ECEN 260 - Final Project

Function Generator and Oscilloscope

Brodric Young

Instructor: Brother Allred
April 8, 2025

Contents

List of Tables

List of Figures

1 Lab Overview

This project involves the design and implementation of a low-cost, dual-purpose electronic diagnostic tool: a combined function generator and oscilloscope built using the STM32 Nucleo development board. The primary motivation behind this project stems from the need for affordable and portable lab equipment for students and hobbyists. As a student of electrical and computer engineering, I frequently work on circuit design and debugging, both during and outside of the academic semester. However, high-quality laboratory instruments are often inaccessible or too expensive to own personally, especially during off-track semesters when university facilities are unavailable. This project aims to bridge that gap by offering a simplified yet practical solution that enables essential testing and signal visualization without requiring significant financial investment.

The device integrates waveform generation and signal capture features into a single, compact unit. It supports user-selectable frequency and amplitude settings for output signals and provides real-time display of input waveforms. While it does not match the accuracy or performance of professional-grade equipment, it delivers functionality that is sufficient for small signal, educational, and prototyping purposes. By utilizing key embedded systems concepts including analog-to-digital and digital-to-analog conversion, UART and I2C communication protocols, and external interrupts, the project not only meets practical needs but also reinforces foundational technical skills. In doing so, it provides a valuable learning experience and a useful tool for continued experimentation and development outside of formal lab environments.

1.1 Objectives

- Design and build a combined function generator and oscilloscope using the STM32 Nucleo development board.
- Provide a portable and cost-effective alternative to commercial lab equipment for students and hobbyists.
- Generate adjustable sine wave outputs with user-selectable frequency and amplitude settings.
- Capture and visualize analog signals using an analog-to-digital converter (ADC).
- Transmit waveform data to a computer display via UART for real-time oscilloscope visualization.
- Display selected signal parameters (e.g., peak-to-peak voltage and frequency) on an I2C-connected LCD.
- Implement user controls using buttons and external interrupts to navigate and adjust signal settings.
- Ensure safe operation at low voltages and currents to minimize risk during use.
- Reinforce embedded systems and circuit design concepts through practical application.

2 Design

2.1 Needs Designed to Meet

This project is designed to address a personal and practical need: having access to an affordable yet functional function generator and oscilloscope. As a student, I often find myself needing reliable equipment for circuit design, testing, and debugging, especially during off-track semesters when I am away from campus and no longer have access to university lab resources. Commercially available signal generators and oscilloscopes can be prohibitively expensive for students on a limited budget, so this project aims to provide a cost-effective alternative that offers sufficient performance for most basic electronic development and troubleshooting tasks. Although this tool will not match the precision or features of high-end laboratory instruments, it is intended to be "good enough" for the purposes it serves, namely, enabling experimentation and learning in a personal workspace. The functionality and usability are prioritized over absolute accuracy, making it an ideal solution for students, hobbyists, or anyone needing low-cost electronic diagnostic tools. Ultimately, this project has the potential to benefit a broader audience by lowering the barrier to hands-on electronics work and supporting continued learning outside of traditional lab environments.

2.2 Effect on Various Areas

While the primary focus of this project is to create a low-cost function generator and oscilloscope for personal and educational use, it is important to evaluate its potential impact across a variety of societal and global factors.

- **Public Health:** The project is not expected to have a significant impact on public health. However, care must be taken to ensure that the device does not unintentionally emit electromagnetic interference that could affect sensitive medical or other equipment in its vicinity.
- **Public Safety:** Although the device is not intended to operate at high voltages or currents, there is still a minimal risk of electrical shock if not properly insulated or used with caution.
- **Public Welfare:** The device may have a modest positive impact on public welfare by making basic diagnostic tools more accessible to individuals who cannot afford commercial equipment. While its effect on the general public is limited, it can be particularly beneficial for students, hobbyists, and educators.
- **Global, Cultural, and Social Factors:** The project is not expected to have a direct impact on global, cultural, or social factors. Its purpose is primarily educational and technical, with limited scope outside of personal and academic use.
- **Environmental Factors:** This project has a minimal environmental impact. Its small size and relatively simple construction mean it will not require significant materials or energy consumption. However, the use of environmentally friendly components and proper disposal of electronic waste should still be considered.

- **Economic Factors:** Economically, the project offers a low-cost alternative to expensive lab equipment, making it more accessible to individuals with limited budgets. It has the potential to reduce financial barriers to entry for students and hobbyists pursuing electronics-related interests or education.

2.3 Major Course Concepts

This project incorporates several core concepts from major coursework, including Analog-to-Digital Converters, Digital-to-Analog Converter, UART communication, the I2C protocol, and digital external interrupts. The ADC is used to convert incoming analog signals from the oscilloscope into digital data that can be processed and analyzed. In contrast, the DAC is responsible for generating output waveforms for the function generator by converting digital signals into analog form. Two displays are integrated into the design, one is an LCD module connected via the I2C protocol, which shows key parameters like the selected output frequency and peak-to-peak voltage. The second is a computer interface that serves as a display for the oscilloscope, receiving waveform data over UART communication. Additionally, digital external interrupts are used to capture user input from pushbuttons, allowing the user to cycle through and select specific frequency and amplitude settings, which are then reflected on the LCD. These combined technologies enable both functional waveform generation and signal visualization, while also providing an interactive and responsive user interface.

3 Specifications

This project is designed to create an affordable and functional function generator and oscilloscope using the STM32 Nucleo development board. The system must meet several specific requirements to ensure it provides useful functionality for students, hobbyists, and engineers:

- **Function Generator**: Capable of generating sine waveforms with adjustable frequencies and amplitudes. The frequency range spans from 1 Hz to 1 MHz, with a resolution to adjust the frequency in small increments (a tenth of the current order of magnitude, for example adjusting to 3kHz from 2kHz or to 40kHz from 50kHz) or large increments (an order of magnitude, for example from 10Hz to 100Hz or 100kHz to 10kHz). The amplitude range is adjustable from 0 to 3.3V peak-to-peak with resolution in adjustments of 0.1V for smaller adjustments or 1V for larger adjustments.
- **Oscilloscope**: The system includes an analog-to-digital converter (ADC) for capturing incoming analog signals. The oscilloscope has two separate channels, each capable of visualizing waveform data of a wide range of frequencies and voltages ranging from 0V to 3.3V on a computer monitor via UART communication. The monitor displays voltage on the y-axis for reference.
- **User Interface**: The function generator is controlled via physical pushbuttons that allow users to adjust the frequency and amplitude of the output waveform. Buttons for frequency and amplitude adjustments are separated into the small and large increments described in the function generator bullet point. Additionally, the system includes a reverse button that inverts the direction of adjustment for frequency and amplitude.
- **Communication Protocols**: The system uses the I2C protocol to interface with the LCD display for adjustment of output frequency and voltage. UART communication transmits waveform data to a connected computer for real-time oscilloscope display using a python script to decode and display the waveform.
- **Portability and Power**: The system is designed to be compact and portable, powered via a USB connection, making it easy to transport and use in a variety of environments. It operates at low voltages to ensure safe usage and minimize electrical shock risks.

3.1 Operating Steps

1. **Setup**: Connect the STM32 Nucleo board to the power source via a USB cable. Ensure the device is properly powered and the I2C and UART communication lines are connected to the LCD and computer, respectively. The LCD display will show the default frequency and amplitude values.
2. **Adjust Frequency**: Use the “Frequency Small” and “Frequency Big” buttons to adjust the output frequency. Pressing the small button increases or if the reverse button is also pressed then decreases the frequency in small increments (a tenth of the current order of magnitude), while the big button increases or decreases it by a factor of 10.

3. **Adjust Amplitude**: Use the "Vpp Small" and "Vpp Big" buttons to adjust the amplitude of the sine wave output. The small button adjusts the amplitude in 0.1V increments, while the big button adjusts the amplitude in 1V increments with the reverse button doing the same thing as for the frequency buttons.

4. **Oscilloscope Display**: View the waveform on the connected computer monitor. The waveform data is transmitted over UART for display on the computer by running a python script on the computer. Voltage is displayed on the y-axis for reference.

3.2 Known Limitations

- **Signal Precision**: The system is not designed to match the precision of commercial oscilloscopes and signal generators. Although it provides basic functionality, its accuracy may not be sufficient for high-precision measurements required in advanced lab environments.

- **Frequency Range**: The output frequency range is limited to 1 Hz to 1 MHz, which may not be suitable for applications requiring very high or very low frequencies.

- **Limited Waveform Types**: The system currently only generates sine waveforms. Other waveform types such as square or triangular waves are not yet supported.

- **Resolution and Display**: The resolution of the ADC and DAC may limit the granularity of waveform capture and output. Higher resolution systems may be necessary for more detailed or precise measurements.

- **Power Limitations**: The function generator is limited to lower output voltages (up to 3.3 V peak-to-peak) and may not provide sufficient voltage for certain high-power applications.

- **Oscilloscope Input Pin**: The input pins for both oscilloscope channels are currently unprotected from high or negative voltages and currents. An attempt was made to fabricate a circuit to protect these pins but has so far been unsuccessful. The maximum and minimum recommended voltages and currents are from 0V-3.3V and 0mA-5mA.

3.3 Parts List

- STM32 Nucleo-L476RG Development Board
- USB cable (for powering the STM32 Nucleo)
- 16x2 I2C LCD Display Module
- Pushbuttons (for frequency and amplitude adjustments)
- Computer with UART communication software (for oscilloscope display)
- Breadboard and jumper wires

4 Schematic

The following schematic illustrates the connections between the STM32 Nucleo-L476RG development board, the buttons used for adjusting frequency and amplitude, the connected displays, probes for the oscilloscope, and power supply leads from the function generator. The Nucleo board serves as the central control unit, interfacing with the pushbuttons, LCD display via the I2C protocol, and the external monitor for waveform display via UART communication.

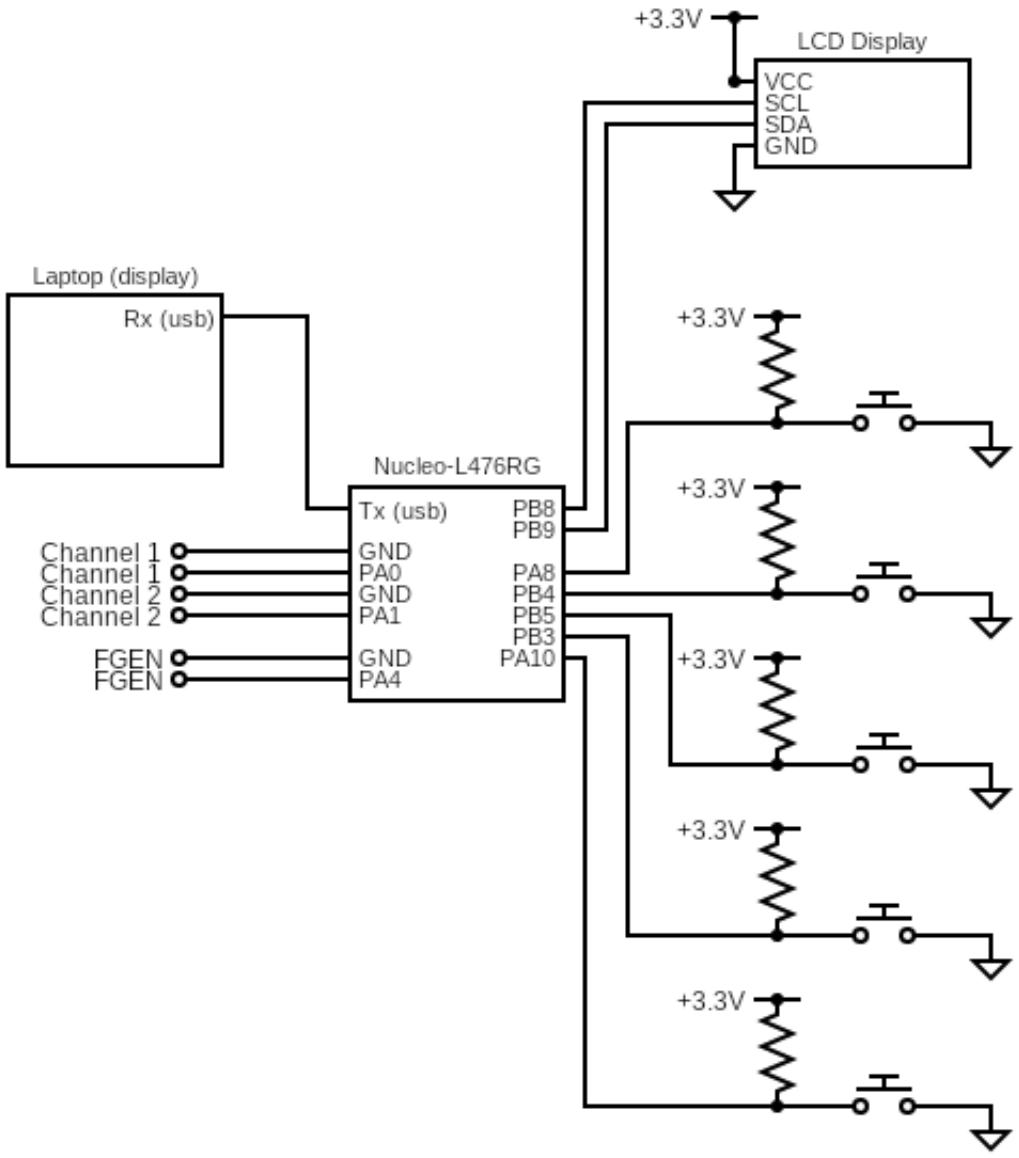

Figure 1: Schematic diagram

5 Test Plan and Test Results

To ensure the functionality and reliability of the system, a structured test plan was developed focusing on common use cases, edge cases, and potential error scenarios. For standard functionality, the system was tested across a wide range of frequency and amplitude settings to verify that the function generator produced expected waveforms and that the oscilloscope correctly displayed them on the UART-connected display. Edge cases included testing the system at the lowest and highest supported frequency values (1 Hz to 1 MHz) and amplitudes (0 to 3.3V peak-to-peak), checking for accuracy and stability in waveform output and capture. Other scenarios included rapid or repeated button presses (debounce testing) and seeing how the frequency and amplitude values displayed on the LCD wrap back around on the last available value button press. These tests helped confirm that the system is user-friendly, fault-tolerant within reason, and suitable for practical use. The demo video of the working project can be seen here: demo-video, and the technical explanation video walking through the technical details of the project can be seen here: explanation-video.

5.1 Test Plan Procedure

Table 1: Test plan with expected and observed results

Case	Test Scenario	Steps	Expected Results	Actual Results
1	Mid-range waveform generation test	Set frequency to 10kHz and amplitude to 1.6V	Sine wave at 10kHz, 1.6Vpp shown on UART display and matching LCD readout	Output matched expectations, waveform stable and accurate
2	Minimum frequency and amplitude test	Set frequency to 1Hz, amplitude to 0V	Flat line or minimal variation on waveform; LCD shows 1Hz, 0V	Flat line observed; LCD correctly displayed values
3	Maximum frequency and amplitude test	Set frequency to 1 MHz, amplitude to 3.3V	High-frequency waveform with max amplitude; LCD shows correct values	flat line at 3.3V; LCD was correct

Case	Test Scenario	Steps	Expected Results	Actual Results
4	Min frequency, max amplitude test	Set frequency to 1Hz, amplitude to 3.3V	Slowly varying waveform with full range amplitude	Waveform displayed as expected with full swing
5	Max frequency, min amplitude test	Set frequency to 1 MHz, amplitude to 0V	Flat line or minimal signal due to 0V amplitude	Showed flat signal; LCD readout correct
6	Button debounce test	Press buttons rapidly in succession	Only one increment per valid press, no skips or double steps	Buttons functioned correctly; no skipped or extra inputs
7	Frequency wrap-around test	Increase frequency past 1 MHz	Frequency resets to 1Hz and displays correctly on LCD	Wrap-around occurred as expected; LCD updated properly
8	Amplitude wrap-around test	Increase amplitude past 3.3V	Amplitude resets to 0V and is displayed on LCD	Functioned as expected; LCD displayed 0V
9	Reverse Button	Change frequency and amplitude with REVERSE active	LCD updates with current frequency and amplitude settings	LCD updated consistently with user changes

6 Code

6.1 Code Overview

The code section provides a detailed breakdown of the main pieces of software used to operate the function generator and oscilloscope. This includes the infinite loop found in the main function (6.2), the functions for updating sine wave values to be passed into the DAC (6.3), and the external interrupt callback responding to button presses (6.4). The full code in the main.c file can be found in my GitHub repository here: main.c. The additional Python script for generating the graph to display waveforms on a computer can be found here: python-wave-viewer.

6.2 Main Function

```
int main(void)
{
  ...

  lcd_init(); // Initialize the LCD
  lcd_backlight(1); // Turn on backlight

  // Write the label text on the LCD
  lcd_write_string("Peak-Peak: "); // first line, ends column 11 (4 columns left)
  lcd_set_cursor(1, 0);
  lcd_write_string("Frequency: "); // second line, ends column 11 (4 columns left)

  // start the timer for DAC and set up initial sine wave output for function generator
  HAL_TIM_Base_Start(&htim2);
  update_sine_wave(Vpp, freq);

  while (1)
  {
    /* LCD display */
    // Vpp value display
    lcd_set_cursor(0, 11);
    Vpp_to_string(Vpp, Vpp_str);
    lcd_write_string(Vpp_str);

    // Frequency value display
    lcd_set_cursor(1, 11);
    freq_to_string(freq, freq_str);
    lcd_write_string(freq_str);


    /* oscilloscope */
    // get channel 1 sample value
    HAL_ADC_Start(&hadc1);
    HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY);
```

```
35      uint16_t channel_1 = HAL_ADC_GetValue(&hadc1);
36
37      // get channel 2 sample value
38      HAL_ADC_Start(&hadc2);
39      HAL_ADC_PollForConversion(&hadc2, HAL_MAX_DELAY);
40      uint16_t channel_2 = HAL_ADC_GetValue(&hadc2);
41
42      char buffer[16];
43      char *ptr = buffer;
44
45      // Convert channel_1
46      char str1[6]; // Max 5 digits + null
47      uint16_to_str(channel_1, str1);
48      char *s1 = str1;  // pointer to traverse the array
49      while (*s1) *ptr++ = *s1++;
50
51      *ptr++ = ',';
52
53      // Convert channel_2
54      char str2[6];
55      uint16_to_str(channel_2, str2);
56      char *s2 = str2;  // pointer to traverse the second string
57      while (*s2) *ptr++ = *s2++;
58
59      *ptr++ = '\n';
60      *ptr = '\0';
61
62      // send information to laptop running python script to display waveforms
63      HAL_UART_Transmit(&huart2, (uint8_t*)buffer, (uint16_t)(ptr - buffer),
    HAL_MAX_DELAY);
64    }
65  }
```

6.3 Sine Wave Updates

```
1  // scales the 3.3V (33) sine wave to another given pk-pk value
2  void generate_sine_wave(uint32_t new_Vpp)
3  {
4    for (int i = 0; i < 128; i++) {
5      new_sine_wave[i] = (uint32_t)((sine33[i] * new_Vpp) / 33);
6    }
7  }
8
9  // modifies the Auto-Reload Register in order to adjust wave frequency
10 void set_trigger_frequency(uint32_t wave_frequency) {
11   uint32_t trigger_frequency = wave_frequency * SAMPLES;
12   uint32_t arr = (80000000 / trigger_frequency) - 1; // internal clk = 80MHz
13
14     __HAL_TIM_SET_AUTORELOAD(&htim2, arr);
15     HAL_TIM_GenerateEvent(&htim2, TIM_EVENTSOURCE_UPDATE);
16     HAL_TIM_Base_Start(&htim2);  // Restart timer
17 }
```

```
18
19 // Stops DMA, then calls the other functions to adjust amplitudew and
       frequency, then starts DMA again
20 void update_sine_wave(uint32_t new_Vpp, uint32_t new_freq)
21 {
22     HAL_DAC_Stop_DMA(&hdac1, DAC_CHANNEL_1); // Stop DAC DMA
23     generate_sine_wave(new_Vpp);            // Update sine wave values
24     set_trigger_frequency(new_freq);          // Update frequency
25     HAL_DAC_Start_DMA(&hdac1, DAC_CHANNEL_1, new_sine_wave, SAMPLES,
       DAC_ALIGN_12B_R); // Restart DAC DMA
26     HAL_TIM_Base_Start_IT(&htim2);
27 }
```

6.4 Button Press Interrupts

```
1  // called whenever a button is pushed (used for amplitude and frequency
       adjustment)
2  void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
3  {
4      uint32_t current_time = HAL_GetTick(); // gets current time for button
       debouncing purposes
5
6      if (GPIO_Pin == VPP_SMALL_Pin) {
7          // Debounce check for VPP_SMALL button
8          if (current_time - last_Vpp_small_press > debounce_delay) {
9              last_Vpp_small_press = current_time;
10             // Check REVERSE pin state to determine direction of change
11             if (HAL_GPIO_ReadPin(REVERSE_GPIO_Port, REVERSE_Pin) == 0) {
12                 // Decrease Vpp by 1 if not already at 0; else wrap around to
       33
13                 if (Vpp > 0) Vpp -= 1;
14                 else Vpp = 33;
15             } else {
16                 // Increase Vpp by 1 if below 33; else wrap around to 0
17                 if (Vpp < 33) Vpp += 1;
18                 else Vpp = 0;
19             }
20         }
21     }
22
23     else if (GPIO_Pin == VPP_BIG_Pin) {
24         // Debounce check for VPP_BIG button
25         if (current_time - last_Vpp_big_press > debounce_delay) {
26             last_Vpp_big_press = current_time;
27             // Check REVERSE pin state to determine direction of change
28             if (HAL_GPIO_ReadPin(REVERSE_GPIO_Port, REVERSE_Pin) == 0) {
29                 // Decrease Vpp by 10, or wrap around to 33 if below 10
30                 if (Vpp >= 10) Vpp -= 10;
31                 else Vpp = 33;
32             } else {
33                 // Increase Vpp by 10 if result will be <= 33; else wrap to 0
34                 if (Vpp <= 23) Vpp += 10;
```

```
35                  else Vpp = 0;
36              }
37          }
38      }
39
40      else if (GPIO_Pin == FREQ_SMALL_Pin) {
41          // Debounce check for FREQ_SMALL button
42          if (current_time - last_freq_small_press > debounce_delay) {
43              last_freq_small_press = current_time;
44              // REVERSE pressed: decrease frequency based on range
45              if (HAL_GPIO_ReadPin(REVERSE_GPIO_Port, REVERSE_Pin) == 0) {
46                  if      (freq < 1)        freq = 1000000;      // Wrap to max
47                  else if (freq <= 10)      freq -= 1;
48                  else if (freq <= 100)     freq -= 10;
49                  else if (freq <= 1000)    freq -= 100;
50                  else if (freq <= 10000)   freq -= 1000;
51                  else if (freq <= 100000)  freq -= 10000;
52              } else {
53                  // REVERSE not pressed: increase frequency based on range
54                  if      (freq > 1000000)  freq = 1;            // Wrap to min
55                  else if (freq >= 100000)  freq += 100000;
56                  else if (freq >= 10000)   freq += 10000;
57                  else if (freq >= 1000)    freq += 1000;
58                  else if (freq >= 100)     freq += 100;
59                  else if (freq >= 10)      freq += 10;
60                  else if (freq >= 1)       freq += 1;
61              }
62          }
63      }
64
65      else if (GPIO_Pin == FREQ_BIG_Pin) {
66          // Debounce check for FREQ_BIG button
67          if (current_time - last_freq_big_press > debounce_delay) {
68              last_freq_big_press = current_time;
69              // REVERSE pressed: divide frequency by 10 or wrap to 100000 if at
    min
70              if (HAL_GPIO_ReadPin(REVERSE_GPIO_Port, REVERSE_Pin) == 0) {
71                  if (freq > 1) freq /= 10;
72                  else freq = 100000;
73              } else {
74                  // REVERSE not pressed: multiply frequency by 10 or wrap to 1
   if at max
75                  if (freq < 1000000) freq *= 10;
76                  else freq = 1;
77              }
78          }
79      }
80
81      // Update the sine wave output with the current amplitude (Vpp) and
   frequency
82      update_sine_wave(Vpp, freq);
83  }
```

7 Conclusion

This project successfully accomplished the goal of designing and building a low-cost function generator and oscilloscope using the STM32 Nucleo development board. Through careful integration of key components such as the ADC, DAC, UART, I2C, and digital interrupts, the system can generate adjustable output waveforms and visualize analog input signals on a computer display. The implementation of user-friendly input controls and real-time display feedback made the device interactive and functional for basic circuit testing and debugging. The final result demonstrates that it is possible to create an effective and affordable piece of lab equipment for personal or educational use, especially for students working on electronics projects outside of traditional lab environments. I had attempted to build a protection clipper and clamper circuit using zener diodes, resistors, and a capacitor in order to prevent high or negative voltages/currents from entering and damaging the microcontroller but my circuit did not function as expected from my simulations. The fabricated circuit and simulations for the attempted circuit can be found here. Future additions to the project would include a similar but working circuit for inputs and outputs to support a wider range, including negative voltages.

Throughout the design and development process, this project reinforced many of the core concepts learned in embedded systems and digital hardware courses. Working with UART and I2C provided additional experience with common communication protocols, while implementing ADC and DAC functionality deepened understanding of signal conversion, timing, sinusoidal functions, and properties of function generators and oscilloscopes. Using digital interrupts for user input tied back to earlier labs as well. This project not only consolidated classroom knowledge learned through this semester, but also provided valuable experience in system integration and debugging—skills that will be essential in future academic and professional engineering work.