5-1-2022

Dual Automated Colorimetric Detection of Bacteria Using A Lateral Flow Assay

Reginald Conley II

DUAL AUTOMATED COLORIMETRIC DETECTION OF BACTERIA USING A LATERAL FLOW ASSAY

by

REGINALD CONLEY II

Presented to the Faculty of the Honors College of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

HONORS BACHELOR OF SCIENCE IN BIOMEDICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2022

ACKNOWLEDGMENTS

I would like to thank my mentors Dr. Kytai T. Nguyen and Dr. Khosrow Behbehani for their continued guidance and support. Dr. Nguyen's knowledge on bacterial assays was instrumental in outlining this project's goals and creating the testing methodology. Dr. Behbehani's advice on the software testing was invaluable.

I would also like to thank my senior design team members: Samuel Fuentes and Babin Khadka. Without their contributions to our group project, I would not have been able to conduct this Honors Project. Their contributions our senior design project helped to create the foundation upon which this Honors Project lays.

Lastly, I would like to thank the Bioengineering Department of the University of Texas at Arlington and the Honors College for their generous support and funding. Without it, this project would not have been completed.

May 03, 2022

ABSTRACT

DUAL AUTOMATED COLORIMETRIC DETECTION OF BACTERIA USING A LATERAL FLOW ASSSAY

Reginald Conley, B.S. Biomedical Engineering

The University of Texas at Arlington, 2022

Faculty Mentor: Kytai T. Nguyen & Khosrow Behbehani

Bacterial infections are one of the biggest issues facing hospitals today; they prolong hospital stays, worsen patient outcomes, and decrease the quality of life for these patients. In the food and water industry, bacteria are causes for food recalls and non-potable water. The objects of this research were to develop a biosensor that could detect *E. coli* and *S. aureus* bacteria simultaneously using the lateral flow assay platform and to generate a software program that could accurately read results of the LFA. The biosensor was not built due to antibody shortages from the COVID-19 pandemic. The software program was developed using MATLAB. Twenty images of LFA test results were subjected to three conditions: the standard condition, the decreased image intensity condition, and the decreased image contrast condition. The program successfully identified the results of the

LFA test for all twenty images under all three conditions. The minimal pixel depth required for accurate results was also investigated, and as a result, it was determined to be 8-bits.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 The Bacteria Problem

1.1.1 Nosocomial Infections

Nosocomial infection due to bacteria is one of the largest issues facing hospitals today; they prolong hospital stays, worsen patient outcomes, and decrease patient satisfaction. These nosocomial infections affect more than three million people every year in the U.S., costing hospitals over $28 billion dollars. An estimated additional $12.4 billion dollars are lost because of decreased productivity and early death of patients (CDC, 2019).

Traditional bacterial culturing to detect bacterial infection using a sample from the patient can take days before the results are known (Tabak et al., 2018). These slow response times delay the physician's ability to create and implement the optimal treatment protocol, resulting in longer healing times for the patient and increased spreading of the infection to other patients. In developing countries, this problem is more exasperated (Ayukekbong, 2017).

1.1.2 Bacterial Infections in other Industries

Healthcare is not the only industry with a bacteria problem as bacteria can also be found in our food and water. The U.S. Food Industry estimates that there are at least four million cases of foodborne illness every year. In a study conducted by Qiu et al. (2021) over the ten-year span of 2006-2016, they found that bacteria accounted for 58% of all foodborne disease outbreaks. More than 83% of all food recalls are due to bacteria contamination and can cost companies more than hundreds of millions of dollars between recalls and FDA fines (Scallan et al., 2011). The yearly economic burden of bacteria related food outbreaks is estimated at $54 billion dollars (Scharff, 2012). Every year there are approximately 7.2 million cases of waterborne illnesses due to bacteria, parasites, and fungi. In all, 94% of deaths due to waterborne illnesses are caused by bacteria (Collier, 2020).

1.2 Our Solution: Group Project Overview

There is a need for a low-cost device capable of rapid and accurate detection of bacteria. To meet this demand, this senior design group created two biosensors using the lateral flow assay platform. Both biosensors would output a distinct colorimetric change visible to the user when either *E. coli* or *S. aureus*, two bacteria that are common causes of bacterial infections, were present. The biosensor utilized magnetic nanoparticles with the ability bind to both gram-positive and gram-negative bacteria as the signaling particle.

1.3 Working Principles of a Lateral Flow Assay

The lateral flow assay (LFA) is a type of biosensors that detects a specific analyte through immunohistochemistry (Figure 1.1). It utilizes capillary action to carry the detection particles and features antibody lines for bacterial binding. The LFA features several prominent parts including the sample pad, conjugate pad, antibody test line, antibody control line, and absorbent pad (Mahmoudi et al., 2019).

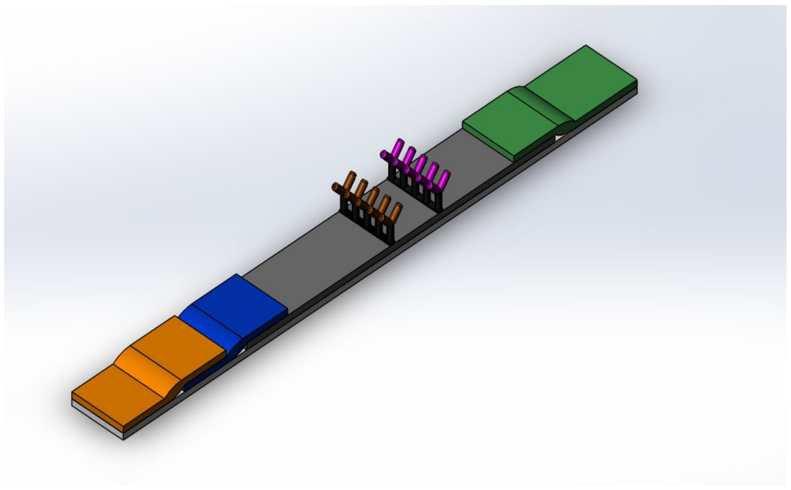


Figure 1.1: An example LFA built in SolidWorks.
Color is used to differentiate the parts.

1.3.1 Sample Pad

In Figure 1.1, the sample pad is in orange. In most LFA strips the sample pad is composed of a glass fiber pad. This fiber pad is pretreated with a buffer reagent such as phosphate buffered saline and other additives like Tween 20. The sample pad will soak in the solution and then be dried before use. The purpose of it is to absorb the liquid of the sample while regulating the sample's pH and act as a surfactant to decrease friction between the sample bacteria and the glass fibers that make up the sample pad (Koczula & Gallotta, 2016).

1.3.2 Conjugate Pad

As the sample leaves the sample pad, it will flow into the conjugate pad in blue. Like the sample pad, the conjugate pad consists of a buffer to further regulate the pH of the sample and a surfactant to increase the flow rate. However, the buffers will not be identical. Conjugate buffers tend to focus more on maintaining nanoparticle morphology and adding blocking reagents such as bovine serum albumin (BSA) to inhibit nonspecific binding on the flow pad (Alam et al., 2021).

The most critical job of the conjugate pad is to house the signaling particle. This signaling particle will be responsible for binding to the bacteria and producing a detectable signal when the bacteria in the patient sample are bound to the antibodies. This detectable signal can be visible, thermal, or electrochemical (Sadeghi et al., 2021). To get the nanoparticles into the conjugate pad, the particles are suspended in the conjugate buffer and then the glass fiber pad is soaked in the solution. Once the pad is dried, the signaling particle and any reagents in the solution will be dried onto the conjugate pad.

1.3.3 Flow Pad & Antibody Lines

The typical flow pad, in gray, used in an LFA is made from a nitrocellulose membrane, which is a compound composed of cellulose esters. The nitrocellulose membrane is then pretreated with blocking agents to reduce nonspecific binding of the signaling particles to the membrane itself (Tang et al., 2021).

Placed on the nitrocellulose membrane by either a piezoelectric technique or a nonspecific binding technique is the antibody line. On a typical LFA test there are two antibody lines. The magenta-colored antibody line is known as the control line. This line tells the user that the test functioned properly. In a test that functions properly, this line will form regardless of whether the analyte being tested for is present. The brown colored antibody line is known as the test line (Figure: 1.1). This line will only form if the analyte being tested for is present (Boehringer & O'Farrell, 2021).

1.3.4 Absorbent Pad

The absorbent pad is shown in green. This pad is made of cellulose. The absorbent pad function is to absorb excess liquid that travels beyond the antibody lines. Because the LFA works on capillary action, back flow of any liquid would impede the forward progress of the signaling particles to detect bacteria correctly. Back flow can cause the LFA to give incomplete or inaccurate results (Castillo-León et al., 2021).

1.4 Types of Lateral Flow Assays

There are two main categories of LFAs. The LFAs that produce results visible to the user are termed colorimetric LFAs. The other type of LFAs do not produce colorimetric results and instead rely on specialized equipment to determine if a test line has formed. The focus of this project was on colorimetric detection of bacteria using LFAs.

Within the realm of colorimetric detection LFAs, the standard detection method is gold nanoparticles conjugated to antibodies. This allows for the antibody-nanoparticle conjugate to bind to its target analyte and produce a reddish color at the test line that is easy for the user to observe (Zhao et al., 2018).

The issue with this design is gold nanoparticles are expensive to buy due the challenges of producing particles with uniform diameter. In addition to this, the use of antibodies ultimately limits the particles' ability to bind to various substances due to the high specificity of antibodies for their unique analyte (Andryukov et al., 2020). The use of nonspecific antibodies has increased the range of substances that the gold nanoparticle can bind to at the sacrifice of specificity (Posthuma-Trumpie et al., 2009). With this strategy, there is still no way to distinguish between two different bacteria in the same biosensor. Another common problem with nonspecific antibodies they lack the ability to bind to both gram-positive and gram-negative bacteria. The nonspecific antibodies will be able to bind to one type of bacterial wall at the sacrifice of binding to the other type.

The other option is to place multiple gold nanoparticles with different antibodies attached to them on the lateral flow assay. This lowers the concentration of antibodies for a particular substance. This causes less of the gold nanoparticles to show up on their corresponding test line and produce that color change. This results in a weak/nonvisible output to the user. Dual detection with exclusive binary answers remains a challenge for gold nanoparticles (Sin et al.,2014; Kozel et al.,2017).

The use of iron oxide nanoparticles has become of more interest beside gold nanoparticles. Their magnetic properties in addition to their brown color allow for use in colorimetric lateral flow assays (Liu et al., 2011). Multiple iron oxide-antibody conjugates have been used in the detection of bacteria (Moyano et al., 2020; Poonlapdecha et al., 2018). However, the same issue with the gold nanoparticles remains for the iron oxide nanoparticles.

To fix this issue, new antibody-free nanoparticles, called label-free particles, are being developed. One method using positively charge gold nanoparticles has shown promise and is able detect multiple bacteria (Bu et al., 2019). However, this design was not able to detect multiple bacteria within the same lateral flow strip; the design still requires separate assays to detect different bacteria. A similar design label free with gold nanoparticles uses a change in fluorescence to determine if E. coli is present. While this method shows some promise results; however, the output is almost nonvisible and requires specialized equipment for reading/detection (Song et al., 2016).

A label-free design using iron oxide nanoparticles with a copper sulfide coating has shown promise. In this study they were able to build a lateral flow assay to detect E. coli, gram negative, by this method (Bu *et al.*, 2020). While it theoretically can detect other bacteria as well, it needs to be tested to prove this. This was the focus of my senior design group's project.

1.5 Significance of this Honors Project

While colorimetric lateral flow assays work well, there is a need to create an assay capable of testing for multiple bacteria simultaneously. The creation of this dual detection assay using label free nanoparticles will show that it is possible and eliminate dual detection as a limitation of colorimetric assays. Additionally, completing this project will improve the usefulness of our group's biosensor for real world application where testing for multiple bacteria can now be done with a single sample measurement from the patient.

More importantly, creating this first prototype will lay down a template for a variety of future assay studies with real world applications. One example is bacterial detection in water. Using dual detection will allow for true broad spectrum bacterial detection using multiple test line antibodies. It will also set the stage for bacterial discrimination using the lateral flow assay platform. An example of this future application would be using this design to distinguish if the patient's *Staph* infection is MRSA or *S. aureus* using two different sets of antibodies.

Equally important to dual detection, is creating a way to automate the results of the test. This allows for easy determination of the results and can act as a secondary check point to confirm the results. This would eliminate a possible user error where they interpret the results of the LFA incorrectly. Interpretation would be especially important if the LFA strip is testing for multiple bacteria, knowing which test line corresponds to which bacteria is crucial.

CHAPTER 2

METHODOLOGY

2.1 Dual Detection LFA

The LFA strip was built with the same parts used my senior design group. The sample and conjugate pads were cut from a glass fiber pad. The flow pad was the same pretreated M180 nitrocellulose membrane. The absorbent pad was cut from GE Whatman paper. The overall dimensions of the LFA strip were 6 mm wide by 65 mm long.

The copper sulfide capped iron oxide nanoparticles (MNPs) were fabricated using the same technique as my senior design group. The iron oxide nanoparticles were capped in 2 mg increments. 2 mL of 10 mg/ml of Carboxylated iron oxide nanoparticles (Fe_3O_4-COOH) were treated with 25 µL of 0.5 mg/ml solution of Cetyltrimethylammonium bromide (CTAB). The reagents were mixed for 30 minutes using a shaker. It was important not to use a magnetic stir bar as the iron oxide particles are magnetic and would clump together on the magnet. After shaking, the iron oxide particles were centrifuged and washed with deionized water three times. The particles were then resuspended in 25 µL of 0.04 M sodium sulfide (NaS) and shaken for three hours. 25 µL of 0.04 M Copper (II) sulfate ($CuSO_4$) was then added to the solution dropwise and shaken for 3 hours. The copper sulfide capped iron oxide nanoparticles were centrifuged and washed twice with water and then once with ethanol. The particles were then dried in the oven at 60 degrees Celsius for 12 hours.

Bacterial culturing was done with E. coli and S. aureus. Initial samples were taken and allowed to grow on separate agar plates. These plates were then stored in the 4-degree Celsius refrigerator. This kept the bacteria alive but stopped them from continuing to grow. All bacteria used in this study were grown from the initial colonies that formed on those agar plates. For tests with bacteria, a single uniform colony was removed from the agar plate and placed in a test tube with 2 mL of LB broth. It was then allowed to incubate overnight. All CFU/ml calculations were done using a spectrometer at OD 600.

2.1.1 Antibody Test Line Placement

For the dual detection, multiple antibody test lines had to be placed on the flow pad. The volume and concentration of the antibody tests lines were 6 μL of the 4mg/ml solution. Three possible designs were created. The first was to place the *E. coli* line in front of the *S. aureus* line and have both span the entire width of the strip. The second design was to place the *S. aureus* line in front of the *E. coli* line and have both span the entire width of the strip. The final design was to use two test lines that only span half of the strip (Figure 2.1). The reason for this was putting one test line in front of the other created a 'wall' of antibody-bacteria-nanoparticle conjugates at the first test line. Any bacteria-nanoparticle conjugates trying the reach the other test line would have to travel over the 'wall' formed by test line in front of it. Without knowing the effects of placing one test line in front of the other all three designs had to be tested.

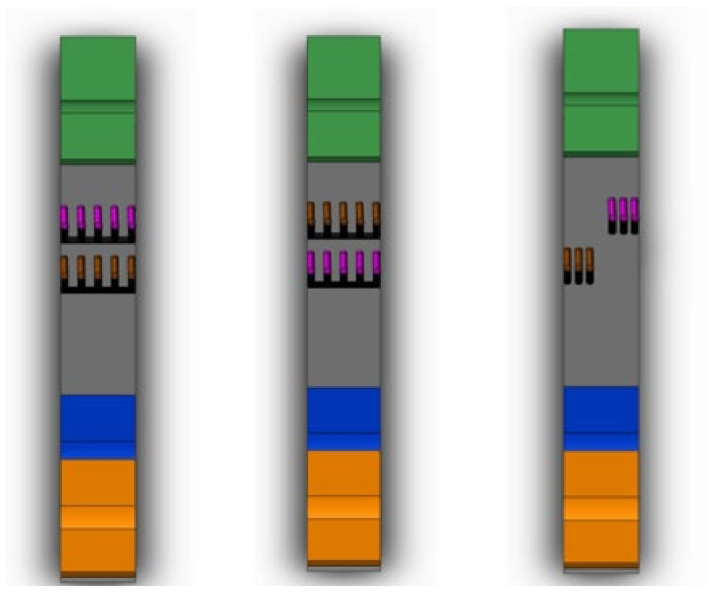


Figure 2.1: Design #1 (Left) with the E. coli line in front of S. aureus. Design #2(Middle) with S. aureus line in front of E. coli. Design #3 (Right) with two half

2.1.2 Sample Specifications

All tests were conducted using an identical sample set up. In this set up 50 μL of 1 mg/mL of MNPs were combined with 50 μL of bacteria. The MNP concentration was held constant for the initial specificity (2.1.3) and sensitivity (2.1.4) testing. The MNP concentration was varied during the variable testing (2.1.5). The bacterial concentration was held constant during the specificity testing (2.1.3) and varied in the other two tests (2.1.4 & 2.1.5). All conducted tests used the same volume. The initial volume of the combined MNP and bacteria sample was 100 μL. From that 100 μL, 75 μL were placed on the LFA strip.

2.1.3 Specificity

Specificity testing was done for all three designs. The nanoparticle concentration was held constant at 1 mg/ml. The bacteria concentration was held constant at 1 x 10^8

CFU/ml. Each design ran three tests in triplicate. Any design that failed either of these three tests would not move onto the next set of testing.

The first test was to determine that the strip could accurately detect E. coli without a false line arising at the S. aureus test line. The second test was done with S. aureus to determine if the strip could detect S. aureus without a false line forming at the E. coli test line. The final test was a combined sample of both bacteria, to determine if the LFA strip could form two test lines simultaneous when both bacteria were present.

2.1.4 Sensitivity

Sensitivity testing was done to determine the lower limit of detection for the three LFA designs. All tests were done in triplicate. 25 μL of both bacteria along with 50 μL of MNPs were added together for a total volume of 100 μL. 75 μL were used to run the test. For sensitivity testing, the initial starting concentration was 1 x 10^7 CFU/ml of each bacterium. After every successful test, the concentration of each bacterium was lowered by a factor of 10 (i.e., 10^7 to 10^6 to 10^5 to …) until no test lines were visible. At that point, the concentration was increased by a factor of 5 to test visibility. It was then changed by a factor of 1 until the test failed again. Once it failed again, the final lower limit of detection was known.

For example, if the test worked at 1 x 10^7 CFU/ml but failed at 1 x 10^6 CFU/ml, the next test would be right in the middle at 5 x 10^6 CFU/ml. Depending on whether that test fails or succeeds, the CFU/ml would change by a factor of 1 to either 6 x 10^6 CFU/ml or 4 x 10^6 CFU/ml. The bacterial concentration would continue to change by a factor of one until the results were no longer visible on the LFA test. Once that happened, the lower limit of detection would be found.

2.1.5 Variable Testing

The design that had the lowest limit of detection continued to variable testing. There were two variables to be tested. The first was the effect of MNP concentrations on the lower limit of detection. Three different MNP concentrations were tested: 0.5 mg/ml, 1.5 mg/ml, and 2.0 mg/ml. For each concentration the antibody test line volume and concentration were held constant at 6 µL of the 4mg/ml solution. The initial test was conducted at the lower limit of detection found in sensitivity testing (2.1.4). If the new test produced visible test lines, the bacteria concentration was lowered further to find the new lower limit of detection.

The second variable tested was the effect of antibody concentrations on the lower limit of detection. For this test, the MNP concentration was held constant at 1 mg/ml. The antibody volumes changed to 4 µL, 8 µL, and 10 µL per test. Similarly, the initial test was conducted at the lower limit of detection found in sensitivity testing (2.1.4). If the new test produced visible test lines, the bacteria concentration was lowered further to find the new lower limit of detection.

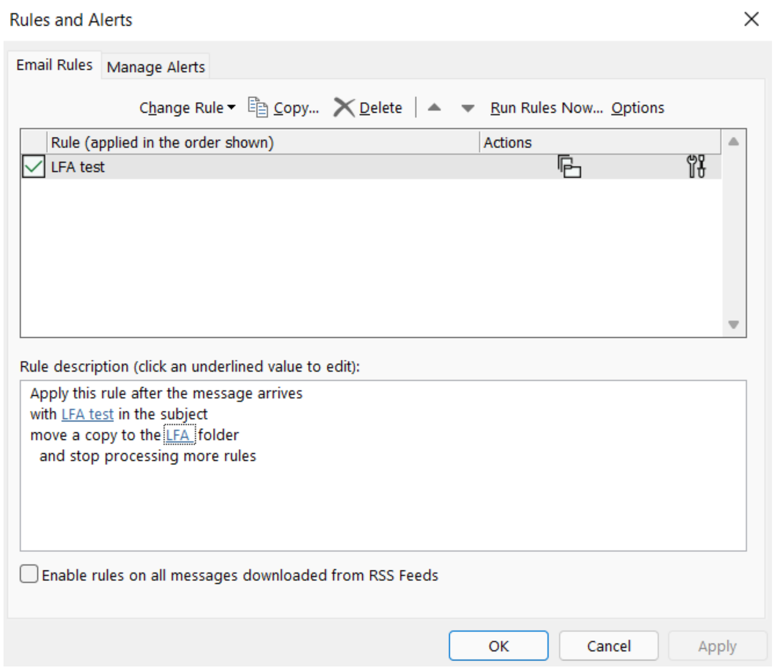
2.2 Automated Line Detection

The automated line detection program was written using MATLAB 2019a. The program was run on a Lenovo PC featuring an Intel Core i5 (64 bit) processor with 8 GB of RAM. The operating system was Windows 11. Microsoft Outlook 365 version 2203, build 16.0.15028.20218 was used for email. The methods used to construct this program were broken into three sections: The Image Acquisition (2.2.1), Image Processing (2.2.2), and Program Output (2.2.3) sections.

2.2.1 Image Acquisition

The first step to get the image to the program required the user to take a picture of their LFA test. There were three requirements of the user. The height of the LFA test had to also be the height of the image; the LFA test had to span from the top to the bottom of the picture. The width of the image should be the width of the LFA.

Once the image was taken by the user, it was emailed and uploaded in the system for analysis and further labelled as the ‘LFA Test’. Using Outlook’s Rule function, a copy of any email with this subject would get sent to a special subfolder. Capitalization of the email subject did not affect transfer to this subfolder (Figure 2.2).



✓ LFA test

Rule description (click an underlined value to edit):

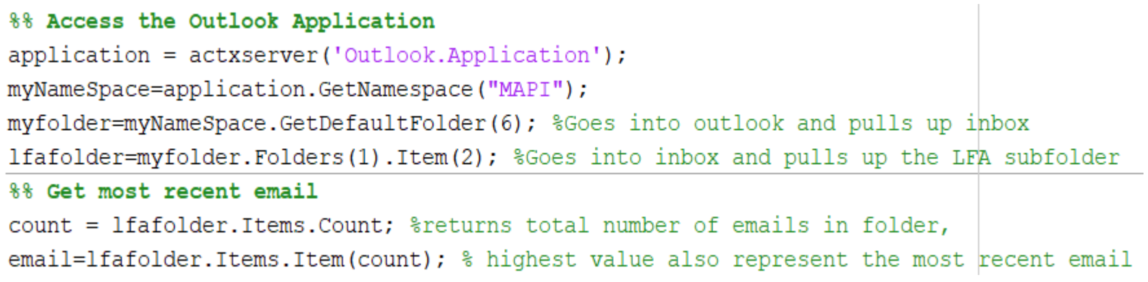
Apply this rule after the message arrives
with LFA test in the subject
move a copy to the LFA folder
and stop processing more rules

☐ Enable rules on all messages downloaded from RSS Feeds

Figure 2.2: The Outlook Rule used to send a copy of emails with the subject ‘LFA test’ to the subfolder.

Once the user sent the email it would arrive in the specified inbox and then the MATLAB program could then be run. The program utilized MATLAB’s COM interface to access and control the Microsoft Outlook application through its Virtual Basic

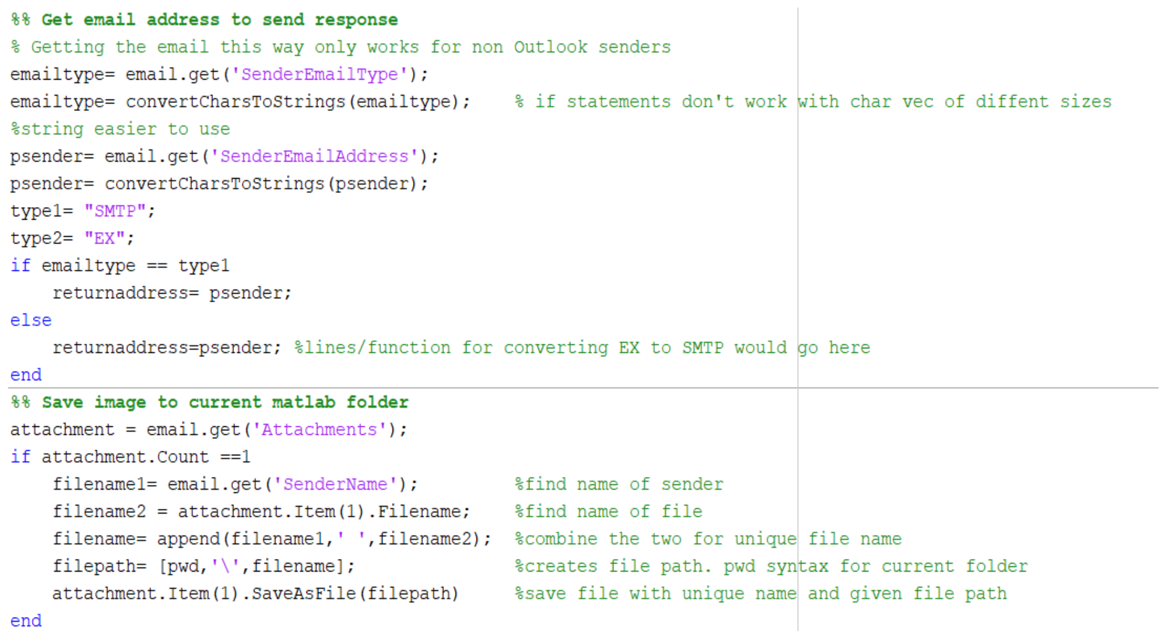
Applications (VBA). Microsoft VBA allowed for macroscopic control using outside applications such as MATLAB. The first step was to use the actxserver function to call the Outlook application. The actxserver function created a local OLE Automation server. This allowed for remote control Outlook without actively using the application. After accessing the application, the program was coded access the subfolder with the LFA tests (Figure 2.3).



```
%% Access the Outlook Application
application = actxserver('Outlook.Application');
myNameSpace=application.GetNamespace("MAPI");
myfolder=myNameSpace.GetDefaultFolder(6); %Goes into outlook and pulls up inbox
lfafolder=myfolder.Folders(1).Item(2); %Goes into inbox and pulls up the LFA subfolder
%% Get most recent email
count = lfafolder.Items.Count; %returns total number of emails in folder,
email=lfafolder.Items.Item(count); % highest value also represent the most recent email
```

Figure 2.3: The lines of code required to find the correct email

Once the email had been found the program withdrew the email address as well as the attachment for use later in response email and image analysis. The program saved the email under the variable ‘psender’. The attachment was saved under ‘filename’. An If statement was used to find the attachment as well as save it under a name unique to every sender (Figure 2.4). There was a limitation of the acquisition of the email address which is explained in the discussion section of this report.

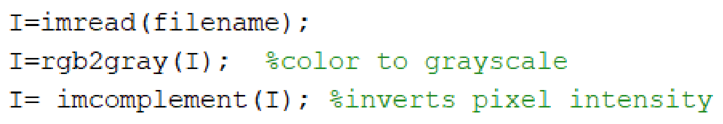


```
%% Get email address to send response
% Getting the email this way only works for non Outlook senders
emailtype= email.get('SenderEmailType');
emailtype= convertCharsToStrings(emailtype);    % if statements don't work with char vec of diffent sizes
%string easier to use
psender= email.get('SenderEmailAddress');
psender= convertCharsToStrings(psender);
type1= "SMTP";
type2= "EX";
if emailtype == type1
    returnaddress= psender;
else
    returnaddress=psender; %lines/function for converting EX to SMTP would go here
end
%% Save image to current matlab folder
attachment = email.get('Attachments');
if attachment.Count ==1
    filename1= email.get('SenderName');          %find name of sender
    filename2 = attachment.Item(1).Filename;    %find name of file
    filename= append(filename1,' ',filename2); %combine the two for unique file name
    filepath= [pwd,'\',filename];              %creates file path. pwd syntax for current folder
    attachment.Item(1).SaveAsFile(filepath)     %save file with unique name and given file path
end
```

Figure 2.4: The lines of code required to extract the user email and image attachment.

2.2.2 Image Processing

Once the image attachment had been saved, it was processed with the function ‘imread’ which could read multiple imaging formats including .JPG, .PNG, .JPEG, and .TIFF. Once the image was loaded it was converted from color to grayscale using the function ‘rgb2gray’. Converting the image to grayscale was crucial for edge detection and counting as those functions operate on differences in grayscale intensity. Once in grayscale, the image was inverted using the function ‘imcomplement’. Imcomplement switched the pixel intensity of every pixel. Bright pixels became dark and dark pixels became bright (Figure 2.5). This was important for the ‘bwboundaries’ function later in the program.

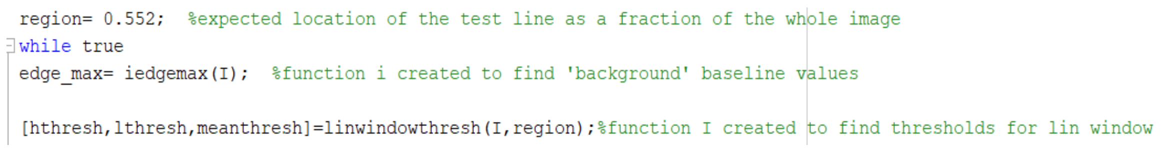


```
I=imread(filename);
I=rgb2gray(I);  %color to grayscale
I= imcomplement(I); %inverts pixel intensity
```

Figure 2.5: The lines of code required to load the image and convert to inverted grayscale.

The remaining portions the image processing was done within a while true loop. What this loop did was continue to run until a certain condition was met that broke the loop. Those conditions in this code were the detection of no control line, detection of only the control line, or the detection of both the control line and the test line.

The next step to detecting a line was to find the background intensity and begin construction of the Linear Window. This was done using two subroutines that I created and stored in the functions called 'iedgemax' and 'linwindowthresh'. For a detailed look at the subroutines in this program, please refer to Appendix B. Iedgemax calculated the noise in the user taken picture. The subroutine went around the outer edges of the image and calculated the average pixel intensity as a baseline. Linwindowthresh used the value stored in the variable 'region' to calculate the average pixel intensity of the control line (Figure 2.6).



```
region= 0.552;  %expected location of the test line as a fraction of the whole image
while true
edge_max= iedgemax(I);  %function i created to find 'background' baseline values

[hthresh,lthresh,meanthresh]=linwindowthresh(I,region);%function I created to find thresholds for lin window
```

Figure 2.6: The lines of code required for the noise and linear window thresholds.

The average intensity was then used as the upper threshold of the linear window and the average intensity minus one unit was used as the lower threshold. By keeping the thresholds of the linear window one integer apart a step filter was created (Figure 2.7). What the step filter did was take any pixel with a value less than or equal to the lower threshold and reassign that pixel's value to zero. Any pixel with a value greater than or equal to the upper threshold had their value reassigned to 256. This removed the majority of unwanted objects and edges in the image.

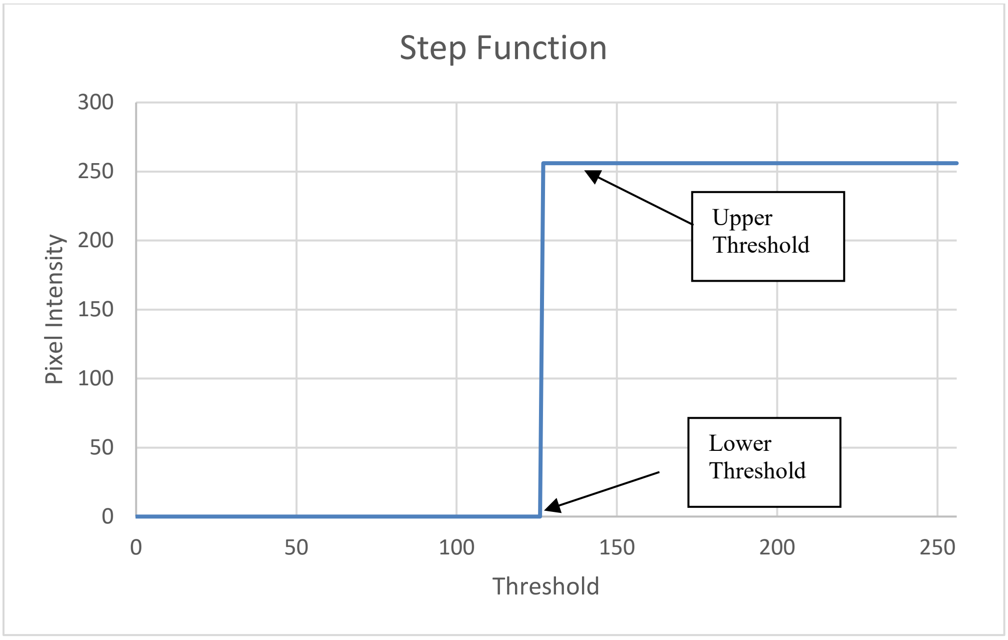


Figure 2.7: Pictural representation of the step function to be created

The variable ‘region’ told linwindowthresh where the control line could be found. Region had to be hard coded into the program as it states the fraction of the whole image where the control line can be found. This single line of code would need to be readjusted for testing with my LFA strips at a future date.

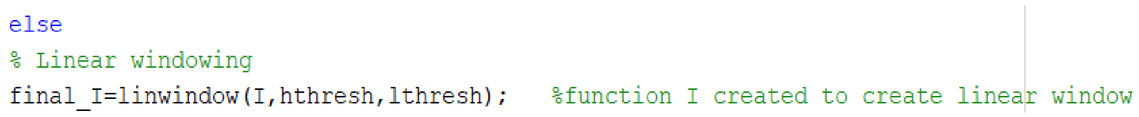
For the LFA tests that failed (did not produce a control line), the linwindowthresh function would still calculate the average intensity where the control line should have formed. To prevent the incorrect answer, an if statement was used (Figure 2.8). The if statement tells the program to break the while loop if the background noise from iedgemax was greater than the upper threshold from linwindowthresh. What this if statement did was determine if the edge of the image had a greater than or equal mean pixel intensity in comparison to the location of the control line. If it did, then no control formed during the LFA test. A new variable is formed named ‘txt’. This variable stored a string to be used later. Once the while loop broke, the program moved on to the output section discussed later.

```
if meanthresh< edge_max
    txt='The test failed. Please contact us.';

    break
```

Figure 2.8: The lines of code required to determine if the LFA test failed

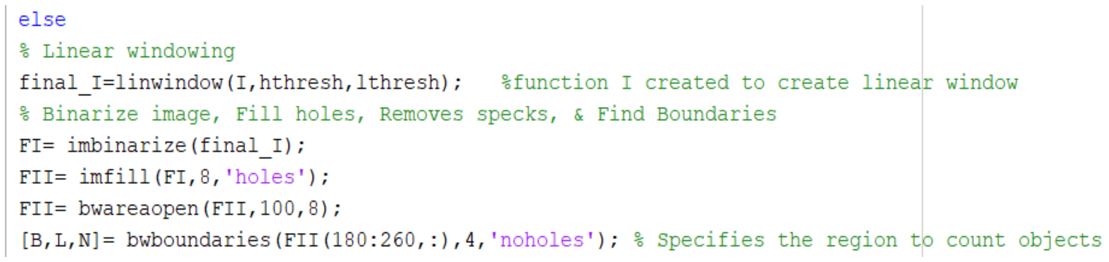
If the upper threshold was greater than idegemax, the else portion of the if statement became active (Figure 2.9). In this portion the subroutine function ‘linwindow’ created the actual step function seen in Figure 2.7 using the values from linwindowthresh. It then filtered the image through the linear window/step function of Figure 2.7.



```
else
% Linear windowing
final_I=linwindow(I,hthresh,lthresh);   %function I created to create linear window
```

Figure 2.9**:** The lines of code required to create the linear window seen in Figure 2.7 and filter the image with it

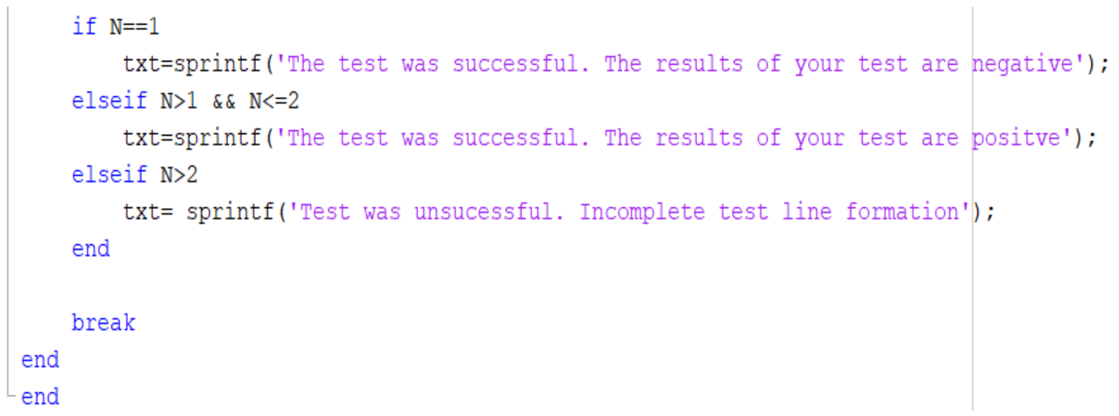
After the linear windowing, the image was binarized using the function ‘imbinarize’ and any small objects were removed using the function ‘imfill’. A second object removal was done using the function ‘bwareopen’. Once the small objects were removed, the function ‘bwboundaries’ was used to count the remaining objects within the specified region of the image and store them in the variable ‘N’ (Figure 2.10). These objects would be the test and control lines.



```
else
% Linear windowing
final_I=linwindow(I,hthresh,lthresh);   %function I created to create linear window
% Binarize image, Fill holes, Removes specks, & Find Boundaries
FI= imbinarize(final_I);
FII= imfill(FI,8,'holes');
FII= bwareaopen(FII,100,8);
[B,L,N]= bwboundaries(FII(180:260,:),4,'noholes'); % Specifies the region to count objects
```

Figure 2.10: The lines of code required for object detection and counting

A nested if statement was added to determine the proper response based on the number of objects found. The response was then stored in the variable ‘txt’ to be used in the output section of the code. Once the if statement terminated, the while loop would break (Figure 2.11).



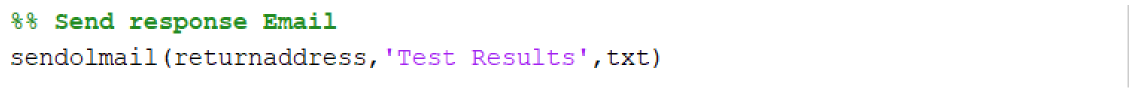
```
    if N==1
        txt=sprintf('The test was successful. The results of your test are negative');
    elseif N>1 && N<=2
        txt=sprintf('The test was successful. The results of your test are positve');
    elseif N>2
        txt= sprintf('Test was unsucessful. Incomplete test line formation');
    end

    break
end
end
```

Figure 2.11: The lines of code required to determine proper output responses

2.2.3 Program Output.

After the image processing, the program sent the user a response email with the results of their test using the subroutine function ‘sendolmail’ created by the MathWorks Help Team. The variable ‘txt’ could have three different possible values depending on the results of the image processing portion of the program. That was what allowed for the different responses depending on the image sent by the user (Figure 2.12).



```
%% Send response Email
sendolmail(returnaddress,'Test Results',txt)
```

Figure 2.12**:** The line of code to send the user an email with the results of their test

2.2.4 Program Testing

Due to the antibody supply chain issues and shortages as a result of the COVID-19 pandemic, the program was unable to be tested on LFA strips. To compensate for this, all images of completed LFA tests were taken from the study conducted by Çam et al. (2017). Keeping the images from one source, in essence keeping the images uniform, was done intentionally. This program was originally developed to be used in conjunction with my specific LFA strips for bacterial detection and recognition. Keeping the images uniform, even if the program is optimized for another LFA design, allowed the overall architecture of the program to remain the same. This allows easy readjustment to my LFA strips at a future date.

The study by Çam et al. (2017) was chosen due to the large number of images available as well as the quality of the images. All the images are of poor quality (8-bit depth). Additionally, the test and control lines of their study had inconsistent edges that lacked the typical sharpness seen on a commercial product. Because of these issues, they were a perfect testing set for creating a robust program.

The MATLAB program was tested under four conditions. The first condition was labeled the standard condition. This was the image as they were. The second condition was to simulate poor lighting. This was done by decreasing the image intensity. Using the Photos application of the Lenovo computer, the image intensity was changed to -90. The third condition was to simulate blurry photos. This was done by decreasing the image contrast to -90. The fourth condition was to determine the minimal image depth required to produce accurate results. While images are not usually below 8 bits, the lowest pixel depth was calculated for the intention of finding the programs limits. This was done by dividing the 8-bit images by powers of two decrease the image bit depth to 6-bit and 4-bit. Testing of the minimal image depth was done on images under the standard conditions.

.

CHAPTER 3

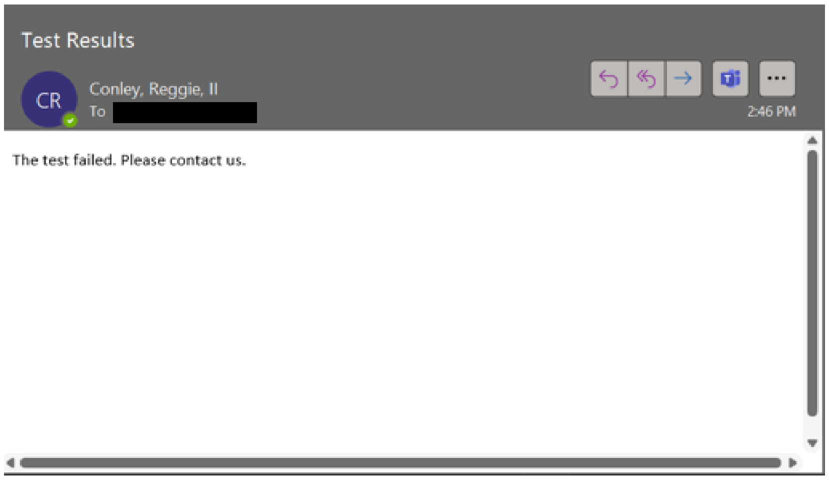
RESULTS

3.1 Dual Detection with LFA

Due to supply chain issues from the COVID-19 pandemic, the antibodies required to conduct the dual detection testing did not arrive before the project deadline.

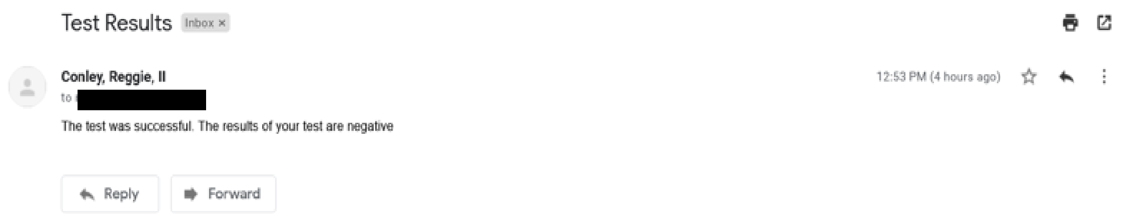
3.2 Automated Line Detection

The program can accurately respond to both SMTP and EX mail protocols (Figure 3.1 & 3.2).



The test failed. Please contact us.

Figure 3.1: Response email sent to Outlook mail user for a test with no visible lines


Test Results

Conley, Reggie, II

to [redacted]

12:53 PM (4 hours ago)

The test was successful. The results of your test are negative

Figure 3.2: Response sent to an email user for a test with only the control line visible

3.2.1 Standard Conditions

For the standard conditions' tests, twenty images were tested without any modification. For the images, the program correctly identified the presence or absence of the control and test lines in all 20 images. The linear window correctly mitigated all noise in seventeen of the twenty images (Appendix C & Table 3.1). The linear window failed for images 6, 14, and 18. (Table 3.2).

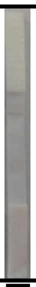
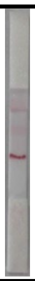
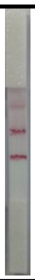
	Condition 1		
	No lines Visible	Control Line Visible	Control & Test Line Visible
User Input			
Computer Computations			
Program Output	The test failed. Please contact us.	The test was successful. The results of your test are negative.	The test was successful. The results of your test are positive.

Table 3.1: Three examples of the results calculated by the computer

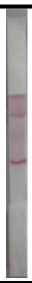
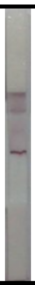
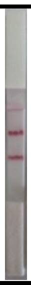
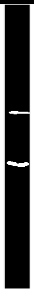
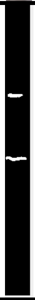
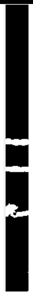
	Condition 1 FAILS		
	Control Line Visible	Control Line Visible	Control & Test Line Visible
User Input			
Computer Computations			

Table 3.2: The three examples of the linear windowing failing to work under normal conditions

3.2.2 Poor Image Lighting

The brightness of the twenty images was decreased to simulate images taken under poor lighting. The program was able to accurately determine the results of the LFA tests at -90 image brightness for all images. In seventeen out of the twenty tests, the linear window accurately filtered all noise (Appendix C & Table 3.3). Similar to Condition 1, the linear window failed to correctly evaluate images 6,14, and 18 (Table 3.4).

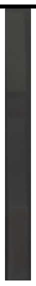
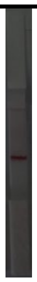
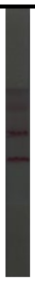
	Condition 2		
	No lines Visible	Control Line Visible	Control & Test Line Visible
User Input			
Computer Computations			
Program Output	The test failed. Please contact us.	The test was successful. The results of your test are negative.	The test was successful. The results of your test are positive.

Table 3.3: Three examples of the linear window working under poor lighting conditions

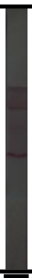
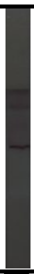
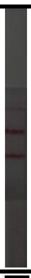
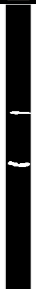
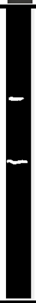
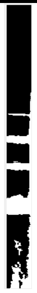
	Condition 2 FAILS		
	Control Line Visible	Control Line Visible	Control & Test Line Visible
User Input			
Computer Computations			

Table 3.4: The three examples of the linear windowing failing under poor lighting conditions

3.2.3 Poor Image Quality

To simulate blurry images, the twenty images' contrast was decreased by -90. The program produced accurate results for all twenty images. For seventeen of the twenty images the linear window filtered all noise (Appendix C & Table 3.5). For images 6, 14, and 18, the linear window failed again (Table 3.6).

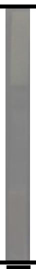
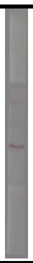
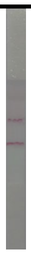
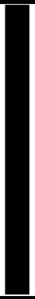
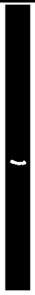
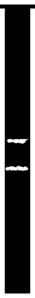
	Condition 3		
	No lines Visible	Control Line Visible	Control & Test Line Visible
User Input			
Computer Computations			
Program Output	The test failed. Please contact us.	The test was successful. The results of your test are negative.	The test was successful. The results of your test are positive.

Table 3.5: Three examples of the linear window working under poor quality conditions

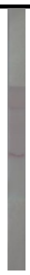
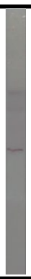
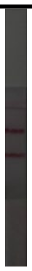
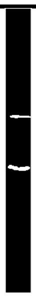
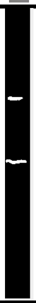
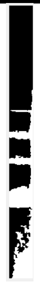
	Condition 3 FAILS		
	Control Line Visible	Control Line Visible	Control & Test Line Visible
User Input			
Computer Computations			

Table 3.6: The three examples of the linear windowing failing under poor quality conditions

3.2.4 Minimal Image Depth

For the twenty images, the minimal pixel depth required for accurate results was tested. Decreasing the depth from 256 to 64 impacted on the program's ability to detect accurate lines. The accuracy fell from 100% to 75% (15/20). After disregarding the five blank images with no test lines, the accuracy of the program decreased to 66.67% (10/15). For the control line tests only, the program accuracy was 77.78% (7/9). For the control and test line the program accuracy was 50% (3/6) (Appendix D & Table 3.7).

For the 4-bit images as the depth decreased from 256 to 16, the accuracy fell to 25% (5/20). Excluding the blank tests that did not form any lines, the accuracy was 0% (0/15) (Appendix D & Table 3.7).

Image Number	6-bit	4-bit
Image 1	√	√
Image 2	√	√
Image 3	√	√
Image 4	√	√
Image 5	√	√
Image 6	√	X
Image 7	√	X
Image 8	√	X
Image 9	√	X
Image 10	X	X
Image 11	√	X
Image 12	√	X
Image 13	√	X
Image 14	X	X
Image 15	√	X
Image 16	X	X
Image 17	√	X
Image 18	X	X
Image 19	√	X
Image 20	X	X

Table 3.7: The image results after reducing the users' pictures to 6-bit and 4-bit depth. √ means the program detected the LFA test. X means the program failed to accurately detect the successfully LFA test.

CHAPTER 4

DISCUSSION

4.1 Automated Line Detection

The MATLAB program accurately determined the results for all sixty images under all the conditions. This was due to the region of the image that was selected for edge detection and object counting as well as the linear window technique used to filter the image. In addition to discussing the results of the VBA, this chapter focuses on the filtering of the image using a linear window as the success of the linear window varied across some images. Table 4.1 gives a pictural example of how the linear window can fail and still produce the correct output. The red box is a representation of the region the program selects for edge detection and object counting.

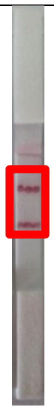
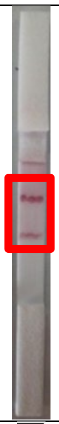
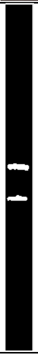
	Linear Window Fails	Linear Window Works
User input Two tests both have control line and test line formation visible to user. The test is positive		
What the computer calculates		
Output of the program	The test was successful. The results of your test are positive.	The test was successful. The results of your test are positive.

Table 4.1: Example of two correct LFA tests showing how partial failure of the linear window does not affect the results.

4.1.1 Receive and Send Emails

The program was able to enter Microsoft Outlook and find the correct subfolder with the LFA test emails. The script's default setting was to always pull out the newest email. This can easily be changed to find a certain email by writing 'email=lfafolder.Items.Item(count-#)' where '- #' is the number of emails you want to go down from the top email (Figure 2.3). The response email as seen in Figure 3.1 and Figure 3.2 was sent to the correct address and features the email content associated with the programs detection results. The script can send responses back to any email address whether given in the standard SMTP or EX format.

One of the advantages of this program was it made use of a Microsoft application. In doing so, there was no need for the program to contain email passwords or other sensitive information that could be vulnerable to data breaches. Because the program was written with the intention of being used on real patients, precautions were taken to avoid any unnecessary exposure to possible data breaches. The program itself was entirely offline. All information was stored locally and rewritten with each new run. This made it difficult for the any users' information to be stolen without hacking the entire computer.

4.1.2 Downloading and Reading Email Attachments

The script could download any attachment type given. The attachments were stored locally and saved for use. After the program finished the variables would be overwritten by the next test. The program could read common image file types such as .JPG, .JPEG, .TIFF, and .PNG.

4.1.3 Standard Conditions

Under standard conditions, the linear window accurately filtered the LFA strip for seventeen of the twenty (85%) images (Appendix C). For the three images that failed, the linear window identifies a false line (Table 3.2). In images 6 and 14, there was a clear visible line that forms outside of the location of the antibody lines. The absorbent pad was not absorbing the particles so much as it was stopping the flow of the particles. In the images it was clear that the absorbent pad acted as a wall causing the particles to build up in that region. This was very similar to the buildup of particles seen at the antibody line.

The linear window was not able to distinguish between the two types of buildup because of the pixel intensity of the fake line. The control line is used to set the threshold values for the linear window. As the image goes through the linear window, any pixel intensities above the threshold go to 256 (become white) and any below the intensity go to 0 (black). The pixel intensities of the false line were similar to the control line, so the linear window filtered them to 256. Once the pixels of the false line were converted to 256, the edge detection program counted the fake line as valid.

In essence, the build up at the absorbent pad led to the formation of an incorrect line. To the user, it was obvious that it was due to the absorbent pad and should be disregarded. The program could not make that distinction. The program saw the fake line's intensity being comparable to the control line intensity and as such, the fake line was not filtered out by the linear window.

Image 18 is a unique case. There was no formation of a red line. Instead, it was shown as a dark region near the sample/conjugate pad. The fact that it showed up as an object is due to the nature of the linear window and corresponding binarization of the image.

The edge detection program worked on binary images. Binary images, by definition, place a one for white pixels and a zero for black pixels. Because the test lines are dark, the images had to be inverted using the function 'imcomplement'. This made the dark parts of the image bright, and the light parts of the image dark. However, all dark spots were also brightened. What happened in image 18 was the dark spot was of comparable intensity to the control line. As a result, the linear window did not filter and remove the region allowing the region to remain bright white when it should not have been.

4.1.4 Poor Image Lighting

Under decreased image intensity the images that failed the standard conditions also failed this condition. The images that were successful under standard condition were also successful under this condition (Appendix C). Based on the results, the images' brightness had minimal effect on the programs accuracy. The program still chose the same regions as valid antibody lines. The effect was minimal and not none as image 18 illustrated.

For image 18, the computer-generated image looked different under decreased intensity (Table 3.4). The regions outlined as test line have drastically increased. As the image intensity is decreased, the range in pixel values decreased as well. Ultimately the white shows that those pixels were within the same intensity range as the control line and so they were not filtered to zero as they should have been.

This drastic example highlights the fact that the program will become more inaccurate as the image gets darker. At zero intensity (-100) the program would fail all images as the intensity for all pixels would be zero.

4.1.5 Poor Image Quality

With decreased image contrast, the sharpness of the image was decreased. The image was smoothed and blended to decrease pixel intensity differences. This test was especially useful for looking at the selectivity of the linear window and the program's ability to determine its thresholds. Like the other tests, the same seventeen images worked, while images 6, 14, and 18 failed (Appendix C & Table 3.6).

4.1.6 Image Depth

As expected, pixel depth had the greatest effect on the programs ability to accurately determine the results of the test. For 6-bit images, it was an overall drop in accuracy of 25%. The control line only images saw a 23% decrease in accuracy. The control and test lines saw the biggest drop in accuracy of 50%. As the bit depth decreased, the overall range of possible pixel values did as well. Unlike image intensity or image contrast, which results in changes to pixel values, changes in pixel depth are much more drastic. The range of values in a pixel has much less. As a result, some pixels that would not have the same value in the 8-bit image, now had the same value in the 6-bit or 4-bit image. This caused the linear window to treat those pixel values as part of the antibody line. The result was large regions of the LFA strip sharing the same white color and being treated as part of the same object.

4.2 Program Limitations & Future Work

The most critical future work should be to conduct the dual detection testing. There is a market available for a device of this nature that can detect both gram-positive and gram-negative bacteria. The variety of industries this device could be used in include hospitals, clinics, food safety, and water safety. The device is inexpensive and easy to manufacture, making the potential profits incredible. Conducting the tests to see if this dual detection is possible along with optimization to include sensitivity are valuable future studies.

In terms of the MATLAB automation program, there were a few of limitations that can be addressed with future work. There was one limitation regarding the send and receive email portion of the program using the Microsoft Outlook 356 email. Because the script was written to utilize Microsoft's VBA, significant revisions to the script would be needed to optimize it for accessing Gmail, Yahoo mail, or another web browser-based email account. There are ways to fix this; however, it is not recommend using a web-based account such as Gmail because it would require storing one's email password within the code.

Another issue that arose was MATLAB's ability to read some image file types. Most Android phones use .JPG or .PNG files which the program could accurately read. The biggest issue was the Apple's iPhone takes pictures in the .HEIC format. Many individuals use iPhones and MATLAB's function 'imread' cannot read the .HEIC format.

Part of the future work should include fixing this limitation. MATLAB and the MATLAB file exchange do not have any codes to read .HEIC files or convert .HEIC files into a readable format. One possible work worth exploring is downloading a third party software program with .HEIC conversion capabilities and then using MATLAB's ability to utilize the computer's command prompt. Using this command prompt, the user can order the third-party software to convert the image. One limitation of this method would be that any users of the MATLAB program would also have to have the third-party software downloaded on their computer.

There is one other limitation regarding image acquisition. The current program requires the height of the image to be the same as the height of the LFA. This was done to allow for easy determination of where the expected control line would be in the image. If the image was not the height of the LFA test, the program would use the wrong region of the code for the control line. This would create an inaccurate threshold for the linear window and ultimately cause an inaccurate result. Another limitation of the program was that it required manual inputs for the location of the region that contained the control and test lines. This region specified where to search for edges and objects. Given that the code was to be optimized for my specific LFA only, the limitation was not an issue. If the code were to be used commercially for only one LFA type it also would not be an issue. However, if the code were to be used for multiple types of LFA's this limitation would render the program unusable. Thus, part of the future work of this study should look at expanding the program to be able to detect the test and control lines for any LFA test. Ideally, it would be able to take an image of any size and accurately determine the results of the test. This would also solve the issue of needing the height of the image to be equal to the height of the LFA.

Other future work to the program should be a way to distinguish between multiple positive control lines. Unfortunately, the study conducted by Çam et al. (2017) did not attempt dual detection. Ideally, the program would be coded to know which test line was for what bacteria and then if the test line was present, it would identify the line accordingly. In the response email to the user, the program would then tell the user which bacteria the user tested positive for.

CHAPTER 5

CONCLUSION

5.1 Dual Detection LFA

Bacterial infections are a global issue effecting healthcare, food, and water industries across the world. The need for rapid detection is only exasperated by how quickly patient outcomes can change once they are infected. In healthcare, rapid detection allows for treatment to begin sooner, and helps to stop the spread to other patients through mass testing and isolation of infected patients. In the food and water industry, bacterial detection is preventative measure that can save the public from getting sick and companies millions of dollars in food recalls.

The planned LFA to be fabricated was not completed due to antibody supply chain issues caused by the COVID-19 pandemic. However, the methodology remains as an asset for future testing once the antibodies would arrive.

5.2 Automated Line Detection

The automation program, which was originally intended to be an ancillary method for validating the planned LFA's results, became the primary focus of this work. As a result, an automation program capable of notifying the patient/customer of their bacterial LFA results was developed. The program was able to enter Microsoft Outlook Mail, find the correct email and download the picture of the sender's test. The program was able to read the downloaded image and filter it using a linear window. The filtered image was subjected to edge detection and object counting functions which determined the results of the test before sending an email back to original sender with those results.

The program required the dimensions of the LFA strips to be the same in addition to the height of the image take by the user. All twenty images had to feature the LFA strip height equal to the image height. The twenty images from a study conducted by Çam et al. (2017) met these criteria and were used to test the program. For each of the three conditions, standard, decreased image intensity, and decreased image contrast, there was a set of twenty images. The program was able to accurately determine the output of the LFA for all twenty images in each set resulting in program being a perfect sixty for sixty. The lowest bit depth with perfect accuracy was an 8-bit image. 6-bit images resulted in program accuracy falling to 75% and 4-bit images resulted in program accuracy dropping to 0%. Thus, this study was able to successfully develop an automated detection system that could accurately identify and send the results of an LFA test from a user generated 8-bit image.

APPENDIX A

MATLAB SOFTWARE PROGRAM

```
% In outlook i have created a subfolder in outlook and linked it
using RULE
% This RULE sends any email with the subject 'LFA test' to the
LFA folder
%% Access the Outlook Application
application = actxserver('Outlook.Application');
myNameSpace=application.GetNamespace("MAPI");
myfolder=myNameSpace.GetDefaultFolder(6); %Goes into outlook and
pulls up inbox
lfafolder=myfolder.Folders(1).Item(2); %Goes into inbox and pulls
up the LFA subfolder
%% Get most recent email
count = lfafolder.Items.Count; %returns total number of emails in
folder,
email=lfafolder.Items.Item(count); % highest value also represent
the most recent email
%% Get email address to send response
% Getting the email this way only works for non Outlook senders
emailtype= email.get('SenderEmailType');
emailtype= convertCharsToStrings(emailtype);    % if statements
don't work with char vec of diffent sizes
%string easier to use
psender= email.get('SenderEmailAddress');
psender= convertCharsToStrings(psender);
type1= "SMTP";
type2= "EX";
if emailtype == type1
    returnaddress= psender;
else
    returnaddress=psender; %lines/function for converting EX to
SMTP would go here
end
%% Save image to current matlab folder
attachment = email.get('Attachments');
if attachment.Count ==1
    filename1= email.get('SenderName');          %find name of
sender
    filename2 = attachment.Item(1).Filename;    %find name of
file
    filename= append(filename1,' ',filename2);  %combine the two
for unique file name
    filepath= [pwd,'\',filename];               %creates file
path. pwd syntax for current folder
    attachment.Item(1).SaveAsFile(filepath)     %save file with
unique name and given file path
end
%% Image detection
I=imread(filename);
I= imresize(I,[440 38]);    %resizes image if necessary % maybe
remove this later
I=rgb2gray(I);  %color to grayscale
I= imcomplement(I); %inverts pixel intensity
```

```
region= 0.552;  %expected location of the test line as a fraction
of the whole image
while true
edge_max= iedgemax(I);  %function i created to find 'background'
baseline values

[hthresh,lthresh,meanthresh]=linwindowthresh(I,region);%function
I created to find thresholds for lin window
if meanthresh< edge_max
    txt='The test failed. Please contact us.';

    break
else
% Linear windowing
final_I=linwindow(I,hthresh,lthresh);   %function I created to
create linear window
% Binarize image, Fill holes, Removes specks, & Find Boundaries
FI= imbinarize(final_I);
FII= imfill(FI,8,'holes');
FII= bwareaopen(FII,100,8);

% Specifies the region to count objects
[B,L,N]= bwboundaries(FII(180:260,:),4,'noholes');

    if N==1
        txt=sprintf('The test was successful. The results of your
test are negative');
    elseif N>1 && N<=2
        txt=sprintf('The test was successful. The results of your
test are positve');
    elseif N>2
        txt= sprintf('Test was unsucessful. Incomplete test line
formation');
    end

    break
end
end
%% Send response Email
sendolmail(returnaddress,'Test Results',txt)
```

APPENDIX B

SUBROUTINE FUNCTIONS USED IN THE PROGRAM

iedgemax
```
function [edge_max]= iedgemax(I)

sr= size(I,1);  %size of the rows
sc= size(I,2);  %size of the columns
It1r= round(sr/20);
It2rr= sr-length(It1r);
It1c= round(sc/10);
It2cc= sc-length(It1c);

edge1= I(1:sr,1:It1c);  % all rows left edge of strip
edge2= I(1:sr,It2cc:sc);% all rows right edge of strip
edge3= I(1:It1r,1:sc);% all columns top edge of strip
edge4= I(It2rr:sr,1:sc);% all columns botom edge

%T Finds average pixel value for each edge of the image
edgemean1= round(mean(edge1,'all'));
edgemean2= round(mean(edge2,'all'));
edgemean3= round(mean(edge3,'all'));
edgemean4= round(mean(edge4,'all'));
edgemeans= [edgemean1,edgemean2,edgemean3,edgemean4];

%Keeps highest edge
edge_max= mean(edgemeans,'all');
end
```

linwindowthresh
```
function [hthresh,lthresh,meanthresh]=linwindowthresh(I,region)
 %I must be gray scale image
 % region is the region of pixels where the control line can be
found must
 % be hard coded in. It is given as:
 %(row pixel # where control line is)/(# total row pixels)

Ir= region*size(I,1);
% creates index from I (image) to use for average intensity
Ir=round(Ir,0);
Ir_high= Ir+5;
Ir_low=Ir-4;
Ic= size(I,2)/2;
Ic= round(Ic,0);
Ic_left= Ic-10;
Ic_right= Ic+10;
I_index= I(Ir_low:Ir_high,Ic_left:Ic_right);
I_index=double(I_index);

% Finds average intensity at and around control line
meanthresh= mean(I_index,'all');
meanthresh= round(meanthresh,0);
hthresh=meanthresh;
lthresh=meanthresh-1;
```

```
end
```

<u>linwindow</u>

```
function [final_I]=linwindow(I,hthresh,lthresh)
% I must be gray scale image
I1= double(I);
% Mask 1 is used to 'filter' out all values less lthresh values
less than
% lthresh get set to zero
mask1= I1< hthresh & I1>lthresh;
% img1 normailizes the values between the threshold in a linear
fashion in
% the case of this function matters very little as the difference
in
% threshold is 1
img1= mask1.*(I1-lthresh)/(hthresh-lthresh)*256;
% mask 2 finds all values greater than hthresh
mask2= I1>=hthresh;
% img2 scale all values above thresh to max brightness
img2= 256*mask2;
% final_I concats the two imgs into a single matrix. Image is now
filtered
% through linear window
final_I=img1+img2;
end
```

APPENDIX C

IMAGES UNDER THE STANDARD, POOR LIGHTING, AND POOR QUALITY CONDTIONS

Image 1

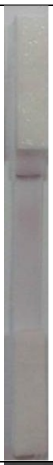
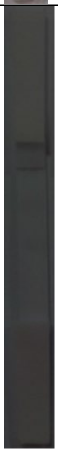
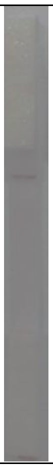
	User input from email	What computer computes	Response sent via email to user
Standard Condition			The test failed. Please contact us
Poor Image Lighting			The test failed. Please contact us
Poor Image Quality			The test failed. Please contact us

Image 2

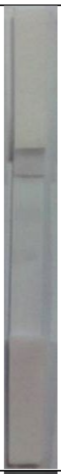
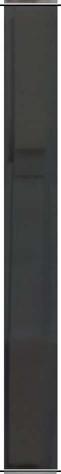
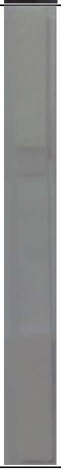
	User input from email	What computer computes	Response sent via email to user
Standard Condition			The test failed. Please contact us
Poor Image Lighting			The test failed. Please contact us
Poor Image Quality			The test failed. Please contact us

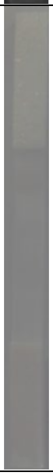
Image 3

	User input from email	What computer computes	Response sent via email to user
Standard Condition			The test failed. Please contact us
Poor Image Lighting			The test failed. Please contact us
Poor Image Quality			The test failed. Please contact us

Image 4

	User input from email	What computer computes	Response sent via email to user
Standard Condition			The test failed. Please contact us
Poor Image Lighting			The test failed. Please contact us
Poor Image Quality			The test failed. Please contact us

Image 5

	User input from email	What computer computes	Response sent via email to user
Standard Condition			The test failed. Please contact us
Poor Image Lighting			The test failed. Please contact us
Poor Image Quality			The test failed. Please contact us

LFA Strips with only the control line

Image 6 LINEAR WINDOW FAILED

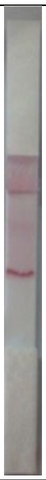
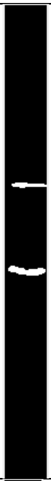
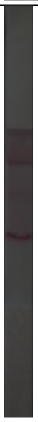
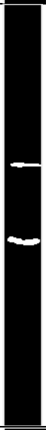
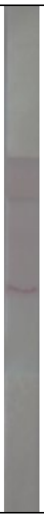
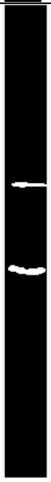
	User input from email	What computer computes	Response sent via email to user
Standard Condition			The test was successful. The results of your test are negative.
Poor Image Lighting			The test was successful. The results of your test are negative.
Poor Image Quality			The test was successful. The results of your test are negative.

Image 7

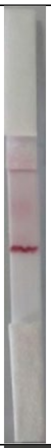
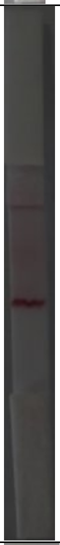
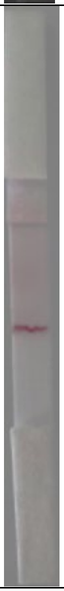
	User input from email	What computer computes	Response sent via email to user
Standard Condition			The test was successful. The results of your test are negative.
Poor Image Lighting			The test was successful. The results of your test are negative.
Poor Image Quality			The test was successful. The results of your test are negative.

Image 8

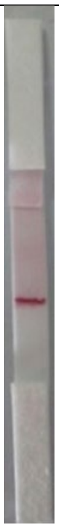
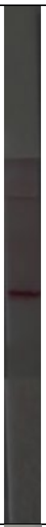
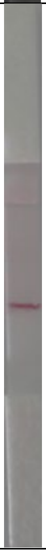
	User input from email	What computer computes	Response sent via email to user
Standard Condition			The test was successful. The results of your test are negative.
Poor Image Lighting			The test was successful. The results of your test are negative.
Poor Image Quality			The test was successful. The results of your test are negative.

Image 9

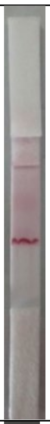
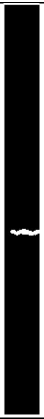
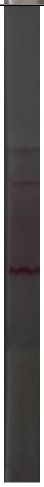
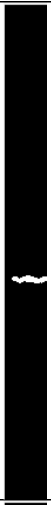
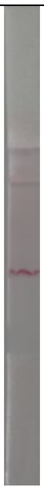
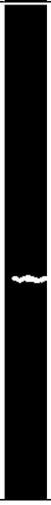
	User input from email	What computer computes	Response sent via email to user
Standard Condition			The test was successful. The results of your test are negative.
Poor Image Lighting			The test was successful. The results of your test are negative.
Poor Image Quality			The test was successful. The results of your test are negative.

Image 10

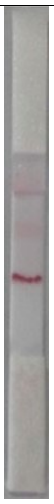
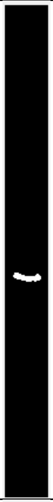
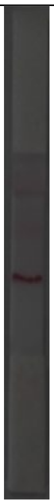
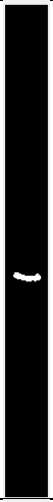
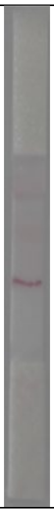
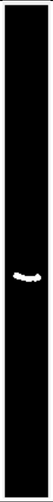
	User input from email	What computer computes	Response sent via email to user
Standard Condition			The test was successful. The results of your test are negative.
Poor Image Lighting			The test was successful. The results of your test are negative.
Poor Image Quality			The test was successful. The results of your test are negative.

Image 11

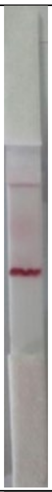
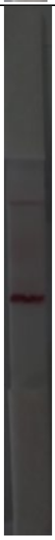
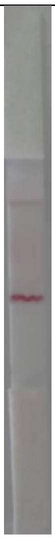
	User input from email	What computer computes	Response sent via email to user
Standard Condition			The test was successful. The results of your test are negative.
Poor Image Lighting			The test was successful. The results of your test are negative.
Poor Image Quality			The test was successful. The results of your test are negative.

Image 12

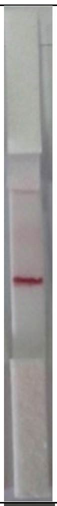
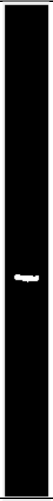
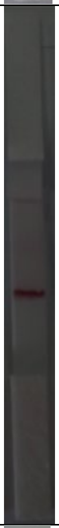
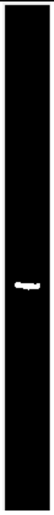
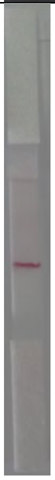
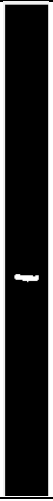
	User input from email	What computer computes	Response sent via email to user
Standard Condition			The test was successful. The results of your test are negative.
Poor Image Lighting			The test was successful. The results of your test are negative.
Poor Image Quality			The test was successful. The results of your test are negative.

Image 13

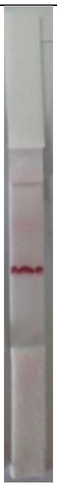
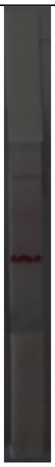
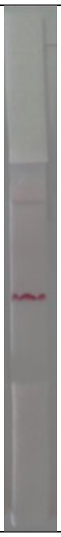
	User input from email	What computer computes	Response sent via email to user
Standard Condition			The test was successful. The results of your test are negative.
Poor Image Lighting			The test was successful. The results of your test are negative.
Poor Image Quality			The test was successful. The results of your test are negative.

Image 14 LINEAR WINDOW FAILED

	User input from email	What computer computes	Response sent via email to user
Standard Condition			The test was successful. The results of your test are negative.
Poor Image Lighting			The test was successful. The results of your test are negative.
Poor Image Quality			The test was successful. The results of your test are negative.

LFA strips with control line and test line

Image 15

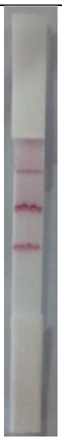
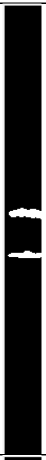
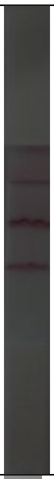
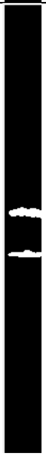
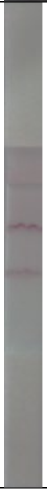
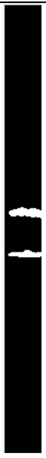
	User input from email	What computer computes	Response sent via email to user
Standard Condition			The test was successful. The results of your test are positive.
Poor Image Lighting			The test was successful. The results of your test are positive
Poor Image Quality			The test was successful. The results of your test are positive

Image 16

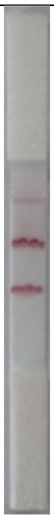
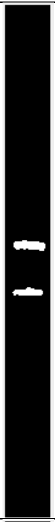
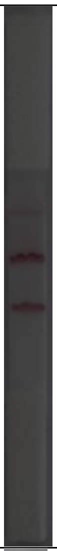
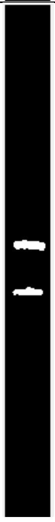
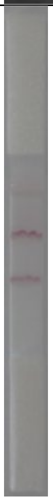
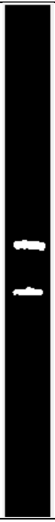
	User input from email	What computer computes	Response sent via email to user
Standard Condition			The test was successful. The results of your test are positive.
Poor Image Lighting			The test was successful. The results of your test are positive
Poor Image Quality			The test was successful. The results of your test are positive

Image 17

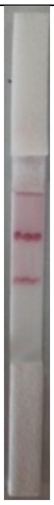
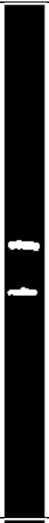
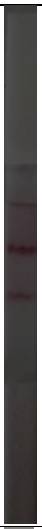
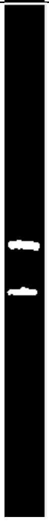
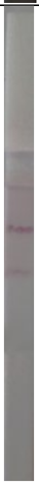
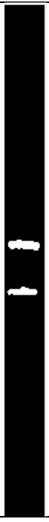
	User input from email	What computer computes	Response sent via email to user
Standard Condition			The test was successful. The results of your test are positive.
Poor Image Lighting			The test was successful. The results of your test are positive
Poor Image Quality			The test was successful. The results of your test are positive

Image 18 PROGRAM FAILED

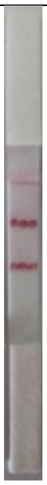
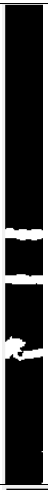
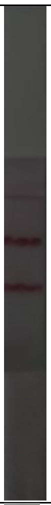
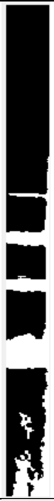
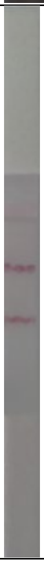
	User input from email	What computer computes	Response sent via email to user
Standard Condition			The test was successful. The results of your test are positive.
Poor Image Lighting			The test was successful. The results of your test are positive.
Poor Image Quality			The test was successful. The results of your test are positive.

Image 19

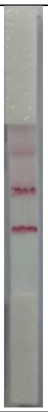
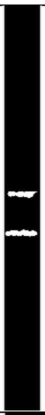
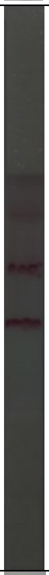
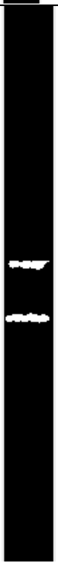
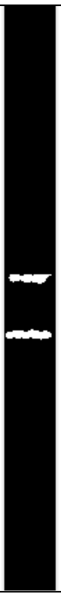
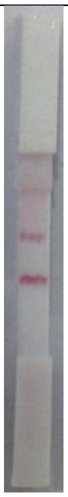
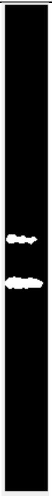
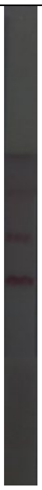
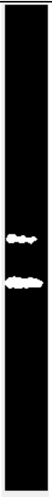
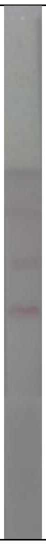
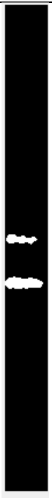
	User input from email	What computer computes	Response sent via email to user
Standard Condition			The test was successful. The results of your test are positive.
Poor Image Lighting			The test was successful. The results of your test are positive
Poor Image Quality			The test was successful. The results of your test are positive

Image 20

	User input from email	What computer computes	Response sent via email to user
Standard Condition			The test was successful. The results of your test are positive.
Poor Image Lighting			The test was successful. The results of your test are positive
Poor Image Quality			The test was successful. The results of your test are positive

APPENDIX D

IMAGE DEPTH REDUCTION TESTS

The results of the image depth reduction tests for all 20 LFA strips.

Image Number	6-bit	4-bit
Image 1		
Image 2		
Image 3		

Image 4		
Image 5		
Image 6		

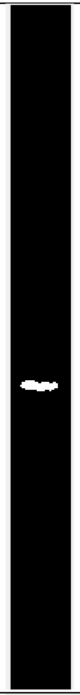
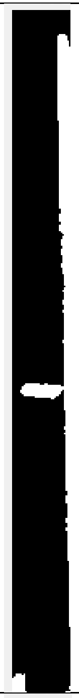
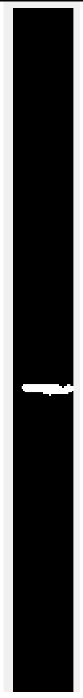
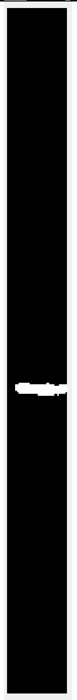
Image 7		
Image 8		

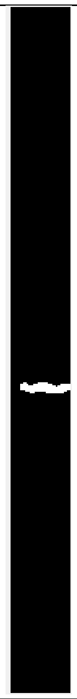
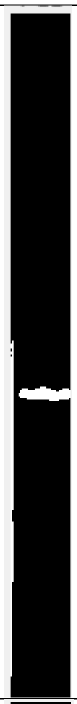
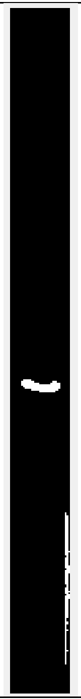
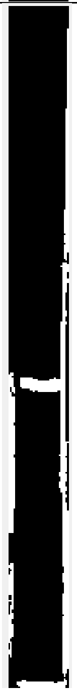
Image 9		
Image 10		

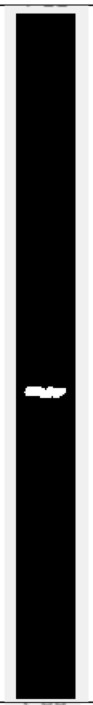
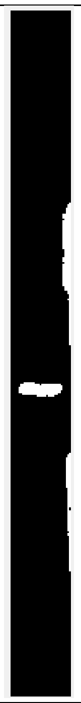
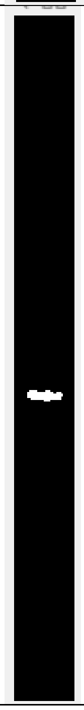
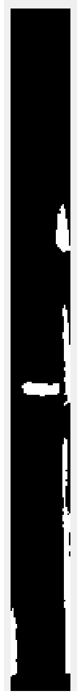
Image 11		
Image 12		

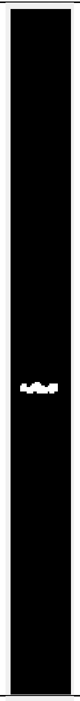
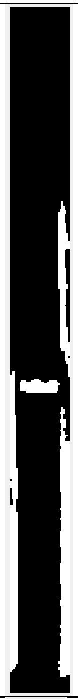
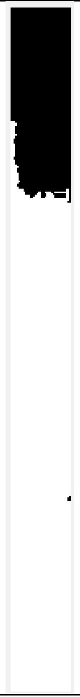
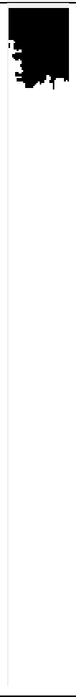
Image 13		
Image 14		

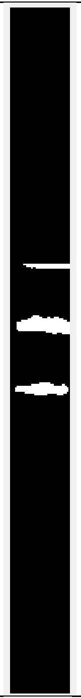
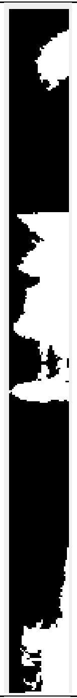
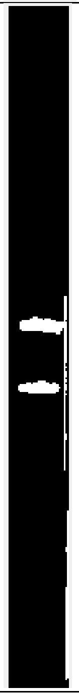
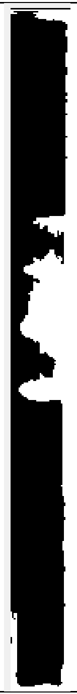
Image 15		
Image 16		

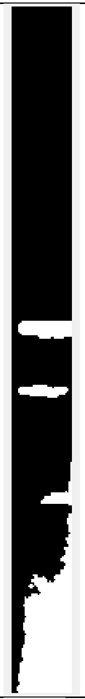
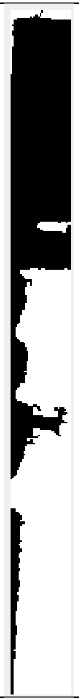
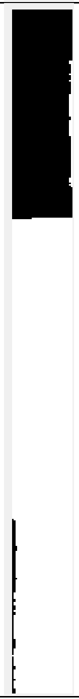
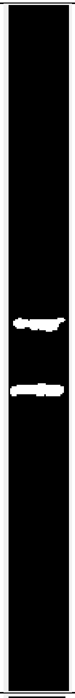
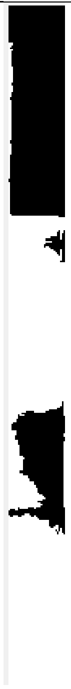
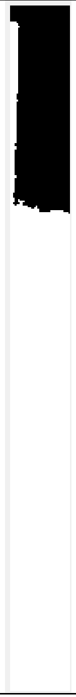
Image 17		
Image 18		

Image 19		
Image 20		

REFERENCES

Alam, N., Tong, L., He, Z., Tang, R., Ahsan, L., & Ni, Y. (2021). Improving the sensitivity of cellulose fiber-based lateral flow assay by incorporating a water-dissolvable polyvinyl alcohol dam. *Cellulose (London, England)*, *28*(13), 8641–8651. https://doi.org/10.1007/s10570-021-04083-3

Andryukov, B. G., Besednova, N. N., Romashko, R. V., Zaporozhets, T. S., & Efimov, T. A. (2020). Label-free biosensors for laboratory-based diagnostics of infections: Current Achievements and New Trends. *Biosensors, 10*(2), 11. https://doi.org/10.3390/bios10020011

Ayukekbong, J. A., Ntemgwa, M., & Atabe, A. N. (2017). The threat of antimicrobial resistance in developing countries: Causes and control strategies. *Antimicrobial resistance and infection control*, *6*(47). https://doi.org/10.1186/s13756-017-0208-x

Boehringer, H. R., & O'Farrell, B. J. (2021). Lateral flow assays in infectious disease diagnosis. *Clinical Chemistry*, *68*(1), 52–58. https://doi.org/10.1093/clinchem/hvab194

Bu, T., Jia, P., Liu, J., Liu, Y., Sun, X., Zhang, M., Tian, Y., Zhang, D., Wang, J., & Wang, L. (2019). Diversely positive-charged gold nanoparticles based biosensor: A label-free and sensitive tool for foodborne pathogen detection. *Food chemistry: X,* 3, 100052. https://doi.org/10.1016/j.fochx.2019.100052

Bu, T., Tian, Y., Sun, X., Wang, Q., Liu, Y., Bai, F., Zhao, S., &amp; Wang, L. (2020). Fe3o4@cus-based immunochromatographic test strips and their application to label-free and dual-readout detection of escherichia coli O157:H7 in food. *Food Chemistry,* 332, 127398. https://doi.org/10.1016/j.foodchem.2020.127398

Çam, D., & Öktem, H. A. (2017). Optimizations needed for lateral flow assay for rapid detection of pathogenic E. coli. *Turkish journal of biology = Turk biyoloji dergisi*, *41*(6), 954–968. https://doi.org/10.3906/biy-1705-50

Castillo-León, J., Trebbien, R., Castillo, J. J., & Svendsen, W. E. (2021). Commercially available rapid diagnostic tests for the detection of high priority pathogens: Status and challenges. *The Analyst*, *146*(12), 3750–3776. https://doi.org/10.1039/d0an02286a

Centers for Disease Control and Prevention. (2019, November 13). More people in the United States dying from antibiotic-resistant infections than previously estimated. Centers for Disease Control and Prevention. https://www.cdc.gov/media/releases/2019/p1113-antibiotic-resistant.html.

Collier, S. (2020, October 27). *Findings*. Centers for Disease Control and Prevention. Retrieved April 9, 2022, from https://www.cdc.gov/healthywater/surveillance/burden/findings.html

Koczula, K. M., & Gallotta, A. (2016). Lateral flow assays. *Essays in biochemistry*, *60*(1), 111–120. https://doi.org/10.1042/EBC20150012

Kozel, T. R., & Burnham-Marusich, A. R. (2017). Point-of-Care Testing for Infectious Diseases: Past, Present, and Future. Journal of clinical microbiology, 55(8), 2313–2320. https://doi.org/10.1128/JCM.00476-17

Liu, C., Jia, Q., Yang, C., Qiao, R., Jing, L., Wang, L., Xu, C., & Gao, M. (2011). Lateral flow immunochromatographic assay for sensitive pesticide detection by using Fe3O4 nanoparticle aggregates as color reagents. *Analytical chemistry*, *83*(17), 6778–6784. https://doi.org/10.1021/ac201462d

Mahmoudi, T., de la Guardia, M., Shirdel, B., Mokhtarzadeh, A., & Baradaran, B. (2019). Recent advancements in structural improvements of lateral flow assays towards point-of-care testing. *TrAC Trends in Analytical Chemistry*, *116*, 13–30. https://doi.org/10.1016/j.trac.2019.04.016

Moyano, A., Serrano-Pertierra, E., Salvador, M., Martínez-García, J. C., Rivas, M., & Blanco-López, M. C. (2020). Magnetic lateral flow immunoassays. *Diagnostics (Basel, Switzerland), 10*(5), 288. https://doi.org/10.3390/diagnostics10050288

Poonlapdecha, W., Seetang-Nun, Y., Wonglumsom, W., Tuitemwong, K., Erickson, L. E., Hansen, R. R., & Tuitemwong, P. (2018). Antibody-conjugated ferromagnetic nanoparticles with lateral flow test strip assay for rapid detection of Campylobacter jejuni in poultry samples. *International journal of food microbiology, 286,* 6–14. https://doi.org/10.1016/j.ijfoodmicro.2018.07.009

Posthuma-Trumpie, G. A., Korf, J., & van Amerongen, A. (2009). Lateral flow (immuno)assay: Its strengths, weaknesses, opportunities and threats. A literature survey. *Analytical and bioanalytical chemistry*, *393*(2), 569–582. https://doi.org/10.1007/s00216-008-2287-2

Qiu, Q., Dewey-Mattia, D., Subramhanya, S., Cui, Z., Griffin, P. M., Lance, S., Lanier, W., Wise, M. E., & Crowe, S. J. (2021). Food recalls associated with foodborne disease outbreaks, United States, 2006-2016. *Epidemiology and infection*, *149*, e190. https://doi.org/10.1017/S0950268821001722

Sadeghi, P., Sohrabi, H., Hejazi, M., Jahanban-Esfahlan, A., Baradaran, B., Tohidast, M., Majidi, M. R., Mokhtarzadeh, A., Tavangar, S. M., & de la Guardia, M. (2021). Lateral flow assays (LFA) as an alternative medical diagnosis method for detection of virus species: The intertwine of nanotechnology with sensing strategies. *Trends in Analytical Chemistry*, *145*, 116460. https://doi.org/10.1016/j.trac.2021.116460

Scallan, E., Hoekstra, R. M., Angulo, F. J., Tauxe, R. V., Widdowson, M., Roy, S. L....Griffin, P. M. (2011). Foodborne Illness Acquired in the United States—Major Pathogens. *Emerging Infectious Diseases*, *17*(1), 7-15. https://doi.org/10.3201/eid1701.p11101.

Scharff, Robert (2012). Economic burden from health losses due to foodborne illness in the United States. *Journal of Food Protection, 75*(1), 123–131. https://doi.org/10.4315/0362-028x.jfp-11-058

Sin, M. L., Mach, K. E., Wong, P. K., & Liao, J. C. (2014). Advances and challenges in biosensor-based diagnosis of infectious diseases. *Expert review of molecular diagnostics, 14*(2), 225–244. https://doi.org/10.1586/14737159.2014.888313

Song, C., Li, J., Liu, J., & Liu, Q. (2016). Simple sensitive rapid detection of Escherichia coli O157:H7 in food samples by label-free immunofluorescence strip sensor. *Talanta*, *156-157*, 42–47. https://doi.org/10.1016/j.talanta.2016.04.054

Tabak, Y. P., Vankeepuram, L., Ye, G., Jeffers, K., Gupta, V., & Murray, P. R. (2018). Blood Culture Turnaround Time in U.S. Acute Care Hospitals and Implications for Laboratory Process Optimization. *Journal of clinical microbiology, 56*(12), e00500-18. https://doi.org/10.1128/JCM.00500-18

Tang, R., Alam, N., Li, M., Xie, M., & Ni, Y. (2021). Dissolvable sugar barriers to enhance the sensitivity of nitrocellulose membrane lateral flow assay for COVID-19 nucleic acid. *Carbohydrate polymers*, *268*, 118259. https://doi.org/10.1016/j.carbpol.2021.118259

Zhao, S., Wang, S., Zhang, S., Liu, J., &amp; Dong, Y. (2018). State of the art: Lateral flow assay (LFA) biosensor for on-site rapid detection. *Chinese Chemical Letters, 29*(11), 1567–1577. https://doi.org/10.1016/j.cclet.2017.12.008

BIOGRAPHICAL INFORMATION

Reginald Conley II began his college career in the fall of 2018 at the University of Texas at Arlington (UTA). He was accepted into the Honors College prior to admission.

During his time at UTA, Reginald joined the Tissue Engineering tract to focus on his passion for the human body. His research included writing MATLAB software for spectroscopy data acquisition, the creation of the lateral flow assay for his senior design project, and the subsequent extension of that research for this honors project.

Reginald will be graduating summa cum laude in the Spring of 2022 with an Honors Bachelor of Science in Biomedical Engineering. He plans to finish this project and begin the patent process with his Senior Design group. Reginald is also applying to medical school for Fall 2023. His career goal, as of finishing this work, is to become a cardiothoracic surgeon and continue to invent diagnostic tools capable of improving patient outcomes.