Arbitrary Waveform Generator

Leandro Bracho
Nick Joseph
Oscar Medrano
Robin Smithson

FINAL REPORT

REVISION – 1
30 April 2024

FINAL REPORT
FOR
Arbitrary Waveform Generator

TEAM 63

APPROVED BY:

_____
Robin Smithson Date

_____
Prof. Lusher Date

_____
T/A Date

Table of Contents

Arbitrary Waveform Generator

Leandro Bracho
Nick Joseph
Oscar Medrano
Robin Smithson

CONCEPT OF OPERATIONS

REVISION – 4
30 April 2024

CONCEPT OF OPERATIONS
FOR
Arbitrary Waveform Generator

TEAM 63

APPROVED BY:

_____
Robin Smithson Date

_____
Prof. Lusher Date

_____
T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
-	9/14/2023	Arbitrary Waveform Generator		Draft Release
1	9/28/2023	Arbitrary Waveform Generator	Robin Smithson	Revision 1
2	11/07/2023	Arbitrary Waveform Generator	Robin Smithson	Revision 2
3	12/3/2023	Arbitrary Waveform Generator	Robin Smithson	Revision 3
4	04/30/2024	Arbitrary Waveform Generator	Robin Smithson	Revision 4

Table of Contents

List of Tables

No table of figures entries found.

List of Figures

1. Executive Summary

Having an easy-to-use arbitrary waveform generator (AWG) is important for the debugging of electrical systems. Traditional AWGs tend to be both costly and challenging to navigate. Our goal is to redefine this experience by developing an affordable AWG that is not only easy to use but also cost effective. This objective will be achieved through the implementation of a budget-friendly AWG, controllable via Bluetooth connectivity with a dedicated mobile app. The app allows users to select waveform types, a freehand mode, and set frequency in Hertz, enhancing the overall usability of AWGs.

2. Introduction

Test validation is a vital step in the design and fabrication of circuit boards for any electrical engineering project. One of the most valuable tools for this is the arbitrary waveform generator (AWG) which enables the generation of real-world waveforms to stimulate the circuit. To enhance the usability of an AWG, a mobile interface will be added to replace the many buttons and knobs that are usually found in standard AWGs. By designing an AWG with a mobile interface, anyone with an Android smartphone can connect to the AWG to create their own waveform. This project focuses on developing a small-scale AWG that is cheaper and more compact than other generators while improving the user experience.

2.1. Background

There are AWGs on the market that allow users to produce real-world waveforms along with the usual preset waveforms such as the sine, square, triangle, and sawtooth waveforms. The problem with standard AWGs in the current market is how unintuitive they can be to the average user. With countless buttons and knobs to work with, it can be confusing to operate the AWG. Our design takes away all the confusion by providing the user with the ability to work an AWG with only their phone. There are no current AWGs that are programable through a user's phone, by allowing the user a much easier way to use an AWG without any restrictions in the functionality, the AWG becomes a more powerful and intuitive tool.

2.2. Overview

The primary purpose of an AWG is to assist in validating circuit design by generating a waveform to stimulate the circuit. This system is used to create and generate simple and complex waveforms. The creation of waveforms will be determined through an app where you can either draw your own waveform or pick a premade waveform. The waveform's amplitude, frequency, offset, and data points can all be edited through the app. From the app the waveform will be

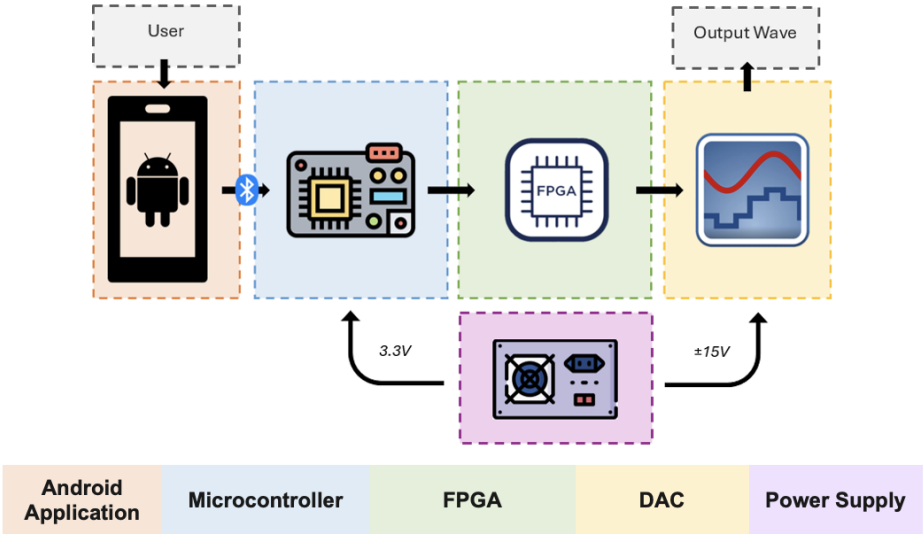


Figure 1. AWG Block Diagram

converted into a CSV file and exported to the AWG. In the AWG, the CSV file is converted from its digital information to analog values that can be output as a voltage waveform.

2.3. Referenced Documents and Standards

- IEEE Standard for Verilog IEEE 1364-2005
- IEEE Bluetooth communication standards: IEEE 802.15.1

3. Operating Concept

3.1. Scope

This project is designed to provide engineers with a technical challenge that will incorporate concepts of hardware and software to build a fully functioning arbitrary waveform generator (AWG). The AWG will interact with a mobile app that will function as the main user interface. The app will have preloaded waves the user can choose from and adjust such as sine, sawtooth, square, and triangle wave, while also having a "Freeform" mode that will allow the user to draw any wave with their finger. The app will upload the wave data to a microcontroller through Bluetooth. The data will then be analyzed, noise filtered out from the graph, and processed by a DAC into a wave the power supplies can create. The AWG will use external memory to store graph data and call on the data to produce the wave.

3.2. Operational Description and Constraints

The arbitrary waveform generator is a cheap alternative to the thousands of dollars an arbitrary waveform generator would cost on the market. With this waveform generator, you can draw a waveform on your phone and have that waveform transmitted to the AWG. In this transmission you can determine the voltage and frequency of the waveform. For this system to work the following requirements must be met:

- The phone must be connected via Bluetooth to the system.
- The voltage must be between ±10V.
- The frequency is between 1 Hz and 20 kHz.
- Produce a maximum of 65,535 data points.

3.3. System Description

The system is divided into five subsystems: the DACs, Microcontroller, Systems Coding, App Development, and Power System.

App Development: The app is the primary interface between the user and the AWG. For this initial build, the app is purely for Android devices. The amplitude, frequency, offset, and number of data points are editable parameters depending on the user's preferences. The user can choose between four predefined waves, or they can use a custom mode that allows the user to hand-draw a wave that will be interpreted into a list of points. The app will have a preview window that allows users to see their functions before uploading to the AWG. The app connects through an on-board Bluetooth module and transmits hex values of each data point to be processed by the AWG's internal components to form the wave.

Microcontroller: The FPGA will utilize Verilog to communicate the inputs to the microcontroller. Bluetooth will also be a necessary component of the microprocessor so that it can connect directly to the Android phone. The microprocessor then stores the data, and outputs it to the DAC.

FPGA: The systems coder is responsible for designing the code within the arbitrary waveform generator. They must take the transmitted data and manipulate the hardware to produce the desired waveform.

DAC: The system is made up of two four-quadrant multiplying DACs along with transimpedance amplifiers to generate a bipolar voltage output, a specification necessary to satisfy the output range requirement. With the addition of a +10V voltage reference, a ±10V output range is achieved. The DACs themselves are used to convert the given digital signals from the FPGA into analog signals enabling users to generate, control, and output any waveform through the app.

Power System: The power system is responsible for converting a +12V input into the required voltages to power the DAC system, Microcontroller, and FPGA using a system of DC-DC converters.

3.4. Modes of Operations

The AWG has only one operation, which involves generating a signal wave. By using the app, you will be able to choose which wave you want to generate and will be able to customize the amplitude, frequency, and desired data points of the graph.

3.5. Users

The AWG is intended for engineers to create low voltage waves to test different circuit applications with a variety of waveforms. The knowledge to use the AWG will be designed such that anyone can use the device with no prior training. The installation process involves installing the app on your phone and connecting it to the AWG through Bluetooth. Once connected the main interface will take place on the app.

3.6. Support

A user manual will be provided detailing how to use and operate the AWG, including a troubleshooting procedure if errors or operation complications occur. There will also be installation instruction of how to download the app with a printout diagram of the app’s UI and what each item in the app performs.

4. Scenario(s)

4.1. Generate a Waveform

When one needs to test systems by applying a known waveform, the arbitrary waveform generator will be able to apply a given waveform to the system. To accomplish this the user simply needs to draw the given waveform in the app and designate the appropriate amplitude, frequency, and offset in the app to generate the wanted waveform.

Utilizing this system has a variety of testing applications. It can be used for anything involving circuits, from electrical engineering lab experiments, home improvement projects, and hobbies.

4.2. Wireless Communication Testing

When needing to generate waveforms with specific schemes being able to draw the waveform and then set the amplitude and frequency. This allows for the simulation of diverse and complex scenarios in wireless testing.

4.3. Control System Analysis

To inject specific signals into a control system to identify its dynamic characteristics. This involves inputting a known signal into the system and observing its effects to determine its dynamic characteristics. An AWG works for this because it will be able to provide consistent known input.

5. Analysis

5.1. Summary of Proposed Improvements

The proposed arbitrary waveform generator will:

- Provide a cheap alternative to current AWGs.
- Allow the user to draw custom waveforms.
- Allow the user to do all the waveform programing from a mobile device.

5.2. Disadvantages and Limitations

The disadvantages of the waveform generator are:

- Limits on the maximum voltages
- Limits on the maximum and minimum frequencies
- Limits on the data points
- Limited number of waveforms

5.3. Alternatives

There are no exact alternatives for this AWG. There are slight changes that could be made to allow existing products to take the place of it. Buying an arbitrary waveform generator is a valid alternative. However, there are no AWGs that allow the user to draw the desired waveform on a smartphone and then generate said waveform.

5.4. Impact

By controlling the AWG through an app, enhanced functionality, usability, and flexibility is provided for a variety of applications in engineering, research, and education. The app provides an intuitive interface for waveform selection and adjustment. This reduces the learning curve for users and streamlines the process of generating waves. In addition, the ability to draw freehand waveforms expands the possibilities for waveform generation, allowing users to create custom waves tailored to their specific needs.

ARBITRARY WAVEFORM GENERATOR

Leandro Bracho
Nicholas Joseph
Oscar Medrano
Robin Smithson

FUNCTIONAL SYSTEM REQUIREMENTS

REVISION – 3
30 April 2024

FUNCTIONAL SYSTEM REQUIREMENTS
FOR
Arbitrary Waveform Generator

PREPARED BY:

_____
Author Date

APPROVED BY:

_____
Robin Smithson Date

_____
John Lusher, P.E. Date

_____
T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
-	9/28/2023	Arbitrary Waveform Generator		Draft Release
1	11/7/2023	Arbitrary Waveform Generator	Robin Smithson	Revision 1
2	12/3/2023	Arbitrary Waveform Generator	Robin Smithson	Revision 2
3	04/30/2024	Arbitrary Waveform Generator	Robin Smithson	Revision 3

Table of Contents

List of Tables

List of Figures

1. Introduction

1.1. Purpose and Scope

This document will cover each subsystem requirement for the arbitrary waveform generator (AWG). This project shall be an AWG that successfully interfaces with an Android smartphone. The user shall be able to generate waveforms, both preset and custom, from the face of their phone, that will be used to test validate circuits. The waveform shall be generated by either one of two ways. The user will select what function waveform they want, such as sine, ramp, square, and so on. Then, the amplitude, frequency, period, offset, phase, and other parameters will also be set. The other option shall be to draw a waveform with one's finger on the Android app. The app shall then record the waveform, from either of these options, and send the data points to SRAM, via a microcontroller with Bluetooth capabilities and an FPGA. This digital information shall then be sent to the digital to analog converter (DAC), which shall then output the desired waveform to the test circuit.

Figure 1. Project Conceptual Image

1.2. Responsibility and Change Authority

Robin Smithson, the team leader, is responsible for making sure all requirements for the AWG project are met. The sponsor, Dr. John Lusher, and the team leader, Robin Smithson, have authority to approve and make changes as seen fit. Subsystem responsibilities are broken down to the following team members:

- Leandro Bracho: Android App Development and Data Input
- Robin Smithson: MCU Design
- Nicholas Joseph: FPGA Design
- Oscar Medrano: DAC Output Design and Power System

2. Applicable and Reference Documents

2.1. Applicable Documents

The following documents, of the exact issue and revision shown, form a part of this specification to the extent specified herein:

Document Number	Revision/Release Date	Document Title
978-1-5044-4509-2	2/22/2018	IEEE Standard for System Verilog -- Unified Hardware Design, Specification, and Verification Language
0-7381-3335-3	6/14/2002	IEEE Standard for Telecommunications and Information Exchange Between Systems – LAN/MAN – Specific Requirements – Part 15: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Wireless Personal Area Networks (WPANs)
1-4244-1513-06	02/22/2008	J. Noseworthy and J. Kulp, "Standard Interfaces for FPGA Components," MILCOM 2007 - IEEE Military Communications Conference, Orlando, FL, USA, 2007, pp. 1-5, doi: 10.1109/MILCOM.2007.4455287.

Table 1. Applicable Documents

2.2. Reference Documents

The following documents are reference documents utilized in the development of this specification. These documents do not form a part of this specification and are not controlled by their reference herein.

Document Number	Revision/Release Date	Document Title
v1.8	02/09/2023	ESP32-WROVER-E & ESP32-WORVER-IE Datasheet
v3.3	10/13/2022	ESP32 Hardware Design Guidelines

Table 2. Reference Documents

2.3. Order of Precedence

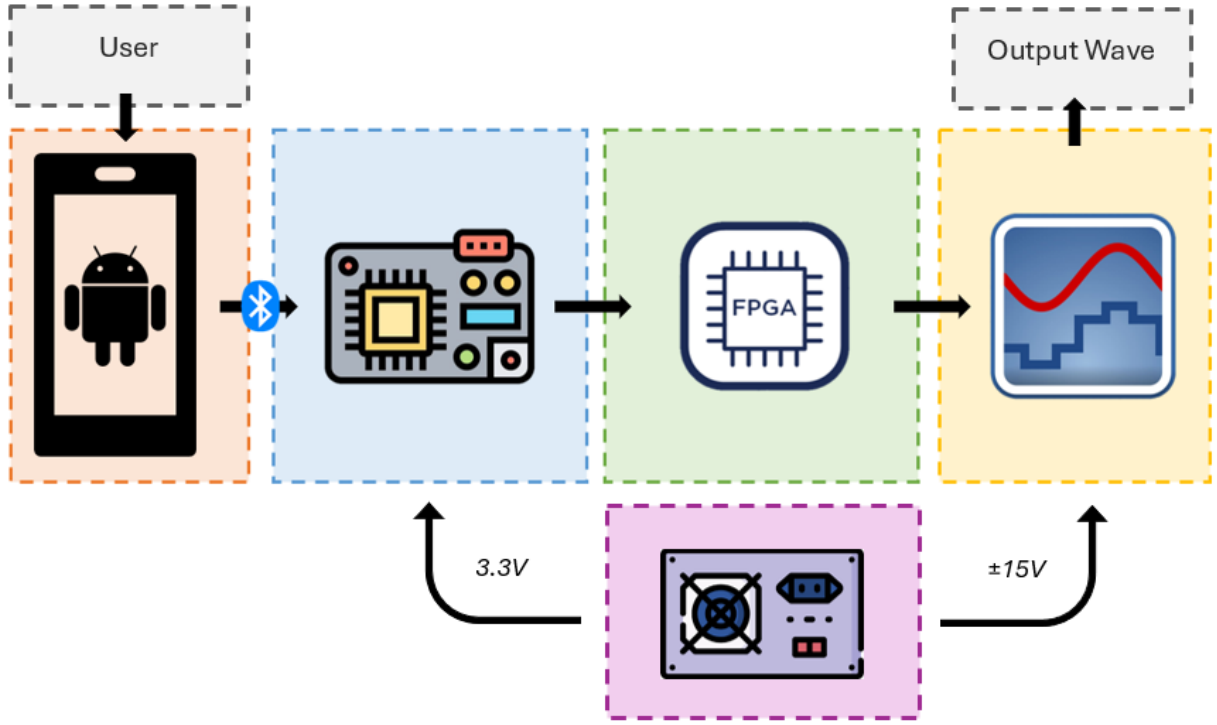
In the event of a conflict between the text of this specification and an applicable document cited herein, the text of this specification takes precedence without any exceptions.

All specifications, standards, exhibits, drawings, or other documents that are invoked as “applicable” in this specification are incorporated as cited. All documents that are referred to within an applicable report are for guidance and information only, except ICDs that have their relevant documents considered to be incorporated as cited.

3. Requirements

3.1. System Definition

The project requires the design of an Arbitrary Waveform Generator (AWG) that interfaces with a smartphone to output various waveforms. The role of the smartphone is to provide the user a way to send information on the desired waveform such as amplitude, frequency, and offset to the AWG through Bluetooth connection. The project has been broken down into its software and hardware components. The software includes app development and the use of the hardware description language, Verilog, to convert the user input into information that the AWG can use. The hardware includes the microcontroller design, the digital to analog converter (DAC) design, and the power system design.



Leandro Bracho	Robin Smithson	Nick Joseph	Oscar Medrano	Oscar Medrano
Android Application	Microcontroller	FPGA	DAC	Power Supply

Figure 2. Block Diagram of AWG

The mobile app is the primary interface between the user and the AWG. For this initial build, the app is purely for Android devices. The user can choose to generate a preset waveform, or they can draw a custom waveform that will be interpreted into a function. The amplitude, frequency, offset, and number of data points will be customizable in the app depending on the user's preferences. A

preview of the waveform will be available to allow users to see their functions before uploading to the AWG. The app connects to the AWG through a Bluetooth module on the MCU that will be able to receive the data points and store the data onto the AWG's external memory. The MCU shall set the clock rate frequency for the data points and export them to the FPGA. The FPGA is responsible for taking the information provided by the microcontroller storing that information as memory and then transferring the information to the DAC. The FPGA will store memory in RAM and then depending on if it is being read to or written to will continue to write into the RAM or begin reading the values over to the DAC. The role of the DAC is to transform the digital input from the FPGA into an electrical signal matching the desired waveform. The DAC itself outputs a current, so a transimpedance amplifier is necessary to convert the current output into a voltage signal.

3.2. Characteristics

3.2.1. Functional/Performance Requirements

3.2.1.1. Mobile Interface

The AWG shall have its interface on an Android app. The app will be the primary way the user interacts with, controls, and inputs waveforms to the AWG.

> *Rationale: This is a core system requirement. The main connection to the Arbitrary Waveform Generator will be a mobile app as this replaces the typical waveform generator user interface.*

3.2.1.2. Function Output

The AWG shall have the capabilities to output a sine, square, ramp, triangle, and hand-drawn function waves, with a frequency from 1 Hz to 20 kHz, and with up to 65,535 data points.

> *Rationale: This is the core system performance requirement as per request of the sponsor.*

3.2.2. Physical Characteristics

3.2.2.1. Mass

The mass of the Arbitrary Waveform Generator shall be less than or equal to 5 kilograms.

> *Rationale: This is a value determined to increase portabilty of the device and allow easier transportation for the customer by keeping the weight low enough for any adult to carry with one hand.*

3.2.2.2. Volume Envelope

The volume envelope of the Arbitrary Waveform Generator shall be less than or equal to 9 inches in height, 7.5 inches in width, and 8.25 inches in length.

Rationale: This is to be determined value by the sponsor but should not be any larger than a typical waveform generator.

3.2.3. Electrical Characteristics

3.2.3.1. Inputs

a. The presence or absence of any combination of the input signals in accordance with ICD specifications applied in any sequence shall not damage the Arbitrary Waveform Generator, reduce its life expectancy, or cause any malfunction, either when the unit is powered or when it is not.

b. No sequence of command shall damage the Arbitrary Waveform Generator, reduce its life expectancy, or cause any malfunction.

Rationale: By design, should limit the chance of damage or malfunction by user/technician error.

3.2.3.2. Power Consumption

The maximum peak power of the system shall not exceed 5 watts.

Rationale: This value has yet to be determined by the sponsor, but the device should not pull a lot of power.

3.2.3.3. Input Voltage Level

The input voltage level for the Arbitrary Waveform Generator shall be +12VDC.

Rationale: Input voltage has yet to be determined by the sponsor, but a standard wall wart adapter will be used to provide power.

3.2.3.4. External Commands

The Arbitrary Waveform Generator shall document all external commands in the appropriate ICD.

Rationale: The ICD will capture all interface details from the low level electrical to the high-level packet format.

3.2.3.5. Data Output

The Arbitrary Waveform Generator shall include an interface compatible with the data system.

Rationale: The Arbitrary Waveform Generator's data points pass directly from the customer's Android smartphone.

3.2.3.6. Diagnostic Output

The Arbitrary Waveform Generator shall include a diagnostic interface of LEDs for power and Bluetooth connection logging.

Rationale: Provides the ability to ensure proper connections are successful.

3.2.3.7. Connectors

The AWG shall use BNC connectors to deliver the waveform to the test circuit.

> *Rationale: The ports will be BNC compatible. The customer will have to supply their own BNC cable.*

3.2.4. Environmental Requirements

The Arbitrary Waveform Generator shall be designed to withstand and operate in the environments and laboratory tests specified in the following section.

> *Rationale: This is a requirement specified by our sponsor due to the intended versatility of the device.*

3.2.4.1. Humidity

The AWG, not including the Android device, shall not exceed 90% RH.

> *Rationale: The component with the least tolerance for humidity, the microcontroller, has an MSL at level 3, according to the ESP32-WROVER datasheet.*

3.2.4.2. Temperature

The AWG shall not exceed an internal temperature of 40 ºC.

> *Rationale: The component with the most temperature intolerance, microcontroller, cannot be in an environment greater than 40 ºC, according to the ESP32-WROVER datasheet.*

3.2.4.3. Ultrasonic Vibrations

The AWG shall not be exposed to any ultrasonic vibration, as it can cause malfunction to the system.

> *Rationale: This is a condition concerning the microcontroller, according to the ESP32-WROVER datasheet.*

4. Support Requirements

The Arbitrary Waveform Generator requires an android smartphone with Wi-Fi and Bluetooth capability to download the required application and provide inputs to the device. The operating system of the smartphone must be greater than Android 7.0 to use the app. The system will include (1) Arbitrary Waveform Generator, (1) Power Cable, (1) Quick Start Guide. The quick start will include infographics of how to turn on the device, how to use the device, how to download the required application, how to navigate the app, and how to contact customer support.

Appendix A: Acronyms and Abbreviations

AWG	Arbitrary Waveform Generator
DAC	Digital-Analog Converter
FPGA	Field Programmable Gate Arrays
Hz	Hertz
ICD	Interface Control Document
kHz	Kilohertz (1,000 Hz)
LED	Light-emitting Diode
mA	Milliamp
MCU	Microcontroller
MHz	Megahertz (1,000,000 Hz)
MSL	Moisture Sensitivity Level
mW	Milliwatt
PCB	Printed Circuit Board
RH	Relative Humidity
TBD	To Be Determined

Appendix B: Definition of Terms

Arbitrary Waveform Generator

Leandro Bracho
Nicholas Joseph
Oscar Medrano
Robin Smithson

INTERFACE CONTROL DOCUMENT

REVISION – 2
30 April 2024

INTERFACE CONTROL DOCUMENT
FOR
Arbitrary Waveform Generator

PREPARED BY:

_____
Author Date

APPROVED BY:

_____
Robin Smithson Date

_____
John Lusher II, P.E. Date

_____
T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
-	9/28/2023	Arbitrary Waveform Generator		Draft Release
1	12/3/2023	Arbitrary Waveform Generator	Robin Smithson	Revision 1
2	04/30/2024	Arbitrary Waveform Generator	Robin Smithson	Revision 3

Table of Contents

List of Figures

1. Overview

This document is provided to detail how the Arbitrary Waveform Generator's internal subsystems are connected, along with how those components communicate with the mobile application. It will discuss the various inputs and outputs of each subsystem and how the components within each subsystem work.

2. References and Definitions

2.1. References

v1.8
ESP32-WROVER-E & ESP32-WORVER-IE Datasheet
9 Feb 2023

v3.3
ESP32 Hardware Design Guidelines
13 Oct 2022

DAC8820 Datasheet (Rev. D)
19 Feb 2008

2.2. Definitions

A	Amp
AWG	Arbitrary Waveform Generator
DAC	Digital to Analog Converter
FPGA	Field Programmable Gate Array
mA	Milliamp
mW	Milliwatt
MCU	Microcontroller
MHz	Megahertz (1,000,000 Hz)
TBD	To Be Determined
V	Volt
W	Watt

3. Physical Interface

3.1. Weight

3.1.1. Microcontroller PCB

Component	Weight (oz)	Number of items	Total Weight (oz)
ESP32-WROVER IE	0.217191	1	0.217191
Capacitor	0.141096	4	0.564384
UART	1.76	1	1.76
Switch	0.073833	2	0.147666
Resistor	0.009161	2	0.018322
Antenna	0.017178	1	0.017178
LED	0.035274	1	0.035274
Terminal Block	0.917123	1	0.917123
Pin Connector	0.282192	1	0.282192

Table 1. Microcontroller Weight

3.1.2. FPGA

Component	Weight (oz)	Number of items	Total Weight (oz)
Board size	N/A	1	N/A
FPGA	N/A	1	N/A
RAM	N/A	1	N/A
Pin connector	N/A	1	N/A
40-Pin Connector	N/A	1	N/A

Table 2. FPGA Weight

3.1.3. DAC PCB

Component	Weight (oz)	Number of items	Total Weight (oz)
DAC8820	N/A	2	N/A
OPA2277UA/2k5	N/A	2	N/A
MAX8776BESA+	N/A	1	N/A
40-Pin Connector	N/A	1	N/A
Capacitor	N/A	8	N/A
BNC Connector	N/A	1	N/A
Terminal Block	N/A	3	N/A

Table 3. DAC Weight

3.1.4. Power System PCB

Component	Weight (oz)	Number of items	Total Weight (oz)
IAB0112D15	N/A	1	N/A
PDSE1-S12-S5-M	N/A	1	N/A
R-78K3.3-2.0	N/A	1	N/A
TSR 1-2412	N/A	1	N/A
Power Jack	N/A	1	N/A
Capacitor	N/A	12	N/A
Inductor	N/A	3	N/A
Terminal Block	N/A	7	N/A

Table 4. Power System Weight

3.2. Dimensions

The dimensions are in millimeters.

3.2.1. Dimension of Microcontroller

Component	Length	Width	Height
ESP32-WROVER IE	31.4	18	3.3
Resistor	1	0.5	0.40
Capacitor (0.1 µF)	0.25	0.13	0.33
Capacitor (22 µF)	0.6	0.3	0.33
UART	914.4	N/A	N/A
Switch	5.2	5.2	1.5
Antenna	84.9	9	0.10
LED	3.5	3.4	0.9
Terminal Block	8	7	12.5
Pin Connector	25	10.1	7

Table 5. Microcontroller Dimensions

3.2.2. Dimension of FPGA

Component	Length	Width	Height
Board size	153	124	3
FPGA	20	20	1.5
RAM	18.54	10.29	1.1
Pin connector	25	10.1	7
40_Pin Connector	102.1	5	7.1

Table 6. FPGA Dimensions

3.2.3. Dimension of DAC

Component	Length	Width	Height
DAC8820	10.2	7.8	2
OPA2277UA/2K5	5	5.8	1.75
MAX876BESA+	5	6.2	1.75
40-PIN Connector	102.1	5	7.1
BNC Connector	6.3	6.3	13.9
Capacitor	1.6	0.8	0.87

Table 7. DAC Dimensions

3.2.4. Dimension of Power System

Component	Length	Width	Height
IAB0112D15	19.6	6	10.1
PDSE1-S12-S5-M	13.2	8.5	7.25
R-78K3.3-2.0	11.5	8.5	17.5
TSR 1-2412	11.7	7.5	10.1
Power Jack	14.4	9	11
Capacitor	N/A	N/A	N/A
Inductor	N/A	N/A	N/A
Terminal Block	8.2	8.2	16.1

Table 8. Power System Dimensions

3.3. Mounting Locations

The AWG can be taken virtually anywhere that the user can go. It is recommended that it is placed on the same surface the test circuit is on, and that the Android smartphone is close by on the same surface. The surface should be a solid, sturdy, and clean surface.

4. Thermal Interface

In this stage of development, there does not appear to be a need for any individual heatsinks on the FPGA, Microprocessor, and DAC. With this, however, there will need to be a fan added onto the box that we assemble the whole project into to provide a constant flow of air so that nothing overheats.

5. Electrical Interface

5.1. Primary Input Power

5.1.1. Primary AWG Unit

The AWG shall be powered by a standard wall wort that can be plugged into an outlet. The power supplied shall be delivered to each electrical component as fit.

5.2. Voltage and Current Levels

5.3 Signal Interfaces

Component	Voltage [V]	Current [A]	Power [W]
UART	5	TBD	TBD
ESP32-WROVER IE	3.3	0.5	1.65
JTAG	3.3	TBD	TBD
FPGA	1.89	.04	N/A
SRAM	3	.025	N/A
DAC8820	3.3	TBD	TBD

Table 9. Recommended Voltage and Current

5.3.1. Android Application

The Android app shall provide the interface to create and view signals imported to the AWG.

5.4. User Control Interface

5.4.1. Android Application

The Android app shall provide the user control components in addition to being the signal interface. From the app, the user will be able to control the wave, frequency, amplitude, number of points, offset, and several other settings.

6. Communications/Device Interface Protocols

6.1. Wireless Communications

6.1.2. Bluetooth

The microcontroller features a Bluetooth capability so that the AWG will be able to connect to the smartphone. This communication is vital, as it is how the Android app will communicate the data points to the microprocessor.

Arbitrary Waveform Generator

Leandro Bracho
Nicholas Joseph
Oscar Medrano
Robin Smithson

SCHEDULE AND VALIDATION

REVISION – 2

Schedule:

	Sep-25	Oct-2	Oct-9	Oct-16	Oct-23	Oct-30	Nov-6	Nov-13	Nov-20	Nov-27	Dec-4
Android Application											
Design UI										Not Started	
Create Text Fields										In Progress	
Add User Input										Completed	
Implement Buttons										Behind Schedule	
Integrate Waveform Window											
Write Wave Functions											
Add Data Export											
Create Freehand Function											
Add Bluetooth Support											
Add Start Screen											
Microcontroller											
Create and Finalize MCU Schematic											
Order Microcontroller Parts											
Solder Components											
Establish Bluetooth Connection											
Import Sample Data											
Program Clock Rate											
Export Sample Data											

Schedule Continued:

	Sep-25	Oct-2	Oct-9	Oct-16	Oct-23	Oct-30	Nov-6	Nov-13	Nov-20	Nov-27	Dec-4
FPGA											
Order FPGA										Not Started	
Order RAM										In Progress	
Design board										Completed	
Order board										Behind Schedule	
Code FPGA											
Solder board											
DAC											
Schematic Design											
Order DAC IC											
Order Output Circuitry Parts											
First PCB Design											
Test Board											
Update as Necessary											
Project Introduction Presentation											
Project Update Presentation											
Final Presentation											
Project Subsystem Demo											
Final Report											

Schedule Continued:

	Jan-22	Jan-29	Feb-5	Feb-12	Feb-19	Feb-26	Mar-4	Mar-11	Mar-18	Mar-25	Apr-1	Apr-8	Apr-15	Apr-22	Apr-29
Android Application															
Assess Current Application Status															
Improve Bluetooth Scan															
Pair & Connect Via Bluetooth															
Transfer Data Via Bluetooth															
UI Improvements															
Microcontroller															
Edit MCU Schematic															
Program Bluetooth Connection															
Import Data															
Export Data															

Schedule Continued:

	Jan-22	Jan-29	Feb-5	Feb-12	Feb-19	Feb-26	Mar-4	Mar-11	Mar-18	Mar-25	Apr-1	Apr-8	Apr-15	Apr-22	Apr-29
FPGA															
Redesign FPGA board															
Solder New Board															
Test System															
testing code															
DAC & Power System															
Schematic Design (DAC)															
Schematic Design (Power Sys)															
Order New Parts															
Solder Boards															
Test DAC System															
Test Power System															

Schedule Continued:

	Jan-22	Jan-29	Feb-5	Feb-12	Feb-19	Feb-26	Mar-4	Mar-11	Mar-18	Mar-25	Apr-1	Apr-8	Apr-15	Apr-22	Apr-29
Project Introduction Presentation															
Project Update (1) Presentation															
Project Update (2) Presentation															
Project Update (3) Presentation															
Project Update (4) Presentation															
Final Presentation															
Project Demo															
Final Report															
Integration															
Power System Designed															
App and MCU Connection															
Complete PCB Schematic															
Final PCBs Ordered															
Parts Still Needed Ordered															
Final PCBs Soldered															
Debugging and Testing															
Final System Adjustments															

Validation Plan:

Test Name	Success Criteria	Methodology	Status
Leandro Bracho			
User Interface	The UI appears correctly in portrait and landscape orientation.	On app homepage check for visual errors in portrait orientation, then flip to landscape and check again.	SUCCESS
Buttons	Each button is clickable with a working dropdown menu	Each button subsequent dropdown item will be pressed, and their options should perform its task.	SUCCESS
Wave Functions	Each wave can be selected and outputs their function correctly.	Each function will be selected and given preset data and user input data to test if the wave outputs correctly.	SUCCESS
User Input	Users can input data and only the acceptable values are passed on	A range of numbers, special characters, and null values will be passed into the inputs.	SUCCESS
Freeform Mode	A user can draw a wave within the chart area	The mode will be selected, and a wave will be drawn to test if the input field's values are updated.	SUCCESS
Export	The app can import a txt file filled with data	Data will be exported, the downloads folder will be checked, and the file will contain its expected values.	SUCCESS
Bluetooth	The app has Bluetooth properties and can scan for local devices.	The device will prompt the user to enable Bluetooth permissions in the app and the app will be able to scan for local Bluetooth Devices.	SUCCESS
Nicholas Joseph			
Power	*FPGA Power*	Need to measure the values inputting into each component on the board	FAILED
FPGA input	The FPGA is written to correctly	Using a coding language write tests to validate that the input leads to the expected results	SUCCESS
FPGA output	The FPGA reads out correctly	Using a coding language write tests to validate that the output leads to the expected results	SUCCESS
SRAM Comm	The FPGA can communicate with the RAM	Have an input write to the S ram, have an oscilloscope measure that there is an input to the RAM	SUCCESS
Writing to Ram	The FPGA can write to the ram	Have the FPGA write to the RAM, measure with an oscilloscope	SUCCESS
Reading from Ram	The ram reads back to the FPGA the correct values	Have a read write cycle run from the FPGA to the RAM and back, ensure the numbers match	SUCCESS
Correct Timing	The timing reading to and from the FPGA is correct	Have multiple cycles run and ensure erroneous errors do not pop up in the system	SUCCESS
FPGA + DAC	The whole system reads in and out correctly	Run system from start to finish and ensure the expected measurements are happening	SUCCESS
FPGA + MCU	The MCU can write to the FPGA correctly	Write a string of values and measure the outputs of a routed pin to see if it is working correctly	

Validation Plan Continued:

Test Name	Success Criteria	Methodology	Status
Oscar Medrano			
Power	DAC, Op-Amp, and Voltage Reference receive power	Need to measure +3.3V on the DAC, +15V and -15V on the op-amp, and +15V on the voltage reference.	SUCCESS
Voltage Reference	Voltage Reference IC outputs a fixed 10V	10V needs to be measured coming out of the MAX876BESA+	SUCCESS
DAC Input	DAC receives 16-bit input and control inputs	Using an AD2, the DAC will receive 16-bits of input and control inputs	SUCCESS
DAC Output	DAC outputs the expected unbuffered current.	The DAC will output the expected 1.66mA current during full-scale operation with a 10V reference.	SUCCESS
Output Circuitry	OPA2277UA/2K5 allows for a +/-10V output	A +/-10V range is measured from the 16-bit range: 0000h-FFFFh	SUCCESS
Robin Smithson			
Bluetooth	The MCU successfully turns on Bluetooth	The app will confirm connection between the phone and MCU is successful, and data will be able to export to the MCU	SUCCESS
Inputs	The MCU successfully imports sample data points	The MCU Serial terminal will display the imported data points with its index number next to it to see that all data points have been imported via Bluetooth	SUCCESS
Data Output	The MCU successfully exports all data points	The clock, MOSI, and Chip Select are seen on the oscilloscope as expected given the imported data points	SUCCESS

Arbitrary Waveform Generator

Leandro Bracho
Nick Joseph
Oscar Medrano
Robin Smithson

SUBSYSTEM REPORTS

REVISION – 1
30 April 2024

SUBSYSTEM REPORTS
FOR
Arbitrary Waveform Generator

TEAM 63

APPROVED BY:

_____
Robin Smithson Date

_____
Prof. Lusher Date

_____
T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
-	12/3/2023	Arbitrary Waveform Generator		Draft Release
1	04/30/2024	Arbitrary Waveform Generator	Robin Smithson	Revision 1

Table of Contents

List of Figures

1. Introduction

The arbitrary waveform generator (AWG) will output any waveform the user desires. Data points will be entered on the app, either by specified values, or by a hand-drawn function. The app will connect to the microcontroller of the AWG via Bluetooth and will import the data points to the microcontroller. The microcontroller will then export the data points to the FPGA, which will then export the data points to the digital-to-analog converter, outputting the waveform into a circuit.

2. Android App Interface Subsystem

2.1. Subsystem Introduction

The Android Application Interface is designed for user interaction to supply data to the Arbitrary Waveform Generator (AWG). The app enables users to input a frequency, amplitude, offset, and desired number of data points. Users can toggle between different wave types using the mode select feature or by returning to the home screen and selecting a new option. Additionally, the app provides a Bluetooth button that displays the current connection status, an undo button to reset hand-drawn waves, and a settings gear where the user can either download or export data.

2.2. Subsystem Details

2.2.1. Design

The application features two screens for user navigation. The start screen serves as a landing page where the user can connect to the device and select which wave they want to start with. The wave screen encompasses the wave graph, input fields, a mode switch, a Bluetooth status, an undo button, a home button, and a settings option.

2.2.1.1. Start Screen

The start screen for the AWG application, as seen in Figures 1-3, provides a simple interface for users to connect with the AWG and select the mode they will start with. If the user does not want to pair to the device the user can skip the connection and then select a mode with an option to connect, shown in Figure 3. After selecting a mode, the app transitions to the home screen and will produce a graph of the selected mode.

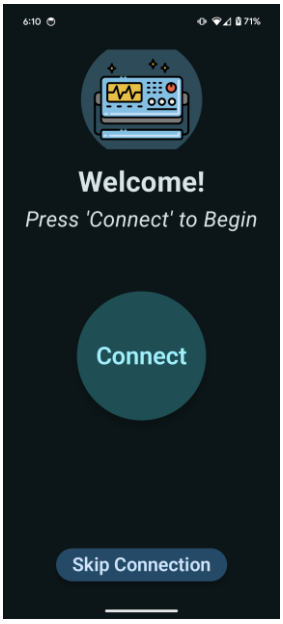


Figure 31. Start Screen

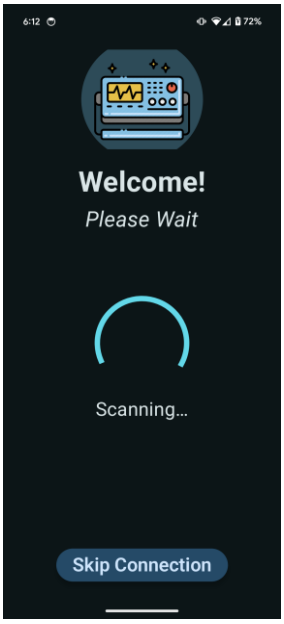


Figure 42. Scanning Screen

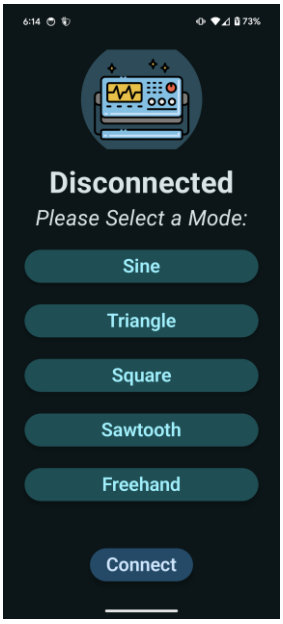


Figure 53. Connection Screen

2.2.1.2. Home Screen

The home screen contains all the main functionality of the app. As mentioned in the design subsection, this includes the wave graph, input fields, a mode switch, a Bluetooth status, an undo button, a home button, and a settings option. Figures 4-8 show each of the five different wave options with smaller details included in each figure. In Figure 4 there is an example of what would happen if the user inputs an unsupported number or symbol. The input field turns red and when attempting to download the file a toast appears informing the user to update the text field, this toast will also display if an export were attempted with an incorrect value. Figure 5 shows the dropdown menu for the settings icon where the user can either download or export. Figure 6 shows what happens when the Bluetooth symbol is pressed if the AWG is not connected, the symbol changes to indicate it is searching and a toast appears with the text “Scanning…”. Figure 7 demonstrates a sawtooth wave with all the input fields edited and the graph correctly representing these changes. Figure 8 displays the progression bar at the top of the screen that displays during a download or an export. The home button will return to the user to the screen shown in Figure 3 and will update the title text based on the connection status. The Bluetooth button has three states, disconnected, scanning, and connected to display to the user what state Bluetooth is currently in. The graph will update in real time as the user inputs new data to reflect the updated parameters. These changes include the x and y axis values updating and the x axis title changing to reflect if the values shown are in milliseconds or nanoseconds. The number of data points displayed is also updated accordingly. The undo button only displays when the Freehand mode is selected and will allow the user to reset their drawings to make a new wave.

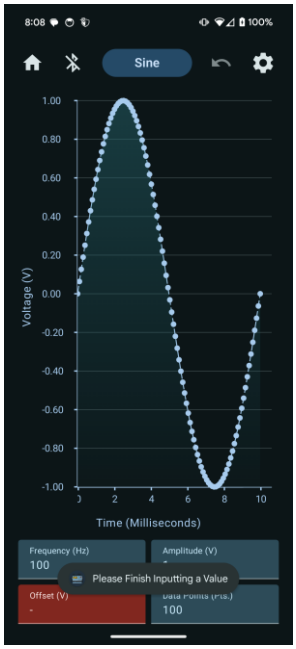


Figure 64. Sine Wave

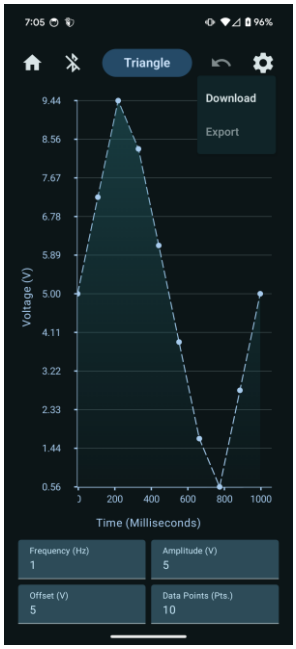


Figure 75. Triangle Wave

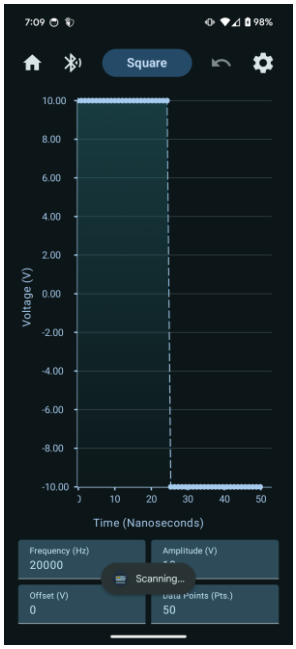


Figure 86. Square Wave

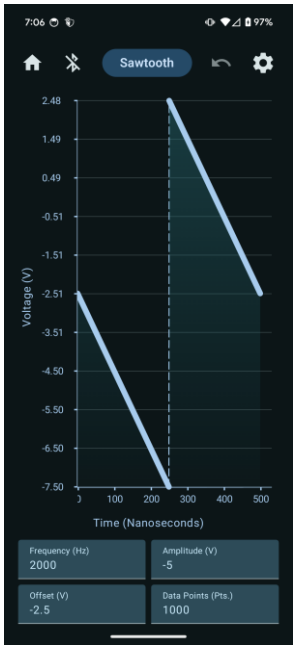


Figure 97. Sawtooth Wave

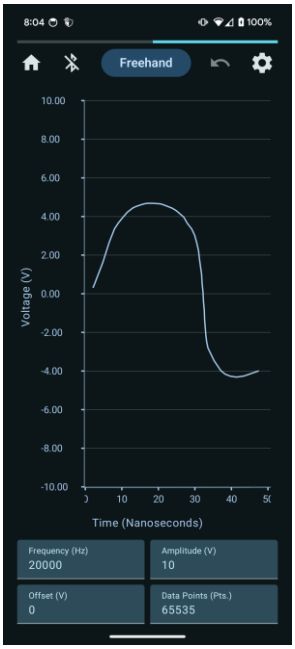


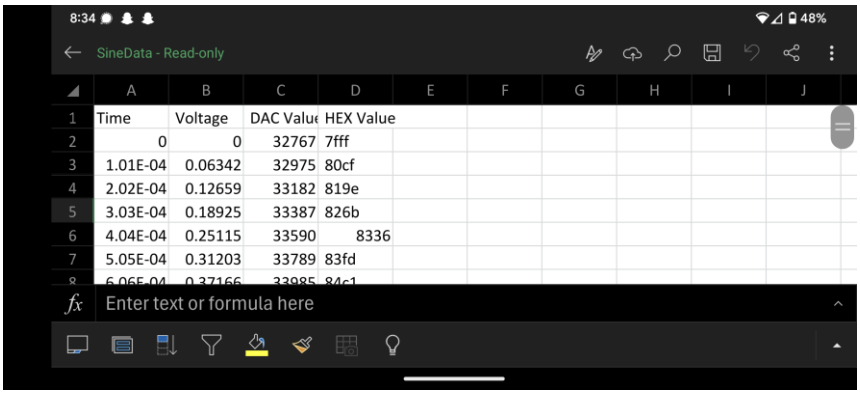
Figure 108. Freehand Wave

2.2.2. Function Implementation

There are several backend functions the application utilizes to export the correct data, display the correct waveforms, and correctly represent a hand drawn line.

2.2.2.1. Download and Export

Whenever the download function is called a .csv file is created and given a name that corresponds to the data that was exported. The file contains a header row with the titles Time, Voltage, DAC Value, and HEX Value with a column associated with each title. Each column contains their respective values and provides the user with the values created based on their wave type and input parameters. To help prevent crashing of the device the export function is treated as an asynchronous routine that frees up space from the main thread. Additionally, the content is written into the file in 8 kilobyte chunks to not overload the application. Figure 9 provides an example of an exported .csv file created for the sine function. When exporting data, only the HEX values shown in Figure 9 are written over to the microcontroller in 512-byte chunks to not overload the microcontroller's buffer size. The export happens in the background, and the download, allowing the user to continue to navigate the application. A drawback with these functions is the inability to cancel them, forcing the user to either restart the application to send new information or to wait for the functions to be completed. A progress bar, as seen in Figure 8, is displayed at the top so the user has a rough estimate of how long it will take for a file to export to the microcontroller.



	A	B	C	D
1	Time	Voltage	DAC Value	HEX Value
2	0	0	32767	7fff
3	1.01E-04	0.06342	32975	80cf
4	2.02E-04	0.12659	33182	819e
5	3.03E-04	0.18925	33387	826b
6	4.04E-04	0.25115	33590	8336
7	5.05E-04	0.31203	33789	83fd

Figure 119. Exported Sawtooth Data

2.2.2.2. Wave Graph Functions

The device generates wave graphs by employing mathematical equations corresponding to the four function graphs and passing in the user's provided frequency, amplitude, offset, and data points. Voltage points are then computed as a function of time within the chosen period. The data points determine the step interval between each point. For use in the download or export functions the voltage points are converted to a DAC value with an equation provided by the DAC's user manual and those integer values are converted to HEX. To prevent screen overcrowding during graphing, the device limits the plotted points after a certain threshold. Only the maximum number of points that won't visually alter the graph are displayed, avoiding unnecessary clutter.

2.2.2.3. Freehand Function

The freehand mode utilizes various functions to accurately display and export the user-drawn line on the graph. The displayed line is composed of multiple smaller lines, with start and end points represented as offsets from the top left corner of the graph. These lines are stored in a list, forming the complete drawn line. During the export of the freehand wave, the number of line points is calculated based on the user's desired data points. If there are too many line points, adjacent lines are combined by using the start point of the first line and the end point of the second line,

effectively reducing the total points by one. Conversely, if there are too few line points, a line is split into two by calculating its midpoint. The midpoint becomes the new start and end point for two lines, increasing the total points by one. Once the total line points match the user's desired data points, each line's start and end points undergo linear interpolation to convert them into corresponding voltage and time values. These new values are stored in a list and subsequently exported.

2.3. Subsystem Validation

Each application component undergoes several validation tests to ensure the component is functioning correctly. Before a new component is added the currently tested component must visually appear correct and have basic functionality with a small margin of error allowed before a new component can be added and tested. If the component can pass its validation test its status will be updated from “FAILED” to “PASS”. These completed tests can be seen in Figures 1-8 as they are the final screens used after passing. Below is a table logging all validation tests.

Component	**Test**	**Status**
Start Screen	Display correctly in portrait & landscape	PASSED
	Request Bluetooth permissions on app launch	PASSED
Buttons	‘Connect’ button attempts to scan and connect to AWG	PASSED
	‘Skip Connection’ button moves to the mode select screen	PASSED
	Mode buttons take the user to their respective graphs	PASSED
Home Screen	Display correctly in portrait & landscape	PASSED
	Buttons produce correct dropdown menus	PASSED
	Dropdown options load correct option	PASSED
	Dropdown dismissed on any press	PASSED
	‘Mode’ button title reflects current mode	PASSED
	‘Home’ button returns user to the start screen	PASSED
Input Fields	Appear correctly in portrait & landscape	PASSED
	Allow only numbers in the field	PASSED
	Reject null or special characters	PASSED
	Limit values within the specified range	PASSED
	Collapse keyboard on focus loss	PASSED
	Maintain entries on mode switch	PASSED
	Highlight red on incorrect input	PASSED
	Offset cannot exceed amplitude limit	PASSED
	Display input range when text field is empty	PASSED
Graph	Display correctly in portrait & landscape	PASSED
	Display selected function	PASSED
	Display up to the max number of data points	PASSED
	Y axis values update according to inputs	PASSED
	X axis values and title update according to inputs	PASSED
	Freehand mode restricts drawing to inside the graph	PASSED
	Graph is zoomable & scrollable	PASSED
Undo Button	Reset button clears drawing in freehand mode	PASSED

	Undo Button is unavailable if Freehand is not selected	PASSED
Download	File created in device's downloads	PASSED
	Overwrites previous file if already created	PASSED
	Incorrect input prevents export	PASSED
	Toast message for successful and failed export	PASSED
	File contains correct values and number of points	PASSED
Export	Correct HEX data is written to the Microcontroller	PASSED
	Incorrect input prevents export	PASSED
	Toast message for successful and failed export	PASSED
	No drawing exports an empty list in freehand mode	PASSED
Progression Bar	Displays during export or download	PASSED
	Progress accurately reflects the current data being written	PASSED
Bluetooth Button	Displays correct icon based on current Bluetooth state	PASSED
	When pressed attempts to scan if disconnected	PASSED
	When pressed if connected notifies the user it is connected	PASSED

2.4 Subsystem Conclusion

The Android application boasts an intuitive and uncluttered user interface, enabling users to effortlessly generate five distinct wave graphs. It includes four commonly used wave functions and a unique arbitrary wave mode that allows users to create personalized waveforms through freehand drawing. The export process is smooth, with files conveniently saved locally to the user's downloads folder.

Bluetooth functionality is effective, requiring users to grant permissions before the application can take advantage of Bluetooth. Clear indicators notify users when a device is successfully connected to the phone. The application performs reliably and meets user expectations. While the overall functionality is sound, potential enhancements could focus on optimizing export times and cancellation and improvements to how the freehand function creates voltage values.

3. Microcontroller Subsystem

3.1. Subsystem Introduction

The purpose of this document is to discuss the Microcontroller Subsystem for the Arbitrary Waveform Generator (AWG). Simply put, the microcontroller is what connects the Android app interface to the rest of the system and allows communication between the two. This report will cover theory, design, operation, physical implementations, and accomplishments for this subsystem.

3.2. Subsystem Details

A schematic and PCB layout for the subsystem is provided below.

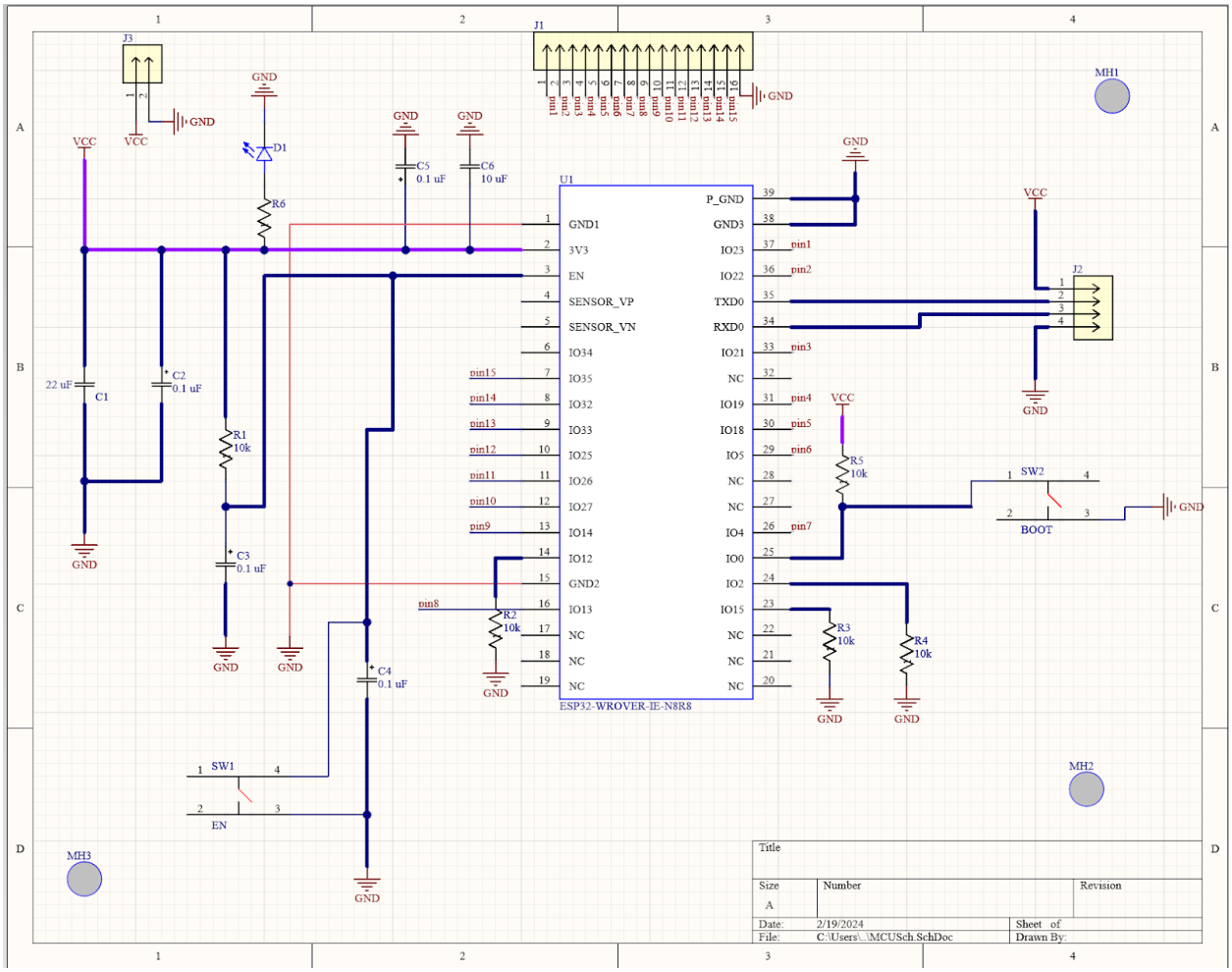


Figure 1012. MCU Schematic

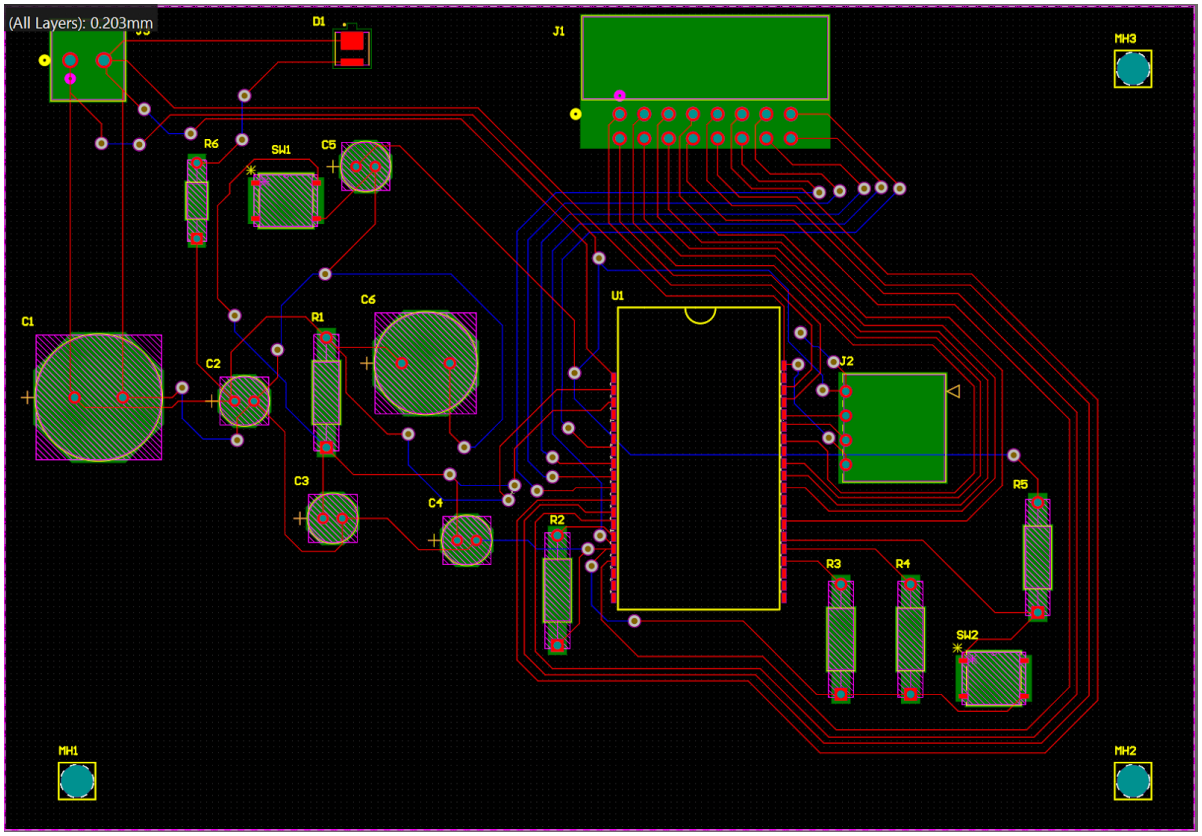


Figure 1113. MCU PCB Layout

An ESP32-WROVER was used as the microcontroller for this subsystem. It has enough I/O pins to allow all the pull-up and pull-down resistors, capacitors, switches, and UART port. There were enough extra I/O pins to accommodate a 16-bit bus that connects to the FPGA. The circuit requires a 3.3 V power supply, with no more than 1 A of current.

One of the key features that sets this microcontroller apart from others is its Bluetooth capability. It was a necessary feature that the AWG system itself be able to have wireless connection to a smartphone. Additionally, an external antenna was attached to the ESP32.

3.3. Subsystem Validation

3.3.1. Power

The MCU first had to be tested to see that it turned on, and that current was flowing everywhere it should be. 3.3 V were supplied to the primary power, along with ground. The green LED lit up as it was supposed to. Pictured in the figure below is the final MCU PCB powered on.

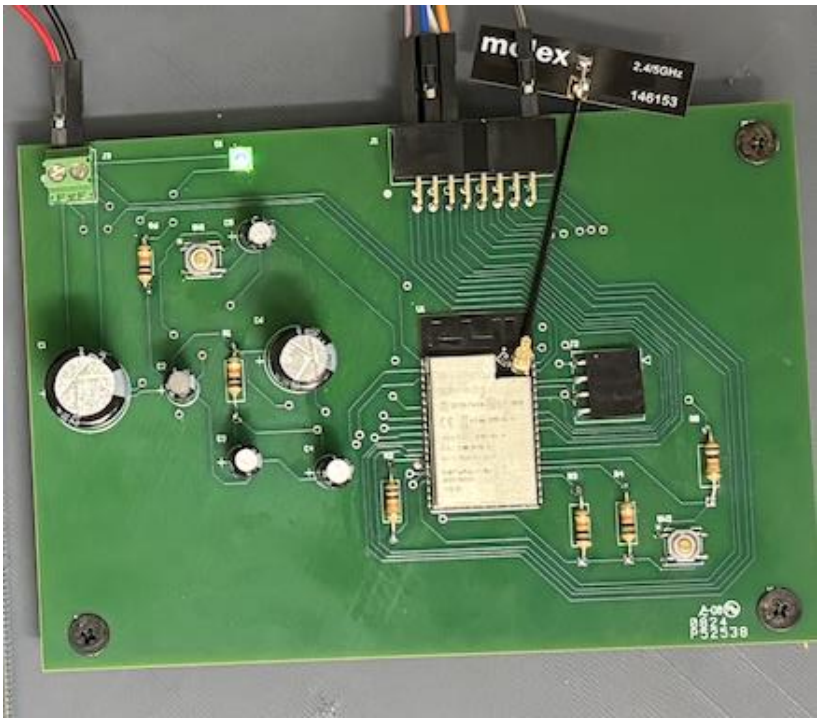


Figure 1214. MCU Circuit Board Powered On

All functions of the board worked as desired. The board successfully powered on, and the Enable and Boot switches successfully allowed the MCU to upload code from the Arduino IDE.

3.3.2. Bluetooth

Bluetooth connection was one of the more difficult characteristics to implement. It was successfully programmed and was capable of importing all 65,535 data points from the Android application.

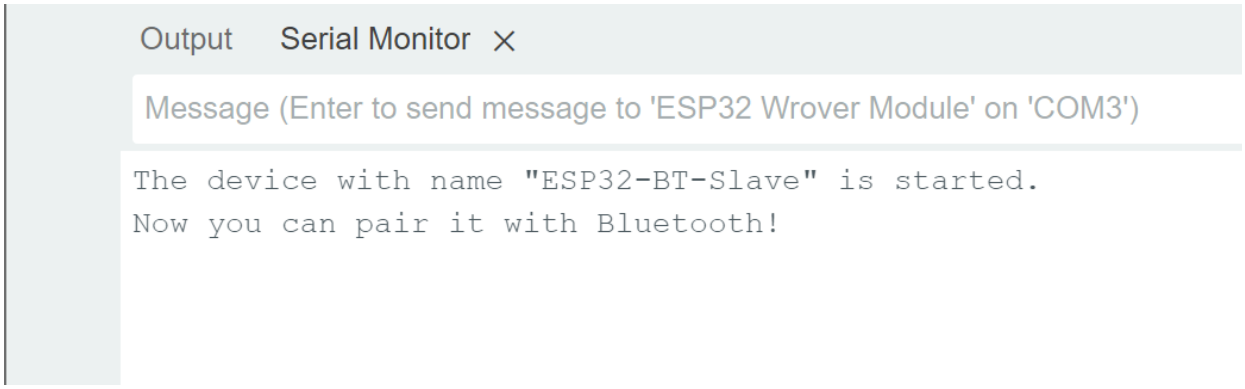


Figure 13. Bluetooth Connection

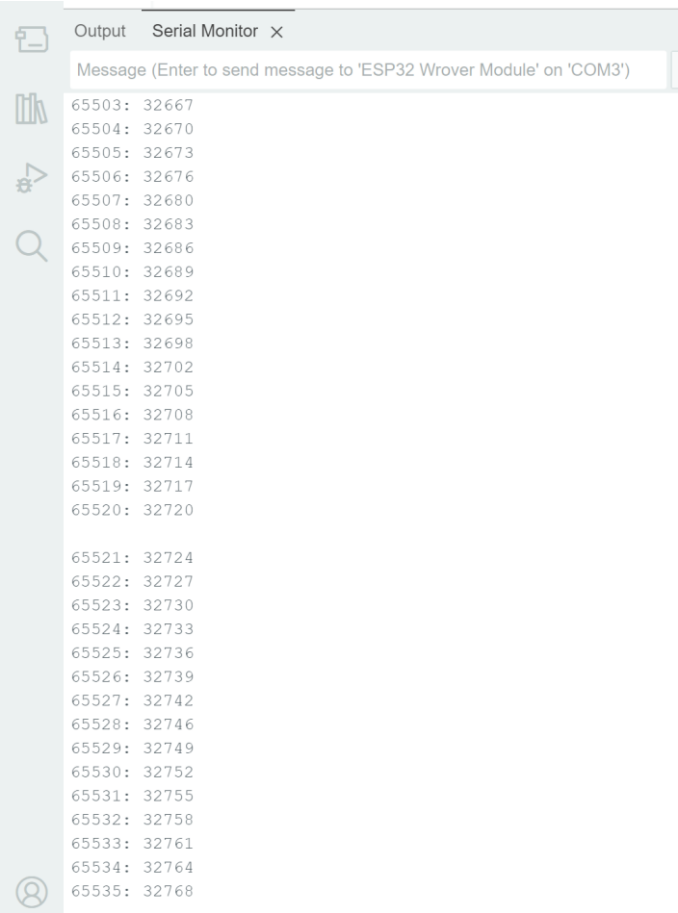


Figure 14. 65,535 Imported Data Points

3.3.3. SPI Communication

Figure 15. SPI Communication

SPI Protocol was utilized to export the data points to the FPGA. The clock, MOSI, and chip select channels successfully exported the data necessary and are shown in the figure above. The blue line is the MISO channel, which would have had verification incoming from the FPGA.

3.4. Subsystem Conclusion

The Microcontroller subsystem successfully implemented both Bluetooth and SPI communication standards to import and export data. Bluetooth Classic was used to connect the MCU, and the Android app using its own unique mac address and deliver up to 65,535 data points. An antenna further increased the range and connection strength between the two subsystems.
SPI protocol then transferred the imported data, providing the hexadecimal values, clock rate, and chip select, as seen in Figure 15.

4. FPGA Subsystem

4.1. Subsystem Introduction

The FPGA subsystem is designed to store information received from the Microcontroller subsystem in the SRAM within the FPGA system. Once stored, this information can be readily accessed and output to the digital-to-analog converter (DAC) system. This report provides an in-depth exploration of the FPGA subsystem's architecture, focusing on its role in information storage and transmission.

4.2. Subsystem Details

A schematic of the FPGA layout is provided below.

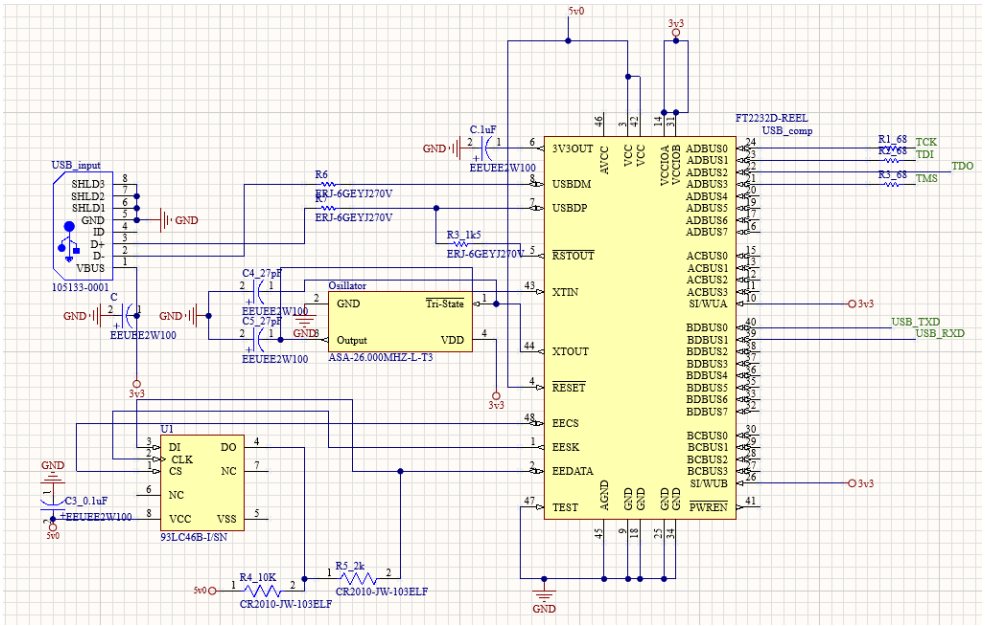


Figure 16. USB input schematic

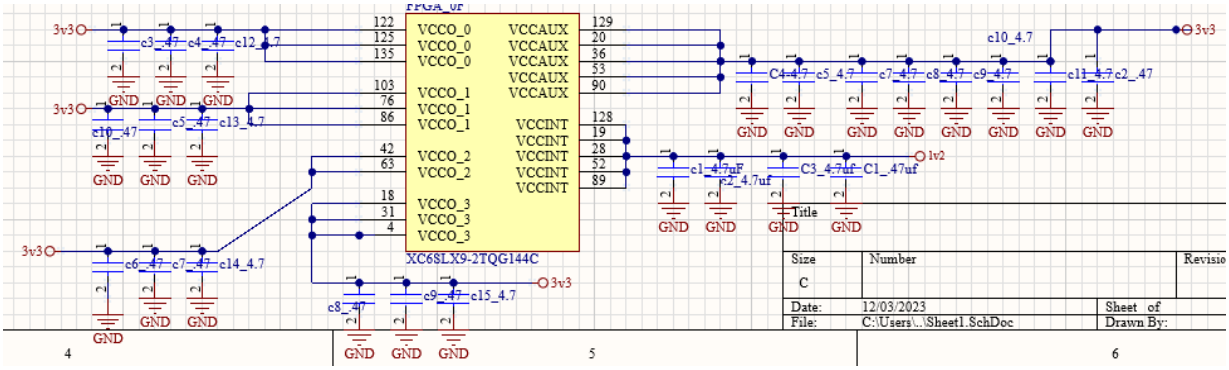


Figure 17. FPGA Power regulation

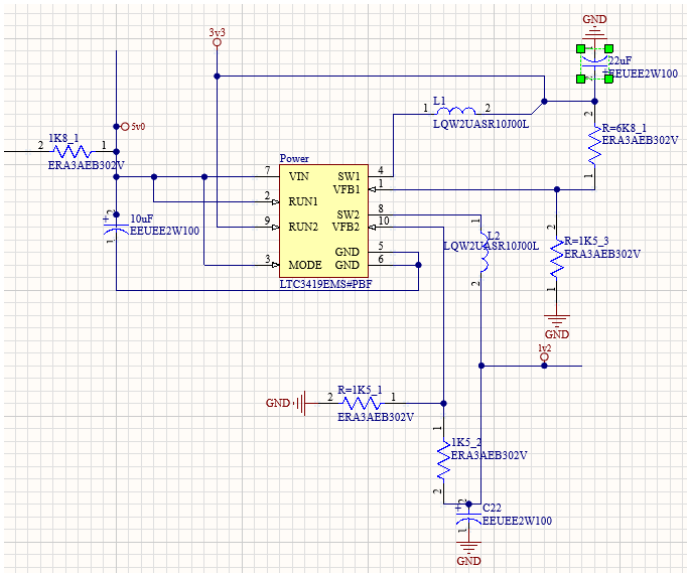


Figure 18. power input schematic

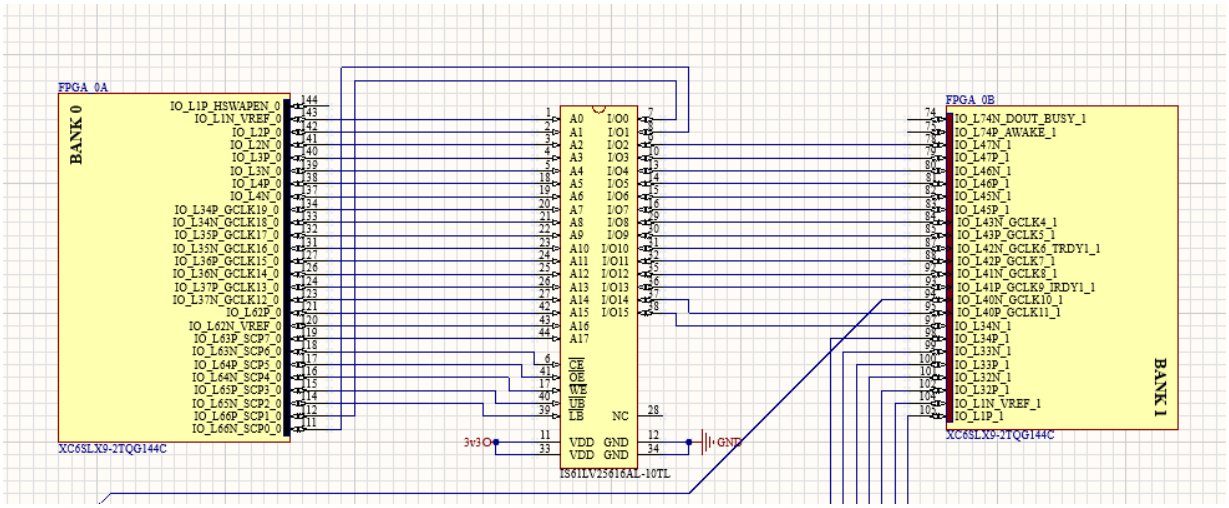


Figure 19. SRAM connection

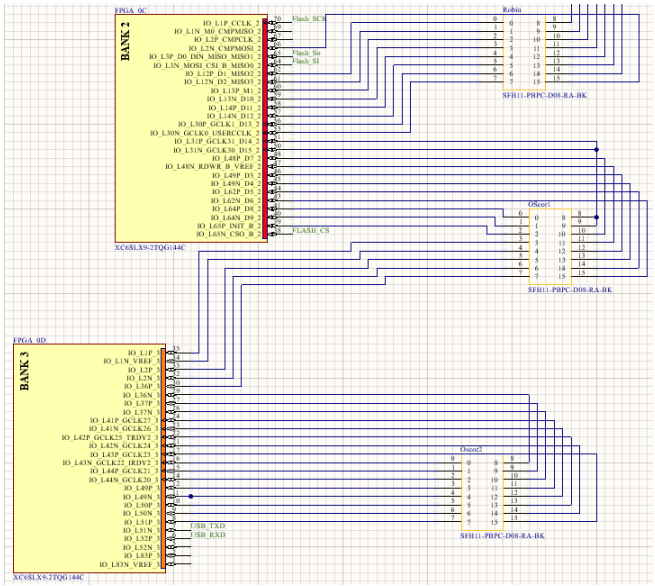


Figure 20. DAC and MCU connection

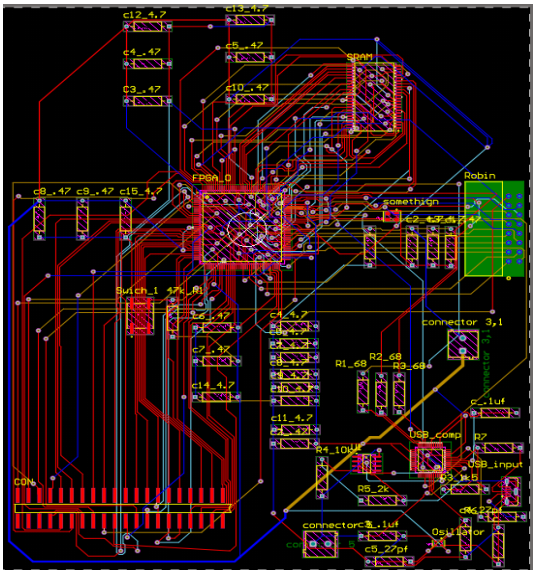


Figure 21. Full FPGA PCB Design

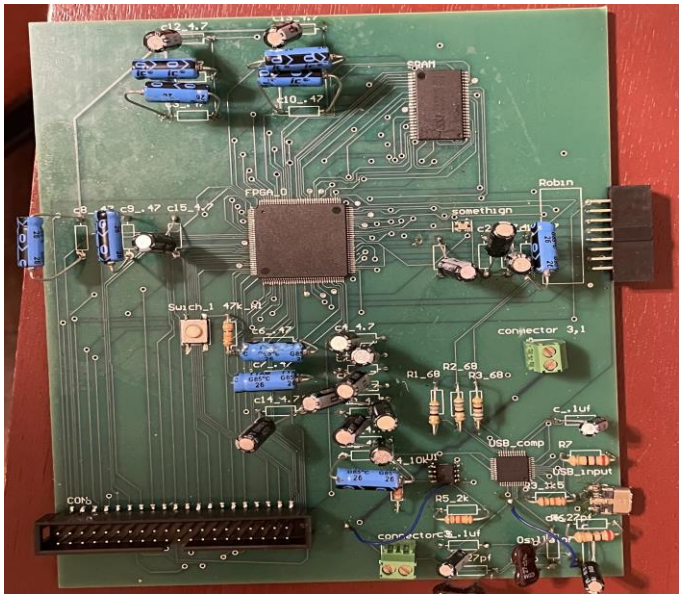


Figure 22. FPGA Circuit board

The Spartan-6 FPGA serves as the conduit for data transfer, utilizing the IO-pins to interface with the SRAM. The SRAM configuration, as outlined in the IS61LV25616AL Data sheet, facilitates toggling between Reading and Writing. Data sourced from the MCU is channeled through this FPGA system. When the data is set for output, the FPGA switches the SRAM to its Reading configuration, retrieves the stored information, and transfers it to the DAC. To power the FPGA, both 3.3 volts and 1.2 volts are necessary, supplied via dedicated power inputs sourced from a 5-volt base input. This setup ensures the requisite voltages for smooth operation of the FPGA subsystem.

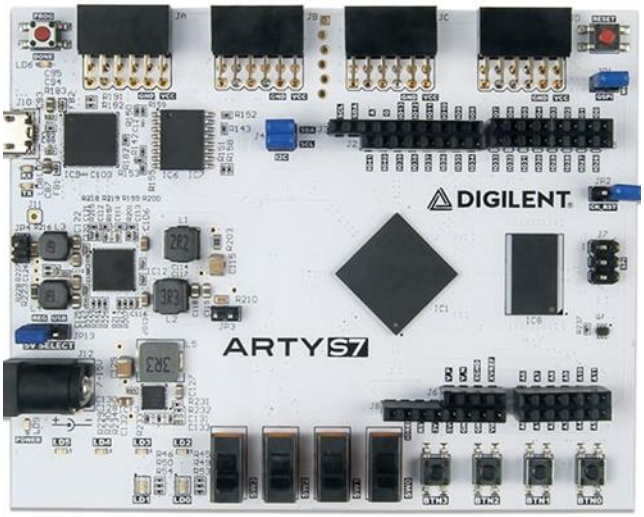


Figure 23. Arty-S7 board

The FPGA programming process involves a micro-USB input that interfaces with a USB controller. This USB controller operates on a 3.3-volt power supply and serves as the conduit for coding the FPGA. This mechanism ensures seamless communication between the programming source and the FPGA, allowing for efficient code transmission and subsequent execution within the FPGA system.

4.3. Subsystem Validation

The system validation report verifies that the FPGA has successfully established communication with the DAC. This successful linkage is crucial as it demonstrates the FPGA's capability to not only store but also efficiently export data into the DAC at the expected frequency and with precise data outputs. This ensures that the system meets the critical performance metrics set for data transfer speed and accuracy, which are essential for real-time data processing and analysis.

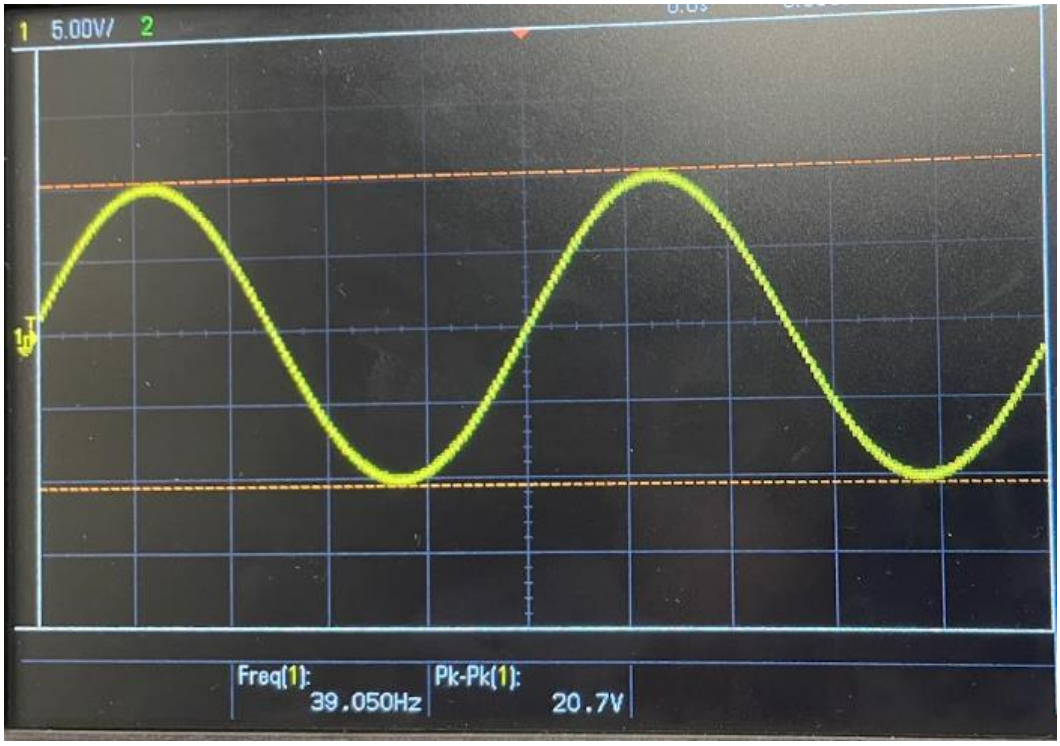


Figure 24. FPGA to DAC validation

The FPGA has successfully established communication with an AD2 to test its capability to receive SPI communication. This successful interfacing confirms the FPGA's proficiency in handling SPI protocol communications. The ability to effectively communicate using SPI protocols is crucial for the FPGA, as it ensures the seamless conversion of analog signals to digital data, which is essential for accurate data processing and analysis within the system. This capability highlights the FPGA's versatility and reliability in managing critical data transmission tasks, further validating its integral role in the system's overall performance and efficiency.

Due to limited time between receiving the FPGA and the Due date we were unable to properly debug the issues between the MCU and the FPGA leading to the inability to communicate between the two systems.

Due to the constrained timeline between the receipt of the FPGA and the project due date, there was insufficient time to thoroughly debug the communication issues between the Microcontroller Unit (MCU) and the FPGA. This shortfall in the debugging process has resulted in a persistent inability to establish communication between these two critical components of the system. The lack of effective communication between the MCU and FPGA is a significant concern, as it impedes the system's capability to perform integrated tasks and manage operations cohesively.

4.4. Subsystem Conclusion

The FPGA subsystem has demonstrated significant capabilities in terms of information storage and transmission, as outlined in this report. The successful establishment of communication with the DAC shows the FPGA's ability to transfer data efficiently and accurately. These achievements affirm the FPGA's role in facilitating real-time data processing and analysis, crucial for the system's performance and operational efficiency.

However, the subsystem faced challenges due to the limited timeframe available for debugging the communication issues between the FPGA and the MCU caused by not having a working FPGA until the beginning of April. This lack of effective communication remains a substantial barrier, preventing the system from achieving fully integrated and cohesive operations. Moving forward, it is imperative to allocate adequate time for thorough testing and debugging to address these issues, ensuring that the FPGA can communicate effectively with all interconnected components and perform its intended functions without disruption.

Overall, while the FPGA subsystem has shown promising results in several areas, the unresolved issues with the MCU communication highlight the need for continued development and optimization to realize the full potential of this technology within the system.

5. Digital-to-Analog Converter Subsystem

5.1. Subsystem Introduction

A key specification for the Arbitrary Waveform Generator is that the voltage range must be between ±10V and have dual channel capabilities. To accomplish this, first, the current outputting digital-to-analog converter used for this project, the DAC8820, must be designed to generate a bipolar output. With a dual op-amp and a fixed voltage reference, the output current of the digital-to-analog converter can be transformed to a voltage and bipolar operation can be achieved. To acquire dual channel capabilities, all that is needed is to design a second instance of the bipolar outputting design is needed, as seen in Figure 25. This subsystem was designed to receive two 16-bit inputs, one for each DAC, and produce an output voltage that ranges from ±10V.

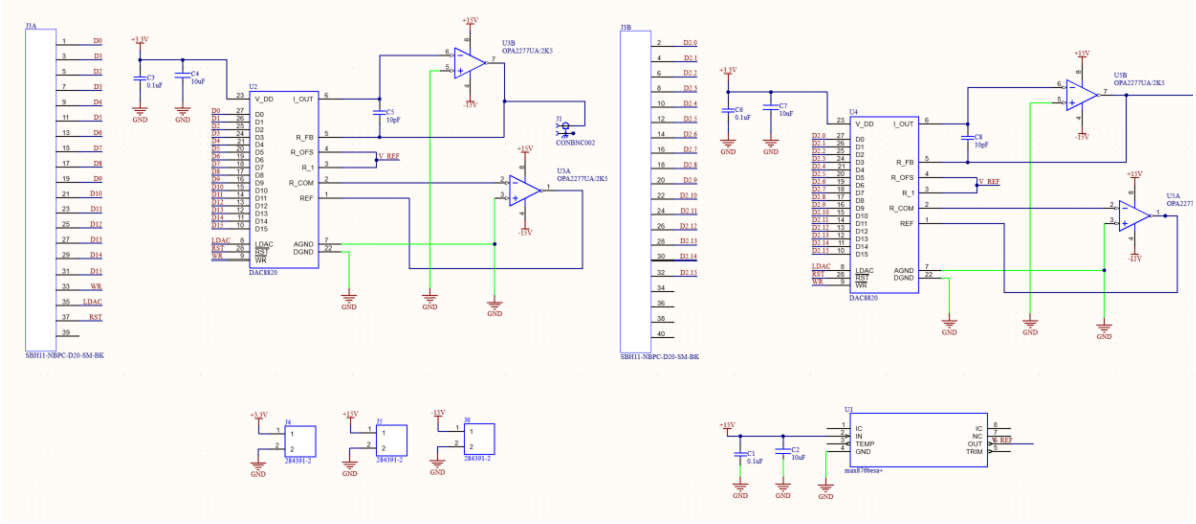


Figure 25. DAC Schematic

5.2. Subsystem Details

The main components for the bipolar output design are the digital-to-analog converter, DAC8820, the dual op-amp, OPA2277UA/2K5, the voltage reference, MAX876BESA+, and the 40-pin connector.

The 40-pin connector serves two purposes, the first being to provide a way to input the required voltage to the three integrated circuits that make up the design. The second is to provide the DAC8820 with the 16-bits of data and the control inputs.

The applied external reference input voltage V_{REF} provided by the MAX876BESA+ in this design will be a fixed +10V per datasheet recommendation. The voltage reference will determine the output current of the DAC8820, with a fixed +10V reference the output current will be 1.66mA.

The dual op-amp OPA2277UA/2K5 serves as a current-to-voltage transformer and allows for full 4-quadrant operations. The reason for the current-to-voltage transformer is because the DAC8820 outputs current and the desired output for the Arbitrary Waveform Generator is voltage. The need for full 4 quadrant operation is so that the output is bipolar and the desired ±10V can be acquired.

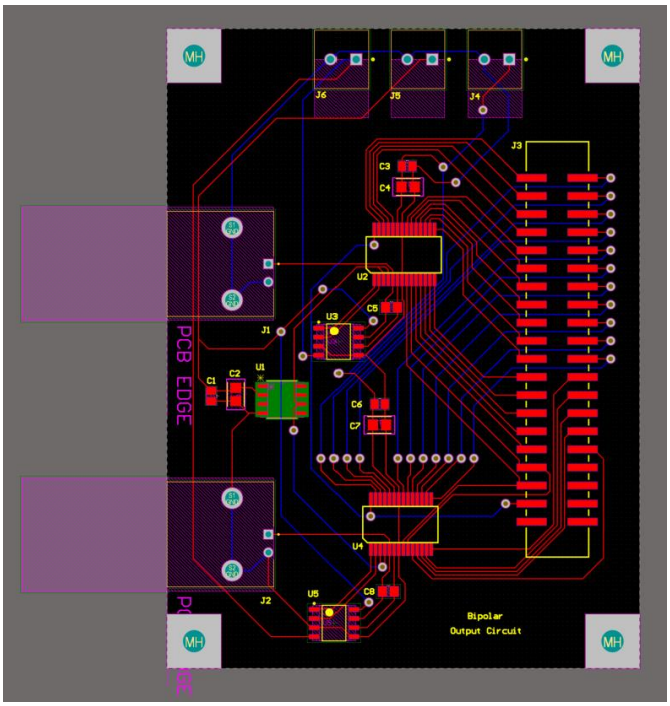


Figure 26. DAC PCB Layout

5.3. Subsystem Validation

The first test that needed to be performed was to test that all integrated circuits were powered on. After applying a +3.3V to the designated pin on the connector, a +3.3V reading must appear at the power supply pin on both DAC8820s. The same process was done with the ±15V on the two OPA2277UA/2K5, and +15V on the single MAX876BESA+. With the MAX876BESA+ powered up, it was also verified that a +10V voltage reference was there for both DAC8820s.

With all the components powered up, the output of the system was tested. Using an AD2 to provide all the necessary pin inputs to both DAC8820s, it was now possible to measure a voltage output based on the 16-bits of data given to each DAC8820 based on this equation found on the datasheet:

$$V_{out} = \left(\frac{D}{32{,}768} - 1\right) \cdot V_{REF},\ where\ 0000h \leq D \leq FFFFh$$

While inputting data ranging from 0000h to FFFFh, it was verified that the DAC8820 was indeed outputting 1.66mA of current and that the bipolar circuitry was functioning, giving the desired ±10V range. Below is the data gathered proving the ±10V specification for both DACs.

DAC 1			DAC 2		
Decimal	Hexadecimal	Voltage [V]	Decimal	Hexadecima	Voltage [V]
65535	FFFF	9.996	65535	FFFF	9.997
60000	EA60	8.308	60000	EA60	8.308
55000	D6D8	6.782	55000	D6D8	6.782
50000	C350	5.256	50000	C350	5.257
45000	AFC8	3.731	45000	AFC8	3.731
40000	9C40	2.205	40000	9C40	2.206
35000	88B8	0.68	35000	88B8	0.68
30000	7530	-0.845	30000	7530	-0.845
25000	61A8	-2.371	25000	61A8	-2.371
20000	4E20	-3.896	20000	4E20	-3.896
15000	3A98	-5.422	15000	3A98	-5.422
10000	2710	-6.947	10000	2710	-6.947
5000	1388	-8.473	5000	1388	-8.473
2500	09C4	-9.236	2500	09C4	-9.236
1250	04E2	-9.617	1250	04E2	-9.617
625	0271	-9.808	625	0271	-9.808
0	0000	-9.998	0	0000	-9.999

Figure 27. Voltage Range Data Table for both DACs

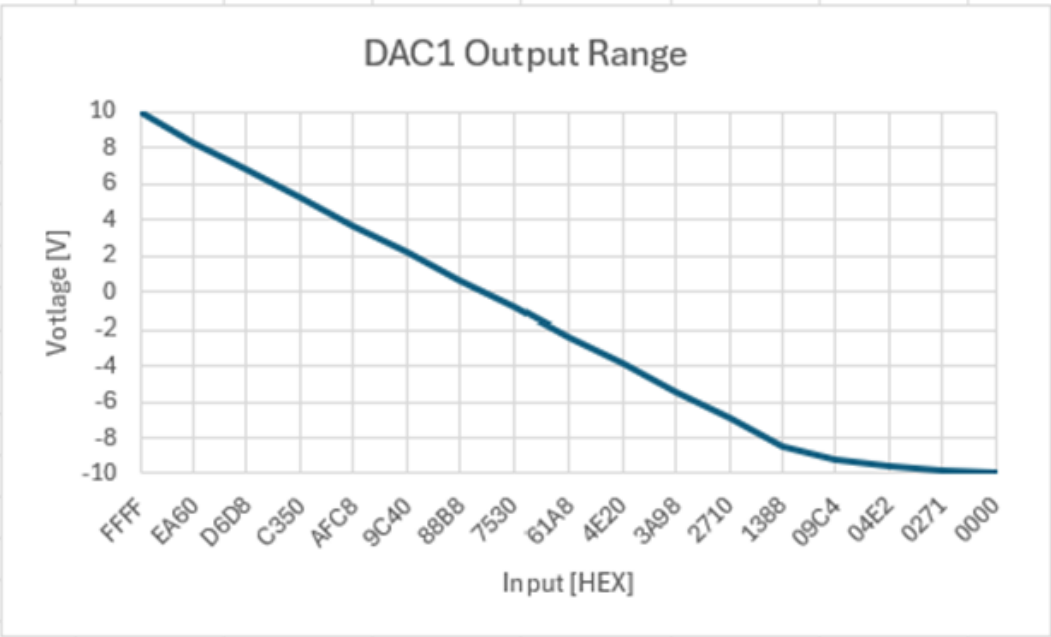


Figure 28. Voltage Range for DAC1

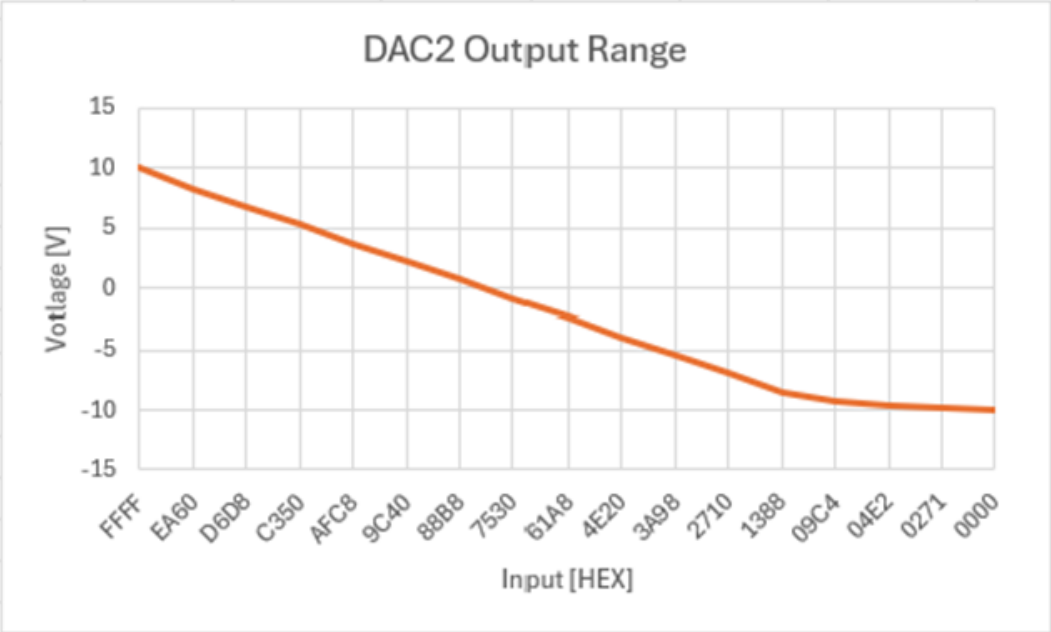


Figure 29. Voltage Range for DAC2

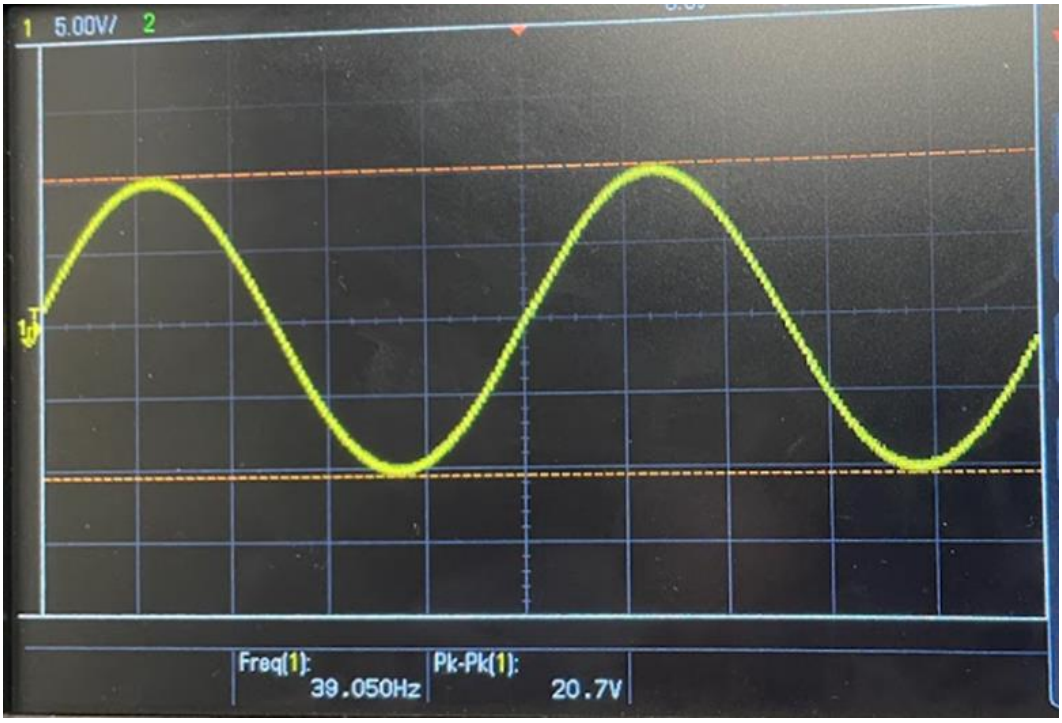


Figure 30. Sine Wave Output from FPGA to DAC Test

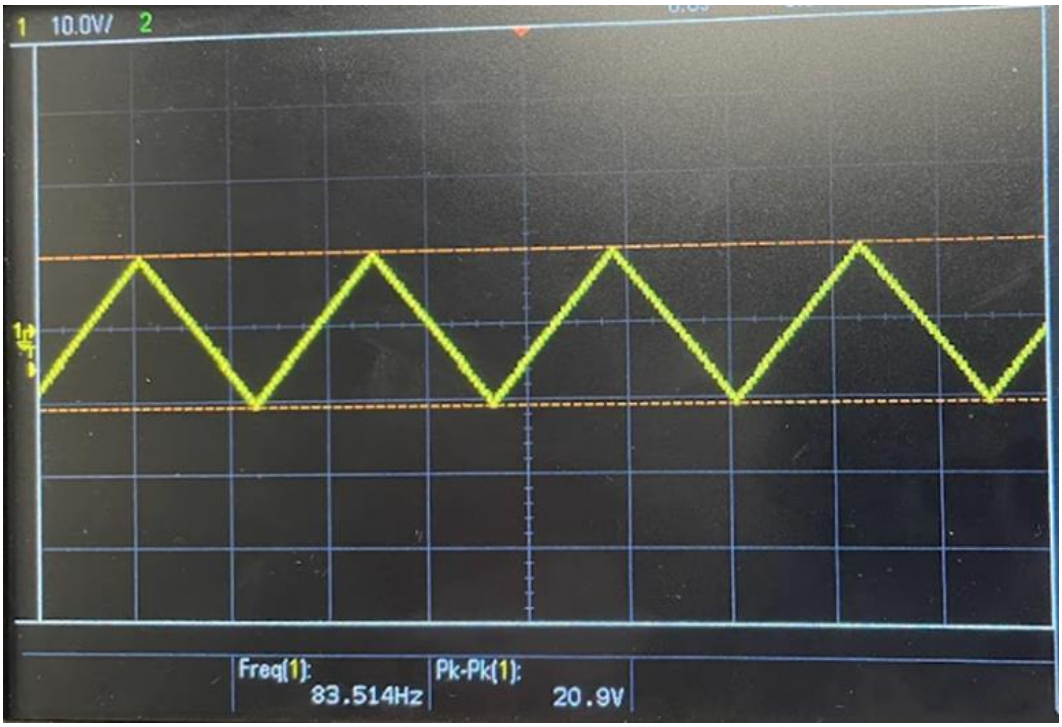


Figure 31. Triangle Wave Output from FPGA to DAC Test

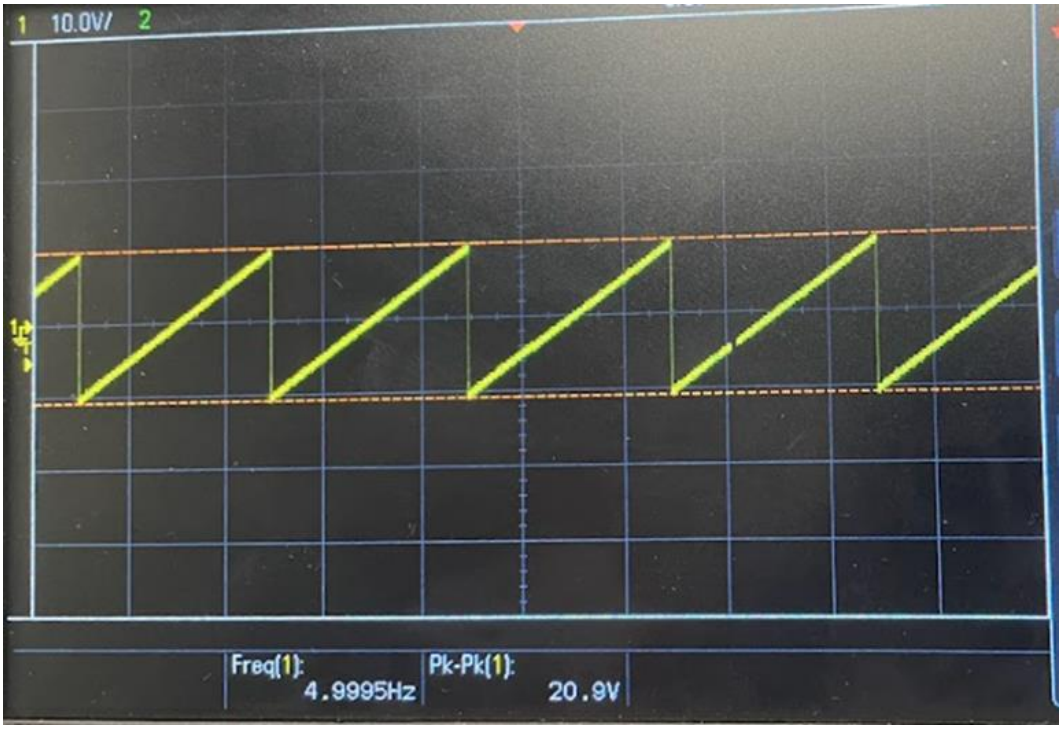


Figure 32. Sawtooth Wave Output from FPGA to DAC Test

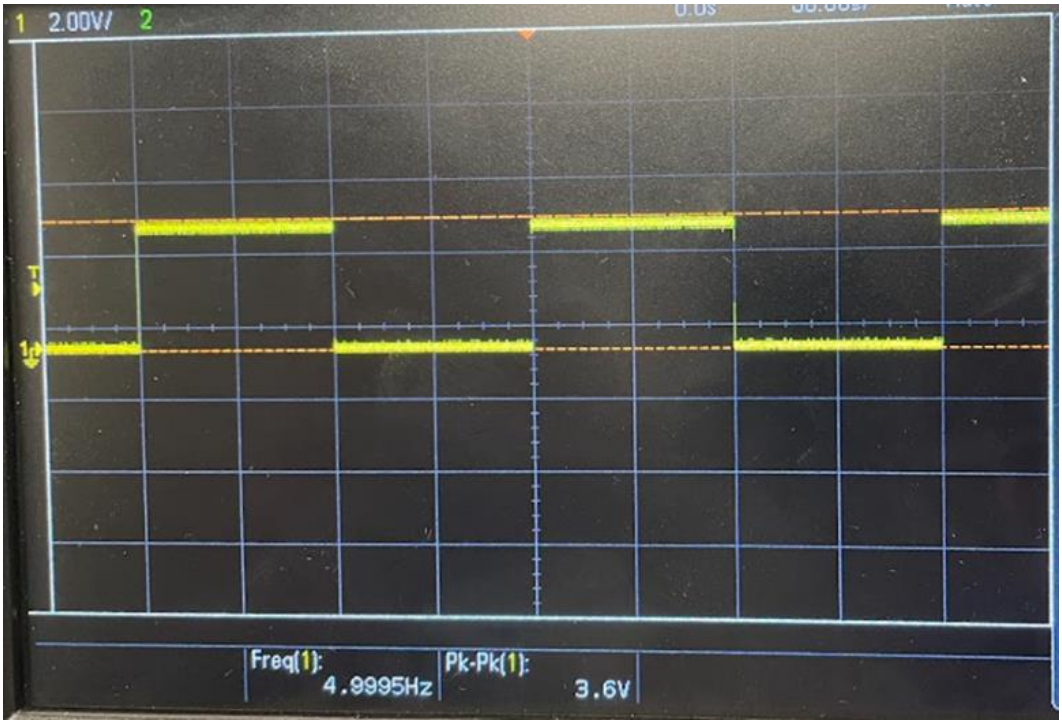


Figure 33. Square Wave Output from FPGA to DAC Test

5.4. Subsystem Conclusion

With all the parts working and desired output acquired, this subsystem is ready to receive data from the FPGA system into both DACs and output desired waveforms. With hardcoded tests from the ARTY-S7, the DAC system is capable of outputting set waveforms given they obey the parameters of the AWG.

6. Power System

6.1. Subsystem Introduction

It is required to design a power system for the arbitrary waveform generator to power each of the subsystems involved. After identifying system requirements, it is determined that the MCU subsystem requires a voltage input of +3.3V, that the DAC subsystem requires the voltage inputs of +3.3V for the DACs, +15V for the voltage reference, and ±15V for the op-amps, and lastly, that the FPGA requires the voltage inputs of +5V, +3.3V, and +1.2V. With the input of +12V from a wall adapter, four DC-DC converters were used to acquire the required voltages for each system.

6.2. Subsystem Details

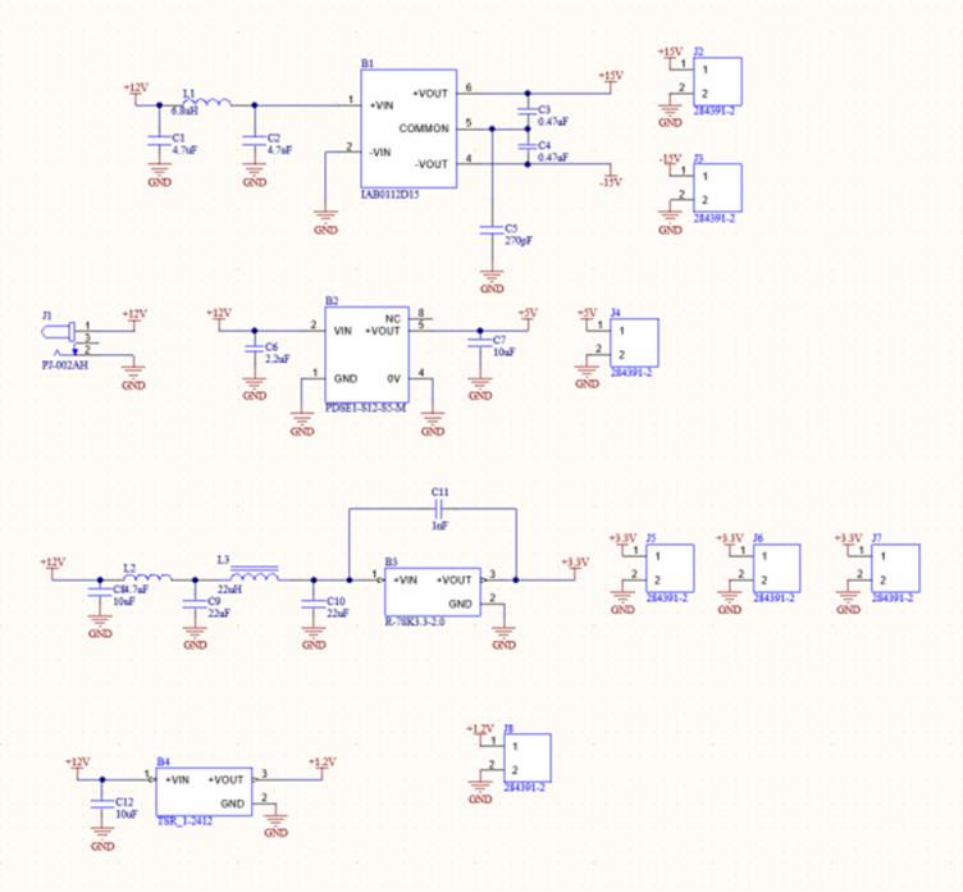


Figure 34. Power System Schematic

The power system is made up of four DC-DC converters, the first used is the IAB0112D15 which outputs ±15V required for the dual op-amps in the DAC subsystem. The next is the PDSE1-S12-S5-M which outputs +5V required for the original FPGA system. The next is the R-78K3.3-2.0 which outputs +3.3V required for the MCU, DAC, and FPGA systems. And lastly, the TSR-1-2412, which outputs +1.2V, is required for the FPGA system. Each of the DC-DC converters can withstand much higher loads than the applications used in the arbitrary waveform generator so there is no need to consider power demand. The design for each of the DC-DC converters is pulled from the datasheets.

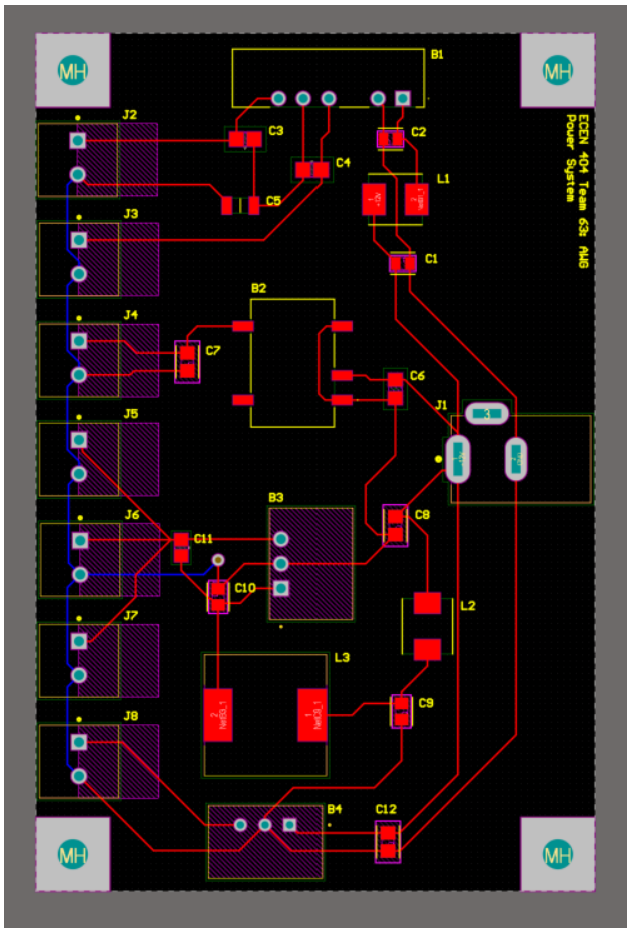


Figure 35. Power System PCB Layout

6.3. Subsystem Validation

To validate the power system, first it is necessary to measure that each DC-DC converter correctly connects to the +12V input voltage. Using the lab multimeter, the input voltage was measured at each converter, similarly, the output voltages were measured at terminal block, ensuring that the correct voltages could be integrated with the other subsystems.

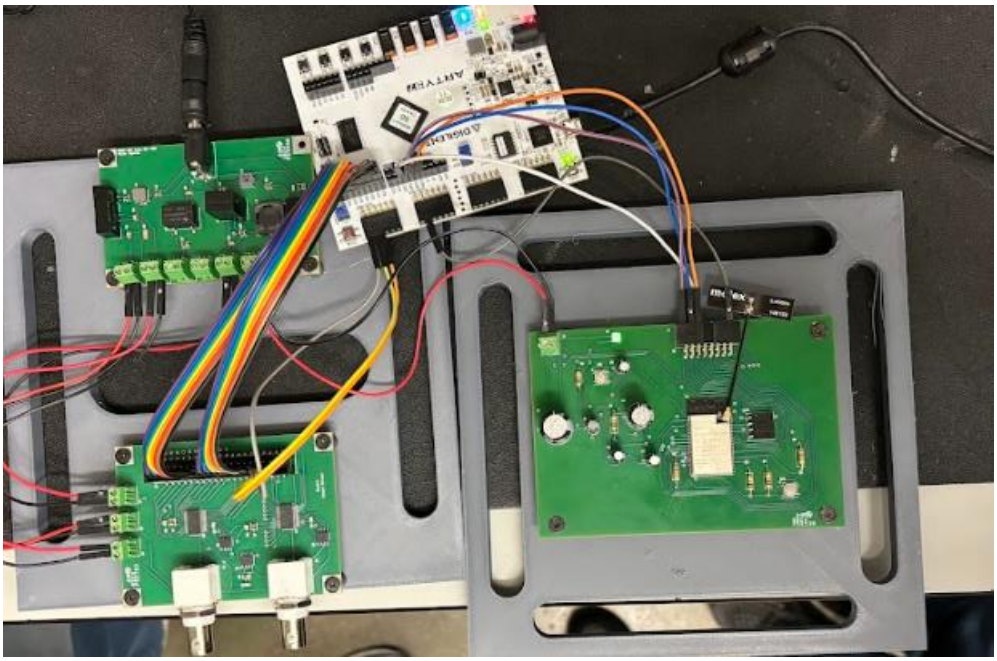


Figure 36. Power System Powering the DAC and MCU Systems

6.4. Subsystem Conclusion

The power system was not designed in the previous semester. Because of this, it was necessary to use a system of DC-DC converters to power the arbitrary waveform generator. Due to the transition to the ARTY-S7, the +5V and +1.2V DC-DC converters went unused. The system successfully powered the MCU and DAC systems, allowing the arbitrary waveform generator to operate.