# ELEC 334 - Project #1

## Objective

The objective of this project is to create **a function generator** which can produce *sine*, *square*, *triangular*, *sawtooth* (*ramp*), *white noise* and *random digital stream* outputs. The output pin will be chosen by you, and will be looked at using an *oscilloscope*.

Additionally, by completing this project, you will

- implement an event driven approach by using multiple interrupts
- understand and experiment interrupt priorities
- generate analog signals using modulation
- implement filter circuits and use laboratory tools to observe signals

## Setup

Install STM32CubeIDE and implement problems on the breadboard. For that, use the **`blinky`** project from the g0 project repo (`https://github.com/fcayci/stm32g0`) and follow `https://youtu.be/UYk_NAnDJlw`. Do not use **standalone** for any problem / lab. That is just for demonstration purposes. You can use one of the undergraduate laboratories for using the scope and measuring the necessary voltages if needed.

- You are required to create a flowchart that shows software operation and a block/connection diagram that shows how things are connected and what modules are in use. You can use online tools to do that. One example is lucidchart (`https://lucid.app`)
- **All codes should be written in C.**
- If a specific pin number is not stated, you are free to pick any appropriate pin to connect your components. Make sure to include this in your block/connection diagram.

## Submission

You should submit the following items organized in a folder:

- **Lab report** - Written in English. PDF file. Should include
    - a cover page
    - a flow chart
    - a block/connection diagram
    - pictures/photos if any/requested
    - code listing
    - at least one paragraph explanation/comment about that problem, code listing
    - final project conclusion about what you've learned.
- **Source files** - should have proper comments and pin connections if there are any external components.
- **Elf files** - generated elf file with debug information.

# Requirements

- Button `#` will be used as an Enter key
- Button * will be used as a dot key
- Button A will be used to define the **Amplitude** of the signal. It will expect a series of numbers after it, and upon pressing the Enter key, it will apply the amplitude. E.g. `A1*20#` will set the amplitude to `1.20` Volts Peak-Peak.
- Button `B` will be used to define the **Frequency** of the signal. It will expect a series of numbers after it, and upon pressing the Enter key, it will apply the frequency. E.g. `B120#` will set the frequency to `120` Hz.
- There should be proper guards for voltage/amplitude inputs and other buttons (i.e. Pressing `B` after `A` should cancel the operation)
    - If no button is pressed for `10` seconds, timeout should happen, and input should be cancelled.
- Button `C` will cycle through modes.
- Button `D` will cycle through displaying the Mode, Amplitude and Frequency of the currently active signal.
- *Sine, square, triangle, sawtooth,* and *white noise* modes should accept both amplitude and frequency information.
- *Digital stream output* mode should randomly generate 0 and 1s at the given frequency. The amplitude should not affect the amplitude of this operation. **This should use a different output pin than the one used for other modes.**
- All operations should be shown on the SSDs.
    - By default, the SSD should display the mode. Sine, square, triangle, sawtooth (ramp), white noise and random digital stream modes. (try to spell each uniquely. ie. for sine, you can write `SInE`, for triangle, you can write `trI`, etc.)
    - Pressing button `A` should empty the SSD and each digit press should push the digit from left and shift the others to the left. . (dot) should be displayed as one digit character of your choosing)
    - Pressing Enter should clear the SSD and should go back to displaying the mode.
- There should be no bouncing on the buttons.
- There should be no brightness difference between Seven Segments.
- There should be no flickering on the SSDs.
- You should calculate and figure out the maximum possible frequencies that you can run.
- Output signals should be clean, so design proper filters based on the frequency range that you support.