Aqua Strings:
Interactive Water Harp

Ryan Bori, Christopher Brown, Trenton Morris, Mikaella Penagos

Dept. of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, 32816-2450

***Abstract* — An original design merging optics, water, and sound; Aqua Strings is a new and interactive way to play music. Unlike a traditional harp, Aqua Strings' strings come from lasers imposed within descending columns of water. With sensors calibrated to read these water-enclosed lasers, the user can strum these water streams to produce beautiful tones. This system is intended to be used in a recreational setting, with the aesthetics to display as a fountain combined with the functionality to be played as an instrument.**

***Index Terms* — Audio system, I2S Digital-analog conversion, Laser diodes, MP3, Photodetectors, Water**

I. INTRODUCTION

In the modern day of interactive technology and immersive art, there is a growing intrigue in systems that can engage multiple senses to create memorable user experiences. *Aqua Strings* is an interactive water harp designed to merge both the tactile and auditory worlds. This project explores the intersection of art and engineering by using laser detection and fluid dynamics to create an instrument that plays with water, light, and sound.

Technically, this project relies on these different systems seamlessly working together. Laser diodes shoot down from above to the bottom of the harp frame, where they are received and read by responding sensors. Meanwhile, a pump mounted in the lower reservoir is cycling water through the pipes and out of several defined channels, where they return to the reservoir in water columns. Finally, our electronics and PCBs are positioned near the top of the frame, monitoring and controlling the entire system. The sound system features custom digital-to-analog converters ready for quick audio playback at the strum of a water column.

Fundamentally, *Aqua String*s combines water streams and responsive audio output triggered by user interactions. For instance, when a user's hand interrupts a laser beam, the system responds in real-time by detecting which string was triggered and playing the respective musical note. The result is a dynamic experience that feels alive – playing strings of water as if they were of a traditional harp.

II. PROJECT COMPONENTS

Aqua Strings can best be introduced through its hardware components: The selection process behind the microprocessor unit, laser, photodiode sensor, audio speaker, and water pump all play a critical role in the multidisciplinary nature of the project. Additionally, each component has gone through careful consideration against other market options to prioritize the most optimal performance.

A. MCU: ESP32

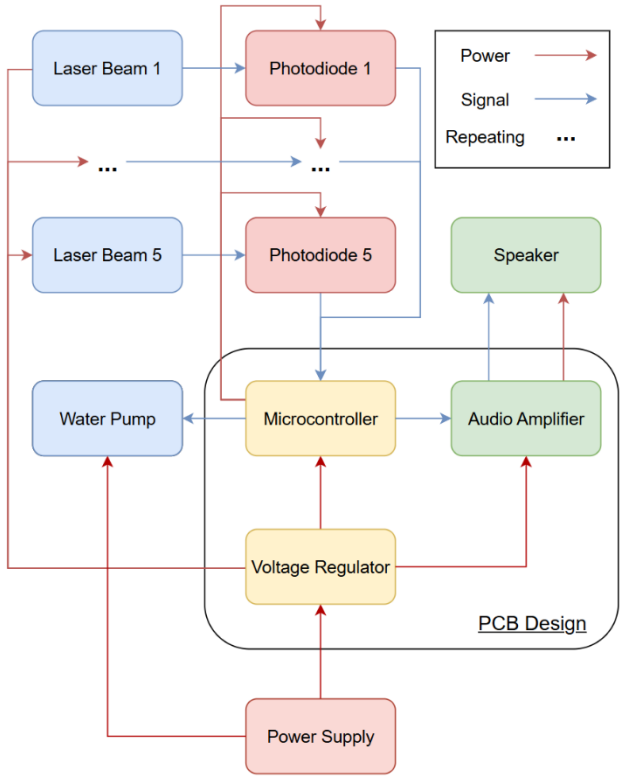
At the core of the control system, the ESP32 microcontroller is selected for a variety of key features: One of the primary reasons being its built-in digital-to-analog converter (DAC). *Aqua Strings* relies on responsive audio output, and the ability to convert digital signals to audio within the same microcontroller unit improves hardware design complexity and reduces board space. With the board cost of 8USD and dimensions 51x25.5 mm, the ESP32 consists of 34 GPIO pins. Such space is sufficient for hardware connections of lasers and photodiode sensors across five water strings. The ESP32 compact footprint and cost-efficient design compared to other MCU alternatives is an ideal choice that is favorable to project standards.

B. Laser Diode

To detect user interaction, *Aqua Strings* uses a laser beam-interruption method where a laser light is directed at a paired photodiode sensor. The laser of choice is a 650nm red laser diode due to its accessibility and cost-effectiveness. Despite their reputation for having a higher absorption rate in water compared to green or blue wavelengths, red was chosen as they offer increased availability and lower unit prices. The selected diode operates at 5mW output power, with a voltage range of 2.8V-5.2V, and produces a dot profile on contact with the sensor.

C. Sensor

To detect laser beam interruptions, *Aqua Strings* uses photodiode sensors positioned opposite each laser emitter. Among several candidates, a 3mm clear, flat head photodiode was chosen for its balance of sensitivity and



affordability. It features a wavelength detection range of 400nm to 1100nm, which is suitable for its corresponding laser diode. It operates within a broad temperature range upward of 100°C, meaning it can more than withstand being under the laser diode for prolonged periods of time. One of the key drivers for selecting this specific photodiode was its cost and availability. The unit cost was approximately $0.36, enabling the ability to prototype without fear of water damage.

D. Speaker

For audio output, *Aqua Strings* utilizes a passive 8Ω, 3W speaker as the device responsible for translating digital signals into audible feedback. This passive speaker does not have an internal amplifier; however, this is compensated for by the project's custom audio amplifier board. This allows the project to have a more deliberate integration of the external component to the PCB. The 3W model selected strikes a suitable balance between compact footprint, moderate power consumption, and sufficient volume output for the audio needs of the project. Due to its low cost ($8 per unit) and simplicity, the passive speaker was a practical option for this component.

E. Water Pump

The Smartpond 155 GPH Submersible Water Pump is responsible for pumping water from the lower reservoir to the PVC waterfall system. The water pump is rated to 2.5 Gallons Per Minute and 3.5ft vertical lift. One of key factors for selecting this water pump was its built-in speed dial. This allowed for live water flow adjustment, changing the pressure within the PVC system at the turn of a dial. The water system could be calibrated with this capability, and the water pressure could be tuned with a potentially changing PVC system. As a safety measure, the submersible water pump has a built-in water sensor that can automatically shut off if water is below a safe level.

III. PROJECT OVERVIEW

To properly showcase the project, the following section will describe our project with an overview from two perspectives. This will consist of a hardware and software overview that explains how the individual components work together and the general logic of the system.

A. Hardware Overview

Aqua Strings operates on several subsystems that are independent and yet also intertwined. These subsystems require multiple components that each need their power and signal connections. Some components need to be told when to turn on or off while others need to communicate with themselves on when to operate. To properly reflect the complexity of the system and to ease its explanation, a hardware block diagram was created.

Fig. 1 Hardware Block Diagram showing the power and signal relations between components.

Figure 1 offers a general look at how each component interacts with one another. The black box in the middle of the diagram reflects the PCB design, which contains a microcontroller, voltage regulator(s), and an audio amplifier. The microcontroller is housed in the main circuit board, which features connection points to other boards and peripherals. The voltage regulator boards control the voltage coming in from the power supply and regulate it to other components of the system. The audio amplifier board boosts the audio signal from the PCB and allows the speaker to output a clear and loud sound. This PCB design is the brains of the hardware, controlling peripherals such as the water pump, laser diodes, and photodiodes. Note that Figure 1, for clarity, has truncated the peripherals to only the first and last laser diode/photodiode pairing. In reality, there are five pairs of lasers and sensors working to operate five different harp strings. To break this figure down, analysis of the hardware overview will start at the bottom of the figure and work its way up.

The system begins with the power supply. *Aqua Strings* was designed to be powered with a single wall connection. Receiving 120VAC, this power is fed into an AC/DC converter that outputs 12VDC. This 12VDC rail will provide power for the entire system, excluding the water pump. This component will have its connections linked directly to the 120-volt input terminals of the power supply. It will be controlled via a relay interfaced with the microcontroller. This setup will allow the microcontroller to turn the pump on or off as necessary.

Moving up from the power supply, the 12-volt output is fed into the voltage regulator boards of the PCB design. These boards regulate this high voltage down to usable ranges of 3.3-volts and 5-volts. The lower 3.3-volt output is used for components like the ESP32 microcontroller and photodiodes. The higher 5-volt output is used for components like the audio amplifier board and laser diodes. Figure 1 showcases this relationship using red "power" arrows that move from the voltage regulator and microcontroller to the peripherals.

Next, the peripherals handle the interaction of the system. Following Figure 1's blue "Signal" arrows, the laser diodes send a signal to the photodiodes in the form of a laser beam. This light received by the sensors is arguably the most important signal of the project. The sensor communicates to the microcontroller its light value reading, and the system can then determine if there has been an interruption in the laser beam. If the photodiode is reading a lower light value, this signal will pass onto the microcontroller and the system will read it as a trigger. Then, the microcontroller will pass on a signal to the audio amplifier board, which will operate the speaker to play the corresponding note.

Ultimately, the PCB was designed to connect all the necessary components through correct power rails and signal channels. It utilizes the main microcontroller board to achieve the majority of these connections, supplemented by the voltage regulator boards for the remaining power connections and the audio amplifier board for the speaker connections. The result is a system that takes light input and outputs audio, while simultaneously operating an independent pump system.

B. Software Overview

To supplement the complexity of this project's multiple subsystems, equally robust logic is required to ensure correct operation. To clarify this explanation, a software block diagram has been created.

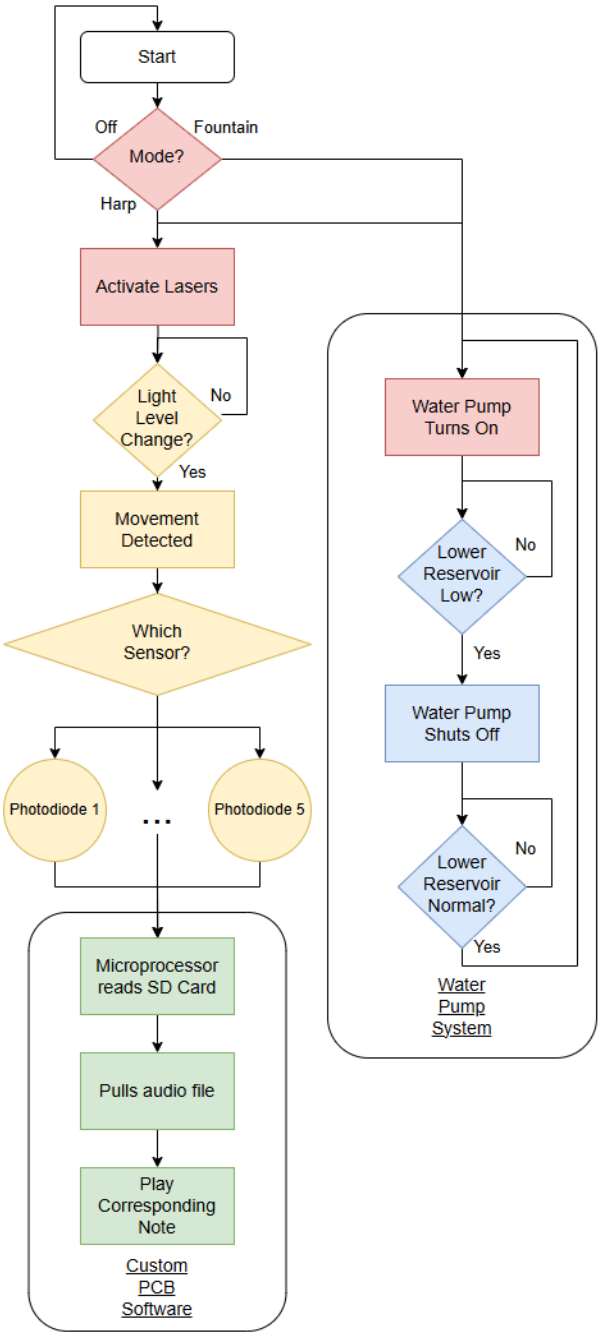


Fig. 2 Software Block Diagram showing the logic of the system.

Figure 2 offers a comprehensive summary of the operation of *Aqua Strings*. The two black boxes indicate defined subsystems within the project. The first box relates to the software logic of the PCB and specifically the microcontroller. The second box relates to the logic of the water pump and its response to different water levels.

Above the Custom PCB Software box is the logic for the laser beam-sensor system. This software block diagram could be split into the left column, featuring the light and PCB logic, and the right column, featuring the water system logic. To break this figure down, analysis of the software overview will start at the top and work its way down.

The logic of the system begins with reading what mode the project is in. By default, of the three different modes, the system begins in the off position. If the user were to switch the project into fountain mode, the water pump system would be enabled. In this mode, only the right side of Figure 2 is active. Specifically, the electronics relating to light and sound (located on the left column) would not be active. Fountain mode strictly reduces *Aqua Strings* to only operate its water strings. For the water pump system, once activated, the water pump turns on. As it pumps water from the lower reservoir through the pipes, the lower reservoir drains. Ultimately, this water pours back down as water streams. If the lower reservoir drains to a critical level, the water pump is set to shut down to protect against potential damage. This gives the system or the user time to add water back to the lower reservoir. Assuming normal operation and a sufficient water level, the water pump should operate continuously in this mode.

The other active mode for this project is harp mode. Harp mode is like fountain mode in that they both activate the water pump system. The main difference is that the light and sound electronics are also active (left column). When the user switches the project into harp mode, both the left and right column of figure 2 activate. When the lasers and sensors are active, the system waits until it detects a light level change. When one occurs, it indicates that there's been a trigger in one of the beams. The system will then determine which strings were played and signal the microprocessor to pull the relevant audio file. The microSD card on the PCB stores audio files relating to each of the five notes. Once selected, the PCB will signal the audio amplifier to play the correct note through the speakers.

Ultimately, this project operates on three core logic subsystems: the water pump, the laser detection, and the microprocessor audio playback. When all of these systems work in tandem, the logic of the project operates seamlessly.

IV. HARDWARE DESIGN

When put together Aqua Strings consists of 5 PCBs (Main board, two 3.3V regulators, one 5V regulator and the audio board) that plug into one another. Each has its own function and is kept to its own board for easy maintenance in case known issues arise. Aqua Strings design involves flowing water, so it is crucial to maintain wet and dry zones when dealing with both water and electronic components.

A. Main Board

The main board serves as the central hub for all control as it integrates the ESP32-WROOM-32 module, which handles all high-level processing and reacts as needed through I/O control. When breadboarding our project we use a standard ESP32 Development Board so when it came to designing our own board it was decided to pull out the same pins in the order the development board had to serve as extra test points or for further unexpected utilization. As originally designed, the pins that were pulled out (other than the development board pins) laser diodes, photodiodes receivers, the audio system, mode selector, and other general interface components. Sigal clarity is ensured through the use of pull-up and pull-down resistors on input lines along with the use of decoupling capacitors near key ICs to aid in stabilizing high-frequency switching.

Several subsystems interface with the microcontroller through a multitude of connectors on the board. A microSD card module is connected via dedicated SPI lines for audio file storage while UART communication is established with an external chip for serial integration. Additionally, there is a level shifter that is included to handle logic conversions between 3.3V and 5V devices, allowing for combability between different sensor types and communication protocols. Lastly, there is a MOSFET used to control the state of the laser diodes to aid in detection accuracy.

To support expandability and unexpected maintenance, the Main PCB includes multiple headers groups by function and need, which simplify wiring and allow modular replacement of subsystems such as the regulator boards (3.3V and 5V), audio amplifier board, rotary dial, water pump relay and the previously mentioned systems. The PCB's design emphasizes separation of signal paths to reduce noise, especially between analog and digital sections. Every header group is placed in a convenient spot since the overall dimensions of our PCBs need to be considered since they will all sit in the upper shelving unit of the project.

B. Voltage Regulators

The voltage regulating system consists of two buck (step-down) converters based on the LM2576 switching regulator, one using a 3.3V chip and the other using a 5V one. These regulators were selected on account of their reputation amongst the staff, specifically recommended by Dr. Weeks himself. The voltage regulator is arguably the most important portion of the PCB design, as its viability determines the ability for the rest of the project to function. Each regulator chip is capable of stepping the 12V DC

output from the power supply down to its respective voltage output. This process involves high frequency switching of internal transistors to transfer energy efficiently through an inductor, which is essential for reducing heat while also increasing efficiency.

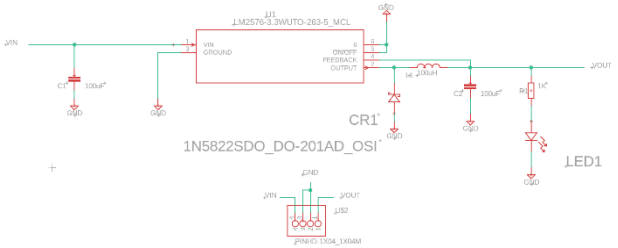


Fig. 3 LM2576 Switching Regulator schematic for 3.3V and 5V output.

As outlined in the above figure, the output from each LM2576 chip is smoothed using a 100 µF capacitor (C2) and a 100 µH but this inductor varies **d**epending on where the regulator is being plugged in. For example, the current draw for the main board is much higher than the current draw of the audio amplifier board so the main board regulators require an inductor that withstands the current draw. On these regulator boards, there is a Schottky diode (CR1) which is placed in parallel with the inductor to allow continuous current flow during the off phase of the switching cycle, preventing voltage spikes that damage components. With this combination of components, the regulator can produce a steady DC voltage at the output pin (+VOUT) and is then routed to the necessary connections downstream.

For monitoring and debugging purposes, each regulator includes a status LED (LED1) connected through a current-limiting resistor (R1), which lights up when there is an output voltage. This type of setup helps isolate a potential issue that could be discovered when testing the boards.

C. Audio Amplifier

Fig. 4 TPA2016D2 Audio Amplifier board schematic (Latest re-design)

The audio board used to process the audio files from the microSD to be played on the attached speakers has gone through three re-designs and two different chips. The final design consists of using the internal DAC of the ESP32 and the TPA2016D2 stereo Class D amplifier. This IC is controlled over I2S and includes built in features such as automatic gain control (AGC), dynamic range compression and volume control, which is adjusted through the uploaded code. The audio portion of this project uses two speakers, but the system uses mono audio so the inputs to the amplifer are connected (IN_P). This input also passes throgh decoupling capacitors and resistors before entering the chip to help filter noise and these components need to be placed as close to the chip as possible otherwise there is a drop in efficiency, as dictated by the data sheet. The power inputs (PVDD, AVDD) also use decoupling capacitor to suppress ripples from the voltage regulators. The chip is powered through the 5V regulators off the main board into VDD and then there is a separate 3.3V regulator attached to this board to power I2S VDD to ensure clean rails.

As previously mentioned, the system using mono audio to simplifiy the design therefore the output pins of the TPA2016D2 chip will play the same audio (OUT_L_P and OUT_R_P) and branch to two speakers. Volume and audio behavior are managed by sending control data over I2C lines (SDA, SCL) which are connected back to the main PCB. Additional control signals SHUTDOWN and GPIO25 provide the ability to enable or disable the amplifier under software control.

V. SOFTWARE DESIGN

The software for Aqua Strings is designed as a real-time, event-driven system designed to run on a dual-core ESP32 microcontroller. The core architectural philosophy is a State-Driven Event Model [1], where the system transitions between defined operational states such as OFF, FOUNTAIN, and HARP, in response to user input or sensor events [2]. This model ensures predictable behavior and robust handling of various operating conditions. The entire system is written in C++ within the Arduino framework, Utilizing the FreeRTOS real-time operating system to manage concurrent tasks efficiently across the ESP32's two processing cores. This design prioritizes modularity, with distinct software modules responsible for specific functionalities. The primary software modules include a System State Manager, which implements the main state machine and organizes the behavior of all other modules, a Sensor Manager, responsible for all aspects of sensor data acquisition and processing; a high-priority Audio Player, which manages audio file loading, I2S configuration, and playback, and a simple Pump Controller, which activates the water pump relay as commanded. This modular approach decouples the hardware interactions from the application logic, making the system easier to debug, maintain, and extend [3]. The overall software design emphasizes non-blocking execution to ensure the system remains responsive, managing intensive operations like file

loading through lower-priority tasks that do not interfere with time-critical operations like sensor polling and audio streaming [4].

A. Real-Time Operating System and Task Management

To achieve true concurrency and meet the strict real-time requirements of a musical instrument, the software leverages the ESP32's dual-core architecture and the FreeRTOS kernel [5]. The workload is carefully partitioned between the two cores to isolate time-critical functions from less-urgent background processes. Core 0 is designated as the Real-Time Core, dedicated to the high priority audioTask. This task, which runs at the highest priority level (configMAX_PRIORITIES - 1), manages the I2S data stream for audio playback, ensuring that sound generation is never delayed by other system activities. This core-task affinity is explicitly enforced using the xTaskCreatePinnedToCore FreeRTOS function [6]. In contrast, Core 1 serves as the Application and I/O Core, handling all other system logic and I/O operations. The tasks running on this core include the sensorTask for polling the photodiodes, the systemStateTask for managing the main application logic, and any future tasks for connectivity, such as a wifiTask. While these tasks are critical, they are more tolerant of minor timing variations than the audio stream, making this an effective strategy for division of labor and preventing stuttering or glitches in the audio output [5].

Inter-task communication is managed using FreeRTOS queues, which provide a thread-safe way for passing messages [7]. For instance, when the sensorTask detects a valid strum, it sends an event message containing the specific string's ID to a queue that the systemStateTask is listening on. This frees sensor detection from application logic, allowing each task to operate independently. Shared hardware resources, such as the SPI bus for SD card access, are protected using semaphores (mutexes) to prevent race conditions [8]. This ensures that only one task can access a shared resource at any given time, maintaining data integrity and system stability. For example, a semaphore prevents the future wifiTask from accessing the SD card while the AudioPlayer is loading a new sound bank into PSRAM.

B. Receiving and Processing Sensor Data

The accurate and reliable detection of user interactions is fundamental to Aqua Strings' functionality; we achieve this through a multi-stage software process that converts raw analog sensor readings into musical trigger events. The system's photodiodes are connected to the ESP32's 12-bit Analog-to-Digital Converter (ADC), which yields a raw sensor value between 0 and 4096. To adapt to varying ambient light conditions, a calibration routine is executed at startup. This routine calculates a detection threshold by averaging the baseline ambient light reading (taken with lasers off) and the direct laser reading (taken with lasers on), using the formula: threshold = (laser_on_value + ambient_baseline) / 2. This method establishes a detection midpoint that is resilient to environmental light changes [9].

To prevent false triggers from transient noise, such as electrical interference or minor water turbulence, the sensorTask applies a moving average filter to the raw ADC data. This is a simple and effective debouncing technique where the task maintains a circular buffer of the most recent sensor readings and uses the average as the reference signal value [10]. This smoothing process ensures that a "strum," which is defined as a single instance where the filtered ADC value dips below the calculated threshold, is only registered when there is a sustained drop in light intensity. The system is also designed to handle polyphony. If multiple strings are detected as strummed within a short time window, the software interprets this as a chord and commands the AudioPlayer to mix and play the corresponding notes simultaneously.

C. Audio Pipeline

The audio pipeline is meticulously designed for high-quality, low-latency sound. All audio files are stored on an SD card as 8-bit, 96kHz mono .wav files. When first plugging in the system, a default Harp C major pentatonic scale sound bank is fully pre-loaded from the SD card into the ESP32's external PSRAM. This strategy was chosen to eliminate SD card read latency during playback, which is a significant bottleneck in real-time audio applications [11]. Accessing audio data from PSRAM at CPU speeds is important for enabling seamless polyphony, as it allows the system to mix multiple audio streams concurrently without the I/O bottleneck of direct streaming from the SD card.

Audio is rendered using the ESP32's built-in Digital-to-Analog Converter (DAC), configured in I2S_MODE_DAC_BUILT_IN mode [12]. The AudioPlayer performs an on-the-fly conversion, upscaling the 8-bit mono samples from PSRAM into 16-bit stereo frames before writing them to the I2S DMA buffer. To prevent the audible "pops" or "clicks" that occur when abruptly starting or stopping a DAC, the software implements a precise playback sequence. Before playback, a buffer of digital silence is written to the I2S bus to smoothly ramp up the DAC's output voltage. After the audio finishes, another silence buffer is written to ramp the signal down. This technique is essential for preventing the sudden DC offset changes that cause these audio artifacts [13].

D. System State and Error Handling

The software is designed to be robust. The System State Manager monitors the health of critical components during initialization. The system includes error flags for key failure conditions, including an SD card mount failure, an unresponsive I2C device, or a PSRAM allocation failure. Upon detecting an error, the software logs a descriptive message and enters an error state [14]. In this state, all interactive functionality is disabled, the water pump is turned off, and no audio is attempted. This fail-safe approach prevents the system from entering an undefined or potentially damaging state and provides clear feedback for troubleshooting [15]. A manual check and restart are required to clear the error and reattempt initialization.

E. Stretch Goal: WIFI controls

A primary stretch goal is to implement an interactive controller by configuring the ESP32 to operate in Wi-Fi Access Point (AP) mode, a capability supported by the ESP-IDF framework [16]. This creates a self-contained network, allowing a user to connect directly to the instrument with a smartphone or laptop. A lightweight embedded web server running on the Application Core would serve as a control webpage [17]. This interface would allow for real-time adjustment of audio gain and volume, and a dropdown menu would enable dynamic swapping of .wav file sets from the SD card into PSRAM. For a responsive user experience, the interface would feature remote playback triggers and live feedback of currently playing notes. This bidirectional, real-time communication would be achieved using the WebSocket protocol, which is ideally suited for low-latency, full-duplex communication between a web client and an embedded device, avoiding the overhead of traditional HTTP polling [18]. Commands from the web interface, such as playing a note or changing volume, would be submitted as events into the existing FreeRTOS queues.

VI. CONCLUSION

The systems above comprise all of what *Aqua Strings* is. This complexity is required for a project that leverages optics, fluid dynamics, digital-analog conversion, and audio playback to create an immersive user experience. All these systems working together allow for a functional and interactive water harp to exist. Throughout development, the team overcame mechanical, electrical, and software integration challenges. By applying lessons learned during our undergraduate coursework and supplementing them with hands-on testing, the team built a fully operational prototype. *Aqua Strings* represents what's possible at the intersection of art and engineering.

ACKNOWLEDGEMENT

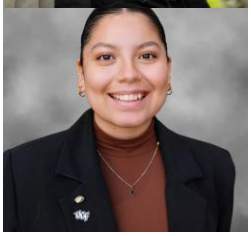
We would like to acknowledge Professor Weeks for his continued support throughout the semester. His determination to help his students is reflected in this project.

BIOGRAPHY

Ryan is a graduating senior at the University of Central Florida studying Computer Engineering. His interests include embedded systems and firmware development, with a focus on low-level hardware software integration.

Christopher is a graduating senior at the University of Central Florida studying Electrical Engineering. After graduating, he has plans to work as an Electrical Engineer involving motor control work for water treatment facilities throughout Texas and Florida.

Trenton is a graduating senior at the University of Central Florida studying Electrical Engineering on the Power and Renewable Energy Track. He aspires to work in the field of power generation.



Mikaella is a graduating senior at the University of Central Florida studying Electrical Engineering with a minor in music. She is looking forward to working in the power transmission sector throughout the Florida region.

REFERENCES

[1] M. Samek, *Practical UML Statecharts in C/C++: Event-Driven Programming for Embedded Systems*, 2nd ed. Newnes, 2008.

[2] R. Adamczyk, "State Machine," in *Pattern Languages of Program Design 5*, M. Kircher and P. Jain, Eds. Addison-Wesley, 2006.
[3] C. Walls, "Modular Development with C++," *Overload*, vol. 129, pp. 10-14, Oct. 2015.
[4] H. Kopetz, *Real-Time Systems: Design Principles for Distributed Embedded Applications*, 2nd ed. Springer, 2011.
[5] Espressif Systems, *ESP32 Technical Reference Manual*, Version 4.4, 2022. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf
[6] Espressif Systems, "FreeRTOS (IDF) - ESP-IDF Programming Guide," 2022. [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/freertos_idf.html
[7] R. Barry, *Using the FreeRTOS Real Time Kernel - A Practical Guide*, Real Time Engineers Ltd, 2016.
[8] M. T. Heath, *Scientific Computing: An Introductory Survey*, 2nd ed. McGraw-Hill, 2002. (Chapter 12 discusses mutual exclusion for preventing race conditions).
[9] J. Dueck, "Simple Automatic Threshold Generation For Embedded Systems," *element14 Community*, Aug. 2021. [Online]. Available: https://community.element14.com/members-area/personalblogs/b/blog/posts/simple-automatic-threshold-generation-for-embedded-systems
[10] S. S. S. R. R. Alluri and V. K. Vummadi, "Comparative Analysis of Moving-Average Filter and Savitzky-Golay Filter for Denoising of Photo-Plethysmography Signal," *TEM Journal*, vol. 14, no. 2, pp. 1681–1688, May 2025.
[11] M. Jacob, "Why Dynamic Memory Allocation is a Problem in Real-Time Systems," *Medium*, Nov. 2022. [Online]. Available: https://medium.com/@mjacob_51453/why-dynamic-memory-allocation-is-a-problem-in-real-time-systems-e75865662712
[12] P. Schatzmann, "I2S - The Way to Go For Audio," *GitHub*, 2021. [Online]. Available: https://github.com/pschatzmann/arduino-audio-tools/wiki/I2S-The-Way-to-Go-For-Audio
[13] T. G. (tannewt), "DAC output pops at beginning and end of audiocore.WaveFile playback," *GitHub Issue #1090*, Adafruit/CircuitPython, Dec. 2018. [Online]. Available: https://github.com/adafruit/circuitpython/issues/1090
[14] A. N. Sloss, D. Symes, and C. Wright, *ARM System Developer's Guide: Designing and Optimizing System Software*. Morgan Kaufmann, 2004. (Chapter 8 discusses exception and fault handling strategies).
[15] P. Koopman, "Better Embedded System Software," *Dr. Dobb's Journal*, vol. 32, no. 2, pp. 42-49, Feb. 2007.
[16] Espressif Systems, "ESP-IDF Wi-Fi API Guide - SoftAP," 2022. [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/wifi.html#esp32-wi-fi-softap
[17] M. Schwartz, J. Takkinen, and M. Collins, "Creating a RESTful API on an ESP32," *in Proceedings of the 2019 ASEE Annual Conference & Exposition*, Tampa, FL, 2019.
[18] MDN Web Docs, "The WebSocket API (WebSockets)," Mozilla. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API. Accessed: Jul. 7, 2025.