XFM Vending Machine

Joseph Olin
Michael Roark
Branden Wagner

C458 Project Paper

May 2, 2017

Abstract

This project designed and implemented an autonomous robotic system that acts as a bar waiter. We used a robotic arm mounted on a mobile robot to bring consumables to customers who were sitting at the bar counter. The robot decided the delivery of consumables based on a color-coded card that each customer presented to the robot. We accomplished a robot control software architecture utilizing Arduino controlled sensors and effectors. Our project demonstrated how a familiar task, such as tending bar, can be handled by a low cost intelligent robot.

Introduction

Attending customers at a counter is a repetitive task. First, the waiter/waitress must ask the customer if they desire a consumable. Then, the waiter must retrieve the consumable from where it is stored. Once the consumable is retrieved, the waiter makes a pass over the length of the counter, giving customers the order they desired. One notable model for attending customers can be found at Brazeiros in Louisville, Kentucky.

At Brazeiros, customers are given a color-coded coaster, with one side being red and other side being green. Based on the color of the coaster the customer shows, the customer may or may not be continually served freshly prepared meat. Our project proposed to take the repetitive task of attending customers at a counter and make that task the responsibility of an intelligent robot rather than a human.

When designing this robot we were concerned with making a low-cost and flexible platform which could be expanded upon with ease. We chose to begin with the base from a previous project which utilized the rover to avoid simple obstacles using physicomimetics. We altered the design with a simpler, more cost-effective option that utilized a line follower sensor. This leOriginally, the arm took approximately 14.6 seconds to deliver a consumable. After further breaking down the movements the arm makes, unnecessary movements were cut out of the delivery process. This optimization improved the arm's delivery time from 14.6 seconds per delivery to approximately 9.3 seconds.d us to the choice of using a simple robotic arm to deliver the consumables.

Using these components we were able to make a simple and cost-effective robot which could perform in a wide set of configurations. We designed the environment to match a simple countertop delivery area, and measured the XFM's performance in successful deliveries. We designed tests to fine-tune performance of the arm and color sensor. These tests and the results are described in the sections below.

Method

Process

We employed a human to load the robot's magazine with consumables (mint patties). Customers would each have a double-sided card with a certain color (e.g. blue) on one side and a different color

(e.g. green) on the other side. This is similar to how restaurants such as Brazeiros operate, where one color indicates the customer wants an order and the other color indicates the customer does not desire an order.

The robot would make cycles through the counter, ignoring unoccupied spaces and customers who indicated they didn't desire an order. The robot's goal was to deliver items as quickly as possible only to customers who desired an order. Our operating model was based on this particular idea, and dictated how we constructed our environment.

Environment

The environment consisted solely of the counter, which was simulated with a long, thin table. There was a line for the robot to follow and there were cards for the customers to mark their order and red cards to mark the ends of the path where the robot would reverse direction.

The storage and loading area was on the robot's deck. Sitting on the deck was a plastic storage magazine, which contained up to eight mint patties.

The bot travel area was the space on the counter where the bot moved. The space stretched from one end of the counter to the other end. The bot would move forward from the far end of the counter following a predetermined, taped path. As the bot followed the path it read the green color cards which the customers laid on the counter alongside the tape, dropping off mint patties in the appropriate places. Once the bot reached a red card it traversed back over the taped path. We also used blue cards to serve as false orders which the robot was intended to ignore.

The customer area was the area closer to the edge of the counter where customers displayed their color-coded card to the bot, and where the bot also placed the customer's order.

Sensors

The sensors needed for the bot to perform the aforementioned tasks were: one IR reflectance sensor array and a Red Green Blue (RGB) color sensor.

The Infrared (IR) reflectance sensor array was for allowing the bot to follow the taped navigation path and for notifying the bot when it had reached the loading area (where the tape would end).

The color sensor was used to notify the bot what color card the customer was displaying, thus notifying the bot if it needed to give the customer a consumable.

Performance Metrics

Three primary performance metrics we used in this project were: the number of color cards the bot accurately detected, the number of successful deliveries the arm made once it knew a delivery

needed to be made, and the speed at which the arm delivered consumables.

System Functionalities

- Bot will move forward along the counter, following taped path using the line follower sensor
- Bot will recognize colored cards using the RGB sensor
- Bot will move back along the counter, following the taped path
- Arm will grab and hold on to a consumable
- Arm will unload a consumable onto the order card

State Diagram

Our robot's behavior was a reactive system, as shown in Figure 1. It ran a path-finding algorithm that depended on a strip of black tape to maintain its path. It moved along its path until it sensed red, indicating the end of its path. If it sensed the color green it stopped, and delivered a mint to the users.

System Architecture

Figure 2 outlines the robot's system architecture. The three core pieces of the system were the Arduino Mega ADK, Adafruit Motor shield, and Braccio Arm Shield. These three were stacked on top of each other with each lead of the Braccio slotting into the pins of the Adafruit Motor shield which then slot into the Arduino Mega ADK. This essentially made the boards function as one unit, but this has been expanded for clarity.

The Mega ADK had more analog and digital pins available in the front of the board. This allowed us to connect components to the board directly to improve the real estate taken up by the wiring, giving us more room for the arm to operate.

Every component with the exception of the Braccio arm in the Architecture was powered by the Arduino's 5V source, and grounded by the Arduino's GND. The arm was powered and grounded by the Braccio arm shield by use of the serial pins on the board. The Pololu Reflectance Array, and TCS3200 GBB Color Sensor input and output pins are all connected to the Arduino board. The Pololu LEDs are connected to the Arduino 5V. The TCS3200 GBB Color Sensor OE was grounded to the Arduino, and LEDs powered by the Arduino.

Results

The following data was recorded in the full testing environment using the initial version of the software. We observed that the color reading sensor failed to detect the correct color 3.7% of the time. The arm failed to deliver the mint at a rate of 7.3%. The estimated average error rate of this configuration was 5.0%.

Table 1. Test data for 2 Green Cards, 1 Blue Card

Passes	Missed color readings	Missed arm deliveries	Total color readings	Total arm deliveries	Color reading fail rate	Arm delivery fail rate
30	5	5	137	68	3.7%	7.3%

The following data was recorded in the full testing environment with the addition of two green cards and two blue cards. We used the initial version of the software.

We observed that the color reading sensor failed to detect the correct color 11.3% of the time. The arm failed to deliver the mint at a rate of 21.4%. The estimated average error rate of this configuration was 15.2%.

Table 2. Test data for 4 Green Cards, 2 Blue Cards.

Passes	Missed color readings	Missed arm deliveries	Total color readings	Total arm deliveries	Color reading fail rate	Arm delivery fail rate
30	25	29	221	135	11.3%	21.4%

The following data was recorded to demonstrate further optimizations performed upon the color sensor, without any operation of the arm.

The following data was recorded in the full testing environment with the addition of two green cards and two blue cards. We disabled the arm functions, and updated the code to include our Euclidean Radius Algorithm.

We observed that the color reading sensor failed to detect the correct color 0.95% of the time. This improved estimated average error rate over the original configuration roughly tenfold.

Table 3. Test data for 4 Green Cards, 2 Blue Cards (Color Sensor Only).

Passes	Missed color readings	Missed arm deliveries	Total color readings	Total arm deliveries	Color reading fail rate	Arm delivery fail rate
30	2	N/A	210	N/A	0.95%	N/A

We collected the following data by disabling all systems except the arm, and coding the arm to repeatedly execute the delivery motion. We measured the amount of time spent for the robot arm to complete each delivery motion, and kept track of the total time to complete thirty cycles. The optimized movement cycle showed 36.3% speed increase over the original.

Table 4. Test data for Braccio Arm Delivery Optimization.

Metric	Original	Optimized
Deliveries	30	30
Total Time (seconds)	439	280
Seconds Per Delivery	14.6	9.3

Discussion

Data Interpretation

Our study of the initial data we gathered indicated there was room for optimization within the robotic system. Each of the three points in our performance metrics had potential for improvement. However, we chose to perform optimizations upon only two of these points for the time being.

The color reading subsystem performed quite well for the first test at a failure rate of 3.7%. However, the second data set saw this rate jump to 11.3%. To improve the color detection failure rate, we tweaked our color reading algorithm so it would discard readings above a specified color radius. This dropped the color detection failure rate to 0.95%. Since we didn't operate the arm while testing this, the failure rate is likely lower than it would be with the system in full operation. However, it is still significantly better than it previously had been.

The second optimization we performed was improving the speed with which the arm delivered each individual food item. We observed the arm took approximately 14.6 seconds to deliver a consumable. After further breaking down the movements the arm makes, unnecessary movements were cut out of the delivery process. This optimization improved the arm's delivery time from 14.6 seconds per delivery to approximately 9.3 seconds per delivery, a 36.3% speed increase.

We didn't implement optimizations upon the arm's delivery functionality. It was observed that the robot's performance degraded significantly in the last several passes, which correlated with the fact that the robot's servo motors got increasingly warm as it continued to operate. Most noticeably, the arm appeared to drop more consumables as the gripper motor got warmer. This suggested that the issue

would have to be resolved using more robust hardware.

Additional Research

Our approach was to define the problem first, then design the robot. As the problem was refined, we also needed to alter the design. Some designs solving a similar problem include Zexuan, et al (2015), and Casavela (2012). Our design intended to combine movement of a wheeled base with the delivery capabilities of an arm into an autonomous delivery system for small solid objects.

We decided to use a color coding system for ordering based on observations of the system in use at Brazeiros. The Brazeiros system uses a green card to start service and a red card to stop service. (Brazeiros 2017) Initially we tuned the sensor to take raw readings from the color sensor and mapped them to an RGB scale of 0 to 255. These RGB values were the basis for our algorithm that took the readings and compared the values to near random values to determine if the color was red or green. Joseph Olin then implemented a Euclidean distance algorithm that treated the various predetermined color values as a point, and then compared them to the readings to determine the closest color value. We refined this algorithm further by setting an acceptable error, which was treated as a radius. This treated each predetermined color value as an origin point, and if the reading fell within the radius of the color it was accepted and the dispatcher dispatched commands.

Path planning is a difficult obstacle to overcome for autonomous robots. Zexuan, et al, (2015) showed an example the procedures needed to plan transportation along a set path. Our project used a combination of physical aids and programming to overcome obstacles in the path. We intended to use a fixed and relatively small path traversed in two directions, so we decided to explore another algorithm.

The algorithm we decided to use was based on code found for the Pololu Reflectance Sensor Array. The sensor array, after calibration, would return a value of 0 to 7000, and each number was used to dictate how hard the rover needed to move to the left or right. Initially our robot had stuttering issues when we used the code directly. Since our robot moves in a straight line we were able to cut back the speed at which the rover adjusted its path. This reduced the amount of stuttering that was present in our movement, and this helped to ease the robot's operation.

Casavela (2012) explains the challenges he overcame in using C++ to program an Arduino. We were able to draw many lessons to design the necessary programs to drive our systems. Casavela's paper also gave several insights on how to engineer robotic systems and properly interface them. This helped to assemble, and integrate the systems involved. This was especially helpful during our integration and system tests where we needed to take relatively sparse and highly decomposed problems, and bring them together to compose a solution to our original problem.

Conclusion

After implementing the XFM Vending Machine robotic system, we discovered how easily and affordably one might use an intelligent robot to perform simple, everyday activities. Although there was no previous robotics experience amongst any who worked on the robotic system, we succeeded in the getting the robot to fulfill its task within the specified performance metrics. Namely, the robot can read colors and deliver food with acceptable failure rates and do so at an acceptable speed. In doing so, we have demonstrated how such systems are relatively straightforward to create, even for those with little to no prior robotic experience. Even with that said, more optimizations might be performed upon the system, namely the arm, to further lower the delivery failure rate.

Appendices

Figure 1. XFM State Transition Diagram.

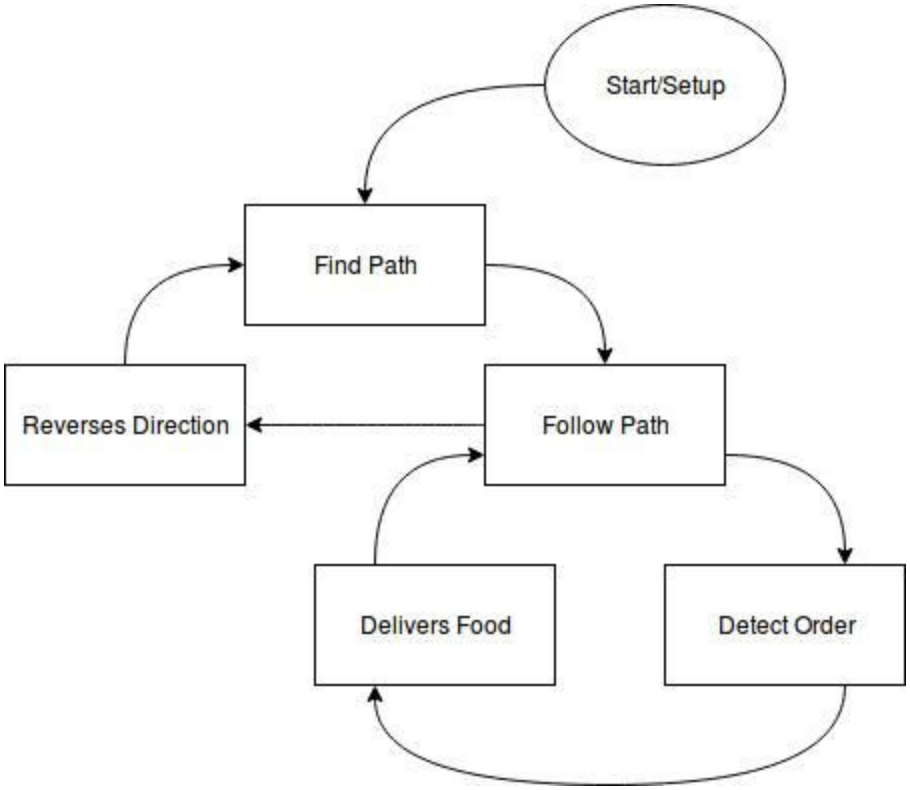
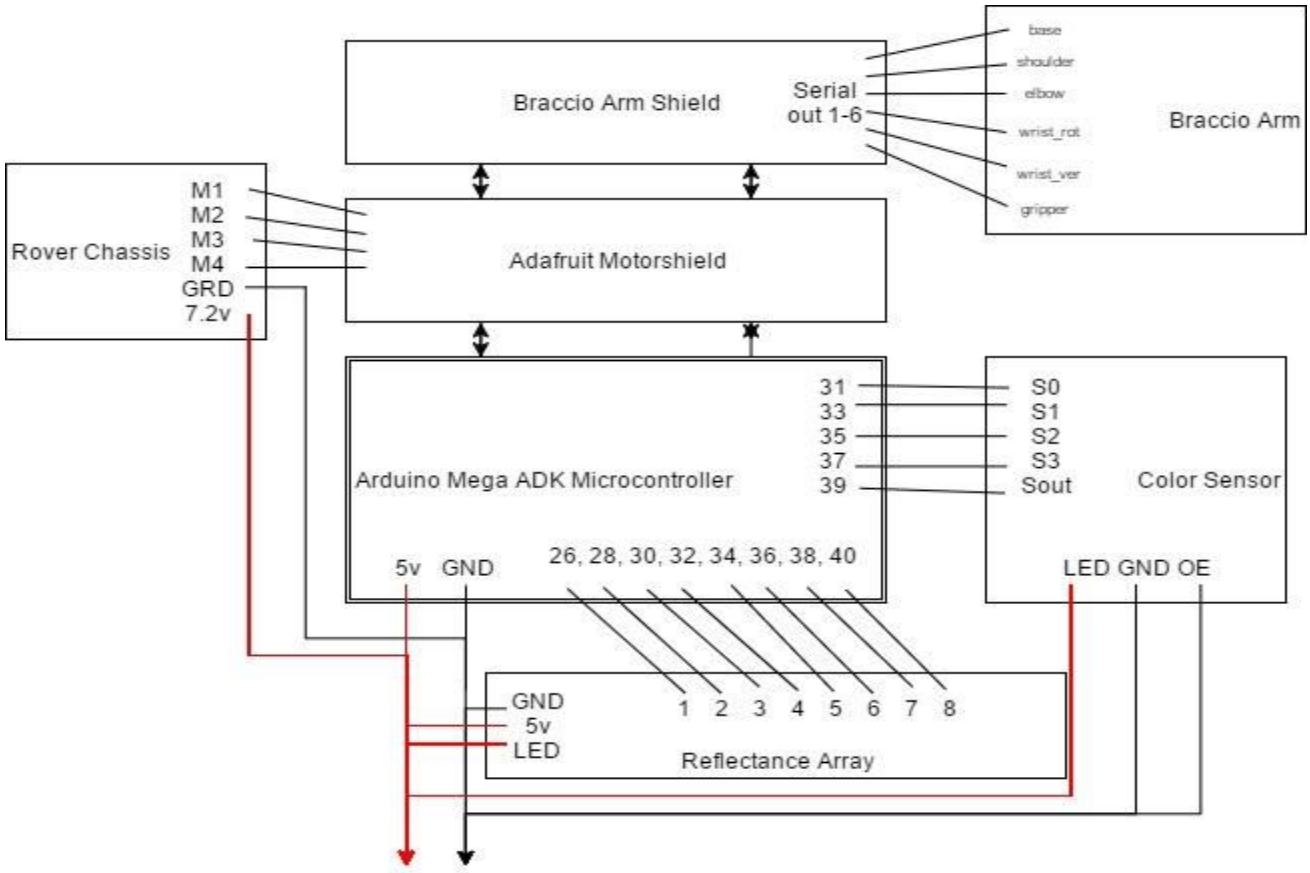


Figure 2. System Architecture Diagram.



Division of Labor

Version Control Manager – Branden Wagner

Bot Navigation – Michael Roark

Bot Color Reading – Joseph Olin

Bot Loading and Unloading – Branden Wagner and Michael Roark

Presentation Preparer – Joseph Olin and Branden Wagner

Development Timeline

Physical Development

Task	Approximate Date Completed (2017)	Description
Assemble arm	March 3	The arm came in a kit and had to be assembled
Test wheel motors	March 14	The robot motors had to be tested to ensure all were working properly
Setup rover platform	March 14	A sturdier, plywood platform was required on top of the chassis so the arm could be mounted
Mount arm on platform	March 14	The arm was secured with screws to the wood platform
Wire everything together	March 21	The wired circuit for getting power to all the servos was completed
Mount color sensor	March 21	The color sensor was mounted to the underside of the chassis using screws and metal brackets
Mount line follower sensor	April 7	The line follower was mounted to the front underside of the chassis using screws, metal brackets, and hot glue

Software Development

Task	**Approximate Date Completed (2017)**	**Description**
Arm movement	March 9	Write code to enable the arm to grab consumables
Bot navigation	March 9	Write code to make the bot drive around via its wheels
Move bot and operate arm	March 14	Write code to ensure the bot can both operate the arm and drive around
Tune color sensor	April 7	Tune the sensor inputs to get more accurate sensor output. Then, map the sensor output to RGB values
Integrate the 3 components	April 14	Write code that ensures the robot can operate the arm as desired, drive on the table, and read colors
Tune line follower sensor	April 28	Tune the line follower sensor so the robot follows the taped line on the table
Integrate all 4 components	April 30	Ensure the robot can drive on the table (following the line), operate the arm, and read colors

References

Braccio. (n.d.). Retrieved May 02, 2017, from http://www.arduino.org/products/tinkerkit/arduino-tinkerkit-braccio

Brazeiros . (2017). Meats. Retrieved February 16, 2017, from http://www.brazeiros.com/menu/meats/

Casavela, S. (2012). C++ PROGRAM FOR DRIVING OF AN AGRICOL ROBOT. Annals Of The University Of Petrosani Mechanical Engineering, 1411-19.

Ju, B. (2016). Robotic gripper can throw darts, balls - with no arm motion | Cornell Chronicle. Retrieved February 16, 2017, from http://www.news.cornell.edu/stories/2012/02/robotic-gripper-can-now-throw-things

Pololu. (2014). QTR-8A Reflectance Sensor Array: QTR-8A and QTR-8RC Reflectance Sensor Array User's Guide. Pololu Corporation. Retrieved May 2, from https://www.pololu.com/docs/pdf/0J12/QTR-8x.pdf

SparkFun. (2010). Rover 5 Chassis : Rover 5. SparkFun Retrieved May 2, 2017, from https://www.sparkfun.com/products/10336

TCS3200 Color Sensor (SKU:SEN0101). (n.d.). Retrieved May 02, 2017, from https://www.dfrobot.com/wiki/index.php/TCS3200_Color_Sensor_(SKU:SEN0101)

Zexuan, Z., Jun, X., Jian-Qiang, L., Fangxiao, W., & Qingfu, Z. (2015). Global path planning of wheeled robots using multi-objective memetic algorithms. Integrated Computer-Aided Engineering, 22(4), 387-404. doi:10.3233/ICA-150498