**MINISTRY OF EDUCATION AND TRAINING**

**FPT UNIVERSITY**

Capstone Project Document

**Design and implementation of**

**Products Sorting System based on color**

|  |  |
| --- | --- |
| **Group 01** | |
| **Group members** | Nguyễn Hùng Mạnh – Team Leader – SE60893  Nguyễn Khang Hy – Team Member – SE61100  Nguyễn Nhật Linh – Team Member – SE61040  Phạm Nhật Hưng – Team Member – SE61119 |
| **Supervisor** | Mr. Nguyễn Đức Lợi |
| **Ext. Supervisor** | N/A |
| **Capstone Project code** | PSSC |

-Ho Chi Minh City, 11/05/2015-

*This page is intentionally left blank*

Table of Contents

[Table of Contents 3](#_Toc427377817)

[List of Tables 6](#_Toc427377818)

[Definitions, Acronyms and Abbreviations 10](#_Toc427377819)

[A. Report No.1 Introduction 11](#_Toc427377820)

[1. Project Information 11](#_Toc427377821)

[2. Introduction 11](#_Toc427377822)

[3. Current Situation 11](#_Toc427377823)

[4. Problem Definitions 11](#_Toc427377824)

[5. Proposed Solution 12](#_Toc427377825)

[5.1 Feature functions 12](#_Toc427377826)

[5.2 Advantages and disadvantages 12](#_Toc427377827)

[6. Functional Requirements 13](#_Toc427377828)

[6.1 Detect color of product 13](#_Toc427377829)

[6.2 Sorting product based on color 13](#_Toc427377830)

[6.3 Filter and get product by color and quantity 13](#_Toc427377831)

[6.4 Manage information of product 13](#_Toc427377832)

[7. Role and Responsibility 13](#_Toc427377833)

[B. Report No.2 Software Project Management Plan 14](#_Toc427377834)

[1. Problem Definition 14](#_Toc427377835)

[1.1 Name of this Capstone Project 14](#_Toc427377836)

[1.2 Problem Abstract 14](#_Toc427377837)

[1.3 Project Overview 14](#_Toc427377838)

[2. Project organization 18](#_Toc427377839)

[2.1 Software Process Model 18](#_Toc427377840)

[2.2 Role and responsibilities 18](#_Toc427377841)

[2.3 Tools and Techniques 19](#_Toc427377842)

[3. Project Management Plan 19](#_Toc427377843)

[3.1 Software development life cycle 20](#_Toc427377844)

[3.2 Phase Detail 20](#_Toc427377845)

[3.3 All Meeting Minutes 22](#_Toc427377846)

[4. Coding Convention 22](#_Toc427377847)

[C. Report No. 3 Software Requirement Specification 23](#_Toc427377848)

[1. User Requirement Specification 23](#_Toc427377849)

[2. System Requirement Specification 23](#_Toc427377850)

[2.1 External Interface Requirement 23](#_Toc427377851)

[2.2 System Overview Use Case 37](#_Toc427377852)

[2.3 List of Use Case 37](#_Toc427377853)

[3. Software System Attribute 48](#_Toc427377854)

[3.1 Usability 48](#_Toc427377855)

[3.2 Reliability 48](#_Toc427377856)

[3.3 Availability 49](#_Toc427377857)

[3.4 Security 49](#_Toc427377858)

[3.5 Maintainability 49](#_Toc427377859)

[3.6 Portability 49](#_Toc427377860)

[3.7 Performance 49](#_Toc427377861)

[4. Conceptual Diagram 49](#_Toc427377862)

[D. Report No.4 Software Design Description 51](#_Toc427377863)

[1. Design Overview 51](#_Toc427377864)

[2. System Architectural Design 52](#_Toc427377865)

[2.1 Hardware Program Architecture description 52](#_Toc427377866)

[2.2 Desktop Application Architecture description 53](#_Toc427377867)

[3. Component Diagram 53](#_Toc427377868)

[4. Detailed Description 54](#_Toc427377869)

[4.1 Class Diagram 54](#_Toc427377870)

[4.2 Class Diagram Explanation 55](#_Toc427377871)

[4.3 Flowchart Diagram 60](#_Toc427377872)

[4.4 Sequence Diagram 64](#_Toc427377873)

[4.5 Activity Diagram 68](#_Toc427377874)

[5. User Interface Design 77](#_Toc427377875)

[5.1 Repository Interface Design 77](#_Toc427377876)

[5.2 Report Interface Design 80](#_Toc427377877)

[5.3 Configuration Interface Design 82](#_Toc427377878)

[6. Database Design 83](#_Toc427377879)

[6.1 Logical Diagram 83](#_Toc427377880)

[6.2 Data Dictionary 83](#_Toc427377881)

[7. Algorithms 85](#_Toc427377882)

[7.1 K – Nearest Neighbor Algorithms 85](#_Toc427377883)

[E. System Implementation & Test 89](#_Toc427377884)

[1. Introduction 89](#_Toc427377885)

[1.1 Overview 89](#_Toc427377886)

[1.2 Test Approach 89](#_Toc427377887)

[2. Database Relationship Diagram 89](#_Toc427377888)

[2.1 Physical Diagram 89](#_Toc427377889)

[2.2 Data Dictionary 89](#_Toc427377890)

[3. Performance Measures 91](#_Toc427377891)

[4. Test Plan 91](#_Toc427377892)

[4.1 Features to be tested 91](#_Toc427377893)

[4.2 Features not to be tested 92](#_Toc427377894)

[5. System Testing Test Case 93](#_Toc427377895)

[5.1 Component Testing 93](#_Toc427377896)

[5.2 Integration Testing 104](#_Toc427377897)

[F. Software User’s Manual 121](#_Toc427377898)

[1. Installation Guide 121](#_Toc427377899)

[1.1 Setting up environment 121](#_Toc427377900)

[1.2 Deployment 121](#_Toc427377901)

[2. User Guide 122](#_Toc427377902)

[2.1 Desktop Application Guide 122](#_Toc427377903)

[2.2 Machine Simulator Guide 130](#_Toc427377904)

[G. Appendix 132](#_Toc427377905)

List of Tables

[Table 1: Definitions 10](#_Toc427377906)

[Table 2: Roles and Responsibilities 13](#_Toc427377907)

[Table 3: Hardware requirements for PC (or laptop0 Desktop Application 17](#_Toc427377908)

[Table 4: Roles and Responsibilities 19](#_Toc427377909)

[Table 5: Phase 1: Requirement Analysis 21](#_Toc427377910)

[Table 6: Phase 2: Design 21](#_Toc427377911)

[Table 7: Phase 3: Implementation 21](#_Toc427377912)

[Table 8: Phase 4: Testing 22](#_Toc427377913)

[Table 9: Phase 5: Maintenance 22](#_Toc427377914)

[Table 10: Block Diagram – Overview 25](#_Toc427377915)

[Table 11: Block Diagram – Input Marble. 26](#_Toc427377916)

[Table 12: Block Diagram – Output Marble 27](#_Toc427377917)

[Table 13: Adruino Mega 2560 - Specification 28](#_Toc427377918)

[Table 14: TCS3200 - Input and Output Pins 30](#_Toc427377919)

[Table 15: TCS3200 - Select output frequency 30](#_Toc427377920)

[Table 16: TCS3200 - Select Photodiode Type 30](#_Toc427377921)

[Table 17: Infrared Sensor - Input and Output Pins 31](#_Toc427377922)

[Table 18: Infrared Sensors connect to Arduino Mega 32](#_Toc427377923)

[Table 19: Servo MG996R - Specification 33](#_Toc427377924)

[Table 20: Servo MG90S - Specification 33](#_Toc427377925)

[Table 21: Servo – Input and Output Pins 33](#_Toc427377926)

[Table 22: Servos connect to Arduino Mega 34](#_Toc427377927)

[Table 23: 12V DC Motor GA-37 - Specification 35](#_Toc427377928)

[Table 24: L298 Dual H-Bridge - Specification 36](#_Toc427377929)

[Table 25: 12A DC-to\_DC step down - Specification 37](#_Toc427377930)

[Table 26: <User> Input Marble Use Case Diagram 40](#_Toc427377931)

[Table 27: <User> Output Marble Use Case Diagram 43](#_Toc427377932)

[Table 28: <User> View Order Detail Use Case Diagram 46](#_Toc427377933)

[Table 29: Configure Ports Use Case Diagram 48](#_Toc427377934)

[Table 30: Component Dictionary 54](#_Toc427377935)

[Table 31: <Desktop App> Marble Attribute 56](#_Toc427377936)

[Table 32: <Desktop App> Marble Method 56](#_Toc427377937)

[Table 33: <Desktop App> Repository Attribute 56](#_Toc427377938)

[Table 34: <Desktop App> Repository Method 56](#_Toc427377939)

[Table 35: <Desktop App> Order Attribute 57](#_Toc427377940)

[Table 36: <Desktop App> Order Method 57](#_Toc427377941)

[Table 37: <Desktop App> OrderDetail Attribute 57](#_Toc427377942)

[Table 38: <Desktop App> OrderDetail Method 57](#_Toc427377943)

[Table 39: <Desktop App> PortSetting Attribute 57](#_Toc427377944)

[Table 40: <Desktop App> PortSetting Method 57](#_Toc427377945)

[Table 41: < K – Nearest Neighbor > KSelection Attribute 58](#_Toc427377946)

[Table 42: < K – Nearest Neighbor > KSelection Method 58](#_Toc427377947)

[Table 43: < K – Nearest Neighbor > ColorAnalysisItem Attribute 58](#_Toc427377948)

[Table 44: < K – Nearest Neighbor > ColorAnalysisItem Method 58](#_Toc427377949)

[Table 45: < K – Nearest Neighbor > ColorAnalysisTable Attribute 58](#_Toc427377950)

[Table 46: < K – Nearest Neighbor > ColorAnalysisTable Method 58](#_Toc427377951)

[Table 47: < K – Nearest Neighbor > ColorDetection Attribute 59](#_Toc427377952)

[Table 48: < K – Nearest Neighbor > ColorDetection Method 59](#_Toc427377953)

[Table 49: < K – Nearest Neighbor > Global Attribute 59](#_Toc427377954)

[Table 50: <Hardware> Input Marble Details Activity Diagram 69](#_Toc427377955)

[Table 51: <Hardware> Output Marble Details Activity Diagram 71](#_Toc427377956)

[Table 52: <User> Input Marble Activity Diagram 72](#_Toc427377957)

[Table 53: <User> Output Marble Activity Diagram 74](#_Toc427377958)

[Table 54: <User> View Report Details 75](#_Toc427377959)

[Table 55: <User> Configure Ports Activity Diagram 76](#_Toc427377960)

[Table 56: <Fields> Manage Repository Interface 78](#_Toc427377961)

[Table 57: < Button/Hyperlinks> Manage Repository Interface 79](#_Toc427377962)

[Table 58: <Fields> Create Customer Profile Interface 79](#_Toc427377963)

[Table 59: < Button/Hyperlinks> Create Customer Profile Interface 80](#_Toc427377964)

[Table 60: <Fields> Search Customer Order Interface 81](#_Toc427377965)

[Table 61: < Button/Hyperlinks> Search Customer Order Interface 81](#_Toc427377966)

[Table 62: < Button/Hyperlinks> View Order Details Interface 81](#_Toc427377967)

[Table 63: <Fields> Configure Ports Interface 82](#_Toc427377968)

[Table 64: < Button/Hyperlinks> Configure Ports Interface 82](#_Toc427377969)

[Table 65: Entity Data Dictionary 84](#_Toc427377970)

[Table 66: Entity Attribute Data Dictionary 85](#_Toc427377971)

[Table 67: Color Sample of Marbles 86](#_Toc427377972)

[Table 68: New Sample or Query Point 86](#_Toc427377973)

[Table 69: Distance between Query Point and Color Sample 87](#_Toc427377974)

[Table 70: Table Data dictionary 90](#_Toc427377975)

[Table 71: Table Attribute Data Dictionary 91](#_Toc427377976)

[Table 72: TCS3200 Color Sensor Testing 94](#_Toc427377977)

[Table 73: Servos Testing 100](#_Toc427377978)

[Table 74: Summary: Position of each servo 101](#_Toc427377979)

[Table 75: Infrared Sensors Testing 104](#_Toc427377980)

[Table 76: Input Marble Testing 114](#_Toc427377981)

[Table 77: Output Marble Testing 119](#_Toc427377982)

[Table 78: Export Order Details Testing 120](#_Toc427377983)

[Table 79: <Desktop App Guide> Configures Ports Steps 123](#_Toc427377984)

[Table 80: <Desktop App Guide> Input Marble Steps 125](#_Toc427377985)

[Table 81: <Desktop App Guide> Output Marble Steps 127](#_Toc427377986)

[Table 82: <Desktop App Guide> View Order Details steps 128](#_Toc427377987)

[Table 83: <Desktop App Guide> Export Order Details Step. 129](#_Toc427377988)

[Table 84: <Machine Simulator> Configure Ports Steps 130](#_Toc427377989)

[Table 85: <Machine Simulator> Input Marble Steps 131](#_Toc427377990)

List of Figures

[Figure 1: Diameter of marble 16](#_Toc427377991)

[Figure 2: The colors of marbles used as products 16](#_Toc427377992)

[Figure 3: Boundaries of the system 17](#_Toc427377993)

[Figure 4: Waterfall development model 18](#_Toc427377994)

[Figure 5: Block Diagram – Overview 23](#_Toc427377995)

[Figure 6: Block Diagram – Input Marble 26](#_Toc427377996)

[Figure 7: Block Diagram – Output Marble 27](#_Toc427377997)

[Figure 8: <Hardware> Arduino Mega 2560 28](#_Toc427377998)

[Figure 9: <Hardware> Color Sensor TCS3200 29](#_Toc427377999)

[Figure 10: <Hardware> Block Diagram of Color Sensor TCS3200 29](#_Toc427378000)

[Figure 11: TCS3200 connects to Arduino. 30](#_Toc427378001)

[Figure 12: <Hardware> Infrared Sensor V1 31](#_Toc427378002)

[Figure 13: Infrared Sensor returns bit one 31](#_Toc427378003)

[Figure 14: Infrared Sensor returns bit zero 32](#_Toc427378004)

[Figure 15: <Hardware> Servo Tower Pro MG996R. 32](#_Toc427378005)

[Figure 16: <Hardware> Servo Tower Pro MG90S 33](#_Toc427378006)

[Figure 17: <Hardware> Servo - Move based on PPM 34](#_Toc427378007)

[Figure 18: <Hardware> 12V DC Motor GA37 35](#_Toc427378008)

[Figure 19: <Hardware> L298 Dual H-Bridge Motor controller 35](#_Toc427378009)

[Figure 20: <Hardware> L298 Dual H-Bridge - Principle diagram 36](#_Toc427378010)

[Figure 21: <Hardware> 12A DC-to-DC step down module 37](#_Toc427378011)

[Figure 22: System Overview Use Case 37](#_Toc427378012)

[Figure 23: <User> Overview Use Case Diagram 38](#_Toc427378013)

[Figure 24: <User> Input Marble Use Case Diagram 38](#_Toc427378014)

[Figure 25: <User> Output Marble Use Case Diagram 40](#_Toc427378015)

[Figure 26: <User> View Order Detail Use Case Diagram 43](#_Toc427378016)

[Figure 27: Configure Ports Use Case Diagram 47](#_Toc427378017)

[Figure 28: Conceptual Diagram 49](#_Toc427378018)

[Figure 29: System Architectural Design 52](#_Toc427378019)

[Figure 30: PSSC Component Diagram 53](#_Toc427378020)

[Figure 31: Desktop App Class Diagram 54](#_Toc427378021)

[Figure 32: K – Nearest Class Diagram 55](#_Toc427378022)

[Figure 33: <User> Input Marble Flowchart Diagram 60](#_Toc427378023)

[Figure 34: <User> Output Marble Flowchart Diagram 61](#_Toc427378024)

[Figure 35: <User> View Order Details Flowchart Diagram 62](#_Toc427378025)

[Figure 36: <User> Configure Ports Flowchart Diagram 63](#_Toc427378026)

[Figure 37: <User> Input Marble Sequence Diagram 64](#_Toc427378027)

[Figure 38: <User> Output Marble Sequence Diagram 65](#_Toc427378028)

[Figure 39: <User> View Order Details Sequence Diagram 66](#_Toc427378029)

[Figure 40: <User> Configure Ports Sequence Diagrams 67](#_Toc427378030)

[Figure 41: <Hardware> Input Marble Details Activity Diagram 68](#_Toc427378031)

[Figure 42: <Hardware> Output Marble Details Activity Diagram 70](#_Toc427378032)

[Figure 43: <User> Input Marble Activity Diagram 72](#_Toc427378033)

[Figure 44: <User> Output Marble Activity Diagram 73](#_Toc427378034)

[Figure 45: <User> View Report Details 75](#_Toc427378035)

[Figure 46: <User> Configure Ports Activity Diagram 76](#_Toc427378036)

[Figure 47: Manage Repository Interface 77](#_Toc427378037)

[Figure 48: Create Customer Profile Interface 79](#_Toc427378038)

[Figure 49: Search Customer Order Interface 80](#_Toc427378039)

[Figure 50: View Order Details Interface 81](#_Toc427378040)

[Figure 51: Configure Ports Interface 82](#_Toc427378041)

[Figure 52: Logical Diagram 83](#_Toc427378042)

[Figure 53: K – Nearest Neighbor Flowchart 88](#_Toc427378043)

[Figure 54: Physical Diagram 89](#_Toc427378044)

[Figure 55: Connect to SQL Server with Windows Authentication mode 121](#_Toc427378045)

[Figure 56: Install PSSC Desktop Application 122](#_Toc427378046)

[Figure 57: USB Cable 122](#_Toc427378047)

[Figure 58: <Desktop App Guide> User configures ports connection 123](#_Toc427378048)

[Figure 59: <Desktop App Guide> User input marbles into machine 124](#_Toc427378049)

[Figure 60: <Desktop App Guide> User waits application update input marble 124](#_Toc427378050)

[Figure 61: <Desktop App Guide> User inputs package on conveyor belt 125](#_Toc427378051)

[Figure 62: <Desktop App Guide> User selects marble to output 125](#_Toc427378052)

[Figure 63: <Desktop App Guide> User inputs order customer information 126](#_Toc427378053)

[Figure 64: <Desktop App Guide> User waits machine to get out marbles 126](#_Toc427378054)

[Figure 65: <Desktop App Guide> User search customer order 127](#_Toc427378055)

[Figure 66: <Desktop App Guide> Show order details 128](#_Toc427378056)

[Figure 67: <Desktop App Guide> User Export Order Details 129](#_Toc427378057)

[Figure 68: <Machine Simulator> User configure ports connection 130](#_Toc427378058)

[Figure 69: <Machine Simulator> User input marble 131](#_Toc427378059)

# Definitions, Acronyms and Abbreviations

|  |  |
| --- | --- |
| **Name** | **Definition** |
| PSSC | Product Sorting System based on color |
| GUI | Graphic User Interface |

Table 1: Definitions

1. Report No.1 Introduction
2. Project Information

* Project name: **Design and implementation of Products Sorting System based on color**
* Project Code: **PSSC**
* Product Type: **Product Sorting Machine & Desktop Application**
* Start Date: **May 11th, 2015**
* End Day: **August 29th, 2015**

1. Introduction

Throughout history, the production activities in many areas have always been changing. Many questions related to manufacturing problems that we are facing are How to improve productivity? How to save costs while still highly profitable? By specific evidence, we will see the answer of these questions is very clear. To improve it, we have applied the science and technology to production. Nowadays, we can see many giant production lines in factories, it can do many things that the people cannot do in the long time. In small companies or business, we still have same machines but the size is smaller. In these systems or machines, we often face the problems about product classification. Maybe sort product by weight, by size, by material or by color… That becomes basic needs of manufacturing.

Base on real demand, we decide to make a project about product classification. We will design and implement of products sorting based on color. It is a convenient system to be highly applicable in the product line. It will determine color of products the sorting them. The system also has the central controller to monitor information about product. We hope it will contribute to improve productivity and production time for small business.

1. Current Situation

In the current production, sorting products by color based primarily on two solutions:

First, some companies or private business use modern equipment to create their line production system. These systems works extremely well, we could input a large number of products at the same time, and then they will detect color of all products and sort quickly. We could easily find a few systems or machine on the current market like Buller YJT Rice Color Sorting Machines, Tea Sorter, and Bean Color Sorter.

Second, some small businesses or farmers still work manually. After having products from harvest, they filter and sort them by hand like tomato, cerise… To improve productivity, they hire many workers. However, this way still has some problems.

1. Problem Definitions

Disadvantages of current situation:

* With the first solution, that we introduce above (in part A-3 Current Situation) is the best solution in production. The difficult of this way is investment funds and maintenance cost.
* With the second solution, the productivity improved by hiring many workers, but the quality of output product is not sure. Because of the problems of workers, their health may be affect the result of product. On the other hand, they do many steps from input to output, so it takes much time. All the information of input or output product is stored manually.

1. Proposed Solution

According to the difficulties recognized in part Problem Definitions, the solution that we use to solve problems based on color detection technology.

Product sorted system will have three main parts:

* Desktop Application.
* Hardware
* Mechanical part.

Classify by module, we have three blocks:

* Block 1: Inputting products and defining color of them.
* Block 2: Sorting product by color or getting product by color and quantity.
* Block 3: Controlling system and managing information of product.

More details is in Feature functions part below.

* 1. Feature functions

The basic and important feature functions are based on three blocks listed below:

* Block 1:
  + User can input product to the system.
  + Detect color of product.
* Block 2:
  + Sort product based on color, which detect in block 1.
  + Get product by color and quantity.
  + Control by central controller.
* Block 3:
  + Manage input and output product on database.
  + Manage repository, quantity of current product.
  + Count product.
  + Update real-time.
  + Control the system directly.
  1. Advantages and disadvantages

The advantages and disadvantages of the proposed solution:

* Advantage:
  + Cost savings, because the components of system are cheap and could buy at any electronic shops.
  + Easy to move to other places.
  + Easy to maintain and replace, it has separate modules.
  + Speed up the process of color detection and sorting.
  + Have the database to manage quantity of product.
* Disadvantages:
  + The quantity of input product is not much at the same time.
  + System only determine a color if the system has the sample of this color.
  + The input product must have big size, at least like marbles.
  + It depends on the mechanical solution.

1. Functional Requirements

Function requirements of the system listed as below:

* 1. Detect color of product
* User will input many products with different color to system. The system will determine color of product.
  1. Sorting product based on color
* When one product has color, which will be determine successfully, the system will filter and sort it to repository (or cart, basket) which has the same color.
  1. Filter and get product by color and quantity
* User inputs the number of product by color. The system will get them exactly and make as package.
  1. Manage information of product
* The information of product will be stored in a database.
* The system can manage about quantity of each input or output products by color and monitor on the screen to see the data, which is updated real time.

1. Role and Responsibility

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Full Name | Role | Position | Contact |
| 1 | Nguyễn Đức Lợi | Project Manager | Instructor | [loind@fpt.edu.vn](mailto:loind@fpt.edu.vn) |
| 2 | Nguyễn Hùng Mạnh | Developer | Leader | [manhnhse60893@fpt.edu.vn](mailto:manhnhse60893@fpt.edu.vn) |
| 3 | Nguyễn Khang Hy | Developer | Member | [hynkse61100@fpt.edu.vn](mailto:hynkse61100@fpt.edu.vn) |
| 4 | Nguyễn Nhật Linh | Developer | Member | [linhnnse61040@fpt.edu.vn](mailto:linhnnse61040@fpt.edu.vn) |
| 5 | Phạm Nhật Hưng | Developer | Member | [hungpn61119@fpt.edu.vn](mailto:hungpn61119@fpt.edu.vn) |

Table 2: Roles and Responsibilities

1. Report No.2 Software Project Management Plan
2. Problem Definition
   1. Name of this Capstone Project

* Design and Implementation of Products Sorting System based on color (PSSC)
  1. Problem Abstract

To improve the production activities, many product classification systems created and sold widely in the market. Understand about needs, we suggest to design and implementation of product sorting system based on color.

The targets of PSSC are farmers or small companies or business. Although PSSC is not as good as the exited systems in market, it still has full basic functionalities. In addition, the system also has an application to manage all information of input or output products. That will help the user can use the system easily.

* 1. Project Overview
     1. Current Situation

In the market, we have some ways to sorted product.

* Using S.PRECISION 12R-6SXM-756 machine to sort rice. The machine uses new technologies like CDD (2048 pixels), so it improve about productivity.
  + Advantage:
    - The machine can detect product, which has small size.
    - The machine sort products very fast.
    - The hardware easily replaces when they have problem.
  + Disadvantage:
    - High cost to maintain
    - Hard to move to other place.
* The farmers use hand to filter and sorter. Major product is vegetable or fruit and this method often used in preservation step.
  + Advantage:
    - People can classify product with complex shapes.
  + Disadvantage:
    - People use their eyes to recognize color of product. It may be not exactly.
    - The time to complete the job depends on the number of workers or experiment of them.
    1. The Proposed System

Based on some defects from other exit systems in the world and based on financial and knowledge condition of group. We decided to choose hardware simple and inexpensive. With structural frame by mica make our system lightness and easy to transport. The system includes three main part: Desktop Application, Hardware of sorter machine and Mechanical Components.

* + - 1. Desktop Application
* User can view information about input or output products in real time.
* User can input number of products by color and by quantity to get out of system.
* User can test hardware of sorter machine to know it is still good or not.
  + - 1. Hardware of sorter machine
* Arduino is the main board to control the softer machine.
* Infrared sensor is used to check the exited of product in system. It places in some parts of machine like the input product part” or the conveyor belt. It helps the system save energy.
* Color sensor used to detect color of input products.
* DC Motor used to run the conveyor belt.
* DC Servo used to make the filter gates in the system. These gates help to sorter products into right repository.
* L298 Dual H-Bridge to control DC motor.
* 12 A DC-to-DC step down module used to convert voltage from high to low for other hardware.
  + - 1. Mechanical Components

The core of system is a machine, which designed based on three blocks of system.

* Block 1: Inputting products and define color of them.
  + The machine has a straight conveyor belt to input products.
  + On the conveyor belt has an infrared sensor to check products, which input or not.
  + On the conveyor belt has two mini servos to make filter gates. When the signal from the infrared sensor is high level that means exited input product. The filter gate will open and put the product in front of color sensor to detect color. Then, it sends the signal to PC’s desktop Application to update database.
* Block 2: Sorting product by color or get product by color and quantity.
  + The machine has five repositories to contain different products by color. Each repository has three servos to make filter gates. These gates help products go into right repository.
  + User can get products out of system in this part via desktop application.
* Block 3: Control system and manage information of product.
  + Mechanical part includes a conveyor belt.
  + The conveyor belt has some packages on it. The products got out will drop into these packages.
    1. Boundaries of the System
* System is available for manager.
* The language of system is English.
* The input or output products are marbles. The boundaries of marbles are list below:
  + Diameter of marble is 15 mm (or 1.5 cm).

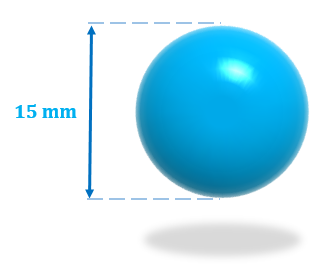


Figure 1: Diameter of marble

* + The colors of marbles used in system are red, blue, green, purple and pink.

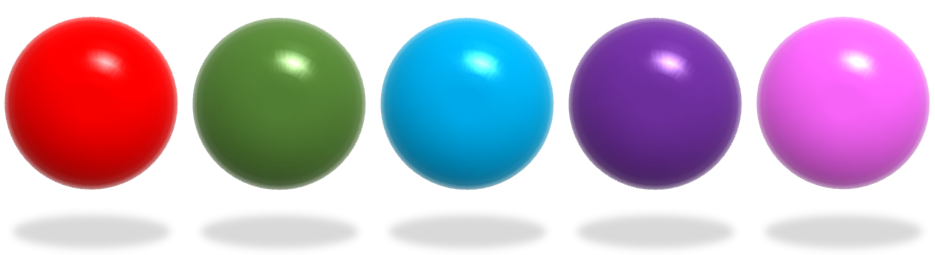
****

Figure 2: The colors of marbles used as products

* + The system only classifies marbles which have color like color sample in database of hardware program.
  + The system cannot classifies marbles that have two or more colors.
* The boundaries of mechanical parts includes:
  + The input product part only accepts at least 7 - 10 marbles at the same time.
  + The repository part only contains at least 7 – 9 marbles.
  + When user get out marbles from system. The maximum marbles for each package is five.
* The complete product includes:
  + The product sorter machine based on color.
  + The Desktop Application for manager to control the machine.
  + The database to storage all information about products.
  + All the documents of the project.

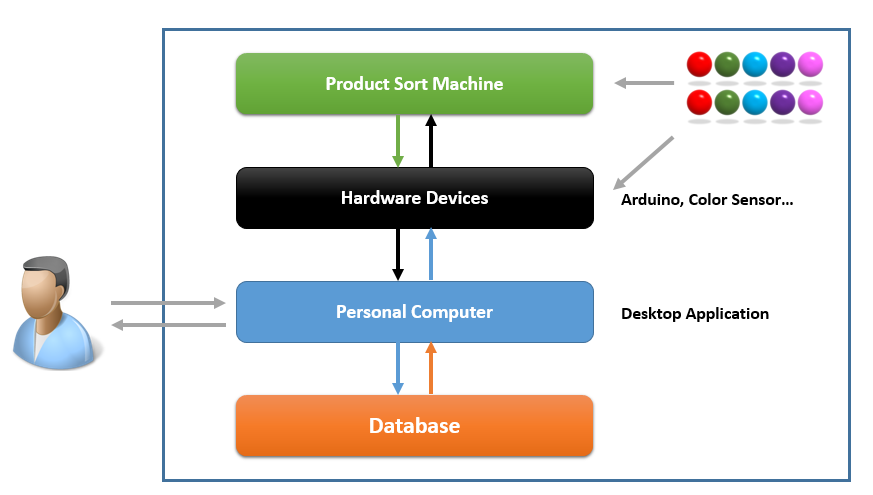


Figure 3: Boundaries of the system

* + 1. Development Environment
       1. Hardware requirements

For PC (or laptop) Desktop Application

|  |  |  |
| --- | --- | --- |
| **Windows** | **Minimum Requirements** | **Recommended** |
| **Operating System** | Windows 7/8 | Windows 7/8 |
| **Computer Processor** | Intel Core 2 Duo 6600 | Intel Core 2 Duo or higher |
| **Computer Memory** | 1 GB | 2 GB or more |
| **USB port** | 2 | 4 or more |

Table 3: Hardware requirements for PC (or laptop0 Desktop Application

For Sorter Machine

|  |  |
| --- | --- |
| **Components** | **Hardware** |
| **Mainboard** | Arduino Atmega 2560 |
| **Communication** | USB Cable |
| **Sensor Devices** | Color Sensor TCS3200, Infrared Sensor V1 |
| **Motors** | 12V DC Motor GA37, Servo RC MG996R, Mini Servo RC MG90S, L298 Dual H-Bridge Motor controller |
| **Power Source** | Power Supply 450W, 12A DC to DC step-down module |

* + - 1. Software requirements
* Windows 7/8: operating system for developing and deploying.
* SQL Server Express 2008/2012: used to create and manage database for PSSC.
* Visual Studio 2013: used to develop desktop application.
* Arduino IDE 1.0.6: used to develop Arduino program.
* Fritzing 0.9.2.b: used to drawing Arduino Circuit Board with other hardware.
* Google Code & Tortoise SVN: user for source control.
* Dev C/C++: use to develop hardware program.
* StarUML 2.2.0, Microsoft Visio 2010/2013: used to create models and diagrams.
* Skype, Gmail: used to communication and meeting.

1. Project organization
   1. Software Process Model

Project developed under waterfall model.

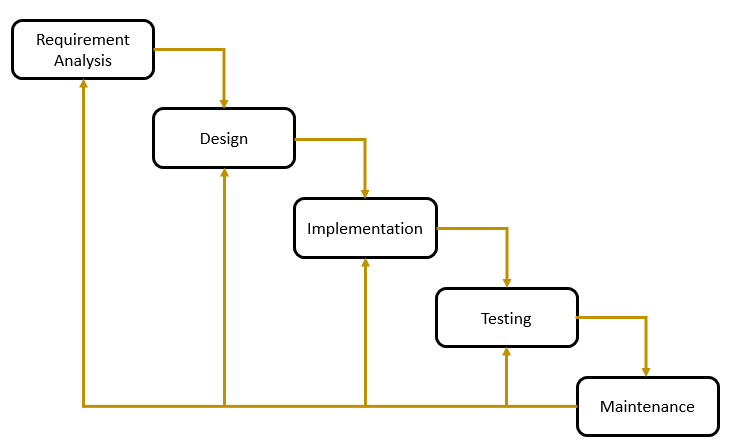


Figure 4: Waterfall development model

For more details information: <http://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm>

* 1. Role and responsibilities

|  |  |  |  |
| --- | --- | --- | --- |
| No | Full name | Role in Group | Responsibilities |
| 1 | Nguyễn Đức Lợi | Supervisor, Project manager | * Specify user requirement * Control the development process * Support about technique and business analysis * Review document |
| 2 | Nguyễn Hùng Mạnh | Team Leader, Developer, Tester | * Managing process * Managing budget * Dividing tasks for team member * Buying hardware * Training team member * GUI design * Mechanical components design * Create test plan * Clarifying requirements * Prepare document * Coding * Testing |
| 3 | Nguyễn Khang Hy | Team Member,  Developer, Tester | * Buying hardware * GUI design * Mechanical components design * Create test plan * Clarifying requirements * Prepare document * Coding * Testing |
| 4 | Nguyễn Nhật Linh | Team Member,  Developer, Tester | * GUI design * Mechanical components design * Create test plan * Clarifying requirements * Prepare document * Coding * Testing |
| 5 | Phạm Nhật Hưng | Team Member, Developer, Tester | * Training team member * GUI design * Mechanical components design * Create test plan * Clarifying requirements * Prepare document * Coding * Testing |

Table 4: Roles and Responsibilities

* 1. Tools and Techniques
* Hardware program:
  + Tools: Dev C/C++ 4.9.9.2, Visual Studio 13, Arduino IDE 1.0.6.
  + Libraries: Arduino library, Servo and DC Motor librabies.
* Desktop Application:
  + Tools: Visual Studio 2013.
  + Libraries: C# .NET Framework.
  + Techniques: Entity Framework, ADO. NET.
* Database: MS SQL Server 2012 Express (x64).

1. Project Management Plan
   1. Software development life cycle

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Phase | Description | Deliverable | Resource needed | Dependencies and Constrains | Risks |
| Requirement Analysis | - Collect requirements.  - Identity and clarify requirements for the system in general. | - Introduction of proposed system.  - Software requirement specification.  - Project plan and task schedule  - | 14 man-days | N/A | - Missing requirement  - Only leader understand scope of project. |
| Design | - Architecture.  - Detail design.  - Database design.  - Mechanical design. | - Software design document.  - Mechanical document. | 30 man – days | Depend on “Requirement Analysis”. | - Lack of experience about mechanical design. |
| Implementation | - Implement all code.  - Assemble mechanical components. | - Desktop Application.  - Hardware program.  - Sorter machine. | 50 man - days | Depend on “Design”. | - Lack of knowledge of team members about hardware. |
| Testing | - Components testing.  - Integration testing.  - System testing.  - Fix bugs. | - Test document. | 10 man - days | Depend on “Implementation”.  The sorter machine is complete. | - Lack of member.  - Lack of experience.  - Missing test case. |
| Maintenance | - Integrated installation. | User manual document. | 6 man - days | The final product already. | N/A |

* 1. Phase Detail
     1. Phase 1: Requirement Analysis

|  |  |  |
| --- | --- | --- |
| Task | Description | Author |
| 1. Collect requirements. | List current systems, which have same functions, give out their strengths and weakness. | ManhNH, HyNK, LinhNN, HungPN |
| 2. Identify and clarity main functions. | Define which main functions system should provide. | ManhNH, HyNK |
| 3. Create System introduction. | Complete introduction report. | LinhNN, HungPN |
| 4. Software project Management Plan. | Create plan and job schedule for team member. | ManhNH |
| 5. SRS | Complete SRS document | ManhNH, HyNK, LinhNN, HungPN |

Table 5: Phase 1: Requirement Analysis

* + 1. Phase 2: Design

|  |  |  |
| --- | --- | --- |
| Task | Description | Author |
| 1. Architecture Design | Design components and communicate method between them. | ManhNH, HyNK |
| 2. Detailed Design | Define objects, classes and function for hardware’s program and controller system. | ManhNH, HyNK, LinhNN, HungPN |
| 3. Database Design | Design tables. | HyNK |
| 4. Desktop Application GUI Design | Design user interface for control application. | HungPN |
| 5. Mechanical Components Design | Design mechanical components and select material. | ManhNH, HyNK, LinhNN, HungPN |

Table 6: Phase 2: Design

* + 1. Phase 3: Implementation

|  |  |  |
| --- | --- | --- |
| Task | Description | Author |
| 1. Suggestion Algorithms | Propose and compare algorithms to find a good solution and increase system performance. | ManhNH, HyNK, LinhNN, HungPN |
| 2. Hardware Program | Apply algorithms to develop program of color sensor.  Program for hardware devices via AT Mega processor on Arduino board. | ManhNH, HyNK |
| 3. Desktop Application | Program desktop application to control the system. | LinhNN, HungPN |
| 4. Unit testing | Create and perform unit testing. | LinhNN, HungPN |
| 5. Mechanical Assembly | Assemble hardware devices and mechanical components. | ManhNH, HyNK, LinhNN, HungPN |

Table 7: Phase 3: Implementation

* + 1. Phase 4: Testing

|  |  |  |
| --- | --- | --- |
| Task | Description | Author |
| 1. Components testing | Write test case for each component corresponding to each block or mechanical part of system and test system. | LinhNN |
| 2. Integration testing | Write test case for system and test system. | HyNK |
| 3. System testing | Write test case for real system in real time and test system. | HungPN |
| 4. Fix bugs | Correct bugs, repair mechanical components during test. | ManhNH |

Table 8: Phase 4: Testing

* + 1. Phase 5: Maintenance

|  |  |  |
| --- | --- | --- |
| Task | Description | Author |
| 1. Installation guide | Write installation guide. | LinhNN |
| 2. User Manual | Writer user manual. | HungPN |

Table 9: Phase 5: Maintenance

* 1. All Meeting Minutes

Refer to Meeting Minutes folder.

1. Coding Convention

C/C++: Using to develop program and solve algorithm on hardware.

Summary:

* Naming Convention:
  + Using Pascal case for class name.
  + Using Camel case for function, variable’s name.
  + The #define and global variable’s name must uppercase and separate by underscore. Ex: GLOBAL\_VARIABLE
* Commenting Convention:
  + Place the comment on the separate line with function.
  + Place the comment at the end of the line, which has calculation formula.

More details about code conventions for C/C++ language by Google:

[https://google-styleguide.googlecode.com/svn/trunk/cppguide.html](%20https:/google-styleguide.googlecode.com/svn/trunk/cppguide.html)

C#: Using to develop desktop application

Summary:

* Naming Convention.
  + Use Camel case for variable’s name.
  + Use Pascal case for class’s name, function’s name.
  + Global variable’s name must uppercase and separate by underscore.

More detail about code conventions for C# language by Microsoft:

<https://msdn.microsoft.com/en-us/library/ff926074.aspx>

1. Report No. 3 Software Requirement Specification
2. User Requirement Specification

* Product used for PSSC is marbles. (The details of marble is listed in Report 2 – part 1.3.3)
* PSSC detects marbles and sorts them to repositories based on color.
* Desktop application is used to control all the components of PSSC.
* Desktop application has database to store all information of input or output marbles.
* Desktop app provides function that user can use:
  + View number of current marbles in repositories.
  + Get marbles from repositories by color and quantity.
  + View input marbles history.
  + View order history.

1. System Requirement Specification
   1. External Interface Requirement
      1. User Interface

* User controls system via desktop application with many forms, so form must be simple, clearly and easy to use.
* Mechanical parts must be designed easily for user to input marbles or get them out of system.
  + 1. Hardware Interface
       1. Block Diagram – Overview

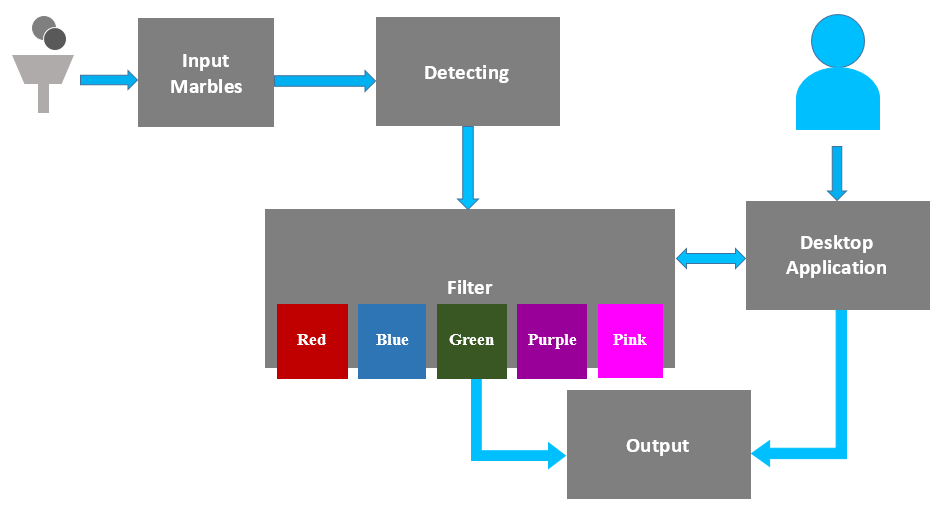


Figure 5: Block Diagram – Overview

|  |  |
| --- | --- |
| Block Name | Description |
| Input Marble | - Through this block, user could input directly marbles to the system. The limit of marbles is from 7 to 10 for each input times.  - The component includes mechanical part, one infrared sensor and two mini servos.   * Mechanical part: make marbles run easily in the system. * Infrared sensor: to detect user inputs marble or not. * Mini servos: to get only one marble at the same time. Then push it to Detecting Block to classify color. |
| Detecting | - When one marble runs to this block, it starts to check color of this marble and concludes.  - The component includes one Arduino board, one TCS3200 color sensor, one infrared sensor and one big servo.   * Color sensor: To detect input marble color then return signals to Arduino board. The signals are square way. The meaning of them are frequency of three basic color (Red, Green, and Blue). * Arduino board: Containing program and algorithm to conclude about color of marble. * Infrared sensor: To specify the right position of marble. If marble is in front on of the color sensor, the system will turn on the color sensor to classify color of marble. * Big servo: After finished detecting color of marble. This servo opens to push this marble to Filter Block. |
| Filter | - Base on the color of the marble, the system opens right repository for this marble. It means red marble will run to red repository and so on with other colors. If repository of this marble is not enough space, the marble will be pushed into the external repository.  - The component includes five infrared sensors and five big servos.   * Infrared sensor: Each sensor checks one repository based on color. If the repository is full space, the system pushes marble into external repository. * Big servo: These servos are filter gate. They make marble run right direction. |
| Desktop Application | - User views the current quantity of marbles in repositories based on color.  - User also gets marbles from system based on color and quantity.  - Desktop Application connects Filter block via USB Serial Port (UART). |
| Output Marble | - To get marbles based on color and quantity.  - User could export order history of customer or print it.  - The component includes one infrared sensors, ten mini servos, one DC motor and L298 H-Dual Bridge.   * Infrared sensor: Check user inputs package or box on conveyor belt or not. * Mini servos: Each repository has two mini servos to get marbles based on quantity. * DC motor: to run the conveyor belt. * L298 H-Dual Bridge: to control conveyor belt run right direction and power supply to DC motor. |

Table 10: Block Diagram – Overview

* + - 1. Block Diagram Details

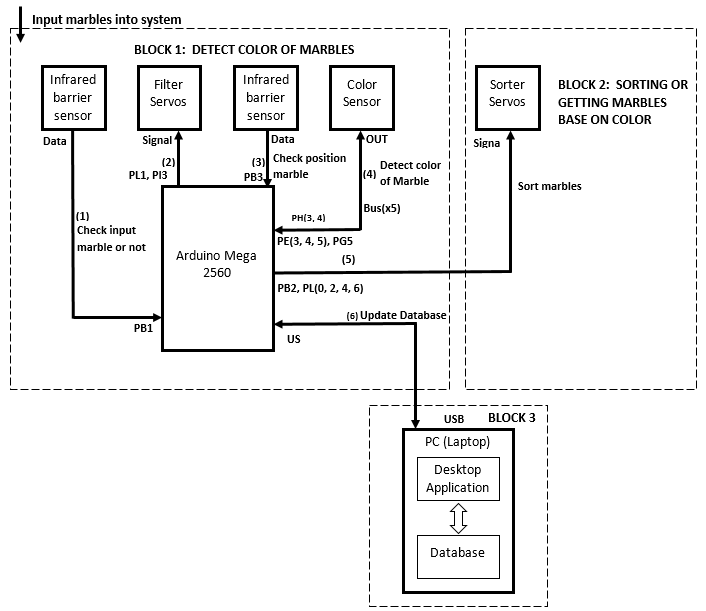


Figure 6: Block Diagram – Input Marble

|  |
| --- |
| **Block Diagram - Input Marble Description** |
| 1. Infrared sensor checks user inputs marble or not. If has input marbles, return bit 0 to Arduino, otherwise return bit 1. |
| 2. If has input marbles, Arduino controls servos to get only one marble at the same time and push this marble to color classifying area. |
| 3. Infrared sensor checks marble in front of color sensor or not. If has marble, return bit 0 to Arduino, otherwise return bit 1. |
| 4. If has marble, Arduino controls color sensor to detect color of marble. After detecting finished, color sensor sends square wave signals to Arduino. Arduino has algorithm to conclude what color of this marble is. |
| 5. Then Arduino controls servos to sorter the marble into right repository base on color. |
| 6. Arduino sends information of this marble to desktop application via USB Serial port to update database. |

Table 11: Block Diagram – Input Marble.

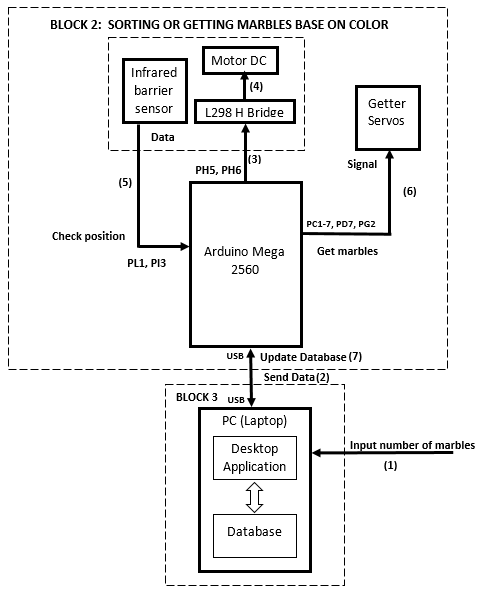


Figure 7: Block Diagram – Output Marble

|  |
| --- |
| **Block Diagram - Output Marble Description** |
| 1. User inputs number of marbles based on color via desktop application. |
| 2. Desktop application sends all output marbles information to Arduino board via USB serial port. |
| 3. After receiving data, Arduino controls L298 H-Dual Bridge. |
| 4. L298 H-Dual Bridge runs DC motor. It means running the conveyor belt. |
| 5. Infrared sensor checks user inputs package or not. If has package, return bit 0 to Arduino, otherwise return bit 1. |
| 6. If has package, Arduino controls getter servos to get marbles based on quantity and color. Then drop all output marbles to the package. |
| 7. Arduino sends signal to desktop application to update new repository quantity. |

Table 12: Block Diagram – Output Marble

* + - 1. Arduino Mega 2560

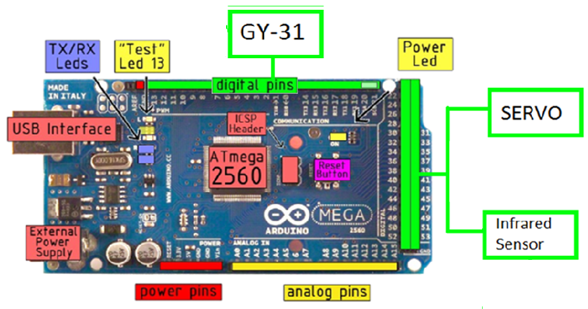


Figure 8: <Hardware> Arduino Mega 2560

Overview:

Arduino Mega 2560 is powerful circuit board with many features. PSSC uses two Arduino boards to control other hardware and sends signals between mechanical parts and desktop applications. When color sensor detects color of marbles, it must run K-Nearest Neighbor algorithm to find color, so it takes much time and it needs more RAM to store samples of colors. Then, the system controls many servos and conveyor belt to sort marbles or get them out. If using one Arduino board, system still do all things but performance slow down, system may be delay. Therefore, system needs to use two Arduino boards, one of them for algorithm and other for controlling.

Specification:

|  |  |  |
| --- | --- | --- |
| **No.** | **Name** | **Description** |
| 1 | Microcontroller | ATmega 2560 |
| 2 | Operating Voltage | 5V (via USB port) |
| 3 | Input Voltage | 7-12V |
| 4 | Input Voltage (limits) | 6-20V |
| 5 | DC Current per I/O Pin | 40 mA |
| 6 | DC Current for 3.3V Pin | 50 mA |
| 7 | Digital I/O Pins | 54 (15 Pins for PWM output) |
| 8 | Analog Input Pins | 16 (10 bit) |
| 9 | Clock Speed | 16 MHz |
| 10 | Flash Memory | 256 KB |
| 11 | SRAM | 8 KB |
| 12 | EEPROM | 4 KB |

Table 13: Adruino Mega 2560 - Specification

More details about Arduino Mega 2560:

<http://www.arduino.cc/en/Main/ArduinoBoardMega2560>

Or

<https://arduino-info.wikispaces.com/MegaQuickRef>

* + - 1. Color Sensor TCS3200

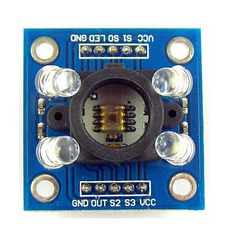


Figure 9: <Hardware> Color Sensor TCS3200

Overview:

Reason for PSSC to use color sensor TCS3200:

* Cheap sensor.
* Easy to buy via market.
* Detect color quickly.
* Only detect exactly object, which has one of colors red, green or blue. This problem could be solved by coding and algorithm, so the senor can detect more colors.

Specification:

* Block Diagram:

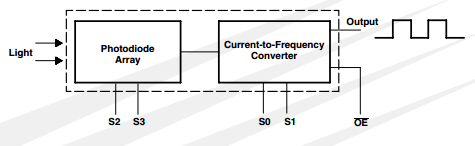


Figure 10: <Hardware> Block Diagram of Color Sensor TCS3200

* Input and Output Pins:

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | I/O | Description |
| 1 | VCC (x2) |  | 5V |
| 2 | GND (x2) | O | Power supply ground. |
| 3 | S0, S1 | I | Select output frequency scaling. |
| 4 | S3, S4 | I | Select photodiode type. |
| 5 | LED |  | Create White Balance Environment |
| 6 | OUT |  | Output frequency (a square wave) |

Table 14: TCS3200 - Input and Output Pins

* Select output frequency scaling by S0, S1:

|  |  |  |
| --- | --- | --- |
| **S0** | **S1** | **Output Frequency Scaling** |
| L | L | Power down |
| L | H | 2% |
| H | L | 20% |
| H | H | 100% |

Table 15: TCS3200 - Select output frequency

* Select Photodiode type by S3, S4:

|  |  |  |
| --- | --- | --- |
| **S3** | **S4** | **Photodiode type** |
| L | L | Red |
| L | H | Bule |
| H | L | Clear |
| H | H | Green |

Table 16: TCS3200 - Select Photodiode Type

More details about Color Sensor TCS3200:

<http://ams.com/eng/Products/Light-Sensors/Color-Sensor/TCS3200>

Connect to Arduino Mega:

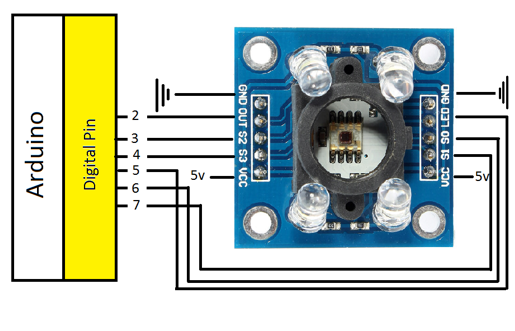


Figure 11: TCS3200 connects to Arduino.

* + - 1. Infrared Sensor V1

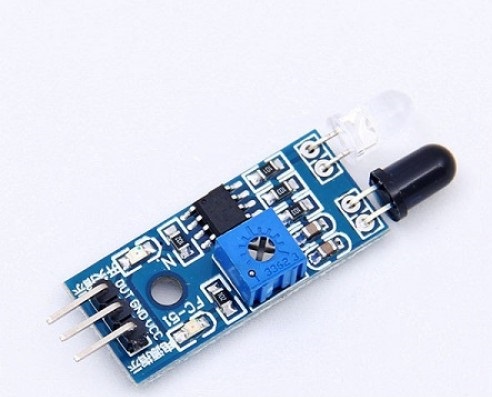


Figure 12: <Hardware> Infrared Sensor V1

Overview:

In PSSC, Infrared Sensor V1 is used to detect the position of marbles on the conveyor belt. If user does not input any marbles, infrared sensor sends bit 1 to Arduino, otherwise it sends bit 0. When Arduino receives bit 0, it sends signals and triggers other hardware to work.

Specification:

* Input and Output Pins:

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | I/O | Description |
| 1 | VCC |  | 5V |
| 2 | GND |  | Power supply ground. |
| 3 | OUT | O | If no object in front of sensor, it returns bit 1, otherwise it returns bit 0. |

Table 17: Infrared Sensor - Input and Output Pins

* If no object in front of sensor, it returns bit 1, otherwise it returns bit 0.

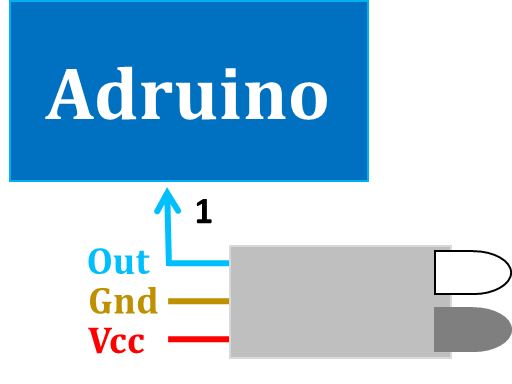


Figure 13: Infrared Sensor returns bit one

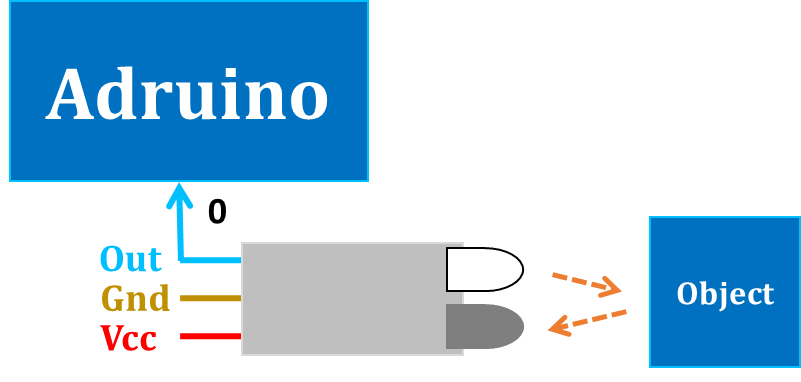


Figure 14: Infrared Sensor returns bit zero

**Connect to Arduino Mega:**

In PSSC system, we use eight infrared sensors. Output pin of each sensor connects to Arduino board is list below:

|  |  |
| --- | --- |
| **Name of infrared sensors** | **Arduino pin** |
| IP\_Gate | 48 (Arduino 1) |
| IP\_TCS230 | 46 (Arduino 1) |
| IP\_Red | 40 (Arduino 1) |
| IP\_Green | 38 (Arduino 1) |
| IP\_Blue | 36 (Arduino 1) |
| IP\_Purple | 34 (Arduino 1) |
| IP\_Pink | 32 (Arduino 1) |
| IP\_Check | 10 (Arduino 2) |

Table 18: Infrared Sensors connect to Arduino Mega

* + - 1. Servo RC



Figure 15: <Hardware> Servo Tower Pro MG996R.



Figure 16: <Hardware> Servo Tower Pro MG90S

Specification:

* Servo MG996R:

|  |  |  |
| --- | --- | --- |
| **No.** | **Name** | **Description** |
| 1 | Dimension | 40.7\*19.7\*42.9mm |
| 2 | Stall torque | 9.4kg/cm (4.8V)  11kg/cm(6V) |
| 3 | Operating speed | 0.17sec/60degree(4.8V)  0.14sec/60degree(6V) |
| 4 | Operating voltage | 4.8-7.2V |
| 5 | Temperature voltage | 0o-550C |
| 6 | Weight | 55.0g |

Table 19: Servo MG996R - Specification

* Servo MG90S:

|  |  |  |
| --- | --- | --- |
| **No.** | **Name** | **Description** |
| 1 | Dimension | 22\*12\*29mm |
| 2 | Stall torque | 1.8kg/cm (4.8V)  2.2kg/cm(6V) |
| 3 | Operating speed | 0.10sec/60degree(4.8V)  0.08sec/60degree(6V) |
| 4 | Operating voltage | 4.8-6.0V |
| 5 | Temperature voltage | -30o-600C |
| 6 | Weight | 13.4g |

Table 20: Servo MG90S - Specification

* Input and Output Pins:

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | I/O | Description |
| 1 | VCC |  | Depend on servo type |
| 2 | GND |  | Power supply ground. |
| 3 | Signal | I | The servo will move based on the signal (pulse) sent to signal wire. It set the angle of the arm on servo. |

Table 21: Servo – Input and Output Pins

* This picture describes how servo move based on the PPM:

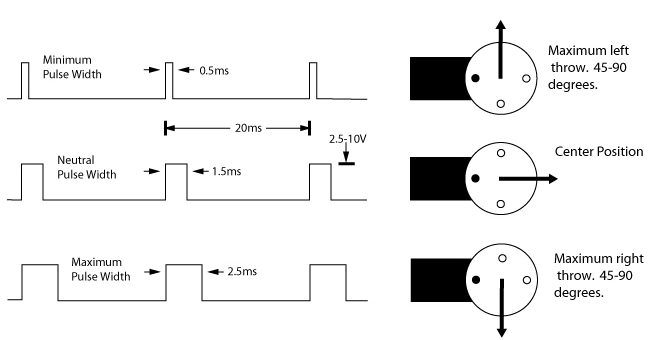


Figure 17: <Hardware> Servo - Move based on PPM

More details about Controlling Servo based on PPM:

<http://www.hooked-on-rc-airplanes.com/servo-tutorial.html>

<http://arduino.vn/reference/xung-ppm>

Connect to Arduino Mega:

In PSSC system, we use eighteen servors. Signal pin of each servo connects to Arduino board is list below:

|  |  |  |
| --- | --- | --- |
| **Name of servos** | Type of servo | **Arduino Pin** |
| LServo\_Tcs230 | MG996R | 53 (Arduino 1) |
| LServo\_Red | MG996R | 51 (Arduino 1) |
| LServo\_Green | MG996R | 49 (Arduino 1) |
| LServo\_Blue | MG996R | 47 (Arduino 1) |
| LServo\_Purple | MG996R | 45 (Arduino 1) |
| LServo\_Pink | MG996R | 43 (Arduino 1) |
| MServo\_Gate1 | MG90S | 52 (Arduino 1) |
| MServo\_Gate2 | MG90S | 50 (Arduino 1) |
| MServo\_Red\_Gate1 | MG90S | 52 (Arduino 2) |
| MServo\_Red\_Gate2 | MG90S | 53 (Arduino 2) |
| MServo\_Green\_Gate3 | MG90S | 50 (Arduino 2) |
| MServo\_Green\_Gate4 | MG90S | 51 (Arduino 2) |
| MServo\_Blue\_Gate5 | MG90S | 48 (Arduino 2) |
| MServo\_Blue\_Gate6 | MG90S | 49 (Arduino 2) |
| MServo\_Purple\_Gate7 | MG90S | 46 (Arduino 2) |
| MServo\_Purple\_Gate8 | MG90S | 47 (Arduino 2) |
| MServo\_PinkGate9 | MG90S | 44 (Arduino 2) |
| MServo\_PinkGate10 | MG90S | 45 (Arduino 2) |

Table 22: Servos connect to Arduino Mega

* + - 1. 12V DC Motor GA37



Figure 18: <Hardware> 12V DC Motor GA37

Overview:

DC Motor GA-37 is a strong engine. It has a high ratio gearbox inside, so the torque is high. This engine can be used to make kinds of vehicles or robotic.

In PSSC, DC Motor GA-37 is used to run conveyor belt.

Specification:

|  |  |  |
| --- | --- | --- |
| No. | Name | Description |
| 1 | Input Voltage | 6-12V |
| 2 | Full speed | 5600rpm |
| 3 | Speed with gearbox | 6V ~ 30rpm  12V ~ 60rpm |
| 4 | Weight | 175.0g |

Table 23: 12V DC Motor GA-37 - Specification

* + - 1. L298 Dual H-Bridge Motor controller

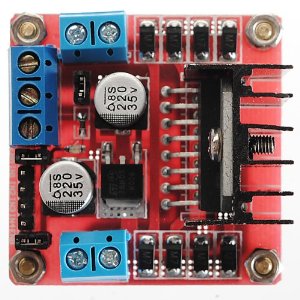


Figure 19: <Hardware> L298 Dual H-Bridge Motor controller

Overview:

L298 Dual H-Bridge Motor could control two DC motors or 1 step motor. In PSSC, DC Motor GA37 need high input voltage to run well (12V). Arduino board could provide this voltage, but it is not good. Because Arduino board still provides power to other hardware and control them. So that, L298 Dual H-Bridge Motor can use to provide high voltage to DC motor and Arduino board can control motor via this hardware. The performance of system will not be affected.

Specification:

* Principle diagram:

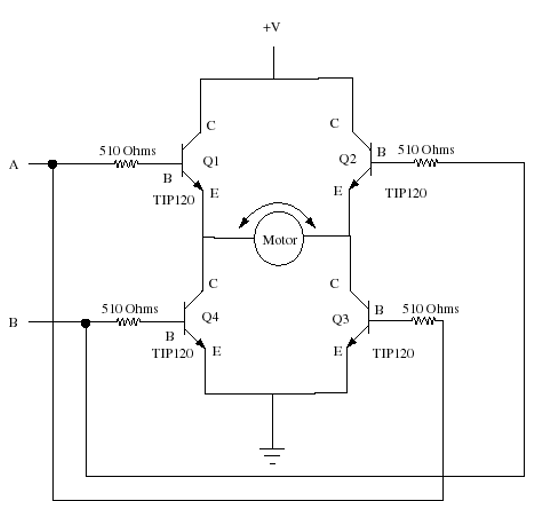


Figure 20: <Hardware> L298 Dual H-Bridge - Principle diagram

Select run mode:

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **Description** |
| 0 | 0 | Stop |
| 0 | 1 | Run Counter-Clockwise |
| 1 | 0 | Run Clockwise |
| 1 | 1 | N/A |

Table 24: L298 Dual H-Bridge - Specification

* + - 1. 12A DC to DC step-down module

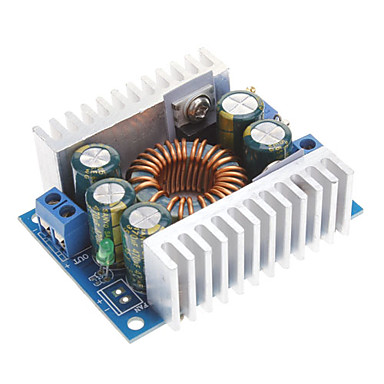


Figure 21: <Hardware> 12A DC-to-DC step down module

Overview:

In PSSC, the input voltage of some MG996R servos is 7.2 V, so this module will convert from 12V to 7.2V for these servos.

Specification:

|  |  |  |
| --- | --- | --- |
| **No.** | **Name** | **Description** |
| 1 | Input Voltage | 4.5-30V |
| 2 | Output voltage | 0.8-30V |
| 3 | Output current | 0–12A 100W (up to 200W) |
| 4 | Working temperature | -40o–85oC |
| 5 | Working frequency | 300kHz |

Table 25: 12A DC-to\_DC step down - Specification

* + 1. Software Interface
* Desktop application runs on computer (pc or laptop).
* Computer must install SQL Server Express 2008/2012 with .NET Framework 4.0 or more.
  + 1. Communication Protocol
* Arduino Mega board connects to computer via USB port. It knows as Serial protocol.
  1. System Overview Use Case

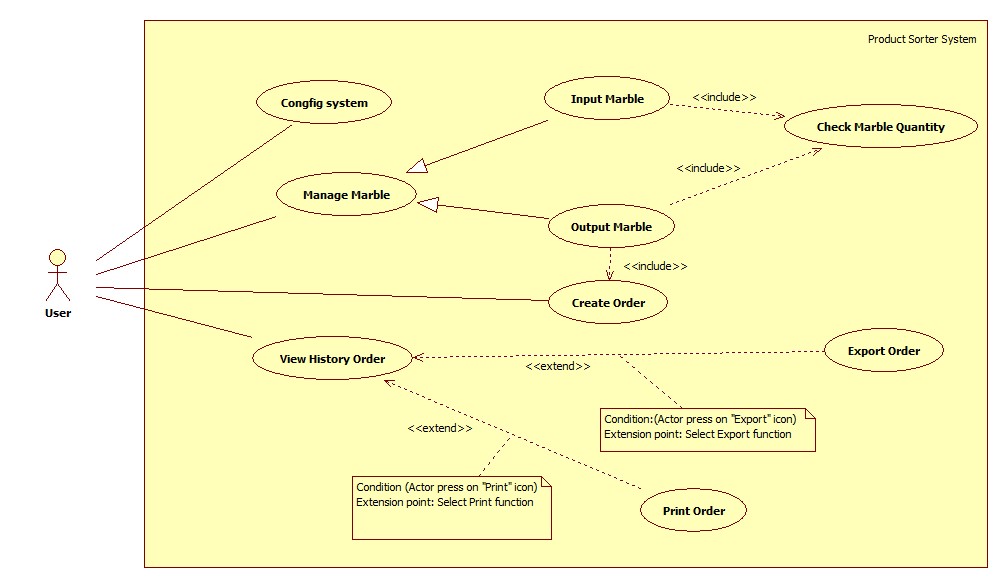


Figure 22: System Overview Use Case

* 1. List of Use Case
     1. <User> Overview Use Case

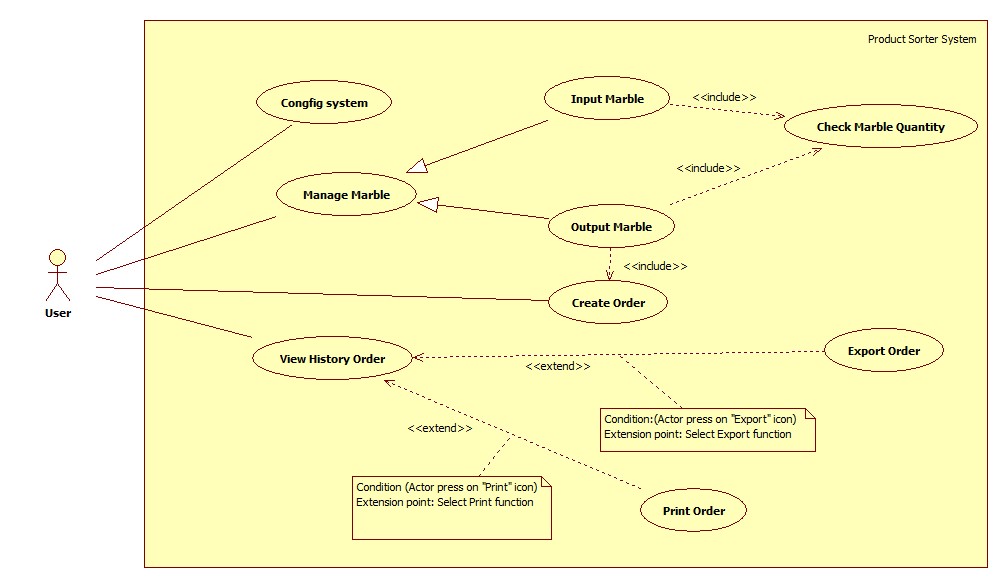


Figure 23: <User> Overview Use Case Diagram

* + - 1. <User> Input Marble Use Case Diagram

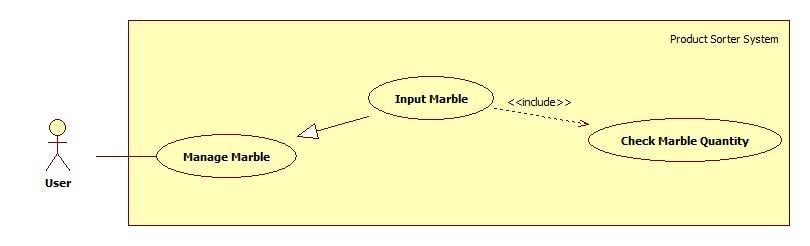


Figure 24: <User> Input Marble Use Case Diagram

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Specification** | | | |
| **USE CASE-1 SPECIFICATION** | | | |
| **Use-case No.** | PSSC001 | **Use-case Version** | 2.0 |
| **Use-case Name** | Input Marble | | |
| **Author** | Nguyen Nhat Linh | | |
| **Date** | 30/05/15 | **Priority** | High |
| **Actor:**   * User.   **Summary:**   * This use case allow user input marble to system.   **Goal:**   * Marble input to right repository in system based on color.   **Triggers**   * User click on “REPOSITORY” menu.   **Preconditions:**   * Program is ready.   **Post Conditions:**   * **On Success**: Marble put to repository and update information to database. * **On Failure**: Show error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | User click on “REPOSITORY” Menu. | Application will navigate to “REPOSITORY” Menu.   * “Marble Management Form”: Form. * “Red”: Textbox. * "Green”: Textbox. * "Blue”: Textbox. * "Purple”: Textbox. * "Pink”: Textbox. * "Total”: Textbox. * "Size Limits”: Textbox. * "Red": NumericUpDown. * "Green": NumericUpDown. * "Blue": NumericUpDown. * "Purple": NumericUpDown. * "Pink": NumericUpDown. * “Output Limits”: Textbox. * "Package x": NumericUpDown. * "Get Marbles”: Button. * "Clear”: Button. * Table Monitor: DataGridView. * Log: RichTextbox. | | 2 | User Input 1 marble into PSSC machine | PSSC machine moves marble to right repository. Then system shows message: "Receive 1 [Color: Red, Green, Blue, Purple or Pink] Marble! “, updates information to Database.  [Alternative 1]  [Exception 1, 2] |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | User clicks on other menu. | Application will navigate to that menu and show content. |   **Exceptions:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | Connection between machine and application is failed. | Show error message: “Please check machine connection!” | | 2 | User input 1 marble into PSSC machine, but repository is not enough space for marble. | Show message box: “Not enough space for marble!” and PSSC machine moves marble to extend repository. |   **Relationships:**   * N/A   **Business Rules:**   * Move marble to right repository. * Update information to database. * When not enough space, move marble to extend repository. * Avoiding exception. | | | |

Table 26: <User> Input Marble Use Case Diagram

* + - 1. <User> Output Marble Use Case Diagram

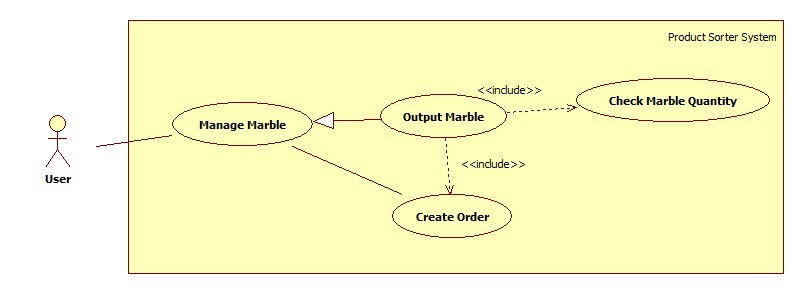


Figure 25: <User> Output Marble Use Case Diagram

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Specification** | | | |
| **USE CASE-1 SPECIFICATION** | | | |
| **Use-case No.** | PSSC002 | **Use-case Version** | 2.0 |
| **Use-case Name** | Output Marble | | |
| **Author** | Pham Nhat Hung | | |
| **Date** | 30/05/15 | **Priority** | High |
| **Actor:**   * User.   **Summary:**   * This use case allow user get marbles for customer based on color and quantity, save order information.   **Goal:**   * Output marbles to customer based on color and quantity, save order information, save order information.   **Triggers**   * User click on “REPOSITORY” menu.   **Preconditions:**   * Program is ready.   **Post Conditions:**   * **On Success**: User get marbles form conveyor belt. * **On Failure**: Show error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | User click on “REPOSITORY” Menu. | Application will navigate to “REPOSITORY” Menu.   * “Marble Management Form”: Form. * “Red”: Textbox. * "Green”: Textbox. * "Blue": Textbox. * "Purple": Textbox. * "Pink": Textbox. * "Total": Textbox. * "Size Limits": Textbox. * "Red": NumericUpDown. * "Green": NumericUpDown. * "Blue": NumericUpDown. * "Purple": NumericUpDown. * "Pink": NumericUpDown. * “Output Limits”: Textbox. * "Package x": NumericUpDown. * "Get Marbles”: Button. * "Clear”: Button. * Table Monitor: DataGridView. * Log: RichTextbox. | | 2 | User Choose number of marbles and number of packages. | Application shows in Table Monitor :   * Marble Color: Name of Marble. * Output Marble: Quantity of Mable. * Status: “Waiting”.   [Exception 1, 2, 3, 4] | | 3 | User Click "Get Marbles" | Application show Customer Form :form   * “Name”: Textbox, string, min length: 1, max length: 50. * “Email”: Textbox, email format, min length: 1, max length: 20. * “Phone”: Textbox, number, min length: 1, max length: 20. * “Address”: Textbox, string ,min length: 1. * “Save”: Button. * “Exit”: button.   [Exception 5] | | 4 | User fill information then click "Save" Button | Save customer information to database. Then PSSC machine outputs marble to the boxes on conveyor belt.  Application shows success message on Table Monitor:   * Marble Color :Name of Marble * Output Marble: Quantity decrease by one per one output marble. * Status: When output marble is zero, "Waiting" status becomes to "Success" status.   [Exception 6, 7, 8, 9, 10, 11] |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | User clicks on other menu. | Application will navigate to that menu and show content. |   **Exceptions:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | Connection between machine and application is failed. | Show error message: “Please check machine connection!” | | 2 | User does not select any marbles. | Show error message: "Please select marble!” | | 3 | User selects too many for one package. | Show error message: “Output Marbles must be less than or equal 5”. | | 4 | User selects many packages, but not enough marble for all. | Show error message: “Not enough [Color] marbles in repository for [Number] packages!”  [Color]: Color of marble that user selects.  [Number]: Number of package that user selects. | | 5 | User does not input package to conveyor belt. | Show error message: “Please input box to conveyor belt” | | 6 | Name is empty. | Show message: “The Name field is Required.” | | 7 | Email is empty. | Show message: “The Email field is Required.” | | 8 | Phone is empty. | Show message: “The Phone field is Required.” | | 9 | Address is empty. | Show message: “The Address field is Required.” | | 10 | User inputs wrong email format. | Show message: “The email is not valid.” | | 11 | User inputs alphabet characters or special characters to phone Textbox. | Show message: “Phone field accepts only Digits.” |   **Relationships:**   * N/A   **Business Rules:**   * User must input box (package) on conveyor belt before system output marbles. * Total marbles that user selects must be less than five marbles for one package. * System allows output when total marbles of all packages are available in repositories. * Customer information must be filled complete before getting marbles. * Avoiding exception. | | | |

Table 27: <User> Output Marble Use Case Diagram

* + - 1. <User> View Order Detail Use Case Diagram

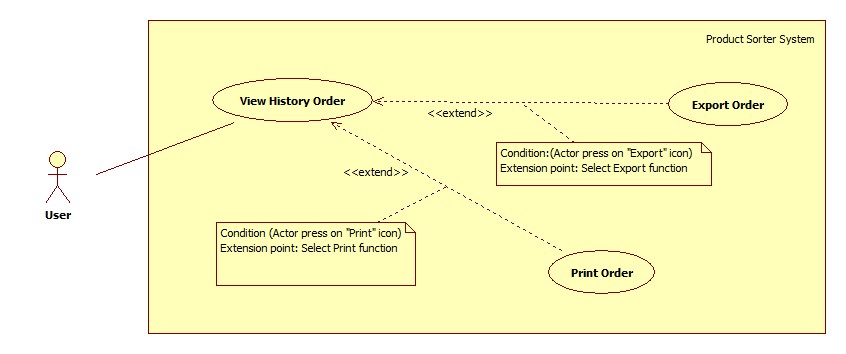


Figure 26: <User> View Order Detail Use Case Diagram

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Specification** | | | |
| **USE CASE-1 SPECIFICATION** | | | |
| **Use-case No.** | PSSC003 | **Use-case Version** | 2.0 |
| **Use-case Name** | View History | | |
| **Author** | Nguyen Khang Hy | | |
| **Date** | 30/05/15 | **Priority** | High |
| **Actor:**   * User.   **Summary:**   * This use case allow user views order history.   **Goal:**   * User views order history.   **Triggers**   * User click on “REPORT” menu.   **Preconditions:**   * Program is ready.   **Post Conditions:**   * **On Success**: Show order details of one customer for user to export or print. * **On Failure**: Show error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | User click on “REPORT” menu. | Application will navigate to “REPORT” Menu.   * “Report Form”: form. * Search Textbox: Search key * “Name”: Textbox. * "Email”: Textbox * "Phone”: Textbox * "Order ID”: Textbox * "Order Date”: Textbox * "Total": Textbox * "View Details”: Link * Order Table : List Order information | | 2 | User fills key to Search Textbox. | Order Table list all order match search key. | | 3 | User choose order then click "View Details" | Show "Customer Order Details":   * Customer Information: Information of customer * Order Details: Information of Order * "Print”: button * "Export”: button   [Alternative 1, 2, 3] |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | User clicks on other menu. | Application will navigate to that menu and show content. | | 2 | User clicks "Print" icon. | Application connects to all printers in system for user to select. | | 3 | User clicks "Export" button  And select export format (Pdf ,Word or Excel) | Export order detail of customer to file for user. |   **Exceptions:**   * N/A   **Relationships:**   * N/A   **Business Rules:**   * Version of excel is “xlsx” (MS 2007, 20120, 2013) * Version of word is “docx” (MS 2007, 20120, 2013) * Export Excel file with format      * Export Word file with format      * Export PDF file with format | | | |

Table 28: <User> View Order Detail Use Case Diagram

* + - 1. <User> Configure Ports Use Case Diagram

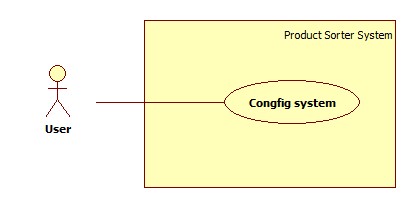


Figure 27: Configure Ports Use Case Diagram

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Specification** | | | |
| **USE CASE-1 SPECIFICATION** | | | |
| **Use-case No.** | PSSC004 | **Use-case Version** | 2.0 |
| **Use-case Name** | Configure System | | |
| **Author** | Nguyen Hung Manh | | |
| **Date** | 30/05/15 | **Priority** | High |
| **Actor:**   * User.   **Summary:**   * This use case allow user configure ports in system.   **Goal:**   * Application and PSSC machine connects successfully.   **Triggers**   * User click on “CONFIGURATION” menu.   **Preconditions:**   * Program is ready.   **Post Conditions:**   * **On Success**: Application and PSSC machine connect Successfully and save setting to database. * **On Failure**: Show error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | User click on “CONFIGURATION” Menu. | Application will navigate to “CONFIGURATION” Menu.  “Port Setting”: form.   * “Input Port”: Combo Box. * “Output Port”: Combo Box. * “Save”: Button. * "Testing”: Button * "Exit”: Button | | 1 | User choose Input Port and Output Port then click "Save" button | System Save Port and show success message  [Exception 1] | | 2 | User choose Input Port and Output Port then click "Testing" button | System test connections and show success message  [Exception 2] | | 4 | User "click Exit button". | System exit Port setting form |   **Alternative Scenario:**  **N/A**  **Exceptions:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | Cannot save port setting | Show message box: “Cannot save [Port Name]. This port already used!”  [Port Name]: Name of port that user selects. | | 2 | Testing connection fail | Show message box: “[Port Name] is not opened!"  [Port Name]: Name of port that user selects. |   **Relationships:**   * N/A   **Business Rules:**   * Show exactly ports in System. * Application and PSSC machine connect successfully when user chooses right ports. * Avoiding exception. | | | |

Table 29: Configure Ports Use Case Diagram

1. Software System Attribute
   1. Usability

* User controls all system components via only one desktop application.
* Desktop application is installed on personal computer or laptop.
* User can learn how to use the system in 20 minutes.
* The system deploy quickly and easily.
  1. Reliability
* Number of errors on input products that occurred is 1/200 (errors/products). Error: The system detects wrong color of product.
  1. Availability
* The mechanical components has two modules. When one of them has problem, it will not affect to all the system and data is not missing.
* Hardware components are easy to find in market.
  1. Security
* The system only needs one person to manage and control, so the security is not necessary.
  1. Maintainability
* Easy to replace hardware components or upgrade.
* The mechanical components it is easy to replace.
  1. Portability
* The hardware and mechanical components is combined in one machine.
* Easy to move.
  1. Performance
* Color detection speed is one product in three milliseconds.
* The conveyor belt of sorter mechanical part is designed based on knowledge of physics; include slanted surface and gravity, so the product runs to repository automatically and quickly.

1. Conceptual Diagram

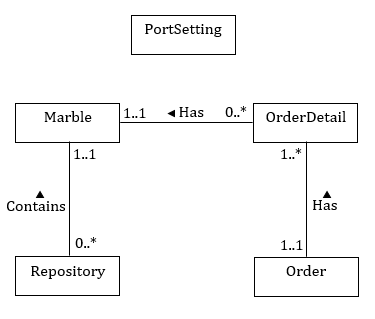


Figure 28: Conceptual Diagram

Data Dictionary

|  |  |
| --- | --- |
| **Entity Data dictionary: describe content of all entities** | |
| **Entity Name** | **Description** |
| PortSettings | Describe all port settings store in the system. |
| Marble | Describe all marble information in the system. |
| Repositories | Describe all repository for each marble in the system. |
| Orders | Describe all orders stored in the system. |
| OrderDetails | Describe all order detail in the system. |

1. Report No.4 Software Design Description
2. Design Overview

* This document describes the technical and user interface design of PSSC System. It includes the architectural design, the detailed design of common functions and business functions and the design of database model.
* The architectural design describes the overall architecture of the system and the architecture of each main component and subsystem.
* The detailed design describes static and dynamic structure for each component and functions. It includes class diagrams, class explanations and sequence diagrams for each use cases.
* The database design describes the relationships between entities and details of each entity.
* Document overview:
* Section 2: gives an overall description of the system architecture design.
* Section 3: gives component diagrams that describe the connection and integration of the system.
* Section 4: gives the detail design description, which includes class diagram, class explanation, flowchart diagram, sequence diagram and activity diagram to details the application functions.
* Section 5: describe screens design.
* Section 6: describe a fully attributed ERD.
* Section 7: describe algorithms (HOW).

1. System Architectural Design

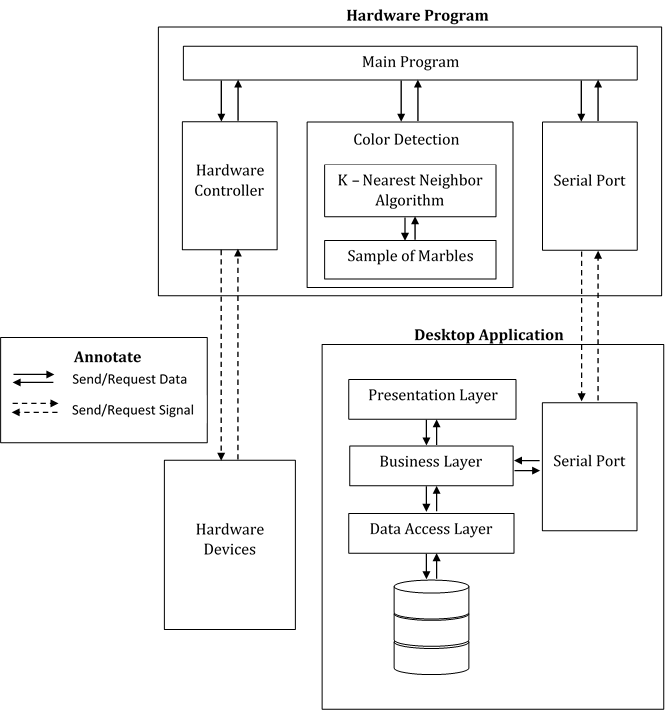


Figure 29: System Architectural Design

* 1. Hardware Program Architecture description

Hardware program is developed on Arduino Kit. The program language used is C++. The program include four parts:

- Main Program receives all input data or signal from Desktop Application and Hardware. Based on type of data or signal, Main Program calls functions to control hardware devices or reply to Desktop Application.

- Color Detection is the core of hardware program. It contains the sample of five colors and K – Nearest Neighbor Algorithm. Arduino uses this core to detect color of input marble and return result for Main Program.

- Hardware Controller is the set of functions to control all hardware devices like color sensor, servo, motor…

- Serial Port is the library of Arduino IDE. Main Program part uses this library to communicate with Desktop Application, transfer and receive signal data.

* 1. Desktop Application Architecture description

Desktop application is developed based on three-layer architecture. The language programing is C# and the technologies used are Entity Framework and Linq in .NET.

The architecture has three main parts:

- Data Access Layer: this layer builds the SQL command based on the request from Business Layer to query or update data from database. Then return the results to the business layer.

- Business Layer: this layer is a bridge between to Data Access layer and Presentation Layer. The logic of system is developed at this layer.

- Presentation Layer: this layer is used to get data from user and passing to the business layer to process. Data received is showed on Desktop GUI for user.

The external part is:

- Serial port is the library of .NET Framework. Application uses this library to communicate with Arduino Kit, transfer and receive signal data.

1. Component Diagram

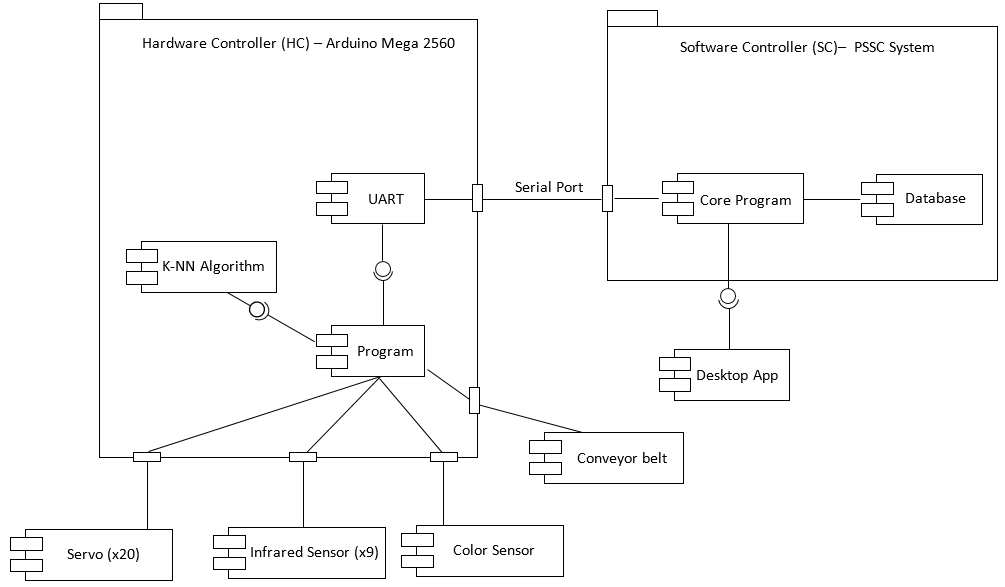


Figure 30: PSSC Component Diagram

|  |  |
| --- | --- |
| Component dictionary: describe component | |
| Component Name | Description |
| <HC> Program | Control all other hardware like servos, sensors. |
| <HC> K-NN Algorithm | Algorithm to detect color of marble. |
| <HC> UART | Use serial port to connect to PC, Laptop via USB port. |
| <SC> Core Program | Handle process between Arduino, desktop app and database. |
| <SC> Database | SQL Database to store information about repository and orders. |
| Desktop App | Help user control all system. |
| Servo | Servo RC helps marble run right way in machine. |
| Infrared Sensor | Detect position marble in machine. |
| Color Sensor | Detect color of marble. |
| Conveyor belt | Output marble to customer. |

Table 30: Component Dictionary

1. Detailed Description
   1. Class Diagram
      1. Desktop App Class Diagram



Figure 31: Desktop App Class Diagram

* + 1. K – Nearest Neighbor Class Diagram



Figure 32: K – Nearest Class Diagram

* 1. Class Diagram Explanation
     1. <Desktop App> Marble

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| MarbleId | int | Private | Unique identifier of Marble, auto increment. |
| MarbleColor | string | Private | Color of Marble |
| MarbleDescription | string | Private | Description of Marble |
| Price | float | Private | Price of Marble |
| DateOfCreation | string | Private | Date Marble Create |
| DateOfLastModification | string | Private | Last time Modification of marble |

Table 31: <Desktop App> Marble Attribute

Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| Getter | Attribute type | Public | Get attribute value |
| Setter | void | Public | Set value of attribute |

Table 32: <Desktop App> Marble Method

* + 1. <Desktop App> Repository

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| RepositoryId | int | Private | Unique identifier of Repository, auto increment. |
| RepositoryName | string | Private | Name of Repository |
| RespositoyDescription | string | Private | Description of Repository . |
| Quantity | int | Private | Quantity of marble in Repository. |
| QuantityLimits | int | Private | Max Quantity of marble in Repository |
| MarbleId | int | Private | ID of Marble |
| DateOfCreation | string | Private | Date Repository Create |
| DateOfLastModification | string | Private | Last time Modification of Repository |

Table 33: <Desktop App> Repository Attribute

Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| Getter | Attribute type | Public | Get attribute value |
| Setter | void | Public | Set value of attribute |

Table 34: <Desktop App> Repository Method

* + 1. <Desktop App> Order

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| OrderId | int | Private | Unique identifier of Order, auto increment. |
| OrderCode | string | Private | Code of an item. |
| OrderDate | string | Private | Date create Order |
| Total | float | Private | Total price of order |
| CustomerName | string | Private | Name of customer. |
| CustomerEmail | string | Private | Email of customer. |
| CustomerPhone | string | Private | Phone of customer. |
| CustomerAddress | string | Private | Address of customer. |

Table 35: <Desktop App> Order Attribute

Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| Getter | Attribute type | Public | Get attribute value |
| Setter | void | Public | Set value of attribute |

Table 36: <Desktop App> Order Method

* + 1. <Desktop App> OrderDetail

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| OrderDetailId | int | Private | Unique identifier of Order Details, auto increment. |
| OrderId | int | Private | ID of Order. |
| MarbleId | int | Private | ID of Marble. |
| Quantity | int | Private | Quantity of Marble |
| UnitPrice | float | Private | Price of an Marble |

Table 37: <Desktop App> OrderDetail Attribute

Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| Getter | Attribute type | Public | Get attribute value |
| Setter | void | Public | Set value of attribute |

Table 38: <Desktop App> OrderDetail Method

* + 1. <Desktop App> PortSetting

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| PortId | int | Private | Unique identifier of Port, auto increment. |
| Name | string | Private | Name of Port. |
| Value | string | Private | Value of Port. |

Table 39: <Desktop App> PortSetting Attribute

Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| Getter | Attribute type | Public | Get attribute value |
| Setter | void | Public | Set value of attribute |

Table 40: <Desktop App> PortSetting Method

* + 1. <K – Nearest Neighbor> KSelection

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| corlorName | uint8\_t | Private | Color of marble |
| distanceValue | float | Private | Distance from new marble to sample marble |

Table 41: < K – Nearest Neighbor > KSelection Attribute

Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| get | Attribute type | Public | Get attribute value |
| set | void | Public | Set value of attribute |

Table 42: < K – Nearest Neighbor > KSelection Method

* + 1. <K – Nearest Neighbor> ColorAnalysisItem

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| corlorName | uint8\_t | Private | Color of marble |
| count | int | Private | Count of marble |
| sum | float | Private | Sum of Distance |
| mean | float | Private | Mean of Distance |

Table 43: < K – Nearest Neighbor > ColorAnalysisItem Attribute

Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| get | Attribute type | Public | Get attribute value |
| set | void | Public | Set value of attribute |

Table 44: < K – Nearest Neighbor > ColorAnalysisItem Method

* + 1. <K – Nearest Neighbor> ColorAnalysisTable

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| listItem | ColorAnalysisItem | Private | List Marble |
| maxCountValue | int | Private | Max count value of marble |

Table 45: < K – Nearest Neighbor > ColorAnalysisTable Attribute

Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| calAnalysisTable | void | Public | Create analysis table for calculating |
| matchColor | uint8\_t | Public | If two or more colors have same quantity sample, this function will find color that have smallest mean value |
| findMaxCountInArray | int | Public | Find nearest color |
| findNumOfMaxCount | int | Public | Find colors have same marbles |

Table 46: < K – Nearest Neighbor > ColorAnalysisTable Method

* + 1. <K – Nearest Neighbor> ColorDetection

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| percentageRed | uint8\_t | Private | Percent Red of marble |
| percentageGreen | uint8\_t | Private | Percent Green of marble |
| percentageBlue | uint8\_t | Private | Percent Blue of marble |
| frequencyClear | float | Private | Frequency of clear |

Table 47: < K – Nearest Neighbor > ColorDetection Attribute

Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| get | Attribute type | Public | Get attribute value |
| set | void | Public | Set value of attribute |
| findClosetColor | uint8\_t | Public | Find color of marble |

Table 48: < K – Nearest Neighbor > ColorDetection Method

* + 1. <K – Nearest Neighbor> Global

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| sample | uint8\_t | Private | Sample of 5 colors |

Table 49: < K – Nearest Neighbor > Global Attribute

* 1. Flowchart Diagram
     1. <User>Input Marble

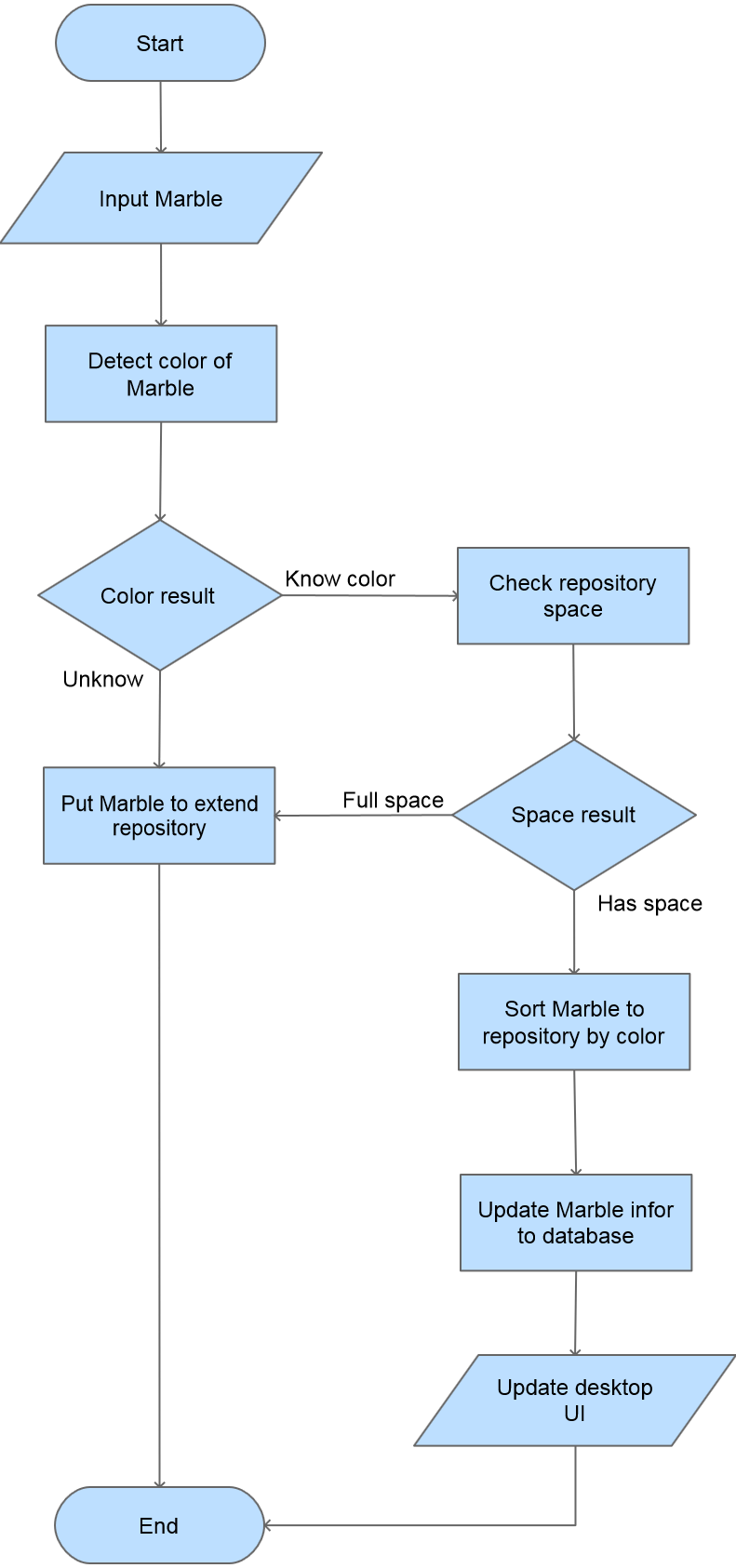


Figure 33: <User> Input Marble Flowchart Diagram

* + 1. <User>Output Marble

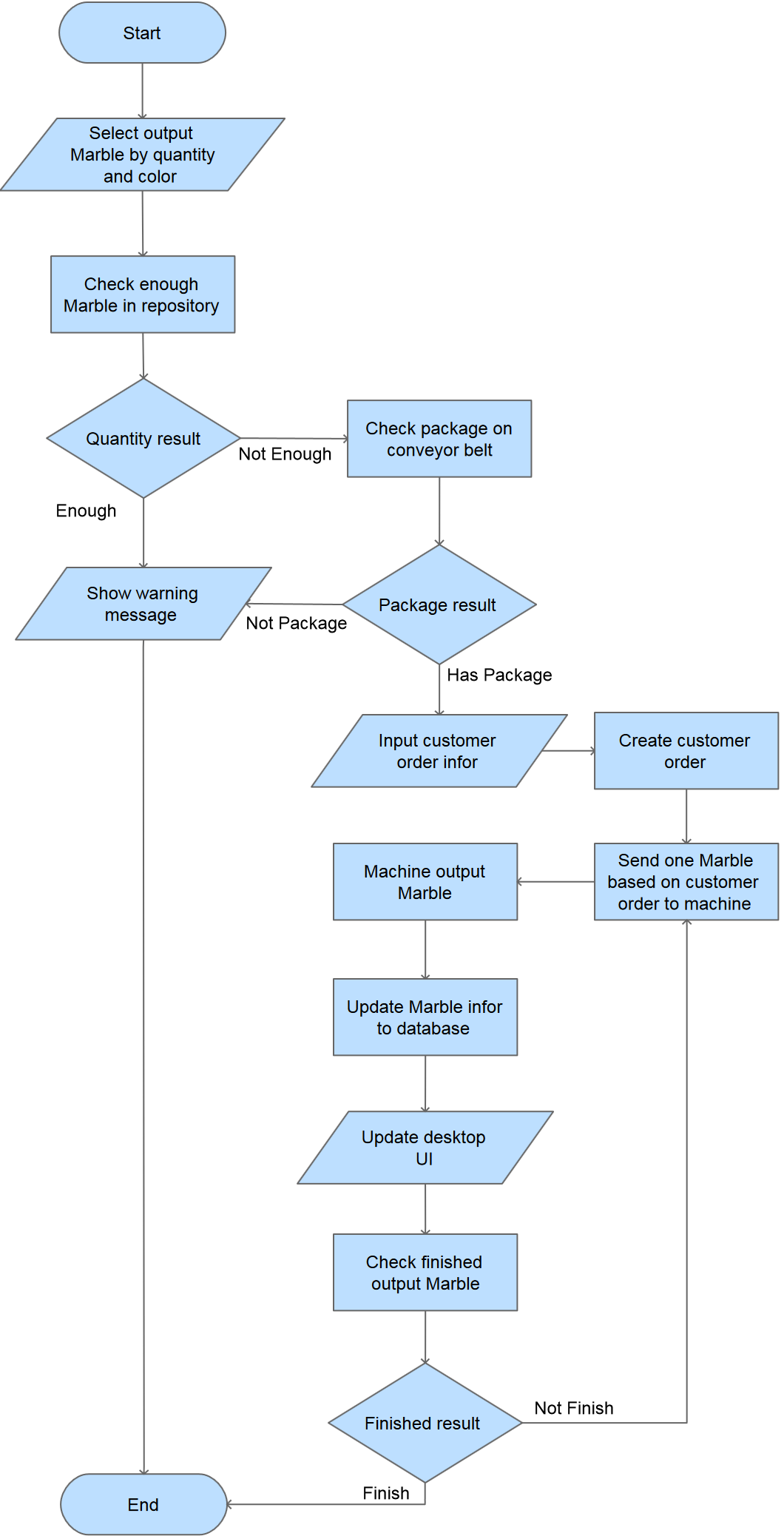


Figure 34: <User> Output Marble Flowchart Diagram

* + 1. <User>View Order Details

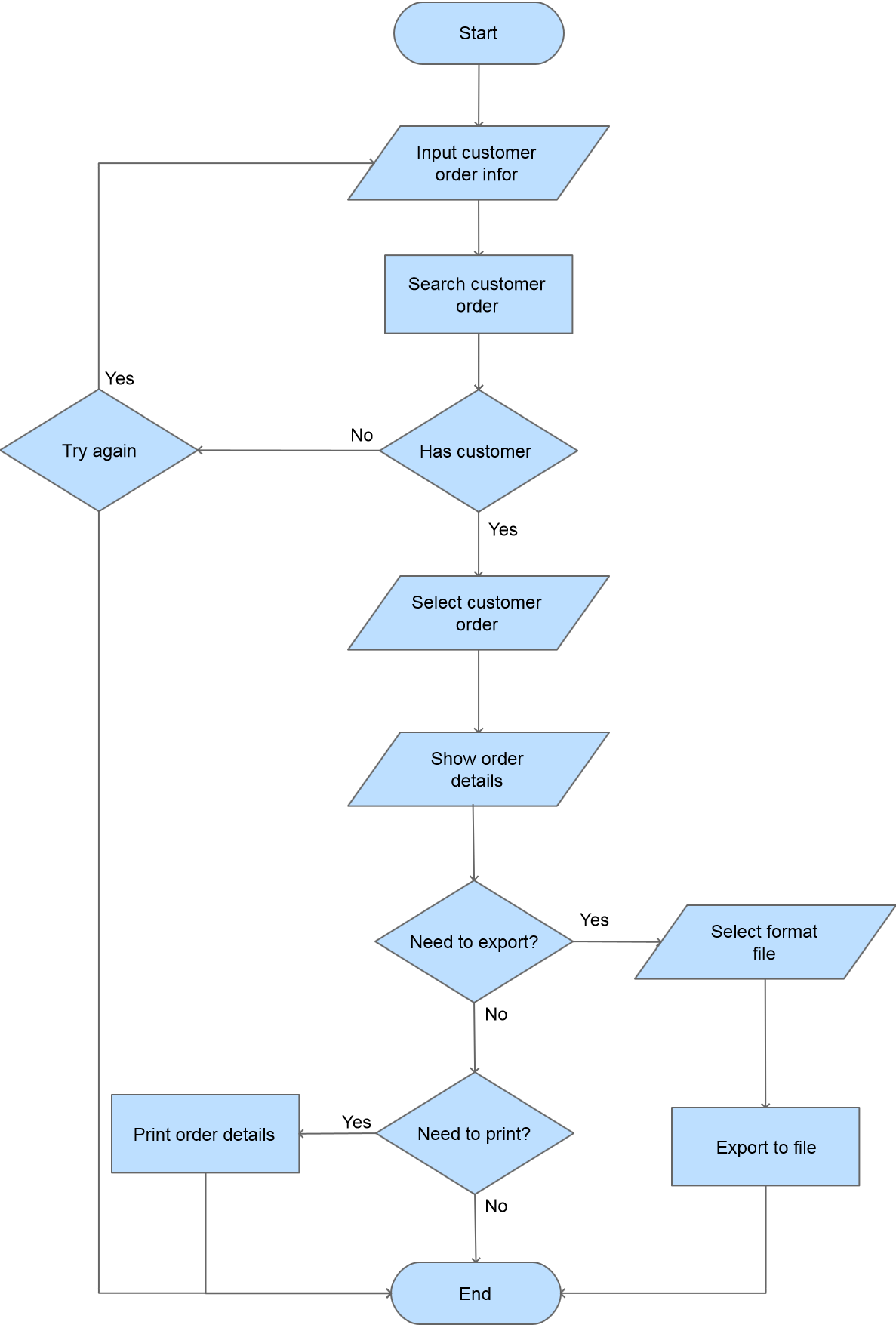


Figure 35: <User> View Order Details Flowchart Diagram

* + 1. <User>Configure Ports

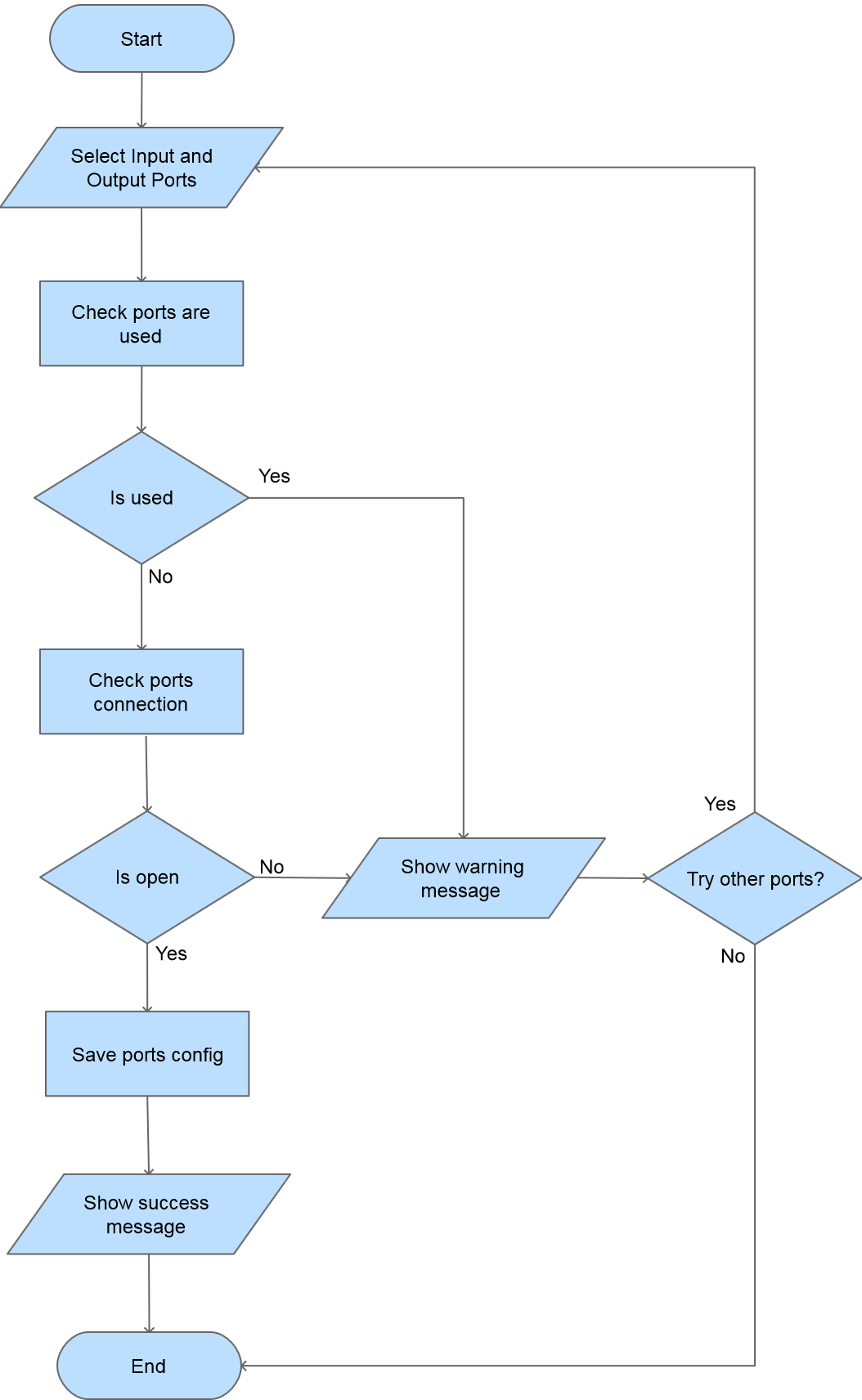


Figure 36: <User> Configure Ports Flowchart Diagram

* 1. Sequence Diagram
     1. <User> Input Marble

Summary: This program shows how user inputs new marble into the system.



Figure 37: <User> Input Marble Sequence Diagram

* + 1. <User> Output Marble

Summary: This program shows how user gets marbles out of system.



Figure 38: <User> Output Marble Sequence Diagram

* + 1. <User> View Order Details

Summary: This program shows how user views customer order details.



Figure 39: <User> View Order Details Sequence Diagram

* + 1. <User> Configure Ports

Summary: This diagram shows how user configures ports between desktop application and product sorter machine.



Figure 40: <User> Configure Ports Sequence Diagrams

* 1. Activity Diagram
     1. <Hardware> Input Marble Details

Summary: This diagram shows how the machine detects color of marbles and sort them when user input.

****

Figure 41: <Hardware> Input Marble Details Activity Diagram

|  |  |  |
| --- | --- | --- |
| 1. Start activity. User input marble to machine. | 2. Arduino controls infrared sensor to input marbles. | 3. Infrared sensor check marble input or not. |
| 4. Infrared sensor returns checking value. | 5. Marble is not input. Check again. | 5a. Has marble. |
| 6a. Arduino send PWM to open 2 servos to accept only one marble at the same time. | 7a. Arduino sends high value to turn on other infrared sensor to check position of marble. | 8a. Infrared sensor check position current marble. |
| 9a. Infrared sensor returns checking value. | 10a. No marble in front of color sensor. Check again. | 11a. Has marble in front of color sensor. |
| 12a. Arduino sends slow value to turn off Infrared sensor. | 13a. Arduino controls color sensor. | 14a. Arduino send signal to pin S1, S2, S3, S4 of color sensor. |
| 15a. Color sensor detects color of marble. | 16a. Color sensor returns value to Arduino. | 17a. Arduino calculates frequency of color and percentage. |
| 18a. Arduino selects infrared sensor of repository based on color. | 19a. Arduino controls this infrared sensor. | 20a. Infrared sensor checks this repository is full or not. |
| 21a. Infrared sensor returns checking value. | 22a. Repository is full, Arduino sends PWM to all servos. | 23a. All servos close. |
| 24a. Open servo at color sensor. | 25a. Marble runs to extend repository. | 26a. End activity. |
| 22b. Select one servo of repository based on color. | 23b. Arduino does two task at the same time. | 24b. Send information of input marble to desktop app to save to database. |
| 25b. End activity. | 26b. Open servo at color sensor. | [After 26b step, continue with 2 steps: 25a and 26a.] |

Table 50: <Hardware> Input Marble Details Activity Diagram

* + 1. <Hardware> Output Marble Details

Summary: This diagram shows how the machine get marbles.



Figure 42: <Hardware> Output Marble Details Activity Diagram

|  |  |  |
| --- | --- | --- |
| (A) is the point to start or loop all the step form other steps. If user selects many packages, this point is used. | | |
| 1. Start activity. User selects marble to get out. | 2. User clicks “Get Marbles” button. | 3. Desktop app checks real marbles in repository. |
| 4. Return value after checking repository. | 5. Not enough marbles for user get out. | 6. End activity. |
| 5a. Has enough marble for user get out. | 6a. send code “1001” to Arduino. | 7a. Arduino sends PWM to DC motor to run conveyor belt. |
| 8a. Infrared sensor starts checking. | 9a. Infrared sensor checks package for output marbles. | 10a. Infrared sensor returns checking value. |
| 11a. No packages on conveyor belt, Arduino starts to check waiting time. | 12a. Get current distance of time form starting conveyor belt. | 13a. Time < 3s, continue to run conveyor belt. |
| 13b. Time > = 3s. | 14b. Stop conveyor belt. | 15b. Arduino sends code “2101” to desktop app. |
| 16b. Show message for user. | 17b. End activity. | 11c. Has package on conveyor belt. |
| 12c Stop conveyor belt to start getting marbles out. | 13c. Arduino sends code “2001” to decktop application. | 14c. Desktop app shows customer order form to get information of customer. |
| 15c. User fills customer forms. | 16c. User click “Save” button. | 17c. Desktop app creates customer order information and saves to database. |
| 18c. Desktop app sends first marble information based on customer order. | 19c. Arduino selects 2 servos of repository based on color. | 20c. Arduino sends PWM to these servos. |
| 21c. These servos filter only one marble to get out. | 22c. Desktop app updates database and GUI. | 23c. Desktop app checks output marbles finish or not. |
| 24c. Return checking value. | 25c. If not finished, run again from step 18c.. | 25d. all marbles output finished |
| 26d. Desktop app sends code “1002” to Arduino. | 27d. Arduino sends PWM to run conveyor belt in 3s. | 28d. Run finished, send code “2002” to desktop app to confirm. |
| 29d. Desktop app checks quantity of packages. | 30d. Return checking value. | 31d. Has more packages, go to step (A). |
| 31d. No package. End activity. |  |  |

Table 51: <Hardware> Output Marble Details Activity Diagram

* + 1. <User> Input Marble

Summary: This diagram shows how the machine detects color of marbles and sort them when user input.



Figure 43: <User> Input Marble Activity Diagram

|  |  |  |
| --- | --- | --- |
| 1. User inputs marbles to system. | 3. Machine detect marble color. | 3. Return detecting value. |
| 4. Detect fails, put marble to extend repository. | 4a. Detect success, start check repository based on color. | 5a. Return checking value. |
| 6a. Full space, put marble to extend repository. | 6b. Has space in repository for this marble. | 7b. Machine moves marble to right repository. |
| 8b. End activity. | 9b. Machine sends color information for desktop app. | 10b. Desktop app, get information of input marble. |
| 11b. Update repository in database. | 12b. Update repositories in desktop app. | 13b. End activity. |

Table 52: <User> Input Marble Activity Diagram

* + 1. <User> Output Marble

Summary: This diagram shows how the machine detects color of marbles and sort them when user input.



Figure 44: <User> Output Marble Activity Diagram

|  |  |  |
| --- | --- | --- |
| 1. Start activity. User selects “Repository” menu. | 2. Get current marble quantity. | 3. Load quantity to desktop app. |
| 4. User selects output marbles. | 5. User clicks “Get Marbles” button to start activity. | 6. Check real marble quantity. |
| 7. Return checking value. | 8. Not enough marbles, show-warning message. | 9. End activity. |
| 8a. Enough marbles, check package on conveyor belt. | 9a. Return checking result. | 10a. Send checking result to desktop app. |
| 11a. No package, show warning message. | 11b. Desktop app shows order form to get customer order information. | 12b. Users fill customer information. |
| 13b. User clicks “Save” button. | 14b. Desktop app starts to create order. | 15b. Start activity. |
| 16b. Call insert order to database. | 17b. Send one marble information to machine. | 18b. Machine output marble. |
| 19b. Send success message to confirm. | 20b. Desktop app start activity. Call received message. | 21b. Update quantity of repository to database. |
| 22b.Update quantity to desktop UI. | 23b. Start to check all marbles output finished or not. | 24b. Return checking value. |
| 25b. Result is finished. End activity. | 25c. Result is not finish, start get continuous form step17b. |  |

Table 53: <User> Output Marble Activity Diagram

* + 1. <User> View Order Details

Summary: This diagram shows how the machine detects color of marbles and sort them when user input.



Figure 45: <User> View Report Details

|  |  |  |
| --- | --- | --- |
| 1. Start activity. User clicks “Report” menu. | 2. List all order. | 3. Show list order on desktop app. |
| 4. User finds customer order. | 5. Start activity. | 6. Has order, show result. |
| 8. User selects one customer order. | 9. Desktop app shows order details form. | 10. Get details of customer. |
| 11. Update desktop UI. | 6a. Not find any customer order. End activity. | 12a. User click “Export” icon, desktop app exports to word, excel or pdf. |
| 13a. End activity. | 12b. User clicks “Print” icon. | 13b. Send report to printer to print. |
| 14b. End activity. |  |  |

Table 54: <User> View Report Details

* + 1. <User> Configure Ports

Summary: This diagram shows how the machine detects color of marbles and sort them when user input.



Figure 46: <User> Configure Ports Activity Diagram

|  |  |  |
| --- | --- | --- |
| 1. Start activity. User click “Configuration” menu. | 2. Desktop app get list current ports. | 3. User selects input port and output port. |
| 4. User click “Testing” button to test connection. | 5. Start to check ports connection. | 6. Return checking value. |
| 7. Port open, show success message. | 8. End activity. | 4a. Use clicks “Save” button. |
| 5a. Update ports to database. | 6a. End activity. | 7b. Port not open, show error message. |
| End activity. |  |  |

Table 55: <User> Configure Ports Activity Diagram

1. User Interface Design
   1. Repository Interface Design
      1. Manage Repository

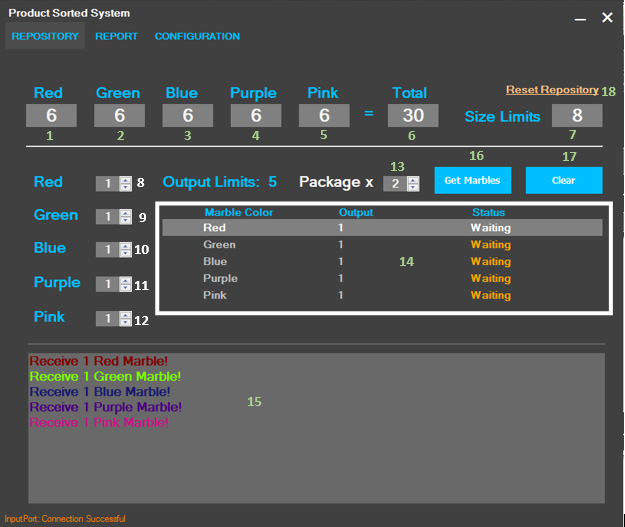


Figure 47: Manage Repository Interface

Fields

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Description** | **Read only** | **Mandatory** | **Control Type** | **Data Type** | **Length** |
| 1 | Red Repository | Number red Marble | Yes | Yes | Textbox | string | N/A |
| 2 | Green Repository | Number Green Marble | Yes | Yes | Textbox | string | N/A |
| 3 | Blue Repository | Number Blue Marble | Yes | Yes | Textbox | string | N/A |
| 4 | Purple Repository | Number Purple Marble | Yes | Yes | Textbox | string | N/A |
| 5 | Pink Repository | Number Pink Marble | Yes | Yes | Textbox | string | N/A |
| 6 | Total Repository | Total Marble | Yes | Yes | Textbox | string | N/A |
| 7 | Size Limits | Limit Number Marble of Repository | Yes | Yes | Textbox | string | N/A |
| 8 | Output Red Marble | Select number of red marbles | Yes | Yes | Textbox | string | N/A |
| 9 | Output Green Marble | Select number of green marbles | Yes | Yes | NumericUpDown | decimal | N/A |
| 10 | Output Blue Marble | Select number of blue marbles | Yes | Yes | NumericUpDown | decimal | N/A |
| 11 | Output Purple Marble | Select number of purple marbles | Yes | Yes | NumericUpDown | decimal | N/A |
| 12 | Output Pink Marble | Select number of pink marbles | Yes | Yes | NumericUpDown | decimal | N/A |
| 13 | Output Package | Select number of packages | Yes | Yes | NumericUpDown | decimal | N/A |
| 14 | Output Status | Update real time output marbles | Yes | Yes | DataGridView | N/A | N/A |
| 15 | Log | Log all input and output marbles | Yes | Yes | RichTextbox | N/A | N/A |

Table 56: <Fields> Manage Repository Interface

Button/Hyperlinks

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Function** | **Description** | **Validation** | **Outcome** |
| 16 | Get marbles | Get marble based on customer order | Number Marble select less than 5 and total marbles in repository | Transfer to Customer Form |
| 17 | Clear | Clear all selected marbles | N/A | N/A |
| 18 | Reset repository | Reset repository and database to default value | N/A | N/A |

Table 57: < Button/Hyperlinks> Manage Repository Interface

* + 1. Create Customer Profile

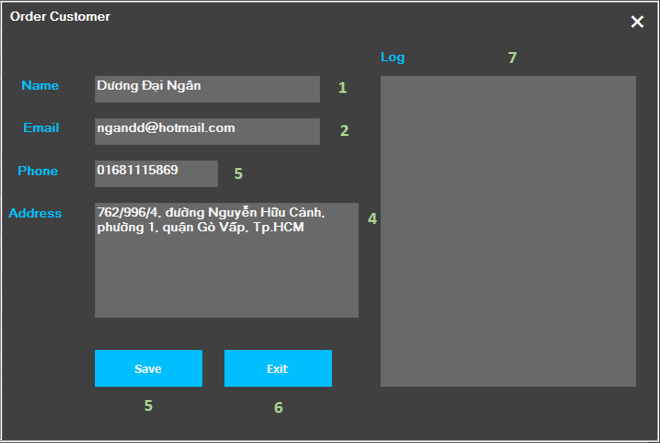


Figure 48: Create Customer Profile Interface

Fields

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Description** | **Read only** | **Mandatory** | **Control Type** | **Data Type** | **Length** |
| 1 | Name | Fill customer name | No | Yes | TextBox | string | N/A |
| 2 | Email | Fill customer email | No | Yes | TextBox | string | N/A |
| 3 | Phone | Fill customer phone | No | Yes | TextBox | string | N/A |
| 4 | Address | Fill customer address | No | Yes | TextBox | string | N/A |
| 5 | Log | Show validate or error message | Yes | Yes | RichTextBox | string | N/A |

Table 58: <Fields> Create Customer Profile Interface

Button/Hyperlinks

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Function** | **Description** | **Validation** | **Outcome** |
| 6 | Save | Save customer information and start output marble | Valid Email | Transfer to Repository Form |
| 7 | Exit | Exit customer form | N/A | Transfer to Repository Form |

Table 59: < Button/Hyperlinks> Create Customer Profile Interface

* 1. Report Interface Design
     1. Search Customer Order

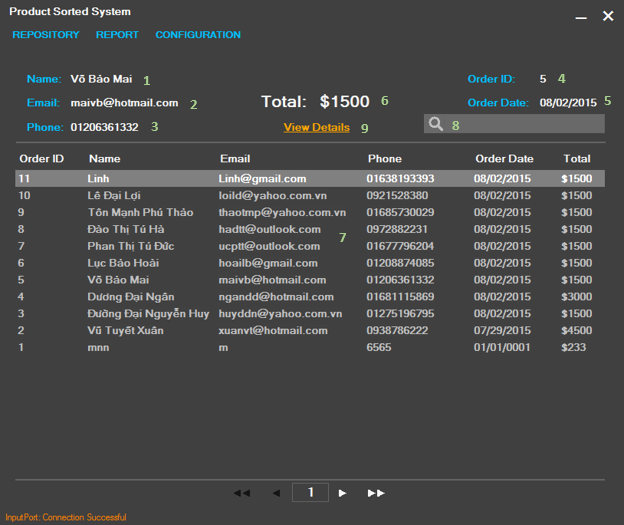


Figure 49: Search Customer Order Interface

Fields

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Description** | **Read only** | **Mandatory** | **Control Type** | **Data Type** | **Length** |
| 1 | Name | Display customer name | Yes | Yes | Label | string | N/A |
| 2 | Email | Display customer email | Yes | Yes | Label | string | N/A |
| 3 | Phone | Display customer phone | Yes | Yes | Label | string | N/A |
| 4 | Order ID | Order ID | Yes | Yes | Label | string | N/A |
| 5 | Order Date | Date of make one order | Yes | Yes | Label | string | N/A |
| 6 | Total | Total price of order | Yes | Yes | Label | string | N/A |
| 7 | Order Information | List history order | Yes | Yes | DataGridView | string | N/A |
| 8 | Search | Search by order Information | NO | No | Textbox | string | N/A |

Table 60: <Fields> Search Customer Order Interface

Button/Hyperlinks

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Function** | **Description** | **Validation** | **Outcome** |
| 8 | View Details | View Order Details | Selected Order | Transfer to Report Details form |

Table 61: < Button/Hyperlinks> Search Customer Order Interface

* + 1. View Order Details

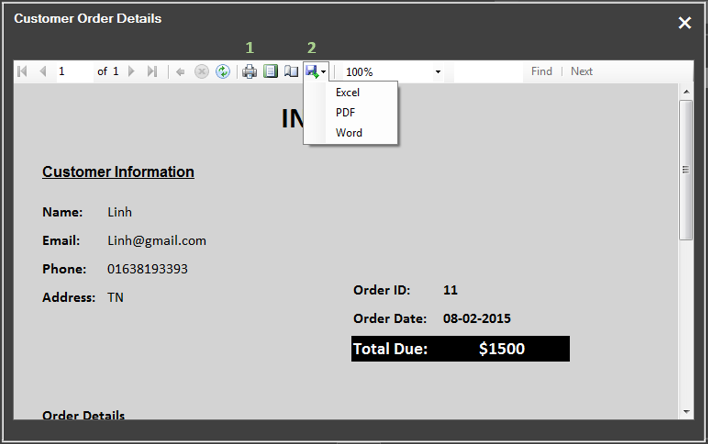


Figure 50: View Order Details Interface

Fields

N/A

Button/Hyperlinks

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Function** | **Description** | **Validation** | **Outcome** |
| 1 | Print | Print order detail | N/A | N/A |
| 2 | Export | Export order detail | N/A | N/A |

Table 62: < Button/Hyperlinks> View Order Details Interface

* 1. Configuration Interface Design

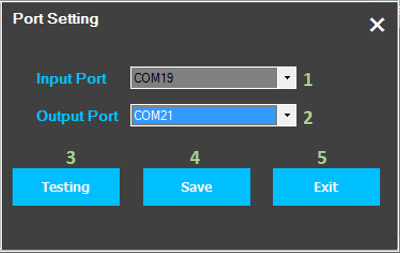


Figure 51: Configure Ports Interface

Fields

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Description** | **Read only** | **Mandatory** | **Control Type** | **Data Type** | **Length** |
| 1 | Input Port | Input Port of System | Yes | Yes | ComboBox | String | N/A |
| 2 | Output Port | Output Port of System | Yes | Yes | ComboBox | String | N/A |

Table 63: <Fields> Configure Ports Interface

Button/Hyperlinks

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Function** | **Description** | **Validation** | **Outcome** |
| 3 | Testing | Testing port connection | N/A | N/A |
| 4 | Save | Save port to database | N/A | N/A |
| 5 | Exit | Exit configuration form | N/A | Transfer to Repository Form |

Table 64: < Button/Hyperlinks> Configure Ports Interface

1. Database Design
   1. Logical Diagram

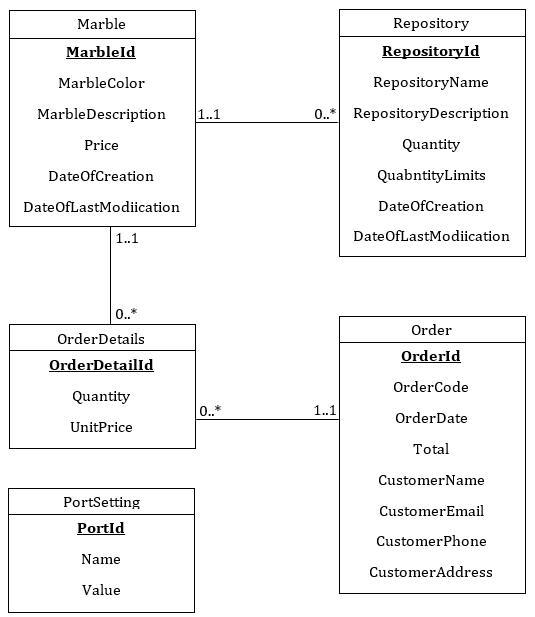


Figure 52: Logical Diagram

* 1. Data Dictionary

|  |  |
| --- | --- |
| **Entity Data dictionary: describe content of all entities** | |
| **Entity Name** | **Description** |
| Marble | Describe all marble information in the system. |
| Repositories | Describe all repository for each marble in the system. |
| Orders | Describe all orders stored in the system. |
| OrderDetails | Describe all order detail in the system. |
| PortSettings | Describe all port settings store in the system. |

Table 65: Entity Data Dictionary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity name** | **Attributes** | **Description** | **Domain** | **Null** |
| Marble | MarbleId | Unique identifier of Marble, auto increment. | int | No |
| MarbleColor | Color of Marble, which system uses to detect. | nvarchar(50) | Yes |
| MarbleDescription | Description of Marble | nvarchar(MAX) | Yes |
| Price | Price of Marble | float | Yes |
| DateOfCreation | Date Marble Create | datetime | Yes |
| DateOfLastModification | Last time Modification of marble | datetime | Yes |
| Repositories | RepositoryId | Unique identifier of Repository, auto increment. | int | No |
| RepositoryName | Repository’s name | nvarchar(50) | Yes |
| RespositoyDescription | Repository’s Description | nvarchar(MAX) | Yes |
| Quantity | Current number of marbles in Repository. | int | Yes |
| QuantityLimits | Max Quantity of Repository | int | Yes |
| DateOfCreation | Date create repository | datetime |  |
| DateOfLastModification | Last time modify repository | datetime | Yes |
| Orders | OrderId | Unique identifier of Order, auto increment. | int | No |
| OrderCode | Code of order. | nvarchar(MAX) | Yes |
| OrderDate | Date create Order | datetime | Yes |
| Total | Total price of order | float | Yes |
| CustomerName | Customer’s name. | nvarchar(50) | Yes |
| CustomerEmail | Customer’s email. | nvarchar(50) | Yes |
| CustomerPhone | Customer’s phone. | nvarchar(20) | Yes |
| CustomerAddress | Customer’s address. | nvarchar(MAX) | Yes |
| OrderDetails | OrderDetailId | Unique identifier of Order Details, auto increment. | int | No |
| Quantity | Quantity of order marble based on color. | int | Yes |
| UnitPrice | Price of an Marble | float | Yes |
| PortSettings | PortId | Unique identifier of Port, auto increment. | int | No |
| Name | Port’s name. | nvarchar(50) | Yes |
| Value | Port’ value. | nvarchar(50) | Yes |

Table 66: Entity Attribute Data Dictionary

1. Algorithms
   1. K – Nearest Neighbor Algorithms
      1. Definition

K – Nearest Neighbors is a type of instance-based learning, or lazy learning. The purpose of the K - Nearest Neighbors (K-NN) algorithm is the algorithm used to specify new form or new sample (called Query point) based on the attributes and class of the available sample (called Training Data), the samples are located in a system or database table called the sample space.

For more details information:

<http://www.saedsayad.com/k_nearest_neighbors.htm>

* + 1. Define Problem

Color sensor has four basic color filters, which used to transform a random color to parameters. These parameters is represent to frequency of three basic color: Red, Green, Blue and Clear. There are three methods to handle data, which we get from sensor:

Method 1: Transform color to RGB model. RGB space is the industry standard for computer graphics so we can identify constituent and percentage of three basic color. Because we need to transform from data to RGB model many times to calculate, it takes a lot of time. The signal may be not exactly because it depends on hardware and environment.

Method 2: Transform color to HSL model shown color more detail and algorithm is supported a lot. However, same as RGB, it take a lot of time to calculate and the signal may be not exactly because it depends on hardware and environment.

Method 3: Find the nearest sample color with the color, which we need to identify. With this method, we do not need to have knowledge about color, handle all form of return signal (frequency, wavelength…) with the only algorithm and we also handle interferences of signal. However, it require a lot of RAM to save the sample environment, which we must collected manually.

With the simple request of this system, we choose method 3 to determine color of the object. The algorithm is K-Nearest Neighbors algorithm.

* + 1. Solution

To solve this problem, we should follow these steps:

* Step 1: Collect a training set of all color we need to identify.
* Step 2: Get signal or information of new sample to examine.
* Step 3: Calculate the distance from new sample to k points from training set (k is an integer – depending your choice). After that, sort result ascending.
* Step 4: Identify new sample.

Example:

Step 1: Collect a training set:

In this step, after measuring some red and blue marbles many times by hardware program. Because color sensor only returns four values, include frequency of red color, frequency of green color, frequency of blue color and frequency of clear color (or origin frequency). Therefore, we would standardize data in percentage. We have:

- Red percent = Red frequency/Clear frequency.

- Green percent = Green frequency/Clear frequency.

- Blue percent = Blue frequency/Clear frequency.

In this example, we only choose two colors: Red and Blue

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Color | % Frequency Red | % Frequency Green | % Frequency Blue |
| 1 | BLUE | 13 | 29 | 50 |
| 2 | BLUE | 14 | 31 | 52 |
| 3 | RED | 65 | 19 | 25 |
| 4 | RED | 65 | 18 | 24 |
| 5 | RED | 65 | 18 | 24 |

Table 67: Color Sample of Marbles

Step 2: When a marble is detected by TCS3200 color sensor, we have new sample (or Query point).

|  |  |  |
| --- | --- | --- |
| New Sample (Or Query point) | | |
| 13 | 30 | 51 |

Table 68: New Sample or Query Point

Step 3: Calculate the distance from Query point to all elements in training set.

We use Euclidean Distance to calculate distance between of two points:

We have the results:

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Color Sample** | **Query Point** | **Distance** |
| 1 | BLUE(13, 29, 50) | Q(13, 30, 51) | 1.414213562 |
| 2 | BLUE(14,31,52) | 1.73050808 |
| 3 | RED(65,19,25) | 59.316924877 |
| 4 | RED(65,18,24) | 59.80802622 |
| 5 | RED(65,18,24) | 59.80802622 |

Table 69: Distance between Query Point and Color Sample

Step 4: Identify new sample:

* K =1: We can see if we choose only one sample from training set, the shortest distance from new sample to one point in training set is 1.414213562. Therefore, we conclude that: new sample is BLUE.
* K = 2: In this case, we can see have 2 points from training set nearest to new sample. In addition, all of them are BLUE. Therefore, we conclude that: new sample is BLUE.
* K = 3: In this case, we have 2 BLUE points and 1 RED point, so we can concluded that: new sample is BLUE
* K = 4: In this case, we have 2 BLUE points and 2 RED points. Therefore, we cannot give concluded. We need to calculate the average of them:

BLUE average = (1.414213562 + 1.732050808)/2 = 1.573132185

RED average = (59.16924877 + 59.80802622)/2 = 59.4886375

We see that, BLUE average is smaller than RED average. Therefore, we conclude that: sample is BLUE.

* K = 5: In this case, we have 2 BLUE points and 3 RED points. Therefore, we conclude that: new sample is RED.

We have problem when we choose K = 5. Although the input marble is BLUE, but the result RED. This problem appears when we choose samples in training set unequal (3 RED and 2 BLUE). To solve this problem, we should select samples of each colors equal in training set.

* + 1. Complexity

Suppose we have n samples in training set. Each of sample has dimension d. Therefore, we have:

* O(d) to compute distance to one sample.
* O(nd) to find one nearest neighbor, we must calculate to all n elements in training set.
* O(knd) to select k nearest samples to conclude.

So the total complexity is O(knd). However, we need a large number of samples for K – Nearest Neighbor to work well!

* + 1. Flowchart

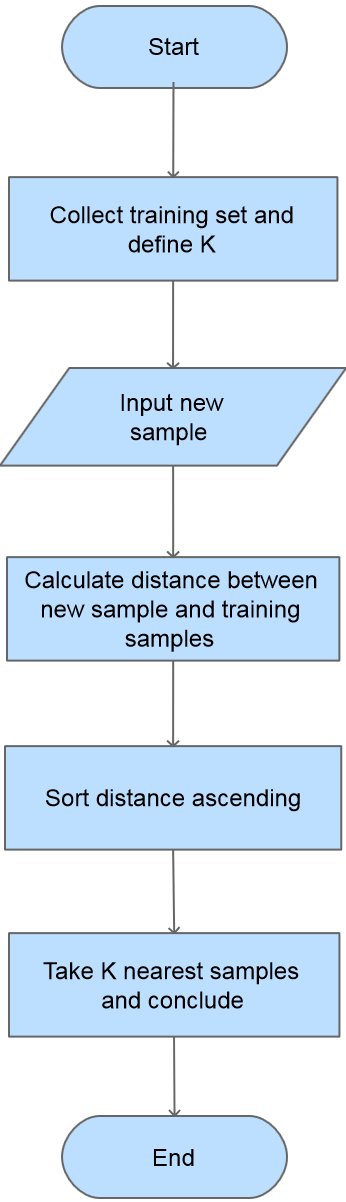


Figure 53: K – Nearest Neighbor Flowchart

1. System Implementation & Test
2. Introduction
   1. Overview

In this section, we provide all information about implementation information and testing procedure of PSSC system includes test plans, test cases, test result and the reason details for failed test cases.

In some test cases, we also make summary based on test results to improve performance of PSSC system.

* 1. Test Approach

- Goal of testing: to ensure all hardware work well and system meet requirement from user.

- Method: Component testing, system testing and black box testing.

1. Database Relationship Diagram
   1. Physical Diagram

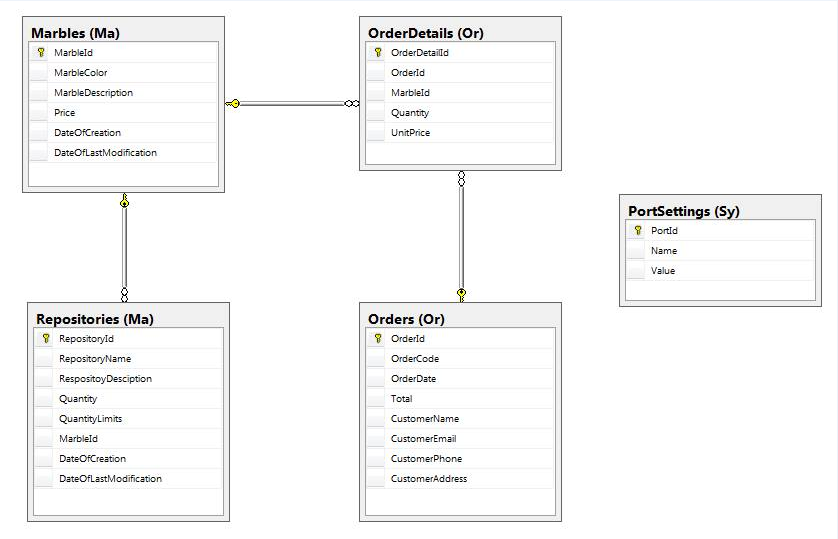


Figure 54: Physical Diagram

* 1. Data Dictionary

|  |  |
| --- | --- |
| **Table Data dictionary: describe content of all tables** | |
| **Table Name** | **Description** |
| Marble | Describe all marble profiles in the system. |
| Repositories | Describe all repository profiles in the system. |
| Orders | Describe all orders stored in the system. |
| OrderDetails | Describe all order detail in the system. |
| PortSettings | Describe all port settings store in the system. |

Table 70: Table Data dictionary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table name** | **Attributes** | **Description** | **Domain** | **Null** |
| Marble | MarbleId {PK} | Unique identifier of Marble, auto increment. | int | No |
| MarbleColor | Color of Marble | nvarchar(50) | Yes |
| MarbleDescription | Description of marble | nvarchar(MAX) | Yes |
| Price | Price of marble | float | Yes |
| DateOfCreation | Date create marble | datetime | Yes |
| DateOfLastModification | Date modify marble | datetime | Yes |
| Repositories | RepositoryId {PK} | Unique identifier of Repository, auto increment. | int | No |
| RepositoryName | Repository’s name | nvarchar(50) | Yes |
| RespositoyDescription | Repository’s Description. | nvarchar(MAX) | Yes |
| Quantity | Quantity of marble in repository. | int | Yes |
| QuantityLimits | Max Quantity of marble in Repository | int | Yes |
| MarbleId {FK} | ID of marble | int | Yes |
| DateOfCreation | Date create repository. | datetime |  |
| DateOfLastModification | Date modify repository. | datetime | Yes |
| Orders | OrderId {PK} | Unique identifier of Order, auto increment. | int | No |
| OrderCode | Code of an item. | nvarchar(MAX) | Yes |
| OrderDate | Date create Order | datetime | Yes |
| Total | Total price of order | float | Yes |
| CustomerName | Name of customer. | nvarchar(50) | Yes |
| CustomerEmail | Email of customer. | nvarchar(50) | Yes |
| CustomerPhone | Phone of customer. | nvarchar(20) | Yes |
| CustomerAddress | Address of customer. | nvarchar(MAX) | Yes |
| OrderDetails | OrderDetailId {PK} | Unique identifier of Order Details, auto increment. | int | No |
| OrderId {FK} | Order’s Id. | int | Yes |
| MarbleId {FK} | Marble’s Id. | int | Yes |
| Quantity | Quantity of order marble. | int | Yes |
| UnitPrice | Price of an marble. | float | Yes |
| PortSettings | PortId {PK} | Unique identifier of Port, auto increment. | int | No |
| Name | Port’ name. | nvarchar(50) | Yes |
| Value | Port’s value. | nvarchar(50) | Yes |

Table 71: Table Attribute Data Dictionary

1. Performance Measures

* Time for detecting one marble: 2 ~ 5 seconds.

1. Test Plan
   1. Features to be tested

We separate two parts for testing: hardware and software.

* + 1. Hardware

After assembling hardware devices to mechanical part, we test all them to ensure that all hardware devices working well.

The list hardware devices that we will test:

|  |  |
| --- | --- |
| Function | Description |
| TCS3200 Color Sensor | * We test with five basic colors red, green, blue, purple and pink. * Each marbles we test 100 times. If ratio of detect successful case is large than 80%, the result is “Passed”, otherwise the result is “Failed”.   Ratio = [Number of success case] / 100 times |
| 18 x Servo RC | * For each servo after testing, we need to find the right position – the angel of servo when it opens and closes. |
| 8 x Infrared Sensor | * We check the return value of this sensor when the marble is in front of and otherwise. |

To see the testing results, we use Arduino board to control these devices with some basic codes and displays status of them on monitor screen.

* + 1. Software

The features of user will be focused and list below:

|  |  |
| --- | --- |
| Function | Description |
| Input marble | * We input marbles with five colors (red, green, blue, purple, pink) many times to ensure that system detect successfully. * We also test with some special marbles like black marble, dirty marble or marble with decal paper to find the boundary of system. |
| Output marble | * User selects number of marbles by application; the machine will get marbles based on this information. |
| View Order Details | * User selects history order and view details. The extend functions that we will test are export order details. |

* 1. Features not to be tested

We do not test features:

* Check connection between PSSC machine and desktop application.
* Check user input the box on conveyor belt.
* Manage marble.
* View order details.

1. System Testing Test Case
   1. Component Testing
      1. TCS3200 Sensor Color

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Test Case Description | Test Case Procedure | **Expected output** | **Inter-test Case Dependence** | **Result** | **Test Date** | **Note** |
| TC001 | TCS230 Sensor with red marble. | After integrating this sensor to mechanical part, input one red marble directly by hand. | - Serial Monitor of Arduino IDE display: “Red”. | N/A | Test 100 times:  - Successful: 96  - Failed: 4  Conclude:  - Ratio: 96%  - Result: Pass | 20/07/2015 | With 4 failed case, the color that sensor detect is pink. |
| TC002 | TCS230 Sensor with green marble. | After integrating this sensor to mechanical part, input one green marble directly by hand. | - Serial Monitor of Arduino IDE display: “Green”. | N/A | Test 100 times:  - Successful: 100  - Failed: 0  Conclude:  - Ratio: 100%  - Result: Pass | 20/07/2015 |  |
| TC003 | TCS230 Sensor with blue marble. | After integrating this sensor to mechanical part, input one blue marble directly by hand. | - Serial Monitor of Arduino IDE display: “Blue”. | N/A | Test 100 times:  - Successful: 98  - Failed: 2  Conclude:  - Ratio: 96%  - Result: Pass | 20/07/2015 | With 2 failed case, the color that sensor detect is purple. |
| TC004 | TCS230 Sensor with purple marble. | After integrating this sensor to mechanical part, input one purple marble directly by hand. | - Serial Monitor of Arduino IDE display: “Purple”. | N/A | Test 100 times:  - Successful: 100  - Failed: 0  Conclude:  - Ratio: 100%  - Result: Pass | 20/07/2015 |  |
| TC005 | TCS230 Sensor with pink marble. | After integrating this sensor to mechanical part, input one pink marble directly by hand. | - Serial Monitor of Arduino IDE display: “Pink”. | N/A | Test 100 times:  - Successful: 97  - Failed: 3  Conclude:  - Ratio: 97%  - Result: Pass | 20/07/2015 | With 3 failed case, the color that sensor detect is purple. |

Table 72: TCS3200 Color Sensor Testing

* + 1. Servo RC

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Test Case Description | Test Case Procedure | **Expected output** | **Inter-test Case Dependence** | **Result** | **Test Date** | **Note** |
| TC006 | Servo name:  “LServo\_Tcs230” | 1. After integrating this servo to mechanical part, connect this servo to Arduino Board.  2. Input one marble to machine. | - Servo opens and closes in right position.  Marble is not jammed. | N/A | Pass | 20/07/2015 | After testing 50 times, the best right position for this servo :  - open: 30 degrees  - close : 110 degree |
| TC007 | Servo name: “LServo\_Red” | 1. After integrating this servo to mechanical part, connect this servo to Arduino Board.  2. Input one red marble to machine. | - Servo opens and closes in right position.  - Red marble run into red repository. | N/A | Pass | 20/07/2015 | After testing 30 times, the best right position for this servo :  - open: 38 degrees  - close : 118 degree |
| TC008 | Servo name:  “LServo\_Green” | 1. After integrating this servo to mechanical part, connect this servo to Arduino Board.  2. Input one green marble to machine. | - Servo opens and closes in right position.  - Green marble run into green repository. | N/A | Pass | 20/07/2015 | After testing 35 times, the best right position for this servo :  - open: 23 degrees  - close : 103 degree |
| TC009 | Servo name:  “Servo\_Blue” | 1. After integrating this servo to mechanical part, connect this servo to Arduino Board.  2. Input one blue marble to machine. | - Servo opens and closes in right position.  - Blue marble run into Blue repository. | N/A | Pass | 20/07/2015 | After testing 20 times, the best right position for this servo :  - open: 37 degrees  - close : 117 degree |
| TC010 | Servo name:  “LServo\_Purple” | 1. After integrating this servo to mechanical part, connect this servo to Arduino Board.  2. Input one purple marble to machine. | - Servo opens and closes in right position.  - Purple marble run into purple repository. | N/A | Pass | 20/07/2015 | After testing 22 times, the best right position for this servo :  - open: 39 degrees  - close : 129 degree |
| TC011 | Servo name:  “LServo\_Pink” | 1. After integrating this servo to mechanical part, connect this servo to Arduino Board.  2. Input one pink marble to machine. | - Servo opens and closes in right position.  - Pink marble run into pink repository. | N/A | Pass | 20/07/2015 | After testing 27 times, the best right position for this servo :  - open: 3 degrees  - close : 83 degree |
| TC012 | Servo name:  “MServo\_Gate” | After integrating this servo to mechanical part, connect this servo to Arduino Board. | - Servo opens and closes in right position. Servo is not jammed. | N/A | Pass | 20/07/2015 | After testing 28 times, the best right position for this servo :  - open: 138 degrees  - close : 60 degree |
| TC013 | Servo name:  “MServo\_Gate2” | After integrating this servo to mechanical part  , connect this servo to Arduino Board. | - Servo opens and closes in right position. Servo is not jammed. | N/A | Pass | 20/07/2015 | After testing 32 times, the best right position for this servo :  - open: 18 degrees  - close : 85 degree |
| TC014 | Servo name:  “MServo\_Red\_Gate1” | After integrating this servo to mechanical part, connect this servo to Arduino Board. | - Servo opens and closes in right position. Servo is not jammed. | N/A | Pass | 20/07/2015 | After testing 25 times, the best right position for this servo :  - open: 148 degrees  - close : 58 degree |
| TC015 | Servo name:  “MServo\_Red\_Gate2” | After integrating this servo to mechanical part, connect this servo to Arduino Board. | - Servo opens and closes in right position. Servo is not jammed. | N/A | Pass | 20/07/2015 | After testing 24 times, the best right position for this servo :  - open: 35 degrees  - close : 115 degree |
| TC016 | Servo name:  “MServo\_Green\_Gate3” | After integrating this servo to mechanical part, connect this servo to Arduino Board. | - Servo opens and closes in right position. Servo is not jammed. | N/A | Pass | 20/07/2015 | After testing 24 times, the best right position for this servo :  - open: 155 degrees  - close : 75 degree |
| TC017 | Servo name:  “MServo\_Green\_Gate4” | After integrating this servo to mechanical part, connect this servo to Arduino Board. | - Servo opens and closes in right position. Servo is not jammed. | N/A | Pass | 20/07/2015 | After testing 22 times, the best right position for this servo :  - open: 25 degrees  - close : 115 degree |
| TC018 | Servo name:  “MServo\_Blue\_Gate5” | After integrating this servo to mechanical part, connect this servo to Arduino Board. | - Servo opens and closes in right position. Servo is not jammed. | N/A | Pass | 20/07/2015 | After testing 24 times, the best right position for this servo :  - open: 145 degrees  - close : 70 degree |
| TC019 | Servo name:  “MServo\_Blue\_Gate6” | After integrating this servo to mechanical part, connect this servo to Arduino Board. | - Servo opens and closes in right position. Servo is not jammed. | N/A | Pass | 20/07/2015 | After testing 20 times, the best right position for this servo :  - open: 35 degrees  - close : 120 degree |
| TC020 | Servo name:  “MServo\_Purple\_Gate7” | After integrating this servo to mechanical part, connect this servo to Arduino Board. | - Servo opens and closes in right position, marble is not jammed | N/A | Pass | 20/07/2015 | After testing 19 times, the best right position for this servo :  - open: 155 degrees  - close : 75 degree |
| TC021 | Servo name:  “MServo\_Purple\_Gate8” | After integrating this servo to mechanical part, connect this servo to Arduino Board. | - Servo opens and closes in right position. Servo is not jammed. | N/A | Pass | 20/07/2015 | After testing 20 times, the best right position for this servo :  - open: 20 degrees  - close : 105 degree |
| TC022 | Servo name:  “MServo\_PinkGate9” | After integrating this servo to mechanical part, connect this servo to Arduino Board. | - Servo opens and closes in right position. Servo is not jammed. | N/A | Pass | 20/07/2015 | After testing 28 times, the best right position for this servo :  - open: 140 degrees  - close : 65 degree |
| TC023 | Servo name:  “MServo\_PinkGate10” | After integrating this servo to mechanical part, connect this servo to Arduino Board. | - Servo opens and closes in right position. Servo is not jammed. | N/A | Pass | 20/07/2015 | After testing 18 times, the best right position for this servo :  - open: 20 degrees  - close : 100 degree |

Table 73: Servos Testing

The table below is the summary of right position of servos:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No.** | **Mechanic Part** | **Name** | **Size** | **Close (degree)** | **Open (degree)** | **Difference** |
| 1 | Input | LServo\_Tcs230 | Large | 30 | 110 | 80 |
| 2 | Input | LServo\_Red | Large | 38 | 118 | 80 |
| 3 | Input | LServo\_Green | Large | 23 | 103 | 80 |
| 4 | Input | Servo\_Blue | Large | 37 | 117 | 80 |
| 5 | Input | LServo\_Purple | Large | 39 | 129 | 90 |
| 6 | Input | LServo\_Pink | Large | 3 | 83 | 80 |
| 7 | Input | MServo\_Gate1 | Mini | 138 | 60 | 78 |
| 8 | Input | MServo\_Gate2 | Mini | 18 | 85 | 67 |
| 9 | Output | MServo\_Red\_Gate1 | Mini | 148 | 58 | 90 |
| 10 | Output | MServo\_Red\_Gate2 | Mini | 35 | 115 | 80 |
| 11 | Output | MServo\_Green\_Gate3 | Mini | 155 | 75 | 80 |
| 12 | Output | MServo\_Green\_Gate4 | Mini | 25 | 115 | 90 |
| 13 | Output | MServo\_Blue\_Gate5 | Mini | 145 | 70 | 75 |
| 14 | Output | MServo\_Blue\_Gate6 | Mini | 35 | 120 | 85 |
| 15 | Output | MServo\_Purple\_Gate7 | Mini | 155 | 75 | 80 |
| 16 | Output | MServo\_Purple\_Gate8 | Mini | 20 | 105 | 85 |
| 17 | Output | MServo\_PinkGate9 | Mini | 140 | 65 | 75 |
| 18 | Output | MServo\_PinkGate10 | Mini | 20 | 100 | 80 |

Table 74: Summary: Position of each servo

* + 1. Infrared Sensor

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Test Case Description | Test Case Procedure | **Expected output** | **Inter-test Case Dependence** | **Result** | **Test Date** | **Note** |
| TC024 | Infrared sensor  (8 sensors) | 1. After integrating this sensor to mechanical part  2. Put “red” marble in font of this sensor directly by hand. | - Before no marble, Serial Monitor of Arduino IDE display “1”  - After having “red” marble, Serial Monitor of Arduino IDE display: “0”. | N/A | Pass | 20/07/2015 |  |
| TC025 | Infrared sensor  (8 sensors) | 1. After integrating this sensor to mechanical part  2. Put “green” marble in font of this sensor directly by hand. | - Before no marble, Serial Monitor of Arduino IDE display “1”  - After having “green” marble, Serial Monitor of Arduino IDE display: “0”. | N/A | Pass | 20/07/2015 |  |
| TC026 | Infrared sensor  (8 sensors) | 1. After integrating this sensor to mechanical part  2. Put “blue” marble in font of this sensor directly by hand. | - Before no marble, Serial Monitor of Arduino IDE display “1”  - After having “blue” marble, Serial Monitor of Arduino IDE display: “0”. | N/A | Pass | 20/07/2015 |  |
| TC027 | Infrared sensor  (8 sensors) | 1. After integrating this sensor to mechanical part  2. Put “purple” marble in font of this sensor directly by hand. | - Before no marble, Serial Monitor of Arduino IDE display “1”  - After having “purple” marble, Serial Monitor of Arduino IDE display: “0”. | N/A | Pass | 20/07/2015 |  |
| TC028 | Infrared sensor  (8 sensors) | 1. After integrating this sensor to mechanical part  2. Put “pink” marble in font of this sensor directly by hand. | - Before no marble, Serial Monitor of Arduino IDE display “1”  - After having “pink” marble, Serial Monitor of Arduino IDE display: “0”. | N/A | Pass | 20/07/2015 |  |
| TC029 | Infrared sensor  (8 sensors) | 1. After integrating this sensor to mechanical part  2. Put black marble in font of this sensor directly by hand. | - Before no marble, Serial Monitor of Arduino IDE display “1”  - After having black marble, Serial Monitor of Arduino IDE display: “0”. | N/A | Fail | 20/07/2015 | Cannot detect object with black color. |

Table 75: Infrared Sensors Testing

Summary: Infrared sensor cannot detect black object, include black marble.

* 1. Integration Testing
     1. Input Marble

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Test Case Description** | **Test Case Procedure** | **Expected output** | **Inter-test Case Dependence** | **Result** | **Test Date** | **Note** |
| TC030 | User input 1 marble. | User input 1 “red” marble to machine. | - 1 red marble runs to red repository.  - Update information to database and display desktop application screen. | N/A | Test 100 times:  - Successful: 97  - Failed: 3  Conclude:  - Ratio: 97%  - Result: Pass | 21/07/2015 | With 3 failed case, the color that sensor detect is pink. |
| TC031 | User input 1 marble. | User input 1 “green” marble to machine. | - 1 green marble runs to green repository.  - Update information to database and display desktop application screen. | N/A | Test 100 times:  - Successful: 100  - Failed: 0  Conclude:  - Ratio: 100%  - Result: Pass | 21/07/2015 |  |
| TC032 | User input 1 marble. | User input 1 “blue” marble to machine. | - 1 blue Marble runs to blue repository.  - Update information to database and display desktop application screen. | N/A | Test 100 times:  - Successful: 97  - Failed: 3  Conclude:  - Ratio: 97%  - Result: Pass | 21/07/2015 | With 3 failed case, the color that sensor detect is purple. |
| TC033 | User input 1 marble. | User input 1 “purple” marble to machine. | - 1 purple Marble runs to purple repository.  - Update information to database and display desktop application screen. | N/A | Test 100 times:  - Successful: 96  - Failed: 4  Conclude:  - Ratio: 96%  - Result: Pass | 21/07/2015 | With 4 failed case, the color that sensor detect is pink. |
| TC034 | User input 1 marble. | User input 1 “pink” marble to machine. | - 1 pink marble runs to pink repository.  - Update information to database and display desktop application screen. | N/A | Test 100 times:  - Successful: 95  - Failed: 5  Conclude:  - Ratio: 95%  - Result: Pass | 21/07/2015 | With 6 failed case, the color that sensor detect is purple or the marble jump over the hole. |
| TC035 | User input 10 marbles. | User input 10 red marbles to machine. | - 9 red marbles run to red repository.  - 1 red marble runs to external repository.  - Update information to database and display desktop application screen. | N/A | Test 1 time:  - Result: Pass | 21/07/2015 |  |
| TC036 | User input 10 marbles. | User input 10 green marbles to machine. | - 9 green marbles run to green repository.  - 1 green marble runs to external repository.  - Update information to database and display desktop application screen. | N/A | Test 1 time:  - Result: Pass | 21/07/2015 |  |
| TC037 | User input 10 marbles. | User input 10 blue marbles to machine. | - 9 blue marbles run to blue repository.  - 1 blue marble runs to external repository.  - Update information to database and display desktop application screen. | N/A | Test 1 time:  - Result: Pass | 21/07/2015 |  |
| TC038 | User input 10 marbles. | User input 10 purple marbles to machine. | - 9 purple marbles run to purple repository.  - 1 purple marble runs to external repository.  - Update information to database and display desktop application screen. | N/A | Test 1 time:  - Result: Pass | 21/07/2015 |  |
| TC039 | User input 10 marbles. | User input 10 pink marbles to machine. | - 9 pink marble run to pink repository.  - 1 pink marble runs to external repository.  - Update information to database and display desktop application screen. | N/A | Test 1 time:  - Result: Pass | 21/07/2015 |  |
| TC040 | User input random marbles. | User input 5 red marbles, 7 blue marbles, 2 green machine. | - 5 red marble runs to Blue repository  - 7 blue marbles run to blue repository.  - 2 green marbles run to green repository.  - Update information to database and display desktop application screen. | N/A | Test 1 time:  - Result: Pass | 21/07/2015 |  |
| TC041 | User input random marbles. | User input 6 green marbles, 2 pink marbles to machine. | - 6 green marbles run to green repository.  - 2 pink marbles run to pink repository.  - Update information to database and display desktop application screen. | N/A | Test 1 time:  - Result: Pass | 21/07/2015 |  |
| TC042 | User input random marbles. | User input 2 purple marbles, 2 red marbles to machine. | - 2 purple marbles run to purple repository.  - 1 red marble runs to red repository  - Update information to database and display desktop application screen. | N/A | Test 1 time:  - Result: Pass | 21/07/2015 |  |
| TC043 | User input random marbles. | User Input 5 red marbles, 7 green marbles, 4 blue marbles, and 2 purple marbles to machine. | - 5 red marble run to Blue repository  - 7 green marbles run to green repository.  - 4 blue marbles run to blue repository.  - 2 purple marbles run to purple repository.  - Update information to database and display desktop application screen. | N/A | Test 1 time:  - Result: Pass | 21/07/2015 |  |
| TC044 | User input random marbles. | User input 5 red marbles, 5 blue marbles, 2 green marbles, 2 pink marbles, 1 purple marbles to machine. | - 5 red marble run to Blue repository  - 5 blue marbles run to blue repository.  - 2 green marbles run to green repository.  - 2 pink marbles run to pink repository.  - 1 purple marbles run to purple repository.  - Update information to database and display desktop application screen. | N/A | Test 1 time:  - Result: Pass | 21/07/2015 |  |
| TC045 | User input 9 red marbles, 9 blue marbles, 9 green marbles, 9 pink marbles, and 9 purple marbles. | User input 9 red marbles, 9 blue marbles, 9 green marbles, 9 pink marbles, and 9 purple marbles to machine. | - 9 red marbles run to red repository.  - 9 blue marbles run to Blue repository.  - 9 green marbles run to green repository.  - 9 pink marble runs to pink repository.  - 9 purple runs to purple repository.  - Update information to database and display desktop application screen. | N/A | Test 1 time:  - Result: Pass | 21/07/2015 |  |
| TC046 | User input marble with cover black duct tape. | User input 1 red marble with 50% cover black tape to machine. | -1 red marble runs to pink repository.  - Update information to database and display desktop application screen. | N/A | Test 10 times:  - Successful: 8  - Failed: 2  Conclude:  - Ratio: 80%  - Result: Pass | 21/07/2015 | With 2 failed case, the color that sensor detect is pink. |
| TC047 | User input marble with cover black duct tape. | User input 1 red marble with 80% cover black tape to machine. | -1 red marble runs to pink repository.  - Update information to database and display desktop application screen. | N/A | Test 10 times:  - Successful: 5  - Failed: 5  Conclude:  - Ratio: 50%  - Result: Failed | 21/07/2015 | With 5 failed case, the color that sensor detect is pink. |
| TC048 | User input marble with cover black duct tape. | User input 1 green marble with 50% cover black tape to machine. | -1 green marble runs to green repository.  - Update information to database and display desktop application screen. | N/A | Test 10 times:  - Successful: 10  - Failed: 0  Conclude:  - Ratio: 100%  - Result: Pass | 21/07/2015 |  |
| TC049 | User input marble with cover black duct tape. | User input 1 green marble with 80% cover black tape to machine. | -1 green marble runs to green repository.  - Update information to database and display desktop application screen. | N/A | Test 10 times:  - Successful: 9  - Failed: 1  Conclude:  - Ratio: 90%  - Result: Pass | 21/07/2015 | With 1 failed case, the color that sensor detect is purple. |
|  | User input marble with cover black duct tape. | User input 1 blue marble with 50% cover black tape to machine. | -1 blue marble runs to blue repository.  - Update information to database and display desktop application screen. | N/A | Test 10 times:  - Successful: 8  - Failed: 2  Conclude:  - Ratio: 80%  - Result: Pass | 21/07/2015 | With 2 failed case, the color that sensor detect is purple. |
| TC050 | User input marble with cover black duct tape. | User input 1 blue marble with 80% cover black tape to machine. | -1 blue marble runs to blue repository.  - Update information to database and display desktop application screen. | N/A | Test 10 times:  - Successful: 9  - Failed: 1  Conclude:  - Ratio: 90%  - Result: Pass | 21/07/2015 | With 1 failed case, the color that sensor detect is purple. |
| TC051 | User input marble with cover black duct tape. | User input 1 purple marble with 50% cover black tape to machine. | -1 purple marble runs to purple repository.  - Update information to database and display desktop application screen. | N/A | Test 10 times:  - Successful: 10  - Failed: 0  Conclude:  - Ratio: 100%  - Result: Pass | 21/07/2015 |  |
| TC052 | User input marble with cover black duct tape. | User input 1 purple marble with 80% cover black tape to machine. | -1 purple marble runs to purple repository.  - Update information to database and display desktop application screen. | N/A | Test 10 times:  - Successful: 10  - Failed: 0  Conclude:  - Ratio: 100%  - Result: Pass | 21/07/2015 |  |
| TC053 | User input marble with cover black duct tape. | User input 1 pink marble with 50% cover black tape to machine. | -1 pink marble runs to pink repository.  - Update information to database and display desktop application screen. | N/A | Test 10 times:  - Successful: 8  - Failed: 2  Conclude:  - Ratio: 100%  - Result: Pass | 21/07/2015 | With 2 failed case, the color that sensor detect is purple. |
| TC054 | User input marble with cover black duct tape. | User input 1 pink marble with 80% cover black tape to machine. | -1 pink marble runs to pink repository.  - Update information to database and display desktop application screen. | N/A | Test 10 times:  - Successful: 4  - Failed: 6  Conclude:  - Ratio: 40%  - Result: Failed | 21/07/2015 | With 6 failed case, the color that sensor detect is purple. |
| TC055 | Special marble | User input 1 green marble with 50% cover red decal. | - 1 green marble runs to green repository. | N/A | Test 10 times:  - Successful: 3  - Failed: 7  Conclude:  - Ratio: 30%  - Result: Failed | 21/07/2015 | With 7 failed case, the color that sensor detect is red. |
| TC056 | Special marble | User input 1 black marble to machine. | - 1 black marble runs to external repository. | N/A | Test 10 times:  - Successful: 0  - Failed: 10  Conclude:  - Ratio: 0%  - Result: Failed | 21/07/2015 | With 4 failed case, the color that sensor detect is purple.  With 6-failed case, the infrared sensor cannot check position this marble. |

Table 76: Input Marble Testing

* + 1. Output Marble

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Test Case Description** | **Test Case Procedure** | **Expected output** | **Inter-test Case Dependence** | **Result** | **Test Date** | **Note** |
| TC057 | User output 1 marble and 1 package. | 1. User put box to conveyor belt.  2. User selects 1 pink marble and 1 packed then click "Get marble".  3. User fill all customer ID and click "Save" | - 1 pink marble output for user.  - Update information to database and display desktop application screen. | TC045 | Pass | 22/07/2015 |  |
| TC058 | User output 1 marble and 1 package. | 1. User put box to conveyor belt.  2. User selects 1 red marble and 1 packed then click "Get marble".  3. User fill all customer ID and click "Save" | - 1 red marble output for user.  - Update information to database and display desktop application screen. | TC045 | Pass | 22/07/2015 |  |
| TC059 | User output 1 marble and 1 package. | 1. User put box to conveyor belt.  2. User selects 1 purple marble and 1 packed then click "Get marble".  3. User fill all customer ID and click "Save" | - 1 purple marble output for user.  - Update information to database and display desktop application screen. | TC045 | Pass | 22/07/2015 |  |
| TC060 | User output 1 marble and 1 package. | 1. User put box to conveyor belt.  2. User selects 1 green marble and 1 packed then click "Get marble".  3. User fill all customer ID and click "Save" | - 1 green marble output for user.  - Update information to database and display desktop application screen. | TC045 | Pass | 22/07/2015 |  |
| TC061 | User output 1 marble and 1 package. | 1. User put box to conveyor belt.  2. User selects 1 blue marble and 1 packed then click "Get marble".  3. User fill all customer ID and click "Save" | - 1 blue marble output for user.  - Update information to database and display desktop application screen. | TC045 | Pass | 22/07/2015 |  |
| TC062 | User output 6 marble and 2 package. | 1. User put box to conveyor belt.  2. User selects 6 blue marble and 2 packages then click "Get marble". | Show warning message "Not enough blue marbles in repository for 2 packages" | TC045 | Pass | 22/07/2015 |  |
| TC063 | User output 6 marble and 2 package. | 1. User put box to conveyor belt.  2. User selects 3 blue marble, 2 red marbles and 2 packages then click "Get marble".  3. User fill all customer ID and click "Save" | - 2 packages output for user. Each package has 3 blue marble, 2 red marbles.  - Update information to database and display desktop application screen. | TC045 | Pass | 22/07/2015 |  |
| TC064 | User output 6 marble and 2 package. | 1. User put box to conveyor belt.  2. User selects 2 blue marble, 2 red marbles, 1 pink marble and 2 packages then click "Get marble".  3. User fill all customer ID and click "Save" | - 2 packages output for user. Each package has 3 blue marble, 3 red marbles and 1 pink marble.  - Update information to database and display desktop application screen. | TC045 | Pass | 22/07/2015 |  |
| TC065 | User output random marbles. | 1. User put box to conveyor belt.  2. User selects 3 red marbles, 3 green marble and 1 packages then click "Get marble".  3. User fill all customer ID and click "Save" | Show warning message “Output marbles must less than or equal 5". | TC045 | Pass | 22/07/2015 |  |
| TC066 | User output random marbles. | 1. User put box to conveyor belt.  2. User selects 2 red marbles, 1 pink marble, 1 green marble and 1 packages then click "Get marble".  3. User fill all customer ID and click "Save" | - 2 red marbles, 1 pink marble and 1 green marble output for user.  - Update information to database and display desktop application screen. | TC045 | Pass | 22/07/2015 |  |

Table 77: Output Marble Testing

* + 1. Export order details

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Test Case Description** | **Test Case Procedure** | **Expected output** | **Inter-test Case Dependence** | **Result** | **Test Date** | **Note** |
| TC067 | User exports order details. | 1. In Customer Order Details, Click "Export” menu.  2. Choose "PDF" and location. | Output PDF file and save at right position that user choose. | N/A | Pass | 21/07/2015 |  |
| TC068 | User exports order details. | 1. In Customer Order Details, Click "Export” menu.  2. Choose "Word" and location. | Output Word file and save at right position that user chooses. | N/A | Pass | 21/07/2015 |  |
| TC069 | User exports order details. | 1. In Customer Order Details, Click "Export” menu.  2. Choose "Excel" and location. | Output Excel file and save at right position that user chooses. | N/A | Pass | 21/07/2015 |  |

Table 78: Export Order Details Testing

1. Software User’s Manual
2. Installation Guide
   1. Setting up environment
      1. Hardware requirements

Laptop or personal computer with minimum requirement:

* OS: Windows 7/8
* CPU: Intel Core 2 Duo 6600
* Ram: 2 GB
* USB Support: 2 or more
  + 1. Software requirements
* Microsoft .NET Framework 4.5
* Microsoft SQL Server 2012 Express
  1. Deployment
     1. Install .NET Framework 4.5
* Download .NET Framework 4.5 at link and install:

<http://www.microsoft.com/en-us/download/details.aspx?id=30653>

* + 1. Install Microsoft SQL Server 2012 Express
* After installing successfully, open SQL Management Studio and Connect to Server with "Windows Authentication” mode.

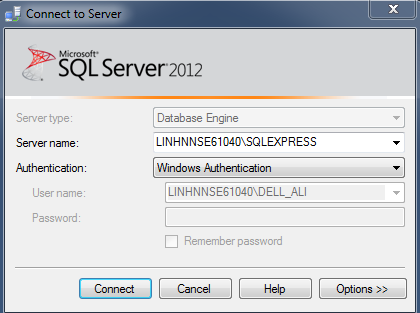


Figure 55: Connect to SQL Server with Windows Authentication mode

* When PSSC Desktop Application runs the first time, it will check PC or Laptop to have SQL Server or not. If system has SQL Server, application creates database automatically, otherwise it shows error and stops working.
  + 1. Install PSSC Desktop Application
* Extract the installation package.
* Run setup file and follow the steps of setup wizard.

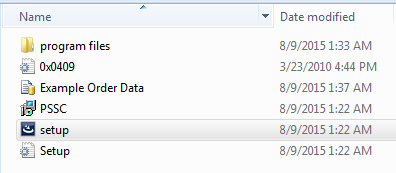


Figure 56: Install PSSC Desktop Application

Note: MS SQL Server must be installed before running this application.

* + 1. Connect PSSC Machine
* Use two USB cables to connect between PSSC Machine and computer. Then wait windows installs driver automatically.
* The image of USB cable is showed below:



Figure 57: USB Cable

1. User Guide
   1. Desktop Application Guide
      1. Configure Ports

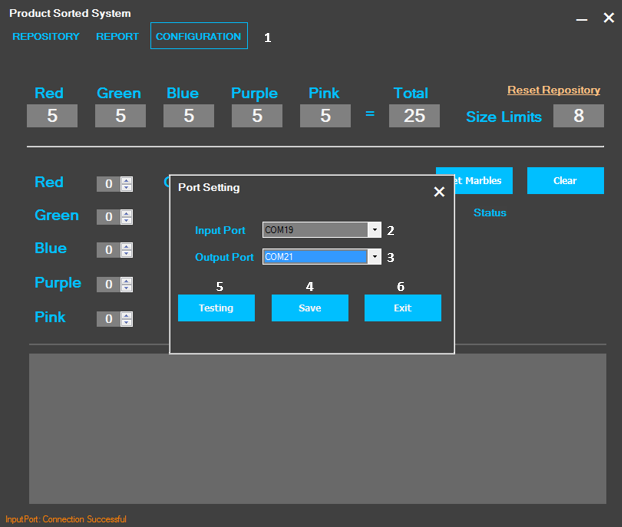


Figure 58: <Desktop App Guide> User configures ports connection

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | Click on CONFIGURATION" menu at “1”. |
| 2 | Select Input Port at “2”. |
| 3 | Select Output Port at “3”. |
| 4 | User click "Save"Button at “4”. |
| 5 | User click "Testing"Button at”5”. |
| 6 | User click "Exit"Button at “6”. |

Table 79: <Desktop App Guide> Configures Ports Steps

* + 1. Input Marble

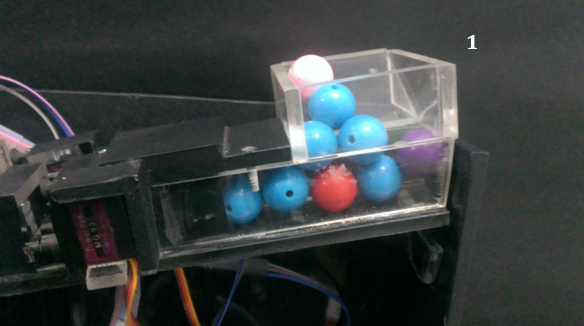


Figure 59: <Desktop App Guide> User input marbles into machine

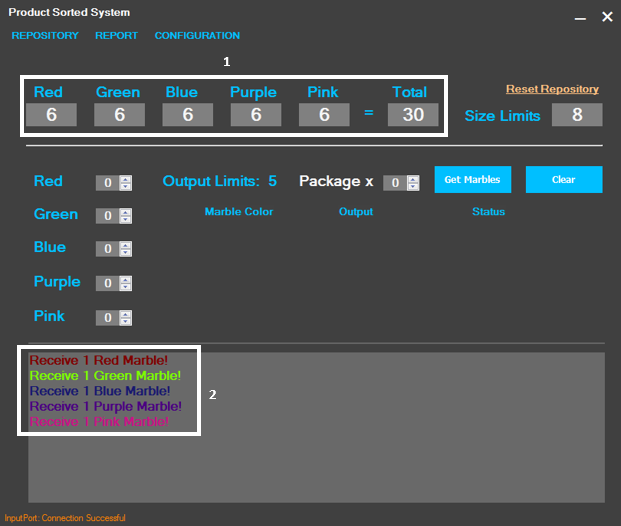


Figure 60: <Desktop App Guide> User waits application update input marble

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | User input marbles to product sorter machine. |
| 2 | User waits application update information about quantity at “2” |
| 3 | User waits application show success message to confirm at “3” |

Table 80: <Desktop App Guide> Input Marble Steps

* + 1. Output Marble



Figure 61: <Desktop App Guide> User inputs package on conveyor belt

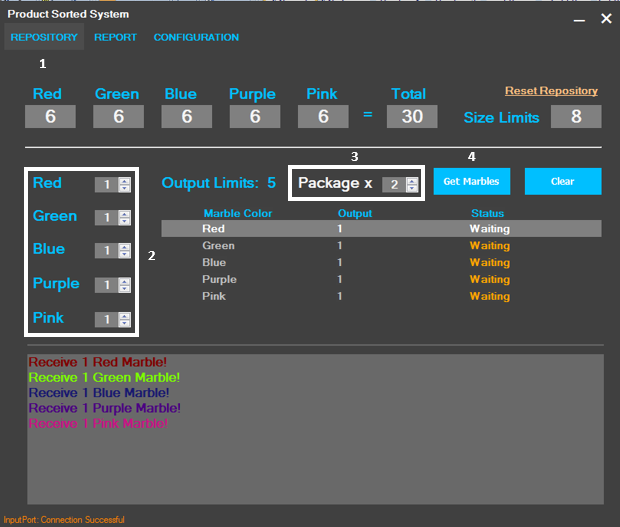


Figure 62: <Desktop App Guide> User selects marble to output

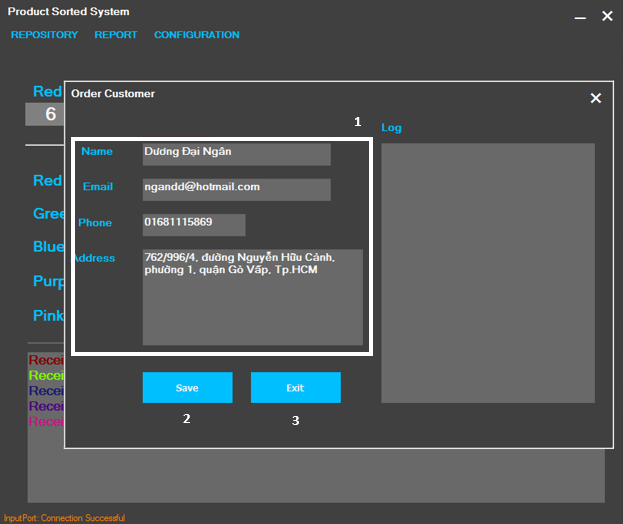


Figure 63: <Desktop App Guide> User inputs order customer information

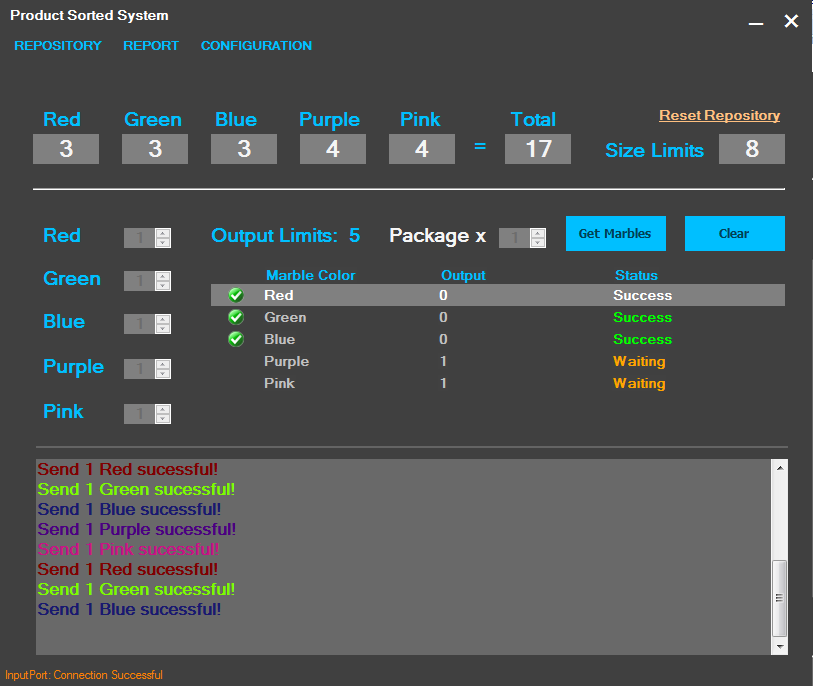


Figure 64: <Desktop App Guide> User waits machine to get out marbles

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | User puts package on conveyor belt. |
| 2 | Click on “REPOSITORY" menu at “1”. |
| 3 | Select number of marbles to get out by color at “2” in Main Screen. |
| 4 | Select number of packages at “3” in Main Screen. |
| 5 | Click "Get Marbles" button at “4” in Main Screen, Customer Screen will be showed. |
| 6 | Input customer information at “1” in Customer Screen. |
| 7 | Click “Save” button at “2” to start get marbles out or click “Exit” at “3” to cancel. |
| 8 | User waits machine to get out marbles and update success message on desktop application. |

Table 81: <Desktop App Guide> Output Marble Steps

* + 1. View customer order.
       1. View order details

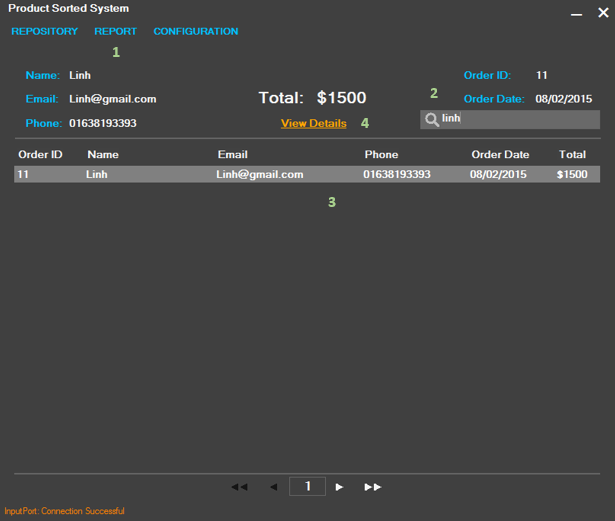


Figure 65: <Desktop App Guide> User search customer order

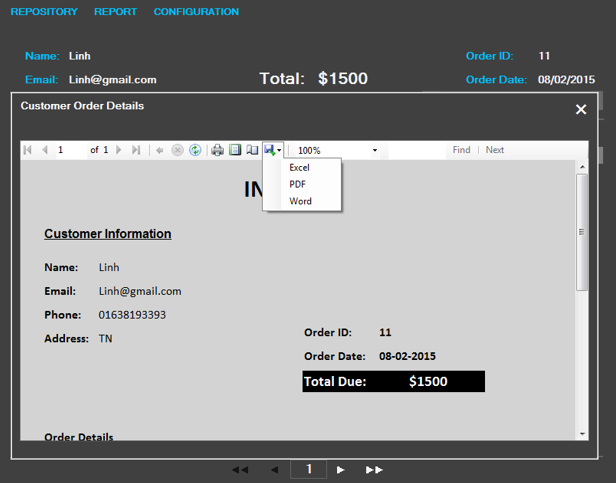


Figure 66: <Desktop App Guide> Show order details

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | Click “REPORT" Menu. |
| 2 | User input customer information at “2” to search. |
| 3 | Select customer from searching result at “3”. |
| 4 | Click “View Details” at “4” |

Table 82: <Desktop App Guide> View Order Details steps

* + - 1. Export order details

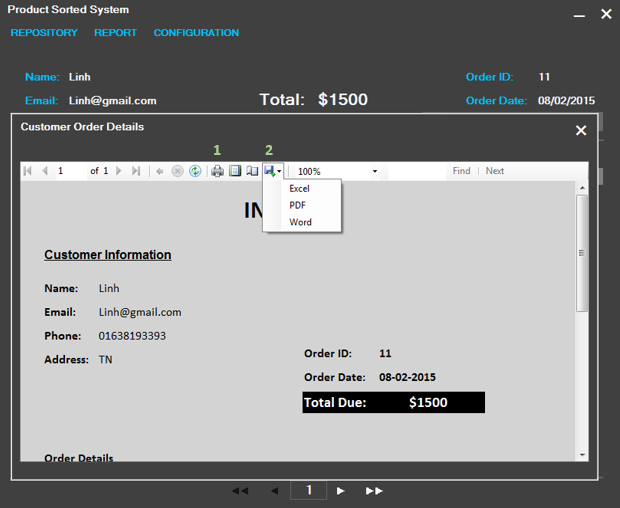


Figure 67: <Desktop App Guide> User Export Order Details

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | If user wants to print, do step 2.  If user wants to export, do step 3. |
| 2 | Click “Printer” icon at “1” to print. |
| 3 | Click “Floppy Drive” icon at “2”. Then select output file format. |

Table 83: <Desktop App Guide> Export Order Details Step.

* 1. Machine Simulator Guide
     1. Introduction

Machine simulator is desktop application. Developer could develop and test functions of main program without real machine.

* + 1. Configures Ports

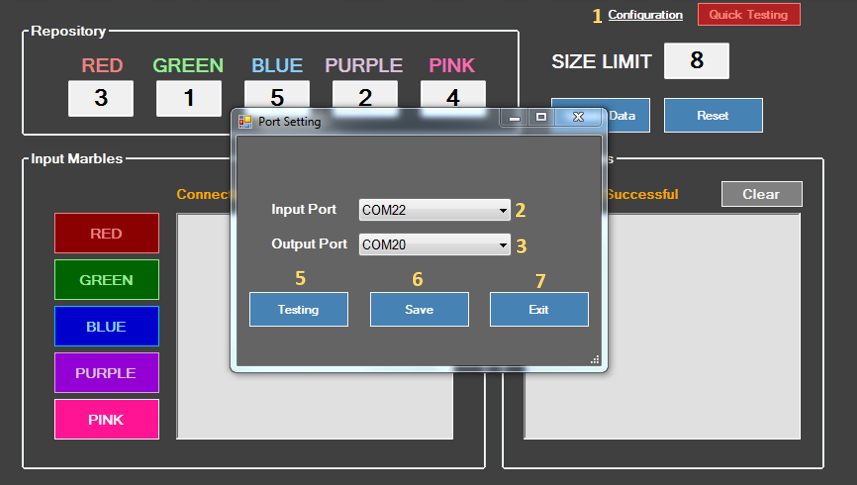


Figure 68: <Machine Simulator> User configure ports connection

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | Click “Configuration" at “1”. |
| 2 | Select Input Port at “2”. |
| 3 | Select Output Port at “3”. |
| 4 | User click "Save"Button at “4”. |
| 5 | User click "Testing"Button at”5”. |
| 6 | User click "Exit"Button at “6”. |

Table 84: <Machine Simulator> Configure Ports Steps

* + 1. Input Marble

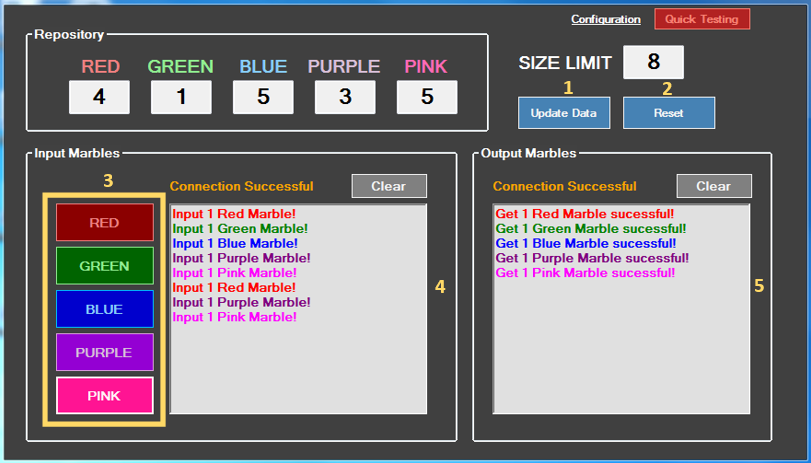


Figure 69: <Machine Simulator> User input marble

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | Click “Update Data" button at “1” to load current marbles. Or click “Reset” button at “2” to renew. |
| 2 | Press one of five button colours at “3” to input marbles to desktop app.  Input or output marble information is updates at “3” and “4” |

Table 85: <Machine Simulator> Input Marble Steps

1. Appendix

1. Aruino Mega

* http://arduino-info.wikispaces.com/MegaQuickRef
* https://www.arduino.cc/en/Main/ArduinoBoardMega

2. TCS3200 Sensor Color

* http://www.dfrobot.com/image/data/SEN0101/TCS3200%20TCS3210.pdf

3. Pulse Width Modulation

* http://arduino.vn/reference/xung-pwm4. Pulse Position Modulation
* http://tbe.vn/chia-se-kien-thuc/7056-phuong-phap-dieu-xung-pwm-la-gi.html

4. Pulse Position Modulation

* http://www.pcbheaven.com/wikipages/Pulse\_Position\_Modulation/

5. L298 Dual H – Bridge

* http://www.hocavr.com/index.php/hardware/hbridge

6. Servo RC

* http://www.digitalnemesis.com/info/docs/rcservo/
* http://www.hooked-on-rc-airplanes.com/servo-tutorial.html

7. Infrared Sensor

* http://www.robosoftsystems.co.in/roboshop/media/catalog/product/pdf/IR\_single.pdf

8. Serial Port (UART)

* http://www.hocavr.com/index.php/lectures/uart

9. K – Nearest Neighbors Algothrim

* http://www.saedsayad.com/k\_nearest\_neighbors.htm
* https://en.wikipedia.org/wiki/K-nearest\_neighbors\_algorithm