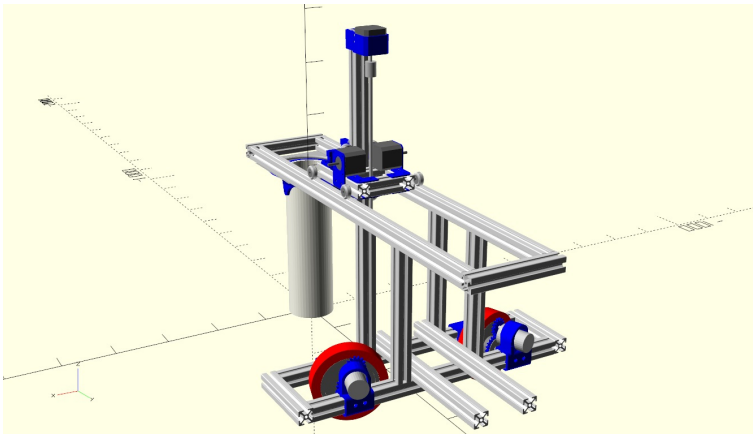
Department of Computer Science & Engineering
The University of Texas at Arlington

Architectural Design Specification
CSE 4316: Senior Design I
Fall 2016



Team Demeter
Polar FarmBot

Arun Kalahasti
Bipin Ghimire
Saman Shrestha
Santosh Pradhan
Travis Matthews

Revision History

Revision	Date	Author(s)	Description
1.0	11.14.2016	AK	official release

CONTENTS

List of Figures

List of Tables

1 INTRODUCTION

Polar FarmBot is a CNC robot that will automate the agricultural growing process. It will be able to plant seeds, water plants, measure soil conditions, and remove weeds. The system will not be able to harvest plants nor remove pests from the plants. It will be stationed in a backyard and will be used to grow various fruits and vegetables such as carrots, lettuce, watermelon, onions, and much more along with all kinds of spices and herbs.

Polar FarmBot will have a central pole in which an arm pivots around. The arm will extend outward from the central pole at variable lengths depending on the size specified by the consumer. At the end of the arm there will be a support that extends downward to a set of wheels that allow the arm to move in a circular direction around the central pole. Along the arm there will be a gantry system that will be powered by a motor to move horizontally back and forward on the arm. The gantry will also have a separate motor that will power a separate arm that moves up and down in the vertical direction. Attached to the rod will be the seeder, water pump, and soil monitor tool assembly. Cables and connectors will be routed through the arm, down the central pole, and back out to a control box located just outside the gardening plot. Within the control box, there will be a motor driver and raspberry pi. The motor driver will power all the motors and sensors on the gantry. The raspberry pi will connect to the internet that will allow for a web based user interface to interact with the system. The control box will also house a power supply that will power all the motors, sensors, motor driver, and raspberry pi. There will also be connectors for the vacuum for picking up seeds and also a connector for water that will allow the robot the ability to water plants.

The robot will also connect back to a computer that will have a web interface for users to interact with the robot. From this interface a user will be able to layout a plot, monitor overall plant growth and soil conditions, and see whether or not the plants are ready for harvesting. To go along with the web interface, there will be a companion application developed for android. This app will have all the same functionality as the web app just on a mobile platform. Polar FarmBot will have multiple cameras attached to it to allow monitoring progress remotely.

2 System Overview

The Polar FarmBot will consist of set of software systems which will function together to plan and execute the process required to plant and grow crops. The primary system layer is the Core Logic Layer. It is responsible for assessing the machine and crops, planning for future actions, and orchestrating the actions needing to be executed immediately. The web client layer is the system a user runs on a browser and allows the user to interact with the system. The web API layer serves data to the web client layer and allows an interface to the FarmBot from other devices. The database layer stores configuration data and historical data for recall and processing. Finally, the motor control layer executes control commands and provides sensor data to the system.

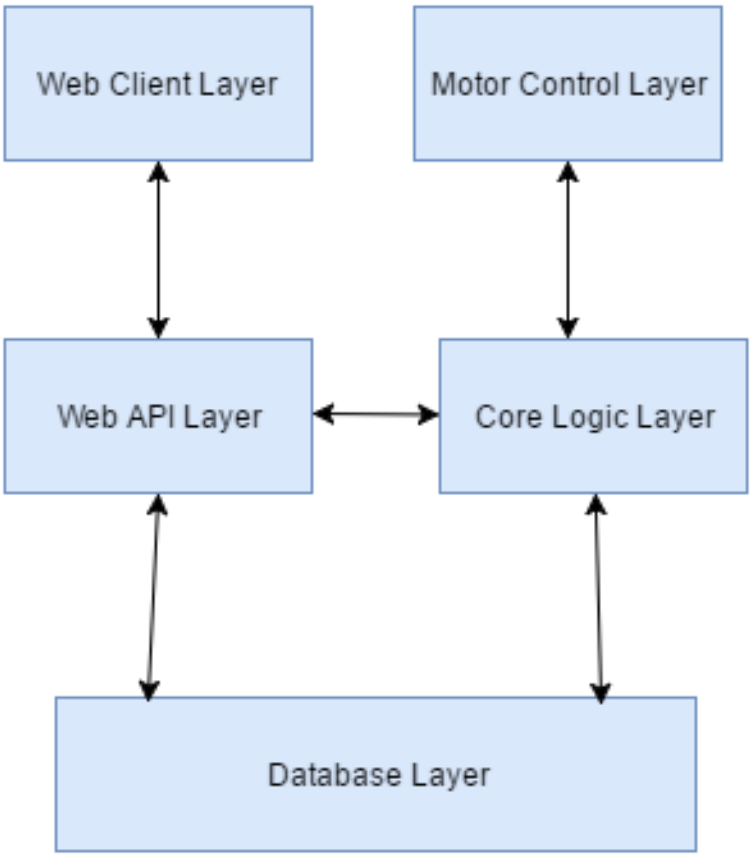


Figure 1: A simple architectural layer diagram

2.1 Web API Description

The Web API Layer serves data to the web client and any api callers through a http gateway. Access is controlled through an authentication engine which will only allow API calls from a registered client. Data is fetched by a data interface and transformed as required by the core program to be served to any requesting client.

2.2 Database Description

The Database Layer is the primary store of all data in the application. It will keep a persistent store of configuration for the application so that the system can be restarted from a blank state without losing critical information such as the plants in the garden, the growth state of those plants, and actions made to those plants. The database layer will also compress data to maximize space efficiency.

2.3 Core Logic Description

The Core Logic Layer acts as the central controller for the total system. Its most important job is to orchestrate a command pipeline to execute tasks on itself and the rest of the system. Secondarily, it also is responsible for generating commands for the motor control interface. Finally, it drives communication to the motor control unit based on instructions in the command pipeline. This communication is pushed through a serial interface via a G-Code executor routine. Most communication to other layers is accomplished through a restful html API.

2.4 Motor Control Description

Motor Control Layer consists of Serial Command Interface, G-Code processor, I/O Handler, Stepper Controller, Sensor Controller. This layer is responsible to execute the processes in the command pipeline to control the motors.

2.5 Web Client Description

The Web Client Layer displays system information to the user. It will consist of a JavaScript framework, static content, and a data service.

3 Subsystem Definitions & Data Flow

The data flow in the Farm Bot architectural design starts from the I/O sub layer in the Web Client layer. The user gives the input through the web interface and it reaches to the Web API Layer. In Web API layer, the data flows to the core logic layer through web gateway. The data will also flow between the core and the User/Access Authentication layers to validate the user access. The core will be able to access database as per the request from user through database interface. In the Core Logic Layer, G-Code Generator generates the G-Code in the form of processes which are lined up by the Scheduler based on their priority in the command pipeline. Then, those processes are transmitted to the Motor Control Layer through the Serial Command Interface and are executed in the G-code Processor. These processed signals help motor to move to the precise coordinates.

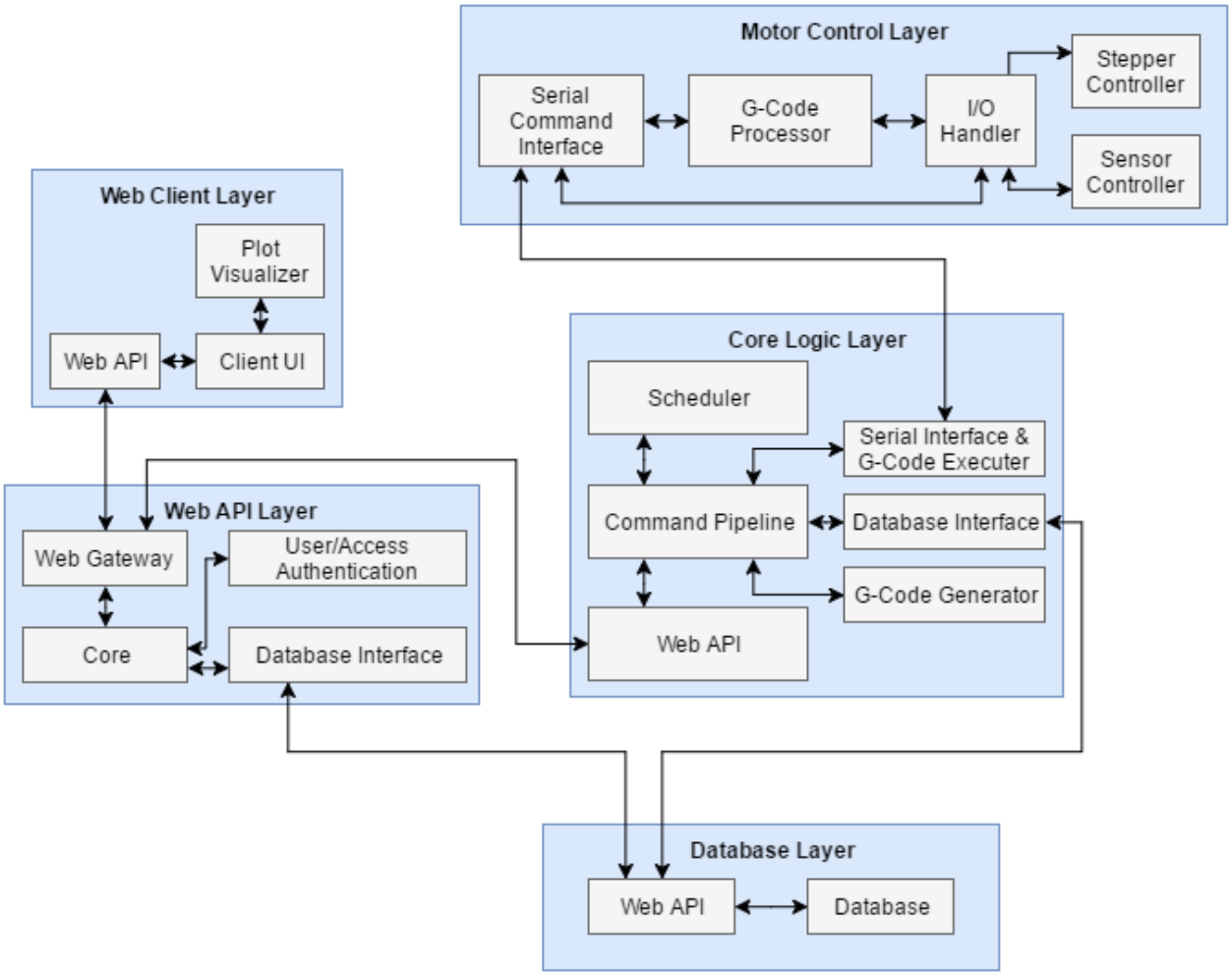


Figure 2: A simple data flow diagram

4 Core Logic Layer Subsystems

The Core Logic Layer subsystems are centered around planning and executing the main actions needed by the machine.

4.1 Command Pipeline

The Command Pipeline is a ordered que of actions which are known to need execution. Each action is read off the top of the list and executed at an indicated time. The command pipeline must also assign an action to the appropriate system for execution.

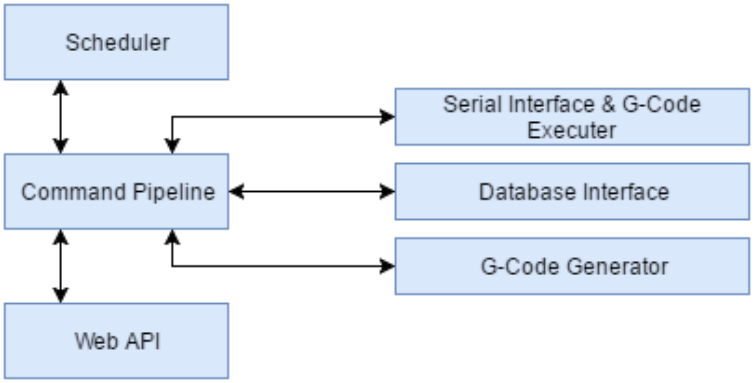


Figure 3: Example subsystem description diagram

4.1.1 Assumptions

The command pipeline assumes that the other system components are always available to execute commands.

4.1.2 Responsibilities

The command pipeline must accurately maintain the order of execution for its tasks.

4.1.3 Subsystem Interfaces

The command pipeline must be able to interface with all the other systems it executes commands for. The primary interface for these transactions is through a html web API.

Table 2: Subsystem interfaces

ID	Description	Inputs	Outputs
#01	Scheduler bus	Upcomming task events	Future Actions needed
#02	Serial Pipeline	Serial Data	Serial Commands
#03	G-Code interface	Serial Data	Serial Commands

4.2 Scheduler

The Scheduler subsystem is built to store requested command events as well as request their timely execution by the command pipeline

4.2.1 Assumptions

The scheduler is assumed to be constantly running.

4.2.2 Responsibilities

The primary purpose of the scheduler was to ensure commands have been executed in a timely manner. This necessitates an internal time tracking system to ensure commands are being executed when they are scheduled to.

4.2.3 Subsystem Interfaces

The scheduler subsystem only needs to interface directly with the command pipeline. It must be able to send and recieve full command instructions as used by the command pipeline.

Table 3: Subsystem interfaces

ID	Description	Inputs	Outputs
#01	Scheduler bus	Upcoming task events	Future Actions needed

4.3 Web API

The Web API subsystem is to allow interfacing to the rest of the FarmBot systems.

4.3.1 Assumptions

The Web API subsystem is assumed to not need user authentication for security.

4.3.2 Responsibilities

The Web API subsystem must listen for input from other layers within the FarmBot architecture. If the input is not provided in a sensible manner, it must be transformed in the subsystem.

4.3.3 Subsystem Interfaces

The Web API subsystem provides an unsecured html interface.

Table 4: Subsystem interfaces

ID	Description	Inputs	Outputs
#01	HTML	html requests	html responses

4.4 Database Interface

The Database Interface subsystem allows data transactions to the central database.

4.4.1 Assumptions

The data passing though the database interface is assumed to be in the final form required before storing. The Database Layer may not always be available.

4.4.2 Responsibilities

The database interface must attempt to send data to the Database Layer of the FarmBot. Data which cannot immediately be pushed to the data layer will be buffered within the subsystem until a timeout or the Database Layer accepts the data.

4.4.3 Subsystem Interfaces

The Database interface only communicates to the command pipeline subsystem and the database layer API.

Table 5: Subsystem interfaces

ID	Description	Inputs	Outputs
#01	Command Pipeline	Data store requests	responses

4.5 G-Code Generator

The G-Code generator subsystem is the software element which converts command actions in the pipeline into executable machine movement code.

4.5.1 Assumptions

The development of this subsystem assumes that complex actions can be consistently digested into a series of motor control actions.

4.5.2 Responsibilities

The G-Code generator will be responsible for creating all motor control actions.

4.5.3 Subsystem Interfaces

The G-Code Generator only communicates to the command pipeline subsystem.

Table 6: Subsystem interfaces

ID	Description	Inputs	Outputs
#01	Command Pipeline	Actions	G-Code Commands

4.6 Serial Interface

The serial interface connects the core logic layer to the motor control layer.

4.6.1 Assumptions

We are assuming the motor control layer requires no input from the rest of the systems beyond coordinate actions. We are also assuming the motor control layer acknowledges receiving all instructions and if no response is received by the motor control layer the instruction was unable to be executed.

4.6.2 Responsibilities

The Serial Interface is responsible for parsing action in the command pipeline sending them to the motor control layer.

4.6.3 SUBSYSTEM INTERFACES

The Serial Interface only communicates to the command pipeline subsystem and the motor control layer.

Table 7: Subsystem interfaces

ID	Description	Inputs	Outputs
#01	Command Pipeline	G-Code Commands	Serial Strings

5 Motor Control Layer Subsystems

The Motor Control Layer controls the machine frame and executes actions requested by the core logic layer. The commands will be sent through a serial interface in the G-Code syntax. The Motor Control Layer will parse these commands, translate them to the machine required machine configuration, and manipulate the motors on the frame to match. If the commands are sensor data requests the I/O handler will translate the raw sensor data into a string format and transmit it back across the serial interface.

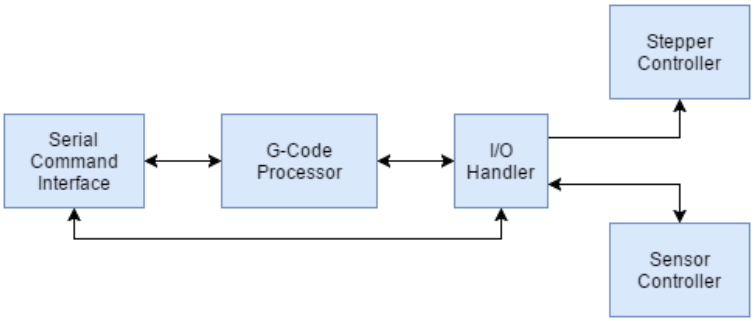


Figure 4: Example subsystem description diagram

5.1 Serial Command Interface

Ther Serial Command Interface is the only method of communication available to the entire Motor Control Layer. This subsystem receives character input through a serial bus and must parse them into individual instructions. These instructions are then routed to the appropriate subsystem for execution. The interface will also build return statements and transmit them as requested by the other Motor Control Layer subsystems.

5.1.1 Assumptions

The Serial Command Interface should not receive commands longer than the limiting buffer. Any commands which are longer than that will not be executed, returning an error through the Serial Command Interface to the control system. The control system is assumed to understand that a command was unable to be parsed and attempt to recover without further action from the Serial Command Interface.

5.1.2 Responsibilities

The Serial Command Interface is responsible for parsing all character input into parametrized commands for consumption by the appropriate subsystem. The interface must request a retransmit if there is an error in parsing the input or return a command acknowledged message if parsing was successful. When the Serial Control Interface sends a message to the Core Logic Layer it does not expect a result.

5.1.3 Subsystem Interfaces

Table 8: Subsystem interfaces

ID	Description	Inputs	Outputs
#01	Serial Bus Interface	characters over bus	characters over bus

5.2 G-Code Processor

The G-Code Processor processes Cartesian coordinate commands into polar configurations which can be used to control the machine.

5.2.1 Responsibilities

This subsystem must intercept control commands which were sent using Cartesian coordinates and convert them to polar coordinates before passing them to the I/O Handler subsystem.

5.2.2 Subsystem Interfaces

Table 9: Subsystem interfaces

ID	Description	Inputs	Outputs
#01	Serial Command Interface	String array	N/A
#02	I/O Handler	N/A	String array

5.3 I/O Handler

The I/O Handler executes motor control as requested by the Serial Command Interface and the G-Code Processor. These commands are received as an array of strings. The I/O Handler calculates the required motor actions to bring the machine into the desired state and passes the actions to the Motor Control Layer. If the Serial Command Interface requests sensor data, the I/O Handler will parse the raw data into an established format to send back through the Serial Command Interface.

5.3.1 Assumptions

The I/O Handler should never have to parse unknown sensor data.

5.3.2 Responsibilities

The I/O Handler must maintain an internal store of machine state and be able to execute some default motor action to detect and establish machine state.

5.3.3 Subsystem Interfaces

Table 10: Subsystem interfaces

ID	Description	Inputs	Outputs
#01	Stepper Controller	Number of steps required	N/A
#02	Sensor Controller	N/A	Raw Sensor Data

5.4 Motor Control

The Motor Control subsystem is the primary driver for the motors on the FarmBot. It will execute motion commands as requested by the I/O Handler subsystem.

5.4.1 Assumptions

The I/O Handler may request actions from the Motor Control system which would be harmful to the machine. The Motor Control system should prioritize safe operation over faithful command execution.

5.4.2 Responsibilities

The Motor Control subsystem should execute commands given to it in the expected format.

5.4.3 Subsystem Interfaces

Table 11: Subsystem interfaces

ID	Description	Inputs	Outputs
#01	I/O Handler interface	Commands	N/A
#02	Stepper Motor Driver	N/A	step direction

5.5 Sensor Control

The Sensor Control subsystem allows easier interfacing with sensor systems.

5.5.1 Assumptions

All sensors will be configured for input parsing in the Sensor Control subsystem.

5.5.2 Responsibilities

The Sensor Control system will only be responsible for unifying and normalizing sensor input data.

5.5.3 Subsystem Interfaces

Table 12: Subsystem interfaces

ID	Description	Inputs	Outputs
#01	I/O Handler Interface Sensor Requests Sensor Configuration	Sensor Data	
#02	Sensors	Sensor Configuration	Sensor Data

6 Database Layer Subsystems

The Database layer provides an interface to store and retrieve data for the other layers. It helps maintaining the persistency and reliability of the data.

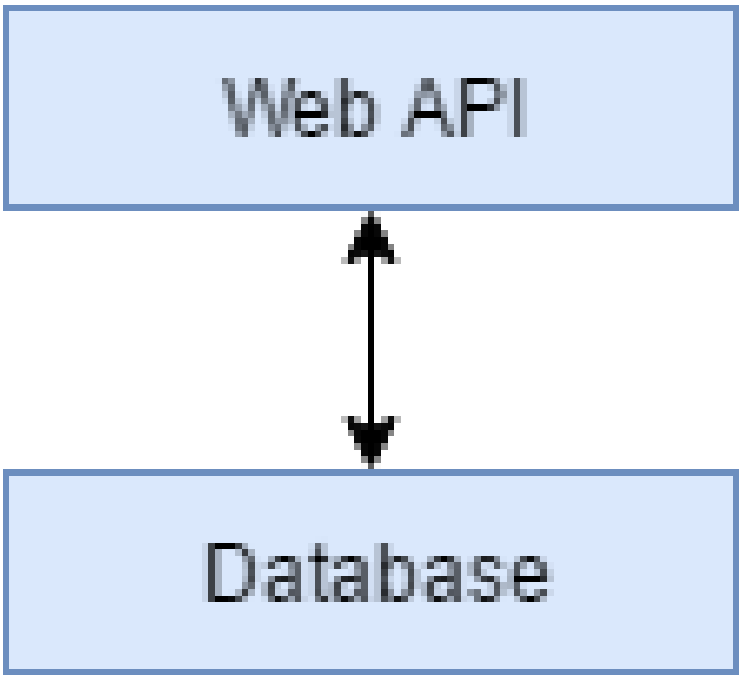


Figure 5: Example subsystem description diagram

6.1 Web API

The Web API allows access of data for clients

6.1.1 Assumptions

The Web API makes an assumption that bad calls will be ignored and security will not be needed.

6.1.2 Responsibilities

Web API should parse the html request.

6.1.3 Subsystem Interfaces

Table 13: Subsystem interfaces

ID	Description	Inputs	Outputs
#01	HTML Interface	HTML Requests	JSON Objects
#02	Database Interface	JSON Objects	JSON Objects

6.2 Database

The Database provides routes to all data and stores the final data that is ready to be fetched by other layers.

6.2.1 Assumptions

The Database should only store valid data and ignore invalid data.

6.2.2 Responsibilities

The Database should store data and keep it ready to deliver to requesters.

6.2.3 Subsystem Interfaces

The database itself does not have any system interfaces.

7 Web Service Layer Subsystems

This layer allows access for data web based clients. Clients connect to the gateway to request resources. If the request passes the authentication, it will be sent to the core.

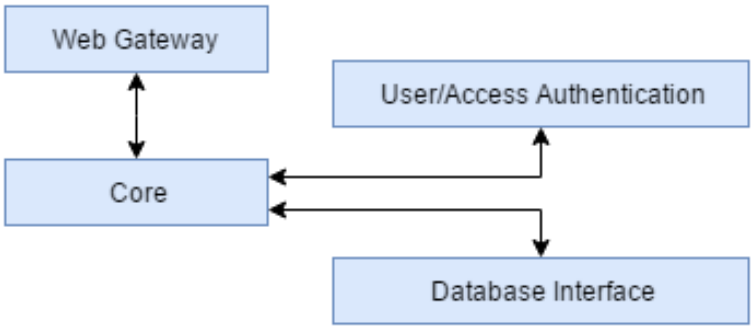


Figure 6: Example subsystem description diagram

7.1 Web Gateway

Web Gateway is a html controller that parses the html request.

7.1.1 Assumptions

The Web Gateway subsystem makes the assumption that any improperly formatted html request will be ignored.

7.1.2 Responsibilities

Web Gateway should parse the html request.

7.1.3 Subsystem Interfaces

Table 14: Subsystem interfaces

ID	Description	Inputs	Outputs
#01	HTML Interface	HTML Requests	JSON Objects
#02	Database Interface	JSON Objects	JSON Objects

7.2 User/Access Authentication

User/Access Authentication validates the html request based on the tokens.

7.2.1 Assumptions

The User/Access Authentication subsystem makes the assumption that any unauthorized html tokens will be ignored.

7.2.2 Responsibilities

User/Access Authentication should check the html request for its validity.

7.2.3 Subsystem Interfaces

Table 15: Subsystem interfaces

ID	Description	Inputs	Outputs
#01	HTML Interface	HTML Requests	JSON Objects
#02	Database Interface	JSON Objects	JSON Objects

7.3 Database Interface

Database Interface helps connect the core to the actual database to fetch and store the data in the actual database.

7.3.1 Assumptions

The database Interface subsystem makes the assumption that the data passing through it is in the final form so that it can be stored in the actual database.

7.3.2 Responsibilities

The database Interface should allow the data to pass to the Database Layer. Invalid data should be buffered within the subsystem before passing it to the Database Layer.

7.3.3 Subsystem Interfaces

Table 16: Subsystem interfaces

ID	Description	Inputs	Outputs
#01	Database Interface	JSON Objects	JSON Objects

7.4 Core

The Web Service Layer Core subsystem will distribute all the parsed html requests to the appropriate layer in the total system.

7.4.1 Assumptions

Core subsystem makes the assumption that there would be some subsystem that would take care of the distributed parsed requests.

7.4.2 Responsibilities

The core subsystem should make sure to distribute the parsed html requests to the appropriate subsystem.

7.4.3 SUBSYSTEM INTERFACES

Table 17: Subsystem interfaces

ID	Description	Inputs	Outputs
#01	HTML Interface	HTML Requests	JSON Objects
#02	Database Interface	JSON Objects	JSON Objects

8 Web Client Layer Subsystems

A browser interface that allows the user to interact with the entire system.

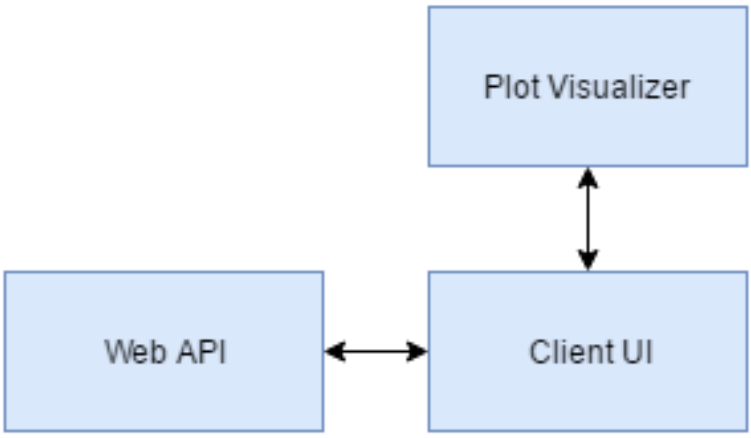


Figure 7: Example subsystem description diagram

8.1 Web API

The Web API makes calls to the web service layer. It also stores and provides user authentication credentials.

8.1.1 Assumptions

The Web API cannot assume the web service layer is always operating and will respond.

8.1.2 Responsibilities

It should maintain connection to web service layer. It should pass all requests from other subsystems. It should keep user authentication credentials so that API calls can be validated.

8.1.3 Subsystem Interfaces

Table 18: Subsystem interfaces

ID	Description	Inputs	Outputs
#01	HTML Interface	HTML Requests	JSON Objects

8.2 Plot Visualizer

Take data and convert it into image. It should take the plot data and graph out the elements and generate a visual example.

8.2.1 Assumptions

The input data is valid.

8.2.2 Responsibilities

It should convert data to image and be able to generate visual example.

8.2.3 Subsystem Interfaces

Table 19: Subsystem interfaces

ID	Description	Inputs	Outputs
#01	Plot data	JSON Object	Image or JS Canvas

8.3 Client UI

A JavaScript based UI to allow the user to see the status of the farmbot, send commands, and schedule various activities.

8.3.1 Assumptions

The client browser will run the needed JavaScript. Unsupported browsers will be notified and rejected.

8.3.2 Responsibilities

The Client UI should maintain minimal resource usage to maintain the expected functionality.

8.3.3 Subsystem Interfaces

Table 20: Subsystem interfaces

ID	Description	Inputs	Outputs
#01	Javascript	User Interaction	Web Page