|  |
| --- |
|  |
| Lab 2: Sensor Data Acquisition, Digitizing, Filtering, and Digital I/O |
| ECSE 426 Microprocessor Systems |
| Group 4: Sidney Ng 260507001  Meng Yin Tao 260480207 |
|  |
|  |

Due: February 23rd 2015

|  |
| --- |
| [Type the abstract of the document here. The abstract is typically a short summary of the contents of the document. Type the abstract of the document here. The abstract is typically a short summary of the contents of the document.] |

Contents

[1. Problem Statement 2](#_Toc412123759)

[2. Theory and Hypothesis 2](#_Toc412123760)

[2.1. Analog-to-Digital Conversion 2](#_Toc412123761)

[2.2. Kalman Filter 2](#_Toc412123762)

[2.3. Data Interpretation 2](#_Toc412123763)

[2.4. Pulse Wave Modulation 2](#_Toc412123764)

[3. Implementation 3](#_Toc412123765)

[3.1. Data Processing 3](#_Toc412123766)

[3.1.1. Data Acquisition and Digitizing (ADC) 3](#_Toc412123767)

[3.1.2. Data Filtering (Kalman Filter) 3](#_Toc412123768)

[3.1.3. Data Conversion 3](#_Toc412123769)

[3.2. Visual Feedback 3](#_Toc412123770)

[3.2.1. Visual Display (GPIO) 3](#_Toc412123771)

[3.2.2. Alarm (PWM) 3](#_Toc412123772)

[4. Testing and Observations 4](#_Toc412123773)

[4.1. Terminal Window 4](#_Toc412123774)

[4.2. Visual Feedback 4](#_Toc412123775)

[5. Conclusion 4](#_Toc412123776)

[References 5](#_Toc412123777)

[Appendix A – Matlab Simulation Results 6](#_Toc412123778)

Lab 2: Sensor Data Acquisition, Digitizing, Filtering, and Digital I/O

# 1. Problem Statement

Data acquisition and signal processing are common operations performed by embedded microprocessor systems. Within the scope of this experiment, we will implement a system which collects data from a temperature sensor and displays the results using LEDs.

# 2. Theory and Hypothesis

## 2.1. Analog-to-Digital Conversion

* Resolution and step sizes
* How to interpret the output of an ADC

## 2.2. Kalman Filter

* What’s a Kalman Filter?
* Meaning of variables
* Choice of values (from Matlab simulation)

## 2.3. Data Interpretation

* Translate to understandable numbers
* Show the equations and values provided in Reference Manual (need reference!!)

## 2.4. Pulse Wave Modulation

* What’s a PWM?
* How does duty\_cycle and period influence the fade-in fade-out effect

# 3. Implementation

## 3.1. Data Processing

* Controlled by interrupt, only executes every 0.02s

### 3.1.1. Data Acquisition and Digitizing (ADC)

* Temperature sensor
* Sampling frequency – set by systick & interrupt

We used Systick, a software implementation of a system clock, for a periodic sampling of the temperature sensor values. We were required to sample the sensor at a frequency of 50 Hz (reference LabExperiment2 Functional Requirements section). The system’s clock has a frequency of 168 MHz, so we set up the systick clock by setting it’s frequency as the system clock speed divided by 50.

* ADC configuration (refer to previously mentioned theory and user manual)

Initializing the ADC required configuring the common ADC settings and the specific ADC settings. For the common ADC settings, we set the following: ADC\_Mode as ADC\_Mode\_Independent, ADC\_Prescaler as ADC\_Prescaler\_Div2, ADC\_DMAAccessMode as ADC\_DMAAccessMode\_Disabled, and ADC\_TwoSamplingDelay as ADC\_TwoSamplingDelay\_5Cycles. The mode is independent since we only needed to use one ADC component. The prescaler is set to div2 because that is the smallest division choice available. DMA Access Mode was disabled since we did not need to use any direct memory accesses. The two sampling delay indicates the amount of cycles to pass between two samples are taken. We chose 5 cycles since that was the highest frequency.

For the specific ADC settings, we configured as the following: ADC\_Resolution as ADC\_Resolution\_12b, ADC\_ScanConvMode as DISABLED, ADC\_ContinuousConvMode as DISABLED, ADC\_ExternalTrigConv as ADC\_ExernalTrigConvEdge\_None, ADC\_DataAlign as ADC\_DataAlign\_Right, and ADC\_NbrOfConversion as 1. We set the resolution as 12 bits since that was the highest resolution size. We disabled the Scan Conversion Mode since we only needed to do the conversion in a single channel for one sensor. We disabled continuous conversion mode because we did not need a continuous sampling. We set the external trigger conversion edge to none since we are using a software interrupt. We set data align to right because that’s how integers in C are represented. The number of conversions is set to one since we are only doing one conversion per sample.

For the channel configurations, we set the following: ADCx as ADC1, ADC\_Channel as ADC\_Channel\_16, Rank as 1, and ADC\_SampleTime as ADC\_SampleTime\_480Cycles. The temperature sensor is hardwired to ADC1 on channel 16. We only had one channel, so the value for rank did not matter. We selected 480 cycles as sample time because \_\_\_\_\_\_

* Mention that the data is passed to the Kalman Filter

The sampled data taken from the ADC, is passed forwards to the Kalman filter.

### 3.1.2. Data Filtering (Kalman Filter)

* Mention that the data comes from the ADC

We reused the Kalman filter from Lab Experiment 1, with some changes. Instead of an array of inputs and outputs, each iteration will only pass in a scalar input value and pass back one output value. The parameters that we used for the Kalman filter we determined experimentally (see Section 4.1). The output values are passed as input to the conversion function.

* Implementation of the filter using previously determined (optimal) variables
* Mention that the data is sent to conversion

### 3.1.3. Data Conversion

* Mention that the data received from the filter needs to be translated in order to be interpreted
* First get the voltage
* Second get the temperature
* Mention that the temperature changes are displayed using LEDs

## 3.2. Visual Feedback

* Independent of the interrupt, can run while data processing is waiting

### 3.2.1. Visual Display (GPIO)

* Mention that when temperature is below the threshold, we display the change
* GPIO configuration (refer to user manual?)

For the GPIO settings we used the following: GPIO\_Pin as GPIO\_Pins 12 to 15 or’d together, GPIO\_Mode as GPIO\_Mode\_OUT, GPIO\_Speed as GPIO\_Speed 100MHz, GPIO\_OType as GPIO\_Otype\_PP, GPIO\_PuPd as GPIO\_PuPd\_NOPULL. We set the pins 12 to 15 since these are the pins hardwired to the LEDs. We set the mode as out since we are writing to the LEDs. We set the speed as 100 MHz since it is the fastest speed. We set the output type as PP and PullUp and PullDown as NOPULL so that \_\_\_\_\_\_\_

* How to track temperature increase and decrease

### 3.2.2. Alarm (PWM)

* Mention that when temperature is above the threshold, we use and alarm
* Increment duty\_cycle by 10% to achieve fade-in fade-out

# 4. Testing and Observations

## 4.1. Terminal Window

* Display intermediate values
  + For Kalman Filter -- Compare against Matlab results
* Know when we passed the threshold

## 4.2. Visual Feedback

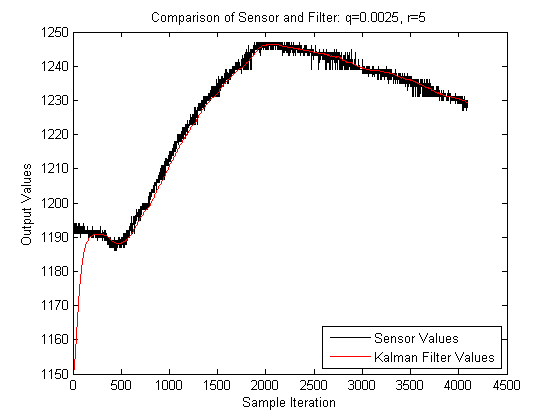
* Observe the clockwise/counter-clockwise LEDs
* Observe fade-in fade-out effect beyond threshold

# 5. Conclusion

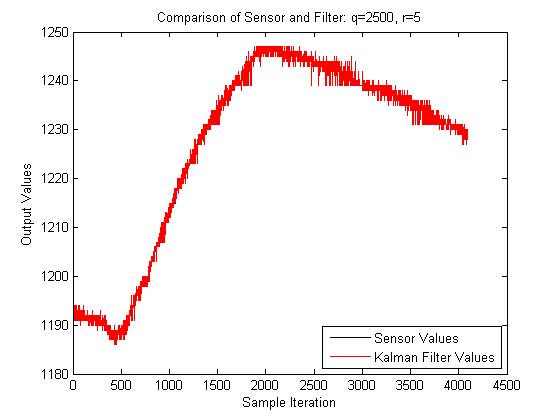
# References

# Appendix A – Matlab Simulation Results

Figure



Figure



Figure

