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: 12 February 2015 @ 11:41:00 PM

Table of Contents

Document Revision History

Revision Number	Revision Date	Description	Rationale
0.1	12-Feb-15	First Draft	Initial DDS

1. 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.

1.1 Purpose and Use

Sidewalk Sketcher is a smart robot that will sketch two-colored image with chalk. The purpose and use of Sidewalk sketcher is to create a banner on a smooth surface. Our product can be used in many fields. For the business people, it can be used as advertising tool. For general public, it can be used for sharing information and for some it can be used as a decoration tool to artistically sketch the floor. We will be targeting an audience that needs some source of advertising, share information and have an interest in floor art. Our product will be made publicly available and it should be affordable for public.

1.2 Project Description

Our robot Sidewalk Sketcher has two sets of modules, which run independent of each other. One module is the user interface, which will be using a 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 a 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 the image on the designated area.

2. 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.

2.1 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.

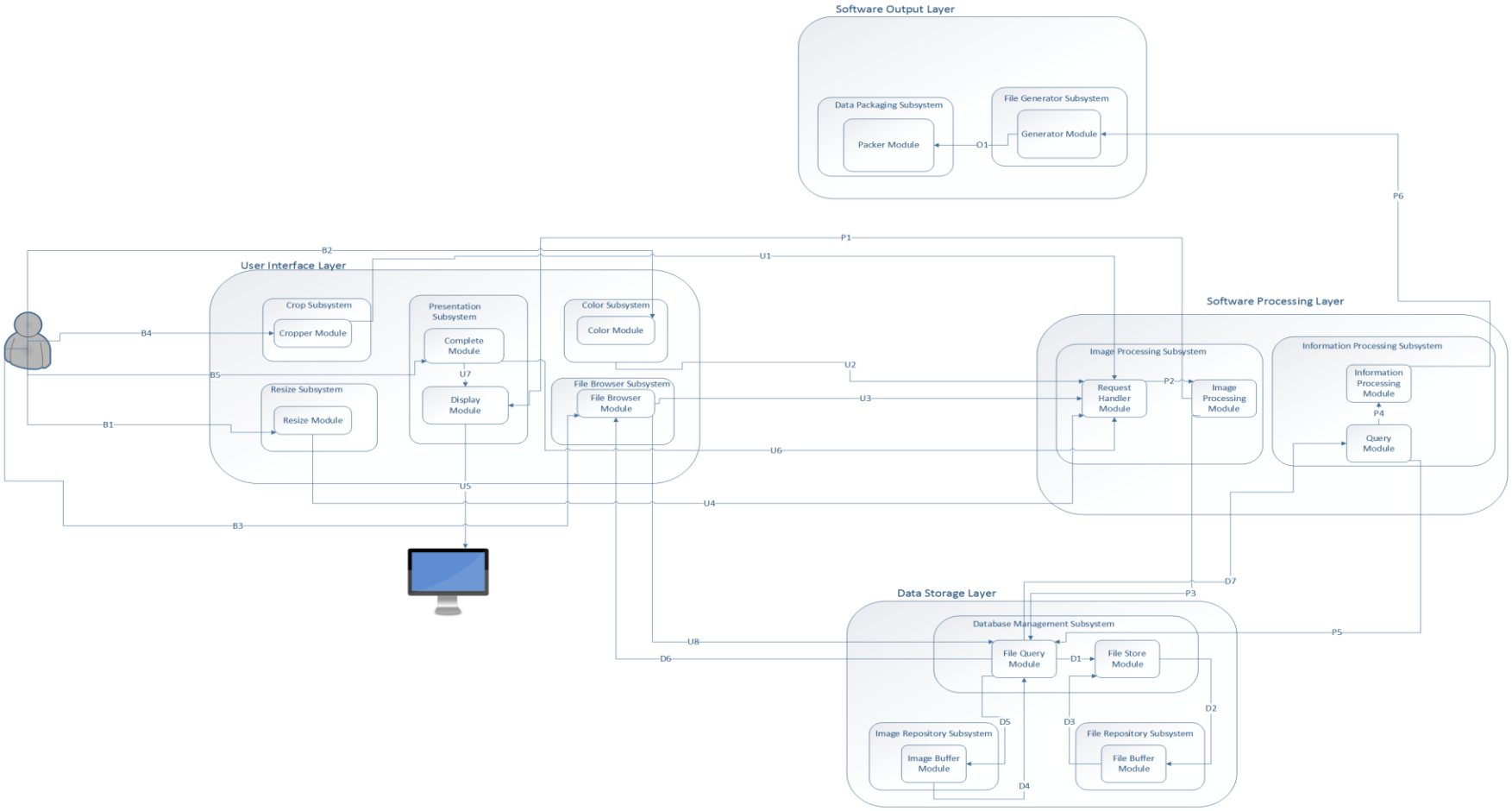
The input layers will be in charge of translating the information received into data that particular part of the system can use. From the software side, the software will translate the image file into an image that will fit the device's requirements. From the hardware

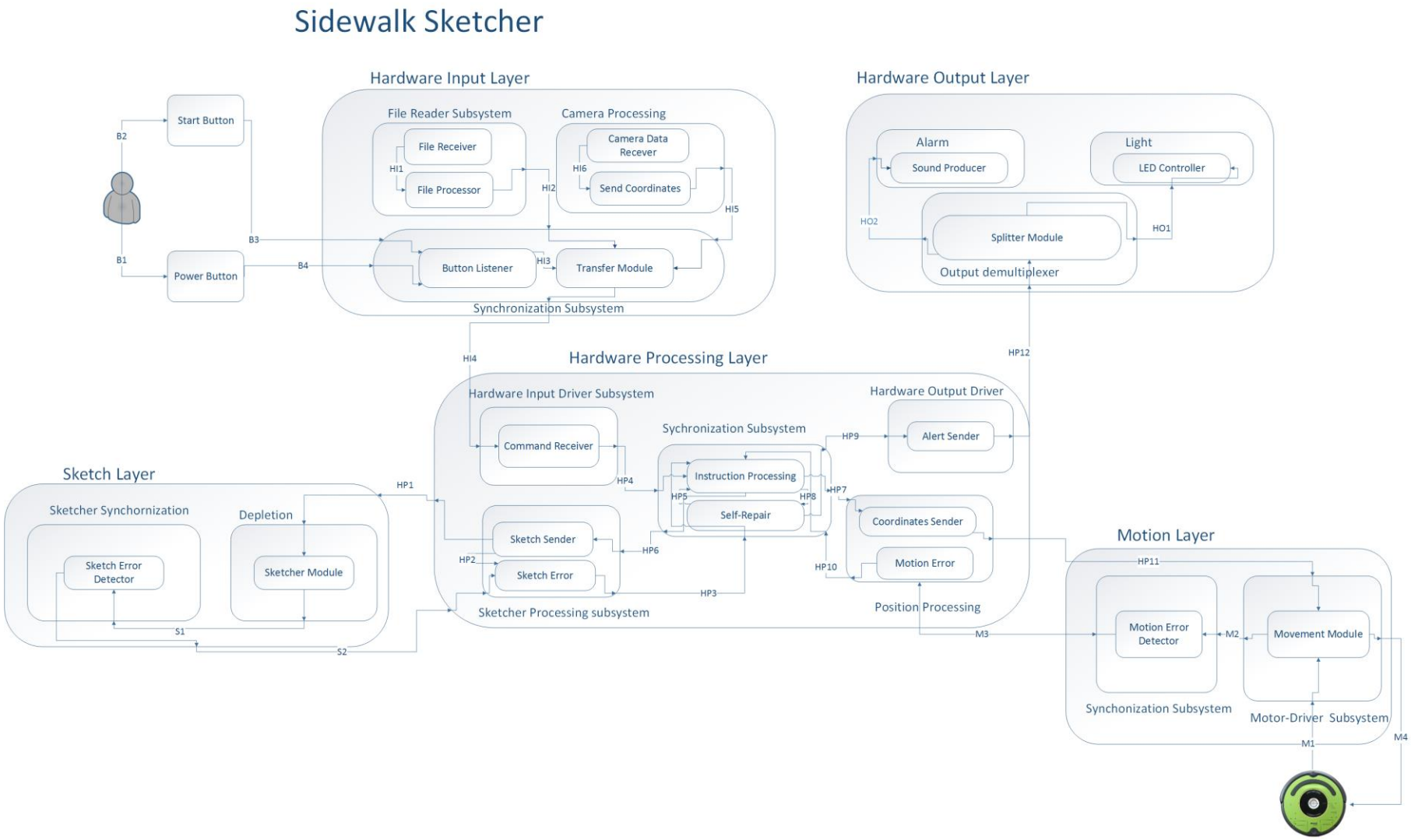
side, the hardware will parse the instructions received and store the data locally to get the sketcher ready for execution. The processing layers will be the units where the actual processing will be performed, analogous to a computer's CPU. The output layers will be in charge of sending the right data to the appropriate places such as the user or the hardware

2.2 Module Decomposition

This subsection shows a visual breakdown of each subsystem in the architecture diagram, shown in Figure 2-1, into their individual modules. The Breakdown is shown separately for software part and hardware part.

Sidewalk Sketcher




Sidewalk Sketcher
Hardware Input Layer
Start Button
B2
B1
Power Button
B3
B4
File Reader Subsystem
File Receiver
HI1
File Processor
HI2
Camera Processing
Camera Data Recever
HI6
Send Coordinates
HI5
Button Listener
HI3
Transfer Module
Synchronization Subsystem
Hardware Output Layer
Alarm
Sound Producer
Light
LED Controller
HO2
Splitter Module
HO1
Output demultiplexer
HI4
HP12
Hardware Processing Layer
Hardware Input Driver Subsystem
Command Receiver
HP4
Sychronization Subsystem
Instruction Processing
HP5
HP8
Self-Repair
HP7
Hardware Output Driver
HP9
Alert Sender
Coordinates Sender
Motion Error
HP10
Position Processing
Sketch Layer
HP1
Sketcher Synchornization
Sketch Error Detector
Depletion
Sketcher Module
S1
S2
Sketch Sender
HP6
HP2
Sketch Error
HP3
Sketcher Processing subsystem
Motion Layer
HP11
Motion Error Detector
M2
Movement Module
Synchonization Subsystem
Motor-Driver Subsystem
M3
M1
M4

2.3 Module Functional Descriptions- 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 selected image is then filtered in software processing layer into outlines of the image. The filtered image is then sent back to user interface layer to display on the screen. This layer consists of cropping subsystem, resize subsystem and color selector subsystem in order to present its corresponding functionality.

File Browser Subsystem

2.3.1 **File Browser Module:** The File browser module makes a request for an image file with the users computer database for file type, JPEG, PNG, and JPG etc. The received file is then sent to software processing layer to filter the image into outlined image.

Resize Subsystem

2.3.2 **Resize Module:** The resize subsystem allows user to enter the final product dimension of sketched image. This subsystem passes the data to software processing layer where the data is converted into instructions.

Presentation Subsystem

2.3.3 **Complete Module:** The complete module provides user a medium to tell system if the final image is ready. When the user inputs its complete flag the message is sent to software processing layer to finalize the image.

2.3.4 **Display Module:** 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.

Cropping Subsystem

2.3.5 **Image Cropper Module:** The Image Cropper module allows user to crop the image into desired size. When the user inputs the desired size a request is made to software processing system where the request is handled by request handler and cropping is done in image processing module.

Color Selector Subsystem

2.3.6 **Color Module:** The color module allows user to choose two colors from a list of colors presented by sidewalk sketcher interface. Once the color is selected the colors are then a request is made to software processing layer where image

request handler handles the request and the two colors are applied in image processing layer. Finally the image is sent to presentation layer for display.

2.4 Module Functional Descriptions- Software Processing Layer

The purpose of the Software Processing Layer is to handle and process the events that it receives from User Interface Layer. This layer filters the image and stores in the database. Software Processing Layer retrieves the image file from the database and converts the image file to the vector file.

Image Processing Subsystem

2.4.1 **Image Processing Module:** The Image Processing module receives events from the request handler as well as receives the image from the file browse subsystem. This module filters the image, which eventually gets displayed in the presentation layer. After image is filtered, this module saves the image file in the database.

2.4.2 **Request Handler Module:** The Request Handler accepts input from the user interface and passes the events to Image processing module.

Information Processing Subsystem

2.4.3 **Information Processing Module:** Information Processing Module is responsible converting the image file that it receives from Query module and converts those image files to vector files.

2.4.4 **Query Module:** Query module is responsible for accessing the database for retrieving the image files.

2.5 Module Functional Descriptions- Software Output Layer

The purpose of the Software Output Layer is to generate the instruction file and pack the file with needed data. 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 Packaging Subsystem

2.5.1 **Packer Module:** Packer module receives the instruction file from the generator module and packs the file with the needed data.

File Generator Subsystem

2.5.2 **Generator Module:** File Generator module is responsible for converting the vector list that it receives from Information processing module into the

Instruction file. After generating the instruction file, it passes the file to Packer module.

2.6 Module Functional Descriptions- 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.

Database Manager Subsystem

2.6.1 **File Query Module:** 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.

2.6.2 **File Store Module:** 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.

Image Repository Subsystem

2.6.3 **Image Buffer Module:** 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.

File Repository Subsystem

2.6.4 **File Buffer Module:** 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.

2.7 Module Functional Descriptions- 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

2.7.1 **File Receiver**: This module receives instruction file from software output layer, stores this file to local memory and sends the path to the file to the file processor module.

2.7.2 **File Processor:** File processor receives the path of the stored instruction file from file receive module, fetches the file from storage, extracts the instructions from the file, generates a list based on that and sends it to the transfer module of synchronization input subsystem.

Camera Processing Subsystem

2.7.3 **Camera Data Receiver:** This module receives continuous feed from the camera data receiver, puts these camera data into appropriate data structure and passes it to position calculator.

2.7.4 **Send Coordinates:** This modules gets camera data from camera data receiver, processes it and produces location co-ordinates of the rover's position and passes this co-ordinates to the transfer module.

Synchronization Subsystem

2.7.5 **Receiver Module:** Receiver module receives instruction file from transfer module of file reader subsystem and location co-ordinates from camera processing and sends it to transfer module of synchronization input subsystem.

2.7.6 **Button Listener:** This modules handles click event of power button and start button. Upon click this module sends the id of the button that has been clicked to the transfer module.

2.7.7 **Transfer Module:** This modules acquires data from Button Listener, File Processor and Position Calculator. Puts them into a dictionary data type and sends it to Hardware Processing Layer.

2.8 Module Functional Descriptions- 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

2.8.1 **Command Receiver:** This module will receive a python dictionary that will hold an Enum class as the key and a value pertaining to the key. Based on what the key is, will determine what method will be executed in this module.

Synchronization Subsystem

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

2.8.3 **Self-Repair:** In this module the Sidewalk Sketcher attempts to repair itself incase an error has occurred. If the Sidewalk Sketcher is unable to repair itself, it will alert the user with the appropriate alert.

Hardware Output Driver

2.8.4 **Alert Sender:** This module will result raw alerts grouped into a list from the Synchronization Subsystem and it will format all of the alerts into a python dictionary where each key represents a different type of alert. The three alerts will be a stop alert, light alert, and sound alert.

Sketcher Processing Subsystem

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

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

Position Processing Subsystem

2.8.7 **Coordinates Sender:** This module will current coordinates of the Sidewalk Sketcher and the coordinates that the device should move towards next.

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

2.9 Module Functional Descriptions- 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

2.9.1 **Sound Producer:** This module receives bit strings from splitter module and produces sound through the speaker connected to it**.**

Light Subsystem

2.9.2 **LED Controller:** This module receives bit strings from splitter module and generates light output on the LEDs connected based on that.

Output Demultiplexer

2.9.3 **Splitter Module:** This module receives string input from receiver module, splits it into two strings, one for sound alert and another for light alert.

2.10 Module Functional Descriptions- 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

2.10.1 **Motion Error Detector:** This module deals with monitoring the Sidewalk Sketcher and making sure that it performs the instruction that it’s supposed to perform.

Motion- Driver Subsystem

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

2.11 Module Functional Descriptions- 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

2.11.1 **Sketch Error Detector:** This module deals with monitoring the Sidewalk Sketcher and making sure that it performs the instruction that it's supposed to perform.

Depletion Subsystem

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

2.12 Module Producer- Consumer Matrix

Producer-Consumer Relationships is divided into two sections: software section and hardware section. The following table shows the relationship between data that is produced by modules and the modules that consume it. It shows overall data flow between the modules and depicts which modules have heavier workload. Producers are represented in the rows on the left and the consumers are represented in the columns on the top.

	User Input	Cropper	Resize	Color	Display	File Browser	Complete	Request Handler	Image Processing	Information Processing	Query	File Query	File Store	Image Buffer	File Buffer	Generator	Packer
User Input		B4	B1	B2	B3		B5										
Cropper								U1									
Resize								U4									
Color								U2									
Display																	
File Browser								U3				U8					
Complete						U7		U6									
Request handler									P2								
Image Processing					P1							P3					
Information Processing																P6	
Query										P4		P5					
File query						D6					D7		D1	D5			
File Store															D2		
Image Buffer												D4					
File Buffer													D3				
Generator																	O1
Packer																	

	User Input	File Receiver	File Processor	Camera Data Receiver	Send Coordinates	Button Listener	Transfer	Command Receiver	Instruction Processing	Self Repair	Alert Sender	Coordinates Sender	Motion Error	Sketch Sender	Sketch Error	Sketch Error Detector	Sketcher	Motion Error Detector	Movement	Splitter	LED Controller	Sound Producer
User Input																						
File Receiver			HI1																			
File Processor							HI2															
Camera Data Receiver					HI6																	
Send Coordinates							HI5															
Button Listener							HI3															
Transfer								HI4														
Command Receiver									HP4													
Instruction Processing										HP8		HP7		HP6								
Self Repair									HP5		HP9											
Alert Sender																				HP12		

	User Input	File Receiver	File Processor	Camera Data Receiver	Send Coordinates	Button Listener	Transfer	Command Receiver	Instruction Processing	Self Repair	Alert Sender	Coordinates Sender	Motion Error	Sketch Sender	Sketch Error	Sketch Error Detector	Sketcher	Motion Error Detector	Movement	Splitter	LED Controller	Sound Producer
Coordinates Sender																			Hp11			
Motion Error									HP10													
Sketch Sender																	HP1					
Sketch Error									HP3													
Sketch Error Detector															S2							
Sketcher																S1						
Motion Error Detector													M3									
Movement																		M 2				
Splitter																					HO1	HO2
LED Controller																						
Sound Producer																						

2.13 Module Data Flow Table

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 modules.

Data Element	Data Type	Description
U1	Java Event ID	Event ID of user interaction.
U2	Java Event ID	Event ID of user interaction.
U3	Java Event ID	Event ID of user interaction.
U4	Java Event ID	Event ID of user interaction.
U5	Image	Display the image.
U6	Java Event ID	Event ID of user interaction.
U7	Image	Display the filtered image.
U8	Java Event ID	Event ID of user interaction.
P1	Image File	Sends the image for display
P2	Java Event ID	Sends the Event ID to Image Processing Module.
P3	Image File	Store the filtered image.
P4	Image File	Passes the retrieved image file to Information Processing
P5	Queries	Sends the queries to retrieve the image file.
D1	Image File	Sends Image file to store.
D2	Queries	Sends the queries to retrieve the data file
D3	Data file	Response with data file
D4	Image file	Response with Image file
D5	Queries	Sends the queries to retrieve the data file
D6	Image file	Response with the image file.
D7	Image file	Response with the image file.

O1	Instruction file	When generator module completes the conversion of the vector file into the instruction file it transfer to packer module.
HI1	Instruction file	After receiving the instruction file from software component passes to file Processor
HI2	List of Instruction	Instructions from the instruction file.
HI3	Event Id	Event ID of User Instruction
HI4	Instruction Lists and Coordinates	Transfer from transfer module to command receiver
HI5	Coordinates	Sends Coordinates
HP1	List	InstructionList and the current Coordinates
HP2	List	InstructionList and the current Coordinates
HP3	List	InstructionList and the current Coordinates
HP4	List	InstructionList and the current Coordinates
HP5	List	InstructionList and the current Coordinates
HP6	List	InstructionList and the current Coordinates
HP7	List	InstructionList and the current Coordinates
HP8	List	InstructionList and the current Coordinates
HP9	List	InstructionList and the current Coordinates
HP10	List	InstructionList and the current Coordinates
HP11	List	InstructionList and the current Coordinates
HP12	String	sends list of string with appropriate output.
M1	List	InstructionList and the current Coordinates
M2	List	InstructionList and the current Coordinates
M3	List	InstructionList and the current Coordinates

M4	List	InstructionList and the current Coordinates
S1	List	InstructionList and the current Coordinates
S2	List	InstructionList and the current Coordinates
HO1	String	Alarm Alert
HO2	String	Light Alert

3. 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.

3.1 Raspberry Pi Model B+

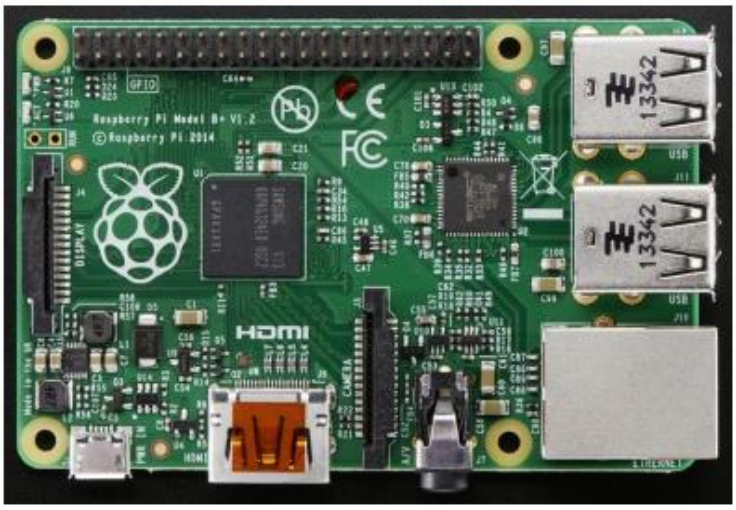


Figure 3-1 Raspberry Pi Model B+

3.1.1 Quantity:

The Sidewalk Sketcher will require one Raspberry Pi.

3.1.2 Purpose:

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

3.1.3 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

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

3.2 2GB Micro SD Card

Figure 3-2 8GB Micro SD card

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

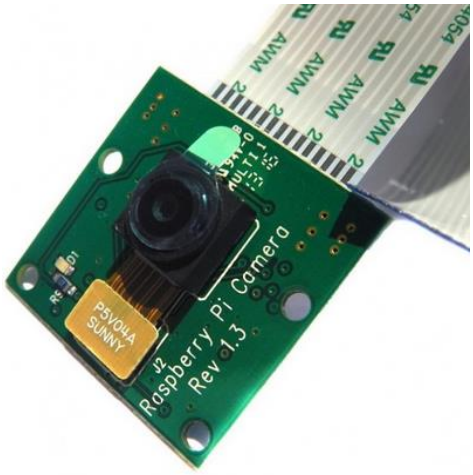
3.2.2 **Purpose:** The operating system running on the Raspberry Pi will be stored on the memory card. Most of the system data will also be stored on the SD card.

3.2.3 **Specifications:**

Model	SanDisk
Capacity	2GB
Transfer Rate	4Mbps

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

3.3 Raspberry Pi Camera



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

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

3.3.3 **Specifications:**

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

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

3.4 360 Degrees Panoramic Lens

3.4.1 **Quantity:** The Sidewalk Sketcher will require one 360 degrees panoramic lens.

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

3.4.3 **Specifications:**

Model	"Naked" Panoramic Dot optic
Manufacturer	Kogeto
Size	35 X 27mm
Weight	3 ounces

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

3.5 Roomba® iCreate 2

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

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

3.5.3 **Specifications:**

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

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

3.6 Stepper Motor

3.6.1 **Quantity:** The Sidewalk Sketcher will require one stepper motor.

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

3.6.3 **Specifications:**

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

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

3.7 LED's

3.7.1 **Quantity:** The Sidewalk Sketcher will use two LED's.

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

3.7.3 **Specifications:**

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

3.7.4 **Interfaces:** These LED's will interface with the Raspberry Pi.

3.8 Broom Stick

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

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

3.8.3 **Specifications:**

Size	1 inch diameter by ~5 feet
Quantity	4

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

3.9 Swim Noodle

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

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

3.9.3 **Specifications:**

Size	51.5 x 1.8 x 1.8 inches
Quantity	3

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

3.10 Speakers

3.10.1 **Quantity:** The Sidewalk Sketcher will use one speaker.

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

3.10.3 **Specifications:**

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

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

3.11 Wood

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

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

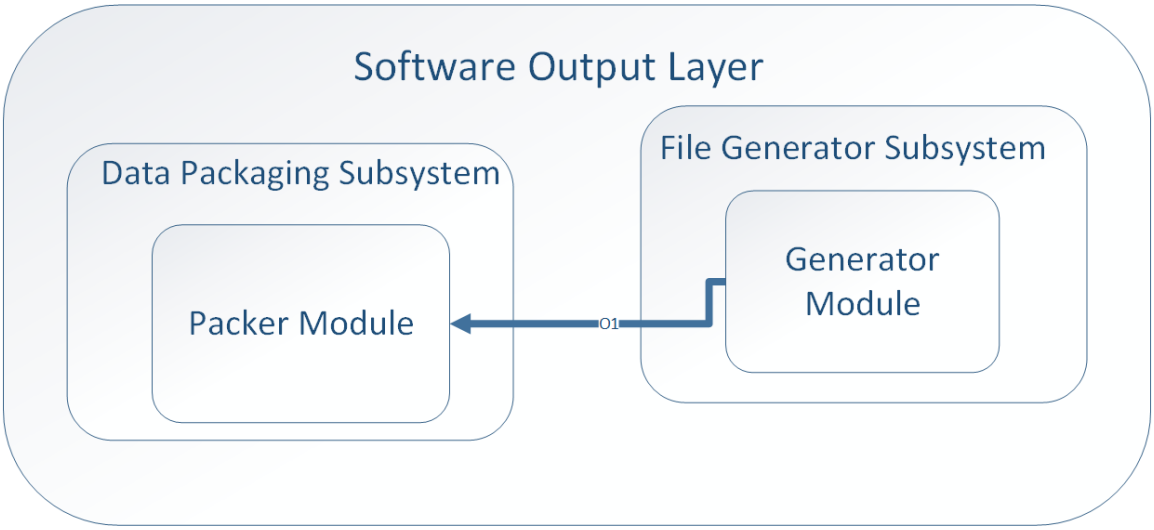
3.11.3 **Specifications:**

Size	2 inches by 4 inches by 8 feet
Quantity	1

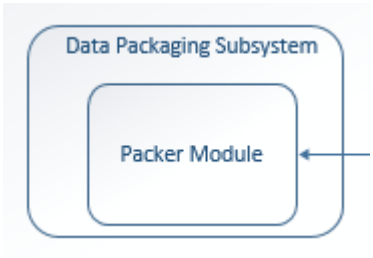
3.11.4 **Interfaces:** These pieces of wood will serve as the base for the wooden sticks.

4. Software Output Layer

The purpose of the Software Output Layer is to generate the instruction file and pack the file with needed data. 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.



4.1 Data Packaging Subsystem



4.1.1 Packer Module

Prologue

Packer module receives the instruction file from the generator module and packs the file with the needed data.

Interfaces

Source	Sink	Input to Sink	Return from Sink
Generator	Packer	Instruction file	N/A

External Data Dependencies

N/A

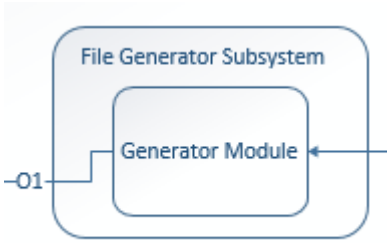
Internal Data Dependencies

Receives Instruction file from Generator Module.

Pseudo Code

```
public void PACK_File()
{
GeneratorModule.GET_FILE( ) ;
//Code to pack the files and data;
}
```

4.2 File Generator



4.2.1 Generator Module

Prologue

File Generator module is responsible for converting the vector list that it receives from Information processing module into the Instruction file. After generating the instruction file, it passes the file to Packer module.

Interfaces

Source	Sink	Input to Sink	Return from Sink
Information Processing	Generator	Vector file	N/A
Generator	Packer	Instruction file	N/A

External Data Dependencies

N/A

Internal Data Dependencies

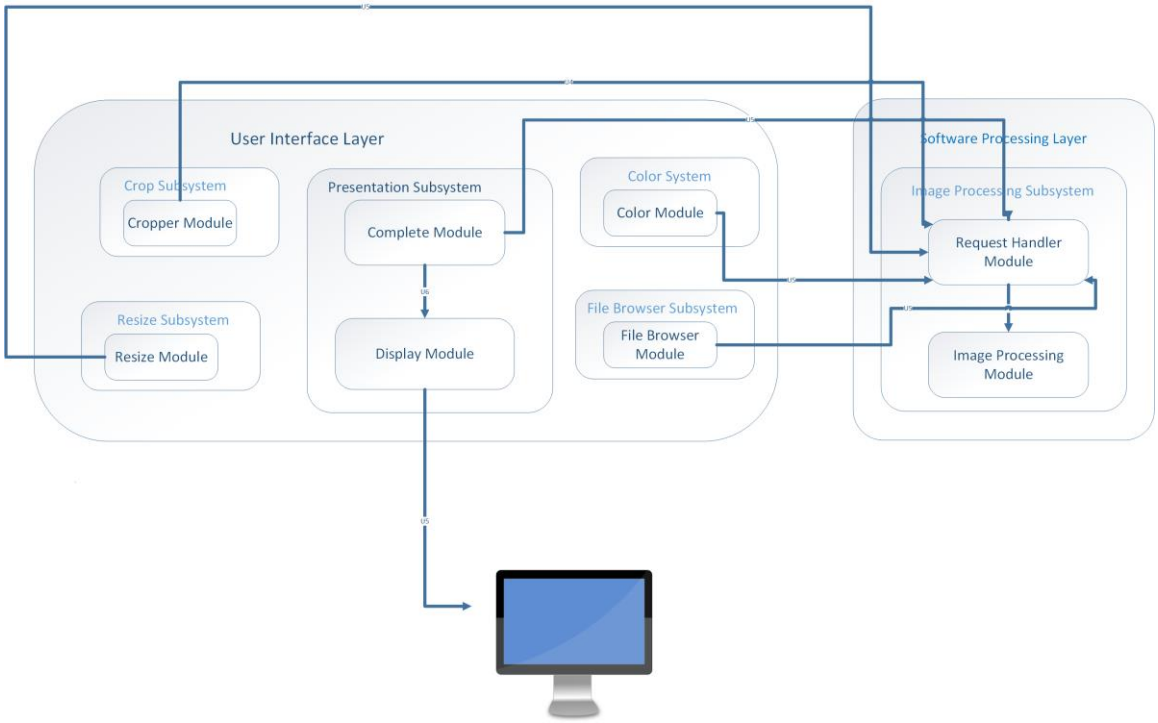
Vector file from Information Processing module

Pseudo Code

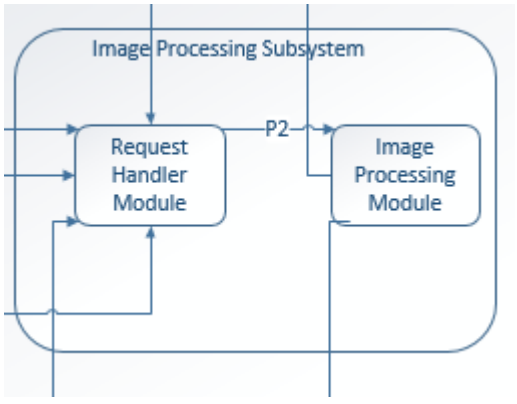
```
public void GENERATOR_FILE ( )
{
InformationProcessingModule.GET_FILE();
//Generate instruction file;
Packer.SET_DATA( myFile);
}
```

5. Software Processing Layer

The purpose of the Software Processing Layer is to handle and process the events that it receives from User Interface Layer. This layer filters the image and stores in the database. Software Processing Layer retrieves the image file from the database and converts the image file to the vector file.



5.1 Image Processing



5.1.1 Request Handler Module

Prologue

The Request Handler accepts input from the user interface and passes the events to Image processing module.

Interfaces

Source	Sink	Input to Sink	Return from Sink
User Interface	Request Handler	Event ID	N/A
Request Handler	Image Processing	Event ID	N/A

External Data Dependencies

N/A

Internal Data Dependencies

Event ID from User Interface Layer.

Pseudo Code

```
Public Event GET_EVENT ( )
{
return event;
}
Public void REQUEST_HANDLER ( )
{
ImageProcessingModule.SET_EVENT( );
}
```

5.1.2 Image Processing Module

Prologue

The Image Processing module receives events from the request handler as well as receives the image from the file browse subsystem. This module filters the image, which eventually gets displayed in the presentation layer. After image is filtered, this module saves the image file in the database.

Interfaces

Source	Sink	Input to Sink	Return from Sink
Request Handler	Image Processing	Event Id, Image	N/A
Image Processing	Database Manager	Image File	N/A

External Data Dependencies

N/A

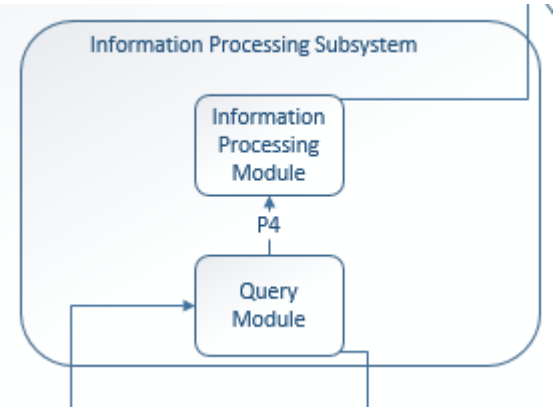
Internal Data Dependencies

Event Id, Image from Request Handler,

Pseudo Code

```
public void IMAGE_PROCESSING ( )
{
RequestHandler.GET_EVENT ( );
// Code for filter and other task;
Presentation.SET_EVENT(  Image );
}
```

5.2 Information Processing



5.2.1 Information Processing Module

Prologue

Information Processing Module is responsible converting the image file that it receives from Query module and converts those image files to vector files.

Interfaces

Source	Sink	Input to Sink	Return from Sink
Query	Information Processing	Image files	N/A
Information Processing	Generator	Vector files	N/A

External Data Dependencies

N/A

Internal Data Dependencies

Image files from Query Module.

Pseudo Code

```
public void INFORMATION_PROCESSING  ( )
{
Query.GET_FILE ( );
// Code for converting image file to vector File ;
}
```

5.2.2 Query Module

Prologue

Query module is responsible for accessing the database for retrieving the image files.

Interfaces

Source	Sink	Input to Sink	Return from Sink
Request Handler	Query	Event Id	N/A
Database	Query	Images file	N/A

External Data Dependencies

N/A

Internal Data Dependencies

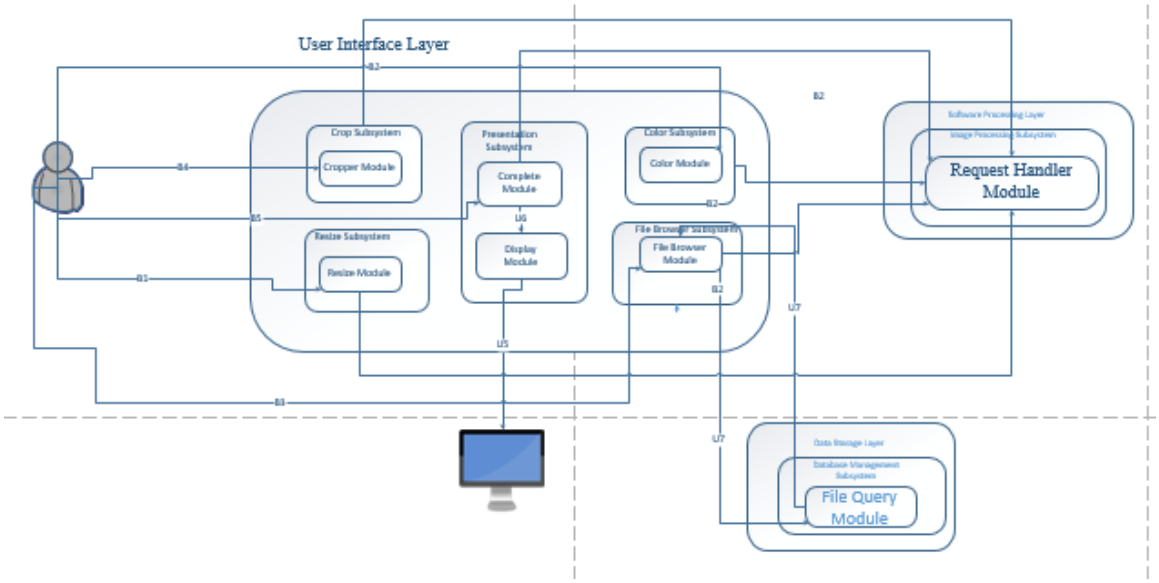
Image files from the database.

Pseudo Code

```
public void QUERY  ( )
{
DatabaseManagement.GET_IMAGE();
}
```

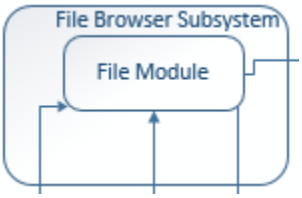
6. User Interface Layer

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 selected image is then filtered in software processing layer into outlines of the image. The filtered image is then sent back to user interface layer to display on the screen. This layer consists of cropping subsystem, resize subsystem and color selector subsystem in order to present its corresponding functionality.



The file browser subsystem allows user to select an image file from the users computer. This subsystem consists of two modules where one module gets the file from the user computer and other module makes a request to filter the image into outlined image.

6.1 File Browser Subsystem



6.1.1 File Browser Module

Prologue

The File browser module makes a request for an image file with the user's computer database for file type, JPEG, PNG, and JPG etc. The received file is then sent to software processing layer to filter the image into outlined image.

Interface

Source	Sink	Input Data to sink	Output
Windows computer hard drive	File Browser Module	Image File	N/A
File Browser module	Software Processing Layer	Image File	Filtered Image File

External Dependencies

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

Internal Dependencies.

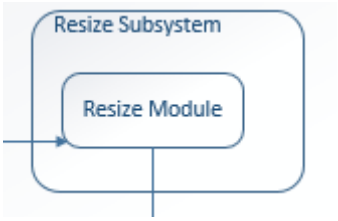
The action for the module will depend on files present on users computer. The source code for this module will be developed using Java language.

Pseudo code

```
public ImageGET_IMAGE ()
{
//Open the JFile chooser and request an image
return fileName;
}
public void SET_IMAGE( Image myImage)
{
this.image=myImage;
}
```

6.2 Resize Subsystem

The resize subsystem will allow user to input the dimension of image to be sketched. The dimension is then sent to Software processing layer where the data is used by its modules to create instructions for sidewalk sketcher.



6.2.1 Resize Module

Prologue

The resize subsystem allows user to enter the final product dimension of sketched image. This subsystem passes the data to software processing layer where the data is converted into instructions.

Interface

Source	Sink	Input Data to sink	Output
User	Resize Module	New dimensions for the image	N/A
Resize Module	Software Processing Layer	Dimension	Instructions for Sidewalk Sketcher

External Dependencies

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

Internal Dependencies.

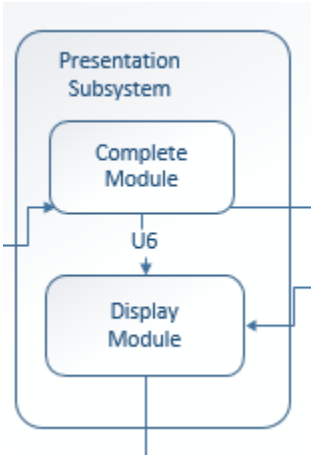
The action for the module will be developed using Java language.

Pseudo code

```
Public void GET_DATA()
{
//Gets data will get the dimension from the user
return data;
}
public float SET_DATA ( float myData)
{
//Set dimension into information processing
This.data=myData;
}
```

6.3 Presentation Subsystem

The presentation subsystem provides a medium for user to communicate with sidewalk sketcher interface. This subsystem consists of module, which handles the GUI part of the user interface as well as shows the changes that user made on the image.



6.3.1 Complete Module

Prologue

The complete module provides user a medium to tell system if the final image is ready. When the user inputs its complete flag the message is sent to software processing layer to finalize the image.

Interface

Source	Sink	Input Data to sink	Output
User	Complete Module	Complete Flag	
Complete Module	Software Processing Layer	Image File	Finalized Image

External Dependencies:

This module depends on the user to input a flag when the final image is ready.

Internal Dependencies.

The action for the module depends GUI button, which will be developed using Java language.

Pseudo code

```
Public void SET_FLAG( boolean flag )
{
// If Complete button pressed then finalize the image
this.Flag= flag;
}
```

6.3.2 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
Software Processing Layer	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.

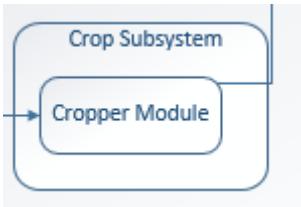
The action for the module depends on windows operating system, which will be developed using Java language.

Pseudo code

```
public void Run_GUI ( ){
//Create a user interface for side walk sketcher.
}
public Image GET_IMAGE( ){
    //Request image from image database module to display the action performed.
}
```

6.4 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.



6.4.1 Image Cropper Module

Prologue

The Image Cropper module allows user to crop the image into desired size. When the user inputs the desired size a request is made to software processing system where the request is handled by request handler and cropping is done in image processing module.

Interface

Source	Sink	Input Data to sink	Output
User	Cropper Module	X and Y position of four corners from cropping image	N/A
Cropper Module	Software Processing Layer	X and Y position of four corners from cropping image	Cropped image

External Dependencies

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

Internal Dependencies.

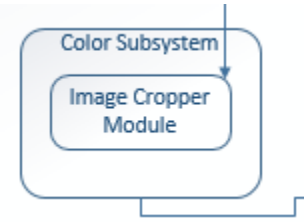
The action for the module will be developed using Java language.

Pseudo code

```
public float GET_DATA( )
{
//Gets data of desired portion of image.
return data;
}
public void SET_DATA ( float myData)
{
Set data will set the desired portion of image to be cropped.
This.data=myData;
}
```

6.5 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.



6.5.1 Color Selector Subsystem

Prologue

The color module allows user to choose two colors from a list of colors presented by sidewalk sketcher interface. Once the color is selected the colors are then a request is made to software processing layer where image request handler handles the request and the two colors are applied in image processing layer. Finally the image is sent to presentation layer for display.

Interfaces

Source	Sink	Input Data to sink	Output
User	Color Picker Module	Selects two colors from color chooser	N/A
Color Picker Module	Software Processor Layer	Sends two color chosen by user	Two colored image.

External Dependencies

The module requires the user two select two different colors to process the image.

Internal Dependencies.

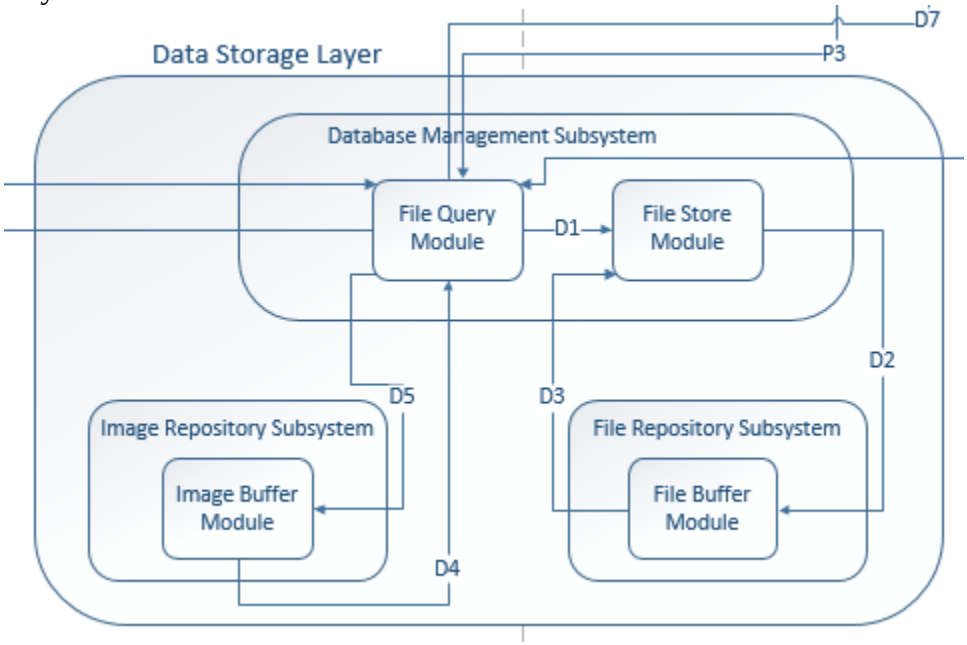
The action for the module depends on color picker module, which will be developed using Java language.

Pseudo code

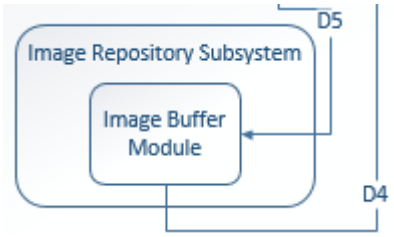
```
Public array GET_COLOR({
    //Get image from temporary memory of presentation layer
return array;
}

public void SET_COLOR ( array myColor) {
//Receives data user
This.color = myColor;
}
```

7. 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.



7.1 Image Repository



7.1.1 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
File Query	Image Buffer	None	Image File
Image Buffer	File Store	Image File	None

External Data Dependencies

None

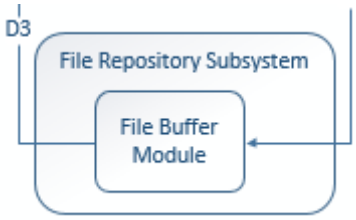
Internal Data Dependencies

The image must be stored on hardware in order to store and request the image.

Pseudo Code

```
imgList[location] = imgFile;
imgQuery(dataFile);
```

7.2 File Repository



7.2.1 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 Query	File Buffer	None	Data File
File buffer	File Store	Data File	None

External Data Dependencies

None

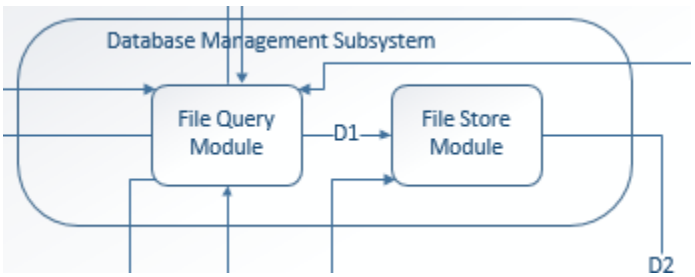
Internal Data Dependencies

The data file must be in storage in order to be stored and requested.

Pseudo Code

```
dataList[location] = dataFile;
dataQuery(dataFile);
```

7.3 Database Management



7.3.1 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 File	None	Image File
File Query	Data File	None	Data File

External Data Dependencies

None

Internal Data Dependencies

The image or data file must be in storage in order to be requested

Pseudo Code

```
dataQuery(dataFile);
imgQuery(dataFile);
```

7.3.2 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 File	Image File	None
File Store	Data File	Data File	None

External Data Dependencies

None

Internal Data Dependencies

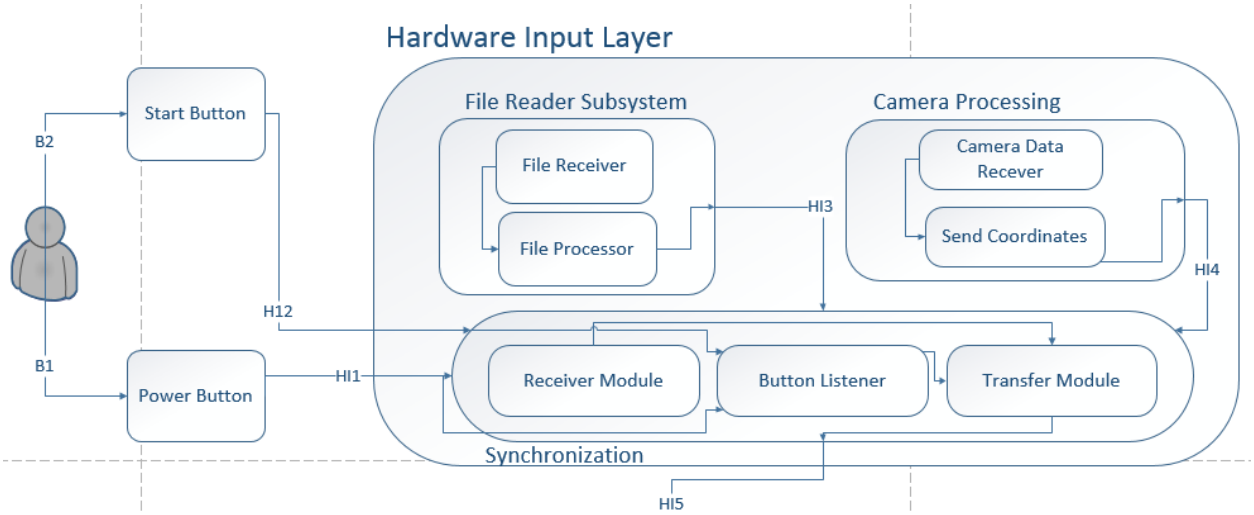
The image or data file must be in available in order to store in storage.

Pseudo Code

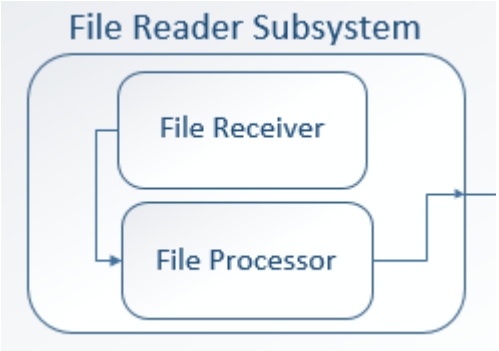
```
dataList[location] = fileName;

imgList[location] = imgFile;
```

8. 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.



8.1 File Reader Subsystem



8.1.1 File Receive Module

Description

This module receives instruction file from software output layer, stores this file to local memory and sends the path to the file to the file processor module.

Interfaces

Source	Sink	Input Data to sink	Output
Transfer Module	File Receive Module	Instruction file	Flag
File Receive Module	File Processor	File Path	Flag

External Data Dependencies

None

Internal Data Dependencies

Instruction File, File Path

Pseudo Code

```
def fileTransferHandler():
    #handles the file transfer
    saveFile(transferredFile)

def saveFile(transferredFile):
    #stores the received file i.e. transferredFile into the pi's memory
```

8.1.2 File Processor

Prologue

File processor receives the path of the stored instruction file from file receive module, fetches the file from storage, extracts the instructions from the file, generates a list based on that and sends it to the transfer module of synchronization input subsystem.

Interfaces

Source	Sink	Input Data to sink	Output
File Receive Module	File Processor	File Path	Flag
File Processor	Transfer Module	A List of Instructions	Flag

External Data Dependencies

None

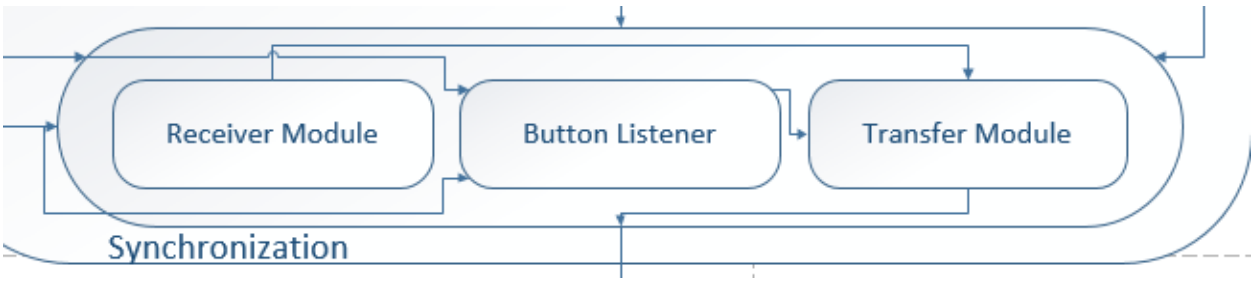
Internal Data Dependencies

File Path and List of Instruction

Pseudo Code

```
def fileProcess(passedFile):
    #instruction_list = array();
    #Get each line of the transferredFile
        #put each instruction into the instruction_list
    #send the instruction_list into the receiver module of #Synchronization Input Subsystem
```

8.2 Synchronization Input Subsystem



8.2.1 Receiver Module

Prologue

Receiver module receives instruction file from transfer module of file reader subsystem and location co-ordinates from camera processing and sends it to transfer module of synchronization input subsystem.

Interfaces

Source	Sink	Input Data to sink	Output
Transfer Module	Receiver Module	List of Instruction	Flag
Receiver Module	Transfer Module	List of Instructions and Location co-ordinates	Flag

External Data Dependencies

None

Internal Data Dependencies

List of Instruction

Pseudo Code

```
def transferDataFile(instructionFile):
    #receives the instruction list from file processor
    #and sends it to transfer module

def transferCameraCoordinates(locationCordinates):
    #receive location information every second
    #from position calculator and transfer it to the transfer #module
```

8.2.2 Button Listener

Prologue

This modules handles click event of power button and start button. Upon click this module sends the id of the button that has been clicked to the transfer module.

Interfaces

Source	Sink	Input Data to sink	Output
Button	Button Listener	Button Id (Integer)	Flag
Button Listener	Transfer Module	List	Flag

External Data Dependencies

Start Button ID, Power Button ID

Internal Data Dependencies

Button Id and List

Pseudo Code

```
def onClickEventListener():
    #listens to power button and start button and upon click
    #generate a list with id of buttons and
    #send it to transfer module.
```

8.2.3 Transfer Module

Prologue

This modules acquires data from Button Listener, File Processor and Position Calculator. Puts them into a dictionary data type and sends it to Hardware Processing Layer.

Interfaces

Source	Sink	Input Data to sink	Output
Button Listener	Transfer Module	List with Button Id	Flag
Position Calculator	Transfer Module	List of Coordinates	Flag
File Processor	Transfer Module	List of Instructions	Flag
Transfer Module	Command Receiver	Dictionary	Flag

External Data Dependencies

None

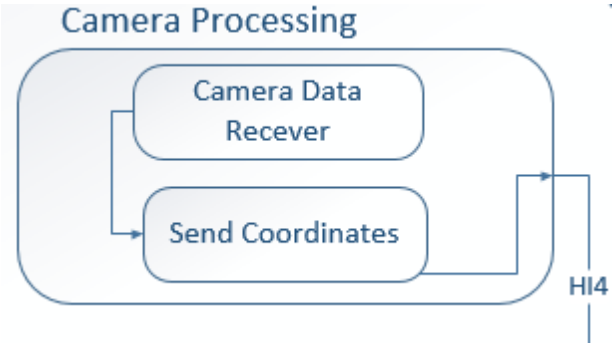
Internal Data Dependencies

Button Id, Co-ordinates and instructions.

Pseudo Code

```
def createDictionary(List):
    initialize dictionary
    case "this method called by File Processor":
        key = "Instructions"
    case "this method called by Positon Calculator":
        key = "co-ordinates"
    case "this method called by Button Listener":
        key = "button"
    value = List
    send_dictionary(dictionary)

def send_dictionary(dictionary):
    #sends dictionary to Command Receiver of
    #hardware processing layer
```

8.3 Camera Processing



8.3.1 Camera Data Receiver

Prologue

This module receives continuous feed from the camera data receiver, puts these camera data into appropriate data structure and passes it to position calculator.

Interfaces

Source	Sink	Input Data to sink	Output
Camera	Camera Data Receiver	Stream of String	Flag
Camera Data Receiver	Position Calculator	Image	Flag

External Data Dependencies

None

Internal Data Dependencies

String and Image

Pseudo Code

```
def getCameraFeed():
    while(True):
        #get Camera feed
        #snap an image every second and send it to
        #position calculator module
```

8.3.2 Position Calculator

Prologue

This modules gets camera data from camera data receiver, processes it and produces location co-ordinates of the rover's position and passes this co-ordinates to the transfer module.

Interfaces

Source	Sink	Input Data to sink	Output
Camera Data Receiver	Position Calculator	Image	Flag
Position Calculator	Transfer Module	A list of co-ordinates (tuple)	Flag

External Data Dependencies

None

Internal Data Dependencies

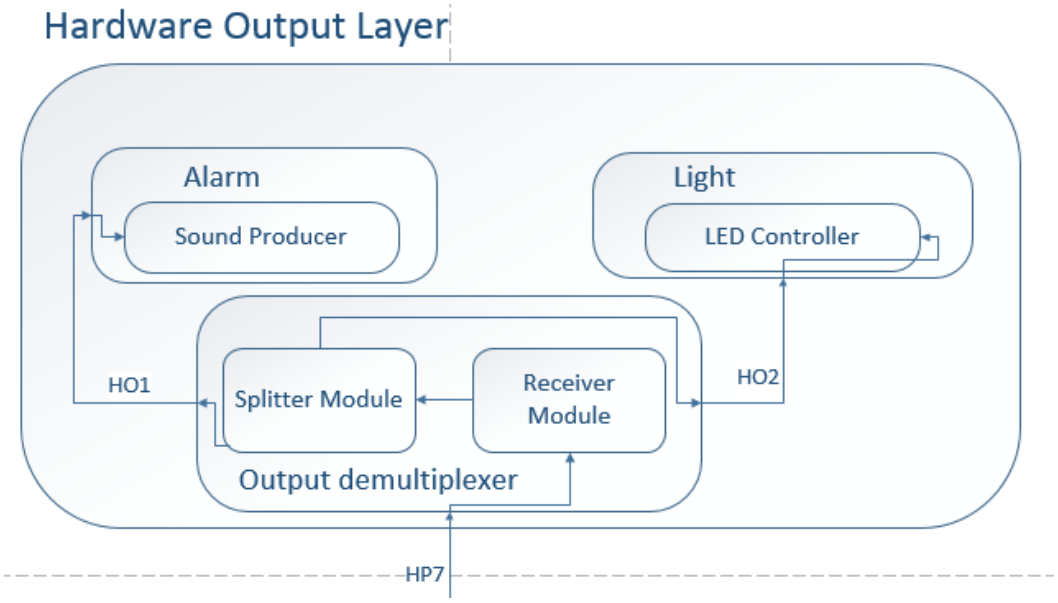
Image and List

Pseudo Code

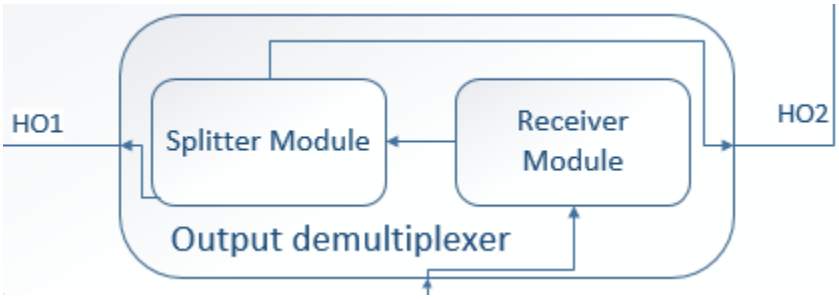
```
def getRoverPosition():
    #gets position co-ordinates from position calculator module
    sendRoverPositionToSIS(posX,posY)

def sendRoverPositionToSIS(posX, posY):
    #sends position co-ordinate to receiver module of #synchronization input subsystem.
```

9. 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.



9.1 Output De-multiplexer



9.1.1 Splitter Module

Prologue

This module receives string input from receiver module, splits it into two strings, one for sound alert and another for light alert.

Interfaces

Source	Sink	Input Data to sink	Output
Receiver Module	Splitter Module	Dictionary	Flag
Splitter Module	Sound Producer	String	Flag
Splitter Module	LED controller	String	Flag

External Data Dependencies

None

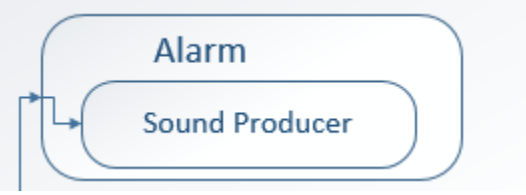
Internal Data Dependencies

Python Dictionary, data needed for alert generation

Pseudo Code

```
def parseAlert(dictionary):
    #get key to see what method to pass it to
    # value = data for command
    case: light
        lightAlert()
    case: alarm
        alarmAlert()
    case: stop
        stopAlert()
```

9.2 Alarm Subsystem



9.2.1 Sound Producer

Prologue

This module receives bit strings from splitter module and produces sound through the speaker connected to it.

Interfaces

Source	Sink	Input Data to sink	Output
Splitter Module	Sound Producer	String	Flag
Sound Producer	Speaker	Analog data	Flag

External Data Dependencies

None

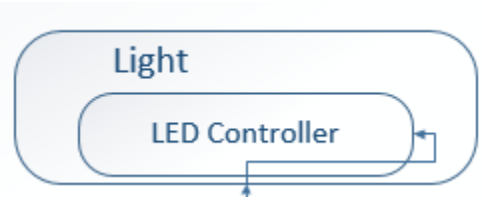
Internal Data Dependencies

String Data, Analog data to Speaker

Pseudo Code

```
def produceSoundAlert(string):
    #produce sound based on the passed string
```

9.3 Light Subsystem



9.3.1 LED controller

Prologue

This module receives bit strings from splitter module and generates light output on the LEDs connected based on that.

Interfaces

Source	Sink	Input Data to sink	Output
Splitter Module	LED controller	String	Flag
LED controller	LEDs	Analog data	Flag

External Data Dependencies

None

Internal Data Dependencies

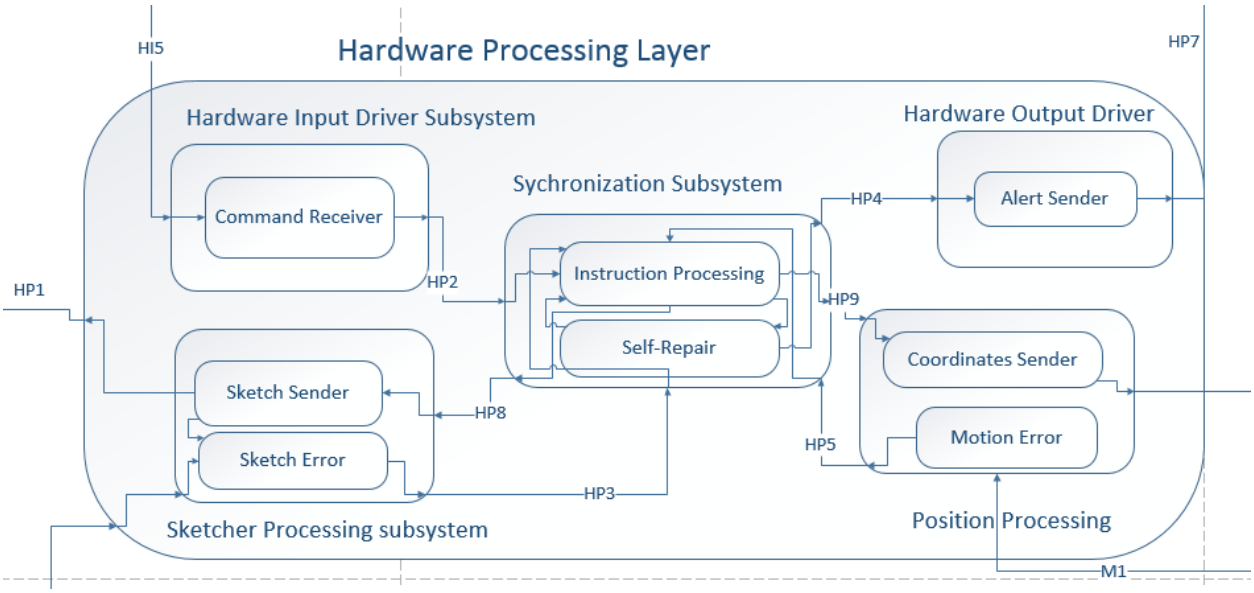
String Data and Analog data to LED

Pseudo Code

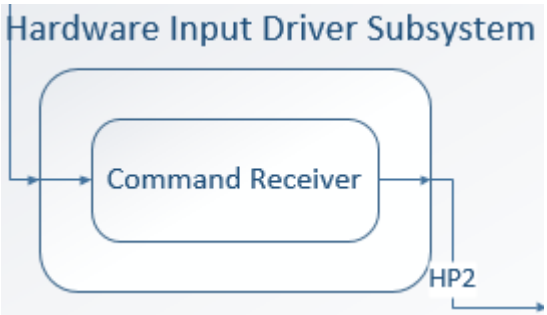
```
def generateLightOutput(String):
    #based on the passed string light alert in LED is
    #generated
```

10. 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.



10.1 Hardware Input Driver Subsystem



10.1.1 Command Receiver

Prologue:

This module will receive a python dictionary that will hold an Enum class as the key and a value pertaining to the key. Based on what the key is, will determine what method will be executed in this module.

Interfaces:

Source	Sink	Input Data to Sink	Output
Transfer Module	Command Receiver	Dictionary	List of parsed output

External Data Dependencies:

None

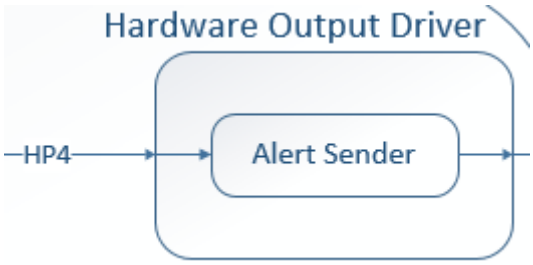
Internal Data Dependencies:

Python dictionary

Pseudo Code:

```
def parseInput(dictionary):
    #get key to see what method to pass it to
    # value = data for command
    case: command
        getCommand()
    case: instruction
        getList()
    case coordinates
        getCoordinates()
```

10.2Hardware Output Driver Subsystem



10.2.1 Alert Sender

Prologue:

This module will result raw alerts grouped into a list from the Synchronization Subsystem and it will format all of the alerts into a python dictionary where each key represents a different type of alert. The three alerts will be a stop alert, light alert, and sound alert.

Interfaces:

Source	Sink	Input Data to Sink	Output
Alert Sender	Receiver Module	List	Dictionary

External Data Dependencies:

None

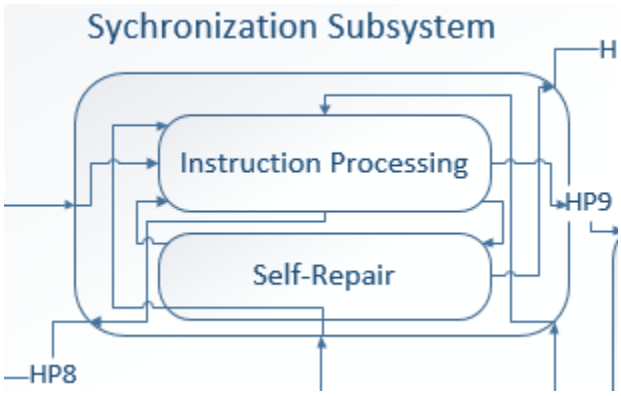
Internal Data Dependencies:

Python dictionary

Pseudo Code:

```
def sendOutputData(list):
    #for every element
        #parse errors to
        #{key: value}
        case: light
            distionary.Add{EnumAlert.Light: dataError}
        case: alarm
            distionary.Add{EnumAlert.Light: dataError}
        case: stop
            distionary.Add{EnumAlert.Light: dataError}
```

10.3 Synchronization Subsystem



10.3.1 Instruction Processing

Prologue:

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

Interfaces:

Source	Sink	Input Data to Sink	Output
Command Receiver	Instruction Processing	List	None
Sketch Error	Instruction Processing	List	None
Motion Error	Instruction Processing	List	None
Instruction Processing	Sketch Sender	List	List
Instruction Processing	Motion Sender	List	List
Instruction Processing	Alert Sender	List	Dictionary

External Data Description:

None

Internal Data Dependencies:

InstructionList and the current Coordinates

Pseudo Code:

```
def getNextPosition():
    NextCoordinates = InstructionList(Count).split()(InstructionList.len)
def incCount():
    #if the Sketcher has finished sketching and moving:
        #Count = Count + 1
        syncTime()
def syncTime():
    #log time
```

10.3.2 Self-Repair

Prologue:

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

Interfaces:

Source	Sink	Input Data to Sink	Output
Self-Repair	Alert Sender	List	Dictionary
Instruction Processing	Self-Repair	List	List

External Data Description:

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

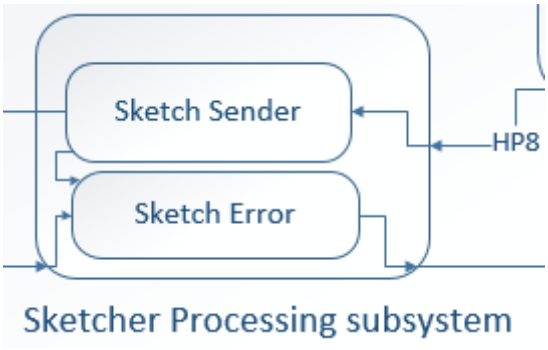
Internal Data Dependencies:

Error messages will trigger this module.

Pseudo Code:

```
def selfAlign():
    #if error from Motion layer == out of position:
        #send the same position
        sendCoordinates(current, Coordinates)
        #if error continues
            sendMotionError()
def sendRepairInstruction():
    #if error from Sketch layer == chalk out of position
        #resend instruction
        sendNextInstructions()
        #if error continues
            sendSketchError()
```

10.4 Sketcher Processing Subsystem



10.4.1 Sketch Sender

Prologue:

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

Interfaces:

Source	Sink	Input Data to Sink	Output
Sketch Sender	Sketcher Module	List	List
Instruction Processing	Sketch Sender	List	List

External Data Description:

None

Internal Data Dependencies:

InstructionList

Pseudo Code:

```
def sendNextInstructions(InstructionList):
    return InstructionList(Count)
def isInstructionSketched():
    #has the robot finished sketching
    return isFinished
```

10.4.2 Sketch Error

Prologue:

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

Interfaces:

Source	Sink	Input Data to Sink	Output
Sketch Error	Self-Repair	List	List
Sketcher Module	Sketch Error	List	List

External Data Description:

An error occurs with the Sketch Layer that triggers this module

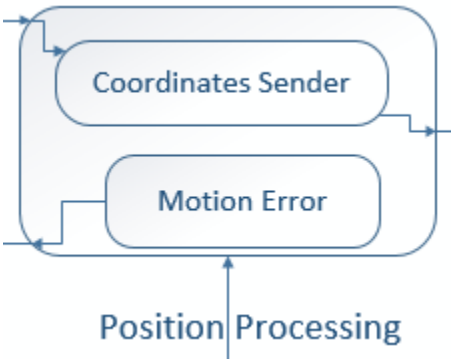
Internal Data Dependencies:

Error message sent as a string

Pseudo Code:

```
def sendSketchError():
    return error
```

10.5Position Processing Subsystem



10.5.1 Coordinates Sender

Prologue:

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

Interfaces:

Source	Sink	Input Data to Sink	Output
Coordinates Sender	Sketcher Module	List	List
Instruction Processing	Coordinates Sender	List	List

External Data Description:

None

Internal Data Dependencies:

Coordinates

Pseudo Code:

```
def sendCoordinates(current, destination):
    coordinateList = []
    coordinateList(0) = current
    coordinateList(1) = destination
    return coordinateList
def isRobotSet():
    #has the robot finished moving
    return isFinished
```

10.5.2 Motion Error

Prologue:

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

Interfaces:

Source	Sink	Input Data to Sink	Output
Motion Error	Self-Repair	List	List
Movement Module	Motion Error	List	List

External Data Description:

An error occurs with the Motion Layer that triggers this module

Internal Data Dependencies:

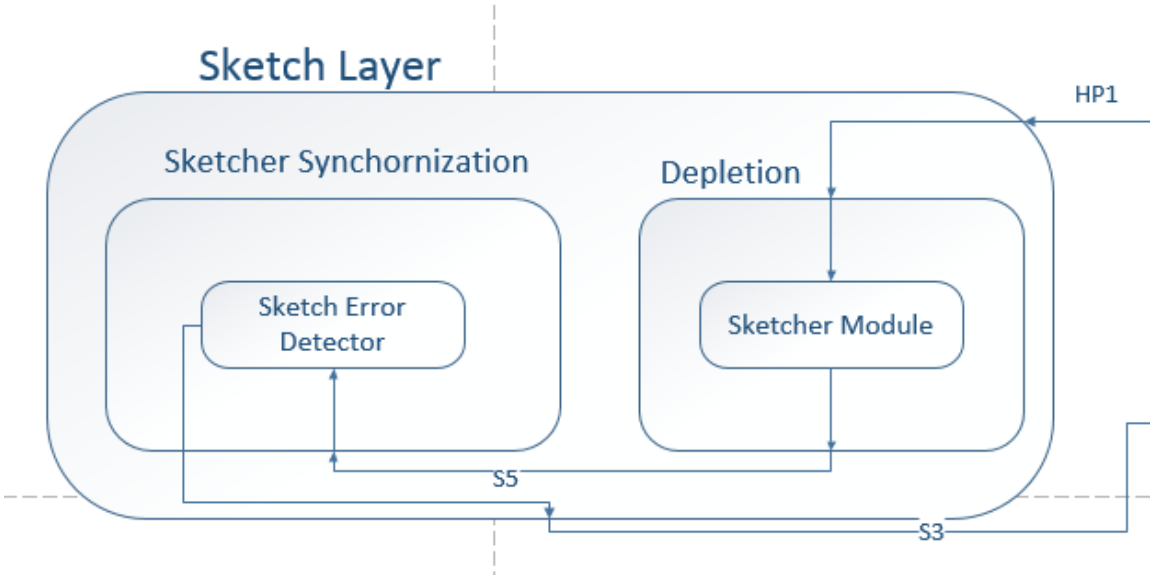
Error message sent as a string

Pseudo Code:

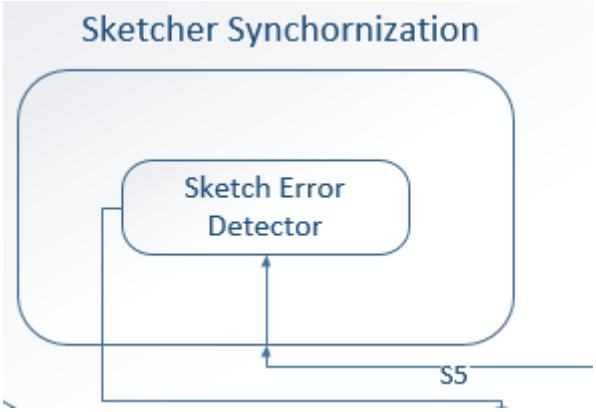
```
def sendMotionError():
    return error
```

11. 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.



11.1Sketcher Synchronization Subsystem



11.1.1 Sketch Error Detector

Prologue:

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

Interfaces:

Source	Sink	Input Data to Sink	Output
Sketcher Module	Sketch Error Detector	List	List
Sketch Error Detector	Sketch Error	List	List

External Data Description:

None

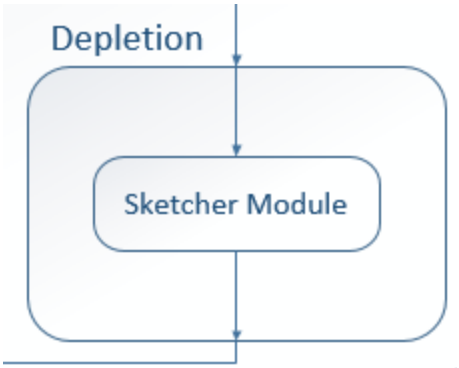
Internal Data Dependencies:

The instruction needed to be perform.

Pseudo Code:

```
def sendSketchError():
    return error
def sendIsDepleted():
    return error
```

11.2Depletion Subsystem



11.2.1 Sketcher Module

Prologue:

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

Interfaces:

Source	Sink	Input Data to Sink	Output
Sketcher Module	Sketch Error Detector	List	List
Sketch Sender	Sketcher Module	List	List

External Data Description:

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

Internal Data Dependencies:

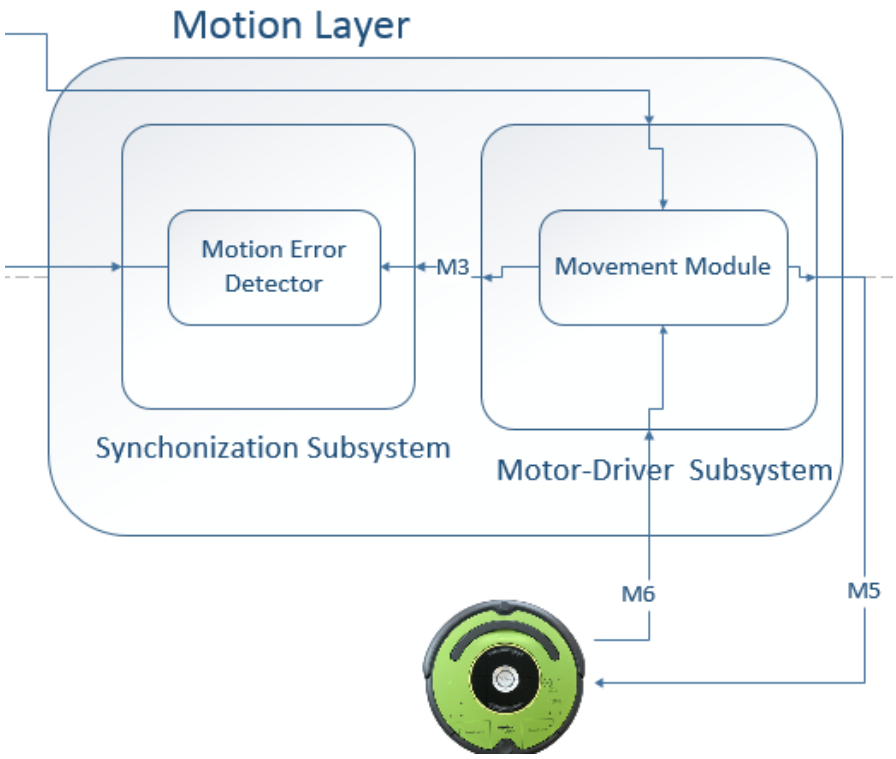
The instruction needed to be perform.

Pseudo Code:

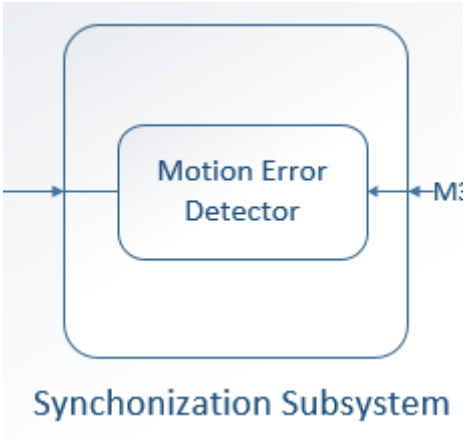
```
Degrees = 0
def sendActions(NextInstruction):
    InstructionListSplit = NextInstruction.split()
    if InstructionListSplit.len == 1:
        #translate the code into readable code to
        #interface with the stepper motor
        if Degrees % DEGREE_REQUIRES_PUSH:
            #send Push instruction to move the stepper motor
            #translate the code into readable code to
            #Degrees updates
        if Degree == DEGREE_DEPLETED
            isDepleteed = True
            return DEPLETEDERROR
    return None
```

12. 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.



12.1Motion Synchronization Subsystem



12.1.1 Motion Error Detector

Prologue:

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

Interfaces:

Source	Sink	Input Data to Sink	Output
Movement Module	Motion Error Detector	List	List
Motion Error Detector	Motion Error	List	List

External Data Description:

None

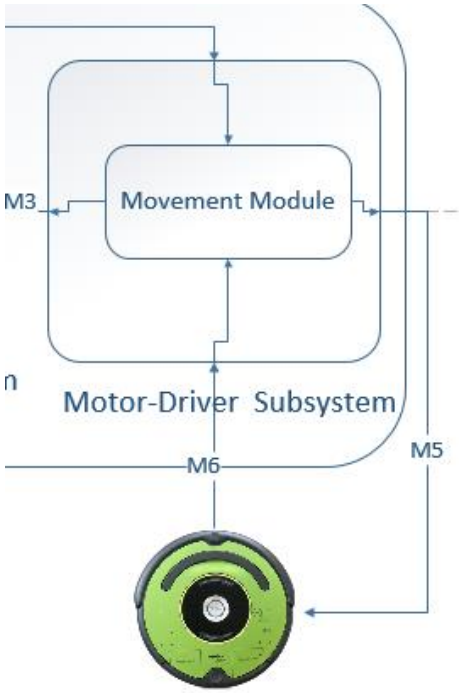
Internal Data Dependencies:

The instruction needed to be perform.

Pseudo Code:

```
def sendMotionError():
    return error
```

12.2Motion-Driver Subsystem



12.2.1 Movement Module

Prologue:

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

Interfaces:

Source	Sink	Input Data to Sink	Output
Movement Module	Motion Error Detector	List	List
Motion Sender	Movement Module	List	List

External Data Description:

The system will have to communicate directly with iCreate 2.

Internal Data Dependencies:

The instruction needed to be perform.

Pseudo Code:

```
def moveCreate(current, destination);
    errorList = []
    errorTurning = turnCreate(current, destination)
    errorMoving = moveTo(destination)
    if errorTurning != None:
        errorList.append(errorTurning)
    if errorMoving != None:
        errorList.append(errorMoving)
    return errorList;
def turnCreate(current, destination):
    #calculate angle of turnCreate
    #turn the robot
def moveTo(coordinates):
    #calculate distance to travel
    #travel to the desired position
```

13. Quality Assurance

The Quality Assurance section provides a detailed description of tests that each subsystem will partake and describe how the overall system will be tested during the development stage.

13.1 Unit Testing

To conduct the unit testing, a white box approach will be taken. In order to determine the testing success or failure, control data will be compared to the expected results. The following subsections will provide a brief description of how each module will be tested within each subsystem.

User Interface Layer

13.1.1 **File Request:** This module will request files from the user's computer to load for the user interface.

13.1.2 **File Handler:** The File Handler module will grab the image from the File Request module in order to send to the Presentation Subsystem for display.

13.1.3 **Display:** The Display module will take the image from the Image Processing module and display the image to the user.

13.1.4 **Image Cropper:** The Image Cropper module will receive data of the modified cropped image from the user.

13.1.5 **Color Picker:** The Color Picker module will receive data of the selected two colors from the user.

13.1.6 **Resize:** The Resize module will receive data of the newly sized image from the user.

Software Processing Layer

13.1.7 **Request Handler:** The Request Handler module will be able to accept inputs from the User Interface Layer and respond to different events provided by the user.

13.1.8 **Image Processing:** The Image Processing module receives data from the Request Handler module in order to generate data for the Data Packager Subsystem and Information Processing Subsystem

13.1.9 **Information Processing:** The Information Processing module receives data form the Image Processing module and generates data to return back to the Image Processing module.

13.1.10 **File Request Handler:** The File Request module will send files to be generated and received the generated files to return to the Information Processing module

13.1.11 **Query Module:** The Query module will receive data from the File Query module to transfer to the Information Processing module

Data Storage Layer

13.1.12 **Image Buffer:** The Image Buffer module will hold the image files until requested by the File Query module.

13.1.13 **File Query:** The File Query module will request the data file form the File Buffer or the image file from the Image Buffer module.

13.1.14 **File Store:** The File Store module will store the data file in the File Buffer module or the image file in the Image Buffer Module.

13.1.15 **File Buffer:** The File Buffer module will hold the data files until requested by the File Query Module.

Software Output Layer

13.1.16 **Transfer Module:** The Transfer Module will receive data from the Data Packer module and File Generator module to be transferred to the File Reader Subsystem

13.1.17 **Packer Module:** The Packer Module will receive data from the Image Processing module and return the packed data to the Transfer Module.

13.1.18 **Generator Module:** The File Generator Module will retrieve the required file from the File Request Handler module and convert the file into an instruction to transfer to the File Transfer module.

Hardware Input Layer

13.1.19 **File Receive:** The File Receive module receives the instruction file from the Software Output Layer, stores this file to local memory and sends the path file to the File Processor module.

13.1.20 **File Processor:** The File Processor module receives the path file, fetches the file from storage, extracts the information from the file and generates a list of instructions.

13.1.21 **Receiver Module:** The Receiver Module receives the instruction file from the transfer module and location coordinates from Camera Processing module

13.1.22 **Button Listener:** The Button Listener module listens for events of power and start up button.

13.1.23 **Transfer Module:** The Transfer module receives data from the Button Listener, File Processor, and Position Calculator module and sends sit to the Hardware Processing layer.

13.1.24 **Camera Data Receiver:** The Camera Data Receiver Module receives continuous feed of data from the markers and puts the data into an appropriate data structure for calculation.

13.1.25 **Position Calculator:** The Position Calculator module receives data from the Camera Data Receiver module and calculates the location coordinates of the Sidewalk Sketcher.

Hardware Processing Layer

13.1.26 Command Receiver: The Command Receiver module will define the list of elements to determine the positioning of the Sidewalk Sketcher and the remaining instructions to complete the image.

13.1.27 Sketch Sender: The Sketch Sender module will send instructions to the Sketch Layer

13.1.28 Sketch Error: The Sketch Error module will alert the Sidewalk Sketcher if there is an error during the sketch process.

13.1.29 Instruction Processing: The Instruction Processing module will be able to handle and process instructions.

13.1.30 Self-Repair: The Self-Repair module will attempt to readjust itself and reposition the chalk during the sketch process.

13.1.31 Alert Sender: The Alert Sender module will be able to receive alerts to be handled to notify the user.

13.1.32 Coordinate Sender: The Coordinate Sender will send current coordinates of the Sidewalk Sketcher.

13.1.33 Motion Error: The Motion Error module will send an error if the robot is arriving at the wrong location than specified.

Sketch Layer

13.1.34 Sketch Error Detector: The Sketch Error Detector module will send an error if the actions perform is incorrect.

13.1.35 Sketcher: The Sketcher module will execute the steps to sketch the image while maintaining the actions being performed.

Motion Layer

13.1.36 Motion Error Detector: The Motion Error Detector module will monitor the Sidewalk Sketcher to ensure there is no incorrect action executed.

13.1.37 Movement: The Movement module will receive instructions of coordinates and directions.

Hardware Output Layer

13.1.38 Splitter Module: The Splitter module receives string input from the Receiver Module, parses into two strings, one for the sound alert and another for the light alert.

13.1.39 Sound Producer: The Sound Producer module receives bit strings from the Splitter Module and produces sounds through the speakers.

13.1.40 LED Controller: The LED Controller module receives bit strings from the Splitter Module and generates light output to the LED bulbs.

13.2 Component Testing

Component testing will be conducted using a white box approach against control data to the expected results in order to determine the test is a success or failure. Provided below is the brief description of how each layer will be tested.

13.2.1 **User Interface Application:** The user must be able to navigate through the User Interface Application to upload an image from the Database Manager and edit the image. The image should be able to send through processing for the Sidewalk Sketcher to begin execution.

13.2.2 **Image Conversion:** The image file should be able to convert to a postscript file and the instructions must be defined for the Sidewalk Sketcher to execute the sketch.

13.2.3 **Alarm Notifications:** The Sidewalk Sketcher must be able to notify the user when the LED Controller or Sound Producer receives a request from the Error Handler.

13.2.4 **Sketch in Motion:** The Sidewalk Sketcher should be able to process the instructions given by the converted Image File to sketch the image in motion while maintain correct positioning.

13.3 Integration Testing

Each layer component will be tested individually and then integrated into the system. The User Interface Layer will be tested against the Software Input Layer, Data Storage Layer, and Software Processing Layer by sending input peripherals through the User Interface Layer. The processed input provided will then be sent to the Hardware Input Layer. The Hardware Input Layer will then be tested with the Hardware Output Layer and the Hardware Processing Layer by using the input from the Software Output Layer. The Hardware Processing Layer will then test with the Hardware Output Layer, Sketch Layer, and Motion Layer by using external inputs from the robot itself.

13.4 System Verification Testing

In order to test for System Verification, testing will be conducted using a white box approach. To ensure that the entire system is built as specified, the system must be broken down into individual test components, testing every module of the system. The individual components must be integrated into the system for a system test after each component has met the acceptance criteria descripted in the System Requirements Specification document.

13.5 Test Cases

Test Case	Expected Result
The user clicks the brose button in the user interface to select an image that the user wish to print	The user sees the image in the user interface
The user selects the resize button to adjust the dimensions of the image	The user sees the image resized
The user selects the crop button in the user interface and drags the mouse over the image to get the desired portion of the image	The user sees the portion of the image with the part the user cropped no longer on the screen
A user selects the color the wish the image to be printed	The user sees the color of the image modified
The user selects the image he wishes to load to the Sidewalk Sketcher	The user sees a confirmation message demonstrating that the image was successfully transferred to the hardware.

14. 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.

14.1 User Interface Layer and Software Output Layer

Requirement Number	Requirement Name	Cropper Module	Resize Module	Complete Module	Display Module	Color Module	File Browser Module	Packer Module	Generator Module
3.1	Sketch Image	x	x	x	x	x	x	x	x
3.2	Sidewalk Sketcher Multicolor			x	x	x			
3.7	Sketch Dimensions		x	x	x				
3.8	Chalk Switch								
8.1	Image Loading			x	x		x		
8.5	Push Chalk								
8.6	Finished Image								
8.7	Depletion Sensing								
8.8	Chalk Reload								
8.9	Positioning								

14.2 Software Processing Layer and Data Storage Layer

Requirement Number	Requirement Name	Request Handler Module	Image Processing Module	Information Processing Module	Query Module	File Query Module	File Store Module	Image Buffer Module	File Buffer Module
3.1	Sketch Image	x	x	x	x	x	x	x	x
3.2	Sidewalk Sketcher Multicolor	x	x						
3.7	Sketch Dimensions	x	x	x		x	x		
3.8	Chalk Switch								
8.1	Image Loading	x	x						
8.5	Push Chalk								
8.6	Finished Image								
8.7	Depletion Sensing								
8.8	Chalk Reload								
8.9	Positioning								

14.3 Hardware Input Layer and Hardware Output Layer

Requirement Number	Requirement Name	File Receiver	File Processor	Camera Data Receiver	Send Coordinates	Button Listener	Transfer Module	Sound Producer	Led Controller	Splitter Module
3.1	Sketch Image	x	x	x	x	x	x	x	x	x
3.2	Sidewalk Sketcher Multicolor									
3.7	Sketch Dimensions	x	x				x			
3.8	Chalk Switch									
8.1	Image Loading	x	x				x			
8.5	Push Chalk									
8.6	Finished Image								x	
8.7	Depletion Sensing									
8.8	Chalk Reload							x		
8.9	Positioning			x	x					

14.4 Sketch Layer, Hardware Processing Layer and Motion Layer

Requirement Number	Requirement Name	Sketch Error Detector	Sketcher Module	Command Receiver	Sketch Sender	Sketch Error	Instruction Processing	Self-Repair	Alert Sender	Coordinates Sender	Motion Error	Motion Error Detector	Movement Module
3.1	Sketch Image	x	x	x	x	x	x	x	x	x	x	x	x
3.2	Sidewalk Sketcher Multicolor												
3.7	Sketch Dimensions			x			x						
3.8	Chalk Switch												
8.1	Image Loading												
8.5	Push Chalk			x									
8.6	Finished Image												
8.7	Depletion Sensing			x									
8.8	Chalk Reload												
8.9	Positioning									x	x	x	x

15. Acceptance Criteria

The Acceptance Criteria describes the necessary steps to take in order to test the Sidewalk Sketcher for customer satisfaction. The plan includes the necessary packages and installation information, acceptance test plan, and the acceptance criteria.

15.1 Packaging and Installation

Sidewalk Sketcher will be packaged with the following items below:

- (1) iCreate 2
- (1) Raspberry Pi micro controller
- (1) Red LED bulb
- (1) Yellow LED bulb
- (1) Speaker
- (1) Charging Station
- (2) Color Chalk
- (3) Markers
- (1) USB 2.0 Cable
- (1) Installation CD
- (1) User Manual

The operating system running on the Raspberry Pi is Raspian. The Installation CD requires operating system of Windows 7 or higher.

15.2 Acceptance Testing

To meet the acceptance criteria, the acceptance testing will be conducted. These acceptance tests and overall plan will be further discussed in the System Test Plan documentation.

15.3 Acceptance Criteria

The Sidewalk Sketcher must meet the following criteria below in order to meet the development team and the customer's satisfaction. The following requirements are either critical or high priority.

- The Sidewalk Sketcher should provide a two color verification
- The Sidewalk Sketcher should provide a sketch of maximum dimensions
- The Sidewalk Sketcher should print the desired image

16. Appendix

16.1 Raspberry Pi 1 Model B+

Raspberry pi is a single-board computers about the size of a credit card. These are developed by Raspberry Pi Foundation with the intention of promoting the teaching of basic computer science in schools. There are various models of Raspberry Pi and B+ is the model of the pi we are using for this project. This model constitutes 700MHz Broadcom SoC processor, 512MB RAM, microUSB, 40-pins of GPIO, HDMI port, 4 USB ports, MicroSD card socket and a camera and DSI Display connector. The pi is used as the main computer in the rover.

16.2 Raspberry Pi Camera

The Raspberry Pi Camera Module is a custom designed add-on for Raspberry Pi. It attaches to Raspberry Pi by way of one of the two small sockets on the board upper surface. This interface uses the dedicated CSI interface, which was designed especially for interfacing to cameras. The CSI bus is capable of extremely high data rates, and it exclusively carries pixel data

16.3 Roomba® iCreate 2

Roomba iCreate 2 is a programmable robot manufactured by iRobot based on their Roomba vacuum cleaning platform. The robot is explicitly designed for robotics development and has wheels fo movement and sensors to help it perceive the surrounding.

16.4 360 Degrees Panoramic Lens

360 Degrees Panoramic lens is the panoramic lens manufactured by Kogeto that enables the Raspberry Pi camera to take a 360 view of the surrounding. This 360 view is important to locate location markers and based on the location markers rover's location is calculated. This lens can be used for taking both images and videos.