Software Engineering 104
Final Project
Monmouth University

Jessica Kostiou
June-August 2013

Table of Contents:

Problem Statement

This project demonstrates how a robot can take on a certain personality (cute or un-cute) based on the robot's movement, sound and light gestures.

Introduction and Overview

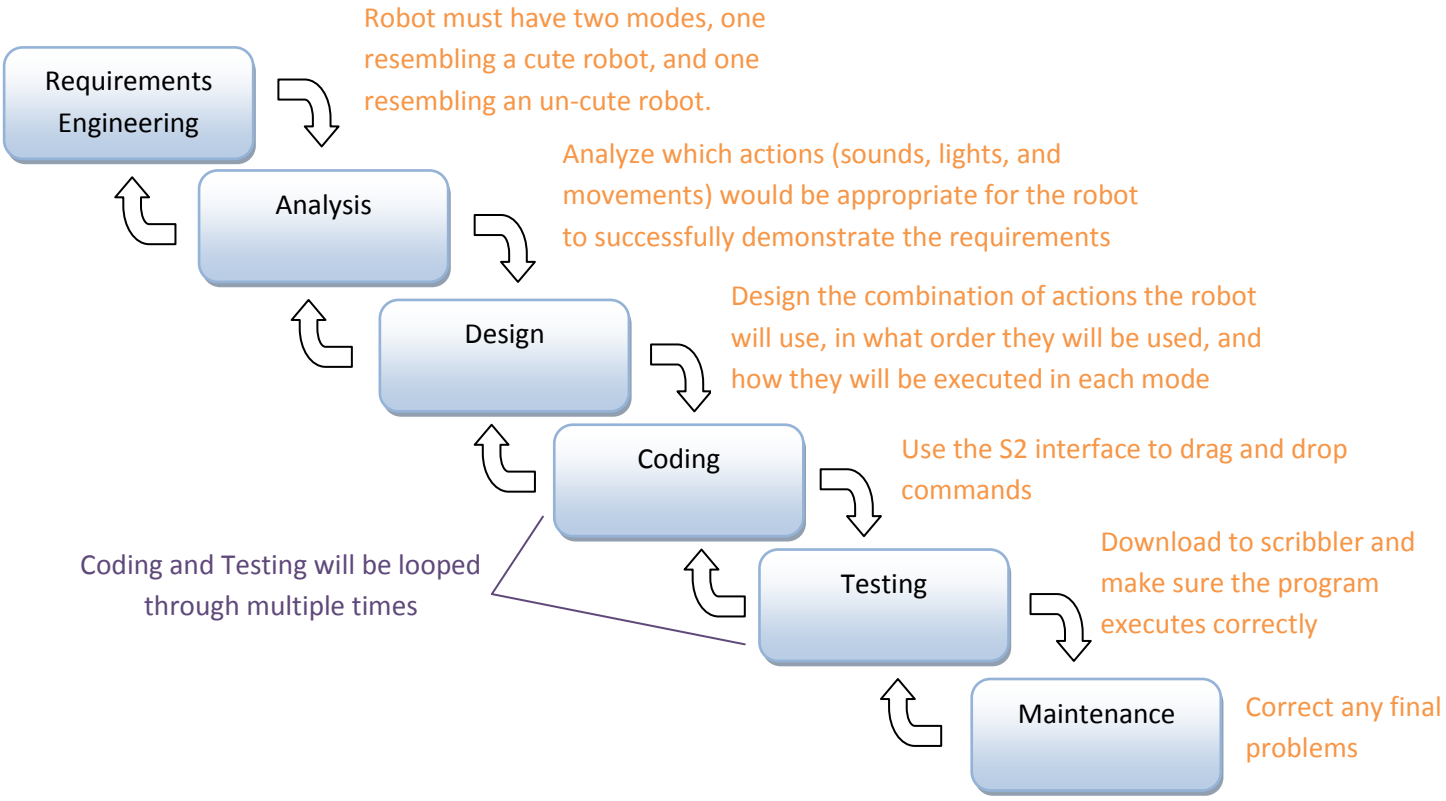
This project is derived from a similar project originating from the Museum of Modern Art in New York called "The Tweenbot Project." This exhibit was created to observe human robot interaction in an attempt to get individuals to guide the robot back to the museum. These people proved to be helpful in guiding the robot back to its "home" at MOMA.

The real question however, is whether the people assisting the robot were helping it because they were truly trying to be helpful, or if they were just reacting to the "adorable, lost little robot." This poses the question; can robots take on a personality type, even if they truly have no personality?

This project tests that question by designing two modes to the Scribbler S2 robot. These modes will make the robot act in a cute or un-cute manner. This will be done by programming the robot with varying assortments of movements, light signals, and sounds. The different "personalities" will then be tested in public and observed to see how humans interact based on the mode the robot is set on.

Process Model

Modified Waterfall Model:



The Modified Waterfall Method will most successfully allow for planning and development of programming a scribbler robot. Its linear structure allows the designer to go from one well defined step to the next. The modified waterfall method also allows the developer to go back to the previous step if necessary. More importantly, the Modified Waterfall Method allows for repeated coding and testing. This is necessary to test all the different combinations of actions in this project that need to successfully resemble a cute or un-cute personality for the robot (Robbie).

It is important that each step is executed in order for the robot to work correctly. The steps below must be followed for any project using the Modified Waterfall Method:

- Requirements Engineering: determine what the requirements of the project are
- Analysis: decide on how to meet those requirements
- Design: plan how to execute the requirements
- Coding: physically write the program
- Testing: ensure the code works
- Maintenance: correct any final issues

More specifically, for each step in the process model for the scribbler robot the corresponding orange text needs to be completed before moving onto the next step. If it is not completed, it will check and loop back to the previous step until the requirement is met. This is necessary to create a complete and well-developed product. After every step in this method is completed, the developer should have a fully functioning robot with two modes (cute and un-cute), and no bugs or errors in programming.

Requirements

Use cases:

Use Case #	1: Robot must act in either a cute or un-cute way
Description	User must be able to switch robot into either cute or un-cute mode determined by finger placement over the sensors of the robot.
Actors	User
Steps	
1.	Turn on robot
2.	Cover the sensor corresponding to the desired mode
3.	Place the robot on the floor
4.	Observe human response

Use Case #	**2:** Bumping into an object
Description	Robot shall be able to detect and bump into an object and respond accordingly given the mode
Actors	Robot
Steps	
1.	Turn on robot
2.	Place robot onto floor
3.	Power into program
4.	Select either cute or un-cute mode by covering sensors
5.	Allow robot to search for object using infrared detectors
6.	Sense object with crash detection
7.	Bump into object
8.	Respond to object appropriately given the mode

Use Case #	**3:** Running into a black line on a completely white surface
Description	Robot must be able to detect and search for a line
Actors	Robot
Steps	
1.	Turn on robot
2.	Place robot on floor
3.	Power into program
4.	Select either cute or un-cute mode by covering sensors
5.	Allow robot to search for line with color detection sensors
6.	Robot successfully recognizes black line

Requirements Tables:

Requirement #: 1	**Category:** Functional	**Use Case #: 1**
Description: Robot shall have a cute mode.		**Source:** Assignment outline
Criterion: •Must successfully make the robot have a "cute" demeanor using only sounds, lights and movements. •Shall be operated by covering the left light sensor.		
Dependencies: The user must cover the lights.		**Conflicts:** •User may cover the wrong combination •Must be sufficient lighting

Requirement #: 2	**Category:** Functional	**Use Case #: 1**
Description: Robot shall have an un-cute mode.		**Source:** Assignment outline
Criterion: •Must successfully make the robot have an "un-cute" demeanor using only sounds, lights and movements. •Shall be operated by covering no lights.		
Dependencies: The user must cover the lights.		**Conflicts:** •User may cover the wrong combination •Must be sufficient lighting

Requirement #: 3 **Category:** Functional **Use Case #: 2**

Description:
Robot shall be able to move around objects.

Source:
Assignment outline

Criterion:
Robot shall be able to maneuver around objects

Dependencies:
Correctly functioning infrared object detectors.

Conflicts:
May be a delay in detection.

Requirement #: 4 **Category:** Functional **Use Case #: 2**

Description:
Robot shall be able to react to bumping into an object

Source:
Assignment outline

Criterion:
Robot shall be able to move itself away from and react to an object after it has been bumped into.

Dependencies:
Correctly functioning infrared object detectors.

Conflicts:
- May be a delay in detection
- Object is moved by robot and is not detected

Requirement #: 5 **Category:** Functional **Use Case #: 3**

Description:
Robot shall be able to follow a line.

Source:
Assignment outline

Criterion:
Robot must continue to follow the path of a line once it is found.

Dependencies:
Correctly functioning line-detection sensors.

Conflicts:
The surface is not completely white and scribbler misreads a line, or cannot read it at all.

Requirement #: 6 **Category:** Functional **Use Case #: 3**

Description:
Robot shall be able to re-find a line again after it is lost.

Source:
Requirement 5

Criterion:
In the event where the robot loses the line, it must be able to look for and successfully find the line again.

Dependencies:
Correctly functioning line-detection sensors.

Conflicts:
The surface is not completely white and scribbler misreads a line, or cannot read at all

Requirement #: 7	**Category:** Functional	**Use Case #: 1**
Description: Robot may utilize its sounds at some point in both modes.		**Source:** User
Criterion: Shall be able to effectively sound to help enhance cute/ un-cute features.		
Dependencies: Correctly functioning lights.		**Conflicts:** •Speakers may malfunction / short out. • May have to stop actions to allow for lights to flash because of one processor.

Requirement #: 8	**Category:** Functional	**Use Case #: 1**
Description: Robot may utilize its lights at some point in both modes.		**Source:** User
Criterion: Shall be able to effectively blink lights to help enhance cute/ un-cute features.		
Dependencies: Correctly functioning lights.		**Conflicts:** •Lights may burn out or short out. • May have to stop actions to allow for lights to flash because of one processor.

Requirement #: 9	**Category:** Functional	**Use Case #: 1**
Description: Actions shall be fluid / relevant.		**Source:** Assignment outline
Criterion: All actions in either mode must make sense together in the order they are placed in. They must not flow together awkwardly or in a way that does not pertain to each specific mode.		
Dependencies: The programmer must correctly analyze the actions of the robot as they are programming,		**Conflicts:** Execution of actions may not appear the way it is illustrated in the drag and drop environment.

Requirement #: 10	**Category:** Non-Functional	**Use Case #: 1**

Description:
Robot must evoke the appropriate reaction from the viewer.

Source:
Assignment outline

Criterion:
Robot must receive the appropriate response from the viewers based on if it is trying to be "cute" or "un-cute." If the robot receives the opposite of the reaction that is expected of the viewer, the project is unsuccessful.

Dependencies:
The viewer is an individual who will give a definite variance in emotion based on the mode the robot is currently in.

Conflicts:
- The viewer may have an unclear opinion of what is "cute" and "un-cute" in relation to robots (or in general.)
- The viewer is unemotional or unaffected by the performance.

Requirement #: 11	**Category:** Non-Functional	**Use Case #: 1**

Description:
The robot's modes shall be noticeably different.

Source:
Assignment outline

Criterion:
The viewer shall be able to differentiate which mode is which when accessed by the robot. There should be no event where the user cannot tell which mode is "cute" and which mode is "un-cute." In the event where this happens, the program is unsuccessful.

Dependencies:
- The programmer must determine which actions are "cute" and "un-cute" when programming.
- The viewer must be qualified to differentiate between "cute" and "un-cute."

Conflicts:
Either the programmer or the viewer cannot successfully determine what is "cute" or "un-cute."

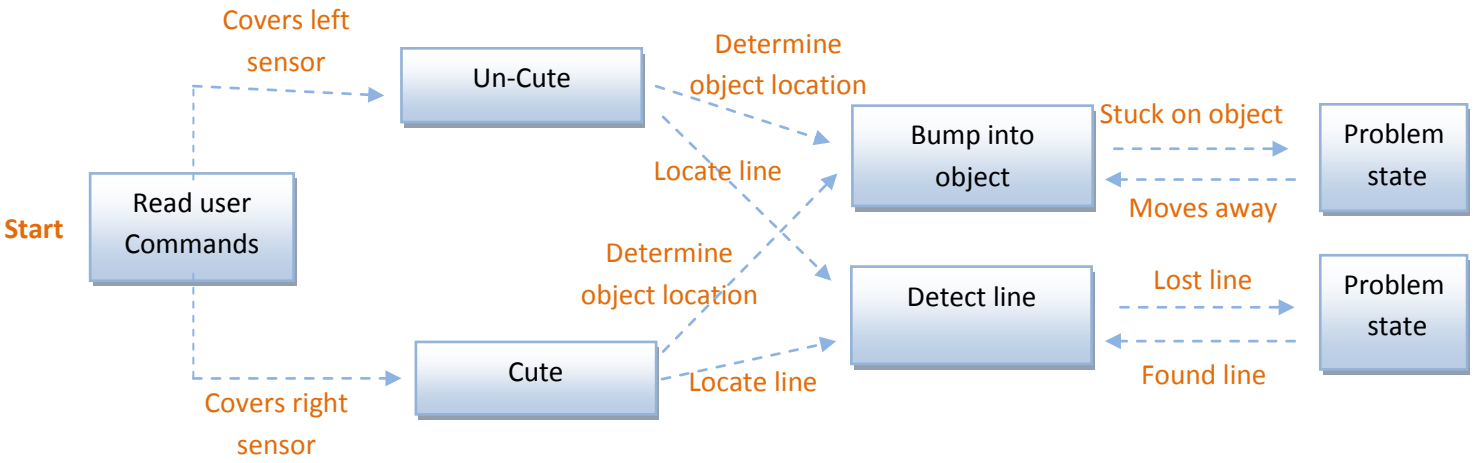
Software Design

Concept of Execution:

The program run by the robot will be determined by the user at runtime. There will be two modes booted into by covering two different combinations of light sensors on the top of the scribbler robot. The cute mode will be accessed by covering the left most light, and the un-cute mode will be accessed by not covering any lights.

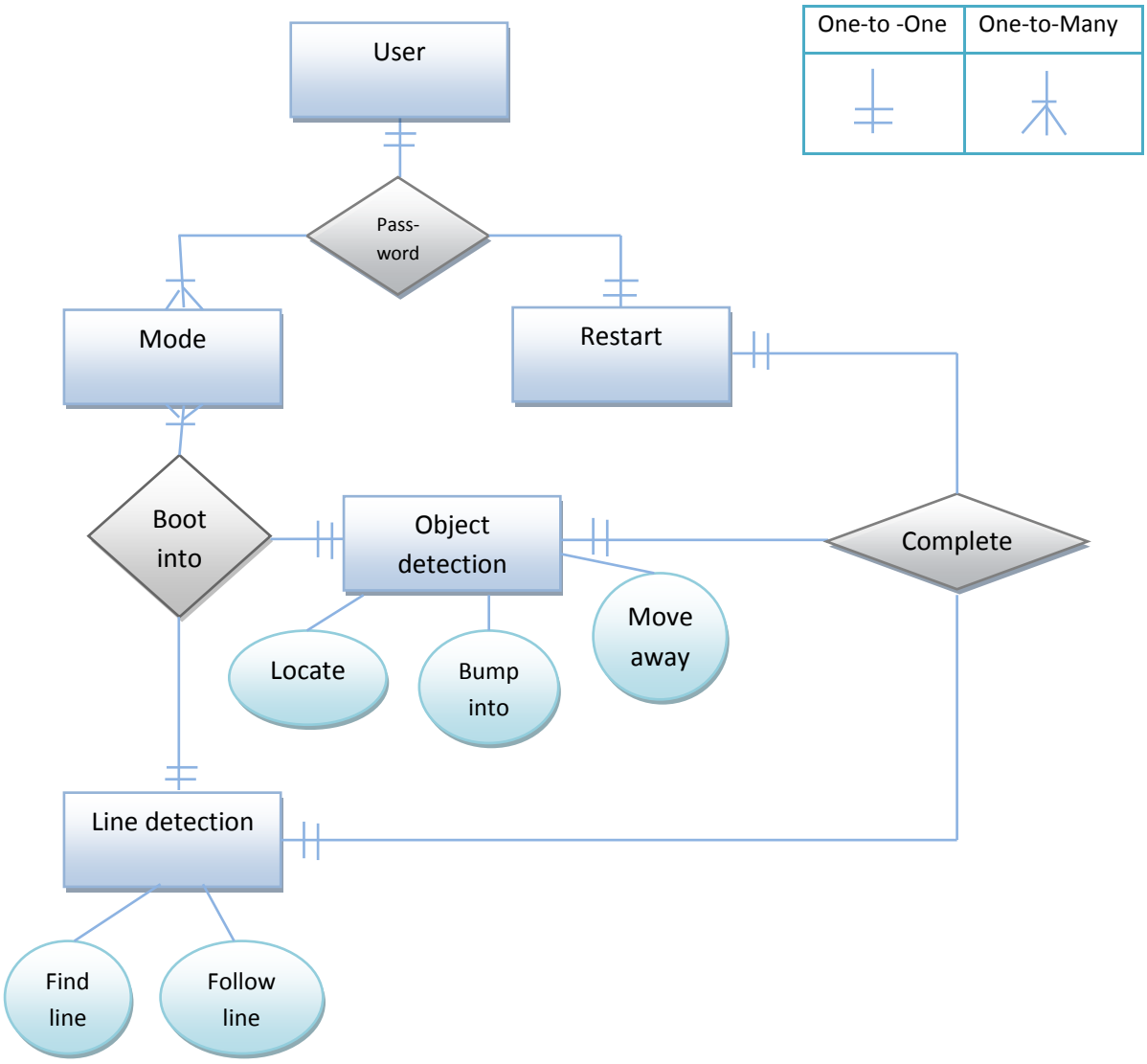
Likewise, each mode will test the requirements listed above. Both the cute and un-cute modes will have the robot bump into an object and follow lines in the appropriate fashion. Once this mode is chosen, the program will follow other modules, such as navigating through other objects, detecting objects, getting un-stuck from other objects, etc. The robot will also be designed to continue following a line once it is located and re-locate a line when it is lost.

State Transition Diagram:



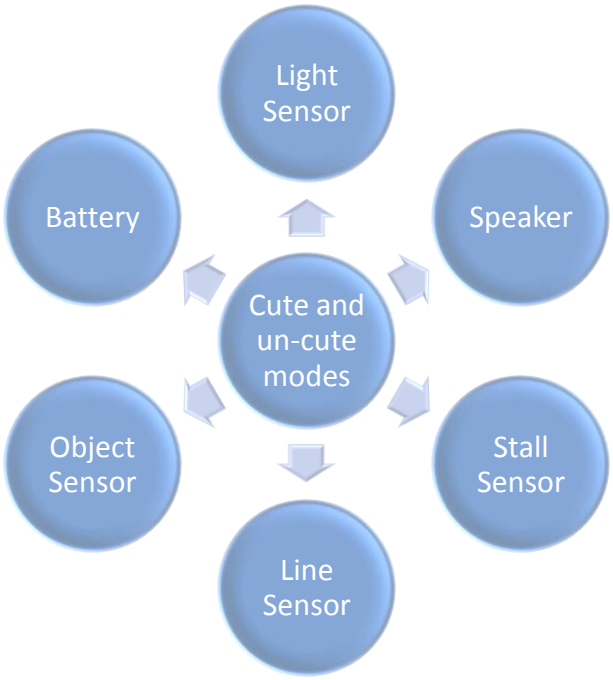
The states of the Scribbler S2 robot in relation to this program are listed above represented by boxes. Each description in orange is a situation relating to the state. First the mode is determined by the user. Both modes will bump into objects and detect lines. Each state has the potential for error, so when this occurs the error is evaluated and corrected by the problem state.

Entity- Relationship Diagram

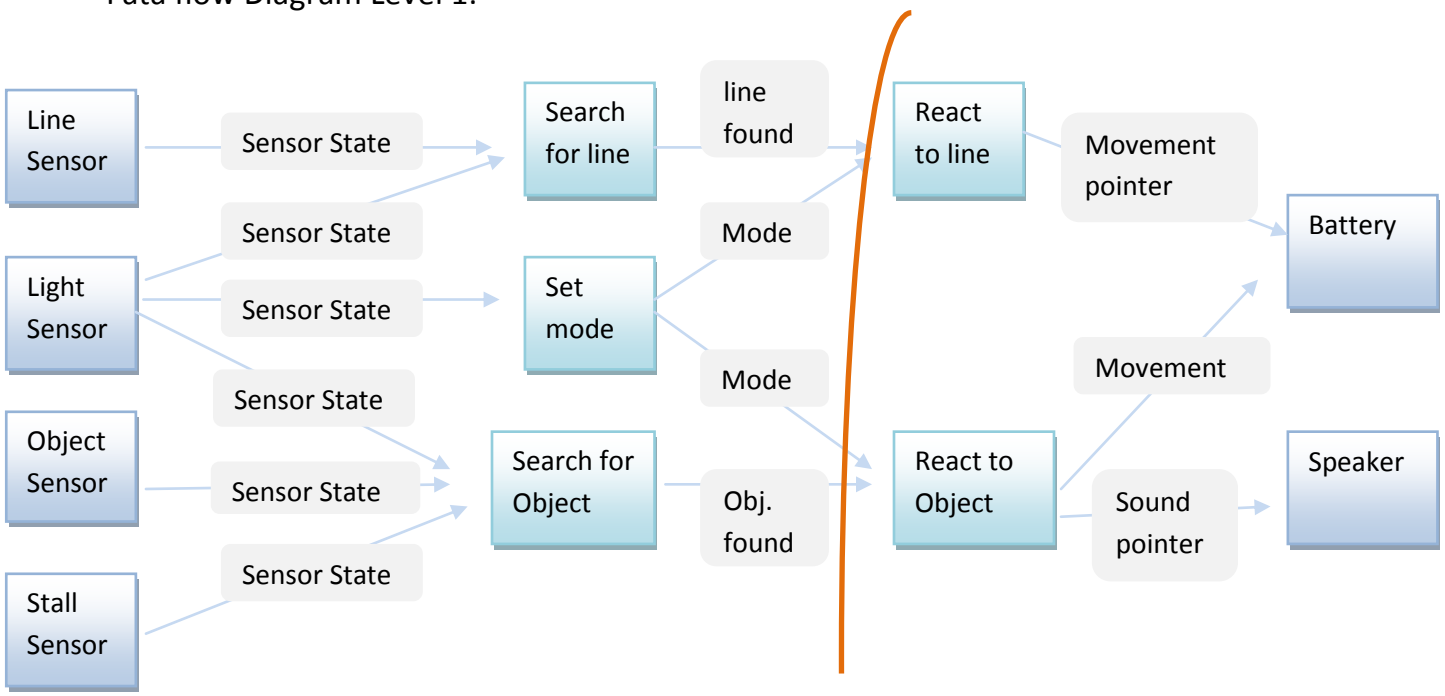


None of the entities in this diagram are mandatory. Each must be completed in order for the program to successfully meet all of its requirements. It is first started by the user who enters a light combination which either boots it into a mode, or restarts the program if the combination is incorrect. The mode determines which task the robot is going to perform: cute to line detection and un-cute to object detection. After the mode's tasks are completed, the program restarts.

Data Flow Diagram level 0:

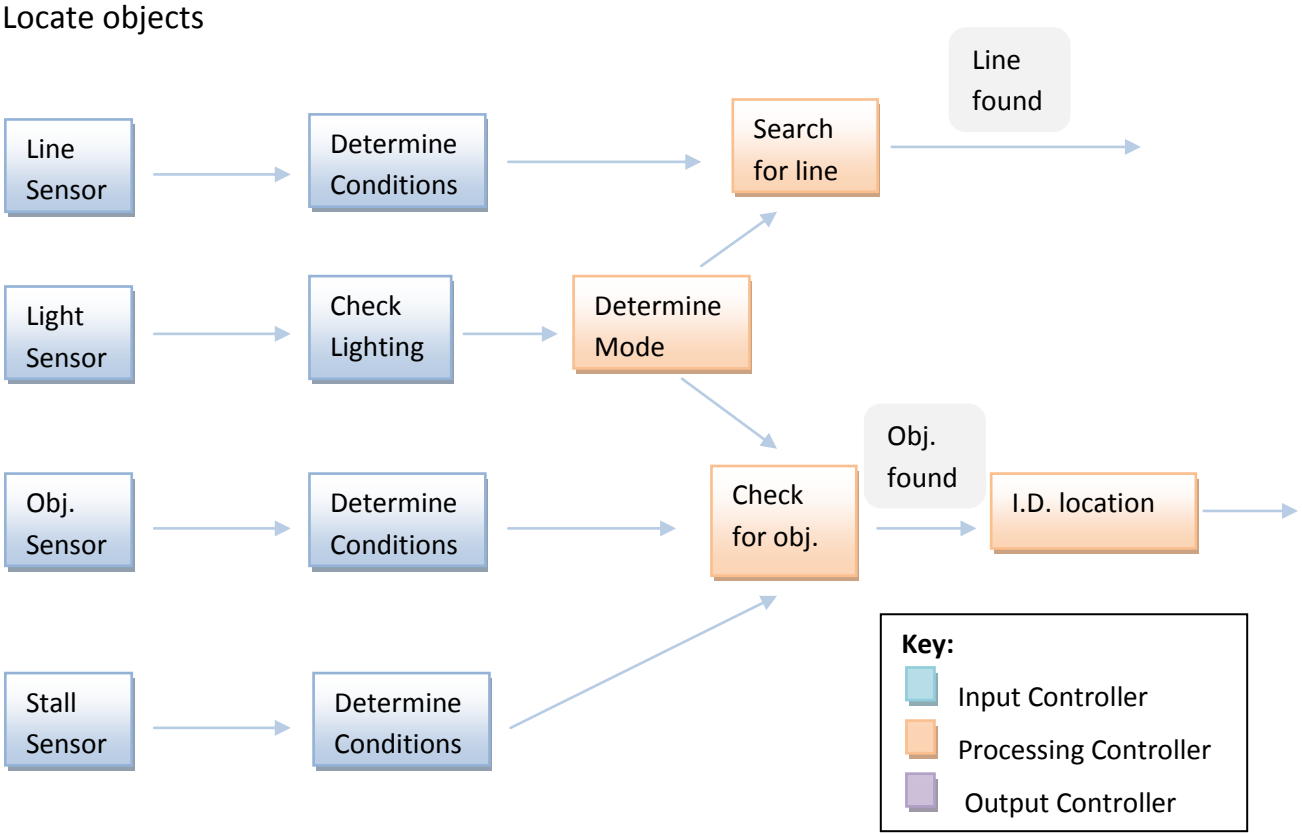


Fata flow Diagram Level 1:



Data Flow Diagram Level 2:

Locate objects



Line Sensor: Determines the conditions involving the line sensor.

Light Sensor: Checks the lighting in the environment.

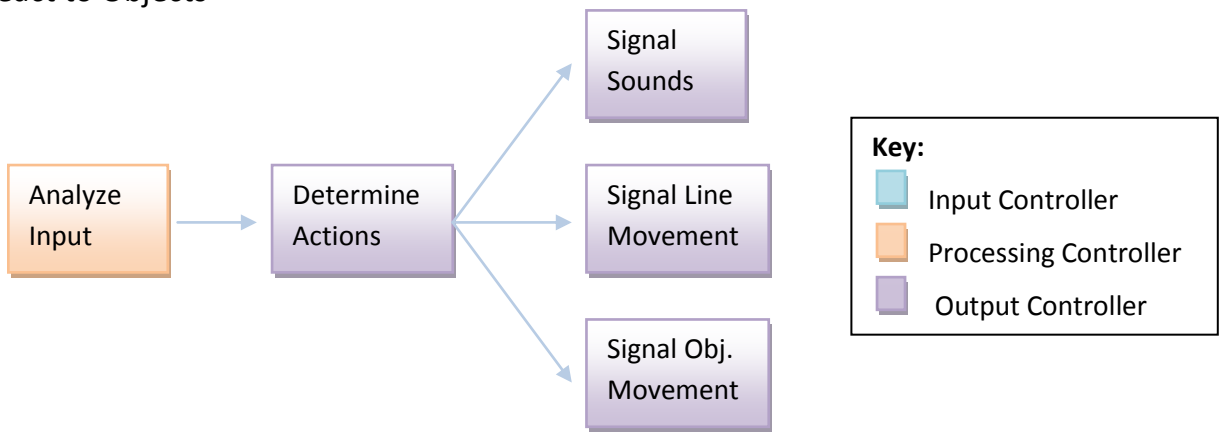
Objet Sensor: Determines the conditions involving the object sensor.

Stall Sensor: Determines the conditions involving the stall sensor.

Determine mode: based on lighting, S2 will boot into either the cute or un-cute mode.

Search for line: Checks the ground to determine if there is a line.

React to Objects



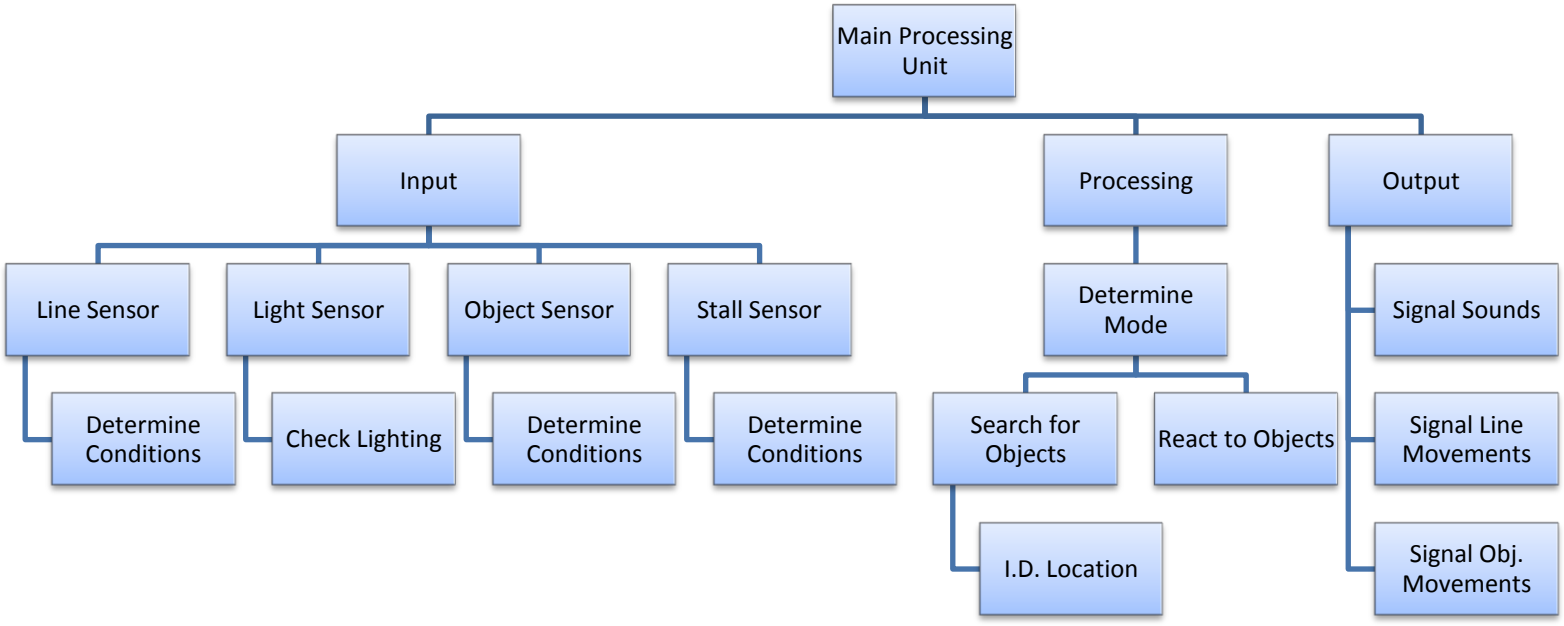
Analyze Input: Based on that the sensor gathers from the environment, the S2 determines what reaction to have based on the mode it is in.

Signal Sounds: S2 sound is projected out by its speakers.

Signal Line Movement: Follow line on surface.

Signal Object Movement: Move away from object/ recover from running into object.

Software Item Detailed Design



The input stage gathers information from the S2's four sensors: line, light, object and stall. The Light sensor does this by reading the amount of light in from the area through its three light sensors. This determines the mode of the program. The remaining three sensors do so by determining the conditions of the environment.

The Processing stage physically boots the robot into either the cute or un-cute mode. From that point the robot does two things: searches for objects and lines and reacts to objects and lines. Once the robot finds the object it identifies its location.

From then on, the program has the robot create a sound or perform a movement based on what mode it is in. The robot can move in two ways: either in reaction to a line, or an object.

Test Plan

Introduction:

The system is designed to be powered into a cute or un-cute mode depending on which light sensor the user covers. The robot then reacts to objects and follows lines in a manner dependent on which mode it is in.

In order to test the system, a test environment must be created on a plane white surface where the robot can make out lines drawn on the surface. Lines and objects will then be added. A non-reflective black tape, or paint must be used to create lines. The robot will be turned on and powered into each mode and placed into the environment by the tester. It will then avoid objects and follow lines in either a cute or un-cute manner.

In the event where the robot runs into an object, the robot should spin around and continue moving in another direction. In the event where this does not happen, the test has failed. In this case the programmer must return to the program and determine a solution. If a robot cannot follow or react to a line when it is driven upon, the test has also failed and the programmer will have to return to the code and correct the problem.

The robot must also be deemed cute or un-cute by the user. The tester should also check to make sure the robot is reacting to objects in the appropriate fashion. If it does not, the test has failed and actions must be re-programmed.

Test Cases:

Test Case #: 1	**Req. #: 1, 2**
Description: Robot powers into correct mode based on light sensor combination	**Conditions:** •Must be sufficient lighting •Must cover correct combination
Expected Results: •Cover left light sensor: power into cute mode •Cover no light sensor: power into un-cute mode	**Upon Failure:** •Make sure test case is valid •Consult software • Locate problem and correct code

Test Case #: 2	**Req. #: 4, 6**
Description: Robot follows a line	**Conditions:** •A line Exists •Place on line
Expected Results: •Center light flash •Sound signal •Continue straight	**Upon Failure:** •Make sure test case is valid •Consult software • Locate problem and correct code

Test Case #: 3	**Req. #: 4, 6**
Description: Robot detects a line on the left	**Conditions:** •A line Exists •Place robot to right of line
Expected Results: •Left light flash •Turn slightly left	**Upon Failure:** •Make sure test case is valid •Consult software • Locate problem and correct code

Test Case #: 4	**Req. #: 4, 6**
Description: Robot detects a line on the right	**Conditions:** •A line Exists •Place robot to left of line
Expected Results: •Right light flash •Turn slightly right	**Upon Failure:** •Make sure test case is valid •Consult software • Locate problem and correct code

Test Case #: 5	**Req. #: 3, 6, 7**
Description: Robot detects an object on the left	**Conditions:** •An object must exist •Place object on left
Expected Results: •Left light flashes •Sound signal •Robot turns to the right	**Upon Failure:** •Make sure test case is valid •Consult software • Locate problem and correct code

Test Case #: 6	**Req. #: 3, 6, 7**
Description: Robot Detects an object on the right	**Conditions:** •An object must exist •Place object on right
Expected Results: •Right light flashes •Sound signal •Robot turns to the left	**Upon Failure:** •Make sure test case is valid •Consult software • Locate problem and correct code

Test Case #: 7	**Req. #: 3, 6, 7**
Description: Robot Detects an object straight ahead	**Conditions:** •An object must exist •Place object in front of robot
Expected Results: •Center light flashes •Sound signal •Robot turns completely around	**Upon Failure:** •Make sure test case is valid •Consult software • Locate problem and correct code

Test Case #: 8	**Req. #: 3, 6, 7**
Description: Robot runs into an object	**Conditions:** •An object must exist •Place object in front of robot
Expected Results: •Al lights flash •Reverse •Turns completely around •Sound signal	**Upon Failure:** •Make sure test case is valid •Consult software • Locate problem and correct code

Test Case #: 9	**Req. #: 1**
Description: Robot performs in a cute fashion	**Conditions:** •Must be a test viewer •Must observe and record viewers reactions
Expected Results: Viewer agrees that the robot is cute	**Upon Failure:** •Make sure test case is valid •Consult viewer to determine errors • Edit code

Test Case #: 10	**Req. #: 2**
Description: Robot performs in an un-cute fashion	**Conditions:** •Must be a test viewer •Observe and record viewers reactions
Expected Results: Viewer agrees that the robot is un-cute	**Upon Failure:** •Make sure test case is valid •Consult viewer to determine errors • Edit code

Traceability Table:

Req. #	Requirement Description	Source of Requirement	SW Module(s) implementing the Requirement	Test Case(s) testing the requirement	Requirements Traceability Test Case #
1	Have a cute mode	Assignment outline	Main processing unit / cute mode	9-Performs cute	Tests to make sure the viewer perceives the actions as cute
2	Have an un-cute mode	Assignment outline	Main processing unit / un-cute mode	10-Performs un-cute	Tests to make sure the viewer perceives the actions as un-cute
3	Able to move around objects	Assignment outline	Object detection sensor / un-cute and cute modes	5- Obj. on left 6-Obj. on right 7- Obj. in front	Tests to detect objects on the left, right, and front and react to them appropriately
4	Able to react to bumping into an object	Assignment outline	Object detection sensor / un-cute and cute modes	8- Runs into object	Tests to make sure the robot acknowledges and reacts to bumping into something
5	Able to follow a line	Assignment outline	Line detection sensor/ un-cute and cute modes	2- Follow line 3-Line on left 4- Line on right	Tests to detect lines on the left, right and center and make sure robot turns properly to stay on line
6	Able to re-find a line	Requirement 4	Line detection sensor / un-cute and cute modes	2- Follow line	Tests to make sure robot stays straight on a line
7	May utilize sounds	User	Cute/ un-cute modes	All	Every test case makes sure the robot properly utilizes sounds to emphasize cute or un-cute mode/ emphasize actions
8	May utilize lights	User	Cute/ un-cute modes	5- Obj. on left 6-Obj. on right 7- Obj. in front 8- Run into obj.	Tests to make sure objects are being detected by flashing lights
9	Movements are fluid	Assignment outline	Cute/ un-cute modes	11- Actions and sounds cohesive	Tests to make sure sounds do not interfere with motions
10	Evoke appropriate user reaction	Assignment outline	Cute/ un-cute modes	9-Performs cute 10-Performs un-cute	Tests to make sure the user reacts appropriately to each mode
11	Modes are noticeably different	Assignment outline	Cute/ un-cute modes	9-Performs cute 10-Performs un-cute	If the user reacts to each mode appropriately (above), the modes are therefore different