1500) | 1. Add Neck\_Horizontal (-200)  2. Add Neck\_Horizontal (200) | 1. Neck (S14) turn to the right  2. Neck (S14)turn to the left | KhoaCDN | QuanNM | Passed |
| UC31 | Neck\_Vertical() | Reset all neck servos (S13,S14,S29) to Init Position (1500) | 1. Add Neck\_Vertical(-200)  2. Add Neck\_Vertical(200) | 1. Neck (S29) go Up  2. Neck (S29) go Down | KhoaCDN | QuanNM | Passed |
| UC32 | Leg\_Lift() | Call Leg\_Reset\_All() function | 1. Add Leg\_Lift(0,200)  2. Add Leg\_Lift(0,200)  3. Add Leg\_Lift(0,200)  4. Add Leg\_Lift(0,200)  5. Add Leg\_Lift(0,200)  6. Add Leg\_Lift(0,200) | 1. Right Front Leg (S1) go Up  2. Right Front Leg (S5) go Up  3. Right Front Leg (S9) go Up  4. Left Front Leg (S17) go Up  5. Left Front Leg (S21) go Up  6. Left Front Leg (S25) go Up | KhoaCDN | QuanNM | Passed |
| UC33 | Leg\_Drop() | Call Leg\_Reset\_All() function | 2. Add Leg\_ Drop (0,200)  2. Add Leg\_ Drop (1,200)  3. Add Leg\_ Drop (2,200)  4. Add Leg\_ Drop (3,200)  5. Add Leg\_ Drop (4,200)  6. Add Leg\_ Drop (5,200) | 1. Right Front Leg (S1) go Down  2. Right Front Leg (S5) go Down  3. Right Front Leg (S9) go Down  4. Left Front Leg (S17) go Down  5. Left Front Leg (S21) go Down  6. Left Front Leg (S25) go Down | KhoaCDN | QuanNM | Passed |
| UC34 | Leg\_Stand() | Call Leg\_Reset\_All() function | 1.Add Leg\_Stand (0)  2.Add Leg\_Stand (1)  3.Add Leg\_Stand (2)  4.Add Leg\_Stand (3)  5.Add Leg\_Stand (4)  6.Add Leg\_Stand (5) | 1. S2 Leg get the same position with S1’s.  1. S6 Leg get the same position with S5’s.  1. S10 Leg get the same position with S9’s.  1. S18 Leg get the same position with S17’s.  1. S22 Leg get the same position with S21’s.  1. S26 Leg get the same position with S25’s. | KhoaCDN | QuanNM | Passed |
| UC35 | Leg\_Forward() | Call Leg\_Reset\_All() function | 1.Add Leg\_Forward (0,200)  2.Add Leg\_Forward (1,200)  3.Add Leg\_Forward (2,200)  4.Add Leg\_Forward (3,200)  5.Add Leg\_Forward (4,200)  6.Add Leg\_Forward (5,200) | 1. Right Front Leg (S0) move forward  2. Right Front Leg (S4) move forward  3. Right Front Leg (S8) move forward  4. Left Front Leg (S16) move forward  5. Left Front Leg (S20) move forward  6. Left Front Leg (S24) move forward | KhoaCDN | QuanNM | Passed |
| UC36 | Leg\_Backward() | Call Leg\_Reset\_All() function | 1.Add Leg\_ Backward (0,200)  2.Add Leg\_ Backward (1,200)  3.Add Leg\_ Backward (2,200)  4.Add Leg\_ Backward (3,200)  5.Add Leg\_ Backward (4,200)  6.Add Leg\_ Backward (5,200) | 1. Right Front Leg (S0) move backward  2. Right Front Leg (S4) move backward  3. Right Front Leg (S8) move backward  4. Left Front Leg (S16) move backward  5. Left Front Leg (S20) move backward  6. Left Front Leg (S24) move backward | KhoaCDN | QuanNM | Passed |
| UC37 | Leg\_Reset() |  | 1. Add Leg\_Reset (0)  2. Add Leg\_Reset (1)  3. Add Leg\_Reset (2)  4. Add Leg\_Reset (3)  5. Add Leg\_Reset (4)  6. Add Leg\_Reset (5) | 1. Reset Right Front Legs to Init Position. (S0 = 1500, S1-S2 = 1800)  2. Reset Right Center Legs to Init Position. (S4 = 1500, S5-S6 = 1800)  3. Reset Right Rear Legs to Init Position. (S8 = 1500, S9-S10 = 1800)  4. Reset Left Front Legs to Init Position. (S16 = 1500, S17-S18 = 1200)  5. Reset Left Front Legs to Init Position. (S20 = 1500, S21-S22 = 1200)  6. Reset Left Front Legs to Init Position. (S24 = 1500, S25-S26 = 1200) | KhoaCDN | QuanNM | Passed |
| UC38 | Leg\_Reset\_All() |  | 1. Add Leg\_Reset\_All() | 1. Reset ALL Legs to Init Position. (see UC37) | KhoaCDN | QuanNM | Passed |

#### Function test : Tripod A, Tripod B movement

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| No. | Function / Feature | Pre-condition | Step of testing | Expected Result | Created by | Executed by | Result |
| UC39 | Tripod\_A\_Lift() | Call Leg\_Reset\_All() function | 1. Add Tripod\_A\_Lift(200) | 1. Tripod-A Legs (Left Front, Left Rear, Right Center) go Up | KhoaCDN | QuanNM | Passed |
| UC40 | Tripod\_A\_Drop() | Call Leg\_Reset\_All() function | 1. Add Tripod\_A\_Drop(200) | 1. Tripod-A Legs (Left Front, Left Rear, Right Center) go Down | KhoaCDN | QuanNM | Passed |
| UC41 | Tripod\_A\_Forward() | Call Leg\_Reset\_All() function | 1. Add Tripod\_A\_Forward (200) | 1. Tripod-A Legs (Left Front, Left Rear, Right Center) move Forward | KhoaCDN | QuanNM | Passed |
| UC42 | Tripod\_A\_Backward() | Call Leg\_Reset\_All() function | 1. Add Tripod\_A\_Backward (200) | 1. Tripod-A Legs (Left Front, Left Rear, Right Center) move Backward | KhoaCDN | QuanNM | Passed |
| UC43 | Tripod\_A\_Left() | Call Leg\_Reset\_All() function | 1. Add Tripod\_A\_Left(200) | 1. Tripod-A Legs turn left. ( Right Center Legs move Forward, Left Front & Left Rear move Backward ) | KhoaCDN | QuanNM | Passed |
| UC44 | Tripod\_A\_Right() | Call Leg\_Reset\_All() function | 1. Add Tripod\_A\_Right(200) | 1. Tripod-A Legs turn right. ( Right Center Legs move Backward, Left Front & Left Rear move Forward) | KhoaCDN | QuanNM | Passed |
| UC45 | Tripod\_B\_Lift() | Call Leg\_Reset\_All() function | 1. Add Tripod\_B\_Lift(200) | 1. Tripod-B Legs (Right Front, Right Rear, Left Center) go Up | KhoaCDN | QuanNM | Passed |
| UC46 | Tripod\_B\_Drop() | Call Leg\_Reset\_All() function | 1. Add Tripod\_B\_Drop(200) | 1. Tripod-B Legs (Right Front, Right Rear, Left Center) go Up | KhoaCDN | QuanNM | Passed |
| UC47 | Tripod\_B\_Forward() | Call Leg\_Reset\_All() function | 1. Add Tripod\_B\_Forward (200) | 1. Tripod-B Legs (Right Front, Right Rear, Left Center) move Forward | KhoaCDN | QuanNM | Passed |
| UC48 | Tripod\_B\_Backward() | Call Leg\_Reset\_All() function | 1. Add Tripod\_B\_Backward (200) | 1. Tripod-B Legs (Right Front, Right Rear, Left Center) move Backward | KhoaCDN | QuanNM | Passed |
| UC49 | Tripod\_B\_Left() | Call Leg\_Reset\_All() function | 1. Add Tripod\_B\_Left(200) | 1. Tripod-B Legs turn left. ( Left Center Legs move Backward, Right Front & Right Rear move Forward ) | KhoaCDN | QuanNM | Passed |
| UC50 | Tripod\_B\_Right() | Call Leg\_Reset\_All() function | 1. Add Tripod\_B\_Right(200) | 1. Tripod-B Legs turn right. ( Left Center Legs move Forward, Right Front & Right Rear move Backward) | KhoaCDN | QuanNM | Passed |

#### Function test : A-Pod Control

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| No. | Function / Feature | Pre-condition | Step of testing | Expected Result | Created by | Executed by | Result |
| UC51 | APOD\_Forward() | Call Leg\_Reset\_All() function | 1. Add APOD\_Forward  (1, 100, 200, 300)  2. Add APOD\_Forward  (3, 200, 100, 500) | A-Pod moving Forward successfully ( Correct move through 7 State - View “Hexapod.xls” for detail)  1.1. Delay time = 300ms, Loop = 1 Time  1.2. Horizontal interval is 200, Vertical interval is 100  2.1 Delay time = 500ms, Loop =3 Times  2.2. Horizontal interval is 100, Vertical interval is 200 | KhoaCDN | QuanNM | Passed |
| UC52 | APOD\_Backward() | Call Leg\_Reset\_All() function | 1. Add APOD\_Backward  (1, 100, 200, 300)  2. Add APOD\_Backward  (3, 200, 100, 500) | A-Pod moving Backward successfully ( Correct move through 7 States - View “Hexapod.xls” for detail)  Input values is same with UC51 | KhoaCDN | QuanNM | Passed |
| UC53 | APOD\_TurnLeft() | Call Leg\_Reset\_All() function | 1. Add APOD\_TurnLeft  (1, 100, 200, 300)  2. Add APOD\_TurnLeft  (3, 200, 100, 500) | A-Pod turn Left successfully ( Correct move through 7 States - View “Hexapod.xls” for detail)  Input values is same with UC51 | KhoaCDN | QuanNM | Passed |
| UC54 | APOD\_TurnRight() | Call Leg\_Reset\_All() function | 1. Add APOD\_ TurnLeft  (1, 100, 200, 300)  2. Add APOD\_TurnLeft  (3, 200, 100, 500) | A-Pod turn Right successfully ( Correct move through 7 States - View “Hexapod.xls” for detail)  Input values is same with UC51 | KhoaCDN | QuanNM | Passed |
| UC55 | APOD\_WaveTail() | Call Tail\_Wave |  |  |  |  |  |
| UC56 | APOD\_Lift() | Call Leg\_Reset\_All() function | 1. Add APOD\_Lift(500)  2. Add APOD\_Lift(700) | 1. APOD raise up, Servo Legs 1-5-9-17-21-25 increase to Position 2000  2. Position (1500+700) > 2100 (code limit), APOD do nothing. | KhoaCDN | QuanNM | Passed |
| UC57 | APOD\_Drop() | Call Leg\_Reset\_All() function | 1. Add APOD\_Drop(300)  2. Add APOD\_Drop(500) | 1. APOD lower down, Servo Legs 1-5-9-17-21-25 decrease to Position 1200  2. Position (1500-500) <=1000 (code limit), APOD do nothing | KhoaCDN | QuanNM | Passed |
| UC58 | APOD\_Neck\_Rotate\_Left() | Reset Servo13 to Init Position (1500) | 1. Add APOD\_Neck\_Rotate\_Left (400)  2. Add APOD\_Neck\_Rotate\_Left (800) | 1. Servo 13 increase to 1900 Position, A-POD neck turn Left.  1. Position (1500+800) >2100 (code limit), A-POD do nothing. | KhoaCDN | QuanNM | Passed |
| UC59 | APOD\_Neck\_Rotate\_Right() | Reset Servo13 to Init Position (1500) | 1. Add APOD\_Neck\_Rotate\_Right (400)  2. Add APOD\_Neck\_Rotate\_Right (800) | 1. Servo 13 decrease to 1100 Position, A-POD neck turn Right.  1. Position (1500-800) <1000 (code limit), A-POD do nothing. | KhoaCDN | QuanNM | Passed |
| UC65 | APOD\_Mandible\_ Nip() | Reset all mandible servos (S3,S19) to Init Position (1500) | 1. Add APOD\_Mandible\_Nip (100)  2. Add APOD\_Mandible\_Nip (300) | 1. S3 = P1600, S19 = P1400. APOD mandible nip.  2. S3 = 1800, S19 = 1200 < 1300( code limit), APOD do nothing | KhoaCDN | QuanNM | Passed |
| UC66 | APOD\_Mandible\_ Release() | Reset all mandible servos (S3,S19) to Init Position (1500) | 1. Add APOD\_Mandible\_Nip (200)  2. Add APOD\_Mandible\_Nip (400) | 1. S3 = P1300, S19 = P1700. APOD mandible release.  2. S3 = 1100, S19 = 1900 > 1800( code limit), APOD do nothing | KhoaCDN | QuanNM | Passed |
| ? | ? |  |  |  |  |  |  |

### Test : Servo Control

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| No. | Function / Feature | Pre-condition | Step of testing | Expected Result | Created by | Executed by | Result |
| UC67 | PW\_Start | 1. Start Program  2. HC\_05 connected.  3. STM32F4, SSC32, Servos all connected & power up.  4. Servos in Idle stage | Send command  “S1” to start | All Servos set Pulse with Pulse is 1500 | KhoaCDN | CuongPAD | Passed |
| UC68 | PW\_Stop | 1. Start Program  2. HC\_05 connected.  3. STM32F4, SSC32, Servos all connected & power up.  4. Servos in Working stage | Send command  “S0” to stop | All Servos release to Idle Stage | KhoaCDN | CuongPAD | Passed |
| UC69 | PW\_Set | 1. Start Program  2. HC\_05 connected.  3. STM32F4, SSC32, Servos all connected & power up.  4. Servos in Working stage | Send command  “0x23 0x01 0x04  0xB0 0x54 0x02” | Set servo 01 with Pulse is 1200 | KhoaCDN | CuongPAD | Passed |
| UC70 | PW\_Set | 1. Start Program  2. HC\_05 connected.  3. STM32F4, SSC32, Servos all connected & power up.  4. Servos in Working stage | Send command  “0x23 0x01 0x04  0x07 0x08 0x02” | Set servo 01 with Pulse is 1800 | KhoaCDN | CuongPAD | Passed |

### Test : System Ingration

#### System Test : Ingration all system part : PC device, HC\_05, STM32F4, SSC32, Servos

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| No. | Function / Feature | | Pre-condition | Step of testing | Expected Result | Created by | Executed by | Result |
| UC71 | Connect Device | | 1. Start Program  2. HC\_05 active | 1. Click “Find” device, select HC\_05, click OK | 1. Connected successfully with HC\_05 | KhoaCDN | CuongPAD | Passed |
| UC72 | Direct Control mode | Start() | 1. Start Program  2. HC\_05 connected.  3. STM32F4, SSC32, Servos all connected & power up. | 1. Switch to Live Controller tab  2. Click “Start” button | 1. APOD run. All legs move to Init Position (1500) | KhoaCDN | CuongPAD | Passed |
| UC73 | Forward() | 1. Repeat UC68  2. APOD is running. | 1. Switch to Live Controller tab  2. Click “↑” button | 1. APOD move forward correctly | KhoaCDN | CuongPAD | Passed |
| UC74 |  | Backward() | 1. Repeat UC68  2. APOD is running. | 1. Switch to Live Controller tab  2. Click “↓” button | 1. APOD move backward correctly | KhoaCDN | CuongPAD | Passed |

# SOFTWARE USER’S MANUAL (SUM)

# APPENDIX

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