Department of Computer Science and Engineering  
The University of Texas at Arlington

Team Sketchers

Sidewalk Sketcher

Team Members:   
*Jesus Parra  
Pranil Maharjan  
Sabin Bajracharya*

*Nischal Pandey  
Lam Pham*

Late Updated: 10 February 2015 @ 7:45:00 AM

Table of Contents

[1. Introduction 4](#_Toc411317667)

[2. Architecture Overview 4](#_Toc411317668)

[3. System Hardware Description 4](#_Toc411317669)

[3.1 Raspberry Pi Model B+ 4](#_Toc411317670)

[3.2 2GB Micro SD Card 5](#_Toc411317671)

[3.3 Raspberry Pi Camera 6](#_Toc411317672)

[3.4 Panoramic Lens 7](#_Toc411317673)

[3.5 Roomba® iCreate 2 8](#_Toc411317674)

[3.6 Stepper Motor 9](#_Toc411317675)

[3.7 LED’s 10](#_Toc411317676)

[3.8 Broom Stick 10](#_Toc411317677)

[3.9 Swim Noodle 11](#_Toc411317678)

[3.10 Speakers 11](#_Toc411317679)

[3.11 Wood 12](#_Toc411317680)

[4. Software Input Layer 13](#_Toc411317681)

[4.1 Image Reader 13](#_Toc411317682)

[4.2 Converter System 13](#_Toc411317683)

[4.3 Transfer Data 14](#_Toc411317684)

[5. Software Output Layer 15](#_Toc411317685)

[5.1 Data Packager 15](#_Toc411317686)

[5.2 File Generator 15](#_Toc411317687)

[5.3 File transfer 15](#_Toc411317688)

[6. Software Processing Layer 16](#_Toc411317689)

[6.1 Image Processing 16](#_Toc411317690)

[6.2 Information Processing 16](#_Toc411317691)

[7. User Interface Layer 17](#_Toc411317692)

[7.1 File Browser Subsystem 18](#_Toc411317693)

[7.2 File Browser Subsystem 19](#_Toc411317694)

[7.3 Cropping Subsystem 19](#_Toc411317695)

[7.4 Resize Subsystem 19](#_Toc411317696)

[7.5 Color Selector Subsystem 19](#_Toc411317697)

[8. Data Storage Layer 19](#_Toc411317698)

[8.1 Image Repository 19](#_Toc411317699)

[8.2 File Repository 20](#_Toc411317700)

[8.3 Database Management 20](#_Toc411317701)

[9. Hardware Input Layer 22](#_Toc411317702)

[9.1 File Reader Subsystem 22](#_Toc411317703)

[9.2 Synchronization Input Subsystem 22](#_Toc411317704)

[9.3 Camera Processing 22](#_Toc411317705)

[10. Hardware Output Layer 23](#_Toc411317706)

[10.1 Alarm Subsystem 23](#_Toc411317707)

[10.2 Output De-multiplexer 23](#_Toc411317708)

[11. Hardware Processing Layer 24](#_Toc411317709)

[11.1 Hardware Input Driver Subsystem 24](#_Toc411317710)

[11.2 Hardware Output Driver Subsystem 24](#_Toc411317711)

[11.3 Synchronization Subsystem 25](#_Toc411317712)

[11.4 Sketcher Processing Subsystem 27](#_Toc411317713)

[11.5 Position Processing Subsystem 28](#_Toc411317714)

[12. Sketch Layer 29](#_Toc411317715)

[12.1 Sketcher Synchronization Subsystem 29](#_Toc411317716)

[12.2 Depletion Subsystem 30](#_Toc411317717)

[13. Motion Layer 31](#_Toc411317718)

[13.1 Motion Synchronization Subsystem 31](#_Toc411317719)

[13.2 Motion-Driver Subsystem 31](#_Toc411317720)

[14. Quality Assurance 33](#_Toc411317721)

[15. Requirements Mapping 33](#_Toc411317722)

[16. Acceptance Criteria 34](#_Toc411317723)

[17. Appendix 34](#_Toc411317724)

# Introduction

This document provides a detail design specification of the robot Sidewalk Sketcher. It will explain the project over view as well as elaborate the hardware and software components used in the robot. This document will describe the architecture over view of the robot and give an insight description of layers used. In addition it will provide knowledge about Sidewalk Sketchers’ subsystems and their modules. Next the document will provide an explanation of how each layer, subsystem and modules communicate through dataflow diagram. A table of requirement mapping is provided to show how the requirements are accomplished by the robot and finally, an over view of acceptance criteria is provided for the quality assurance.

# Architecture Overview

# System Hardware Description

This section describes the hardware components that will make up the Sidewalk Sketcher. The hardware information provided here covers the quantity required, manufacturer specifications, its intended role in the system, a brief description of how it operates, and other components that will interface with the hardware.

## Raspberry Pi Model B+



**Figure 3-1** Raspberry Pi Model B+

### Quantity: The Sidewalk Sketcher will require one Raspberry Pi.

### Purpose: The Raspberry Pi will be the main processor of the Sidewalk Sketcher. This device will program the iCreate, control the chalk flow, and just overall control the physical component of the Sidewalk Sketcher. Its 40 GPIO pins will be used to retrieve and send information to the iCreate and a stepper motor controlling the chalk unit. This device will also be in charge of receiving the output provided by the user interface and using this information to sketch an image.

### Specifications:

|  |  |
| --- | --- |
| Model | Raspberry Pi B+ |
| SoC | Broadcom BCM2835 |
| CPU | 700 MHz ARM1176JZF-S core |
| GPU | 25 MHz Broadcomm IV |
| RAM | 512MB |
| USB | 4 2.0 Onboard USB ports |
| Video Out | HDMI uo to 1920x1200 resolution, PAL, NTSC |
| Audio Out | 3.5mm Jack, HDMI |
| Storage | Onboard SD/MM/SDIO card slot |
| Network | 10/100mbps Ethernet, RJ-45 |
| Peripherals | GPIO, UART, SPI, IIC +3.3, +5.0V |
| Power | 700 mA, 5V via MicroUSB or GPIO header |
| OS | Raspbian |

### Interfaces: Since the Raspberry Pi is the central unit of control, it will interface with the iCreate, a stepper motor, LEDs and speakers.

## 2GB Micro SD Card



**Figure 3-2** 8GB Micro SD card

### Quantity: The Sidewalk Sketcher will require one 2GB micro SDHC memory card.

### Purpose: The operating system running on the Raspberry Pi will be stored on the memory card. Most of the system data will also be stored on the SD card.

### Specifications:

|  |  |
| --- | --- |
| Model | SanDisk |
| Capacity | 2GB |
| Transfer Rate | 4Mbps |

### Interfaces: The SD card will interface with the Raspberry Pi via its SD slot.

## Raspberry Pi Camera



### Quantity: The Sidewalk Sketcher will require one Raspberry Pi Camera.

### Purpose: This camera will be used with the panoramic lens to calculate the position of the robot using trigonometry based on the markers.

### Specifications:

|  |  |
| --- | --- |
| Model | Raspberry Pi Camera Board v1.3 |
| Manufacturer | Omnivision 5647 Camera Module |
| Resolution | 2592 x 1944 |
| Video | Supports 1080p @ 30fps, 720p @ 60fps and 640x480p 60/90 Recording |
| Type | 15-pin MIPI Camera Serial Interface |
| Size | 20 x 25 x 9mm |
| Weight | 3g |

### Interfaces: The camera will interface with the Raspberry Pi through the ribbon cable.

## Panoramic Lens



### Quantity: The Sidewalk Sketcher will require one panoramic lens.

### Purpose: The panoramic lens will be used with the Raspberry Pi to have a 360 degree view of the surrounding to view the markers. With these markers, the relative position of the iCreate will be calculated.

### Specifications:

|  |  |
| --- | --- |
| Model | Universal 3 in 1 Clip Fisheye Lens Wide Angle Lens Macro Lens |
| Manufacturer | Action1st |
| Size | 20 x 25 x 9mm |
| Weight | 1.6 ounces |

### Interfaces: The panoramic lens will interface with the camera by being placed on top of the camera.

## Roomba® iCreate 2



### Quantity: The Sidewalk Sketcher will require one Roomba ® iCreate 2.

### Purpose: The iCreate 2 will provide the movement of the Sidewalk Sketcher. It will carry the Raspberry Pi along with the stepper motor that will control the chalk piece.

### Specifications:

|  |  |
| --- | --- |
| Model | iRobot Create® 2 Programmable Robot |
| Manufacturer | iRobot |
| Size | 13.39 in Diameter by 3.62 in |
| Weight | 7.9 pounds |

### Interfaces: The iCreate 2 will interface directly with the Raspberry Pi.

## Stepper Motor



### Quantity: The Sidewalk Sketcher will require one stepper motor.

### Purpose: The stepper motor will be attached to a chalk holder that would be used to control the chalk.

### Specifications:

|  |  |
| --- | --- |
| Model | 28BYJ48-12-300-01 |
| Manufacturer | Changzhou Fulling Motor Co.,Ltd |
| Rated Voltage | 12V |
| No. of Phase | 2 |
| Resistance per Phase | 300 |
| Gear Reduction Ratio | 1:64 |
| Step Angle | 5.625º /64 |

### Interfaces: The stepper motor will interface directly with the Raspberry Pi.

## LED’s



### Quantity: The Sidewalk Sketcher will use two LED’s.

### Purpose: The LED’s will serve the purpose to alert the user when the robot is low on battery and a reload of chalk is needed.

### Specifications:

|  |  |
| --- | --- |
| Model | LED - Basic Red 5mm and LED - Basic Yellow 5mm |
| Manufacturer | Chine Young Sun LED Technology Co. LTD |
| Peak Forward Current | 30mA |
| Operation Temperature | -40~85º C |

### Interfaces: These LED’s will interface with the Raspberry Pi.

## Broom Stick



### Quantity: The Sidewalk Sketcher will have four broom stick wooden poles.

### Purpose: The wooden sticks will be used as the trunk for the markers. They will be attached to a wooden based made from wood.

### Specifications:

|  |  |
| --- | --- |
| Size | 1 inch diameter by ~5 feet |
| Quantity | 4 |

### Interfaces: These wooden poles will be mounter on top of a bass created from wood.

## Swim Noodle



### Quantity: The Sidewalk Sketcher will use 3 different color swim noodes.

### Purpose: The swim noodles will serve the purpose to label the markers so that the camera is able to calculate the relative position using trigonometry and polar coordinates.

### Specifications:

|  |  |
| --- | --- |
| Size | 51.5 x 1.8 x 1.8 inches |
| Quantity | 3 |

### Interfaces: These swim noodles will be mounted on top of the wooden sticks and will be simply serve the purpose to label each marker.

## Speakers



### Quantity: The Sidewalk Sketcher will use one speaker.

### Purpose: The purpose of these speakers will be to amplify the alarm the Sidewalk Sketcher will generate when the chalk is near depletion.

### Specifications:

|  |  |
| --- | --- |
| Size | 32 mm by 5.1mm |
| Frequency | 280 Hz |
| Manufacturer | DB Unlimited |

### Interfaces: This speaker will interact directly with the Raspberry Pi.

## Wood



### Quantity: The Sidewalk Sketcher will require 4 pieces of 2” by 4: by 12” which can be attained from one piece of 2” by 2: by 8’.

### Purpose: These pieces of wood will serve as the based for the markers used for position because they will allow the markers to stand without much/if any movement.

### Specifications:

|  |  |
| --- | --- |
| Size | 2 inches by 4 inches by 8 feet |
| Quantity | 1 |

### Interfaces: These pieces of wood will serve as the base for the wooden sticks.

# Software Input Layer

The purpose of the Software Input Layer is to accept input from the User Interface and output the image in way the system can manipulate. This layer is responsible for reading the image file, converting the image to an appropriate data file, and outputting the data to the Software Processing Layer for final processing.

## Image Reader

### Image Scan

**Prologue**

The Image Scan will read the image as the Image Reader Subsystem analyzes the image file. The module will maintain the image file upon request and is responsible for verifying that the image sent by the user interface is a valid image. If not the Image Scan is to deny the image before conversion.

**Interfaces**

|  |  |  |  |
| --- | --- | --- | --- |
| Source | Sink | Input Data to sink | Output |
| Image Scan | Image Converter | Image File | Data File |

**External Data Dependencies**

This action requires the user to send the image to process

**Internal Data Dependencies**

N/A

**Pseudo Code**

If (buffer != null)

{

parse buffer

grab last value

Check file name,

if valid, continue

Else

report invalid image

}

## Converter System

### Image Converter

**Prologue**

The Image Converter will receive the image file from the Image Scan and convert the image into a data file. This Data file will be used to instruct the Sidewalk Sketcher on how to plot the image. After the Image Converter has successfully completed the conversion, the data file will be transferred to the Transfer Data Subsystem for deployment.

**Interfaces**

|  |  |  |  |
| --- | --- | --- | --- |
| Source | Sink | Input Data to sink | Output |
| Image Converter | Image Transfer | Data File | Data File |

**External Data Dependencies**

N/A

**Internal Data Dependencies**

This action will be written in Java to parse the data file for conversion

**Psuedo Code**

tempImgFile = imgBuffer

dataFile = imageConvert(tempImgFile)

## Transfer Data

### Image Transfer

**Prologue**

This module is responsible for retrieving the data file from the Image Converter module and deploying the data file to the Image Processing subsystem. The Image Transfer module will be on stand by for any incoming images to be transferred. Once the Image Transfer module has completed the transfer, the Image Transfer module will return to stand by. The Image Transfer module will not have access in modifying the data file.

**Interfaces**

|  |  |  |  |
| --- | --- | --- | --- |
| Source | Sink | Input Data to sink | Output |
| Image Transfer |  | Data File | Data File |

**External Data Dependencies**

N/A

**Internal Data Dependencies**

This action will be written in Java for the Image Transfer module

**Pseudo Code**

While(transferFile != null)

{

deployFile(transferFile)

}

# Software Output Layer

The purpose of the Software Output Layer is to provide processed information or requested data to the hardware component in a format that it will be able to process. The output of the software component for the Sidewalk Sketcher will be handled in this layer and this will ultimately serve as the input for the hardware component.

## Data Packager

## File Generator

## File transfer

# Software Processing Layer

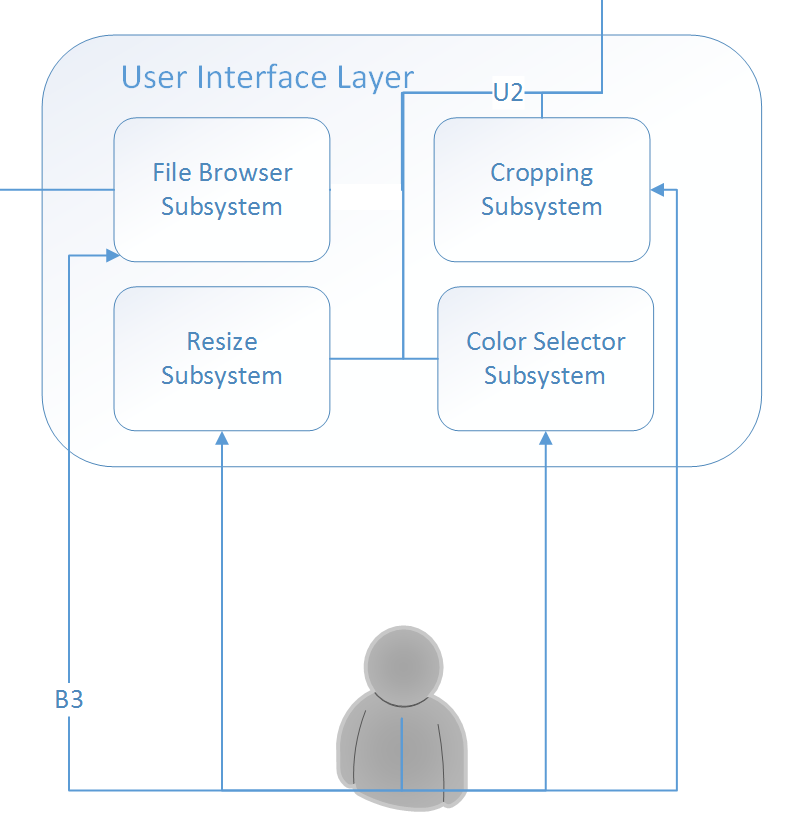
The purpose of the Software Processing Layer is to handle all the software input data, process that data and then send it to the output layer. Also, the processing layer should retrieve and save files in the data storage layer. This layer will handle all of the processing involved in the back end logic that the user will request through the User Interface Layer.

## Image Processing

## Information Processing

# User Interface Layer

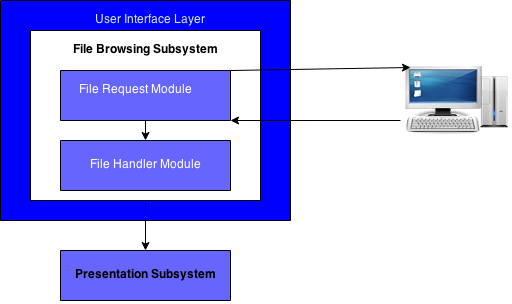
The User Interface Layer provides user a medium to create processed image required for our robot Sidewalk sketcher. In this layer the file browser subsystem allows user to select an image from the user computer. The image is then displayed through presentation layer. Next, the layer provides user options to resize the image to be produced through resize subsystem, crop image through cropping subsystem and choosing colors to print final image through color selector subsystem. All these options are displayed on screen through presentation subsystem.



## File Browser Subsystem

The file browser subsystem allows user to select an image file from the users computer. The selected image is then used for image processing by other layers and subsystem.

**File request module**



**Prologue**

The File request module makes a request for an image file with the users computer database for file type, JPEG, PNG,JPG etc.

**Interface**

|  |  |  |  |
| --- | --- | --- | --- |
| Source | Sink | Input Data to sink | Output |
| Windows computer | File Request Module | Request image file | N/A |
| File request module | Windows Database | N/A | Image File |

**External Dependencies**

The module requires user to select the appropriate image file to complete the process

**Internal Dependencies.**

The action for the module will be developed using Java language.

**Service Dependencies**

The action listener will completely depend on GUI developed in java.

**Pseudo code**

Below is the example of the

Get\_Image {

Open the Jfile chooser and request an image

}

Set\_Image{

Set an image in an array

}

## File Browser Subsystem

## Cropping Subsystem

## Resize Subsystem

## Color Selector Subsystem

# Data Storage Layer

The Data Storage Layer contains all the subsystems that manages and holds a repository of all the image and data files saved by or accessed by the application. These images and data files may be requested from the Database Manager Subsystem for Image Processing in the Software Processing Layer.

## Image Repository

### Image Buffer

**Prologue**

The Image Buffer module has the responsibility to maintain the image files stored. The Image Buffer receives the image files from the File Store module and sends the image file to the File Query module. The Image Buffer module will be done as a background process as the system requests for the image files.

**Interfaces**

|  |  |  |  |
| --- | --- | --- | --- |
| Source | Sink | Input Data to sink | Output |
| Image Buffer | File Store | Image File | N/A |
| Image Buffer | File Query | N/A | Image File |

**External Data Dependencies**

N/A

**Internal Data Dependencies**

These background functions will be written in Java

**Pseudo Code**

## File Repository

### File Buffer

**Prologue**

The File Buffer module has the responsibility to maintain all the converted data files that are stored. The File Buffer will receive data files from the File Store module and send data files to the File Query module. The File Buffer module will be a background process as the system requests for the data files.

**Interfaces**

|  |  |  |  |
| --- | --- | --- | --- |
| Source | Sink | Input Data to sink | Output |
| File Buffer | File Store | Data File | N/A |
| File Buffer | File Query | N/A | Data File |

**External Data Dependencies**

N/A

**Internal Data Dependencies**

This background process will be written in Java

**Pseudo Code**

## Database Management

### File Query

**Prologue**

The File Query module has the responsibility to requests specific files when requested by the system. Upon request, the File Query module will query for data files from the File Buffer module or image files from the Image Buffer module.

**Interfaces**

|  |  |  |  |
| --- | --- | --- | --- |
| Source | Sink | Input Data to sink | Output |
| File Query | Image Buffer | N/A | Image File |
| File Query | File Buffer | N/A | Data File |

**External Data Dependencies**

N/A

**Internal Data Dependencies**

This background process will be written in Java

**Pseudo Code**

### File Store

**Prologue**

The File Store module has the responsibility to store specific files when requested by the system. Upon request, the File Store module will store the image files to the Image Buffer module or data files to the File Buffer.

**Interfaces**

|  |  |  |  |
| --- | --- | --- | --- |
| Source | Sink | Input Data to sink | Output |
| File Store | Image Buffer | Image File | N/A |
| File Store | File Buffer | Data File | N/A |

**External Data Dependencies**

N/A

**Internal Data Dependencies**

This background process will be written in Java

**Pseudo Code**

# Hardware Input Layer

This layer is responsible for reading input from Software Output Layer, Sketch layer and Motion layer, packaging all these inputs and sending it to the hardware processing layer. This layer includes the File reader subsystem, Sensor Reader subcomponent, Transfer Data, Camera Processing and Synchronization Input Subsystem. Input from Power Button and Start Button is also read in this layer.

## File Reader Subsystem

## Synchronization Input Subsystem

## Camera Processing

# Hardware Output Layer

All the hardware outputs in the form of light and sound are taken care of in this layer. It includes alarm subsystem, output De-multiplexer and Light subsystem. This layer depends on the output from hardware output driver subsystem of Hardware processing layer.

## Alarm Subsystem

## Output De-multiplexer

# Hardware Processing Layer

The purpose of the Hardware Processing Layer is to analyze and process data as well communicate with the remainder of the hardware layers: Hardware Input Layer, Hardware Output Layer, Sketch layer and the Motion Layer. This layer is the central processing unit of the hardware. Essentially this is the microcontroller in the hardware that controls all of the motions including the Hardware Input Layer, the Hardware Output Layer, the Sketching Layer and the Motion Layer. The sections below provide a detailed description of this layer and its subcomponents – Hardware Input Driver Subsystem, Hardware Output Driver Subsystem, Synchronization Subsystem, Position Processing Subsystem, and the Motion Subsystem.

## Hardware Input Driver Subsystem

\*\*\*Picture of whole Subsystem split into modules\*\*\*\*

### Set Global Variables

#### Description:

##### This module will receive a python list with two elements. The first element of the list will be another list that has all of the instructions that the robot should follow in order to sketch the image. The second element of the list will contain the current position of the Sidewalk Sketcher. This module will set these two element to the global scope of the project instance so that can be accessed through the entire hardware without explicitly passing them as parameters.

#### Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Parameter | Data Returned |
| setListElements | Receive the input from the Hardware Input Layer | None | None |

#### External Data Dependencies:

##### None

#### Internal Data Dependencies:

##### Python list containing 2 elements

#### Pseudo Code:

##### 

## Hardware Output Driver Subsystem

### Send Alerts

#### Description:

##### This module will result raw alerts grouped into a list from the Synchronization Subsystem and it will format all of the alerts into a python class that will have 3 classes each representing a different type of alert. The three alerts will be a stop alert, light alert, and sound alert.

#### Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Parameter | Data Returned |
| sendOutputData | Receive raw list from the Sync. Subsystem and then format it into a class structure for the Output Layer | List | Class Object |

#### External Data Dependencies:

##### None

#### Internal Data Dependencies:

##### Python list containing an class object

#### Pseudo Code:

##### 

## Synchronization Subsystem

### Instruction Processing

#### Description:

##### This module deals with getting the instructions and coordinates from the Hardware Input Driver Subsystem, storing certain variables and distributing information to Sketcher Processing Subsystem and the Position Processing Subsystem.

#### Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Parameter | Data Returned |
| getNextPosition | Extracts the next coordinates from the InstructionList | None | None |
| incCount | Increments the Count variable representing how man instructions have been sketched | None | None |
| syncTime | Ensures the sketching process is performed synchronously by keeping a log of time each instruction gets performed | None | None |

#### External Data Description:

##### None

#### Internal Data Dependencies:

##### InstructionList and the current Coordinates

#### Pseudo Code:

##### 

### Self-Repair

#### Description:

##### In this module the Sidewalk Sketcher attempts to repair itself incase an error has occurred. If the Sidewalk Sketcher is unable to repair itself, it will alert the user with the appropriate alert.

#### Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Parameter | Data Returned |
| selfAlign | The device will attempt to set itself to the desired position | None | None |
| sendRepairInstruction | The device will attempt to reposition the chalk mechanism. | None | String |

#### External Data Description:

##### An external error either in the Sketch or Motion layer has occurred.

#### Internal Data Dependencies:

##### Error messages will trigger this module.

#### Pseudo Code:

##### 

## Sketcher Processing Subsystem

### Send Next Sketch Instruction

#### Description:

##### This module will send the next instruction to the Sketch layer as a string and decrease the count of the overall length of instructions every time an instruction is sent.

#### Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Parameter | Data Returned |
| sendNextInstruction | Send the next instruction | List | String |
| isInstructionSketched | Returns a confirmation that the robot is ready for the next instruction | None | Boolean |

#### External Data Description:

##### None

#### Internal Data Dependencies:

##### InstructionList

#### Pseudo Code:

##### 

### Send Sketch Error

#### Description:

##### This module will send the Synchronization Layer an error if it present while sketching the robot. The errors that may occur include nearing chalk depletion, having a chalk broken, or any unexplained error that the system might encounter.

#### Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Parameter | Data Returned |
| sendSketchError | Send a sketch error to the Sync. Subsystem if an error exists | None | String |

#### External Data Description:

##### An error occurs with the Sketch Layer that triggers this module

#### Internal Data Dependencies:

##### Error message sent as a string

#### Pseudo Code:

##### 

## Position Processing Subsystem

### Send Current Coordinates

#### Description:

##### This module will current coordinates of the Sidewalk Sketcher and the coordinates that the device should move towards next.

#### Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Parameter | Data Returned |
| sendCoordinates | Send the current and next coordinates | None | List |
| isRobotSet | Returns a confirmation that the robot is ready for the next movement | None | Boolean |

#### External Data Description:

##### None

#### Internal Data Dependencies:

##### Coordinates

#### Pseudo Code:

##### 

### Send Motion Error

#### Description:

##### This module will send the Synchronization Layer an error if it present while the robot I moving from point A to point B in the coordinate system of the Sidewalk Sketcher. Errors of this transition include the robot not moving, arriving at a different location than expected or any other unexpected error.

#### Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Parameter | Data Returned |
| sendMotionError | Send a sketch error to the Sync. Subsystem if an error exists | None | String |

#### External Data Description:

##### An error occurs with the Motion Layer that triggers this module

#### Internal Data Dependencies:

##### Error message sent as a string

#### Pseudo Code:

##### 

# Sketch Layer

The purpose of the Sketch Layer is to analyze all of the actions that will be performed by the sketching device that will drag the chalk, pick up the chalk when it’s not drawing, and send the update to the Hardware Processing Layer when the chalk nears depletion This layer will communicate directly to the Hardware Processing Layer and the Hardware Input Layer. With the Hardware Processing Layer, it will commute to let the processor know whether or not the chalk is near depletion along with whether the device should be in its writing state or in its floating state (floating meaning it is picked up because it is passing an area that there is no lines to be drawn) This layer will communicate with the Hardware Input Layer because it will have to send senor information to make sure the chalk is connecting to the ground when it is supposed to be sketching the image.

## Sketcher Synchronization Subsystem

### Detect Error

#### Description:

##### This module deals with monitoring the Sidewalk Sketcher and making sure that it performs the instruction that it’s supposed to perform.

#### Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Parameter | Data Returned |
| sendSketchError | Sends a message if there is an error during the sketch process | None | String |
| sendIsDepleted | Sends a message if the chalk has depleted | None | String |

#### External Data Description:

##### None

#### Internal Data Dependencies:

##### The instruction needed to be perform.

#### Pseudo Code:

##### 

## Depletion Subsystem

### Perform Instruction

#### Description:

##### This module will actually be in charge of performing the instruction that is sent through the Hardware Processing Layer. In this module the system will keep a variable that will keep track of the piece of chalk being used. This module will also control the stepper motor and how many degrees it has turned. These degrees will be used to calculate the remainder of the piece of chalk.

#### Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Parameter | Data Returned |
| sendAction | Sends the required instruction to the stepper motor | NextInstruction | Error message |

#### External Data Description:

##### The system will have to communicate directly with the stepper motor.

#### Internal Data Dependencies:

##### The instruction needed to be perform.

#### Pseudo Code:

##### 

# Motion Layer

The purpose of the Motion Layer is to analyze the mechanical motion that will perform from the Sidewalk Sketcher and to ensure that the device is on the correct path. This layer will communicate directly to the Hardware Processing Layer and the Hardware Input Layer. With the Hardware Processing Layer which will commute to let the processor know where in the robot is at any given position so that the processing unit can determine where to go next and let this layer know the updated information. This layer will communicate with the Hardware Input Layer because it will have to send the positioning data collected from the data and sync this input to know its absolute position relative to the marker devices used for positioning.

## Motion Synchronization Subsystem

### Detect Error

#### Description:

##### This module deals with monitoring the Sidewalk Sketcher and making sure that it performs the instruction that it’s supposed to perform.

#### Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Parameter | Data Returned |
| sendMotionError | Sends a message if there is an error during the movement | None | String |

#### External Data Description:

##### None

#### Internal Data Dependencies:

##### The instruction needed to be perform.

#### Pseudo Code:

##### 

## Motion-Driver Subsystem

### Perform Instruction

#### Description:

##### This module will actually be in charge of performing the instruction that is sent through the Hardware Processing Layer. In this module the system will use the coordinates received to determine where the robot currently is, where it is facing and what direction it should turn to.

#### Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Parameter | Data Returned |
| moveCreate | Sends the required instruction to the iCreate 2 | current, destination | List |

#### External Data Description:

##### The system will have to communicate directly with iCreate 2.

#### Internal Data Dependencies:

##### The instruction needed to be perform.

#### Pseudo Code:

##### 

# Quality Assurance

# Requirements Mapping

This section provides an overview of the key system requirements and how each layer provides its functionality to meet the requirement through a table. Note that the layers present in the table are only layers that complete key requirements. Some layers that are required to run the system are not present in this table.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Requirement Number | Requirement Name | Hardware Input Layer | Hardware Output Layer | Sketch Layer | Motion Layer | UI Layer | Hardware Processing Layer |
| 3.1 | Sketch Image |  |  | x |  |  |  |
| 3.2 | Sidewalk Sketcher Multicolor |  |  |  |  | x |  |
| 3.7 | Sketch Dimensions |  |  |  |  | x |  |
| 3.8 | Chalk Switch |  |  |  |  |  | x |
| 8.1 | Image Loading |  |  |  |  |  | x |
| 8.5 | Push Chalk |  |  |  |  |  | x |
| 8.6 | Finished Image |  | x |  |  |  |  |
| 8.7 | Depletion Sensing | x |  |  |  |  |  |
| 8.8 | Chalk Reload |  | x |  |  |  |  |
| 8.9 | Positioning |  |  |  | x |  |  |

# Acceptance Criteria

# Appendix