**See3PO Software Requirements Specifications (SRS)**

**Version 0.8**

1. [INTRODUCTION](http://code.google.com/p/see3po/wiki/RequirementsSpec#1._Introduction)
   1. [Product Overview](http://code.google.com/p/see3po/wiki/RequirementsSpec#1.1_Product_Overview)
   2. [Purpose](http://code.google.com/p/see3po/wiki/RequirementsSpec#1.2_Purpose)
   3. [Scope](http://code.google.com/p/see3po/wiki/RequirementsSpec#1.3_Scope)
   4. [Reference](http://code.google.com/p/see3po/wiki/RequirementsSpec#1.4_Reference)
   5. [Definition and Abbreviation](http://code.google.com/p/see3po/wiki/RequirementsSpec#1.5_Definition_and_Abbreviation)
2. [OVERALL DESCRIPTION](http://code.google.com/p/see3po/wiki/RequirementsSpec#2._Overall_Description)
   1. [Product Perspective](http://code.google.com/p/see3po/wiki/RequirementsSpec#2.1_Product_Perspective)
   2. [Product Functions](http://code.google.com/p/see3po/wiki/RequirementsSpec#2.2_Product_Functions)
   3. [User Characteristics](http://code.google.com/p/see3po/wiki/RequirementsSpec#2.3_User_Characteristics)
   4. [General Constraints](http://code.google.com/p/see3po/wiki/RequirementsSpec#2.4_General_Constraints)
   5. [Assumptions and Dependencies](http://code.google.com/p/see3po/wiki/RequirementsSpec#2.5_Assumptions_and_Dependencies)
3. [SPECIFIC REQUIREMENTS](http://code.google.com/p/see3po/wiki/RequirementsSpec#3._Specific_Requirements)
   1. [External Interface Requirements](http://code.google.com/p/see3po/wiki/RequirementsSpec#3.1_External_Interface_Requirements)
      1. [User Interface](http://code.google.com/p/see3po/wiki/RequirementsSpec#3.1.1_User_Interfaces)
      2. [Hardware Interface](http://code.google.com/p/see3po/wiki/RequirementsSpec#3.1.2_Hardware_Interfaces)
      3. [Software Interface](http://code.google.com/p/see3po/wiki/RequirementsSpec#3.1.3_Software_Interfaces)
      4. [Communications Protocols](http://code.google.com/p/see3po/wiki/RequirementsSpec#3.1.4_Communications_Protocols)
   2. [Sofware Product Features](http://code.google.com/p/see3po/wiki/RequirementsSpec#3.2_Software_Product_Features)
      1. [GUI](http://code.google.com/p/see3po/wiki/RequirementsSpec#3.2.1_Graphical_User_Interface)
      2. [Floor Plan Processing](http://code.google.com/p/see3po/wiki/RequirementsSpec#3.2.2_Floor_Plan_Processing)
      3. [Path Finding](http://code.google.com/p/see3po/wiki/RequirementsSpec#3.2.3_Path_Finding)
   3. [Sofware Product Features](http://code.google.com/p/see3po/wiki/RequirementsSpec#3.3_Software_System_Attributes)
      1. [Reliability](http://code.google.com/p/see3po/wiki/RequirementsSpec#3.3.1_Reliability)
      2. [Availability](http://code.google.com/p/see3po/wiki/RequirementsSpec#3.3.2_Availability)
      3. [Security](http://code.google.com/p/see3po/wiki/RequirementsSpec#3.3.3_Security)
      4. [Maintainability](http://code.google.com/p/see3po/wiki/RequirementsSpec#3.3.4_Maintainability)
      5. [Performance](http://code.google.com/p/see3po/wiki/RequirementsSpec#3.3.5_Performance)
4. [Revision History](http://code.google.com/p/see3po/wiki/RequirementsSpec#4_Revision_History)

**1. Introduction**

**1.1 Product Overview**

See3PO is a path finding robot, which can find the shortest path to reach its destination from its current location and reach there with minimum human intervention.

**1.2 Purpose**

This is a research project. The main purpose of this project is to make a robot find the shortest path, between its current location and destination, in a given floor plan. The robot should also be capable of updating its current route at a regular interval, in order to take care of any possible hindrance in it's path, that could have come to existence since the last time it calculated its route. And finally it should then reach the destination.

**1.3 Scope**

* Assumptions: Algorithm to be provided by MERL is delivered on time.
* Deliverables: GUI, Floor plan processing program, Path finding program..........

When provided with a given floor plan in acceptable image format, the robot will be able to get its current location on the same, using the algorithm provided by MERL. It will then pass that location information to appropriate programs to compute the shortest path to the destination. It then will move along the computed path towards it's destination. However, the computation of the route is not an one time event, this will continue to be executed at a regular interval, in order to rule out any possible hindrance on the route that could cause the robot to not to reach its destination and if such problems come into existence since the last time the path was calculated, the robot will then re-evaluate the route and follow the newly calculated route.

**1.4 Reference**

1. [System Architecture](http://code.google.com/p/see3po/wiki/SystemArchitecture)
2. [Use Cases](http://code.google.com/p/see3po/wiki/UseCases)

**1.5 Definition and Abbreviation**

*This section needs to be populated each time we use new abbreviations while working on SRS.*

|  |  |
| --- | --- |
| **Terms** | **Meaning** |
| SRS | System Requirement Specification |
| MERL | Mitsubishi Electric Research Laboratories |
| WhereAmI | A Vision-Based Locating Program supplied by MERL |
| Host PC | A Desktop PC used to interact wirelessly to the robot |
| See3PO | The Pathfinding Robot |

**2. Overall Description**

**2.1 Product Perspective**

See3PO will autonomously navigate its environment. Using only visual tools and a floorplan,   
the robot will walk to any reachable locations as directed by the user.

**2.2 Product Functions**

The User will be able to direct the robot to a desired location within its environment.  
The robot will use the WhereAmI software to keep up-to-date information about its current position.  
Pathfinder will find the optimal route to the target location from the current position.  
The Host PC will direct the robot, step by step, to the target, testing accuracy along the way

**2.3 User Characteristics**

The user should require no training.

**2.4 General Constraints**

* Wireless Communication may limit the range and accuracy of See3PO's movement.
* Current plans include known, indoor locatios with accessible floorplans.
* Some preparation may be necessary for WhereAmI to accurately work in a given environment.

**2.5 Assumptions and Dependencies**

It is assumed that the entire area to be covered by the robot will have the following properties:

* Consistent wireless reception
* Smooth, dry, level flooring
* A relatively consistent floor plan

**3. Specific Requirements**

**3.1 External Interface Requirements**

**3.1.1 User Interfaces**

* The user will be the operator of the See3PO robot.
* From the Host PC interface, the user will control connections to the robot and have the ability to delegate tasks to the robot.
* From the Robot interface, the user can change settings such as the IP address of the Host pc.

**3.1.2 Hardware Interfaces**

* Use wireless network (WiFi[?](http://code.google.com/p/see3po/w/edit/WiFi)) over TCP/IP to communicate between host and robot.
* Host acts as server and robot acts as client while establishing connection.
* Robot is able to connect or disconnect from host through interface.
* Robot can configure the IP address of host so the host can be deployed on different environment.
* Use serial port with cable over RS-232 to communicate between robot and control unit.
* Robot has interface for controlling its movement therefore it can be tested without receiving command from host.

**3.1.3 Software Interfaces**

* The Operating System must be Windows, and has been tested on Windows XP, Windows Vista and Windows 7.
* Host PC’s must have the most current version of .Net Framework.

**3.1.4 Communications Protocols**

Please refer to **Apendix A: Communication Protocols of See3PO**.

**3.2 Software Product Features**

**3.2.1 Graphical User Interface**

The primary GUI will reside on the Host PC and handle the following tasks:

* Connection to the Robot
* Floor Plan Loading:
  + From Image
  + From Saved Floor Plan files
* Automated and Manual placement of Robot within the floorplan
* Selection of Robot Destination

The GUI will have three main components

* A Live View panel will show a video feed from the robot.
* A text output window will display connection information and other messages
* A Floorplan window will handle the following input:
  + Setting floor plan scale by drawing a scale line and inputing the distance drawn
  + Manually setting the Robot's current position
  + Selecting the desired destination.

It will also display the following output:

* + A simplified image of the floorplan
    - One pixel equals one square step.
    - Color will indicate walkablility.
  + The Robot's current position will be indicated by a red arrow.
  + The current path will be indicated by a series of blue lines.
  + The destination will be indicated by a blue "X"

**3.2.2 Floor Plan Processing**

Floor plan processing consists of two components:

* Floor Plan
* Floor Tile

Floor Plan takes an image of an actual floor plan, (currently supporting both .bmp and .jpg format) and it converts the same to an array of floor tile objects. This creation of floor tile array works in the following way:

* The dimension of each tile is set as the scale selected in GUI.
* Each tile that has RGB component darker than that of the color gray will be treated as un-walkable, else it will be walkable.

Floor plan has another functionality that connects each tile in the it, with the four neighboring walkable tiles of the same(top, bottom, right and left). If any of the neighboring tile is un-walkable, it is not added to this list and the program continues with the next neighbor of the original tile, if any, or else with the next tile in the array of floor tiles. This walkable neighboring tiles information is kept in the floor tile object itself. Other than this floor plan also works with GUI in setting the start and target tile in it.

Floor tile, as the name suggests, corresponds to the tile object for a floor plan. It has a location in X-Y co-ordinate system for the floor plan, with (0,0) being the location for the first tile of a floor plan. A tile could be either walkable or un-walkable, as mentioned earlier. Also it keeps a list of neighbors of itself as mentioned before.

For more details please look into [Floor Plan Document](http://code.google.com/p/see3po/wiki/FloorPlanProcessing).

**3.2.3 Path Finding**

* Requirement: Develop a simple path planning algorithm to generate a near optimal path between two locations.
* Diliverable: Ability to automatically generate a near-optimal path for the robot to follow.
* Design:
  + **Strategy Disign Patterh**: we define an interface to adapt other path finding algorithms for further usages
  + **Adapter Design Pattern**: The implementation of the interface is in a concrete class QGPathFinder, which serves as an **adapter**. The class GQPathFinder transfers FloorPlan and FloorTile into a graph structure and returns a near optimal path between the starting and target point as a list of FloorTile.
  + We used a C# graph library **QuickGraph** since we have established works using C#.

For more details please look into [Path Finding Document](http://code.google.com/p/see3po/wiki/PathFinding).

**3.3 Software System Attributes**

**3.3.1 Reliability**

* It is extremely that the robot act correctly in the case of a lost connection. To accomplis this, the robot will recieve discrete commands. If the robot has not recieved a next command before completion of its current move, it will stop.
* To ensure that moves are accurate, between each move, the robot will acquire its current position and compare this to an expected value. If the values don't match, the path will be recalculated.
* Because the robot will deal with discrete directions (for, as of this writing), a readjustment algorithm will be used to bring the robot to true, in the case of a misalignment.

**3.3.2 Availability**

This software will be made free for download from [the wiki download page](http://code.google.com/p/see3po/downloads/list)

**3.3.3 Security**

Because the robot is searching for a specific host by IP, it would be difficult to hijack the robot with another PC.

**3.3.4 Maintainability**

All source code except for WhereAmI will be publicly available at [the See3PO Google Code site](http://code.google.com/p/see3po/)

**3.3.5 Performance**

Performance will be weighed against reliability. By reducing the calls to WhereAmI and the Pathfinder, performance is expected to increase, while reliability decreases. We intend to find a balance of the two.

**4 Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision Date** | **Description Date** | **Document Version** | **Author** |
| 12/02/2009 | Created outline for draft verion of SRS | 0.1 | [Frank](http://code.google.com/p/see3po/wiki/Frank) |
| 12/04/2009 | Added Introduction | 0.2 | [Debarati](http://code.google.com/p/see3po/wiki/Debarati) |
| 12/05/2009 | Added Added Hardware/Communications Requirements | 0.3 | [Jacky](http://code.google.com/p/see3po/wiki/Jacky) |
| 12/06/2009 | Added User Interface Requirements | 0.4 | [Frank](http://code.google.com/p/see3po/wiki/Frank) |
| 12/07/2009 | Added section 3.2.2 Floor Plan Processing | 0.5 | [Debarati](http://code.google.com/p/see3po/wiki/Debarati) |
| 12/07/2009 | Added Sections 2.1, 2.2. 2.4 (some skipped) | 0.6 | [Frank](http://code.google.com/p/see3po/wiki/Frank) |
| 12/08/2009 | Added Sections 3.2.3 Path Finding | 0.7 | [Nick](http://code.google.com/p/see3po/wiki/Nick) |
| 12/14/2009 | Added Section 3.3 Software System Attributes | 0.8 | [Frank](http://code.google.com/p/see3po/wiki/Frank) |

**Apendix A: Communication Protocols of See3PO**

Version: 2.0

Created by: Jacky

Modified by: Jacky

Last update: 10/23/2009

**Index:**

[1. Communication between local brain and control host 2](#_Toc244087935)

[1.1. Configurations 2](#_Toc244087936)

[1.2. Protocol: Host to local brain 2](#_Toc244087937)

[1.2.1. Motors Control Commands 2](#_Toc244087938)

[1.2.2. Motor System/Test Commands 3](#_Toc244087939)

[1.2.3. Servo Commands 3](#_Toc244087940)

[1.2.4. Disconnect Command 3](#_Toc244087941)

[1.3. Protocol: Local brain to host 4](#_Toc244087942)

[1.3.1. Disconnect Command 4](#_Toc244087943)

[2. Communication between local brain and ATOM 5](#_Toc244087944)

[2.1. Configurations 5](#_Toc244087945)

[2.2. Motors Control Commands 5](#_Toc244087946)

[2.3. Motor System/Test Commands 6](#_Toc244087947)

[3. Communication between local brain and servo control 7](#_Toc244087948)

[3.1. Configurations 7](#_Toc244087949)

[3.2. Servo Commands 7](#_Toc244087950)

1. Communication between local brain and control host
   1. Configurations

|  |  |
| --- | --- |
| Communication method | Ethernet over TCP/IP |
| Host setting | Static IP: 192.168.3.104 on host |
| Client Setting | DHCP from AC, 192.168.3.X on local brain |
| Other | Wireless, limited range. |

* 1. Protocol: Host to local brain
     1. Motors Control Commands

Using 8-byte fixed length packet to send commands to local brain. The following table shows the definition of each byte in a packet.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Byte 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0x01 | 0x10 | 0x11 | LWS\_H | LWS\_L | RWS\_H | RWS\_L | 0xEF |

* Byte 0: 0x01 = Motor commands
* Byte 1: 0x10 = Start byte of a command
* Byte 2: 0x11 = Keep driving
* Byte 3 and Byte 4: Left Wheels Speed

Speed is a 2-byte integer. Byte 3 is the high byte of the speed, byte 4 is the low byte.

* Byte 5 and Byte 6: Right Wheels Speed

Speed is a 2-byte integer. Byte 5 is the high byte of the speed, byte 6 is the low byte.

* Byte 7: End of transmission.

It is the ending byte of a packet, constant value 0xEF.

* + 1. Motor System/Test Commands

Using 8-byte fixed length packet to send commands to local brain. The following table shows the definition of each byte in a packet.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Byte 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0x01 | 0x10 | 0x00 | Sounds | 0x00 | 0x00 | 0x00 | 0xEF |

* Byte 0: 0x00 = System commands
* Byte 1: 0x10 = Start byte of a command
* Byte 2: 0x00 = System command (make sounds)
* Byte 3: Make 5 different kinds of sounds

Value from 1 to 5.

* Byte 4, Byte 5, and Byte 6: reserved
* Byte 7: End of transaction.

It is the ending byte of a packet, constant value 0xEF.

* + 1. Servo Commands

Dynamic length packet is used to send string commands to local brain. The last byte of the packet should be a “\r”.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Byte 0 | 1 | 2 | .. | .. | n-1 | n |
| 0x02 | “#” |  |  |  |  | “\r” |

* Byte 0: 0x02 = Servo commands
* Byte 1 ~ Byte n-1: the string command data in ASC II code

The string format is: “#16 P1500 #17 P1500 #18 P1400 …”

“#16 P1500” tells servo #16 moves to position 1500.

* Byte n: End of line.
  + 1. Disconnect Command

|  |  |  |
| --- | --- | --- |
| Byte 0 | 1 | 2 |
| 0x00 | 0x02 | EOT:0xEF |

* Byte 0: 0x00 = System commands
* Byte 1: 0x02 = disconnect from the host.
* Byte 2: End of transmission.
  1. Protocol: Local brain to host
     1. Disconnect Command

|  |  |  |
| --- | --- | --- |
| Byte 0 | 1 | 2 |
| 0x00 | 0x02 | EOT:0xEF |

* Byte 0: 0x00 = System commands
* Byte 1: 0x02 = disconnect from the host.
* Byte 2: End of transmission.

1. Communication between local brain and ATOM
   1. Configurations

|  |  |
| --- | --- |
| Communication method | Serial port over RS232 |
| Host setting | "COM1", 9600, No Parity, 8 bits data, 1 stop bits |
| Client Setting | 9600, No Parity, 8 bits data, 1 stop bits |
| Other |  |

* 1. Motors Control Commands

Using 7-byte fixed length packet to send commands to ATOM. The following table shows the definition of each byte in a packet.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Byte 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 0x10 | 0x11 | LWS\_H | LWS\_L | RWS\_H | RWS\_L | 0xEF |

* Byte 0: 0x10 = Start byte of a motor command
* Byte 1: 0x11 = Keep driving
* Byte 2 and Byte 3: Left Wheels Speed

Speed is a 2-byte integer. Byte 2 is the high byte of the speed, byte 3 is the low byte.

* Byte 4 and Byte 5: Right Wheels Speed

Speed is a 2-byte integer. Byte 4 is the high byte of the speed, byte 5 is the low byte.

* Byte 6: End of transmission.

It is the ending byte of a packet, constant value 0xEF.

* 1. Motor System/Test Commands

Using 7-byte fixed length packet to send commands to ATOM. The following table shows the definition of each byte in a packet.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Byte 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 0x10 | 0x00 | Sounds | 0x00 | 0x00 | 0x00 | EOT:0xEF |

* Byte 0: 0x10 = Start byte of a motor command
* Byte 1: 0x00 = System command (make sounds)
* Byte 2: Make 5 different kinds of sounds

Value from 1 to 5.

* Byte3, Byte 4, and Byte 5: reserved
* Byte 6: End of transmission.

It is the ending byte of a packet, constant value 0xEF.

1. Communication between local brain and servo control
   1. Configurations

|  |  |
| --- | --- |
| Communication method | Serial port over RS232 |
| Host setting | "COM2", 115200, No Parity, 8 bits data, 1 stop bits |
| Client Setting | 115200, No Parity, 8 bits data, 1 stop bits |
| Other |  |

* 1. Servo Commands

Dynamic length packet is used to send string commands to servo control. The last byte of the packet should be a “\r”.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | .. | .. | n-1 | n |
| “#” |  |  |  |  | “\r” |

* Byte 1 ~ Byte n-1: the string command data in ASC II code

The string format is: “#16 P1500 #17 P1500 #18 P1400 …”

“#16 P1500” tells servo #16 moves to position 1500.

* Byte n: End of line.