Remy: AR Assisted Cooking

Principal Investigators

Mona Elokda, Ben Hylak

Advisors

Professor Gillian Smith, Professor Erin Solovey, Professor Mughal

Special Contributions

Boiling Detection: Maria Medina

Table of Contents

Table of Figures 7

Acknowledgements 9

Abstract 11

Executive Summary 12

1. Introduction 14

2. Meet Remy 19
- Walkthrough 20
- System Capabilities 26

3. Related Work 28
- The Smart Kitchen 28
- AR + Smart Kitchen 29
- Context Aware Systems 32

4. Design 34
- Design Considerations 34
- Design Process 35
  - Personas 35
  - Practical Futurism 36
  - Scenario Mapping 36
- Design Patterns 37
  - Anchor to Context 38
  - Conform to Context 40
  - Contextual, not explicit, input 42
  - Adaptive Minimalism 43

5. Implementation 45
- The Hat 47
  - Unity Game Engine 48

Software Architecture 49
Model 49
Burners 50
Recipes 50
Managers 52
DatabaseManager 52
SpeechManager 53
RecipeManager 53
NotificationManager 53
State Machines 53
Burners 54
BurnerBehaviour 55
BurnerMiniMirror 59
Anchoring System 59
Improving Device Input 60
Gaze 60
Image Tracking 61
Hand Tracking 63
Adaptive Transparency Shader 63
Speech Recognition 65
The Rat 66
Purpose 66
Hardware 67
Raspberry Pi 3 B+ 68
Fisheye Camera 68
Thermal Camera 69
Coral Edge TPU Accelerator 70
Construction 71
Electronic 71
Enclosure 74
Software 77
Temperature Detection 78
Burner On Detection 82

Pot Detection 84
Food State Detection 88
Boiling 89
Pancakes 92

6. Evaluation 94
System Usability 94
Recruitment 94
Safety Considerations 95
Setup 96
Dependent Measures 97
Procedure 99
Survey Results 100
No Signs of Simulator Sickness 101
System Usability Results 102
Single Ease Question (SEQ) 102
Discussion 103
System Accuracy 111
Burner Temperature 111
Detecting when a Burner is On 111
Detecting Boiling Water 112
Detecting the Presence of a Pot 113
Pancake State Detection 113

7. Reflections on Design and Tradeoffs 115

8. Future Work 117

9. Conclusion 119

References 119

Appendix 125
Appendix A (Experiment Surveys) 126
Appendix B (Personas) 132
Appendix C (AMG8833 Datasheet) 135

Appendix D (SEQ Results) 139
Appendix E (SUS Results) 141

Table of Figures

Figure 1: Hand Gestures in Minority Report 20
Figure 2: Ramen Recipe Flowchart 25
Figure 3: Ramen Anchored Voice Prompt 26
Figure 4: Instructions anchored to a ramen package 27
Figure 5: Instructions anchored to headset 27
Figure 6: Waiting to Boil 27
Figure 7: Pot Detected Indicator 28
Figure 8: Voice Prompt Anchored to Pot 28
Figure 9: Hand Contextual Anchoring “Mini Mirror” 29
Figure 10: Taking ramen out of package 29
Figure 11: Ramen in the process of being added 29
Figure 12: Timer set once Ramen is added 30
Figure 13: Burner On Visualization 31
Figure 14: Notifications 31
Figure 15: Explicit Input from CounterIntelligence 34
Figure 16: Projection mapping example from CounterIntelligence 35
Figure 17: Display Modes from AR-based context-aware assembly support system 37
Figure 18: Example Storyboard 41
Figure 19: Contextual Mapping on Burner Dial 43
Figure 20: Timers are floating billboards 45
Figure 21: Timers on the rim of pans/pots 45
Figure 22: Overall System Diagram 49
Figure 23: Firebase Node Example 51
Figure 24: The Hat Overall System Architecture 53

Figure 25: Burner State Diagram 60
Figure 26: Text is obscured by the indicator ring 68
Figure 27: Adaptive transparency makes text more legible 68
Figure 28: Electronic Assembly 72
Figure 29: Field of View of Fisheye Camera 73
Figure 30: The Regular Camera 73
Figure 31: Field of View of Thermal Camera from datasheet 74
Figure 32: Field of View of Thermal Camera on top of Stove 74
Figure 33: Block Diagram of Hardware Components 76
Figure 34: Enclosure 3D design (Top view) 79
Figure 35: Enclosure 3D design (Bottom View) 79
Figure 36: Enclosure and Wall Anchor 3D Design 80
Figure 37: The Rat Mounted over The Stove 81
Figure 38: Thermal Camera Raw Output 83
Figure 39: Thermal Camera Interpolated Image 84
Figure 40: Thermal Processing Flowchart 85
Figure 41: Thermal Processing Output Images 86
Figure 42: Burner On Detection Flowchart 88
Figure 43: Pot Detection Labeling Example 89
Figure 44: Pot Detection Example 90
Figure 45: Pot Detection Flowchart 92
Figure 46: Boiling Detection Masked 94
Figure 47: Boiling Detection Mask Applied to Image 94
Figure 48: Boiling Difference, Threshold, and Bounded Rectangle Images 95
Figure 49: Pancake States 9
Figure 50: Simulated Kitchen Setup 100
Figure 51: Experiment GUI 101
Figure 52: Responses to Single Ease Questions 107
Figure 53: Participant Holding Onto The Ramen Packet 112
Figure 54: Participant using hand-anchored voice prompt 113
Figure 55: Example of what the participant might see 113
Figure 56: Pancakes made with the Rat 118

Acknowledgements

This project would not have been possible without the help and support of the following people:

- Our advisors, Professor Gillian Smith, Professor Erin Solovey, and Professor Maqsood Mughal for their support and helpful feedback.
- Our classmate, Maria Medina, for implementing boiling detection and for her support along the way.
- Our friend, Línda Perla-Giron Blanco, for helping us edit this report
- The experiment participants who helped us understand how detached monitoring can fit into people's lives.

Abstract

Our project introduces the concept of “detached monitoring” in a context-adaptive cooking system. The system has two parts: *the Rat*, a device mounted above the stove and *the Hat*, an augmented reality (AR) headset worn by the user. The Rat provides information about the user's actions and the food being cooked. This information, combined with information from the Hat, is used to determine the user's context. Instructions and status information are then embedded in the user's environment via the Hat.

The system was piloted with 7 participants in a kitchen setting. The results indicated that users found the tasks easier the more detached monitoring it incorporated, and, overall, found detached monitoring to be intuitive.

Executive Summary

Although the smart-kitchen has been heralded as the future of domestic living since the Jetsons, this promise has yet to be realized beyond showrooms and research labs. “Smart” appliances can be awkward to control, especially while cooking. They are often controlled through multiple mobile apps, or a smart speaker. These interfaces require the user to shift their hands or attention from their task in order to provide input. They also typically can't detect the user's context: what they are making, what they are doing, where they are in the home.

Our project introduces the concept of “detached monitoring” in a context-adaptive cooking system.The system has two parts: *the Rat,* a device mounted above the stove and *the Hat,* an augmented reality (AR) headset worn by the user. The rat, which can be retrofitted onto existing stoves, combines a thermal and RGB camera to understand what is happening on and around the stove. This includes detecting if a burner has been left on, pancake needs to be flipped, or that a user has completed a action (like flipping a pancake). This information, combined with information from the AR headset, is fed to a context resolution system that determines what the user is making and how the input should be used. It then determines the instructions and status information that are naturally mapped into the user's environment via the hat. For example, if a user places a pot on the stove,

the system detects this action and anchors a voice prompt directly above the pot. Alternatively, if the system detects that the user has poured in pancake mix, a "Waiting to Cook" label and a spinning ring will be augmented above that pot, and the user will be notified when they need to take further action.

We also introduce a set of four novel heuristics we developed for designing AR applications in a domestic setting: contextual anchoring, contextual mapping, contextual input and adaptive minimalism.

The system was piloted with 7 participants in a kitchen setting. The results indicated that users found the tasks easier the more detached monitoring it incorporated, and, overall, found detached monitoring to be intuitive.

1. Introduction

Cooking is difficult -- it requires following precise instructions, maintaining focus for long periods of time, and multi-tasking. Consumers have long been seeking ways to reduce this difficulty with technology, starting with cast-iron stoves from 200 A.D. More recently, in the 1960s, novel tools like electric blenders and toasters entered the kitchen to make cooking more time efficient [23]. Although they did reduce physical labor, they did not necessarily make cooking any safer or less stressful.

Since the invention of mobile phones, meal-prep kits and on-demand delivery services have been rapidly growing in popularity (nearly 4x as fast as the rest of the restaurant industry). But, these recent solutions are wasteful; they require excess packaging and resources for transportation..

Since the Jetsons, dreams of a "smart kitchen" have promised to alleviate this difficulty. But, this promise has yet to be realized beyond showrooms and research labs. Current implementations of smart kitchens often involve a slew of "smart" appliances, such as a "smart" blender or "smart" fridge. But kitchen appliances like these have the lowest rate of adoption among smart-home devices.

These appliances can be awkward to control, especially while cooking. They are often controlled through multiple mobile apps, potentially one for each

appliance, or a single smart speaker. These interfaces require the user to shift their hands and/or attention from their task in order to provide input. They are also typically unable to detect the user's context: what they are making, what they are doing, where they are in the home.

Augmented Reality (AR) is an emerging human-computer interaction modality that has potential for overcoming some of these challenges. Unlike smartphones, which are largely limited in output on a small screen, AR head mounted displays (HMD) can digitally embed information in the user's world. This allows for a more natural mapping of information, and can convey this information without requiring the user to shift their attention from their surroundings, which could be useful in the kitchen. Cooking often requires large amounts of information — measurements, ingredients, instructions — and furthermore, requires that said instructions be executed upon with precise timing. A momentary lapse of attention can lead to burnt food or, worst case, a fire. In fact, nearly half of all home fires are caused by cooking [1].

Although AR allows for output to be directly mapped to the environment, this natural mapping does not exist for input. Hand gestures, seen both in research and science fiction movies like *Minority Report [28] as illustrated in Figure 1*, are one of the most common forms of input in AR today. Hand gestures used today include movements like swiping and pinching, as well as key poses like a thumbs up or a

fist. But gestural input has a number of flaws. For one, these gestures lack any type of tactile feedback. Tactile feedback -- for example, the feeling of a click -- is a critical component of interaction [14]. Additionally, these physically unsupported, repetitive gestures can become tiresome, in what is known as gorilla arm [32].

Figure 1: Hand Gestures in Minority Report [26]

With this in mind, there are several factors to consider in the design, and we explored these in our project. First, instead of requiring the user to perform artificial gestures, we explore whether it would be preferable to detect when the user has completed an instruction, and have the system proceed to the next step. That is, to have a natural, rather than explicit, input. For example, if a user is cooking chicken and it is time to flip the chicken, the system can proceed to the next step once the chicken is flipped — it would not require an explicit gesture like a

swipe or a thumbs up. These gestures consume the user's hand, which are necessary for the tasks they are performing.

Further, we explore the benefits of providing this natural input — both when the user is present and when they are not.. In addition to detecting when an action has been done, it may be desirable to know when an action should be done. For example, many meals require waiting until a pan is hot before starting. Other meals, like cooking pancakes or eggs, require the food to have a certain appearance (like bubbles around the edges) before proceeding. These steps often require long periods of waiting, and a momentary lapse of attention could ruin the meal. It is not practical to use the camera on the user's headset for this task — the user will not always be looking at the stove. They may be preparing a different part of the meal elsewhere in the kitchen, or perhaps completing a different activity elsewhere in the home (for instance, the bathroom).

We propose a new input system, which we call "detached monitoring." With a detached monitoring system, AR is enriched with the context of the user's environment — what we call "contextual input." The user can work alongside it, or leave for short periods of time with the peace of mind that their task is being supervised. It informs the AR system both when action should be taken, and once an action has been taken. It has the potential to reduce stress and the need for explicit input, while also improving safety.

In the following chapters, we will describe Remy's different features and the design considerations behind it. We will then discuss in detail the methods we used to implement these features in the implementation chapter. Then, we will discuss evaluation methods and the results we obtained. The report will end with a conclusion, a discussion of future work, and a discussion of design tradeoffs.

2. Meet Remy

Remy[1] is a cooking aide developed for the realities of life. It assists the user when they're in the kitchen, and when they're not -- when they're following a recipe, and when they're not. The system is able to sense the user's context and adapt appropriately.

Remy has two parts: a device mounted above the stove (the "Rat") and an augmented reality (AR) headset worn by the user (the "Hat"). The Rat, which can be retrofitted onto existing stoves, has a suite of sensors including an RGB and thermal camera. Using custom computer vision algorithms and machine learning, the Rat can detect important events such as when burners are left on, when water is boiling, or when pancakes need to be flipped. The Rat also detects the user's actions to understand what they are doing.

The second part is the “Hat," our software application that runs on a Magic Leap AR headset. The primary purpose of the Hat is to embed information in the user's environment. It is what allows the user to see instructions and information as they go about their task. In addition, the device can collect input like gaze, hand position and voice input.

[1] The naming scheme of our system was inspired by the Pixar movie "Ratatouille." In the movie, "Remy" is an ambitious young rat with a passion to cook. He ends up helping a young chef learn how to cook by hiding in his hat and tugging on his hair. Hence, "Remy," "Rat," and the "Hat."

Information from both the Hat and the Rat is fed to a context resolution system that determines what the user is making and how the input should be used. For example, it decides when a recipe should be automatically started, when a prompt should be shown or when the user should be notified of a dangerous situation.

Walkthrough

Remy is designed to adapt to the context of the user. As such, there is no single entrypoint, and no set order of steps that a user must follow. What we can describe, then, is not the path a user must follow but the paths a user might follow given an objective. In this scenario, a user wants to prepare a package of instant ramen. Figure 2 shows the different paths a user might take when making Ramen.

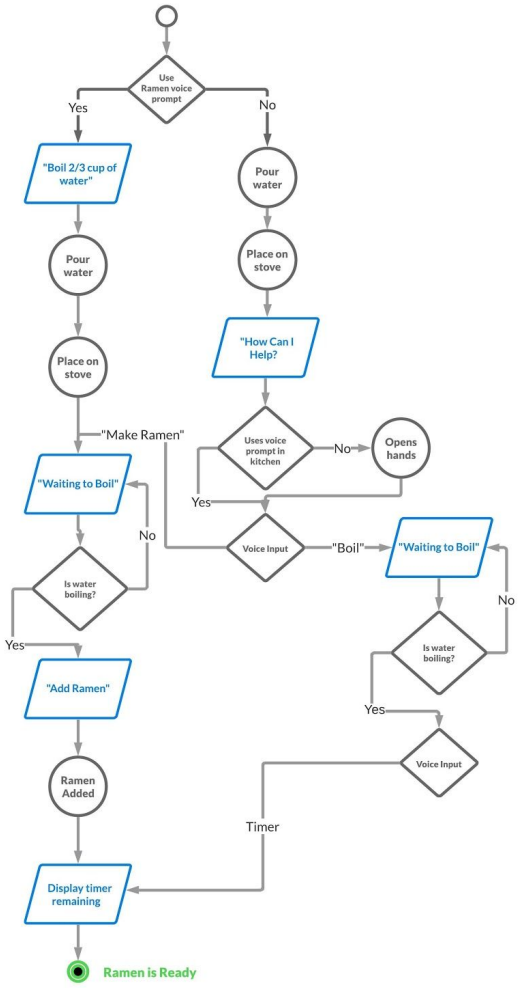


Figure 2: Ramen Recipe Flowchart

If this user does not have experience with preparing the ramen, they will likely search the package for instructions. In this case, we highlight the package

and give them a voice prompt as seen in Figure 2. If they say "make," we will start the process for making ramen.

Figure 3: Ramen Anchored Voice Prompt

The next step will be to pour 2/3 cup of water as shown in Figure 4. If they place the package down the instructions will go to heads-up mode (Figure 5).

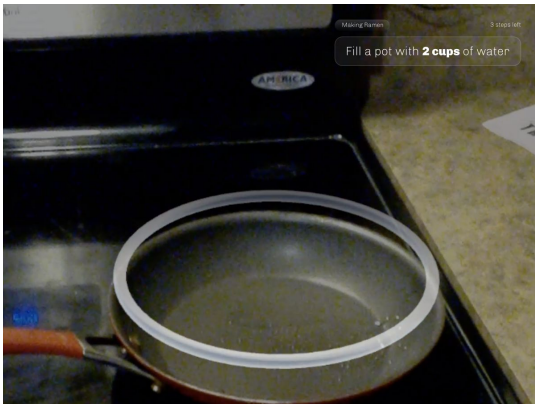


Figure 4 (Left): Instructions anchored to a ramen package
Figure 5 (Right): Instructions anchored to headset

Once they place the pot of water on the stove, the system automatically starts waiting for the water to boil.

Figure 6: Waiting to Boil

But if the user has made ramen before, they likely know approximately how much water to use. They might not even bother taking the package out of the cabinet until the water is boiling. So, perhaps they began by pouring water and

placing it on the stove. Once they do, they'll see a white ring indicating the pot has been detected. If they look at the pot, a voice prompt will appear.

Figure 7 (Left): Pot Detected Indicator

Figure 8 (Right): Voice Prompt Anchored to Pot

If they simply say "make ramen," Remy will pick up from where they are, detecting the pot with water, and start monitoring the pot until it boils.

What if the user has left the kitchen without telling Remy what they're making? Wherever they are in the home, they can open their hand and say "stove" to summon a miniature version of their stove. Saying "make ramen" will get them to the same point.

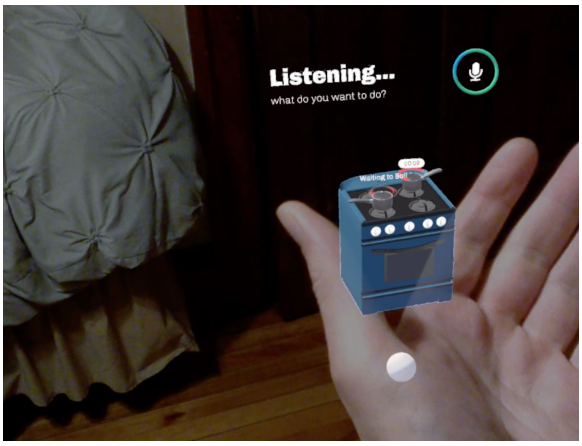


Figure 9: Hand Contextual Anchoring "Mini Mirror"

Once the water boils, in any of the above cases, the user will be notified. The system will automatically move to the next step: "Add Noodles." When Remy detects that noodles have been added, the system once again moves to the next step: setting a 3 minute timer.

Figure 10 (Left): Taking ramen out of package

Figure 11 (Right): Ramen in the process of being added

Figure 12: Timer set once Ramen is added

But what if the user is an expert -- what if they're not preparing instant ramen, but a traditional family ramen recipe? In this case, when they place water on the stove they can simply say "boil" -- Remy will notify them when it boils. Once the water boils, and they add more ingredients, Remy will be there ready to set a timer. Remy will never know every family's recipe -- but it will still help along the way.

System Capabilities

Preparing Ramen is just one of the system's many features. For instance, Remy can detect when users accidentally leave burners on. This ability is provided by a fusion of input -- the temperature of the burner and whether or not there is a

pot on the stove. Remy can communicate this situation through a combination of augmented visualizations or notifications.

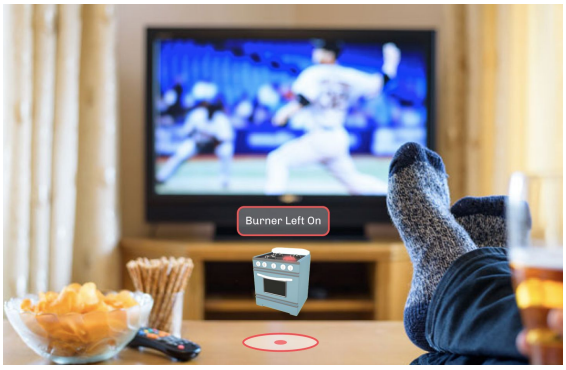


Figure 13: Burner On Visualization (Left)
Figure 14: Notifications (Right)

The system also enables users to complete more complicated recipes, like making pancakes, and enables users to check the status of their stove by simply looking at their hand.

3. Related Work

Before starting the design process, we conducted a research to explore how technology has been used in the kitchen. We also looked into projects that used AR in the kitchen as well as projects that focus on context awareness.

The Smart Kitchen

Over the years, there has been no shortage of "smart kitchen" projects [19, 28]. Many of these projects involve "smart" sensors and appliances distributed around the kitchen. For example, Stander et al. in their research on smart kitchen Infrastructures [28] use sensors to track cooking and enable remote control of kitchen appliances. It contains a set of sensors and RFID tags on utensils, in drawers, and throughout the kitchen. Nearly every appliance in the kitchen is made "smart," including a scale, blender, steamer, and a coffee machine. The entire system is controlled through a mobile application. However, in many cases, it would be easier to simply turn on the blender, rather than using a mobile phone. Having to juggle a mobile phone while cooking is difficult and potentially dangerous. Although the technical accomplishes of this project are interesting, it ignores some realities of domestic living.

AR + Smart Kitchen

CounterIntelligence [13] is one of the few “smart kitchen” projects that incorporates a form of AR. CounterIntelligence uses projectors to display information onto objects and surfaces in a kitchen. It also embeds a number of physical sensors throughout the kitchen, including above the stove, in the fridge and even inside cabinet drawers.Instead of having to take out a mobile phone to see information, multiple projectors were used to display information on surfaces throughout the kitchen. With this projected information, users provide explicit input once they've completed a step. In Figure 15, a user moves to the next step by pressing a "next" button projected onto their counter. This explicit input introduces the inconvenience of having to stop a task to provide input. Further, if the user's hands are dirty (as they often are during cooking), interacting with the system may dirty their kitchen. Enabling context-aware input is one way to avoid these problems, as we explore with Remy.

Figure 15: Explicit Input from CounterIntelligence

The system is described as being "context aware," but the paper does not describe how it adapts to the user with that context. In fact, CounterIntelligence requires users to follow specific, sequential steps, instead of conforming to the user's actions. Having a system that is able to infer actions would allow the user to avoid explicit input, and truly adapt to the user, which we explore with Remy

The Counterintelligence project also doesn't consider how the user will interact with the system outside of the kitchen. Practically speaking, users will not always be in the kitchen during the cooking process. They may be in the bathroom, for example, or watching TV in the living room as water boils or as noodles cook.

Lastly, the type of "Augmented Reality" used in CounterIntelligence differs from the modern meaning. CounterIntelligence uses projection mapping to display information throughout the kitchen. Projection mapping uses a traditional projector to project information onto surfaces as seen in Figure 15. Although this has the benefit of allowing users to use the system without a HMD, it has a number of drawbacks. For example, any surface that needs information projected onto it would require a separate projector. It also requires the said surfaces to be flat and requires the room be darkened so that projections are visible. These are serious drawbacks that limit the potential and practicality of such a system.

Figure 16: Projection mapping example from CounterIntelligence

With modern, three dimensional, AR systems, the digital and physical worlds are more closely intertwined. The user wears a head mounted device that allows the information to be displayed anywhere in their environment. The information doesn't have to be flat or against a surface -- it can have volume and float in mid air. It also allows information to be anchored to an object, even if it's something the user is holding.

Context Aware Systems

While we were unable to find cooking projects that used three dimensional AR, two projects shed a light on how it could benefit the kitchen.

Khuong et. al [11] developed a context aware assembly support system that tracks LEGO block assembly status in real-time and automatically recognizes error and completion states at each step. The user, wearing a head mounted display, could see where they should attach new blocks and which blocks are correct/incorrect. Once they installed a a block, the system would automatically advance to the next step.

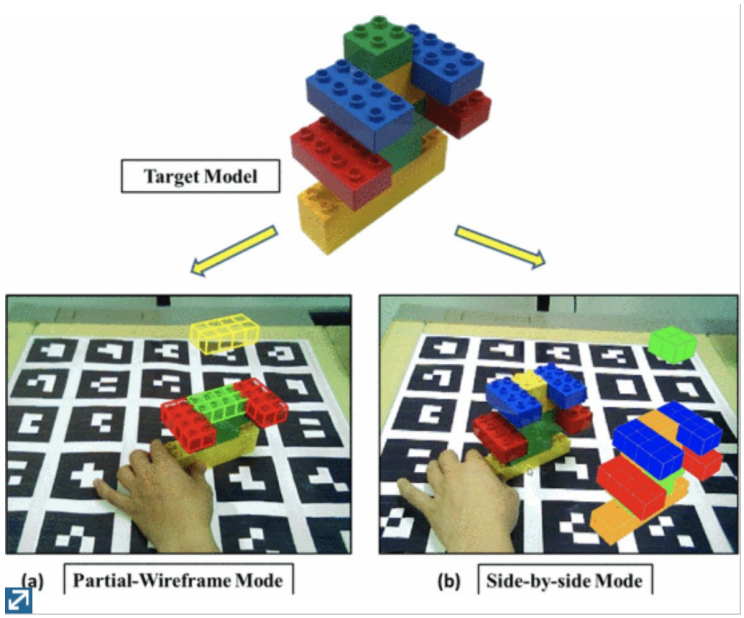


Figure 17: Display Modes from AR-based context-aware assembly support system

GuideMe is a mobile AR application that provides guidance in using home appliances [16]. The project compared AR instructions to paper and video based instructions. Although they found that paper instructions had lower error rates, users reported a lower cognitive load while using AR instructions. The higher error rates might be explained by lack of familiarity with the medium and needing to hold a phone (as it was not head mounted).

4. Design

Design Considerations

The kitchen is a unique space in the home. It has a great deal of cultural significance and is an integral part of family life. This considered, our project necessarily involves not just designing technology, but determining how to design an experience. We began with familiarizing ourselves with designing for this space. In Designing Technology for Domestic Spaces: A Kitchen Manifesto [2], Bell and Kaye comprehensively outlined a number of pitfalls when designing technology for the kitchen. Their paper asks how one designs "...not for efficiency, but for experience, affect, and desire. The challenge is to make sense of people's daily practices so that these practices can meaningfully inform design and innovation." Along these lines, rather than forcing users to behave a certain way, the authors encourage designers to "find and support rituals of domesticity." Although the authors do not discuss implementation specifics, much less AR, the paper provided a solid starting point. It became clear that domestic spaces pose unique challenges and that we needed to deeply consider how people actually behave in the kitchen.

Design Process

Personas

To begin, we defined potential users of our system (personas). Personas are representations of specific users, each with unique needs. The personas we created include the "Busy Bee" (young professionals who are short on time), the "College Kid" (someone who is so inexperienced they could burn ramen) and the "Octopus" (a parent trying to juggle three toddlers while preparing breakfast). Next, for each persona, we came up with an extensive list of contexts users might find themselves in while using our system. This includes cooking, of course, but also situations that arise while cooking. For example, the Busy Bee might leave the burner on in a rush to an important meeting. The College Kid might be staring at a jar of tomato sauce, wondering how to make pasta. An Octopus might need to run to their room to grab Tylenol for their kid with 102 degree fever, while they are preparing breakfast for the rest of the kids. After developing this list of contexts, two additional design goals emerged:

1. **People are not always in the kitchen while cooking**
2. **Different people will complete the same task in different ways, depending on situational factors.**

Practical Futurism

This led us to create a design manifesto we call "Practical Futurism." Although the details of this design manifesto are out of the scope of the paper, the gist is to design for how life really is. In life, we burn things, we pour too much and we drop batter all over the floor. In life, we realize we are late to an eye doctor's appointment in the middle of making lunch. Life is messy. Designers creating applications for mobile phones have more room to ignore this -- the application's stage is not the world but a small phone screen. But, when designing AR applications this becomes imperative -- the user's world is the stage.
Wit we narrowed down the specific scenarios we

Scenario Mapping

Next, we revisited the scenarios we had previously identified during the persona creation process. We decided to design our system around a diverse, yet specific, set of scenarios. The scenarios we chose were: setting a timer, making Ramen, and making pancakes. These scenarios were chosen because each differs in the level of complexity and system involvement, while also being representative of other tasks. For example, making pancakes is similar to making chicken (each requires flipping after reaching a certain state).

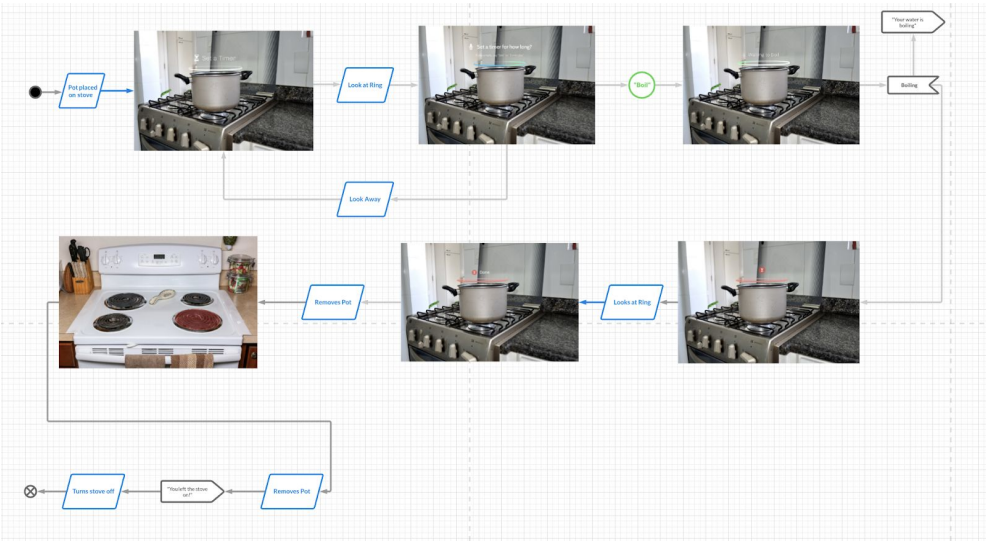


Figure 18: Example Storyboard

We then mapped out each scenario as seen in Figure 18. These maps blended both design elements and user actions, and provided an effective way to communicate and evaluate the designs, prior to implementation.

Design Patterns

When designing a mobile app, there are countless guidelines and best practices that can be employed. For example, both Apple and Google offer practical design guidelines tailored to their specific platform. Although Magic Leap does offer a limited set of design guidelines, these are tailored for experiences that

demand users' complete attention (like games). Our system works in tandem, and blends into the real world. Due to these differences, we found the Magic Leap guidelines to be largely inapplicable.

Unable to find an existing set of heuristics tailored for AR, we referred back to the timeless heuristics developed by Nielsen Norman Group [17]. These include "visibility of system status," "match between system and the real world," and "user control and freedom." While designing with these heuristics in mind, we developed a number of new guidelines that we strived to emphasize throughout our work.

Anchor to Context

Remy does not have a centralized user interface — instead the user interface is distributed across the kitchen, anchored to objects — more specifically, contexts. This allows direct mappings between digital information and the source of the said information. Our system has a plethora of examples. When a user is setting a timer or starting a recipe, they speak directly to the burner itself as seen in Figure 7 (left). Once this timer is set, we know which burner the timer is for and can place a timer directly around the rim of the pot as seen in Figure 12. This is particularly useful if the user is cooking multiple things simultaneously -- when a timer finishes, they are able to see precisely which pot the timer is for.

Similarly, when the user is following a recipe, instructions are anchored to the relevant item. This could be the bowl they're using, Ramen package they were looking at, or burner cooking the meal. If the user needs to know what the first step for a certain meal is, they need to look no further than where their food is. Again, this is particularly useful if the user is cooking multiple dishes simultaneously -- it could be easy to forget which dish is which and add the right ingredient to the wrong pot.

Another example is how we communicate setting the temperature of the stove. Rather than verbalize the setting ("Medium High") we directly augment the level onto the dial. This also allows the system to communicate with a higher level of fidelity.

Figure 19: Contextual Mapping on Burner Dial

If a burner is left on, we directly augment a pulsing red disk onto the stove. This directly communicates which burner is left on in a natural fashion. Additionally, when a burner is left on, we have a pulsing red ring around the dial for the burner that needs to be turned off, further clarifying the mapping between the information and how to remedy the situation. Interesting, in *Design of Everyday Things*, Don Norman specifically calls attention to the often confusing mapping between burners and dials.

If the context of the stove is not available -- for example, the user is in a different room -- we recreate a miniaturized version and anchor the UI appropriately (an idea we call "Mini Mirror'). An example of this is available in Figure 9. When the user opens up a voice prompt using a hand gesture, they see a miniature stove mirroring the state of their actual stove. The UI is anchored just like it would be on their real stove. This, again, carries the benefits of "natural mappings" as described by Don Norman [18].

Conform to Context

This rule is related to contextual anchoring, but still distinct. Fitting the context is more of an aesthetic concern. As much as possible, we believe the UI should be conform to the context and object -- essentially becoming an extension of

the object. This results in a true augmentation of an object or context. This can be seen throughout a number of UI elements -- for example, timers.

Figure 20 (Left): Timers are floating billboards

Figure 21 (Right): Timers on the rim of pans/pots

The above images show two designs we considered during the design process. On the left, the timers do not conform and are simply displayed as floating billboards. The design on the right, used by our system, conforms to the context, in this case the rim of the pans/pots. This design best implements a number of heuristics provided by Nielsen Norman Group, including "match between system and the real world" and "aesthetic and minimal design."

Contextual, not explicit, input

Unlike traditional systems which rely almost entirely on explicit input, our system depends mostly on implicit contextual input. That contextual input has two flavors: behavioral and environmental.

Behavioral input describes actions, both intentional and subliminal, that the user takes that can be used as input. For example, if the user is making pancakes and they are told to turn the stove on, they are automatically advanced to the next step once the burner is turned on. When a user places a pot on the stove, a white ring appears to let them know it's been detected. If the user looks at the pot they just placed for more than a certain period after said action, it transitions to listening or voice input. Behavioral input is often multilayered and the combination of multiple input sources. When speaking to a voice assistant, like Alexa or Google Assistant, there is no contextual basis for the conversation. In our system, we use objects as a contextual basis -- a burner or a package of ramen, for instance (enabling the user to "speak to" the object). In addition to being the perfect place to anchor the UI to (contextual anchoring), the knowledge of what the object is enables the system to suggest potential actions and infer meaning with less verbosity from the user.

Environmental input describes events that happen in the user's environment, indirectly related to the user's actions. For example, if the temperature of a burner is not decreasing over a certain time period, and there is no pot on the stove, the system can determine that the stove has been left on. It covers inputs related to food: if water is boiling, if a pancake needs to be flipped. Environmental input is typically used for monitoring food and automatically advancing to the next step of a recipe.

Adaptive Minimalism

In recent years, designers have been striving for a "minimal" UI, which in 2D user interfaces often means removing the non essentials. In the context of AR, a minimalistic user interface is important: it obscures as little of the user's real world as possible. Obscuring the real world is especially dangerous in a safety critical context like cooking. Our user interface has adaptive minimalism — that is, the minimalism adapts to the user's context. One example of this happens when instructions are anchored to a pot or pan. The instructions move to the rim of the pot or pan when the user's hands come near. This allows a user to see the instructions from afar, while also retaining full visibility of their task (for example, adding noodles to a pot of boiling water).

Another example of this adaptive minimalism are the augmented rings around pots placed on the stove. The ring fades out towards the back, depending on the direction the user is looking at the pot. This helps make text more visible and avoids obscuring anything that might be behind the pan.

5. Implementation

To meet the aforementioned goals, we have implemented a system with two primary components: a device mounted above the stove (the “Rat”) and the software running on the AR HMD worn by the user (the “Hat”). The Rat monitors the stove and the user's actions near the stove, and sends this input to the Hat. The Hat also detects user input, like speech, hand position and gaze, but the primary purpose is to display information in the user's environment.

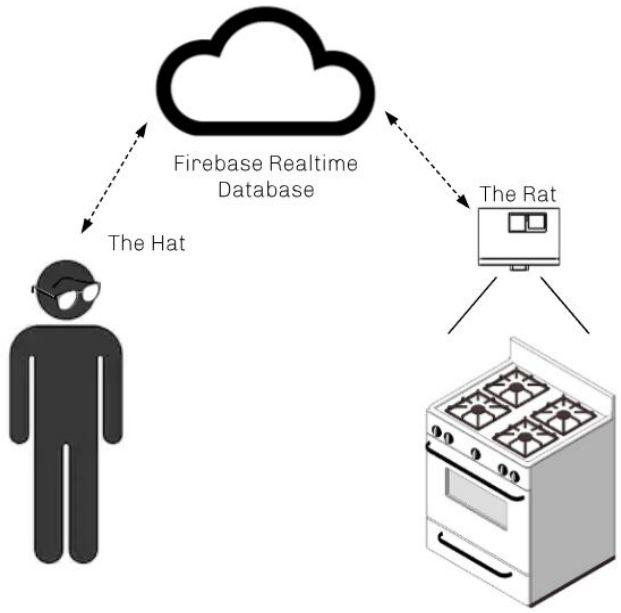


Figure 22: Overall System Diagram

The Rat and the Hat communicate through Firebase. Firebase is a NoSQL database that enables real time communication between the two devices (< 30ms readtime). Since the Rat is providing input to the Hat, low latency communication is required. Firebase was also chosen because it is easy to develop and because it would allow communication over different wifi networks so the user can still get notifications when they leave the house. The database has a root node that represents the whole stove, and a child node for each burner. Each burner has attributes including the temperature, whether or not a pot is detected, and whether or not boiling water is detected. The following figure shows an example of one of the four nodes.

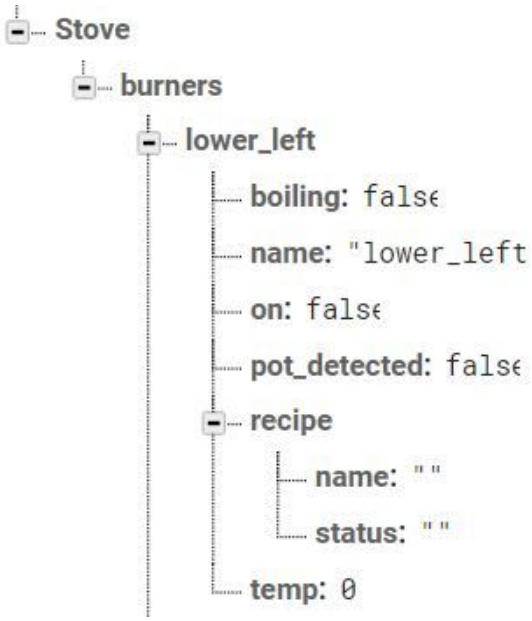


Figure 23: Firebase Node Example

The Hat

"The Hat" allows the user to see information embedded in the world around them. It also provides additional input to determine the user's intentions and proximity to the stove. The system we implemented runs on a Magic Leap One AR headset.

The Magic Leap One offers a robust set of input methods, including hand key pose detection, hand tracking, eye tracking, and more. It is untethered, so the user can move about the kitchen without being physically connected to a more powerful computer. At the time of writing, the Field of View (FOV) was better than the only potential replacement, the Microsoft Hololens (40° x 30° compared to 29° x 17°). This difference in FOV allows the user to see more of the system, particularly in their peripheral vision. Additionally, the Magic Leap One is more comfortable to wear, largely due to the weight distribution (all the computing happens in a puck that clips on the waist). This weight reduction may also improve depth perception by reducing the added inertia, as described by [10].

Unity Game Engine

Our application is built on top of the Unity game engine. Unity was chosen due to its compatibility with the Magic Leap One and ease of development compared to other methods (for example, Unity offers a graphical interface where objects in the scene can be configured).

First, we will disambiguate a number of Unity specific terms that will be used explain our architecture. The building blocks of a Unity program are GameObjects. GameObjects have a position, rotation, scale, as well as components. These components define the behavior or ability of GameObjects. For example, in a video game, an item that can be picked up might have a `PickupBehavior` component attached. All of the GameObjects are contained by the world, which in Unity is called the scene. Returning to the video game analogy, the scene contains all of the characters, buildings, mountains etc. Everything in the scene is what the user could possibly see if they looked/walked around, but not everything in a scene is visible at any given moment.

Every frame, all of the GameObjects are updated. This happens by calling an `Update()` function attached to each component. Each GameObject controls its own state, and is fairly isolated from other GameObjects.

Components and other scripts used by Unity are written in C#.

Software Architecture

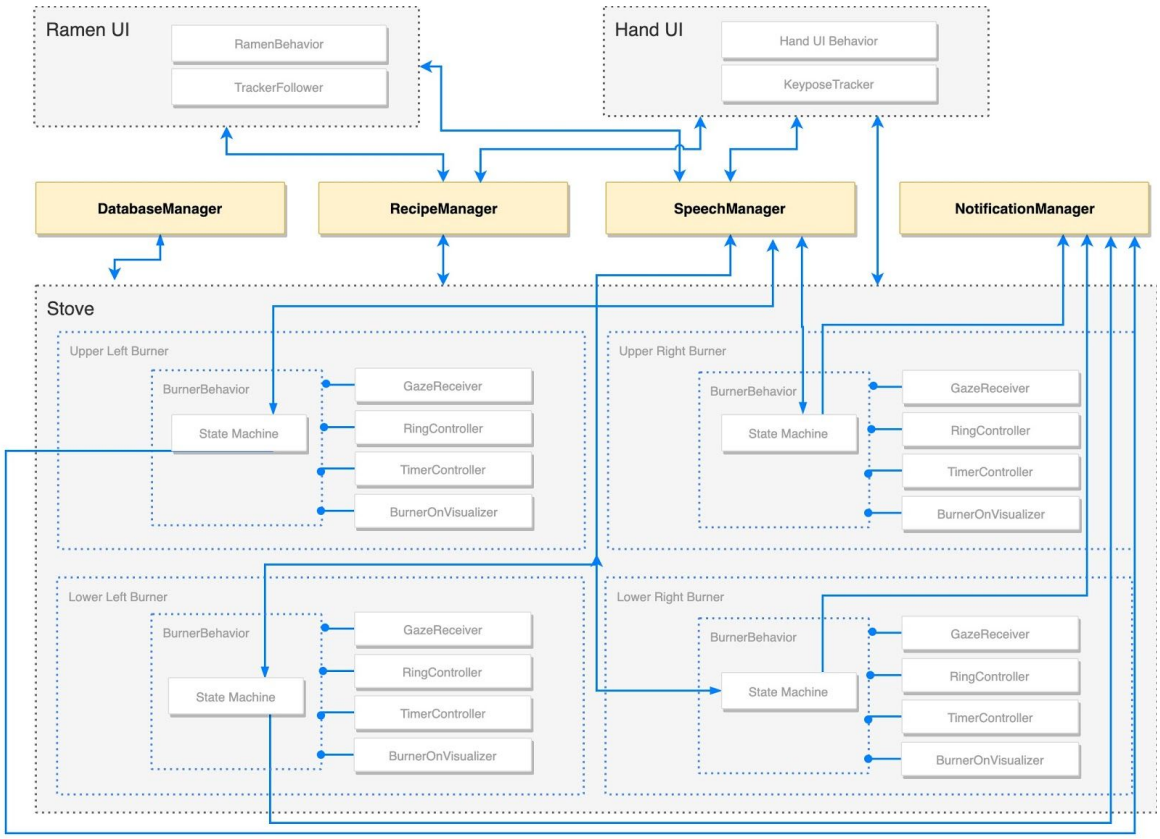


Figure 24: The Hat Overall System Architecture

Model

In a computer application, the "model" represents the underlying data behind the user interface. It is encapsulated from the rest of the system so the way

the information is displayed can be easily changed. The model for our system has two primary components: burners and recipes.

Burners

The Burner data class has a number of attributes, including the temperature, and whether or not a pot is detected, burner is on, or boiling is detected. All of these attributes are exposed as `ReactiveProperties` from the UniRx[2] library. This allows other classes to bind themselves to the data model and update themselves automatically when the data changes. This programming paradigm is known as "reactive programming." [22]

Recipes

Recipes contain the instructions for preparing various meals. Although our system currently only supports making pancakes and ramen, the Recipe system was written in a extensible way to easily support new recipes.

Recipes contains a queue of recipe steps (`RecipeStep`). Each recipe step has the following attributes:

- Instruction or Wait Explanation (string)

[2] UniRx is an open source implementation of the reactive programming paradigm for Unity. It is nearly identical to other "Rx" implementations such as JavaRx and SwiftRx. More information can be found here: https://github.com/neuecc/UniRx

    - Each recipe step has either an instruction or wait explanation. An instruction is something the user must do, and a wait explanation is for a step that requires something to happen (like a pan to preheat or water to boil).
    - If a wait explanation is provided, an indeterminate state will be displayed above the anchor.
- Anchor (Anchorable)
    - The `Anchorable` system will be explained in more detail under Key Components, but this is the location that the UI for the current step will be anchored to.
- Requires Burner (bool)
    - Whether or not this step requires a burner to proceed. For example, if the step is "Boil 2/3 Cup of Water," a pot of water must be placed on the stove to proceed
    - This tells the `RecipeManager` (explained in the "Managers" section) to consume the first unused burner with a pot placed on it.
- On Enter (lambda function)
    - Called when the step is first entered. Handles setup for the step.
- On Complete (lambda function)

- Called when the step is done. Cleans up the user interface in preparation for the step.

The recipe class has an `Update` function that is called by the `RecipeManager` every frame. This function checks whether or not the current step has been satisfied and, if so, proceeds to the next step. If a recipe is started in the middle of a user making something, it will automatically skip steps until it catches up to the user.

Managers

In our program, there are four "Manager" classes that orchestrate how GameObjects behave with one another. Those classes are `DatabaseManager`, `RecipeManager`, `SpeechManager` and `NotificationManager`.

DatabaseManager

`DatabaseManager` manages the connection with Firebase and updating the data layer. It is responsible for deserializing the json response from Firebase, and mapping the values to burner locations. Burner data objects are stored in an array within this class.

SpeechManager

`SpeechManager` handles the streaming speech recognition service. This includes controlling access, keeping track of whether or not the resource is in use, permissions, and initiating/closing requests.

RecipeManager

`RecipeManager` is a singleton responsible for starting and managing recipes. It is capable of handling multiple recipes at the same time. To start a recipe, any script can instantiate a recipe and pass it to `StartRecipe`. `RecipeManager` also handles assigning burners to recipes when needed, and releasing them when appropriate.

NotificationManager

`NotificationManager` handles sending notifications to the user and dismissing them when appropriate. It is implemented as queue since in the current implementation, only one notification can be viewed as a time. Safety critical notifications will always take priority and be displayed first.

State Machines

State machines are used throughout our program, from burners to notifications. Encapsulating functionality into states makes the code more flexible

and easier to maintain. For example, to add a new capability to the burner voice prompt, we would create a state for the said capability and add a trigger to the voice input state. This minimizes the number of classes that have to be modified. It also helps us handle transitions: individual states can detect that a transition is required and cleanup after themselves (for example, hiding unnecessary UI components).r

In our system, a state machine was implemented using a "State" class with an "Update" function. This "Update" function returns a State. This return value is either the current state or the next state. Components use this state machine by having an attribute of type state, which is assigned to an initial value. In the components update function, the update function of the state is called and the return value is assigned to the state attribute. If the state was changed, it will be updated on the next frame.

Burners

There is a GameObject in the scene for each of the four burners. The position (e.g. "Upper Left") is set for each burner in the editor. Each GameObject has a `BurnerBehaviour` class which controls the state and high level functionality for the burner. Additionally, each burner GameObject has a number of child GameObjects including a "`BurnerOnVisualizer,`" a timer, and an indicator

ring. This allows the UI functionality to be encapsulated. For example, if a burner has been left on, the `BurnerBehaviour` class will enable the `BurnerOnVisualizer`. The `BurnerOnVisualizer` class will handle pulsing the red disk displayed on top of the burner. Once it is turned off, the `BurnerOnVisualizer` will be disabled by the `BurnerBehaviour.`

BurnerBehaviour

As mentioned above, a `BurnerBehaviour` component is attached to each Burner. Each `BurnerBehaviour` has its own state machine. Burner states are broken up into four categories `InputStates`, `TimerStates`, `BoilingStates`, and `RecipeStates` (these are not states themselves, but packages for states). The state diagram below illustrates how and why the burner moves through its states.

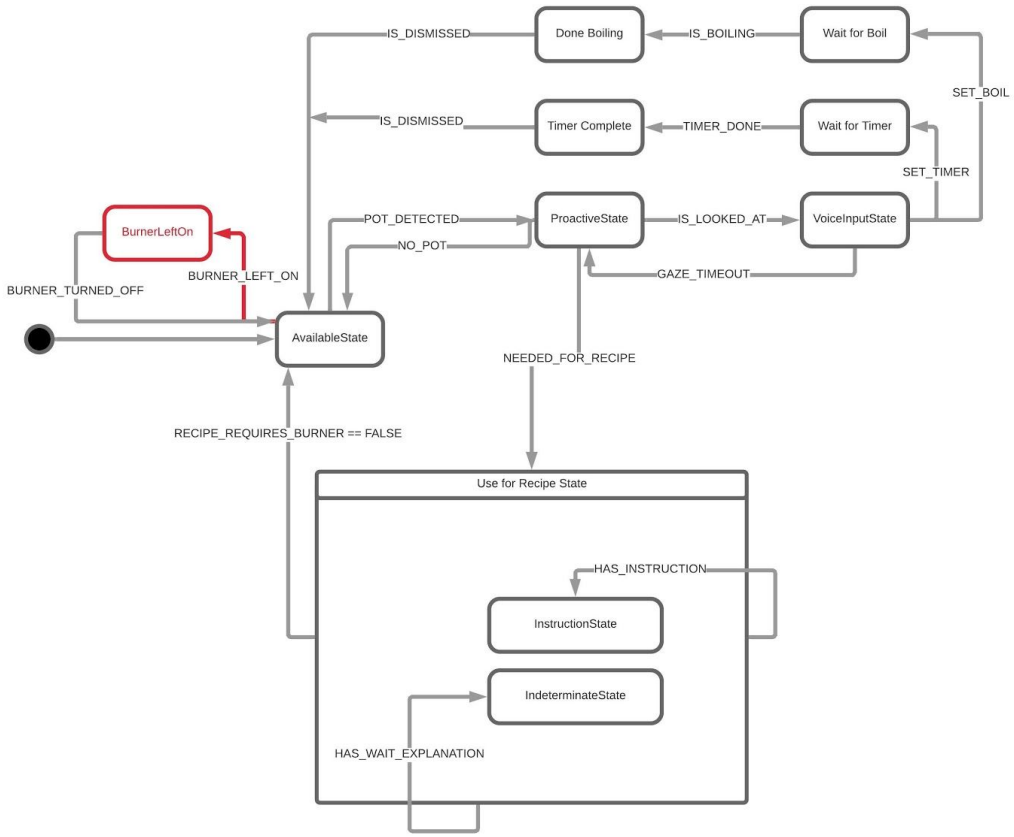


Figure 25: Burner State Diagram

The following are explanations of each state's behavior:

- `AvailableState`
    - When available, the burner is checking to see if a pot is added. It is also checking to see if it's been left on for more than 5 seconds, in which case it transitions to `BurnerLeftOn` state.

- `BurnerLeftOnState`
    - In this state, a glowing red disk is augmented directly on top of the burner (as seen previously in Figure 13). Once the burner has been turned off, the system returns to `AvailableState`
- `InputStates`
    - `ProactiveState`
        - When a pot is placed on the stove, and the burner is not needed for a recipe, the burner is transitioned to a proactive state. In this state, a white ring is displayed above the pot to indicate that it has been detected. If the user looks at the ring for more than ~ .3 seconds, the burner transitions to `VoiceInputState`
    - `VoiceInputState`
        - In this state, the ring transitions to be blue and wavy. Voice prompts are displayed and, as the user talks, the recognized text is displayed as well. This state also handles errors (in which case, the recognized text will flash red and clear) as well as determining how to process the input.
- `BoilStates`
    - `WaitForBoilState`

        - If the user asks the system to monitor boiling, this state will be initiated. The ring switches to a red indeterminate mode, as seen in Figure 6, and "Waiting to Boil" is displayed. Once the water boils, the system transitions to `DoneBoilingState`
    - `DoneBoilingState`
        - Once the water is done boiling, the ring transitions to pulsing green and "Done" is displayed. Once dismissed, the burner returns to `ProactiveState`
- `TimerStates`
    - `WaitForTimerState`
        - If a timer is set, this state will be initiated. The ring transitions to a timer and the time remaining is displayed. Once the desired time has elapsed, the system switches to `TimerDoneState.`
    - `TimerDoneState`
        - Once the water is done boiling, the ring transitions to pulsing green and "Done" is displayed. Once dismissed, the burner returns to available.
- `UseForRecipeState`
    - `UseForRecipeState` is entered once `RecipeManager` determines that the burner should be used for a recipe. It is the only state which

has a sub-state. It assigns this sub-state based on the current recipe step -- for example, if the current `RecipeStep` is waiting for water to boil, the sub-state will be the `WaitingToBoil` state.

BurnerMiniMirror

`BurnerMiniMirror` is an extension of the `BurnerBehaviour` class. It contains all of the same components and child GameObjects, but instead of updating its own state it copies the state of a target burner. This is used for the Mini Mirror display shown in Figure 9.

Anchoring System

One unique ability of our system is contextual anchoring: that is, the UI can anchor itself to physical contexts in the users environment. To enable this, we developed a unique anchoring system. The system has two primary components: `Anchorable` and anchor points.

Any context (represented as a GameObject) that can be anchored to (for example, a Ramen package or burner) extends from the `Anchorable` class. It also defines one or more transforms (scale, position, and rotation) the UI can anchor to -- what we call anchor points.

The `Anchorable` class has two functions: `IsInView()` and `GetBestAnchorPoint()`. `IsAvailable` returns whether or not the `Anchorable`

is visible to the user. For example, if the Ramen package has been placed on the counter or ripped open it is no longer visible. In this case, the `Anchorable` class will fall back to an anchor in the upper corner of the user's display. Next, `GetBestAnchorPoint` returns the best anchor point to anchor to. For example, the default anchor for a pot is in the middle, but if the user's hands are near the pot it moves to an alternative anchor point along the edge of the rim.

Improving Device Input

Although the Magic Leap has a number of different input methods, they are difficult to incorporate out of the box (due to instability, inaccuracies, or implementation specific details.) We developed a number of strategies for circumventing these deficits.

Gaze

One input source used is gaze tracking. Gaze tracking provides a 3D vector, with the camera at origin, that represents where the user's eyes are looking. We use this in order to understand where the user is directing their attention. For example, in Figure 5 (Left) a white ring is shown above a pot to indicate a detection, and this ring turns into a voice prompt when the user directs their attention towards it.

Although it is easy to detect which GameObject is being looked at in a given moment, this information alone is not very useful. Eyes tend to jump quickly between fixation points, in movements known as saccades [21]. These saccades happen extremely quickly, and users might not even notice them happening. In our system, we usually combine three measures: whether or not the GameObject is being looked at; if so how long it's been looked at, and if not how long it's been since it was looked at. The former is usually used to initiate an action (like showing a voice prompt) and the latter is usually used to exit an action (a timeout).

In order to more easily incorporate gaze tracking into our system, we created two new classes: `GazeCaster` and `GazeReceiver`. A single `GazeCaster` is added to any Unity Scene that uses eye-tracking. This class constantly gets the latest gaze direction and performs a Physics raycast into the environment. Lastly, any GameObject that wishes to somehow utilize raycasting adds a Physics Collider and our `GazeReceiver` component. The `GazeReceiver` component provides a variety of useful data including duration of current gaze and duration since last gaze.

Image Tracking

Another input source is image tracking. Magic Leap is capable of tracking image targets in the real world and providing the position and location. There are a number of problems with this tracking. For one, it can be extremely noisy even in

the best of circumstances. This noise can be extremely jarring, and potentially nauseating, for users. Additionally, optimal conditions are extremely hard to achieve. Lighting will worsen this problem, as will small image targets. The inaccuracies get worse as the image targets move further away from the headset. And if the image target is too close to the headset, the rendered content can become invisible.

To resolve this problem, we created the `TrackerFollower` class. The class is assigned an Magic Leap `ImageTracker` to follow and a tracking speed. `TrackerFollower` uses linear interpolation in order to smooth out the image tracking data and follow more smoothly. Different tracking speeds make sense for different applications. Something that is handheld should have a higher tracking speed than something expected to be stationary.

`TrackerFollower` automatically decreases following speed as the image tracker gets further away. This helps counteract the loss of accuracy. It can also, if enabled, ensure the image target is outside of the camera's clipping plane.
The last feature of `TrackerFollower` is the ability to automatically switch between multiple targets for the same object. For example, if you want to follow a Ramen package, the user could be looking at either the front or back of the package. `TrackerFollower` can be provided both and will track whatever is available/closest.

Hand Tracking

Magic Leap provides hand tracking, including a location of the user's palm and recognition of a small set of key-poses (like thumbs up, open hand, and fist). Like eye tracking, hand tracking can be extremely noisy and we often want to know more about how long the key-pose has been active for.

To work around this issue, we implemented `KeyPoseTracker`. `KeyPoseTracker` is provided a a key-pose to track. From that, it tracks a number of different data points:

- Hand velocity
- Smoothed (median + linear interpolation) hand position
- Key-pose duration
- Duration since key-pose
- Whether or not the hand is stationary
- Duration of stationary
- Duration since stationary

Adaptive Transparency Shader

The adaptive transparency shader makes it easier to read text when it is anchored to a burner. The figures below illustrate legibility without (left) and with (right) the adaptive transparency shader.

Figure 26: Text is obscured by the indicator ring (Left)

Figure 27: Adaptive transparency makes text more legible (Right)

This effect was achieved using what is known as a shader. A shader changes how a mesh is rendered by the camera. In this case, the shader, which is calculated for every pixel, does the following:

```
// this is psuedocode
// this function is run on every single pixel location being
// shaded on an objecte
void surf()
{
      Position meshCenter= GetCenterOfMesh();
      Vector directionUserIsFacing = meshCenter - CameraPosition;
      Position closestPointOnMesh = GetClosestPointOnMesh(meshCenter,
```

```
                                        Radius,
                                        directionUserIsFacing);

//the farther away this pixel is from being the closest in line with
the user's gaze the more transparent the pixel is
      float pixelAlpha  = 1 - abs(distance(closestPointOnMesh,
                                  ThisPixelsWorldLocation))
      SetTransparency((pixelAlpha)
}
```

Speech Recognition

Speech recognition is not provided by Magic Leap. There were a number of existing Speech Recognition packages that worked with Magic Leap, but these did not offer streaming. Streaming speech recognition predicts what the user is saying as they talk, rather than recording an audio clip and submitting it for recognition. For our project, we modified an Azure Speech Recognition [6] library for use with Magic Leap and other IL2CPP devices. This modified package has been open sourced and is available for download at the following link: http://github.com/bhylak/magicleap-streaming-stt

The Rat

Purpose

In order to successfully monitor the user's kitchen environment both when the user is in the kitchen or in another room, we needed a device that would always be in the kitchen continuously monitoring the environment. The device is meant to provide information about the user's environment in the kitchen to determine actions taken and actions that need to be taken.

Before building the Rat, we defined a set of requirements that the Rat has to meet. The requirements are summarized in the following table.

Requirement	Reasons
The Rat needs to consistently monitor the kitchen	Maintain safety since the user can leave the burner on any time of the day; Decrease the need for explicit input
All operations need to run locally on the Rat	Maintain the privacy of the user
The Rat should be mounted securely above the stove	To get a good view of the stove top and surroundings; Not in the user's way during cooking
Low latency	To detect user's actions in real time so the user is not left wondering whether or not explicit input is needed

Hardware

To meet the above criteria, we determined that the following hardware modules are needed:

- A microcontroller with high processing power
- An RGB camera to detect objects placed on the stove and status of recipes.
- A thermal camera to detect which burners are on.

A block diagram of the different parts can be seen in Figure 28. The Coral USB accelerator was added later on in the design process to speed up machine learning inferencing. The following is a picture of the electronic assembly.

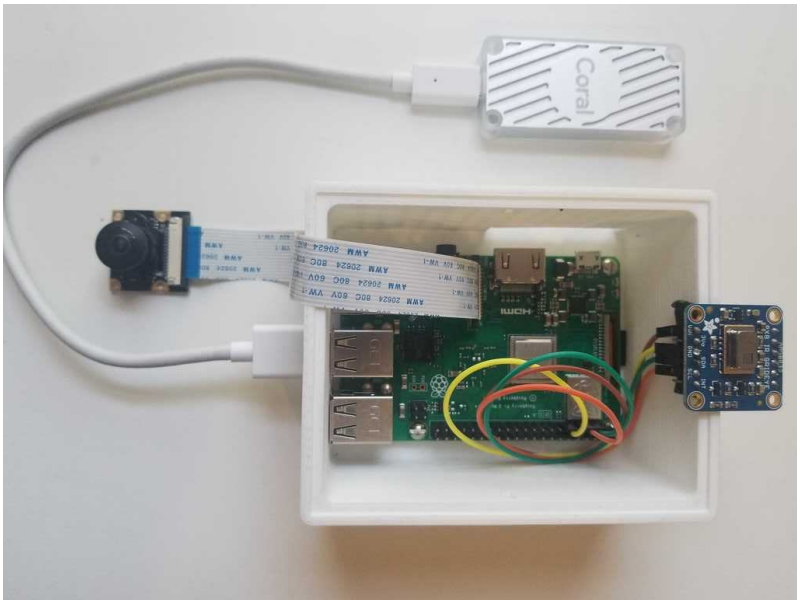


Figure 28: Electronic Assembly

Raspberry Pi 3 B+

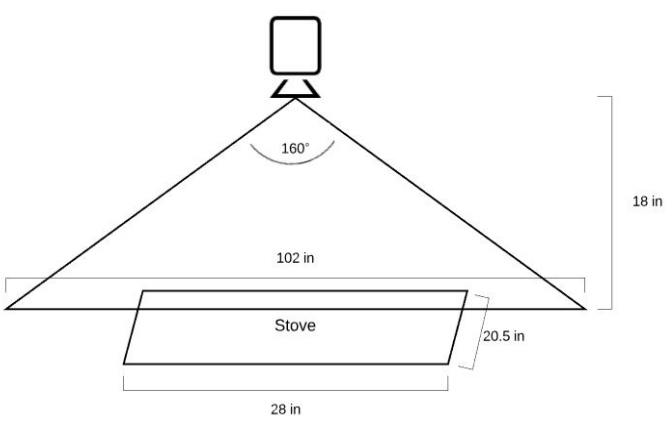
For the microcontroller, we decided to use a Raspberry Pi, which is a single-board computer used widely for prototyping purposes. We used the Raspberry Pi model 3B+ for its high processing power, affordability, and prototyping capabilities.

The high processing power of the Raspberry Pi was a priority since it would be used to run multiple different operations at the same time. These operations include the use of thermal detection, computer vision, and machine learning. All of these processes require high processing devices especially when run at the same time. However, for machine learning applications, the high processing power of the Raspberry Pi was still not sufficient. Therefore, we used a google coral accelerator to run machine learning processes. The coral accelerator will be discussed in more detail later in this chapter.

Fisheye Camera

The SainSmart Wide Angle Fish-Eye is a Raspberry Pi compatible camera that has a 160° viewing angle. The reason we used a fisheye camera was due to the narrow field of view of a regular Raspberry Pi camera, which has a horizontal field of view of 62.2° and a vertical field of view of 48.8°.

Due to the fact that the camera needed to capture a stove top with the dimensions 28” x 20.5” and the device is mounted at a height of 18”, we needed to use a fisheye camera since our calculations showed that the field of view of the regular camera would not be sufficient. This is shown in the following figures of the horizontal fields of view of each of the cameras.



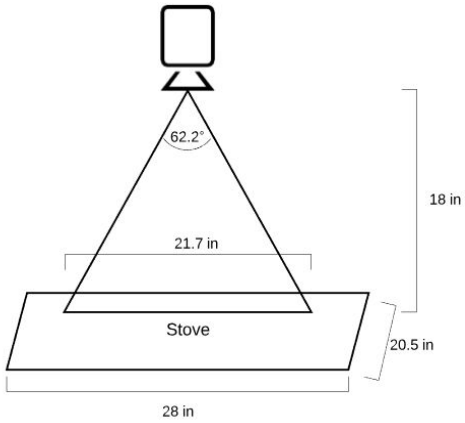


Figure 29: Field of View of Fisheye Camera (Left)

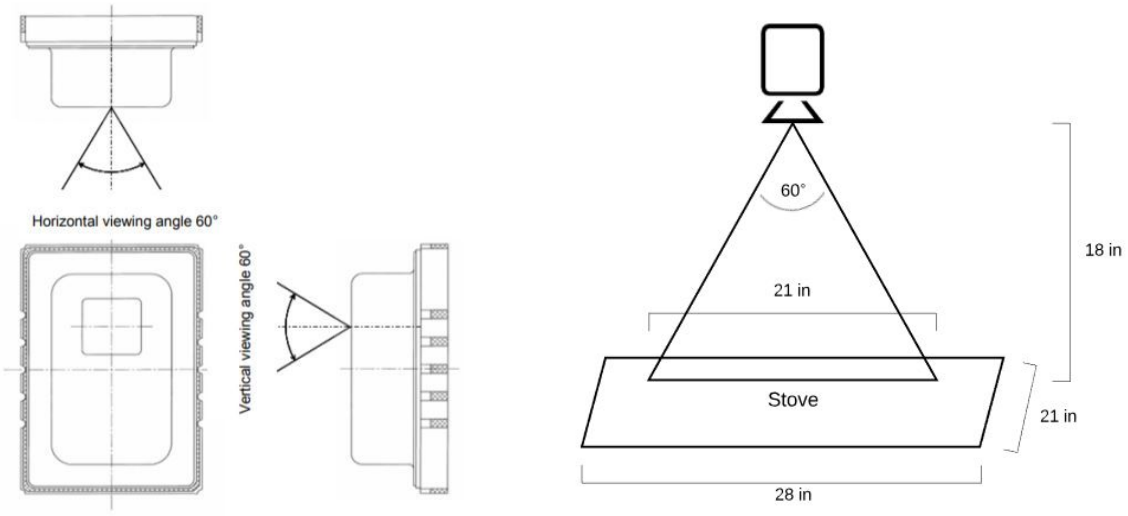
Figure 30: The Regular Camera (Right)

Thermal Camera

We decided to use the Adafruit AMG8833 IR Thermal Camera for our thermal detection purposes due to its affordability and compatibility with the Raspberry Pi. The main limitation of the thermal camera, however, is the temperature range of 0°C to 80°C. For the purposes of our prototype, the

temperature change that the thermal camera is able to detect can still allow us to detect when a burner is turned on and when a burner is left on.

Although the field of view of the thermal camera does not cover the whole stove, it covers enough of the burners to be able to get their temperatures. The camera has a viewing angle of 60**º**.



Appendix C

Figure 31: Field of View of Thermal Camera from datasheet

Figure 32: Field of View of Thermal Camera on top of Stove

Coral Edge TPU Accelerator

Google Edge TPU Coral accelerator was added later on in the design process to speed up machine learning processes run on the Raspberry Pi. The accelerator

uses Google Edge TPU to run machine learning inferences. Edge TPU is an ASIC (Application Specific Integrated Circuit) designed to provide high performance machine learning inferencing for Tensorflow Lite models. The Coral accelerator is designed for any Linux device with a USB port and is therefore compatible with the Raspberry Pi. The Coral team recommends using a USB 3 port to get the best inference speeds that can reach 100 FPS. The accelerator is still compatible, however, with the USB 2 port of the Raspberry Pi.

Construction

Electronic

The following block diagram shows the connections made to the Raspberry Pi in order to integrate the various parts of the system. The integration between all the hardware pieces included three connections, the thermal camera to the Raspberry Pi, the fisheye camera to the Raspberry Pi, and the Coral Accelerator to the Raspberry Pi.

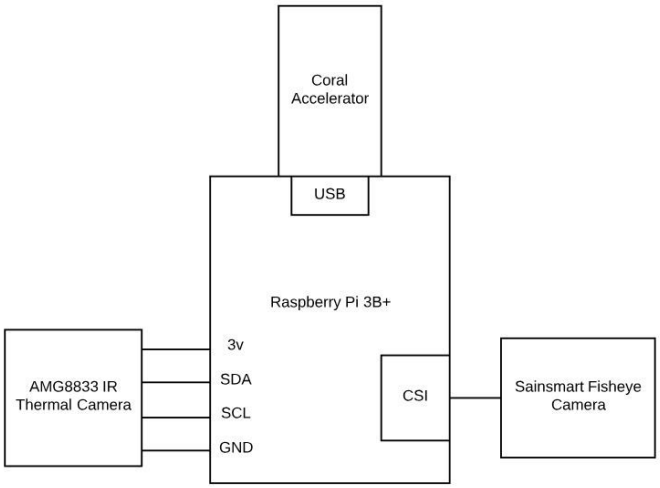


Figure 33: Block Diagram of Hardware Components

The Coral Accelerator can simply be plugged into a USB port on the Raspberry Pi and does not require any additional connections. Similarly, the Fisheye camera uses a CSI-2 (Camera Serial Interface), which is the most widely used camera interface in the mobile industry, and can be connected to the CSI port on the Raspberry Pi. The thermal camera connection, however, is slightly more complex and is discussed in more detail below.

As seen in the block diagram, the connection between the thermal camera and the Raspberry Pi uses four pins on the thermal camera. These pins are split into power pins and logic pins as follow:

Power Pins:

- 3Vo, which is a 3.3V output from the onboard voltage regulator which converts the voltage input from 3-5V to a safe voltage.
- GND which is the common ground pin for power and logic pins.

Logic Pins:

- SCL, which is the I^2C clock pin that connects to the SCL pin on the Raspberry Pi. This pin includes a 10k pull up resistor needed for I^2C connections.
- SDA, which is the I^2C data pin and it connects to the SDA pin on the Raspberry Pi. This pin also includes a 10k pull up resistor.

I^2C is a serial protocol for two wire interface that connects low speed devices like microcontrollers to similar peripherals in embedded systems. I^2C bus allows for connecting almost unlimited number of I^2C devices using only two wires that include pull up resistors. Each I^2C slave has a 7-bit address that is unique to each bus. The microcontroller generates the clock using the SCL pin and gets the data through SDA.

Enclosure

We needed to design an enclosure for our electronic parts both to protect them and be able to mount them directly over the stove. Before starting the design, we had the following criteria:

- Both the thermal camera and the fisheye camera should get a view of all four burners on the stove.
- The enclosure needs to be compact
- The enclosure needs to have holes for mounting the Raspberry Pi
- Power input, USB ports, and the HDMI port needed to be exposed
- The horizontal position of the enclosure needs to be adjustable in case we need to move it to a different stove
- The whole design needs to be securely mounted to the backsplash

The 3D design of the enclosure that includes the Raspberry Pi, the thermal camera, and the fisheye camera can be seen in the figures below.

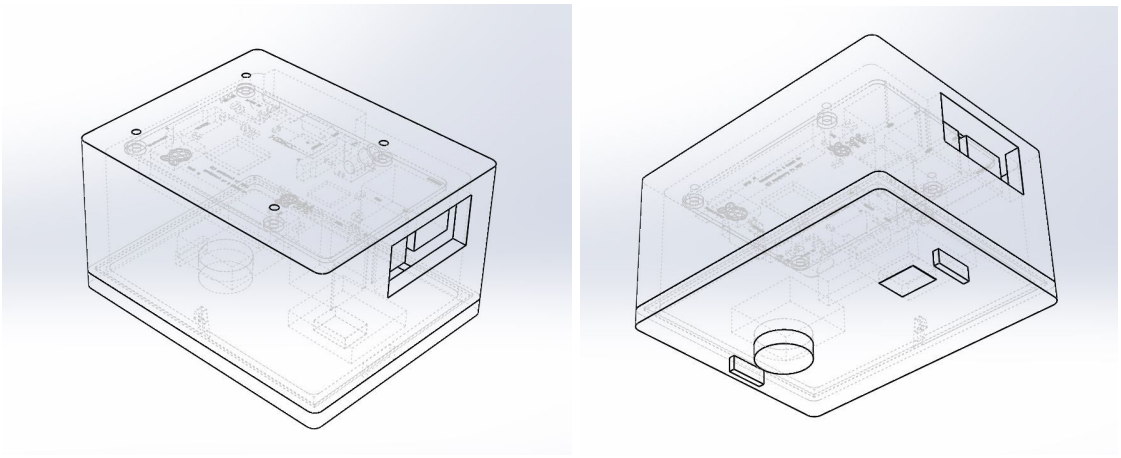


Figure 34: Enclosure 3D design (Top view)

Figure 35: Enclosure 3D design (Bottom View)

The enclosure rests on a part that can slide over two Aluminum rods attached to a back piece that gets mounted to the backsplash. The back piece is designed to be bulky and larger than the other components in the design so that it can support the weight of all the other components. The complete design can be seen in the figure below.

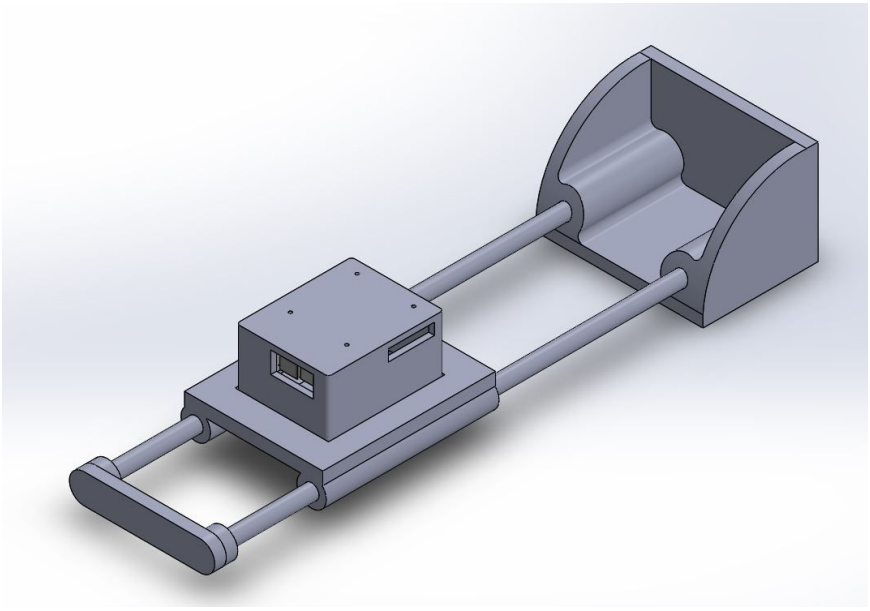


Figure 36: Enclosure and Wall Anchor 3D Design

All parts of the design were 3D printed, assembled and mounted over the stove. The following is a picture of the design after being mounted on top of the sove.

Figure 37: The Rat Mounted over The Stove

Software

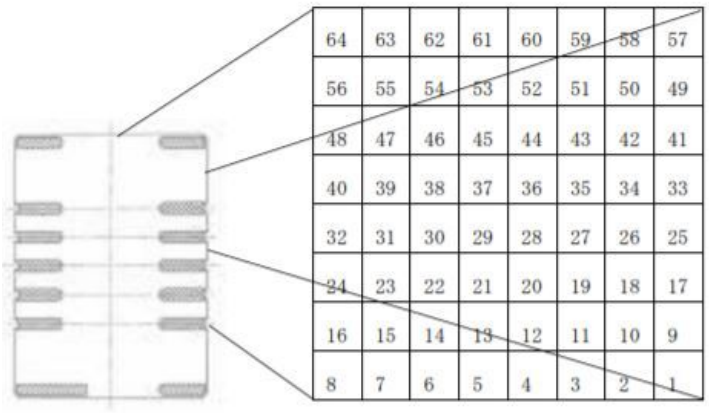
The Rat provides information about the user's environment and actions. Therefore, the images provided by the thermal camera and the fisheye camera need to be processed to collect useful information about the kitchen environment. This information is collected through processes that run on the Rat, including temperature detection, burner on detection, pot detection and food state detection. These are discussed in more detail below.

The software processes were designed with the following assumptions about the stove environment:

- The stove is electric with four burners. Although the processes might work for gas stoves and induction cooktops, this has not been tested.
- The area of the stove top is not larger than 21" x 28".
- The Rat is mounted a minimum of 18" above the stove so the thermal camera has a view of all four burners

Temperature Detection

One of the main things that we need the device to do is detection of temperature on different burners to detect whether or not a burner is on. The output image from the thermal camera is an 8x8 pixel array as shown in Figure 38. This image alone is not enough to get useful information about each burner.



Appendix C

Figure 38: Thermal Camera Raw Output

Before processing the image to get burner temperatures, we needed to refine the visualization from the thermal camera to a higher resolution. We used interpolation, which is a mathematical method used to construct data points between discrete points of data to get a refined measurement. After interpolation, we get a 32x32 pixel image. The image from the thermal camera of all burners of the stove being on is shown below. Red represents the highest temperature while blue represents the lowest temperature.

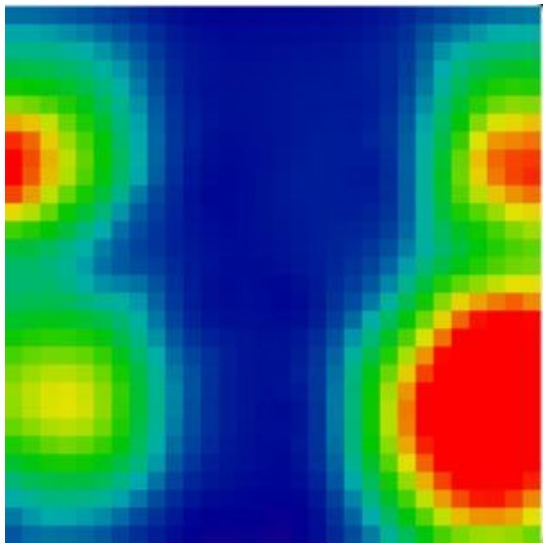


Figure 39: Thermal Camera Interpolated Image

Our initial method of getting temperature of burners was to split up the image from the thermal camera into four quarters representing each burner and take the average of the pixel readings in each quarter. That method proved to be inaccurate, however, due to the fact that it takes into account the cooler parts of the stove and affects the temperature readings.

To detect the burner temperatures more accurately, we apply computer vision methods using the OpenCV library [20]. OpenCV is an open source library developed for real-time computer vision and image processing applications. The process applied to the thermal camera image to detect the temperature of each burner is summarized in the following flowchart.

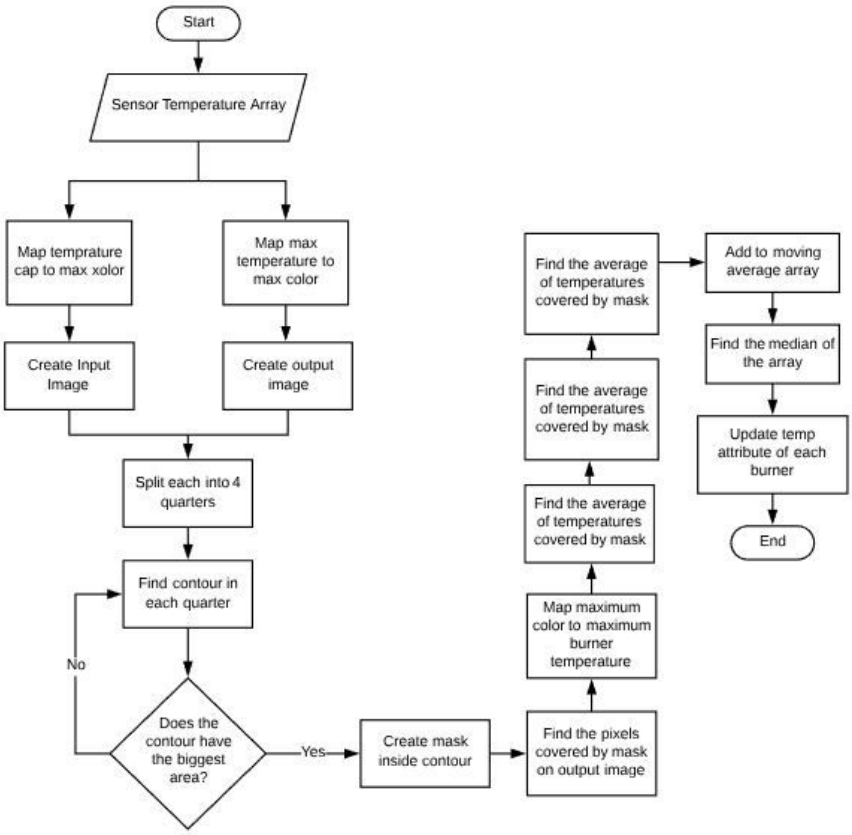


Figure 40: Thermal Processing Flowchart

The following figure shows the processed images of each quarter of the stove with the mask applied to the each burner area.

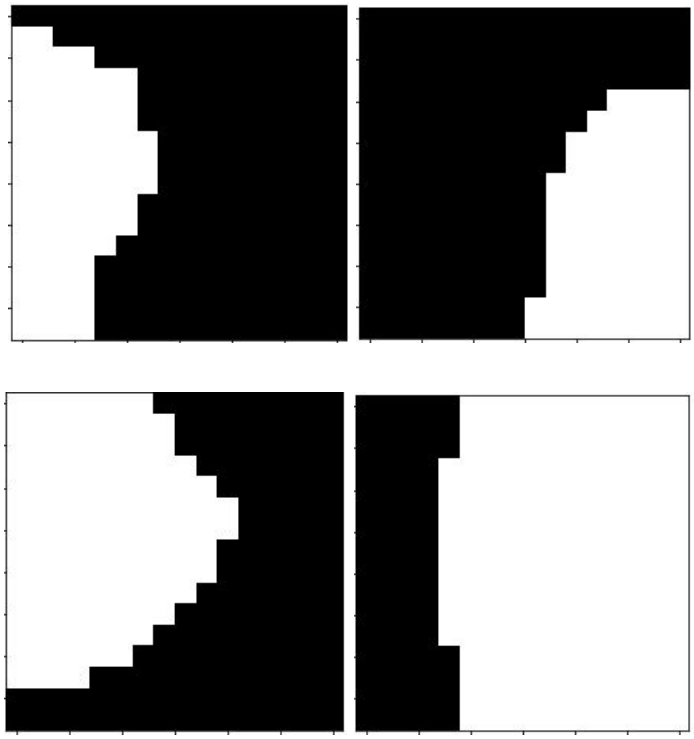


Figure 41: Thermal Processing Output Images

Burner On Detection

Besides getting the temperature of each burner, the Rat also needs to determine if a burner is on. One approach would be to check whether or not the temperature is higher than a certain threshold. But, this approach does not account for a burner that has just recently been turned off. A hot burner can take several minutes to cool down after being turned off.

Therefore, we instead collect temperature differentials over a time window to check for decreasing trends. In order to account for temporary temperature spikes caused by the user's hands or other burners, we take the median of a set of temperature values over a time window before checking for decreasing temperatures.

In order to constantly process a collection of data points both for getting a median of temperatures and getting temperature differentials, we used a moving average array. A moving average is used to store a maximum number of data points in an array over time. Once the array reaches its maximum size, data points that were stored earlier get pushed out of the beginning of the array and a new data point is added to the end of the array.

The following diagram sums up the burner on detection process.

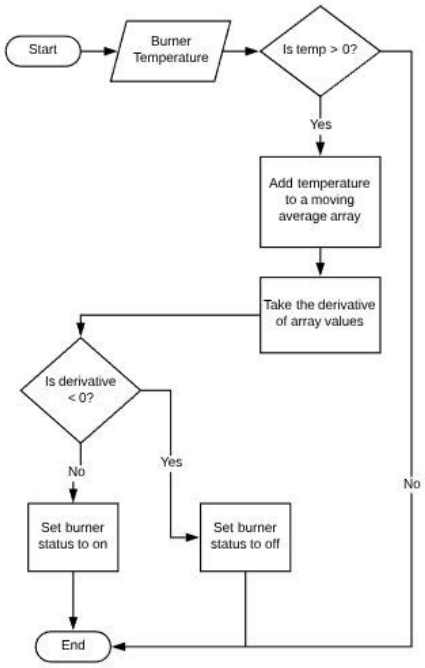


Figure 42: Burner On Detection Flowchart

Pot Detection

Pot detection is an important form of behavioral input. For example,when the user is making Ramen, the device can start detecting whether or not water is boiling based on the presence of a pot. It is also important to know which burner is being used so that UI components, like instructions, can anchor itself to the correct burner.

The first step in the process was to collect a set of pot images using the Fisheye camera on the Rat. We collected 250 images of different pots with varied

lighting conditions, positions, orientations, and combinations. Next, the location of the pot in each image pictures need to be recorded in a process known as labeling. In this process, a bounding box is drawn around each pot. We used an open source software called LabelImage [31] for this task. The following figure shows the interface we used and an example image.

Figure 43: Pot Detection Labeling Example

Next, we trained a machine learning model using the Tensorflow library. Tensorflow is widely used for machine learning applications because it makes the process of training and running machine learning models easier. However, training Tensorflow machine learning models from scratch can take millions of images and is very time consuming. Therefore, we used a technique known as transfer learning to train the model more quickly and accurately. Transfer learning is a method that allows for general purpose models to be retrained for a more specific

task [5]. In this case, we re-trained a Mobilenet V2 [30] Tensorflow model to detect pots.

Since training a machine learning model takes a long time on devices with low processing power, we used Google Cloud TPU to train our model which accelerated the training process significantly to take around 30 minutes [24].

After the training finishes an inference graph is created. The inference graph can then be used by the Tensorflow library to detect objects and their locations in images. One of the test images used to verify the success of the trained model is shown below. The model detected a pot with a 90% probability.

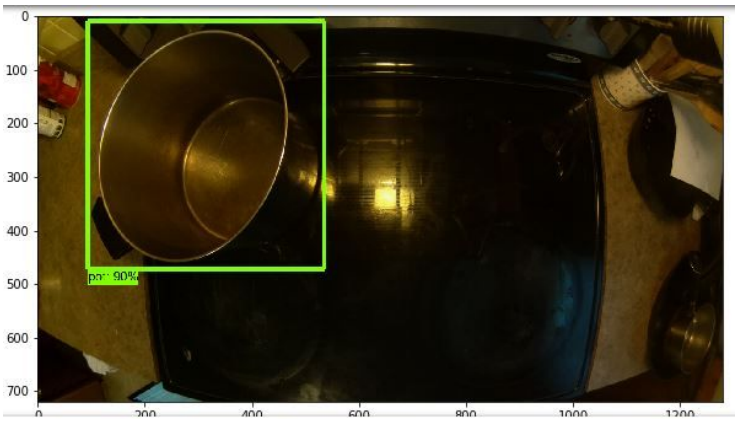


Figure 44: Pot Detection Example

Using a Tensorflow model for real time applications is inefficient since the processing time it takes is 0.067 FPS (Frames Per Second). However, Tensorflow offers a lighter version of a Tensorflow model that is called a TFLite model (Tensorflow Lite), with the disadvantage of decreasing accuracy. After converting the pot detection graph to a tflite graph, the speed increased to 0.2 FPS. However, that was still slow considering that UI elements need to be updated in real time depending on the results of the tflite detection. Therefore, we decided to use Coral Accelerator to run the pot detection model.

To use the coral accelerator with the pot detection model, all we needed to do was use the TPU object detection script included in the Edge TPU package with the TFLite model that we had created earlier. When tested with our pot detection model, the coral accelerator increased the framerate 26x from 0.2FPS to 5 FPS. Next, to detect which burner the pot is placed on, we used the detection boxes generated by the Edge TPU object detection class. The detection box around a pot is returned in an array in the following format [x_min, y_min, x_max, y_max]. The following flowchart shows the process of detecting which burner the pot is placed on.

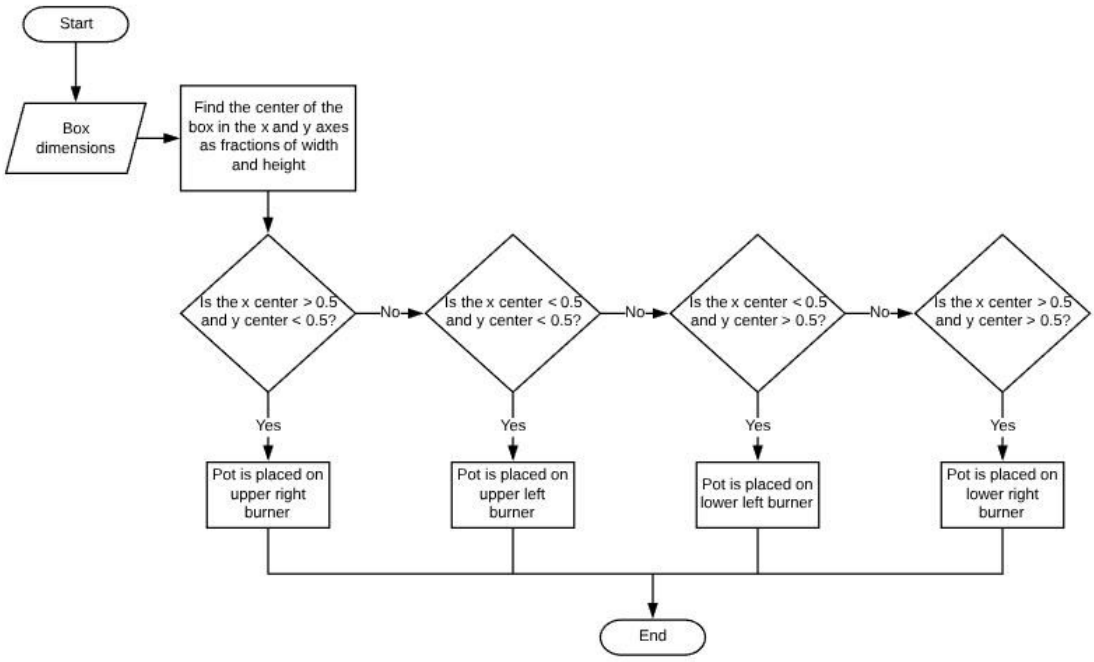


Figure 45: Pot Detection Flowchart

Food State Detection

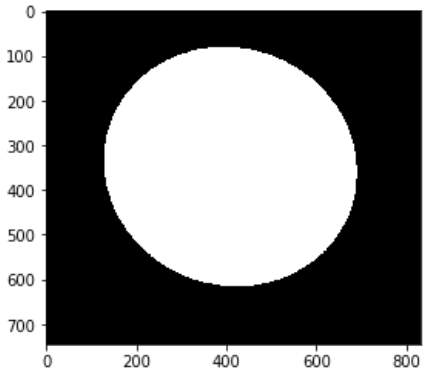
To demonstrate how the Rat can be used to detect different states of food, we implemented two different use cases boiling detection and pancake detection. The device can detect when water starts to boil and when a pancake is poured, ready to flip, or has been flipped.

Boiling

Boiling water is an essential part of many recipes including making Ramen, which we will be using to test the system. Therefore, it is important for the Rat to be able to detect when water starts boiling. Since boiling is a very dynamic process, machine learning won't be effective in this case. We used image differencing techniques provided by OpenCv and scikit-image [25] libraries to detect boiling. The main technique we used to find the differences between images is the Structural Similarity Index (SSIM) [29]. SSIM provided a concrete measure of similarity between two given images based on a range from -1 which indicates completely different images to 1 which indicates identical images.

To detect boiling, the main difference that need to be taken into account between consecutive frames is mainly the bubbles that arise in water when it starts boiling. Therefore, areas outside of the pot such as the stove top and the counters would make SSIM calculations less accurate since they contribute to adding unnecessary noise in the image. Therefore, the image is first cropped to only include the stove top area. The image is then split into four quadrants corresponding to each burner. Even after splitting the image to only include the pot area on each burner, there was still some noise around the pot that could affect the SSIM calculations.

To remove that noise, the image is further modified to include the area closest to the rim of the pot. That is done using OpenCv to find the largest elliptical contour in the image. The ellipse is then drawn onto a black mask to create a new mask. When this mask applied to the original image, it outputs the pot region of the burner. The following figures show the masking process.



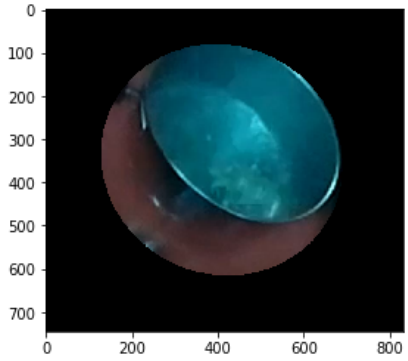


Figure 46: Boiling Detection Masked (Left)

Figure 47: Boiling Detection Mask Applied to Image (Right)

After this mask is applied, the image is used in an SSIM comparison process. In this process, every frame recorded by the Rat is compared to the frame previous to it. The SSIM comparison function returns a difference score and an image highlighting the differences between the two frames. For visualization purposes, the following figures shows the difference image, the image converted to

grayscale, and bounding boxes drawn using OpenCv to show the areas of difference between the two frames.

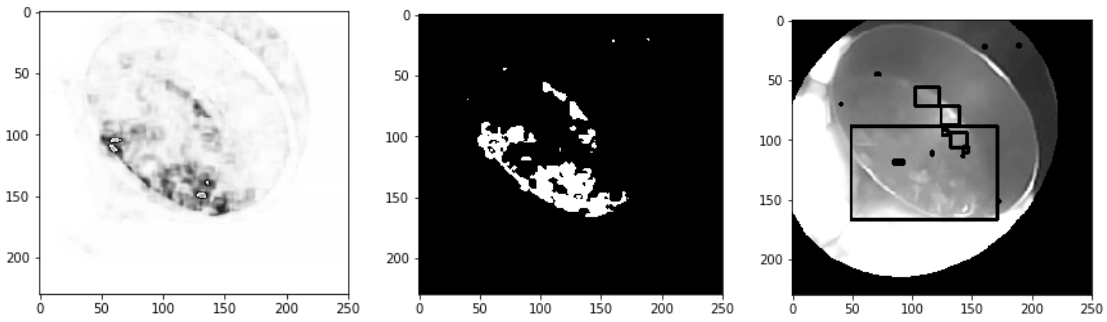


Figure 48: Boiling Difference, Threshold, and Bounded Rectangle Images (Grayscale)

Although every SSIM is stored, we do not take every number into account to determine if boiling occurs. That is because the SSIM might drop for reasons other than water boiling, such as someone passing their hand over the pot or the pot being slightly moved around. To avoid this issue, the median of five SSIM readings is stored into a list. We then take the average and use it to determine whether or not water is boiling. If the average SSIM drops under a certain threshold representing a larger difference between the frames, a boiling flag is set to true. If thirty seconds pass where that flag is set to true, we check the temperature of the water to see if it is in the boiling temperature range. If the temperature is in the boiling range, then the boiling status of the burner is set to true.

Pancakes

It is important for the Rat to be able to monitor the progress of a recipe. That means that it should be able to recognize all the different states of a recipe once it's placed on the stove. In our case, we used making pancakes as an example of a recipe that can be monitored using the rat. The different states of a pancake are: not placed in pan, placed in pan (not ready to flip), ready to flip, and flipped.

We used a retrained image classification model run on the Coral EdgeTPU in order to detect the different pancake states. The data collected for the training process consisted of pictures of pancakes that we made. We made around 30 pancakes and took 70 images per state with the Rat's Fisheye camera (a total of 280 images). We used data augmentation in order to generate more images for the training process. Data augmentation is an automatic process used to increase the size of datasets using for machine learning purposes [9]. We performed random operations on the images that include rotating, flipping (horizontally and vertically), adding noise and blurring. We increased the size of our dataset to 420 images using image augmentation. The following figure shows a sample of our dataset for all 4 states.

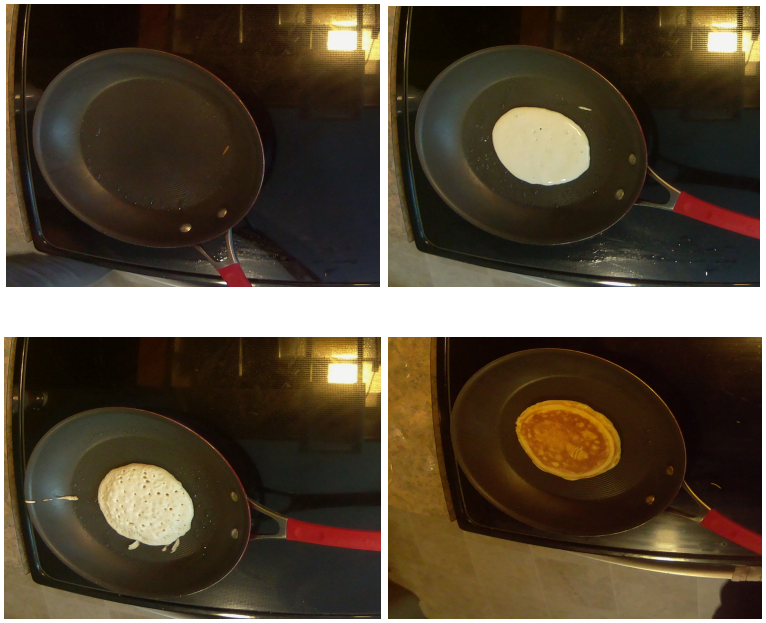


Figure 49: Pancake States

Using the same training technique we used for pot detection to train a pancake state detection model resulted in a low accuracy Edge TPU TFLite model. Therefore, we used Google Cloud AutoML [3]. AutoML is designed to train machine learning models on the cloud. AutoML enabled us to optimize training for use with Edge TPU and resulted in a highly accurate model.

6. Evaluation

After the system was complete, the evaluation for the system was divided into three sections: system usability, accuracy, and awareness. By separating these evaluations, we were able to explore each area in more depth.

System Usability

To test the system's usability, we conducted a study in a simulated kitchen environment. The study was performed to evaluate the merits of detached monitoring, particularly in its ability to reduce the need for physical presence while cooking and adapt to the user's context without explicit input. Additionally, we wanted to discover how usable our AR application was in a real world setting (because there is very little research in this area, the challenges are largely unknown).

Recruitment

For the experiment, we searched for participants who met the following criteria:

- Over the age of 18
- Had familiarity with cooking pancakes and ramen
- No known history of seizures/photosensitive epilepsy

To find participants, we posted on school social media pages, posted flyers around campus and contacted relevant student groups and classes. We recruited a total of 7 participants, ranging from ages 18 to 30. Two of the participants identified as females, and the remaining five identified as male.

Safety Considerations

We took a number of steps to ensure the safety of our participants. First, we created a mock kitchen in a lieu of a real, operable kitchen. This is because the effects of our system and AR in general on depth perception were largely unknown. There was a non-trivial risk that the system could affect user's ability to perceive depth, or reduce awareness of their surroundings, either of which could result in a burn or other injury. We monitored users' actions throughout the experiment to look for errors, like misjudging the distance to a pot, that could cause complications when using the system in a real kitchen.

Additionally, users completed a Simulator Sickness Survey before and after using our system. The Simulator Sickness Survey has users self-identify their level of discomfort across a wide range of symptoms, like burping, eye-strain and dizziness. The survey had two primary uses. First, it allowed us to filter users that had symptoms which could be exacerbated by AR. Users that had more than one

mild symptom in any category would not be allowed to participate. Second, it allowed us to track whether or not our system was causing Simulator Sickness.

Setup

First, we created the mock kitchen in our lab. The mock kitchen had a printed stove with switches to turn the burners on and off, a sink area with a pitcher of water, measuring cups, pans and other required materials. It also had plastic vegetables, and real packages of Ramen.

Figure 50: Simulated Kitchen Setup

Although the Rat was mounted above the stove for added realism, due to the mock kitchen setting, all of the responses from the Rat were simulated. (For example, since our fake stove could not actually boil water, another researcher would simulate the signal that is sent to the Hat). We created a GUI using Python to easily simulate the responses from the Rat, which can be seen below.

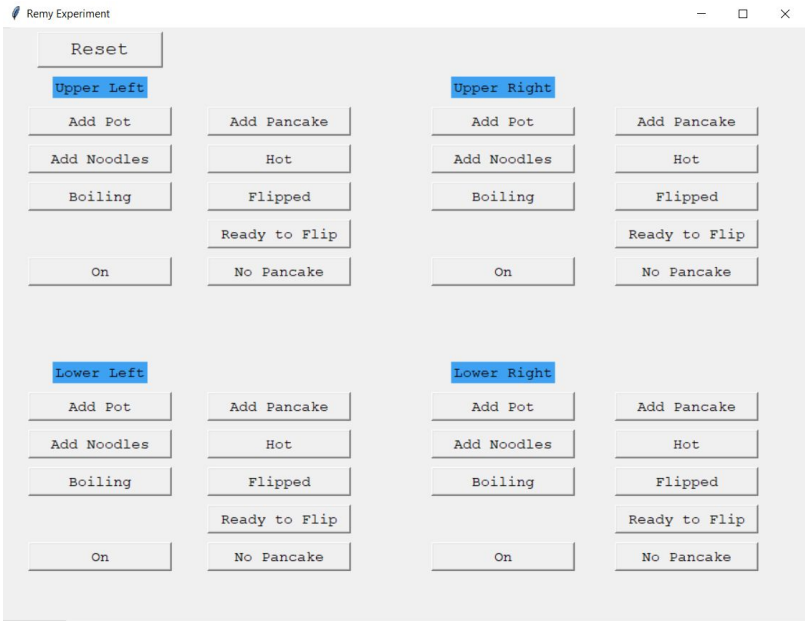


Figure 51: Experiment GUI

Dependent Measures

In our evaluation, we used a variety of surveying techniques, along with observation. The first surveying method was the Simulator Sickness Questionnaire, as mentioned previously.

The following two surveying techniques were chosen based on research provided by Laubheimer [12]. The first is a two-part post-task survey, administered after every task. Laubheimer's experience in the field indicated that lengthy surveys after each task tired users out, and most questions were often redundant. Instead, they suggest asking a Single Ease Question, or SEQ. The SEQ asks the user to rate the task from Very Easy to Very Difficult on a 7 point scale, which we have them complete on a web survey. After the SEQ, Laubheimer recommends asking the user to verbally explain why they scored the task accordingly.

After all tasks have been completed, we then administered the System Usability Survey (SUS). NNG recommends this survey at the culmination of a user study, in order to understand how the participants perceive the system as a whole. The system usability survey has 10 questions, each ranging from Strongly Disagree (1) to Strongly Agree (7).

After the SUS, we also collect demographic data like age, gender, and experience level with AR/VR and cooking. Finally, we had a short, semi-structured interview to provide an outlet for any uncommunicated feedback. A list of all the surveys used in the experiment can be found in Appendix A.

As mentioned, we also relied on observation throughout the experiment. Participants were instructed to think out loud as they completed the tasks. Each

participant was video recorded so we could further analyze their behavior post-experiment.

Procedure

We began with explaining the experiment, including potential risks, and reviewing the consent form with participants. Once the consent form was signed, we introduced them to the Magic Leap. We first showed them the device, and explained some of the capabilities. Afterwards, participants went through fitting and visual calibration, using built in Magic Leap utilities. This calibration improves eye tracking and overall visual quality, and also familiarizes participants with using an AR headset. Participants were then walked over to the mock kitchen and introduced to the various tools and components.

Next, participants were asked to complete three tasks: reheating vegetables, making ramen and, lastly, making pancakes. One of the most important aspects of our evaluation is that we did not tell users how to complete the task. This is in contrast to the evaluation for CounterIntelligence where users had to follow a predetermined set of steps like "Put one egg into a small pot & fill the pot with enough HOT water to cover the egg [and then] bring the water to a simmer & let simmer for 3 min. "

Instead, we gave participants a goal: make ramen or make pancakes, for example. The only additional information we provided for the task was related to constraints of our simulated environment (for example, we told participants to pretend like there was real pancake mix in the bowl.) Our goal was to understand how people naturally complete common tasks in the kitchen.

When a task had a long wait, the experimenter would sit down at two chairs located about two meters from the stove. This was to encourage participants to leave the stove, although they were not required to. If participants did not leave the stove naturally, we would ask them why. We instructed participants to keep the Magic Leap in-between tasks, unless they were experiencing discomfort (at which point, we would have ended the experiment)

As mentioned, after each task participants were asked to fill out a Single Ease of Use Question (SEQ). We then proceeded with asking the participant how they decided what score to give the task.

Once all three tasks were completed, participants removed the headset and provided final feedback.

Survey Results

The last participant (7) in our study encountered a number of device related technical problems, which resulted in the inability to complete the first and second

task. For this reason, we discarded their responses to the System Usability Survey and the Single Ease of Use Question. Their responses to the Simulator Sickness Survey, as well as observations gleaned from the trial, will still be used.

No Signs of Simulator Sickness

After an hour of using our system, the majority of participants did not show any increase in simulator sickness, as reported by the Simulator Sickness Survey. Two participants did, however, report a small increase (from "None" to "Slight") in eyestrain. Additionally, one participant reported a decrease in two symptoms: fullness of head and stomach awareness. Although this only occured for one participant, it is nonetheless interesting to note that our system did not worsen their existing symptoms. Although it seems that the Magic Leap headset did cause eye strain after an hour, the lack of other symptoms associated with simulator sickness is a positive sign.

Additionally, participants did not show signs of impaired depth perception nor impaired environmental awareness. None of the participants accidentally touched a hot surface, or made a mistake that could be attributed to the effects of wearing an AR headset. In other words, had the experiment taken place in a real kitchen, none of the participants in this study would have injured themselves.

System Usability Results

Five out of the six participants found the system easy to use and reported that they would like to use the system frequently. Four participants reported that they would not need help from a technical person to use the system. This, of course, means that two participants said they would need help. Those two participants were the first participants to try out the system. They encountered errors, mostly related to voice input, that were fixed directly after their participation. One out of the six participants reported that they did not feel very confident using the system. That participant also encountered technical issues that required our intervention. A summary of the answers to each of the System Usability Scale questions can be found in Appendix E.

Single Ease Question (SEQ)

Responses to Single Ease Questions are summarized in the following chart. The difficulty levels that could be assigned to each task varied from 1 being the easiest to 7 being the most difficult. None of the participants assigned a difficulty level higher than 5 to any of the tasks. Participants reported that making pancakes was the easiest task, while setting a timer was the most difficult.

Figure 52: Responses to Single Ease Questions

Discussion

Detached monitoring was intuitive

One of the clearest findings from this experiment is that participants found detached monitoring to be intuitive. In fact, the tasks that relied more on detached input were rated as easier in the SEQ. Participants understood early in the task that their actions were being detected, without being told, and came to expect it. In contrast, participants often did not initially understand that they could provide voice input due to a lack of visual cues and prompts. This is especially surprising when considering how commonplace voice input is, and how uncommonplace detached monitoring is.

Participant 1 said that detached input was “...calming, almost. It takes the panic out of putting things on the stove.” While making ramen, participant 3 made a mistake by putting the noodles in the water and accidentally dropping in a sealed flavor packet before it was even on the stove, much less heated. But, when completing a task with more detached input (making pancakes), the same participant executed the task flawlessly and remarked “It was like somebody was watching over me." Participant 5 said that "almost completely autonomous.” Although they were performing all of the actions, the lack of explicit input led to this feeling of autonomy.

Accommodating different paths

As previously mentioned, we actively avoided prescribing a "correct" way of completing each task in the experiment. This effort was reflected in the diversity of ways our participants completed each task. For example, while making Ramen, four participants began by looking for instructions on the package, while three started by simply pouring water into the pot. One participant did not bother with a measuring cup, explaining that he simply pours out any excess water after making the ramen noodles. Some participants initiated the recipe following process, while others did not realize this was an option and simply prepared the ramen by using the boiling monitoring and timers.

Contextual anchoring is effective -- with caveats

Contextual anchoring was found to be highly effective in our experiment. Contextual anchors to a pot or burner were by far the most usable. Participants did not need to be told where the UI was -- they found it naturally as they completed their task. Participant 6 particularly mentioned that they appreciate how "clear it was to see which timer went to which pot." Participants largely understood, and liked, that they could "leave" the UI where it was while they were waiting for a task to finish. While we were talking in proximity to the stove, waiting for a step to complete, all participants would occasionally glance over to check the status.

A few issues did arise from the use of contextual anchoring. Three participants reported momentarily not being sure where exactly to look, or having to search for a UI. This happened when the interface was just out of the participant's FOV, or if the participant was expecting a certain UI to be available that simply wasn't. For example, some participants were expecting to see instructions for the next step while waiting for the pan to preheat. Although there was no next step available, participants had no way of knowing and would visually scan around the kitchen looking for the step. One participant expressed a desire

for directional cues (like arrows) to guide them towards what they should be looking at.

Adaptive minimalism went largely unnoticed (a good thing!)

Participants mostly did not remember the adaptive minimalism after completing a task. At the end of the second task, we asked participants if they saw the instructions move when adding food to a pot or pan. Only one participant could initially recall seeing this happen. However, after seeing the feature again in the third task, participants generally expressed that they remembered it from before. One participant remarked that it's “like its paying attention to you." The fact that participants did not notice the feature, but also had no trouble viewing the inside of the pot, indicates that the adaptive minimalism was successful. The point, after all, is to get out of the way.

Most users did not find the entry point available from the package of Ramen

The user interface anchored to the package of Ramen was, by far, our least successful UI element. This was partly exacerbated by the combination of Magic Leap's poor image tracking and the significant near clipping plane (0.37 meters). Magic Leap's image tracking suffered at large distances, and required the package to be held relatively close to the headset. At the same time, the near clipping plane prevented things from being viewed when less than 0.37 meters from the glasses.

It also takes a while to recognize the package, and most users have already found the instructions on that package by that time. If eye tracking was more precise, it would be preferred to simply detect that the user is reading the instructions and then offer to start a recipe automatically.

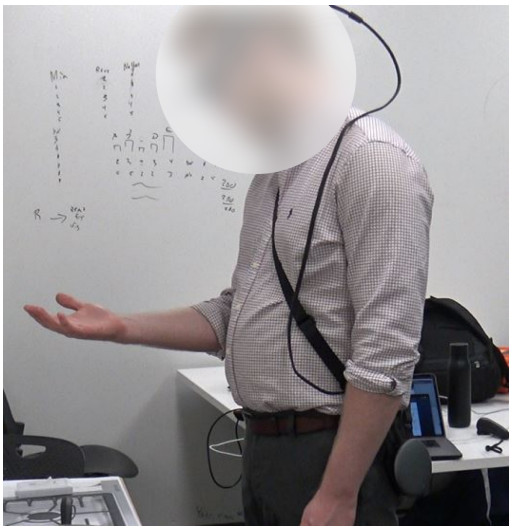
When instructions are anchored to something that's held, users are afraid to put it down.

For the participants that used the Ramen package entry point successfully, they all seemed to hesitate after seeing the initial instruction. Some participants were reluctant to put down the package. In fact, in the picture below, participant 3 was so reluctant to put down the package that they turned on the burner with one finger while holding the package in the same hand. Participants who displayed this behavior said that they did not want to lose the information -- or the user interface in general. Some participants also seemed to think that subsequent steps would also show up on the package.

Figure 53: Participant Holding Onto The Ramen Packet

Participants found the gesture for a voice prompt to be easy and comfortable -- once shown how to do it correctly

The only hand gesture in our system is to summon a handheld voice prompt. In our experiment, we taught users how to invoke the prompt prior to the third task (pancakes). For the first two participants, we tried to verbally tell them how without showing them, but they were unable to do so without seeing how. Once participants were shown how, they were largely successful. After being taught, we instructed users to open/close a voice prompt multiple times and asked if it was physically exhausting. All participants indicated that it was not. One participant specifically remarked that it was "just like taking out a phone."



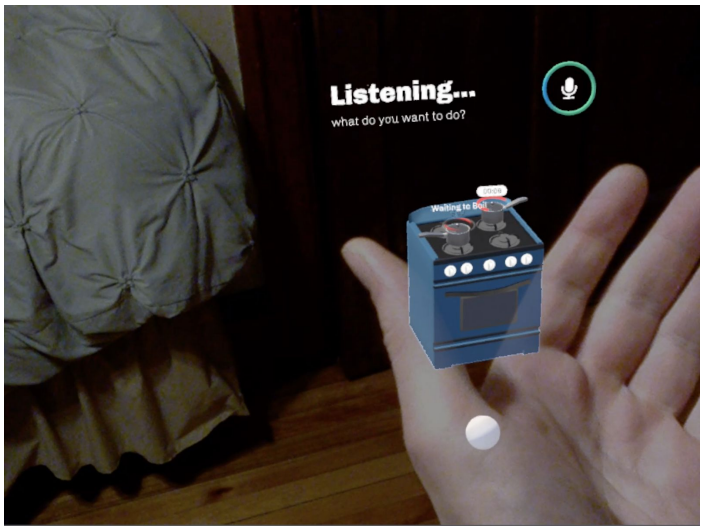


Figure 54: Participant using hand-anchored voice prompt (Left)

Figure 55: Example of what the participant might see (Right)

Participants successfully used gaze to interact with elements of the system, without knowing it

We asked the last three participants to place three pots done and set timers, one at a time. The goal was to determine if participants could use their gaze to select a specific burner. All three of the participants we asked were able to complete this mini-task successfully, without knowing that gaze was the input method. They seemed to naturally look at the burner while talking to it, avoiding any intentional redirection of gaze.

Status and time estimations

Participants largely agreed that in order to feel confident enough to leave the stove, they needed a way to stay informed on the status of the task, including an estimate of time remaining. Notifications are useful, but may be too late. Unlike other participants, participant 7 decided not to sit down and converse while preparing pancakes. Instead, they continued to watch the stove. When asked why they didn't want to leave, they illustrated the problem by walking to the other side of the lab, and pretending to complete activities of daily living, including laundry. When they got a notification that the pancake needed to be flipped, they said they were afraid it would burn before they had a chance to flip it. This was the reason they were wary of leaving the stove — they didn't want to have to rush back immediately.

Participants need to have an estimate of time remaining so they understand how long of a "leash" they have — how far they can go and make it back in time. The handheld stove visualization was later implemented in response to this feedback. A related finding is that participants expressed a need to know what's coming next. It helps them prepare in advance and complete the task in a sequence that makes the most sense for their context.

System Accuracy

To evaluate the performance of each process run by the Rat, a series of tests were conducted where both speed and accuracy measurements were recorded.

Burner Temperature

To evaluate temperature detection by the Rat, we compared the temperature reading from the Rat to the temperature readings from an IR thermometer. The tests conducted to make the comparison are recording boiling temperature and burner surface temperature.

When running the tests, we realized that the burner surface temperatures and the temperature of hot pans almost always exceed 80°C, which is the maximum temperature the thermal camera could detect. Although the thermal camera could actually give readings higher than 80°C, the accuracy decreased significantly with higher temperatures.

Detecting when a Burner is On

Although the thermal camera could not detect the high temperatures of burner surfaces accurately, it could still detect changes in temperature and could

be used to detect when a burner was on. To evaluate the ability of the Rat to detect when a burner is on, we conducted a test where a burner was turned on for a few minutes and then turned off. We measured the amount of time it took for the device to record that a burner was on after it was turned on. We also recorded the points of the experiments at which false detections were made.

There was a delay of around 5 seconds before the device detected that a burner was on. That is caused by the fact that the burner does not get immediately hotter than the surrounding environment the moment it is turned on. After the burner has been turned on. As the temperature rises, the status detected by the device switches between on and off. Once the temperature reaches a steady level, the device could detect that the burner was constantly on. Similar results were obtained when detecting that a burner was turned off. Once the burner is turned off, the temperature does not significantly drop immediately. Therefore, the status detected by the device switches from on to off for around a 30 seconds. After the temperature starts decreasing more significantly, the device is able to detect that a burner was turned off.

Detecting Boiling Water

Three boiling experiments were conducted to record the delay between visual boiling and boiling detected by the Rat. The device detected boiling from a

minute to a minute and thirty seconds early in all cases. That is caused by the fact that the boiling detection algorithm checks if the water temperature is above a certain threshold before assigning a boiling status. The inaccuracy and inconsistency of temperature readings collected from the thermal camera are the main reason for this issue.

Detecting the Presence of a Pot

We measured the accuracy and speed of the pot detection algorithm by placing a variety of pots on each of the four burner. When tested with pots that the machine learning model was trained with, the device was able to successfully detect when a pot was placed on each of the four burners. However, the device was not able to accurately detect pots that were not included in the training dataset. When a pot from outside the dataset was placed on the stove, the device could detect it with a probability lower than 20%, which is not a high enough probability to avoid false detections and poorly positioned bounding boxes. The speed of the object detection algorithm using Edge TPU was measured to be 5 FPS.

Pancake State Detection

The accuracy of our pancake state detection model can be measured through the quality of pancakes made. Therefore, 3 pancakes were made with the

assistance of the Rat. The quality of the pancakes and the consistency of the results over the three pancakes were evaluated. We also measured the delay of the device to detect each pancake state.

All pancakes states were detected successfully and the pancakes made can be seen in figure 56. The test also showed the consistency of pancake detection across the three pancakes since they all have a similar golden brown color. The speed of pancake detection using Edge TPU is 2.5FPS.

Figure 56: Pancakes made with the Rat

7. Reflections on Design and Tradeoffs

Before settling on our final method of running machine learning models, which is using the Edge TPU Coral accelerator, multiple different methods were considered. The first method we considered was using Azure Custom Vision AI [source]. Although the Custom Vision model was highly accurate and could recognize a wide variety of pots, it had two major drawbacks. The first drawback was the high latency, since its processing speed was 0.25 FPS. The second drawback was that the model was hosted online and therefore has privacy concerns associated with it. Since maintaining privacy of the users was a major design criteria for the Rat, we decided to explore other options for running machine learning models.

Running a re-trained tensorflow model locally on the Raspberry Pi was our next option. However, as mentioned in the implementation section, the processing speed was 0.067 FPS. When we converted the Tensorflow model to a TFLite model and then to an Edge TPU TFLite model, there was a clear tradeoff between accuracy and speed. The Tensorflow model was the most accurate. For example, the pot detection tensorflow model could detect pots with probabilities higher than 90%. TFLite model was significantly faster with a framerate of 0.2 FPS, but with lower accuracy. The TFLite model could detect a pot with probabilities higher than

70%. The Edge TPU TFLite model has a framerate of 5FPS, but the pot detection accuracy drops again to between 50% and 80%. However, the Edge TPU accuracy was still high enough to detect pots it was trained on, so we decided to use it for the purposes of our prototype.

Because the Coral accelerator was released only a few weeks before the end of this project, we considered another inference acceleration option before we could get a Coral accelerator. We considered the Intel Neural Compute Stick 2, which uses a VPU (Vision Processing Unit) to accelerate machine vision tasks. Using Intel Neural Compute Stick for our specific application was not efficient for the following reasons:

- Lack of detailed documentation on how to use the device.
- No direct compatibility with Tensorflow models. Models needed to be converted to a format compatible with the ncsdk.
- Incompatibility with custom trained Tensorflow model.

8. Future Work

Measuring Awareness

Notifications are a key part of detached monitoring. Effective notifications can help prevent errors like burnt pancakes and unattended burners. In this project we implemented two different forms of notifications: diegetic ("heads up") and nondiegetic (virtual objects in the user's environment). Unfortunately, we did not have the time to determine which form was the most effective. In the future, we would like to conduct a controlled study to determine which type of notification can keep users appropriately aware and prevent dangerous situations.

Real Kitchen Environment

Prior to our experiment, we did not know whether or not AR would risk participants' safety. We are now confident that it will not. In the future, we are planning to conduct a follow-up study in a real kitchen environment.

Expanding Detached Monitoring

Our project implemented a single form of detached monitoring, but there are countless applications in the kitchen and beyond. One application we'd like to pursue is a smart kitchen scale. The scale would use AR to show a progress bar as

different ingredients are added to a mixing bowl. It would enable the user to walk away in the middle of pouring an ingredient, effectively "saving" their working state at any given time. Like the Rat, it would also be able to detect actions like when something is placed on the scale or the user is done adding an ingredient.

Thermal Camera

Due to the temperature range limitations of the thermal camera we used, the Rat could not collect accurate surface or food temperatures on each burner. To solve this problem, a thermal camera with a larger temperature range should be used. In a future iteration of the project, we will use the MLX90640 which has a 110° view angel and can measure temperatures between -40°C and 300°C.

Recipes from the Cloud

One potential feature that can be added to the system is accessing and parsing recipes from the cloud for the user to follow. Some of the participants in our experiment expressed interest in being able to look up a recipe using Remy and have the device guide them through the recipe.

9. Conclusion

The primary goal of the project was to explore how detached monitoring can help improve cooking. Based on our research and experiments, we believe that detached monitoring can make cooking easier, safer, and less stressful by guiding the user through recipes and keeping the user informed no matter where they are in the home. Participants in our experiment definitively reported that cooking tasks were easier the more detached monitoring was involved. In the future, AR , combined with detached monitoring, will certainly have a place in the kitchen and the rest of the home.

References

[1] Marty Ahrens. 2018. Home Structure Fires. (December 2018). Retrieved April 21, 2019 from https://www.nfpa.org/News-and-Research/Data-research-and-tools/Building-and-Life-Safety/Home-Structure-Fires

[2] Genevieve Bell and Joseph Kaye. 2002. Designing Technology for Domestic Spaces: A Kitchen Manifesto. Gastronomica 2, 2 (2002), 46–62. DOI:http://dx.doi.org/10.1525/gfc.2002.2.2.46

[3] Anon. Cloud AutoML - Custom Machine Learning Models | Google Cloud. Retrieved April 25, 2019 from https://cloud.google.com/automl/

[4] Jenny Cooper. 2015. Cooking Trends Among Millennials: Welcome to the Digital Kitchen. (June 2015). Retrieved April 24, 2019 from https://www.thinkwithgoogle.com/consumer-insights/cooking-trends-among-millennials/

[5] Brian Curry. 2018. An Introduction to Transfer Learning in Machine Learning. (July 2018). Retrieved April 23, 2019 from https://medium.com/kansas-city-machine-learning-artificial-intelligen/an-introduction-to-transfer-learning-in-machine-learning-7efd104b6026

[6] Erhopf. 2019. Speech-to-text with Azure Speech Services - Azure Cognitive Services. (March 2019). Retrieved April 25, 2019 from https://docs.microsoft.com/en-us/azure/cognitive-services/speech-service/speech-to-text

[7] Patrick Farley. 2019. What is Azure Custom Vision? (March 2019). Retrieved April 24, 2019 from https://docs.microsoft.com/en-us/azure/cognitive-services/custom-vision-service/home

[8] Krista Garcia. 2018. Smart Appliances Haven't Found a Home Yet. (July 2018). Retrieved April 23, 2019 from https://www.emarketer.com/content/smart-appliances-haven-t-found-a-home-yet

[9] Thomas Himblot. 2018. Data augmentation : boost your image dataset with few lines of Python. (March 2018). Retrieved April 23, 2019 from https://medium.com/@thimblot/data-augmentation-boost-your-image-dataset-with-few-lines-of-python-155c2dc1baec

[10] Adam Jones, J.Edward Swan, Gurjot Singh, and Eric Kolstad. 2008. The Effects of Virtual Reality, Augmented Reality, and Motion Parallax on Egocentric Depth Perception. 2008 IEEE Virtual Reality Conference (2008). DOI:http://dx.doi.org/10.1109/vr.2008.4480794

[11] Bui Minh Khuong, Kiyoshi Kiyokawa, Andrew Miller, Joseph J.La Viola, Tomohiro Mashita, and Haruo Takemura. 2014. The effectiveness of an AR-based context-aware assembly support system in object assembly. 2014 IEEE Virtual Reality (VR)(2014). DOI:http://dx.doi.org/10.1109/vr.2014.6802051

[12] Page Laubheimer. 2018. Beyond the NPS: Measuring Perceived Usability with the SUS, NASA-TLX, and the Single Ease Question After Tasks and Usability Tests. (February 2018). Retrieved April 16, 2019 from https://www.nngroup.com/articles/measuring-perceived-usability/

[13] Chia-Hsun Lee, Leonardo Bonanni, and Ted Selker. 2005. CounterIntelligence: Augmented Reality Kitchen. CHI 2239 (January 2005), 45.

[14] Hannah Limerick. 2019. Haptics | 6 reasons touch is important. (April 2019). Retrieved April 24, 2019 from https://www.ultrahaptics.com/news/blog/haptics-touch-important/

[15] Ashley Lutz. 2015. 5 ways millennials' dining habits are different from their parents'. (March 2015). Retrieved April 23, 2019 from https://www.businessinsider.com/millennials-dining-habits-are-different-2015-3

[16] Lars Müller, Ilhan Aslan, and Lucas Krüßen. 2013. GuideMe: A Mobile Augmented Reality System to Display User Manuals for Home Appliances.

Lecture Notes in Computer Science Advances in Computer Entertainment(2013), 152–167. DOI:http://dx.doi.org/10.1007/978-3-319-03161-3_11

[17] Jakob Nielsen. 1994. Enhancing the explanatory power of usability heuristics. Conference companion on Human factors in computing systems - CHI 94 (1994). DOI:http://dx.doi.org/10.1145/259963.260333

[18] Donald Norman. 2013. The Design of Everyday Things: Revised and Expanded Edition, New York: Basic Books.

[19] F. Nugroho and A.B. Pantjawati. 2018. Automation and Monitoring Smart Kitchen Based on Internet of Things (IoT). IOP Conference Series: Materials Science and Engineering 384 (2018), 012007. DOI:http://dx.doi.org/10.1088/1757-899x/384/1/012007

[20] Anon. 2019. OpenCV. (April 2019). Retrieved April 23, 2019 from https://opencv.org/

[21] Dale Purves et al. 2019. Neuroscience, New York: Oxford University Press.

[22] Anon. 2019. Reactive programming. (March 2019). Retrieved April 24, 2019 from https://en.wikipedia.org/wiki/Reactive_programming

[23] Jesse Rhodes. 2011. The Evolution of the Modern Kitchen. (May 2011). Retrieved April 23, 2019 from

https://www.smithsonianmag.com/arts-culture/the-evolution-of-the-modern-kitchen-164457607/

[24] Sara Robinson, Aakanksha Chowdhery, and Jonathan Huang. 2018. Training and serving a realtime mobile object detector in 30 minutes with Cloud TPUs. (July 2018). Retrieved April 24, 2019 from https://medium.com/tensorflow/training-and-serving-a-realtime-mobile-object-detector-in-30-minutes-with-cloud-tpus-b78971cf1193

[25] Anon. 2019. scikit-image. (April 2019). Retrieved April 23, 2019 from https://scikit-image.org/

[26] Jeff Sneider. 2014. Steven Spielberg Hiring 'Godzilla' Writer for 'Minority Report' TV Series (Exclusive). (August 2014). Retrieved April 23, 2019 from https://www.thewrap.com/steven-spielberg-hiring-godzilla-writer-for-minority-report-tv-series-exclusive/

[27] Steven Spielberg. 2003. Minority Report.

[28] Marcus Stander, Aristotelis Hadjakos, Niklas Lochschmidt, Christian Klos, Bastian Renner, and Max Muhlhauser. 2012. A Smart Kitchen Infrastructure. 2012 IEEE International Symposium on Multimedia(2012). DOI:http://dx.doi.org/10.1109/ism.2012.27

[29] Anon. Structural similarity index. Retrieved April 23, 2019 from https://scikit-image.org/docs/dev/auto_examples/transform/plot_ssim.html

[30] Tensorflow. 2019. tensorflow/models. (March 2019). Retrieved April 23, 2019 from https://github.com/tensorflow/models/tree/master/research/slim/nets/mobilenet

[31] Tzutalin. 2019. tzutalin/labelImg. (April 2019). Retrieved April 23, 2019 from https://github.com/tzutalin/labelImg

[32] Anon. What is Gorilla Arm? - Definition from Techopedia. Retrieved April 24, 2019 from https://www.techopedia.com/definition/31480/gorilla-arm

Appendix

Appendix A

Simulator Sickness Survey

Instructions : Circle how much each symptom below is affecting you right now

1. General discomfort	None	Slight	Moderate	Severe
2. Fatigue	None	Slight	Moderate	Severe
3. Headache	None	Slight	Moderate	Severe
4. Eye Strain	None	Slight	Moderate	Severe
5. Difficulty Focusing	None	Slight	Moderate	Severe
6. Salivation Increasing	None	Slight	Moderate	Severe
7. Sweating	None	Slight	Moderate	Severe
8. Nausea	None	Slight	Moderate	Severe
9. Difficulty Concentrating	None	Slight	Moderate	Severe
10. « Fullness of the Head »	None	Slight	Moderate	Severe
11. Blurred vision	None	Slight	Moderate	Severe
12. Dizziness with eyes open	None	Slight	Moderate	Severe
13. Dizziness with eyes closed	None	Slight	Moderate	Severe
14. *Vertigo	None	Slight	Moderate	Severe

15. **Stomach awareness	None	Slight	Moderate	Severe
16. Burping	None	Slight	Moderate	Severe

* Vertigo is experienced as loss of orientation with respect to vertical upright.

** Stomach awareness is usually used to indicate a feeling of discomfort which is just short of nausea.

Setting a Timer

Single Ease of Use Question (SEQ)

Overall, this task was...

Easy						Difficult
1	2	3	4	5	6	7

Making Ramen

Single Ease of Use Question (SEQ)

Overall, this task was...

Easy						Difficult
1	2	3	4	5	6	7

Making Pancakes

Single Ease of Use Question (SEQ)

Overall, this task was...

Easy						Difficult
1	2	3	4	5	6	7

System Usability Scale

SUS uses a 5-point scale where 1 corresponds to strongly disagree and 5 corresponds to strongly agree.

1. I think that I would like to use system frequently

Strongly Disagree				Strongly Agree
1	2	3	4	5

2. I found the system unnecessarily complex

Strongly Disagree				Strongly Agree
1	2	3	4	5

3. I thought the system was easy to use

Strongly Disagree				Strongly Agree
1	2	3	4	5

4. I think that I would need the support of a technical person to be able to use the system

Strongly Disagree				Strongly Agree
1	2	3	4	5

5. I found the various functions in the system were well integrated

Strongly Disagree				Strongly Agree
1	2	3	4	5

6. I thought there was too much inconsistency in the system

Strongly Disagree				Strongly Agree
1	2	3	4	5

7. I would imagine that most people would learn to use the system very quickly

Strongly Disagree				Strongly Agree
1	2	3	4	5

8. I found the system very cumbersome to use

Strongly Disagree				Strongly Agree
1	2	3	4	5

9. I felt very confident using the system

Strongly Disagree				Strongly Agree
1	2	3	4	5

Appendix B

System Personas

Parent

A lot to cook, and a lot of mouths to feed.

Contexts

- Making breakfast for 3 kids while also preparing their lunches for school
- Trying to prepare dinner as the baby is crying
- Preparing Thanksgiving for 15 people
- Toddlers running around the kitchen (kid trying to touch stove unattended)
- Sharing the kitchen
- Talking to people while cooking (divided attention)
- go-to meal for family

College Student

Not experienced, not sophisticated, busy and poor

Contexts

- finals week -- even less time than usual, and stressed
- on the go-able food
- food that's hard to mess up
- cooking ramen/macaroni for dinner (just boil water)
- cooking the same thing for the 100th time (repetition)
- shopping for weekly groceries, needs food that fits budget

Health Freak

Counting calories, Paleo, vegan, etc.

Contexts

- craving a certain thing, but want it in healthy meal
- wants to find something to eat that fits calorie range

Busy Bee

Knowledgeable about cooking, but short on time

Contexts

- Meal prepping for the week

- food within timespan [30 mins] -- want to maximize time, food that can be cooked in time span
- on a conference call while cooking
- taking at the look and seeing what can be made (not preparing a complex dish ahead of time)
- wants to cook a quick meal, but doesn't want to feel like a college student (+ wants to eat healthy)
- Dressed for work and doesn't want to get their clothes dirty
- Consistent routine everyday (breakfast + dinner @ certain times)
    - things ready when for them when they get back (Whatever that means) (recipes prepared?)
- eats lunch at work
- tell you what you're missing so you can buy it

Linguini

Amateur chef

Contexts

- Waiting for a pie to bake in the oven (waiting a really long time)
- Cooking things that takes a while, and is not attended

- Cooking things that need to be checked up on periodically
- Lots of ingredients, lots of steps, lots of things to mess up
- Figuring out what flavors would work well together -- combinations of things

Appendix C

AMG8833 Datasheet Highlights

4-2 Main Functions

Item	Value
Pixel number	64 (8×8 Matrix)
External Interface	I²C (fast mode)
Frame rate	Typ. 10 frames/sec or Typ. 1 frame/sec
Operating Mode	Normal Sleep Stand-by (10sec or 60sec intermittence)
Output Mode	Temperature Output
Calculate Mode	No moving average or Twice moving average
Temperature Output Resolution	0.25℃
Number of Sensor Addresses	2 (I²C Slave Address)
Thermistor Output Temperature Range	-20℃～80℃
Thermistor Output Resolution	0.0625℃

4-3 Absolute Maximum Ratings

Item	Specification	Terminal
Applied voltage	-0.3～6.5V	VDD, VPP
Input/Output voltage	-0.3～Vdd+0.3V	SCL, SDA, AD_SELECT
Output current	-10～10mA	INT, SDA
ESD (Human Body Model)	1kV	All Terminals
ESD (Machine Model)	200V	All Terminals

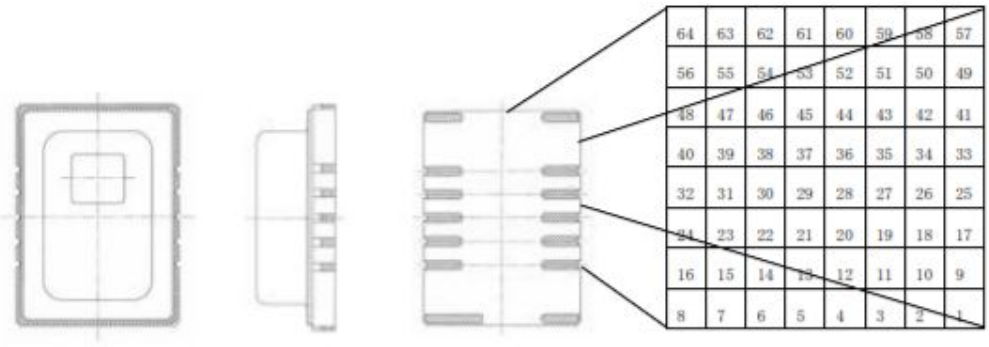
4-4 Ratings

Item	Specification	
	High gain	Low gain
Applied voltage	3.3V±0.3V or 5.0V±0.5V	
Temperature Range of Measuring Object	0℃～80℃	-20℃～100℃
Operating temperature	0℃～80℃	-20℃～80℃
Storage temperature	-20℃～80℃	

4-7 Pixel Array & Viewing Field

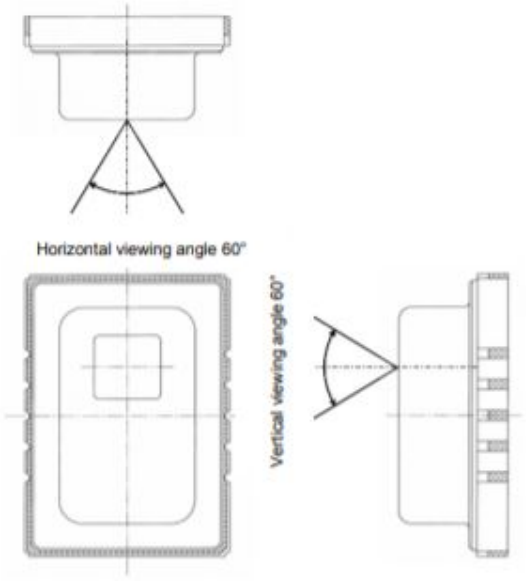
(1)Pixel Array

Pixel Array from 1 to 64 is shown below.



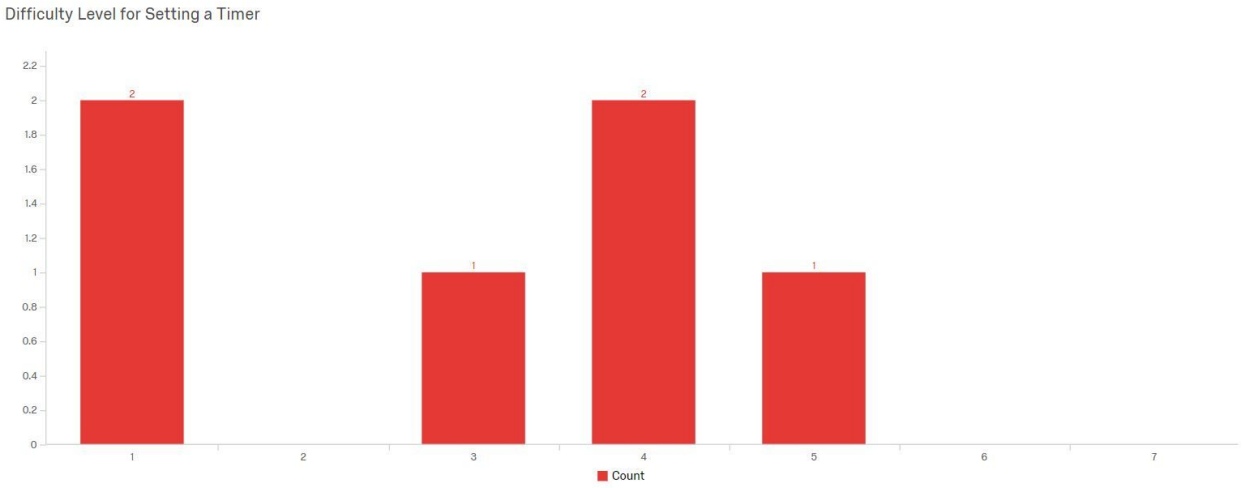
(2)Viewing Field

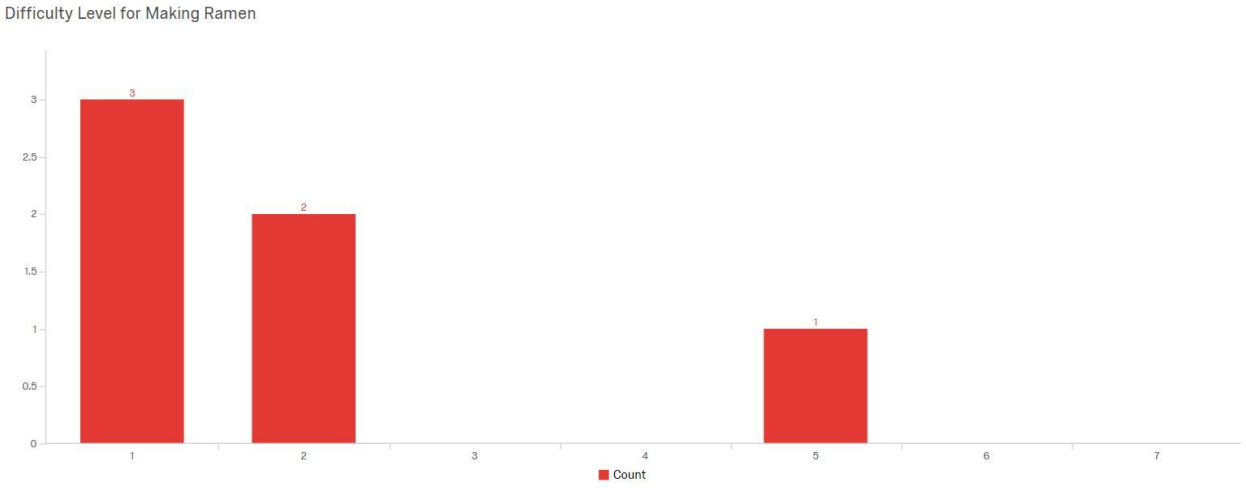
Sensor Viewing Field (Typical) is shown below.

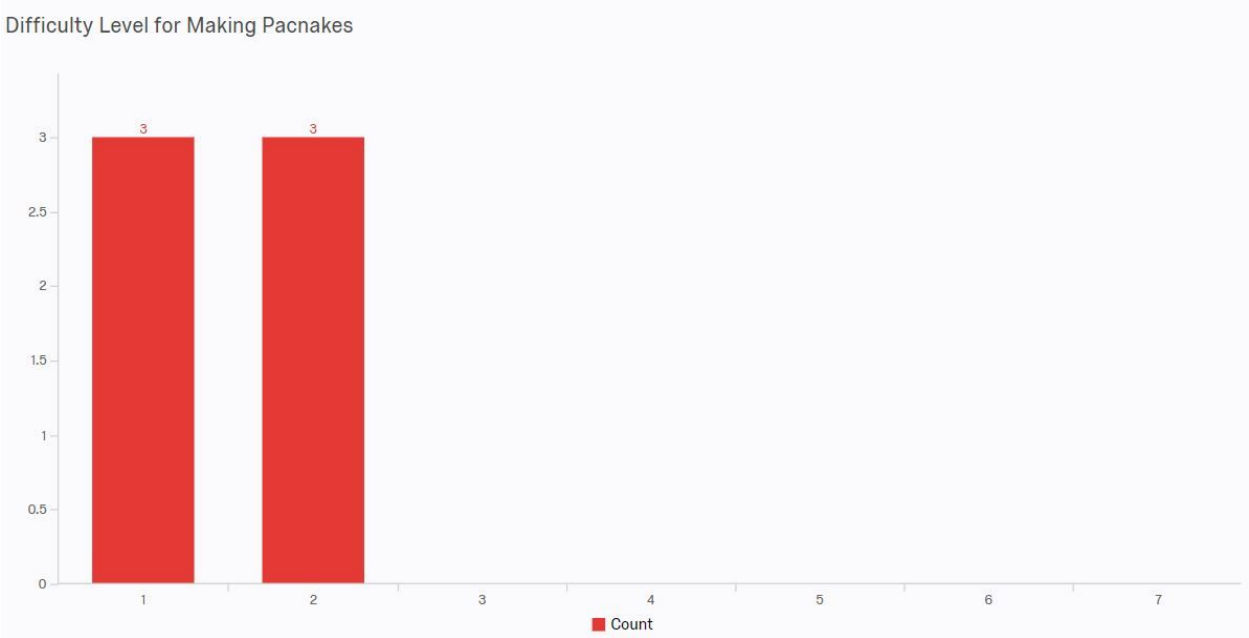


Appendix D

Answers to Single Ease Questions



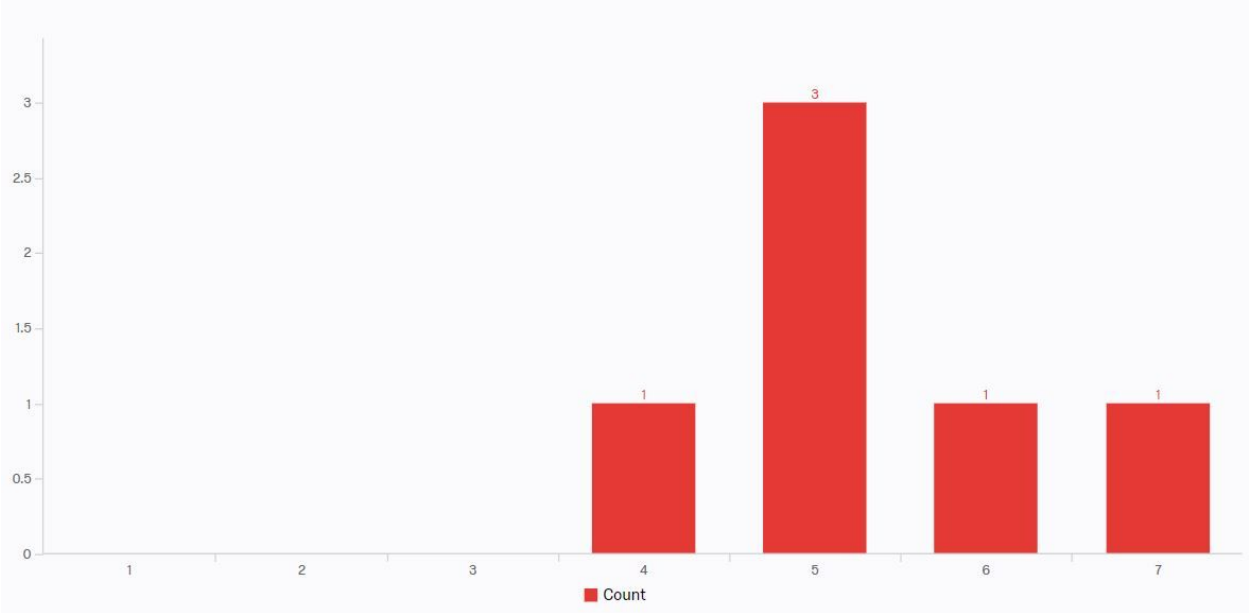



Difficulty Level for Making Pacnakes
3
3
3
2.5
2
1.5
1
0.5
0
1
2
3
4
5
6
7
Count

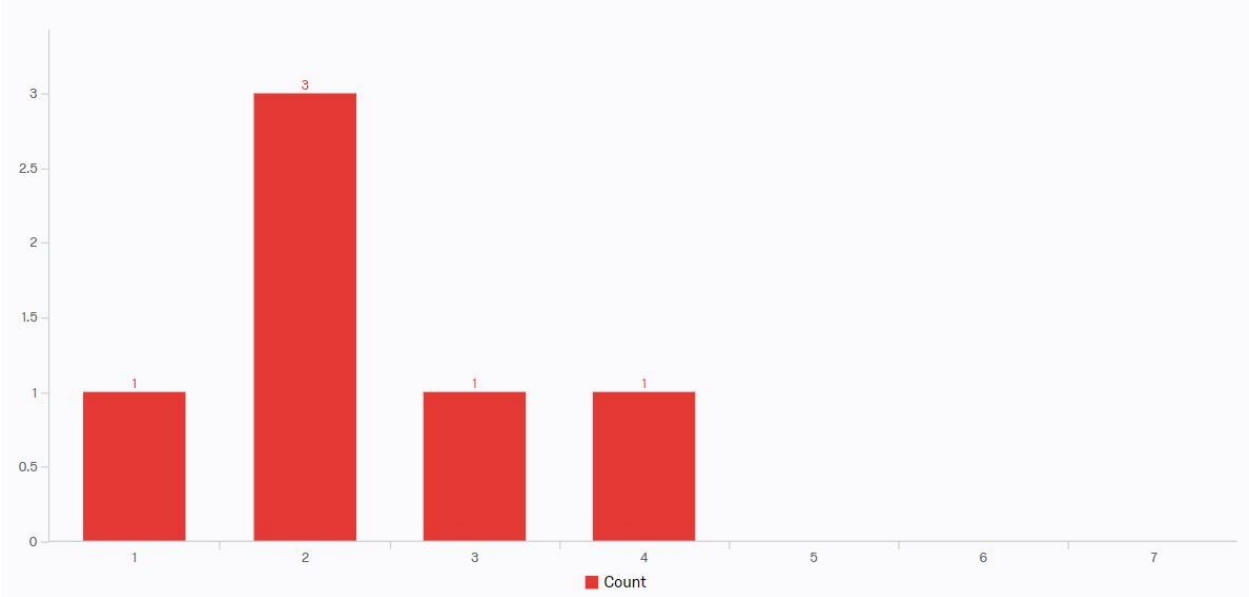
Appendix E

Answers to System Usability Scale

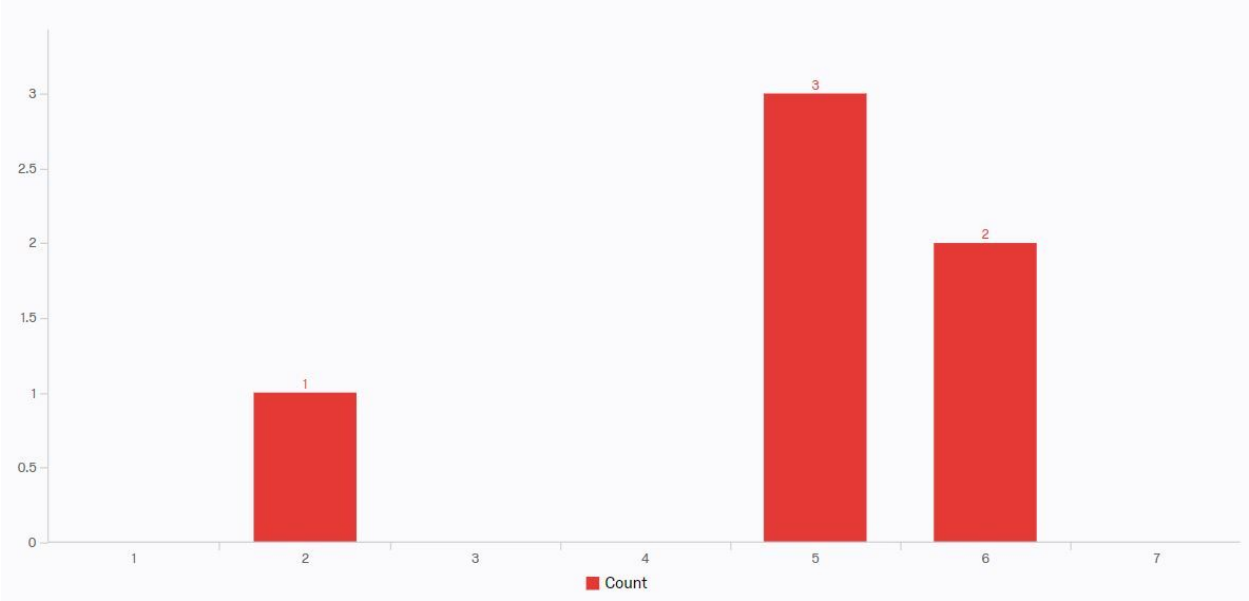
Q1. I think that I would like to use system frequently



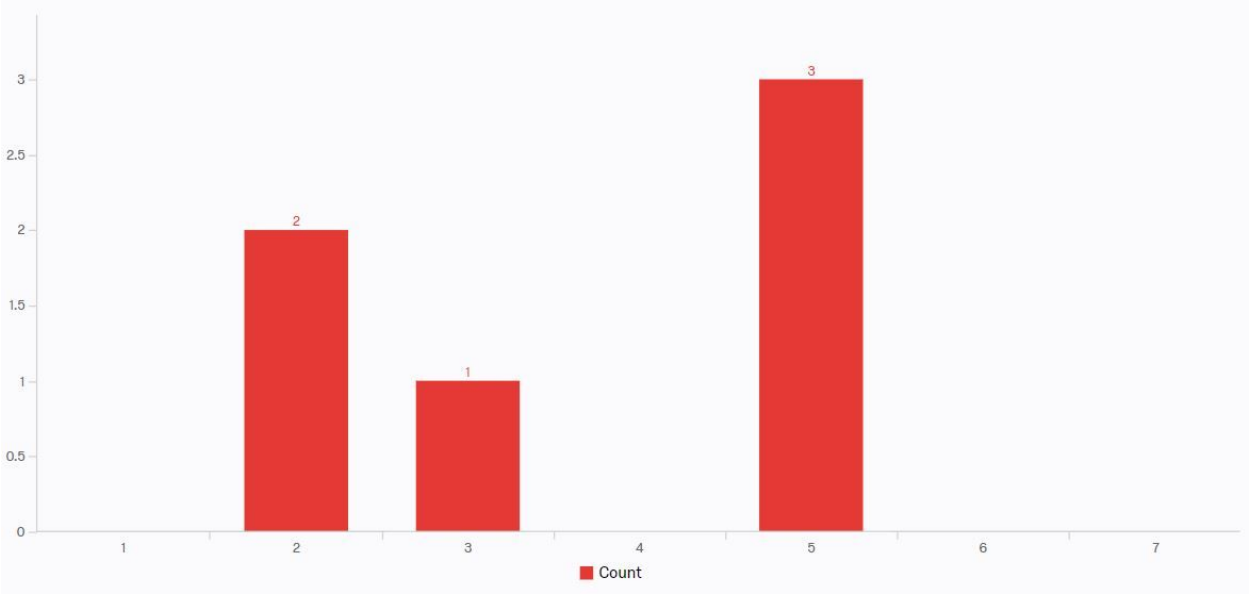
Q2. I found the system unnecessarily complex



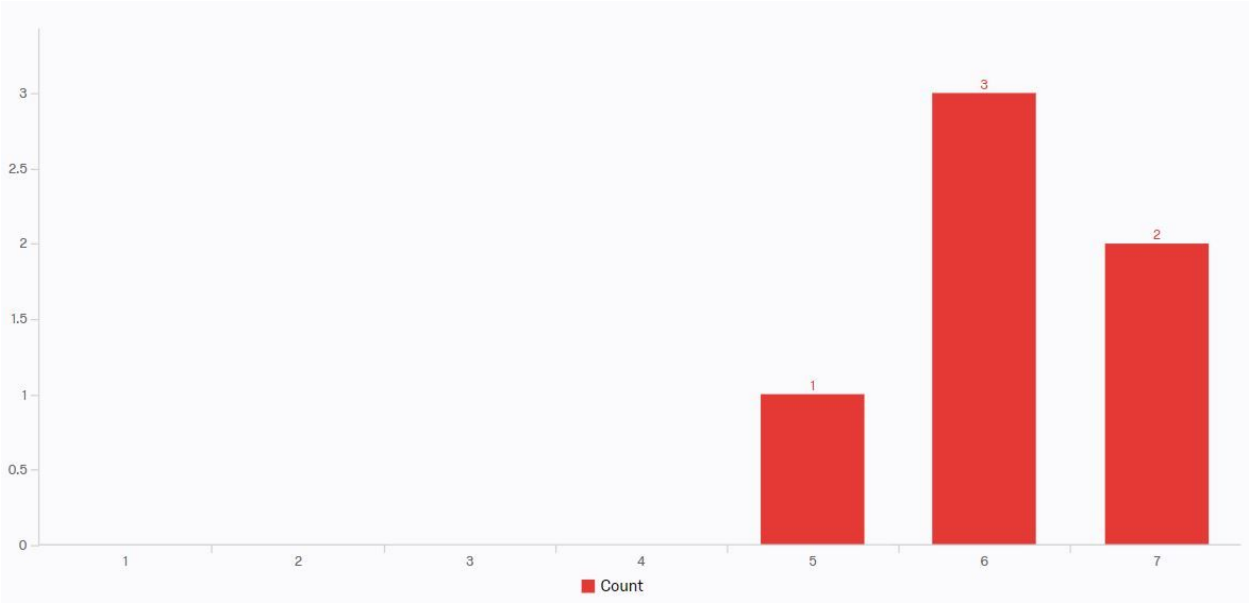
Q3. I thought the system was easy to use



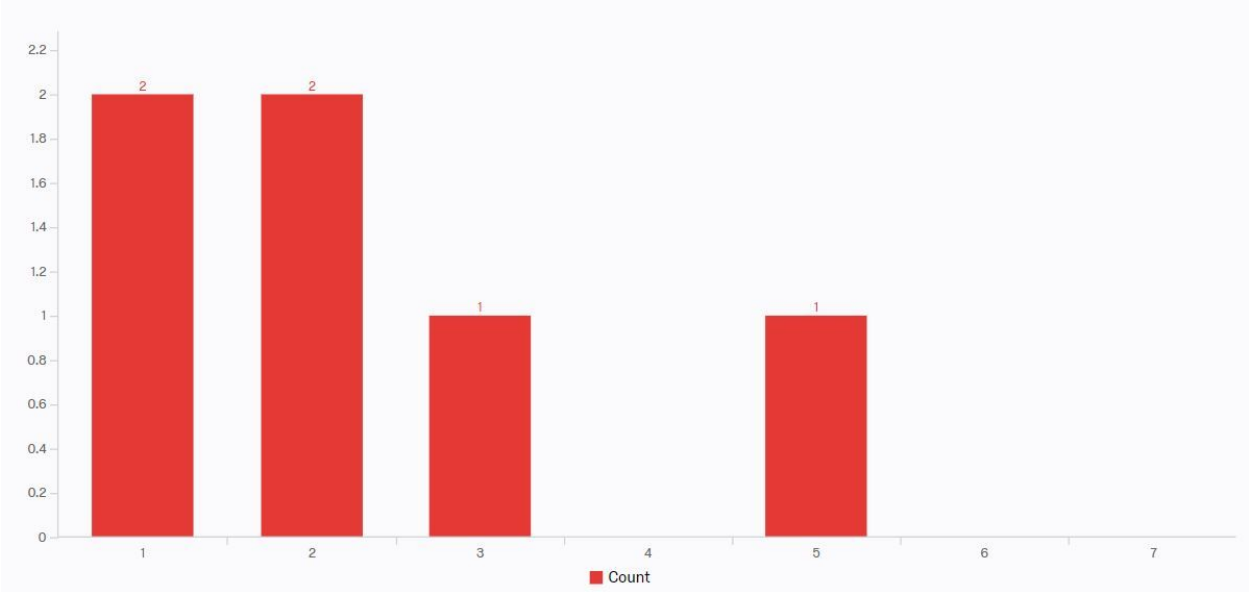
Q4. I think that I would need the support of a technical person to be able to use the system



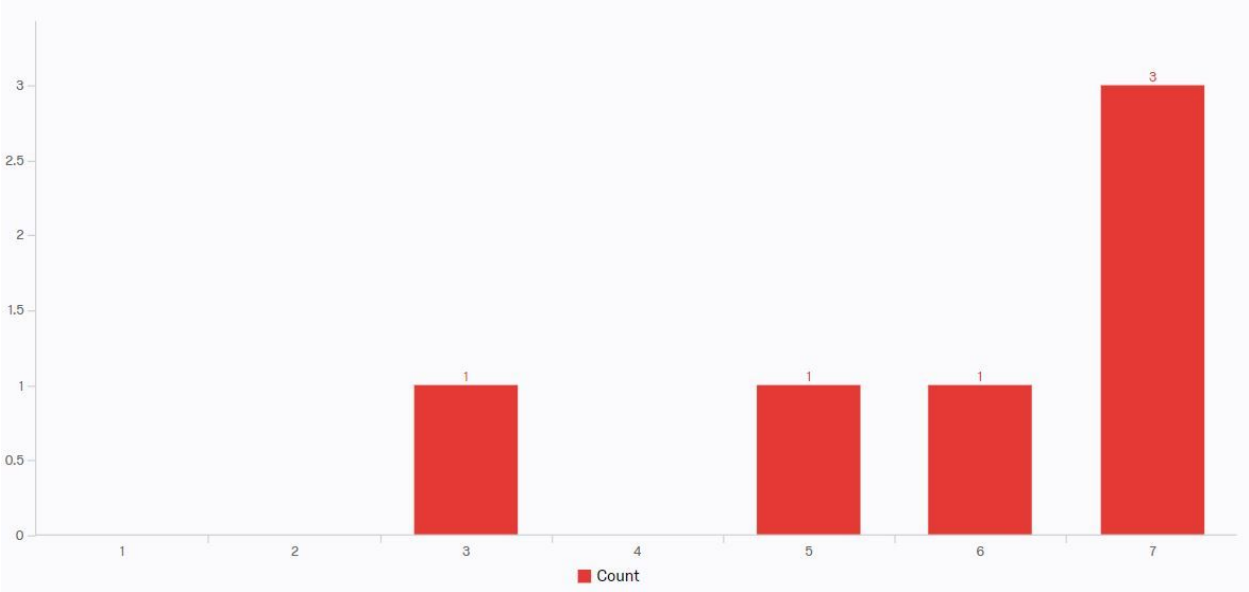
Q5. I found the various functions in the system were well integrated



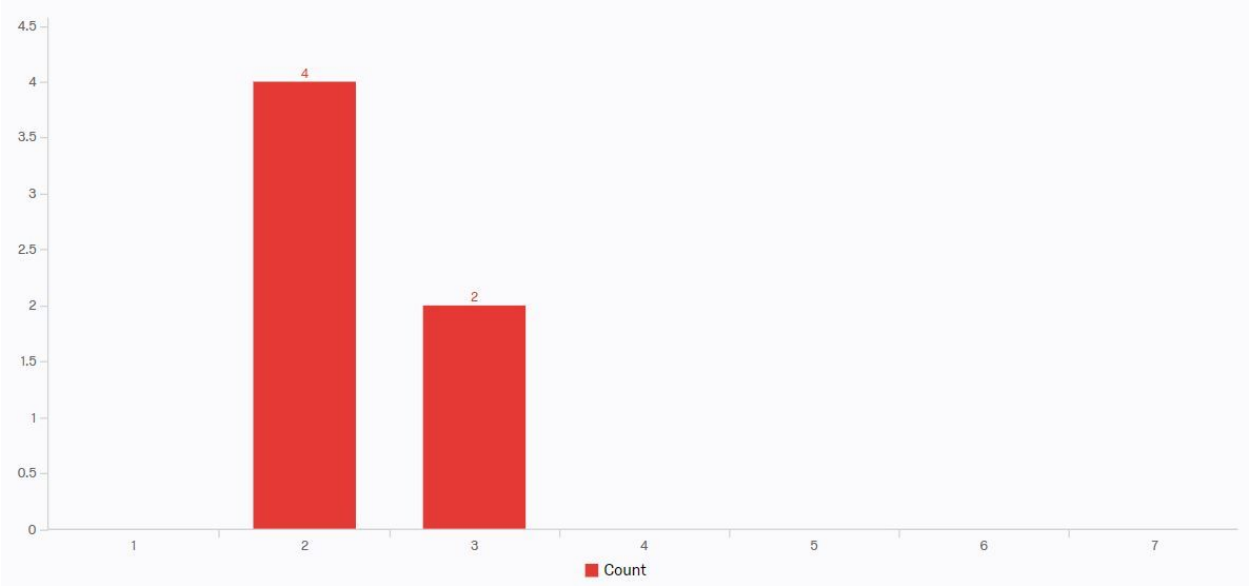
Q6. I thought there was too much inconsistency in the system



Q7. I would imagine that most people would learn to use the system very quickly



Q8. I found the system very cumbersome to use



Q9. I felt very confident using the system