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

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

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

# Table of Contents

[Table of Contents 2](#_Toc373505436)

[List of Figures 7](#_Toc373505437)

[List of Tables 9](#_Toc373505438)

[A. INTRODUCTION 12](#_Toc373505439)

[I. PROJECT INFORMATION 12](#_Toc373505440)

[II. TEAM MEMBER 12](#_Toc373505441)

[III. APOD INTRODUCTION 12](#_Toc373505442)

[1. HEXAPOD 12](#_Toc373505443)

[2. LYNXMOTION APOD 14](#_Toc373505444)

[IV. EXISTING SOLUTIONS 15](#_Toc373505445)

[V. PROJECT INTRODUCTION 16](#_Toc373505446)

[VI. APPROACHES 17](#_Toc373505447)

[VII. OTHER 17](#_Toc373505448)

[1. WHY WE NEED APOD ROBOT? 17](#_Toc373505449)

[2. PRODUCT 17](#_Toc373505450)

[B. SOFTWARE PROJECT MANAGEMENT PLAN (SPMP) 18](#_Toc373505451)

[I. PROBLEM DEFINITION 18](#_Toc373505452)

[1. NAME OF THIS CAPSTONE PROJECT 18](#_Toc373505453)

[2. PROBLEM ABSTRACT 18](#_Toc373505454)

[3. PROBLEM OVERVIEW 18](#_Toc373505455)

[3.1. The Current System 18](#_Toc373505456)

[3.2. The Propose System 18](#_Toc373505457)

[3.3. Boundaries of The System 19](#_Toc373505458)

[3.4. Developing Environment 19](#_Toc373505459)

[3.4.1 Hardware environment: 19](#_Toc373505460)

[3.4.2 Software environment: 19](#_Toc373505461)

[II. PROJECT ORGANIZATION 20](#_Toc373505462)

[1. SOFTWARE PROCESS MODEL 20](#_Toc373505463)

[2. ROLES AND RESPONSIBILITIES 21](#_Toc373505464)

[3. TOOL AND TECHNIQUES 21](#_Toc373505465)

[3.1. For Development 21](#_Toc373505466)

[3.1.1 Hardware environment: 21](#_Toc373505467)

[3.1.2 Software environment: 21](#_Toc373505468)

[3.2. For Management 21](#_Toc373505469)

[3.3. For Communication 21](#_Toc373505470)

[III. PROJECT MANAGEMENT PLAN 22](#_Toc373505471)

[1. Task 22](#_Toc373505472)

[1.1. Initiating and Planning: 22](#_Toc373505473)

[1.2. Software Requirement Analysis 22](#_Toc373505474)

[1.3. Creating Software Design Description 22](#_Toc373505475)

[1.4. Coding 23](#_Toc373505476)

[1.5. Testing 23](#_Toc373505477)

[1.6. Deployment 23](#_Toc373505478)

[2. Task sheet 24](#_Toc373505479)

[3. All meeting minutes 25](#_Toc373505480)

[IV. CODING CONVENTION 25](#_Toc373505481)

[C. SOFTWARE REQUIREMENTS SPECIFICATION (SRS) 26](#_Toc373505482)

[I. USER REQUIREMENT SPECIFICATIONS 26](#_Toc373505483)

[II. SYSTEM REQUIREMENT SPECIFICATIONS 26](#_Toc373505484)

[1. Hardware requirement 26](#_Toc373505485)

[2. Software interface 26](#_Toc373505486)

[3. External interface 26](#_Toc373505487)

[4. Functional requirement 26](#_Toc373505488)

[5. Non-functional requirement 27](#_Toc373505489)

[III. USE-CASE DIAGRAM AND USE-CASE SPECIFICATIONS 27](#_Toc373505490)

[1. Use-case Diagram 27](#_Toc373505491)

[2. Use-case Specifications 28](#_Toc373505492)

[2.1. Sequences Control 28](#_Toc373505493)

[2.1.1 Add State 28](#_Toc373505494)

[2.1.2 Remove states 29](#_Toc373505495)

[2.1.3 Import States 30](#_Toc373505496)

[2.1.4 Export States 32](#_Toc373505497)

[2.2. Direct Control 33](#_Toc373505498)

[2.2.1 Mode Selection 33](#_Toc373505499)

[a) Normal Input 33](#_Toc373505500)

[b) Gamepad Input 34](#_Toc373505501)

[2.2.2 Movement Control 36](#_Toc373505502)

[a) Move Forward 36](#_Toc373505503)

[b) Move Backward 37](#_Toc373505504)

[c) Turn Left 39](#_Toc373505505)

[d) Turn Right 40](#_Toc373505506)

[e) Body Lift 42](#_Toc373505507)

[f) Body Drop 43](#_Toc373505508)

[g) Squeeze Left 44](#_Toc373505509)

[h) Squeeze Right 46](#_Toc373505510)

[i) Manual Grip 47](#_Toc373505511)

[2.3. Object tracking 49](#_Toc373505512)

[2.3.1 Auto Grip 49](#_Toc373505513)

[2.3.2 Choose Target 50](#_Toc373505514)

[2.3.3 Change Target 52](#_Toc373505515)

[2.3.4 Lock Target 53](#_Toc373505516)

[IV. SOFTWARE SYSTEM ATTRIBUTES 54](#_Toc373505517)

[1. Reliability 54](#_Toc373505518)

[2. Availability 55](#_Toc373505519)

[3. Security 55](#_Toc373505520)

[4. Maintainability 55](#_Toc373505521)

[5. Portability 55](#_Toc373505522)

[6. Performance 55](#_Toc373505523)

[V. ENTITY RELATIONSHIP DIAGRAM 56](#_Toc373505524)

[D. SOFTWARE DESIGN DESCRIPTION (SDD) 56](#_Toc373505525)

[I. DESIGN OVERVIEW 56](#_Toc373505526)

[1. Introduction 56](#_Toc373505527)

[2. Architecture 57](#_Toc373505528)

[3. User interface 57](#_Toc373505529)

[II. COMPONENTS 58](#_Toc373505530)

[1. Central processing: 58](#_Toc373505531)

[1.1. Bluetooth module (HC 05) 58](#_Toc373505532)

[1.2. STM32F4 Discovery 59](#_Toc373505533)

[2. Servos control & Sensor reader 60](#_Toc373505534)

[2.1. HG-645MG Servo 60](#_Toc373505535)

[2.2. SSC32 with ATMEGA328P 61](#_Toc373505536)

[2.3. Ultrasonic ranging module (HC SR04) 61](#_Toc373505537)

[III. IMPLEMENTATION APPROACHES 62](#_Toc373505538)

[1. Algorithms and Mechanism 62](#_Toc373505539)

[1.1. RC Servo Control 62](#_Toc373505540)

[1.1.1 HG-645MG Specification 62](#_Toc373505541)

[1.1.2 How does it work? 63](#_Toc373505542)

[1.1.3 Servo’s Speed Control 64](#_Toc373505543)

[1.2. Sensor Reader (HC-SR04) 65](#_Toc373505544)

[1.2.1 HC-SR04 Specifications 65](#_Toc373505545)

[1.2.2 How to read distance value? 66](#_Toc373505546)

[1.3. Multi-legs Control 67](#_Toc373505547)

[1.3.1 Tripod Gait 67](#_Toc373505548)

[1.3.2 Wave Gait 67](#_Toc373505549)

[1.3.3 Ripple Gait 68](#_Toc373505550)

[1.3.4 Implementation Approaches 69](#_Toc373505551)

[a) Walking In Straight Line 69](#_Toc373505552)

[b) Turning Left – Right 74](#_Toc373505553)

[2. SSC32 Board 76](#_Toc373505554)

[2.1. Schematic 76](#_Toc373505555)

[2.2. Technical Point 77](#_Toc373505556)

[2.2.1 Atmega328P Microcontroller (MCU) 77](#_Toc373505557)

[a) Overview 77](#_Toc373505558)

[b) Features 78](#_Toc373505559)

[2.2.2 General Purposes Input / Output (GPIO) 79](#_Toc373505560)

[2.2.3 Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART) 79](#_Toc373505561)

[2.2.4 Serial Peripheral Interface (SPI) 79](#_Toc373505562)

[2.2.5 Shifter (74HC595) 79](#_Toc373505563)

[2.2.6 SPI – Shifter interface 79](#_Toc373505564)

[3. STM32F4 Discovery Board 79](#_Toc373505565)

[IV. ACTIVITY AND CLASS STRUCTURE 79](#_Toc373505566)

[1. Activity 79](#_Toc373505567)

[1.1. APOD Controller (HMI Application) 79](#_Toc373505568)

[1.2. STM32F4 Discovery (Central processing board) 81](#_Toc373505569)

[1.3. SSC32 Board (Servos control and Sensor reader) 82](#_Toc373505570)

[2. Class Structure 83](#_Toc373505571)

[V. SEQUENCE DIAGRAM 84](#_Toc373505572)

[1. Embedded Controller 84](#_Toc373505573)

[1.1. Forward 84](#_Toc373505574)

[1.2. Backward 85](#_Toc373505575)

[1.3. Turn Left 86](#_Toc373505576)

[1.4. Turn Right 87](#_Toc373505577)

[2. Use-cases Sequences 88](#_Toc373505578)

[2.1. Sequences Control 88](#_Toc373505579)

[2.1.1 Add State 88](#_Toc373505580)

[2.1.2 Remove States 89](#_Toc373505581)

[2.1.3 Import States 89](#_Toc373505582)

[2.1.4 Export States 90](#_Toc373505583)

[2.2. Direct Control 91](#_Toc373505584)

[2.2.1 Mode Selection 91](#_Toc373505585)

[a) Normal Input 91](#_Toc373505586)

[b) Gamepad Input 92](#_Toc373505587)

[2.2.2 Movement Control 93](#_Toc373505588)

[a) Forward 93](#_Toc373505589)

[b) Backward 93](#_Toc373505590)

[c) Turn Left 94](#_Toc373505591)

[d) Turn Right 94](#_Toc373505592)

[e) Body Lift 95](#_Toc373505593)

[f) Body Drop 95](#_Toc373505594)

[g) Squeeze Left 96](#_Toc373505595)

[h) Squeeze Right 96](#_Toc373505596)

[2.3. Object Tracking 97](#_Toc373505597)

[2.3.1 Auto Grip 97](#_Toc373505598)

[2.3.2 Choose target 98](#_Toc373505599)

[2.3.3 Change Target 99](#_Toc373505600)

[2.3.4 Lock Target 100](#_Toc373505601)

[VI. USER INTERFACE DESIGN 101](#_Toc373505602)

[1. Live Control 101](#_Toc373505603)

[2. Sequence Player 102](#_Toc373505604)

[3. Configuration 103](#_Toc373505605)

[4. Object Extractor 104](#_Toc373505606)

[E. SOFTWARE TEST DOCUMENTATION (STD) 105](#_Toc373505607)

[F. SOFTWARE USER’S MANUAL (SUM) 105](#_Toc373505608)

[G. APPENDIX 105](#_Toc373505609)

[I. Glossary 105](#_Toc373505610)

[II. References 105](#_Toc373505611)

# List of Figures

[Figure A‑1 CH3-R Walking Robot 13](#_Toc373505612)

[Figure A‑2 AH2 Walking Robot 13](#_Toc373505613)

[Figure A‑3 Lynxmotion APOD 14](#_Toc373505614)

[Figure A‑4 Flowbotics Studio example 1 15](#_Toc373505615)

[Figure A‑5 Flowbotics Studio example 2 15](#_Toc373505616)

[Figure A‑6 Lynxmotion Visual Sequences 16](#_Toc373505617)

[Figure B‑1 System Boundary Overview 19](#_Toc373505618)

[Figure B‑2 Spriral model 20](#_Toc373505619)

[Figure B‑3 Project plan 25](#_Toc373505620)

[Figure B‑4 Project plan 25](#_Toc373505621)

[Figure C‑1: Use cases diagram 27](#_Toc373505622)

[Figure C‑2 Add State Use-case diagram 28](#_Toc373505623)

[Figure C‑3 Remove States Use-case diagram 29](#_Toc373505624)

[Figure C‑4 Import States Use-case diagram 30](#_Toc373505625)

[Figure C‑5 Export States Use-case diagram 32](#_Toc373505626)

[Figure C‑6 Normal input Use-case diagram 33](#_Toc373505627)

[Figure C‑7 Gamepad Input Use-case diagram 34](#_Toc373505628)

[Figure C‑8 Move Forward Use-case diagram 36](#_Toc373505629)

[Figure C‑9 Move Backward Use-case diagram 37](#_Toc373505630)

[Figure C‑10 Turn Left Use-case diagram 39](#_Toc373505631)

[Figure C‑11 Turn Right Use-case diagram 40](#_Toc373505632)

[Figure C‑12 Body Lift Use-case diagram 42](#_Toc373505633)

[Figure C‑13 Body Drop Use-case diagram 43](#_Toc373505634)

[Figure C‑14 Squeeze Left Use-case diagram 44](#_Toc373505635)

[Figure C‑15 Squeeze Right Use-case diagram 46](#_Toc373505636)

[Figure C‑16 Manual Grip Use-case diagram 47](#_Toc373505637)

[Figure C‑17 Auto Grip Use-case diagram 49](#_Toc373505638)

[Figure C‑18 Choose Target Use-case diagram 50](#_Toc373505639)

[Figure C‑19 Change Target Use-case diagram 52](#_Toc373505640)

[Figure C‑20 Lock Target Use-case diagram 53](#_Toc373505641)

[Figure C‑21 56](#_Toc373505642)

[Figure D‑1 APOD Controller System Architecture 57](#_Toc373505643)

[Figure D‑2 Live Control User interface design 57](#_Toc373505644)

[Figure D‑3 Sequence Player user interface design 58](#_Toc373505645)

[Figure D‑4 Bluetooth Communication module HC05 59](#_Toc373505646)

[Figure D‑5 STM32F4 Discovery Board 60](#_Toc373505647)

[Figure D‑6 HC 645MG Servo 61](#_Toc373505648)

[Figure D‑7 SSC32 Board 61](#_Toc373505649)

[Figure D‑8 HCSR04 Distance Sensor 62](#_Toc373505650)

[Figure D‑9 RC Servo block 63](#_Toc373505651)

[Figure D‑10 Servo Signal illustration 63](#_Toc373505652)

[Figure D‑11 HC 645MG Signal Details 64](#_Toc373505653)

[Figure D‑12 HC-SR04 Specifications 65](#_Toc373505654)

[Figure D‑13 HC-SR04 Best practice angle 65](#_Toc373505655)

[Figure D‑14 HC-SR04 Operating sequence 66](#_Toc373505656)

[Figure D‑15 Tripod Gait illustration 67](#_Toc373505657)

[Figure D‑16 Wave Gait illustration 68](#_Toc373505658)

[Figure D‑17 Wave Gait step order 68](#_Toc373505659)

[Figure D‑18 Ripple Gait illustration 69](#_Toc373505660)

[Figure D‑19 APOD leg position block 70](#_Toc373505661)

[Figure D‑20 APOD leg swing diagram 71](#_Toc373505662)

[Figure D‑21 APOD Leg full step diagram 71](#_Toc373505663)

[Figure D‑22 Tripod A at Front position 72](#_Toc373505664)

[Figure D‑23 Tripod A at Neutral position 73](#_Toc373505665)

[Figure D‑24 Tripod A at Rear position 73](#_Toc373505666)

[Figure D‑25 Tripod A at Left position 75](#_Toc373505667)

[Figure D‑26 Tripod A at Right position 75](#_Toc373505668)

[Figure D‑27 SSC32 Schematic 77](#_Toc373505669)

[Figure D‑28 80](#_Toc373505670)

[Figure D‑29 81](#_Toc373505671)

[Figure D‑30 82](#_Toc373505672)

[Figure D‑31 83](#_Toc373505673)

[Figure D‑32 84](#_Toc373505674)

[Figure D‑33 85](#_Toc373505675)

[Figure D‑34 86](#_Toc373505676)

[Figure D‑35 87](#_Toc373505677)

[Figure D‑36 88](#_Toc373505678)

[Figure D‑37 89](#_Toc373505679)

[Figure D‑38 89](#_Toc373505680)

[Figure D‑39 90](#_Toc373505681)

[Figure D‑40 91](#_Toc373505682)

[Figure D‑41 92](#_Toc373505683)

[Figure D‑42 93](#_Toc373505684)

[Figure D‑43 93](#_Toc373505685)

[Figure D‑44 94](#_Toc373505686)

[Figure D‑45 94](#_Toc373505687)

[Figure D‑46 95](#_Toc373505688)

[Figure D‑47 95](#_Toc373505689)

[Figure D‑48 96](#_Toc373505690)

[Figure D‑49 96](#_Toc373505691)

[Figure D‑50 97](#_Toc373505692)

[Figure D‑51 98](#_Toc373505693)

[Figure D‑52 99](#_Toc373505694)

[Figure D‑53 100](#_Toc373505695)

[Figure D‑54 Live Control Interface Design 101](#_Toc373505696)

[Figure D‑55 Sequence Player Interface Design 102](#_Toc373505697)

[Figure D‑56 Configuration Interface Design 103](#_Toc373505698)

[Figure D‑57 Object Extractor Interface Design 104](#_Toc373505699)

# List of Tables

[Table A‑1 Team Members 12](#_Toc373505700)

[Table B‑1 Roles and Responsibilities 21](#_Toc373505701)

[Table B‑2 Initiating and Planning 22](#_Toc373505702)

[Table B‑3 Software Requirement Analysis 22](#_Toc373505703)

[Table B‑4 Creating Software Design Description 23](#_Toc373505704)

[Table B‑5 Coding 23](#_Toc373505705)

[Table B‑6 Testing 23](#_Toc373505706)

[Table B‑7 Deployment 24](#_Toc373505707)

[Table C‑1 29](#_Toc373505708)

[Table C‑2 30](#_Toc373505709)

[Table C‑3 32](#_Toc373505710)

[Table C‑4 33](#_Toc373505711)

[Table C‑5 34](#_Toc373505712)

[Table C‑6 36](#_Toc373505713)

[Table C‑7 37](#_Toc373505714)

[Table C‑8 38](#_Toc373505715)

[Table C‑9 40](#_Toc373505716)

[Table C‑10 41](#_Toc373505717)

[Table C‑11 43](#_Toc373505718)

[Table C‑12 44](#_Toc373505719)

[Table C‑13 46](#_Toc373505720)

[Table C‑14 47](#_Toc373505721)

[Table C‑15 49](#_Toc373505722)

[Table C‑16 50](#_Toc373505723)

[Table C‑17 51](#_Toc373505724)

[Table C‑18 53](#_Toc373505725)

[Table C‑19 54](#_Toc373505726)

[Table D‑1 70](#_Toc373505727)

[Table D‑2 APOD Leg full forward step 72](#_Toc373505728)

[Table D‑3 Forward Sequence 74](#_Toc373505729)

[Table D‑4 Backward Sequence 74](#_Toc373505730)

[Table D‑5 Turn Left Sequence 76](#_Toc373505731)

[Table D‑6 Turn Right Sequence 76](#_Toc373505732)

[Table D‑7 Live Control Interface Design Description 101](#_Toc373505733)

[Table D‑8 Sequence Player Interface Design Description 102](#_Toc373505734)

[Table D‑9 Configuration Interface Design Description 103](#_Toc373505735)

[Table D‑10 Object Extractor Interface Design Description 104](#_Toc373505736)

This page intentionally left blank.

# INTRODUCTION

## PROJECT INFORMATION

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

## TEAM MEMBER

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Name** | **Role** | **Contact** |
| **1** | 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

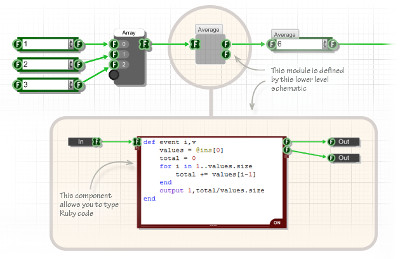


Figure A‑4 Flowbotics Studio example 1

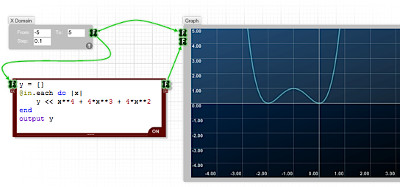


Figure A‑5 Flowbotics Studio example 2

* Lynxmotion Visual Sequences: $39.94

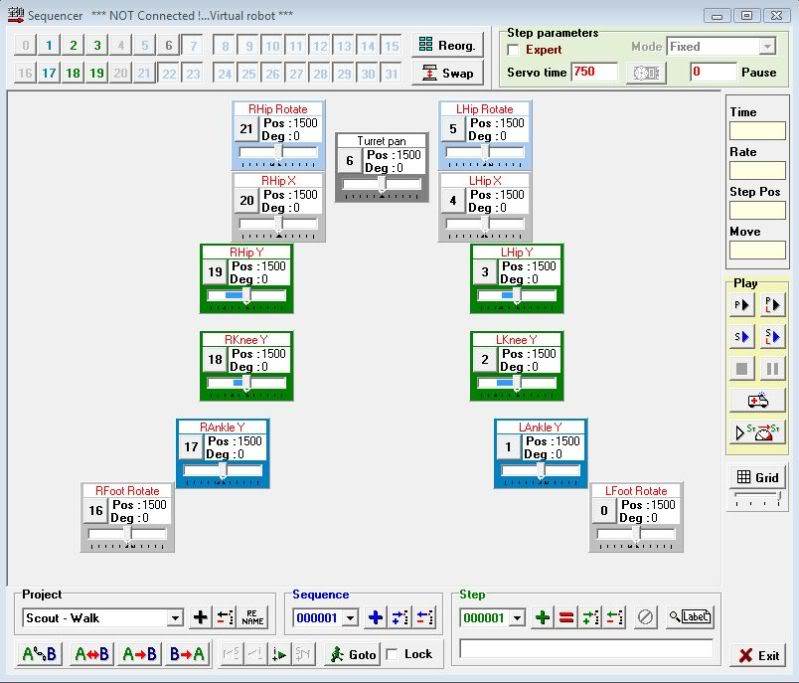


Figure A‑6 Lynxmotion Visual Sequences

## PROJECT INTRODUCTION

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

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

## APPROACHES

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

## OTHER

### WHY WE NEED APOD ROBOT?

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

### PRODUCT

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

# SOFTWARE PROJECT MANAGEMENT PLAN (SPMP)

## PROBLEM DEFINITION

### NAME OF THIS CAPSTONE PROJECT

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

### PROBLEM ABSTRACT

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

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

### PROBLEM OVERVIEW

#### The Current System

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

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

#### The Propose System

The system consists of 4 module:

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

Main features will be provided:

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

#### Boundaries of The System



Figure B‑1 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 ARV studio 4 for main board programming.
* Developing environment Visual studio 2012 with C# language for user interface programming.

## PROJECT ORGANIZATION

### SOFTWARE PROCESS MODEL

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

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

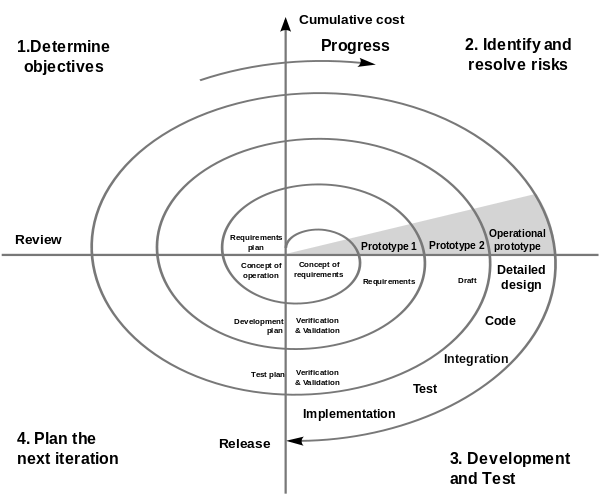


Figure B‑2 Spriral model

### 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

### Task

#### Initiating and Planning:

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

Table B‑2 Initiating and Planning

#### Software Requirement Analysis

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

Table B‑3 Software Requirement Analysis

#### Creating Software Design Description

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

Table B‑4 Creating Software Design Description

#### Coding

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

Table B‑5 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 system include : hexapod robot and hexapod controller simulator |
| Output | Software packages, user manual |
| Deliverables | Software packages, user manual |
| Effort (man-month) | 0.3 mm |
| Dependencies and Constrains | Completion of all other tasks |
| Risks | - Hardware’s errors |

Table B‑7 Deployment

### Task sheet

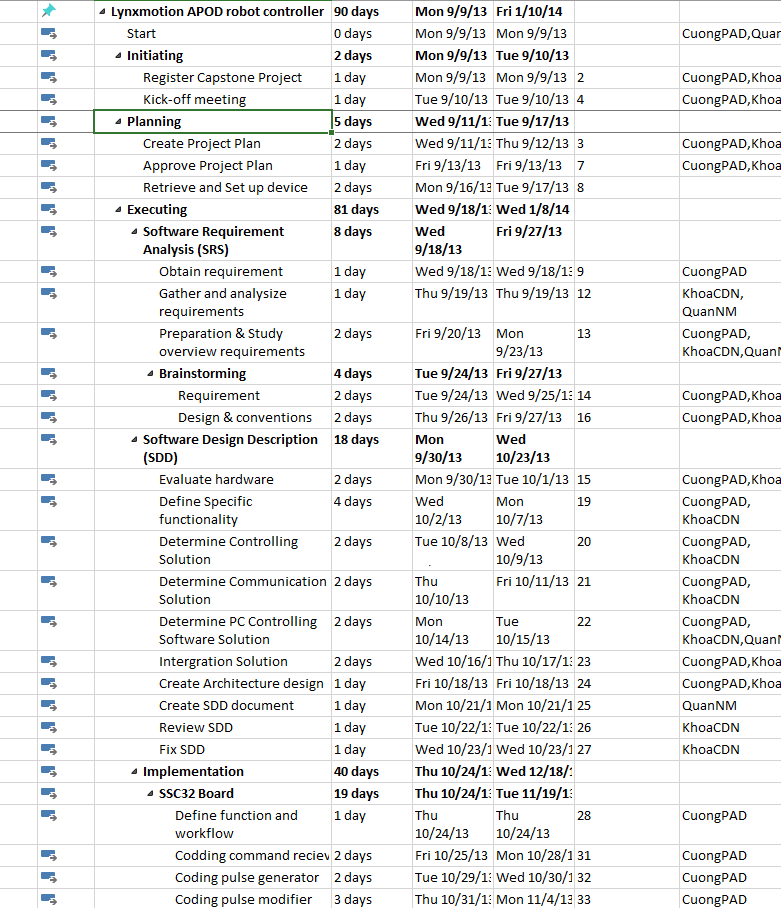


Figure B‑3 Project plan



Figure B‑4 Project plan

### All meeting minutes

<to be updated>

## CODING CONVENTION

The following rules follow:

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

# SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

## USER REQUIREMENT SPECIFICATIONS

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

## SYSTEM REQUIREMENT SPECIFICATIONS

### Hardware requirement

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

### Software interface

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

### External interface

* PS2 Controller

### Functional requirement

* Direction control : user can direct the A-Pod to turn left, right, go forward or backward.
* Camera view control : user can see the view sending by the A-Pod camera.
* Grab control : user can direct the A-Pod to grab things.
* Remote control : enable remote controlling, help the user control the A-Pod through a programmed Bluetooth device.

### Non-functional requirement

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

## USE-CASE DIAGRAM AND USE-CASE SPECIFICATIONS

### Use-case Diagram

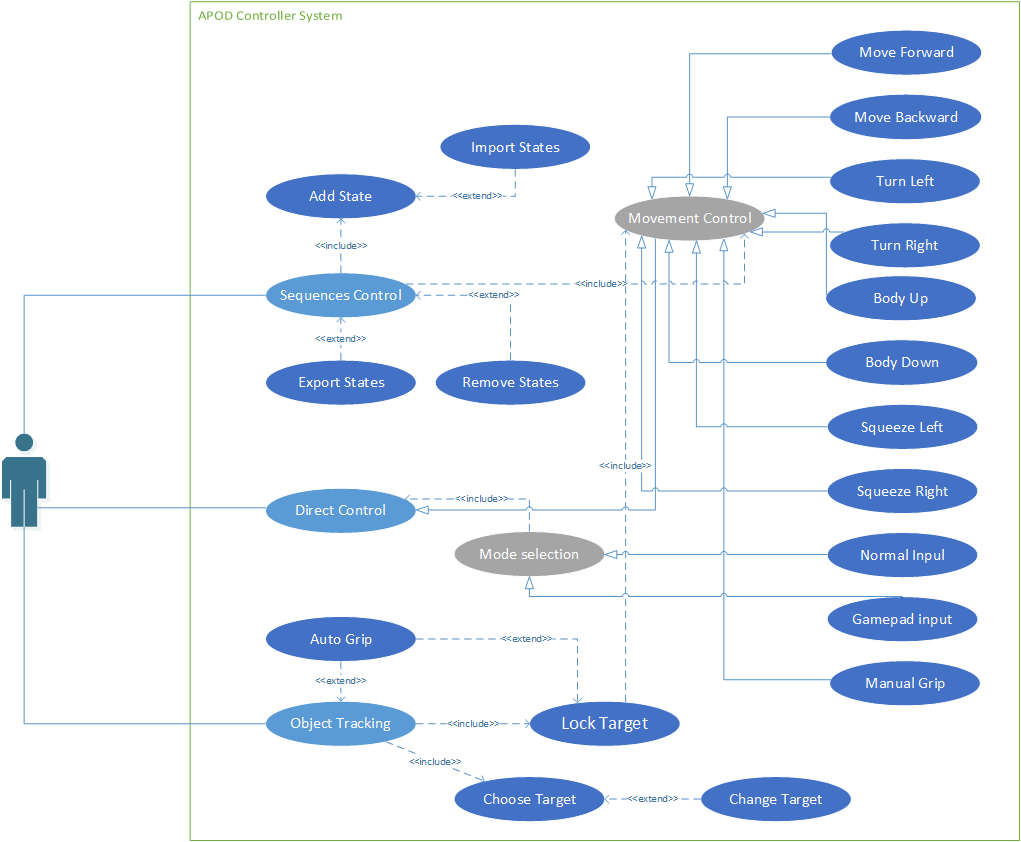
****

Figure C‑1: Use cases diagram

### Use-case Specifications

#### Sequences Control

##### Add State

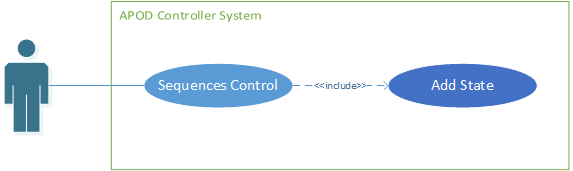


Figure C‑2 Add State Use-case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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

##### Remove states

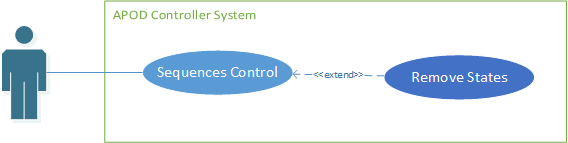


Figure C‑3 Remove States Use-case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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

##### 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

##### Export States

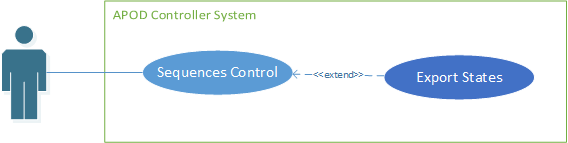


Figure C‑5 Export States Use-case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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

#### Direct Control

##### Mode Selection

###### Normal Input

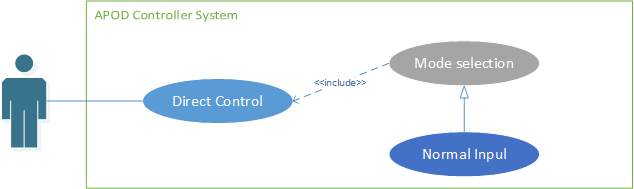


Figure C‑6 Normal input Use-case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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

###### Gamepad Input

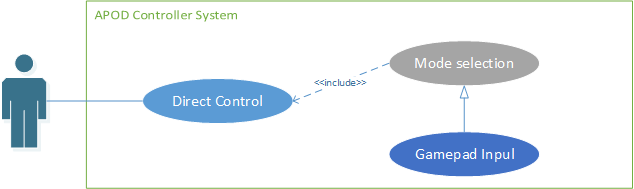


Figure C‑7 Gamepad Input Use-case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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

##### Movement Control

###### Move Forward

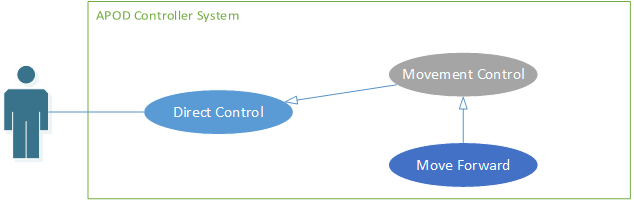


Figure C‑8 Move Forward Use-case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 Backward

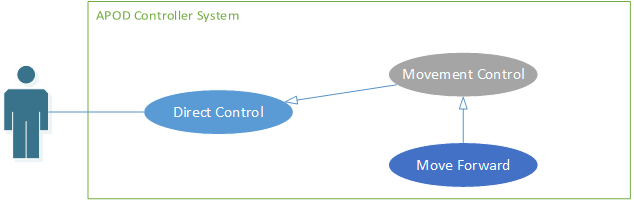


Figure C‑9 Move Backward Use-case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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

###### Turn Left

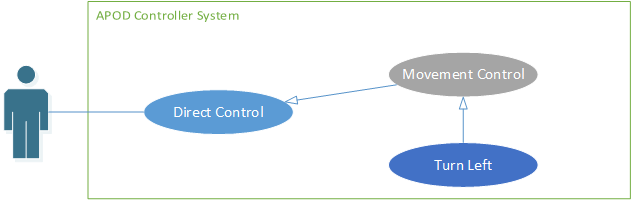


Figure C‑10 Turn Left Use-case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 Right

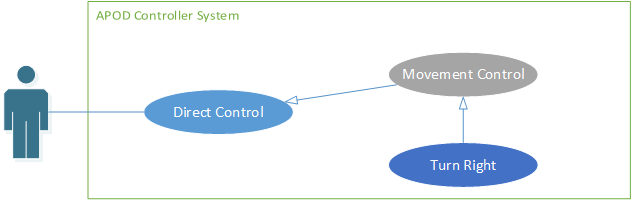


Figure C‑11 Turn Right Use-case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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

###### Body Lift

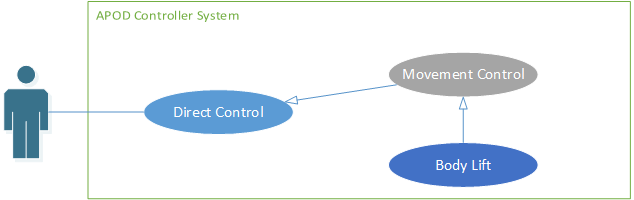


Figure C‑12 Body Lift Use-case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 Drop

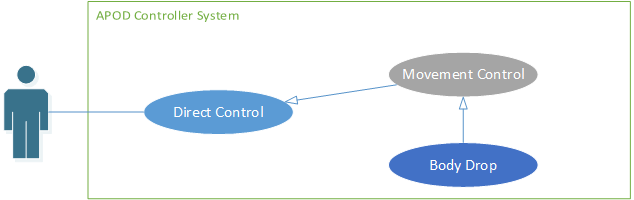


Figure C‑13 Body Drop Use-case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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

###### Squeeze Left

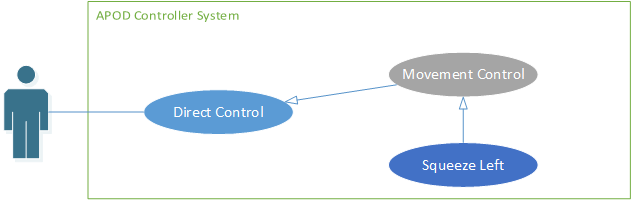


Figure C‑14 Squeeze Left Use-case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 Right

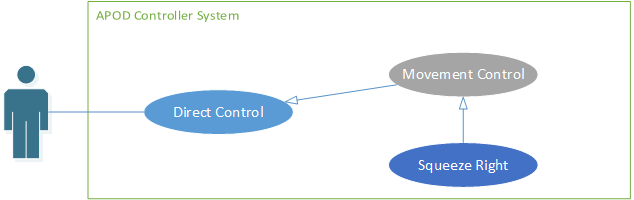


Figure C‑15 Squeeze Right Use-case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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

###### Manual Grip

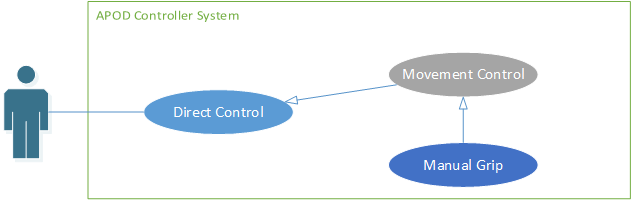


Figure C‑16 Manual Grip Use-case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **MANUAL GRIP 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 Mandibles  **Goal:**  APOD Mandibles Grip  **Triggers**  User click [Grip] 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 ()   **Alternative Scenario:**   * Alt 1: press “I” 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‑15

#### Object tracking

##### Auto Grip

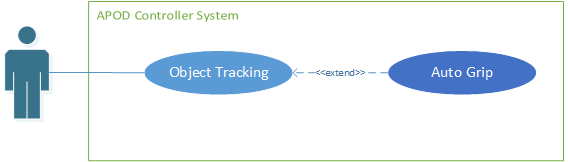


Figure C‑17 Auto Grip Use-case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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

##### 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

##### Change Target

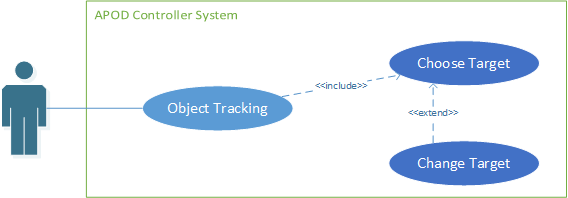


Figure C‑19 Change Target Use-case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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

##### Lock Target

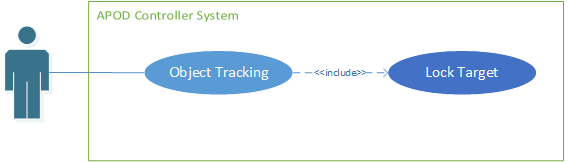


Figure C‑20 Lock Target Use-case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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

## SOFTWARE SYSTEM ATTRIBUTES

### Reliability

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

### Availability

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

### Security

* N/A

### Maintainability

* N/A

### Portability

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

### Performance

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

## ENTITY RELATIONSHIP DIAGRAM

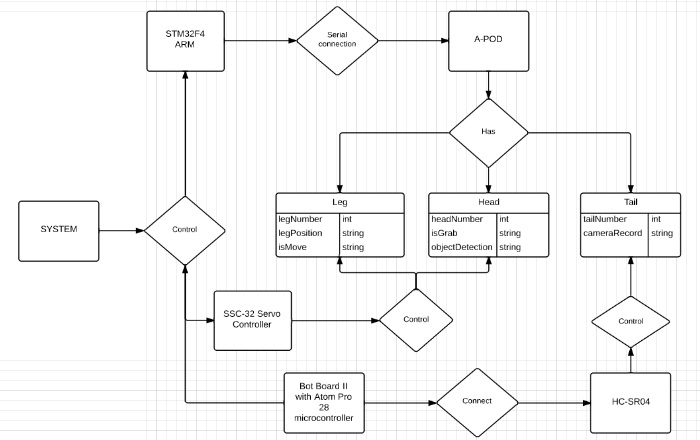


Figure C‑21

# 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

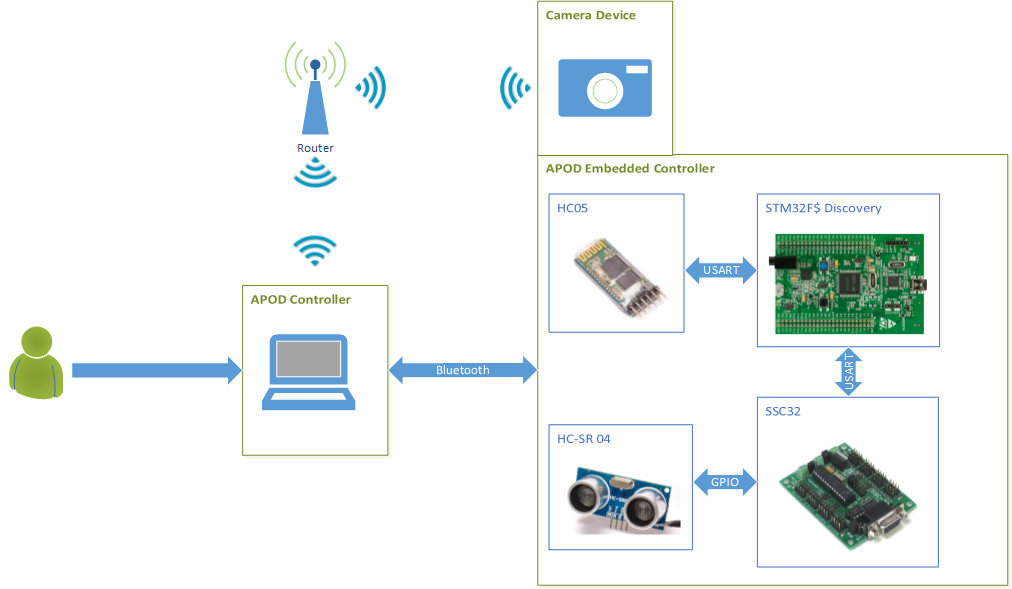


Figure D‑1 APOD Controller System Architecture

### User interface

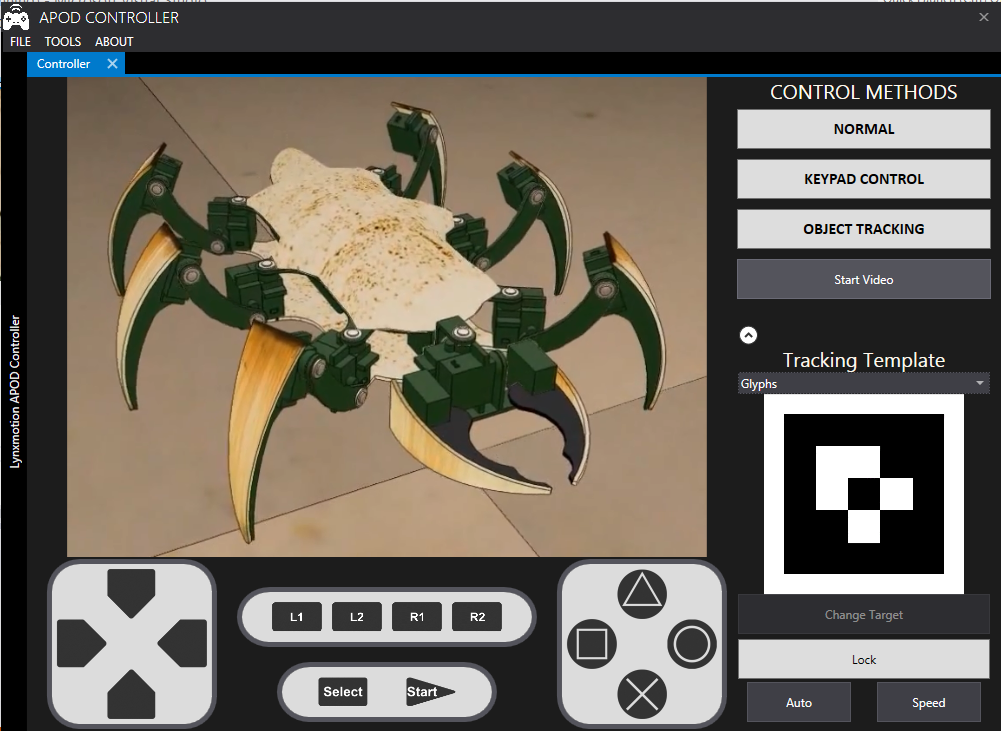


Figure D‑2 Live Control User interface design

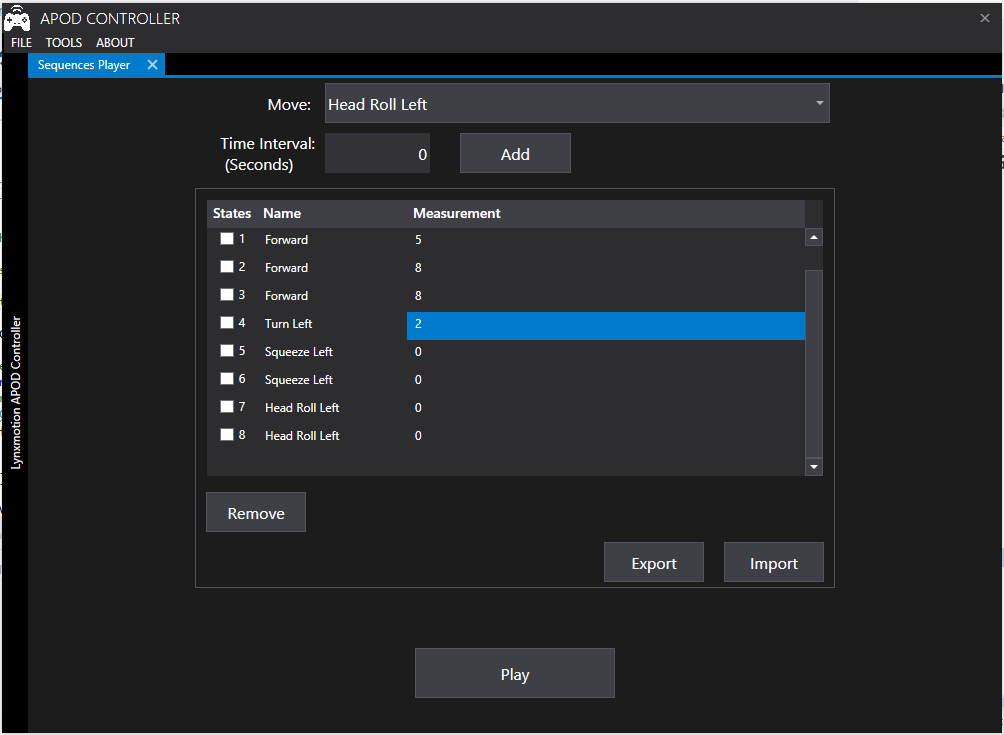


Figure D‑3 Sequence Player user interface design

## COMPONENTS

### Central processing:

#### Bluetooth module (HC 05)

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

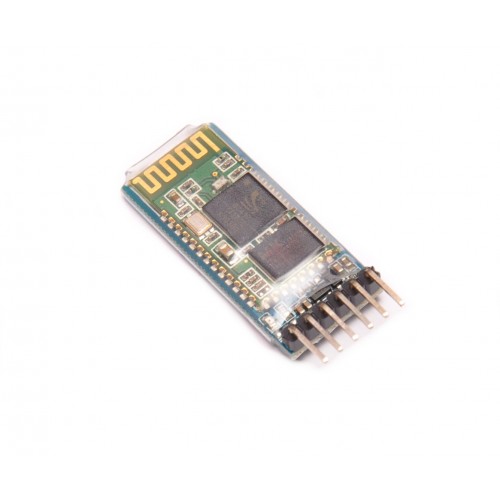


Figure D‑4 Bluetooth Communication module HC05

#### STM32F4 Discovery

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



Figure D‑5 STM32F4 Discovery Board

### Servos control & Sensor reader

#### HG-645MG Servo

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



Figure D‑6 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)

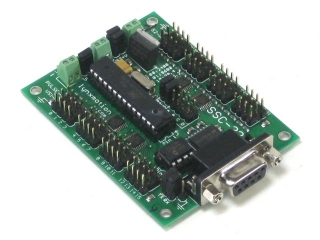


Figure D‑7 SSC32 Board

#### Ultrasonic ranging module (HC SR04)

* Ultrasonic module using for distance measurement, discovering obstacles.



Figure D‑8 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.

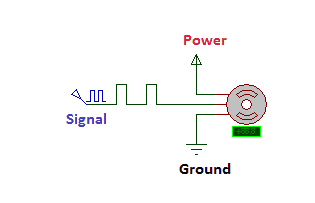


Figure D‑9 RC Servo block

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‑10 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:

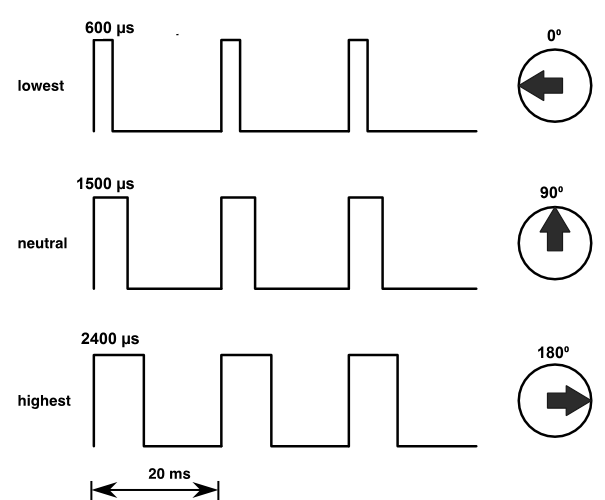


Figure D‑11 HC 645MG Signal Details

##### 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

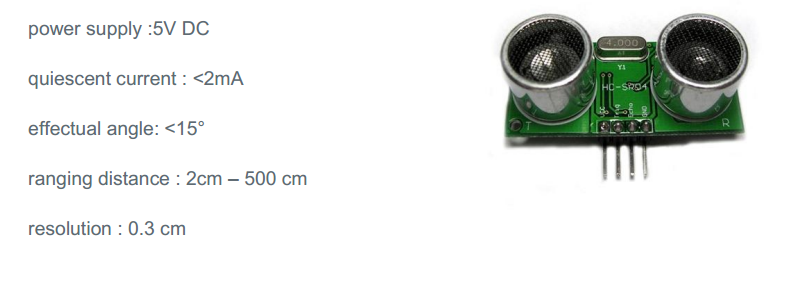


Figure D‑12 HC-SR04 Specifications

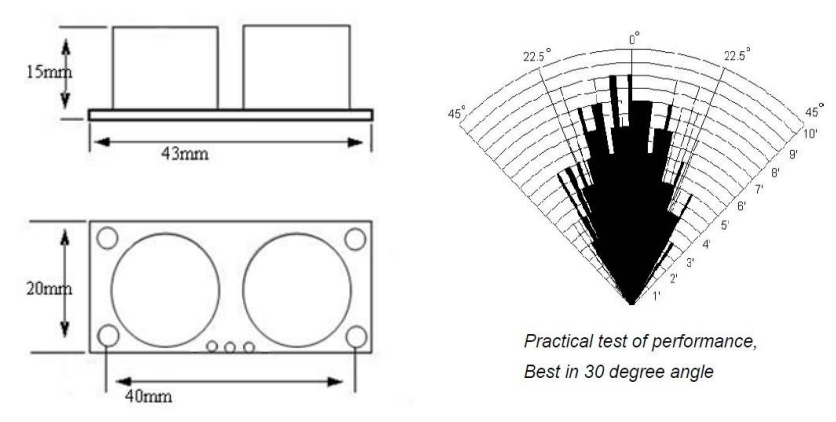


Figure D‑13 HC-SR04 Best practice angle

##### 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‑14 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‑15 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.

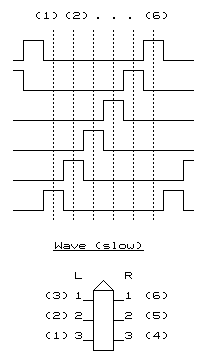


Figure D‑16 Wave Gait illustration

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

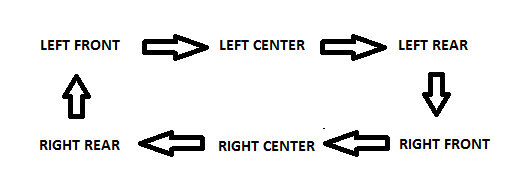


Figure D‑17 Wave Gait step order

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)

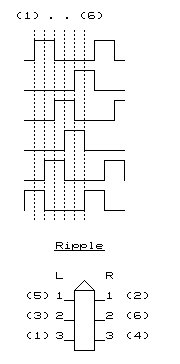


Figure D‑18 Ripple Gait illustration

##### 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

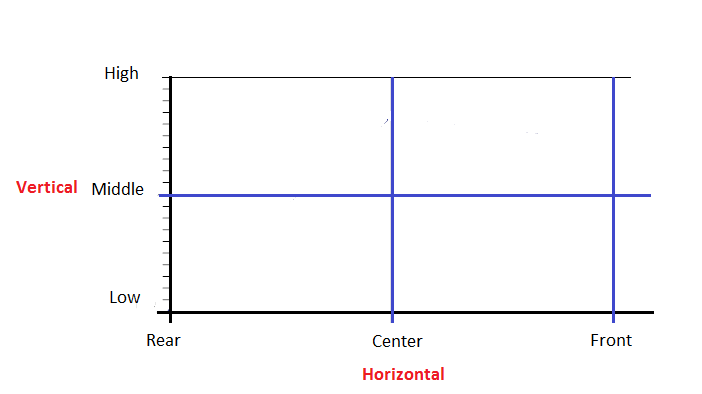


Figure D‑19 APOD leg position block

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

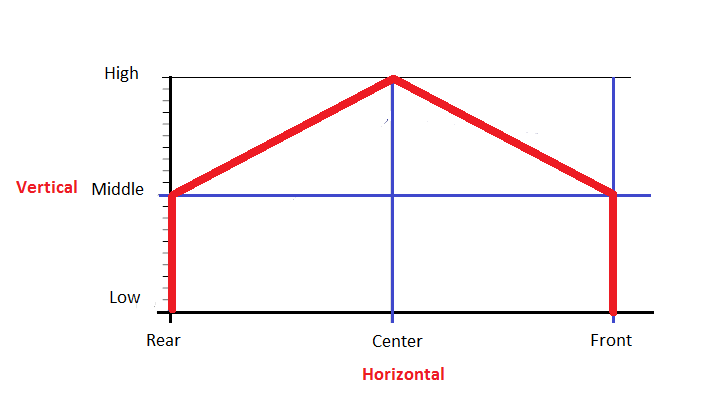


Figure D‑20 APOD leg swing diagram

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:

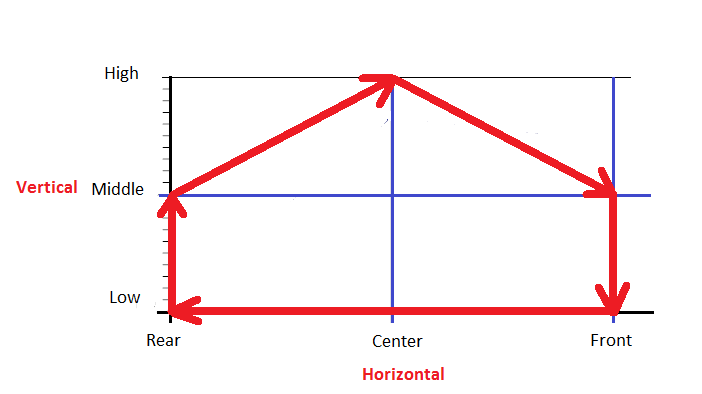


Figure D‑21 APOD Leg full step diagram

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)

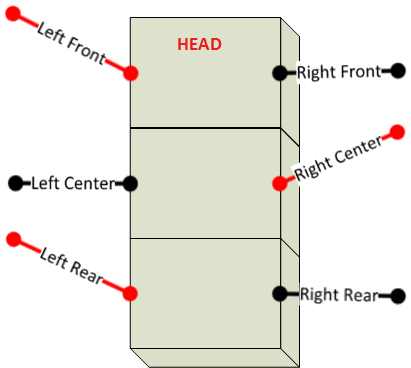


Figure D‑22 Tripod A at Front position

* Position: Center or Neutral

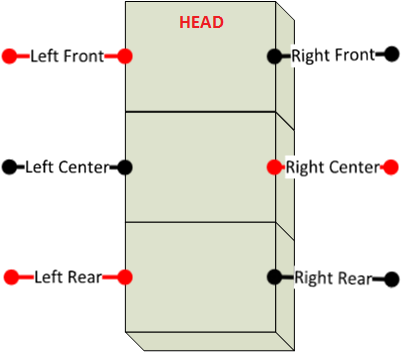


Figure D‑23 Tripod A at Neutral position

* Position: Rear (far from head)

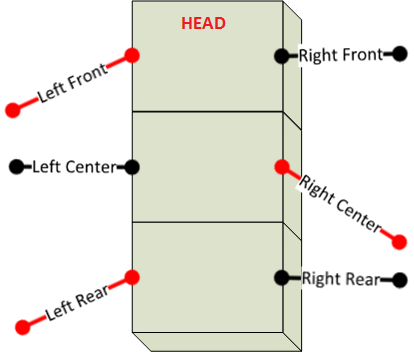


Figure D‑24 Tripod A at Rear position

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

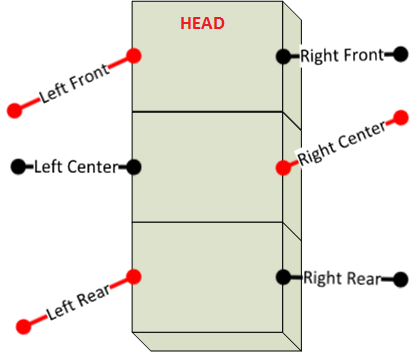


Figure D‑25 Tripod A at Left position

* Position: Right



Figure D‑26 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

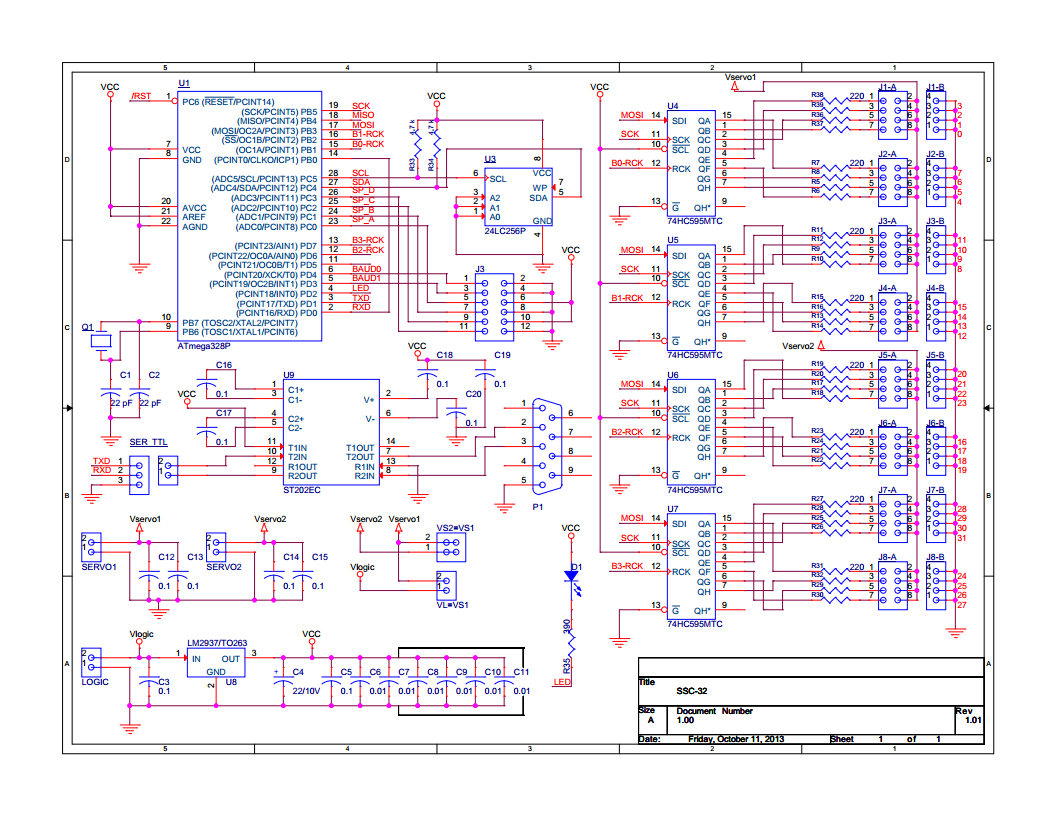


Figure D‑27 SSC32 Schematic

#### 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

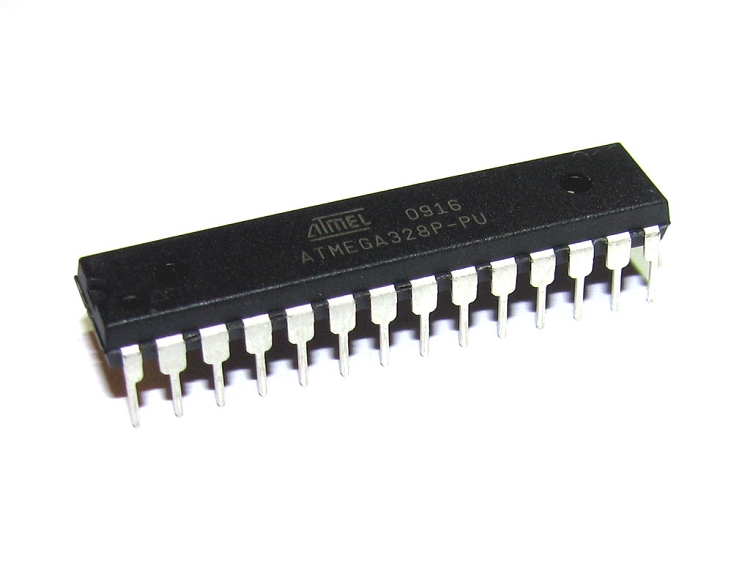


Figure D‑28 ATmega328p Microcontroller

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

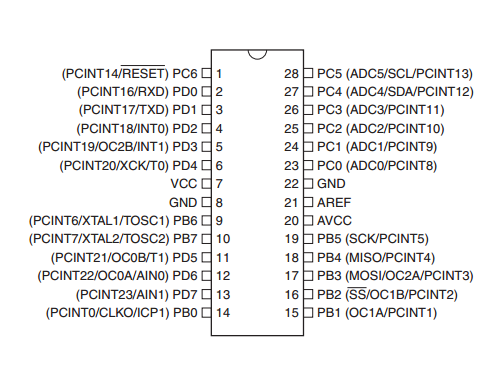


Figure D‑29 ATmega328p 28 Pins DIP

###### Block Diagram

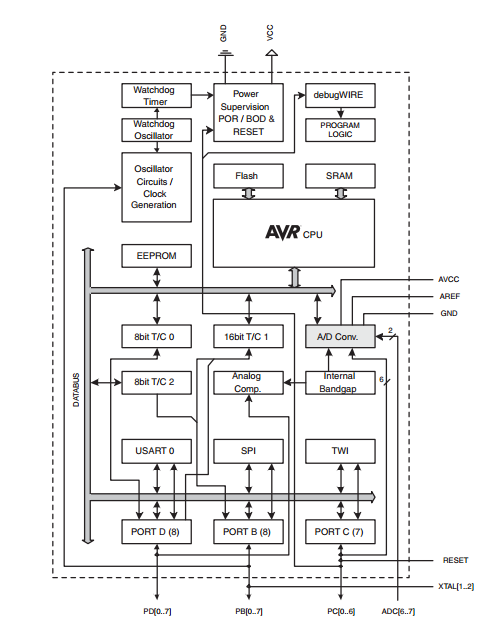


Figure D‑30 ATmega328p Block Diagram

##### 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

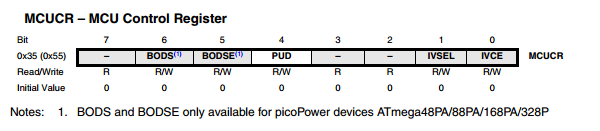


Figure D‑31 MCU Control Register

* 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

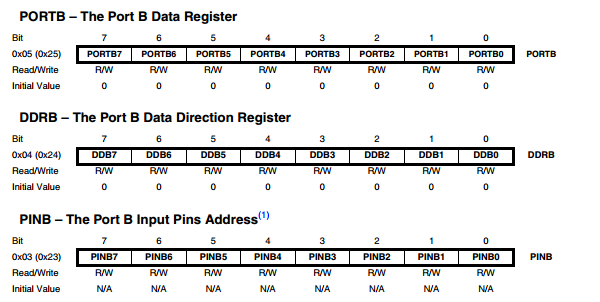


Figure D‑32 PORT B Register Description

PORT C

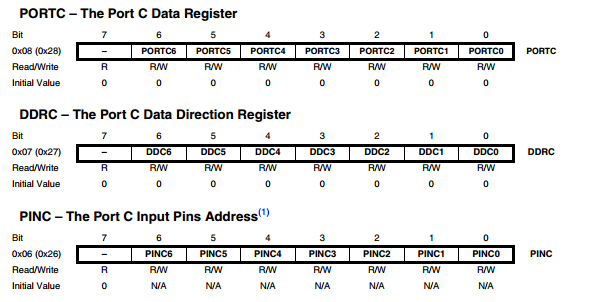


Figure D‑33 PORT C Register Description

PORT D

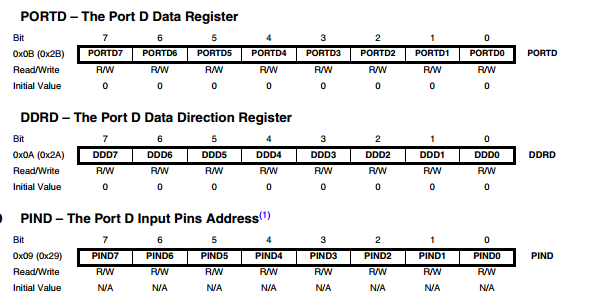


Figure D‑34 PORT D Register Description

##### 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

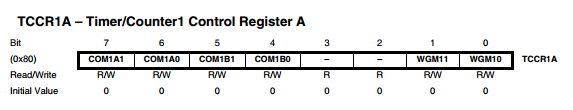


Figure D‑35 Timer 1 Control register A

* 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

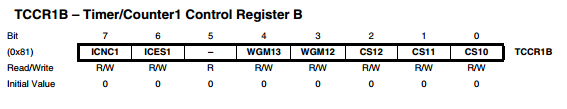


Figure D‑36 Timer 1 Control register B

* 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

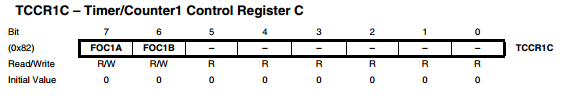


Figure D‑37 Timer 1 Control register C

* 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

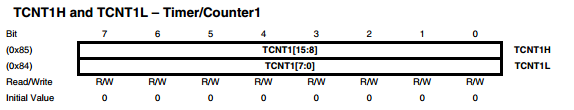


Figure D‑38 Timer 1 Counter

ORC1A

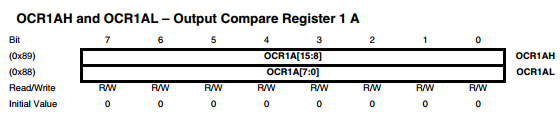


Figure D‑39 Output Compare Register A

ORC1B

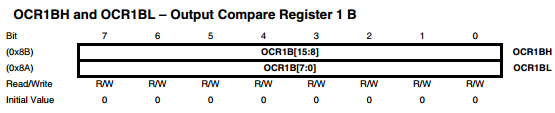


Figure D‑40 Output Compare Register B

IRC1

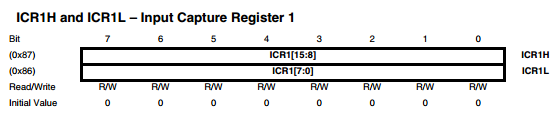


Figure D‑41 Input Capture Register

TIMSK1

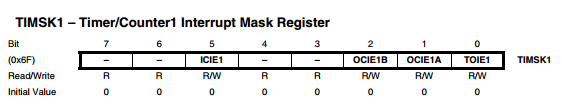


Figure D‑42 Timer/Counter1 Interrupt Mask Register

* 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

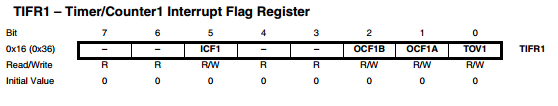


Figure D‑43 Timer 1 Interrupt Flog register

* 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

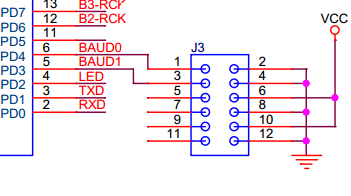


Figure D‑44 Baud Rate setup

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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

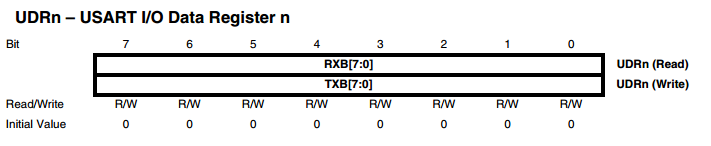


Figure D‑45 USART I/O Data register

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

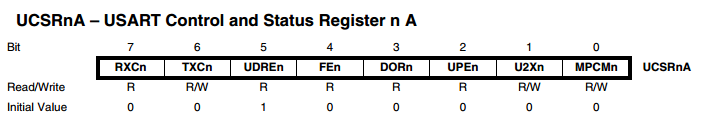


Figure D‑46USART Control and Status Register A

* 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

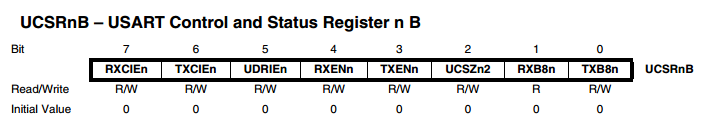


Figure D‑47 USART Control and Status register B

* 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

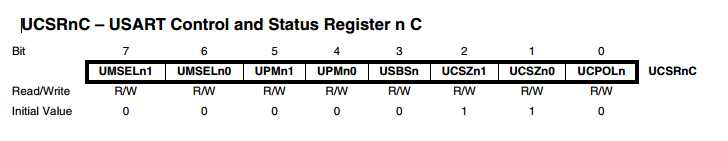


Figure D‑48 USART Control and Status register C

* 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

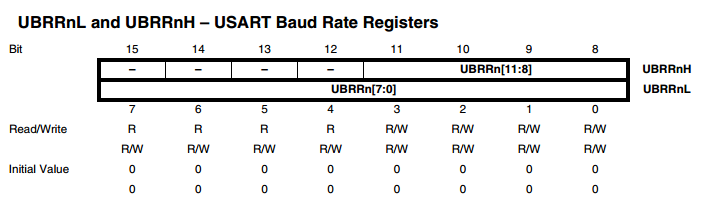


Figure D‑49 USART Baud Rate register

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

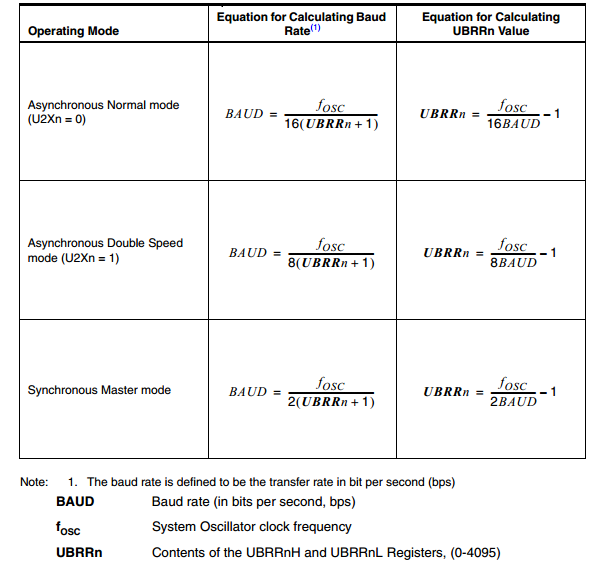


Figure D‑50 Baud Rate Calculation

##### 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

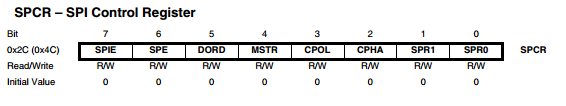


Figure D‑51 SPI Control register

* 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

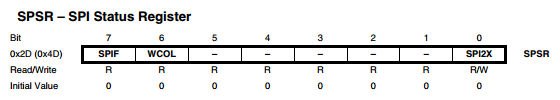


Figure D‑52 SPI Status register

* 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

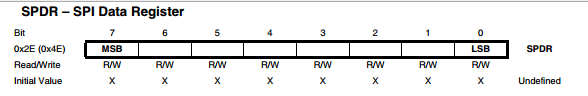


Figure D‑53 SPI Data register

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?

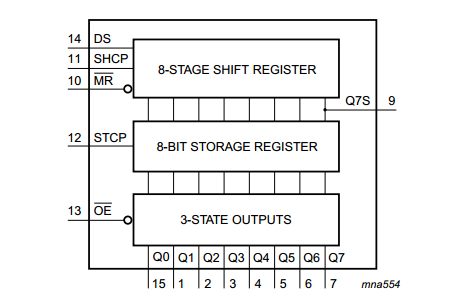


Figure D‑54 74HC595 Block diagram

* **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

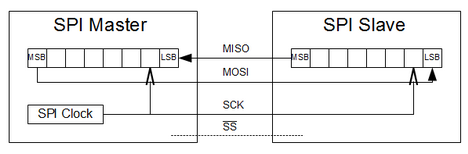


Figure D‑55 SPI Diagram

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:

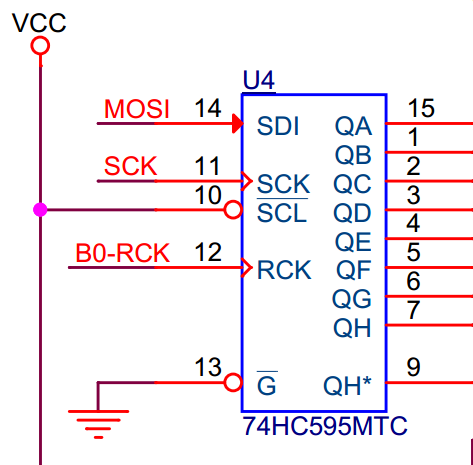


Figure D‑56 SPI- Shift Register

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

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 SC\_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 feature of the ATmega328p.

### STM32F4 Discovery Board

## ACTIVITY AND CLASS STRUCTURE

### Activity

#### APOD Controller (HMI Application)

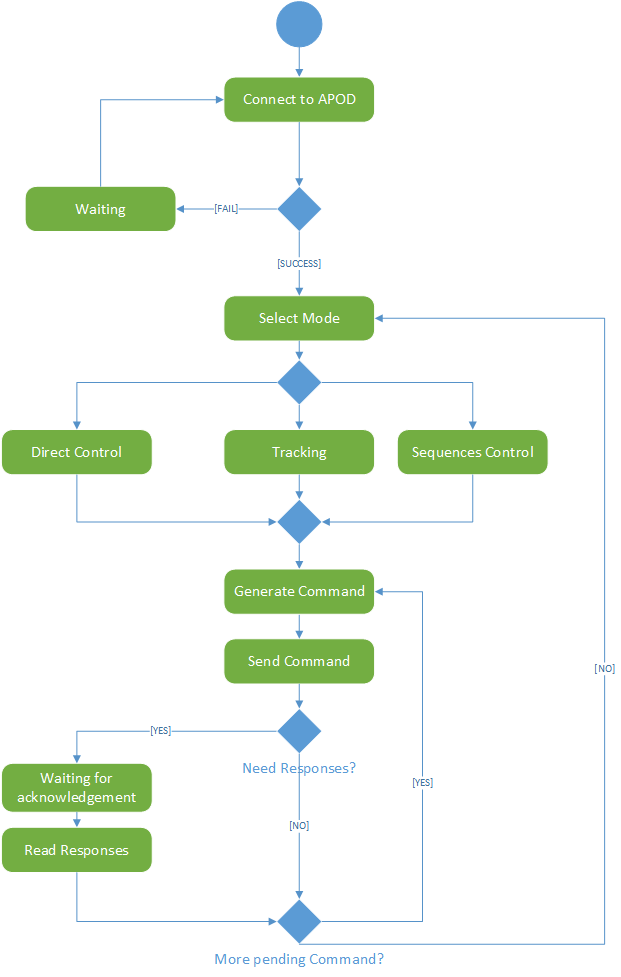


Figure D‑57

#### STM32F4 Discovery (Central processing board)



Figure D‑58

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

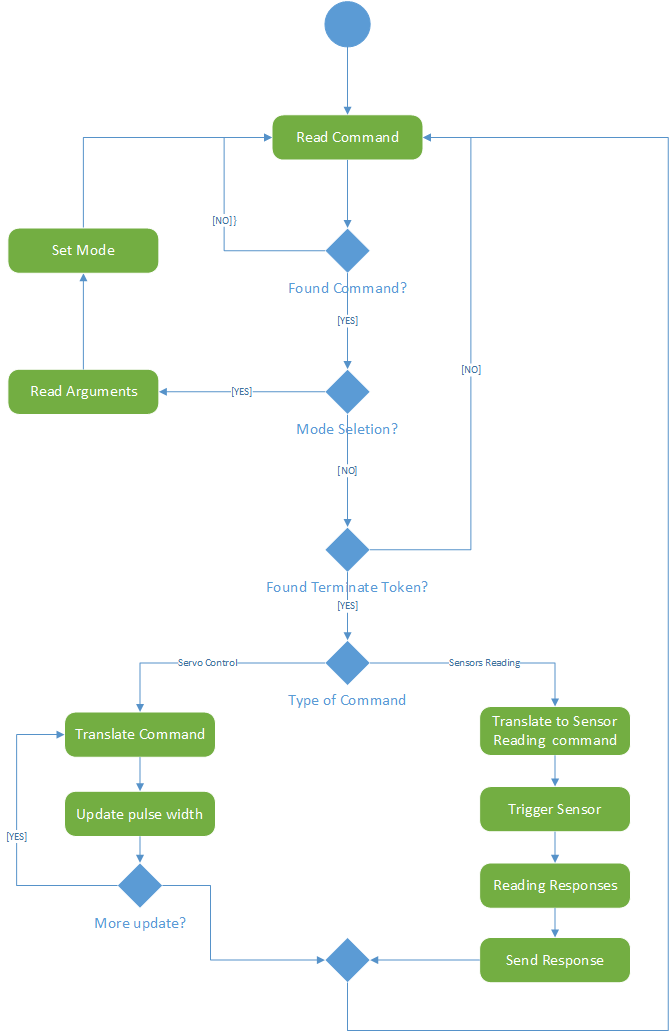


Figure D‑59

### Class Structure

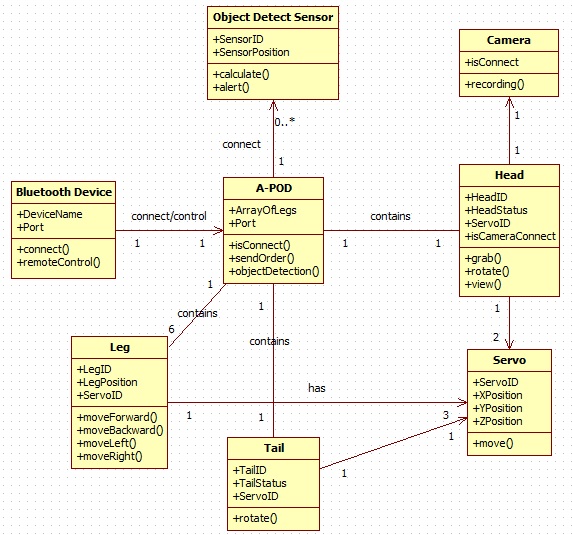


Figure D‑60

## SEQUENCE DIAGRAM

### Embedded Controller

#### Forward

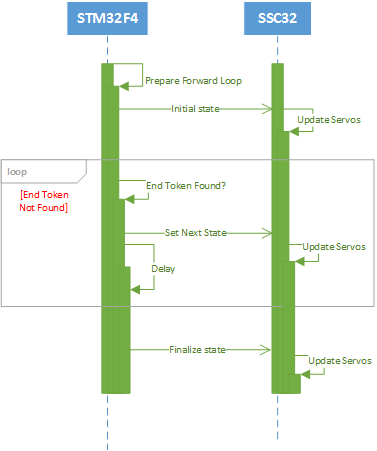


Figure D‑61

#### Backward

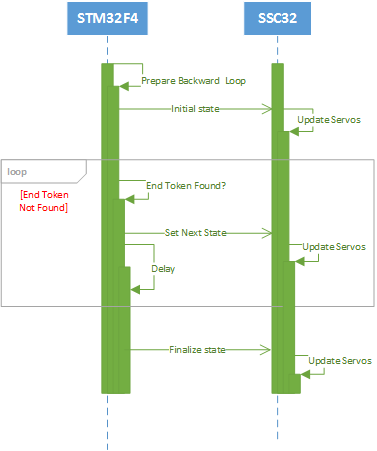


Figure D‑62

#### Turn Left

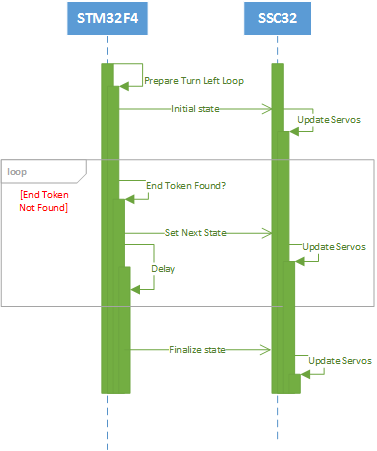


Figure D‑63

#### Turn Right

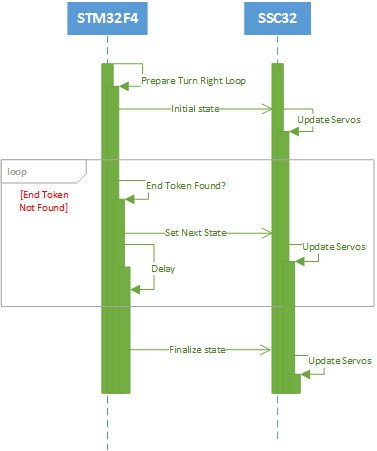


Figure D‑64

### Use-cases Sequences

#### Sequences Control

##### Add State



Figure D‑65

##### Remove States

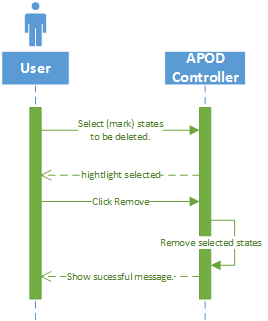


Figure D‑66

##### Import States

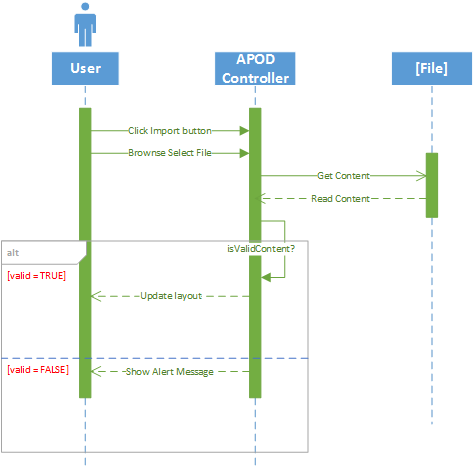


Figure D‑67

##### Export States

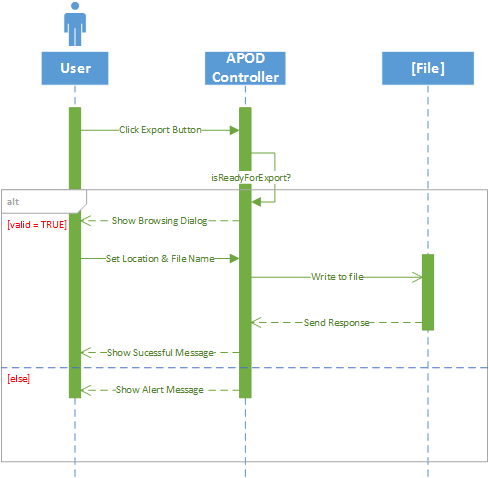


Figure D‑68

#### Direct Control

##### Mode Selection

###### Normal Input

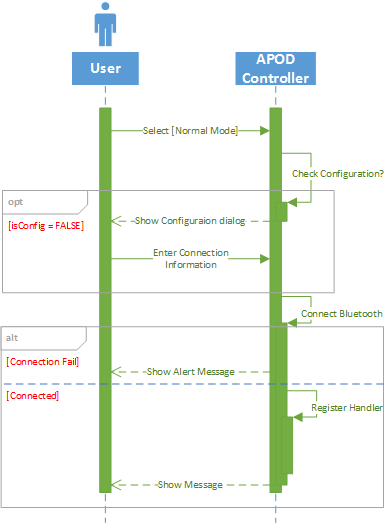


Figure D‑69

###### Gamepad Input

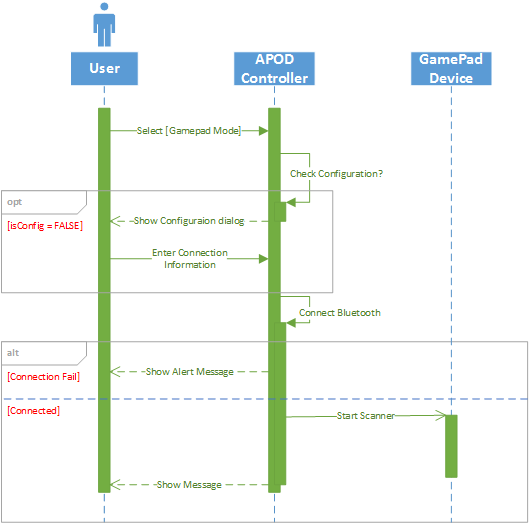


Figure D‑70

##### Movement Control

###### Forward

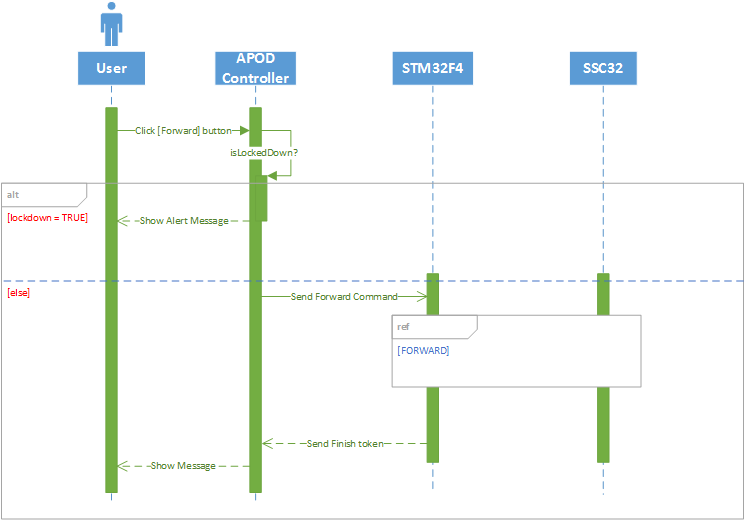


Figure D‑71

###### Backward

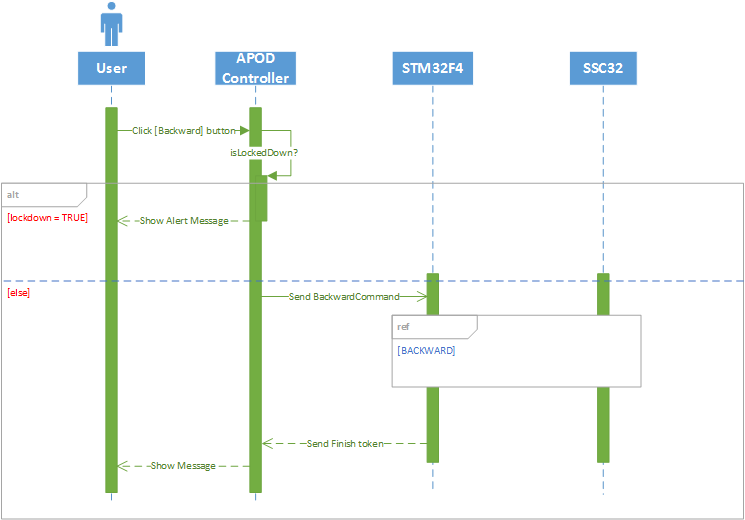


Figure D‑72

###### Turn Left

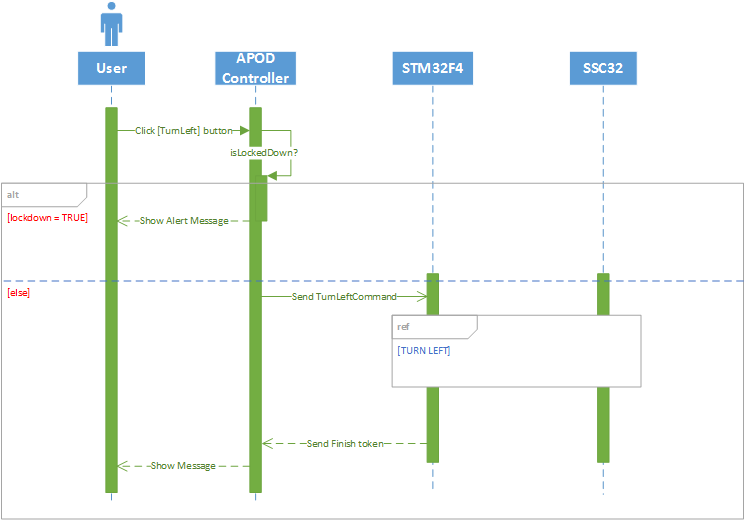


Figure D‑73

###### Turn Right

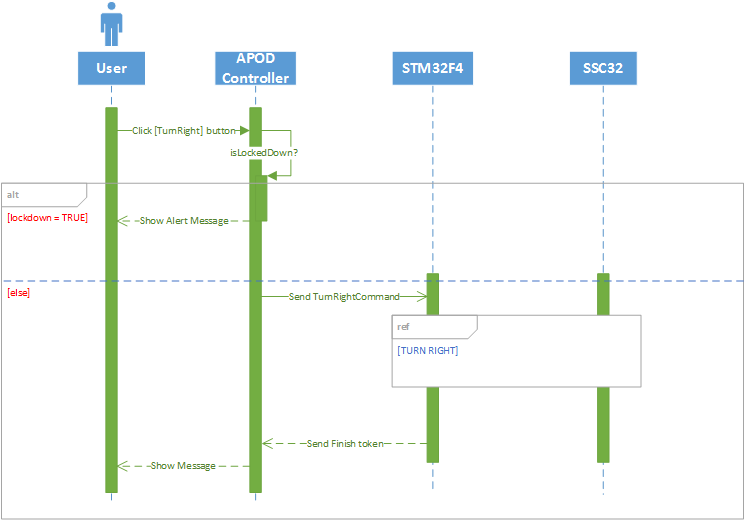


Figure D‑74

###### Body Lift

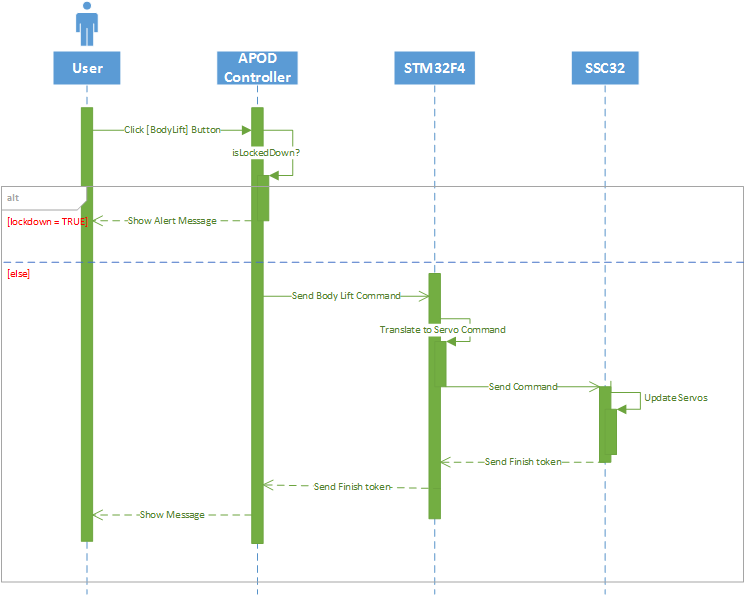


Figure D‑75

###### Body Drop

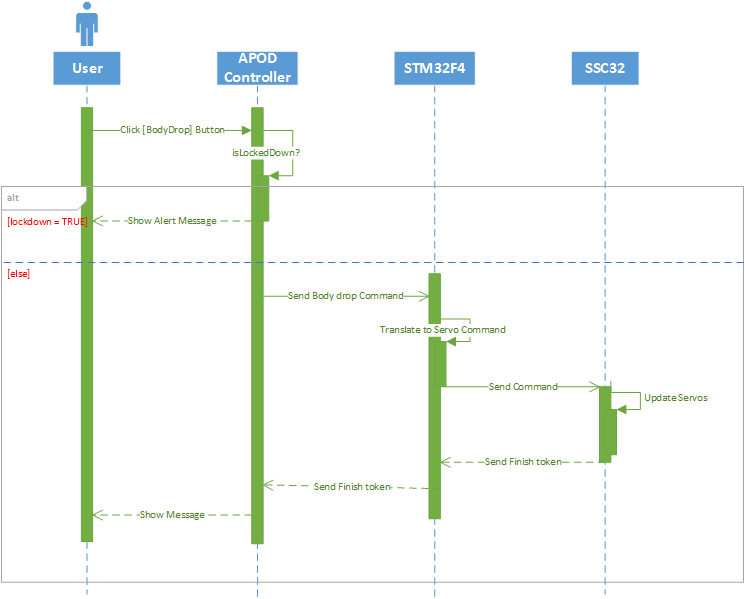


Figure D‑76

###### Squeeze Left

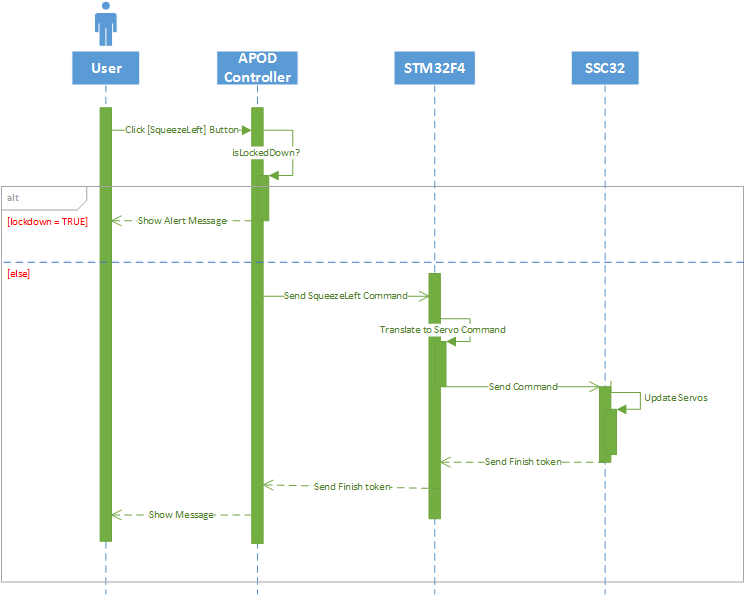


Figure D‑77

###### Squeeze Right

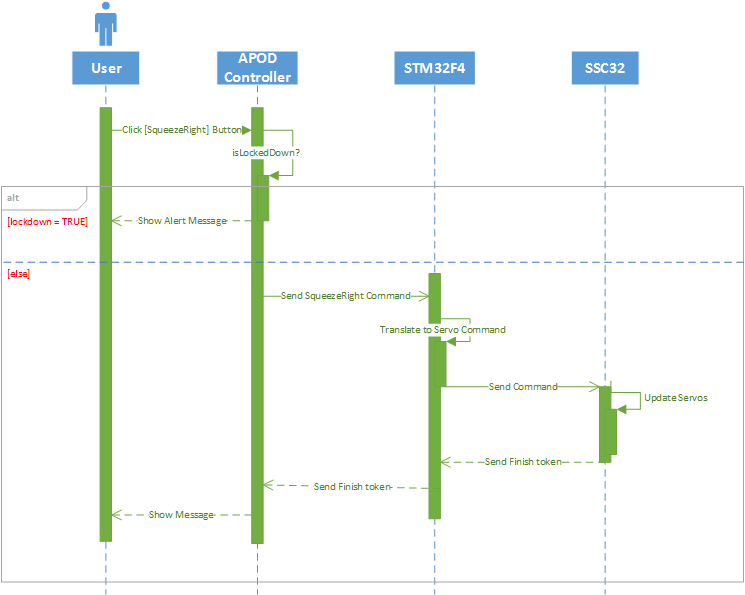


Figure D‑78

#### Object Tracking

##### Auto Grip

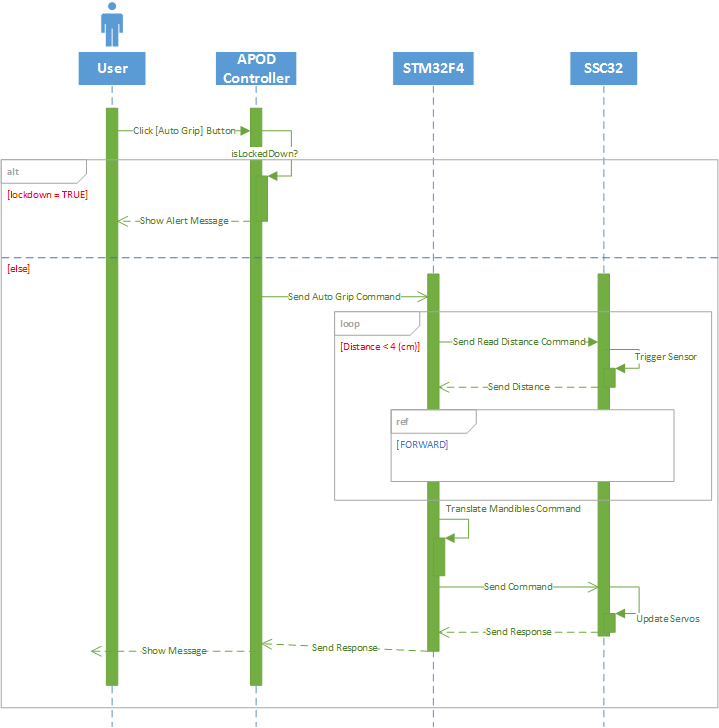


Figure D‑79

##### Choose target



Figure D‑80

##### Change Target

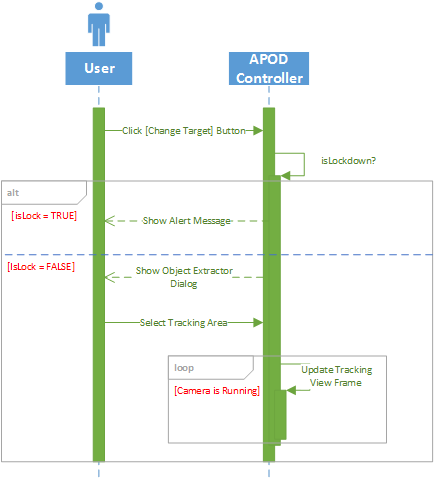


Figure D‑81

##### Lock Target

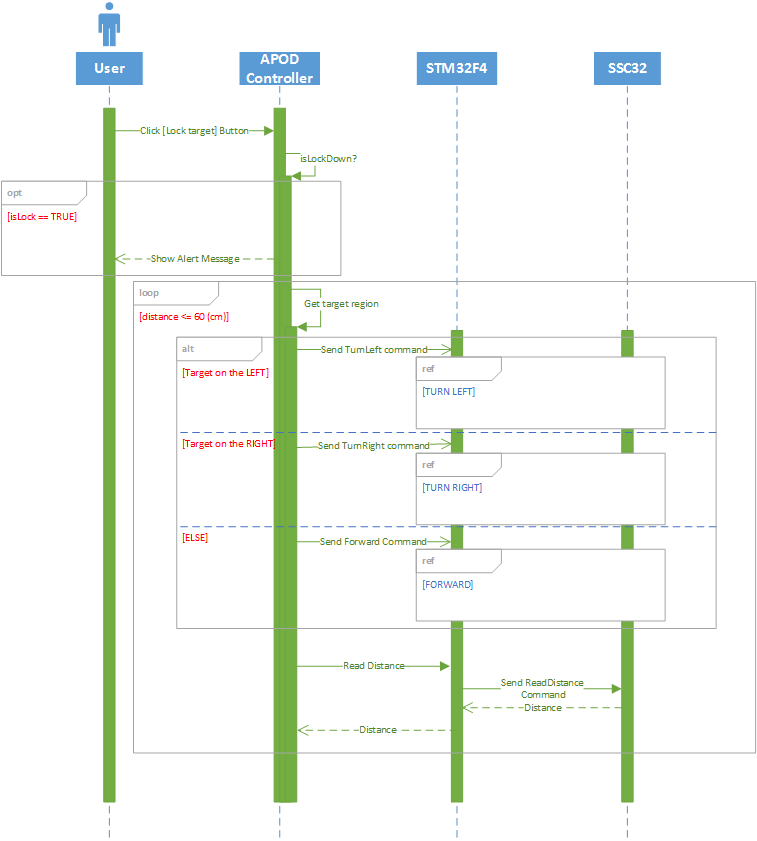


Figure D‑82

## USER INTERFACE DESIGN

### Live Control

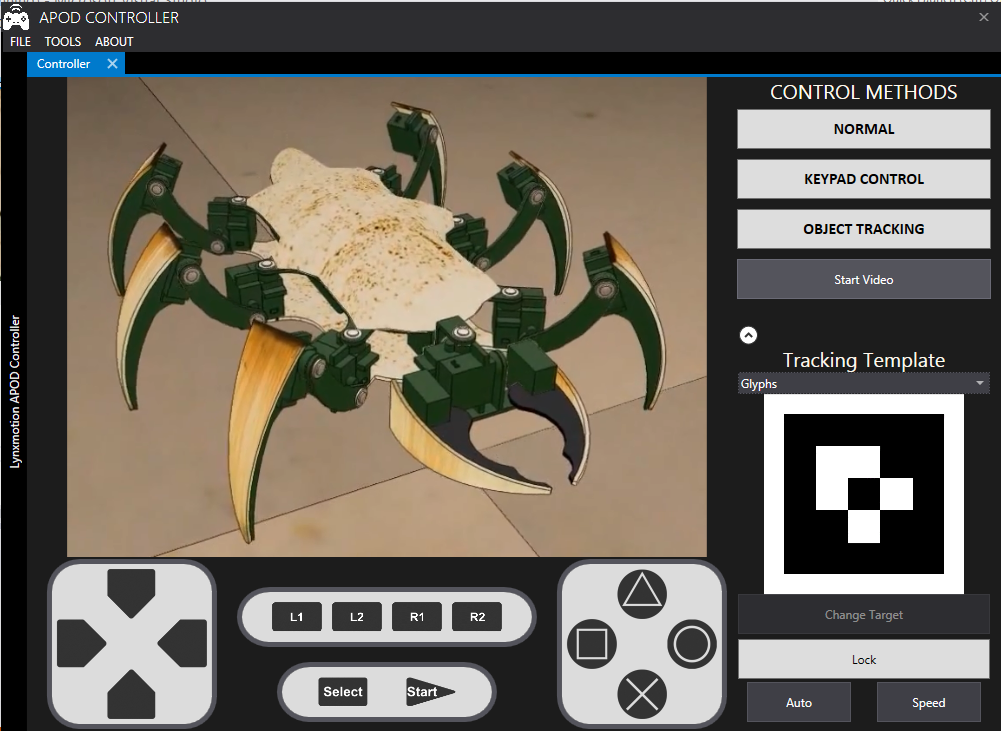


Figure D‑83 Live Control Interface Design

|  |  |  |
| --- | --- | --- |
| 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‑9 Live Control Interface Design Description

### Sequence Player

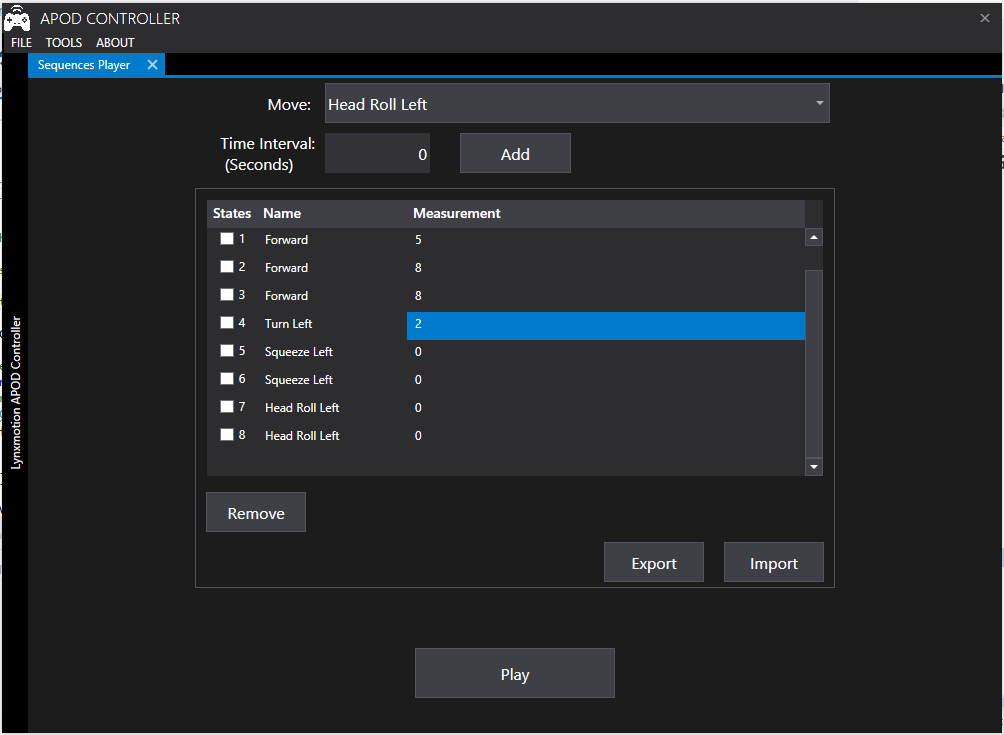


Figure D‑84 Sequence Player Interface Design

|  |  |  |
| --- | --- | --- |
| 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‑10 Sequence Player Interface Design Description

### Configuration

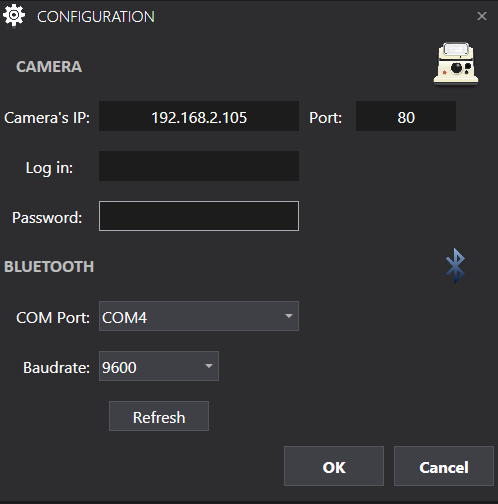


Figure D‑85 Configuration Interface Design

|  |  |  |
| --- | --- | --- |
| 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 baudrate selection |
| 6 | Button | Refresh list of Ports |
| 7 | Button | Accept information |
| 8 | Button | Cancel action |
|  |  |  |
|  |  |  |

Table D‑11 Configuration Interface Design Description

### Object Extractor

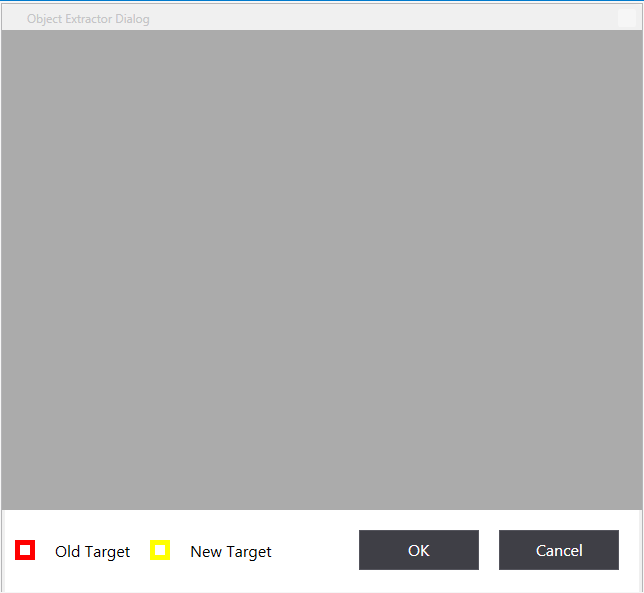


Figure D‑86 Object Extractor Interface Design

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

Table D‑12 Object Extractor Interface Design Description

# SOFTWARE TEST DOCUMENTATION (STD)

# SOFTWARE USER’S MANUAL (SUM)

# APPENDIX

## Glossary

## 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.). *ATmega328.* 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

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

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