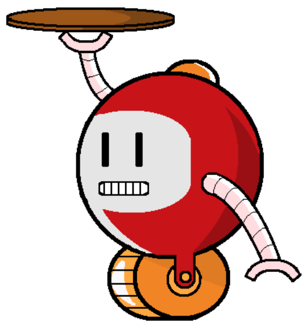


LazyBots

McMaster University

Draft Component Design

SE 4GA6 & TRON 4TB6

Group 9

Karim Guirguis	001307668
David Hemms	001309228
Marko Laban	001300989
Curtis Milo	001305877
Keyur Patel	001311559
Alexandra Rahman	001305735

Table of Contents

List of Tables

List of Figures

1 Revisions

Table 1: VIC Table of Revisions

Date	Revision Number	Authors	Comments
November 24th, 2017	Revision 0	Karim Guirguis David Hemms Marko Laban Curtis Milo Keyur Patel Alexandra Rahman	-

2 Manager System

2.1 Purpose

The following will describe the component software design associated with Manager System. This will be carried out within web based tools to allow managment to access there information Anywhere

2.2 Scope

The scope of this section is associated with any front end user interfaces that the managment staff will use. This includes the MIS/MID and uses relation in regards to the Map Making Page, the Error Viewing page and the Login System.

2.3 Module Decomposition

Manager Login Page: Given the login credentials, will authenticate administrator of the system with the server. Secrets include how it goes about verifying with the server if the credentials are valid. **Manager Station Map Software Page**: Will allow administrator to create or modify the map of the area where Alfred will deliver drinks. This map will then be sent and stored on the server. The secrets of this module includes how the mapping system will translate user input into the map file **Manager Station Request Software Page**: Will allow administrator to execute commands for Alfred, as well as view incoming error codes from Alfred. Secrets include how the errors are decoded from the server.

2.4 Uses Relation

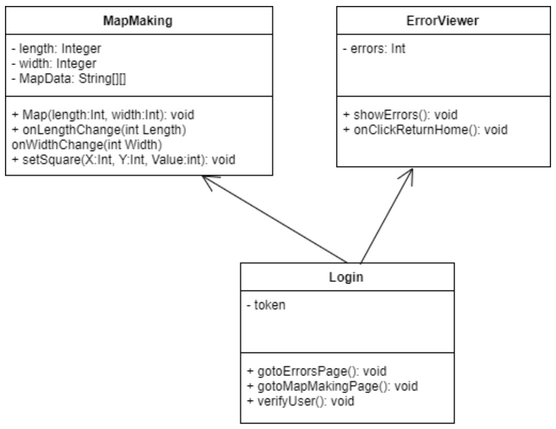


Figure 1: Alfred Uses Relation Diagram

2.5 MIS

2.5.1 Login Page

Uses

- MapMaking (WebPage)
- ErrorViewer (WebPage)

Public Functions

gotoErrorsPage(): void Navigates to the page associated with showing the Managers page for Errors with Alfred. **gotoMapMakingPage(): void** Navigates to the page associated with showing the Managers page for creating a resturant Map. **verifyUser(): void** Determines if the information that the user put into the form on the webpage is correct.

2.5.2 MapMaking Page

Uses None
Public Functions
Map(length:Int, width:Int): void Constructor to object for page load **onLengthChange(int Length)** Sets the length of the map when the user changes it in the form. **onWidthChange(int Width)** Sets the width of the map when the user changes it in the form. **setSquare(X:Int, Y:Int, value:Int): void** Sets the value of the square based on its X,Y position when there is an on click event

2.5.3 ErrorViewer Page

Uses None
Public Functions
showErrors(): void Shows the Errors Associated to Alfred. **onClickReturnHome()** Signals the Robot to return home.

2.6 MID

2.6.1 Login Page

Uses

- MapMaking (WebPage)
- ErrorViewer (WebPage)

Internal Variables
token: String âĂŞ Token for a session with the server.
Functions
public gotoErrorsPage(): void Navigates to the page associated with showing the Managers page for Errors with Alfred. **public gotoMapMakingPage(): void** Navigates to the page associated with showing the Managers page for creating a resturant Map. **public verifyUser(): void** Determines if the information that the user put into the form on the webpage is correct.

2.6.2 MapMaking Page

Uses None
Internal Variables length: Integer - Length of the Map **width: Integer** - Width of the Map **MapData: String[][]**- Storage of the map values
Public Functions
public Map(length:Int, width:Int): void Constructor to object for page load **public onLengthChange(int Length)** Sets the length of the map when the user changes it in the form. **public onWidthChange(int Width)** Sets the width of the map when the user changes it in the form. **setSquare(X:Int, Y:Int, value:Int): void** Sets the value of the square based on its X,Y position when there is an on click event. The Values corresponds to:

- 0: Free to move
- 1: Path is blocked
- 2: Table
- 3: Base

2.6.3 ErrorViewer Page

Uses None
Public Functions
showErrors(): void Shows the Errors Associated to Alfred. where the Errors are in the following format

- LowLiquid: 0x00000001
- LeakingTank: 0x00000010
- LowBattery: 0x00000100
- NoMovement: 0x00001000

onClickReturnHome() Signals the Robot to return home.

3 Alfred System

3.1 Purpose

The following will describe the component software, mechanical and electrical design associated with Alfred's Manager System, Alfred's Drivetrain and Alfred's Image Processing system. These three systems will be ran on the Raspberry Pi.

3.2 Scope

The scope of this section is associated with Alfred's Manager System, Alfred's Drivetrain and Alfred's Image Processing system. The software documentation will provide the MIS and MID, uses relations to describe how the system will be designed to preform its function of being able to drive to a specific location. The mechanical and electrical design will focus on the aspects to provide motion and navigation.

3.3 Module Decomposition

Alfred Manager Module: Endpoint for communication with Alfred. Will manage communication with server, as well as send any errors that Alfred is experiencing. Secrets include Parsing of messages from the server and the pumping system. **Alfred Drive Train Module**: Responsible for driving and managing the motors based on desired route. Will also be sending errors preventing movement to Alfred Manager Module. Secrets include how the robot will preform navigation based on the map, how the robot will control and drive the robot and the inputs from the image processing module. **Image Processing Module**: Will detect any obstacles in the way as well as locate incoming nodes. Will communicate with Alfred Drive Train Module, to determine whether any required action based on results. Secretes include how the image processing will be carried out

3.4 Uses Relation

3.5 MIS

3.5.1 Alfred Manager Class

Uses

- Communication (Static Class)
- State (Enum)
- Drivetrain (Class)
- Order (Class)

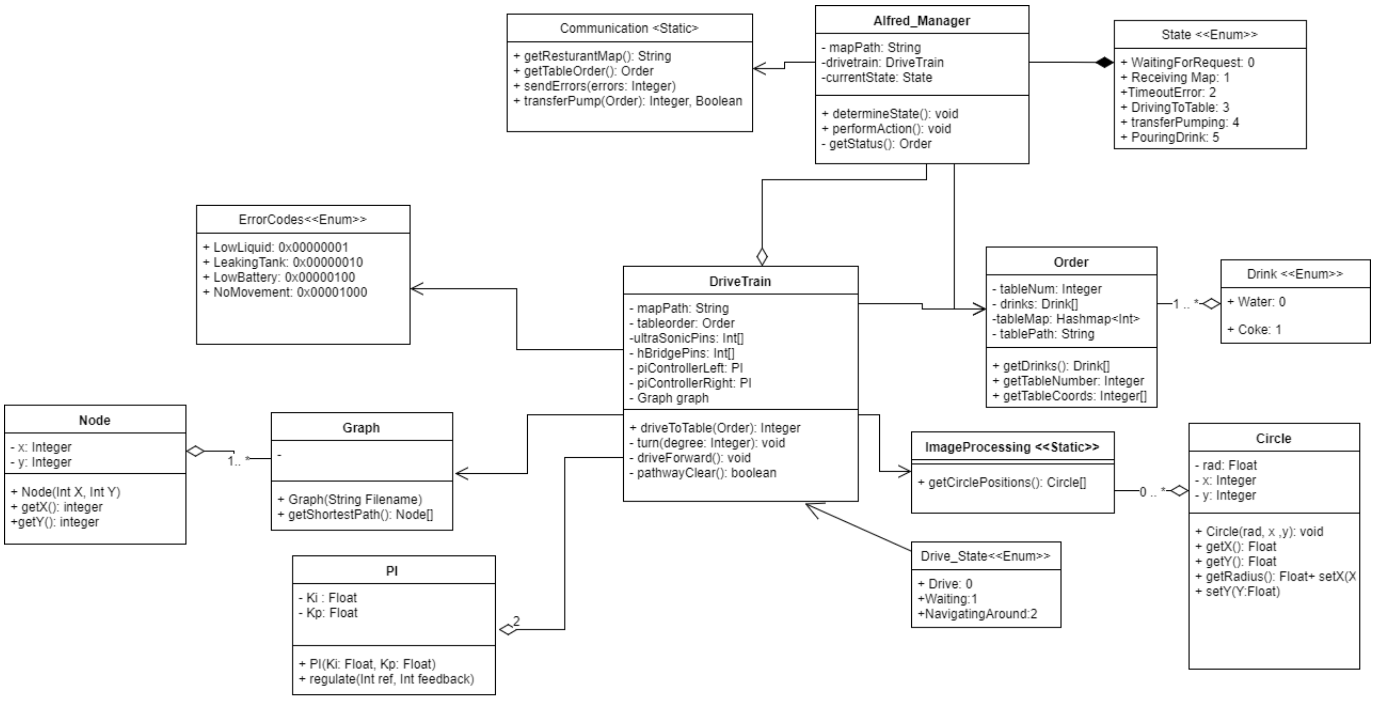


Figure 2: Alfred Uses Relation Diagram

Public Functions
determineState(): void Determines the overall state of Alfred based on the inputs to Alfred. **performAction(): void** Will preform the desired state based on the state of Alfred.

3.5.2 Order Class

Uses

- Drink (Enum)

Public Functions
getDrinks(): Drink[] âĂŞ Returns the list of drinks **getTableNumber**: Integer âĂŞ Returns the table number reference **getTableCoords**: Integer[] âĂŞ Returns the coordinates to the table

3.5.3 Drivetrain Class

Uses

- Order (Class)
- ImageProcessing (Class)
- Drive_State (Enum)
- ErrorCodes (Enum)
- Circle (Class)
- Graph (Class)
- Node (Class)
- PI (Class)

Public Functions driveToTable(Order): Integer This function will preform the driving operation in order to navigate towards the specific table.

3.5.4 Communication Static Class

Uses None **Public Functions getResturantMap(): String** Retrieves and stores the map to be used for navigation. Returns the path of the map **getTableOrder(): Order** Retrieves and returns the TableâĂŹs Order. **sendErrors(errors: Integer)** Sends an integer with the described set of integers. **transfer-Pump(Order): Integer** Preforms communication with the pumping system. Sends the Order data and receives the errors from the pumping system and if it complete.

3.5.5 Graph Class

Uses

- Node (Class)

Public Functions Graph(String Filename) Constructor to create a graph object. Builds the graph based on the path to the map **getShortestPath(): Node[]** Returns a list of nodes that describes the shortest way to get to the destination.

3.5.6 Node Class

Uses None
Public Functions Node(Int X, Int Y) Constructor to the node class, takes in the position of the point along the X Y plane. **getX(): integer** Returns the x coordinate of the node **getY(): integer** Returns the y coordinate of the node

3.5.7 PI Class

Uses None
Public Functions
PI(Ki: Float, Kp: Float) Constructor for the PI controller object taking in the Ki and Kp terms **regulate(Int ref, Int feedback)** Based on the reference to control to and the feedback, will determine the value to set the outputs too.

3.5.8 Imaging Processing Static Class

Uses

- Circle (Class)

Public Functions
getCirclePositions(): Circle[] Gives a list of circles that were within the view of the camera

3.5.9 Circle Class

Uses None
Public Functions
Circle(radius, x, y) Constructor of the circle class, takes in initial radius, x and y values **getX(): Float** Gives the X position of the circle relative to the image **getY(): Float** Gives the Y position of the circle relative to the image **getRadius(): Float** Gives the radius of the circle **setX(X:Float)** Sets the X position of the circle. **setY(Y:Float)** Sets the Y position of the circle. **setRadius(rad : Float)** Sets the radius of the circle.

3.6 MID

3.6.1 Alfred Manager Class

Uses

- Communication (Static Class)
- State (Enum)
- Drivetrain (Class)
- Order (Class)

Internal Variables
mapPath: String âĂŞ The absolute path to the map directory **drivetrain: DriveTrain** âĂŞ An object encapsulates information in regards to the drivetrain **currentState: State** âĂŞ Holds the information in regards to which action will be preformed by Alfred
Functions
public determineState(): void Determines the overall state of Alfred based on the inputs to Alfred based on the following tabular expression:

Previous State	**Conditions**	**Next State**
WaitingForRequest	Order == Null && time < timeout	WaitingForRequest
	Order == Null && timeout <= time	TimeoutError
	Order! = Null	ReceivingMap
ReceivingMap	Order == Null && time < timeout	WaitingForRequest
	Order == Null && timeout <= time	TimeoutError
	Order! = Null	DrivingToTable
TimeoutError	time< try_again	TimeoutError
	try_again <= time	WaitingForRequest
DrivingToTable	Drive_errors == 0	transferPumping
	Drive_errors != 0	WaitingForRequest
transferPumping	Pump_errors ==0 && table_done	WaitingForRequest
	Pump_errors !=0	WaitingForRequest

public performAction(): void Will preform the desired state based on the state of Alfred based on the following table:

State	**Action**

WaitingForRequest	getTableOrder()
	sendErrors(errors)

ReceivingMap	getResturantMap()
TimeoutError	Sleep()
DrivingToTable	driveToTable(Order)
transferPumping	transferPump(Order)

3.6.2 Order Class

Uses

- Drink (Enum)

Internal Variables tableNum: Integer - the reference to the table **drinks: Drink[]** âĂŞ List of drinks for the userâĂŹs table **tableMap: Hashmap<Int>** - Map that takes in a table reference number and returns its X,Y Coord **tablePath: String** âĂŞ gives the path of the table for the hashmap
Functions public getDrinks(): Drink[] Returns the list of drinks **public getTableNumber: Integer** Returns the table number reference **public getTableCoords: Integer[]** Returns the coordinates to the table from the Hashmap of table numbers

3.6.3 Drivetrain Class

Uses

- Order (Class)
- ImageProcessing (Class)
- Drive_State (Enum)
- ErrorCodes (Enum)
- Circle (Class)
- Graph (Class)
- Node (Class)
- PI (Class)

Internal Variables mapPath: String âĂŞ The absolute path to the map directory **tableorder: Order** âĂŞ The order from the next table. **ultraSonicPins: Int[]** âĂŞ The pins dedicated for the ultrasonic sensors **hBridgePins: Int[]** âĂŞ The pins dedicated for the H-bridge **graph: Graph** âĂŞ Object to the graph object to find the shortest path **piControllerLeft: PI** âĂŞ Object used for PI control for the left side of the drivetrain **piControllerRight: PI** âĂŞ Object used for PI control for the left side of the drivetrain
Functions
public driveToTable(Order): Integer This function will preform the driving operation in order to navigate towards the specific table. **private turn(degree: Integer): void** This function will turn relative to its current position the amount desired within the argument. The robot will use the references of the circles to help with alignment by knowing that every node is 90 degrees from one another.
private driveForward(void): void This function will control the robot to move forward provided: $\forall frontultrasonicsensors : d_{ultrasonic} > d_{min}$. This motion will use the PI regulators to provide motion at human speed and will continue until a the next circle is within the middle of the camera.
private pathwayClear (void): boolean This function will determine if robot to move forward provided: $\forall frontultrasonicsensors : d_{ultrasonic} > d_{min}$.

3.6.4 Communication Static Class

Uses None **Internal Variables** None **Functions getResturantMap(): String** Retrieves the map file based off of an FTP protocol and stores the map to be used for navigation. Returns the path of the map **getTableOrder(): Order** Retrieves and returns the TableâĂŹs Order. **sendErrors(errors: Integer)** Sends an integer with the described set of integers. **transferPump(Order): Integer** Performs communication with the pumping system using UART communication. The first 32 bits are the error code of the pumping system and the last bit is the status of the pumping system. Sends the Order data and receives the errors from the pumping system.

3.6.5 Graph Class

Uses

- Node (Class)

Internal Variables None
Functions Graph(String Filename) Constructor to create a graph object. Builds the graph based on the path to the map **getShortestPath(): Node[]** Returns a list of nodes that describes the shortest way to get to the destination. The shortest path will be preformed using the map and DijkstraâĂŹs algorithm.

3.6.6 Node Class

Uses None
Public Functions Node(Int X, Int Y) Constructor to the node class, takes in the position of the point along the X Y plane. **getX(): integer** Returns the x coordinate of the node **getY(): integer** Returns the y coordinate of the node
Internal Variables x: Integer âĂŞ The x coordinate of the node **y: Integer** âĂŞ The y coordinate of the node.
Functions public Node(Int X, Int Y) Constructor to the node class, takes in the position of the point along the X Y plane. **public getX(): integer** Returns the x coordinate of the node **public getY(): integer** Returns the y coordinate of the node

3.6.7 PI Class

Uses None **Internal Variables Ki: Float** âĂŞ Integral temp for the PI controller **Kp: Float** âĂŞ The y coordinate of the node.
Functions public PI(Ki: Float, Kp: Float) Constructor for the PI controller object taking in the Ki and Kp terms **public regulate(Int ref, Int feedback)** Based on the reference to control to and the feedback, will determine the value to set the outputs too. Determines the output based along the following formula: $out = Kp * error + Ki * \int_0^t (error)dt$. Note that the derivative term is not used due to the error associated with derivatives within computing systems.

3.6.8 Imaging Processing Static Class

Uses

- Circle (Class)
- Open CV (External Library)

Internal Variables None
Functions
Public getCirclePositions(): Circle[]
Gives a list of circles that were within the view of the camera. Open CV returns a list of objects which can then be checked to see if they are black circles.

3.6.9 Circle Class

Uses None **Internal Variables**
rad: Float âĂŞ The radius of the circle **x: Integer** âĂŞ The X position of the circle relative to the image
y: Integer - The Y position of the circle relative to the image
Functions
Circle(radius, x, y) Constructor of the circle class, takes in initial radius, x and y values **getX(): Float** Gives the X position of the circle relative to the image **getY(): Float** Gives the Y position of the circle relative to the image **getRadius(): Float** Gives the radius of the circle **setX(X:Float)** Sets the X position of the circle. **setY(Y:Float)** Sets the Y position of the circle. **setRadius(rad : Float)** Sets the radius of the circle.

4 Pumping System

4.1 Purpose

The following will describe the component software, mechanical and electrical design associated with Alfred's Pumping System, Alfred's Drivetrain and Alfred's Image Processing system. This system will be ran on the Arduino Mega.

4.2 Scope

The scope of this section is associated with Alfred's Pumping System. The software documentation will provide the MIS and MID, uses relations to describe how the system will be designed to preform its function of being able to communicate to the raspberry pi and pump drinks. The mechanical and electrical design will focus on the different pumps/sensors that will be associated with the pumping system.

4.3 Module Decomposition

Alfred Pumping Module: Will control pumping system in regards of when to pour, how long and rate of dispensing. Will communicate to the raspberry pi errors pertaining to the pump or container to Alfred Manager Module. secrets include how the system preforms the dispensing of drinks and determination of errors.

4.4 Uses Relation

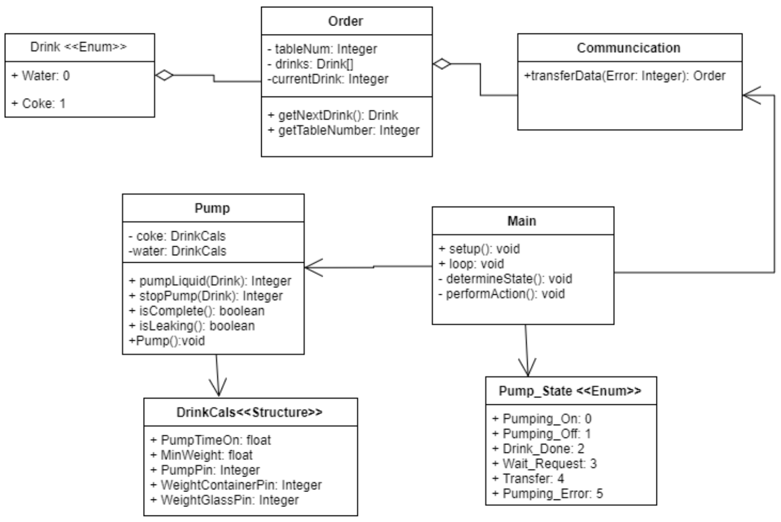


Figure 3: Uses Relation Diagram for the pumping system

4.5 MIS

4.5.1 Main Class

Uses

- Order (Class)
- Drink (Enum)
- Pump (Class)
- Pump_State (Enum)
- Communication (Class)

Internal Variables None
Functions
public setup(): void Setup function for Arduino that initializes the pins that will be used
publicloop(): void Main loop that will preform the main logic of the pumping system

public determineState(): void Based on the previous state of the pumping machine will look at factors such as weight of the liquid, need for communication and errors to determine the next state.
public performAction(): void Based on the state of the device it will:

- Pumping_On: Turn on the voltage for the pump
- Pumping_Off: Turn off the voltage for the pump
- Wait_Request: Will Sleep for a specific amount of time before checking again
- Transfer: Preform transferring via UART
- Pumping_Error: Perform No Action, and ensure all pumping devices are off

4.5.2 Class Pump

Uses

- DrinkCals

public Functions Pump():void Initializes the values of the pumping module **pumpLiquid(Drink): Integer** Turns on the pump for the specific drink type. **stopPump(Drink): Integer** Turns off the pump for the specific drink type. **isComplete(): Boolean** Determines if the drink has been completely filled or not based. **isLeaking(): Boolean** Determines if the containers are losing fluid when there is no pumping. **isSafeTempature()** Determines if the containers are still storing the liquids are at a safe temperature.

4.5.3 Communication Class

Uses

- DrinkCals

Internal Values None
Functions
transferData(Integer): Order
Communication performed used where Orders are received and errors are transferred with the Manager system.

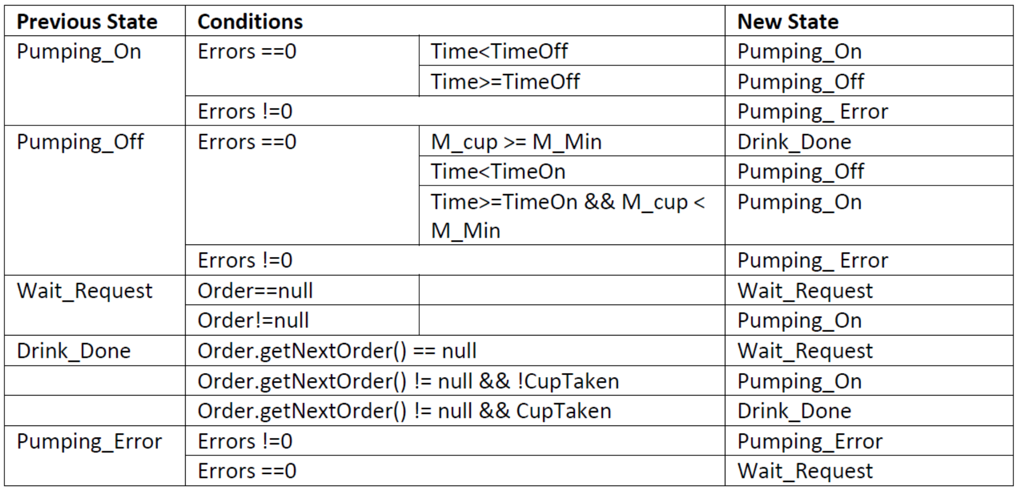
4.6 MID

4.6.1 Main Class

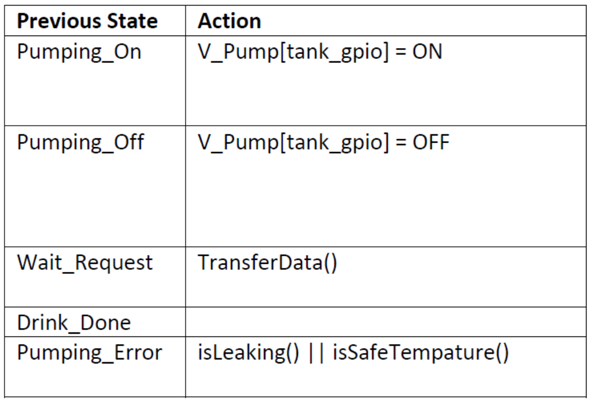
Uses

- Order (Class)
- Drink (Enum)
- Pump (Class)
- Pump_State (Enum)
- Communication (Class)

Internal Variables None
Functions
public setup(): void Setup function for Arduino that initializes the pins that will be used
public loop(): void Main loop that will preform the main logic of the pumping system
public determineState(): void Based on the previous state of the pumping machine will look at factors such as weight of the liquid, need for communication and errors to determine the next state. Which state is summarized in the following table:
public performAction(): void Based on the state of the the pumping system, will preform the following actions:



Previous State	Conditions		New State
Pumping_On	Errors ==0	Time<TimeOff	Pumping_On
		Time>=TimeOff	Pumping_Off
	Errors !=0		Pumping_ Error
Pumping_Off	Errors ==0	M_cup >= M_Min	Drink_Done
		Time<TimeOn	Pumping_Off
		Time>=TimeOn && M_cup < M_Min	Pumping_On
	Errors !=0		Pumping_ Error
Wait_Request	Order==null		Wait_Request
	Order!=null		Pumping_On
Drink_Done	Order.getNextOrder() == null		Wait_Request
	Order.getNextOrder() != null && !CupTaken		Pumping_On
	Order.getNextOrder() != null && CupTaken		Drink_Done
Pumping_Error	Errors !=0		Pumping_Error
	Errors ==0		Wait_Request



Previous State	Action
Pumping_On	V_Pump[tank_gpio] = ON
Pumping_Off	V_Pump[tank_gpio] = OFF
Wait_Request	TransferData()
Drink_Done	
Pumping_Error	isLeaking() || isSafeTempature()

4.6.2 Class Pump

Uses

- DrinkCals

Internal Values DrinkCals coke: Structure holding the calibrations related to Coke products **DrinkCals water**: Structure holding the calibrations related to Water.
Functions public Pump():void Initializes the values of the pumping module **public pumpLiquid(Drink): Integer** Turns on the pump for the specific drink type. **public stopPump(Drink): Integer** Turns off the pump for the specific drink type. **public isComplete(): Boolean** Determines if the drink has been completely filled or not based on the following equation: $Filled := M_min < M_cup$ **public isLeaking(): Boolean** Determines if the containers are losing fluid when there is no pumping based on the following equation: $Leaking := [M_minleak < (M_container1 - M_container_prev1) \wedge Pin7 == 0] \vee [M_minleak < (M_container2 - M_container_prev2) \wedge Pin8 == 0]$
public isSafeTempature() Determines if the containers are still storing the liquids at a temperature greater then the minimum temperature for the liquids based off of the following equations. $OverTempature := (T_container1 < T_min) \vee (T_container2 < T_min)$

4.6.3 Communication Class

Uses

- DrinkCals

Internal Values None
Functions
transferData(Integer): Order
Communication performed used using UART where Orders are received and errors are transferred. The first 32 bits will be the errors associated with the pumping system and the last bit will be if the robot is done.

5 Scheduling

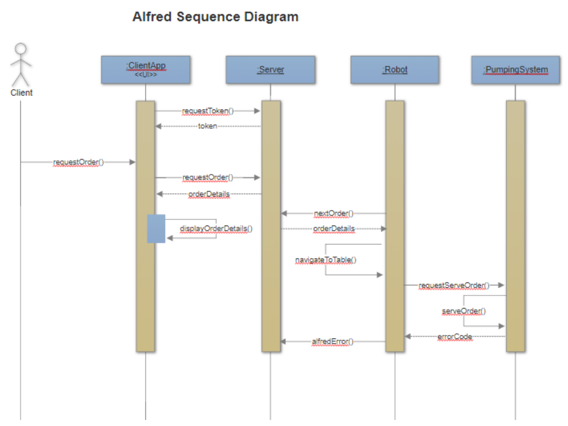


Figure 4: Alfred Sequence Diagram

6 Design Notes

7 Data Dictionary

8 References