ECSE 426 - Microprocessor Systems
Lab Report 2: Timers, Interrupts, Multithreaded, Interrupt-Driven Readings and Peripheral Control

Harley Wiltzer (260690006)
Matthew Lesko (260692352)

March 19, 2018

Contents

List of Figures

List of Tables

1 Abstract

The purpose of experiment 3 is for the programmers to gain experience in utilizing timers and interrupts to accomplish a task which involves converting an analog pulse to digital and displaying its voltage on an LED display, effectively a voltmeter. The purpose of experiment 4 is for the programmers to gain exposure in designing a multithreaded application on a real time operating system (RTOS) running on an embedded system. The task of experiment 4 involves copying over experiment 3's program and subdiving several of its features each to its own concurrently running thread, with the goal of optimizing power usage. This report will explain in detail how the programmers implemented the problems stated below, as well as the challenges they faced, the testing they had done, and the conclusions they have made. By the end of the report, the reader shall understand how the timers available on the STM32F4 board can be used to activate peripherals and generate a pulse, and understand the implementation of multithreaded programs on embedded systems.

2 Problem Statement

The problem is for the developers to first implement a solution for generating a PWM pulse, whose voltage is set by an input on a keypad, that is then fed to a rectifier. Afterwards, the device shall feed the rectifier's output to an ADC to be converted to a digital signal, and have the signal's mean voltage be automatically displayed on an LED display. Furthermore, the program has to be implemented with the use of concurrently-running threads running on an RTOS. The problem can be divided into the following tasks:

- Setting up a timer to act as a PWM pulse generator;
- Setting up a timer to activate the ADC to take an analog sample and convert it to digital;
- Designing a rectifier circuit component that takes the PWM pulse as input and feeds the output to the ADC;
- Testing and optimization of an FIR filter that reduces noise from the output signal of the ADC;
- Setting up the alphanumeric keypad so that the user may input their desired voltage to be displayed on an LED display;
- Mainting the 7-segment display;
- Coding a controller function that automates the changes to be made on the PWM's duty cycle so that the previous or default voltage updates to the target voltage on the LED display;
- Coding a finite state machine function that enables the user to Enter and Delete digits for the target voltage, Reset the target voltage, and put the device to Sleep by using the keypad;
- Implementating the program's features using CMSIS-RTOS and multithreading;
- Reducing the power consumption of the device when it is in sleep mode using CMSIS-RTOS.

3 Theory and Hypothesis

3.1 Theory

3.2 Hypothesis

4 Implementation

4.1 Filtering

The devices uses a modified FIR filter from experiment 2 [source] in order to reduce the noise of the ADC's signal. The two modifications are the following:

- Increasing from 5 coefficients to 10 being used in the moving average;
- Setting the first five coefficients to 0.05 and the last five coefficients to 0.15.

In this way, the 5 earliest samples hav have a significance of 0.05 and the 5 later samples have a significance of 0.15 when calculating the average of the 10 samples. The programmers drew this conclusion by trial and error and the empirical evidence proved that the above modifications to the FIR filter significantly reduced the signal's noise. The programmers have tested mutliple configurations, in which this report shall demonstrate three of them. One can draw comparisons from the following graphs generated by a run-time variables monitoring and visualization tool, STM Studio [source]:

The configuration of this filter's coefficients is [0.2, 0.2, 0.2, 0.2, 0.2]. In regular blue are the unfiltered values, and in light blue are the filtered values. This graph demonstrates the values read at runtime of the unfilted and filtered values that passed through the unmodified FIR filter. One can observe that there is a considerable amount of noise left from filtering.
The configuration of this filter's coefficients is [0.05, 0.05, 0.05, 0.05, 0.05, 0.15, 0.15, 0.15, 0.15, 0.15]. This filter considers the five earliest values to each have a significance of 0.15 and the five later values to each have a significance of 0.05. One can observe by the graph that this filter is a considerable improvement to the unmodified version. Although, the programmers believe that it could be optimized further.
The configuration of this filter's coefficients is [0.15, 0.15, 0.15, 0.15, 0.15, 0.05, 0.05, 0.05, 0.05, 0.05]. This filter considers the five later values to each have a significance of 0.15 and the five early values to each have a significance of 0.05. One can observe by the graph that this filter is an improvement to the previous version. The programmers believe that this version of the filter should perform well enough given the scope of the problem.

5 Testing and Observations

6 Conclusion

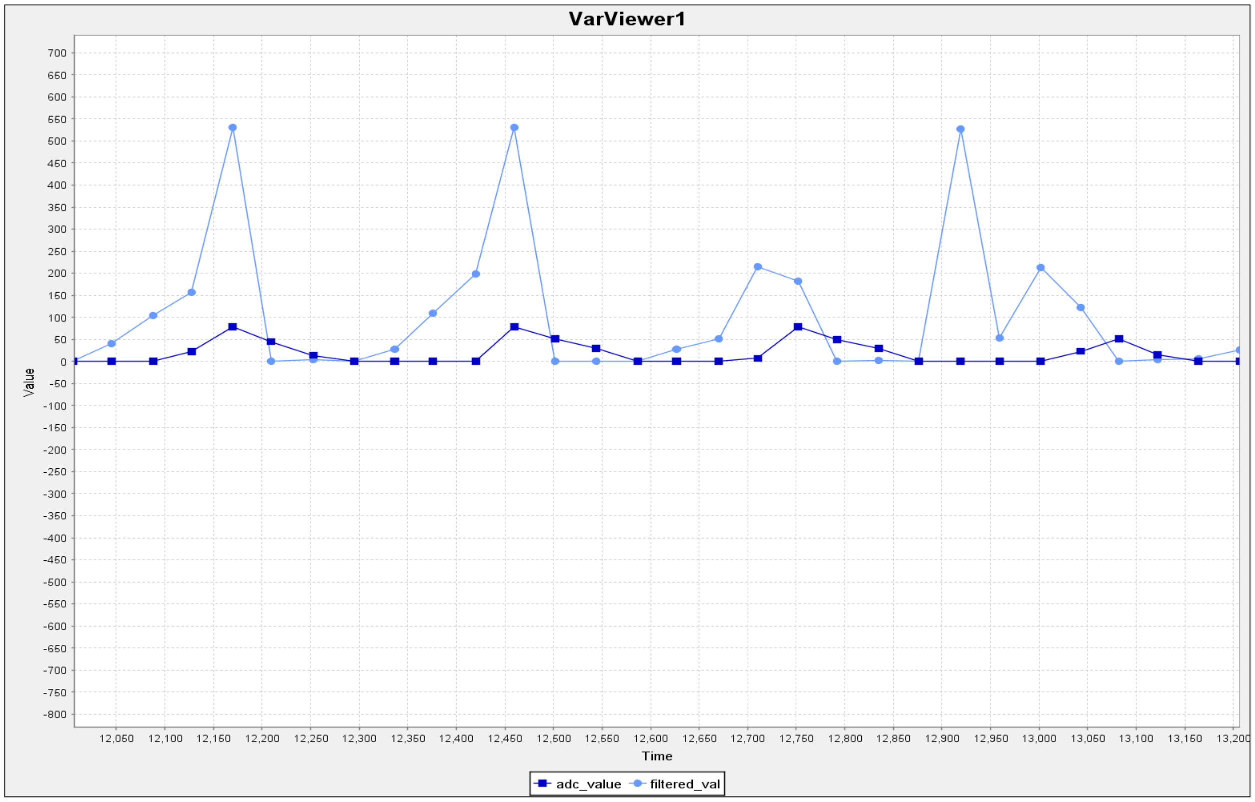

Figure 1: Filtered and Unfiltered Values VS Time (ms), Using Unmodified FIR Filter

Appendix A

GPIO Configuration Parameters

This appendix lists the configuration parameters set for each of the different GPIO pins (or classes of GPIO pins).

User Input Button

Parameter	Value
Mode	`GPIO_MODE_IT_RISING`
Pull	`GPIO_NOPULL`

Display Mode LEDs (4 of these)

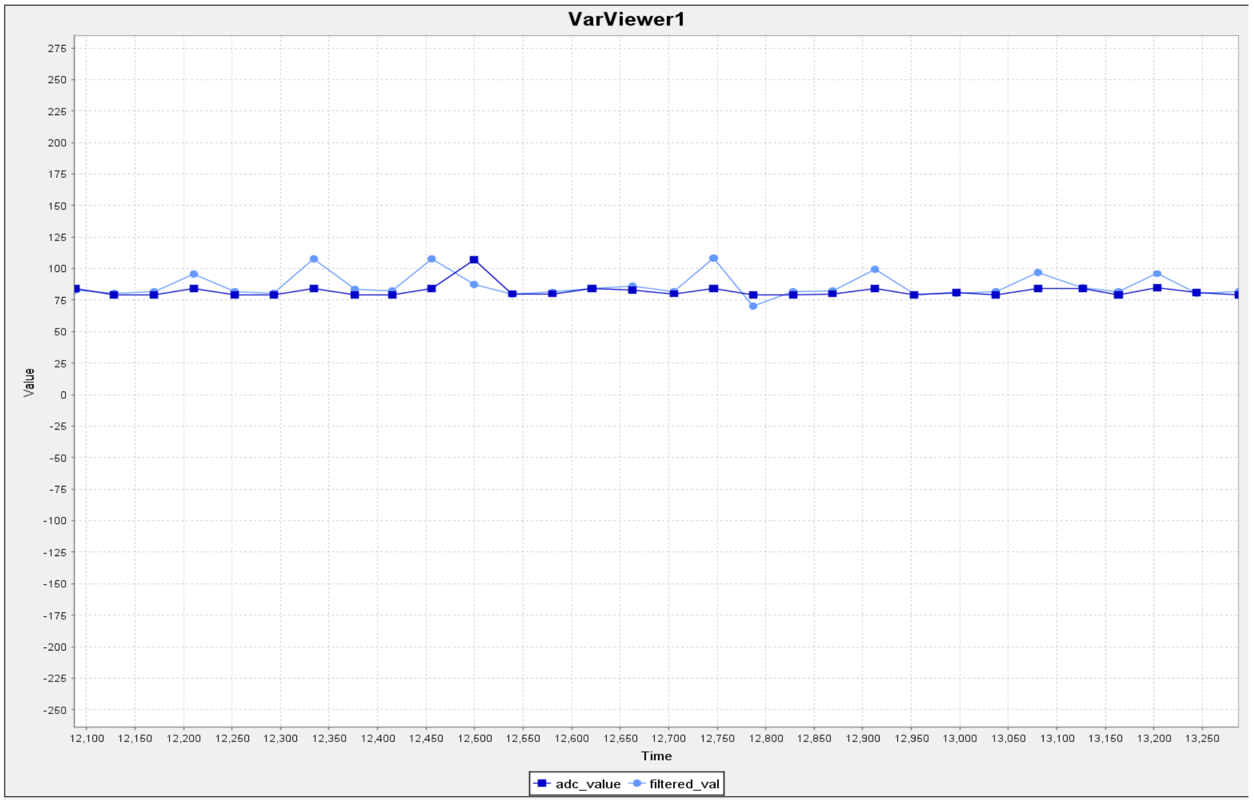

Figure 2: Filtered and Unfiltered Values VS Time (ms), Using High Coeffcients for Early Values

Parameter	Value
Mode	GPIO_MODE_OUTPUT_PP
Pull	GPIO_NOPULL
Speed	GPIO_SPEED_FREQ_LOW

Display Segment Pins (8 of these)

Parameter	Value
Mode	GPIO_MODE_OUTPUT_PP
Pull	GPIO_NOPULL
Speed	GPIO_SPEED_FREQ_LOW

Display Selector Pins (3 of these)

Parameter	Value
Mode	GPIO_MODE_OUTPUT_PP
Pull	GPIO_NOPULL
Speed	GPIO_SPEED_FREQ_LOW

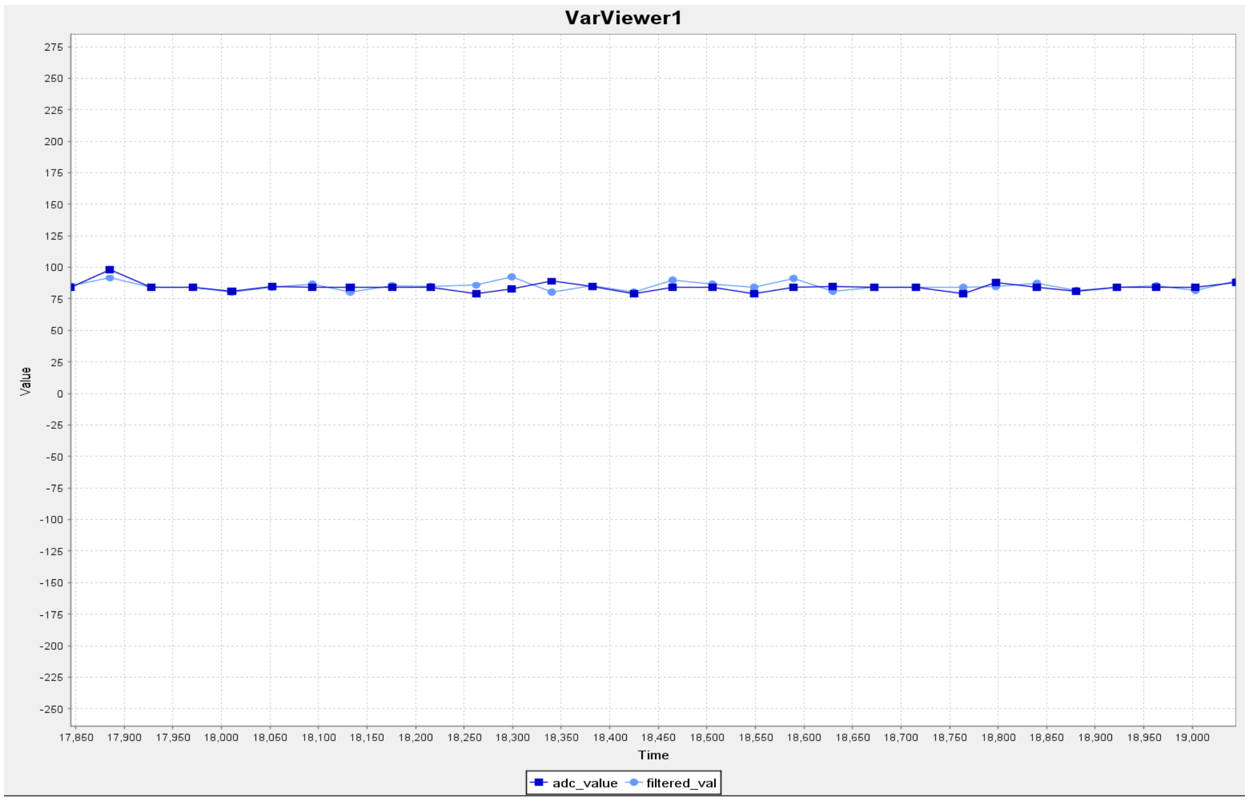

Figure 3: Filtered and Unfiltered Values VS Time (ms), Using High Coeffcients for Late Values

Appendix B

ADC Configuration Settings

ADC Instance Parameters

Parameter	Value
Clock Prescaler	`ADC_CLOCK_SYNC_PCLK_DIV2`
Resolution	`ADC_RESOLUTION_8B`
Scan Conversion Mode	Disabled
Continuous Conversion Mode	Disabled
Discontinuous Conversion Mode	Disabled
External Trigger Conversion Edge	`ADC_EXTERNALTRIGCONVEDGE_RISING`
External Trigger Conversion	`ADC_SOFTWARE_START`
Data Alignment	`ADC_DATAALIGN_RIGHT`
Number of Conversions	1
DMA Continuous Requests	Disabled
EOC Selection	`ADC_EOC_SINGLE_CONV`

ADC Channel Parameters (Channel 1)

Parameter	Value
Rank	1
Sampling Time	`ADC_SAMPLETIME_3CYCLES`

Appendix C

HAL Cube MX Autogenerated Code

Appendix D

Theory References