Final Report - PolyDrop

Dr. Ben Hawkins
Zach Scott and Lilly Paul
ztscott@calpoly.edu, ljpaul@calpoly.edu

Table of Contents

Executive Summary

This report contains explanations and details of the background, engineering design and qualifications, and the final results of the PolyDrop system project. In nearly 20 weeks, the team developed benchmarks and specifications to make a software interface that would control existing hardware to move droplets in a customizable fashion across a platform. After developing the software product to set specifications, it was tested against specific benchmarks. Results show that the PolyDrop system meets all requirements, is a reliable system, and enables researchers and students to study the field of microfluidics more effectively.

Introduction

Project Overview

Our goal is to create a product that will utilizes the OpenDrop system and allows the user to move a droplet across the grid of multiple spaces with one control. Already, the OpenDrop product is built using electro-wetting technology that moves droplets across the grid in an atomic fashion. The hardware is portable and allow's users to conduct point of care testing with desired liquids. The system utilizes manual buttons to control the droplet across the grid, forcing the user to control the path of the liquid. In addition, a software interface to OpenDrop, MicroDrop provides a "point and click" control using a computer and mouse. In addition, MicroDrop relays live visual feedback to the user, and both functions greatly improve the user experience. Our goal is taking this to the next step, allowing the user to preprogram a path for a droplet(s) on OpenDrop's hardware and provide image analysis feedback from a webcam. Our ultimate deliverables is a product that allows users to design paths for the droplet movement, then physically moving the droplet across the grid in that desired path. As well as detecting the drops with the webcam and providing information like color and location on the grid to the user. This functionality will greatly improve the user experience and provide many benefits to the existing system.

Client and Community Partners

For our product, there are three main established community partners. First, DropBot, an open-source Digital Microfluidic automation system. The digital microfluidic system originates from DropBot, which has an established community supporting those who wish to improve it. DropBot was published in 2013 and the community has been growing since. The DropBot is a larger system that is not lightweight, however it offers more user control and real-time measurement of instantaneous drop velocity. The second partner is OpenDrop, the smaller and lightweight version of DropBot. OpenDrop Version 1 was published in 2015, with version 2.0 developed in 2016. OpenDrop is an open-sourced product and is developing a community for support. OpenDrop will serve as the base for our project, and we will be working closely with their open source libraries. We obtained a working OpenDrop system from Gaudi labs in order to avoid having to build one ourselves. The third partner is MicroDrop, the software that layers OpenDrop to create point and click control. Microdrop is open-source as well and they provide their source code online. We have come to the realization that working on top of the Microdrop software is not a possibility so we are basing our UI off of MicroDrop's. Overall, our product consists of these three main community partners and one direct beneficiary being Dr. Hawkins.

Project Goals and Objectives

The original goal of the Capstone project was to create an interface that controlled the Open Drop hardware and allowed users to move multiple drops on multiple paths with just a few clicks. This was accomplished with the AutoDrop software. The project was extended into our senior project because there was still a lot of work to be done polishing the interface and adding new features. On the GUI side, the software did not account for drop collisions, had a storage problem because the arduino could only handle a set number of byte arrays at a time, and took too many button presses to actually create and execute a path. The goal of Zach's work in the project was to fix these problems and make sure the GUI ran smoothly with the addition of the new features Dr Hawkins wanted us to add. The goal of Lilly's work was to add image analysis in order to provide feedback of the drops. The image analysis needed to be able detect drops on the grid, as well as provide information like their location on the grid and color. Zach and Lilly then worked together to integrate the image analysis into the existing GUI.

Project Outcomes and Deliverables

At the end of the Winter Quarter 2016, the Auto Drop software was complete as a capstone project and featured multi-drop control, multi-path options, serial communication with the Open Drop arduino, arduino storage of all the paths before execution, and the ability to delete paths. What it lacked was a good collision detection system and the ability for one drop to absorb another when a collision occurred. It also had a storage space issue on the arduino side of the project, limiting it to set number of communications per cycle, meaning that there was a limit on how many drops could be stored and how many paths could be run at once. With the addition of the video analysis features, we decided to change the path execution from whole path communication to one square at a time movement. This allowed us to run the image analysis and drop detection after every square of movement to determine if the position and color had changed and highlight them on the screen. What we didn't foresee is that this change cause the program to run incredibly slow. We had to wait a whole 4 and a half seconds between each movement. On long paths the program took too long, making our software frustrating for the user. In order to fix this, we entirely revamped the arduino code that was written in capstone to run the paths when they were received instead of storing them to execute later. This change sped up the program greatly, making it so the biggest delay was the image analysis, which was the ideal speed for maximum usability. It also solved the memory issues we had encountered during the testing of the capstone project. In testing our new software, renamed Poly Drop, we did not reach any limit in amount of paths that could be run at once. The image analysis portion of the project also was completed to the desired specifications. The analysis runs after every square movement and data about drop locations and colors is returned to the user.

Background

The Applications and Uses

This project focuses on the bioassay analysis concentration of microfluidics. Most cell assay studies focus on cell differentiation and cell response to foreign substance. Currently, this research is done by studying the response of multiple cells (colonies). This method has been challenged for its accuracy, because it relies on a mean of responses. However, using single samples through microfluidics can provide similar results and eliminates the need for research on an entire cell population. Instead, if one cell can be studied through electrowetting technology, cell assay analysis can become much more precise.

Lab-On-A-Chip Initiative

Digital microfluidics (DMF) is a specific and sub category of microfluidics. DMF has been researched extensively by Aaron Wheeler of the Wheeler Microfluidics Laboratory, who has been one of the pioneers of the “lab-on-a-chip” initiative. DMF mainly involves an electrode array which provides different potentials to specified cells on the array. In effect, the voltage potential changes the shape of a certain liquid droplet and moves in the direction of the force applied by an adjacent potential. A general depiction of this process is below.

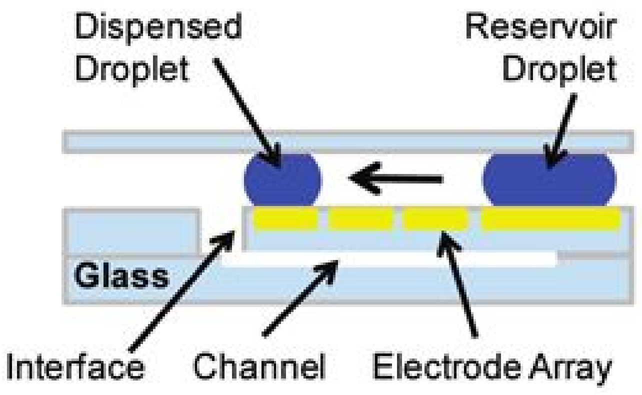


Figure 1: Illustration of digital microfluidics (Wheeler microfluidics Laboratory)

How OpenDrop Digital Microfluidics Works

Open Drop uses digital microfluidics to store, move, transfer, combine and split droplets. It provides this functionality by utilizing high voltage and electronic circuitry. By using a change in high voltage across the grid, droplets can move from node to node. This technique is utilized by separating the hardware into different layers of the circuit.

The different layers include the electrode grid, a hydrophobic surface carrier, the drops themselves and a conductive with grounded side of glass. These layers and their function are explained below.

The first layer is the electrode grid. The grid outputs 190-330 volts onto specific grid nodes which is controlled by the Arduino firmware. If an grid node is off, the node rests at 0 volts. The grid is 16 x 8 nodes, allowing the board to have multiple drops move in different paths. This layer is the beginning the circuit.

The next layer is a hydrophobic surface carrier, placed on top of the electrode grid. This layer serves two purposes. First, to separate the liquid from the high voltage electrode grid which prevents damaging the circuitry. Second, by providing the liquid a surface that has very low friction to easily move the drops. This layer is essential to allow liquid to translate across the grid.

The third layer is the liquid under testing. The simplest liquid to test with is water.

The fourth layer is a glass sheet with the conductive side facing the electrode grid. The ground side of glass faces the outside or user. This final layer completes the circuit.

By preparing the OpenDrop hardware with these layers, the drops are able to move when the electrode grid changes voltage from node to node. The high voltage at one node forms an electric circuit through the hydrophobic surface carrier, the droplet and the glass. When the high voltage node turns off and the neighbor node goes high, the droplet should translate to the new high voltage node. By using this technique and with adequate preparation, the liquid can move anywhere on the grid.

History of OpenDrop

OpenDrop is based from the DropBot digital microfluidics platform. DropBot provided the initial prototype and functionality and OpenDrop improved it with design and portability. OpenDrop was developed using the lab on a chip concept, where point of care tests could be run in essentially any environment with quick results. Thus, the primary improvement is the size of the machine. Dropbot is a large, bulky, multi-piece machine, whereas OpenDrop can fit in one's hand. In addition, OpenDrop requires a lower operating voltage than DropBot. While DropBot and OpenDrop are different, the key takeaway is to use DropBot's active community to our advantage. DropBot has been around longer, thus has faster and more consistent resources we can use for OpenDrop.

Building Our OpenDrop

A first step for our team was replicating OpenDrop's hardware to a working prototype that we could develop on. The easiest solution was to contact Dr. Gaudenz, the founder of OpenDrop, and ask if he would send us a prototype from his lab. The second method, involved searching through the OpenDrop website, locating manufacturing files and finding a company that will create the pcb and solder the board. Thankfully, Dr. Gaudenz send a prototype OpenDrop board that we developed on.

OpenCV

We used OpenCV in the development of our image analysis. OpenCv (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. It was build to promote and accelerate the use of machine perception in products. It cross platform and a BSD-licensed product, making it easy for businesses or individuals to utilize and modify the code. It is used widely in the industry by companies like Google, Microsoft, and Intel. OpenCv is developed in C++, but had C, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. We developed with the Java interface. We are currently using version 3.2.0.

Engineering Specifications

The Personas

Below are the description and details for the two main personas concerning the PolyDrop system. An extensive list containing background and characteristics are provided for each persona followed by a brief summary. Based from the personas, use cases were developed and defined. A detailed description for the important cases are provided at the end of the section.

Persona 1 : “Researcher”

- Background
    - researcher
    - developing new information on bioassay analysis
    - developed in the bio field
- Behavior patterns
    - automates things
    - doesn't want/need to do step by step
- Goals
    - run a speedy test for bioassay, on-site
    - receive quick results and answers
    - quickly interpret results
- Skills
    - how to personally adjust settings on device
    - biomedical to interpret results
- Attitudes
    - time efficient
    - very interested in detailed reports
        - to understand and develop from results
    - “Give me quick, accurate and detailed results”
- Environment
    - inside a research facility
    - possible application-based/publishing based
    - time-sensitive manner
    - multiple tests for validity

Persona 2 : “Student”

- Background
  - education about device usage
  - education involving bioassay results
- Behavior Patterns
  - follows the step by step approach/information
  - no specific pattern
- Goals
  - testing for educational purpose (rather than research)
  - reproducing previous results
- Skills
  - minimal knowledge for biology (shouldn't be designing custom test)
  - minimal knowledge on technology
- Attitudes
  - possibly impatient
  - "Let me just get through this lab"
- Environment
  - education setting
  - tests are not as critical
  - not as detailed reports
  - multiple devices in the room for groups

Brief Descriptions for Personas

1. Researcher

The researcher is an individual who wants to use this product to test their hypothesis via point of care testing. The researcher is interested in running repeated tests and programming them in a an automatic fashion. He/she is interesting in the data of the test, information about paths and grid manipulation and the resulting droplets. Finally, he/she will need to change the administration settings to fine tune the test to specific needs.

2. Student

The student is an individual who wants to learn how to use OpenDrop and operate point of care testing. The student is interested in operating the product at a basic level, simply manipulating drops around the grid. The student is also interested in running the OpenDrop product without running into many problems.

Use Cases

1. The user manually controls droplets via physical buttons on the OpenDrop hardware.

2. Point-and-click manipulations of droplets using the microdrop software and a computer.

3. Users pre-programing paths for a droplet(s) using the PolyDrop user interface.

4. Pre-programming the path of a single droplet to specific grid points.

5. Specifying the merging or splitting of multiple droplets using the PolyDrop user interface.

6. The user wants to see how different colors of drops combine together

Detailed Description of Use Cases

Use Case 1: Pre-programming a path using PolyDrop user interface.

The user wants to create a point of care test with specific liquids. He approaches the product, turns it on, boots it up and inserts his liquids onto the grid. After getting the basics set up on the interface, it is time for the user to create paths for the droplets to move. Instead of manually controlling the movement of the drops, (s)he wants to easily automate this process. This way the test runs faster, doesn't require constant attention and the test can be saved for later usage.

First, the user will select a button to control a specific drop on the grid. Then, the user will be able to mark grids points where the drop will move across. The path from start to end can be adjusted by the user. After creating the desired path, the user will hit complete. The user can do the same for others droplets as well. After completing his test, by preprogramming all desired paths with liquids, the user will click "End Path." Finally, the drops will move according to the paths. After the test is complete, the user can extract the droplets and remove the film atop the grid.

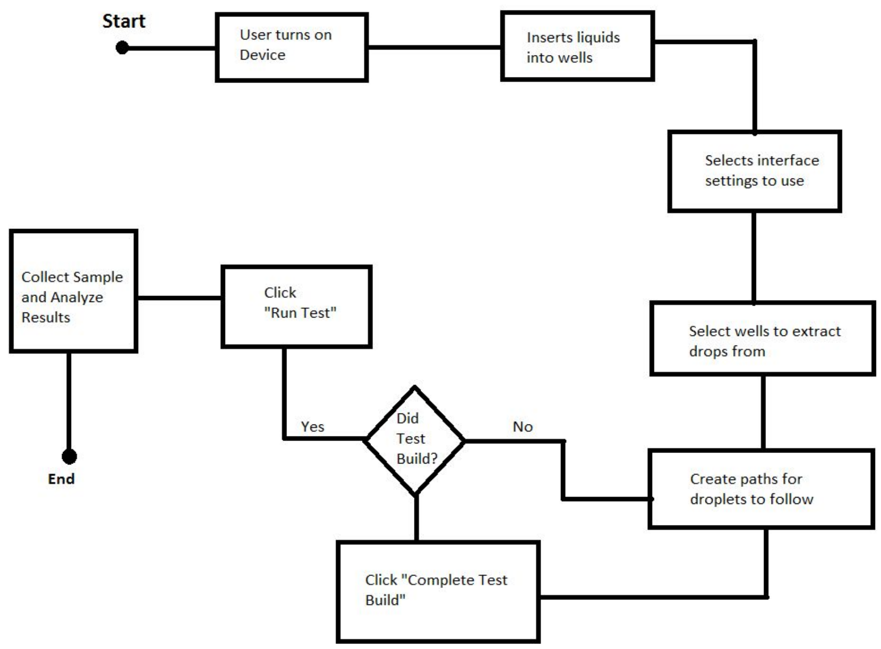

Start
User turns on Device
Inserts liquids into wells
Selects interface settings to use
Select wells to extract drops from
Create paths for droplets to follow
Did Test Build?
No
Yes
Click "Complete Test Build"
Click "Run Test"
Collect Sample and Analyze Results
End

Figure 2. UML Diagram from Use Case 1

<u>Use Case 2: Point-and-Click manipulation of droplets using the PolyDrop software.</u>

In order to use the point and click functionality, the user first powers up the OpenDrop hardware and visually confirms that it receives an adequate amount of power. Next, the user connects OpenDrop to a computer with the associated microdrop software installed. Next, they place a hydrophobic surface carrier over the OpenDrop electrode grid. After, they carefully place the liquid droplets on the electrode grid. The user leaves the hardware and refers to the microdrop software where they verify that they liquid is visible on the UI. Next, they would input the paths for the test they are about to run). He proceeds to begin testing by pointing and clicking on the camera interface that microdrop provides. After testing, the final steps would be to extract the liquid from OpenDrop for either disposal or further analysis, and then finally powering down OpenDrop and PolyDrop.

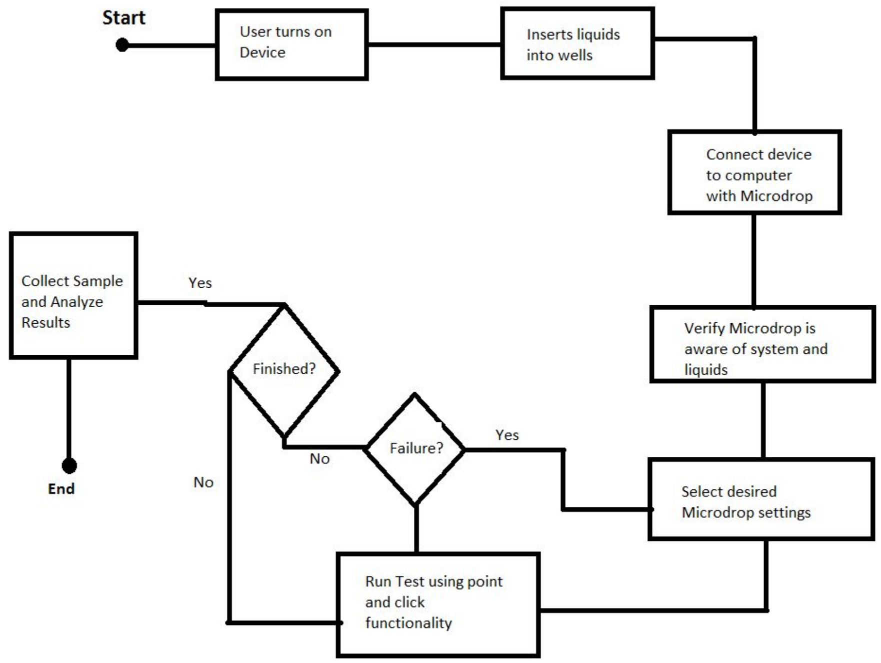

Start
User turns on Device
Inserts liquids into wells
Connect device to computer with Microdrop
Verify Microdrop is aware of system and liquids
Select desired Microdrop settings
Run Test using point and click functionality
Failure?
Yes
No
Finished?
Yes
No
Collect Sample and Analyze Results
End

Figure 3. UML Activity Diagram for Use Case 2

Engineering Specifications

Derived from the personas and use cases, the below engineering specifications were made and met via thoughtful design and engineering. After assessing the use cases, the below specifications outline what the team needed to make in order to accomplish the client's needs. As a team, we created seven engineering specifications near the beginning of this quarter.

1. From the interface, the user can move a droplet from selected starting node to end node.
2. The user can create multiple paths with different drops.
3. The user will be automatically provided a path by selecting a start and end node.
4. The complete system will run for 2 hours without crashing or errors.
5. The Arduino and interface will communicate through reliable serial connection.
6. The user can access a history of previous run paths.
7. The user can add drops anywhere on the electrode grid through the interface.
8. The interface can detect drops on the grid from a webcam and image analysis
9. The user has access to the color and location of each drop detected
10. The interface displays a current image of the grid and drops from the webcam

Final Design

Functional Decomposition

The functional decomposition explanation will follow the top-down approach by introducing modules and sub-modules in various diagrams. This document utilizes functional decompositions for the hardware and state diagrams for the software.

Level 0:

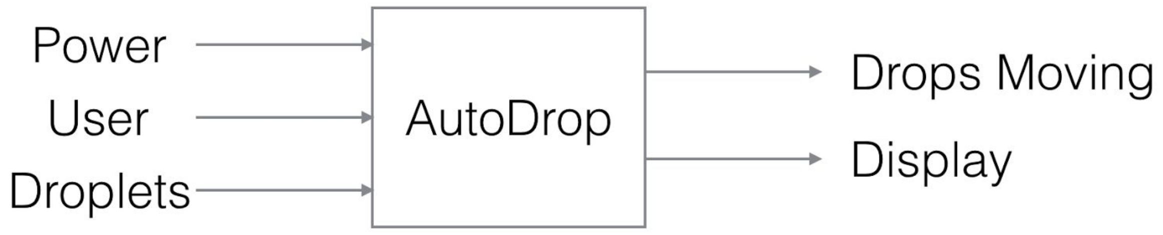


Figure 3: Level 0 Design

At the highest level, the basic inputs are power, user interaction and droplets onto the board. From there, the PolyDrop will coordinate the user interactions and develop methods to move the drops in a certain path. The result will be drops translating across the grid and the display reflecting the movement.

Level 1:

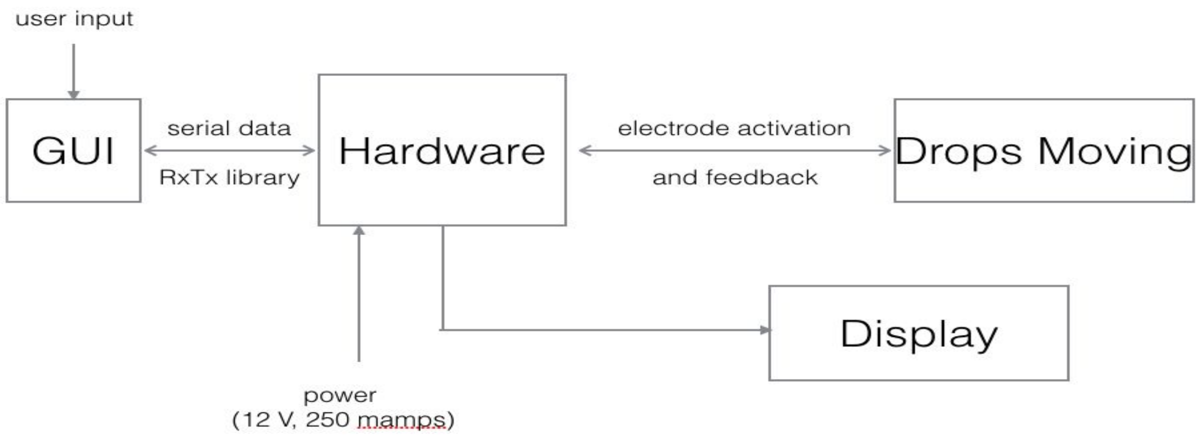


Figure 4: Level 1 Design

The main modules are depicted in Figure 4 and the core module is the Arduino hardware. The decomposition begins with power to the hardware and input from the user on the interface. Using serial communications, the GUI will instruct the hardware what to do. Then, the hardware will move the drops across the grid as the user specified. To validate the correct movement, the electrode voltage will change and the display will visually depict drop movements. At the level one, there are 4 main modules, each uniquely responsible to allow the user to move the drops.

Looking at the application as a whole, our specific project and goals are focused on developing a software layer to create easy interaction between the user and hardware. Essentially, our time was focused on developing an interface to communicate to the hardware. Thus, the next diagrams are software based figures which explain the code flow and states.

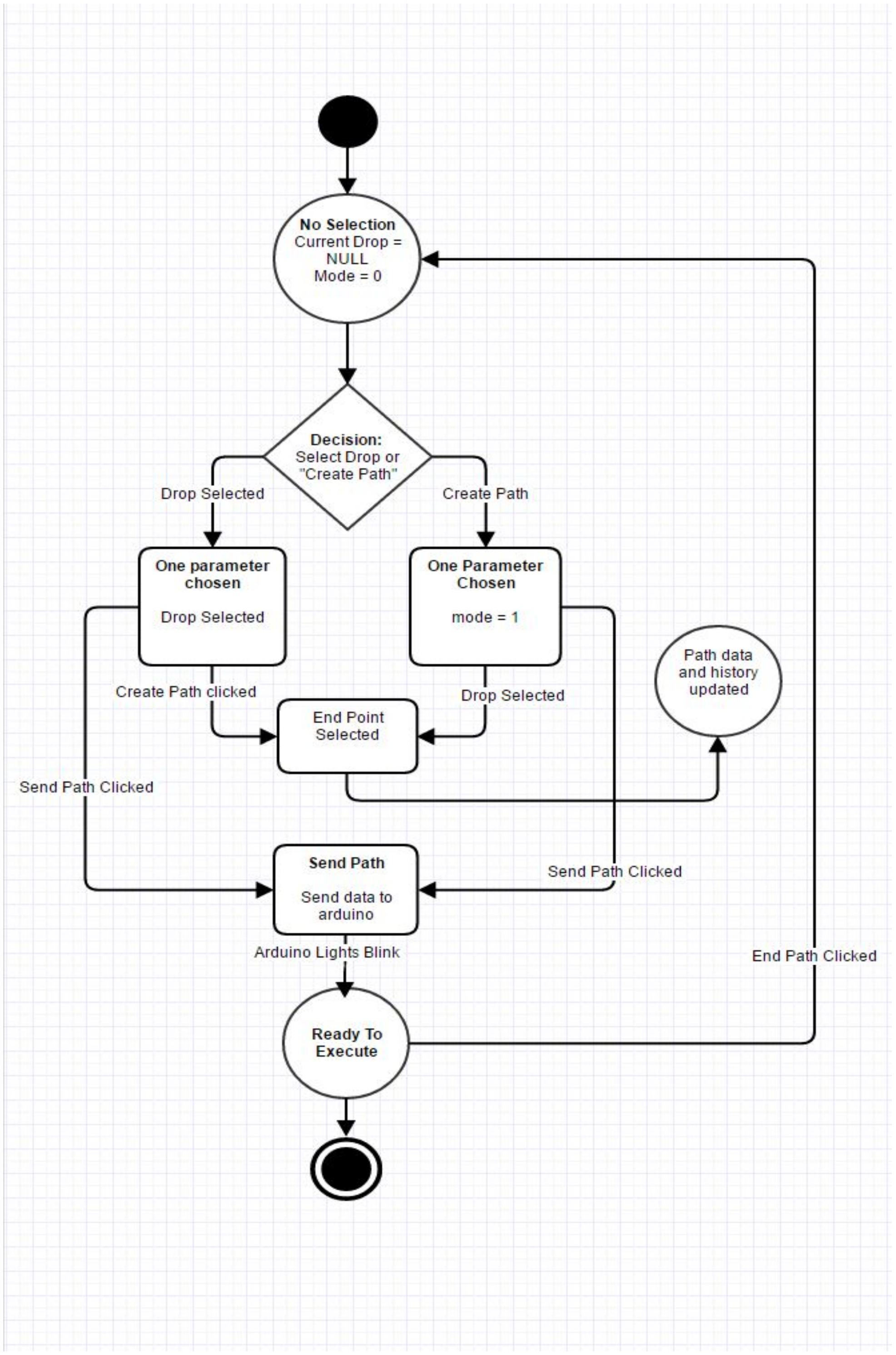


Figure 5. Data Flow Diagram for Java Interface

The above diagram shows the state diagram for the GUI implementation. The program uses a mode variable to determine what state it is in and when it should perform certain tasks. It starts off with the mode equal to zero with no drops selected and the create path button not pressed. The user has a choice of either selecting a drop or clicking the create path button, but both must be done before a path is able to be created. Once a drop is selected and the user has specified that they want to create a path, they must select an endpoint for that drop to move to by clicking one of the grid buttons. Once the grid button is pressed, the colors on the grid will change to indicate the new path and the history list will update. The mode will be set back to zero. The user then has a choice of creating new paths or sending the path data to the arduino for execution. If the send path button is pressed, the GUI will send the current path data over the serial connection and the LED's on the OpenDrop system will flash when the data is received and processed. The user must then hit the End Path button to have the arduino execute the specified paths. After the end path button is pressed, the gui goes back to its original state, but with the new updated drop positions.

We have added a feature allowing drops to be added to the board after the execution has started. It is not in the above state diagram, but if a user presses the Add Drop button and then clicks on a grid square, a drop will be added to both the GUI display and OpenDrop display.

Below is the originalstate diagram for the Arduino. In the first iteration, the arduino functioned as follows. Since the Arduino reacts to the serial communication it receives, it makes sense to use a state diagram which labels the different states the Arduino will enter.

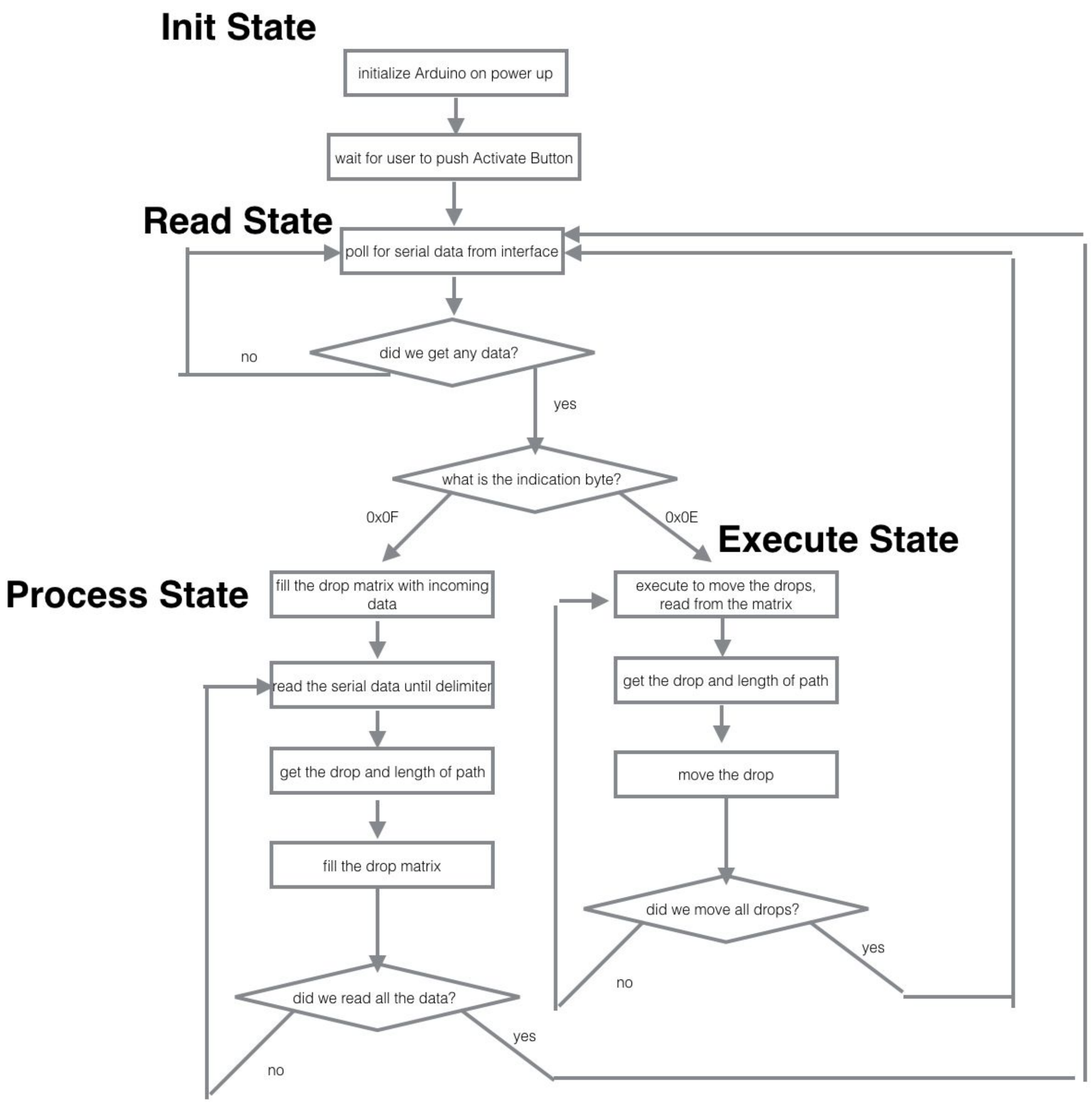

Init State
initialize Arduino on power up
wait for user to push Activate Button
Read State
poll for serial data from interface
did we get any data?
no
yes
what is the indication byte?
0x0F
0x0E
Process State
fill the drop matrix with incoming data
read the serial data until delimiter
get the drop and length of path
fill the drop matrix
did we read all the data?
no
yes
Execute State
execute to move the drops, read from the matrix
get the drop and length of path
move the drop
did we move all drops?
no
yes

Figure 6. State Diagram for Arduino software

Label 6 indicates there are four main states the Arduino is in: Init, Read, Process and Execute. The important states are the last three. In summary, the Read State will poll for data from the serial connection. Once there is data available, the Process State will read the information and fill a 2 dimensional array, drop by paths movement, with the data. Finally, when the execute byte is sent, the Execute State will read from the matrix and move the drops across the grid as specified.

With the change to execute paths automatically, we greatly cleaned up the logic of the state diagram above and in doing so, massively sped up the execution speed of the entire program.

Image Analysis Design

In our image analysis code we start by accessing a USB webcam, take a real time image. Then do a background subtraction with a pre-specified background image. Both images are preprocessed to reduce noise usually from light variability. The background subtraction produces a foreground mask. On the foreground mask we then perform an edge detection. With the resulting contours from the edge detection, we create a bounding rectangle around the contours. This bounding rectangle we overlay back on the original input image. Next, we compute the center point of the rectangle to calculate where it is on the OpenDrop grid, and store that location in column/row format. Then, we compute the mean color of the pixels in the bounding rectangle and store this color. Lastly, we return the original image with the bounding rectangle drawn on it (in the color detected) and display in the GUI. We also return the location and color data for each detected drop. You can see this software flow in the Figures below.

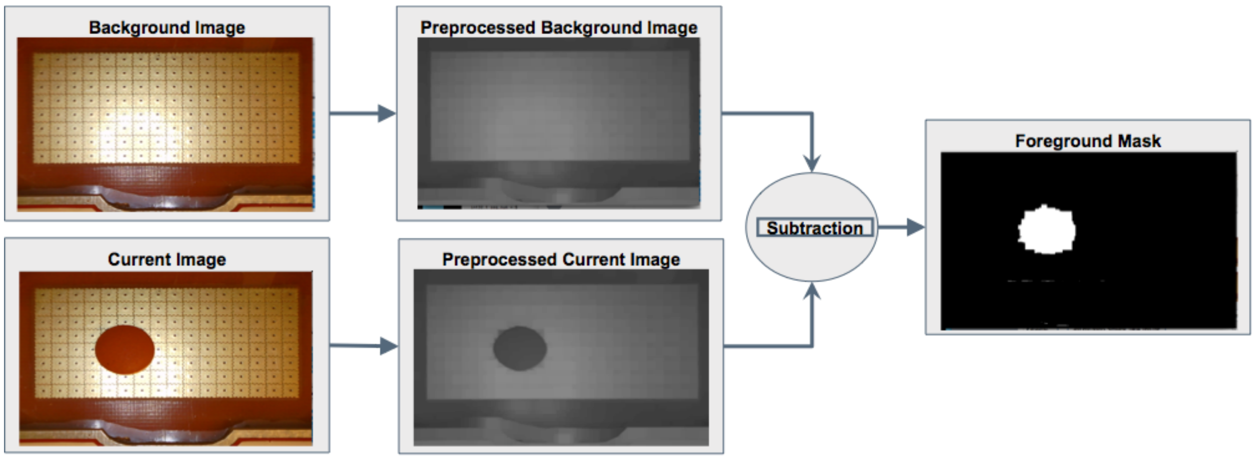


Figure 4: Background Subtraction Flow

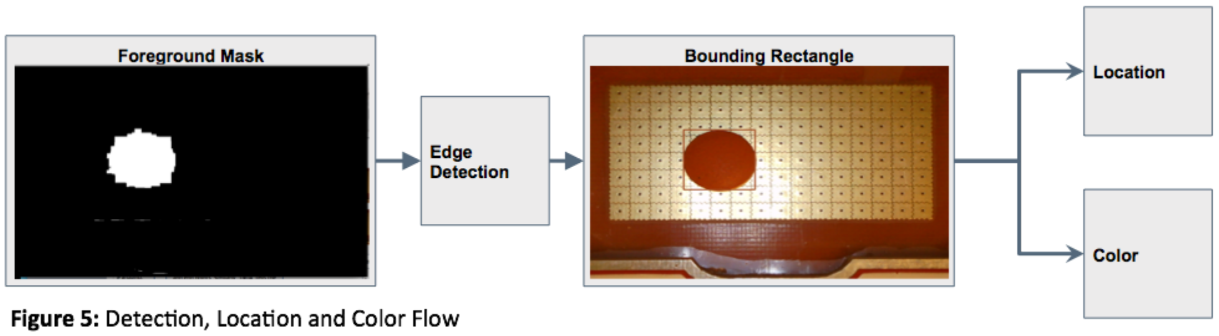
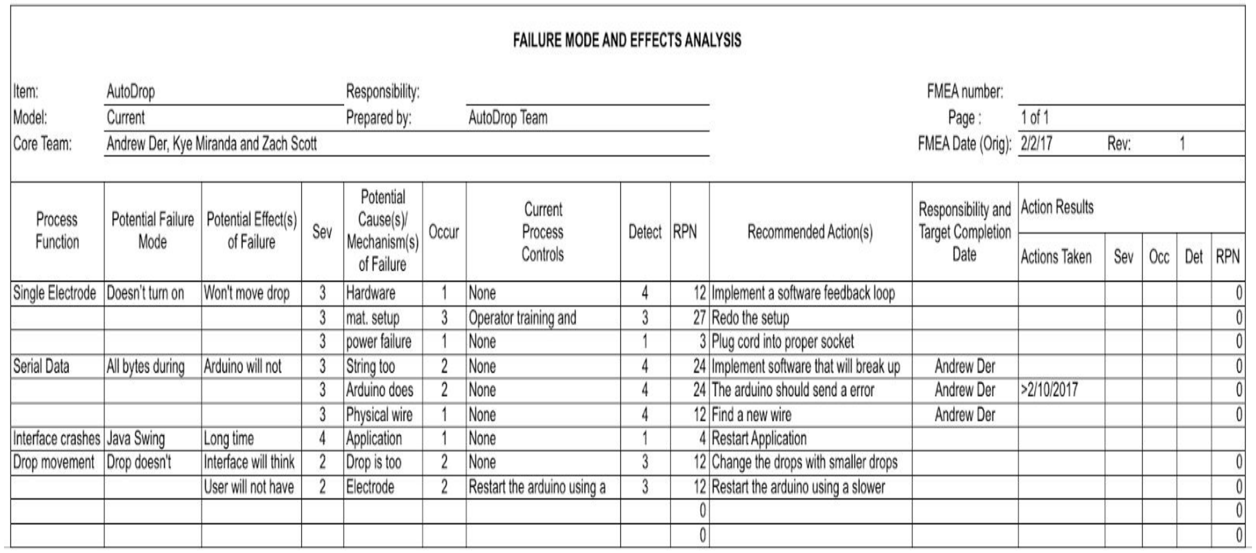


Figure 5: Detection, Location and Color Flow

Design Verification

Failure Mode Effects Analysis (FMEA)



FAILURE MODE AND EFFECTS ANALYSIS

Item:	AutoDrop	Responsibility:		FMEA number:	
Model:	Current	Prepared by:	AutoDrop Team	Page :	1 of 1
Core Team:	Andrew Der, Kye Miranda and Zach Scott			FMEA Date (Orig):	2/2/17 Rev: 1

Process Function	Potential Failure Mode	Potential Effect(s) of Failure	Sev	Potential Cause(s)/ Mechanism(s) of Failure	Occur	Current Process Controls	Detect	RPN	Recommended Action(s)	Responsibility and Target Completion Date	Action Results				
											Actions Taken	Sev	Occ	Det	RPN
Single Electrode	Doesn't turn on	Won't move drop	3	Hardware	1	None	4	12	Implement a software feedback loop						0
			3	mat. setup	3	Operator training and	3	27	Redo the setup						0
			3	power failure	1	None	1	3	Plug cord into proper socket						0
Serial Data	All bytes during	Arduino will not	3	String too	2	None	4	24	Implement software that will break up	Andrew Der					0
			3	Arduino does	2	None	4	24	The arduino should send a error	Andrew Der	>2/10/2017				0
			3	Physical wire	1	None	4	12	Find a new wire	Andrew Der					0
Interface crashes	Java Swing	Long time	4	Application	1	None	1	4	Restart Application						
Drop movement	Drop doesn't	Interface will think	2	Drop is too	2	None	3	12	Change the drops with smaller drops						0
		User will not have	2	Electrode	2	Restart the arduino using a	3	12	Restart the arduino using a slower						0
								0							0
								0							0

Figure 7. Failure Mode and Effects Analysis (larger image in Appendix D)

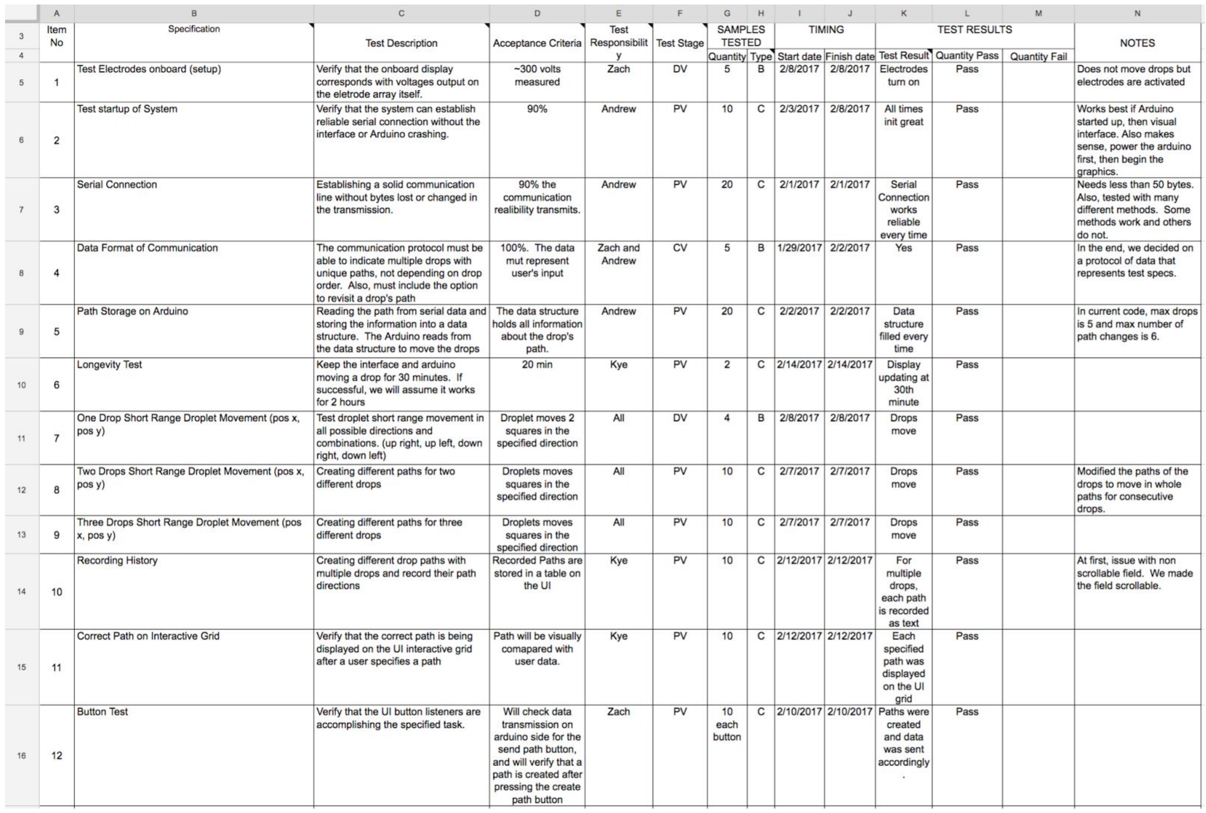
The chart above shows our Failure Mode Effects Analysis table for PolyDrop. It displays possible malfunctions with the system along with information about the severity of failure, likelihood of occurrence, causes, recommended action to correct the problem, and more. The failures were split into 4 categories: Single Electrode, Serial Data, Interface crashes, and Drop movement. There are multiple failure cases for each category. This diagram was used to diagnose errors that occur while the system is in use. The two large process functions are activating an electrode node and sending a serial data which are be explained below.

If the electrode grid is unable to output high voltage on a node, the drop will not move to the desired spot. This failure can happen due to a hardware malfunction, bad material setup or a power failure. Fortunately, solutions are available for every situation.

The second large process function prone to error is sending serial data. If the data is too long, the Arduino is not initialized or if the physical wire is broken, the Arduino will not receive instructions from the interface. Like above, the chart outlines the recommended action for each situation.

Overall, there are 9 process functions that can fail and each has a recommended action. The least dangerous process is the hardware not receiving adequate input power. If this occurs, the hardware cannot output enough voltage in order to move the drop. The solution is to make sure the power converter gives 12 V DC and 250 mA (min) to the Arduino. The most dangerous process is the material setup for the initialization of the hardware. If the material is not set up correctly, for example not enough Rain-X or space between the hydrophobic surface carrier and grid, the drops will not move. In our experience, this happens very often and it is very difficult to move a drop. The solution is to either add more Rain-X to the surface wrap or restart the setup process entirely.

Design Verification Plan



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
3	Item No	Specification	Test Description	Acceptance Criteria	Test Responsibility	Test Stage	SAMPLES TESTED		TIMING		TEST RESULTS			NOTES
4							Quantity	Type	Start date	Finish date	Test Result	Quantity Pass	Quantity Fail	
5	1	Test Electrodes onboard (setup)	Verify that the onboard display corresponds with voltages output on the eletrode array itself.	~300 volts measured	Zach	DV	5	B	2/8/2017	2/8/2017	Electrodes turn on	Pass		Does not move drops but electrodes are activated
6	2	Test startup of System	Verify that the system can establish reliable serial connection without the interface or Arduino crashing.	90%	Andrew	PV	10	C	2/3/2017	2/8/2017	All times init great	Pass		Works best if Arduino started up, then visual interface. Also makes sense, power the arduino first, then begin the graphics.
7	3	Serial Connection	Establishing a solid communication line without bytes lost or changed in the transmission.	90% the communication realibility transmits.	Andrew	PV	20	C	2/1/2017	2/1/2017	Serial Connection works reliable every time	Pass		Needs less than 50 bytes. Also, tested with many different methods. Some methods work and others do not.
8	4	Data Format of Communication	The communication protocol must be able to indicate multiple drops with unique paths, not depending on drop order. Also, must include the option to revisit a drop's path	100%. The data mut represent user's input	Zach and Andrew	CV	5	B	1/29/2017	2/2/2017	Yes	Pass		In the end, we decided on a protocol of data that represents test specs.
9	5	Path Storage on Arduino	Reading the path from serial data and storing the information into a data structure. The Arduino reads from the data structure to move the drops	The data structure holds all information about the drop's path.	Andrew	PV	20	C	2/2/2017	2/2/2017	Data structure filled every time	Pass		In current code, max drops is 5 and max number of path changes is 6.
10	6	Longevity Test	Keep the interface and arduino moving a drop for 30 minutes. If successful, we will assume it works for 2 hours	20 min	Kye	PV	2	C	2/14/2017	2/14/2017	Display updating at 30th minute	Pass		
11	7	One Drop Short Range Droplet Movement (pos x, pos y)	Test droplet short range movement in all possible directions and combinations. (up right, up left, down right, down left)	Droplet moves 2 squares in the specified direction	All	DV	4	B	2/8/2017	2/8/2017	Drops move	Pass		
12	8	Two Drops Short Range Droplet Movement (pos x, pos y)	Creating different paths for two different drops	Droplets moves squares in the specified direction	All	PV	10	C	2/7/2017	2/7/2017	Drops move	Pass		Modified the paths of the drops to move in whole paths for consecutive drops.
13	9	Three Drops Short Range Droplet Movement (pos x, pos y)	Creating different paths for three different drops	Droplets moves squares in the specified direction	All	PV	10	C	2/7/2017	2/7/2017	Drops move	Pass		
14	10	Recording History	Creating different drop paths with multiple drops and record their path directions	Recorded Paths are stored in a table on the UI	Kye	PV	10	C	2/12/2017	2/12/2017	For multiple drops, each path is recorded as text	Pass		At first, issue with non scrollable field. We made the field scrollable.
15	11	Correct Path on Interactive Grid	Verify that the correct path is being displayed on the UI interactive grid after a user specifies a path	Path will be visually comapared with user data.	Kye	PV	10	C	2/12/2017	2/12/2017	Each specified path was displayed on the UI grid	Pass		
16	12	Button Test	Verify that the UI button listeners are accomplishing the specified task.	Will check data transmission on arduino side for the send path button, and will verify that a path is created after pressing the create path button	Zach	PV	10 each button	C	2/10/2017	2/10/2017	Paths were created and data was sent accordingly.	Pass		

Figure 8. Design Verification SpreadSheet (larger image in Appendix D)

Above is the design verification plan which details how the beta prototype was tested. We tested the product after completing the beta prototype in order to identify mistakes and limitations. In general, there are tests for functionality, robustness, reliability and longevity. Most testing included functionality and determining if multiple drops could be moved with multiple paths. Another main testing focus included testing the byte limit of the serial communication protocol. In summary, the tests contained everything from proper initialization, reliable functionality and stress of use. The image analysis was developed and tested separately. Then once working to our satisfaction, we integrated it and performed system tests.

The first major test proceeds to try every combination of initialization steps, only to reveal that a specific order is required. Since there are two different parts to the system, Arduino and computer, it is important to discover the correct way to initialize the system. Correct initialization assures that the interface recognizes and communicates with the hardware with a reliable protocol.

The second test finds the maximum number of drops and spaces the Arduino can handle on the grid. Since the Arduino is a microcontroller with limited memory space, there exists a maximum of usable space. By finding this limit, we can develop PolyDrop such that the user is aware of this limitation.

The third main functional test proceeds to create different and complex paths for multiple drops on the board. This is the main functional test and incorporates the most engineering specifications. Under this testing, we created simple and complex paths with single and multiple drops and programmed them into the board. We used the maximum number of drops and path length possible discovered from above tests.

Other functional tests include if a history of paths is available, if the buttons on the GUI work, how long the system can run without error, and drop detection. The results of these tests are explained in the next two sections.

Verification Test Results

Test	Description	Notes	Verification
One drop path movement	Created 10 paths with one drop with varying directions and lengths	All paths performed as expected	✓
Multiple drop movement	Created 5 paths with 3 different drops, varying order and length	Drops finish all of their paths first, not in order that user specifies. This is by design.	✓
Collision detection test	Test collision detection between drops going multiple directions with different end points	Colsion Detection now works fully, with prompts to the user asking whether they want to combine drops or not.	✓
Delete All test	Test the delete all functionality	Delete All removes all paths, clears the path data and resets drop positions	✓
Delete Last test	Test the feature that deletes the last path in the history	Delete last deletes one path from the list and updates the data and grid colors	✓

Figure 9. GUI Software Test table

All of the features we implemented passed all of the tests we ran with some minor tweaking except the collision detection tests. There was problems with some of the edge cases in the collision detection system, but most of the collisions are detected by coloring the square of the collision red. The problem with the beta version of our software is that it detects the collision, but does not handle the merge by updating the drop list to show only one drop. In our user manual, we recommend that a user should delete the last path when a merge is detected. The fix for the collision detection and drop merging bugs will come in the continuation of our project as Zach's senior design project.

Test	Description	Notes	Verification
Drop Detection - Image Analysis	Test can accurately detect multiple drops	Was able to recognize objects in sized drops in different lighting.	✓
Drop Location - Image Analysis	Test returns the correct row, column for the drop location on the electrode grid	Correctly identifies location on the grid the drop covers.	✓
Drop Color - Image Analysis	Test can accurately detect color of the drop with multiple different colors	Reports back RGB colors values, visually colors the rectangle around the drop with the detected color. Tested with all major colors (red, orange, yellow, green, blue, purple, pink, grey, brown, black, white, and clear drops). Clear drops had mostly blue values.	✓
System Integration	Test the image analysis still worked as expected when called by the GUI. As well as testing the processed image is displaying properly in the GUI.	Image analysis works as expected. Processed image is displayed next to the UI grid. With the processed image displayed the GUI is too large for my MacBook, however it works on Zach's PC with a larger screen.	✓
Movement Feedback	Terminate the path if the image analysis does not move the drop	We were unable to test this properly, as the Open Drop board is broken and cannot move drops.	!

Figure . Image Analysis Software Test table

All image analysis tests passed, we were able to detect multiple drops, with different colors, and sizes. The biggest issue we faces was variability in lighting would affect the whole drop detect. Since we are looking a such small objects (the drops), even a small variation would throw off the whole process. This was fixed by performing preprocessing on both the background image and current image. The preprocessing included dulling, burring, and eroding the pixels in the masked image.

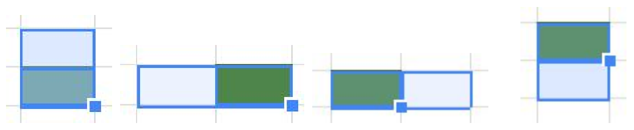
GUI Bugs Found:

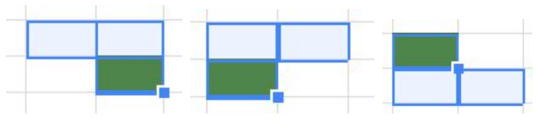
- Had a problem where the corner square would not be colored yellow when creating a path with both x and y movement negative - Fixed
- Had a problem where white squares showing previous position would not update when a new path was created - Fixed
- Had a problem where drop list position would not update - Fixed
- Had a problem where history list overflowed off the screen on long tests - Fixed (added a scroll bar)
- Had a problem where first history list entry would cause the program to throw an Exception - Fixed
- Had a problem where deleting all paths would leave more green squares than expected - Fixed
- Had a problem where delete last path would turn all previous paths yellow when we only wanted one - Fixed
- Had a problem where collisions would not be detected for paths ending on the same square as another dropped - Fixed
- Had a problem where not all of the old paths would turn turquoise after a new path was created - Fixed
- Had a problem where drops would stay in the drop list after they should have merged Fixed
- Had a problem where a collision square would stay red after a delete last path function was completed - Fixed
- Had a problem where the wrong end byte was used on the add drop serial protocol - Fixed
- Had a problem where drops became part of the background mask when not moving for a long time - Fixed
- Had a problem where with slight change in lighting the whole detection would be thrown off - Fixed
- Had a problem where had to take a background image every time starting the program, instead of having the option to use a pre taken background image - Fixed
- Had a problem where all color were reported as grey - Fixed
- Had a problem where using a sleep in the main to allow time for the arduino to move drops was making it so the GUI was never updated with the new process image because the whole cpu was going to sleep - Fixed
- Had a problem where the delay was way too long before next square of movement - Fixed

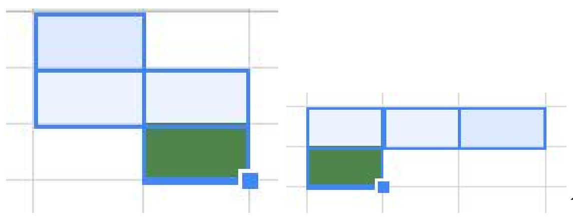
Path Formations:

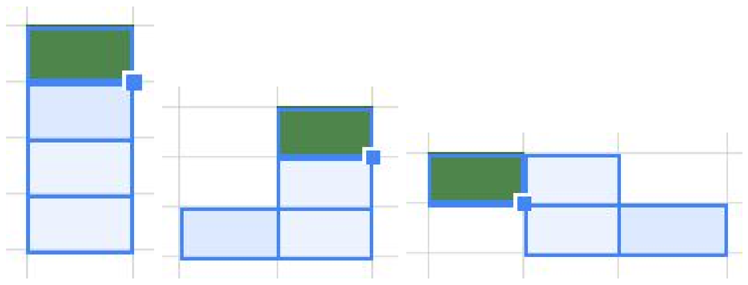
Low Complexity:

The basic path formations we used to test our software were single, double, and triple paths. A few examples of these can be seen below. These formations were tested in multiple scenarios, such as one drop running one path, one drop running multiple paths, multiple drops running one path, and multiple drops running multiple paths.


single

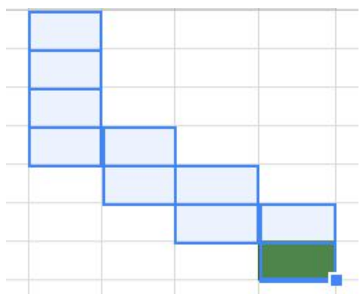
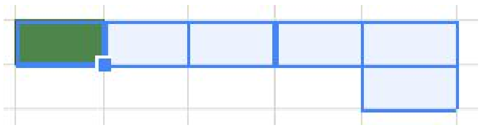
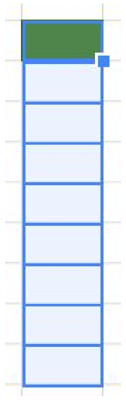
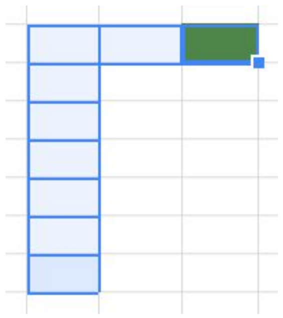
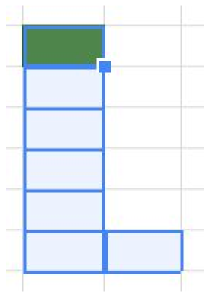
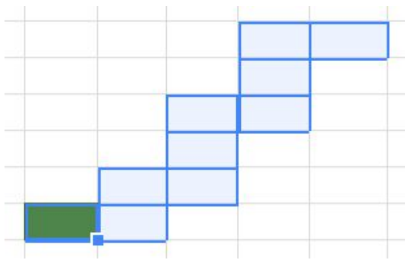

double


triple



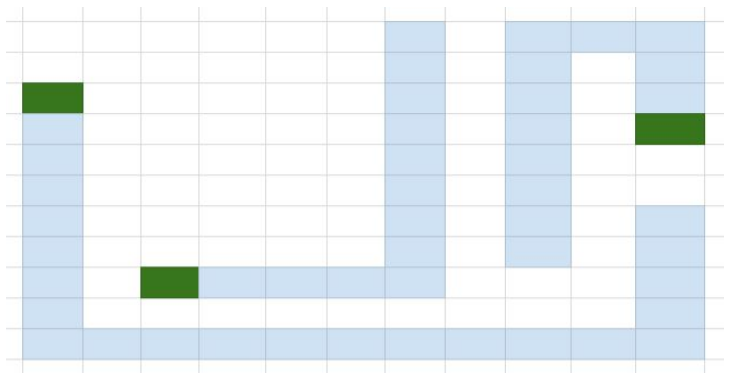
Medium Complexity:

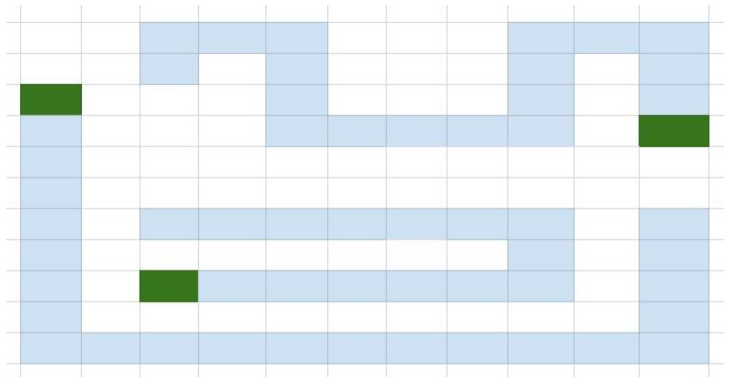
The formations below are the medium complexity path formations we used to test our system. With varying length and direction, these paths challenged the different use cases of path creation that exist in our code.

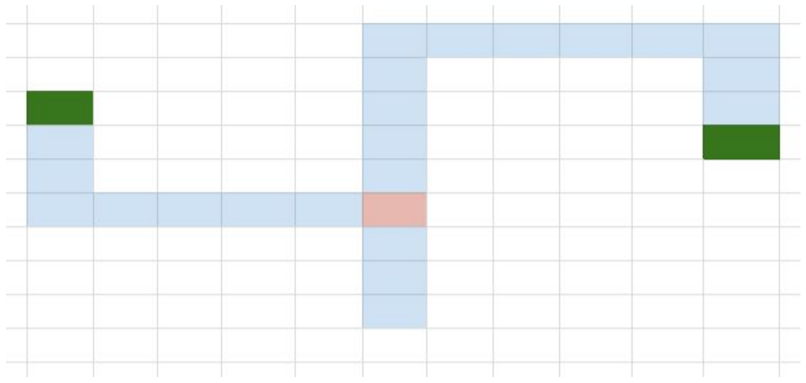







<u>High Complexity:</u>

The formations below show some of the more complex, multi drop tests that we ran to test our system. By running long paths with multiple changes in direction, we can really test the limits of our software. All the paths executed correctly except the collision one, which was explained above.







Original Capstone Firmware Tests:

In addition to testing the arduino firmware code by sending the paths tested on the gui and viewing the results, we also tested the limits of the microcontroller by sending it varying sizes of hard-coded byte arrays. The data from this tests can be seen in the figure below.

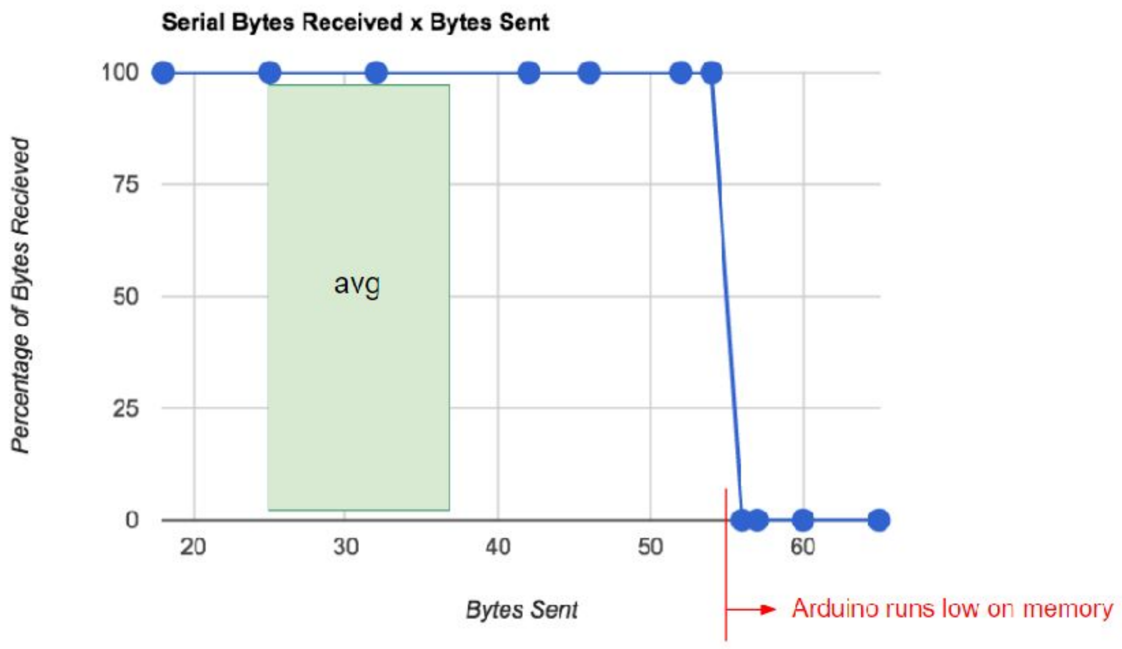


Figure 10. Limits of Arduino memory

As the graph above shows, the arduino crashed when we tried to send it more than 55 bytes. This is due to the Arduino Micro's limitation on global memory space. The average usage for typical path movements is around 30 bytes, so the limitation isn't that bad for the system as a whole. Every drop pattern can still be executed, but longer paths for a lot of drops must be split up into multiple "Send Path" button presses. This issue might be fixed by using an arduino with more memory space or using an external memory device, but it might not be cost effective for normal use.

Current Firmware Tests:
Since the arduino does not store any paths, it never runs out of memory and there is no bytes lost.

System Analysis

After testing PolyDrop with the Design Verification Plan and recording the results, the system meets most (if not all) requirements and passes as a reliable functional device. By design verification, the reliability was proven and limitations were discovered. For instance, the product meets all functional requirements to move different drops across the grid. However, a limitation is the number of drops possible on the board and how collisions are handled on the software. Below will go into detail on which requirements were successfully met, which were not and how it can be improved.

The functional tests include proper initialization, moving a drop and providing a history paths. The PolyDrop system now has a reliable method to initialize such that the Arduino is ready to receive serial data from the interface. Second, the user is able to select a number of drops and create unique paths for them to move. The PolyDrop system will program and manipulate the board to the exact paths the user creates. We tested this with many different paths, starting from simple and short to complex and long. Next, the product displays a history of run paths for the user, which provides a better user experience. Also, it provides the user a second method to log the paths and delete them if necessary. Next, PolyDrop allows the addition of up to five drops anywhere on the board. As a result, the user can begin a drop path anywhere. Finally, the image analysis was able to detect drops, their color and locations. It was also abstracted away so all the main has to do is call one simple function, that returns the necessary information. The processed image is nicely displayed in the GUI next to the UI grid.

Future iterations of Poly Drop can only be achieved if the system is fixed and begins to actually move drops. At that time, we have left the tools to create extra features with the image analysis data, such as failure detection.

Overall, PolyDrop meets all immediate requirements and is a reliable functioning system. We feel we have maximized the software's potential until a working hardware system is created

References

He, Jie-Long, An-Te Chen, Jyong-Huei Lee, and Shih-Kang Fan. "Digital Microfluidics for Manipulation and Analysis of a Single Cell." *IJMS International Journal of Molecular Sciences* 16.9 (2015): 22319-2332. Web.

Abdelgawad, Mohamed, Michael W. L. Watson, and Aaron R. Wheeler. "Lab on a Chip." *Hybrid Microfluidics: A Digital-to-channel Interface for In-line Sample Processing and Chemical Separations* (2009): n. pag. Web

Fobel, Ryan, Christian Fobel, and Aaron R. Wheeler. "DropBot: An Open-source Digital Microfluidic Control System with Precise Control of Electrostatic Driving Force and Instantaneous Drop Velocity Measurement." Applied Physics Letters 102.19 (2013): 193513. Web.

Pollack, Michael G., Richard B. Fair, and Alexander D. Shenderov. "Electrowetting-based Actuation of Liquid Droplets for Microfluidic Applications." *Applied Physics Letters* 77.11 (2000): 1725. Web.

Whitesides, George M. "Cool, or Simple and Cheap? Why Not Both?"*Lab Chip* 13.1 (2013): 11-13. Web.

"Trail: Creating a GUI With JFC/Swing." *Trail: Creating a GUI With JFC/Swing (The Java™ Tutorials)*. N.p., n.d. Web. 20 Mar. 2017.

"User Interface Design Basics." *Usability.gov*. Department of Health and Human Services, 21 May 2014. Web. 20 Mar. 2017.

Atlassian. "Git Tutorials and Training | Atlassian Git Tutorial." *Atlassian*. N.p., n.d. Web. 15 Dec. 2016.

"Arduino Serial Messaging Protocol." *Arduino Serial Messaging Protocol*. N.p., n.d. Web. 20 Nov. 2016.

Appendix

Appendix A

Attached is the user manual from Dr. Urs Gaudenz about the OpenDrop hardware.

OpenDrop

Instructions OpenDrop Prototype V2.1 – Digital Microfludics Plattform

OpenDrop is a new design for an open source digital microfludics platform for research purposes. The device uses recent electro-wetting technology to control small droplets of liquids. Potential applications are lab on a chip devices for automating processes of digital biology.

***** DISCLAIMER *****
This PROTOTYPE is for EXPERIMENTAL USE ONLY. The circuit generates voltages of up to 280 VDC. You the user, bear the responsibility for your own safety and safety of others in using this device. IN NO EVENT SHALL WE BE LIABLE FOR ANY DAMAGES OR LOSSES OF ANY KIND. Be sure that you understand the basic circuit concepts and have the required knowledge to operate it.

Removable Droplet Carrier

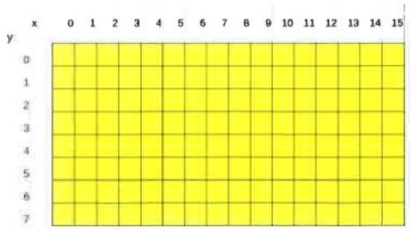
The OpenDrop devices is based on 16×8 electrode array integrated into the printed circuit board. The liquids stay on a thin, hydrophobic foil attached to the droplet carrier.

The foil (5-7 um thick) should lay flat on the electrode array with no air gap in between. A thin film of mineral oil can be applied to the electrodes before applying the foil. The electrodes are gold coted and not isolated. Make sure that no conductive liquids come in contact with the electrodes or flow into the electronic circuit as this may cause short circuits.

A good electro-wetting foil for the device is still under development. Current results are based on a ETFE foil (or Saran wrap) attached to the bottom of the carrier through double sided tape and coated with fluoropel (or Rain-X) water repellent to make the surface hydrophobic.
See here for details: http://www.gaudi.ch/OpenDrop/?p=116

Programming

The OpenDrop is based on the Arduino Micro micro-controller board. The electrodes are addressed in a x-y coordinate system (see picture below). A sample code with the instruction to activate one single electrode can be found on the OpenDrop website (http://www.gaudi.ch/OpenDrop/?p=139).



Instruction on how to install the Arduino programming tool and how to use it can be found here: http://arduino.cc/en/Guide/HomePage

Powering

To power the device use a standard Arduino power adapter rated at 12V DC, 250mA (min) and plug it into the power plug of the Arduino.

To switch the high voltage supply to the electrodes use the toggle switch on the top left. The up position is on. Make sure that the high voltage is switched off before manipulating the device.

Press the right micro-button to activate the device.

Adjust the voltage level using the trimmer potentiometer (blue). Turning the trim-pot clockwise increases the voltage.

For more information including schematics, circuit board design and general updates see: http://www.gaudi.ch/OpenDrop/

Appendix B

For this specification, we are referencing the installation process for MicroDrop onto OpenDrop. The users must be able to make this installation in order to use our plugin. Essentially, MicroDrop must be installed first which is why this is an important specification and step. We want our users to be able to install the software quickly and easily. Below are the steps to do so.

Downloading the Microdrop software

The latest Microdrop release is available here (see what's new in this version). Note that if you have previously installed a pre 1.0 version, you should uninstall it before installing the latest version. By default, every time you launch the application, it will check for updates online.

We also provide a self-extracting portable version which you can run from a folder on your computer or a USB thumb drive. Note that this option does not require Administrator privileges and allows you to easily switch between multiple versions of the software on the same computer.

Installing the control board plugin

Before you can actually use the *Microdrop* software, you will need to install a control board plugin. We currently support the dmf_control_board plugin (used for the DropBot) and have recently added support for the OpenDrop (note that you need to be connected to the internet for this next step).

1. Select "File/Manage Plugins" from the menu.
2. Click the "Download plugin" button.
3. Select "dmf_control_board" or "open_drop" from the list and click "OK".

A few seconds later you should see a message box telling you that the plugin was installed successfully. Plugins are stored in a folder called "Microdrop/plugins" which will be automatically created in your "My Documents" folder (or in the root of the portable distribution). The program will then shut itself down and you will need to restart it to finish the plugin installation.

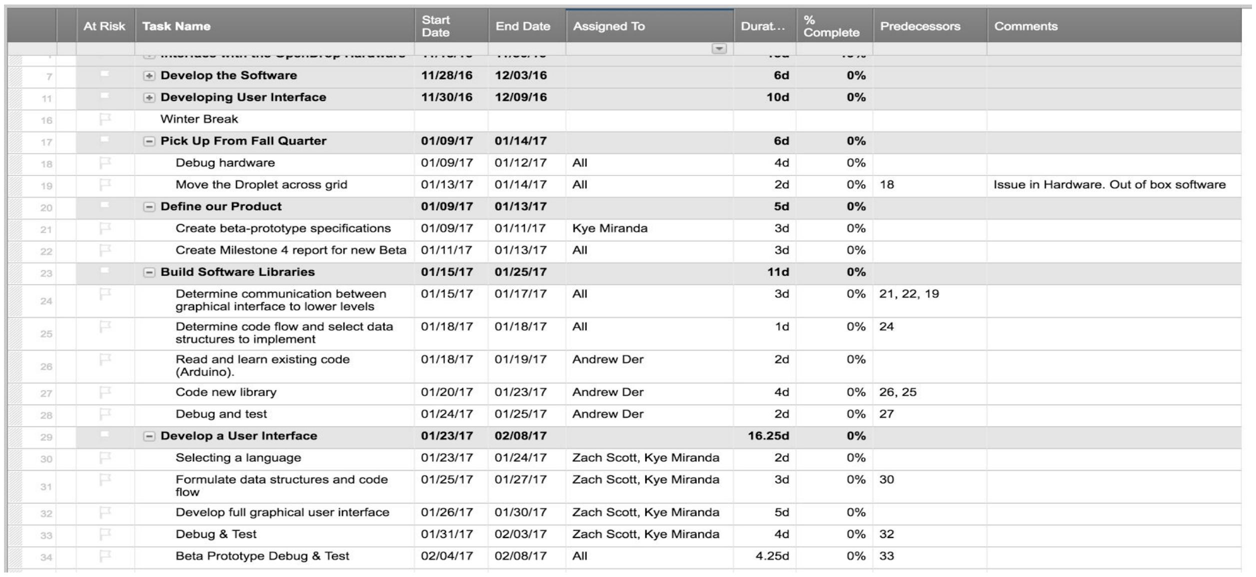
If you haven't flashed the firmware onto your system, you may need to do so before you can connect to it through Microdrop. Here are links to instructions for the DropBot and OpenDrop.

This information comes from ("UserGuide – Microdrop." *UserGuide – Microdrop*. N.p., n.d. Web. 17 Nov. 2016.)

Appendix C

Here are the Gantt sheets for the Fall Quarter and beginning of Winter Qtr. The respective Gantt charts are under Management Plan.

Figure xx: Gantt Sheet Fall Quarter

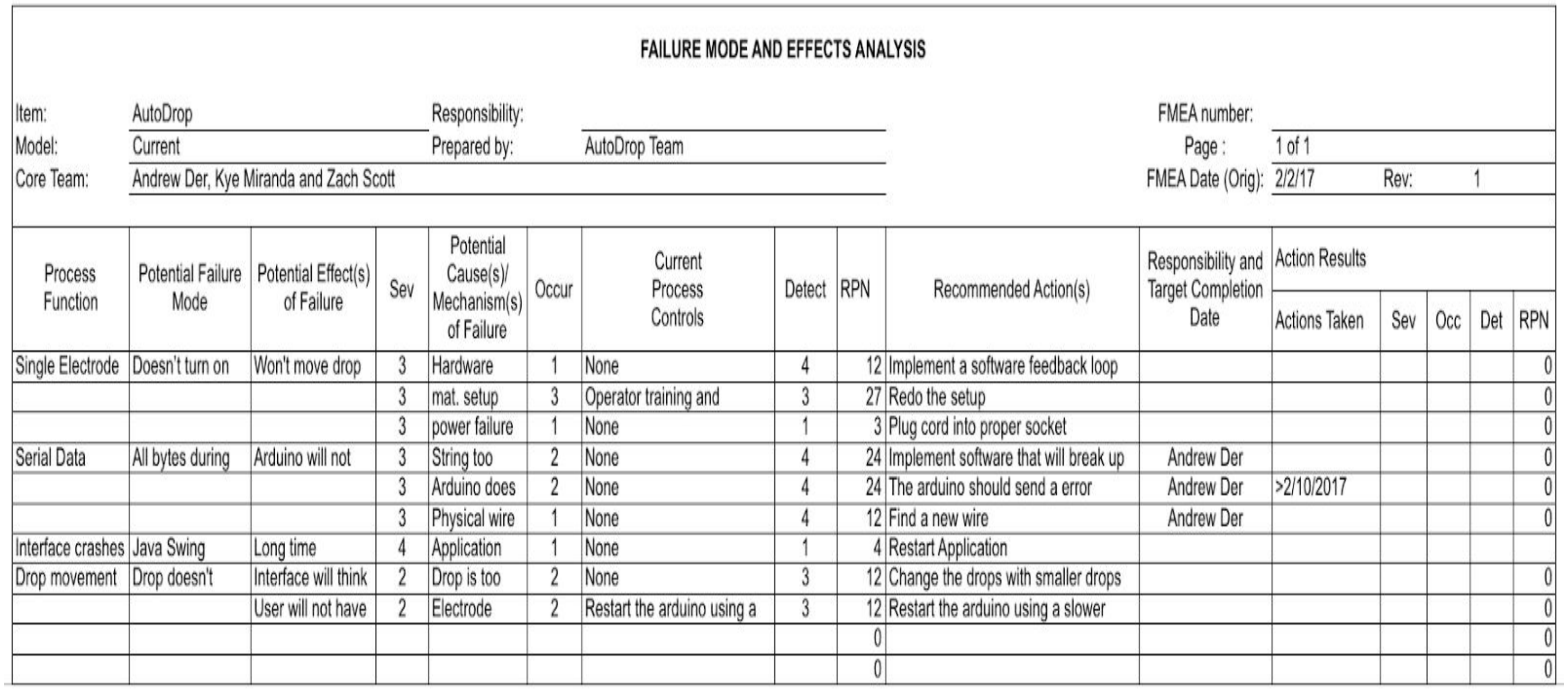


	At Risk	Task Name	Start Date	End Date	Assigned To	Durat...	% Complete	Predecessors	Comments
7		**Develop the Software**	**11/28/16**	**12/03/16**		**6d**	**0%**		
11		**Developing User Interface**	**11/30/16**	**12/09/16**		**10d**	**0%**		
16		Winter Break							
17		**Pick Up From Fall Quarter**	**01/09/17**	**01/14/17**		**6d**	**0%**		
18		Debug hardware	01/09/17	01/12/17	All	4d	0%		
19		Move the Droplet across grid	01/13/17	01/14/17	All	2d	0%	18	Issue in Hardware. Out of box software
20		**Define our Product**	**01/09/17**	**01/13/17**		**5d**	**0%**		
21		Create beta-prototype specifications	01/09/17	01/11/17	Kye Miranda	3d	0%		
22		Create Milestone 4 report for new Beta	01/11/17	01/13/17	All	3d	0%		
23		**Build Software Libraries**	**01/15/17**	**01/25/17**		**11d**	**0%**		
24		Determine communication between graphical interface to lower levels	01/15/17	01/17/17	All	3d	0%	21, 22, 19	
25		Determine code flow and select data structures to implement	01/18/17	01/18/17	All	1d	0%	24	
26		Read and learn existing code (Arduino).	01/18/17	01/19/17	Andrew Der	2d	0%		
27		Code new library	01/20/17	01/23/17	Andrew Der	4d	0%	26, 25	
28		Debug and test	01/24/17	01/25/17	Andrew Der	2d	0%	27	
29		**Develop a User Interface**	**01/23/17**	**02/08/17**		**16.25d**	**0%**		
30		Selecting a language	01/23/17	01/24/17	Zach Scott, Kye Miranda	2d	0%		
31		Formulate data structures and code flow	01/25/17	01/27/17	Zach Scott, Kye Miranda	3d	0%	30	
32		Develop full graphical user interface	01/26/17	01/30/17	Zach Scott, Kye Miranda	5d	0%		
33		Debug & Test	01/31/17	02/03/17	Zach Scott, Kye Miranda	4d	0%	32	
34		Beta Prototype Debug & Test	02/04/17	02/08/17	All	4.25d	0%	33	

Figure xx. Gantt Sheet Winter (Weeks 1 - 4)

Appendix D

Larger images of the Design Verification charts.

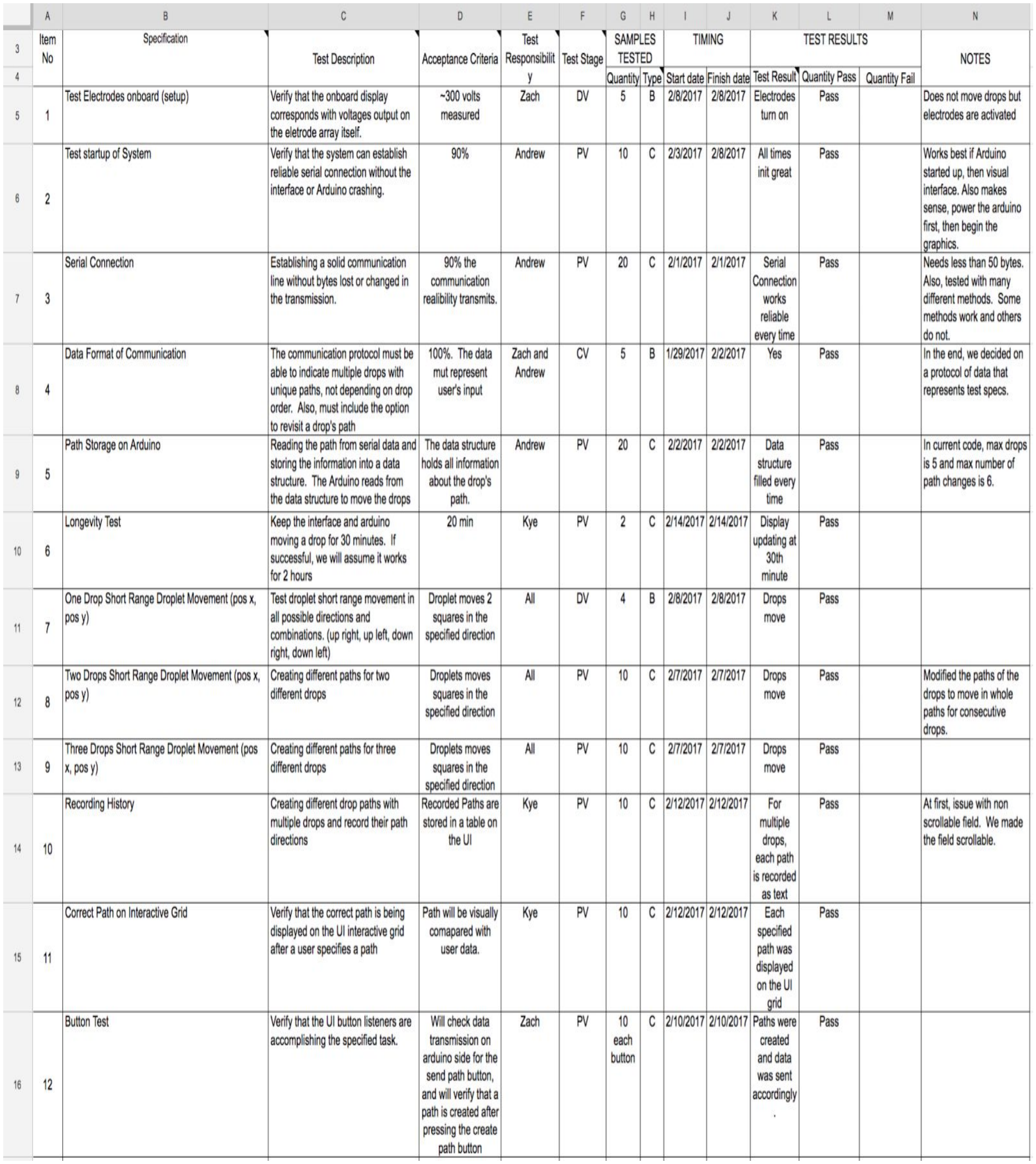


FAILURE MODE AND EFFECTS ANALYSIS

Item:	AutoDrop	Responsibility:		FMEA number:	
Model:	Current	Prepared by:	AutoDrop Team	Page :	1 of 1
Core Team:	Andrew Der, Kye Miranda and Zach Scott			FMEA Date (Orig):	2/2/17 Rev: 1

Process Function	Potential Failure Mode	Potential Effect(s) of Failure	Sev	Potential Cause(s)/ Mechanism(s) of Failure	Occur	Current Process Controls	Detect	RPN	Recommended Action(s)	Responsibility and Target Completion Date	Action Results				
											Actions Taken	Sev	Occ	Det	RPN
Single Electrode	Doesn't turn on	Won't move drop	3	Hardware	1	None	4	12	Implement a software feedback loop						0
			3	mat. setup	3	Operator training and	3	27	Redo the setup						0
			3	power failure	1	None	1	3	Plug cord into proper socket						0
Serial Data	All bytes during	Arduino will not	3	String too	2	None	4	24	Implement software that will break up	Andrew Der					0
			3	Arduino does	2	None	4	24	The arduino should send a error	Andrew Der	>2/10/2017				0
			3	Physical wire	1	None	4	12	Find a new wire	Andrew Der					0
Interface crashes	Java Swing	Long time	4	Application	1	None	1	4	Restart Application						
Drop movement	Drop doesn't	Interface will think	2	Drop is too	2	None	3	12	Change the drops with smaller drops						0
		User will not have	2	Electrode	2	Restart the arduino using a	3	12	Restart the arduino using a slower						0
								0							0
								0							0

FMEA Chart



Item No	Specification	Test Description	Acceptance Criteria	Test Responsibility	Test Stage	SAMPLES TESTED		TIMING		TEST RESULTS			NOTES
						Quantity	Type	Start date	Finish date	Test Result	Quantity Pass	Quantity Fail	
1	Test Electrodes onboard (setup)	Verify that the onboard display corresponds with voltages output on the eletrode array itself.	~300 volts measured	Zach	DV	5	B	2/8/2017	2/8/2017	Electrodes turn on	Pass		Does not move drops but electrodes are activated
2	Test startup of System	Verify that the system can establish reliable serial connection without the interface or Arduino crashing.	90%	Andrew	PV	10	C	2/3/2017	2/8/2017	All times init great	Pass		Works best if Arduino started up, then visual interface. Also makes sense, power the arduino first, then begin the graphics.
3	Serial Connection	Establishing a solid communication line without bytes lost or changed in the transmission.	90% the communication realibility transmits.	Andrew	PV	20	C	2/1/2017	2/1/2017	Serial Connection works reliable every time	Pass		Needs less than 50 bytes. Also, tested with many different methods. Some methods work and others do not.
4	Data Format of Communication	The communication protocol must be able to indicate multiple drops with unique paths, not depending on drop order. Also, must include the option to revisit a drop's path	100%. The data mut represent user's input	Zach and Andrew	CV	5	B	1/29/2017	2/2/2017	Yes	Pass		In the end, we decided on a protocol of data that represents test specs.
5	Path Storage on Arduino	Reading the path from serial data and storing the information into a data structure. The Arduino reads from the data structure to move the drops	The data structure holds all information about the drop's path.	Andrew	PV	20	C	2/2/2017	2/2/2017	Data structure filled every time	Pass		In current code, max drops is 5 and max number of path changes is 6.
6	Longevity Test	Keep the interface and arduino moving a drop for 30 minutes. If successful, we will assume it works for 2 hours	20 min	Kye	PV	2	C	2/14/2017	2/14/2017	Display updating at 30th minute	Pass		
7	One Drop Short Range Droplet Movement (pos x, pos y)	Test droplet short range movement in all possible directions and combinations. (up right, up left, down right, down left)	Droplet moves 2 squares in the specified direction	All	DV	4	B	2/8/2017	2/8/2017	Drops move	Pass		
8	Two Drops Short Range Droplet Movement (pos x, pos y)	Creating different paths for two different drops	Droplets moves squares in the specified direction	All	PV	10	C	2/7/2017	2/7/2017	Drops move	Pass		Modified the paths of the drops to move in whole paths for consecutive drops.
9	Three Drops Short Range Droplet Movement (pos x, pos y)	Creating different paths for three different drops	Droplets moves squares in the specified direction	All	PV	10	C	2/7/2017	2/7/2017	Drops move	Pass		
10	Recording History	Creating different drop paths with multiple drops and record their path directions	Recorded Paths are stored in a table on the UI	Kye	PV	10	C	2/12/2017	2/12/2017	For multiple drops, each path is recorded as text	Pass		At first, issue with non scrollable field. We made the field scrollable.
11	Correct Path on Interactive Grid	Verify that the correct path is being displayed on the UI interactive grid after a user specifies a path	Path will be visually comapared with user data.	Kye	PV	10	C	2/12/2017	2/12/2017	Each specified path was displayed on the UI grid	Pass		
12	Button Test	Verify that the UI button listeners are accomplishing the specified task.	Will check data transmission on arduino side for the send path button, and will verify that a path is created after pressing the create path button	Zach	PV	10 each button	C	2/10/2017	2/10/2017	Paths were created and data was sent accordingly.	Pass		

Design Verification Plan