Interactive Teddy Bear Report

Kenton Romero: kromero2001@tamu.edu

Adam Halldorsson: aehalldor@tamu.edu

Summary

In the age of technology, more and more children are becoming disillusioned from traditional toys and entertainment. The recent trend is a detrimental push toward children increasingly using smart devices while foregoing the experience of interacting with physical toys. The interactive teddy aims to bridge the gap between technology and traditional toys. This report will go in depth over the development of the first iteration of the Interactive Teddy Bear along with the challenges and feats faced in the project. The report will go over the physical hardware and how it is assembled into the bear. There is also a detailed explanation of how the code manages multiple stimuli to react accordingly. The software design will also explain the specific structures and functions used within the code, along with commentary on why certain structures are used and how they can be improved. There is also a journaling of our work on the project from initial brainstorming to completion. The progress log along with links to a demo video, presentation, and source code will be found on the end of this document.

Background & Introduction

Too often do we find ourselves seeing the youth of the day missing out on the simple joys we had with physical entertainment from our childhood. Many studies have shown how the prolonged use of computers can lead to shorter attention spans, depression, and hampered development. However, a transition is to be expected as our defining trait of humanity is the culture that we pass down and evolve in the process. Fortunately old habits die hard, and the teddy bear is a staple of American childhood. We want to bring this icon into the new age, through innovation in motion detection, impact detection, sound detection, and reactive light up eyes. The interactive teddy will allow children to have a friendly life-like toy with a futuristic aesthetic.

In current times, many of the sensors and computing chips in our design are incredibly ubiquitous, allowing for large scale production to lead to a very affordable alternative to the child’s iPad or Tablet. In the future, this technology could even connect to mobile devices, allowing more customization and opportunities for updates after initial purchase. Included

microtransactions may even be able to provide a diverse array of personalities for such bears. However, this product is an early prototype as a proof of concept. In this document, we will be exploring the development of the first iteration of the interactive teddy.

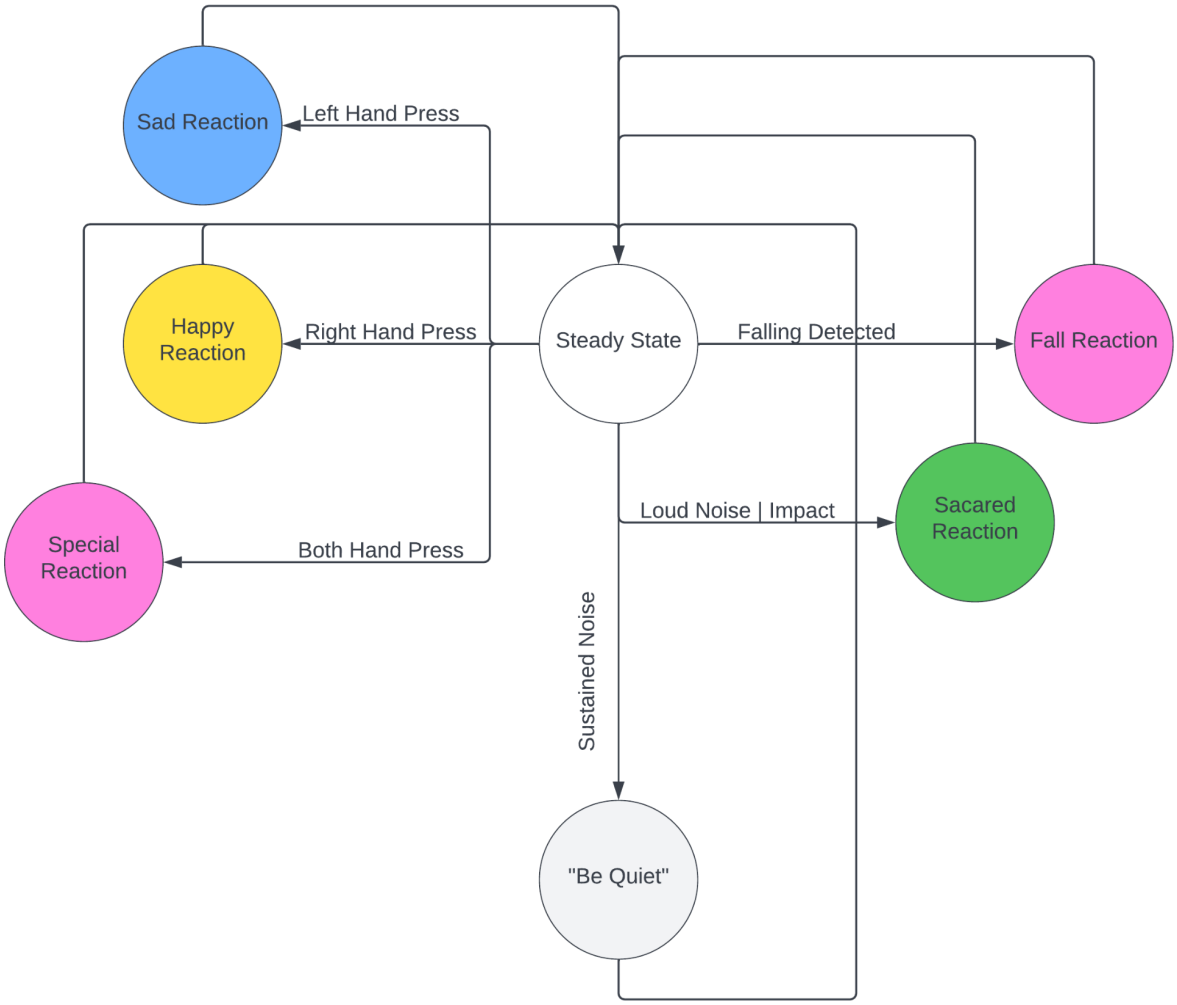
System Design

Hardware*:

- PCB
  - GPIO pin out and accelerometer soldered to main PCB connected above raspberry pi
  - Small PCB boards are used to connect some wire junctions
- RGB LED lights
  - Inside eyes of bear behind glass cabochons, fastened via hot glue, wired together onto a small PCB, then wired to main PCB
- Raspberry Pi 3
  - Main computer, all components connected to, located in bear body, covered in anti-static bag
- Accelerometer (MPU-6050)
  - Connected directly to main PCB, used to detect free fall
- 10,000 mWh battery
  - Used to power entire bear, located in bear body
- Hand-Buttons
  - A special button set up was made to fill the hands of bear and create an easily pressable button for users. Button soldered on PCB base, with a larger sewing button on top to widen the pressable area. Buttons wired through bear arms
- PortableSpeaker
  - connected to Pi via 3.5mm aux, sitting inside back of bear head, facing back, separate battery that can be charge by battery bank during operation
- USB Microphone
  - Connected via usb, used to get audio input, omni-directional mic, located inside body
- Teddy Bear
  - Approximately 45cm tall, 1.1 liter body volume, 0.6 liter head volume. Light Brown in color.

*images may be found in the Technical Steps sub-section

Software Description and System State Diagram:



The interactive teddy currently runs on python. This decision was a compromise of performance for less complexity in coding the bear's algorithm to interact with multiple stimuli. The algorithm can be modeled as a state machine, where the bear starts in the *Steady State* and performs different processes when specific stimuli are detected.

The *Steady State* is a single while loop, where the computer is continuously polling the current forces acting on the bear and triggers the passive, organic blinking of the bear's eyes (this "blinking" refers to the slightly irregular flicker of the white LED eyes while in the *Steady State*). The *Steady State* only triggers the Falling Detection. When the bear is in a free fall, an

if-statement will trigger a sequence for the bear to react. This is the *Fall Reaction* state, which will revert to the *Steady State* upon completion of the sequence.

The rest of the states will follow the same pattern of performing a process (typically playing a specific audio and a change in eye color) and return to the *Steady State* upon completion of sequence. Although the other states follow the same pattern, the way they are triggered are actuated by different means. The first of these is via interrupts and the second is via a separate thread coordinating with the *Steady State* through mutexes. This hodge podge of methods is not ideal in the sense that reactions can be overlapped, causing a previous reaction to be cut-off by another, but due to the nature of this system frequently switching states, the cut-offs are brief and feel organic as if the bear is reacting immediately to new stimulus as a living creature would.

The *Happy, Sad,* and *Special Reaction* states are all handled by a single handler that interrupts the *Steady State* when any of the hand-buttons are pressed on the bear. This handler then does a short polling session (lasting a fraction of a second in real-time) to determine if only one or both hand-buttons have been pressed. This polling allows for slightly dyssynchronous button pressing, as many people, especially children, would not be able to quickly press both buttons in unison. After determining what stimuli is being received, the handler will perform the sequence specified for each case: left-hand-press, right-hand-press, or both-hand-press. These sequences follow the same pattern of state-specific eye activity and audio.

The last two states, *Be Quiet* and *Scared Reaction*, need to activate in a different manner. These states rely on interpreting audio input over time, so a single thread is in charge of continuously interpreting the input from the microphone on the bear. Depending on the noise activity, the bear will react accordingly. For any loud or sudden noise, the bear will be startled (*Scared Reaction*). For a sustained period of elevated noise, the bear will request the environment to “be quiet” (*Be Quiet State*). While testing, the microphone also seemed to give the bear the ability to detect sudden impacts, such as a slap or small punch. This ability was also used to trigger the *Scared State*. Before these states are activated, a mutex is activated to let the main while

loop to not alter the eye state or play any audio. This communication is not handled between the interrupt handler and the audio input thread.

The final product code is bearfinal.py. Some of the functions declared are as follows:

- EyeState(color)
  - This function was created to easily change the color of the bear's eyes through color codes, such as 'off', 'RGB', 'RG', 'GR', 'RB', 'BG', 'R', 'B', etc. This is used very often especially in creating the eye blink animations.
- PlaySound(emotion)
  - This function was originally a dummy function to just print when an audio is supposed to change before implementing audio playback. However, this proved beneficial as this function was an ideal place to create code to choose a random audio reaction from a specific emotion playlist.
  - The emotion objects were 2D arrays of linked lists holding the path for a specific audio and the time it takes for the audio to play.
  - The function takes in a specified emotion object and chooses a random audio track, then plays the sound with a non-blocking command, and returns the time of the specific audio track
- Blink(t,color, color2 = 'null', off = 0.09, on = 1.75)
  - This function was created to facilitate short eye animations for reaction states, allowing for a broad combination of alternating eye colors, variable blink lengths, and variable total time of eye animation
- Falling()
  - Another originally dummy function to print when fall is detected. This function plays a sound, eye animation, and returns.

There are then two threads that manage audio input and the button presses. hand_handler(pin) is the function with the logic to detect button presses and play the corresponding emotion sequence. audio_callback (indata, frames, time, status) is the audio interpreter. This function is continuously tracking

the volume of the surrounding environment. The thread is activated with the ‘with stream:’ code block where the *Steady State* loop is residing.
Multiple cool downs were added to prevent repeat emotion states from what should be considered a single stimulus event. These are implemented by making the program unable to do the same emotion after a certain number of clock cycles, using counters and boolean variables.

Within every *Steady State* loop, the total force enacted on the bear over a moving average is calculated every cycle. This was originally done in order to interpret more advanced motions of the accelerometer, but during this phase, further implementation was not complete.

Passive blinking/flickering of eyes is controlled via a randomly generated timing of blinks to give a natural feel to the bear. This functionality is monitored in the *Steady State* and is temporarily disabled when an emotion state is in progress, with the exception of the “Be Quiet” state not having eye animations.

All sound files are found in the voices folder that is in the same directory as the bearfinal.py code. Testing code for troubleshooting and initial set up serve as the footer to the document.

Materials:

Parts	**Cost**
Raspberry Pi	Provided
Wide Buttons	$4.49
Wires	$3.47
Bright LEDs	$7.47
Portable Speaker	$14.56

Anti-static bags	$6.98
10,000 mWh battery bank	$11.44
Sewing Supplies	Provided
Teddy Bear	$5.00
Charging Cord	Provided
Auxiliary Cord For Speaker	Provided
PCB with Breadboard Layout	$14.56
MPU 6050 (Accelerometer)	Provided
40-pin GPIO Extension Board for Raspberry Pi 3	$7.89
Cabochons	$8.65
USB Microphone	$16.04
Total:	$100.55

Tentative Work Schedule:

Week	**Goal**	**Status**
March 28 - April 1	Acquire all parts	Completed
April 4 - 8	Connect all the pieces externally and start working on functionality	Completed
April 11 - 15 *30%*	Have all parts working	Completed

	externally except the microphone	
April 18 - 22	Record audio for voice, get microphone working, and start putting things inside the bear	Completed
April 25 - 29 *70%*	Test, refine, and finalize the prototype	Completed

Team Coordination Agreement

The following statements are agreed upon by all team members:

- We will be respectful to each other
- We will be on time
- We will be understanding of mistakes and disagreements
- We will give criticism constructively
- We will work together
- We will communicate frequently
- We will make time to work on the project together
- We will be careful with the hardware
- We will write good code that is understandable and efficient

Signatures:

Adam Halldorsson

Adam Halldorsson

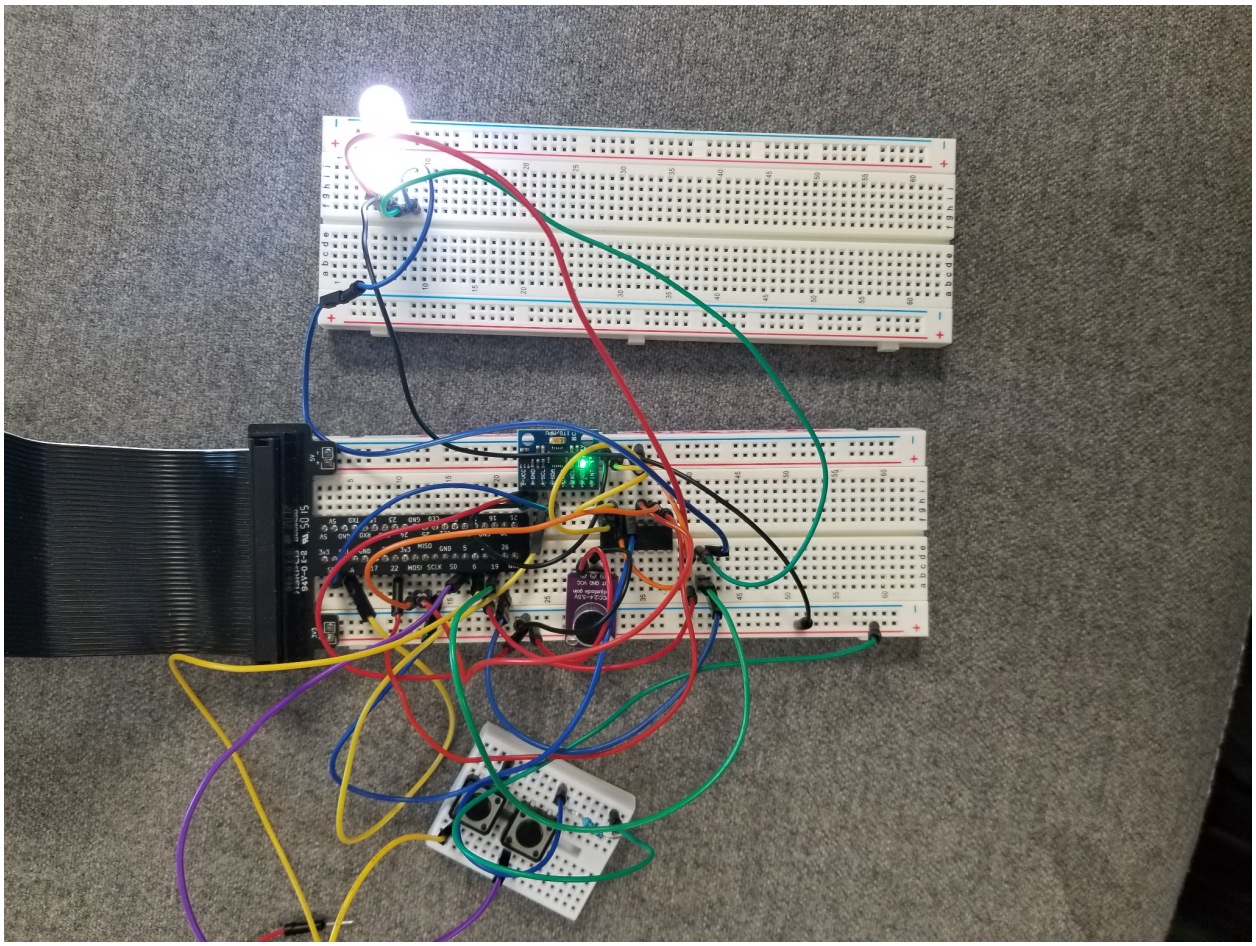
Kenton Romero

Kenton Romero

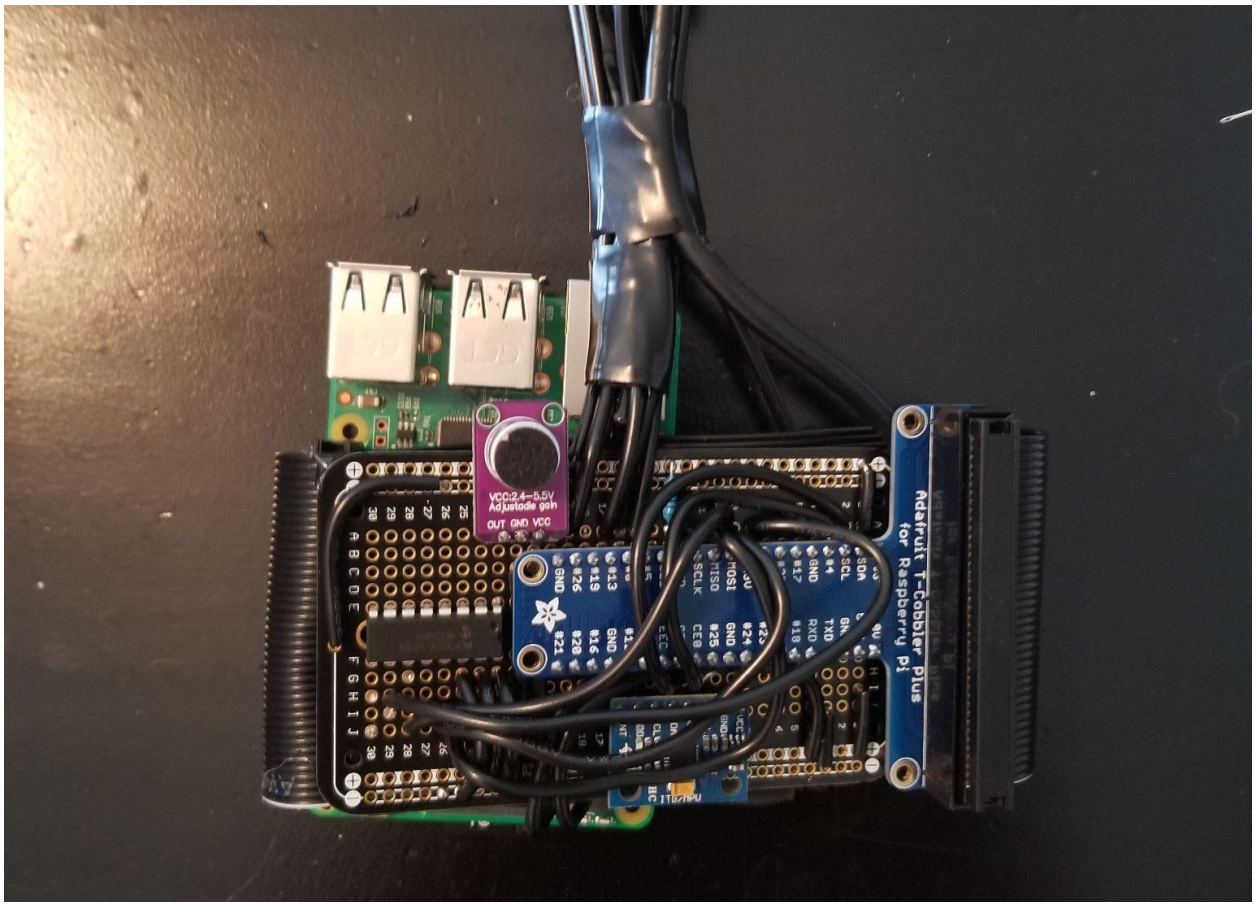
Technical Steps

The original features we had planned for the bear were light up eyes that change colors based on the interaction with the bear, buttons to interact with and output sound effects or voice lines with variation, speed detection, and sound detection.

We began with the eyes lighting up and changing colors in a sequence when buttons were interacted with. We also added the microphone, accelerometer, and ADC to the board to test that they worked but did not add any of their functionality at this point. The initial functionality of the bear was designed outside the bear to make testing and troubleshooting easier, as well as to avoid potential damage to the more flimsy wires and breadboard we used initially.

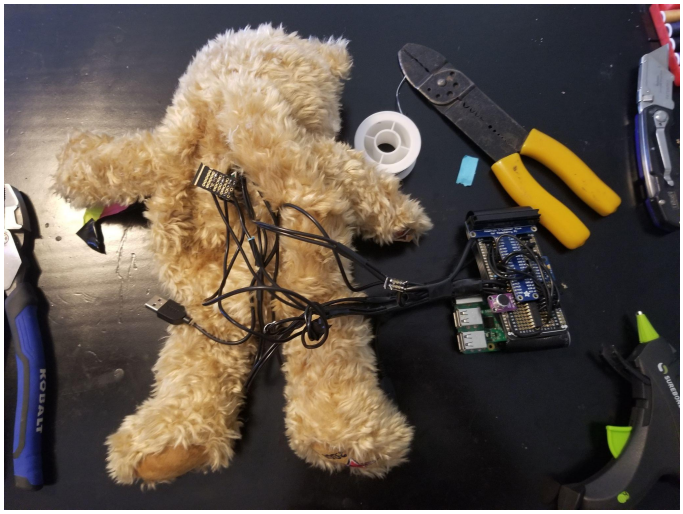


Our next step was to get the accelerometer to detect when the bear had been dropped or thrown. This was our first roadblock because we weren't able to throw around our flimsy breadboard without potentially breaking the components or wiring. Our solution was to purchase a solderable breadboard to more permanently house our components. We transferred the wiring from the basic breadboard to the solderable board and secured all the components in place with extended wires on the eye lights and buttons because they were to be placed away from the Pi and board. The solderable board was much more secure and compact and we felt more comfortable going forward with development.



The eyes were also hot glued to clear silicon cabochons to replace the eyes of the original teddy bear.

To test the accelerometer with drop testing we had to secure the parts inside the bear to cushion the impact. A problem we foresaw when planning was having the parts in the cotton balls of the bear could be dangerous with static electricity so we planned ahead and purchased antistatic bags to place the larger components in and used electrical tape to cover any amount of exposed wires. We removed the eyes on the bear and much of the stuffing to make room for our components. The new LED eyes were hot glued in place of the old eyes, buttons were placed in the bear's hands, and the Pi and board were placed in an antistatic bag before being placed in the torso of the bear. The hands were then sewn up to hide the buttons and a zipper was sewn onto the back of the bear to easily access the Pi and board.



With the components in the bear, we were able to perform drop tests on the bear and tune the accelerometer to be able to detect being dropped. Once the bear was able to detect drops, we added lighting effects and room for sound effects to be implemented later.

After the drop feature was completed we started adding the sound and voice effects to all previous parts. We used a voice changer to record voice lines that the bear would say when affected by stimuli. Sounds that showed joy were added to the right hand, sad sounds were added to the left hand, scared sounds were added to the drop effects, and special voice lines were added when both hands are pressed simultaneously. For the Pi to be able to play the sounds we used the pygame built-in library and a small speaker with aux connection that was placed in the head of the bear. The speaker had to be very compact to be able to fit in the head of the bear so the sounds could come from the ‘mouth’ of the bear and wired to be able to react to the stimuli the Pi receives as fast as possible. Once the speaker was working with our current iteration of the bear, it was placed in the head of the bear.

We then moved on to testing the microphone that we had soldered onto our board to detect loud noises so the bear could react but we ran into an issue. The microphone we had originally planned on using required additional libraries and a more advanced reading system than we had anticipated. We decided to buy an alternative microphone that was USB powered and plugged directly into the Pi. The USB mic was able to interface with python much easier than the original microphone and we were able to detect the decibel level of sounds in the bear's immediate environment. Using this data, we added scared voice lines to the bear when loud sounds were detected nearby as well as warnings from the bear to "be quiet" when long prolonged sounds are detected. After this code was implemented, the microphone was placed into the torso of the bear.

With all of the functionality of the bear completed, we now had to remove the wiring so that the bear was able to be a stand alone unit with no external wiring. We implemented a USB on/off cable to connect to a portable battery bank that was placed in the bear that would power the system when turned on. The next step was to make the Pi boot into the python script immediately when the power switch is flipped and the Pi boots on. Adding the script as a part of the boot sequence was more difficult than initially anticipated because the root directory did not detect the python environment for the script to read. Our solution to this problem was to delay the program start from the boot and hold the program until just before the boot was finished so that the python environment could be detected and the program could be launched by just turning the Pi on. All the external wires were removed and the bear is now a fully wireless standalone unit.

Final Outcome

The original goals for the first Interactive Teddy were ambitious, but with more experience and polish, this bear could be an affordable and exciting toy for any family to have in their household. Future ambitions include voice recognition, mobile-device connectivity, and a true childhood companion. It would almost be like an alexa in a toy. This first iteration proved the feasibility to integrate multiple sensors around the bear and create complex algorithms for the bear to interact in a multitude of ways in its environment. The process was not as straightforward as one would hope, with the original microphone design falling through and the first battery dying mid development, but the team withstood while succeeding in creating a flexible program that can manage multiple asynchronous inputs. The interactive teddy will soon be the standard above the simple teddy bear. As we smartify our TVs, homes, and cars, why not a teddy bear?

Link to project slide presentation:
https://docs.google.com/presentation/d/1aBJkHKzLyJ3t3sAy7a6x_MK2PkfA4xfjnr0s2rUGyh4/edit

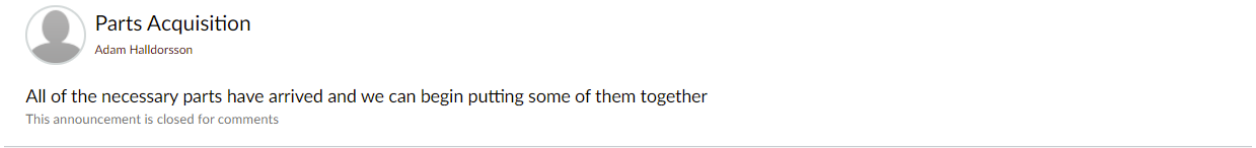
Link to demo video:
https://drive.google.com/file/d/13pumU9rHAdHNi9wx_uzHJ-dJ08TUc0SD/view?usp=drivesdk

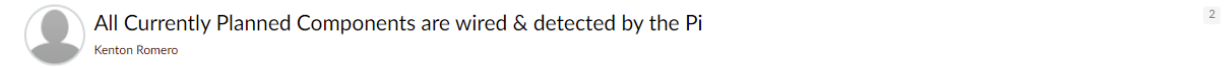
Link to source code:
https://drive.google.com/drive/folders/19-sKUktW-tKtBy7cOKO8j-AhyG15OCU-?usp=sharing

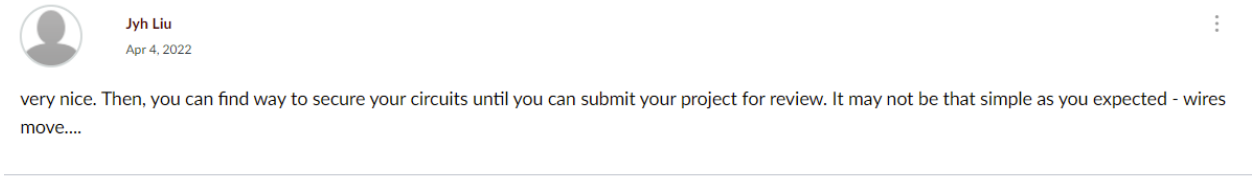
Appendix

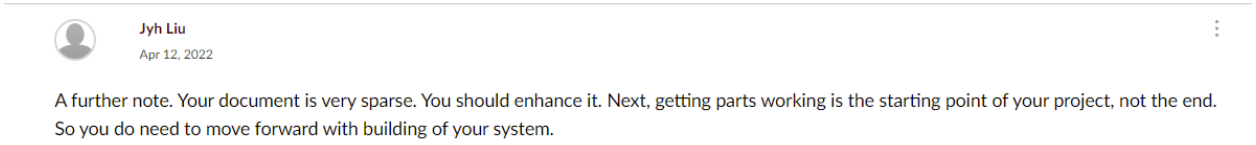
Pictures and videos are included in the report and not in the chat logs to avoid repetition.

April 1, 2022 Announcements

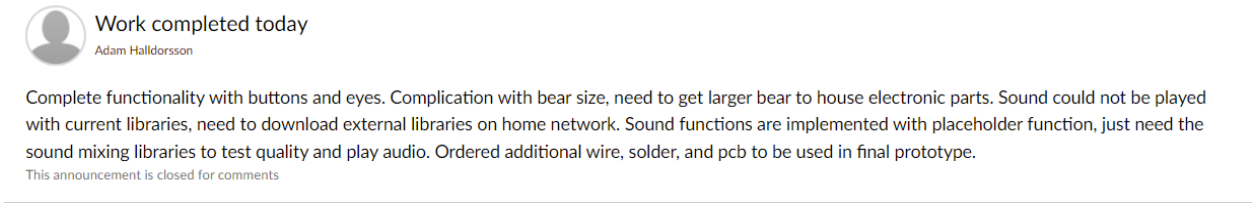




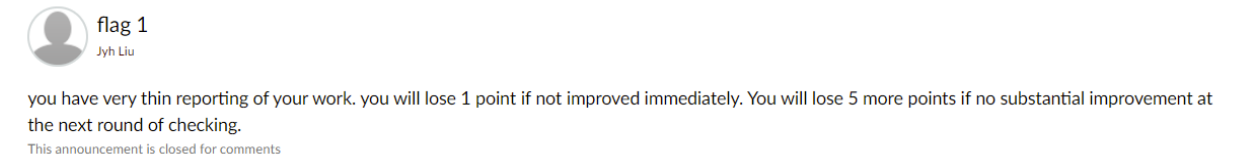




April 13, 2022 Announcement



April 18, 2022 Flag



April 19, 2022 Announcement

Current Code and Parts aquistion.

Kenton Romero

bear.py The code is currently working with use of the external prototype, minus fast motion detection and sound detection. All our custom functions are homed in the main bear.py file. We use multiple different libraries which are installed onto the Raspberry Pi which are found in all the imports at the head of the file.

All parts have arrived, with the new parts listed as follows:

- solder
- 25 ft of 24g wire
- PCB breadboard
- Raspberry pi GPIO pinout PCB

These parts will be used to build a stronger structure of wired electronics along with cleaning up clutter.

This announcement is closed for comments

April 21, 2022 Announcement

Moving breadboard prototype to more permanent board

Adam Halldorsson

Previously we were doing the original testing on a breadboard with everything loosely wired.

We have moved all the wiring over to a solderable board and neatly soldered all of the components into place and have full functionality.

The pi also boots directly into the program so there is no need for user interaction in the OS.

Final steps are to move the components into the housing and secure them to their permanent positions in the bear.

April 25, 2022 Review

Review feedback

Jyh Liu

4/25 So far, I only saw some breadboard hookups, and your statements about something is done. The python code is of course necessary for the control, but it is not possible to assert the progress with some lines of codes, simply because you are building a physical system. As such, and since you have been flagged once, you are losing 2 points from your final grade. You will lose 3 more points if the situation does not improve in next round of checking, or at final project review time.

This announcement is closed for comments

April 25, 2022 Announcement

Bear Assembled

Kenton Romero

Sunday Evening, we worked on assembling the bear together. The bear was successfully wired together, including eyes, hand buttons, microphone, speaker, battery, power switch, and the Pi. The speaker was inserted in the back of the head and the buttons are on the opened slits on the bear's hands. The next steps are to clean up the sewing and insert a zipper in the back of the bear. I showed in class the code working outside of the bear prior to bear assembly. Working bear demo will be submitted tonight.

April 26, 2022 Announcement

Bear Functionality Tested in bear

Kenton Romero

Today, we have closed all the openings in bear. This includes sewing the hands shut with the buttons inside and sewing a zipper on the back of the bear. Majority of our code has already been complete prior to physically assembling the bear, so we included a short video of the functionality. The functions tested are as follows: Eye reactions, sound reactions, left hand press, right hand press, both hand press, and fall detection. The yellow wire is an ethernet from the pi to our laptop in order to troubleshoot, and the black wire is for power (this is only temporary since our battery broke today). The next functionality that must be implemented include thrash detection, fast movement, and loud noise detection.

Some obstacles encountered today include the microphone not working as expected and the battery bank dying. A new microphone and battery bank will be bought tomorrow and included in the bear. This will allow the bear to be stand alone, reacting to various stimuli.

April 28, 2022 Announcements

Purchased new battery and microphone

Kenton Romero

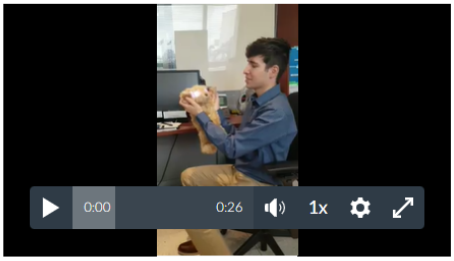
microphone

A new battery was purchased along with a new microphone, since the first implementation did not work as planned. The pi was set up to detect sound from the microphone successfully.

This announcement is closed for comments

Demo of Battery Installed and Microphone Integrated

Kenton Romero



The microphone and battery is working. The final prototype is complete. Next steps is reporting.

This announcement is closed for comments