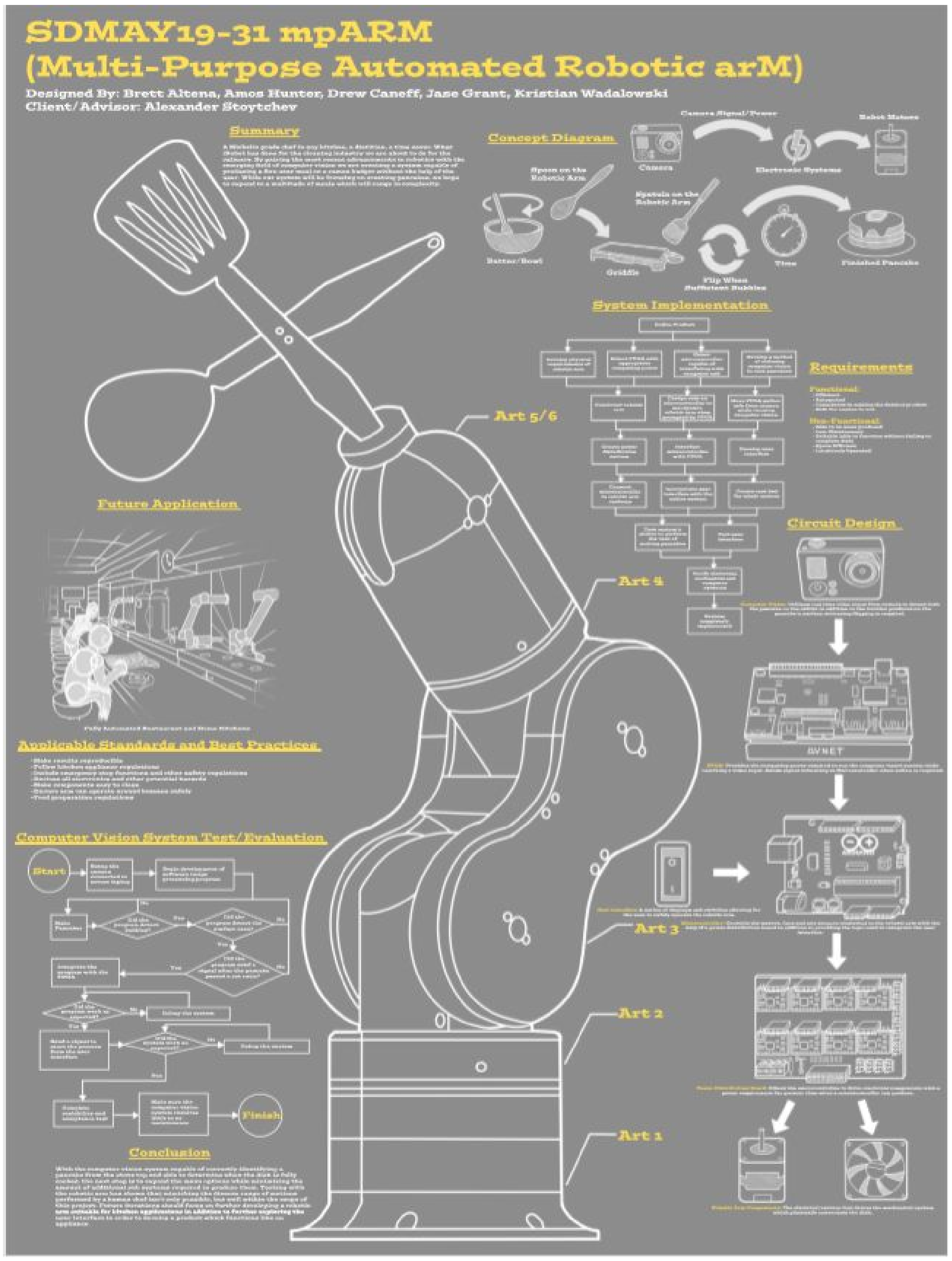
CPR E/ EE 492 Senior Design 2 sdmay19-31

Final Report

mpARM-Multi-Purpose Automatic Robotic Arm


SDMAY19-31 mpARM
(Multi-Purpose Automated Robotic arM)
Designed By: Brett Altena, Amos Hunter, Drew Caneff, Jase Grant, Kristian Wadalowski
Client/Advisor: Alexander Stoytchev
Summary
Concept Diagram
Camera
Electronic Systems
Batter/Bowl
Griddle
Time
Finished Pancake
System Implementation
Requirements
Art 5/6
Art 4
Art 3
Art 2
Art 1
Future Application
Applicable Standards and Best Practices
Circuit Design
Computer Vision System Test/Evaluation
Start
Finish
Conclusion

Executive Summary **4**
List of Definitions: 4
Requirement Specifications **4**
Functional Requirements: 4
Non-Functional Requirements: 5
System Design & Development **5**
Equipment 5
Design Plan 6
Design Objectives 6
System Constraints 7
Design Trade-Offs 7
Arm vs. Specialized Machine 7
Open Source Arm vs. Professionally Made Arm 7
Arm Material: PLA vs. ABS 8
FPGA: Python vs. C 8
Computer Vision vs A Timer 8
Bubble Detection vs. Thermal Imaging 8
Hanging the Arm Upside Down vs. Right-Side Up 9
Top Level Design 9
Safety 9
Kill Switches 10
Project Box 10
Fuses 10
ABS Plastic 10
Food Safety 10
Subsystem Designs 10
Computer Vision 10
Enclosure and Base 11
Arm Movement 12
FPGA 12
Implementation **13**
Computer Vision 13
Technologies Used 13
OpenCV 13
Hough Circles 13
Background Subtraction/ Contour Detection 14
Interface 15
Applied Standards/Best Practices 16
Arm movement 17

Technologies Used 17
Applied Standards/Best Practices 17
Enclosure 18
Technologies Used 18
Applied Standards/Best Practices 18
FPGA 18
Technologies Used 19
Applied Standards/Best Practices 20
Testing, Validation, and Evaluation 21
Test Plan 21
Integration Testing 21
System Testing 21
Computer Vision 21
Arm Movement 21
Enclosure and Base 22
FPGA 22
Validation and Verification 23
Evaluation 23
Performance Metrics 23
Results 23
Pancake flip reliability 23
Experiment 1 23
Experiment 2 25
Project and Risk Management 28
Task Decomposition & Role Responsibilities 28
Schedule: Projected Vs Actual 29
Risks and Mitigation 29
Potential 30
Actual and Mitigation 30
Conclusions 30
Team Information 31
Amos Hunter 31
Brett Altena 31
Drew Caneff 32
Kristian Wadolowski 32
Jase Grant 32
References 33

Executive Summary

The mpARM project is a robotic arm that has been programmed to flip pancakes using a computer vision system. Batter is poured onto a griddle mounted in front of the arm. A camera above the griddle feeds images into a Field Programmable Gate Array (FPGA), which processes the images to determine when a pancake is done. From there, the FPGA sends instructions to an Arduino which translates them into electrical signals that are boosted by a printed circuit board. These signals are fed into the arm and translated as the motions that allow the arm to flip the pancakes.

List of Definitions:

● mpARM: multi-purpose Automated Robotic arM

● ISU: Iowa State University

● Computer Vision: A technology which allows a computer to receive information from a camera or similar device and process that information in such a way as to interpret it and produce a response accordingly

● Iowa State: Short for Iowa State University

● mpARM : Multi-Purpose Automated Robotic Arm

● LED: Light-Emitting Diode

● FPGA: Field Programmable Gate Array, a kind of programmable microchip

● PCB: Printed Circuit Board, an electric circuit printed on a substrate with added components

● IDE: Integrated Development Environment, a software platform for writing code

● 3D Printing: Three Dimensional Printing, a technology which additively prints an object from a substance, in our case ABS filament

●ABS: Acrylonitrile butadiene styrene, a kind of thermoplastic

Requirement Specifications

Functional Requirements:

The projects functional requirements are:

1. It must have the ability to flip a pancake efficiently, so the pancake doesn't burn and the pancake is flipped smoothly.
2. It must have the ability to use computer vision to tell when the pancake needs to be flipped. If it doesn't work correctly the pancake can burn or the arm might try to flip a pancake that isn't in the area because of wrong directions given to the arm.
3. It must be safe to use for the user and the people around the device.
4. It should make pancakes autonomously.

Non-Functional Requirements:

The project's non-functional requirements are:

1. It needs to have low maintenance. If the machine breaks down it will have to be easily fixed by the user rather than having a technician come to fix it.
2. The machine needs to be reliable. The consumer needs to be able to tell it what to make and let it go and come back and the food be done. The food needs to be done right also, so there aren't wasted resources.
3. The system needs to take up space in such a way that it does not interfere with the other work being conducted within the same workspace.
4. Users will need to be able to operate the device with ease. Having the system be very intuitive will be critical in having everyday people being able to operate the device.
5. It can make many kinds of dishes that the consumer wants, fast and efficiently.
6. The system needs to be made on an industrial level. Where there can be a prototype that is manufactured fast and on a commercial scale.
7. The machine must be safe to operate.

System Design & Development

Equipment

- GoPro Hero 3
    - A GoPro Hero 3 is being used as a camera because of its features of high definition, high fps, compactness, and it was used by a team member before in a past project. The GoPro Hero 3 also has limited firmware making more

    - ideal for programming its connection to the FPGA. The team is attempting to limit the distance from the pancake surface to the camera to approximately 30 to 40 cm.
- 20" Griddle with Adjustable Temperature Control
    - A 20” griddle is being used to cook the pancakes at a controlled temperature of 375 degrees Fahrenheit. The length of the griddle is important for heat distribution and the option of detecting up to 3 pancakes at a given time. The user will have to plug in the griddle along with operating the user interface.
- Pancake Batter
    - The team tested several pancake recipes and was able to produce a homemade recipe that shows moderate to high bubbling while cooking. The batter will have to be premixed before the robotic arm can pour it on the griddle.
- Laptop
    - A laptop will be used during the testing stage as a stand-in for the FPGA. The user interface will be text-based and will accept arguments from command line execution or user input through the program.
- Robotic Arm System (Button UI, FPGA, Arduino, and THOR Robotic Arm)
    - The Robotic Arm System is mentioned in this subproject and in the comments in the code, but will not delve into details on the parts unrelated to the computer vision system. The FPGA will run the program and accept input from the user interface and the camera and will output signals to the Arduino Mega.
- Frame and Base
    - The robot needs a base to be mounted to, as does the other equipment. A frame is needed to contain the machine as well as give a place to mount the camera.

Design Plan

Design Objectives

For this project it was thought that a robotic arm would be the most applicable for the purpose of making food autonomously, where the user can leave and come back to a meal completely done. While our design is meant to handle pancakes, we want the ability to progress our work in the future to handle other dishes, so our design needs to be adaptable with only minor modifications. One such modification is the ability to switch out attachments for the different kinds of dishes that this system can make. For this design the robotic arm will be installed on a countertop or table and have easy access to the workspace where the food is being prepared and cooked.

System Constraints

1. Funding - the budget of this system is limited and we have had to make do with some components that work, but could be better. Also, the breaking of components in the system and having to fix them.
2. Time - Having the time to implement all the system functions that we want is not possible in the few months that we have to complete the project. Dealing with broken components also required time.
3. Team - The designing of this system and making of this system some member have had to do things that are new to them or have no experience with. Also, miscommunications can set back the development of the system.
4. System - The system is set to only make pancakes and nothing else.

Design Trade-Offs

Arm vs. Specialized Machine

Our initial research into automated pancake cooking revealed that the most efficient method of producing automated pancakes would be a specialized conveyor belt style machine. In fact we discovered that most types of automated food production implemented some sort of conveyor belt. With this information available we considered the overall goal of our project and decided that we should move forward with the robotic arm design. Our goal was not to make a specialized machine, but to create a generalized cooking machine. With the arm design we open our project to the possibility of future cooking applications, where as the specialized designs lock our project into whatever we specialize in.

Open Source Arm vs. Professionally Made Arm

The arm is a key component in our project, and so the selection of an arm was incredibly important to the project.

A professional quality arm would be ideal. It would either come pre-assembled or would have a carefully documented assembly guide. There would be an operation manual, and access to support from the company that created it.

A open source arm does not guarantee anything that a professional arm does. However it is significantly cheaper. As our budget is incredibly limited, we chose open source to save money.

Arm Material: PLA vs. ABS

We needed to chose the type of plastic to print our arm from. PLA is the cheapest plastic available, however it has a lower melting point which would put the arm at risk when used near the hot cooking surface.

ABS is a more expensive plastic, but the higher melting point was essential for ensuring heat resistance in our arm.

FPGA: Python vs. C

We also came up with different designs for the FPGA, but we finally choose to do it in Python rather that C. Doing it in Python code and having a bitstream increases the speed but with it running in C the pipeline could accelerate the computer vision code faster, but the time to implement this takes longer. For Python it is faster to implement computer vision and it can still run fast which was perfect for the project's deadline.

Computer Vision vs A Timer

We determined early on that we could reliably make good quality pancakes using a set time interval for flipping. However a computer vision system acts as a proof of concept for the use of cameras in cooking. We felt that if we could successfully integrate such a system into our project it would make it easier to progress our project into other forms of cooking.

Bubble Detection vs. Thermal Imaging

We needed to decide how we would use computer vision to determine when pancakes are ready to flip. Thermal imaging was our first choice. However thermal cameras are prohibitivly expensive, and we has no way to confirm that thermal imagine would work.

Based on observation of pancakes cooking we determined that a normal camera could detect the bubbles forming on the pancakes and use those to determine the flip time.

Hanging the Arm Upside Down vs. Right-Side Up

We choose to have the arm mounted to a counter top rather than above the workspace area. We chose this because we had an articulation of the arm that if hung upside down would fall off. Choosing this design took away from the arms reach and the ability to have more components on the workspace. That area is now being taken up by the robotic arm.

Top Level Design

For this design we are using the Thor robotic arm. The claw of the arm has been replaced with a spatula attachment. The modified arm and the griddle are secured to a base, so that the spatula attachment on the arm can flip pancakes on the griddle. A frame is secured around the arm and griddle that will contain the whole system without impeding the flipping motion of the arm. A camera is mounted on the frame, right above the griddle. The camera records the pancakes as they cook and sends the image data to an FPGA. The FPGA uses a computer vision program to determine when the pancakes should be flipped. When the pancakes are ready to be flipped the FPGA sends a serial input to the Arduino. The code on the arduino translates this serial input into various signals. These signals are sent to the PCB which amplifies them. The amplified signals are finally sent to the arm, where they drive the motors. The signals sent through this process will make the arm perform the pre-programmed flipping and serving motions.

Safety

Kill Switches

For the safety of the system, the users and the people around the system we are going to integrate multiple kill switches. There will be a kill switch on the Arduino and the FPGA. The power switch on the power strip will also be a kill switch.

Project Box

The electronics are contained inside of an insulated plastic enclosure. This isolates the user from potentially hazardous electricity, as well as protects the electronics from damage. The project box has buttons and switches on it to allow a user to safely interface with the machine.

Fuses

The motor control board has inherent over-current protection by means of a 30 volt, 7 amp fuse. This prevents fire and electrical hazards.

ABS Plastic

We are using ABS plastic instead of the cheaper PLA plastic, because the higher melting point of ABS is suitable for high temperature environments. This mitigates any risk of structural failure due to heat, contaminating food with melted plastic, or fire hazards caused by melting plastic.

Food Safety

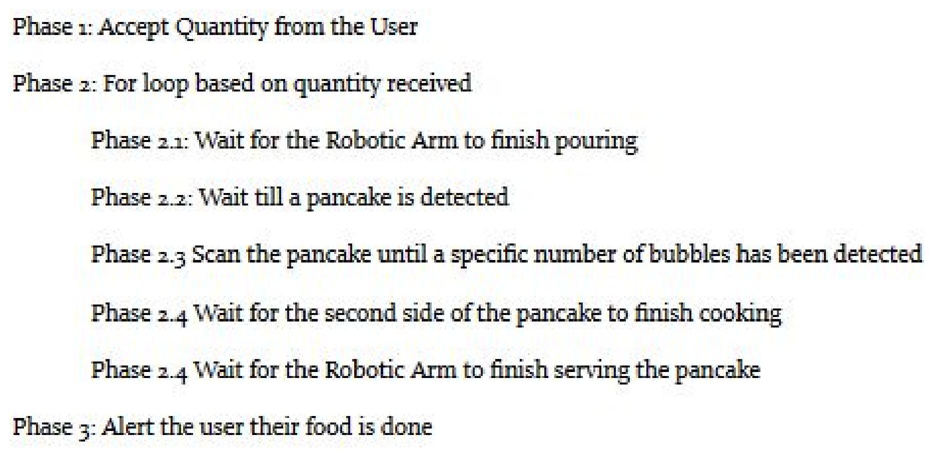
All systems are designed so that no hazardous substances will contaminate the pancakes at any point in the cooking process. Ensuring that the pancakes are safe for human consumption is our top priority.

Subsystem Designs

Computer Vision

The goal of the computer vision code is to determine when a pancake is ready to flip by counting the bubbles that form on its surface. We designed the code to isolate the pancake, and detect the bubbles as they appeared.

The program relies on the following two algorithms: Hough Circles and Background Subtraction/ Contour Detection. Two data structures used in both the detection and scanning phase is named Dataset 1 and Dataset 2. Both of these variables are arrays because Python allows the program to easily iterate through them, and add or pop a value from the beginning or the end of the array. The variable Dataset 1 keeps track of all the pixels where the pancake was detected, and Dataset 2 keeps track of the centers of the bubbles seen so that the program does not recount that bubble on the next iteration. For each algorithm a circle/ contour is shown red if it was found outside the pancake, it is displayed as blue if the circle/ contour was found for the first time (incrementing the number of bubbles found), and green if the circle/ contour was already found and counted.



```
Phase 1: Accept Quantity from the User
Phase 2: For loop based on quantity received
    Phase 2.1: Wait for the Robotic Arm to finish pouring
    Phase 2.2: Wait till a pancake is detected
    Phase 2.3 Scan the pancake until a specific number of bubbles has been detected
    Phase 2.4 Wait for the second side of the pancake to finish cooking
    Phase 2.4 Wait for the Robotic Arm to finish serving the pancake
Phase 3: Alert the user their food is done
```

Fig. 7: A diagram of the program dataflow.

Enclosure and Base

The enclosure was designed to provide a few different functions. Initially, the option of suspending the robot arm's base upside down was considered, so a frame which was sturdy enough and of proper dimensions to support the robotic arm. Although we did not ultimately follow that design path, the thought process influenced our choice of materials, namely aluminum channel for the frame and plywood for the base. Aluminum channel is relatively inexpensive, lightweight yet strong, and easy to cut and work with. Plywood is similarly cost effective and durable.

Another requirement the design fulfilled was the need for a camera mounting. The computer vision needed a camera to be mounted above the griddle to view the batter as it was cooked into a pancake. The frame accomplished this by allowing the camera to be mounted at a specific distance from the griddle surface. This gave the image proper focusing but also allowed the robotic arm sufficient clearance to access the griddle.

A third requirement the design satisfies is the need for a safety enclosure. The frame prevents the arm from moving outside of it, thereby keeping nearby people safe. As a note, the robotic arm in its current state is not strong enough to cause a person harm, but in principle it is a good practice.

Furthermore, the robotic arm requires a power supply and a housing for its electronics. The base provides a substrate on which to mount those as well. The griddle is mounted to the base as well. In addition, the frame and base combination allow the project to be handled as one unit. It made portability much easier and allowed the project to function like a module.

Arm Movement

The arm is moved when the computer vision program sends a signal for a specific action. The arm only completes the following three motions: pouring batter from a bowl onto the griddle, flipping the pancake onto its second side, and serving the pancake onto a nearby plate. Each motion is made of the arm moving a single articulation at a time using the new Arduino code. The code provided by the Thor Arm creators provided us with the Arduino code to utilize the PCB and a graphic user interface. The original plan was to modify their code to accept input from the computer vision program and send commands to the PCB. However, when trying to use their testing interface the corresponding articulation motor would shudder, stop moving and emit a high pitch sound, and then shudder again before stopping. Their code was not able to move the motors in the arm, therefore, the team chose to develop their own Arduino code to integrate into the project.

FPGA

The computer vision will run on the FPGA in python using PYNQ. The computer vision code runs slower on the FPGA and faster on the computer. To make up for this, there will be a overlay that is loaded on the FPGA and a PYNQ computer vision library that will speed up the computer vision to where it can run faster that a computer would be able to run the code.

We are using a FPGA for the computing part of this design. It will take the video from the camera in through a usb port and send it to the computer vision program on the FPGA to analyze what to do next. Once it decides what to do next the FPGA will output the data to an Arduino to tell the arm what to do next.

Implementation

Computer Vision

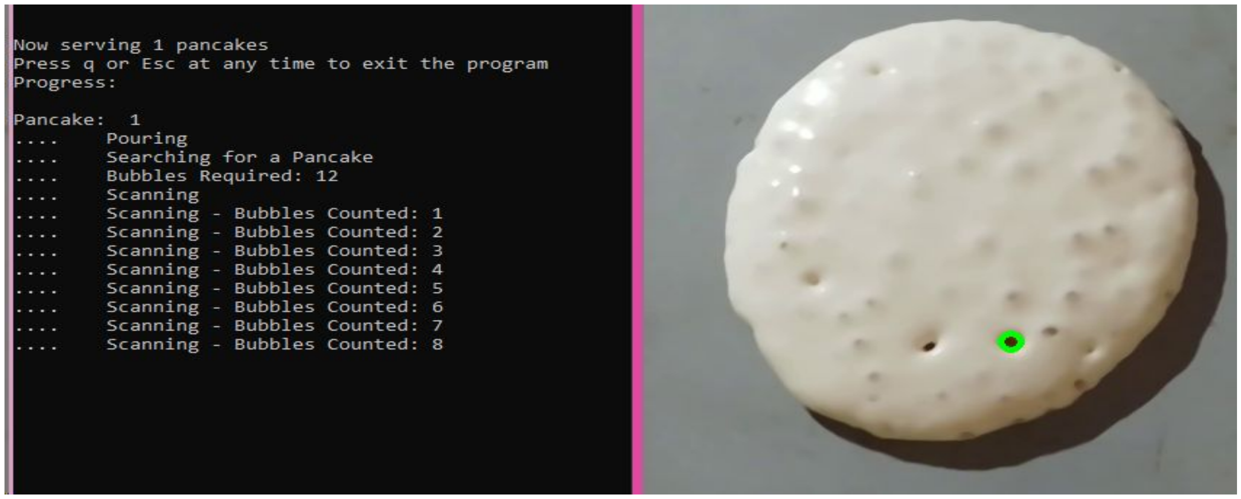
Technologies Used

OpenCV

For the computer vision code we used the python implementation of openCV. OpenCV is a freely available computer vision library developed by Intel. The library is widely used for computer vision applications, and is generally considered to be the best computer vision library available. An alternative to this would have been using Matlab's computer vision libraries, but Matlab uses it's own programming language which would be difficult to run on our hardware.

Hough Circles

The Hough Circles algorithm is used for pancake detection and in one approach it is used for bubble detection. The algorithm works perfectly for detecting pancakes that are perfect/ imperfect circles due to their large size. Hough Circles allows the program to easily obtain the radius of the pancake in terms of pixels. Through testing of the algorithm, we were able to translate how many pixels at a distance of 35 cm away from is equal to the actual pancake diameter measurement. This was done by cutting out paper circles with diameter 9 - 14 cm long, and observing the radius when the program detects the circle. This all led to a hard-coded array within the program to determine how many bubbles are required for thorough cooking by detecting the pancake radius. The program indicates where it believes the pancake is by showing a red circle around the pancake. Dataset 1 consists of all the pixels found within this circle. This ensures that the program will only count bubbles that have a center within Dataset 1. During the cooking process, the bubbles form a near perfect circle when they pop allowing the Hough Circles algorithm to pick up a majority of the bubbles.

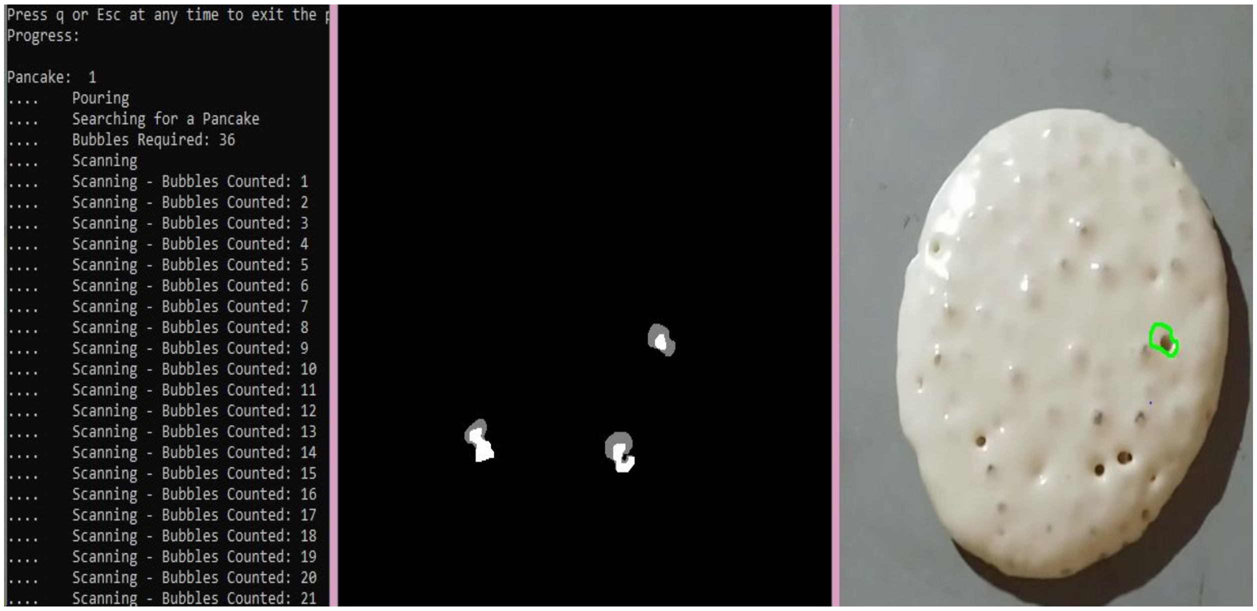


```
Now serving 1 pancakes
Press q or Esc at any time to exit the program
Progress:

Pancake:  1
....    Pouring
....    Searching for a Pancake
....    Bubbles Required: 12
....    Scanning
....    Scanning - Bubbles Counted: 1
....    Scanning - Bubbles Counted: 2
....    Scanning - Bubbles Counted: 3
....    Scanning - Bubbles Counted: 4
....    Scanning - Bubbles Counted: 5
....    Scanning - Bubbles Counted: 6
....    Scanning - Bubbles Counted: 7
....    Scanning - Bubbles Counted: 8
```

Fig. 5: A screenshot of the program using the Hough Circles algorithm, a bubble was found but was accounted for already

Background Subtraction/ Contour Detection

The Background Subtraction/ Contour Detection algorithm focuses on removing non-moving parts of the image frame and highlighting any motion detected. The modified frame then allows Contour Detection to find any objects on the frame that looks semi-circular-ish. The program does not allow any contours that are not found within Dataset 1 or any bubbles that have been found before.



```
Press q or Esc at any time to exit the p
Progress:

Pancake:  1
....    Pouring
....    Searching for a Pancake
....    Bubbles Required: 36
....    Scanning
....    Scanning - Bubbles Counted: 1
....    Scanning - Bubbles Counted: 2
....    Scanning - Bubbles Counted: 3
....    Scanning - Bubbles Counted: 4
....    Scanning - Bubbles Counted: 5
....    Scanning - Bubbles Counted: 6
....    Scanning - Bubbles Counted: 7
....    Scanning - Bubbles Counted: 8
....    Scanning - Bubbles Counted: 9
....    Scanning - Bubbles Counted: 10
....    Scanning - Bubbles Counted: 11
....    Scanning - Bubbles Counted: 12
....    Scanning - Bubbles Counted: 13
....    Scanning - Bubbles Counted: 14
....    Scanning - Bubbles Counted: 15
....    Scanning - Bubbles Counted: 16
....    Scanning - Bubbles Counted: 17
....    Scanning - Bubbles Counted: 18
....    Scanning - Bubbles Counted: 19
....    Scanning - Bubbles Counted: 20
....    Scanning - Bubbles Counted: 21
```

Fig.6: A screenshot of the program using the Background Subtraction/ Contour Detection algorithm, a contour was found but was accounted for already, and 2 other objects found did not fit a bubble's parameters

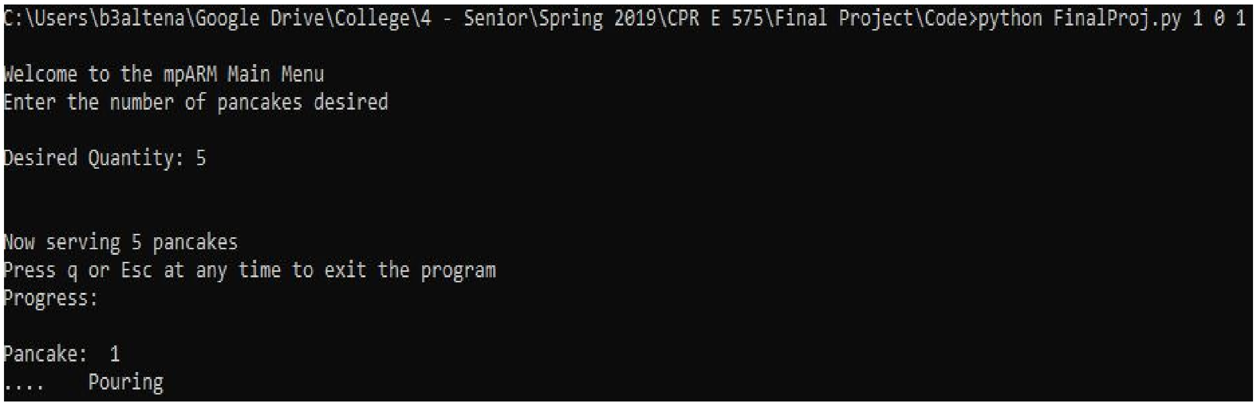
Interface

During the testing phase, the user interface consists of text-based input from the user. The arguments can be taken in through a simple user interface (method 1) or through the command line (method 2). Both methods ensure that the quantity received will not be negative, a string, and not greater than ten. The laptop will then send commands to the Arduino via outputting serial strings. These methods allow the user to decide how they wish to enter the quantity and which algorithm they want to test.

Input Format: python FinalProj.py <version number> <quantity> <approach>

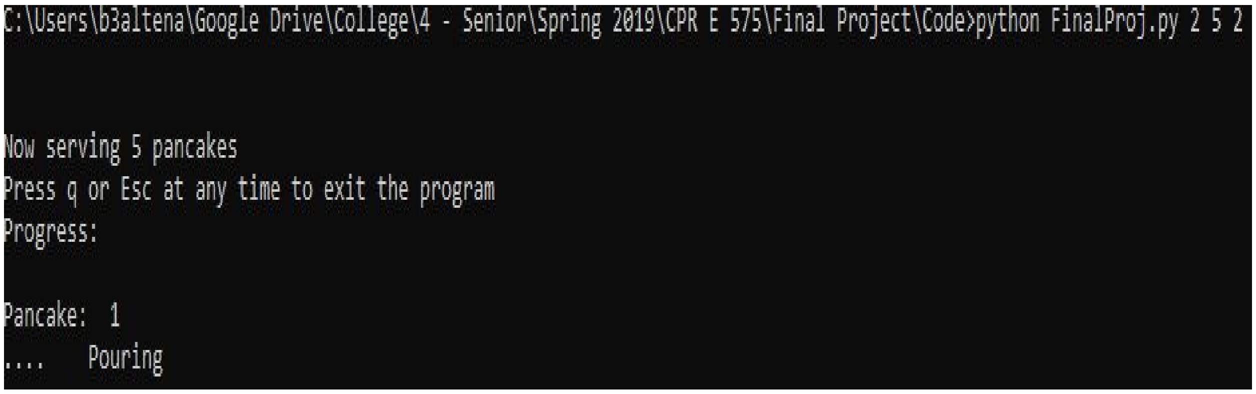
- Version
    - 1: User Interface
    - 2: Command line execution
- Quantity
    - 1 - 10: Only needed for version 2
- Approach

- ○ 1: Hough Circles
- ○ 2: Background Subtraction



```
C:\Users\b3altena\Google Drive\College\4 - Senior\Spring 2019\CPR E 575\Final Project\Code>python FinalProj.py 1 0 1

Welcome to the mpARM Main Menu
Enter the number of pancakes desired

Desired Quantity: 5


Now serving 5 pancakes
Press q or Esc at any time to exit the program
Progress:

Pancake:  1
....    Pouring
```

Fig.8: A screenshot of the execution of Method 1 utilizing the user interface and Hough Circles.



```
C:\Users\b3altena\Google Drive\College\4 - Senior\Spring 2019\CPR E 575\Final Project\Code>python FinalProj.py 2 5 2


Now serving 5 pancakes
Press q or Esc at any time to exit the program
Progress:

Pancake:  1
....    Pouring
```

Fig.9: A screenshot of the execution of Method 2 utilizing command line execution and background subtraction/ contour detection.

Applied Standards/Best Practices

Standards that were used in the implementation of the computer vision program were the following: try to capture only the pancake in the video frame, do not change the lighting during the cooking process, and always allow the user to exit the program at any point. The first standard was used to allow the program to only detect one large circle (pancake) at a time. The second standard was used to prevent false positives in the bubble detection algorithm. The third standard is a safety measure to stop all signals that are being sent to the robotic arm.

Arm movement

Technologies Used

The team measured the voltage and the current at each of the inputs and the outputs of each motor control board on the PCB. The voltage and current was consistent across each motor control board when being sent a signal using their Arduino code and user interface. The motor control board has 2 inputs that it receives from the Arduino code, DIR and STEP. The DIR signal switches from 5V (clockwise) to 0V (counter-clockwise) to signal to the motor which direction it should spin. The STEP signal indicates how fast the motor should be spinning by sending it a PWM wave. This was discovered by measuring the STEP signal with an oscilloscope and continuously sending signals through the user interface.

A team member researched and implemented how to do PWM waves manually in Arduino code. The motor control board first required the voltage to be slowly brought down from 12V (power-saving mode) to 1.2V (running mode). This was done by slowly decreasing the PWM wavelength from 19 ms apart to 500 microseconds apart and increasing when ramping out of the running mode. The code replicated the user interface exactly when it sent signals to the PCB and that is when the team realized that they had also replicated the error that the user interface was having, the motors stopped moving after turning on for half a second. After experimenting, it was realized the PWM was sending signals too fast during the running mode, therefore, the motor stopped turning. After decreasing the PWM wave during the running mode to send signals 3 ms apart, the motors turned as expected. This revolution proved that the code was working correctly and could be used to program robotic arm movements.

Applied Standards/Best Practices

The team took precautions while working with the Arduino, PCB, and the robotic arm such as the following practices: preparing to upload a blank script to the Arduino if something goes wrong immediately, connecting only the pins needed to test the current motor, shutting off the PCB completely before modifications, and ensuring to never cross two of the solder connections. The first practice was used if the motor, the PCB, Arduino, or the arm started to either smoke or break itself. The second practice was used so that we independently monitor the PCB's attributes knowing that it was only receiving signals for one motor control board. The third practice was done so that a team member would not accidentally injure or kill themselves by touching an active current (up to 5 A). The fourth practice was employed to keep the PCB's circuitry safe from short circuiting.

Enclosure

Technologies Used

The arm enclosure is fairly basic in terms of technologies applied. Aluminum channel was used as struts, which are held together with cubes of wood. The fasteners used are wood screws. The screws needed to be countersunk for certain fastenings, so that their heads would be flush with the surface. The frame is fastened to the plywood base with those same screws. The base was covered in a faux marble tape to improve aesthetics and also make the surface smoother and easier to clean. The power supply is a power strip, to consolidate the various plugs of the subsystems into one plug so that the entire machine may run off of just one wall outlet. The electronics (such as the Arduino and motor control PCB) are put into a plastic housing to protect them and to protect people from them.

Applied Standards/Best Practices

Care was taken to ensure that the enclosure was sturdy and covered the entire robotic arm, as is customary in most situations where robotic arms are used, such as in a factory setting. The live electric wiring was enclosed in a plastic box to insulate it from contact with people. Care was taken to avoid shorts and ground faults. The ends of the aluminum struts were sanded to remove burrs of metal which might cut someone. The power supply for the system is grounded to earth. Two kill switches, in addition to the power on and off, are used as safety precautions to allow a person to deactivate the machine immediately. One is on the power supply and the other is on the electronics enclosure.

FPGA

Technologies Used

Operating System

We started designing the computer vision code to run in c. The c implementation of the computer vision program doesn't require a operating system. It uses a jtag to program the vivado pipeline onto the FPGA. To program the FPGA the bitstream of the pipeline is exported to Vivado SDK and programed from there.

When we decided to run the computer vision code using Python an operating system is required. The operating system that was pre-installed on the FPGA was Petalinux, but to run computer vision on the FPGA we need a different operating system that Petalinux. The operating system that we need is an image file of PYNQ and Petalinux.

PYNQ/Jupyter

PYNQ is an operating system that is design to run Python code on the FPGA. It uses Jupyter to run the code. The python runs in segments because of the jupyter notebook format. Jupyter also allows the use of computer vision libraries. We also chose this because PYNQ computer vision system was designed to speed up everything. PYNQ also allows the use of a overlay with this we can speed up the computer vision code to the point where it is faster that a computer.

Vivado

In Vivado Design Suite we had to make a bitstream for the overlay. To make this we had to design a pipeline. The pipeline consists of ip cores that are already created. Then we connected the ip core and also synced the clocks of the ip cores. We did have to alter a few inputs and outputs of the ip cores. Once the pipeline compiles without any errors, we were able to generate a bitstream that was uploaded to PYNQ and saved it its overlay folder.

SDK

This SDK is a program to run C code on the FPGA. It also programs your pipeline on the the FPGA using a jtag. When we are designing the computer vision in C I used this program to initialise all the components and bind the inputs and output of the program using a mask. It was ready to use the computer vision code we developed in C, but then the team decided to switch to Python.

Interface

The interface for the C was a command line using an ssh program like Putty and once the testing was validated we would have a button interface for selecting the amount of pancakes that the user/consumer wants to eat. Python interface is different because the Jupyter notebook way to input is like a inputstream. We can give out inputs, but not run programs from this. If we would have gotten more testing done we were going to have a button interface that would have made things easier for the user.

Camera

The camera that we used for the FPGA was the logitech c270. The goPro hero 3 broke and we got another goPro hero 5 as a fill in, but the goPro hero 5 firmware doesn't allow it to integrate with the FPGA. The c270 was able to be integrated easily. It also allow us to move forward rather that wait for another goPro hero 3 to arrive.

Applied Standards/Best Practices

Standards that were used in the implementation of the FPGA system were the following: have the inputs and outputs be correct, not to have unnecessary things slowing the FPGA down, and always have the kill switch active to kill the program if necessary. The first standard was used to make sure the information got through the system correctly and have the arm respond correctly. The second standard was used to prevent issues with cooking the pancakes and the speed it would take to read a frame. The third standard is a safety measure to stop all signals that are being sent to the robotic arm in case of emergency.

Testing, Validation, and Evaluation

Test Plan

Integration Testing

Our project requires many separate components that need to be integrated into the final design. Prior to integrating any component, the component is tested to confirm it is operational. Since integration between components relies on dataflow, our integrations tests were designed to confirm that the data flow from one component to another was working properly.

Due to time constraints, and the failure of the arm, we were unable to fully integrate the project.

System Testing

Computer Vision

Our computer vision system was tested on video data we recorded of pancakes cooking. Flip times were determined by a human cook, and the goal was that the programs would match the ideal flip time for each pancake in the dataset. The computer vision program was constant modified to detect the pancake better and determine which algorithm proved better with test data after training the program.

Arm Movement

The arm was tested by sending signals to specific motors and seeing whether the articulation worked as expected. Two of the articulations (articulations 1 and 2) were fixed by replacing the gears connected to the stepper motors. The gears were 3D printed and cracked and broken due to constant movement done during testing and because they were made of PLA filament. Articulation 3 is not working because the belt connecting the motor to the pulley is not tight enough. The belt tighteners printed do not provide enough tension to make the belt grip on the lower gear. A zip tie was used to tighten the belt but did not prove to be effective enough. Articulation 4 is not working because due to unknown reasons. When the upper section of the arm is removed and articulation 4's motor was exposed, the motor was turning normally when a signal was being sent to it. Therefore, the issue must be mechanically between the gear on articulation 4 motor and the gear that rotates the upper section. Articulation 5 & 6 do not work due to mechanical reasons as well because the two belts in the upper section have fallen off their gears. This issue has not been fixed because it is expensive timewise to access the inner mechanics of the upper section. If there was more time for testing, the team would work on identifying and correcting the mechanical issues found in the arm.

Enclosure and Base

Testing for the frame involved physically flexing the strut and joints to make sure they would not bend or break. Once the frame was put on the base, the robotic arm was allowed to move and briefly allowed to strain against the enclosure. When the enclosure did not break, it was determined that the frame was sufficient for our purposes. The first piece of plywood material to be used for the base was found to be too small in testing, so it was replaced with a larger one which was of suitable dimensions. Similarly, once the camera was placed on the frame and the image coming from it was seen to be clear and adequate for the computer vision software, it was known that the camera placement was good on the frame. Similarly, once it was demonstrated that the base could hold the robotic arm's base, the griddle, the power supply, and the electronics enclosure, the efficacy of this subsystem was proven. The power supply showed that it was adequate by not tripping with all modules running off of it at once. This showed that it could provide sufficient current to the project and still allow its switch to function as a killswitch.

FPGA

A testing circuit was made for the FPGA. It had a set of four switches, each connected in series to an LED. When the switches were connected to the FPGA and a a switch was closed, a positive signal was sent to the corresponding input on the FPGA and an LED would light up for the pressed button. When the switches were opened, a zero signal was sent to the proper FPGA pin and no LED would light. The switches were debounced with a capacitor to allow for consistent signals. The power source for the tester was a switched mode power supply typically used for charging smartphones.

Testing of the computer vision was accomplished by reading the Jupyter notebook outputs after each segment was ran. Those outputs showed us what was happening throughout the system and let us know what needed to be fixed or changed. The segments would show completed if they finished and failed with what error if the segment could not finish running.

Test of the overlay was much harder because we can't see what the overlay is doing. We only import it into the code like a library. The computer vision code could run without errors, but wouldn't know if the overlay ran correctly or to its maximum potential. The only way for us to read this is at the end where we compared a base time on only software to a time with the overlay integrated into the computer vision program.

Validation and Verification

Evaluation

Performance Metrics

We have two major performance metrics:

Pancake flip reliability, the ability of our computer vision code to reliably flip a pancakes at or near the ideal time.

Arm motion, the ability to reliably move the arm. However, there was not enough time to complete an analysis on the arm movement's reproducibility.

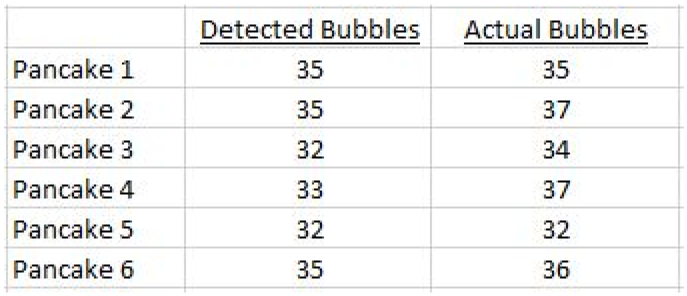
Results

Pancake flip reliability

Experiment 1

A significant portion of the program was made utilizing the first pancake videos and this led to overfitting the code to look for these specific parameters. However, the results of the computer vision program proved successful for these videos by accurately sending a signal to flip the pancake at the same time the pancake was being flipped in the video.

The following table is an example of how each pancake video was tested against the program.



	Detected Bubbles	Actual Bubbles
Pancake 1	35	35
Pancake 2	35	37
Pancake 3	32	34
Pancake 4	33	37
Pancake 5	32	32
Pancake 6	35	36

Table 1: The bubbles detected by the program, the number of bubbles detected from the team's perspective.

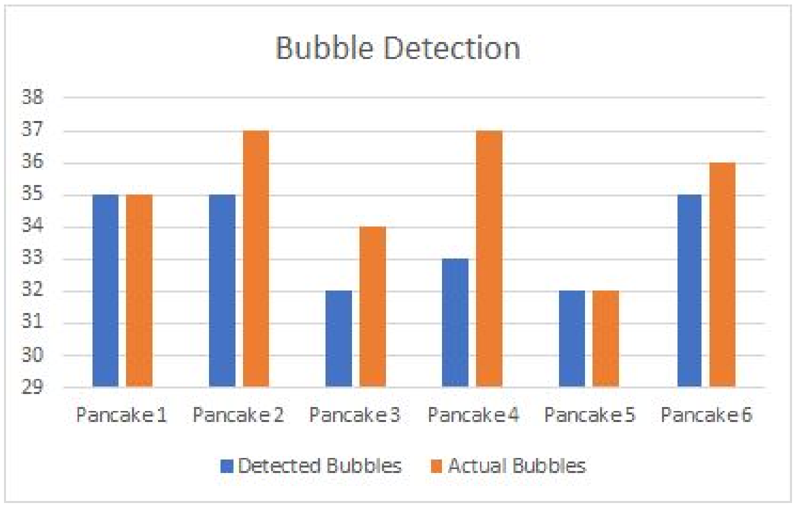
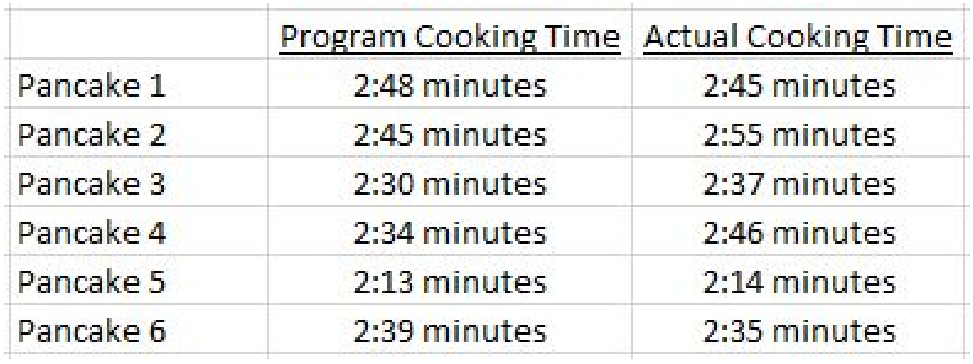


Table 2: Table 1 transformed into a clustered bar graph for easier readability.



	Program Cooking Time	Actual Cooking Time
Pancake 1	2:48 minutes	2:45 minutes
Pancake 2	2:45 minutes	2:55 minutes
Pancake 3	2:30 minutes	2:37 minutes
Pancake 4	2:34 minutes	2:46 minutes
Pancake 5	2:13 minutes	2:14 minutes
Pancake 6	2:39 minutes	2:35 minutes

Table 3: The time taken to cook the pancake, and the time taken to cook the second side of the first experiment.

One of the main reasons that experiment 1 was successful was that these videos were used to develop the training model the code would look for. The Hough Circles approach had the following issues: not able to count all the bubbles clearly due to the angle of the camera therefore the algorithm could not see the circles clearly. However, the algorithm does a good job of ensuring that it does not count the same bubble twice. The Background Subtraction/ Contour Detection approach had the following issues: lighting changes can cause major disruptions and cause false bubbles, can count a bubble more than once due to changing contour shape, false positives can occur due to batter movement. However, the algorithms combined can count every bubble that appears on the surface of a pancake.

Experiment 2

The second experiment did not go as well as expected due to different circumstances during the cooking process. A large shadow was cast from the structure holding up the camera. This interfered with the algorithms approach to detect the pancake in the image. This gave the opportunity of troubleshooting the program to look for very specific parameters when detecting the pancake such as size and color. When the program started to restrict too large or too small pancakes it stopped working for the first sequence of videos due to camera distance from the pancakes. Therefore, the camera distance is required to be equivalent across experiments to ensure that the program is able to pick up the pancake more easily.

As shown in Figure 11A, the program is able to pick out where the pancake is in the frame. In Figure 11B, the program detects a circle found in the shadows combined with a circle found in the light other than the pancake in the middle of the frame. The circle indicates where the program believes the pancake is.

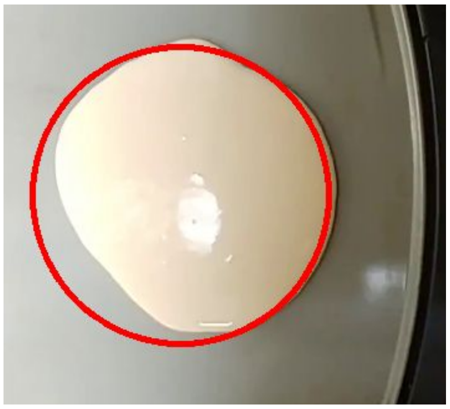


Fig. 11A: The program is able to accurately pinpoint where the pancake is.

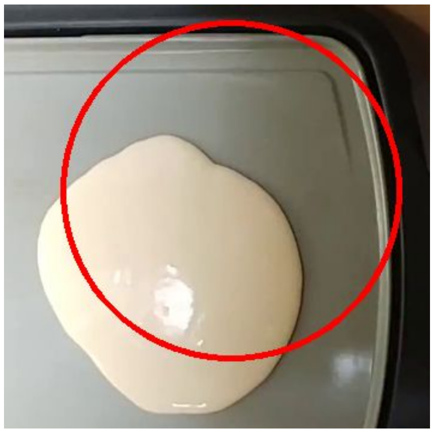
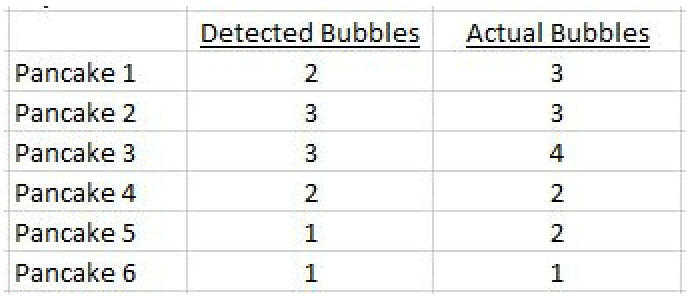


Fig. 11B: The program is not able to accurately pinpoint where the pancake is by finding a circle elsewhere.

This issue of non constant lighting in the frame led to restricting the circle size the program was able to look for. This may cause errors in the future if the pancake size is not consistent (too big or too small). The program would not have this issue if the experiment was done with ideal constants.

The next issue was that the batter used in experiment 2 was left unrefrigerated for two hours before it was utilized. This change in the recipe had unwelcome consequences such as slower cooking and fewer bubbles. The first consequence does not have a significant impact of the programming because it will scan the pancake forever if needed. The second consequence had a significant impact on the program because it was not able to reach its predetermined bubble count. For example, the pancake in Figure 7 had a diameter of 120 meaning that it needed 32 bubbles to be considered fully cooked through on the first side. However, each algorithm only detected less than 10 bubbles on each of the pancakes in experiment 2.

The following table is an example of how well the program did with one of the pancakes in experiment 2. The bubbles detected by the program were significantly less than what the program was expecting (32 bubbles). However, for the bubbles that did appear on the surface, the program was able to capture a majority of them. The program was not able to enter the second part because it continued to scan the pancake surface for bubbles.



	Detected Bubbles	Actual Bubbles
Pancake 1	2	3
Pancake 2	3	3
Pancake 3	3	4
Pancake 4	2	2
Pancake 5	1	2
Pancake 6	1	1

Table 4: The bubbles detected by the program, the number of bubbles detected from the team's perspective. The program was expecting 30+ bubbles to appear on the surface.

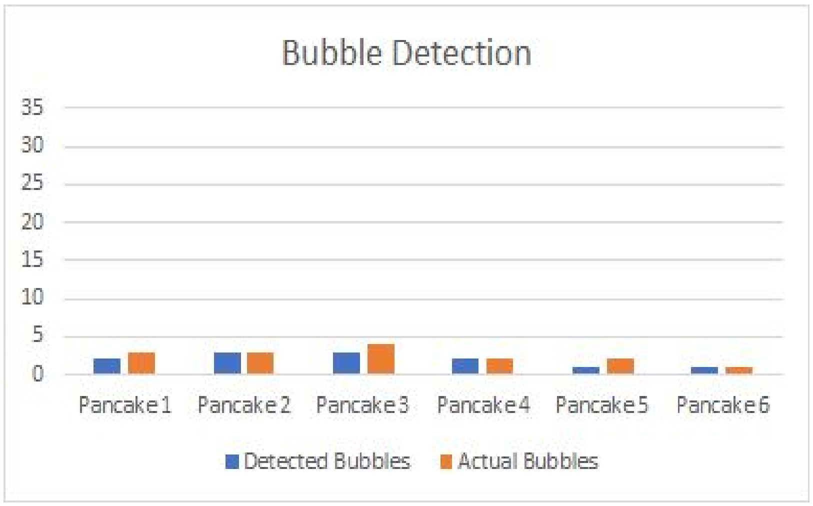
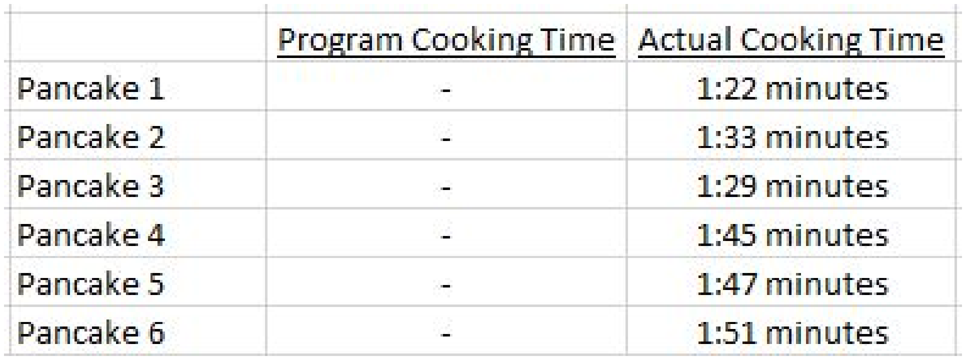


Table 5: Table 4 transformed into a clustered bar graph for easier readability.



	Program Cooking Time	Actual Cooking Time
Pancake 1	-	1:22 minutes
Pancake 2	-	1:33 minutes
Pancake 3	-	1:29 minutes
Pancake 4	-	1:45 minutes
Pancake 5	-	1:47 minutes
Pancake 6	-	1:51 minutes

Table 6: The time taken to cook the pancake, and the time taken to cook the second side of the second experiment. The program failed allowing the pancake to cook forever.

Project and Risk Management

Task Decomposition & Role Responsibilities

Amos Hunter was responsible for the frame, base, power system, and camera placement. This included determining the requirements, producing a design from those requirements, testing the subsystems, integrating them together and with the rest of the project, and evaluating their effectiveness. He also fulfilled the role of scribe for the team, which involved taking meeting notes, summarizing the meeting in an email and sending this email to team members. Communication through Slack (an online messaging and file-sharing service) was also a large part of these responsibilities. Additionally, he provided help with other parts of the project, including the robotic arm and electronic circuitry assembly and troubleshooting.

Drew Caneff was responsible for building the robotic arm, but later became involved in troubleshooting programming issues with it. He focused on 3D printing the arm and assembling it, as well as soldering all of the PCBs. He put together the project box as well. To get all 3D printed parts, he contacted an outside company and collaborated with them to get the largest parts 3D printed because the team did not have access to 3D printers which were large enough.

Brett Altena was the team leader, meeting facilitator, and the main computer vision developer. As the team leader he was responsible for the following: in charge of the general direction that the team takes when making design decisions, regularly take in input from my teammates and decide the best plan of action, prioritize tasks to ensure that the team is working efficiently and will accomplish deliverables as previously agreed upon, and assigning the roles of my team members based on their previous project experience and strengths/ weaknesses. As the meeting facilitator he was responsible for the following: tracking contributions made by members and meeting deliverable deadlines, establish an agenda to achieve short and long-term goals, start and lead team meetings, be the point of contact for the client/ advisor, and lead the team in budget/resource allocation. As the main computer vision developer he was responsible for the following: developing a program to determine when a pancake is ready to move to the next stage by detecting where the pancake is and how many bubbles rise to the surface and manage the budget for the computer vision equipment. Brett was additionally assigned the responsibility of transforming the Thor Arm's Arduino code into original code that met the needs of the project and assist in getting the robotic arm to move within the last two weeks of the project.

Jase Grant was responsible for the programing and implementation of the FPGA to work with C, when computer vision changed to Python he had to re-implement and program the FPGA to work with Python, changing the computer vision code to work with the FPGA, speeding up the system to cook perfect pancakes, and making sure the inputs from the user interface were read correctly by FPGA. He was also the assignment manager for this project.

Kristian Wadolowski was responsible for programming the motions of the robotic arm and dealing with the arduino code. Additionally assigned to design an interface, and to work on the computer visions aspects. As a clerical role, Kristian took on the role of report manager.

Schedule: Projected Vs Actual

The project was not completed on schedule. There were numerous unforeseen setbacks which consumed time and prevented progress. Among these were difficulties with getting the robotic arm to move, repairing certain broken components, programming troubles, and others. However, significant progress was made toward reaching full completion. In the end, the project had a fully constructed robotic arm in an enclosure with power and controls which could move. The computer vision worked fine. With that said, not all of the articulations could move, the FPGA was not incorporated into the project, and the arm could not make pancakes.

Risks and Mitigation

Potential

This project is operating on a limited budget, and the team must be aware that the more expensive components in the design do not cause overspending of allotted funds. It is planned to carefully assess all purchases to reduce the chance of wasted funds, and will use affordable but reliable components wherever possible.

The time frame is incredibly tight. With only two semesters to complete this project while managing other classwork, there is a ever present danger of falling behind schedule. We need to constantly assess our progress, and work hard to meet our time goals as we have set them.

Actual and Mitigation

Careful budgeting and the actions of the team treasurer allowed this team to stay mostly within budget. There were some instances in which it was necessary for team members to spend some of their own money on materials for the project. However, this did not create a great inconvenience.

Despite best efforts and intentions, time proved to be a major issue for this project. Although efforts were made to improve accountability and timely completion of subsystems, various unforeseen circumstances made progress slower than would have been optimal. Mitigation included more frequent meetings, more consistent communication, and advisor feedback among others. This is to say, in spite of mitigation, the time goals were not completely met.

Conclusions

In closing, this project was an excellent learning experience for the team members. It was an exercise in collaboration, team management, documentation, communication with a client, and other professional skills as well as a technical endeavor. Although it was not fully completed, it sufficiently demonstrated our team's ability to take on a long term, large scale project and produce results. Practical lessons learned include to have reasonable constraints on the boundaries of the project, to have systems in place for greater accountability from people, the importance of clearly defined roles, the need to allow plenty of time for project completion. Technical lessons learned include lots of information about computer vision (in fact, two of our group members took a graduate-level course in computer vision during the second semester of this project to better implement the computer vision program), as well as plenty of new information about robotics.

Future work should include completing the unfinished aspects of the current project. If one were to move forward with the project, they would find that our documentation provides sufficient information to continue work. The FPGA could be integrated into the project, the remaining articulations could be made to move, the programming could be completed to allow the arm to make pancakes (and other foods). A helpful direction to take the project in would be to update the publicly available information about the open-source Thor arm. This is because a pitfall the team encountered was a lack of clear and up-to-date information on the open-source online resources.

Team Information

Amos Hunter

Amos Hunter is a senior Electrical Engineering student at Iowa State University who enjoys working on electronics projects. He has an educational focus on controls systems and will begin a career in automation after graduation as a Plant Systems Engineer. Robotics has been a passion of his from an early age, and continues to be an area of interest educationally, professionally, and personally. He has done numerous electronics projects

Brett Altena

His mentor Professor Stoytchev and him came up with this idea of a pancake making robot after Brett's sophomore year. Brett Altena is a senior in Computer Engineering at Iowa State. He has been actively working on robotic projects each year focusing on artificial intelligence utilizing data analytics. He has accepted a job offer from 3M as a Manufacturing Systems Engineer role at their headquarters in St. Paul, MN. His job will focus on optimizing their current manufacturing system by utilizing process mining and machine learning tactics. He enjoys listening to audio books, playing sand volleyball, and challenging his friends to any kind of game.

Drew Caneff

Drew is a Senior in Electrical Engineering with a General Business Minor and diverse experiences both in the College of Design and Life Sciences. He is passionate about many fields within Electrical Engineering, but currently specialize in both biomedical applications as well as nanotechnologies. Outside of Engineering he has contributed in several fields, his favorite being the performance of Dye-Sensitized Solar Cells where he is listed as a contributor of a conference paper published in 2015. On his own and in teams he has contributed to several projects ranging drastically from wearable technology to robotic applications for home use, however, he especially enjoyed helping a design team create a mobile night club capable of syncing lights to music. He loves to be given challenges which require innovative solutions, so much so that at times it is impossible to pull him away from his work. Some of his closest friends often joke about his work ethic, as one of his most used phrases includes “can’t stop won’t stop” whenever he’s asked why he can’t take a break from a particularly interesting task.

Kristian Wadolowski

Kristian Wadolowski is a senior in Software engineering with an interest in automation, data analysis, and data management. He enjoys actively exploring and learning about cultures, languages, and global perspectives. He has experience with 5 languages, and is conversational in 2. He enjoys keeping up with economic, political, and scientific events, and love discussing them with an open mind.

Jase Grant

He is a computer engineering major and has an interest in the embedded systems side of computer engineering. Working with FPGAs, network connection capabilities, and coding of the of those systems are his biggest interests. He is a senior at Iowa State University and he is anxious to graduate so he can start his career and do what he loves for the rest of my life. This is why he is so excited for this project because it is exactly what he hopes and wants to do in the future.

References

1. Thor Robotic Arm. [Online]. Available :
   http://thorrobot.org/
2. Thor - Open Source, 3D Printable Robotic Arm. [Online]. Available :
   https://www.thingiverse.com/thing:1743075
3. Angel LM Thor - DIY 3D Printable Robotic Arm . [Online]. Available :
   https://github.com/AngelLM/Thor
4. Thor | Hackaday.io. [Online]. Available :

   https://hackaday.io/project/12989-thor