***Efficient Embedded Course***

**ASSIGNMENT**

**TIMER PROJECT:**

**CLOCK**

**Issue 1.0**

Contents

[1 Introduction 1](#_Toc78890806)

[1.1 Lab overview 1](#_Toc78890807)

[2 Requirements 1](#_Toc78890808)

[3 Details 2](#_Toc78890809)

[3.1 Hardware 2](#_Toc78890810)

[3.2 Software 2](#_Toc78890811)

# Introduction

## Lab overview

For this project, you will create a clock which displays elapsed time and flashes an LED each second.

# 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
* **LCD module**
* **LEDs**

# Details

## Hardware

For this project you will reuse the text LCD module and LED hardware and driver software which you developed in the GPIO Basic User Interface module.

## Software

Update LCD Flag

Interrupt Timer

PWM

GPIO -> LCD

GPIO -> LED

Update Time

timer\_isr

Main: Update LCD

Figure 1. Communication diagram for system.

Develop software according to the specifications of Figure 1.

The ISR sends information to the main code with the global variable:

* lcd\_update\_requested: main should update LCD

The main code should perform the following:

* Initialize all peripherals and enable interrupts
* Repeat
  + If lcd\_update\_requested is set:
    - Read the current time in a thread safe manner: hour, minute, second, millisecond.
    - Update the LCD with the format shown below.(h:hours, m:minutes, s:seconds, M: milliseconds). Use sprintf with a format specifier %02d to print a decimal value so it is zero-padded to two digits.

|  |
| --- |
| hh:mm |
| ss.MMM |

The ISR should execute every 1 millisecond and perform the following:

* Update the current time information (hour, minute, second, millisecond)
* Request an LCD update every 100 ms
* Flash the LED once per second. Adjust the LED brightness using Pulse Width Modulation. LED brightness should start at 100% at 0ms and fade to 0% by 600ms.