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

|  |
| --- |
| **FPT UNIVERSITY** |
| Capstone Project Document |
| Designing and making  A Lynxmotion APOD robot controller |
|  |
| |  |  | | --- | --- | | **Hexapod Team** | | | **Group Members** | * Phan Anh Dũng Cường * Nguyễn Minh Quân | | **Supervisor** | Mr. Trần Khánh Ninh | | **Ext Supervisor** |  | | **Capstone Project code** | APOD | |
|  |

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

This page intentionally left blank.

# Acknowledgements

We would like to express our appreciation to Mr. Trần Khánh Ninh - our research supervisor – for his support and guideline throughout this project, from the beginning. His experience and advices has been a great resource for us.

We also wish to thanks various people including our family, our friend for their contribution and support for this project.

# Table of Contents

[Acknowledgements 3](#_Toc373746732)

[Table of Contents 4](#_Toc373746733)

[A. INTRODUCTION 11](#_Toc373746734)

[I. PROJECT INFORMATION 11](#_Toc373746735)

[II. TEAM MEMBER 11](#_Toc373746736)

[III. APOD INTRODUCTION 11](#_Toc373746737)

[1. HEXAPOD 11](#_Toc373746738)

[2. LYNXMOTION APOD 13](#_Toc373746739)

[IV. EXISTING SOLUTIONS 14](#_Toc373746740)

[V. PROJECT INTRODUCTION 15](#_Toc373746741)

[VI. APPROACHES 16](#_Toc373746742)

[VII. OTHER 16](#_Toc373746743)

[1. WHY WE NEED APOD ROBOT? 16](#_Toc373746744)

[2. PRODUCT 16](#_Toc373746745)

[B. SOFTWARE PROJECT MANAGEMENT PLAN (SPMP) 17](#_Toc373746746)

[I. PROBLEM DEFINITION 17](#_Toc373746747)

[1. NAME OF THIS CAPSTONE PROJECT 17](#_Toc373746748)

[2. PROBLEM ABSTRACT 17](#_Toc373746749)

[3. PROBLEM OVERVIEW 17](#_Toc373746750)

[3.1. The Current System 17](#_Toc373746751)

[3.2. The Propose System 17](#_Toc373746752)

[3.3. Boundaries of The System 18](#_Toc373746753)

[3.4. Developing Environment 18](#_Toc373746754)

[3.4.1 Hardware environment: 18](#_Toc373746755)

[3.4.2 Software environment: 18](#_Toc373746756)

[II. PROJECT ORGANIZATION 19](#_Toc373746757)

[1. SOFTWARE PROCESS MODEL 19](#_Toc373746758)

[2. ROLES AND RESPONSIBILITIES 20](#_Toc373746759)

[3. TOOL AND TECHNIQUES 20](#_Toc373746760)

[3.1. For Development 20](#_Toc373746761)

[3.1.1 Hardware environment: 20](#_Toc373746762)

[3.1.2 Software environment: 20](#_Toc373746763)

[3.2. For Management 20](#_Toc373746764)

[3.3. For Communication 20](#_Toc373746765)

[III. PROJECT MANAGEMENT PLAN 21](#_Toc373746766)

[1. Phases 21](#_Toc373746767)

[1.1. Initiating and Planning: 21](#_Toc373746768)

[1.2. Software Requirement Analysis 21](#_Toc373746769)

[1.3. Creating Software Design Description 21](#_Toc373746770)

[1.4. Coding 22](#_Toc373746771)

[1.5. Testing 22](#_Toc373746772)

[1.6. Deployment 22](#_Toc373746773)

[2. Task sheet 23](#_Toc373746774)

[IV. CODING CONVENTION 25](#_Toc373746775)

[1. Coding style 25](#_Toc373746776)

[2. Source File Organization 25](#_Toc373746777)

[2.1. C/C++ 25](#_Toc373746778)

[2.1.1 [File name].h: 25](#_Toc373746779)

[2.1.2 [File name].c: 25](#_Toc373746780)

[2.2. C# 25](#_Toc373746781)

[C. SOFTWARE REQUIREMENTS SPECIFICATION (SRS) 25](#_Toc373746782)

[I. USER REQUIREMENT SPECIFICATIONS 25](#_Toc373746783)

[II. SYSTEM REQUIREMENT SPECIFICATIONS 26](#_Toc373746784)

[1. Hardware requirement 26](#_Toc373746785)

[2. Software interface 26](#_Toc373746786)

[3. External interface 26](#_Toc373746787)

[4. Functional requirement 26](#_Toc373746788)

[5. Non-functional requirement 26](#_Toc373746789)

[III. USE-CASE DIAGRAM AND USE-CASE SPECIFICATIONS 27](#_Toc373746790)

[1. Use-case Diagram 27](#_Toc373746791)

[2. Use-case Specifications 27](#_Toc373746792)

[2.1. Sequences Control 27](#_Toc373746793)

[2.1.1 Add State 27](#_Toc373746794)

[2.1.2 Remove states 29](#_Toc373746795)

[2.1.3 Import States 30](#_Toc373746796)

[2.1.4 Export States 31](#_Toc373746797)

[2.2. Direct Control 33](#_Toc373746798)

[2.2.1 Mode Selection 33](#_Toc373746799)

[a) Normal Input 33](#_Toc373746800)

[b) Gamepad Input 34](#_Toc373746801)

[2.2.2 Movement Control 36](#_Toc373746802)

[a) Move Forward 36](#_Toc373746803)

[b) Move Backward 37](#_Toc373746804)

[c) Turn Left 39](#_Toc373746805)

[d) Turn Right 40](#_Toc373746806)

[e) Body Lift 42](#_Toc373746807)

[f) Body Drop 43](#_Toc373746808)

[g) Squeeze Left 44](#_Toc373746809)

[h) Squeeze Right 46](#_Toc373746810)

[i) Manual Grip/Release 47](#_Toc373746811)

[2.3. Object tracking 49](#_Toc373746812)

[2.3.1 Auto Grip 49](#_Toc373746813)

[2.3.2 Choose Target 50](#_Toc373746814)

[2.3.3 Change Target 52](#_Toc373746815)

[2.3.4 Lock Target 53](#_Toc373746816)

[IV. SOFTWARE SYSTEM ATTRIBUTES 54](#_Toc373746817)

[1. Reliability 54](#_Toc373746818)

[2. Availability 55](#_Toc373746819)

[3. Security 55](#_Toc373746820)

[4. Maintainability 55](#_Toc373746821)

[5. Portability 55](#_Toc373746822)

[6. Performance 55](#_Toc373746823)

[D. SOFTWARE DESIGN DESCRIPTION (SDD) 55](#_Toc373746824)

[I. DESIGN OVERVIEW 55](#_Toc373746825)

[1. Introduction 55](#_Toc373746826)

[2. Architecture 56](#_Toc373746827)

[3. User interface 57](#_Toc373746828)

[II. COMPONENTS 58](#_Toc373746829)

[1. Central processing 58](#_Toc373746830)

[1.1. Bluetooth module (HC 05) 58](#_Toc373746831)

[1.2. STM32F4 Discovery 58](#_Toc373746832)

[2. Servos control & Sensor reader 59](#_Toc373746833)

[2.1. HG-645MG Servo 59](#_Toc373746834)

[2.2. SSC32 with ATMEGA328P 60](#_Toc373746835)

[2.3. Ultrasonic ranging module (HC SR04) 60](#_Toc373746836)

[III. IMPLEMENTATION APPROACHES 61](#_Toc373746837)

[1. Algorithms and Mechanism 61](#_Toc373746838)

[1.1. RC Servo Control 61](#_Toc373746839)

[1.1.1 HG-645MG Specification 61](#_Toc373746840)

[1.1.2 How does it work? 62](#_Toc373746841)

[1.1.3 Servo’s Speed Control 63](#_Toc373746842)

[1.2. Sensor Reader (HC-SR04) 64](#_Toc373746843)

[1.2.1 HC-SR04 Specifications 64](#_Toc373746844)

[1.2.2 How to read distance value? 65](#_Toc373746845)

[1.3. Multi-legs Control 66](#_Toc373746846)

[1.3.1 Tripod Gait 66](#_Toc373746847)

[1.3.2 Wave Gait 66](#_Toc373746848)

[1.3.3 Ripple Gait 67](#_Toc373746849)

[1.3.4 Implementation Approaches 68](#_Toc373746850)

[a) Walking In Straight Line 68](#_Toc373746851)

[b) Turning Left – Right 73](#_Toc373746852)

[2. SSC32 Board 75](#_Toc373746853)

[2.1. Schematic 75](#_Toc373746854)

[2.2. Technical Point 77](#_Toc373746855)

[2.2.1 Atmega328P Microcontroller (MCU) 77](#_Toc373746856)

[a) Overview 77](#_Toc373746857)

[b) Pin Configuration 78](#_Toc373746858)

[c) Block Diagram 79](#_Toc373746859)

[2.2.2 General Purposes Input / Output (GPIO) 79](#_Toc373746860)

[a) Configurations 80](#_Toc373746861)

[b) Register Description 80](#_Toc373746862)

[2.2.3 Interrupts 82](#_Toc373746863)

[2.2.4 Timer 82](#_Toc373746864)

[a) Usage and Purposes 82](#_Toc373746865)

[b) Register Description 83](#_Toc373746866)

[2.2.5 Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART) 85](#_Toc373746867)

[a) Usage and Purposes 85](#_Toc373746868)

[b) Baud Rate 85](#_Toc373746869)

[c) Register Description 86](#_Toc373746870)

[2.2.6 Serial Peripheral Interface (SPI) 89](#_Toc373746871)

[a) Usage and Purposes 89](#_Toc373746872)

[b) Register Description 89](#_Toc373746873)

[2.2.7 Shift Register (74HC595) 90](#_Toc373746874)

[a) Overview 90](#_Toc373746875)

[b) How does it work? 90](#_Toc373746876)

[2.2.8 SPI - Shift Register interface 91](#_Toc373746877)

[a) Overview 91](#_Toc373746878)

[b) How does it work? 91](#_Toc373746879)

[3. STM32F4 Discovery Board 92](#_Toc373746880)

[3.1. STM32F407VGT6 Microcontroller (MCU) 92](#_Toc373746881)

[3.1.1 Overview 92](#_Toc373746882)

[3.1.2 Configuration 93](#_Toc373746883)

[3.1.3 Block Diagram 94](#_Toc373746884)

[3.2. General Purposes Input / Output (GPIO) 95](#_Toc373746885)

[3.2.1 Features 95](#_Toc373746886)

[3.2.2 Configuration 95](#_Toc373746887)

[3.2.3 Register Description 96](#_Toc373746888)

[3.3. Interrupts 99](#_Toc373746889)

[Features 99](#_Toc373746890)

[3.4. Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART) 99](#_Toc373746891)

[3.4.1 Usage and Purpose 99](#_Toc373746892)

[3.4.2 Baud Rate 100](#_Toc373746893)

[3.4.3 Register Description 100](#_Toc373746894)

[IV. ACTIVITY AND CLASS STRUCTURE 103](#_Toc373746895)

[1. APOD Controller (HMI Application) 103](#_Toc373746896)

[1.1. Activity 103](#_Toc373746897)

[1.2. Class Diagram 104](#_Toc373746898)

[1.3. External Library 104](#_Toc373746899)

[2. STM32F4 Discovery (Central processing board) 105](#_Toc373746900)

[2.1. Activity 105](#_Toc373746901)

[2.2. Command Sheet 106](#_Toc373746902)

[3. SSC32 Board (Servos control and Sensor reader) 107](#_Toc373746903)

[3.1. Activity 107](#_Toc373746904)

[3.2. Command Sheet 108](#_Toc373746905)

[V. SEQUENCE DIAGRAM 109](#_Toc373746906)

[1. Embedded Controller 109](#_Toc373746907)

[1.1. Forward 109](#_Toc373746908)

[1.2. Backward 110](#_Toc373746909)

[1.3. Turn Left 111](#_Toc373746910)

[1.4. Turn Right 112](#_Toc373746911)

[2. Use-cases Sequences 113](#_Toc373746912)

[2.1. Sequences Control 113](#_Toc373746913)

[2.1.1 Add State 113](#_Toc373746914)

[2.1.2 Remove States 114](#_Toc373746915)

[2.1.3 Import States 114](#_Toc373746916)

[2.1.4 Export States 115](#_Toc373746917)

[2.2. Direct Control 116](#_Toc373746918)

[2.2.1 Mode Selection 116](#_Toc373746919)

[a) Normal Input 116](#_Toc373746920)

[b) Gamepad Input 117](#_Toc373746921)

[2.2.2 Movement Control 118](#_Toc373746922)

[a) Forward 118](#_Toc373746923)

[b) Backward 118](#_Toc373746924)

[c) Turn Left 119](#_Toc373746925)

[d) Turn Right 119](#_Toc373746926)

[e) Body Lift 120](#_Toc373746927)

[f) Body Drop 120](#_Toc373746928)

[g) Squeeze Left 121](#_Toc373746929)

[h) Squeeze Right 121](#_Toc373746930)

[i) Manual Grip/Release 122](#_Toc373746931)

[2.3. Object Tracking 123](#_Toc373746932)

[2.3.1 Auto Grip 123](#_Toc373746933)

[2.3.2 Choose target 124](#_Toc373746934)

[2.3.3 Change Target 125](#_Toc373746935)

[2.3.4 Lock Target 126](#_Toc373746936)

[VI. USER INTERFACE DESIGN 127](#_Toc373746937)

[1. Live Control 127](#_Toc373746938)

[2. Sequence Player 128](#_Toc373746939)

[3. Configuration 129](#_Toc373746940)

[4. Object Extractor 130](#_Toc373746941)

[E. SOFTWARE TEST DOCUMENTATION (STD) 131](#_Toc373746942)

[I. Introduction 131](#_Toc373746943)

[1. Overview 131](#_Toc373746944)

[2. Test Approach 131](#_Toc373746945)

[II. Test Plan 131](#_Toc373746946)

[1. Features to be tested 131](#_Toc373746947)

[2. Features not to be tested 131](#_Toc373746948)

[3. Test Environment 132](#_Toc373746949)

[III. Test Cases 132](#_Toc373746950)

[1. Functional Test 132](#_Toc373746951)

[2. UI Test 147](#_Toc373746952)

[F. SOFTWARE USER MANUAL (SUM) 156](#_Toc373746953)

[I. Installation 156](#_Toc373746954)

[II. Usage 158](#_Toc373746955)

[G. APPENDIX 158](#_Toc373746956)

[I. Glossary 158](#_Toc373746957)

[II. List of Figures 159](#_Toc373746958)

[III. List of Tables 162](#_Toc373746959)

[IV. References 163](#_Toc373746960)

# INTRODUCTION

## PROJECT INFORMATION

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

## TEAM MEMBER

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Name** | **Role** | **Contact** |
| **1** | Phan Anh Dũng Cường | Leader | * Phone: 0972708318 * Email: cuongpadse60774@fpt.edu.vn |
| **2** | Nguyễn Minh Quân | Member | * Phone: 0915699635 * Email: quannm60344@fpt.edu.vn |
|  |  |  |  |
|  |  |  |  |

Table A‑1 Team Members

## 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 CH3-R Walking Robot



Figure A‑2 AH2 Walking Robot

### LYNXMOTION APOD



Figure A‑3 Lynxmotion APOD

* 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

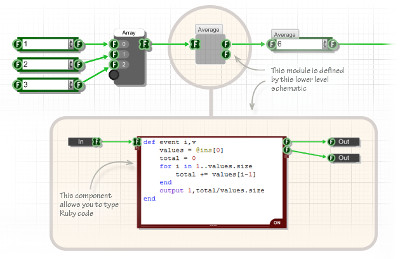


Figure A‑4 Flowbotics Studio example 1

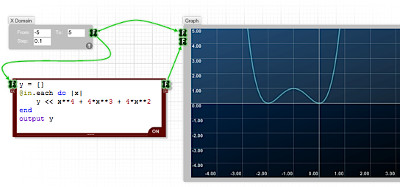


Figure A‑5 Flowbotics Studio example 2

* Lynxmotion Visual Sequences: $39.94

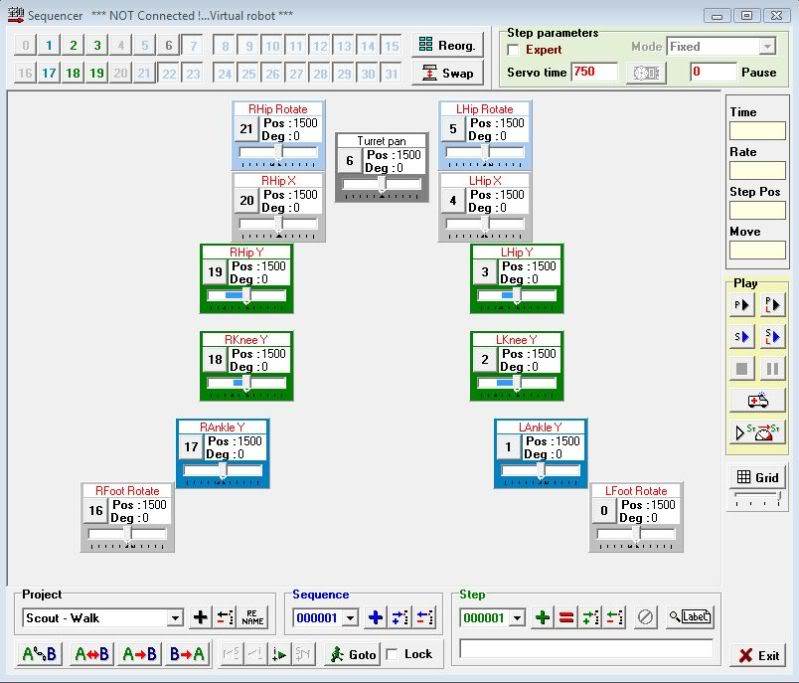


Figure A‑6 Lynxmotion Visual Sequences

## PROJECT INTRODUCTION

This project focus on programming on microprocessor unit (MCU) to developing, control the APOD 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 APOD  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 (Human Machine Interface) program will be provided for user to control the APOD with ease.

## OTHER

### WHY WE NEED APOD ROBOT?

The main purpose of APOD Robot is to be used at home with variable simple tasks. The Project ‘s scope is to control APOD from a distance via a HCI program on PC (or mobile devices). APOD 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 APOD  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 System Boundary Overview

* + 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 AVR 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.

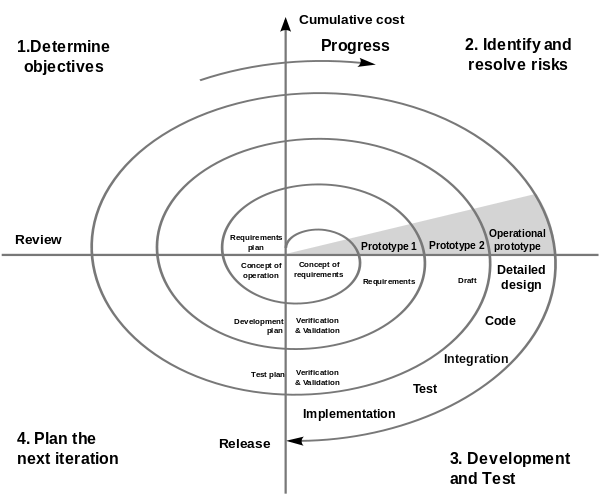


Figure B‑2 Spriral model

### ROLES AND RESPONSIBILITIES

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Full name | Team Role | Responsibilities |
| 1 | Trần Khánh Ninh | Supervisor | Define business  Support in technical issues |
| 1 | Phan Anh Dũng Cường | 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 |
|  |  |  |  |

Table B‑1 Roles and Responsibilities

### 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 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 report
* Skype chat system
* CMS system

## PROJECT MANAGEMENT PLAN

### Phases

#### 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 Initiating and Planning

#### Software Requirement Analysis

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

Table B‑3 Software Requirement Analysis

#### Creating Software Design Description

|  |  |
| --- | --- |
| Description | Designing the controller for APOD robot based on actual requirements |
| Output | Architecture design, circuits diagram, board diagram, algorithms and design specification |
| Deliverables | Software Design Description(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 Creating Software Design Description

#### Coding

|  |  |
| --- | --- |
| Description | Designing the controller for APOD 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 Coding

#### 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 Testing

#### Deployment

|  |  |
| --- | --- |
| Description | Deploying APOD Controller system |
| 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 Deployment

### Task sheet

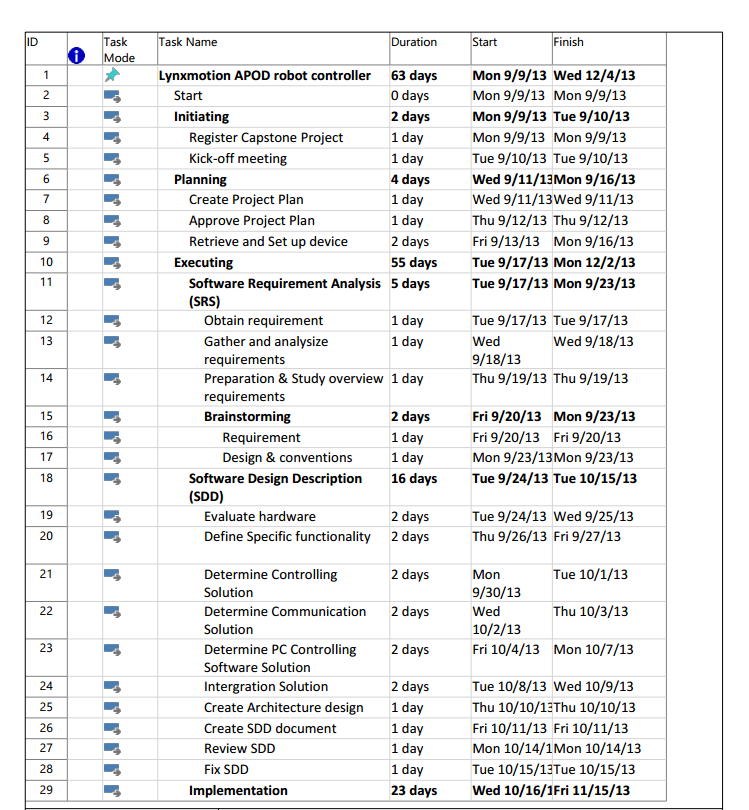


Figure B‑3 Project plan Page 1

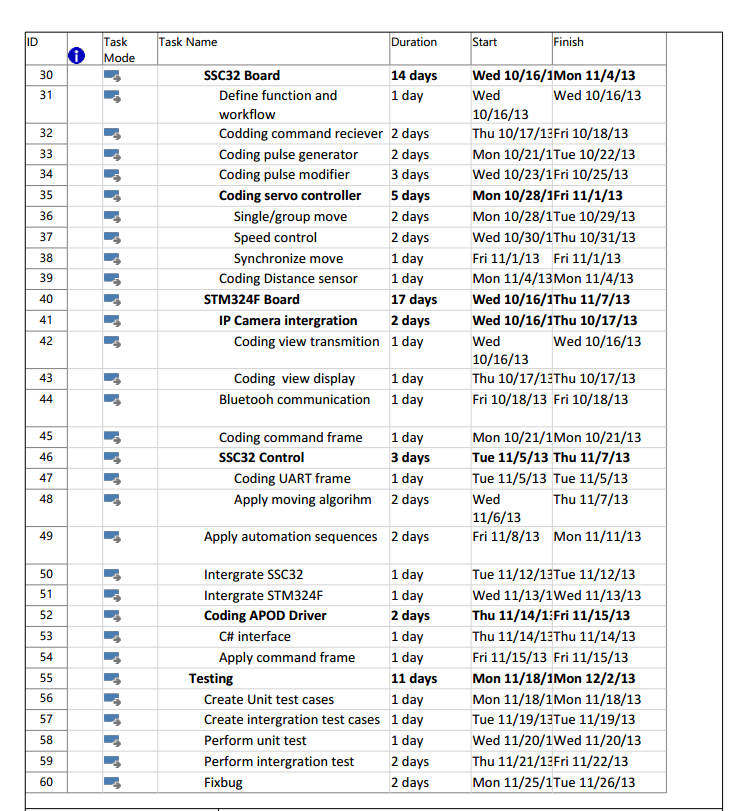


Figure B‑4 Project plan Page 2

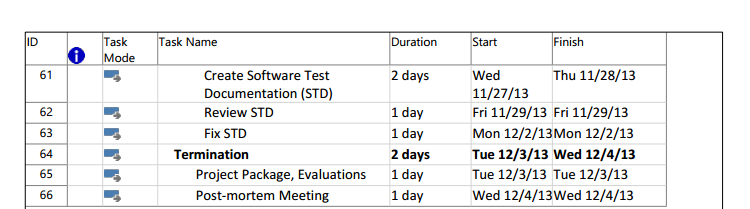


Figure B‑5 Project plan Page 3

## CODING CONVENTION

### Coding style

The following rules follow:

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

### Source File Organization

#### C/C++

Each module must have 2 file [File name].h and [File name].c following below description:

##### [File name].h:

* Macro declaration.
* External declaration of global variable for usage from different module
* Prototype of function/subroutine used by referenced module

##### [File name].c:

* Including [File name].h
* Declaration of external Module/Library supporting the implementation
* Declaration of external variable in [File name].h
* Declaration of variable used only within the scope of this module
* Implementation of all prototype declared in [File name].h

#### C#

* Each module support for 1 function will be places in 1 folder (namespace)
* External Dynamic-link library (DLL) will be place in 1 different folder from the solution with a text file listed all of the DLL needed.

# SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

## USER REQUIREMENT SPECIFICATIONS

* APOD should be able to walk freely in any direction.
* Controlling via a Serial connection like Bluetooth, or directly through a PS2 controller.
* APOD should be able to grab small things like a Coke.
* APOD should be able to recording video.
* APOD 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, APOD should move within 2 seconds

## SYSTEM REQUIREMENT SPECIFICATIONS

### Hardware requirement

* 25 HS-645MG servo
* 1 SSC-32 servo controller
* 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 APOD ‘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 APOD to turn left, right, go forward or backward.
* Camera view control : user can see the view sending by the APOD camera.
* Grab control : user can direct the APOD to grab things.
* Remote control : enable remote controlling, help the user control the APOD through a programmed Bluetooth device.

### Non-functional requirement

* The APOD 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.
* View from camera should be refresh every 0.5 second.
* The APOD 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

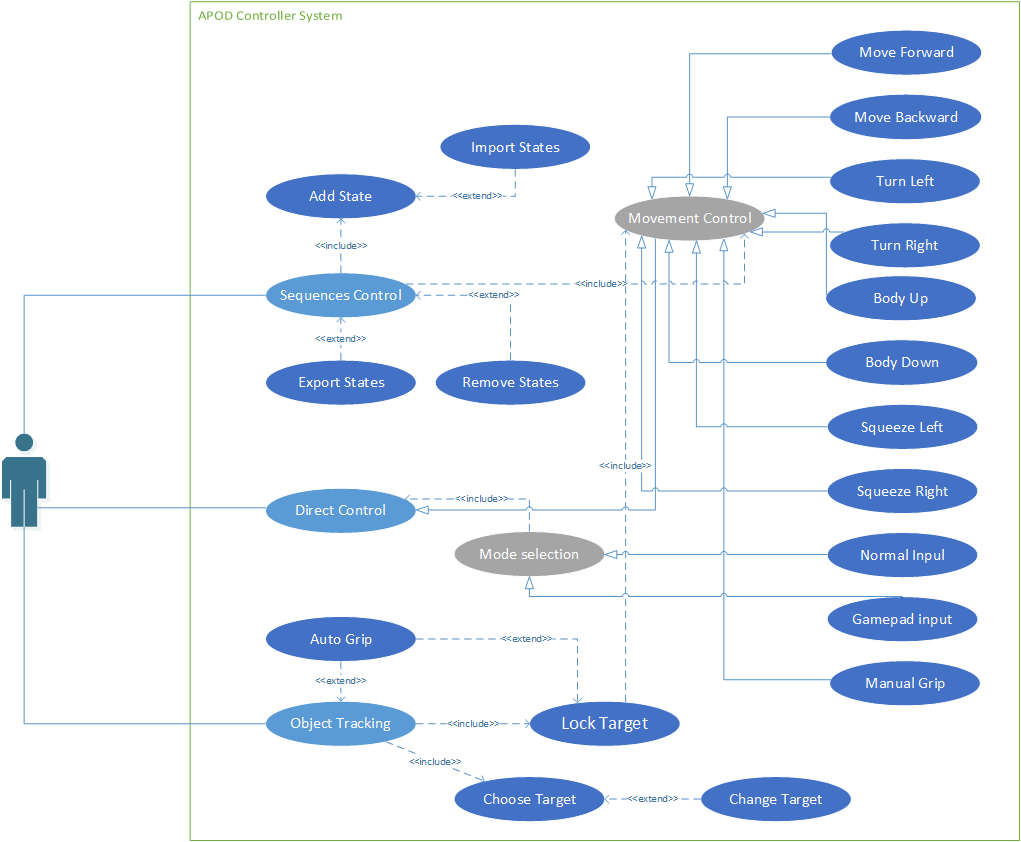
****

Figure C‑1: Use cases diagram

### Use-case Specifications

#### Sequences Control

##### Add State

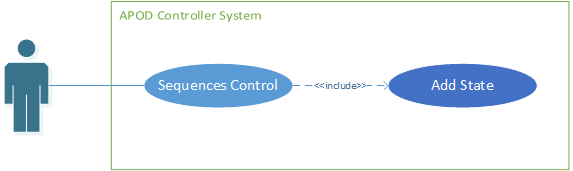


Figure C‑2 Add State Use-case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ADD STATE USE CASE SPECIFICATION** | | | | |
| **Use-case No.** | UC001 | **Use-case Version** | | 1.0 |
| **Use-case Name** | Add State | | | |
| **Author** | CuongPAD | | | |
| **Date** | 05/10/2013 | **Priority** | Normal | |
| **Actor:**  User  **Summary:**  Add a state (move) to sequence  **Goal:**  New state is added to sequence  **Triggers**  User click [Add] button  **Preconditions:**   * Sequence Player feature is selected * Interval (if required) stay within valid range   **Post Conditions:**  NONE  **Main Success Scenario:**   * User select move to add to sequence * Enter interval value (if available) * Click [Add]   **Alternative Scenario:**  NONE  **Exceptions:**  NONE  **Relationships:**  NONE | | | | |

Table C‑1 Add state use-case specification

##### Remove states

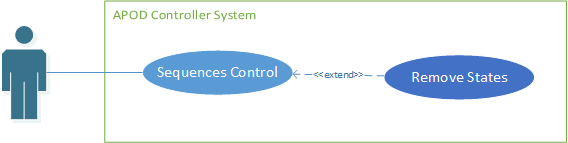


Figure C‑3 Remove States Use-case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **REMOVE STATES USE CASE SPECIFICATION** | | | | |
| **Use-case No.** | UC002 | **Use-case Version** | | 1.0 |
| **Use-case Name** | Remove States | | | |
| **Author** | CuongPAD | | | |
| **Date** | 05/10/2013 | **Priority** | Normal | |
| **Actor:**  User  **Summary:**  Remove selected states from current sequence  **Goal:**  All selected state to be removed  **Triggers**  User click [Remove] Button  **Preconditions:**   * Sequence Player feature is selected   **Post Conditions:**  NONE  **Main Success Scenario:**   * User select all move to be removed from sequence * Click [Remove]   **Alternative Scenario:**  NONE  **Exceptions:**  NONE  **Relationships:**  NONE | | | | |

Table C‑2 Remove states use-case specification

##### Import States



Figure C‑4 Import States Use-case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **IMPORT STATES USE CASE SPECIFICATION** | | | | |
| **Use-case No.** | UC003 | **Use-case Version** | | 1.0 |
| **Use-case Name** | Import States | | | |
| **Author** | CuongPAD | | | |
| **Date** | 05/10/2013 | **Priority** | Normal | |
| **Actor:**  User  **Summary:**  Import existing sequence from file  **Goal:**  Current sequence is replace by new one from file  **Triggers**  User click [Import] Button  **Preconditions:**   * Sequence Player feature is selected   **Post Conditions:**  NONE  **Main Success Scenario:**   * Click [Import] button * Browse to location contain sequence file (\*.xml) * Select file & click [Ok] * Validate new sequence content * Replace current sequence if valid   **Alternative Scenario:**  NONE  **Exceptions:**   * IO Exception   **Relationships:**  NONE | | | | |

Table C‑3 Import States use-case specification

##### Export States

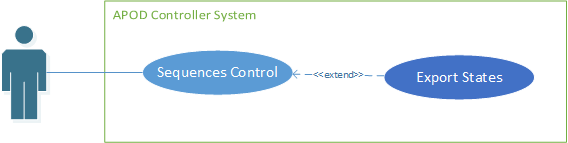


Figure C‑5 Export States Use-case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EXPORT STATES USE CASE SPECIFICATION** | | | | |
| **Use-case No.** | UC004 | **Use-case Version** | | 1.0 |
| **Use-case Name** | Export States | | | |
| **Author** | CuongPAD | | | |
| **Date** | 05/10/2013 | **Priority** | Normal | |
| **Actor:**  User  **Summary:**  Export current sequence to file  **Goal:**  New Sequence file  **Triggers**  User click [Export] button  **Preconditions:**   * Sequence Player feature is selected   **Post Conditions:**  NONE  **Main Success Scenario:**   * Click [Export] button * Browse to location contain sequence file (\*.xml) * Click [Ok]   **Alternative Scenario:**  NONE  **Exceptions:**   * IO Exception   **Relationships:**  NONE | | | | |

Table C‑4 Export states use-case specification

#### Direct Control

##### Mode Selection

###### Normal Input

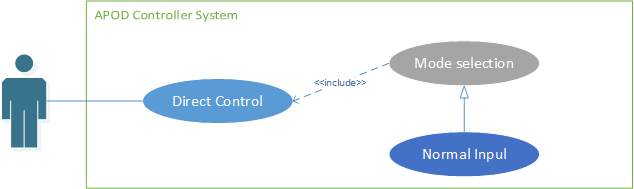


Figure C‑6 Normal input Use-case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **NORMAL INPUT USE CASE SPECIFICATION** | | | | |
| **Use-case No.** | UC005 | **Use-case Version** | | 1.0 |
| **Use-case Name** | Normal Input Selection | | | |
| **Author** | CuongPAD | | | |
| **Date** | 05/10/2013 | **Priority** | Normal | |
| **Actor:**  User  **Summary:**  Using normal input method (keyboard or Mouse)  **Goal:**  Normal input is selected  **Triggers**  User click [Normal Mode] button  **Preconditions:**   * Live control feature is selected   **Post Conditions:**  NONE  **Main Success Scenario:**   * User click [Normal Mode] button * Configuration is checked * Establish Bluetooth connection   **Alternative Scenario:**  NONE  **Exceptions:**   * IO Exception   **Relationships:**  NONE | | | | |

Table C‑5 Normal input use-case specification

###### Gamepad Input

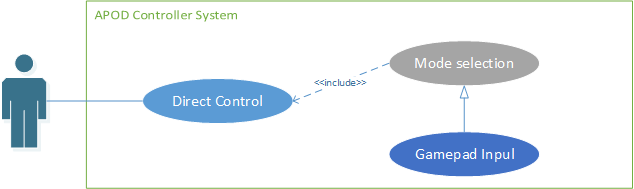


Figure C‑7 Gamepad Input Use-case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **GAMEPAD INPUT** **USE CASE SPECIFICATION** | | | | |
| **Use-case No.** | UC006 | **Use-case Version** | | 1.0 |
| **Use-case Name** | Gamepad Input Selection | | | |
| **Author** | CuongPAD | | | |
| **Date** | 05/10/2013 | **Priority** | Normal | |
| **Actor:**  User  **Summary:**  Using Gamepad input method (Gamepad)  **Goal:**  Gamepad input is selected  **Triggers**  User click [Gamepad Mode] button  **Preconditions:**   * Live control feature is selected * Gamepad is connected   **Post Conditions:**  NONE  **Main Success Scenario:**   * User click [Gamepad Mode] button * Configuration is checked * Gamepad device connection is checked * Establish Bluetooth connection   **Alternative Scenario:**  NONE  **Exceptions:**   * IO Exception   **Relationships:**  NONE | | | | |

Table C‑6 Gamepad input use-case specification

##### Movement Control

###### Move Forward

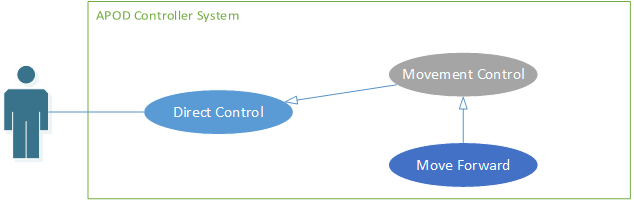


Figure C‑8 Move Forward Use-case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **MOVE FORWARD USE CASE SPECIFICATION** | | | | |
| **Use-case No.** | UC007 | **Use-case Version** | | 1.0 |
| **Use-case Name** | Move Forward | | | |
| **Author** | CuongPAD | | | |
| **Date** | 05/10/2013 | **Priority** | Normal | |
| **Actor:**  User  **Summary:**  Command APOD to move forward  **Goal:**  APOD walking in forward direction  **Triggers**  User click [Forward] button  **Preconditions:**   * Live control feature is selected * Bluetooth connection is established * Normal input mode or Gamepad input mode is selected * [L1] and [L2] is not pressed.   **Post Conditions:**   * APOD stop when button is released   **Main Success Scenario:**   * User click on [Forward] button ()   **Alternative Scenario:**   * Alt 1: press “W” key on keyboard (if Normal input is selected) * Alt 2: press () key on gamepad (if Gamepad input is selected)   **Exceptions:**   * IO Exception   **Relationships:**  NONE | | | | |

Table C‑7 Move Forward use-case specification

###### Move Backward

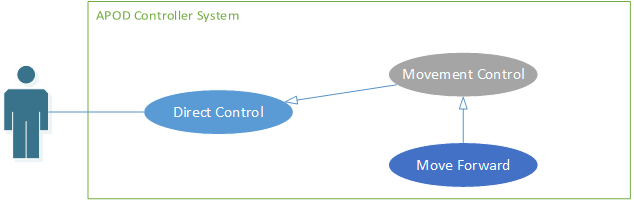


Figure C‑9 Move Backward Use-case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **MOVE BACKWARD USE CASE SPECIFICATION** | | | | |
| **Use-case No.** | UC008 | **Use-case Version** | | 1.0 |
| **Use-case Name** | Move Backward | | | |
| **Author** | CuongPAD | | | |
| **Date** | 05/10/2013 | **Priority** | Normal | |
| **Actor:**  User  **Summary:**  Command APOD to move Backward  **Goal:**  APOD walking in Backward direction  **Triggers**  User click [Backward] button  **Preconditions:**   * Live control feature is selected * Bluetooth connection is established * Normal input mode or Gamepad input mode is selected * [L1] and [L2] is not pressed.   **Post Conditions:**   * APOD stop when button is released   **Main Success Scenario:**   * User click on [Backward] button ()   **Alternative Scenario:**   * Alt 1: press “S” key on keyboard (if Normal input is selected) * Alt 2: press () key on gamepad (if Gamepad input is selected)   **Exceptions:**   * IO Exception   **Relationships:**  NONE | | | | |

Table C‑8 Move Backward use-case specification

###### Turn Left

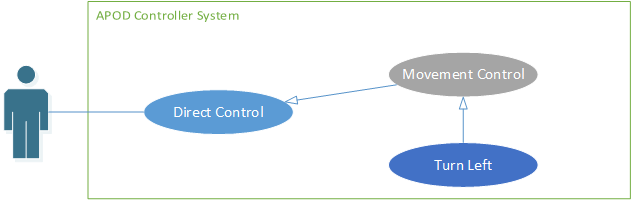


Figure C‑10 Turn Left Use-case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TURN LEFT USE CASE SPECIFICATION** | | | | |
| **Use-case No.** | UC009 | **Use-case Version** | | 1.0 |
| **Use-case Name** | Turn Left | | | |
| **Author** | CuongPAD | | | |
| **Date** | 05/10/2013 | **Priority** | Normal | |
| **Actor:**  User  **Summary:**  Command APOD to turn to the left  **Goal:**  APOD turn in Left direction  **Triggers**  User click [Turn Left] button  **Preconditions:**   * Live control feature is selected * Bluetooth connection is established * Normal input mode or Gamepad input mode is selected * [L1] and [L2] is not pressed.   **Post Conditions:**   * APOD stop when button is released   **Main Success Scenario:**   * User click on [Turn Left] button ()   **Alternative Scenario:**   * Alt 1: press “A” key on keyboard (if Normal input is selected) * Alt 2: press () key on gamepad (if Gamepad input is selected)   **Exceptions:**   * IO Exception   **Relationships:**  NONE | | | | |

Table C‑9 Turn left use-case specification

###### Turn Right

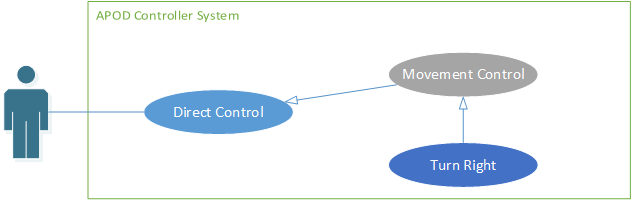


Figure C‑11 Turn Right Use-case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TURN RIGHT USE CASE SPECIFICATION** | | | | |
| **Use-case No.** | UC010 | **Use-case Version** | | 1.0 |
| **Use-case Name** | Turn Right | | | |
| **Author** | CuongPAD | | | |
| **Date** | 05/10/2013 | **Priority** | Normal | |
| **Actor:**  User  **Summary:**  Command APOD to turn to the Right  **Goal:**  APOD turn in Right direction  **Triggers**  User click [Turn Right] button  **Preconditions:**   * Live control feature is selected * Bluetooth connection is established * Normal input mode or Gamepad input mode is selected * [L1] and [L2] is not pressed.   **Post Conditions:**   * APOD stop when button is released   **Main Success Scenario:**   * User click on [Turn Right] button ()   **Alternative Scenario:**   * Alt 1: press “D” key on keyboard (if Normal input is selected) * Alt 2: press () key on gamepad (if Gamepad input is selected)   **Exceptions:**   * IO Exception   **Relationships:**  NONE | | | | |

Table C‑10 Turn right use-case specification

###### Body Lift

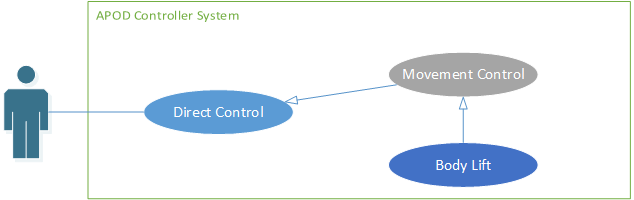


Figure C‑12 Body Lift Use-case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **BODY LIFT USE CASE SPECIFICATION** | | | | |
| **Use-case No.** | UC011 | **Use-case Version** | | 1.0 |
| **Use-case Name** | Body Lift | | | |
| **Author** | CuongPAD | | | |
| **Date** | 05/10/2013 | **Priority** | Normal | |
| **Actor:**  User  **Summary:**  Command APOD to Lift up  **Goal:**  APOD Lift body up  **Triggers**  User click [R1] button  **Preconditions:**   * Live control feature is selected * Bluetooth connection is established * Normal input mode or Gamepad input mode is selected   **Post Conditions:**  NONE  **Main Success Scenario:**   * User click on [R1] button ()   **Alternative Scenario:**   * Alt 1: press “U” key on keyboard (if Normal input is selected) * Alt 2: press () key on gamepad (if Gamepad input is selected)   **Exceptions:**   * IO Exception   **Relationships:**  NONE | | | | |

Table C‑11 Body Lift use-case specification

###### Body Drop

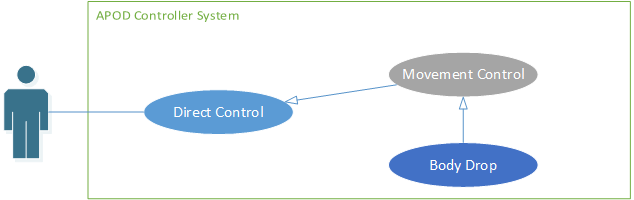


Figure C‑13 Body Drop Use-case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **BODY DROP USE CASE SPECIFICATION** | | | | |
| **Use-case No.** | UC012 | **Use-case Version** | | 1.0 |
| **Use-case Name** | Body Drop | | | |
| **Author** | CuongPAD | | | |
| **Date** | 05/10/2013 | **Priority** | Normal | |
| **Actor:**  User  **Summary:**  Command APOD to Drop down  **Goal:**  APOD Drop body Down  **Triggers**  User click [R2] button  **Preconditions:**   * Live control feature is selected * Bluetooth connection is established * Normal input mode or Gamepad input mode is selected   **Post Conditions:**  NONE  **Main Success Scenario:**   * User click on [R1] button ()   **Alternative Scenario:**   * Alt 1: press “O” key on keyboard (if Normal input is selected) * Alt 2: press () key on gamepad (if Gamepad input is selected)   **Exceptions:**   * IO Exception   **Relationships:**  NONE | | | | |

Table C‑12 Body Drop use-case specification

###### Squeeze Left

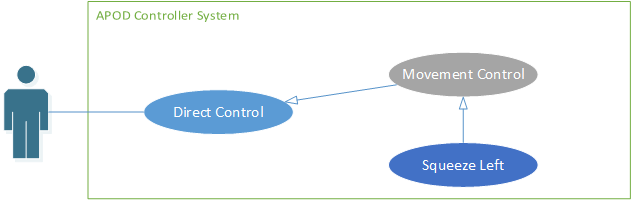


Figure C‑14 Squeeze Left Use-case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SQUEEZE LEFT USE CASE SPECIFICATION** | | | | |
| **Use-case No.** | UC013 | **Use-case Version** | | 1.0 |
| **Use-case Name** | Squeeze Left | | | |
| **Author** | CuongPAD | | | |
| **Date** | 05/10/2013 | **Priority** | Normal | |
| **Actor:**  User  **Summary:**  Command APOD to squeeze to the left  **Goal:**  APOD Drop Squeeze left  **Triggers**  User click [Turn Left] button  **Preconditions:**   * Live control feature is selected * Bluetooth connection is established * Normal input mode or Gamepad input mode is selected * [L2] button is pressed   **Post Conditions:**  NONE  **Main Success Scenario:**   * User press [L2] button () * User click on [Turn Left] button ()   **Alternative Scenario:**   * Alt 1: press “A” key on keyboard while pressing “E” key (if Normal input is selected) * Alt 2: press () key on gamepad while pressing  (if Gamepad input is selected)   **Exceptions:**   * IO Exception   **Relationships:**  NONE | | | | |

Table C‑13 Squeeze Left use-case specification

###### Squeeze Right

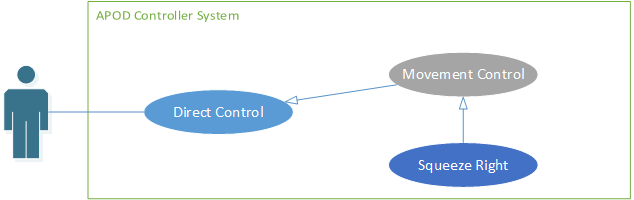


Figure C‑15 Squeeze Right Use-case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SQUEEZE RIGHT USE CASE SPECIFICATION** | | | | |
| **Use-case No.** | UC014 | **Use-case Version** | | 1.0 |
| **Use-case Name** | Squeeze Right | | | |
| **Author** | CuongPAD | | | |
| **Date** | 05/10/2013 | **Priority** | Normal | |
| **Actor:**  User  **Summary:**  Command APOD to squeeze to the right  **Goal:**  APOD Drop Squeeze right  **Triggers**  User click [Turn Right] button  **Preconditions:**   * Live control feature is selected * Bluetooth connection is established * Normal input mode or Gamepad input mode is selected * [L2] button is pressed   **Post Conditions:**  NONE  **Main Success Scenario:**   * User press [L2] button () * User click on [Turn Left] button ()   **Alternative Scenario:**   * Alt 1: press “D” key on keyboard while pressing “E” key (if Normal input is selected) * Alt 2: press () key on gamepad while pressing  (if Gamepad input is selected)   **Exceptions:**   * IO Exception   **Relationships:**  NONE | | | | |

Table C‑14 Squeeze Right use-case specification

###### Manual Grip/Release

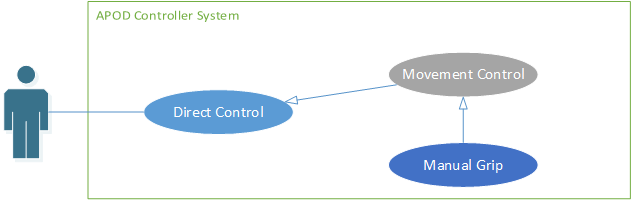


Figure C‑16 Manual Grip Use-case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **MANUAL GRIP/RELEASE USE CASE SPECIFICATION** | | | | |
| **Use-case No.** | UC015 | **Use-case Version** | | 1.0 |
| **Use-case Name** | Manual Grip | | | |
| **Author** | CuongPAD | | | |
| **Date** | 05/10/2013 | **Priority** | Normal | |
| **Actor:**  User  **Summary:**  Command APOD to Grip/Release Mandibles  **Goal:**  APOD Mandibles Grip/Release  **Triggers**  User click [Grip/Release] button  **Preconditions:**   * Live control feature is selected * Bluetooth connection is established * Normal input mode or Gamepad input mode is selected * [L1] and [L2] is not pressed.   **Post Conditions:**   * APOD stop when button is released   **Main Success Scenario:**   * User click on [Grip] button () or [Release] Button ()   **Alternative Scenario:**   * Alt 1: press “I” key on keyboard (if Normal input is selected) * Alt 2: press () or () key on gamepad (if Gamepad input is selected)   **Exceptions:**   * IO Exception   **Relationships:**  NONE | | | | |

Table C‑15 Manual Grip use-case specification

#### Object tracking

##### Auto Grip

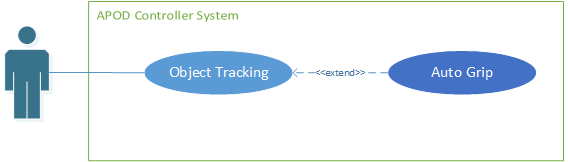


Figure C‑17 Auto Grip Use-case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **AUTO GRIP USE CASE SPECIFICATION** | | | | |
| **Use-case No.** | UC016 | **Use-case Version** | | 1.0 |
| **Use-case Name** | Auto Grip | | | |
| **Author** | CuongPAD | | | |
| **Date** | 05/10/2013 | **Priority** | Normal | |
| **Actor:**  User  **Summary:**  Auto detect object in range and move forward to grab it.  **Goal:**  Object is grabbed  **Triggers**  User click [AUTO GRIP] button  **Preconditions:**   * Live control feature is selected * Bluetooth connection is established   **Post Conditions:**  NONE  **Main Success Scenario:**   * User click [AUTO GRIP] button * Auto grip command is sent to APOD   **Alternative Scenario:**  NONE  **Exceptions:**   * IO Exception   **Relationships:**  NONE | | | | |

Table C‑16 Auto Grip use-case specification

##### Choose Target



Figure C‑18 Choose Target Use-case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CHOOSE TARGET USE CASE SPECIFICATION** | | | | |
| **Use-case No.** | UC017 | **Use-case Version** | | 1.0 |
| **Use-case Name** | Choose Target | | | |
| **Author** | CuongPAD | | | |
| **Date** | 05/10/2013 | **Priority** | Normal | |
| **Actor:**  User  **Summary:**  Enter tracking mode, choosing target to track  **Goal:**  New tracking template is selected  **Triggers**  User click [Object Tracking] button  **Preconditions:**   * Configuration is initiated * Bluetooth connection is established * IP camera stream is working   **Post Conditions:**  NONE  **Main Success Scenario:**   * [Object Tracking] button is clicked * Object extractor is showed * User select tracking template from capture frame. * User click [OK]   **Alternative Scenario:**  NONE  **Exceptions:**   * IO Exception * Time Out Exception   **Relationships:**  NONE | | | | |

Table C‑17 Choose Target use-case specification

##### Change Target

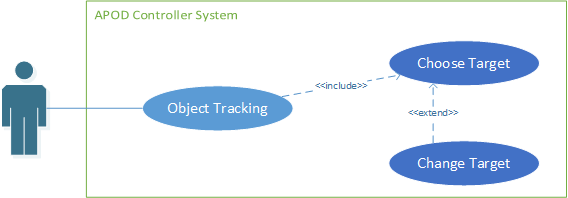


Figure C‑19 Change Target Use-case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ADD POINT USE CASE SPECIFICATION** | | | | |
| **Use-case No.** | UC018 | **Use-case Version** | | 1.0 |
| **Use-case Name** | Change Target | | | |
| **Author** | CuongPAD | | | |
| **Date** | 05/10/2013 | **Priority** | Normal | |
| **Actor:**  User  **Summary:**  Change current tracking template  **Goal:**  New tracking template is selected  **Triggers**  User click [Change target] button  **Preconditions:**   * Configuration is initiated * Bluetooth connection is established * IP camera stream is working * Tracking mode is selected   **Post Conditions:**  NONE  **Main Success Scenario:**   * [Change Target] button is clicked * Object extractor is showed * User select tracking template from capture frame. * User click [OK]   **Alternative Scenario:**  NONE  **Exceptions:**   * IO Exception * Time Out Exception   **Relationships:**  NONE | | | | |

Table C‑18 Change Target use-case specification

##### Lock Target

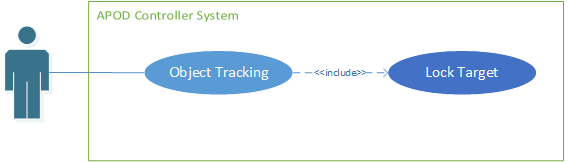


Figure C‑20 Lock Target Use-case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ADD POINT USE CASE SPECIFICATION** | | | | |
| **Use-case No.** | UC019 | **Use-case Version** | | 1.0 |
| **Use-case Name** | Lock Target | | | |
| **Author** | CuongPAD | | | |
| **Date** | 05/10/2013 | **Priority** | Normal | |
| **Actor:**  User  **Summary:**  Start tracking target  **Goal:**  Reach target (within 60 cm)  **Triggers**  User click [Lock] button  **Preconditions:**   * Configuration is initiated * Bluetooth connection is established * IP camera stream is working * Tracking template was set   **Post Conditions:**  NONE  **Main Success Scenario:**   * User click on [Lock] button * APOD keep moving forward (with adjustment) until target is reached * User decide to Grab object or not.   **Alternative Scenario:**  NONE  **Exceptions:**   * IO Exception   **Relationships:**  NONE | | | | |

Table C‑19 Lock Target use-case specification

## SOFTWARE SYSTEM ATTRIBUTES

### Reliability

* Easy to upgrade firmware.
* The APOD 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 APOD system will be shut down and reset automatically.
* Controller has the flexibility that allows changes in hardware design.

### Security

* N/A

### Maintainability

* Each module of the APOD controller system can be easily upgrade.
* APOD embedded controller can be recycled and integrated into other control system.

### Portability

* The APOD controller is depend on hardware so that APOD system do not have the portability attribute. However, the APOD 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

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

# 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

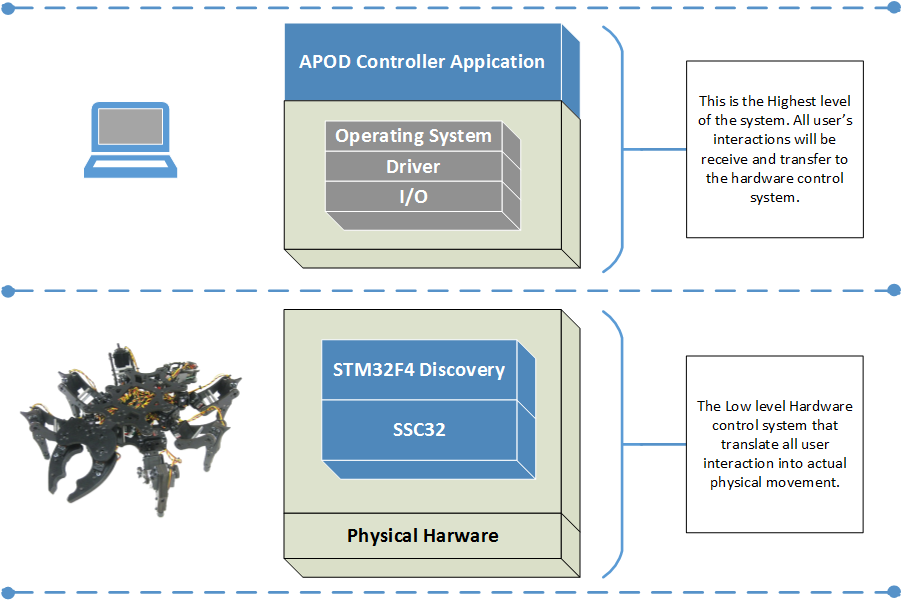


Figure D‑1 APOD Controller Architecture

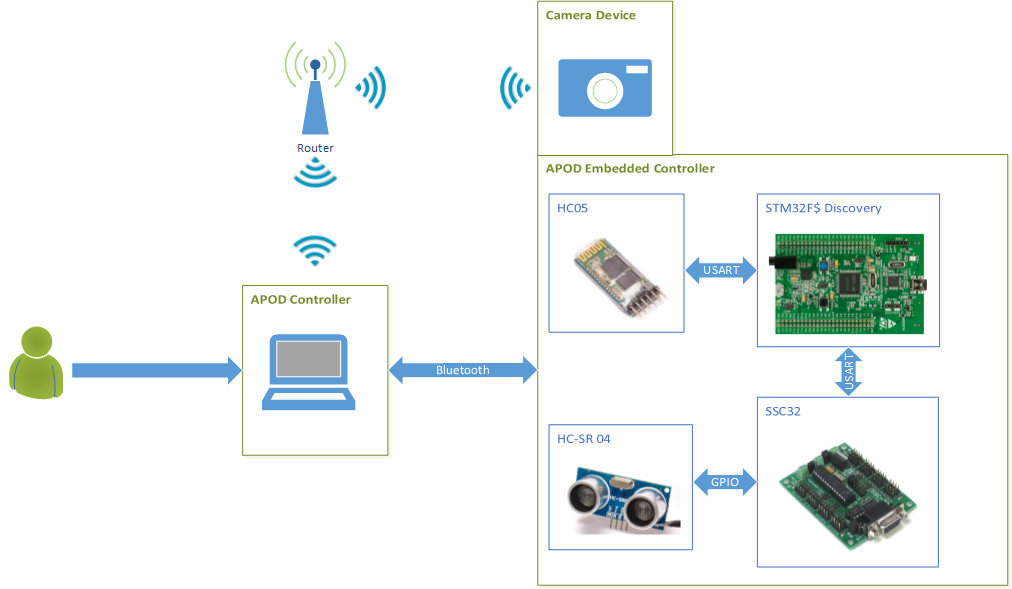


Figure D‑2 APOD Communication Mechanism

### User interface

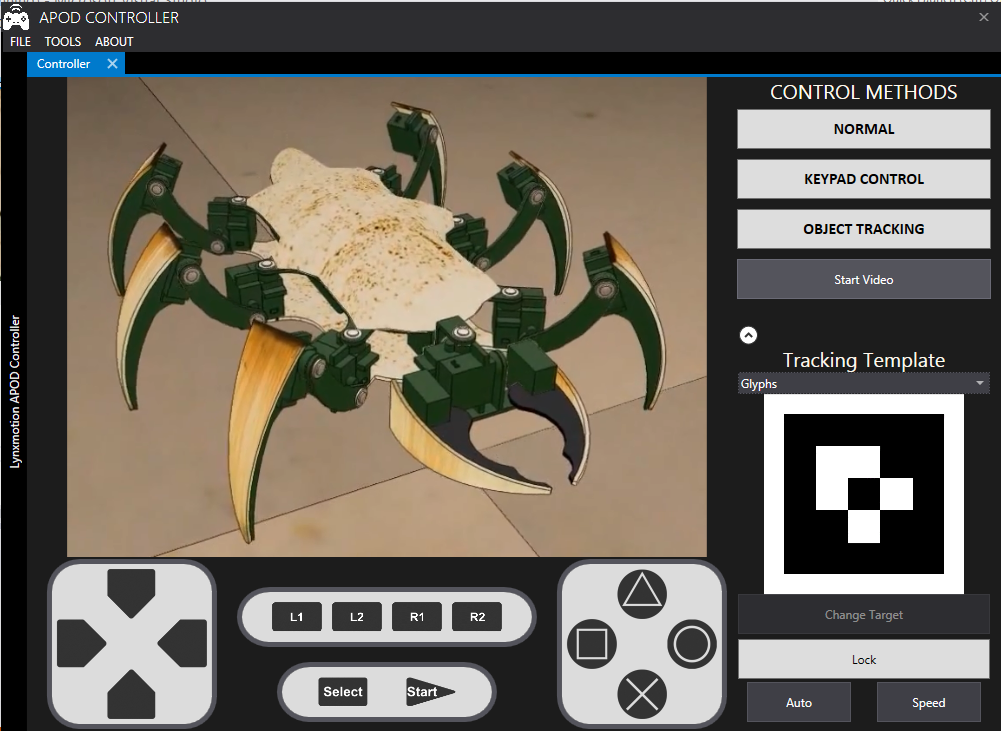


Figure D‑3 Live Control User interface design

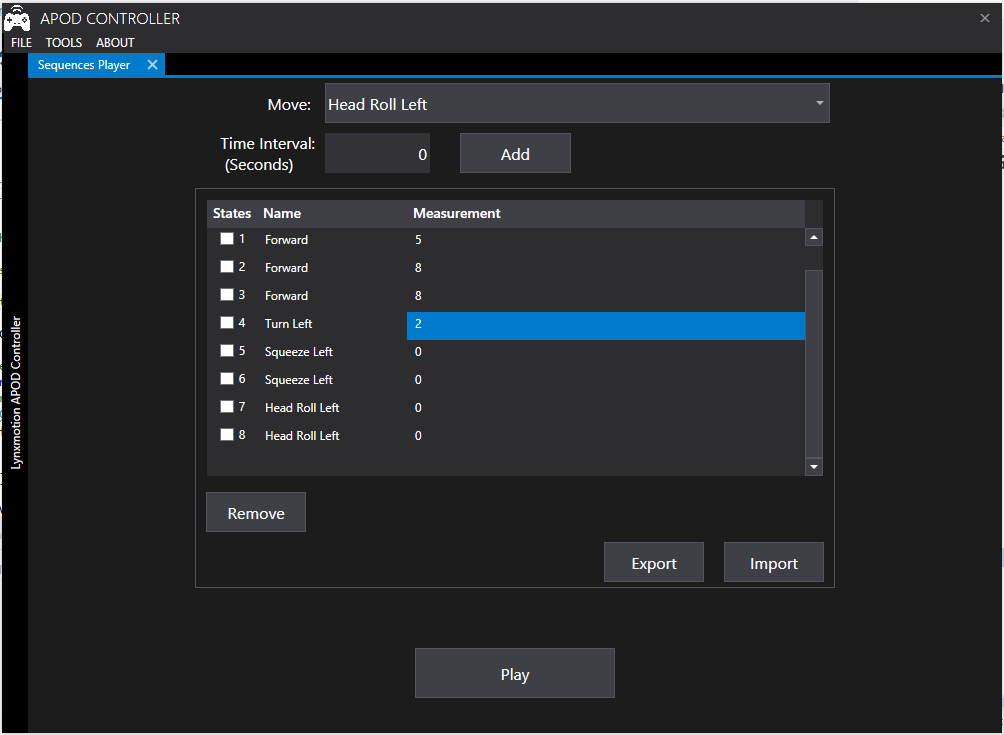


Figure D‑4 Sequence Player user interface design

## COMPONENTS

### Central processing

#### Bluetooth module (HC 05)

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

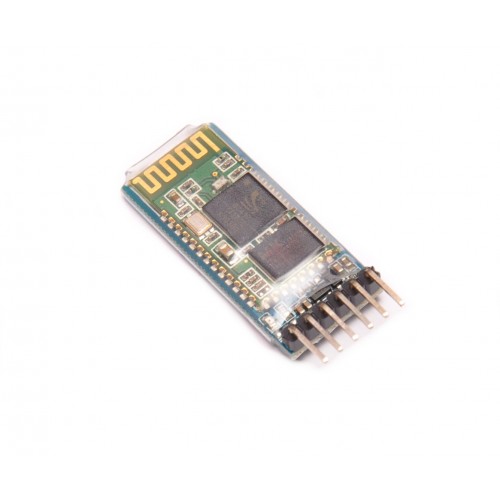


Figure D‑5 Bluetooth Communication module HC05

#### STM32F4 Discovery

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



Figure D‑6 STM32F4 Discovery Board

### Servos control & Sensor reader

#### HG-645MG Servo

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



Figure D‑7 HC 645MG Servo

#### 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)

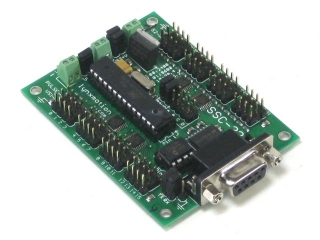


Figure D‑8 SSC32 Board

#### Ultrasonic ranging module (HC SR04)

* Ultrasonic module using for distance measurement, discovering obstacles.



Figure D‑9 HCSR04 Distance Sensor

## IMPLEMENTATION APPROACHES

### Algorithms and Mechanism

#### RC Servo Control

##### HG-645MG Specification

|  |  |
| --- | --- |
| Motor Type: | 3 Pole |
| Bearing Type: | Dual Ball Bearing |
| Speed (4.8V/6.0V): | 0.24 / 0.20 |
| Torque oz./in. (4.8V/6.0V): | 107 / 133 |
| Torque kg./cm. (4.8V/6.0V): | 8.0 / 10.0 |
| Size in Inches: | 1.59 x 0.77 x 1.48 |
| Size in Millimeters: | 40.39 x 19.56 x 37.59 |
| Weight ounces: | 1.94 |
| Weight grams: | 55 |

##### How does it work?

Unlike the AC/DC motor, Servo motor do not rotate continuously. Servo motor rotation range usually fall within 90o to 180o (some special servo can rotate up to 360o) and keep holding the torque at the position as long as its power supply remain or until it receive signal to move the torque.

Most of Servos have 3 wires connector. One wire for power supply (4 – 6v). The second wire is for voltage ground. And the third wire is for receiving control signal.

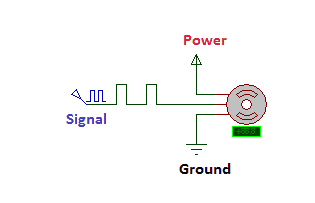


Figure D‑10 RC Servo block

The whole idea to control servo is sending the appropriate signal to servo so that it will move to the intended position. The signal to control servo, basically is a PWM (Pulse Width Modulation) signal of 50 Hz (or refresh rate: 20ms). It is the width of carrier pulse that decide the position of the torque (not the number of pulse).



Figure D‑11 Servo Signal illustration

In order for the servo to keep the position, the signal must be maintain.

In short words, the pulse width of carrier signal will act as a coordinate value of the servo’s torque. Once the coordinate is changed, the servo constantly move the toque to the responding position discarding of its current position. The rotating speed will be depending on the manufacturer of the servo.

For the HG-645MG servo, the corresponding signal is showed below:

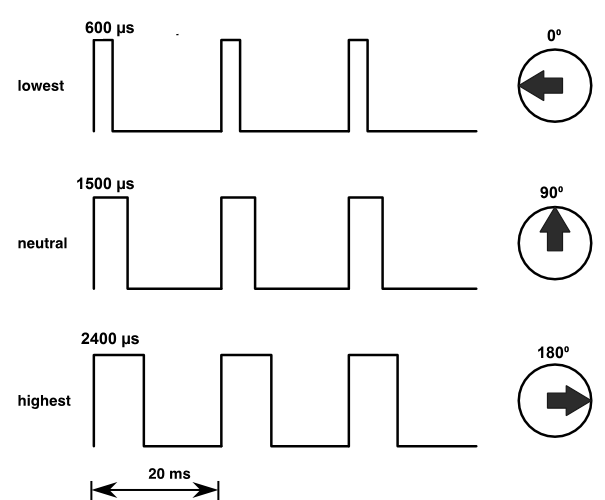


Figure D‑12 HC 645MG Signal Details

##### Servo’s Speed Control

As we know, the servo will always rotate at a constant speed, for example the servo HG-645MG speed is 60o/ 0.2s with 6V supply. With such speed applied to robot’s movement, it can affect the stability of the structure, especially for those in which mechanical part is not so much reliable. How do we slow it down?

The idea is not making the servo rotate a whole distance at once but make it rotate one step at a time until it reach its destination, in short it is “divide and conquer” method. For example, you want your HG-645MG servo to rotate 90o in 1 second and the current position of your servo is 0o. If you just apply the PWM with 1500 µs pulse width (1500 µs stand for 90o position), the servo will move to its destination in only 0.3 second. With the method mentioned above, we divide the whole 90o into ten 9o pieces (or 150 µs step as we call it). The signal pulse width will be modified 10 times, after each time the width will be added 150 µs more and waiting 100 ms until the next change happen. With this implementation, the servo will move 10x9o = 90o in 10x100 = 1000 ms = 1 s. The larger the divisor is, the smoother the servo will move.

For this project, the step is always fix at 20 µs and the waiting time fall in range of 10 ms to 20 ms depend on user purposes.

Let D be the travelling distance (in µs), T be the expected time for the move, δ be the waiting time for each step. We have the speed formula:

#### Sensor Reader (HC-SR04)

##### HC-SR04 Specifications

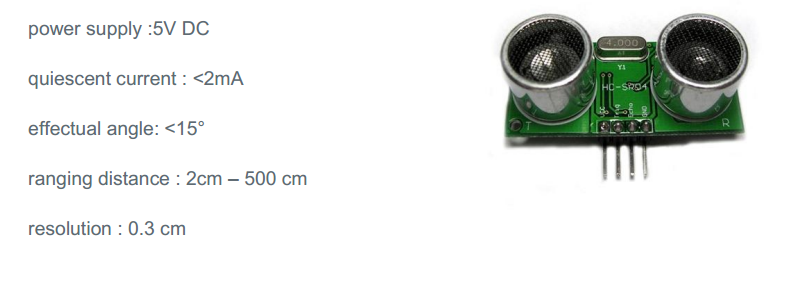


Figure D‑13 HC-SR04 Specifications

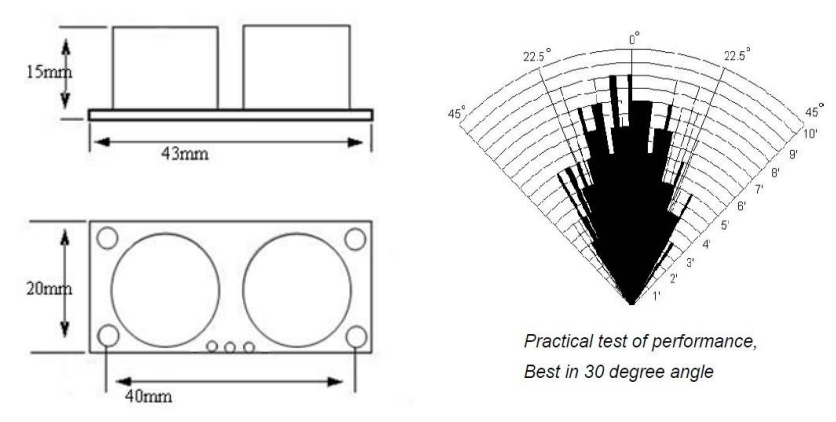


Figure D‑14 HC-SR04 Best practice angle

##### How to read distance value?

The HC-SR04 has 4 pins: VCC, GND, Trigger, Echo.

* VCC is used for power supply
* GND is used for voltage ground
* Trigger and Echo will perform as follow



Figure D‑15 HC-SR04 Operating sequence

First the Trigger pin is pull high for more than 10 µs (10 µs pulse), then waiting for the Echo output to go high and start timing the length of the Echo pin staying high. At this point, a short ultrasonic pulse is transmitted, waiting to be reflected by an obstacle. When the reflected ultrasonic pulse is receive by senor, the Echo output will go low.

The time that Echo output staying high will be used to calculate the distance by following formula given by the manufacturer:

Or

#### Multi-legs Control

The description below is described in the following article by Oricom Technologies– “Analysis of Multi-Legged Animal + Robot Gaits” (2003).

##### Tripod Gait

The tripod gait is the best-known hexapod gait. Hexapod six legs is divide into 2 halves for to control. Each half (or tripod) consists of 3 legs: the front and rear leg of one side and the center leg of the opposite side. More specific:

* Tripod A: left front, right center and left rear
* Tripod B: right front, left center and right rear.



Figure D‑16 Tripod Gait illustration

During walking cycle, the weight of the entire hexapod is shifted from one tripod to another. At a moment, there is always one tripod on the ground while another doing its cycle. “*Since 3 legs are on the ground at all times, this gait is both "statically" and "dynamically" stable*.” - (Oricom Technologies , 2003)

##### Wave Gait

The wave gait implementation make each leg in one side do it cycle after one another leg has finished its cycles and repeat it on the other side’s legs.

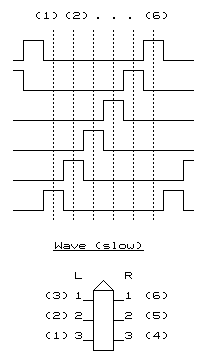


Figure D‑17 Wave Gait illustration

For instance, the turn of leg taking step is as follow: Left Front -> Left Center -> Left Rear -> Right Front -> Right Center -> Right Rear

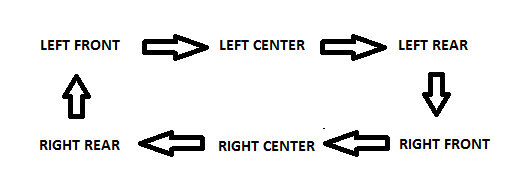


Figure D‑18 Wave Gait step order

Since there is always 5 legs on the ground, this method is the most stable gait but in return, it is also the slowest gait due to its “*one step at a time*”.

##### Ripple Gait

“*The final stride is the Ripple Gait. At first glance the timing of this gait looks somewhat complicated, however, the key to understanding is to recognize that, on each side a local wave comprising non-overlapping lift phases is being performed, and that the 2 opposite side waves are exactly 180 degrees out of phase with one another. For instance, if L3 and R3 are considered to represent the start of each local wave, then notice that R3 starts to move exactly in the middle of the L3-L2-L1 side wave*” .” (Oricom Technologies , 2003)

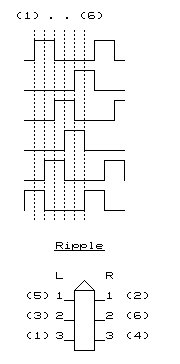


Figure D‑19 Ripple Gait illustration

##### Implementation Approaches

For this project, we will apply **the tripod gait** method to control the APOD walking movement. As we know, 6 legs of APOD will be divided into 2 tripods

We defined 2 tripod as follow:

* **Tripod A: Left front, Right Center, Left Rear**
* **Tripod B: Right front, Left Center, Right Rear**

###### Walking In Straight Line

**Leg’s Position:**

There are 6 basic positions for the APOD’s leg divive into 2 orientation.

* Vertical: High, Middle , Low
* Horizontal: Front, Center, Rear

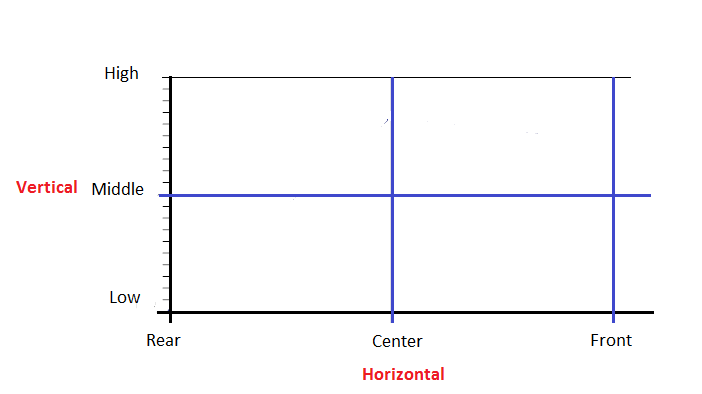


Figure D‑20 APOD leg position block

Each step (swing time) of the APOD leg is represented by a sequence of 4 consecutive states defined by the position of the leg in both team of vertical and horizontal positions.

For example, this is the forward step sequence:

|  |  |  |
| --- | --- | --- |
| State | Vertical | Horizontal |
| 0 | Low to Mid | Rear |
| 1 | Mid to High | Rear to Center |
| 2 | High to Mid | Center to Front |
| 3 | Mid to Low | Front |

Table D‑1 Leg swing sequence

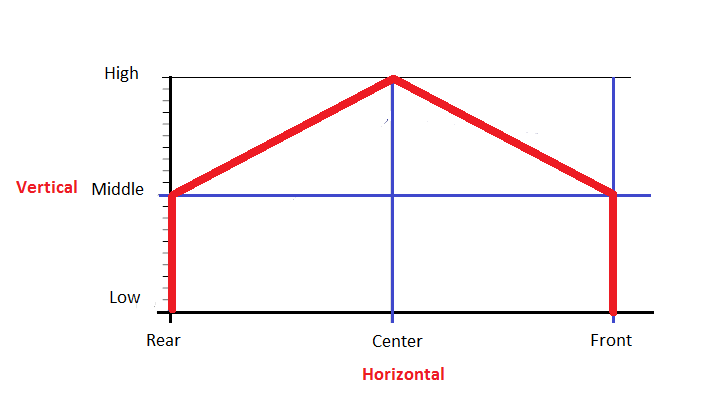


Figure D‑21 APOD leg swing diagram

Depend on the step direction (forward or backward) the sequence may differ: “Front” and “Rear” may be replaing each other’s position.

After making the step, the leg will return to its original position (compare to the body) while keeping the feet on the ground (stand time). This is the point the force was born to push the body moving in the opposite direction of the moving legs.

From the APOD view point, the whole operations of moving leg can be seen as the figure below:

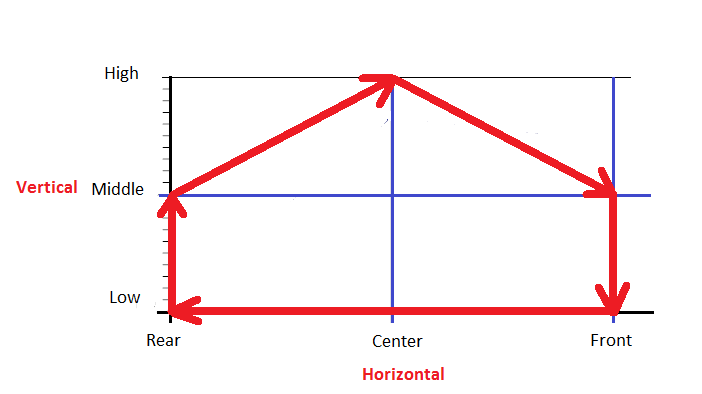


Figure D‑22 APOD Leg full step diagram

The whole operation of APOD’s leg can be seen as this sequence of states (Forward step) looping times to times:

|  |  |  |
| --- | --- | --- |
| **State** | **Vertical** | **Horizontal** |
| 0 | Low | Front to Center |
| 1 | Low | Center to Rear |
| 2 | Low | Rear |
| 3 | Low to Mid | Rear |
| 4 | Mid to High | Rear to Center |
| 5 | High to Mid | Center to Front |
| 6 | Mid to Low | Front |
| 7 | Low | Front |

Table D‑2 APOD Leg full forward step

**Tripod Position**

According to the nature of The Tripod Gait, every 3 legs in one tripod will always move and stop at the same time. In other words, their cycles are exactly the same phase with one another. The term tripod position is defined by the combination of its legs positions.

Let’s take a view of Tripod A (marked with red color) from the up top viewpoint for example:

* Position: Front (close to head)

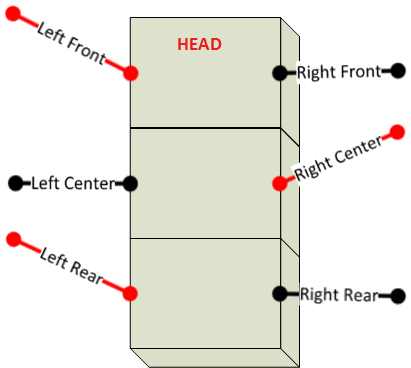


Figure D‑23 Tripod A at Front position

* Position: Center or Neutral

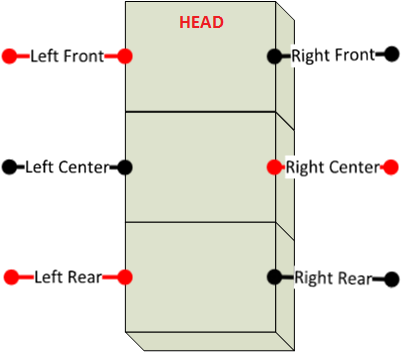


Figure D‑24 Tripod A at Neutral position

* Position: Rear (far from head)

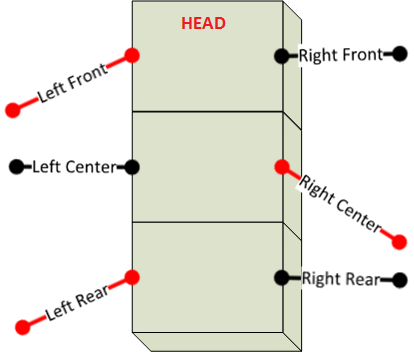


Figure D‑25 Tripod A at Rear position

Since all Tripod A’s Legs making the forward step at once (moving from front to rear), there will be 2 pushing force from both side of the body, from the back to the head, that making the APOD moving forward. This is the fundamental of making the APOD moving forward. The backward move can be achieve by making the reverse step of forward step (the backward step).

**Walking**

Now before we can figure out the walking method of APOD, let’s examine the human walking method first. As we can see, the position of one leg in walking (or running) is always the opposite position of the other one. Or in short, their cycles is exactly 180o out of phase with one another. The similar mechanism is applied for The Tripod Gait walking method. Let Tripod A and B stand for 2 legs of human, then their cycles will always be 180o out of phase with the other one. In conclusion, the full sequence for 2 tripods of The Tripod Gait walking method is described as below

**Forward Sequence**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **State** | **Tripod A** | | **Tripod B** | |
| **Vertical** | **Horizontal** | **Vertical** | **Horizontal** |
| 0 | Low | Front to Center | Mid to High | Rear to Center |
| 1 | Low | Center to Rear | High to Mid | Center to Front |
| 2 | Low | Rear | Mid to Low | Front |
| 3 | Low to Mid | Rear | Low | Front |
| 4 | Mid to High | Rear to Center | Low | Front to Center |
| 5 | High to Mid | Center to Front | Low | Center to Rear |
| 6 | Mid to Low | Front | Low | Rear |
| 7 | Low | Front | Low to Mid | Rear |

Table D‑3 Forward Sequence

**Backward Sequence**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **State** | **Tripod A** | | **Tripod B** | |
| **Vertical** | **Horizontal** | **Vertical** | **Horizontal** |
| 0 | Low | Rear to Center | Mid to High | Front to Center |
| 1 | Low | Center to Front | High to Mid | Center to Rear |
| 2 | Low | Front | Mid to Low | Rear |
| 3 | Low to Mid | Front | Low | Rear |
| 4 | Mid to High | Front to Center | Low | Rear to Center |
| 5 | High to Mid | Center to Rear | Low | Center to Front |
| 6 | Mid to Low | Rear | Low | Front |
| 7 | Low | Rear | Low to Mid | Front |

Table D‑4 Backward Sequence

###### Turning Left – Right

As mention above, we already know 3 basic position of tripod: Front, Center, Rear. In this section, there are 2 more position: Left and Right.

The definition of 2 new position can be described by tripod A as follow:

* Position: Left

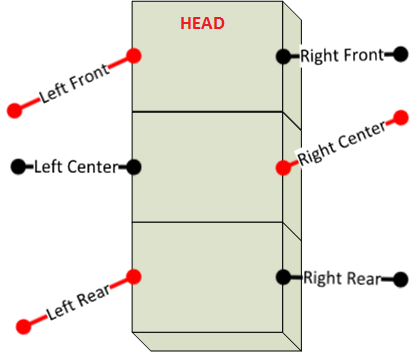


Figure D‑26 Tripod A at Left position

* Position: Right



Figure D‑27 Tripod A at Right position

Let’s take a look when Tripod A making set :

* Left Front and Left Rear take the forward step
* Right Center take the backward step

There will be forces with 2 opposite direction on the APOD body. The left side force push the body forward and the right side force push the body backward. As a result of this combination. The APOD will turn from left to right. In other words, when the tripod making transition from left position to right position (through center position), the APOD will turn from left to right. And when the tripod making transition from right position to left position (through center position), the APOD will turn from right to left.

**The combination sequence for turning the APOD:**

**Turn Left:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **State** | **Tripod A** | | **Tripod B** | |
| **Vertical** | **Horizontal** | **Vertical** | **Horizontal** |
| 0 | Low | Left to Center | Mid to High | Right to Center |
| 1 | Low | Center to Right | High to Mid | Center to Left |
| 2 | Low | Right | Mid to Low | Left |
| 3 | Low to Mid | Right | Low | Left |
| 4 | Mid to High | Right to Center | Low | Left to Center |
| 5 | High to Mid | Center to Left | Low | Center to Right |
| 6 | Mid to Low | Left | Low | Right |
| 7 | Low | Left | Low to Mid | Right |

Table D‑5 Turn Left Sequence

**Turn Right:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **State** | **Tripod A** | | **Tripod B** | |
| **Vertical** | **Horizontal** | **Vertical** | **Horizontal** |
| 0 | Low | Right to Center | Mid to High | Left to Center |
| 1 | Low | Center to Left | High to Mid | Center to Right |
| 2 | Low | Left | Mid to Low | Right |
| 3 | Low to Mid | Left | Low | Right |
| 4 | Mid to High | Left to Center | Low | Right to Center |
| 5 | High to Mid | Center to Right | Low | Center to Left |
| 6 | Mid to Low | Right | Low | Left |
| 7 | Low | Right | Low to Mid | Left |

Table D‑6 Turn Right Sequence

### SSC32 Board

#### Schematic

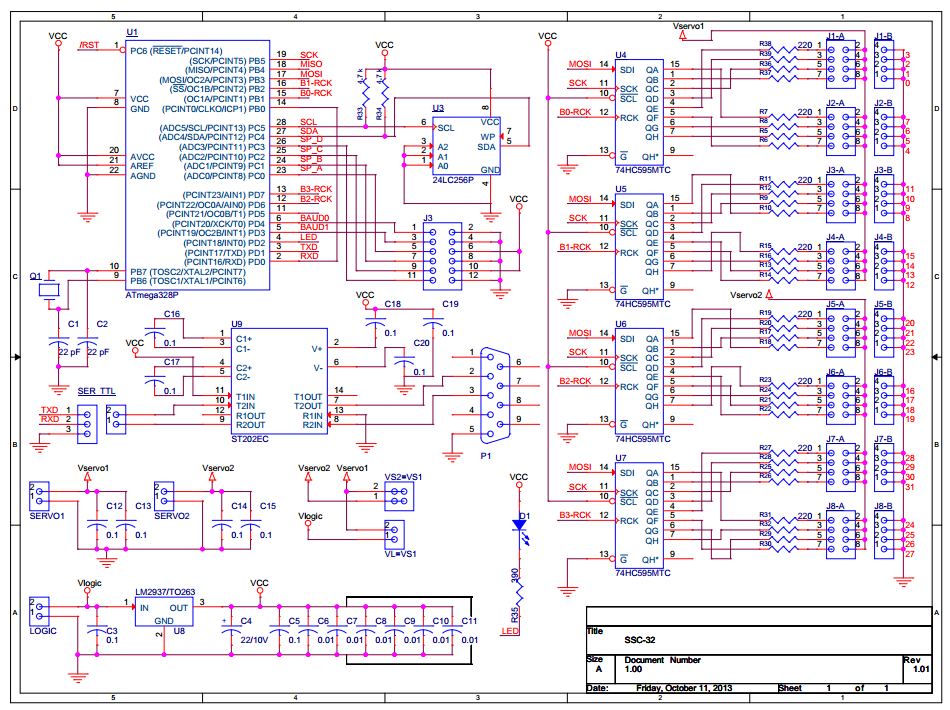


Figure D‑28 SSC32 Schematic

#### Technical Point

##### Atmega328P Microcontroller (MCU)

###### Overview

ATmega328p is a high-performance Atmel picoPower 8-bit AVR RISC-based (Reduced Instruction Set Computing) microcontroller provided with these features:

* 32KB ISP (In-System Programming) flash memory with read-while-write capabilities
* 1024B EEPROM (Electrically Erasable Programmable Read-Only Memory)
* 2KB SRAM (Static Random-Access Memory)
* 23 general purpose I/O lines
* 32 general purpose working registers
* Three flexible timer/counters with compare modes
* Internal and external interrupts
* Serial programmable USART (Universal Synchronous and Asynchronous serial Receiver and Transmitter)
* A byte-oriented 2-wire serial interface
* SPI (Serial Peripheral Interface) serial port
* A 6-channel 10-bit A/D
* Programmable watchdog timer with internal oscillator
* Five software selectable power saving modes

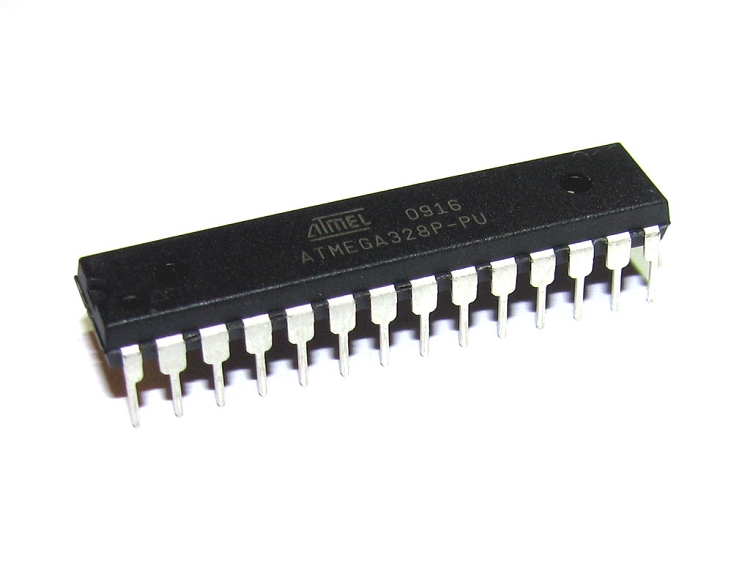


Figure D‑29 ATmega328p Microcontroller

For more information about ATmega328 Features and Specifications, please refer to the ATmega328p Datasheet from Atmel web site:

<http://www.atmel.com/devices/atmega328p.aspx?tab=documents>

###### Pin Configuration

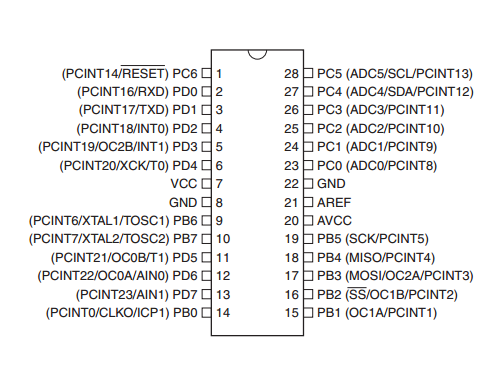


Figure D‑30 ATmega328p 28 Pins DIP

###### Block Diagram

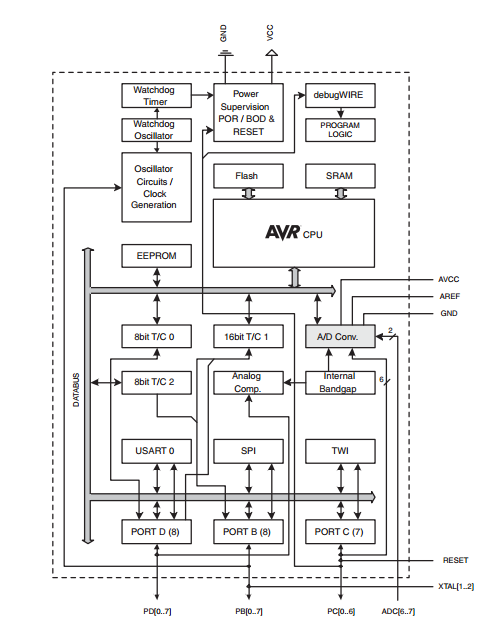


Figure D‑31 ATmega328p Block Diagram

##### General Purposes Input / Output (GPIO)

The ATmega328p has 3 Ports: Port B, Port C, Port D and each of them consists of 8 pin PB[0..7], PC[0..7], PD[0..7]

Each port pin consists of three register bits: DDxn, PORTxn, and PINxn. The DDxn bits are accessed at the DDRx I/O address, the PORTxn bits at the PORTx I/O address, and the PINxn bits at the PINx I/O address. The details will be described later.

Notes:

* “x” represent the port name, can be “B”,”C” or “D”.
* “n” represent the pin number of the x port, vary from 0 to 7 (each port has 8 pins, as mentioned above)

###### Configurations

The DDxn bit in the DDRx Register selects the direction of this pin. If DDxn is written logic one, Pxn is configured as an output pin. If DDxn is written logic zero, Pxn is configured as an input pin.

If PORTxn is written logic one when the pin is configured as an input pin, the pull-up resistor is activated. To switch the pull-up resistor off, PORTxn has to be written logic zero or the pin has to be configured as an output pin.

If PORTxn is written logic one when the pin is configured as an output pin, the port pin is driven high (one). If PORTxn is written logic zero when the pin is configured as an output pin, the port pin is driven low (zero).

Writing a logic one to PINxn toggles the value of PORTxn, independent on the value of DDRxn.

###### Register Description

MCUCR

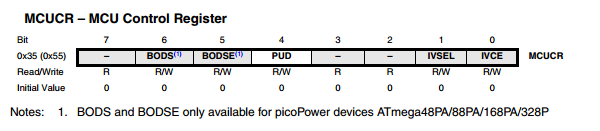


Figure D‑32 MCU Control Register

* Bit 4 - PUD: Pull-up Disable

When this bit is written to one, the pull-ups in the I/O ports are disabled even if the DDxn and PORTxn Registers are configured to enable the pull-ups.

PORT B

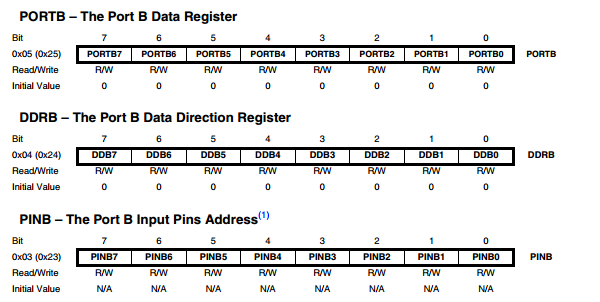


Figure D‑33 PORT B Register Description

PORT C

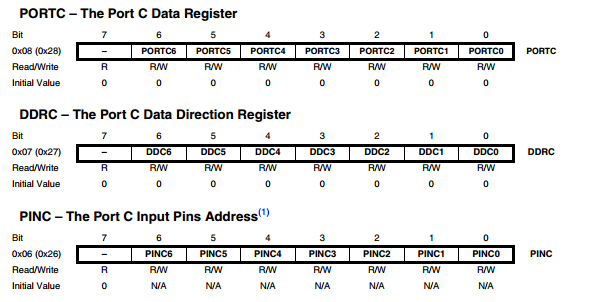


Figure D‑34 PORT C Register Description

PORT D

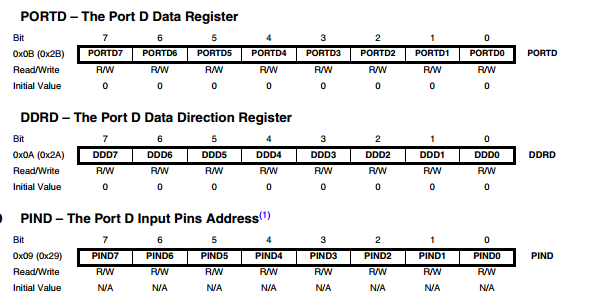


Figure D‑35 PORT D Register Description

##### Interrupts

The ATmega328p provides 26 interrupt vectors for user. In the scope of this project, we only use some of the interrupts services listed in the table below:

|  |  |  |
| --- | --- | --- |
| Number | Source | Definition |
| 1 | PCINT1 | Pin Change Interrupt Request 1 |
| 2 | TIMER1 COMPA | Timer/Counter1 Compare Match A |
| 3 | TIMER1 COMPB | Timer/Coutner1 Compare Match B |
| 4 | TIMER0 COMPA | Timer/Counter0 Compare Match A |
| 5 | USART, RX | USART Rx Complete |
| 6 | USART, TX | USART, Tx Complete |

Table D‑7 Interrupts Usage

##### Timer

###### Usage and Purposes

The main feature of the SSC32 board is to control servos, more specific: 32 HC645MG servos. As mention in previous section, a RC servo is controlled by a signal, which basically is a PWM signal or simply an output pin toggled at specific time (high <-> low). It is the responsibility of Timer making the toggle to generate the PWM signal controlling servos. To support that purpose of this project, we will use 16-bit Timer of the ATmega328p (TIMER/COUNTER 1).

###### Register Description

TTCR1A

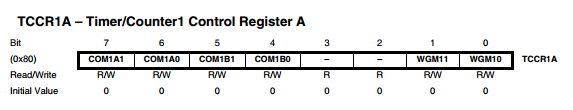


Figure D‑36 Timer 1 Control register A

* Bit 7:6 - COM1A1:0: Compare Output Mode for Channel A
* Bit 5:4 - COM1B1:0: Compare Output Mode for Channel B
* Bit 3:2 - Reserved: read as “0”
* Bit 1:0 - WGM11:0: Waveform Generation Mode

TTCR1B

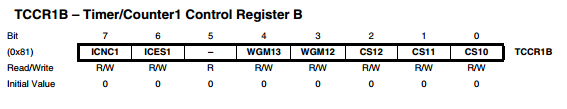


Figure D‑37 Timer 1 Control register B

* Bit 7 - ICNC1: Input Capture Noise Canceler
* Bit 6 - ICES1: Input Capture Edge Select
* Bit 5 - Reserved: Read as “0”
* Bit 4:3 - WGM13:2: Waveform Generation Mode
* Bit 2:0 - CS12:0: Clock Select

TTCR1C

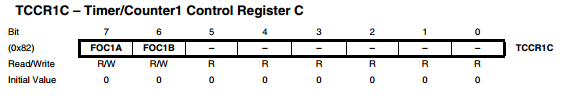


Figure D‑38 Timer 1 Control register C

* Bit 7 - FOC1A: Force Output Compare for Channel A
* Bit 6 - FOC1B: Force Output Compare for Channel B
* Bit 5:0 - Reserved: Read as “0”

TCNT1

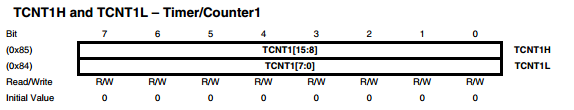


Figure D‑39 Timer 1 Counter

ORC1A

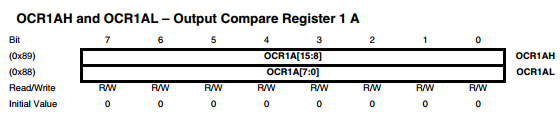


Figure D‑40 Output Compare Register A

ORC1B

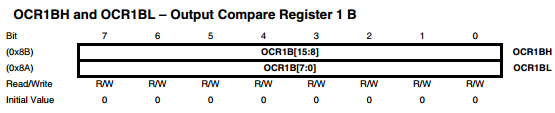


Figure D‑41 Output Compare Register B

IRC1

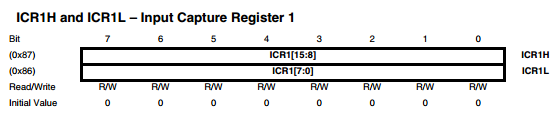


Figure D‑42 Input Capture Register

TIMSK1

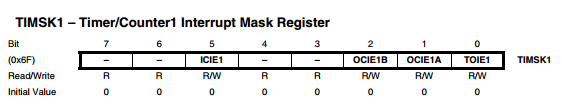


Figure D‑43 Timer/Counter1 Interrupt Mask Register

* Bit 7, 6 - Reserved: Read as “0”
* Bit 5 - ICIE1: Timer/Counter1, Input Capture Interrupt Enable
* Bit 4, 3 - Reserved: Read as “0”
* Bit 2 - OCIE1B: Timer/Counter1, Output Compare B Match Interrupt Enable
* Bit 1 - OCIE1A: Timer/Counter1, Output Compare A Match Interrupt Enable
* Bit 0 -TOIE1: Timer/Counter1, Overflow Interrupt Enable

TIFR1

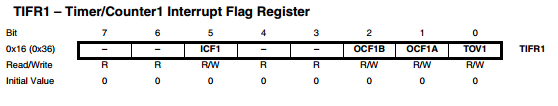


Figure D‑44 Timer 1 Interrupt Flog register

* Bit 7, 6 - Reserved: Read as “0”
* Bit 5 - ICF1: Timer/Counter1, Input Capture Flag
* Bit 4, 3 - Reserved: Read as “0”
* Bit 2 - OCF1B: Timer/Counter1, Output Compare B Match Flag
* Bit 1 - OCF1A: Timer/Counter1, Output Compare A Match Flag
* Bit 0 - TOV1: Timer/Counter1, Overflow Flag

##### Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART)

###### Usage and Purposes

In this project, the use of USART will be the main communication channel between SSC32 module and other modules. The command will be receive through this channel as well as the response from SSC32 (when required).

###### Baud Rate

The SSC32 support method for user to change baud rate of USART via 2 jumper on board baud0 and baud1

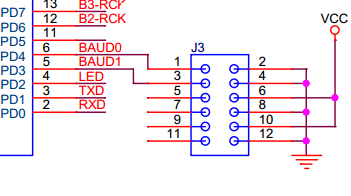


Figure D‑45 Baud Rate setup

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **BAUD0** | **BAUD1** | **Baud Rate** | **UBRR0H** | **UBRR0L** |
| 0 | 0 | 2400 | 1 | 7F |
| 0 | 1 | 9600 | 0 | 5F |
| 1 | 0 | 38400 | 0 | 17 |
| 1 | 1 | 115200 | 0 | 7 |
| Note:  - 1: indicate jumper is close  - 0: indicate jumper open  - The details about UBRR0H and UBRR1H is in the section below. | | | | |

Table D‑8 Baud Rate setup details

After apply the jumper(s) on baud rate selection pin(s), the ATmega328p must be reset for the new baud rate to be applied.

###### Register Description

UDRn

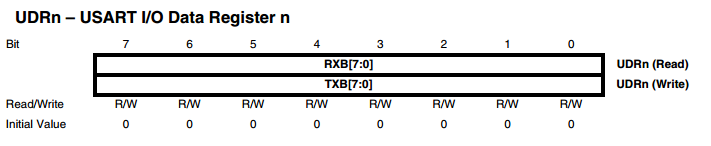


Figure D‑46 USART I/O Data register

The USART Transmit Data Buffer Register and USART Receive Data Buffer Registers share the same I/O address referred to as USART Data Register or UDRn. The Transmit Data Buffer Register (TXB) will be the destination for data written to the UDRn Register location. Reading the UDRn Register location will return the contents of the Receive Data Buffer Register (RXB).

UCSRnA

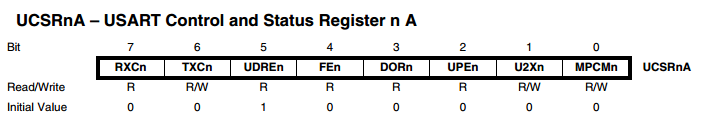


Figure D‑47USART Control and Status Register A

* Bit 7 - RXCn: USART Receive Complete
* Bit 6 - TXCn: USART Transmit Complete
* Bit 5 - UDREn: USART Data Register Empty
* Bit 4 - FEn: Frame Error
* Bit 3 - DORn: Data OverRun
* Bit 2 - UPEn: USART Parity Error
* Bit 1 - U2Xn: Double the USART Transmission Speed
* Bit 0 - MPCMn: Multi-processor Communication Mode

UCSRnB

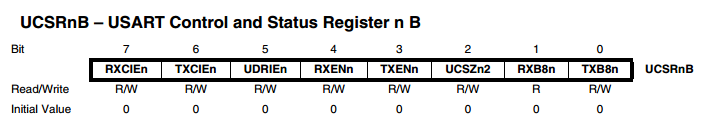


Figure D‑48 USART Control and Status register B

* Bit 7 - RXCIEn: RX Complete Interrupt Enable n
* Bit 6 - TXCIEn: TX Complete Interrupt Enable n
* Bit 5 - UDRIEn: USART Data Register Empty Interrupt Enable n
* Bit 4 - RXENn: Receiver Enable n
* Bit 3 - TXENn: Transmitter Enable n
* Bit 2 - UCSZn2: Character Size n
* Bit 1 - RXB8n: Receive Data Bit 8 n
* Bit 0 - TXB8n: Transmit Data Bit 8 n

UCSRnC

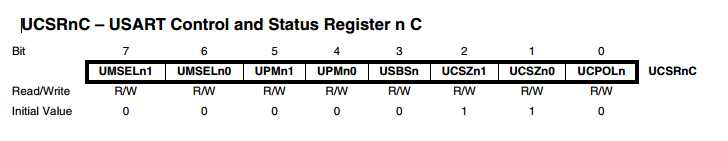


Figure D‑49 USART Control and Status register C

* Bits 7:6 - UMSELn1:0 USART Mode Select
* Bits 5:4 - UPMn1:0: Parity Mode
* Bit 3 - USBSn: Stop Bit Select
* Bit 2:1 - UCSZn1:0: Character Size
* Bit 0 - UCPOLn: Clock Polarity

UBRRnL and UBRRnH

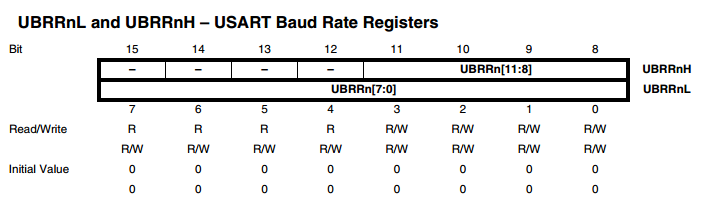


Figure D‑50 USART Baud Rate register

* Bit 15:12 - Reserved: read as “0”
* Bit 11:0 - UBRR[11:0]: USART Baud Rate Register

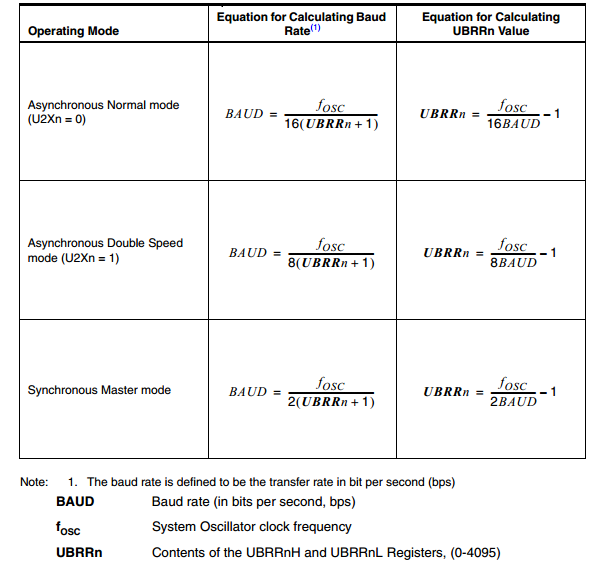


Figure D‑51 Baud Rate Calculation

##### Serial Peripheral Interface (SPI)

###### Usage and Purposes

In this project, SPI communication will only be used for communication between ATmega328P and 74HC595, not for communication between microcontrollers. The detail of SPI- Shift Register interface will be discussed later in this document.

###### Register Description

SPCR

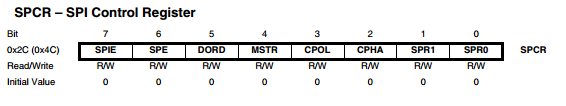


Figure D‑52 SPI Control register

* Bit 7 - SPIE: SPI Interrupt Enable
* Bit 6 - SPE: SPI Enable
* Bit 5 - DORD: Data Order
* Bit 4 - MSTR: Master/Slave Select
* Bit 3 - CPOL: Clock Polarity
* Bit 2 - CPHA: Clock Phase
* Bits 1, 0 - SPR1, SPR0: SPI Clock Rate Select 1 and 0

SPSR

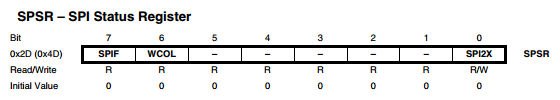


Figure D‑53 SPI Status register

* Bit 7 - SPIF: SPI Interrupt Flag
* Bit 6 - WCOL: Write Collision Flag
* Bit [5:1] – Reserved: Read as “0”
* Bit 0 - SPI2X: Double SPI Speed Bit

SPDR

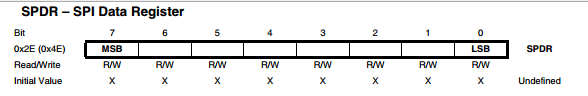


Figure D‑54 SPI Data register

The SPI Data Register is a read/write register used for data transfer between the Register File and the SPI Shift Register

##### Shift Register (74HC595)

###### Overview

74HC595 is a shift register device with 8-bit serial input 8-bit serial or parallel output. This project will use the “serial to parallel conversion” feature of this Shift register.

###### How does it work?

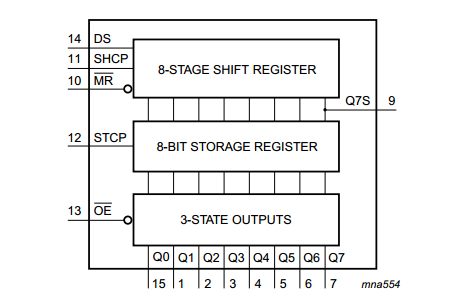


Figure D‑55 74HC595 Block diagram

* **Serial Data Input (DS):** This pin is connected to any port of the microcontroller. This pin is responsible for getting the data serially from the microcontroller.
* **Shift Clock (SH\_CP):** On this pin the clock signal is applied. On the rising edge, that is when the clock rises from negative to positive, the data on the DS line is sampled and it gets stored in the shift register. The bit on DS line is stored on the LSB (least significant place i.e. BIT0). On next pulse this bit moves to BIT1 location. After 8 clock pulses this bit is moved to BIT7 (MSB place). After 8 clock pulses the Shift Register has all 8 bits of a byte and is ready to convert them to parallel.
* **Store Clock (ST\_CP):** You can hold this pin LOW while you get everything setup and nothing on the display pins will change. Then when you are done, and everything is how you want, you pull the pin HIGH and the 74HC595 will display the new settings. So even though we are changing values in the register in 8 steps, it looks like it was just one step
* **MR :** – Will empty the whole shift register, if pulled low, must be pulled high to enable.
* **OE:** – This pin enables the output when tied to ground, & disabled when HIGH.

##### SPI - Shift Register interface

###### Overview

Let’s take a look of how we generate the signal controlling servos using timer. We have 32 servos to be control by the SSC32 Board.

First approach, we will use Output compare feature of timer to generate the PWM signal. With this built-in feature, we can generate highly accurate PWM signal. But the problem is that ATmega328p have only 2 output compare pin for each timer. Even if we use all 3 timers, we can only control 6 servos by this approach.

Second approach, we will use the GPIO to output the PWM signal. The PWM will be toggle manually for each pins in ports. But on total, ATmega328 has only 3 GPIO ports which mean we can only control 8x3 = 24 servos while having 32 servos to control and also wasting all the other features of the MCU.

The third approach is using some external device. In this case, we will use 4 more Shift Registers (74HC595) to output 32 PWM signal (8 pin on each Shift Register).

###### How does it work?

Take a look on SPI diagram

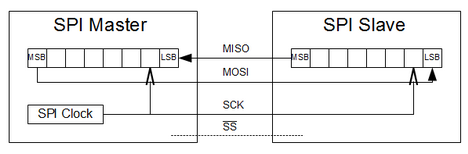


Figure D‑56 SPI Diagram

We know that after the SPI Data register is written the SPI transmitting protocol will be trigger to send out data.

Data in the SPI Data register will be shifted out one bit at a time on the SCK clock, start with the MSB (Most Significant Bit or Most Left Bit). At one clock of SCK, one bit of data will be shift out on MOSI (Master Out Slaver In) pin.

Now we wire the 74HC595 as follow:

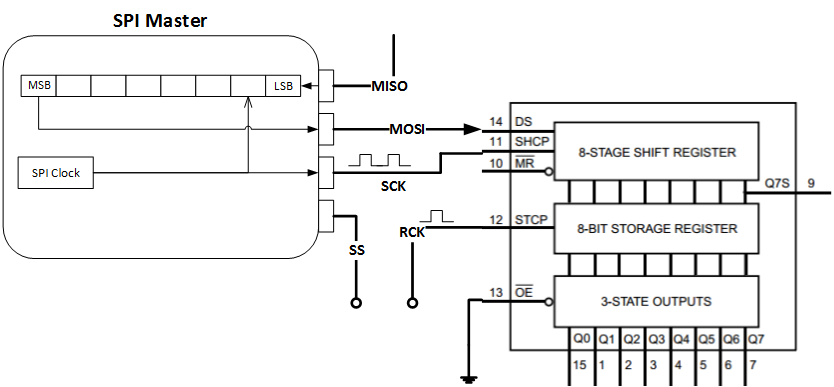


Figure D‑57 SPI- Shift Register

Ignore the MISO (Master In Slaver Out) pin, the MOSI pin is now connect to the DS (Or SDI) pin of the Shift register. SCK will be connected to the SH\_CP Pin of the Shift register (or also SCK).

When the SPI transmitting is trigger, the SPI clock will also be the Shifting clock on the Shift Register. Each bit of the SPI data will follow the clock transfer into the 74HC595 Register through the MOSI – SDI wire. As a result, the SPI become a tool for writing data to shift register automatically. The output of 74HC595 can only change when a high pulse is apply to the ST\_CP Pin (or the RCK as the Figure above).

Using the same MISO and SCK pin for 4 Shifter, 4 more pin for the output trigger. We have extended 6 pins of ATmega328p into 32 pins, which is enough for controlling 32 servos as mentioned above, while not wasting other features of the ATmega328p.

### STM32F4 Discovery Board

#### STM32F407VGT6 Microcontroller (MCU)

##### Overview

This ARM Cortex-M4 32-bit MCU with FPU (Float-Point Unit) has 210 DMIPS (Dhrystone Microprocessor without Interlocked Pipeline Stages), up to 1 MB Flash/192+4 KB RAM (Random Access Memory), USB OTG HS/FS (Universal serial bus on-the-go full-speed/half speed), Ethernet, 17 timers, 3 ADCs, 15 comm. interfaces and a camera.

This device provides the following benefits.

* 168 MHz/210 DMIPS Cortex-M4 with single cycle DSP MAC and floating point unit providing:
  + Boosted execution of control algorithms More features possible for your applications Ease of use
  + Better code efficiency
  + Faster time to market
  + Elimination of scaling and saturation
  + Easier support for meta-language tools
* Designed for high performance and ultra-fast data transfers; ART Accelerator, 32-bit, 7- layer AHB bus matrix with 7 masters and 8 slaves including 2 blocks of SRAM, Multi DMA controllers: 2 general purpose, 1 for USB HS, 1 for Ethernet, One SRAM block dedicated to the core, providing performance equivalent to 0-wait execution from Flash Concurrent execution and data transfers and simplified resource allocation
* Outstanding power efficiency; Ultra-low dynamic power, RTC <1 μA typical in VBAT mode, 3.6 V down to 1.7 V VDD, Voltage regulator with power scaling capability, providing extra flexibility to reduce power consumption for applications requiring both high processing and low power performance when running at low voltage or on a rechargeable battery
* Maximum integration: Up to 1 Mbyte of on-chip Flash memory, 192 Kbytes of SRAM, reset circuit, internal RCs, PLLs, WLCSP package available, providing more features in space constrained applications
* Superior and innovative peripherals providing new possibilities to connect and communicate high speed data and more precision due to high resolution
* Extensive tools and software solutions providing a wide choice within the STM32 ecosystem to develop your applications.

##### Configuration

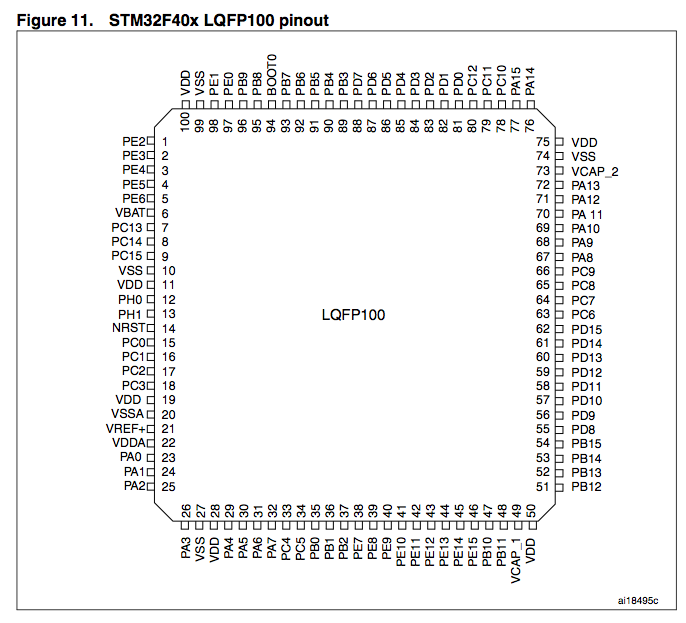


Figure D‑58 STM32F407VGT6 Pin

##### Block Diagram

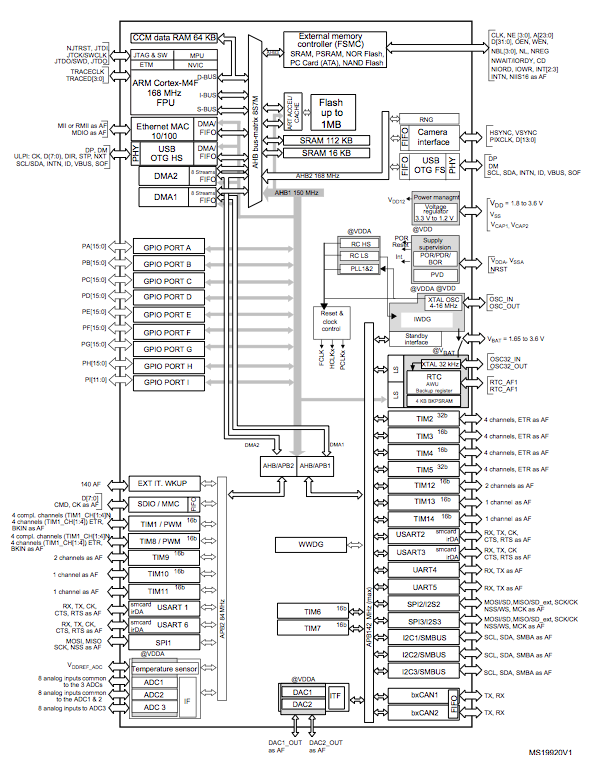


Figure D‑59 STM32F407VGT6 Block Diagram

#### General Purposes Input / Output (GPIO)

##### Features

* Up to 16 I/O s under control
* Output states: push-pull or open drain + pull-up/down
* Output data from output data register (GPIOx\_ODR) or peripheral (alternate function output)
* Speed selection for each I/O
* Input states: floating, pull-up/down, analog
* Input data to input data register (GPIOx\_IDR) or peripheral (alternate function input)
* Bit set and reset register (GPIOx\_BSRR) for bitwise write access to GPIOx\_ODR
* Locking mechanism (GPIOx\_LCKR) provided to freeze the I/O configuration
* Analog function
* Alternate function input/output selection registers (at most 16 AFs per I/O)
* Fast toggle capable of changing every two clock cycles
* Highly flexible pin multiplexing allows the use of I/O pins as GPIOs or as one of several peripheral functions

##### Configuration

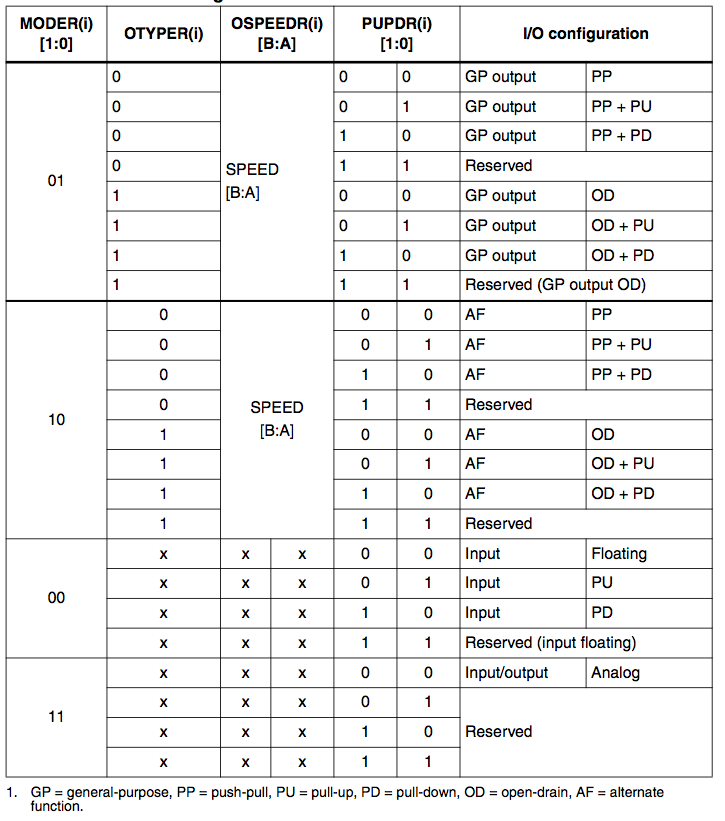


Figure D‑60 STM32F407VGT6 GPIO Configuration

##### Register Description

Each of the GPIOs has four 32-bit memory-mapped control registers (GPIOx\_MODER, GPIOx\_OTYPER, GPIOx\_OSPEEDR, GPIOx\_PUPDR) to configure up to 16 I/Os. The GPIOx\_MODER register is used to select the I/O direction (input, output, AF, analog). The GPIOx\_OTYPER and GPIOx\_OSPEEDR registers are used to select the output type (push- pull or open-drain) and speed (the I/O speed pins are directly connected to the corresponding GPIOx\_OSPEEDR register bits whatever the I/O direction). The GPIOx\_PUPDR register is used to select the pull-up/pull-down whatever the I/O direction.

GPIO port mode register (GPIOx\_MODER) (x = A..I)

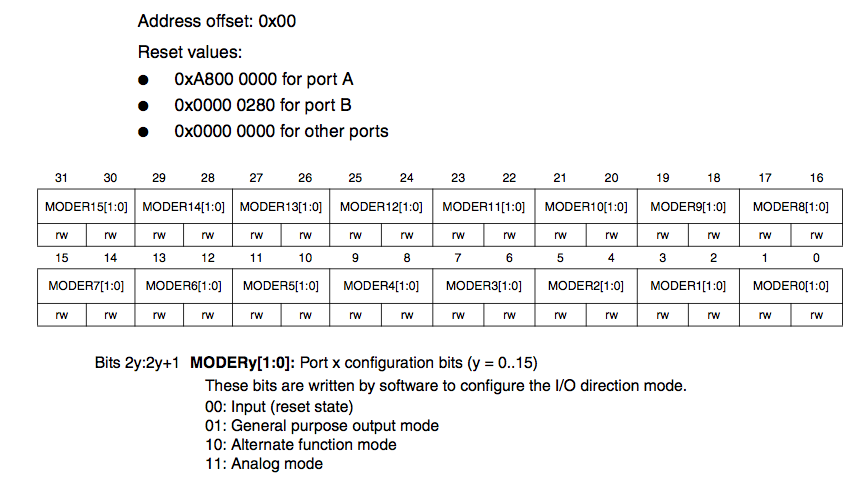


Figure D‑61 GPIO port mode register

GPIO port output type register (GPIOx\_OTYPER) (x = A..I)

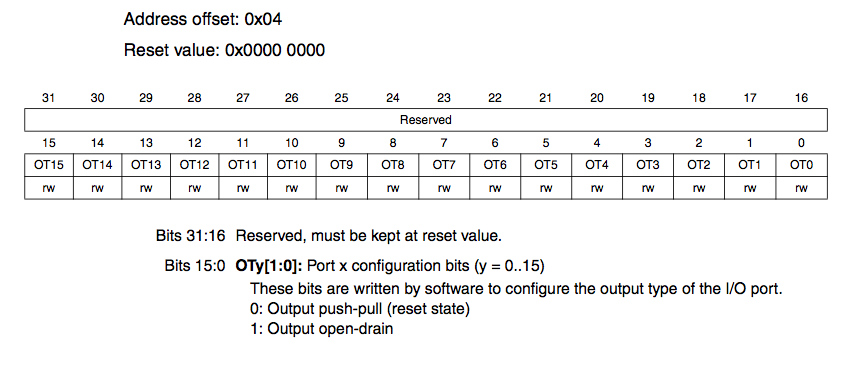


Figure D‑62 GPIO port output type register

GPIO port output speed register (GPIOx\_OSPEEDR)

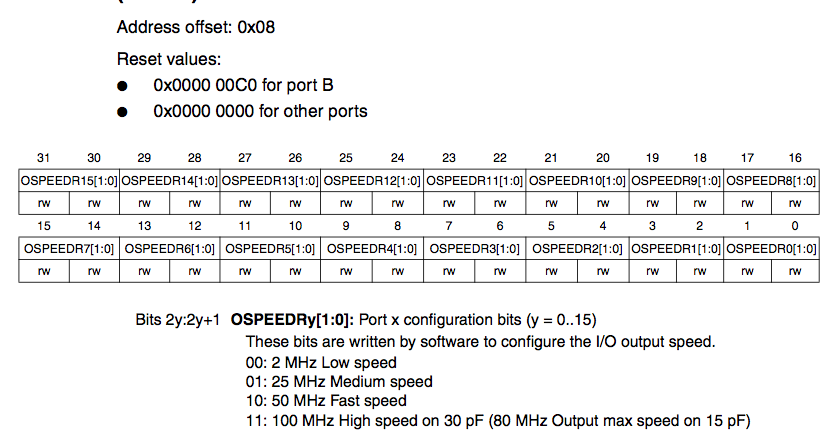


Figure D‑63 GPIO port output speed register

GPIO port pull-up/pull-down register (GPIOx\_PUPDR) (x = A..I)

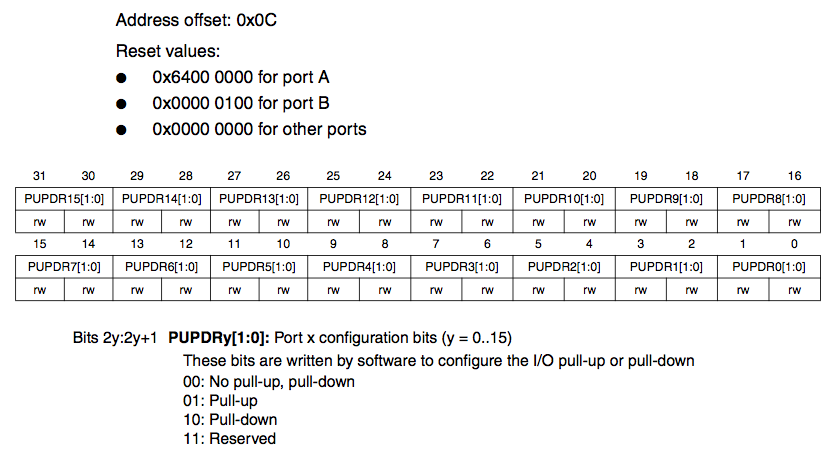


Figure D‑64 GPIO port pull-up/pull-down register

GPIO port input data register (GPIOx\_IDR) (x = A..I)

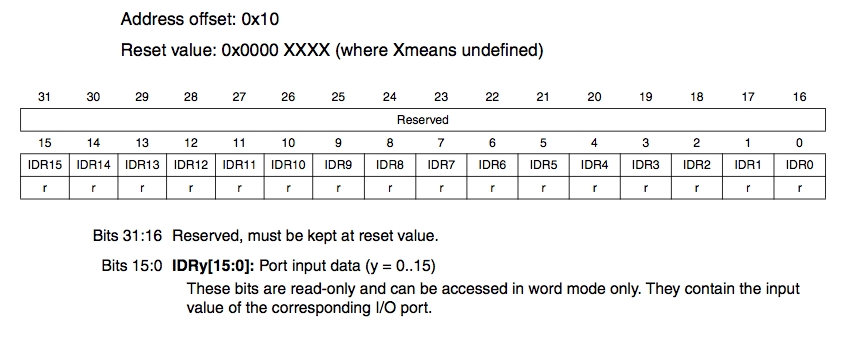


Figure D‑65 GPIO port input data register

#### Interrupts

All ports have external interrupt capability. To use external interrupt lines, the port must be configured in input mode.

The external interrupt/event controller consists of up to 23 edge detectors for generating event/interrupt requests. Each input line can be independently configured to select the type (pulse or pending) and the corresponding trigger event (rising or falling or both). Each line can also masked independently. A pending register maintains the status line of the interrupt requests.

##### Features

The main features of the external interrupt (EXTI) controller are the following:

* Independent trigger and mask on each interrupt/event line G dedicated Status bit for each interrupt line
* Generation of up to 23 software event/interrupt requests
* Detection of external signals with a pulse width lower than the APB2 clock period. Refer to the electrical characteristics section of the STM32F40x and STM32F41x datasheets for details on this parameter.

In this case, we only use Interrupt for USART, include Input (Rx) and Output (Tx).

#### Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART)

##### Usage and Purpose

In this project, the use of USART will be the main communication channel between STM32F4 Discovery module and other modules. The command will be receive through this channel as well as the response from STM32F4 (when required).

##### Baud Rate

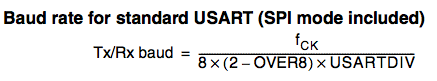
(\*)

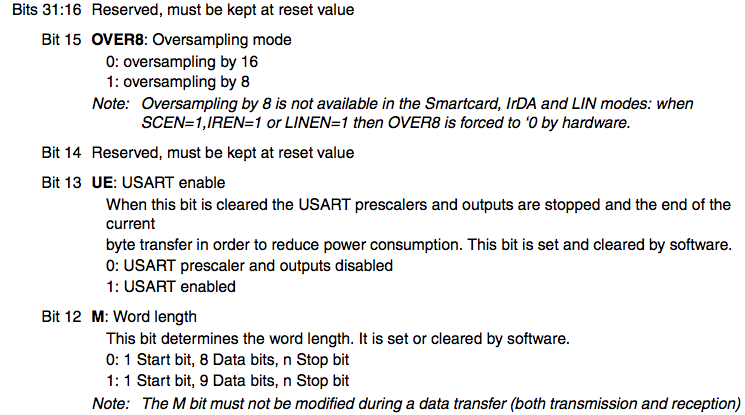
Figure D‑66 Baud Rate Calculation

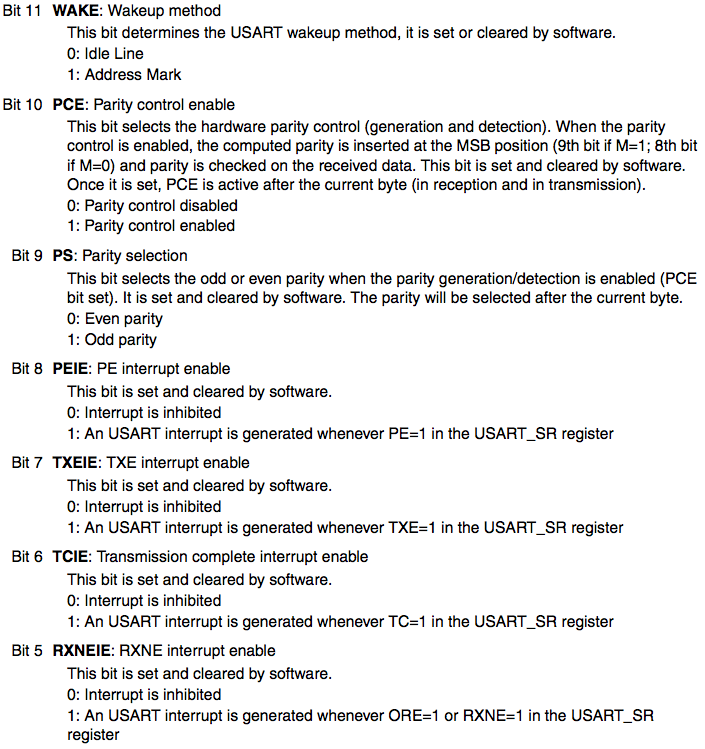
Notes: (\*) Refer to STM32F407VGT6 Datasheet for more details.

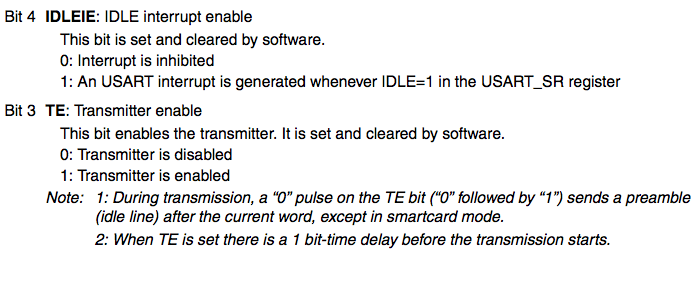
##### Register Description

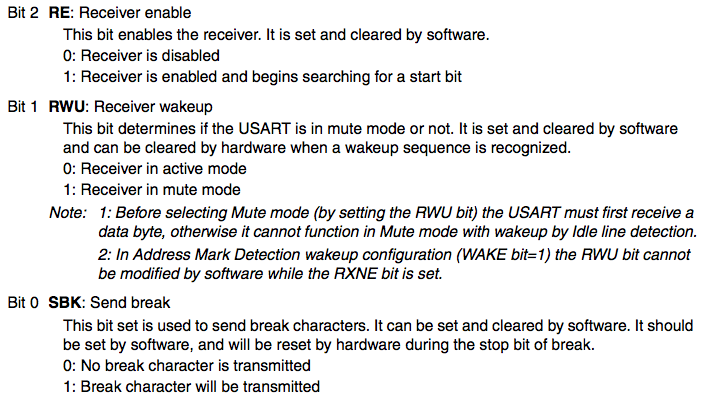


Figure D‑67 STM32F407VGT6 USART Register Description









## ACTIVITY AND CLASS STRUCTURE

### APOD Controller (HMI Application)

#### Activity

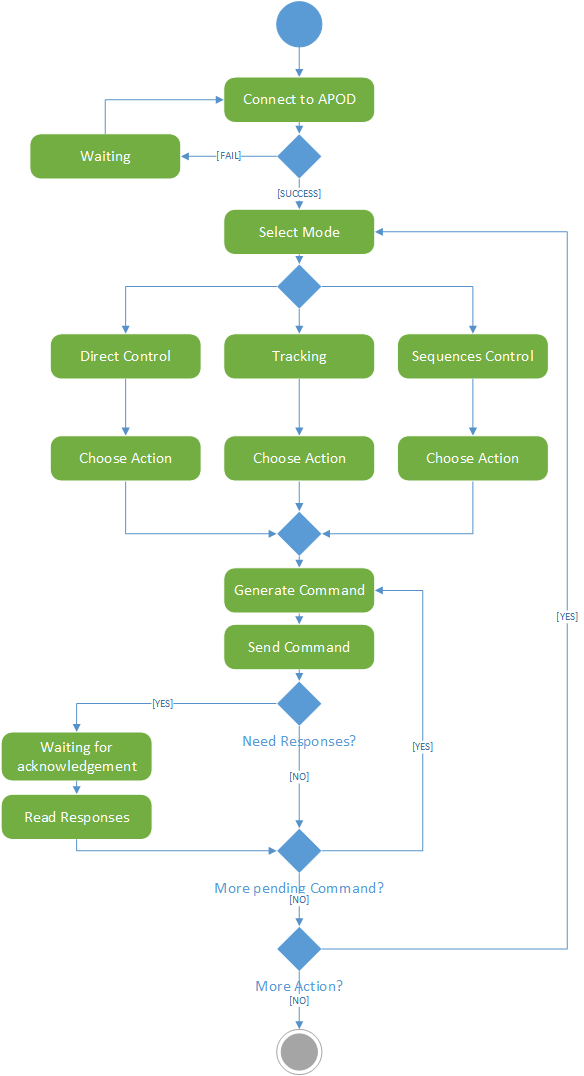


Figure D‑68 APOD Controller Application Activity

#### Class Diagram

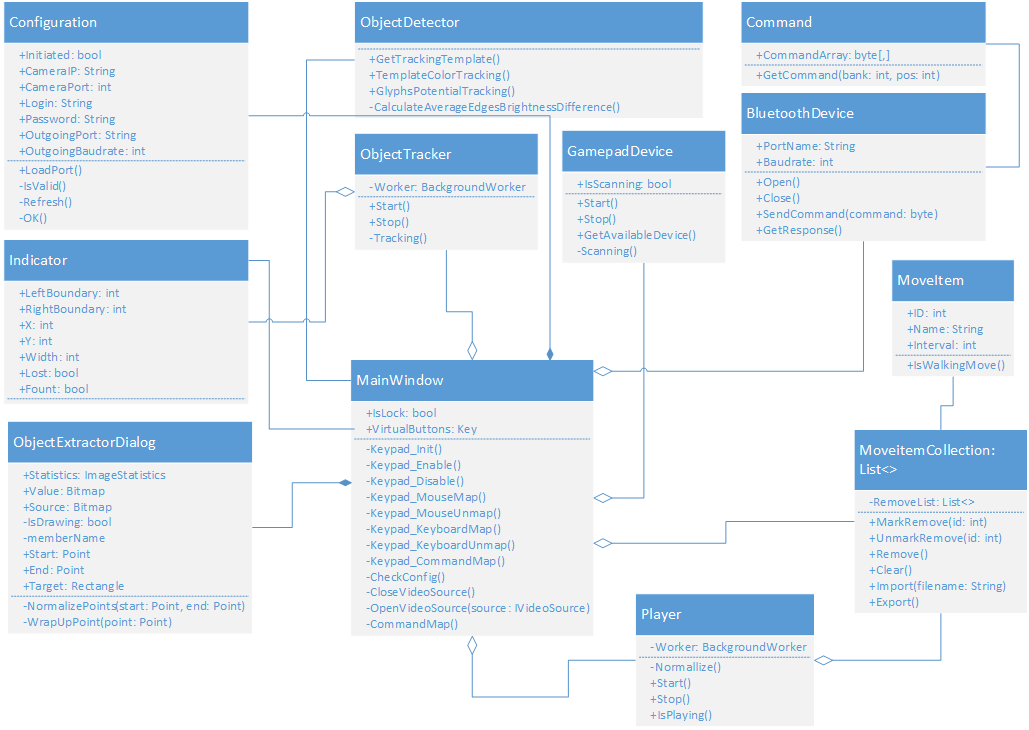


Figure D‑69 Class diagram

#### External Library

**AForge.NET** (Kirillov, aforgenet.com, n.d.)framework including namespaces:

* Aforge
* Aforge.Controls
* Aforge.Imaging
* Aforge.Math
* Aforge.Video
* Aforge.Video.DirectShow
* Aforge.Vision

MahApps.Metro UI toolkit (Jenkins, n.d.):

* MahApps.Metro

### STM32F4 Discovery (Central processing board)

#### Activity



Figure D‑70 STM32F4 Activity

#### Command Sheet

### SSC32 Board (Servos control and Sensor reader)

#### Activity

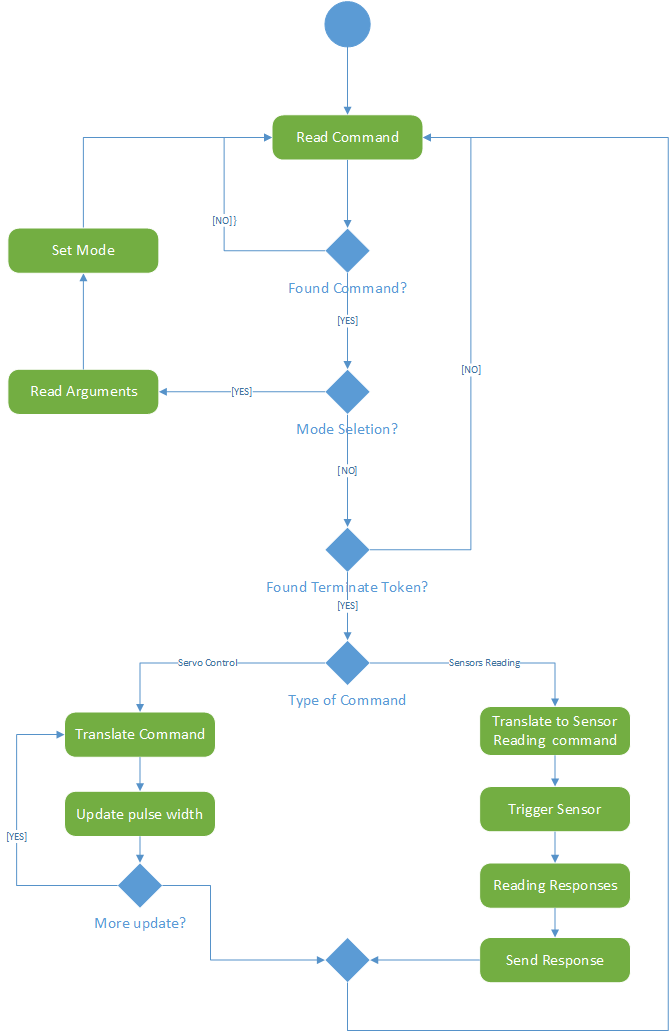


Figure D‑71 SSC32 Activity

#### Command Sheet

|  |  |  |  |
| --- | --- | --- | --- |
| Command | Argument 1 | Argument 2 | Description |
| *[1 byte]* | *[1 byte]* | *[1 byte]* |  |
| ‘I’ | [interval length] | N/A | Selecting speed for servo move.  The [*interval length*] argument take value from 0 to 31 describe the waiting time (in ms) of each 20 µS pulse of servo. (\*) |
| ‘D’ | N/A | N/A | Get the distance sensor value, return 2-bytes response contain the sonar travel time. |
| ‘V’ | [channel] | N/A | Get the ADC value from selected channel. 2 available channel: ‘C’ and ‘D’ |
| ‘S’ | [mode] | N/A | Start-up command for all servos:  - [*mode*] = 1: Start  - [*mode*] = 2: Stop |
| Note: (\*) Refer to the [Implementation Approaches](#_Servo’s_Speed_Control) section for more information. | | | |

Table D‑9 Mode & Reader command (SSC32)

|  |  |  |  |
| --- | --- | --- | --- |
| Command | Argument 1 | Argument 2 | Description |
| *[1 byte]* | *[1 byte]* | *[2 bytes]* |  |
| ‘#’ | [channel] | [pulse width] | Main servo command. Rotate servo at [*channel*] to the [*pulse width*] position.  - [*channel*]: select servo channel  - [*pulse width*]: pulse width value corresponding for the servo position |
| ‘T’ | [mode] | N/A | Terminate command that allow servos to start moving to their new position set up by ‘#’ command.  - [mode] = 0: move with maximum speed  - [mode] = 1: move with limited speed. (\*) |
| Note: (\*) Refer to the [Implementation Approaches](#_Servo’s_Speed_Control) section for more information. | | | |

Table D‑10 Servo Control command (SSC32)

## SEQUENCE DIAGRAM

### Embedded Controller

#### Forward

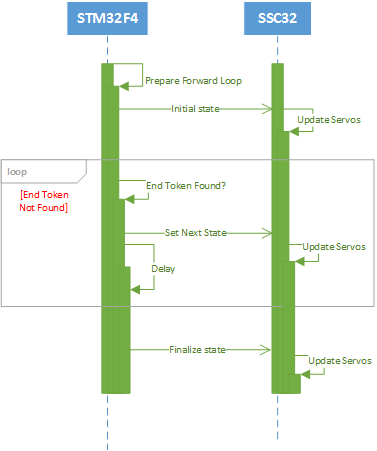


Figure D‑72 Embbed Forward sequence

#### Backward

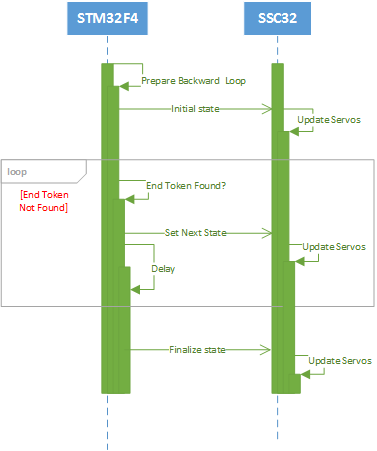


Figure D‑73 Embbed Backward sequence

#### Turn Left

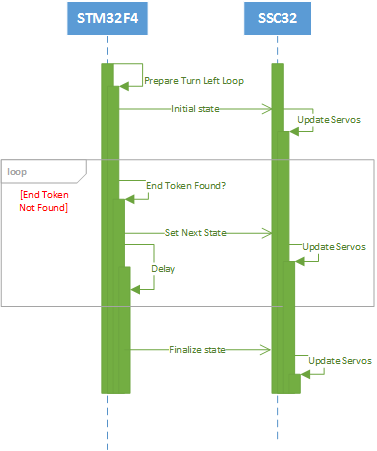


Figure D‑74 Embbed Turn left sequence

#### Turn Right

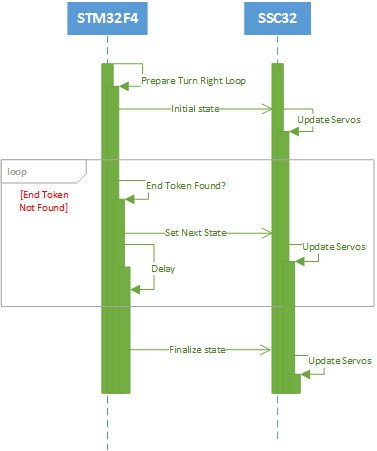


Figure D‑75 Embbed Turn Right sequence

### Use-cases Sequences

#### Sequences Control

##### Add State



Figure D‑76 Add State Use case sequence

##### Remove States

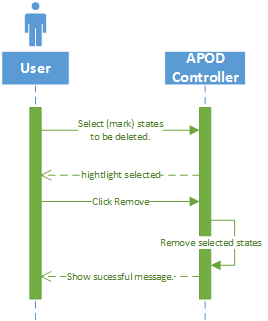


Figure D‑77 Remove States Use case sequence

##### Import States

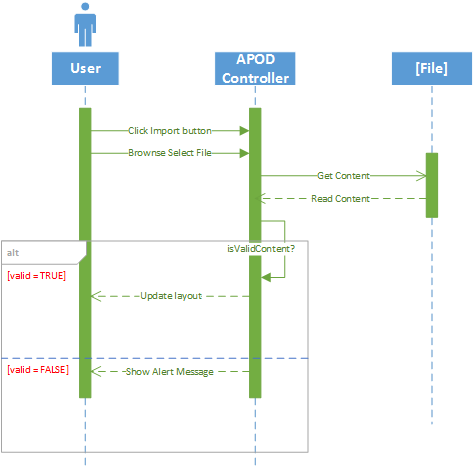


Figure D‑78 Import States Use case sequence

##### Export States

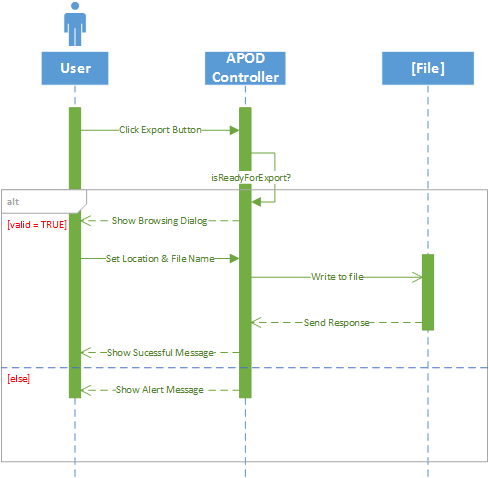


Figure D‑79 Export States Use case sequence

#### Direct Control

##### Mode Selection

###### Normal Input

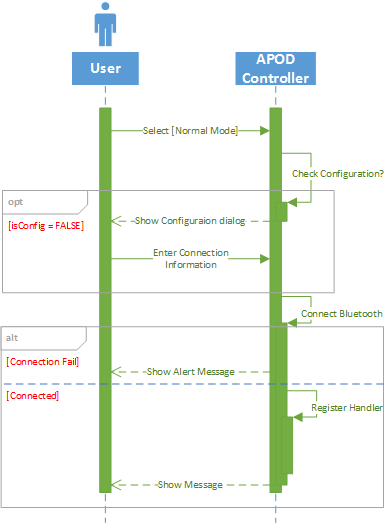


Figure D‑80 Normal Input Use case sequence

###### Gamepad Input

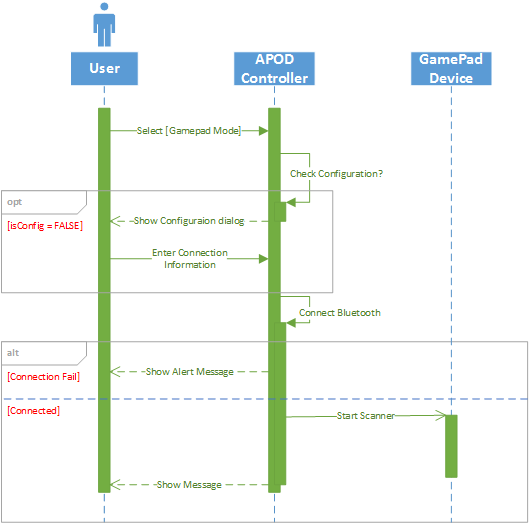


Figure D‑81 Gamepad Input Use case sequence

##### Movement Control

###### Forward

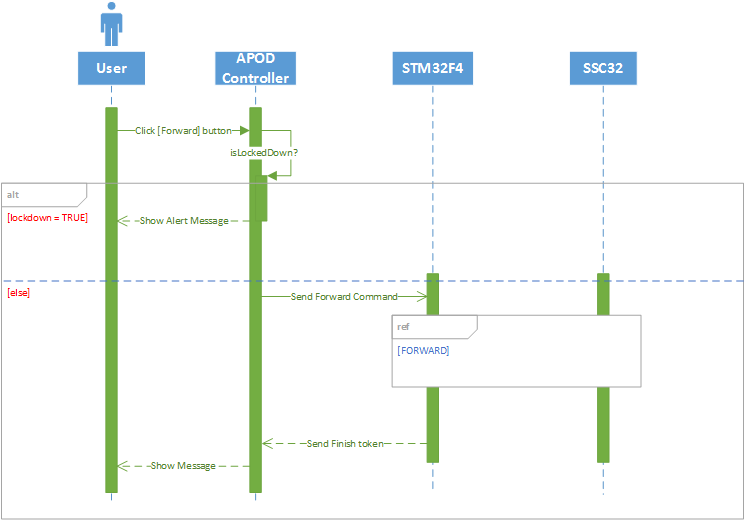


Figure D‑82 Move Forward Use case sequence

###### Backward

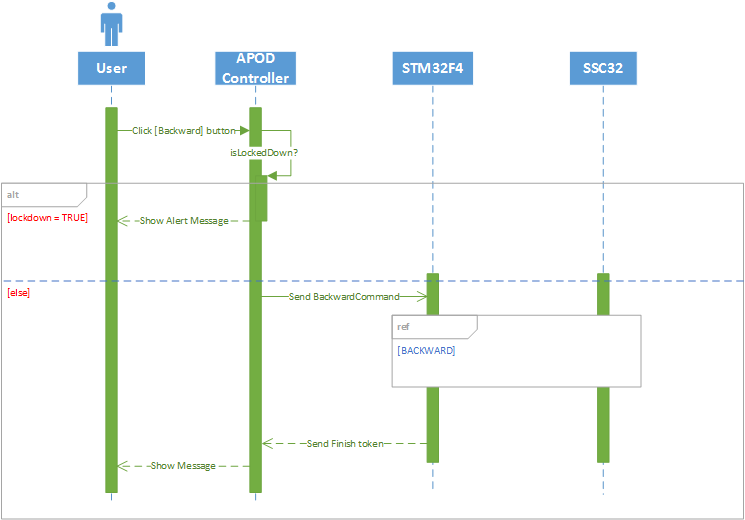


Figure D‑83 Move Backward Use case sequence

###### Turn Left

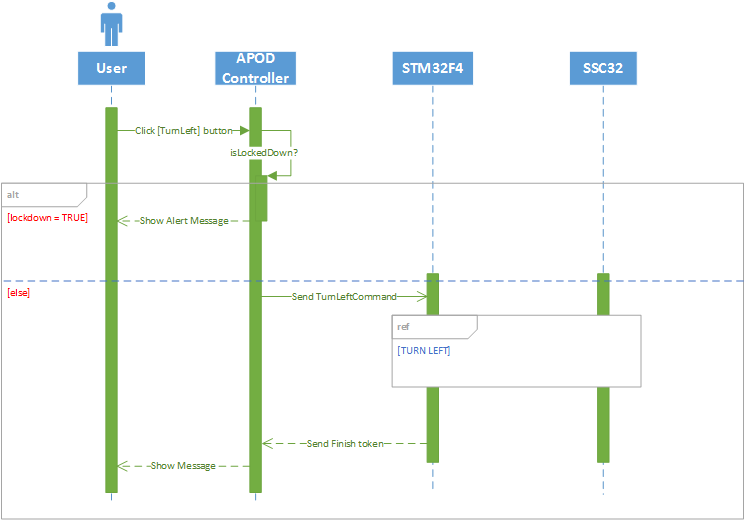


Figure D‑84 Turn Left Use case sequence

###### Turn Right

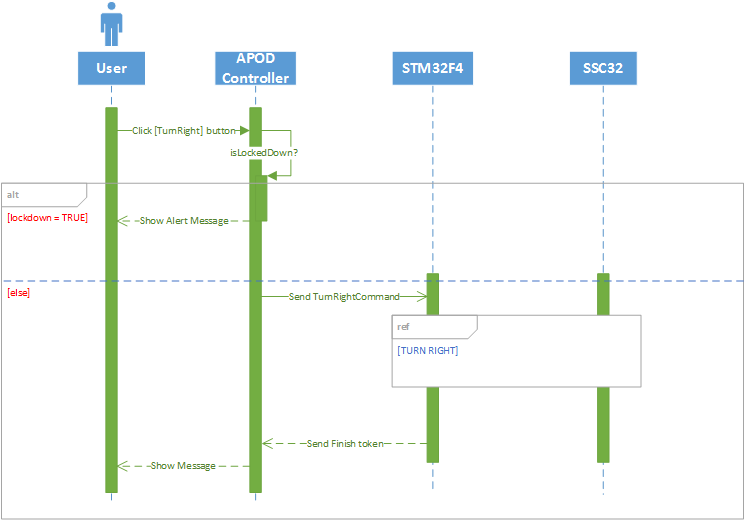


Figure D‑85 Turn Right Use case sequence

###### Body Lift

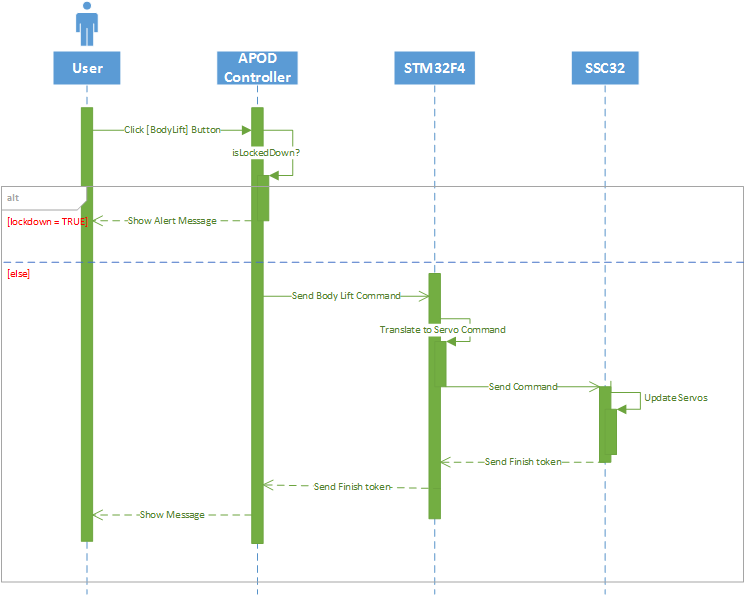


Figure D‑86 Body Lift Use case sequence

###### Body Drop

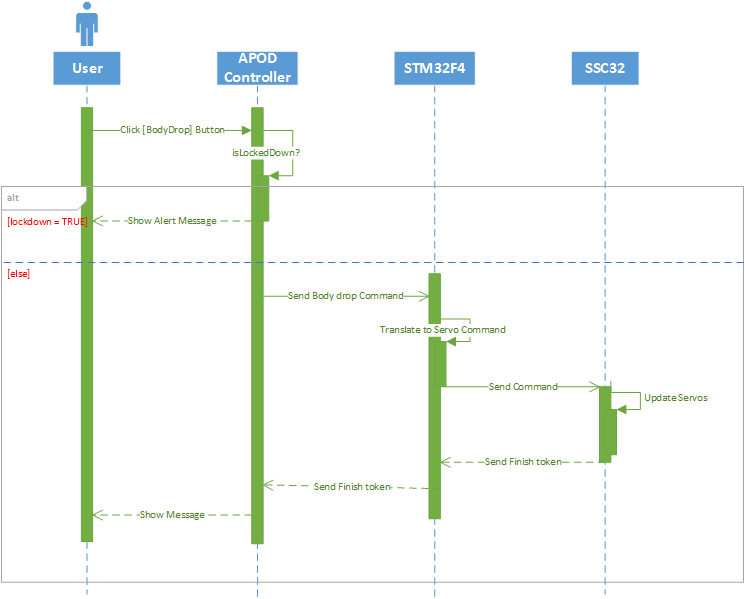


Figure D‑87 Body Drop Use case sequence

###### Squeeze Left

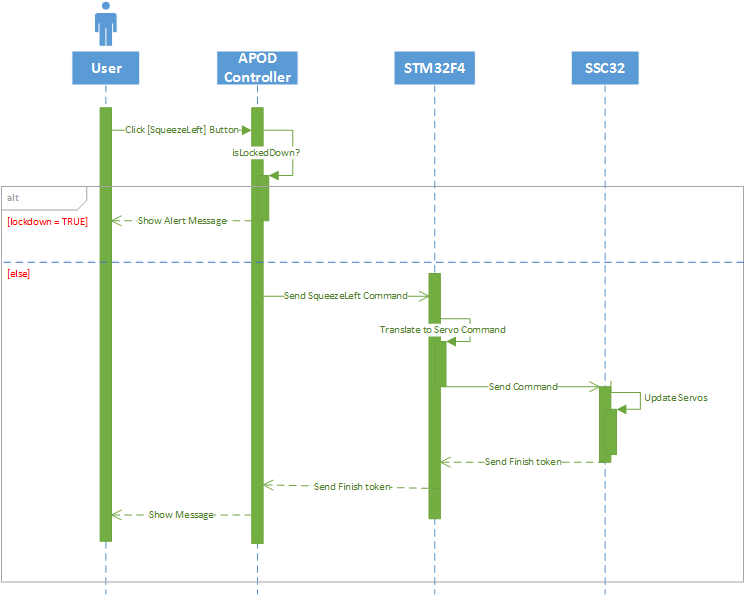


Figure D‑88 Squeeze Left Use case sequence

###### Squeeze Right

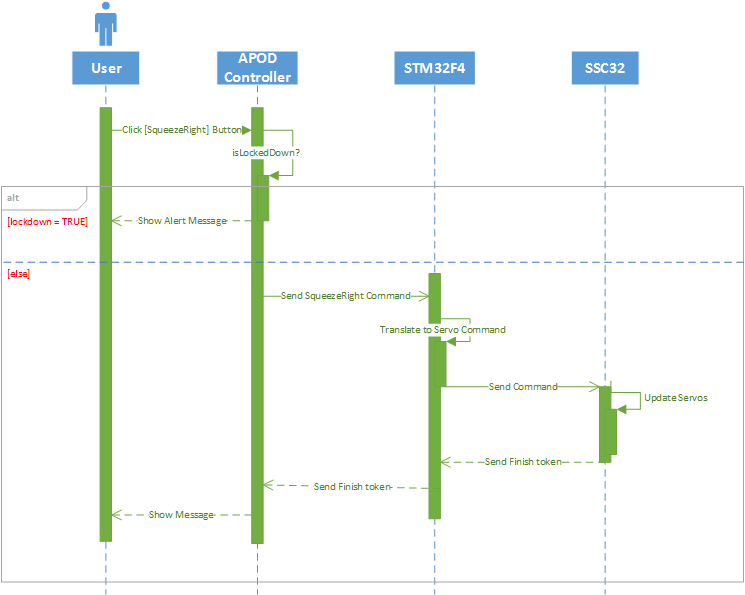


Figure D‑89 Squeeze Right Use case sequence

###### Manual Grip/Release

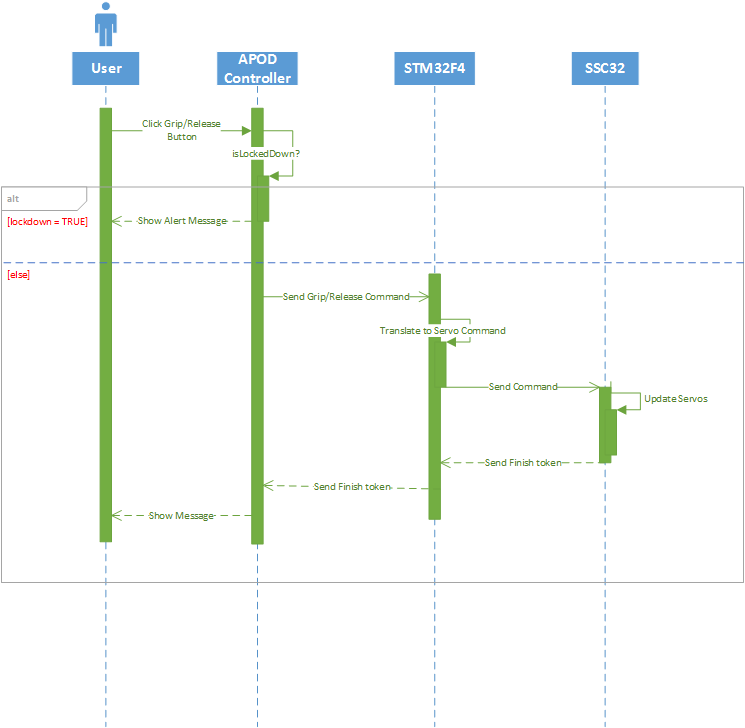


Figure D‑90 Manual Grip/Release Use case sequence

#### Object Tracking

##### Auto Grip

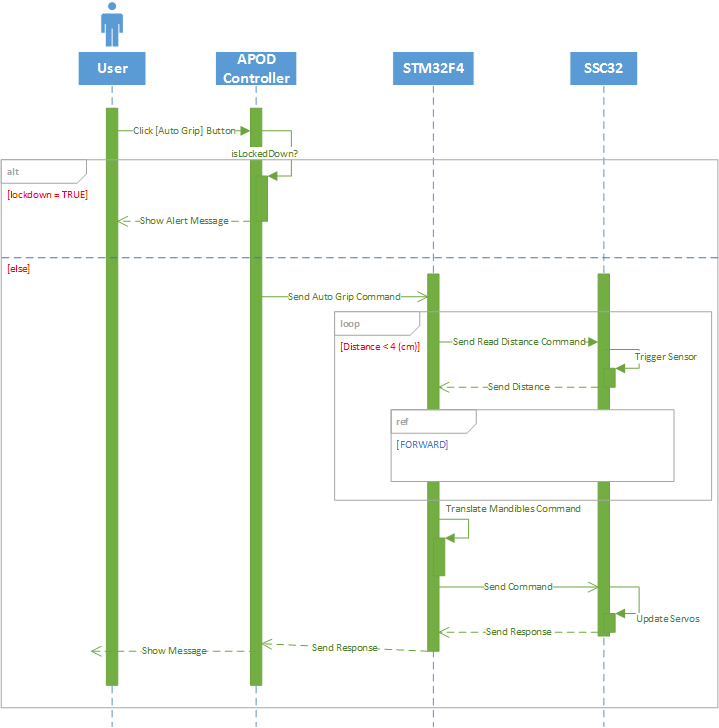


Figure D‑91 Auto Grip Use case sequence

##### Choose target



Figure D‑92 Choose Target Use case sequence

##### Change Target

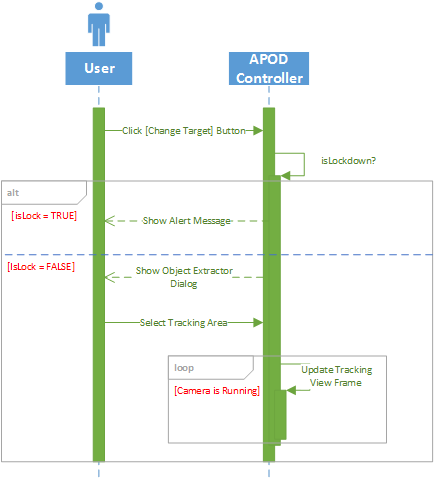


Figure D‑93 Change Target Use case sequence

##### Lock Target

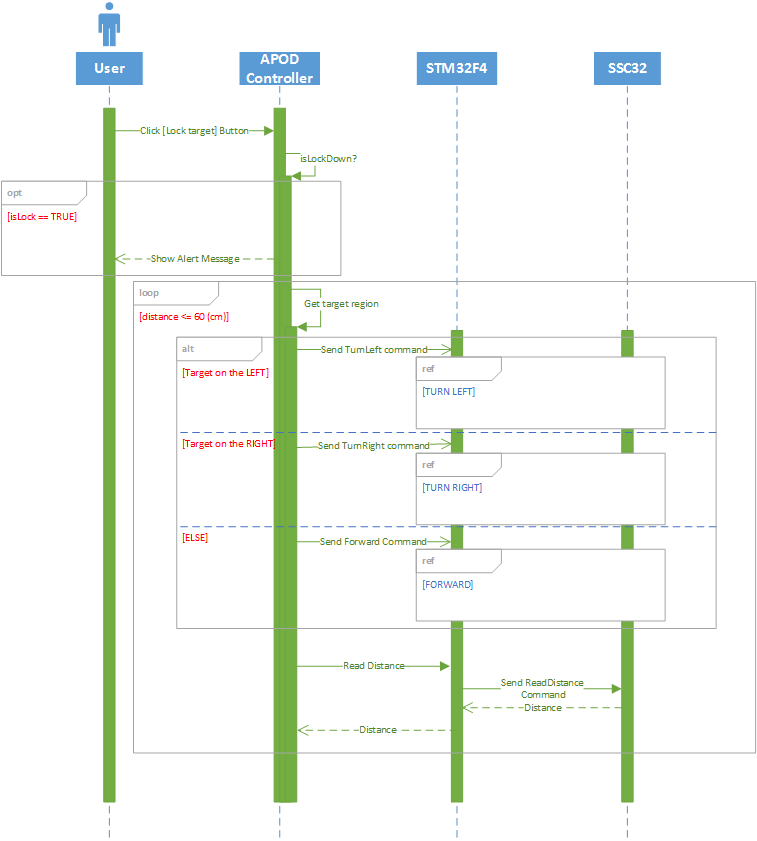


Figure D‑94 Lock Target Use case sequence

## USER INTERFACE DESIGN

### Live Control

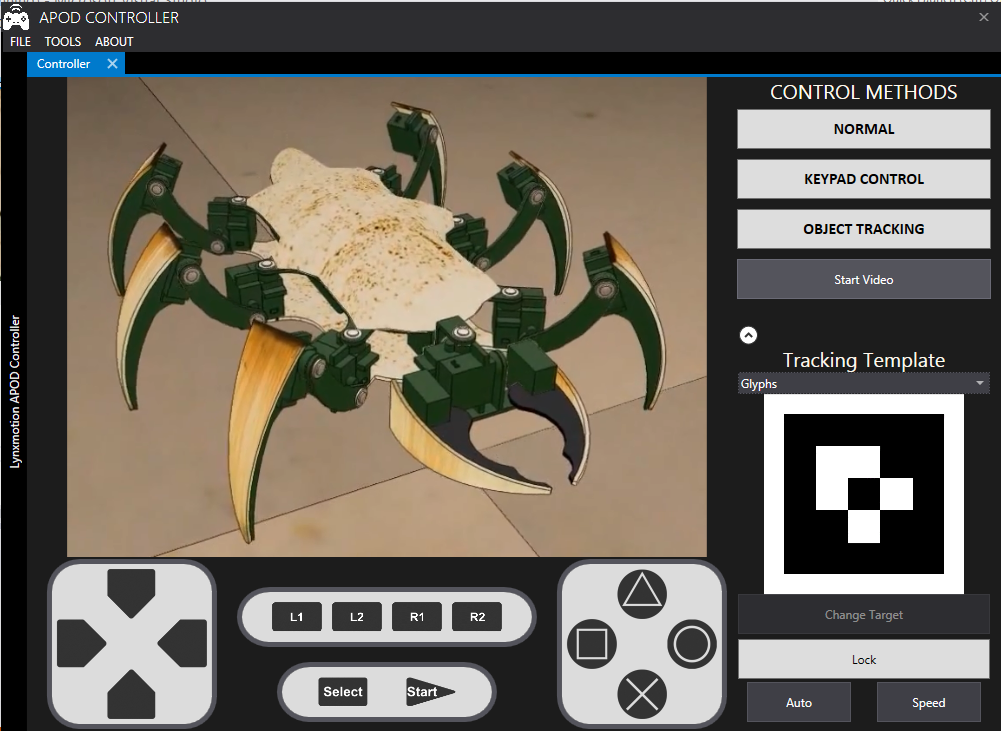


Figure D‑95 Live Control Interface Design

|  |  |  |
| --- | --- | --- |
| Item NO. | Component Name | Description |
| 0 | Menu | Select Features |
| 1 | VideoSourcePlayer | Display Camera IP ‘s View |
| 2 | Checkbox (Button) | Select Normal input mode |
| 3 | Checkbox (Button) | Select Gamepad input mode |
| 4 | Checkbox (Button) | Select object tracking mode |
| 5 | Button | Start streaming video from Camera |
| 6 | Image | Display tracking template |
| 7 | Button | Change Tracking template |
| 8 | Checkbox (Button) | Lock target – Start tracking |
| 9 | Button | Switch speed |
| 10 | Button | Auto grip |
| 11 | Keypad | Action selection |
| 12 | Keypad | Action bank selection |
| 13 | Keypad | Set mode/function |
| 14 | Keypad | Navigation Selection |
|  |  |  |

Table D‑11 Live Control Interface Design Description

### Sequence Player

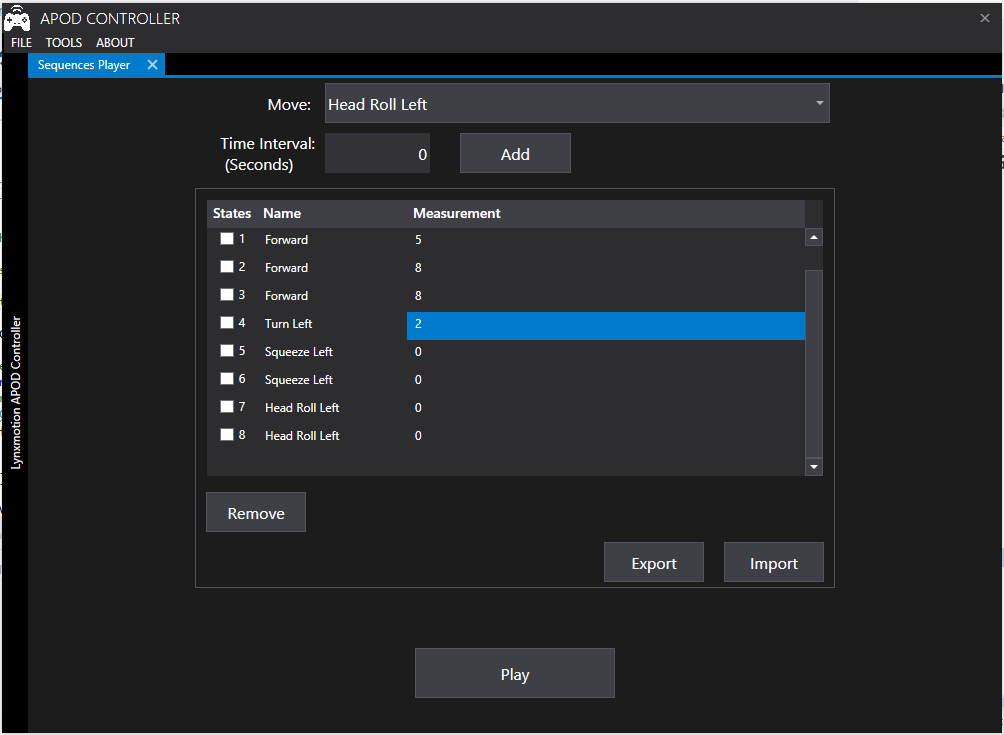


Figure D‑96 Sequence Player Interface Design

|  |  |  |
| --- | --- | --- |
| Item NO. | Component Name | Description |
| 0 | Combo Box | Select action to add |
| 1 | Textbox | Movement interval |
| 2 | Button | Add action to sequence |
| 3 | Data Grid | Display current sequence content |
| 4 | Button | Remove selected states from sequence |
| 5 | Button | Export current sequence to file |
| 6 | Button | Import sequence from file |
| 7 | Button | Start playing the sequence |
|  |  |  |
|  |  |  |
|  |  |  |

Table D‑12 Sequence Player Interface Design Description

### Configuration

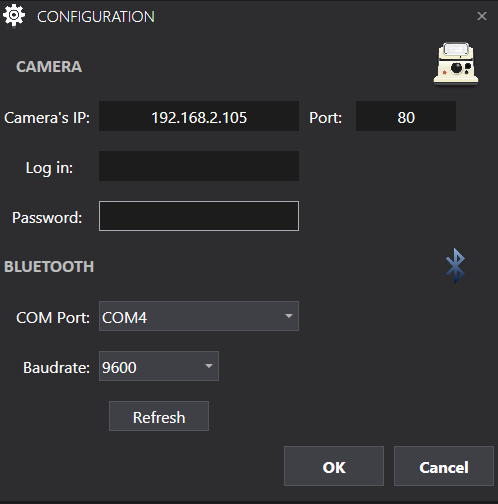


Figure D‑97 Configuration Interface Design

|  |  |  |
| --- | --- | --- |
| Item NO. | Component Name | Description |
| 0 | Textbox | Enter IP of camera |
| 1 | Textbox | Camera port to get stream |
| 2 | Textbox | Username of camera IP |
| 3 | Password Box | Password of camera IP |
| 4 | Combo Box | All available COM ports |
| 5 | Combo Box | Bluetooth baud rate selection |
| 6 | Button | Refresh list of Ports |
| 7 | Button | Accept information |
| 8 | Button | Cancel action |
|  |  |  |
|  |  |  |

Table D‑13 Configuration Interface Design Description

### Object Extractor

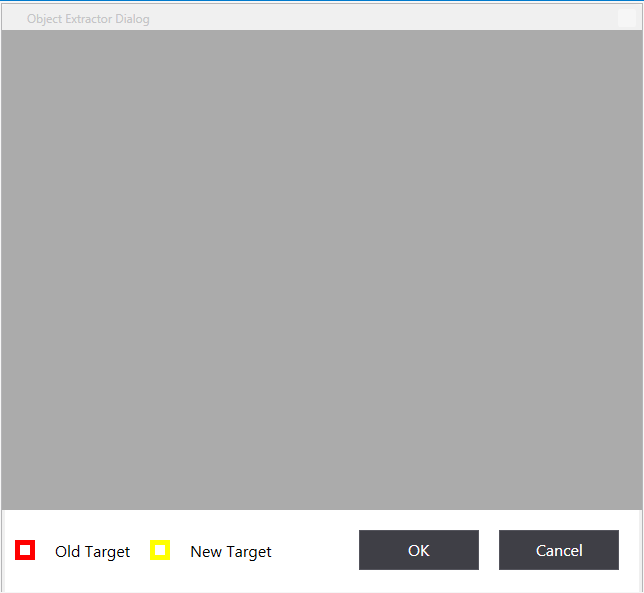


Figure D‑98 Object Extractor Interface Design

|  |  |  |
| --- | --- | --- |
| Item NO. | Component Name | Description |
| 0 | Image | Capture frame for template selection |
| 1 | Button | Accept new template |
| 2 | Button | Cancel action |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Table D‑14 Object Extractor Interface Design Description

# SOFTWARE TEST DOCUMENTATION (STD)

## Introduction

### Overview

The purpose of this project is to build a physical hexapod robotic with named APOD. He has the ability to get balance on his legs, move like a real insect, can record by the camera and auto follow and grip item which is chosen. In the other hand, the robot communicate with Pc through Bluetooth technology, and be Controlled by PS2 controller

### 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

We have two main kind of test phases: Function test (Integration test) and System test.

For the Integration test, we get the functional test to show up the working of each function is right or wrong, so we have to focus on the result of the function in difference cases.

For the System test, all activities of system are needed to check after setup, for example what does he(APOD) do when we start up and observe him if something happen . We create test case to check each activity of him from turn on to turn off and log what he do when we act on. Finally, we record the result

### Features to be tested

Functional Testing:

* Integration Testing: Test all function of robot, C# application
* System Testing: Test all function for APOD – C# application

UI Testing:

* Test UI of C# application

### Features not to be tested

Embedded function on STM32F4 and SSC32

### Test Environment

In addition to the application and any other user specified software, the following list of software should be considered a minimum:

* NET Framework 4.0
* Visual Studio 2012

Hardware tools:

* A PC/laptop that has Bluetooth device and can connect Wireless.
* Wireless Router

Operating System: Window 8

## Test Cases

### Functional Test

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| No | Function | | Pre-condition | Step of Testing | Expected result | Create by | Result |
| Medium Function | Small Function |
| 1 | Start robot | Turn robot on | Robot is turned off | Switch the trigger on robot | Notice led on two board is on | QuanNM | P |
| 2 | Turn robot off | Robot is turned on | Switch the trigger on robot | Notice led on two board is off | QuanNM | P |
| 3 | Connect Bluetooth device | Connect PC/Laptop that has Bluetooth | Robot is turned on | 1.Turn ON Bluetooth board on two device  2.Pair device and enter pin ’1234’ on PC | 1. If two devices can connect to each other, the light of Bluetooth board on Robot is continuous light. If two devices can't connect to each other, the light of Bluetooth board on Robot is flashing light. | QuanNM |  |
| 4 | Operate in “Gamepad Controller” mode | Start robot | 1. Robot and PC/ Laptop connected with each other  2. Environment:  - Using Serial Port communication in win 7/8 | Click “Start” button on PS2 controller | 1.Robot start  2. All servos are set with 1500 pulse width | QuanNM | P |
| 5 | Stop robot | 1. Robot and PC/ Laptop connected with each other  2. Environment:  - Using Serial Port communication in win 7/8  3. Robot is on | Click [Select] button on PS2 controller | 1. Robot is stop.  2. All servos are released. | QuanNM | P |
| 6 | Robot move forward | 1. Robot and PC/ Laptop connected with each other  2. Environment:  - Using Serial Port communication in win 7/8  3. Robot is on | Click [Up Dpad] button on PS2 controller | Robot move forward 1 time | QuanNM | P |
| 7 | Robot move backward | 1. Robot and PC/ Laptop connected with each other  2. Environment:  - Using Serial Port communication in win 7/8  3. Robot is on | Click [Down Dpad] button on PS2 controller | Robot move backward 1 time | QuanNM | P |
| 8 | Turn robot left | 1. Robot and PC/ Laptop connected with each other  2. Environment:  - Using Serial Port communication in win 7/8  3. Robot is on | Click [Left Dpad] button on PS2 controller | Robot turn left 1 time | QuanNM | P |
| 9 | Turn robot right | 1. Robot and PC/ Laptop connected with each other  2. Environment:  - Using Serial Port communication in win 7/8  3. Robot is on | Click [Right Dpad] button on PS2 controller | Robot turn right 1 time | QuanNM | P |
| 10 | Drop robot’s body | 1. Robot and PC/ Laptop connected with each other  2. Environment:  - Using Serial Port communication in win 7/8  3. Robot is on | Click Combo button [Trigger L1 ] + [Trigger R1] on PS2 controller | Robot drop the height 1 time | QuanNM | P |
| 11 | Lift Robot’s body | 1. Robot and PC/ Laptop connected with each other  2. Environment:  - Using Serial Port communication in win 7/8  3. Robot is on | Click Combo button [Trigger L1 ] + [Trigger R2] on PS2 controller | Robot lift the height 1 time | QuanNM | P |
| 12 | Stretch robot forward | 1. Robot and PC/ Laptop connected with each other  2. Environment:  - Using Serial Port communication in win 7/8  3. Robot is on | Click Combo button [Trigger L1 ] + [Up Dpad] on PS2 controller | Robot stretch forward 1 time | QuanNM | P |
| 13 | Stretch robot backward | 1. Robot and PC/ Laptop connected with each other  2. Environment:  - Using Serial Port communication in win 7/8  3. Robot is on | Click Combo button [Trigger L1 ] + [Down Dpad] on PS2 controller | Robot stretch backward 1 time | QuanNM | P |
| 14 | Stretch robot left | 1. Robot and PC/ Laptop connected with each other  2. Environment:  - Using Serial Port communication in win 7/8  3. Robot is on | Click Combo button [Trigger L1 ] + [Left Dpad] on PS2 controller | Robot stretch to left 1 time | QuanNM | P |
| 15 | Stretch robot right | 1. Robot and PC/ Laptop connected with each other  2. Environment:  - Using Serial Port communication in win 7/8  3. Robot is on | Click Combo button [Trigger L1 ] + [Right Dpad] on PS2 controller | Robot stretch to right 1 time | QuanNM | P |
| 16 | Rotate Robot’s neck to left | 1. Robot and PC/ Laptop connected with each other  2. Environment:  - Using Serial Port communication in win 7/8  3. Robot is on | Click [Square] on PS2 controller | Robot’s neck rotate to left 1 time | QuanNM | P |
| 17 | Rotate Robot’s neck to right | 1. Robot and PC/ Laptop connected with each other  2. Environment:  - Using Serial Port communication in win 7/8  3. Robot is on | Click [Circle] on PS2 controller | Robot’s neck rotate to right 1 time | QuanNM | P |
| 18 | Turn robot’s neck up | 1. Robot and PC/ Laptop connected with each other  2. Environment:  - Using Serial Port communication in win 7/8  3. Robot is on | Click Combo button [Trigger L2] + [Up Dpad] on PS2 controller | Robot’s neck turn up 1 time | QuanNM | P |
| 19 | Turn robot’s neck down | 1. Robot and PC/ Laptop connected with each other  2. Environment:  - Using Serial Port communication in win 7/8  3. Robot is on | Click combo button [Trigger L2] + [Down Dpad] on PS2 controller | Robot’s neck turn down 1 time | QuanNM | P |
| 20 | Turn robot’s neck left | 1. Robot and PC/ Laptop connected with each other  2. Environment:  - Using Serial Port communication in win 7/8  3. Robot is on | Click combo button [Trigger L2] + [Left Dpad] on PS2 controller | Robot’s neck turn left 1 time | QuanNM | P |
| 21 | Grip robot’s ripper | 1. Robot and PC/ Laptop connected with each other  2. Environment:  - Using Serial Port communication in win 7/8  3. Robot is on | Click [Triangle] button on PS2 controller | Robot’s Gripper grip 1 time | QuanNM | P |
| 22 | Release robot’s ripper | 1. Robot and PC/ Laptop connected with each other  2. Environment:  - Using Serial Port communication in win 7/8  3. Robot is on | Click combo button [Trigger L2] + [Left Dpad] on PS2 controller | Robot’s Gripper release 1 time | QuanNM | p |
| 23 | Operate in “Normal” mode  (using keyboard) | Start robot | 1. Robot and PC/ Laptop connected with each other  2. Environment:  - Using Serial Port communication in win 7/8 | Click “Start” button on Main PC Interface | 1.Robot start  2. All servos are set with 1500 pulse width | QuanNM | P |
| 24 | Stop robot | 1. Robot and PC/ Laptop connected with each other  2. Environment:  - Using Serial Port communication in win 7/8  3. Robot is on | Click [Select] button on Main PC Interface | 1. Robot is stop.  2. All servos are released. | QuanNM | P |
| 25 | Robot move forward | 1. Robot and PC/ Laptop connected with each other  2. Environment:  - Using Serial Port communication in win 7/8  3. Robot is on | Click [Up Dpad] button on Main PC Interface | Robot move forward 1 time | QuanNM | P |
| 26 | Robot move backward | 1. Robot and PC/ Laptop connected with each other  2. Environment:  - Using Serial Port communication in win 7/8  3. Robot is on | Click [Down Dpad] button Main PC Interface | Robot move backward 1 time | QuanNM | P |
| 27 | Turn robot left | 1. Robot and PC/ Laptop connected with each other  2. Environment:  - Using Serial Port communication in win 7/8  3. Robot is on | Click [Left Dpad] button Main PC Interface | Robot turn left 1 time | QuanNM | P |
| 28 | Turn robot right | 1. Robot and PC/ Laptop connected with each other  2. Environment:  - Using Serial Port communication in win 7/8  3. Robot is turned on | Click [Right Dpad] button Main PC Interface | Robot turn right 1 time | QuanNM | p |
| 29 | Drop robot’s body | 1. Robot and PC/ Laptop connected with each other  2. Environment:  - Using Serial Port communication in win 7/8  3. Robot is on | Click Combo button [Trigger L1 ] + [Trigger R1] Main PC Interface | Robot drop the height 1 time | QuanNM | P |
| 30 | Lift Robot’s body | 1. Robot and PC/ Laptop connected with each other  2. Environment:  - Using Serial Port communication in win 7/8  3. Robot is on | Click Combo button [Trigger L1 ] + [Trigger R2] Main PC Interface | Robot lift the height 1 time | QuanNM | P |
| 31 | Stretch robot forward | 1. Robot and PC/ Laptop connected with each other  2. Environment:  - Using Serial Port communication in win 7/8  3. Robot is on | Click Combo button [Trigger L1 ] + [Up Dpad] Main PC Interface | Robot stretch forward 1 time | QuanNM | P |
| 32 | Stretch robot backward | 1. Robot and PC/ Laptop connected with each other  2. Environment:  - Using Serial Port communication in win 7/8  3. Robot is on | Click Combo button [Trigger L1 ] + [Down Dpad] Main PC Interface | Robot stretch backward 1 time | QuanNM | P |
| 33 | Stretch robot left | 1. Robot and PC/ Laptop connected with each other  2. Environment:  - Using Serial Port communication in win 7/8  3. Robot is on | Click Combo button [Trigger L1 ] + [Left Dpad] on Main PC Interface | Robot stretch to left 1 time | QuanNM | P |
| 34 | Stretch robot right | 1. Robot and PC/ Laptop connected with each other  2. Environment:  - Using Serial Port communication in win 7/8  3. Robot is on | Click Combo button [Trigger L1 ] + [Right Dpad] on Main PC Interface | Robot stretch to right 1 time | QuanNM | P |
| 35 | Rotate Robot’s neck to left | 1. Robot and PC/ Laptop connected with each other  2. Environment:  - Using Serial Port communication in win 7/8  3. Robot is on | Click [Square] on Main PC Interface | Robot’s neck rotate to left 1 time | QuanNM | P |
| 36 | Rotate Robot’s neck to right | 1. Robot and PC/ Laptop connected with each other  2. Environment:  - Using Serial Port communication in win 7/8  3. Robot is on | Click [Circle] on Main PC Interface | Robot’s neck rotate to right 1 time | QuanNM | P |
| 37 |  | Turn robot’s neck up | 1. Robot and PC/ Laptop connected with each other  2. Environment:  - Using Serial Port communication in win 7/8  3. Robot is on | Click Combo button [Trigger L2] + [Up Dpad] on Main PC Interface | Robot’s neck turn up 1 time | QuanNM | P |
| 38 |  | Turn robot’s neck down | 1. Robot and PC/ Laptop connected with each other  2. Environment:  - Using Serial Port communication in win 7/8  3. Robot is on | Click combo button [Trigger L2] + [Down Dpad] on Main PC Interface | Robot’s neck turn down 1 time | QuanNM | P |
| 39 |  | Turn robot’s neck left | 1. Robot and PC/ Laptop connected with each other  2. Environment:  - Using Serial Port communication in win 7/8  3. Robot is on | Click combo button [Trigger L2] + [Left Dpad] on Main PC Interface | Robot’s neck turn left 1 time | QuanNM | P |
| 40 |  | Grip robot’s ripper | 1. Robot and PC/ Laptop connected with each other  2. Environment:  - Using Serial Port communication in win 7/8  3. Robot is on | Click [Triangle] button on Main PC Interface | Robot’s Gripper grip 1 time | QuanNM | P |
| 41 |  | Release robot’s ripper | 1. Robot and PC/ Laptop connected with each other  2. Environment:  - Using Serial Port communication in win 7/8  3. Robot is on | Click combo button [Trigger L2] + [Left Dpad] on Main PC Interface | Robot’s Gripper release 1 time | QuanNM | P |

Table E‑1 Functional Test Cases

### UI Test

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| No | Function | | Pre-condition | Step of Testing | Expected result | Create by | Result |
| Medium Function | Small Function |
| 1 | Normal mode | Connect Bluetooth with normal controller  Connect Bluetooth with normal controller | 1.Application is open.  2.Bluetooth is already to pair  3.Wireless connected | 1. Click “Normal” button.  2.In Configuration, Fill in:  - IP camera:”192.168.2.100”  - Port: “80”  - Admin: “admin”  - Password: NONE  - Com port: COM3 (depend on each computer)  -Baud rate: “9600”  3. Click OK | - Configuration windows is disappear.  -IP camera connected  - Bluetooth pair with robot | QuanNM | P |
| 2 | 1.Application is open.  2.Bluetooth is already to pair  3.Wireless connected | 1. Click “Normal” button.  2.In Configuration, Fill in:  - IP camera: NONE  - Port: NONE  - Admin: “admin”  - Password: NONE  - Com port: NONE(depend on each computer)  -Baud rate: “9600”  3. Click OK | 1.Each blank detail( IP camera, Port, Com port) will noticed by change the color (red)  2. System still not connect | QuanNM | P |
| 3 | 1.Application is open.  2.Bluetooth is already to pair  3.Wireless connected | 1. Click “Normal” button.  2.In Configuration, Fill in:  - IP camera: NONE  - Port: 80  - Admin: “admin”  - Password: NONE  - Com port: COM No.(depend on each computer)  -Baud rate: “9600”  3. Click OK | 1. IP camera is noticed to red color.  2.System still not connect | QuanNM | p |
| 4 | 1.Application is open.  2.Bluetooth is already to pair  3.Wireless connected | 1. Click “Normal” button.  2.In Configuration, Fill in:  - IP camera: “192.168.2.100”  - Port: NONE  - Admin: “admin”  - Password: NONE  - Com port: COM No.(depend on each computer)  -Baud rate: “9600”  3. Click OK | 1. Port is noticed to red color.  2.System still not connect | QuanNM | P |
| 5 | 1.Application is open.  2.Bluetooth is already to pair  3.Wireless connected | 1. Click “Normal” button.  2.In Configuration, Fill in:  - IP camera: “192.168.2.100”  - Port: 80  - Admin: “admin”  - Password: NONE  - Com port: NONE No.(depend on each computer)  -Baud rate: “9600”  3. Click OK | 1. COM Port is noticed to red color.  2.System still not connect | QuanNM | P |
| 6 | 1.Application is open.  2.Bluetooth is already to pair  3.Wireless connected | 1. Click “Normal” button.  2.In Configuration, Fill in:  - IP camera:”192.168.2.100”  - Port: “80”  - Admin: “admin”  - Password: NONE  - Com port: COM3 (depend on each computer)  -Baud rate: “9600”  3. Click OK | Configuration windows is disappear. | QuanNM | p |
| 7 | Connect Bluetooth with normal controller  Connect Bluetooth with Keypad controller | 1.Application is open.  2.Bluetooth is already to pair  3.Wireless connected | 1. Click “Normal” button.  2.In Configuration, Fill in:  - IP camera:”192.168.2.100”  - Port: “80”  - Admin: “admin”  - Password: NONE  - Com port: COM3 (depend on each computer)  -Baud rate: “9600”  3. Click OK | - Configuration windows is disappear.  -IP camera connected  - Bluetooth pair with robot | QuanNM | P |
| 8 | 1.Application is open.  2.Bluetooth is already to pair  3.Wireless connected | 1. Click “Normal” button.  2.In Configuration, Fill in:  - IP camera: NONE  - Port: NONE  - Admin: “admin”  - Password: NONE  - Com port: NONE(depend on each computer)  -Baud rate: “9600”  3. Click OK | 1.Each blank detail( Ip camera, Port, Com port) will noticed by change the color (red)  2. System still not connect | QuanNM | P |
| 9 | 1.Application is open.  2.Bluetooth is already to pair  3.Wireless connected | 1. Click “Normal” button.  2.In Configuration, Fill in:  - IP camera: NONE  - Port: 80  - Admin: “admin”  - Password: NONE  - Com port: COM No.(depend on each computer)  -Baud rate: “9600”  3. Click OK | 1. IP camera is noticed to red color.  2.System still not connect | QuanNM | p |
| 10 | 1.Application is open.  2.Bluetooth is already to pair  3.Wireless connected | 1. Click “Normal” button.  2.In Configuration, Fill in:  - IP camera: “192.168.2.100”  - Port: NONE  - Admin: “admin”  - Password: NONE  - Com port: COM No.(depend on each computer)  -Baud rate: “9600”  3. Click OK | 1. Port is noticed to red color.  2.System still not connect | QuanNM | P |
| 11 | 1.Application is open.  2.Bluetooth is already to pair  3.Wireless connected | 1. Click “Normal” button.  2.In Configuration, Fill in:  - IP camera: “192.168.2.100”  - Port: 80  - Admin: “admin”  - Password: NONE  - Com port: NONE No.(depend on each computer)  -Baud rate: “9600”  3. Click OK | 1. COM Port is noticed to red color.  2.System still not connect | QuanNM | P |
| 12 |  | Start camera | 1.Bluetooth is already connected.  2.Wireless is already connected.  3.Camera is already connected. | Click “Start Video” button | Camera screen appear in camera view frame | QuanNM | P |
| 13 | Object Tracking | Getting item | 1.Bluetooth is already connected.  2.Wireless is already connected.  3.Camera is already connected.  4.Camera screen started | 1. Click “Object tracking” button.  2. Click on item’s color  3. Click Ok | 1. Yellow square focus on chosen item.  2. On the right screen there is a show of item’s color | QuanNM | P |
| 14 | Follow particular Item | 1.Bluetooth is already connected.  2.Wireless is already connected.  3.Camera is already connected.  4.Camera screen started  5. Item is tracked | Click “Auto” button | 1.On APOD:  - Gripper is open widely.  - APOD auto moves to item.  - After he reach the item, then he nips.  2. On Pc:  - Tracking zone is always focus depend on APOD movement | QuanNM | P |
| 15 | Get distance | 1.Bluetooth is already connected.  2.Wireless is already connected.  3.Camera is already connected.  4.Camera screen started  5. Item is tracked | Click “Distance” button | Receive current distance from APOD to item. | QuanNM | P |
| 16 | Sequence Player | Making sequence to perform(1) | 1.Bluetooth is already connected.  2.Wireless is already connected. | 1.Click menu “Tool”, choose “Sequence player” menu.  2. In move: chose [Forward].  3. In Time: Enter [5].  4. click Add.  5.click Play. | APOD move forward in 5 second | QuanNM | P |
| 17 |  | Making sequence to perform(2) | 1.Bluetooth is already connected.  2.Wireless is already connected. | 1.Click menu “Tool”, choose “Sequence player” menu.  2. In move: chose [Forward].  3. In Time: Enter [5].  4. click Add.  5. In move: chose [Turn Left].  6. In Time: Enter [3].  7.click Play. | APOD move forward in 5 seconds, turn left in 3 seconds | QuanNM | P |
| 18 |  | Making sequence to perform(3) | 1.Bluetooth is already connected.  2.Wireless is already connected. | 1.Click menu “Tool”, choose “Sequence player” menu.  2. In move: chose [Forward].  3. In Time: Enter [5].  4. click Add.  5. In move: chose [Turn Left].  6. In Time: Enter [3].  7. In move: chose [Turn Right].  8. In Time: Enter [10].  9.click Play. | APOD move forward in 5 seconds, turn left in 3 seconds, turn right in 10 seconds | QuanNM | P |
| 19 |  | Making sequence to perform(4) | 1.Bluetooth is already connected.  2.Wireless is already connected. | 1.Click menu “Tool”, choose “Sequence player” menu.  2. In move: chose [Backward].  3. In Time: Enter [5].  4. click Add.  5. In move: chose [Head left].  7. In move: chose [toward Front].  8.click Play. | APOD move forward in 5 seconds, Head to left, APOD is toward front | QuanNM | P |
| 20 |  | Making sequence to perform(4) | 1.Bluetooth is already connected.  2.Wireless is already connected. | 1.Click menu “Tool”, choose “Sequence player” menu.  2. In move: chose [Forward].  3. In Time: Enter [-5]. | System prevent to enter “-” | QuanNM | P |
| 21 |  | Export sequence | 1.Bluetooth is already connected.  2.Wireless is already connected. | 1.Click menu “Tool”, choose “Sequence player” menu.  2. In move: chose [Backward].  3. In Time: Enter [5].  4. click Add.  5. In move: chose [Head left].  7. In move: chose [toward Front].  8.click “Export” button.  9.Choose direction.  10. Enter name of file  11. click OK | Sequence file is created with xml type | QuanNM | P |
| 22 |  | Import sequence | 1.Bluetooth is already connected.  2.Wireless is already connected. | 1.Click Import button  2.chose file xml to import.  3.Click OK. | List of sequence in file are listed to perform | QuanNM | P |
| 23 |  | Remove sequence | 1.Bluetooth is already connected.  2.Wireless is already connected. | 1.Click menu “Tool”, choose “Sequence player” menu.  2. In move: chose [Backward].  3. In Time: Enter [5].  4. click Add.  5. In move: chose [Head left].  7. In move: chose [toward Front].  8. stick on [Head Left] combo box  8.click Remove button. | [Head left] is removed in list | QuanNM | p |

Table E‑2 UI Test Cases

# SOFTWARE USER MANUAL (SUM)

## Installation

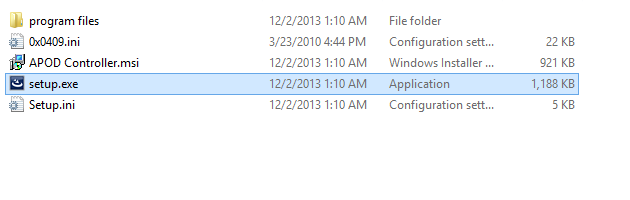


Figure F‑1 Run Setup file in delivered package

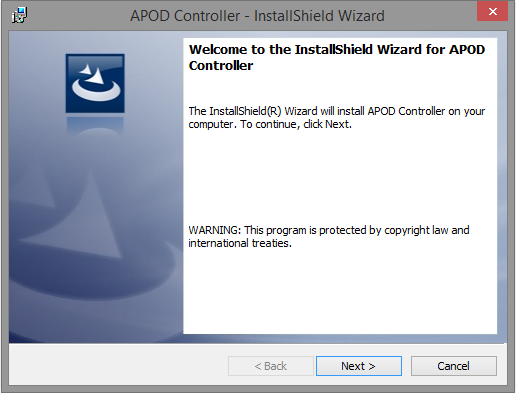


Figure F‑2 Click Next on welcome dialog

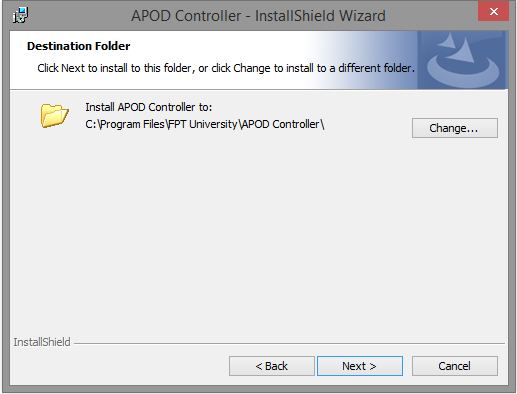


Figure F‑3 Choose Destination

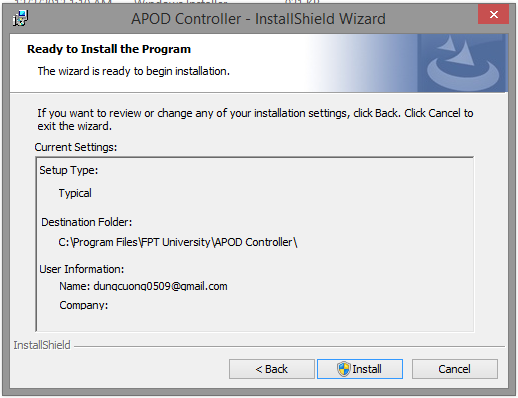


Figure F‑4 Re-Check Setup configuration

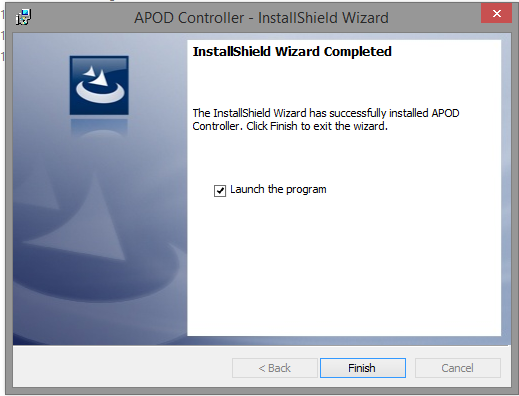


Figure F‑5 Finish Installation

## Usage

# APPENDIX

## Glossary

|  |  |  |
| --- | --- | --- |
| Acronym | Definitions | Notes |
| ADC | Analog - digital converter |  |
| APOD | Name of the hexapod |  |
| AVR | Microcontroller which was developed by Atmel in 1996 |  |
| CMS | Content management system |  |
| COM | Communication port/ serial port |  |
| DLL | Dynamic-link library |  |
| DOF | Degree of freedom |  |
| EEPROM | Electrically erasable programmable read-only memory |  |
| ES | Embedded system |  |
| GND | Ground |  |
| HEXAPOD | A structure with 6 legs |  |
| HMI | Human machine interface |  |
| IS | Information system |  |
| ISP | In-system programming |  |
| IT | Information technology |  |
| MCU | Microcontroller unit |  |
| MISO | Master out slaver in |  |
| MOSI | Master in slaver out |  |
| NiCad | Nickel–cadmium battery |  |
| Ni-MH | Nickel–metal hydride battery |  |
| PIC | Programmable intelligent computer |  |
| PVC | Polyvinyl chloride |  |
| PWM | Pulse width modulation |  |
| RISC | Reduced instruction set computing |  |
| RS232 | Physical serial interface standard |  |
| SCK | Source clock |  |
| SDD | Software design description |  |
| SDM | Systems development method |  |
| SPI | Serial peripheral interface |  |
| SPMP | Software project management plan |  |
| SRAM | Static random-access memory |  |
| SRS | Software requirement specification |  |
| SSC-32 | Serial servo controller board (32 servos) |  |
| STD | Software test document |  |
| SUM | Software user manual |  |
| USART | Universal asynchronous receiver/transmitter |  |
| VCC | IC power-supply pin |  |

Table G‑1 Glossary

## List of Figures

[Figure A‑1 CH3-R Walking Robot 12](#_Toc373746961)

[Figure A‑2 AH2 Walking Robot 12](#_Toc373746962)

[Figure A‑3 Lynxmotion APOD 13](#_Toc373746963)

[Figure A‑4 Flowbotics Studio example 1 14](#_Toc373746964)

[Figure A‑5 Flowbotics Studio example 2 14](#_Toc373746965)

[Figure A‑6 Lynxmotion Visual Sequences 15](#_Toc373746966)

[Figure B‑1 System Boundary Overview 18](#_Toc373746967)

[Figure B‑2 Spriral model 19](#_Toc373746968)

[Figure B‑3 Project plan Page 1 23](#_Toc373746969)

[Figure B‑4 Project plan Page 2 24](#_Toc373746970)

[Figure B‑5 Project plan Page 3 24](#_Toc373746971)

[Figure C‑1: Use cases diagram 27](#_Toc373746972)

[Figure C‑2 Add State Use-case diagram 27](#_Toc373746973)

[Figure C‑3 Remove States Use-case diagram 29](#_Toc373746974)

[Figure C‑4 Import States Use-case diagram 30](#_Toc373746975)

[Figure C‑5 Export States Use-case diagram 31](#_Toc373746976)

[Figure C‑6 Normal input Use-case diagram 33](#_Toc373746977)

[Figure C‑7 Gamepad Input Use-case diagram 34](#_Toc373746978)

[Figure C‑8 Move Forward Use-case diagram 36](#_Toc373746979)

[Figure C‑9 Move Backward Use-case diagram 37](#_Toc373746980)

[Figure C‑10 Turn Left Use-case diagram 39](#_Toc373746981)

[Figure C‑11 Turn Right Use-case diagram 40](#_Toc373746982)

[Figure C‑12 Body Lift Use-case diagram 42](#_Toc373746983)

[Figure C‑13 Body Drop Use-case diagram 43](#_Toc373746984)

[Figure C‑14 Squeeze Left Use-case diagram 44](#_Toc373746985)

[Figure C‑15 Squeeze Right Use-case diagram 46](#_Toc373746986)

[Figure C‑16 Manual Grip Use-case diagram 47](#_Toc373746987)

[Figure C‑17 Auto Grip Use-case diagram 49](#_Toc373746988)

[Figure C‑18 Choose Target Use-case diagram 50](#_Toc373746989)

[Figure C‑19 Change Target Use-case diagram 52](#_Toc373746990)

[Figure C‑20 Lock Target Use-case diagram 53](#_Toc373746991)

[Figure D‑1 APOD Controller Architecture 56](#_Toc373746992)

[Figure D‑2 APOD Communication Mechanism 56](#_Toc373746993)

[Figure D‑3 Live Control User interface design 57](#_Toc373746994)

[Figure D‑4 Sequence Player user interface design 57](#_Toc373746995)

[Figure D‑5 Bluetooth Communication module HC05 58](#_Toc373746996)

[Figure D‑6 STM32F4 Discovery Board 59](#_Toc373746997)

[Figure D‑7 HC 645MG Servo 60](#_Toc373746998)

[Figure D‑8 SSC32 Board 60](#_Toc373746999)

[Figure D‑9 HCSR04 Distance Sensor 61](#_Toc373747000)

[Figure D‑10 RC Servo block 62](#_Toc373747001)

[Figure D‑11 Servo Signal illustration 62](#_Toc373747002)

[Figure D‑12 HC 645MG Signal Details 63](#_Toc373747003)

[Figure D‑13 HC-SR04 Specifications 64](#_Toc373747004)

[Figure D‑14 HC-SR04 Best practice angle 64](#_Toc373747005)

[Figure D‑15 HC-SR04 Operating sequence 65](#_Toc373747006)

[Figure D‑16 Tripod Gait illustration 66](#_Toc373747007)

[Figure D‑17 Wave Gait illustration 67](#_Toc373747008)

[Figure D‑18 Wave Gait step order 67](#_Toc373747009)

[Figure D‑19 Ripple Gait illustration 68](#_Toc373747010)

[Figure D‑20 APOD leg position block 69](#_Toc373747011)

[Figure D‑21 APOD leg swing diagram 70](#_Toc373747012)

[Figure D‑22 APOD Leg full step diagram 70](#_Toc373747013)

[Figure D‑23 Tripod A at Front position 71](#_Toc373747014)

[Figure D‑24 Tripod A at Neutral position 72](#_Toc373747015)

[Figure D‑25 Tripod A at Rear position 72](#_Toc373747016)

[Figure D‑26 Tripod A at Left position 74](#_Toc373747017)

[Figure D‑27 Tripod A at Right position 74](#_Toc373747018)

[Figure D‑28 SSC32 Schematic 76](#_Toc373747019)

[Figure D‑29 ATmega328p Microcontroller 77](#_Toc373747020)

[Figure D‑30 ATmega328p 28 Pins DIP 78](#_Toc373747021)

[Figure D‑31 ATmega328p Block Diagram 79](#_Toc373747022)

[Figure D‑32 MCU Control Register 80](#_Toc373747023)

[Figure D‑33 PORT B Register Description 81](#_Toc373747024)

[Figure D‑34 PORT C Register Description 81](#_Toc373747025)

[Figure D‑35 PORT D Register Description 82](#_Toc373747026)

[Figure D‑36 Timer 1 Control register A 83](#_Toc373747027)

[Figure D‑37 Timer 1 Control register B 83](#_Toc373747028)

[Figure D‑38 Timer 1 Control register C 83](#_Toc373747029)

[Figure D‑39 Timer 1 Counter 84](#_Toc373747030)

[Figure D‑40 Output Compare Register A 84](#_Toc373747031)

[Figure D‑41 Output Compare Register B 84](#_Toc373747032)

[Figure D‑42 Input Capture Register 84](#_Toc373747033)

[Figure D‑43 Timer/Counter1 Interrupt Mask Register 84](#_Toc373747034)

[Figure D‑44 Timer 1 Interrupt Flog register 85](#_Toc373747035)

[Figure D‑45 Baud Rate setup 85](#_Toc373747036)

[Figure D‑46 USART I/O Data register 86](#_Toc373747037)

[Figure D‑47USART Control and Status Register A 86](#_Toc373747038)

[Figure D‑48 USART Control and Status register B 87](#_Toc373747039)

[Figure D‑49 USART Control and Status register C 87](#_Toc373747040)

[Figure D‑50 USART Baud Rate register 88](#_Toc373747041)

[Figure D‑51 Baud Rate Calculation 88](#_Toc373747042)

[Figure D‑52 SPI Control register 89](#_Toc373747043)

[Figure D‑53 SPI Status register 89](#_Toc373747044)

[Figure D‑54 SPI Data register 89](#_Toc373747045)

[Figure D‑55 74HC595 Block diagram 90](#_Toc373747046)

[Figure D‑56 SPI Diagram 91](#_Toc373747047)

[Figure D‑57 SPI- Shift Register 92](#_Toc373747048)

[Figure D‑58 STM32F407VGT6 Pin 93](#_Toc373747049)

[Figure D‑59 STM32F407VGT6 Block Diagram 94](#_Toc373747050)

[Figure D‑60 STM32F407VGT6 GPIO Configuration 96](#_Toc373747051)

[Figure D‑61 GPIO port mode register 97](#_Toc373747052)

[Figure D‑62 GPIO port output type register 97](#_Toc373747053)

[Figure D‑63 GPIO port output speed register 98](#_Toc373747054)

[Figure D‑64 GPIO port pull-up/pull-down register 98](#_Toc373747055)

[Figure D‑65 GPIO port input data register 99](#_Toc373747056)

[Figure D‑66 Baud Rate Calculation 100](#_Toc373747057)

[Figure D‑67 STM32F407VGT6 USART Register Description 100](#_Toc373747058)

[Figure D‑68 APOD Controller Application Activity 103](#_Toc373747059)

[Figure D‑69 Class diagram 104](#_Toc373747060)

[Figure D‑70 STM32F4 Activity 105](#_Toc373747061)

[Figure D‑71 SSC32 Activity 107](#_Toc373747062)

[Figure D‑72 Embbed Forward sequence 109](#_Toc373747063)

[Figure D‑73 Embbed Backward sequence 110](#_Toc373747064)

[Figure D‑74 Embbed Turn left sequence 111](#_Toc373747065)

[Figure D‑75 Embbed Turn Right sequence 112](#_Toc373747066)

[Figure D‑76 Add State Use case sequence 113](#_Toc373747067)

[Figure D‑77 Remove States Use case sequence 114](#_Toc373747068)

[Figure D‑78 Import States Use case sequence 114](#_Toc373747069)

[Figure D‑79 Export States Use case sequence 115](#_Toc373747070)

[Figure D‑80 Normal Input Use case sequence 116](#_Toc373747071)

[Figure D‑81 Gamepad Input Use case sequence 117](#_Toc373747072)

[Figure D‑82 Move Forward Use case sequence 118](#_Toc373747073)

[Figure D‑83 Move Backward Use case sequence 118](#_Toc373747074)

[Figure D‑84 Turn Left Use case sequence 119](#_Toc373747075)

[Figure D‑85 Turn Right Use case sequence 119](#_Toc373747076)

[Figure D‑86 Body Lift Use case sequence 120](#_Toc373747077)

[Figure D‑87 Body Drop Use case sequence 120](#_Toc373747078)

[Figure D‑88 Squeeze Left Use case sequence 121](#_Toc373747079)

[Figure D‑89 Squeeze Right Use case sequence 121](#_Toc373747080)

[Figure D‑90 Manual Grip/Release Use case sequence 122](#_Toc373747081)

[Figure D‑91 Auto Grip Use case sequence 123](#_Toc373747082)

[Figure D‑92 Choose Target Use case sequence 124](#_Toc373747083)

[Figure D‑93 Change Target Use case sequence 125](#_Toc373747084)

[Figure D‑94 Lock Target Use case sequence 126](#_Toc373747085)

[Figure D‑95 Live Control Interface Design 127](#_Toc373747086)

[Figure D‑96 Sequence Player Interface Design 128](#_Toc373747087)

[Figure D‑97 Configuration Interface Design 129](#_Toc373747088)

[Figure D‑98 Object Extractor Interface Design 130](#_Toc373747089)

[Figure F‑1 Run Setup file in delivered package 156](#_Toc373747090)

[Figure F‑2 Click Next on welcome dialog 156](#_Toc373747091)

[Figure F‑3 Choose Destination 157](#_Toc373747092)

[Figure F‑4 Re-Check Setup configuration 157](#_Toc373747093)

[Figure F‑5 Finish Installation 158](#_Toc373747094)

## List of Tables

[Table A‑1 Team Members 11](#_Toc373747589)

[Table B‑1 Roles and Responsibilities 20](#_Toc373747590)

[Table B‑2 Initiating and Planning 21](#_Toc373747591)

[Table B‑3 Software Requirement Analysis 21](#_Toc373747592)

[Table B‑4 Creating Software Design Description 22](#_Toc373747593)

[Table B‑5 Coding 22](#_Toc373747594)

[Table B‑6 Testing 22](#_Toc373747595)

[Table B‑7 Deployment 23](#_Toc373747596)

[Table C‑1 Add state use-case specification 28](#_Toc373747597)

[Table C‑2 Remove states use-case specification 30](#_Toc373747598)

[Table C‑3 Import States use-case specification 31](#_Toc373747599)

[Table C‑4 Export states use-case specification 32](#_Toc373747600)

[Table C‑5 Normal input use-case specification 34](#_Toc373747601)

[Table C‑6 Gamepad input use-case specification 35](#_Toc373747602)

[Table C‑7 Move Forward use-case specification 37](#_Toc373747603)

[Table C‑8 Move Backward use-case specification 38](#_Toc373747604)

[Table C‑9 Turn left use-case specification 40](#_Toc373747605)

[Table C‑10 Turn right use-case specification 41](#_Toc373747606)

[Table C‑11 Body Lift use-case specification 43](#_Toc373747607)

[Table C‑12 Body Drop use-case specification 44](#_Toc373747608)

[Table C‑13 Squeeze Left use-case specification 46](#_Toc373747609)

[Table C‑14 Squeeze Right use-case specification 47](#_Toc373747610)

[Table C‑15 Manual Grip use-case specification 49](#_Toc373747611)

[Table C‑16 Auto Grip use-case specification 50](#_Toc373747612)

[Table C‑17 Choose Target use-case specification 51](#_Toc373747613)

[Table C‑18 Change Target use-case specification 53](#_Toc373747614)

[Table C‑19 Lock Target use-case specification 54](#_Toc373747615)

[Table D‑1 Leg swing sequence 69](#_Toc373747616)

[Table D‑2 APOD Leg full forward step 71](#_Toc373747617)

[Table D‑3 Forward Sequence 73](#_Toc373747618)

[Table D‑4 Backward Sequence 73](#_Toc373747619)

[Table D‑5 Turn Left Sequence 75](#_Toc373747620)

[Table D‑6 Turn Right Sequence 75](#_Toc373747621)

[Table D‑7 Interrupts Usage 82](#_Toc373747622)

[Table D‑8 Baud Rate setup details 86](#_Toc373747623)

[Table D‑9 Mode & Reader command (SSC32) 108](#_Toc373747624)

[Table D‑10 Servo Control command (SSC32) 108](#_Toc373747625)

[Table D‑11 Live Control Interface Design Description 127](#_Toc373747626)

[Table D‑12 Sequence Player Interface Design Description 128](#_Toc373747627)

[Table D‑13 Configuration Interface Design Description 129](#_Toc373747628)

[Table D‑14 Object Extractor Interface Design Description 130](#_Toc373747629)

[Table E‑1 Functional Test Cases 146](#_Toc373747630)

[Table E‑2 UI Test Cases 155](#_Toc373747631)

[Table G‑1 Glossary 159](#_Toc373747632)

## References

*All you need to know about AVR fuses* . (2013, May 30). Retrieved from Embedds: http://www.embedds.com/all-you-need-to-know-about-avr-fuses/

Atmel. (n.d.). *ATmega328p*. Retrieved from Atmel Website: http://www.atmel.com/devices/atmega328p.aspx?tab=overview

Atmel. (n.d.). *ATmega328p.* Retrieved from Atmel Web site: http://www.atmel.com/devices/atmega328.aspx?tab=documents

Electrical and Computer Engineering. (n.d.). *C Coding Standard* . Retrieved from Electrical and Computer Engineering Web site: http://users.ece.cmu.edu/~eno/coding/CCodingStandard.html

Frye, J. (2010, June 16). *SSC-32 Manual*. Retrieved from Lynxmotion Web site: http://www.lynxmotion.com/images/html/build136.htm

ITead Studio. (n.d.). *Ultrasonic Ranging Module HC-SR04.* Retrieved from ITead Studio Web site: http://imall.iteadstudio.com/im120628012.html

Jenkins, P. (n.d.). Retrieved from http://mahapps.com: http://mahapps.com/MahApps.Metro/

Kirillov, A. (n.d.). Retrieved from aforgenet.com: http://www.aforgenet.com/framework/

Kirillov, A. (2010, June 30). *Detecting some simple shapes in images*. Retrieved from http://www.aforgenet.com: http://www.aforgenet.com/articles/shape\_checker/

Kirillov, A. (2010, November 5). *Glyphs' recognition*. Retrieved from http://www.aforgenet.com: http://www.aforgenet.com/articles/glyph\_recognition/

Lötzsch, W. (2012, December 9). *Visual Studio 2012 Metro Styles for WPF*. Retrieved from Codeproject Web site: http://www.codeproject.com/Articles/442856/Visual-Studio-2012-Metro-Styles-for-WPF

NXP . (n.d.). *74HC595PW.* Retrieved from NXP Web site: http://www.nxp.com/products/logic/shift\_registers/74HC595PW.html

Oricom Technologies . (2003, January). *Analysis of Multi-Legged Animal + Robot Gaits*. Retrieved from Oricom Technologies Web site: http://www.oricomtech.com/projects/leg-time.htm#Hex2

STMicroelectronics . (n.d.). *STM32F4DISCOVERY*. Retrieved from STMicroelectronics Web site: http://www.st.com/web/catalog/tools/FM116/SC959/SS1532/PF252419#

STMicroelectronics. (n.d.). *STM32F407VG.* Retrieved from STMicroelectronics Web site: http://www.st.com/web/catalog/mmc/FM141/SC1169/SS1577/LN11/PF252140#

*STMicroelectronics STM32F407VG.* (n.d.). Retrieved from Keil Web site: http://www.keil.com/dd/chip/6103.htm

Tushar. (2013, January 25). *Controlling servo motor with AVR* . Retrieved from Embedds: http://www.embedds.com/controlling-servo-motor-with-avr/

Tushar. (2013, July 26). *Interfacing shift register with AVR* . Retrieved from Embedds Seb site: http://www.embedds.com/interfacing-shift-register-with-avr/

Tushar. (2013, August 4). *Interfacing Ultrasonic Rangefinder with AVR* . Retrieved from Embedds: http://www.embedds.com/interfacing-ultrasonic-rangefinder-with-avr/

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