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: 27 January 2015 @ 4:08:00 PM

Table of Contents

1. Introduction 5

1.1 Purpose and Use 5

1.2 Project Description 5

1.3 Key Requirements 5

2. Meta Architecture 6

2.1 Architectural Vision 6

2.2 Guiding Principles 7

2.3 Design Assumptions 8

2.4 Design Tradeoffs 8

3. Inter-Subsystem Data Flow 10

3.1 Data Flow Diagram 10

3.2 Data flow definitions 11

3.3 Producer- Consumer Relationships 15

4. Layer Definitions 19

4.1 Software Input Layer 19

4.2 Software Output Layer 19

4.3 Software Processing Layer 19

4.4 User Interface Layer 19

4.5 Data Storage Layer 19

4.6 Hardware Input Layer 20

4.7 Hardware Output Layer 20

4.8 Hardware Processing Layer 20

4.9 Sketch Layer 20

4.10 Motion Layer 20

5. Software Input Layer 21

5.1 Image Reader 21

5.2 Converter System 22

5.3 Transfer Data 22

6. Software Output Layer 23

6.1 Data Packager 23

6.2 File Generator 24

6.3 File transfer 24

7. Software Processing Layer 25

7.1 Image Processing 25

7.2 Information Processing 26

8. User Interface Layer 27

8.1 File Browser Subsystem 27

8.2 Cropping Subsystem 28

8.3 Resize Subsystem 28

8.4 Color Selector Subsystem 29

9. Data Storage Layer 30

9.1 Image Repository 30

9.2 File Repository 31

9.3 Database Management 31

10. Hardware Input Layer 32

10.1 File Reader Subsystem 32

10.2 Sensor Reader Subcomponent 33

10.3 Synchronization Input Subsystem 33

10.4 Camera Processing 34

10.5 Transfer Data 34

11. Hardware Output Layer 35

11.1 Alarm Subsystem 35

11.2 Light Subsystem 36

11.3 Output De-multiplexer 36

12. Hardware Processing Layer 37

12.1 Hardware Input Driver Subsystem 37

12.2 Hardware Output Driver Subsystem 38

12.3 Synchronization Subsystem 38

12.4 Sketcher Processing Subsystem 39

12.5 Position Processing Subsystem 39

13. Sketch Layer 40

13.1 Sketcher Synchronization Subsystem 40

13.2 Depletion Subsystem 41

13.3 Piston Subsystem 42

14. Motion Layer 43

14.1 Motion Synchronization Subsystem 43

14.2 Motion-Driver Subsystem 44

14.3 Position Subsystem 44

15. Requirements Mapping 45

16. Operating System Dependencies Section 46

16.1 Software Input Layer 46

16.2 Software Output Layer 46

16.3 Software Processing Layer 46

16.4 Hardware Input Layer 46

16.5 Hardware Output Layer 46

16.6 Sketch Layer 46

16.7 Motion Layer 47

16.8 Hardware Processing Layer 47

17. Testing Considerations Section 48

17.1 Overall Considerations 48

17.2 Software Input Layer 48

17.3 Software Output Layer 48

17.4 Software Processing Layer 49

17.5 User Interface Layer 49

17.6 Data Storage Layer 49

17.7 Hardware Input Layer 49

17.8 Hardware Output Layer 50

17.9 Hardware Processing Layer 50

17.10 Sketcher Layer 50

17.11 Motion Layer 50

# Introduction

## Purpose and Use

This document describes the architectural overview of the Sidewalk Sketcher and provides a detailed explanation on the system’s meta-architecture as well as multiple layers and subsystem used. The document will also provide set of guiding principles upon which the system is developed based on and also highlights on product concept, scope, and key requirements necessary to design the architecture. Ultimately, the ADS document will describe the operating system dependencies and testing considerations needed to achieve customer and client satisfaction.

## Project Description

The Sidewalk Sketcher has two sets of modules, which runs independent of each other. One module is carried out in user computer using Windows 7 or higher operating system and the other module is carried out on Sidewalk Sketchers. The user is required to convert their desired image into two-color image using Sidewalk Sketcher user interface. Next, the processed image is then transferred into the Sidewalk Sketcher system via USB cord. The processed image will be stored into Sidewalk Sketcher system. The system will provide user options to load file, crop image and input the size of image to be sketch. Beside these features, the Sidewalk Sketcher System will also provide an option to choose two colors that will be used and show the final image using those two colors. Finally Sidewalk Sketcher will start sketching image on the designated area.

## Key Requirements

|  |  |  |
| --- | --- | --- |
| Requirement No | Requirement Name | Description |
| 3.1 | Sketch Image | The system should be able to sketch an image with chalk. |
| 3.2 | Sidewalk Sketcher Multicolor | The system should be able to sketch an image with a least two different colors. |
| 3.7 | Sketch Dimensions | The system should be able to print images with a maximum size of 8 by 10 feet and minimum of 2 by 2 feet. |
| 3.8 | Chalk Switch | The Sidewalk Sketcher will have two pieces chalk while sketching. During this process, the device will sketch all parts of the image that require the first chalk color. After the first chalk color is completed, the Sidewalk Sketcher will switch chalk colors by itself and continue the sketching process. |
| 8.1 | Image Loading | The Sidewalk Sketcher should be able to load the image into the Sidewalk Sketcher via USB cord. |
| 8.5 | Push chalk | The Sidewalk Sketcher will have the ability to continually push the chalk while the image is sketching. |
| 8.6 | Finished Image | The Sidewalk Sketcher will alert the user when the image is finished sketching. |

# Meta Architecture

This section describes the various design principles that were used in the design of the system’s architecture. The Meta Architecture 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.

## Architectural Vision

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 that will come in handy while debugging.

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 might want to save the image locally so he will have the option to save the project (Data Storage Layer). The user will then decide to ship 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 it is running low on battery 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.

With this flow in mind there come the main functions of the layers. The input layers will be in charge of translating the information received into information 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.

## Guiding Principles

Before architecture design can be started, it is important to define the guiding principles upon which the Sidewalk Sketcher will be built. This allows the team to have a better understanding of how the system should be designed. The guiding principles for the Sidewalk Sketcher –modularity, reliability, user friendliness, and safety– are described in this subsection.

### Modularity

The Sidewalk Sketcher is specifically designed with modularity as its prime priority. The system is designed to have pseudo-independent components, in the way that many components could be performed without the assistance of other. Yet this is pseudo-independent in the way that each module, is dependent on each other’s interaction. This principle may appear invisible or transparent to the user in that they will not notice the main bulk of it, yet they will have to go through a certain process to sketch an image which is where the modules are essentially formed.

### Reliability

The Sidewalk Sketcher should be a reliable sketching device that will be able to adapt to the image the user wished to sketch. The system should display the image in the software that eventually will sketch with the hardware.

### User Friendliness

The Sidewalk Sketcher will have an intuitive user interface that will set the look and feel for the application. Once the user has installed the software unit along with the drivers for the hardware, the user should easily be able to determine how to use the software. Of course, there will be a manual for both the software and hardware components. Nevertheless, Team Sketcher wishes the user to get a feel for the software without those components.

### Safety

The Sidewalk Sketcher will be safe so that the hardware component shall not harm the user and the software unit shall not compromise any software that may be already installed in the user’s machine.

## Design Assumptions

In order to develop the Sidewalk Sketcher, the team has made assumptions that each user will follow specific steps to achieve the desired image sketch. The list of assumptions is described in this subsection.

### Markers

The Sidewalk Sketcher will have a set of three markers that will aid in determining the position of the device at all times. A crucial assumption for the project is that the markers are places in the appropriate location prior to the robot beginning sketching the image.

### Image

The user will select an image that can be sketched manually for the Sidewalk Sketcher to sketch. The more complicated the image is the lower the quality of the sketch will be. To maximize the user’s experience with this device, the user should select a simple image to be sketch such that if a person were to manually sketch the image, the image will result in good quality given the dimensions of the image.

### Battery Life

The user is highly advised to have the Sidewalk Sketcher fully charged prior to beginning a new sketch. If the sketcher is not fully charged, the device will have to be charged during operation which may result in a lower quality image.

### Chalk

The user is highly advised to have a full length piece of chalk installed in the Sidewalk Sketcher prior to beginning a new sketch. If the sketcher does not have a new piece of chalk installed, the user operation will be halted when the chalk nears depletion and will not resume until the user reloads another piece of chalk. This halt could result in a lower quality image as opposed to the image that would have been created if the sketcher had a full piece of chalk beforehand.

### User Presence

The user is highly advised to be present in the sketching process in case there are any errors that arise that could compromise the image sketch.

## Design Tradeoffs

In order to develop the Sidewalk Sketcher, the team had to consider a few tradeoffs when designing the Sidewalk Sketcher. The list of tradeoffs is described in this subsection.

### Many Layers vs. Few Layers

Based on the concept of modularity there was a debate of whether to have multiple layers of abstraction or whether we should encapsulate the abstraction in fewer layers. The Sidewalk Sketcher design uses the wider spread of abstraction.

### Vector Plotting vs. Bit-Map Plotting

There are two ways to sketch an image, either by drawing lines (vector plotting) or by drawing a bunch of dots (bit map plotting). The way in which we plot the image is heavily based on how we implement the rest of the Sidewalk Sketcher design. The Sidewalk Sketcher design uses the vector plotting approach.

### Camera vs. Sonar Sensors

There are two ways to determine distance of objects relative to one another, either by sound or light. With each of these choices comes several ways of achieving the distance calculation. The Sidewalk Sketcher design uses the camera approach in determining position.

### IRobot Create® vs Scratch

There are two ways to begin our hardware design, either by building off another component (in this case the IRobot Create®) or completely designing a new component from scratch. Due to the time constraint and the learning curve on other aspects of the system, Team Sketchers has decided to go with the IRobot Create® approach.

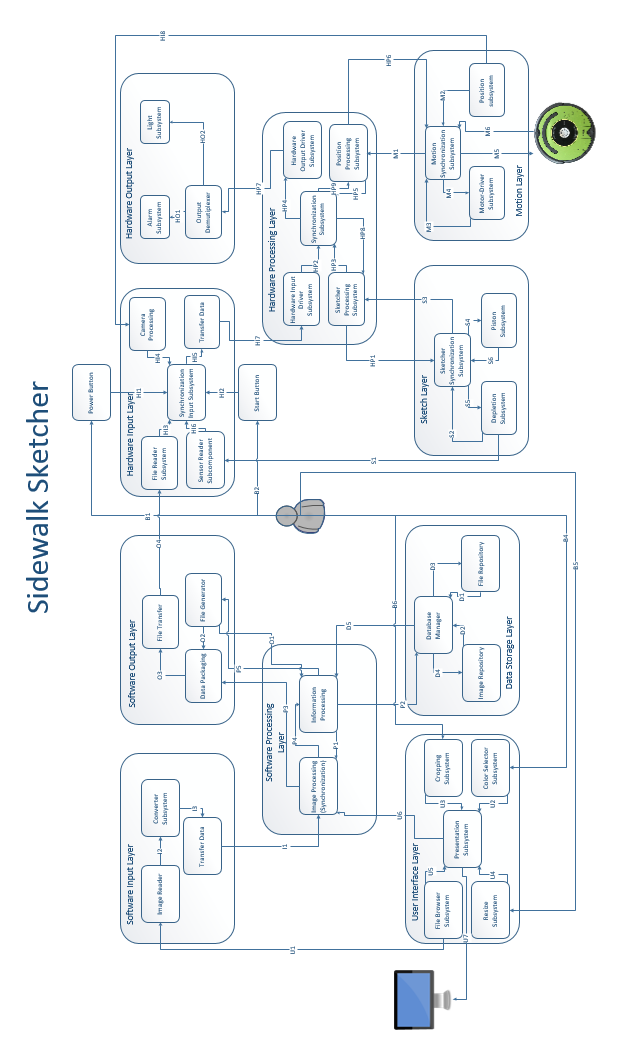
# Inter-Subsystem Data Flow

This section illustrates how the various subsystems interact within Sidewalk Sketcher. Each data flow is marked, which is described in the following subsections. The subsections provide a high level overview of the system and the data flows between each of the layers and subsystems describe how individual data elements are used. The relationship between data producers and their respective consumers is also shown below.

## Data Flow Diagram

The dataflow diagram is divided into two sections: Software section and Hardware section. Overall dataflow diagram consists of ten layers. Starting from the User Interface in software section, data flows to the software input layer passing it to the software processing layer. Software processing layer constantly interacts with the data storage layer to store data or retrieve data. It also interacts with the software output layer to provide needed output.

The Software Output acts as the Hardware input. In hardware input, along with the image data received from the software output, the camera positioning input as well as the sensor reading is integrated and transferred into the hardware processing layer. Hardware processing layer constantly interacts with the sketch layer, motion layer and hardware output layer so that proper processing can be done.

****

## Data flow definitions

The table below gives a description of each element in the data flow diagram. Each description includes how the data element will be used and passed between subsystems.

|  |  |
| --- | --- |
| Elements | Description |
| B1 | The User presses the power button to turn on the Sidewalk Sketcher. |
| B2 | The User presses the Start button to start printing the image. |
| B3 | The User interacts with the User Interface to provide the user input. |
| B4 | The User selects the color options from the UI. |
| B5 | The User resizes the image to desired size. |
| B6 | The User crops the image to print the desired portion of the image. |
| U1 | User Interface layer interacts with the software input layer when user completes browsing the image and pass it to the image reader. |
| U2 | After color selection is made, I get displayed. |
| U3 | After cropping, final image is displayed in the UI. |
| U4 | Desired file or image is displayed. |
| U5 | After resizing the image, processed image is displayed. |
| U6 | Presentation subsystem interacts with Image processing subsystem in software processing layer. |
| U7 | All changes are displayed to the user. |
| I1 | After filtering the image into two colors, the image will be sent for processing in the software-processing layer. |
| I2 | When Image Reader in software input layer receives the image, it passes the image to the converter subsystem, which eventually converts the image into the two colors image. |
| I3 | After changing image into the two-color image, converter subsystem passes it to the transfer subsystem so that it can pass it to the software processing layer. |
| P1 | Information processing subsystem passes the required information to the Image processing subsystem. |
| P2 | Information processing subsystem access the database to complete the queries requested by the user with the help of database manager. |
| P3 | Completing the image processing, image is transfer to the software output layer, which is received by data packaging subsystem. |
| P4 | After processing the image, if the user wants to process the information. Image processing subsystem passes it to the information processing subsystem. |
| P5 | Needed information is passed to the file generator subsystem. |
| O1 | File generator generates the information from the Information processing subsystem. |
| O2 | File generator transfers the files to the data packager for packaging. |
| O3 | Packaged data will be sent to the file transfer that will be the Hardware input. |
| O4 | Processed data is transfer to the Hardware via USB cable, which is received by the file reader. |
| D1 | Response to the requested query from file repository to the database manager. |
| D2 | Response to the requested query from Image repository to the database manager. |
| D3 | Database manager retrieves the file from the file repository. |
| D4 | Database manager retrieves the file from the image repository. |
| D5 | Database manager response to the queries from the information processing subsystem. |
| HI1 | Receives the power signal when the power button is turned on. |
| HI2 | Receives the start signal when the start button is pressed. |
| HI3 | Data received by file reader is synchronized and further processed in synchronization subsystem. |
| HI4 | Since we are using camera for positioning the robot. The user need set up the position of the camera and need to pass it to synchronization input subsystem. |
| HI5 | When everything is set up, synchronization input subsystem passes the data to transfer data subsystem. |
| HI6 | There are sensors that we so for different purpose. In order to start sketching user need to make sure all the sensor is functioning we will acts as the input as well. |
| HI7 | Transfer data subsystem passes the input data to the hardware processing layer for further processing. |
| HI8 | Camera input helps in the motion of the robot. |
| HP1 | As sketch process starts, Hardware processing layer provides the image data so that the sketch layer so that it can sketch the image. |
| HP2 | All the hardware input is received by the hardware input driver in hardware processing layer and pass it to synchronization subsystem. |
| HP3 | While sketching, robot need to update its position, should update status of the chalk or any error so there need the constant communication between sketch layer and Hardware processing layer. |
| HP4 | When robot needs to notify user, synchronization subsystem passes the information to the Hardware output driver. |
| HP5 | Robot needs to know its current position. |
| HP6 | Hardware processing layer needs to communicate with the motion layer to get the position of the robot. |
| HP7 | In case of chalk depletion, battery depletion or completion of the sketch process, user is notified with the output from the Hardware output driver. |
| HO1 | When the Chalk or battery is near depletion, the alarm goes off and notifies the user. |
| HO2 | When the sketching the image is complete, the LED lights notify user about the completion. |
| S1 | The depletion subsystem will use its data as input for the Hardware Input Layer. |
| S2 | The depletion subsystem updates the synchronization subsystem when the piece of chalk is near depletion. |
| S3 | The synchronized data from the Sketch Layer will be transferred to be processed in the Hardware Processing Layer. |
| S4 | The synchronization subsystem will alert the piston mechanism when it is time to be in the high or the low position. |
| S5 | The synchronization system will tell the depletion system to stop action when instructed from the Hardware Processing Layer. |
| S6 | The piston mechanism will send its status to the synchronization layer along with any problems it may have encountered. |
| M1 | The subsystem passes information regarding the position of the device along with any problems that may have been encountered. |
| M2 | The position subsystem sends its updates and current position to the synchronization subsystem. |
| M3 | This subsystem sends its updates and any problems it may have encountered due to the movement of the device. |
| M4 | The synchronization layer lets the motor later what coordinates to move next. |

## Producer- Consumer Relationships

Producer-Consumer Relationships is divided into two sections: software section and hardware section. Those tables demonstrate the relationships between data producers and their respective consumers. Producers are represented in the rows on the left and the consumers are represented in the columns on the top.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **User Input** | **File Browser** | **Resize** | **Presentation** | **Cropping** | **Color Selector** | **Image Reader** | **Converter** | **Transfer Data** | **Image Processing** | **Information processing** | **Database Manager** | **Image Repository** | **File Repository** | **Data Packaging** | **File Generator** | **File Transfer** |
| **User Input** |  | B3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **File Browser** |  |  |  | U5 |  |  | U1 |  |  |  |  |  |  |  |  |  |  |
| **Resize** |  |  |  | U4 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Presentation** |  |  |  |  |  |  |  |  |  | U6 |  |  |  |  |  |  |  |
| **Cropping** |  |  |  | U3 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Color Selector** |  |  |  | U2 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Image Reader** |  |  |  |  |  |  |  | I2 |  |  |  |  |  |  |  |  |  |
| **Converter** |  |  |  |  |  |  |  |  | I3 |  |  |  |  |  |  |  |  |
| **Transfer Data** |  |  |  |  |  |  |  |  |  | I1 |  |  |  |  |  |  |  |
| **Image Processing** |  |  |  |  |  |  |  |  |  |  | P4 |  |  |  | P3 |  |  |
| **Information Processing** |  |  |  |  |  |  |  |  |  | P1 |  | P2 |  |  |  | P5 |  |
| **Database Manager** |  |  |  |  |  |  |  |  |  |  | D5 |  | D4 | D3 |  |  |  |
| **Image Repository** |  |  |  |  |  |  |  |  |  |  |  | D2 |  |  |  |  |  |
| **File Repository** |  |  |  |  |  |  |  |  |  |  |  | D1 |  |  |  |  |  |
| **Data Packaging** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | O3 |
| **File Generator** |  |  |  |  |  |  |  |  |  |  | O1 |  |  |  | O2 |  |  |
| **File Transfer** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **User Input** | **File Transfer** | **File Reader** | **Sensor Reader** | **Synchronization Input** | **Camera Processing** | **Transfer Data** | **Hardware Input Driver** | **Sketcher Processing** | **Synchronization** | **Hardware Output Driver** | **Position Processing** | **Sketcher Sync** | **Depletion** | **Piston** | **Motion Sync** | **Motor Driver** | **Position** | **Output De-multiplex** | **Alarm** | **Light** |
| **User Input** |  |  |  |  | HI1/HI2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **File Transfer** |  |  | O4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **File Reader** |  |  |  |  | HI3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Sensor Reader** |  |  |  |  | HI6 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Sync Input** |  |  |  |  |  |  | HI5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Camera Processing** |  |  |  |  | HI4 |  |  |  |  |  |  |  |  |  |  |  |  | H  I  8 |  |  |  |
| **Transfer Data** |  |  |  |  |  |  |  | HI7 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Hardware Input Driver** |  |  |  |  |  |  |  |  |  | HP2 |  |  |  |  |  |  |  |  |  |  |  |
| **Sketcher Processing** |  |  |  |  |  |  |  |  |  | HP3 |  |  | HP1 |  |  |  |  |  |  |  |  |
| **Sync** |  |  |  |  |  |  |  |  |  |  | HP4 |  |  |  |  |  |  |  |  |  |  |
| **Hardware Output Driver** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | HP7 |  |  |
| **Position Processing** |  |  |  |  |  |  |  |  |  | HP5 |  |  |  |  |  | HP6 |  |  |  |  |  |
| **Sketcher Sync** |  |  |  |  |  |  |  |  | S3 |  |  |  |  |  |  | S5 | S4 |  |  |  |  |
| **Depletion** |  |  |  |  |  |  |  |  |  |  |  |  | S2 |  |  |  |  |  |  |  |  |
| **Piston** |  |  |  |  |  |  |  |  |  |  |  |  | S6 |  |  |  |  |  |  |  |  |
| **Motion Sync** |  |  |  |  |  |  |  |  |  |  |  | M1 |  |  |  |  | M4 |  |  |  |  |
| **Motor Driver** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | M3 |  |  |  |  |  |
| **Position** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | M2 |  |  |  |  |  |
| **Output De-multiplex** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | HO  1 | HO2 |
| **Alarm** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Light** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

# Layer Definitions

This section provides a brief description of all the layers in the Sidewalk Sketcher. It includes the Software Input Layer, Software Output Layer, Software Processing Layer, User Interface Layer, Data Storage Layer, Hardware Input Layer, Hardware Output Layer, Hardware Processing Layer, Sketch Layer and Motion Layer.

## 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.

## 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.

## 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.

## User Interface Layer

The User Interface Layer allows the user to select an image that will be used by Sidewalk Sketcher to draw an image. It will also allow user to crop and resize an image to provide flexibility to users to sketch desired size and portion of selected image. The interface will provide an option to view the chalk color that will be used when the image is actually being sketched. This layer however, will not insure that the image looks in real life like it will look in the user interface because the actual color will depend on what colors are actually loaded into the Sidewalk Sketcher.

## 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.

## 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.

## Hardware Output Layer

This layer will handle of all the notifications that will be sent from this device to the user, such as low batter alert and chalk depletion. This is the only layer that will interfere with the user.

## 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. 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.

## Sketch Layer

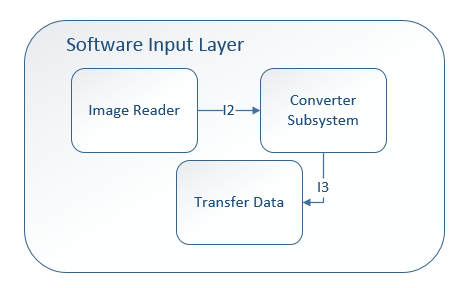
The purpose of the Sketch Layer is to analyze all of the actions that will be performed by the sketching device that will drag the chalk, pick up the chalk when it’s not drawing, and send the update to the Hardware Processing Layer when the chalk nears depletion. This layer will communicate directly 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 information to make sure the chalk is connecting to the ground when it is supposed to be sketching the image.

## 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, it 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.

# 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

### Description: The File Browser Subsystem will provide the Image Reader an image file that will be read before conversion.

### Assumptions: The Image Reader will be able to support .jpeg, .bmp and .png files.

### Responsibilities: The Image Reader is responsible for scanning the image file making sure that it is an appropriate file prior to image conversion.

### Inter-Layer Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Information Required | Information Returned |
| readImg | The Image Reader will receive the image file from the User Interface layer | Image File | None |

### Public Interfaces: This subsystem does not have any external interfaces.

## Converter System

### Description: The Converter Subsystem will accept an input from the File Image Reader for file conversion into an acceptable data file.

### Assumptions: The file has already been approved by the Image Reader.

### Responsibilities: The Converter Subsystem is responsible for formatting the image file to an acceptable data file and outputs to the Transfer Data subsystem for Software Processing Layer.

### Inter-Layer Interfaces: This subsystem does not interface with another layer.

### Public Interfaces: This subsystem does not have any external interfaces.

## Transfer Data

### Description: The Transfer Data subsystem will take the converted image file and transfer the file to the Software Processing layer.

### Assumptions: The file has been fully converted and ready for transfer.

### Responsibilities: The Transfer Data subsystem is responsible for transferring the converted file to the Image Processing subsystem.

### Inter-Layer Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Information Required | Information Returned |
| sendImg | The Transfer Data subsystem will transfer the file to the Image Processing subsystem in the Software Processing Layer | Image File | Image File |

### Public interfaces: This subsystem does not have any external interfaces.

# 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

### Description: Data Packager subsystem helps to wrap up the data, which it receives from Software processing layer. After wrapping the processed data, it passes the data to the file transfer subsystem.

### Assumptions: The data received from software processing layer is in postscript. It is assumed that it interact without any problem with software processing layer and File transfer subsystem.

### Responsibilities: This subsystem will be responsible for packaging the input processed data from the data processor to file transfer subsystem.

### Inter-Layer Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Info Required | Info Returned |
| getData | Receives the data | None | None |

### Public Interfaces: This subsystem does not have any external interfaces.

## File Generator

### Description: This subsystem interacts with the information processing subsystem to generate the requested file by the user.

### Assumptions: Data received from the information processing subsystem is not manipulated.

### Responsibilities: This subsystem is responsible for communicating with information processing subsystem and get required information.

### Inter-Layer Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Info Required | Info Returned |
| retrieveInfo | Communicates with information processing subsystem for needed information. | None | Saved files. |
| receiveData | receives information from the information processing subsystem. | None | Files. |

### Public Interfaces: This subsystem does not have any external interfaces.

## File transfer

### Description: This subsystem receives the postscript data from data packager. It helps to provide the postscript data to hardware input.

### Assumptions: Data is not manipulated while transferring in between the subsystem.

### Responsibilities: This subsystem accepts the data from data packager and provide that data as hardware input.

### Inter-Layer Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Info Required | Info Returned |
| putFile | Sends converted file to the hardware | None | None |

### Public Interfaces: The subsystem does not have any external interfaces.

# 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

### Description: Once the user input is received, the received image is converted into the postscript with the help of image processing layer. Besides converting to the postscript, it also handles the user actions like resizing, cropping, image contrasts and so on.

### Assumptions: It is assumed that the originality of the image remains the same and image is not corrupted.

### Responsibilities: This subsystem will be responsible for processing all the image files and pass it to the output layer.

### Inter-Layer Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Info Required | Info Returned |
| processImage | Handle input command received from the user and process it | Image | Image. |
| receiveData | Receives the image data from Transfer Data subsystem. | None | Image data. |
| Transferdata | Transfers data to data packager. | None | Postscript data. |

### Public Interfaces: This subsystem does not have any external interfaces.

## Information Processing

### Description: This subsystem interacts with the data storage layer in order to store the image file or retrieve the file from the database. It also communicates with the file generator subsystem.

### Assumption: It is assumed that there is no any communicating problem with database.

### Responsibilities: This subsystem will be responsible for saving the image file and retrieving the stored file from the database.

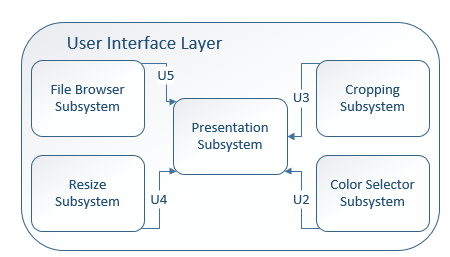
### Inter – Layer Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Info Required | Info Returned |
| storeData | Interacts with the database managers and stores the processed file | Image file | None |
| receiveData | Receives data from database. | Query | Image and data files. |
| transferInfo | Transfers information requested by the file generator subsystem. | None | Files. |
| receiveRequest | Receives requests from the file generator subsystem. | None | None. |

### Public Interfaces: This subsystem does not have any external interfaces.

# User Interface Layer

The User Interface Layer allows the user to select an image that will be used by Sidewalk Sketcher to draw an image. It will also allow user to crop and resize an image to provide flexibility to users to sketch desired size and portion of selected image. The interface will provide an option to view the chalk color that will be used when the image is actually being sketched. This layer however, will not insure that the image looks in real life like it will look in the user interface because the actual color will depend on what colors are actually loaded into the Sidewalk Sketcher.



## File Browser Subsystem

### Description: When Sketcher interface is opened the user can click on select file option to choose the desired image from user’s computer in order to be processed.

### Assumptions: Image files are present in user’s computer and the user is in default working directory.

### Responsibilities: The File Browser Subsystem is responsible to provide an interface for user to select desired image from the user computer. After an image is selected the subsystem is responsible of providing a confirmation button to confirm the selection.

### Inter-Layer Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Parameter | Data Returned |
| selectImage | Selects the required image | String Filename | Image File |

### Public Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Parameter | Data Returned |
| browseListner | Opens file chooser menu | None | None |

## Cropping Subsystem

### Description: After the user selects an image to be processed the user can click on crop image option to print desired portion of selected image.

### Assumptions: The file chosen by user is JPEG, Bit map, PNG format.

### Responsibilities: The Cropping Subsystem is responsible to provide user a frame that can be enlarge or reduced to provide flexibility to select desire portion of the image. It is responsible to provide a crop button that will be clicked after user is done selecting required portion of an image.

### Inter-Layer Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Parameter | Data Returned |
| cropImage | Crops the selected image | Four coordinates | Cropped image |

### Public Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Parameter | Data Returned |
| cropAction | Opens the crop editor menu | None | None |

## Resize Subsystem

### Description: The Resize Subsystem will allow user to enter the size of image that will be sketched by Sketcher.

### Assumptions: None

### Responsibilities: The Resize Subsystem is responsible to provide a text field where the user can enter the size of output image in inch.

### Inter-Layer Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Parameter | Data Returned |
| resizImage | Takes size of image to print | int length, int breadth | Cropped image |

### Public Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Parameter | Data Returned |
| resizeAction | Opens the crop editor menu | None | None |

## Color Selector Subsystem

### Description: The Color Selector Subsystem provides the user the option to select which two colors will be used in the sketch. This subsystem serves the purpose for the user to see the final sketch. However, the actual color of the sketch will be determined by the two actual colors that will be present in the Sidewalk Sketcher.

### Assumptions: Chalk holder 1 is used for high contrast color and chalk holder 2 is used for low contrast color.

### Responsibilities: The Color Selector Subsystem is responsible to provide an option for user to select the chalk color for high contrast and low contrast pixel of image so that the user can have view of final sketch output.

### Inter-Layer Interfaces:

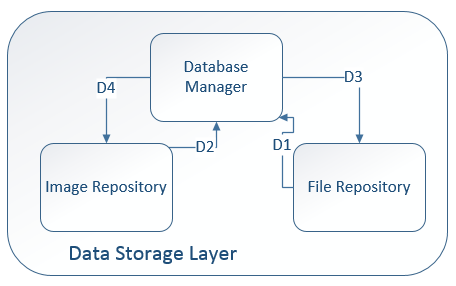
|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Parameter | Data Returned |
| selectColor | Takes colors for high contrast and low contrast color | String high, String low | Two colored image |

### Public Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Parameter | Data Returned |
| selectColorAction | Ask for two colors | None | None |

# 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

### Description: The Image Repository subsystem is responsible for acquiring and holding all image files given from the Database Management.

### Assumption: The file is an acceptable file from the Software Processing Layer and there is enough storage space in the hardware.

### Responsibilities: The Image Repository Layer will receive all image files and output any image files requested from the Database Management.

### Inter-Layer Interfaces: This subsystem does not interface with another layer.

### Public Interfaces: This subsystem does not have any external interfaces

## File Repository

### Description: The File Repository holds all data files of instructions generated and outputted to the Database Management Subsystem.

### Assumption: The Image file is already converted to a plotter instruction data file for the hardware to process

### Responsibilities: This subsystem is responsible for holding all data files for extraction to the File Generator.

### Inter-Layer Interfaces: This subsystem does not interface with another layer.

### Public Interfaces: This subsystem does not have any external interfaces

## Database Management

### Description: The Database Management System sends requests to the File Repository and Image Repository and sends the data to the respective subcomponents.

### Assumption: The Image Repository will contain only Image files, and the File Repository will only contain data files. The Database Management Subsystem will send requests to files that are present in the repository, and throws an exception otherwise.

### Responsibilities: The requests will be made when the Software Processing Layer requires an image/data file for Information Processing. The Database Management Subsystem responsibilities are to grab the requested image/data file.

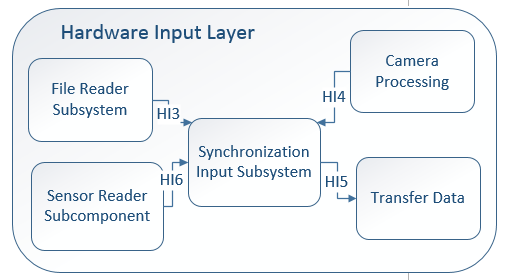
### Inter-Layer Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Info Required | Info Returned |
| grabFile | Interacts with the Information Processing subsystem by requesting files from the Image Repository or File Repository. | Data file | Data File |

### 

# 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

### Description: File sent for software output layer is received by this subsystem.

### Assumptions: File with suitable format is sent from Software Output layer.

### Responsibilities: The file received is checked for its integrity and sent to the synchronization input subsystem.

### Inter-Layer Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Parameter | Data Returned |
| receiveFileFromSOL | Receives data from File transfer subsystem | Boolean | Image File |
| sendFileToSIS | Send file read from file transfer to synchronization input subsystem. | String of bits | Boolean |

### Public Interfaces: This subsystem does not have external interfaces.

## Sensor Reader Subcomponent

### Description: Output from chalk depletion subsystem is received by this subsystem.

### Assumptions: Sensor are working fine and constantly sending their output to this subsystem.

### Responsibilities: The data received from the sensors is converted to appropriate format that can be understood by hardware processing layer and then sent to the SIS (Synchronization input subsystem).

### Inter-Layer Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Parameter | Data Returned |
| receiveSensorData | Receive data from the depletion subsystem. | sensorId | Sensor Data (float) |
| sendSensorData | Sends sensor data to synchronization input subsystem | Sensor Data (float) | Boolean |

### Public Interfaces: This subsystem does not have external interfaces.

## Synchronization Input Subsystem

### Description: Merges data from File reader subsystem, Sensor reader subsystem and camera processing.

### Assumption: File reader subsystem, Sensor reader subsystem and camera processing all are functioning and send desired data to this subsystem.

### Responsibilities: Receives data from File reader subsystem, Camera Processing, Sensor Reader Subcomponent, power button and start button and sends it to transfer data sub- system.

### Inter-Layer Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Parameter | Data Returned |
| receiveFileReaderData | Received data from the Synchronization Subsystem | File line number | Data concerning trigger flags for the output layer |
| receiveSensorReaderData | Receive data from the sensor reader subsystem. | none | Chalk length (float) |
| receiveCameraProcessingData | Receives data from camera processing subsystem. | none | Location data (float) |
| sendSyncedData | Send the synched data to transfer data subsystem | String of all merged data from various sub components | Boolean |

### Public Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Parameter | Data Returned |
| powerButtonHandler | Handles power button push event. | none | integer |
| startButtonHandler | Handles start button push event. | none | integer |

## Camera Processing

### Description: Receives position data from position subsystem in Motion Layer and processes it and passes it to synchronization input subsystem.

### Assumption: Position subsystem is working and sending data to camera processing subsystem.

### Responsibility: It is responsible for receiving data from position subsystem, format it and send it to synchronization input subsystem.

### Inter-Layer Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Parameter | Data Returned |
| receivedPositionData | Received data from the position subsystem. | none | Position data merged as a String |
| sendProcessedDataToSIS | Send the processed data from camera processing subsystem to SIS. | Position data merged as a String | Boolean |

### Public Interfaces: This subsystem does not have external interfaces.

## Transfer Data

### Description: Takes care of data transfer between Hardware input layer and Hardware Processing Layer.

### Assumption: Synchronization Input subsystem sends required data.

### Responsibility: This layer is responsible for receiving data from synchronization input subsystem, formatting the data, and sending it to Hardware input Driver subsystem.

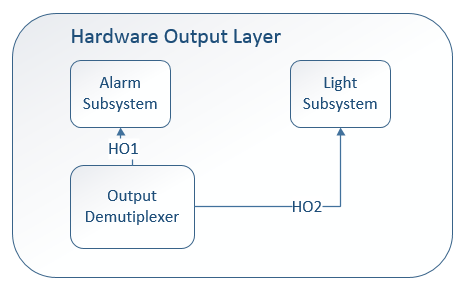
### Inter-Layer Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Parameter | Data Returned |
| receivedSynchorizedData | Received data from the Synchronization input Subsystem | none | String of all merged data |
| sendSynchorizedData | Send the data received from transfer data to HIDS | String of all merged data | Boolean |

### Public Interfaces: This subsystem does not have external interfaces.

# 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

### Description: Receives input from Output de-multiplexer and generates sound alerts.

### Assumptions: The speaker is working and output de-multiplexer is functioning properly.

### Responsibilities: This subsystem is responsible for producing sound alert upon receiving instruction from Output De-multiplexer subsystem in cases when the user needs to be notified. It can produce different sound pattern alerts depending on the instructions received.

### Inter-Layer Interfaces

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Parameter | Data Returned |
| receiveInstructionFromOD | Receives instruction from output de-multiplexer. | none | Data bits |

### Public Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Parameter | Data Returned |
| produceSoundAlert | Causes the speaker in the alert subsystem to produce sound. | Data bits | none |

## Light Subsystem

### Description: Receives input from Output de-multiplexer and causes the LEDs to flash.

### Assumptions: The LEDs and de-multiplexer are functioning properly.

### Responsibilities: It is responsible for generating alert in the form of light on the LEDs upon receiving instruction from output de-multiplexer. It should be able to generate various patterns of light on the LEDs.

### Inter-Layer Interfaces

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Parameter | Data Returned |
| receiveInstructionFromOD | Receives instruction from output de-multiplexer. | none | Data bits |

### Public Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Parameter | Data Returned |
| produceLightAlert | Causes the LEDs in the light subsystem to light up. | Data bits | none |

## Output De-multiplexer

### Description: Receives input from Hardware Output Driver Subsystem in Hardware Processing Layer and sends it to Alarm Subsystem and Light Subsystem in Hardware Output Layer.

### Assumptions: Hardware Output Driver Subsystem sends desired output to the subsystem.

### Responsibilities: It is responsible for receiving output from Hardware Output Driver Subsystem, de-multiplexing it and sending these de-multiplexed outputs to alarm subsystem and light subsystem.

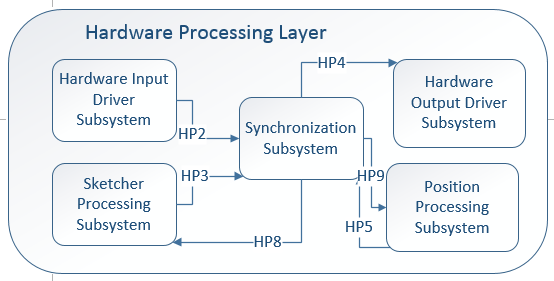
### Inter-Layer Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Parameter | Data Returned |
| receiveInstructionsFromHODS | Receives instruction from hardware output driver subsystem. | none | Data bits |
| sendInstructionsToLS | Sends instruction to lights Subsystem | Data bits | none |
| sendInstructionsToSS | Sends instruction to sound alert Subsystem | Data bits | none |

### Public Interfaces: This subsystem does not have external interfaces.

# Hardware Processing Layer

The purpose of the Hardware Processing Layer is to analyze and process data along with communicating 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

### Description: Once the hardware has received the input from the software it will be synchronized and compacted in the Hardware Input Later. After this is completed, the information will be transferred through the Hardware Processing Layer through this interface subsystem.

### Assumptions: The input layer has originally multiplexed the input information and has already synchronized the information received. The information is de-multiplexed ready to be processed.

### Responsibilities: This subsystem de-multiplexes and processes the data that it receives from the Input later. Essentially this subcomponent is the interface that links the two layers of Hardware Input Layer and the Hardware Processing Layer.

### Inter-Layer Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Parameter | Data Returned |
| receivedInputCompacted | Received input from the Hardware Input Layer | None | Processed data |

### Public Interfaces: This subsystem does not have external interfaces.

## Hardware Output Driver Subsystem

### Description: Once the hardware has processed the input, sketching process and the motion component it will send the information to the output layer when it is required. When the Sidewalk Sketcher reaches a battery percentage of 20% the subsystem will send information to the Hardware Output Layer. Other information scenarios include when the chalk on the device is running low or when the image is finished being sketched.

### Assumptions: The subsystem has received the correct flags from the Synchronization Subsystem letting it know when to connect with the Hardware Output Layer.

### Responsibilities: This subsystem is responsible for interfacing with the Hardware Output Layer by zipping the information required for the output units in a data form that can be understood and process in its destination layer.

### Inter-Layer Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Parameter | Data Returned |
| sendNextInstruction | Send the data required to trigger the output components | Flag identifier | Data to trigger the appropriate output subsystems |

### Public Interfaces: This subsystem does not have external interfaces.

## Synchronization Subsystem

### Description: This is the main processing unit of the Hardware Processing Layer. This subsystem will be controlled using the Raspberry Pi and its many ports. In this subsystem the information will all be stored, scheduled, and distributed through the rest of the subsystems in this layer.

### Assumptions: There are no assumptions in this subsystem.

### Responsibilities: This subsystem is responsible for storing, scheduling, and distributing the information to its necessary destination.

### Inter-Layer Interfaces: This subsystem does not have internal interfaces.

### Public Interfaces: This subsystem does not have external interfaces.

## Sketcher Processing Subsystem

### Description: Once the Synchronization Subsystem has gotten the data from the Hardware Input Driver Subsystem the Synchronization Subsystem will let the Sketch layer know what actions to perform next. This subsystem serves as the interface for this procedure.

### Assumptions: The Synchronization Layer is performing its duty correctly.

### Responsibilities: This unit is responsible for sending the data to the Sketch layer that will ultimately tell it what to do. This unit is also responsible for receiving status updates from the Sketch Layer to and eventually passing them to the Synchronization Subsystem that will be used in knowing how far in the sketching process is the Sidewalk Sketcher.

### Inter-Layer Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Parameter | Data Returned |
| receiveIntructions | Received data from the Synchronization Subsystem | File line number | Data concerning trigger flags for the sketch layer |
| sendStatus | Sends the status of the Sketch Layer to the Synchronization Layer | None | Data of the status of the sketching module |

### Public Interfaces: This subsystem does not have external interfaces.

## Position Processing Subsystem

### Description: Once the Synchronization Subsystem has gotten the data from the Hardware Input Driver Subsystem, the Synchronization Subsystem will let the Motion layer know what actions to perform next. This subsystem serves as the interface for this procedure.

### Assumptions: The Synchronization Layer is performing its duty correctly.

### Responsibilities: This unit is responsible for sending the data to the Motion layer that will ultimately tell it what to do. This unit is also receives status updates from the Motion Layer to be sent to the Synchronization Subsystem that will be used in knowing how far in the sketching process is the Sidewalk Sketcher.

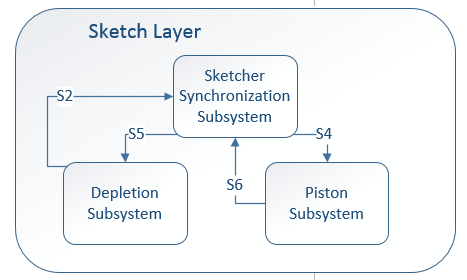
### Inter-Layer Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Parameter | Data Returned |
| receieveMotionResponse | Received the status of the Sketch Layer | None | Data of the status of the sketching module |
| sendStatus | Sends the status of the Motion Layer to the Synchronization Layer | None | Data of the status of the motion module |

### Public Interfaces: This subsystem does not have external interfaces.

# 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

### Description: This subsystem will be in charge of the communication between the Depletion Subsystem and the Piston Subsystem. Through this communication, this subsystem will know when the Sidewalk Sketcher’s chalk is reaching depletion along with having the instructions ready to control the device. When this subsystem receives the instructions to lift up the sketching device, the subsystem will instantly communicate this information with its peer subsystems.

### Assumptions: There are no explicit assumptions for this subsystem.

### Responsibilities: This system will interface with the Hardware Processing Layer by de-multiplexing information received from the Hardware Processing Layer and distributing it throughout the remaining subsystems in this layer. It will also then send the current state of the Sidewalk Sketcher back to the Hardware Processing Layer.

### Inter-Layer Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Parameter | Data Returned |
| receiveIntructions | Received data from the Hardware Processing Layer | File line number | Instructions whether to pick up the sketching device or not |
| sendStatus | Sends the status of the Sketch Layer to the Hardware Processing Layer | None | Data updating the unit of its status and/or problems |

### Public Interfaces: This subsystem does not have external interfaces.

## Depletion Subsystem

### Description: This subsystem will be in charge of doing a sensor processing based on the remaining length of the chalk. This subsystem will receive input from the sensor readings and determine whether the chalk is nearing depletion.

### Assumptions: There are no explicit assumptions for this subsystem.

### Responsibilities: This subsystem will interface with the Sketcher Synchronization Subsystem and the Hardware Input layer to alert the system if the chalk is nearing depletion.

### Inter-Layer Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Parameter | Data Returned |
| sendSensorData | Send input to the Hardware Input Layer | None | Sensor readings used to determine length of chalk |

### Public Interfaces: This subsystem does not have external interfaces.

## Piston Subsystem

### Description: This subsystem will interface with the Hardware Input Layer and the Sketcher Synchronization Subsystem. The main purpose of this subsystem is to pick up the sketching mechanism when the Sidewalk Sketcher is passing an area in the image that does not require vectors to be drawn. This unit will also consecutively work on making sure the chalk is always on the ground sketching the image when the system is supposed to be sketching.

### Assumptions: There are no explicit assumptions for this subsystem

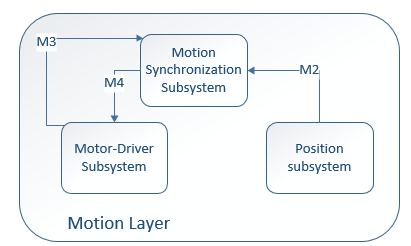
### Responsibilities: This subsystem will is responsible for making sure the Sidewalk sketcher sketches when it’s supposed to and does not when it is not supposed to (when it passes an area in the image that does not required to be sketched)

### Inter-Layer Interfaces: This subsystem does not have internal interfaces.

### Public Interfaces: This subsystem does not have external interfaces.

# Motion Layer

The purpose of the Motion Layer is to analyze the mechanical motion that will be performed by the Sidewalk Sketcher and to ensure that the device is on the correct path. The Hardware Processing Layer will communicate with this layer and let the processor know where 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

### Description: This subsystem will be in charge of communication with the Motion-Driver Subsystem and the Position Subsystem. Through this communication, this subsystem will determine what where the robot needs to go next based on input which will receive from the Hardware Processing Layer. This subsystem will sync the information relative to position and next position instructions.

### Assumptions: There are no explicit assumptions for this subsystem.

### Responsibilities: This system will de-multiplex information received from the Hardware Processing Layer and distribute the information throughout the remaining subsystems in this layer. The subsystem will also then send the current state of the Sidewalk Sketcher back to the Hardware Processing Layer.

### Inter-Layer Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Parameter | Data Returned |
| receiveIntructions | Received data from the Hardware Processing Layer | File line number | Position data |
| sendStatus | Sends the status of the Motion Layer to the Hardware Processing Layer | None | Data updating the unit if it is on the desired path |

### Public Interfaces: This subsystem does not have external interfaces.

## Motion-Driver Subsystem

### Description: This subsystem will be in charge of controlling the motors of the Sidewalk Sketcher and essentially controlling the physical motion of the device. This interface will control with the lowest level of the Sidewalk sketcher providing motion.

### Assumptions: There are no explicit assumptions for this subsystem.

### Responsibilities: This subsystem will interface with the motor drivers and provide the appropriate voltage to the motors in order to move the robot the required position.

### Inter-Layer Interfaces: This subsystem does not have internal interfaces.

### Public Interfaces: This subsystem does not have external interfaces.

## Position Subsystem

### Description: This subsystem will interface with the Hardware Input Layer and the Motion Synchronization Subsystem. The main purpose of this subsystem is to alert the motion layer about the updates of the robot’s position so that the Sidewalk sketcher can be repairs its position dynamically.

### Assumptions: There are no explicit assumptions for this subsystem

### Responsibilities: This subsystem will interface with the Camera Processing Subsystem to get the position of the robot. The subsystem will then send the information to the Motion Synchronization Subsystem in order to synchronize the position the robot.

### Inter-Layer Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Parameter | Data Returned |
| sendPositionData | send position information to the Hardware Input Layer | None | Coordinates relative to the position markers |

### Public Interfaces: This subsystem does not have external interfaces.

# 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 |  |  |

# Operating System Dependencies Section

This section is about the dependencies of our system layers with operating systems, form of libraries and interfaces. The system is designed to be dependent on two different modules. The initial step, which is to setup the user interface for Sidewalk Sketchers on users computer will be built in Python language. The user interface will be built focused Windows users. Next step is to load process image into the Sidewalk Sketcher system, which uses Raspberry Pi that is dependent on Linux kernel-base operating system.

## Software Input Layer

The Software Input layer is dependent on Windows operating system. The user interface for the Windows is created in Python language. Software input layer is responsible of converting the image file into two color image using Open CV library. In addition, the Software Input Layer is responsible for loading and reading image file selected by the user. To read and load the file we will be using CSV library in Python.

## Software Output Layer

Software Output Layer communicates with the interface to send the final image to data manager, which is dependent on Windows operating system of the users’ computer which will require Shutil.py library to be able to move file from one place to another.

## Software Processing Layer

The image chosen by the user will be processed into two-color image on windows machine. The Processed image will be then communicate with Software output layer to transfer the file to Sidewalk Sketchers System.

## Hardware Input Layer

The Hardware input layer is dependent on Linux Kernel-base Operating System. The Hardware Input Layer is responsible to read and load the processed image into the user interface for Raspberry Pi. The interface for the Raspberry Pi will also be created using Python language.

## Hardware Output Layer

The hardware output layer is dependent on Linux Kernel-base Operating System. The hardware output layer is responsible to communicate with alarm subsystem and light subsystem. This layer will communicate with alarm and light subsystem using instructions obtained from Python language operated in Linux.

## Sketch Layer

Packet of data will be received from Hardware Processing layer by sketcher layer being dependent on Linux Kernel which will convert the data into instructions running Python codes.

## Motion Layer

Packet of data will be received from Hardware Processing layer by Motion layer being dependent on Linux Kernel, which will convert the data into instructions running Python codes.

## Hardware Processing Layer

Hardware Processing Layer will obtain data packets from hardware input layer. Hardware processing layer depends on Linux Kernel to run Python codes, which will be using CVS Python library.

# Testing Considerations Section

This section details the testing considerations of all the different layers of the system. These testing considerations are specifically used to help assist the development and the testing process. To properly and efficiently test the functionality, modularity, and effectiveness of each layer, interaction between layers are minimized. These tests are conducted to help ensure that the functionality of each layer can be properly ensured.

## Overall Considerations

### Code Review: Code Reviews will be used to constantly ensure the functions of each layer are being implemented properly. This provides opportunities to find any bugs prior to testing that can help prevent larger problems.

### Ease of Use: Everything should be intuitive and quick to respond to the user. The user should not be able to independently learn the implementation of the function in order to properly use to the system.

### Beginning-to-End: Upon completing use cases, the system will be tested from beginning to end in order to test compatibility, completeness, and system functionality. These results will be compared to the expected output results to ensure the validation of the overall system requirements.

## Software Input Layer

### Modularity: The Software Input Layer must not be dependent on the internal subsystems of other layers.

### Internal System Interaction: The Image Reader will send files to the converter system in order to transfer the files to Image Processing.

### External System Interaction: This layer interacts with the User Interface Layer after the user has submitted any changes and configurations to the image.

## Software Output Layer

### Modularity: The Software Output Layer must not be dependent on the internal subsystems of other layers.

### Internal System Interaction: The Data Packaging will receive data from Image Processing and the File Generator to transfer the file to the Hardware Input Layer

### External System Interaction: This layer interacts with the Software Processing Layer and the Hardware Input Layer. The data file will be sent for processing and output to device in order to begin the Hardware Input Layer.

## Software Processing Layer

### Modularity: The Software Processing Layer must not be dependent on the internal subsystems of other layers.

### Internal System Interaction: The Image Processing subsystem receives and outputs files from the Information Processing Subsystem.

### External System Interaction: This layer interacts with the Software Input Layer, User Interface Layer, Data Storage Layer, and Software Output Layer in order to grab, send, and process information from all layers.

## User Interface Layer

### Modularity: The User Interface Layer must not be dependent on the internal subsystems of other layers.

### Internal System Interaction: There is no internal subsystem interaction.

### External System Interaction: This layer interacts with the Software Processing Layer and the Software Input Layer. The layer also interacts with the User for the File Browser, Cropping, Resize, and Color Selector subsystem.

## Data Storage Layer

### Modularity: The Data Storage Layer must not be dependent on the internal subsystems of other layers

### Internal System Interaction: The Database Manger sends requests to the File Repository and Image Repository in order to grab image or data files.

### External System Interaction: This layer interacts with the System Processing Layer and the Software Output Layer.

## Hardware Input Layer

### Modularity: The Hardware Input Layer must not be dependent on the internal subsystems of other layers

### Internal System Interaction: The File Reader, Sensor Reader, Camera Processing, and Transfer Data subsystem all interacts with the Synchronization Input Subsystem.

### External System Interaction: This layer interacts with the Software Output Layer, Sketch Layer, Hardware Output Layer, Motion Layer, and Hardware Processing Layer. The layer also interacts with the user for power and start button.

## Hardware Output Layer

### Modularity: The Hardware Output Layer must not be dependent on the internal subsystems of other layers

### Internal System Interaction: The Light subsystem throws an event to the Output De-multiplexer which gives an indication to the Alarm Subsystem

### External System Interaction: This layer interacts with the Hardware Input Layer and Hardware Processing Layer

## Hardware Processing Layer

### Modularity: The Hardware Processing Layer must not be dependent on the internal subsystems of other layers.

### Internal System Interaction: The Hardware Input Driver, Hardware Output Driver, Sketcher Processing, and Position Processing will communicate to the Synchronization Subsystem for processing.

### External System Interaction: This layer interacts with the Hardware Input Layer, Hardware Output Layer, Motion Layer, and Sketcher Layer.

## Sketcher Layer

### Modularity: The Sketcher Layer must not be dependent on the internal subsystems of other layers.

### Internal System Interaction: The Piston and Depletion Subsystem communicates with the Sketcher synchronization subsystem for processing.

### External System Interaction: This layer interacts with the Hardware Processing Layer and Hardware Output Layer

## Motion Layer

### Modularity: The Motion Layer must not be dependent on the internal subsystems of other layers.

### Internal System Interaction: The Position and Motor-Driver subsystem communicates with the Motion Synchronization subsystem to process the location and motor of the hardware.

### External System Interaction: This layer interacts with the Hardware Processing Layer and the Hardware Input Layer.