Autonomous Butter Robot

A Senior Project Report

presented to

the Faculty of California Polytechnic State University

San Luis Obispo

Partial Fulfillment

of the Requirements for the Degree

Bachelor of Science in Computer Engineering

by

Michael Hegglin & David Chau

June 2020

Abstract

Don’t you wish your butter would come to you? Well now it can with the patented Michael and David butter robot! Based on an idea from a TV show, our team set out to see if a similar robot was possible to make in real life. The objective was simple. Can we make a small table sized robot that can bring a person butter using image detection software? With that question in mind we set out buying our components. We wanted to keep it small, so we looked up devices that could do simple image processing and from there we based the robot design off what we thought a small homemade rc car might look like. After continuous testing we found that yes, it is possible to make a small table sized butter robot! Now you won’t even have to get out of your chair to butter your waffles.

Table of Contents

Introduction

As Americans get lazier and lazier, they soon won't even be able to stand up to get their own butter. Our product solves this problem! Our butter robot can easily find a piece of bread using image detection, and bring the butter to you! This could be used for anyone one who butters bread in their daily routine.

Stakeholders

Anyone who has a big family or large tables would love to have our product. No more is asking your relative to pass the butter from across the table. The robot could also be improved upon to bring more butter over to someone as soon as their butter runs out. This means our same robot, with a few modifications could potentially be used in a restaurant to deliver the butter after bread has been served to truly allow its users to embrace their laziness in those moments.

Framed Insights and Opportunities

The most important things that came up when talking to people who said they would want a butter robot was: how often would you have to recharge it, how fast would it get me my butter, and can the butter fall off the robot? To address these things, we first decided to use Lego pieces combined with adhesive to construct the body of the robot and also keep the butter from falling out or spilling over the edge. This also meant we could shape the Legos however we

desired for the rest of the robot. Next we looked at our power delivery situation. While our robot could be improved to use wireless/inductive charging and port docking, we decided to pursue using a rechargeable battery pack. This way we could keep the expenses down, and still make it so the consumer would not have to worry about buying more batteries. Finally our team had to decide the speed of the robot. We wanted it to be fast enough so that the user wasn't waiting an awkward amount of time for the butter but slow enough so that it wouldn't fly off the table. After looking at different gearing that could attach to our motors, we decided to go with a gearing that matched that of a slow RC car. These parts were easy to find and purchase to both keep the cost down and adjust for the right speed.

Projects Goals and Objectives

Goal: Bring butter to the user

- Objective 1: Create a robot body to hold the butter
- Objective 2: Use motors to move the body
- Objective 3: Use camera imaging to identify bread
- Objective 4: Use a raspberry pi to keep the robot small
- Objective 4: Use raspberry pi for logic and talk to the arduino to drive the motors

Project Deliverables

- A robot that can bring a user butter to consume
- A raspberry pi that can use a camera imaging to identify bread
- A raspberry pi that can use a camera imaging to center the robot's position on the bread
- A raspberry pi that can use a camera to detect its relative distance to the bread
- A raspberry pi that can send commands to an Arduino
- An Arduino that can tell the motors to start and stop moving
- A battery that can drive the robot so it is independent of wired DC power supply

Project Outcomes

With our project now done, we no longer have to pass our butter to the person next to us. We can simply use our butter robot.

Background

This project was inspired by a comedy TV show called Rick and Morty. In the show, the character Rick spontaneously created an automated robot that passed butter around the table. This robot stood on one wheel and had a camera on top. While the structural design of the cartoon's robot is quite different from ours, the basic premise and logic are the same.

We decided to use Python due to the fact that it is very portable and easy to write. It runs perfectly fine on a raspberry pi and there are many easy to use proven libraries that can be installed specifically for Python. Not only are these libraries open source and easy to use, but someone else has gone through and done the security testing for us. One important library we used was Nanpy, where we can directly tell the Arduino, flashed with Nanpy firmware, what to do under specific conditions.

The biggest ethical concern we had with the project was privacy concerns due to the use of a camera by an autonomous robot moving around people's homes. To solve this issue, everything is disconnected from the internet. No parts of the application need any sort of internet connection to move. Going along with this, the raspberry pi does not save any of the video feed after it is received. All the application does is look at a frame to check if it's still on path to reach the butter, then it throws the frame away once it is done.

Currently there are no other butter robots that we know of on the market today. The biggest competitor with our product would be a normal table container to hold your butter outside the refrigerator so that it is soft and ready to spread. The best part of our robot is that it can be used in this same way if the owner for some reason chooses not to actually run it.

Formal Project Definition

Customer Requirements

- The robot will bring butter to the user though DC motors
- The robot will use camera imaging to detect when bread is there
- The robot will be rechargeable
- The robot will be small (deployable)
- The robot will be relatively inexpensive

Engineering Requirements

Product Requirements Table					
Spec Number	Parameter Description	Requirements	Compliance	Tolerance	Risk
1	Battery Power Having the ability to run under its own power for a meal and be rechargeable	1hr	T, I	Min.	Low.
2	Camera Imaging Use Camera imaging to see if a piece of bread is there	N/A	T, A	Max.	High.
3	Deployability	12”x12”	T, I	Max.	Med.

	Having the ability to put the robot anywhere with low surface area				
4	Butter will be brought to the user	N/A	T, I, A	Max.	High.
5	Price Keep the robot as inexpensive as possible for the buyer	<$200	A	Min.	Low.

Customer Personas

The first type of person who may want our product is Larry Lobster. This type of person already has great enthusiasm for robots in general. They may not know much about them, but they enjoy watching them work and think that they are very cool items to have. This type of person might not use the robot on a daily basis but more as an item to show off when people come over, this person probably has a large dining table or other surface on which the robot can run.

The second type of person to buy are people like Lucas. Lucas is an experienced mechanical engineer, and he understands the technical knowledge that goes into designing and creating a robot. He enjoys messing with things himself and likes to tinker with the robot to make it more accustomed to his needs and wants. This type of person has a good understanding

of how the product is made, and may eventually cannibalize it to create their own project or further develop the product itself.

The final type of person who would want to buy the product is Bertha. Bertha is somebody who has trouble physically getting around her home. This type of person likes that robots can help them with their daily tasks, even if it is as simple as moving butter during a meal. Instead of having to sit or stand up to take some butter, the robot is there to make this process relatively easier.

*** Look in Appendices/Personas for more information about Larry, Lucas, and Bertha ***

Use Cases

The main user of our system is anyone who uses butter at the table for consumption. As a user of the robot, a person would normally want butter placed on the dining table. The person using the robot would have a piece of white bread with them at the table that the robot can identify.

Design

Our robot was designed entirely around size and value. We needed the robot to fit on a normal sized table, and we needed the robot to be relatively inexpensive. To accomplish this, we decided to base our robot off of two common parts: a Raspberry Pi and an Arduino board. We then decided to use standard dc motors, a camera that can be easily attached to the raspberry pi, and Legos to form the structural basis of the robot.

To train the model to detect bread, we used a training model through the TensorFlow platform. Multiple images (at least four hundred) are taken in as a dataset to be compiled into the training model. These images were gone through by hand and the bread was labeled in each photo. This gives the training help with identifying the size and shape of what the piece of bread might look like in an image. This process can be automated with scripts, however, we could not get the new APIs to accurately train a model this way. We then took the stock training script that comes with each of the tensorflow training models, and modified the parameters to what we thought would be best; parameters like number of objects, aspect ratio, and batch size. We based these decisions off internet searches and Tensorflow recommendations taking into account training model type and the size of our data. Next a script using Tensorflow API takes in two separate folders of images for training. The training folder that contains 75% of the images, and a testing folder that contains 25% of the images. Both are used to train the model however, after the training api trains using a certain amount of photos, then it checks if it can detect bread by looking in the testing images folder. This training loop is used, and the learning process of the images took about 24hrs. This speed could be rapidly improved, however the training was done

on a 2016 Macbook Pro with 4 cores and an integrated slow GPU. Package versions can be seen in the Environment table below.

From this training comes what is called a frozen Tensorflow training graph. This graph file contains all the training data learned from the training, along with a mapping file that maps certain objects to the data in the training graph. For us, the training graph only contained bread, so we only had one bread label that would be used in our mapping file. Next we had to convert the Tensorflow graph to a TensorflowLite graph. This has to be done to lighten the processing power needed to find the bread in images. This meant that the graph outputted from the training would be compressed into a smaller graph, and therefore lose accuracy in bread detection. For this reason this was the hardest part about the image detection part of our project. Our team would be able to run and detect objects on the Macbook perfectly, then when compressed and put onto the pie it would fail to accurately detect the bread. Along with this, Tensorflow recently rewrote a lot of their APIs including the API to transform the Tensorflow graph into a TensorflowLite graph. Because of this and because we couldn't get the newest version of Tensorflow to function properly, we were using depreciated packages which made the process particularly hard. The files were then transferred to the Raspberry Pi. Finally, after learning the model, transferring data to the Raspberry Pi, and using a connected camera we could now detect the bread at an adequate accuracy rate.

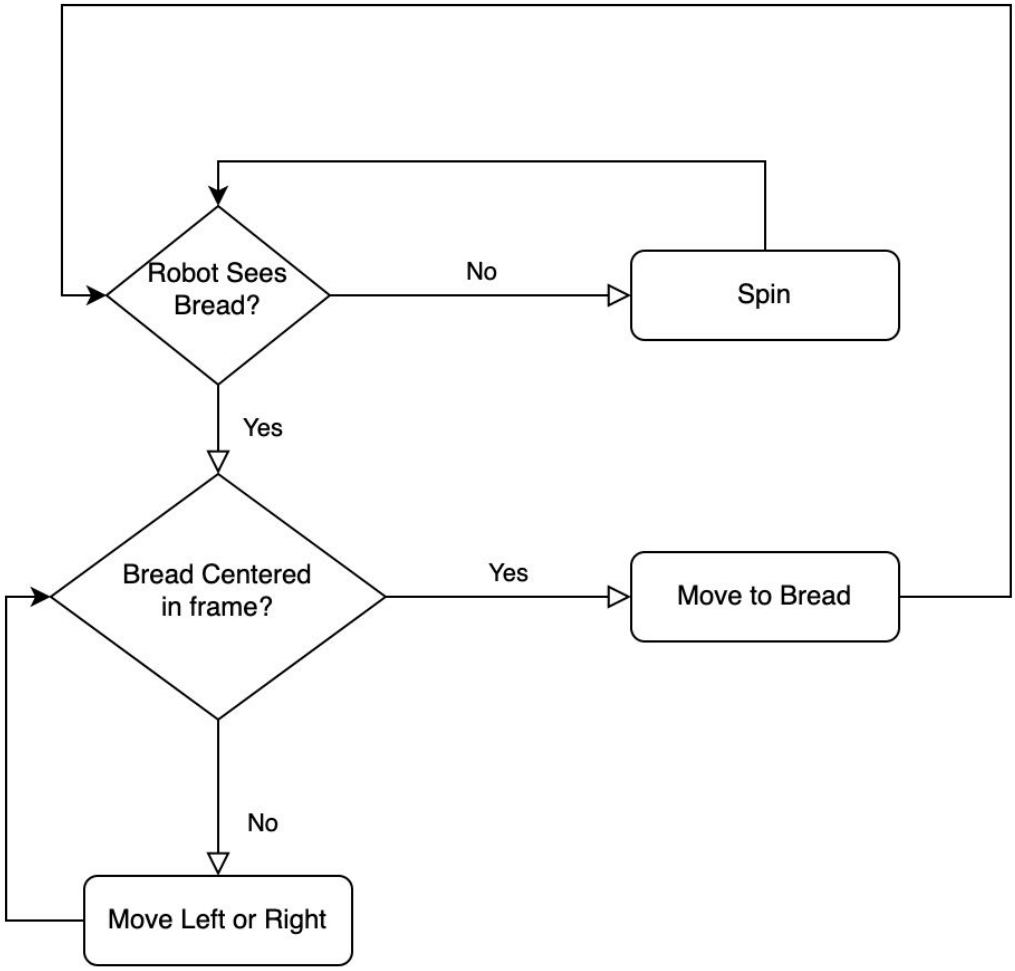
Environment

Package	**MacOs Version**	**Raspberry Pi Version**
Tensorflow	1.14.0	1.14.0
Python	2.7.2	2.7.2
Opencv	2.4.9	2.4.9
virtualenv	16.7.9	16.7.9
matplotlib	3.2.1	3.2.1
MacOs	Catalina 10.15.1	
NOOBS (Raspberry Pi)	3.4.0	
Labellmg was used to draw the bounding boxes on the photos		

Decision Matrix

Decision Matrix					
Criteria					
Options	Price 9	Size 10	Power Consumption 5	Support 6	Totals
Raspberry Pi	7	9	5	10	31
Computer Motherboard	3	3	4	7	17
Raspberry Pi Camera	8	10	10	10	38
Amazon Camera	10	10	10	10	40
DC motors	8	8	8	8	32
AC motors	7	6	5	8	26
Legos	10	10	10	10	40
Steel/Metal Body	9	7	5	5	26
Arduino	8	7	5	9	29
TI msp432	8	7	7	4	26

State Machine



System Testing and Analysis

We broke the testing of our robot down into five categories: camera detection, distance sensing, body design, and motor response. We started with camera detection. Through taking multiple pictures and choosing different training models, we were able to observe live video feed from the Raspberry Pi on a monitor. We waved a piece of bread in front of the camera to see if the model would both run on the Pi and detect the piece of bread. Next, since we were already focusing on camera software, distance sensing naturally came to mind when testing. To do this, we compared the average size of a piece of bread in a single frame, and then we took this number to base the distance of how far or how small the bread looked in the frame compared to the average. We were then able to center the camera off measuring the corners of the detection box from the bottom of the frame.

Body design was relatively easy. Our team decided that with Legos, we could easily shape the body however we wanted which would be perfect for any sudden design changes. This meant we could stack things however we wanted and determine what size we wanted it to be. From here, we focused on motor response. We mounted the motors under our Lego chassis and connected them to the motor boards that are connected to the Arduino board. The team performed the same test of waving a piece of white bread in front of the Pi camera. Knowing that the detection and centering was working through images, we then had to tweak the signals being sent to the motors so they would react to distance and move according to the desired behavior.

Conclusion

The robot was a success. The robot created can accurately detect a piece of white bread on most backgrounds, then can move to where the piece of bread by being able to judge the distance by itself. By having the camera detect bread and distance we removed the problem of video and sensor feed being out of sync. To extend the project, we would create an imaging model that identifies other things on the table so that it can move around them, but still make it to the bread. There could also be improvements with how the robot detects bread. For example you could have the camera memorize who already has had their bread buttered.

Teaming

Team management was very easy for us. Since it was a relatively small team of two people the tasks were fairly easy to divide up. Since we are fourth year computer engineers we both had an understanding of modulating code for reuse, and a good understanding of how hardware components fit together and mix with the software. Michael did a lot of the Tensorflow training, while David did a lot of the Arduino and hardware components, however both people worked on everything in the project because we worked right next to each other for most of the project.

Reflection

Both of us learned just how much computational power it takes for image processing in real time. One of the biggest hurdles for our team was getting the image processing scripts to work on a Raspberry Pi. Even with all of our adjustments we still only get about 4 frames per second. Just thinking of the cameras scientists leave in the wilderness to help detect and count animal movements is mind boggling. The work and thought that must go into those contraptions must be enormous. The second biggest take-away from our project would be the power consumption of modern devices. Once again we can see how important it is to write efficient code for a device. Even with no screen the motors and board we used consumed a lot more power than we first believed. However, now that we have completed the project we believe we are both better engineers.

Appendices

Code

https://github.com/hegglinmichael/SeniorProject

Personas

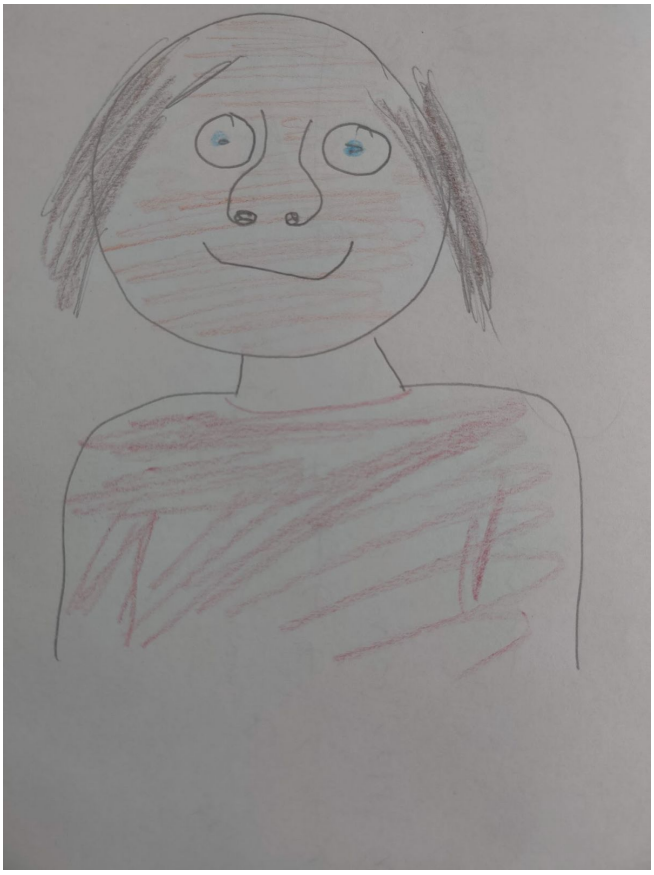


Figure 1. Persona of Larry Lobster

Figure 2. Persona of Bertha

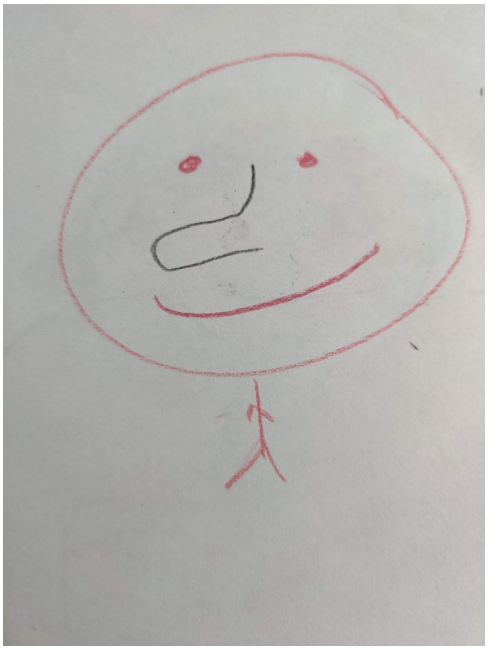


Figure 3. Persona of Lucas

Analysis of Senior Project Design

Please provide the following information regarding your Senior Project and submit to your advisor along with your final report. Attach additional sheets for your responses to the questions below.

Project Title: _____

_____ Quarter / Year Submitted: _____ Student:

(Print Name) _____ (Sign) _____

Advisor: (Print Name) _____ (Initial) _____ Date: _____

Functional Requirements

Our butter robot identifies a piece of white bread using object image detection then moves to the user with butter in store. Its movement is accomplished through dc motors, image detection software on a Raspberry Pi, and an Arduino board.

Primary Constraints

The main limiting factor in this project was the computational power of our Raspberry Pi. We can barely run tensorflow on the board without it freezing up. We really had to tweak the training scripts and training models to make the software runnable on the pie. Most of the light weight models meant for devices with low computational power would not work. Both members of our team had also not built a robot before, or used image detection software. Because of this,

we went into the project pretty open minded, and decided to choose parts that had a lot of support.

Economic

Original Estimated Cost

Item	Price
Raspberry Pi	99.99
Camera	26.99
Legos	29.99
Ultrasonic Sensors	9.59
DC motors	14.59
Arduino Mega	31.8
Micro Servo Motor	12.99
Total	225.94

Actual Final Cost

Item	Price
Motor Drivers	13.99
Raspberry Pi	99.99
Camera	26.99
Legos	29.99
Ultrasonic Sensors	9.59
DC motors	29.18
Arduino Mega	31.8
Micro Servo Motor	12.99
Total	254.52

Original Estimated Development Time

Week	Due Date
1	Initial Meeting
2	Schedule + photos
3	Bread Identification Early Prototype
4	build initial frame with legos + look at servos and arduinos
5	;
6	raspberry pi in frame with camera mounted
7	look at and purchase arduinos + servos + motors + sensors
8	start writing arduino code for servos and motors
9	continue writing arduino code for servos and motors
10	Make sure everything is done that we wanted to accomplish
11	Finals Week no work going on
12	Spring Break
13	figure out how to make robot get bread distance/arduino to pi communication
14	mounting stuff on robot (servos, motors, etc)
15	figure out how to make robot go to bread
16	coding robot to bread
17	catch up on unfinished work
18	make arm to cut butter
19	test robot + fix errors
20	test robot + fix errors

Actual Development Time

Week	Due Date
1	Initial Meeting
2	Schedule + photos

3	Bread Identification Early Prototype
4	Bread Identification Early Prototype
5	Bread Identification Early Prototype
6	Bread Identification Early Prototype
7	look at and purchase arduinos + servos + motors + sensors
8	start writing arduino code for servos and motors
9	continue writing arduino code for servos and motors
10	Make sure everything is done that we wanted to accomplish
11	Finals Week no work going on
12	Spring Break
13	figure out how to make robot get bread distance/arduino to pi communication
14	figure out how to make robot get bread distance/arduino to pi communication
15	mounting stuff on robot (servos, motors, etc)
16	mounting stuff on robot (servos, motors, etc)
17	mounting stuff on robot (servos, motors, etc)
18	test robot + fix errors
19	test robot + fix errors
20	test robot + fix errors

Commercial Basis

Commercial Basis	Cost in USD
Manufacturing Cost	100
Devices Sold/Year	1000000
Purchase Price	250
Profit/Year	150000000
User Cost/Year	1

We believe that using overseas manufacturing we can get costs down to $100 by looking at component cost for how much money it takes a company to make an arduino and Raspberry Pi. While everyone likes butter and this could be in millions of households, the current bugs probably limit the amount of people we will attract, so we played it safe with a million person limit. Finally we looked at the cost of electricity it takes to run your phone for a year. Most sources said it takes slightly less than a dollar to do so, and we believe our robot is comparable to a phone for yearly cost.

Environmental

The only environmental impact from our device would be the electrical and plastic parts that our robot is made of. Because of these parts we believe it would be equivalent to the environmental impact of cell phones if everyone were to buy it.

Manufacturability

The hardest part about manufacturing a robot would be putting all the components on the robot body perfectly, and not making the robot too heavy for the user. We would have to replace the lego body with a special plastic frame that we would have to make molds for.

Sustainability

The biggest maintenance with our robot is having to restart the Raspberry Pi. The pie will eventually need to be restarted, and then you would have to bring back up the image detection software running on it. For the average user this could be a big problem.

Our project could also use a lot of upgrades. For example, having a specially made plastic mold for the body instead of legos would create a better looking device that people might be more willing to put on their tables. Another upgrade would be allowing the camera to detect other objects in the field of view, this means the robot could dodge the objects that aren't butter instead of plowing through them. The last change I would make would be having a charging station. This means that a user would never have to replace the batteries in their robot.

Ethical

Any ethical problems that could possibly arise would only be around the robot instead of from the robot. The robot's only purpose is to identify and butter bread, so it is not and should not connect to any network, and the robot is confined to only whatever surface on which the user places it.

Health and Safety

The only way this robot would up you in any danger is if you now stopped walking because anything could be brought to you as long as you have strategically placed pieces of white bread.

Social and Political

This won't have any social or political side effects. It is a small butter robot.

Development

We believe that the both of us learned a lot about image detection software during this project; especially with python and Tensorflow. All image detection was done via python image detection libraries and all detection models were from Tensorflow. There are a lot of different models that can be used to train with. All of these models have different training parameters and ways they train the model. This meant that a lot of time was spent going over which model would be best for light weight use, and which parameters to tweak to make the bread more detectable.

The next thing that was learned was the building of a robot. Both of us did not have any robotics experience before starting this project. This meant that we never had built a robot body,

or used the Arduino. A lot of thought was put into how we could easily build the robot so that we could easily make changes quickly, but still give the robot enough strength to not break during use.