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

Patrol Crusaders

Neighborhood Patrol Drone

Architecture Design Specification

Team Members:

Ashesh Rana

Nabin Adhikari

Shreemila Rajwant

Arvind Swaminathan

**Last Updated: Tuesday, January 27, 2015**

**Document Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Revision Number | Revision Date | Description | Rationale |
| **0.1** | 11/30/2014 | Architectural Design Specification Initial Draft for submission. | Initial draft of ADS |
| **0.2** | 12/07/2014 | Review of ADS | Review of initial draft |
| **0.3** | 12/08/2014 | Review of ADS | Correction after peer review |
| **0.4** | 12/08/2014 | Review of ADS for Submission | Proof reading and adding OS dependencies |
| **1.0** | 12/09/2014 | ADS- Gate Review Submission | Fig. 2 corrected |

Table of Contents

[1. Introduction 7](#_Toc405899591)

[1.1 Purpose and Use 7](#_Toc405899592)

[1.2 Project Description 7](#_Toc405899593)

[1.3 Key Requirements 8](#_Toc405899594)

[2. Meta Architecture 10](#_Toc405899595)

[2.1 Architectural Vision 10](#_Toc405899596)

[2.2 Guidelines Principles 10](#_Toc405899597)

[2.3 Assumptions 11](#_Toc405899598)

[2.4 Trade offs 11](#_Toc405899599)

[3. Layer Definition 13](#_Toc405899600)

[3.1 Hardware Input Layer 13](#_Toc405899601)

[3.2 Application Input Layer 13](#_Toc405899602)

[3.3 Processing Input Layer 14](#_Toc405899603)

[3.4 Database Management Layer 14](#_Toc405899604)

[3.5 Presentation Layer 14](#_Toc405899605)

[3.6 Detailed Architectural Design 14](#_Toc405899606)

[4. Hardware Input Layer 15](#_Toc405899607)

[4.1 Hardware Event Handler Subsystem 15](#_Toc405899608)

[4.2 Video Input Handler Subsystem 17](#_Toc405899609)

[5. Application Input Layer 19](#_Toc405899610)

[5.1 Administrator Event Handler Subsystem 19](#_Toc405899611)

[5.2 User Event Handler Subsystem 20](#_Toc405899612)

[5.3 Data formatter 22](#_Toc405899613)

[6. Processing Layer 24](#_Toc405899614)

[6.1 Main Processor: 24](#_Toc405899615)

[6.2 Hardware Data Controller 26](#_Toc405899616)

[6.3 Application Data Controller 27](#_Toc405899617)

[6.4 API Handler 29](#_Toc405899618)

[7. Database Layer 31](#_Toc405899619)

[7.1 Database Manager 31](#_Toc405899620)

[7.2 Database 33](#_Toc405899621)

[8. Presentation Layer 35](#_Toc405899622)

[8.1 Output Controller Subsystem 35](#_Toc405899623)

[8.2 Output Formatter Subsystem 36](#_Toc405899624)

[9. Inter-Subsystem Data Flows 38](#_Toc405899625)

[9.1 Data Flow Diagram 38](#_Toc405899626)

[9.2 Data Flow Definitions 39](#_Toc405899627)

[9.3 Producer-Consumer Relationship Table 42](#_Toc405899628)

[10. Requirements Mapping 43](#_Toc405899629)

[11. Operating System Dependencies 44](#_Toc405899630)

[11.1 Application Input Layer 44](#_Toc405899631)

[11.2 Hardware Input Layer 44](#_Toc405899632)

[11.3 Processing Layer 44](#_Toc405899633)

[11.4 Database Layer 44](#_Toc405899634)

[11.5 Presentation Layer 45](#_Toc405899635)

[12. Testing Considerations 46](#_Toc405899636)

[12.1 Application Input Layer 46](#_Toc405899637)

[12.2 Hardware Input Layer 46](#_Toc405899638)

[12.3 Processing Layer 47](#_Toc405899639)

[12.4 Database Layer 47](#_Toc405899640)

[12.5 Presentation Layer 47](#_Toc405899641)

**List of Figures**

**Fig. 1 -** General Description 7

**Fig. 2 -** Layer Definition 13

**Fig. 3 -** Detail Architectural design 14

**Fig. 4 -** Hardware Event Handler 15

**Fig. 5 -** Video Input Handler 17

**Fig. 6 -** Administrator Event Handler 19

**Fig. 7 -** User Event Handler 20

**Fig. 8 -** Data Formatter 22

**Fig. 9 -** Main Processor 24

**Fig. 10 -** Hardware Data Controller 26

**Fig. 11 -** Application Data Controller 27

**Fig. 12 -** API Handler 29

**Fig. 13 -** Data Manager 31

**Fig. 14 -** Database 33

**Fig. 15 -** Output Controller 35

**Fig. 16 -** Output Formatter 36

**Fig. 17 -** Data Flow 38

**List of Tables**

**Table. 1 –** Key Requirements 9

**Table. 2 –** Inter-Layer Interface (Hardware Event Handler) 16

**Table. 3 –** Public Layer Interface (Hardware Event Handler) 16

**Table. 4 –** Inter-Layer Interface (Video Input Layer) 18

**Table. 5 –** Public Interface (Video Input Layer) 18

**Table. 6 –** Public Interface (Admin Event Handler) 20

**Table. 7 –** Public Interface (User Event Handler) 21

**Table. 8 –** Inter-Layer Interface (Data Formatter) 23

**Table. 9 –** Main Processor ( Interface) 25

**Table. 10 –** Inter-Layer Interface (Hardware Data Controller) 27

**Table. 11 –** Inter-Layer Interface (Application Data Controller) 28

**Table. 12 –** Inter-Layer Interface (API Handler) 29

**Table. 13 –** Inter-Layer Interface (Database Manager) 32

**Table. 14 –** Public Interface (Database) 34

**Table. 15 –** Inter-Layer Interface (Output Controller) 36

**Table. 16 –** Public Interface (Output Formatter) 37

**Table. 17 –** Dataflow 41

**Table. 18 –** Producer-Consumer 42

**Table. 19 –** Requirements Mapping 43

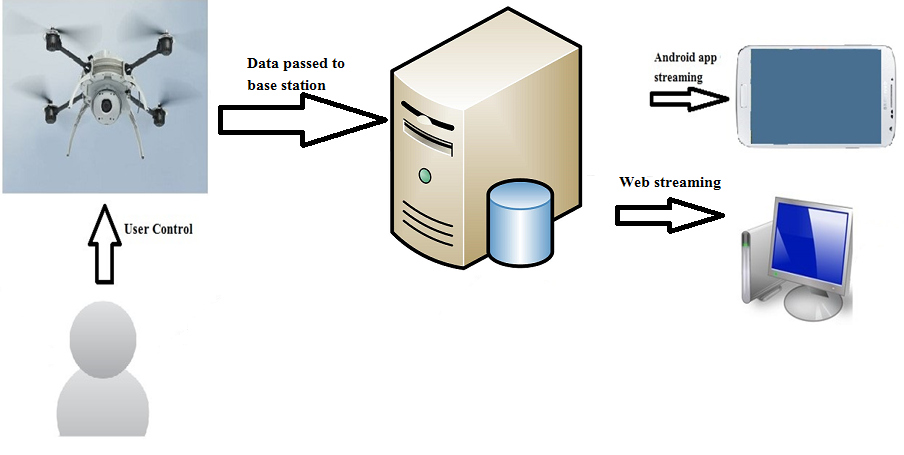
# Introduction

The Architectural design will describe the architecture for the Neighborhood Patrol Drone (NPD). This document will provide the system’s architecture, principle guidelines, layers, subsystems, dependencies, and testing considerations. This section will include product concept, scope, and key requirements.

## Purpose and Use

The purpose of the Neighborhood Patrol Drone is to provide added security to a neighborhood. It does so by patrolling over a predefined path and record video as well as take pictures. Once it has taken a picture or a video, real time image processing is conducted on the video to detect any irregular activities. The video is streamed to a server where the user can access it via a mobile phone using an Android application or from a website. The recorded videos are stored on the server for five days for reviewing purposes.

## Project Description



**Fig. 1 -** General Description

## Key Requirements

|  |  |  |
| --- | --- | --- |
| **No.** | **Requirements** | **Description** |
| 3.1 | Flight Trajectory and Height | NPD has the ability to fly forward and hover if necessary as directed in a specified trajectory and height. It also includes a mechanism to fly and land it safely when needed. |
| 3.2 | Capturing Live Video and images | NPD has a camera that records the videos and save it in the database for 5 days and be able to get images from those videos |
| 3.3 | Notification System | NPD have a capability to send an email notification and text message to the admin of the system and the users of the system if it senses any irregular activities while in operation. |
| 3.4 | Human Override Capability | The user can control the drone using a remote control and will have an option to switch back to auto pilot. |
| 3.5 | Energy detection capability | The system will detect the battery level. The level of the battery will be transmitted to the base station. |
| 3.7 | Registration | The admin can add new user to the system as a local user who can view live streaming videos, captured still images and videos, and get notification. |
| 3.8 | Log In/Log Out | The admin and user can log into the web application and android application to view the live streaming videos, captured still images and videos. |
| 3.9 | Battery Power | The vehicle must fly for at least half an hour before needing to recharge. |
| 4.2 | Android Application | Android application will be released on the Google Play as a free downloads to watch captured video and pictures for the users. |
| 4.3 | Web Application | A website will be created to access videos and pictures captured by the drone for the user |
| 4.4 | User manual | The final product will have a user manual printed in English. The user manual will include information on how to assemble the device as well as installation software for the system. It will have complete information about the specifications and features of the product. |
| 4.5 | Base station | The base station will consists of a computer and a running server which will be used for image processing and broadcasting videos and images to the web and android applications. The base station will also works as a final resting place and charging station for the drone. |
| 4.6 | Source Code Disk | A source code disk will be included in the package. This disk will contain all the source code for the system and digital version of user manual. |
| 7.3 | Troubleshooting Guide | The details of system configuration and maintenance procedure to the user as a CD ROM disk and written troubleshooting guide. It will consist of all the required material for hardware reconfiguration and software reconfiguration. The data recovery procedure is also provided in a guide if any data loss occurred during the products usage |
| 8.1 | System Operation | In order to operate the system, some legal issues need to be taken into consideration that is why the system will not be operated outside specified area. |
| 8.6 | FAA Regulation | To operate drone lawfully, user will have to obtain permission from FAA. |

**Table. 1 –** Key Requirements

# Meta Architecture

This section will describe the various design principles followed throughout the construction of the system’s architecture. It will also elaborate the architecture vision, guiding principles and the tradeoffs associated with the architecture design of Neighborhood Patrol Drone (NPD).

## Architectural Vision

NPD will be used for the patrolling of the neighborhood and alert will be send on real time to the neighborhood if any suspicious activity is found. This system consists of five layers i.e. Hardware Input Layer, Application Input Layer, Processing Layer, Database Management Layer, and Presentation Layer. Hardware Input Layer and Application Input Layer will receive and send the data to the Processing Layer. The Processing Layer will perform the business process on the received data and send it to the database management layer where the data is managed, formatted and saved in an appropriate location. The layer also prepares data for Presentation Layer. User views the system response in the Presentation Layer.

## Guidelines Principles

### Durability:

NPD will be used for the security system, so durability is one of the main guideline principles. Since the system is used for security, it should not break down while in operation. So, the hardware used in the system will be used from the reliable sources and the durability of the hardware will be taken into consideration.

### User Friendliness:

NPD will be built with ease-in-use principle. The remote controller for the drone will have conspicuous controls. User manual with trouble shooting guide will be created for user guidance for overall operation of the system. The user interface for the android app and the website will include proper naming convention on its actions and labels to make obvious about its functionality. It will be easy to set up for the user to put the system in use.

### Performance:

This is the major principle of NPD. The application will be highly responsive. As the system is concerned with security, it will send out the real time alert. The system will also be conservative on the latency of the application responses.

### Security:

NPD will be secure as it will store all the video and photo data in a secure location. Only authorized viewers will be able to access those stored images. User information will also be protected.

### Modularity:

NPD will be designed so that each part is independent of each other. If a user doesn’t meet some of the required assumptions, the system should still be configurable to perform a subset of built-in functions. In this system some more features can be added in the future as per the request of the application users independent of any components being used at prototyping level.

## Assumptions

In order to develop NPD, the team has made assumptions. The following are the list of the assumption made.

### Internet Access:

The neighborhood must have internet access so that a video captured by the drone can be sent to the base station real time. At the base station, the video will be processed and a notification can be sent if there are any suspicious activities.

### Environment:

It is assumed that there will be minimal interference of the environmental factors with the system.

### Knowledge:

We assume that team has enough knowledge of AndroidTM programming and web programming as well as drone technology to make this product. We assume that we will be able to finish the most critical requirements stated in the system requirements specifications document.

### Active email address:

We assume that all the user and administrator have the working email in order to use the system (mostly the application parts).

### FAA Regulations:

We assume that we can test product by obtaining the FAA license for flying the drone.

## Trade offs

Below are some tradeoffs that we will need to consider when designing the Neighborhood Patrol Drone architecture.

### Affordable Vs Performance

We decided to make our system affordable to the user so that the system can reach to the far denser user than limited users. However, we are not able to make the system perform to its full capacity because using high power equipment and software systems will raise the price of the total system. We can’t use night vision camera to detect objects in night, use powerful battery to power the system.

### Modularity Vs Scalability:

We decided to make our system very modular so that if user wants to change any components in future it will be easy to change. However, in doing so the expansion of the system hasn’t been taken into consideration at the moment. Changing cameras, drone and other general components can be done, but expanding the system to run on ISO or Linux Operating system haven’t been taken into account.

### Two Input Layer Vs Single Input Layer:

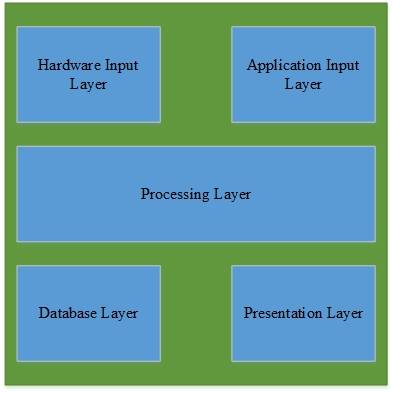
We decided to use two input layers (Hardware Input Layer and Application Input Layer) instead of having single input layer (Input Layer) because it helps to make the layers and functionality of the subsystems easy to differentiate and show the data flows properly in a recognizable way. It was considered, but modified for using two input layers in order to follow the guiding principle of modularity and simplicity.

### Presentation Layer Vs UI Layer:

The tradeoff of using the Presentation Layer vs. UI layer helped us to differentiate the confusion of user thinking that UI layer as both input layer for application and output layer of the system. To remove any confusion and follow top down data flow of the system we decided to use Presentation Layer as our output layer of the system and Application Input Layer as an independent input layer.

# Layer Definition

This section will describe the high level description of our system. Our system will consist of five separate layers i.e. Hardware Layer, Application Layer, Processing Layer, Database Management Layer and Presentation Layer. The current architecture is based on the guiding principles presented in section 2.2 above. A brief description of each layer will be provided in the following sections.



**Fig. 2 -** Layer Definition

## Hardware Input Layer

The purpose of this layer is to gather the data from the hardware component of the system such as drone (Pilot Module), video camera, battery, and controller. The data from the hardware parts will be collected by respective subsystem inside the Hardware Input Layer and will send it to the Processing Input Layer.

## Application Input Layer

The purpose of this layer is to gather the input data from user via web application and android app .This layer contains all the business logic such that the gathered data are formatted through respective subsystems and send it to the Processing Layer.

## Processing Input Layer

The purpose of this layer is to accept the formatted data sent from Hardware Input Layer and Application Input Layer, and analyze it. After the data has been analyzed it is then sent to a main processing component where it is further processed and the appropriate action will be generated. This layer also contains a component to deal with API requests to the Internet as well as their responses.

## Database Management Layer

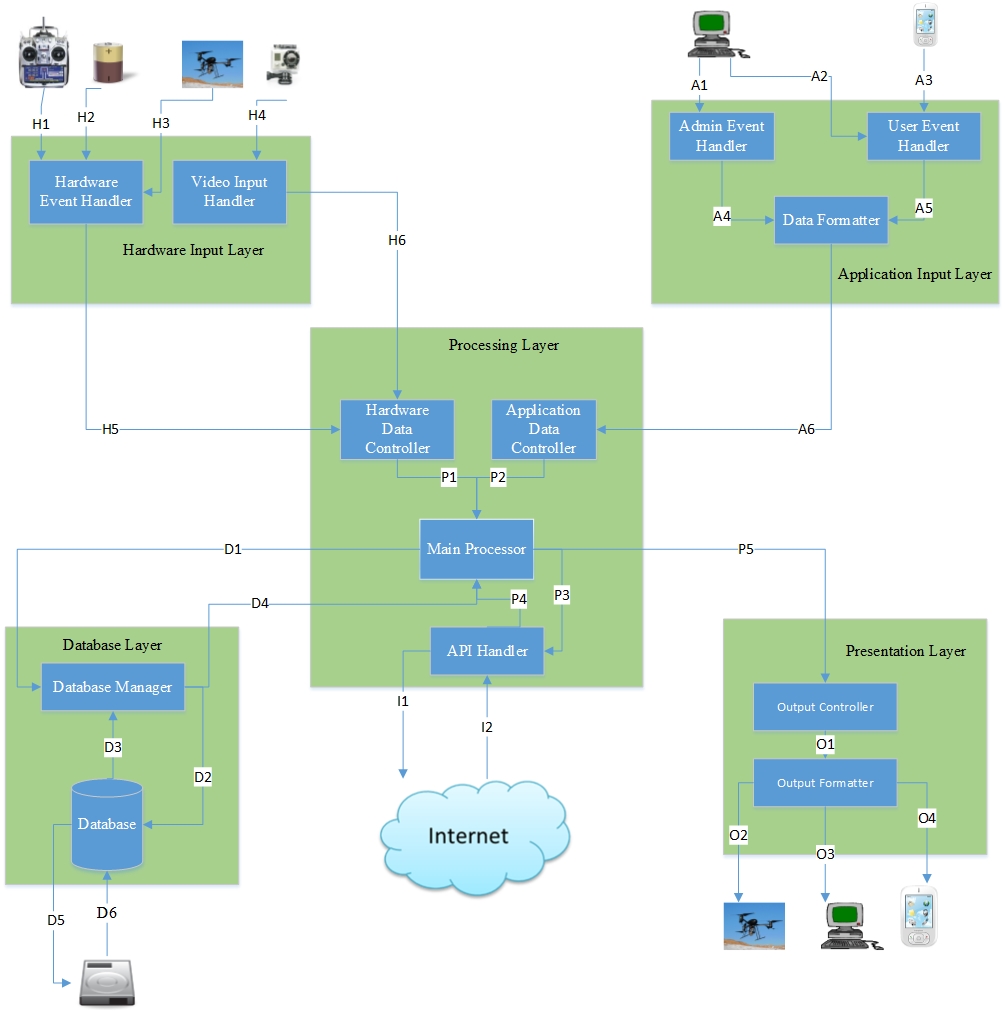
This layer will store the analyzed data received from the subsystem of processing input layer i.e. main processing component. The subsystems in this layer manage all data and store it to its designated place. After receiving a formatted query, the database sends formatted data back to the Processing Layer.

## Presentation Layer

This layer will display the information to the user via drone, admin home page or user home page. Data to be displayed is sent from the Processing Layer to the Presentation Layer and is then transmitted to the respective output formatter based on the data received.

## Detailed Architectural Design

The diagram below represents high level architectural design of NPD with five different layers, subsystem and the data flow between layers and subsystems.

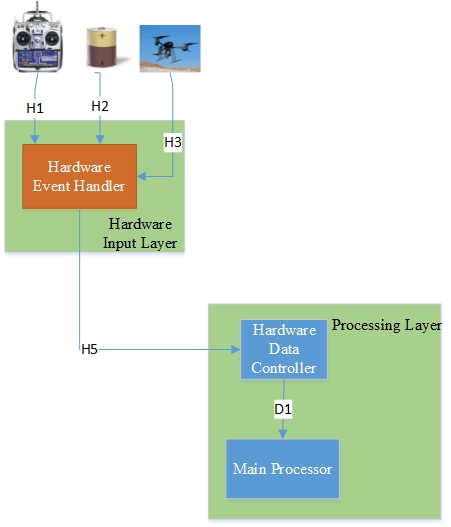
****

**Fig. 3 -** Detail Architectural design

# Hardware Input Layer

This section will elaborate about the Hardware Input Layer. This layer will take input from the hardware components of the system i.e. (drone Pilot Module), batteries, and controller. The subsystems of this system are described below along with their functions.

## Hardware Event Handler Subsystem



**Fig. 4 -** Hardware Event Handler

### General Description:

The hardware event handler receives input from the hardware components of the drone that include the controller and the battery. Once such an input is detected by the handler, the input is processed and a message is generated to be passed to the Data Processing Layer to take the required action.

### Assumption:

Assume that the input received by the sub system could be either analog or digital.

### Responsibilities:

The handler sends a message to the Data Processing Layer according to the input received. On the controller the user can:

* Switch state of autopilot
* Increase or decrease height of drone
* Change speed of drone

The battery level is measured using a voltmeter whose input is received by the handler. The handler can also receive an input from the drone (Pilot Module), if the connection is lost with the controller. The message is decoded and sent to the Data Processing Layer which will take an appropriate action after further processing.

### Inter-Layer Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Description** | **Information Required** | **Information Returned** |
| sendEvent | The hardware event handler will receive both analog and digital inputs from the controller, battery or the drone. These inputs are converted to a digital message that is sent to the Hardware Input Controller subsystem in the data processing layer. | Raw data from controller action, from voltmeter attached to battery or from the drone when connection is lost to controller. | Formatted digital message containing input data from hardware event. |

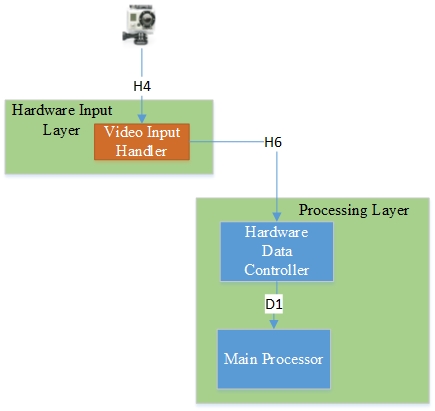
**Table. 2 –** Inter-Layer Interface (Hardware Event Handler)

### Public Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Description** | **Information Required** | **Information Returned** |
| controllerDataInput | The handler detects either digital or analog inputs from the controller and converts it into a digital message. | Raw controller input data. | Formatted digital message of controller input |
| batteryDataInput | The reading from the voltmeter attached to the battery will be converted into a message. | Voltmeter reading | Formatted data received from the voltmeter |
| droneSignalStatus | The drone sends an input to the handler if it loses contact with the controller. | Drone signal status | Formatted message containing the status of the drones connection with the controller |

**Table. 3 –** Public Layer Interface (Hardware Event Handler)

## Video Input Handler Subsystem



**Fig. 5 -** Video Input Handler

### General Description:

The Video Input Handler takes the video input from the camera present on the drone. The video is packaged and formatted to be sent to the Data Processing Layer. The video is received through a wireless video transmitter present on the drone.

### Assumptions:

Assume that the camera transmits the video to a range of at least 50 meters.

### Responsibilities:

The video Input Handler receives the video from the camera as an input. The video is then formatted. The formatted video is then sent to the Data Processing Layer for further processing.

### Inter-Layer Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Description** | **Information Required** | **Information Returned** |
| SendVideo | Sends the video after being formatted to the Hardware Input Controller subsystem in the data processing layer. | Formatted video feed | Formatted video from drone |

**Table. 4 –** Inter-Layer Interface (Video Input Layer)

### Public Interfaces:

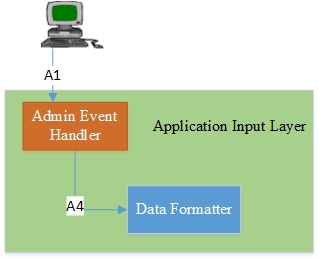
|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Description** | **Information Required** | **Information Returned** |
| RecieveVideo | Receives the raw video feed from the drone and formats it | Raw video feed from the drone | Formatted video feed from the camera on board the drone |

**Table. 5 –** Public Interface (Video Input Layer)

# Application Input Layer

This section will present detail description of the Application Input Layer. This layer takes the input from the two types of the users namely the Administrator (the admin) and the local users (the user) via the Web Application and the Android™ application. Each subsystem of this layer will be presented below.

## Administrator Event Handler Subsystem



**Fig. 6 -** Administrator Event Handler

### General Description:

The Admin Event Handler accepts inputs from the administrator through the web application. This subsystem will determine the type of messages needed to be delivered to the rest of the system in order to satisfy the request or an action received from the administrator.

### Assumptions:

All administrative interactions will be unique which will allow the Admin Event Handler to properly respond to the event.

### Responsibilities:

The Admin Event Handler will detect the type of action that has occurred through the admin user interface of the application. The Admin Event Handler will trigger a response message and sent it to the Input Data Formatter. The response will be unique to a particular type of actions.

### Inter-Layer Interfaces:

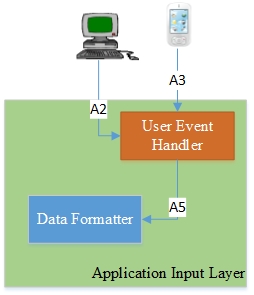
This subsystem does not interact with other layers.

### Public Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Description** | **Information required** | **Information Returned** |
| adminEventListener | The Admin Event Handler will wait to receive messages from the administrator through the web application. | None | Admin Input Data (New user information, login credentials, etc.) |

**Table. 6 –** Public Interface (Admin Event Handler)

## User Event Handler Subsystem



**Fig. 7 -** User Event Handler

### General Description:

The User Event Handler accepts inputs from the user through the Web Application and Android™ application. This subsystem determines the type of messages needed to be delivered to the rest of the system in order to satisfy any request or action received from the user.

### Assumptions:

All user interactions will be unique which will allow the User Event Handler to properly respond to the event.

### Responsibilities:

The User Event Handler will detect the type of action that has occurred through the user interface of the application. The User Event Handler will trigger a response message and send it to the Data Formatter. The response will be unique to a particular type of actions.

### Inter-Layer Interfaces:

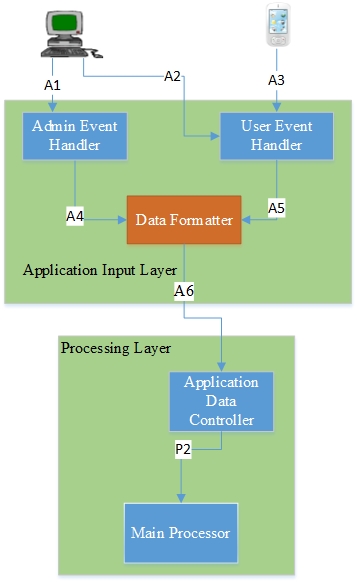
This subsystem does not interact with other layers.

### Public Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Description** | **Information required** | **Information Returned** |
| userEventListener | The User Event Handler will wait to receive messages from the user through the web application or Android™ Application. | None | User Input Data (login credentials) |

**Table. 7 –** Public Interface (User Event Handler)

## Data formatter



**Fig. 8 -** Data Formatter

### General Description

The Data Formatter accepts input from the Admin Event Handler and the User Event Handler. The data will be pre-processed by the formatter and converted into the format which is acceptable for the rest of the system. Then, the formatted data will be sent to the Application Data Controller which is necessary for creating a proper response for both Admin and the User’s interaction.

### Assumptions

The Data Formatter must be able to accept various forms of input from the Admin Event Handler and the User Event Handler.

### Responsibilities

The Data Formatter should accept the digital inputs in various forms and translate it into an acceptable form. The formatter will structure each data in such a manner that they will be easy to process.

### Inter-Layer Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Description** | **Information required** | **Information Returned** |
| sendFormattedData | The Data Formatter will send formatted data to the Application Data Controller in the Processing Layer. | Event data generated by the Android™ Application and the Web Application. | None |

**Table. 8 –** Inter-Layer Interface (Data Formatter)

### Public Interfaces:

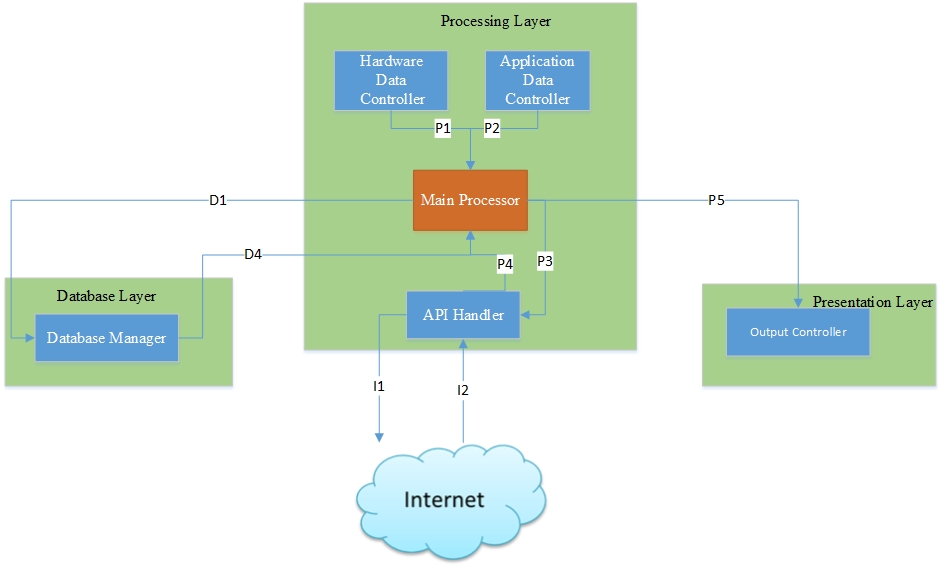
This subsystem does not interact externally.

# 

# Processing Layer

This section will elaborate about the working of processing layer. This layer will receive input from both hardware layer and application layer. The detail description of this layer is explained below.

## Main Processor:



**Fig. 9 -** Main Processor

### General Description:

The Main Processor accepts input from Application Data Formatter, Hardware Input Layer Controller. The Main Processor Unit will also be able to handle data transfer request over internet. It also communicates with Database Management Layer and saves and retrieves data and also sends data to Presentation Layers as output.

### Assumptions:

The process will be able to handle any given amount task at any given time.

### Responsibilities:

Main Processor is the main component among all other component. It handles all the data to and from other layers. The sources of input are Hardware Event Handler, Video Input Handler, and Data Formatter. The data that is passed into this module is pre-processed, so this module will make decision whether to communicate with Data Storage Management Layer or API Handler based on the data that has been passed into it. This module also communicates with presentation layer in order to produce output of the request input.

### Inter - Layer Interfaces:

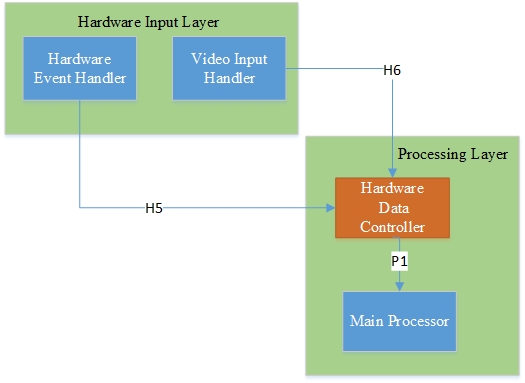
|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Description** | **Information Required** | **Information Returned** |
| saveData | The main processor will send data to the database layer that needs to be saved. | Videos and user information that needs to be saved. | None |
| sendData | The main processor will send data to the presentation layer for output. | Videos and user information that needs to be output. | None |

**Table. 9 –** Main Processor ( Interface)

### Public Interfaces:

This subsystem does not interact externally.

## Hardware Data Controller



**Fig. 10 -** Hardware Data Controller

### General Description:

The Hardware Data Controller gets information from Hardware Event Handler, Video Input Handler. The data received is divided and then identified. The Hardware Layer will collect the data and determines the next step that need to be taken. The collected data is sent to Main Processor for further processing and from there appropriate action will be taken so as to meet the request made by the hardware.

### Assumptions:

Hardware Data controller assumes that the data sent to it is already formatted so as to meet the criteria set by the Hardware Data controller which enable it to easily distinguish the type the data that is being sent to it by the hardware.

### Responsibilities:

The responsibilities of the subsystem is to gather the information sent to it from Hardware Event Handler, Video Input Handler and identify the action that the system needs to provide in order to satisfy the request of the hardware. The data then will be sent to Main Processor which corresponds to the fully formatted data and generates a response.

### Inter-Layer Interfaces:

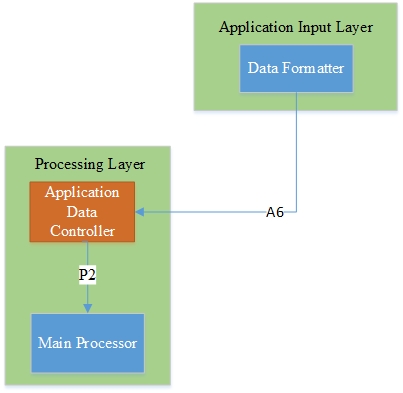
|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Description** | **Information Required** | **Information Returned** |
| hardwareDatahandler | The Hardware Data Controller will identify the type of data that is being passed to it from hardware event handler. | None | Identified hardware data. |

**Table. 10 –** Inter-Layer Interface (Hardware Data Controller)

### Public Interfaces:

This subsystem does not interact externally.

## Application Data Controller



**Fig. 11 -** Application Data Controller

### General Description:

The Application Data Controller will get information from Data Formatter and the received data will be separated and identified according to the user request. The identified data will be sent to the Processing Layer from where an appropriate action will be taken to satisfy the request of the user.

### Assumptions:

Application Data Controller assumes that the data received from the Data Formatter is formatted according to the criteria set by Application Data Controller so as to easily distinguish the request made by the user.

### Responsibilities:

The responsibilities of the subsystem is to gather the information sent to it from data formatter and identify the action that the system need to provide in order to satisfy the request of the user. The data then will be sent to Main Processor which will then corresponds to the fully formatted data and generates a response.

### Inter-Layer Interface:

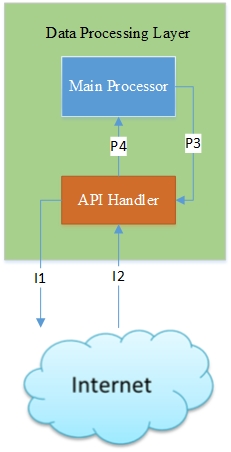
|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Description** | **Information Required** | **Information Returned** |
| appDataController | The application data controller will get the preprocessed data from the application layer. | None | Analyzed data from application layer. |

**Table. 11 –** Inter-Layer Interface (Application Data Controller)

### Public Interfaces:

This subsystem does not interact externally.

## API Handler



**Fig. 12 -** API Handler

### General Description:

The API Handler acts as an interface between the Main Processor and any external API interactions. The API Handler will handle any request made by user that needs access to the network in order to fetch data that is required by the Processing Layer.

### Assumptions:

It is assumed that the API Handler will be able to handle any kind of request that is made by the user and respond properly.

### Responsibilities:

The API Handler links Processing Layer to the data that is not present in the system. The Main Processor will send request to the API Handler and the handler will gather the correct information through the network and then will send it to the Main Processor

### Inter – Layer Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Description** | **Information Required** | **Information Returned** |
| apiHandler | Sends data over the internet for API request. | Information required for a request. | Information requested. |

**Table. 12 –** Inter-Layer Interface (API Handler)

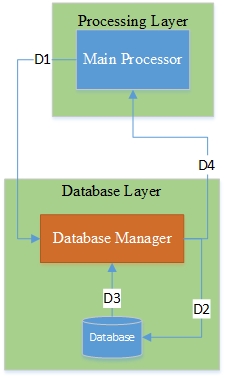
### Public Interfaces:

This subsystem does not interact externally.

# Database Layer

This section describes the subsystem of database management layer. This layer receives the data from the subsystem of processing layer .i.e. main processor. Database management layer has subsystem and they are database manager and database

## Database Manager



**Fig. 13 -** Data Manager

### General Description:

Database Manager Subsystem retrieves the data from Processing Layer and separates the business logic and stores it in the database. It can also authenticate the user account by validating with the account saved in Database Subsystem. The Database Manager Subsystem is the only component that will have the access to the Database .The Main Processor from Processing Layer has to call the Database Manager Subsystem in order to retrieve data from Database.

### Assumptions:

The Database Manager Subsystem will assume that the data being provided is correct and complete.

### Responsibilities:

This layer will be responsible to accept data from the Processing Layer, format it and store it in the appropriate location. It will also be responsible for locating and retrieving stored data from the Database Subsystem, changing data format if necessary, and passing it to the Main Processor Subsystem of Processing Layer in the Data Processing Layer.

### Inter – Layer Interfaces:

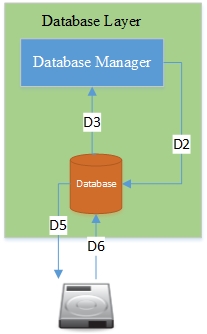
|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Description** | **Information Required** | **Information Returned** |
| saveData | Saves data to the appropriate location sent by the main processor. | Data to be saved, types of data (video, picture, battery status ad user information). | Success or Failure |
| queryData | Received data from main processor are formatted and located to appropriate location. | Data to be retrieved, types of data (video, pictures, battery status and user information). | Success or Failure |
| validateUser | Validate the user account of the user using the system applications. | User name and password. | Success or Failure |

**Table. 13 –** Inter-Layer Interface (Database Manager)

### Public Interfaces:

This subsystem does not interact externally.

## Database



**Fig. 14 -** Database

### General Description:

The Database Subsystem will receives all the formatted data from the Database Manager and such data will be stored in the appropriate location in a secure manner. All the communications with the Database Subsystem will be done via database Manager Subsystem.

### Assumption:

None.

### Responsibilities:

Database subsystem will be responsible to store all the data of the system. Database manager subsystem will send the formatted data such as pictures, video, and user’s account information to the Database Subsystem and will retrieve all the data from the Database Subsystem by sending queries.

### Inter-layer Interfaces:

This subsystem does not interact with other layers.

### Public Interfaces:

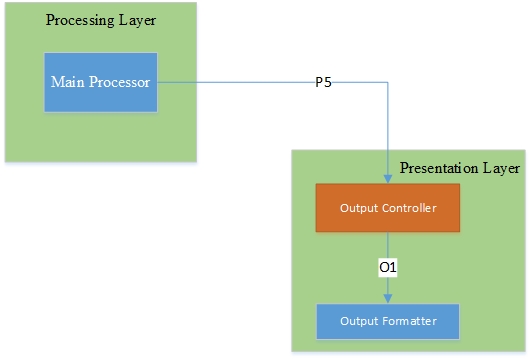
|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Information Required | Information Returned |
| storeData | Database stores the formatted data on the external hard drive. | Formatted data to be store | None |
| RetrieveData | Database subsystem can retrieve data from external hard drive on the request of database manager. | Queries | Success of failure |

**Table. 14 –** Public Interface (Database)

# Presentation Layer

This section will present detail description of the Presentation Layer. This layer gets input from Processing Layer which will be processed and presented as an output.

## Output Controller Subsystem



**Fig. 15 -** Output Controller

### General Description:

The Output Controller receives all the data that needs to be outputted to the user from the Processing Layer. It then analyzes the type of output, either hardware or user application, and then routes it appropriately through the Output formatter Subsystem.

### Assumptions:

The Processing of the output has already been done completely in the Processing Layer and there exists an output location for each input into the Output Controller Subsystem.

### Responsibilities:

The Output Controller receives the data to be outputted to the user from the Processing Layer. Such an output can be either to a user application (android and web applications) or to the drone itself (e.g. change height of the drone). The controller then sorts the data according to the required output and then routes it to either the drone or the application through the Output Formatter. Output to the drone is sorted in digital format.

### Inter-Layer Interfaces:

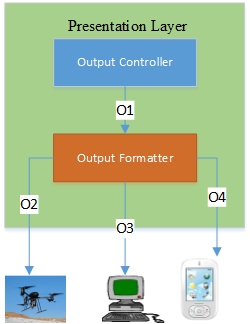
|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Description** | **Information Required** | **Information Returned** |
| outputListener | Waits for data from the processing layer. This data needs to be outputted to the user application or to the hardware. | None | Data to be outputted to the user. |

**Table. 15 –** Inter-Layer Interface (Output Controller)

### Public Interfaces:

This subsystem does not interact externally.

## Output Formatter Subsystem



**Fig. 16 -** Output Formatter

### General Description:

The output formatter’s job is to receive the output data from the Output Controller, which varies with destination, and format the output for that specific destination. The destination can be either the android application, web application or the drone itself.

### Assumptions:

The output device is available.

### Responsibilities:

The Output Formatter receives the data to be outputted from the Output Controller. This data is specific to the output destination. The Output Formatter then formats the output according the required destination and is then forwarded to the destination.

### Inter-Layer Interfaces:

This subsystem not interact with other layers.

### Public Interfaces:

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Description** | **Information Required** | **Information Returned** |
| outputAndroidApp | Formats the data specifically to be outputted to the android application. | Data to be outputted to the user | None |
| outputWebApp | Formats the data specifically to be outputted to the web application. | Data to be outputted to the user | None |
| outputDrone | Formats the data specifically to be outputted to the drone. | Data to be outputted to the drone | None |

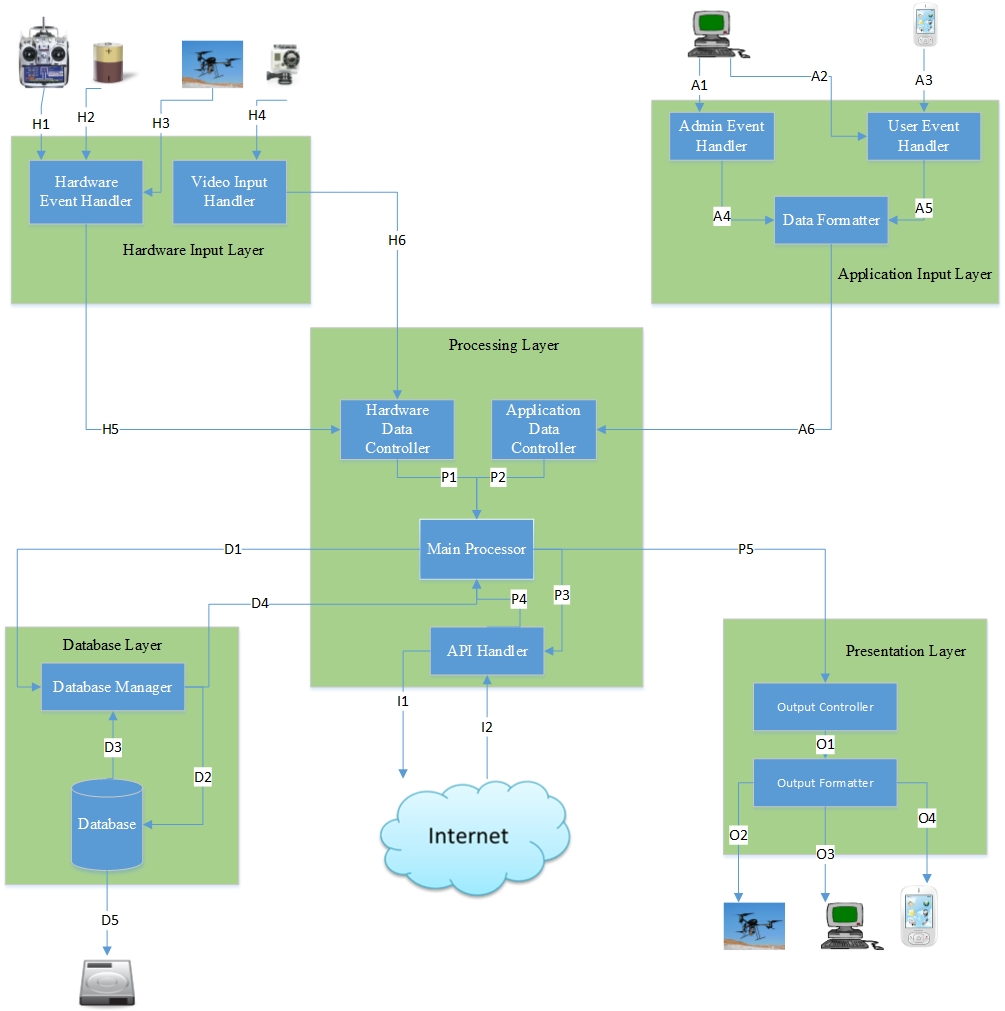
**Table. 16 –** Public Interface (Output Formatter)

# Inter-Subsystem Data Flows

In this section the data flows for the architectural layers of the NPD will be presented. This section will show each layer and their logical subsystems, as well as how they interact with each other. The flow of data between each subsystem which are either from same layer or the different layers will be discussed. Every logical subsystem of the architectural layer has an important function in the layer it resides. Therefore, the way that data flows between them is very essential for NPD to function properly.

## Data Flow Diagram

The diagram presented below will give a representation of the architecture of the NPD showing each layer, subsystem, of the NPD as well as the data flows between each layers of the NPD. This will easily demonstrate the flow between layers and how each layer interacts with other layer to correctly function. This data flow diagram will makes it easier to understand the functions of each subsystem and how they work on a higher level.



**Fig. 17 -** Data Flow

## Data Flow Definitions

The following table will show the breakdown of the data flows in the architectural data flow diagram presented in section 9.1. This table will show each of the data flows, what the data is, and where the data is going. The table will also show the information that is passed between each of the subsystems so that it can give a better understanding of the system and not only the layers of the system.

|  |  |  |  |
| --- | --- | --- | --- |
| **Data**  **Element ID** | **Description** | **Source** | **Sink** |
| **A1** | Admin enters any input into the Web application or requests functionality from the application. | Admin | Admin Event Handler |
| **A2** | User enters any input into the web application or requests functionality from the application. | User | User Event Handler |
| **A3** | User enters any input into the Android™ application or requests functionality from the application | User | User Event Handler |
| **A4** | An event from the admin is processed and sent to the Data Formatter to format the request into information usable by another subsystem. | Admin Event Handler | Data Formatter |
| **A5** | An event from the user is processed and sent to the Data Formatter to format the request into information usable by another subsystem. | User Event Handler | Data Formatter |
| **A6** | After the message has been formatted it is passed along the Application Data Controller to find out what the data is and what it is being used for. | Data Formatter | Application Data Controller |
| **D1** | The fully processed information has will be passed to the Database Manager in order to get ready to be stored. | Main Processor | Database Manager |
| **D2** | The formatted data will be passed to the Database which will store the information. | Database Manager | Database |
| **D3** | The needed stored data is requested from the Database and is formatted into data that is readable by the system. | Database | Database Manager |
| **D4** | When stored data is requested it is passed from the Database Manager to the Main Processor. | Database Manager | Main Processor |
| **D5** | The processed data has been stored on hard drive which is the storage unit for the NPD. | Database | Hard Disk |
| **D6** | Information requested is retrieved by the Database from Hard Disk. | Hard Disk | Database |
| **H1** | The Hardware Event Handler will listen the input in the form of a button press and then pass the appropriate information. | Controller | Hardware Event Handler |
| **H2** | The Hardware Event Handler will get signal from battery about its power. | Battery | Hardware Event Handler |
| **H3** | The Hardware Event Handler will get information about drone movement from the Drone (Pilot Module). | Drone(Pilot Module) | Hardware Event Handler |
| **H4** | The Video Input Handler will get video frame as input from the video camera. | Video Camera | Video Input Handler |
| **H5** | The hardware event that has been occurred is relayed to the Hardware Input Controller for action to be taken by the system. | Hardware Event Handler | Hardware Input Controller |
| **H6** | The video event that has been occurred is relayed to the Hardware Input Controller for action to be taken by the system. | Video Input Handler | Hardware Input Controller |
| **I1** | When needed, the API Request Handler will make a call to the Internet in order to get requested information. | API Handler | Internet |
| **I2** | Information from the Internet is passed to the NPD’s system by using APIs which will be interpreted and formatted by API Handler so the system can use them. | Internet | API Handler |
| **P1** | Once hardware data has been received and analyzed, it is passed to the Main Processor for performing the requested action. | Hardware Input Controller | Main Processor |
| **P2** | Once the application data has been received and analyzed, it is passed to the Main Processor for performing the requested action. | Application Data Controller | Main Processor |
| **P3** | The Main Processor makes a call to the API Handler in order to get the external information that is needed for performing the requested action. | Main Processor | API Handler |
| **P4** | Data received from the APIs will be parsed and then sent for processing. | API Handler | Main Processor |
| **P5** | The processed information will be passed to the Output Controller so that it can be output. | Main Processor | Output Controller |
| **O1** | The output data is passed to the Output Formatter so that it can be put in the correct format. | Output Controller | Output Formatter |
| **O2** | The formatted data is passed to the Drone (Pilot Module), so that it can fly. | Output Formatter | Drone(Pilot Module) |
| **O3** | The formatted display data is passed to the Admin Screen. | Output Formatter | Admin Screen |
| **O4** | The formatted display data is passed to the User Screen. | Output Formatter | User Screen |

**Table. 17 –** Dataflow

## Producer-Consumer Relationship Table

The following table will show the relationships between data flows in a different way and helps to give a different perspective from producer consumer relationships.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Consumer | Admin | User | Drone(Pilot Module) | Batteries | Video Camera | Controller | Hardware Event Handler | Video Input Handler | Admin Event Handler | User Event Handler | Data Formatter | Hardware Input Controller | Application Data Controller | Main Processor | API Handler | Database Manager | Database | Hard Disk | Output Controller | Output Formatter | Admin. Screen | User Screen | Internet |
| Producer |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Admin |  |  |  |  |  |  |  |  |  | **A1** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| User |  |  |  |  |  |  |  |  |  |  | **A2 A3** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Drone(Pilot Module) |  |  |  |  |  |  |  | **H3** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Batteries |  |  |  |  |  |  |  | **H2** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Video Camera |  |  |  |  |  |  |  |  | **H4** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Controller |  |  |  |  |  |  |  | **H1** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Hardware Event Handler |  |  |  |  |  |  |  |  |  |  |  |  | **H5** |  |  |  |  |  |  |  |  |  |  |  |
| Video Input Handler |  |  |  |  |  |  |  |  |  |  |  |  | **H6** |  |  |  |  |  |  |  |  |  |  |  |
| Admin Event Handler |  |  |  |  |  |  |  |  |  |  |  | **A4** |  |  |  |  |  |  |  |  |  |  |  |  |
| User Event Handler |  |  |  |  |  |  |  |  |  |  |  | **A5** |  |  |  |  |  |  |  |  |  |  |  |  |
| Data Formatter |  |  |  |  |  |  |  |  |  |  |  |  |  | **A6** |  |  |  |  |  |  |  |  |  |  |
| Hardware Input Controller |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **P1** |  |  |  |  |  |  |  |  |  |
| Application Data Controller |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **P2** |  |  |  |  |  |  |  |  |  |
| Main Processor |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **P3** | **D1** |  |  | **P5** |  |  |  |  |
| API Handler |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **P4** |  |  |  |  |  |  |  |  | **I1** |
| Database Manager |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **D4** |  |  | **D2** |  |  |  |  |  |  |
| Database |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **D3** |  | **D5** |  |  |  |  |  |
| Hard Disk |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **D6** |  |  |  |  |  |  |
| Output Controller |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **O1** |  |  |  |
| Output Formatter |  |  |  | **O2** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **O3** | **O4** |  |
| Admin. Screen |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| User Screen |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Internet |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **I2** |  |  |  |  |  |  |  |  |

**Table. 18 –** Producer-Consumer

# Requirements Mapping

The requirements mapping takes every top level requirement of our system and maps it to the layer in which that requirement is met. Thus it shows how data flows for each of the requirements. This mapping also allows us to identify test cases for different functionality for each layer.

The processing layer receives every input data at some point of time. All the data inputted by the user definitely comes out of the system through the Presentation Layer. Therefore, the Processing Layer and Presentation Layer play a major part in satisfying most of the requirements.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No.** | **Requirements** | **Application Input** | **Hardware Input** | **Processing** | **Database** | **Presentation** |
| **3.1** | Flight Trajectory and Height |  | **√** | **√** |  |  |
| **3.2** | Capturing Live Video and images |  | **√** |  |  |  |
| **3.3** | Notification System |  |  | **√** |  | **√** |
| **3.4** | Human Override Capability |  | **√** | **√** |  | **√** |
| **3.5** | Energy Detection Capability |  | **√** | **√** |  |  |
| **3.7** | Registration | **√** |  | **√** |  | **√** |
| **3.8** | Log In/Log Out | **√** |  | **√** |  | **√** |
| **3.9** | Battery Power |  | **√** | **√** |  | **√** |
| **4.5** | Base station | **√** | **√** | **√** | **√** | **√** |

**Table. 19 –** Requirements Mapping

# Operating System Dependencies

This section will elaborate on libraries, operating systems, and APIs which will be used by each layer. These topics will be a high level description as no specific hardware/software components have been decided by the development team.

## Application Input Layer

The web application will be dependent on the versions of the web browsers like Google Chrome(version 23.0.1271.95), Mozilla(version 31.2.0), and Internet Explorer(version 10.0) while the Android™ application will be dependent on the Android™ operating system version 4.03 or newer. There may be additional Android™ specific libraries which will help with the transfer of information between components on the Android™ system.

## Hardware Input Layer

The Hardware Input Layer will be dependent on an embedded operating system, Tiny OS, Linux, etc. The hardware peripherals will be dependent on analog to digital conversion libraries if the outputs are not already in a digital format. The libraries used will be decided upon the operating system of the controller, video camera, drone (Pilot Module), and batteries. The hardware peripherals will also depend upon the operating system used by the drone (Pilot Module).

## Processing Layer

The Processing Layer will be dependent on radio wave frequencies for communicating with the Hardware Input Layer and internet communication with the Application Input Layer. The communication between the Processing Layer and the Application Layer will depend upon the libraries such as internet connection libraries, Android™ Application libraries, the Web Application libraries and the operating system being used by the base station computer. The Application Layer will also be the main base for our image processing operation whereas the pilot module’s operating system will handle the most of the processing of hardware inputs. This layer will contain the main programming portion of our products functions.

## Database Layer

The Database Layer will use the same operating system as the web application layer which will allow HTML, JSP, the JSON or XML data to be processed/ stored correctly. This layer will also be dependent on SQL server and a database driver like JDBC/OLEDB, and Java/.Net libraries.

## Presentation Layer

The Presentation Layer will need NPD operating system’s input/output libraries so that it can prepare output accordingly. This layer will need the web application and Android™ Application libraries to communicate with the system libraries using which the output can be presented to the user in a GUI form. The layer also depends upon the libraries of the pilot module as the operating system used on the pilot module will determine the data being passed to the drone during its operation.

# Testing Considerations

This section will describe how each layers of the system will be validated. The each layer has specified functions and properties which will be performed without compromising the stand-alone principle of those layers and their independency will not be broken. As a development team we will be comparing each layer to the guiding principles used to create the layers to ensure they are consistent and represent the principles stated in the document above.

## Application Input Layer

### Modularity:

The Application Input Layer must be an independent layer which doesn’t depend upon the internal mechanisms of other layers. The layer must be a standalone layer of the system.

### Internal-System Interaction:

The subsystems of this layer will only interact and communicate with the Processing Layer of the architecture.

### External-System Interaction:

The subsystems of this layer will be directly interactive with the user and administrator. The administrator must be able to set user settings, add new local users, and set other local settings of the system.

## Hardware Input Layer

### Modularity:

The Hardware Input Layer must be an independent layer which does not depend upon the internal mechanisms of other layers. The layer must be a standalone layer of the system.

### Internal-System Interaction:

The subsystems of this layer will only interact and communicate with the Processing Layer of the architecture.

### External-System Interaction:

The subsystems of this layer will be gathering data from the system components like drone battery, controller, video camera, pilot module. This layer will gather data from the system components like battery power of the drone, information about controller, video, and other data from the pilot module.

## Processing Layer

### Modularity:

The Processing Layer must be an independent layer which does not depend upon the internal mechanisms of other layers. The layer must be a standalone layer of the system.

### Internal-System Interaction:

The subsystems of this layer will receive data from the Application Input Layer, the Database Layer, and the Hardware Input Layer. Similarly, the subsystems of this layer will only send data to the Presentation Layer and Database Layer of the system.

### External-System Interaction:

The subsystems of this layer will directly interact with the internet to send API requests and receive the responses. This interaction must be validated as the request and response should be authenticated before parsing.

## Database Layer

### Modularity:

The Database Layer must be an independent layer which does not depend upon the internal mechanisms of other layers. The layer must be a standalone layer of the system.

### Internal-System Interaction:

The subsystems of this layer will only send and receive data from the Processing Layer.

### External-System Interaction:

There will be a Hard Drive of the host computer which will act as the data storage of the system. It will be the only external interaction this layer will have.

## Presentation Layer

### Modularity:

The Presentation Layer must be an independent layer which does not depend upon the internal mechanisms of other layers. The layer must be a standalone layer of the system.

### Internal-System Interaction:

The subsystems of this layer will interact with the Processing Layer from which it will receive data. This layer will not receive and send data to any other layer of the system.

### External-System Interaction:

The subsystem of this layer will send the processed data as well as packaged data to the corresponding output devices such as computer screen, Android™ screen, Drone, or LCD screen of the system.