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

**FPT UNIVERSITY**

|  |
| --- |
| Capstone Project Document |
| Designing a mobile robot navigation and  target tracking system   |  |  | | --- | --- | | **Group 2 – Embedded System** | | | **Group Members** | Bui Ha Duong Leader SE60772  Tran Vo Hoang Member SE60814  Truong Buu Hoang Duy Member SE60672  Duong Minh Tuan Member SE60642  Nguyen Huu Tai Member 60042 | | **Supervisor** | Tran Khanh Ninh | | **Ext Supervisor** | N/A | | **Capstone Project Code** | RNT | |

Ho Chi Minh City, 01/2014

# Introduction

## Scenario

Nowadays, the technology has the long steps if compared to the previous; it contributes to the development of science. From the initial idea is a mobile thing could help us in searching, detecting object(s) - the mobile robot navigation is a little part of the development contribution, and it could be used to search, find and relocate object in our environment. The one demand of science is to discovery, study new and strange place/ environments, there are many mobile robot used in this way like discovering harsh climate and dangerous place such as deep holes or in the deep of sea, new environment like Moon and Mars surfaces.



Figure ‑ Curiosity Rover reaching Mars

The mobile robot has to work on the harsh climate and dangerous place where people cannot or hard to work: oxygen concentration is low or very low, radiation is unidentified, large temperature fluctuation, disaster like sandstorm, sleet etc., damaged by external impact by living creature, environment element. Following these reason and requirement, making a new kind of mobile robot with main missions are navigating, detecting and tracking object(s) is very important.

Besides that, this technology will have many benefits in the modern life. With algorithms for navigating, detecting, tracking object(s) and path planning, we can make new autonomous tools that help us to do things, works. For example: Google’s Self-Driving car, Robot Vacuum Cleaner…



Figure ‑ Google’s Self-Driving Car



Figure ‑ Robot Vacuum Cleaner

With the interest in science, our team had decided to develop a mobile robot can help us in navigating and detecting object(s) with hope to contribute in the development of science.

## Existing Project



Figure ‑ Robot X1 - National University of Singapore



Figure ‑ the autonomous Urbie is designed for various urban operations, including military reconnaissance and rescue operations.



Figure ‑ Autonomous Robot use Microsoft Kinect sensor

**Main features:**

* Autonomously navigation from unknown outdoor environment to indoor environment and navigation to a specified indoor location while avoiding obstacles;
* Searching, identifying and engaging all pre-specified targets in an uncertain indoor environment before it returns to the starting point;
* Negotiate a staircase before the entrance of the building;
* Identifying, locating and manipulating elevator buttons in order to autonomously operate the elevator.

## Proposed Solution and Approach

Our team proposed an autonomous mobile robot system has some advantages like:

* Low cost
* Light weight
* Low power consuming
* Vision based
* Two-wheeled robot platform
* Comparatively small in size
* Easy implementation
* Respond quickly with commands
* Efficient real-time solution for tracking object
* Can perform with a high degree of autonomy, which is particularly desirable in fields such as space exploration, cleaning floors…
* Gain information about the environment
* Work for an extended period with/without human intervention
* Move either all or part of itself throughout its operating environment with/without human assistance

So that, we have our approach:

* **Hardware**: Based on quality and cost, we mainly use products of Texas Instruments.
* **Firmware**: We use the TIVA™ C Series TM4C123G LaunchPad for the mobile robot’s brain.
* **Software**: We use C# .NET and EmguCV library for Graphical User Interface (GUI) and Image Processing.

## Project Overview

### Hardware Overview

### Firmware Overview

### Software Overview

## Scope of Project

The scope of this project is a prototype of an “Autonomous Mobile Robot”, includes hardware, firmware and software.

**Hardware – Mobile Robot:**

* Creating a Two-wheeled mobile robot
* Creating an Encoder system for mobile robot
* Creating a Driver circuit for dc servo motor
* Creating a Multi-sensor navigation system for the autonomous mobile robot

**Firmware:**

* Developing a motion control system for two-wheeled mobile robot
  + Go Ahead
  + Go Back
  + Turn Left
  + Turn Right
* Developing PID Algorithm for speed control
* Developing Communication between mobile robot and software
* Developing Communication between mobile robot and sensors

**Software:**

* Developing an integrated software environment for mobile robot navigation and path planning.
* Applying Computer Vision for tracking object(s)
* Applying Computer Vision for obstacles detection and avoidance.
* Applying Computer Vision for navigation and path planning.

## Team Introduction

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Roles and Responsibility** | | | | |
| **No** | **Full Name** | **Role** | **Position** | **Contact** |
| 1 | Trần Khánh Ninh | Project Owner | Instructor | NinhTK@fpt.edu.vn |
| 2 | Bùi Hà Dương | Project Manager/Developer | Team leader | DuongBHSE60772@fpt.edu.vn |
| 3 | Trần Võ Hoàng | Developer/Tester | Team member | HoangTVSE60814@fpt.edu.vn |
| 4 | Trương Bửu Hoàng Duy | Developer/Tester | Team member | DuyTBHSE60672@fpt.edu.vn |
| 5 | Dương Minh Tuấn | Developer/Tester | Team member | TuanDMSE60642@fpt.edu.vn |
| 6 | Nguyễn Hữu Tài | Developer/Tester | Team member | TaiNH60042@fpt.edu.vn |

Table ‑ Team member and role in the project

# Software Project Management Plan

## Problem definition

### Name and main points of Capstone Project

**English**: Designing a mobile robot navigation and target tracking system

**Vietnamese**: Thiết kế hệ định vị và tìm đường đi cho robot di động

Students design one 2-wheels mobile robot, microcontroller board, program system to control the mobile robot, receive and process image to detect object(s), finding right way. Using difference kind of sensor such as: acceleration sensor, infrared sensor, supersonic sensor … for navigating, detecting…

### Problem Abstract

After many decades, there are a lot of methods to process image for detecting object(s) by analyzing the color, the shape, the edge, middle mass and blob method, etc. They have difference advantage and disadvantage so if we want to have a good result from image processing we have known what the methods are and how we could use them.

Edge detection is one of the fundamental steps in image processing, image analysis, image pattern recognition, and computer vision techniques. Edge detection is the name of a method which aims at identifying points in a digital image at which the image brightness changes sharply or discontinuities. The points at which image brightness changes sharply are typically organized into a set of curved line segments termed edges.

The colors in the object and the background should have a significant color difference in order to segment objects successfully using color based methods. The image is captured in 8-bit, unsigned integer, BGR format. In other words, captured images can be considered as 3 matrices, BLUE, RED and GREEN with integer values ranges from 0 to 255.

About the detecting object(s) and dodge the obstacles, there are many related methods such as VFF - Virtual Force Field and PFM - Potential Field Method etc. but they are not perfectly methods and also have problems, as PFM there are 4 significant problems that are inherent to PFMs and independent of the particular implementation:

1. Trap situations due to local minima (cyclic behavior).

2. No passage between closely spaced obstacles.

3. Oscillations in the presence of obstacles.

4. Oscillations in narrow passages.

With these methods and problems above, we have many works have to be done before building a system go with mobile robot.

### Project Overview

#### The Current System

There are many projects about mobile robot with many difference missions such as: navigating, finding and detecting, overcome off-road, discovering harsh climate place. With the high professional level, their shape is very suitable for mission but hard to complete other mission, fixed hardware is hard to change or upgrade so there might be their disadvantages.



Figure ‑ Adept Mobile Robots Seekur system



Figure ‑ Mobile Detection Assessment and Response System (MDARS)

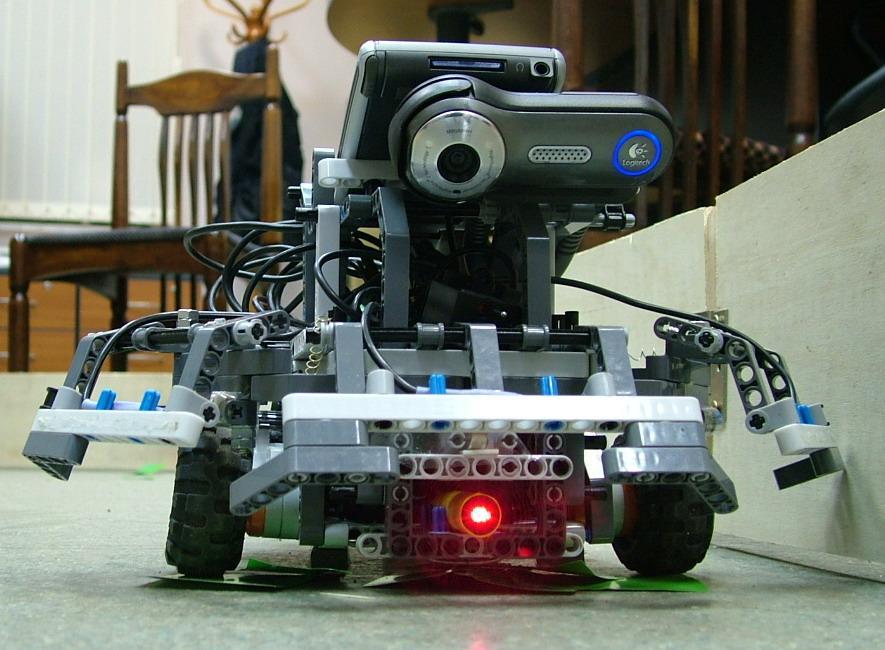


Figure ‑ Mobile robot Gnom No.9

#### The Proposed System

The system will use common and high performance hardware with good support from the manufacture. Hardware structure is easy to fix, change and upgrade without any damage. The mobility of robot and its shape would be suitable for more than one mission such as navigating, finding and detecting, tracking and chasing... etc.

#### Boundaries of the System

The mobile robot navigate by using compass sensor, infrared sensor for determine the direction.

Using camera for image processing, detect and track object(s).

Using infrared sensor, compass sensor to determine the direction to move and turn.

Using Bluetooth, supersonic to determine distance from robot to object(s), switch using 2 sensors for difference distance.

## 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 ‑ SPIRAL process model

### Roles and Responsibilities

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Full name | Team Role | Scrum Team Role | Responsibilities |
| 1 | Trần Khánh Ninh | Supervisor | Stake Holder | * Define business * Support in technical issues |
| 2 | Bùi Hà Dương | Team Leader | Product Owner, Team leader | * Managing process * Designing hardware * Building hardware * Clarifying requirements * Support technique * Prepare documents * Coding * GUI design * Create test case * Testing * Creating test plan and test cases |
| 3 | Trần Võ Hoàng | Team Member | Team Member | * Coding * Building hardware * Clarifying requirements * Support technique * Testing * Prepare documents |
| 4 | Trương Bửu Hoàng Duy | Team Member | Team Member | * Coding * Building hardware * Clarifying requirements * Support technique * Testing * Prepare documents |
| 5 | Dương Minh Tuấn | Team Member | Team Member | * Coding * Building hardware * Clarifying requirements * Support technique * Testing * Prepare documents |
| 6 | Nguyễn Hữu Tài | Team Member | Team Member | * Coding * Building hardware * Prepare documents * Clarifying requirements * Support technique * Testing |

Table ‑ Member’s roles

### Tools and Techniques

#### For Development

**Hardware environment:**

* Tiva C Series EK-TM4C123GXL LaunchPad



Figure ‑ Tiva™ C Series TM4C123G LaunchPad

* 6A Dual Brushed DC or Single Bipolar Stepper Motor Driver



Figure ‑ The DRV8412/32 are high performance, integrated dual full bridge motor drivers with an advanced protection system.

* Dual DC Motors



Figure ‑ Dual DC Motors

* Ultrasonic sensor HY-SRF05



Figure ‑ HY-SRF05

* Infrared sensor



Figure ‑ Infrared sensor

* Compass sensor



Figure ‑ Compass cmps03

* eZ430-RF2500 RF Module

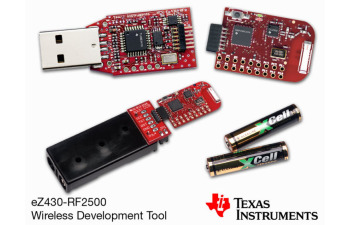


Figure ‑ RF transmitter & receiver

* Ip camera



Figure ‑ Sony Xperia Sola as an Ip Camera

**Software environment:**

* Microcontroller programming tools



Figure ‑ Code Composer Studio 5.4



Figure ‑ Visual Studio 2012

* Circuit supporting tools

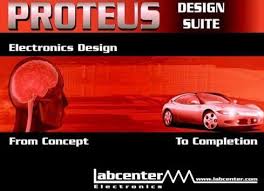


Figure ‑ Proteus 7.8

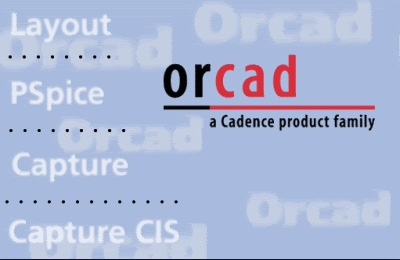


Figure ‑ Orcad 9.2

#### For Management and Documentations

* Task tracking



Figure ‑ Microsoft Project 2010

* Source and documents management



Figure ‑ Tortoise SVN

* Drawing diagrams



Figure ‑ StarUML

* Writing documents



Figure ‑ Microsoft Office 2010

#### For Communication

* Gmail, Facebook group for daily report and meeting minutes.
* Skype, Yahoo chat system

## Project management plan

### Tasks

#### 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 ‑ Task- Initiating & Planning

#### Software Requirements Analysis

|  |  |
| --- | --- |
| **Description** | Analyzing software requirements based on available embedded system and simulation 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 simulation hardware |

Table ‑ Task – Software Requirements Analysis

#### Creating Software Design Description

|  |  |
| --- | --- |
| **Description** | Designing the libraries interface to communicate between operating system and simulation hardware 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 ‑ Task – Creating Software Design Description

#### Coding

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

Table ‑ Task - 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 ‑ Task – Testing

#### Deployment

|  |  |
| --- | --- |
| **Description** | Deploying system include : a mobile robot navigation |
| **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 ‑ Task - Deployment

### Documentation

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

|  |  |  |
| --- | --- | --- |
| **REPORT NO.** | **DESCRIPTION** | **DELIVERY DATE** |
| 1 | Introduction | 2014.01.16 |
| 2 | Software Project Management Plan (SPM) | 2014.01.24 |
| Problem Definition |  |
| Project organization |  |
| Project management plan |  |
| Coding convention |  |
| 3 | Software Requirements Specifications (SRS) | 2014.02.05 |
| User Requirement Specification |  |
| System Requirement Specification (Specific Requirements) |  |
| External Interface Requirements |  |
| System Features |  |
| Use Case Diagram & Use Case Specification |  |
| Software System Attributes |  |
| 4 | Software Design Description (SDD) |  |
| Design Overview |  |
| System Architectural Design |  |
| Component Diagram |  |
| Detailed Description of Components |  |
| Sequence Diagram |  |
| User Interface Design |  |
| 5 | Software Test Documentation (STD) |  |
| Test Plan |  |
| Test Cases |  |
| Checklists |  |
| 6 | Software User’s Manual (SUM) |  |
| Installation Guide |  |
| User’s Guide |  |
| FINAL | Completed version of Report No.1 to Report No.6 – Soft copy |  |
| Completed version of Report No.1 to Report No.6 – Hard copy |  |

Table ‑ Report schedule

### Implementation

As Product Owner, I expect to have the CDs contain all source code and deployable mobile robot navigation and target tracking system at the end of project

Application must be developed and tested completely. Those feature need to be implemented

|  |  |  |
| --- | --- | --- |
| **MODULE** | **DESCRIPTION** | **DELIVERY** **DATE** |
| 0 | Built on Simulation hardware |  |
| 1 | Design board & Write controller Code |  |
| Control Hardware |  |
| Get sensors value |  |
| Get Feedback from sensors |  |
| Create Spec message definition |  |
| 2 | C code Control |  |
| UI |  |
| Interact & send message through UART |  |
| 3 | Move definition motor values to positions |  |
| 4 | Demo with deployable mobile robot navigation and  target tracking system problem |  |
| Algorithm |  |
| Object Tracking |  |
| Obstacles Avoidance |  |
| Path Planning |  |
| Final | CDs contain all source code and deployable mobile robot navigation and  target tracking system as final version |  |

Table ‑ Implement schedule

## Conventional rules

### Names

#### Naming convention

No name shall exceed 31 characters in length. Use standard abbreviations to shorten name length while maintaining understandability.

#### Function name

Usually every function performs an action, so the name should make clear what it does: check\_for\_errors() instead of error\_check(), dump\_data\_to\_file() instead of data\_file(). This will also make functions and data objects more distinguishable.

Structs are often nouns. By making function names verbs and following other naming conventions programs can be read more naturally.

Suffixes are sometimes useful:

* + *max* - to mean the maximum value something can have.
  + *cnt* - the current count of a running count variable.
  + *key* - key value.

For example: retry\_max to mean the maximum number of retries, retry\_cnt to mean the current retry count.

Prefixes are sometimes useful:

* *is* - to ask a question about something. Whenever someone sees *Is* they will know it's a question.
* *get* - get a value.
* *set* - set a value.

For example: is\_hit\_retry\_limit.

#### Variable name on the stack

Use all lower case letters.

Use '\_' as the word separator.

With this approach the scope of the variable is clear in the code.

Now all variables look different and are identifiable in the code.

**For example:**

int handle\_error (int error\_number) {

int error= OsErr();

Time time\_of\_error;

ErrorProcessor error\_processor;

}

#### Global Variables

Global variables should be prepended with a 'g\_'.

Global variables should be avoided whenever possible.

It's important to know the scope of a variable.

**For example:**

Logger g\_log;

Logger\* g\_plog;

#### Global constants

Global constants should be all caps with '\_' separators.

It's tradition for global constants to be named this way. You must be careful to not conflict with other global *#define*s and enum labels.

**For example:**

const int A\_GLOBAL\_CONSTANT= 5;

### Formatting

#### Array declaration

Example is not acceptable:

static uint8 digits[10] = {1,2,3,4,5,6,7,8,9,0};

While the next example is not acceptable:

static uint8 digits[20] = {0,1,2,3,4,5,6,7,8,9,10,

11, 12, 13, 14, 15, 16, 17, 18, 19 };

The preceding example shall be rewritten as:

static unit8 digits[20] =

{

0, 1, 2, 3, 4, 5, 6, 7, 8, 9,

10, 11, 12, 13, 14, 15, 16, 17, 18, 19

};

Multidimensional arrays shall accent their row and column initializations by explicitly enclosing each column’s values within its own set of braces. The following construct is required:

static uint32 salary\_tbl[3][5] =

{

{ 500, 550, 600, 650, 700 }, // ...

{ 600, 670, 740, 810, 880 }, // ...

{ 740, 840, 940, 1040, 1140 } // ...

};

While both of the following examples are not allowed:

static uint32 salary\_tbl[3][5] =

{

500, 550, 600, 650, 700,

600, 670, 740, 810, 880,

740, 840, 940, 1040, 1140

};

static uint32 salary\_tbl[3][5] =

{

500, 550, 600, 650, 700, 600, 670, 740, 810, 880, 740, ...

};

#### Conditional Expression

The use of the conditional expression ?: shall be discouraged. Testability requirements mandate that only one executable statement exists per line to allow for the setting of breakpoints on every executable statement during emulator debugging. The conditional expression ?: prevents the setting of breakpoints in this manner.

The construct:

memset( ccr-.line[FIRST\_DOT\_LOC],‘.’, num\_dots ? num\_dots : MAX\_DOTS );

num\_dots5 ( num\_dots, MAX\_DOTS ) ? num\_dots1 1:0;

Shall be rewritten similar to the following:

if ( num\_dots.5 MAX\_DOTS )

num\_dots5 0;

else

num\_dots11;

memset( ccr-.line[FIRST\_DOT\_LOC],‘.’, num\_dots );

#### When braces are need

All if, while and do statements must either have braces or be on a single line.

All if, while and do statements require braces even if there is only a single statement within the braces.

**For example:**

if (1 == somevalue) {

somevalue = 2;

}

It ensures that when someone adds a line of code later there are already braces and they don't forget.

#### Add comment to closing braces

Adding a comment to closing braces can help when you are reading code because you don't have to find the begin brace to know what is going on.

while(1) {

if (valid) {

} /\* if valid \*/

else {

} /\* not valid \*/

} /\* end forever \*/

#### Parens () with Key Words and Function Policy

Do not put parens next to keywords. Put a space between.

Do put parens next to function names.

Do not use parens in return statements when it's not necessary.

Keywords are not functions. By putting parens next to keywords keywords and function names are made to look alike.

**Example:**

if (condition){

}

while (condition){

}

strcpy(s, s1);

return 1;

#### A line should not exceed 78 Characters

Lines should not exceed 78 characters.

Even though with big monitors we stretch windows wide our printers can only print so wide. And we still need to print code.

The wider the window the fewer windows we can have on a screen. More windows is better than wider windows.

We even view and print diff output correctly on all terminals and printers.

#### One variable per line

Related to this is always define one variable per line:

Not:

char \*\*a, \*x;

Do:

char \*\*a = 0; /\* add doc \*/

char \*x = 0; /\* add doc \*/

The reasons are:

Documentation can be added for the variable on the line.

It's clear that the variables are initialized.

Declarations are clear which reduces the probability of declaring a pointer when you meant to declare just a char.

### Documentation

Consider your comments a story describing the system. Expect your comments to be extracted by a robot and formed into a man page. Class comments are one part of the story, method signature comments are another part of the story, method arguments another part, and method implementation yet another part. All these parts should weave together and inform someone else at another point of time just exactly what you did and why.

# Software Requirements Specification

## User Requirement Specifications

* Mobile Robot should be able to move freely in any direction.
* Controlling via a Serial connection like RF, or directly through a PS2 controller.
* Controller should be simple and easy.
* Mobile Robot can identify and detect objects, avoiding objects and find way.
* Mobile Robot can navigate according to a stationary object or motion (like a ball).
* Mobile Robot can find the way out of labyrinth.

## System Requirement Specifications

### 1. Hardware requirement

* Tiva C Series EK-TM4C123GXL LaunchPad
* 6A Dual Brushed DC or Single Bipolar Stepper Motor Driver
* Dual DC Motors
* Ultrasonic sensor HY-SRF05
* Infrared sensor
* Compass sensor
* eZ430-RF2500 RF Module
* IP camera
* PS2 controller

## Functional Requirement

### Communication protocols

N/A

### Use cases

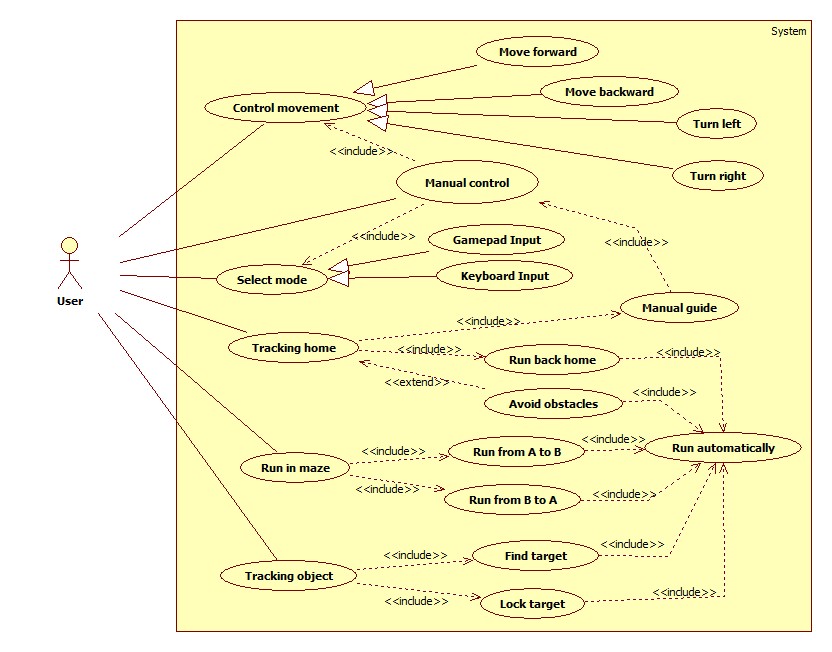


Figure ‑ Navigation mobile robot’s use cases

#### Move forward

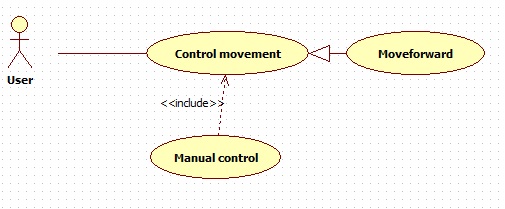


Figure ‑ Move forward use case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **MOVE FORWARD USE CASE SPECIFICATION** | | | | |
| **Use-case No.** | UC001 | **Use-case Version** | | 1.0 |
| **Use-case Name** | Move forward | | | |
| **Author** | HoangTV | | | |
| **Date** | 12/02/2014 | **Priority** | Normal | |
| **Actor:**  User  **Summary:**  This use case allows user controls the robot move forward.  **Goal:**  Robot moves forward.  **Triggers**  User push up the analog stick in gamepad mode or ↑ button in keyboard mode then release  **Preconditions:**  - Configuration is initiated  - Power is on.  - RF connection is established  **Post Conditions:**  -Success: Robot moves forward.  -Failure: Robot doesn’t move forward.  **Main Success Scenario:**   1. Power up/ Make sure power is on 2. Push up the analog stick or ↑ button in keyboard mode then release 3. Robot moves forward   **Alternative Scenario:**  N/A  **Exceptions :**  N/A  **Relationships:**  Included case Manual control. | | | | |

Table ‑ Move forward use case specification

#### Move backward

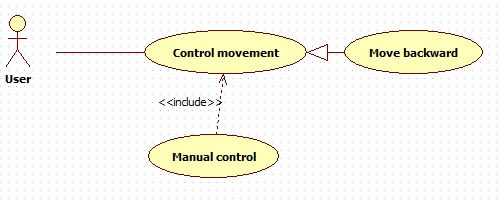


Figure ‑ Move backward use case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **MOVE BACKWARD USE CASE SPECIFICATION** | | | | |
| **Use-case No.** | UC002 | **Use-case Version** | | 1.0 |
| **Use-case Name** | Move backward | | | |
| **Author** | HoangTV | | | |
| **Date** | 12/02/2014 | **Priority** | Normal | |
| **Actor:**  User  **Summary:**  This use case allows user controls the robot move backward.  **Goal:**  Robot moves backward.  **Triggers**  User push down the analog stick in gamepad mode or ↓ button in keyboard mode then release  **Preconditions:**  - Configuration is initiated  - Power is on.  - RF connection is established  **Post Conditions:**  -Success: Robot moves backward.  -Failure: Robot doesn’t move backward.  **Main Success Scenario:**   1. Power up/ Make sure power is on 2. Push down the analog stick or ↓ button in keyboard mode then release 3. Robot moves backward   **Alternative Scenario:**  N/A  **Exceptions :**  N/A  **Relationships:**  Included case Manual control. | | | | |

Table ‑ Move backward use case specification

#### Turn left

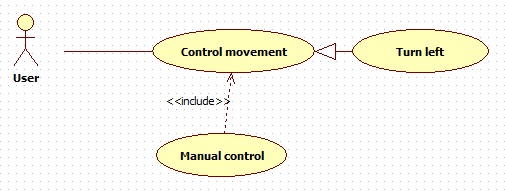


Figure ‑ Turn left use case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TURN LEFT USE CASE SPECIFICATION** | | | | |
| **Use-case No.** | UC003 | **Use-case Version** | | 1.0 |
| **Use-case Name** | Turn left | | | |
| **Author** | HoangTV | | | |
| **Date** | 12/02/2014 | **Priority** | Normal | |
| **Actor:**  User  **Summary:**  This use case allows user controls the robot turn left.  **Goal:**  Robot turns left.  **Triggers**  User pushes the analog stick to the left in gamepad mode or ← button in keyboard mode then release  **Preconditions:**  - Configuration is initiated  - Power is on.  - RF connection is established  **Post Conditions:**  -Success: Robot turns left.  -Failure: Robot doesn’t turn left.  **Main Success Scenario:**   1. Power up/ Make sure power is on 2. Push the analog stick to the left in gamepad mode or ← button in keyboard mode then release 3. Robot turns left   **Alternative Scenario:**  N/A  **Exceptions :**  N/A  **Relationships:**  Included case Manual control. | | | | |

Table ‑ Turn left use case specification

#### Turn right

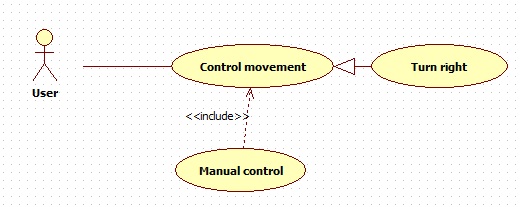


Figure ‑ Turn right use case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TURN RIGHT USE CASE SPECIFICATION** | | | | |
| **Use-case No.** | UC004 | **Use-case Version** | | 1.0 |
| **Use-case Name** | Turn right | | | |
| **Author** | HoangTV | | | |
| **Date** | 12/02/2014 | **Priority** | Normal | |
| **Actor:**  User  **Summary:**  This use case allows user controls the robot turn right.  **Goal:**  Robot turns right.  **Triggers**  User pushes the analog stick to the right in gamepad mode or → button in keyboard mode then release  **Preconditions:**  - Configuration is initiated  - Power is on.  - RF connection is established  **Post Conditions:**  -Success: Robot turns right.  -Failure: Robot doesn’t turn right.  **Main Success Scenario:**   1. Power up/ Make sure power is on 2. Push the analog stick to the right right in gamepad mode or → button in keyboard mode then release 3. Robot turns right   **Alternative Scenario:**  N/A  **Exceptions :**  N/A  **Relationships:**  Included case Manual control. | | | | |

Table ‑4 Turn right use case specification

#### Gamepad Input

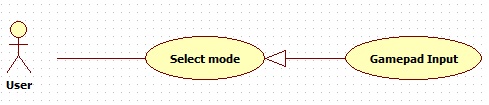


Figure ‑ Gamepad Input use case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **GAMEPAD INPUT USE CASE SPECIFICATION** | | | | |
| **Use-case No.** | UC005 | **Use-case Version** | | 1.0 |
| **Use-case Name** | Gamepad input | | | |
| **Author** | HoangTV | | | |
| **Date** | 12/02/2014 | **Priority** | Normal | |
| **Actor:**  User  **Summary:**  Using gamepad input method.  **Goal:**  Gamepad input mode is selected.  **Triggers**  User select Gamepad mode.  **Preconditions:**  - Configuration is initiated  - Power on  **Post Conditions:**  N/A  **Main Success Scenario:**   1. User select manual mode. 2. User select gamepad input mode 3. Establish RF connection.   **Alternative Scenario:**  N/A  **Exceptions :**  N/A  **Relationships:**  Included case Manual control. | | | | |

Table ‑ Gamepad Input use case specification

#### Keyboard Input

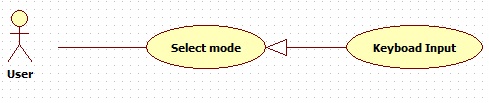


Figure ‑ Keyboard Input use case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **KEYBOARD INPUT USE CASE SPECIFICATION** | | | | |
| **Use-case No.** | UC006 | **Use-case Version** | | 1.0 |
| **Use-case Name** | Keyboard input | | | |
| **Author** | HoangTV | | | |
| **Date** | 12/02/2014 | **Priority** | Normal | |
| **Actor:**  User  **Summary:**  Using keyboard input method.  **Goal:**  Keyboard input mode is selected.  **Triggers**  User select Keyboard mode.  **Preconditions:**  - Configuration is initiated  - Power on  **Post Conditions:**  N/A  **Main Success Scenario:**   1. User select manual mode. 2. User select keyboard input mode 3. Establish RF connection.   **Alternative Scenario:**  N/A  **Exceptions :**  N/A  **Relationships:**  Included case Manual control. | | | | |

Table ‑ Keyboard Input use case specification

#### Find target

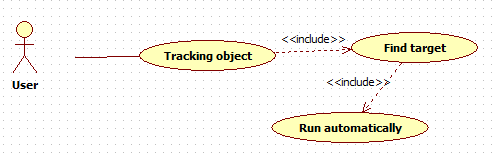


Figure ‑ Find target use case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **FIND TARGET USE CASE SPECIFICATION** | | | | |
| **Use-case No.** | UC007 | **Use-case Version** | | 1.0 |
| **Use-case Name** | Find target | | | |
| **Author** | DuyTBH | | | |
| **Date** | 13/02/2014 | **Priority** | Normal | |
| **Actor:**  User  **Summary:**  Enter tracking object mode, find target to track.  **Goal:**  The mobile robot finds exactly where the target is and toward the target  **Triggers**  Enter tracking object mode  **Pre-conditions:**  - Configuration is initiated  - RF connection is established  - IP camera stream is working  **Post-conditions:**  - The mobile robot stops turn left when the target is found  **Main Success Scenario:**   1. The mobile robot standing still and turning left until it finds the target 2. When the target is found by IP camera, the mobile robot toward the target   **Alternative Scenario:**  N/A  **Exceptions:**  N/A  **Relationships:**  N/A | | | | |

Table ‑ Find target use case specification

#### Lock target

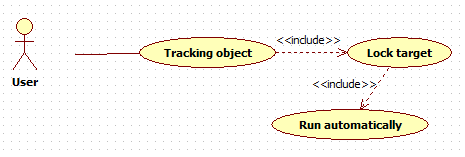


Figure ‑ Lock target use case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **LOCK TARGET USE CASE SPECIFICATION** | | | | |
| **Use-case No.** | UC008 | **Use-case Version** | | 1.0 |
| **Use-case Name** | Lock target | | | |
| **Author** | DuyTBH | | | |
| **Date** | 13/02/2014 | **Priority** | Normal | |
| **Actor:**  User  **Summary:**  Start tracking target  **Goal:**  The mobile robot runs closer to the target (within 20cm).  **Triggers**  Target is found  **Pre-conditions:**  - Configuration is initiated  - RF connection is established  - IP camera stream is working  - Target is found  **Post-conditions:**  N/A  **Main Success Scenario:**   1. The mobile robot runs toward the target. 2. The mobile robot stops when the distance between it and the target was about 20 cm.   **Alternative Scenario:**  N/A  **Exceptions:**  N/A  **Relationships:**  N/A | | | | |

Table ‑ Lock target use case specification

#### Manual guide

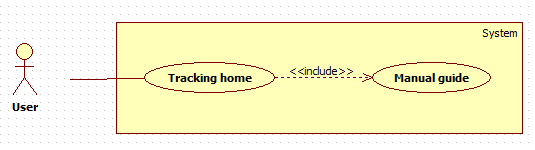


Figure ‑ Manual guide use case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **MANUAL GUIDE USE CASE SPECIFICATION** | | | | |
| **Use-case No.** | UC009 | **Use-case Version** | | 1.0 |
| **Use-case Name** | Manual guide | | | |
| **Author** | DuyTBH | | | |
| **Date** | 13/02/2014 | **Priority** | Normal | |
| **Actor:**  User  **Summary:**  Enter home tracking mode  **Goal:**  Directions for the mobile robot to go from point A to point B by manual control.  **Triggers**  Enter home tracking mode  **Pre-conditions:**  - Configuration is initiated  - RF connection is established  - IP camera stream is working  - Controller mode is established (keyboard, gamepad)  **Post-conditions:**  N/A  **Main Success Scenario:**   1. The mobile robot running closer to point B. 2. After the completion of the way from A to B, the mobile robot stops at point B and remember the way back to point A.   **Alternative Scenario:**  N/A  **Exceptions:**  N/A  **Relationships:**  N/A | | | | |

Table ‑ Manual guide use case specification

#### Run back home

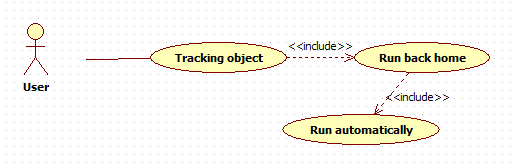


Figure ‑ Run back home use case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **RUN BACK HOME USE CASE SPECIFICATION** | | | | |
| **Use-case No.** | UC010 | **Use-case Version** | | 1.0 |
| **Use-case Name** | Run back home | | | |
| **Author** | DuyTBH | | | |
| **Date** | 13/02/2014 | **Priority** | Normal | |
| **Actor:**  User  **Summary:**  Home tracking starts  **Goal:**  The mobile robot automatically goes back to home (from point B back to point A)  **Triggers**    **Pre-conditions:**  - Configuration is initiated  - RF connection is established  - IP camera stream is working  - Manual guide is completed (the way from point A to point B is finished)  **Post-conditions:**  N/A  **Main Success Scenario:**   1. From point B, the mobile robot running closer to point A 2. After the completion of the way back, the mobile robot will stop exactly at point A   **Alternative Scenario:**  N/A  **Exceptions:**  N/A  **Relationships:**  N/A | | | | |

Table ‑ Automatically back home use case specification

#### Avoid obstacles

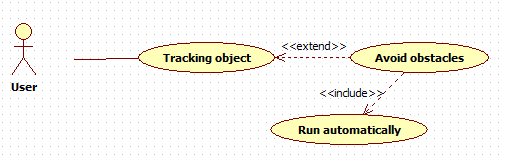


Figure ‑ Avoid obstacles use case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **AVOID OBSTACLES USE CASE SPECIFICATION** | | | | |
| **Use-case No.** | UC011 | **Use-case Version** | | 1.0 |
| **Use-case Name** | Avoid obstacles | | | |
| **Author** | DuyTBH | | | |
| **Date** | 13/02/2014 | **Priority** | Normal | |
| **Actor:**  User  **Summary:**  Home tracking starts  **Goal:**  On the way go back to home (from point B back to point A), the mobile robot avoids obstacles automatically.  **Triggers**  Obstacles appear on the way back home  **Pre-conditions:**  - Configuration is initiated  - RF connection is established  - IP camera stream is working  - Manual guide is completed (the way from point A to point B is finished)  - Ultrasonic sensor is ready.  **Post-conditions:**  N/A  **Main Success Scenario:**   1. From point B, the mobile robot running closer to point A 2. If obstacles appear, the mobile robot will avoid and keep going back to home 3. After the completion of the way back (without touching any obstacles), the mobile robot will stop exactly at point A   **Alternative Scenario:**  N/A  **Exceptions:**  N/A  **Relationships:**  N/A | | | | |

Table ‑ Avoid obstacles use case specification

#### Run in maze (from A to B)

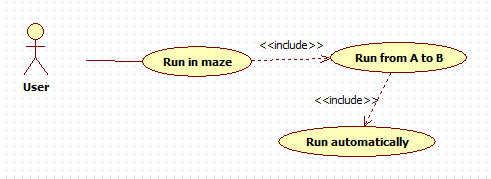


Figure ‑ Run in maze (from A to B) use case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **AVOID OBSTACLES USE CASE SPECIFICATION** | | | | |
| **Use-case No.** | UC012 | **Use-case Version** | | 1.0 |
| **Use-case Name** | Run in maze (from A to B) | | | |
| **Author** | DuyTBH | | | |
| **Date** | 13/02/2014 | **Priority** | Normal | |
| **Actor:**  User  **Summary:**  Run in maze mode starts  **Goal:**  The robot run automatically in a maze from point A (bottom left corner) to point B (top right corner)  **Triggers**  Press button on the robot  **Pre-conditions:**  - Configuration is initiated  - Power is on  - Infrared sensor is ready.  **Post-conditions:**  N/A  **Main Success Scenario:**   1. In a maze, from start point A (bottom left corner) the mobile robot running closer to point B (top right corner) 2. After the completion of the way from A to B (without touching any edges in the maze), the mobile robot will stop exactly at point B   **Alternative Scenario:**  N/A  **Exceptions:**  N/A  **Relationships:**  N/A | | | | |

Table ‑ Run in maze (from A to B) use case specification

#### Run in maze (from B to A)

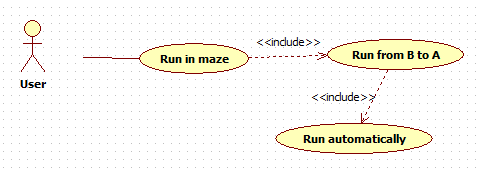


Figure ‑ Run in maze (from B to A) use case diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **AVOID OBSTACLES USE CASE SPECIFICATION** | | | | |
| **Use-case No.** | UC013 | **Use-case Version** | | 1.0 |
| **Use-case Name** | Run in maze (from B to A) | | | |
| **Author** | DuyTBH | | | |
| **Date** | 13/02/2014 | **Priority** | Normal | |
| **Actor:**  User  **Summary:**  Run in maze mode starts  **Goal:**  The robot run automatically in a maze from point B (top right corner) back to start point A (bottom left corner)  **Triggers**  When robot reach point B (top right corner)  **Pre-conditions:**  - Configuration is initiated  - Power is on  - Infrared sensor is ready  - Robot reach point B (top right corner)  **Post-conditions:**  N/A  **Main Success Scenario:**   1. In a maze, from point B (top right corner) the mobile robot running closer to start point A (bottom left corner) 2. After the completion of the way back to start point A (without touching any edges in the maze), the mobile robot will stop exactly at point A   **Alternative Scenario:**  N/A  **Exceptions:**  N/A  **Relationships:**  N/A | | | | |

Table ‑ Run in maze (from B to A) use case specification

## Non-Functional Requirement

### Reliability

* All Controller Applications are easily to upgrade firmware.
* All Controller Applications can be replaced easily by loaded into chip if the controller has problem.
* All Controller Applications are guarantee by quality testing in:
* Stability constraints.
* Functionality.
* Reliability.
* It’s mostly depending on hardware reliability.

### Availability

* The embedded kit is everywhere to buy, or can buy ones which have similar function.

### Security

* N/A

### Maintainability

* Easy to upgrade or replace hardware.
* Easy to update software.

### Portability

* The embedded kit is small to be moved easily.

### Performance

* N/A

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