|  |
| --- |
|  |
| 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

[List of Tables 2](#_Toc412477390)

[List of Figures 2](#_Toc412477391)

[1. Problem Statement 3](#_Toc412477392)

[2. Theory and Hypothesis 3](#_Toc412477393)

[2.1. Analog-to-Digital Conversion 3](#_Toc412477394)

[2.2. Kalman Filter 4](#_Toc412477395)

[2.3. Data Interpretation 4](#_Toc412477396)

[2.4. Pulse Wave Modulation 5](#_Toc412477397)

[3. Implementation 6](#_Toc412477398)

[3.1. Data Processing 6](#_Toc412477399)

[3.1.1. Data Acquisition and Digitizing (ADC) 6](#_Toc412477400)

[3.1.2. Data Filtering (Kalman Filter) 7](#_Toc412477401)

[3.1.3. Data Conversion 7](#_Toc412477402)

[3.2. Visual Feedback 8](#_Toc412477403)

[3.2.1. Visual Display (GPIO) 8](#_Toc412477404)

[3.2.2. Alarm (PWM) 9](#_Toc412477405)

[4. Testing and Observations 9](#_Toc412477406)

[4.1. Matlab Simulation 9](#_Toc412477407)

[4.2. System Feedback 9](#_Toc412477408)

[5. Conclusion 9](#_Toc412477409)

[References 10](#_Toc412477410)

[Appendix A – Matlab Simulation Results 11](#_Toc412477411)

# List of Tables

[Table 1: ADC Initialization Parameters 6](#_Toc412474401)

[Table 2: ADC Configuration Parameters 6](#_Toc412474402)

[Table 3: Channel Configuration Parameters 7](#_Toc412474403)

[Table 4: GPIO Configuration Parameters 8](#_Toc412474404)

[Table 5: Modulo to GPIO Pin 8](#_Toc412474405)

# List of Figures

[Figure 1: Kalman Filter Parameter q=0.0025, r=5 11](file:///C:\Users\TQ.AlexTao\Documents\Bei\ecse426\Lab%202%20Base%20Project\LAB%202.docx#_Toc412474406)

[Figure 2: Kalman Filter Parameter q=2500, r=5 12](#_Toc412474407)

[Figure 3: Kalman Filter Parameter q=0.0025, r=500 13](#_Toc412474408)

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

An analog-to-digital converter (ADC) translates an analog input into a digital value which can be transmitted over a data bus. The ADC used for this implementation has 12-bit resolution and operates in a range of 0 to 3 volts [1]. The resolution and the range define the step size of the ADC according to equation (1).

(1)

Where Stepsize is the voltage range represented by the same digital value

Range is the total possible analog value [3v - 0v = 3v]

Numberstep is the number of possible digital values. [212 = 4098]

By substituting the given numerical values into equation (1), the step size of our ADC is found to be 0.000732V / step.

To interpret the output of the ADC, we need to perform the reverse calculation as shown in equation (2).

(2)

Where Valueanalog is the analog value corresponding to the digital value

Value­digital is the digital value at the output of the ADC

Stepsize is the voltage range represented by the same digital value

For example, the output of the ADC is 1080, then the corresponding analog value should be approximately 0.790629575V.

The many-to-one mapping of the ADC leads to loss of precision. Therefore, the calculated analog value will differ from the actual analog value measured at the input of the ADC. For this reason, a filter should be inserted between the output of the ADC and the computation of the analog value.

## 2.2. Kalman Filter

Kalman Filter is a low pass-filter used to block high frequency noise based on the current state. Its state is defined by five parameters: process noise covariance (q), measurement noise covariance (r), value (x), estimation error covariance (p) and kalman gain (k). As the matlab simulation shows (see Appendix A), q and r are the key parameters to be initialized when implementing a Kalman Filter. Together, these two parameter set how close the filtered value should follow the raw input. While q and r are fixed during the initialization of the filter, x, p and k are updated as the filter runs. A good initial approximation on those three parameters will result in the filtered value to converge more quickly.

At each iteration, a new value p, k and x is calculated and updated. These calculations are preformed according to equations (3) to (6) [2].

(3)

(4)

(5)

(6)

Where measurement is the current sample value.

The updated value for x is the output of the filter at this iteration. As equation (5) shows, this value is used in the next iteration to filter the next input.

## 2.3. Data Interpretation

After inserting a filter between the ADC output and computing the analog value, the voltage resulting from equation (2) should be more accurate. This value can in turn be translated into the corresponding temperature using equation (7) [1], [3].

(7)

Where Temp˚C is the temperature corresponding to a given voltage

V­sense is the voltage at the sensor computed by converting the digital value

V25 is the reference temperature at 25˚C [0.76V]

Avg\_Slope is the average slope of temperature vs V­­sense

## 2.4. Pulse Wave Modulation

The objective of Pulse Width Modulation (PWM) is to vary the enable time, or duty cycle, within a period. Every period, the duty cycle is incremented until it reaches the full period duration, and then decremented until it reaches 0 duration. The process is continually repeated. By adjusting the duty cycle, the duration of the time the LED is turned on varies. If the period is short enough, then the human eye will not pick up on the difference between an on or off state and instead interpret the varying duty cycle times as differences in light intensity. Thus a fading effect is achieved.

# 3. Implementation

The overall system is controlled by a 168MHz system core clock. It is divided of two main components, data processing and visual feedback. The implementation of each is detailed in the following sections. The processor execution mainly resides within a main loop. Inside of the loop, the program checks whether the interrupt handler has set an interrupt flag. If it has, then it will execute the data processing portion. The portion controlling the LED display lies outside of the conditional check for the interrupt flag.

## 3.1. Data Processing

The data processing component is controlled by the clock operating at a rate derived from the system core clock. This software implemented system clock is called Systick, which counts up to a specific number of pulses before throwing an interrupt. The required sampling frequency of 50Hz is achieved by dividing the system core clock of 168MHz by 50Hz. The Systick interrupt handler will throw an interrupt every 0.02ms and allow one data sample to be taken and processed.

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

The built in temperature sensor is connected to ADC1 through channel 16. To collect data from the temperature sensor, we initialized the ADC as summarized in table 1.

Table : ADC Initialization Parameters

|  |  |
| --- | --- |
| Initialization Parameter | Value |
| ADC\_Mode | ADC\_Mode\_Independent |
| ADC\_Prescaler | ADC\_Prescaler\_Div2 |
| ADC\_DMAAccessMode | ADC\_DMAAccessMode\_Disabled |
| ADC\_TwoSamplingDelay | 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 specific ADC setting, we imposed the following configurations.

Table : ADC Configuration Parameters

|  |  |
| --- | --- |
| Configuration Parameter | Value |
| ADC\_Resolution | ADC\_Resolution\_12b |
| ADC\_ScanConvMode | DISABLED |
| ADC\_ContinuousConvMode | DISABLED |
| ADC\_ExternalTrigConv | ADC\_ExernalTrigConvEdge\_None |
| ADC\_DataAlign | ADC\_DataAlign\_Right |
| ADC\_NbrOfConversion | 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 are represented in C. The number of conversions is set to one since we are only doing one conversion per sample.

The channel on which samples are taken is configured as shown in table 3.

Table : Channel Configuration Parameters

|  |  |
| --- | --- |
| Configuration Parameter | Value |
| ADCx | ADC1 |
| ADC\_Channel | ADC\_Channel\_16 |
| Rank | 1 |
| ADC\_SampleTime | ADC\_SampleTime\_480Cycles |

As previously mentioned, the temperature sensor is hardwired to ADC1 over channel 16. Since this is the only used, the value for rank does not matter. By setting ADC sample time to 480 cycles, we have a total conversion period of 492 cycles [4].

Finally, the sampled data taken from the ADC is forwarded to the Kalman filter.

### 3.1.2. Data Filtering (Kalman Filter)

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.

### 3.1.3. Data Conversion

Once the digital data filtered, it can be converted to its corresponding analog value and then used to compute the temperature. First to convert the filtered digital data, we used equation (2). Second, the resulting voltage is in turn fed into equation (7). Finally, the resulting temperature value is passed on to the LED for display.

## 3.2. Visual Feedback

The visual display for your program used two operational modes, one below a threshold temperature and one above a threshold temperature. The threshold temperature we set to 55 degrees Celsius (see section 4.2).

To configure the GPIO to output to the LEDs, we used the settings as in table 4.

Table : GPIO Configuration Parameters

|  |  |
| --- | --- |
| **Configuration parameter** | **Value** |
| **GPIO\_Pin** | GPIO\_Pin\_12 | GPIO\_Pin\_13 | GPIO\_Pin\_14 | GPIO\_Pin\_15 |
| **GPIO\_Mode** | GPIO\_Mode\_OUT |
| **GPIO\_Speed** | GPIO\_Speed\_100MHz |
| **GPIO\_OType** | GPIO\_Otype\_PP |
| **GPIO\_PuPd** | 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. By setting the output type to push-pull, the output is controlled by a pair of transistors. By setting pull-up / pull-down to no pull, the default output value is undefined. By using no pull we attempt to control the LEDs by setting and resetting them instead of have them go to a default state.

### 3.2.1. Visual Display (GPIO)

The operational mode when the temperature is below the threshold indicates changes in temperature. Every two degree change in temperature, the current LED will turn off and another LED will turn on. If the change is positive, then the next clockwise LED is turned on, otherwise, the next counter-clockwise LED is turned on. To specify the LED we implemented a counter. Then we took the modulo 4 of the counter and specified a results as in table 5. Whenever temperature changed, we can then increment or decrement the counter accordingly. The reference temperature is updated at every two degree change.

Table : Modulo to GPIO Pin

|  |  |
| --- | --- |
| **Modulo Result** | **GPIO\_Pin** |
| 0 | 12 |
| 1 | 13 |
| 2 | 14 |
| 3 | 15 |

### 3.2.2. Alarm (PWM)

The operational mode when the temperature is above the threshold acts as an alarm indicating that the temperature has reached a critical point. The alarm is implemented by using pulse width modulation (see section 2.4) when enabling the LEDs. The duty cycle of the PWM is implemented as a percentage. The actual duty cycle that the counter counts to is found by multiplying the duty cycle percentage by the set period. In this way, PWM can be reused with a different period without specifying the interval of increase for the duty cycle. Instead, adjusting the duty cycle only requires adding or subtracting 10% to the percentage value.

# 4. Testing and Observations

## 4.1. Matlab Simulation

In order to test the Kalman filter and obtain optimal values for the parameters, we collected data from printing out one run of the system. Then in Matlab we implemented a copy of the Kalman filter code and imported the sampled data. Finally we plotted both the filter output and the sampled data while varying the Kalman filter state parameters q and r. We noticed from the Kalman filter implementation that x, p, and k all changed at the first iteration, so there was no need to select specific values for them through experimentation (see Section 2.2). X is the estimated error, so we simply selected a value close to the raw data for 38 degrees Celsius. We left p and k equal to 0 to begin. We noticed that as we held r constant and increased q from 0.0025 to 25, the noise increased (Figures 1-5). Holding q constant at 0.0025 and increasing r from 5 to 500 the curve was not representative of the sampled data (Figures 6-8). The most optimal values we found to be q = 0.0025 and r = 5 as seen in Figure 1.

## 4.2. System Feedback

To test that our system was working, we printed filtered values, intermediate values, and converted temperature to the terminal window. Then we corroborated changes in the LEDs with the printed values. We made sure that the LEDs only changed when the temperature changed by two degrees. We also checked that an increase in temperature enabled the next clockwise LED and a decrease in temperature enabled the next counter-clockwise LED. We checked that the alarm started flashing when we heated up the processor past the threshold. Additionally, we checked the fading in and fading out of the PWM.

For our temperature values, we noticed that the values were higher than should be observed, given that we worked at room temperature. Our temperature values stayed at around 38 degrees Celsius when turned on without heating up or cooling down. However, according to the Lab Experiment 2 Document, the sensors are prone to error and that values above 30 degrees can be expected, so we have not attempted to adjust the values.

# 5. Conclusion

* Take into account the wrap-around issue
* Consider using average when converting digital to analog

# References

[1] STMicroelectronics. *STM32F405xx STM32F407xx Datasheets*. 2013, pp. 129-134.

[2] *ECSE426 Microprocessor Systems Lab 1: One-Domensional Kalman Filter*. Winter 2015, pp. 1-2.

[3] STMicroelectronics. *RM0090 Reference manual*. 2011, pp. 229-230.

[4] STMicroelectronics. *RM0090 Reference manual*. 2011, pp. 215.

# Appendix A – Matlab Simulation Results

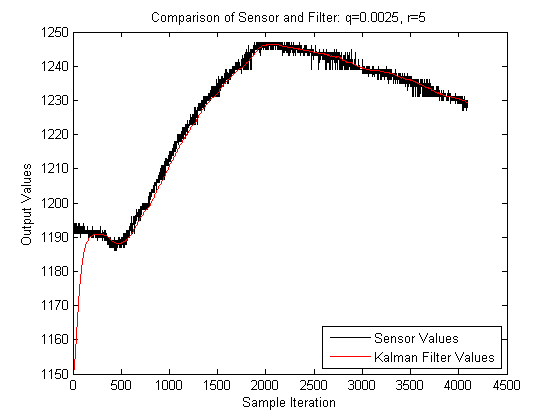


Figure : Kalman Filter Parameter q=0.0025, r=5

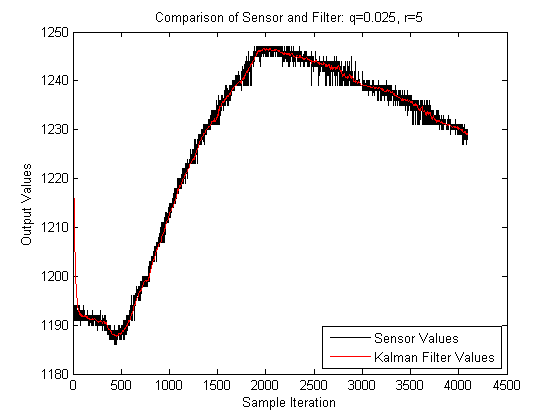


Figure : Kalman Filter Parameters q = 0.025, r = 5

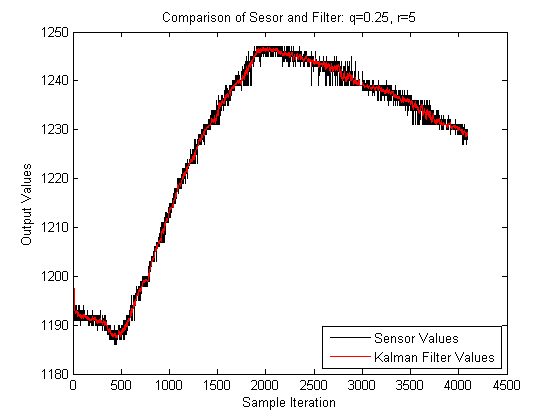


Figure : Kalman Filter Parameters q = 0.25, r = 5

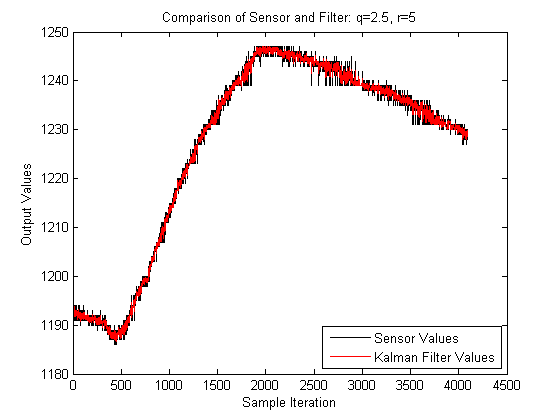


Figure : Kalman Filter Parameters q = 2.5, r = 5

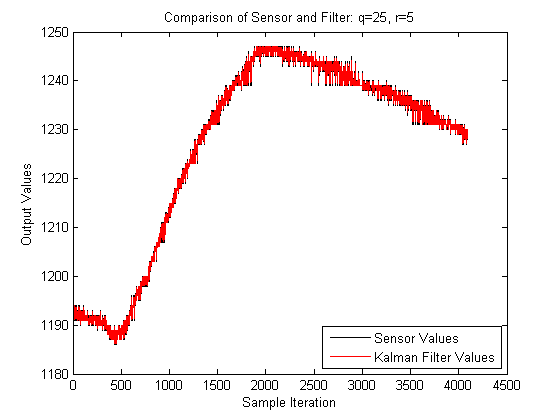


Figure : Kalman Filter Parameters q = 25, r = 5

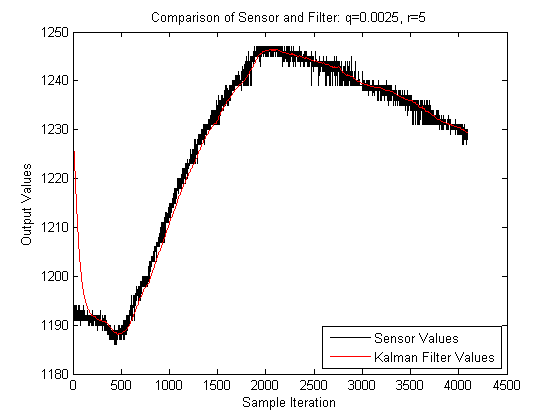


Figure : Kalman Filter Parameters q = 0.0025, r = 5

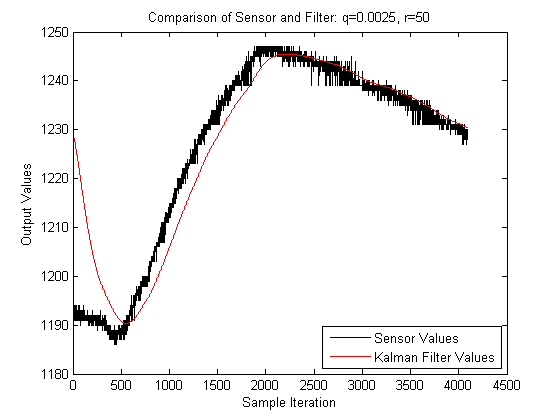


Figure : Kalman Filter Parameters q = 0.0025, r = 50

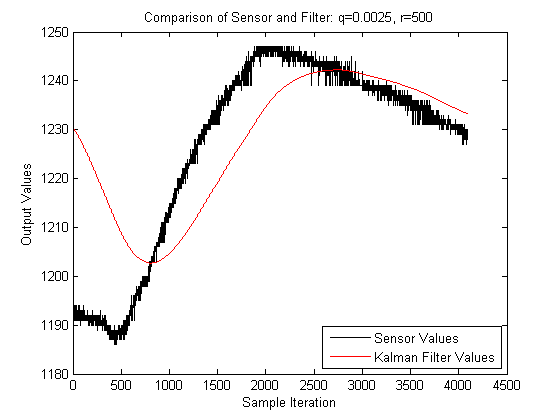


Figure : Kalman Filter Parameters q = 0.0025, r = 500