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: 9 February 2015 @ 10:56:00 PM

Table of Contents

[1. Introduction 4](#_Toc411021423)

[2. Architecture Overview 4](#_Toc411021424)

[3. System Hardware Description 4](#_Toc411021425)

[3.1 Raspberry Pi Model B+ 4](#_Toc411021426)

[3.2 2GB Micro SD Card 5](#_Toc411021427)

[3.3 Raspberry Pi Camera 6](#_Toc411021428)

[3.4 USB-Serial Cable 6](#_Toc411021429)

[3.5 Roomba® iCreate 2 6](#_Toc411021430)

[3.6 Stepper Motor 6](#_Toc411021431)

[3.7 LED’s 6](#_Toc411021432)

[3.8 Broom Sticks 6](#_Toc411021433)

[3.9 Swim Noodle 6](#_Toc411021434)

[3.10 Speakers 6](#_Toc411021435)

[3.11 Plywood 6](#_Toc411021436)

[4. Software Input Layer 6](#_Toc411021437)

[4.1 Image Reader 6](#_Toc411021438)

[4.2 Converter System 6](#_Toc411021439)

[4.3 Transfer Data 6](#_Toc411021440)

[5. Software Output Layer 7](#_Toc411021441)

[5.1 Data Packager 7](#_Toc411021442)

[5.2 File Generator 7](#_Toc411021443)

[5.3 File transfer 7](#_Toc411021444)

[6. Software Processing Layer 8](#_Toc411021445)

[6.1 Image Processing 8](#_Toc411021446)

[6.2 Information Processing 8](#_Toc411021447)

[7. User Interface Layer 9](#_Toc411021448)

[7.1 File Browser Subsystem 9](#_Toc411021449)

[7.2 Cropping Subsystem 9](#_Toc411021450)

[7.3 Resize Subsystem 9](#_Toc411021451)

[7.4 Color Selector Subsystem 9](#_Toc411021452)

[8. Data Storage Layer 9](#_Toc411021453)

[8.1 Image Repository 9](#_Toc411021454)

[8.2 File Repository 9](#_Toc411021455)

[8.3 Database Management 9](#_Toc411021456)

[9. Hardware Input Layer 9](#_Toc411021457)

[9.1 File Reader Subsystem 9](#_Toc411021458)

[9.2 Synchronization Input Subsystem 10](#_Toc411021459)

[9.3 Camera Processing 10](#_Toc411021460)

[10. Hardware Output Layer 11](#_Toc411021461)

[10.1 Alarm Subsystem 11](#_Toc411021462)

[10.2 Output De-multiplexer 11](#_Toc411021463)

[11. Hardware Processing Layer 11](#_Toc411021464)

[11.1 Hardware Input Driver Subsystem 11](#_Toc411021465)

[11.2 Hardware Output Driver Subsystem 11](#_Toc411021466)

[11.3 Synchronization Subsystem 11](#_Toc411021467)

[11.4 Sketcher Processing Subsystem 11](#_Toc411021468)

[11.5 Position Processing Subsystem 11](#_Toc411021469)

[12. Sketch Layer 11](#_Toc411021470)

[12.1 Sketcher Synchronization Subsystem 12](#_Toc411021471)

[12.2 Depletion Subsystem 12](#_Toc411021472)

[13. Motion Layer 13](#_Toc411021473)

[13.1 Motion Synchronization Subsystem 13](#_Toc411021474)

[13.2 Motion-Driver Subsystem 13](#_Toc411021475)

[14. Quality Assurance 14](#_Toc411021476)

[15. Requirements Mapping 14](#_Toc411021477)

[16. Acceptance Criteria 15](#_Toc411021478)

[17. Appendix 15](#_Toc411021479)

# 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

This section describes the various design principles that were used in the design of the system’s architecture. This section elaborates on the architectural vision of Team Sketchers, the guiding principles that serve as the foundation for the system architecture, assumptions as well as tradeoffs associated with the architecture design of the Sidewalk Sketcher.

## Architecture Description

The architecture design of the Sidewalk Sketcher is based on the principle of modularity. The project as a whole can be divided into simply two components: hardware and software. However, due to the various areas of work that this project will encounter, Team Sketchers have decided to make modularity a big principle. The modularly principle consists of the following: each layer will have a synchronization subsystem that in effect will be a controller for that particular layer. This subsystem will be in charge of distribution of data among the other subsystems along with being the primary interface in interacting with other layers. As a result, we have several layers all interconnected to communicate with one another to create a traceable process.

The architecture consists of ten layers that are in essence divided between hardware and software.

Software: Software Input, Software Output, Software Processing, UI, and Data Storage.   
Hardware: Hardware Input, Hardware Output, Hardware Processing, Sketch, and Motion.

Each of these layers consists of several subsystems that will interface with one another and interface with other layers. The layers of the Sidewalk Sketcher can better be observed by following the flowchart, yet the basic outline is as follows:

The user will interface with the UI (UI Layer). This layer will attain its input (Software Input Layer) and then perform modifications to the image (Software Processing Layer). After this is completed, the user will have the option to save the project (Data Storage Layer). The user will then decide to transfer the desired image to the hardware (Software Output Layer) so the system will create the vector instructions and send it to the hardware via USB cable (Hardware Input Layer). The Sidewalk Sketcher will then be placed in the desired position to start the image and the markers will be set to their designated places and the hardware will begin execution when instructed. During this execution (Hardware Processing Layer), the Sidewalk Sketcher will keep track of the chalk depletion (Sketch Layer) along with its current position (Motion Layer). In case the Sidewalk Sketcher detects the battery is running low or reaching chalk depletion, the device will alert the user (Hardware Output Layer). Once the user has fixed this problem, the Sidewalk Sketcher will continue to sketch the image until the process is completed.

The input layers will be in charge of translating the information received into data that particular part of the system can use. From the software side, the software will translate the image file into an image that will fit the device’s requirements. From the hardware side, the hardware will parse the instructions received and store the data locally to get the sketcher ready for execution. The processing layers will be the units where the actual processing will be performed, analogous to a computer’s CPU. The output layers will be in charge of sending the right data to the appropriate places such as the user or the hardware.

## Module Decomposition

## Module Functional Descriptions- User Interface Layer

## Module Functional Descriptions- Software Input Layer

## Module Functional Descriptions- Software Processing Layer

## Module Functional Descriptions- Software Output Layer

## Module Functional Descriptions- Hardware Input Layer

## Module Functional Descriptions- Hardware Processing Layer

## Module Functional Descriptions- Hardware Output Layer

## Module Functional Descriptions- Motion Layer

## Module Functional Descriptions- Sketch Layer

## Module Functional Descriptions

# 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. It’s 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 |
| Class | 4 |
| Transfer Rate | 4Mbps |

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

## Raspberry Pi Camera

## USB-Serial Cable

## Roomba® iCreate 2

## Stepper Motor

## LED’s

## Broom Sticks

## Swim Noodle

## Speakers

## Plywood

# 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 Subsystem

### Packer Module

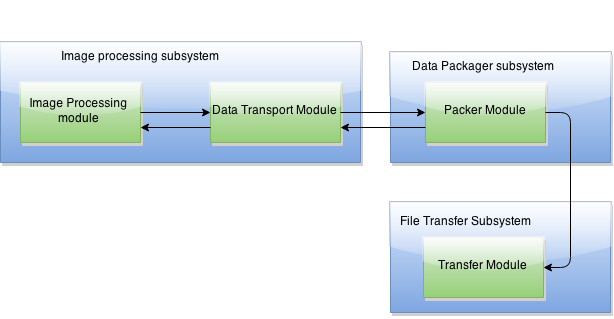


Figure 5-1 Packer Module

**Description**

Packer module receives the processed image data from Data Transport module and sends the data to file transfer module.

**Interfaces**

|  |  |  |  |
| --- | --- | --- | --- |
| **Source** | **Sink** | **Input to Sink** | **Return from Sink** |
| Data Transport | Packer | ImageObject | N/A |
| Packer | File Transfer | ImageObject | N/A |

Table 5-1 Packer Interface

**External Data Dependencies**

N/A

**Internal Data Dependencies**

Receives Image data from data transport module.

**Pseudo Code**

**public** **class** Packer {

**int** Postdata;

**public** Packer(**int** data) {

Postdata = data;

}

public **int** transfer() {

**return** Postdata;

}

}

## File Generator

### Generator Module

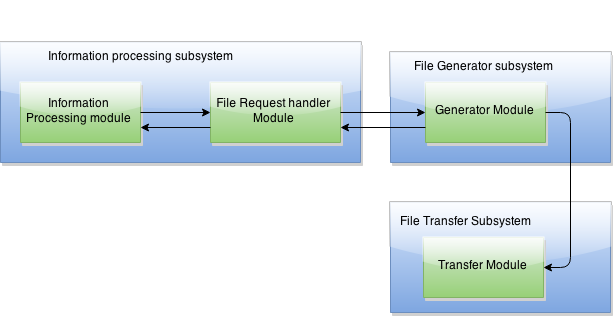
****

Figure 5-2 Generator Module

**Description**

File Generator module is responsible for retrieving required file from File request handler module and transfer that file to File Transfer module.

**Interfaces**

|  |  |  |  |
| --- | --- | --- | --- |
| **Source** | **Sink** | **Input to Sink** | **Return from Sink** |
| File Generator | File Request | File request | File data |
| File Generator | File Transfer | File data | N/A |

Table 5-3 Generator Interface

**External Data Dependencies**

N/A

**Internal Data Dependencies**

File Data from File request handler module

**Pseudo Code**

Path file = ...;

**try**{ (InputStream in = Files.newInputStream(file);

BufferedReader reader = **new** BufferedReader(**new** InputStreamReader(in)))

String line = **null**;

**while** ((line = reader.readLine()) != **null**) {

System.***out***.println(line);

}

} **catch** (IOException x) {

System.err.println(x);

}

## File transfer

### Transfer Module

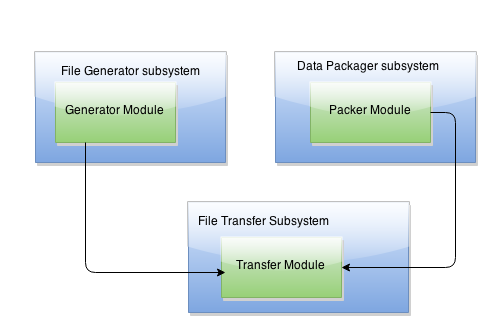


Figure 5-3 Transfer Module

**Description**

Transfer Module receives data from Packer module as well as from Generator module which eventually is the Hardware Input.

**Interfaces**

|  |  |  |  |
| --- | --- | --- | --- |
| **Source** | **Sink** | **Input to Sink** | **Return from Sink** |
| Packer | Transfer | ImageObject | N/A |
| Generator | Transfer | File data | N/A |

Table 5-3 Transfer Interfaces

**External Data Dependencies**

N/A

**Internal Data Dependencies**

Image Data from packer module and File data from Generator module.

**Pseudo Code**

**public** **class** FileTransfer {

**int** Postdata;

**public** FileTransfer(**int** data) {

Postdata = data;

}

**public** **int** transfer() {

**return** Postdata;

}

}

# Software Processing Layer

The purpose of the Software Processing Layer is to handle and process the data and the requests that have been received from User Interface Layer and Software Input Layer. This layer generates database queries and requests for communicating with the Data storage layer as well as translate information received from the Data storage layer making a link between the two layers bi- directional. This layer also handles requests from software output layer and provides response with requested data.

## Image Processing

### Data Receiver Module

### Macintosh HD:Users:pranilmaharjan:Downloads:Untitled Diagram-3.png

Figure 6-1 Data Receiver Module

**Description**

The data receiver module is responsible for receiving data from Transfer data module and passes the data to Image processing module.

**Interfaces**

|  |  |  |  |
| --- | --- | --- | --- |
| **Source** | **Sink** | **Input to Sink** | **Return from Sink** |
| Transfer data | Data Receiver | Image Object | N/A |
| Data Receiver | Image Processing | CommandRequest Object | N/A |

Table 6-1 Data Receiver Interfaces

**External Data Dependencies**

N/A

**Internal Data Dependencies**

Image object from Transfer data Module received from User Interface

**Pseudo Code**

**public** **class** ReadingImage {

**public** **void** getImage() {

**try** {

BufferedImage image =

ImageIO.read(getClass().getResource("Sunset.jpg")) ;

// Do something with the image.

} **catch** (IOException e) {

e.printStackTrace();

}

}

**public** **static** **void** main(String[] args) {

ReadingImage ri = **new** ReadingImage();

ri.getImage();

}

### Request Handler Module

### Macintosh HD:Users:pranilmaharjan:Downloads:Untitled Diagram-4.png

Figure 6-2 Request Handler Module

**Description**

The Request Handler accepts input from the UI subsystem and responds to different events generated by user actions on the User Interface.

**Interfaces**

|  |  |  |  |
| --- | --- | --- | --- |
| **Source** | **Sink** | **Input to Sink** | **Return from Sink** |
| Data Transfer | Request Handler | Event ID | N/A |
| Request Handler | Image Processing | InputObject | N/A |

Table 6-2 Request Handler Interfaces

**External Data Dependencies**

N/A

**Internal Data Dependencies**

Event ID from Data Transfer Module action performed by User.

**Pseudo Code**

**private** **class** ButtonHandler **implements** ActionListener

{

// handle button event

**public** **void** actionPerformed( ActionEvent event )

{

// what function needs to be performed

}

}

### Data Transport Module

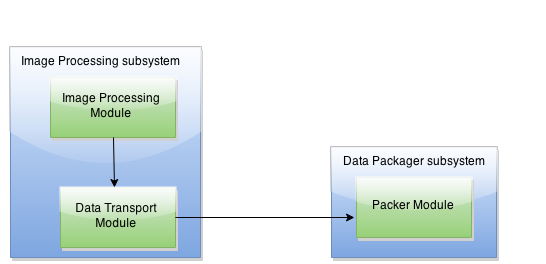
****

Figure 6-3 Data Transport Module

**Description**

Data Transport module is responsible for sending the processed data to the Output layer. Data Transport module receives the processed data from Image processing module and pass the data to Packer module.

**Interfaces**

|  |  |  |  |
| --- | --- | --- | --- |
| **Source** | **Sink** | **Input to Sink** | **Return from Sink** |
| Image Processing | Data Transport | postscript data | N/A |
| Data Transport | Packer | postscript data | N/A |

Table 6-3 Data Transport Interfaces

**External Data Dependencies**

N/A

**Internal Data Dependencies**

PostScript data from Image processing Module

**Pseudo Code**

**class** DataTransport {

**int** Postdata;

public **void** receive(**int** data) {

Postdata = data;

}

**int** transfer() {

**return** Postdata;

}

}

### Image Processing Module

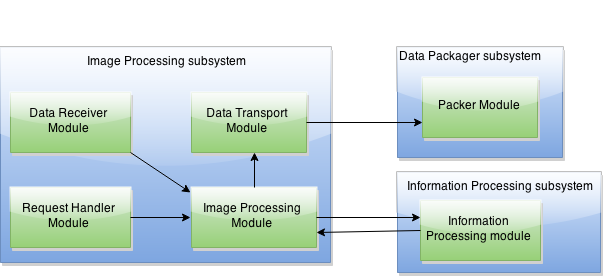
****

Figure 6-4 Image Processing Module

**Description**

The Image Processing module receives the data from the Data receiver and also handles request input processed from Request Handler. This module also communicates with information processing subsystem for generating required information while processing the image and finally transfers the processed data to software output layer via data transport module.

**Interfaces**

|  |  |  |  |
| --- | --- | --- | --- |
| **Source** | **Sink** | **Input to Sink** | **Return from Sink** |
| Request Handler | Image Processing | Event Id | N/A |
| Data Receiver | Image Processing | ImageObject | N/A |
| Image Processing | Data Transport | PostScript | N/A |
| Image Processing | Information Processing | Event Id/ ImageObject | FileObject |

Table 6-4 Image Processing Interface

**External Data Dependencies**

N/A

**Internal Data Dependencies**

Event Id from Request Handler, Image Object from Data Receiver and FileObject from Information Processing module.

**Pseudo Code**

receiveEvent(Action Event)

{

//Receives event from Request handler

}

receiverData(Image data)

{

//Receives image data from Data receiver

}

requestFile(**int** file)

{

//Receives needed file from Information Processing module

}

sendData(**int** data)

{

//Processed data is transferred to data transport module

}

## Information Processing

### Information Processing Module

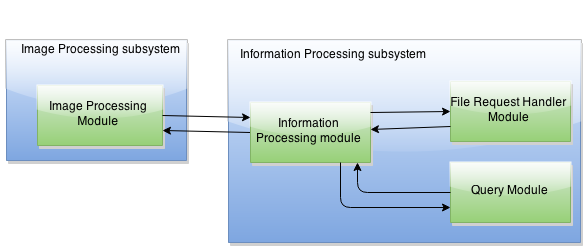
****

Figure 6-5 Information Processing Module

**Description**

Information Processing Module is responsible for receiving the data as well as requests from the Image processing module. It also handles the file request from the file request handler and also communicates with query module for accessing data from any files.

**Interfaces**

|  |  |  |  |
| --- | --- | --- | --- |
| **Source** | **Sink** | **Input to Sink** | **Return from Sink** |
| Image Processing | Information Processing | ImageObjects/ EventRequests | Image Data / Action performed |
| File request handler | Information Processing | File Request | File Data |
| Information Processing | Query module | Queries | Files/images |

Table 6-5 Information Processing Interfaces

**External Data Dependencies**

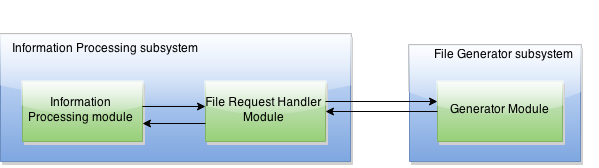
N/A

**Internal Data Dependencies**

Image Objects from Image Processing Module and file data from query module.

**Pseudo Code**

### File Request Handle Module

****

**Description**

File Request Handler receives file requests from File generator and passes to Information Processing module, which analyzes the requests and provides the appropriate response.

**Interfaces**

|  |  |  |  |
| --- | --- | --- | --- |
| **Source** | **Sink** | **Input to Sink** | **Return from Sink** |
| File Generator | File Request | File Request | File Data |
| Information Processing | File Request | File Data | N/A |

Table 6-6 File Request Interfaces

**External Data Dependencies**

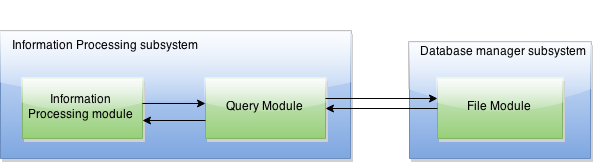
N/A

**Internal Data Dependencies**

File Data from Information processing module.

**Pseudo Code**

### Query Module

****

**Description**

Query module is responsible for accessing the database after receiving requests from Information processing module.

**Interfaces**

|  |  |  |  |
| --- | --- | --- | --- |
| **Source** | **Sink** | **Input to Sink** | **Return from Sink** |
| Information Processing | Query | ImageObject request/ fileObject request | Image or files |
| File query | Query | Images or files | N/A |

**External Data Dependencies**

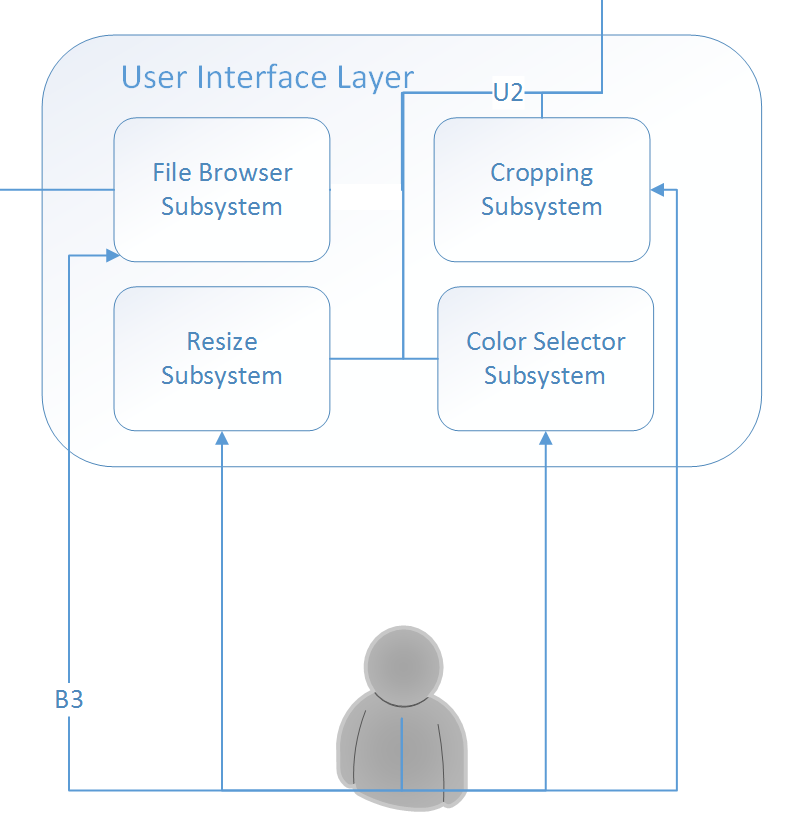
N/A

**Internal Data Dependencies**

**Pseudo Code**

# User Interface Layer

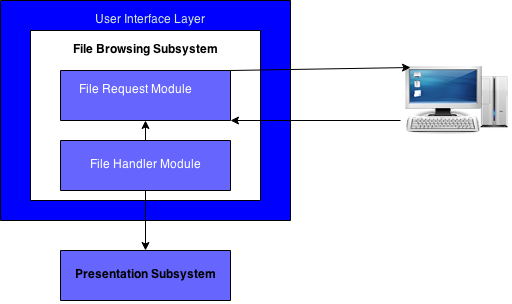
The User Interface Layer provides user a medium to create processed image required for our robot Sidewalk sketcher. In this layer the file browser subsystem allows user to select an image from the user computer. The image is then displayed through presentation layer. Next, the layer provides user options 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

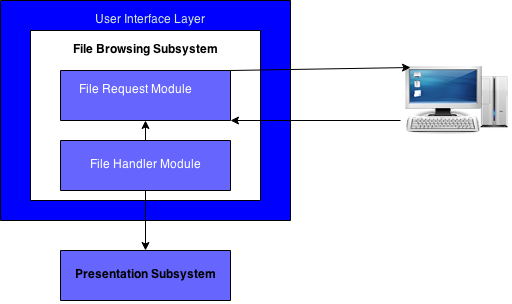
}

**SET\_IMAGE{**

Set an image in an array

}

### File Handler Module



**Prologue**

The File handler module gets an image file from file request modules and then sends it to presentation subsystem to display.

**Interface**

|  |  |  |  |
| --- | --- | --- | --- |
| **Source** | **Sink** | **Input Data to sink** | **Output** |
| File Request Module | File Handler Module | N/A | Image File |
| File Handler Module | Presentation Subsystem | Image File | N/A |

**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 {**

Gets image stored in an array of file request module.

}

**SET\_IMAGE {**

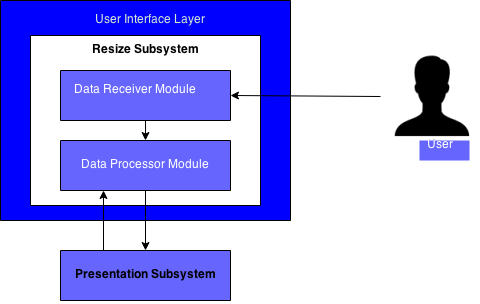
Sends image to presentation layer

}

## Resize Subsystem

The resize subsystem will allow user to input new dimension for selected image. The new dimension is then used by its modules to produce a new resized image.

### Data Receiver



**Prologue**

The data receiver module receives data from the user and then the data is then sent to data processor module.

**Interface**

|  |  |  |  |
| --- | --- | --- | --- |
| **Source** | **Sink** | **Input Data to sink** | **Output** |
| User | Data Receiver Module | New dimensions for the image | Dimension |
| Data Receiver Module | Data Processor Module | Dimension | N/A |

**External Dependencies**

The module requires user to enter new dimension for the image.

**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\_DATA {**

Gets data will get the dimension from the user

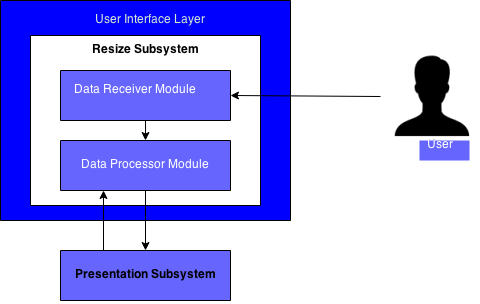
}

**SET\_DATA {**

Set data will set the dimension in variables.

}

### Data Processor Module



**Prologue**

The data processor module receives data from the data receiver module. This module then request image from presentation subsystem to resize image using the data. Once the resize is complete it returns the new image to presentation Subsystem.

**Interface**

|  |  |  |  |
| --- | --- | --- | --- |
| **Source** | **Sink** | **Input Data to sink** | **Output** |
| Data Receiver Module | Data Processor Module | New dimensions |  |
| Presentation subsystem | Data Processor Module | Image | N/A |
| Data processor Module | Presentation Subsystem | Image and dimensions | New dimensional image |

**External Dependencies**

N/A

**Internal Dependencies.**

The action for the module depends on data receiver module which 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

**SET\_DATA {**

Receives data from data receiver module.

}

**GET\_IMAGE {**

Get image from database of presentation layer

}

**PROCESS\_IMAGE {**

Process image with given data

}

**SET\_IMAGE {**

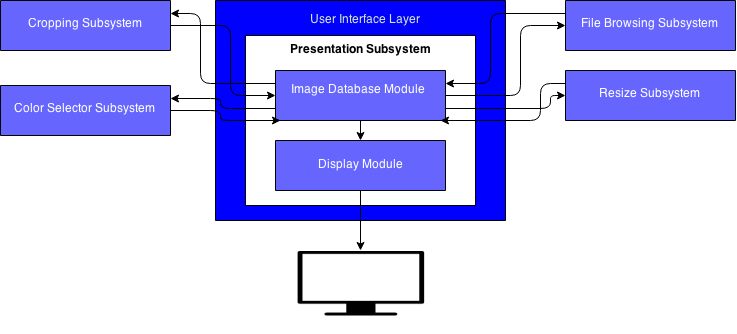
Set new image back to database of presentation layer

}

## Presentation Subsystem

The presentation subsystem communicates with all the subsystem present in user interface layer. It creates a medium for user to interact with the sidewalk sketcher user interface.

### Image Database Module



**Prologue**

The image database module works as a storage where image that was loaded and modified are stored. This module can store one image at a time and if any changes are made in the image the old image will be replaced.

**Interface**

|  |  |  |  |
| --- | --- | --- | --- |
| **Source** | **Sink** | **Input Data to sink** | **Output** |
| File Browsing Subsystem  Resizing Subsystem  Cropping Subsystem  Color Selector Subsystem | Image Database Module | Image File | N/A |
| Image Database Module | Display Module | Image File | Image |

**External Dependencies:**

N/A

**Internal Dependencies.**

The action for the module depends on all the subsystems of user interface layer, which 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

**SET\_IMAGE {**

Put image in an array of image received from other subsystems

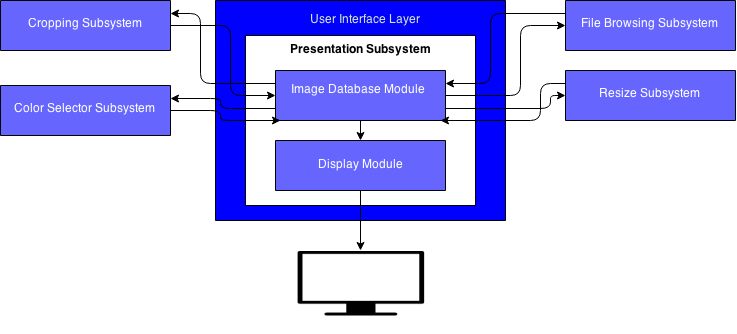
}

**GET\_IMAGE {**

Request image from other subsystems according to the action performed.

}

### Display Module



**Prologue**

The display module provides a visual aid to the user. It displays the GUI used for sidewalk sketcher interface as well as show the changes in image made by the user.

**Interface**

|  |  |  |  |
| --- | --- | --- | --- |
| **Source** | **Sink** | **Input Data to sink** | **Output** |
| Image Database Module | Display Module | Image File | Image |
| Display Module | Windows Operating System | GUI File | Sidewalk Sketcher Interface |

**External Dependencies:**

The display module depends on the user turning on the display in order to display see the output of the system as well as action performed.

**Internal Dependencies.**

The action for the module depends on windows operating system which 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

**Run\_GUI {**

Create a user interface for sidewalk sketcher.

}

**GET\_IMAGE {**

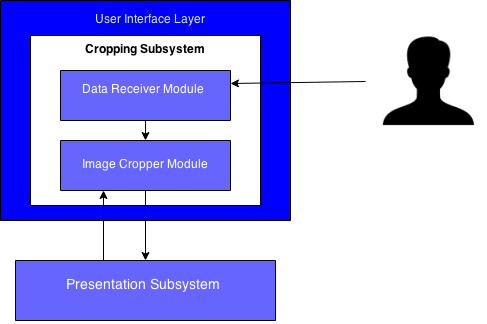
Request image from image database module to display the action performed.

}

## Cropping Subsystem

Cropping subsystem allows user to crop an image to desirable size. It gives user flexibility to choose which part of an image to be cropped by using its modules.

### Data Receiver Module



**Prologue**

The data receiver module receives data from the user and then the data is then sent to data processor module.

**Interface**

|  |  |  |  |
| --- | --- | --- | --- |
| **Source** | **Sink** | **Input Data to sink** | **Output** |
| User | Data Receiver Module | X and Y position of four corners from cropping image | N/A |
| Data Receiver Module | Data Processor Module | X and Y position of four corners from cropping image | N/A |

**External Dependencies**

The module requires user to select the desired portion of image to be cropped.

**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\_DATA {**

Gets data of desired portion of image.

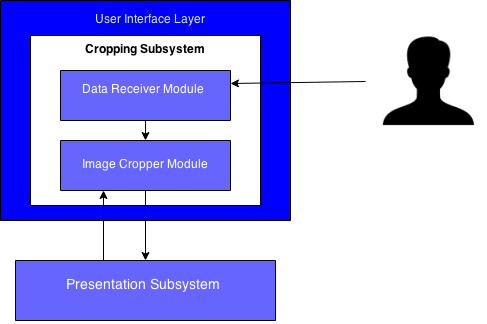
}

**SET\_DATA {**

Set data will set the desired portion of image to be cropped.

}

### Image Cropper Module



**Prologue**

The image cropper module receives data of desired image size from the data receiver module. This module then request image from presentation subsystem to crop an image using the data. Once the cropping is complete it returns the new image to presentation Subsystem.

**Interface**

|  |  |  |  |
| --- | --- | --- | --- |
| **Source** | **Sink** | **Input Data to sink** | **Output** |
| Data Receiver Module | Data Cropper Module | Data of new size image |  |
| Presentation subsystem | Data Processor Module | Image | N/A |
| Data Cropper Module | Presentation Subsystem | Image and new size of image | New image |

**External Dependencies**

N/A

**Internal Dependencies.**

The action for the module depends on data receiver module, which 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

**SET\_DATA {**

Receives data from data receiver module.

}

**GET\_IMAGE {**

Get image from database of presentation layer

}

**CROP\_IMAGE {**

Crop image with given size

}

**SET\_IMAGE {**

Set new image back to database of presentation layer

}

## Color Selector Subsystem

Color Selector subsystem allows user to choose two colors that will be used while processing selected image into two color image. This subsystem will provide user options to choose different colors using its module.

## 

## 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\*\*\*\*

### Receive Raw Data

#### Description:

##### This module will receive raw data from the hardware input layer. This data will contain a python list structure that will contain each line of the file loaded from the user interface. Each element of the list will contain a line in the file which will represent instructions for the Sidewalk Sketcher to follow. This module will also include the current position of the robot. In this module the system will store the set instructions and location and organize it in a way that can be passed to the Synchronization subsystem.

#### Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Parameter | Data Returned |
| getRawInput | Receive the input from the Hardware Input Layer | None | Python list of 2 elements, one with another list of instructions and the other of position coordinates |

#### External Data Dependencies:

##### None

#### Internal Data Dependencies:

##### Python list containing 2 elements

#### Pseudo Code:

##### 

## Hardware Output Driver Subsystem

## Synchronization Subsystem

## Sketcher Processing Subsystem

## Position Processing Subsystem

# 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

## Depletion Subsystem

# 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

## Motion-Driver Subsystem

# 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