***Efficient Embedded Course***

**ASSIGNMENT**

**INTERRUPT PROJECT:**

**HUMAN RESPONSE TIMER**

**Issue 1.0**

Contents

[1 Introduction 1](#_Toc78805198)

[1.1 Lab overview 1](#_Toc78805199)

[2 Requirements 1](#_Toc78805200)

[3 Details 2](#_Toc78805201)

[3.1 Hardware 2](#_Toc78805202)

[3.2 Software 2](#_Toc78805203)

[3.3 Testing 3](#_Toc78805204)

[3.4 Extra Credit Options 3](#_Toc78805205)

# Introduction

## Lab overview

For this project, you will create a device which makes sounds through a speaker based on how the user presses several switches.

# Requirements

In this lab, we will be using the following hardware and software:

* **Keil µVision5 MDK IDE**
  + Please see the included Getting Started with Keil guide on how to download and install Keil.
* **STM32 Nucleo-L552ZE-Q**
  + For more information, click [here](https://www.st.com/en/evaluation-tools/nucleo-l552ze-q.html).
* **Logic Analyzer or Oscilloscope** 
  + Required to monitor the interrupt signals
* **3x Switches/Buttons**
* **Speaker module**
* **330 Ω Resistor**
* **1 µF Capacitor**

# Details

## Hardware



Figure 1. Schematic diagram

Use three momentary switches SW1-SW3 (or one multi-way switch) to control the device. Drive a speaker SP1 from a GPIO output (labelled Audio) using capacitor C1 to block DC current. Resistor R1 is optional and reduces the volume of the sound.

Please see the included Nucleo-L552ZE-Q pins legend (NUCLEO\_L552ZE\_pins.docx) for the pinout of the Arduino-included Zio connectors for CN7, CN8, CN9 and CN10.

## Software

Write code in C to do the following:

* The tone should sound while SW2 is pressed down.
* The frequency of the tone should rise while SW1 is pressed.
* The frequency of the tone should fall while SW3 is pressed.

The following software design is suggested:

* Create an initialization function which configures GPIO inputs and outputs based on which pins to which you’ve wired your switches and speaker.
* Use the delay-loop function delay\_us which has been provided in the demo project to cycle at the correct frequency.
* Create a function play\_tone(unsigned int duration\_ms) which generates a square wave with the given period (specified in microseconds) and duration (milliseconds). This can be done by toggling the audio output pin, waiting for a time delay, and repeating this process. Calculate the necessary value to pass to delay\_us based on period (inverse of frequency). Calculate the number of times to toggle the output based on period and duration.
* Create a function update\_buttons(void) to repeatedly check to see if any switches are pressed and respond accordingly.
  + If SW1 or SW3 is pressed, adjust the period accordingly. Limit the value of period to within 100 microseconds and 10,000 microseconds.
  + The processor executes this loop very quickly if SW2 is not pressed, so the value of period will quickly reach the upper or lower limit if SW1 or SW3 is pressed. To slow the code down, remember if the previous state of the buttons and only respond if the button state has changed.
* If SW2 is pressed, call play\_tone(unsigned int duration\_ms) with a duration of your choice.