Software Formalization

Year: 2023 Semester: Spring Team: 18 Project: RDNT

Creation Date: February 18, 2023 Last Modified: February 18, 2023

Author: Kahaan Patel Email: pate1410@purdue.edu

Assignment Evaluation:

| **Item** | **Score (0-5)** | **Weight** | **Points** | **Notes** |
| --- | --- | --- | --- | --- |
| **Assignment-Specific Items** | | | | |
| **Third Party Software** |  | x2 |  |  |
| **Description of Components** |  | X3 |  |  |
| **Testing Plan** |  | x3 |  |  |
| **Software Component Diagram** |  | x4 |  |  |
| **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*

1.0 Utilization of Third Party Software

For the following project we will use the following third party libraries:

ESP-DSP component library [1]

* This library has very low level code for digital signal processing with the ESP-IDF framework. Many parts of the library are optimized for computation on the ESP32 chip.
* We will use this library for FFT calculation. This has FFT calculations optimized for our chip so we can very quickly sample and isolate frequencies.
* This does not require any attribution or licensing.

ESP32\_LED\_STRIP component library [2]

* This library has streamlined processes to control the PWM waves required to drive the LEDS. This will make our job much easier when taking frequencies from the FFT operations.
* This does not require any attribution or licensing.

2.0 Description of Software Components

We will complete all sampling of input analog signals using our ADC. We are using the Continuous Mode Driver API that is native to the ESP-IDF. We are using the API with custom configuration. The fields that we are configuring include the size of each sample frame, the size of the total sample buffer, the callback that will be called when the ADC has completed filling a sample frame.

The main firmware that will be run on our microcontroller is the FFT algorithm from the ESP-DSP library, a component library of the ESP Framework. This will be used as the backbone for our analog signal processing, which then translates input signals into commands to send to the LEDs. We are using this library without substantial changes but configuring to the sample configurations used by the ADC driver.

The Android application used to control the LEDs’ configuration is being developed by our team through using Gatt Profile and UART on ESP32 devices. We will be using the Gatt library on android to establish connection and send/receive messages. Additionally for the color picking tool we would be using guhoang:colorpicker. This will be used on the user side to choose colors and set a static color on the led strips. Other than that we will be utilizing the standard library to set pages and buttons for user input and onClick listeners for implementing actions.

To summarize :

The Android app, the firmware for the microcontroller, and the signal processing algorithms make up the project's three core software components.

Android App: The Android app is in charge of controlling the user interface and giving the user a way to communicate with the system. The onboard analog microphone, an AUX-in port, and a Bluetooth signal are the three various audio sources that can be chosen by the user. The software will also provide the user the option to choose from a variety of designs and color palettes for the LED strips. The software will accomplish this by utilizing a variety of Android components, including activities, fragments, and views, and Bluetooth to connect with the microcontroller.

Microcontroller Firmware: The firmware of the microcontroller is in charge of reading input samples from the chosen audio source and applying the signal processing algorithms to them. It will read the input signal using an analog-to-digital converter (ADC) and reduce noise. After that, it will separate frequencies from the signal using an FFT technique. The microcontroller will use this processed data to transmit messages to a message sender, which will then operate the LED strips using the FFT output. Also, the microcontroller will control the logic for switching between modes and utilize Bluetooth to connect with the Android app.

Signal Processing Algorithms: The signal processing algorithms are in charge of lowering the input sample's noise level and separating the signal's frequencies. By combining the input signal with a Gaussian function, the 1-D Gaussian filter lowers noise on the signal. The noise-reduced signal is subsequently transformed from the time domain to the frequency domain by the FFT technique. By providing PWM signals to the data port of the LED strips, the microcontroller drives the LED strips using this frequency data.

In conclusion, the Android app connects to the microcontroller over Bluetooth and offers the user interface. The firmware of the microcontroller reads input samples from the chosen audio source, applies signal processing algorithms to them, and then manages the LED strips. The Gaussian filter and FFT method are used in the signal processing algorithms to remove noise from the input sample and separate frequencies from the signal.

3.0 Testing Plan

1. To test our ADC Continuous Mode Driver we will connect our ADC input pin to multiple DC signals from the AD2 wave generator. Then we will compare the mean of many samples (multisampling) to confirm that our sample frames have readings acceptable to our signals. We must test the ADC driver first because without correct voltage readings, nothing sent to the FFT operations will be accurate, regardless of the correctness of the FFT operations.
2. FFT will be tested thoroughly, starting by sending in a controlled analog signal generated by an AD2. Using this, we will know what the FFT should find as the fundamental frequency and return that value. Once we have proven that the FFT can handle a single constant sine wave, we can set the AD2 to sweep through a range of frequencies and see if it can handle transitions between frequencies. After this has been proven to work, we can move onto sending analog signals through the AUX jack or Microphone. This will get the algorithm to try to react to actual music, which will confirm full functionality of our project.
3. How to test Android App : The features excluding ESP32 will be tested using the simulator and making sure the button and components that work on click are responding. To check the connection , we will be setting up the bluetooth services on both sides and ensure the messages are sent and received for both the APP and device. Other then that:
   1. Test the app's user interface to see if it is simple to use and intuitive.
   2. To make the app more user-friendly, gather user feedback.
   3. To guarantee that it functions on all supported devices, test the app on several Android devices.
   4. To make sure the app is compatible with all Android versions, test it on various Android versions.
   5. Check the app's response and load times to make sure they are quick and responsive.
   6. To make sure the software is not consuming excessive amounts of memory, test its memory use.
   7. To make sure that user data is secure, test the app's data encryption.
   8. To make sure that only users who have been given permission may use the app's features, test the app's authentication and authorization.
   9. This testing plan, in its entirety, involves testing each project component to ensure that it is operating as intended and that the software is user-friendly, compatible, quick, and secure.

4.0 Sources Cited:

[1] “Espressif DSP Library,” *ESP-DSP Library — Espressif DSP Library v1.2.0-17-g94321ab documentation*. [Online]. Available: https://espressif-docs.readthedocs-hosted.com/projects/esp-dsp/en/latest/index.html. [Accessed: 18-Feb-2023].

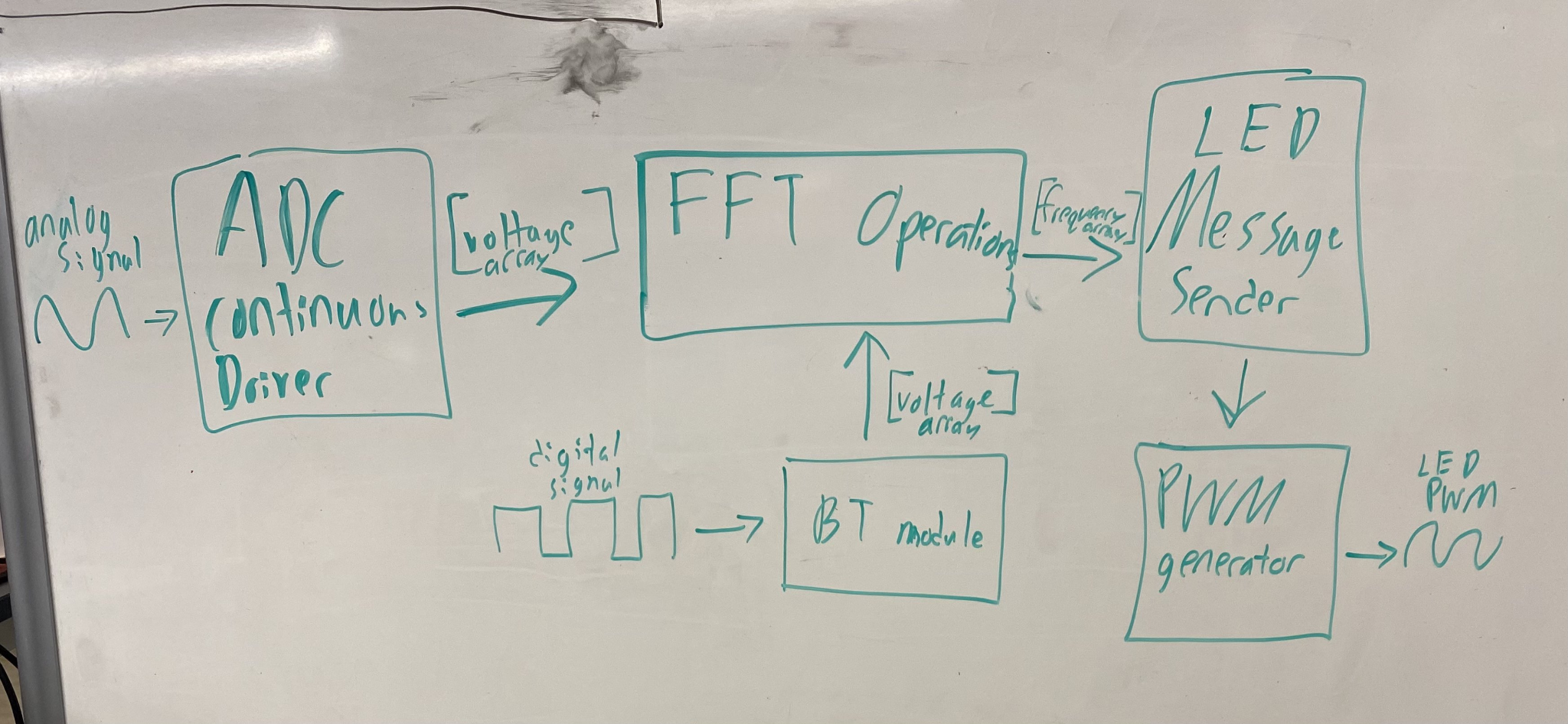
[2]L. Bruder, “Lucas-Bruder/ESP32\_LED\_STRIP: ESP32 led Strip Library,” *GitHub*. [Online]. Available: https://github.com/Lucas-Bruder/ESP32\_LED\_STRIP. [Accessed: 18-Feb-2023].

[3] Android Developers. "BluetoothGatt | Android Developers," September 29, 2021. <https://developer.android.com/reference/android/bluetooth/BluetoothGatt>.

[4] Android Developers. "BluetoothAdapter | Android Developers," September 29, 2021.<https://developer.android.com/reference/android/bluetooth/BluetoothAdapter>.

Appendix 1: Software Component Diagram

*Use this appendix to provide a diagram of the software components of your design. Depending on the nature of your design, multiple software component diagrams or function call structures may be appropriate.*

**

