Component Analysis

Year: 2022 Semester: Fall Team: 12 Project: R.A.C.H.E.L.

Creation Date: September 16, 2022 Last Modified: September 16, 2022

Member 1: Bartosz Stoppel Email: bstoppel@purdue.edu

Member 2: Micah Morefield Email: mmorefie@purdue.edu

Member 3: Jack Myers Email: myers395@purdue.edu

Member 4: James Hubbard Email: hubbar31@purdue.edu

Assignment Evaluation:

| **Item** | **Score (0-5)** | **Weight** | **Points** | **Notes** |
| --- | --- | --- | --- | --- |
| **Assignment-Specific Items** | | | | |
| **Analysis of Component 1** |  | x2 |  |  |
| **Analysis of Component 2** |  | x2 |  |  |
| **Analysis of Component 3** |  | x2 |  |  |
| **Bill of Materials** |  | x6 |  |  |
| **Writing-Specific Items** | | | | |
| **Spelling and Grammar** |  | x2 |  |  |
| **Formatting and Citations** |  | x1 |  |  |
| **Figures and Graphs** |  | x2 |  |  |
| **Technical Writing Style** |  | x3 |  |  |
| **Total Score** |  | | |  |

5: Excellent 4: Good 3: Acceptable 2: Poor 1: Very Poor 0: Not attempted

General Comments:

*Relevant overall comments about the paper will be included here*

IMPORTANT NOTE: This team assignment (Component Analysis) and the Bill of Materials are separate team assignments due in subsequent weeks but will be graded together after both have been submitted. Note the weighting of the Bill of Materials in the rubric above.

1.0 Component Analysis:

The primary hardware components of our design that we will analyze are the microcontroller, 4x4 keypad matrix camera system, and piezoelectric sensor. The microcontroller acts as the “interface controller” for the game, handling any user inputs or signals from sensors, to be shared to the laptop for the main game logic. The 4x4 keypad matrix will be used for user input to make modifications to the game, score, and other game concerns. Our camera system will detect where the ball’s current position is during gameplay. Our piezoelectric sensors are used to detect ball bounces, supporting the camera’s data to determine where the ball bounced.

1.1 Microcontroller:

Our choice of microcontroller was determined mostly by ease of access. As we already had multiple STM32F091RCT6 development boards, and had easy access to those chips, we opted for this device. We could begin working day 1. Likewise, each of us has had experience with this chip, and we could all easily contribute to the microcontroller code without having to learn a new system. Because most of our project deals with interfacing a camera with OpenCV, which we have little experience with, we decided we should keep everything else as familiar as possible, to reduce the amount of time spent in the lab learning.

We found a similar MCU, STM32F103C8, that is a good second option. Likewise, Micah has previous experience with an LPC55S69, an impressive device shown in the comparison chart below. Early on in our development, we became worried that our chosen microcontroller may not be strong enough. The following factors were necessary components to our project that were considered:

1. We will be handling ADC conversions from multiple sources. We need multiple ADC channels, and would like more than 1 ADC controller.

* With the absence of a second ADC controller, we will need to guarantee that the ADC sample rate is high enough such that we can sample from multiple channels without losing information. We need each channel to be sampled at approximately 10 kHz.

1. Our ADC inputs will need some signal processing methods applied to them before being understood by our microcontroller, including a band-pass filter. A DSP-capable microcontroller would solve this, in software, given a good enough DSP accelerator.

* If we were to use a bandpass FIR filter on the LPC55S69, on 8 signals (4 ADC cycles), we need the filter to process 2 signals faster than 40kHz (25 μs). Using the onboard PowerQuad Hardware Accelerator, a 12-tap FIR filter takes approximately 2 ms [1]. Thus, we could not compute “real time” filtered data; we’d need buffered data, which will not work for our project.

1. We need at least 1 UART peripheral.

In Table 1 below, a comparison of the three chips is highlighted.

| Statistic / Chip | STM32F091RCT6 [2] | STM32F103C8T6 [3] | LPC55S69 [4] |
| --- | --- | --- | --- |
| ADC Channels | 16 | 2 | 5\* |
| ADC Controllers | 1 | 2 | 2\* |
| ADC Sample Size | 12 bit | 12 bit | 16 bit |
| ADC Sample Rate | 1 MHz | 1 MHz | 1 MHz |
| Clock Rate | 48 MHz | 72 MHz | 150 MHz |
| ARM DSP | No | No | Yes |
| UART Peripherals | 8 | 3 | 9 |
| Price | $0.00 | $7.62 | $3.42+ |
| Lead Time | 0 days | 4 days | 365 days+ |

\* This ADC can handle 10 channels with 1 controller, or 5 channels with 2 controllers.

+ Micah could probably borrow one for free, right now, through his research team.

Table 1. Microcontroller Comparison Chart

1.2 Keypad/User Interface:

For user interaction, we have a few options. Novel systems are possible through our existing hardware. For example, Google’s MediaPipe [5] offers a hand detection algorithm that can communicate with OpenCV through its dataset library [6]; these two softwares in conjunction could allow the user to issue specific commands using hand signs. That being said, systems such as this would be stretch goals, and as development of our product progresses, the necessity for a simpler interaction method is evident; a specific hardware device is required. Buttons are a simple and intuitive interface for users to interact with as long as specific keystrokes and their function are made clear. A keypad interface gives the user 16 possible commands, at each specific state, and saves our team the engineering effort of assembling our own button matrix/panel. Button functions can be communicated through our projector, the primary feedback mechanism.

When considering a keypad matrix, there are a variety of options. The main metrics that constrain our choice are ghosting prevention, hardware/software debouncing, and what interface is supported between the micro and the keypad.

The first keypad we considered was chosen, as with most of the components, considering availability. At our disposal was an AdaFruit NeoTrellis 4x4 keypad. This specific keypad is special in that it not only comes with a button matrix, but an additional PCB to decipher and process the user input, before it is sent to the MCU using I2C. This I2C data can then be deciphered by the recipient with adafruit’s Arduino/C++ library [7]. The benefit to this communication is simplicity. After plugging in the keypad and installing the library we could have a working keypad interface without any deployed engineering effort. However, setting up a keypad was a specific hardware PSSC that we wanted to devote time to; using such a device as the AdaFruit NeoTrellis, leaves no space for our own solution, unless we write our own libraries to make use of the I2C communication bus. However, such a task would result in recreating an already existing software library, and would ultimately exist as an engineering effort for the sake of a PSSC. A better fit for our project is a barebones keypad matrix, without any proprietary PCB or software, that allows us the freedom to multiplex and decipher the input ourselves.

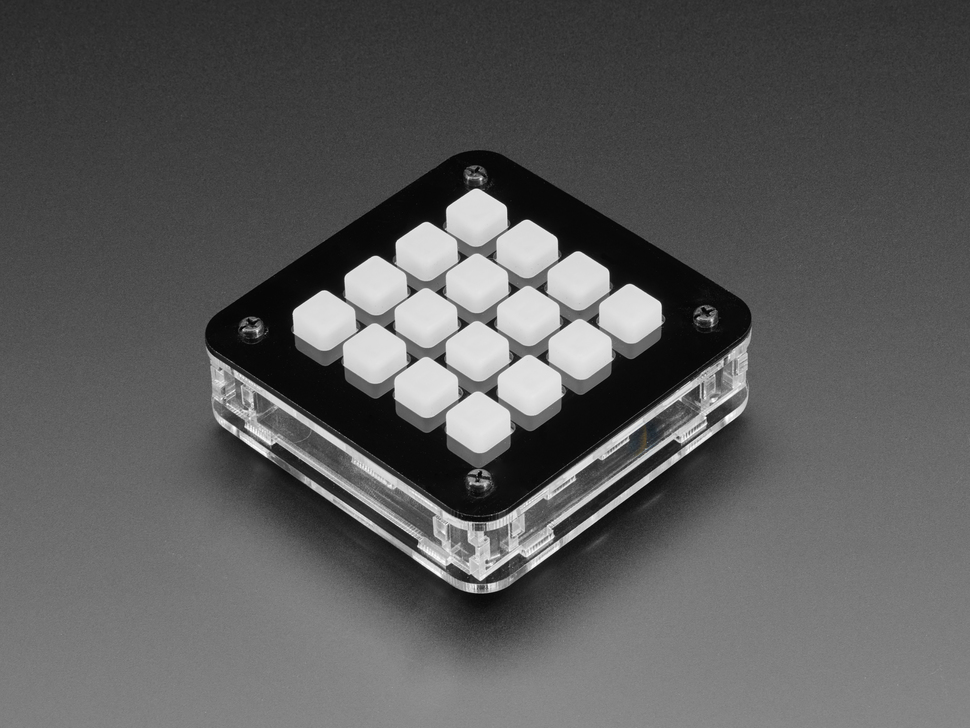
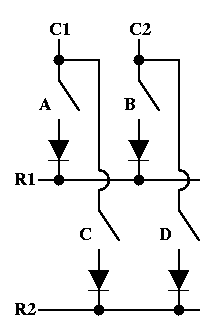
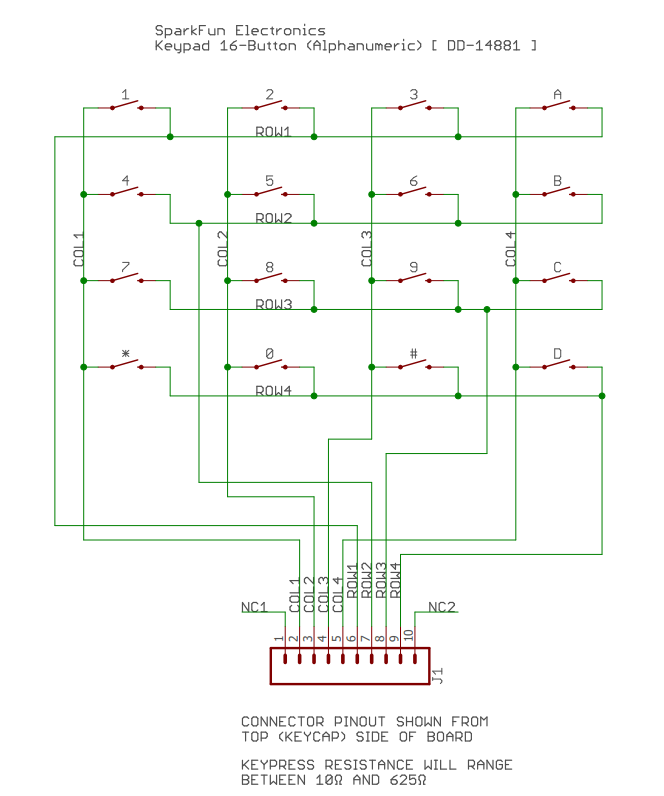
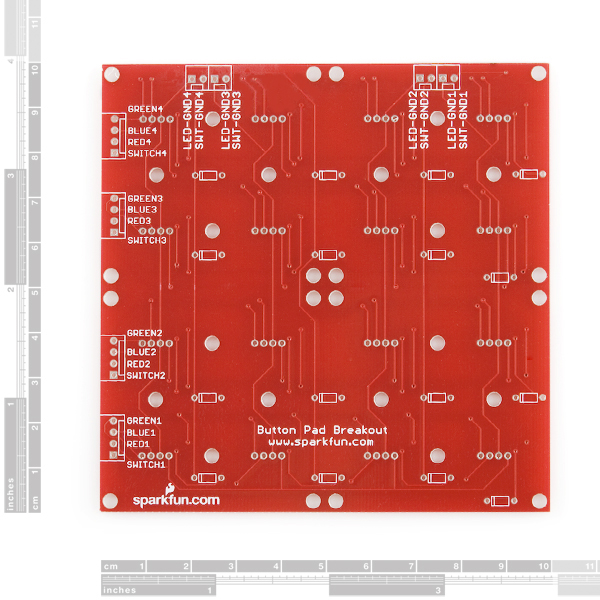
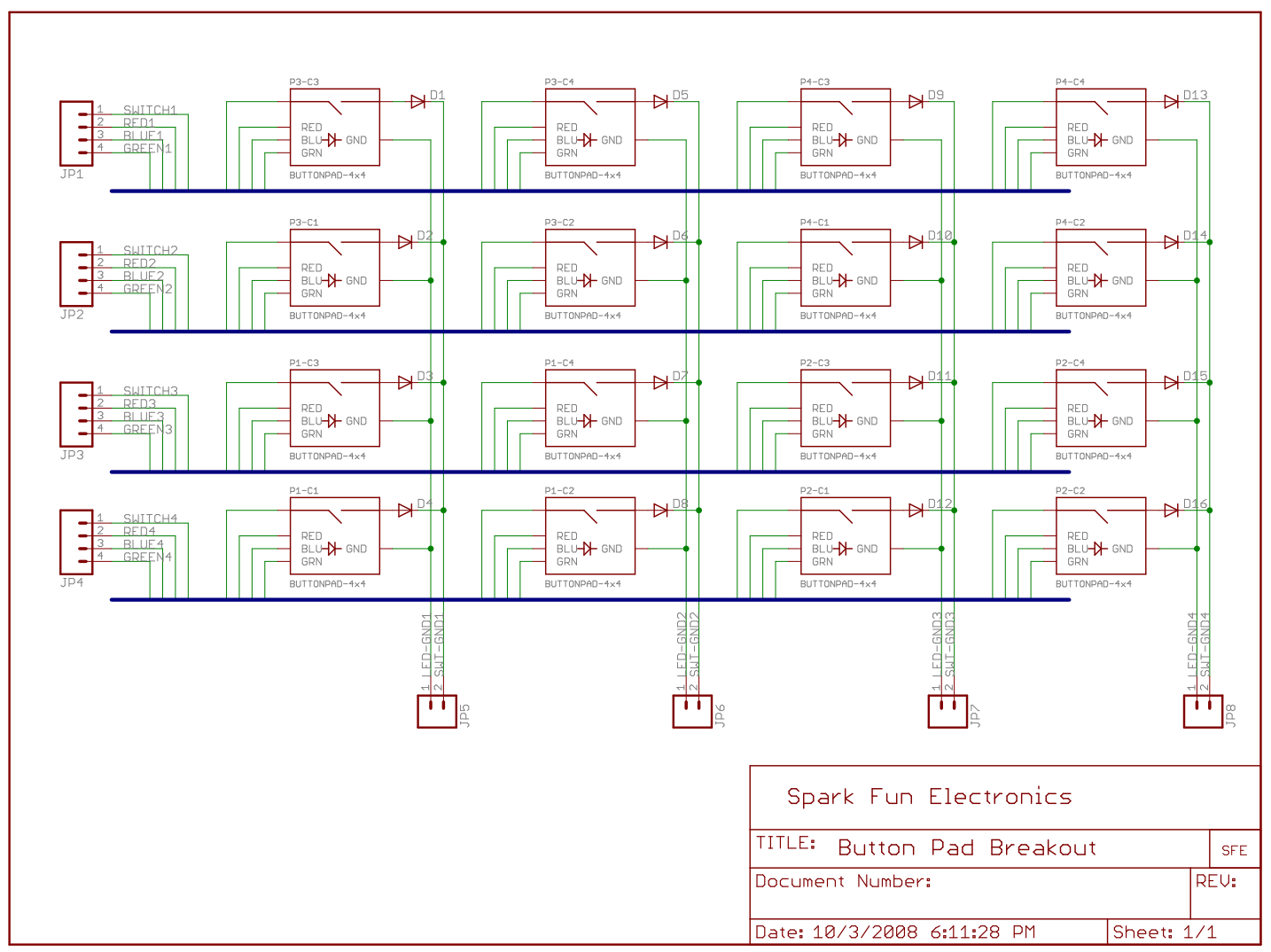


Fig. 1. Adafruit Neotrellis 4x4

Our second keypad option was again dictated by availability, however, our entire team also had experience with this hardware. The Sparkfun 16 button keypad is a minimal option for this component, consisting of only the matrix for multiplexing output [8, Fig. 2]. These four inputs and four outputs allow us to easily interact with this keypad through the GPIO pins on our MCU. Additionally, the MCU can handle tasks such as software debouncing. In almost all metrics, this keypad is perfect for our application, however it falls short in one: ghosting. This Sparkfun product is amazing for its simplicity, but because of this, the matrix does not make use of any extra components, such as diodes. The only way to eliminate ghosting is to add additional hardware as such [9, Fig. 3]. While this option comes close to satisfying our desires, it is incomplete.

Fig 2. Sparkfun 16 Button Schematic Fig 3. Anti-Ghosting Method

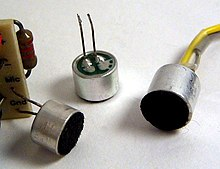
For our final option we choose the component primarily because of, you guessed it, availability. At our disposal is a Sparkfun 4x4 Button Pad Breakout. As seen in [10, Fig. 4] This board has the pads to easily insert diodes and LEDS. This is perfect for our project, as it gives us the freedom to implement anti-ghosting measures while still allowing us to multiplex the button matrix ourselves. One note is that this option exists only as a PCB, thus the added conductive pads, diodes, and optional LEDs, must be acquired and assembled by us. This extra work is not extremely time consuming and worth the tradeoff for a better end user experience.

Fig 4. Sparkfun 4x4 Button Pad Schematic Fig 5. Sparkfun 4x4 PCB

1.3 Microphones:

Sound will play a key role in our project’s ability to keep track of the game state. It’s important that the microphones we choose be well-suited to the sounds we want to listen to. The ideal microphone for our project is very sensitive, because the sound of a table tennis ball striking the table is a relatively small signal.

Electret Condenser Microphone (ECM):

Fig 6: Electret microphones [11]

Electret condenser microphones sense sound through the vibration of a permanently charged diaphragm. In [12] it was concluded that, when used in direct contact with the sound source, these microphones are most sensitive to background noise from the surroundings. Because of the complaints aimed at a similar commercial product (Stiga SensorScore) about household noises triggering points to score, the team was wary of any microphone which was excessively sensitive to background noise.

Accelerometer:

Accelerometer contact microphones generally employ MEMS accelerometers to measure vibrations, with a sensitivity well into the audible range [13]. This type of sensor was shown in [12] to have the lowest sensitivity to background noise and the lowest internal noise floor of these three sensor types. These sensors’ usefulness is derived from their ability to simultaneously sense constant accelerations and accelerations that are varying at a high frequency. The reason the team didn’t choose this type of sensor is that the sensor used in [13] was custom made for the study, and a similar sensor would likely cost a large amount of money and take a long time to arrive.

Piezoelectric Microphone:

Fig 7: Piezoelectric contact microphones

The piezoelectric contact microphone is the quintessential contact microphone. These sensors are cheap and readily available, and they are sensitive to vibrations in the frequency range we are interested in, namely the range of the sound of a table tennis ball bouncing on the table. Furthermore, in [12] it was concluded that piezoelectric microphones reject nearly as much background noise as an accelerometer, however their noise floor is higher. The team chose to use this type of sensor in our final design because it is simple to use, cheap, and readily available (unlike MEMS devices).



1.4 Camera:

The camera is the main input for the laptop and gathering game data. The camera, alongside with Piezo microphones, is what will be converting the tennis table game into information that we will be working with in our software. The camera’s feed will be handled by a laptop as it may be too vigorous to process through for MCUs and PCUs. Criteria that the camera must meet for prototyping and commercial needs include having both a color/depth sensor, a fast-ball-capturing frame rate, and mountability. For prototyping reasons, we are not entirely set on using a color or depth sensor in order to detect where a ball is on the screen. So, we have been looking for a camera that can capture both types of feeds. The fast-ball-capturing frame rate that we have decided to be sufficient enough is 30 FPS. Less than 30 FPS, we might not have enough time to capture the exact ball bounce location; faster than 60 FPS, we might not be able to process all the frames fast enough. Mountability will be key later on when we decide on how to suspend the camera above the table. This camera and the projector will be the most cost heavy peripherals in the project, so the price of the entire product will depend on the camera/projector choice. Therefore, finding a cheaper camera could come at an advantage to lower production cost.

**Asus Xtion Pro Live [15]**

This is the camera that was available for us in the senior design lab. A previous team had purchased it and it meets all of our criteria. It is small, has a 30 FPS depth sensor and a 60 FPS color sensor. It is also available at $300 on the manufacturers website [15], which is expensive, but it has been in the market for a while and it will be available to purchase at a lower cost on 3rd party vendors. We are currently using this camera, but are open for change as the main reason for the choice of this camera was its immediate availability.

The camera requires the install of OpenNI2 in order to access the camera feed data. It is an open source software that will convert the camera feed to data that will be used by our blob detection algorithm in order to find a ball within the feed.

**Xbox Kinect [16]**

The Xbox kinect is the most widely known camera sensor for depth and color sensing. The kinect captures at 30 FPS for both feeds and is small enough to mount anywhere. The Kinect was popular enough to be in supply everywhere and has been in the market long enough that the price for a kinect for those that do not have one is less than $30. We already own one so we do not have to consider this price for prototyping. And since the kinect was so popular, there are many projects and repositories that utilize the Kinect, which makes many libraries and tools available at our disposal.

There exists a 2.0 Version of the Kinect that was released for the former Xbox 1. It comes at around $20 on average from vendors with used products (GameStop, eBay, …). The differences include a wider FOV and larger resolution [17]. So if the ball becomes undetectable because it is too close to the camera (~within 1 ft for Xtion Pro Live from testing) then a Kinect 2.0 could be useful.

**Intel® RealSense™ Depth Camera D455 [18]**

We included this camera because it is the modern day standard. It is very small and it offers the industry's best color/depth sensors. The RealSense is a failsafe if the Kinect and Xtion Pro Live fail to capture the ball’s movement accurately. This can have an output frame rate up to 90 FPS[18].

2.0 Sources Cited:

*Throughout this and other papers, use of the IEEE citation style should be used. Use of embedded hyperlinks for all web-based sources is required. A reference to the IEEE citation style format is provided* [*here*](http://www.ieee.org/documents/ieeecitationref.pdf)*.*

[1] Mouser, “LPC55S69 User Manual,” 24-Oct-2019. [Online]. Available: <https://www.mouser.com/pdfDocs/NXP_LPC55S6x_UM.pdf>, P. 1079. [Accessed 16-Sep-2022].

[2] STMicroelectronics, “STM32F091RC,” 2017. [Online]. Available: <https://www.st.com/en/microcontrollers-microprocessors/stm32f091rc.html#overview>.

[Accessed 16-Sep-2022].

[3] STMicroelectronics, “STM32F103C8,” 2017. [Online]. Available: <https://www.st.com/en/microcontrollers-microprocessors/stm32f103c8.html#overview>.

[Accessed 16-Sep-2022].

[4] NXP, “LPC55S69 Product Data Sheet,” 6-Aug-2021. [Online]. Available: <https://www.nxp.com/docs/en/nxp/data-sheets/LPC55S6x_DS.pdf>. [Acccessed 16-Sep-2022].

[5] “Hands,” *mediapipe*. [Online]. Available: <https://google.github.io/mediapipe/solutions/hands.html>. [Accessed: 16-Sep-2022].

[6] “Framework for working with different datasets,” *OpenCV*. [Online]. Available: <https://docs.opencv.org/4.x/d8/d00/group__datasets.html>. [Accessed: 16-Sep-2022].

[7] A. Industries, “Adafruit 4x4 NeoTrellis feather M4 Kit Pack,” *adafruit industries blog RSS*. [Online]. Available: <https://www.adafruit.com/product/4352>. [Accessed: 16-Sep-2022].

[8] “Keypad - 16 button (alphanumeric),” *DD-14881 - SparkFun Electronics*. [Online]. Available: <https://www.sparkfun.com/products/retired/14881>. [Accessed: 16-Sep-2022].

[9] *Keyboard matrix help*. [Online]. Available: <https://www.dribin.org/dave/keyboard/one_html/>. [Accessed: 16-Sep-2022].

[10] M. #430122, M. #160590, and Rsp, “Button pad 4x4 - breakout PCB,” *COM*. [Online]. Available: <https://www.sparkfun.com/products/8033>. [Accessed: 16-Sep-2022].

[11] “Electret microphone,” *Wikipedia*, 31-Jul-2022. [Online]. Available: <https://en.wikipedia.org/wiki/Electret_microphone>. [Accessed: 17-Sep-2022].

[12] A. Carullo, “ICSV 2015,” in *PERFORMANCE COMPARISON OF DIFFERENT CONTACT MICROPHONES USED FOR VOICE MONITORING*, 2015. [Online]. Available: <https://www.researchgate.net/publication/293174544_PERFORMANCE_COMPARISON_OF_DIFFERENT_CONTACT_MICROPHONES_USED_FOR_VOICE_MONITORING>. [Accessed: 17-Sep-2022]

[13] P. Gupta, M. J. Moghimi, Y. Jeong, D. Gupta, O. T. Inan, and F. Ayazi, “Precision wearable accelerometer contact microphones for longitudinal monitoring of mechano-acoustic cardiopulmonary signals,” *Nature News*, 12-Feb-2020. [Online]. Available: <https://www.nature.com/articles/s41746-020-0225-7>. [Accessed: 17-Sep-2022].

[14] “Contact microphone,” *Wikipedia*, 07-May-2022. [Online]. Available: <https://en.wikipedia.org/wiki/Contact_microphone>. [Accessed: 17-Sep-2022].

[15] [Online]. Available: <http://xtionprolive.com/asus-3d-depth-camera/asus-xtion-pro-live>. [Accessed: 16-Sep-2022].

[16] “Kinect” Wikipedia. [Online]. Available: <https://en.wikipedia.org/wiki/Kinect>.

[Accessed: 16-Sep-2022].

[17] “The difference between Kinect v2 and v1” *Skarred Ghost .* [Online]. Available: <https://skarredghost.com/2016/12/02/the-difference-between-kinect-v2-and-v1/>

[Accessed: 16-Sep-2022]

[18] “Intel® RealSense™ Depth Camera D455” *Intel Real Sense.* [Online]. Available: <https://store.intelrealsense.com/buy-intel-realsense-depth-camera-d455.html>

[Accessed: 16-Sep-2022]