Capacitive Touch Augmentative and Alternative Communication (AAC) for dogs

https://github.com/WilsonRL/FinalProject/tree/**Finished**

Rebecca Wilson, McIntosh Webber

Project summary

In 2018, a speech-language pathologist taught her puppy to communicate by pressing buttons with prerecorded words, including “outside,” “play,” and “water.” Her dog currently uses over 45 different buttons in sequences of up to five words to communicate [1]. Many dog (and cat) owners have now taught their pets to communicate with recordable buttons. Cognitive scientists are currently investigating the language capacity of non-human animals using recordable buttons and video recordings of the animals using the buttons [2]. Our project aims to create a capacitive touch-based augmentative and alternative communication (AAC) system for dogs and their owners that tracks the learner’s progress.

Goals and objectives

There are several disadvantages to the currently available AAC systems for training dogs: (1) the buttons are large and take up a large amount of floor space, (2) the buttons can be difficult for smaller pets to press, (3) it can be difficult for the owner to hear the word being played if they're in a different room or it’s noisy and (4) there isn’t an automated way to track the button presses and learner’s progress.

Our final project aims to address these issues by creating a capacitive touch-based AAC system that can be used to train dogs to communicate and track their learning progress. More specifically, we plan to use Adafruit’s capacitive touch sensor breakout MPR121 - STEMMA QT) to create 12 capacitive touch-based “buttons” that will each play a different word when touched. We will also track and graph the number of times each word is pressed in total and per day.

GPIO goals

We **cleverly** incorporated GPIO into our project by connecting a 12-channel capacitive touch sensor breakout (MPR121 - STEMMA QT) and a STEMMA Speaker to a Feather (RP2040) equipped with an AirLift FeatherWing ESP32 WiFi CoProcessor (see figure 1). We also added a button that downloads new mp3 files when pressed. The Feather is powered by a 3.7 V 400 mA lithium ion battery. We will refer to this wireless capacitive touch system as the “AAC board.”

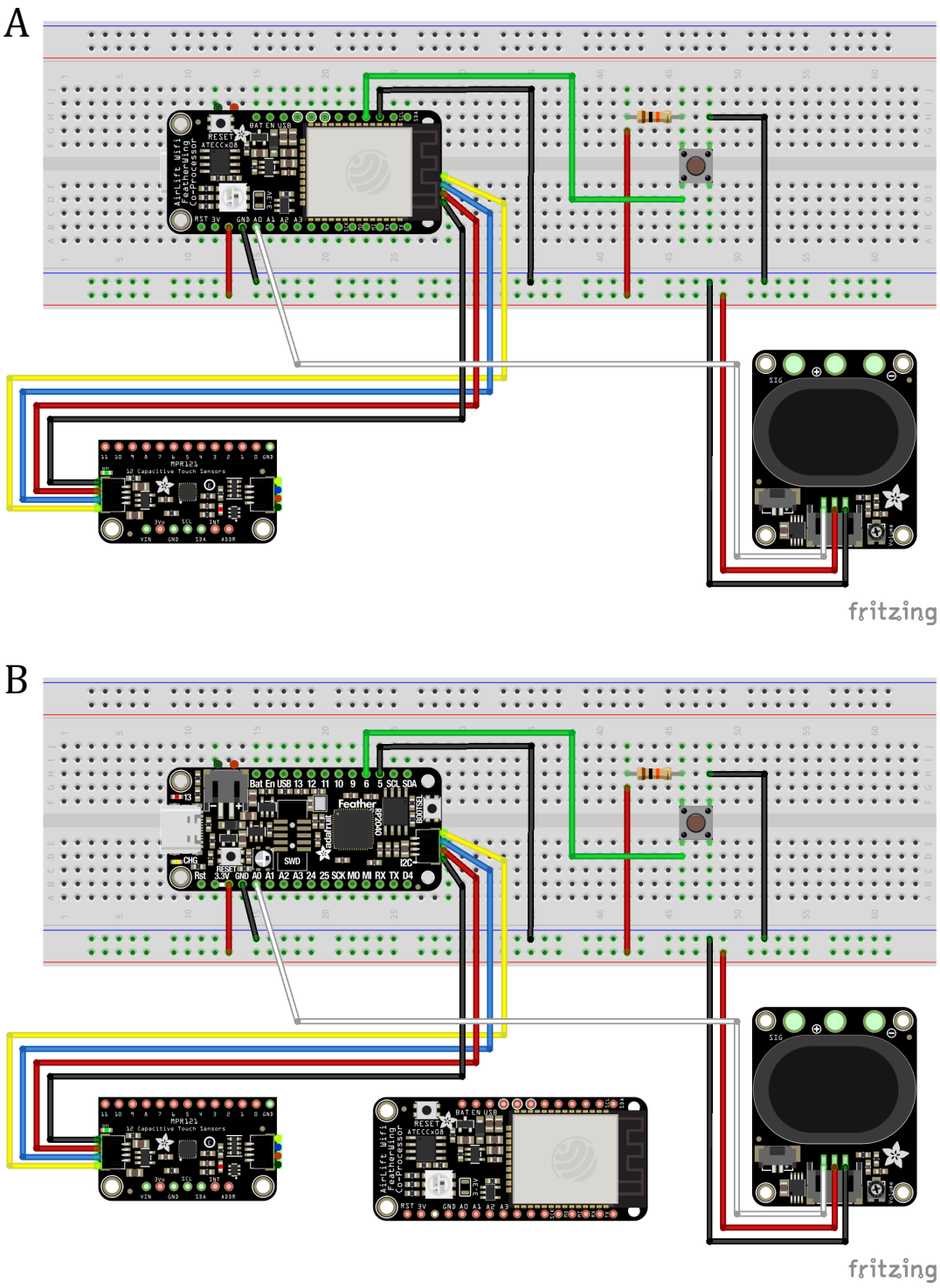
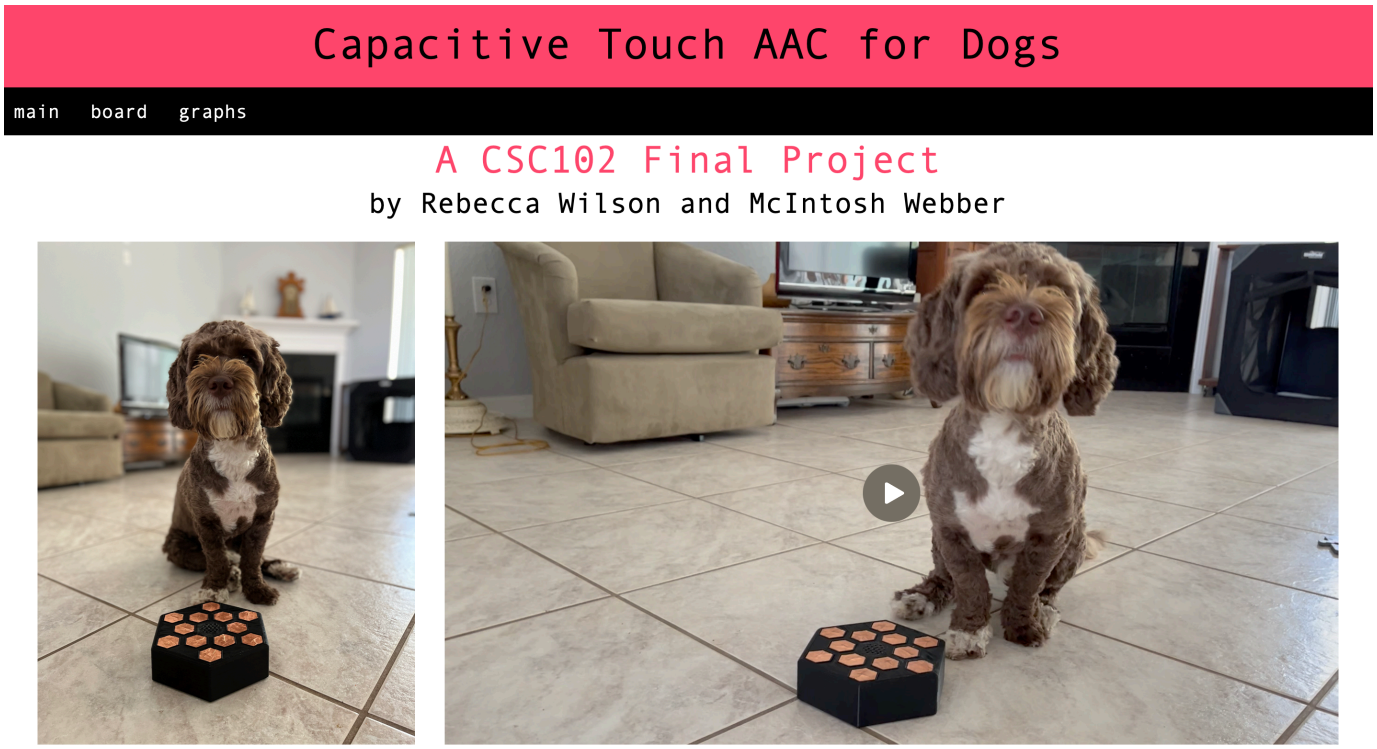


Figure 1. Fritzing diagram of AAC board. A 12-channel capacitive touch sensor breakout (MPR121 - STEMMA QT) and a STEMMA Speaker are attached to a Feather (RP2040) equipped with (A) and without (B) an AirLift FeatherWing ESP32 WiFi CoProcessor. The Feather is powered by a 3.7 V 400 mA lithium ion battery. A button is connected to the Feather at pin D6 that downloads new mp3 files when pressed. The Feather D5 pin is connected to ground to allow wireless file downloads. This wire needs to be disconnected in order to download files via Mu on the computer.

GUI goals

We created a graphical user interface (GUI) using the web application framework Python Flask, HTML, and CSS. The main page shows the AAC board and a video of the board in use (Figure 2A). A second page (Figure 2B) allows users to set up their AAC board by entering a word/phrase for each of the twelve capacitive touchpads. The website/server then uses the text to generate an mp3 file for each word using the python gtts library and makes the files available for download by the AAC board. The AAC board includes a button to download the mp3 files. Each time a capacitive touchpad is touched on the AAC board, it uploads the associated word to a feed we created on Adafruit's IO cloud service. The GUI can then fetch the data (button presses) from the feed, count the number of times each button was pressed, and display that information as a bar graph (Figure2C and D).

A



B

enter words to setup board

main board graphs

Button 00:
enter word for button 0

Button 01:
enter word for button 1

Button 02:
enter word for button 2

Button 03:
enter word for button 3

Button 04:
enter word for button 4

Button 05:
enter word for button 5

Button 06:
enter word for button 6

Button 07:
enter word for button 7

Button 08:
enter word for button 8

Button 09:
enter word for button 9

Button 10:
enter word for button 10

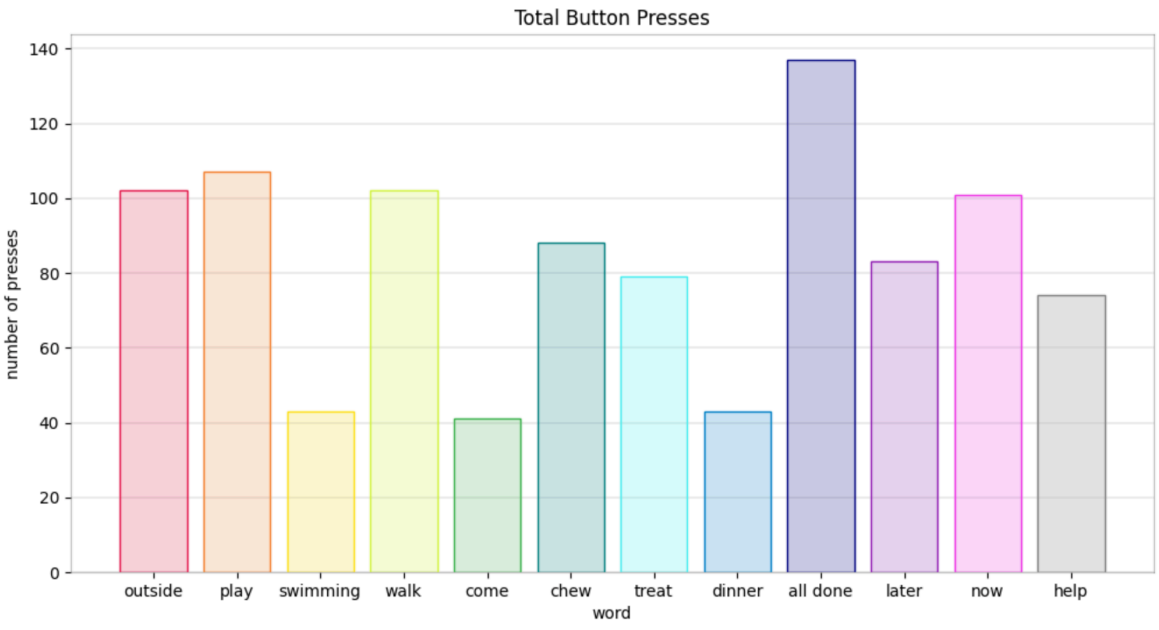
Button 11:
enter word for button 11

Submit

C

Data Tracking

main board graphs



D

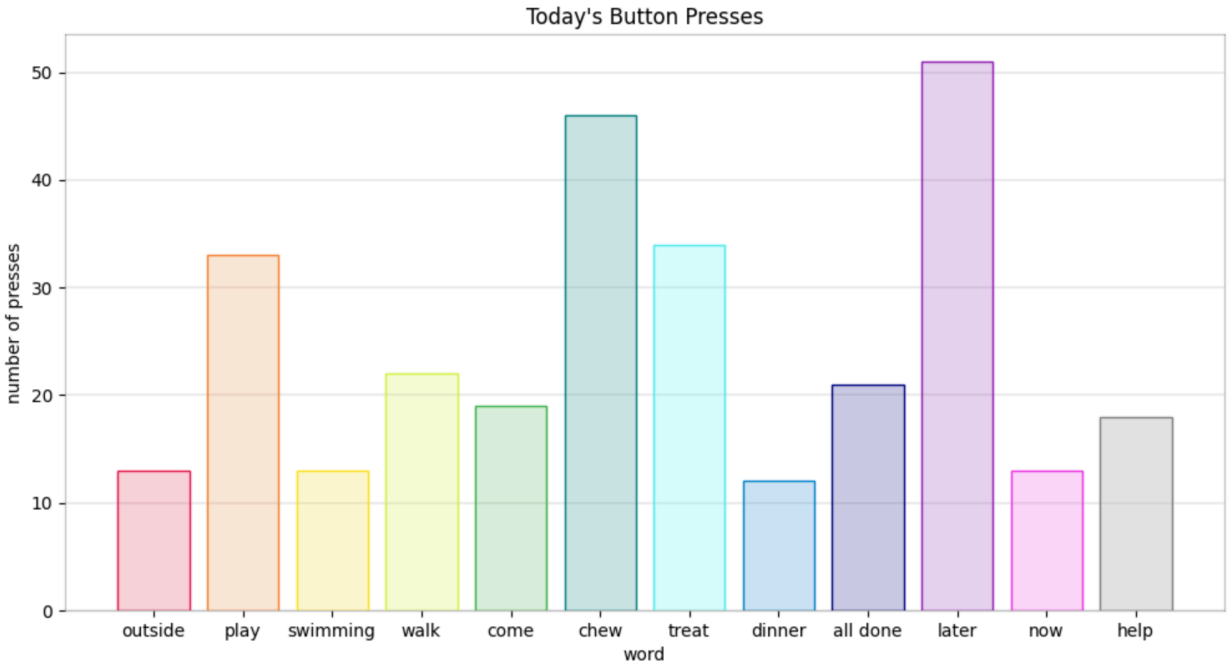


Figure 2. Screenshots of GUI. Screenshots of the graphical user interface created in Python Flask are shown. (A) Main page showing AAC board and video of it in use. (B) Form to enter words that are converted to MP3 files and made available for download by the AAC board. (C) Bar graph showing the total times each word was pressed. (D) Bar graph showing the number of times a word was pressed on the current day.

Timeline

Project Title	Week														Finals
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
Task 1 - Brief write-up (due Feb 9 - week 4)															
- get project idea approval															
- create write-up															
- order electronics															
Task 2 - preliminary prototype (due March 25 - week 9)															
- connect sensor to raspberry pi															
- create and add sound files to code															
- write code to analyze sensor touches															
- write code to display analysis on GUI															
- investigate google's text-to-speech as an alternative to sound files															
- add of wireless component															
- design and create housing for sensor															
- update write-up															
Task 3 - working prototype (due April 15 - week 12)															
- add light to indicate when mp3 files finish downloading															
- add graph that shows daily button presses															
- train dog to use sensor															
Task 4 - final run-through (due May 2 - week 14)															
- create and practice presentation															
Final project presentation															

Future development plans

If you were to continue working on this project, what would you do?

The biggest problem we encountered while testing our project was the durability and functionality of the physical board, so designing a functional enclosure to safely house the electronics would be our main focus if we were to continue this project. During testing, it was apparent that the touchpads need to be visually distinct, so the dog can easily see what they need to touch and asymmetric so the dog has a reference to know which button is being pressed.

Where could you go from here to make it better?

In addition to improving the design of the electronic's enclosure, we would like to either add a display on the AAC board that displays the last word or the last couple of words pressed and/or have the board send a text message of the word that was pressed.

What could be done to make it have increased broader impact?

For a broader audience to benefit from this project, an enclosure that safely houses the electronic components and has visually distinctive touchpads needs to be designed. If after redesigning the enclosure, it appears possible to train a dog to press the touchpads, we believe the next step would be to recreate the electronics portion using a different microcontroller. We would like to use the Raspberry Pi Zero 2 W which wasn't available when we started our project, but is less expensive than using the Feather RP2040 equipped with the WiFi Coprocessor. This would allow us to write the code associated with the AAC board in Python instead of CircuitPython. We think this would allow for a more direct transfer of data from the AAC board to the server/website for graphing.

Lessons learned

What did you learn by working on the project throughout the course?

We learned how to:

- create a website and server using Flask, Python, HTML, and CSS
- connect to the server to download files on a wireless device running CircuitPython
- assemble a wireless capacitive touch sensor
- detect and process input from the capacitive touch sensor with CircuitPython
- send and retrieve data from Adafruit's Internet-of-Things (IoT) cloud service
- 3D print an enclosure for the electronics

How did it relate to The Science of Computing curriculum?

This experience related to the Science of Computing curriculum in three main ways:

- programming

- creating a graphical user interface
- problem-solving
- wiring circuits

How was the experience beneficial to problem-solving in general?

This experience required us to problem-solve by researching solutions related to both choosing and wiring hardware and writing the program. This gave us the confidence to solve more difficult problems in the future.

What did you learn that will benefit you in future courses in the Computer Science curriculum?

The most important thing we learned from this project is that when designing a physical product, it's crucial to test the big-picture concept as soon as possible. We were so focused on the tiny details of the code that we didn't consider whether the dog would be able to recognize and distinguish between twelve symmetrical and flat touchpads. The physical buttons used by dogs are easy to see, and they feel the button being pressed. The capacitive touchpads don't provide the same sensory input when pressed, so it may be more difficult for the dog to learn. We could have tested this earlier in the project, but it didn't occur to us that it may be a problem. In the future, when we do a project, we'll be more thoughtful to keep the big-picture in mind and test all aspects of the project as soon as possible.

The second thing we learned from the project is not to rely too heavily on other people's products. The Saturday before our project was due, Adafruit made a change to their IoT cloud service that made the portion of our project that depends on it fail to work. It took a significant amount of time to discover the change they made and figure out how to fix it.

References

[1] https://www.hungerforwords.com/our-story/

[2] https://how.theycantalk.org/c/home/our-approach-to-research