|  |
| --- |
|  |
| 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

|  |
| --- |
| The goal of this experiment is to implement a temperature monitor which collects data via a temperature sensor. The data is processed to finally display a change in temperature using LEDs. The following report will discuss the detailed implementation and results of this experiment. |

Contents

[List of Tables 2](#_Toc412485702)

[List of Figures 2](#_Toc412485703)

[1. Problem Statement 3](#_Toc412485704)

[2. Theory and Hypothesis 3](#_Toc412485705)

[2.1. Analog-to-Digital Conversion 3](#_Toc412485706)

[2.2. Kalman Filter 4](#_Toc412485707)

[2.3. Data Interpretation 4](#_Toc412485708)

[2.4. Pulse Wave Modulation 5](#_Toc412485709)

[3. Implementation 6](#_Toc412485710)

[3.1. Data Processing 6](#_Toc412485711)

[3.1.1. Data Acquisition and Digitizing (ADC) 7](#_Toc412485712)

[3.1.2. Data Filtering (Kalman Filter) 8](#_Toc412485713)

[3.1.3. Data Conversion 8](#_Toc412485714)

[3.2. Visual Feedback 8](#_Toc412485715)

[3.2.1. Visual Display (GPIO) 9](#_Toc412485716)

[3.2.2. Alarm (PWM) 9](#_Toc412485717)

[4. Testing and Observations 10](#_Toc412485718)

[4.1. Matlab Simulation 10](#_Toc412485719)

[4.2. System Feedback 10](#_Toc412485720)

[5. Conclusion 10](#_Toc412485721)

[References 11](#_Toc412485722)

[Appendix A – Matlab Simulation Results 12](#_Toc412485723)

# List of Tables

[Table 1: ADC Initialization Parameters 7](#_Toc412484837)

[Table 2: ADC Configuration Parameters 7](#_Toc412484838)

[Table 3: Channel Configuration Parameters 7](#_Toc412484839)

[Table 4: GPIO Configuration Parameters 8](#_Toc412484840)

[Table 5: Modulo to GPIO Pin 9](#_Toc412484841)

# List of Figures

[Figure 1: Pulse Modulation 5](#_Toc412485724)

[Figure 2: Data Flow 6](#_Toc412485725)

[Figure 3: Kalman Filter Parameter q=0.0025, r=5 12](file:///P:\microprocessor%20labs\ecse426\Lab%202%20Base%20Project\LAB%202.docx#_Toc412485726)

[Figure 4: Kalman Filter Parameters q = 0.025, r = 5 12](#_Toc412485727)

[Figure 5: Kalman Filter Parameters q = 0.25, r = 5 13](#_Toc412485728)

[Figure 6: Kalman Filter Parameters q = 2.5, r = 5 13](#_Toc412485729)

[Figure 7: Kalman Filter Parameters q = 25, r = 5 14](#_Toc412485730)

[Figure 8: Kalman Filter Parameters q = 0.0025, r = 5 14](#_Toc412485731)

[Figure 9: Kalman Filter Parameters q = 0.0025, r = 50 15](#_Toc412485732)

[Figure 10: Kalman Filter Parameters q = 0.0025, r = 500 15](#_Toc412485733)

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 temperature monitor using the STM32F407VG discovery board. The temperature monitor 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 approximately 0.000732V / step.

To interpret the output of the ADC, we need to perform the reverse computation 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.

Due to the many-to-one mapping of the ADC, we expect a 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 parameters 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 for 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. We expect the output to match that of the Matlab simulation.

## 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*

We expect the temperature to correctly reflect the room temperature (25˚C).

## 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, 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. Figure 1 shows the pulse modulation. We expect to achieve a fading effect.

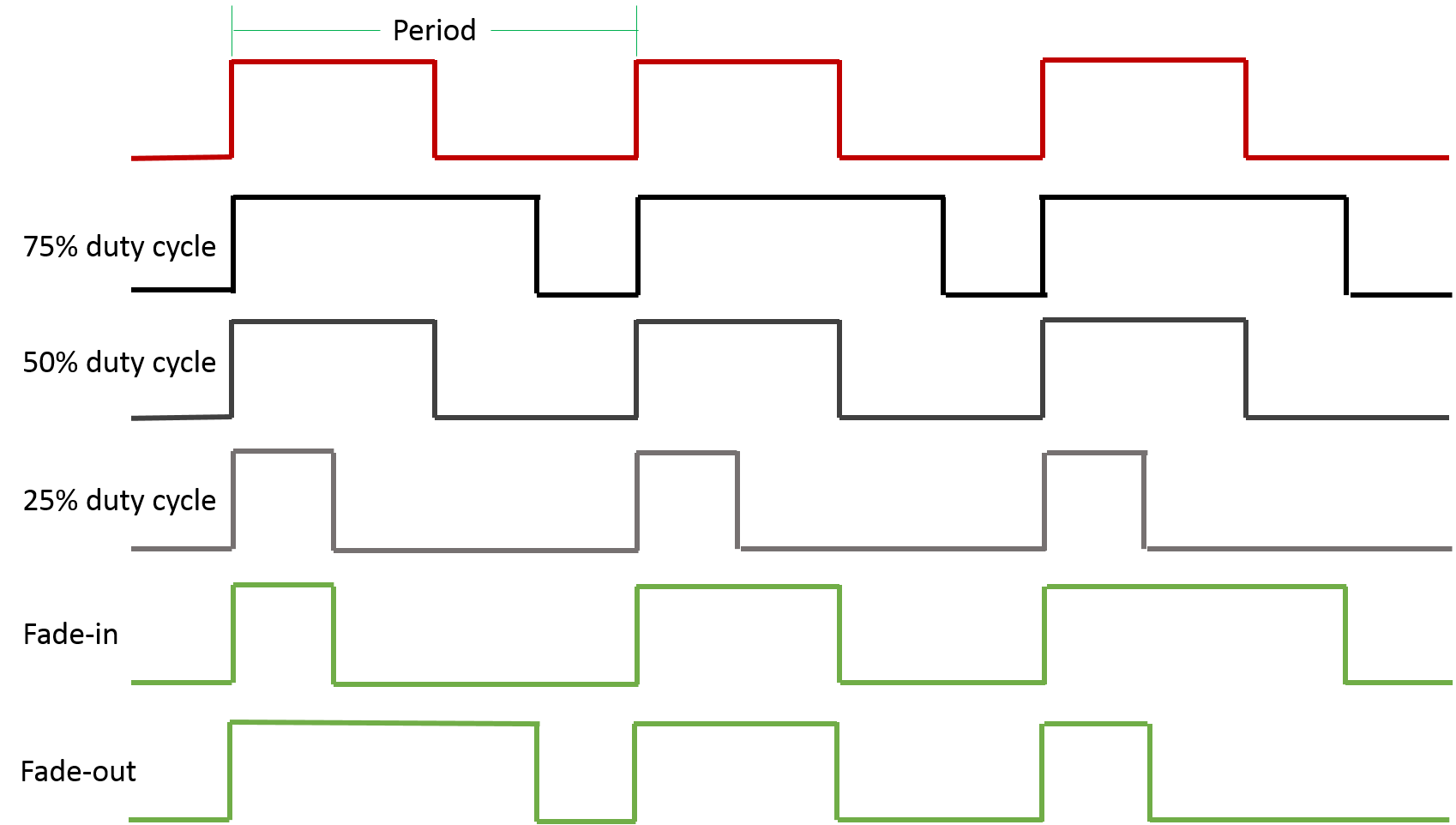


Figure 1: Pulse Modulation

# 3. Implementation

The overall system (see Figure 2) 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.

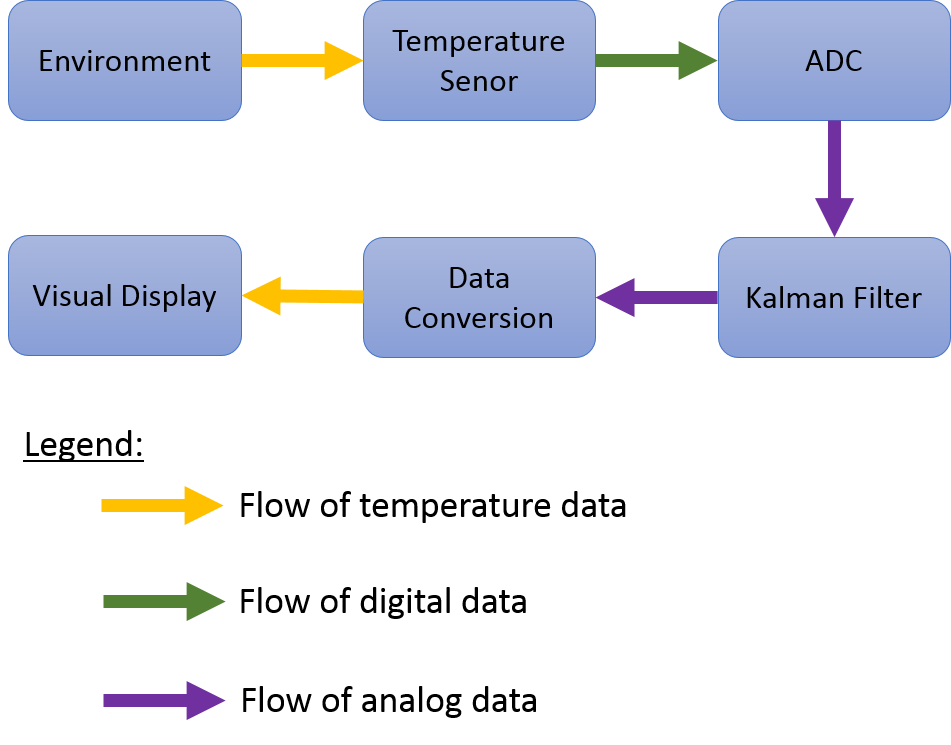


Figure 2: Data Flow

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 [3]. To collect data from the temperature sensor, we initialized the ADC as summarized in table 1.

Table 1: 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 is disabled since we do 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 2: 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 it is 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 this is 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 3: 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 channel 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

Independent of the interrupt state, the visual display for our program uses two operational modes, one below a threshold temperature and one above a threshold temperature. The threshold temperature is 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 4: 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. For very two degrees 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 take the modulo 4 of the counter value the result maps to a GPIO pin as shown in table 5. Whenever temperature changes, the counter is incremented or decremented accordingly. The reference temperature is updated at every two degrees change.

Table 5: 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 periods without specifying the interval of increase for the duty cycle. Instead, adjusting the duty cycle only requires changing the percentage value.

# 4. Testing and Observations

## 4.1. Matlab Simulation

In order to test the Kalman filter and to 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 simulation that the value of x, p, and k 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˚C. 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 3-7). Holding q constant at 0.0025 and increasing r from 5 to 500 the curve was not representative of the sampled data (Figures 8-10). 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 varied the temperature using a hair dryer. 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 changes when the temperature changes 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 starts to flash when the temperature has been brought up past the threshold. Additionally, we checked the fading in and fading out of the PWM.

For our temperature values, we noticed that the values are higher than what should be observed, given that the sensor is operating at room temperature. Our temperature values stayed at around 38˚C 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].

# 5. Conclusion

In this experiment we implement a functional system that detects the temperature changes of a microprocessor and displays visual feedback to a user using LEDs. Data from the sensor is processed by a series of converters and a Kalman filter. After successfully processing of the data, the LEDs turn on and off sequentially clockwise when temperature rises and turn on and off sequentially counter-clockwise when temperature falls. Past the threshold temperature, the LEDs blink to alert the user that the temperature is too high. This temperature monitor can be further applied to a system requiring critical temperature monitoring such as a car engine or even a nuclear power plant.

# 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.

[5] A. Suyyagh, *ECSE426 Microprocessor Systems Lab 2: Sensor Data Acquisition, Digitizing, Filtering, and Digital I/O*. Winter 2015, pp. 3.

# Appendix A – Matlab Simulation Results

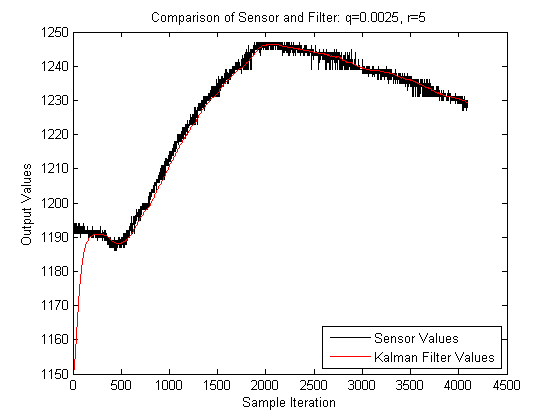


Figure 3: Kalman Filter Parameter q=0.0025, r=5

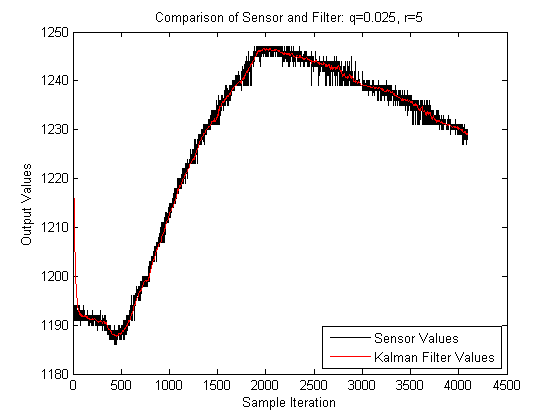


Figure 4: Kalman Filter Parameters q = 0.025, r = 5

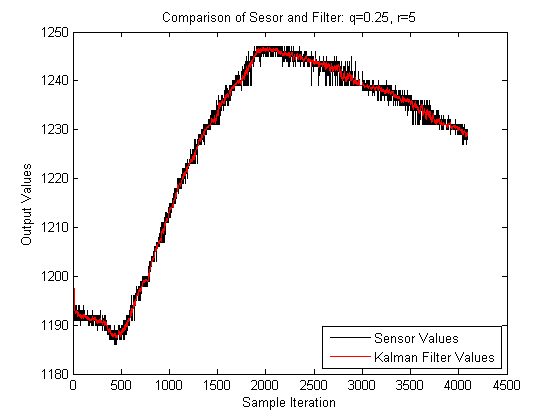


Figure 5: Kalman Filter Parameters q = 0.25, r = 5

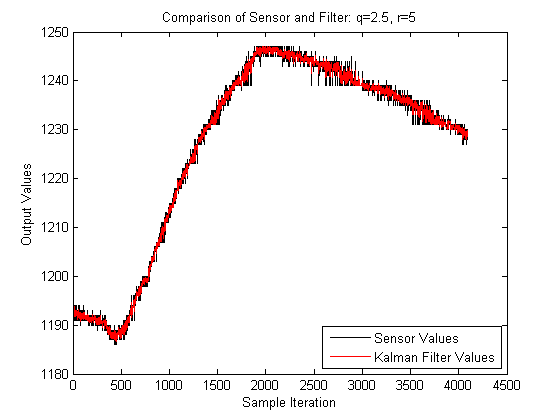


Figure 6: Kalman Filter Parameters q = 2.5, r = 5

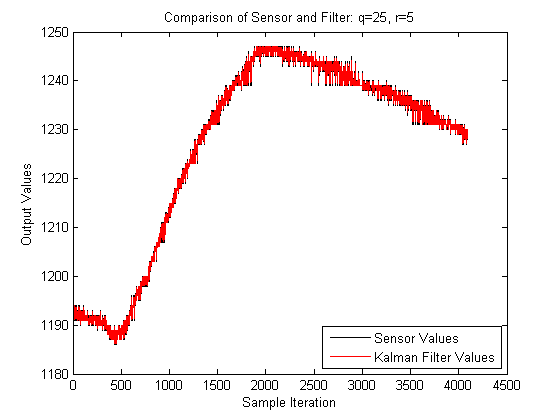


Figure 7: Kalman Filter Parameters q = 25, r = 5

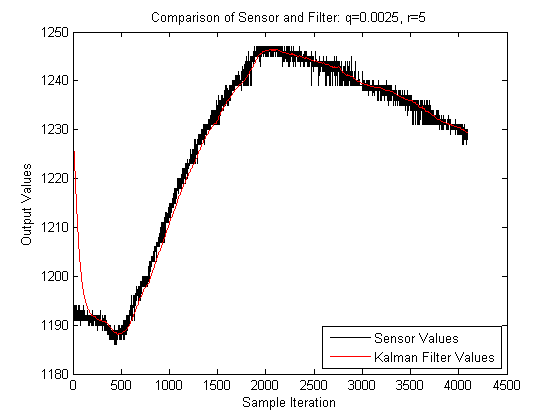


Figure 8: Kalman Filter Parameters q = 0.0025, r = 5

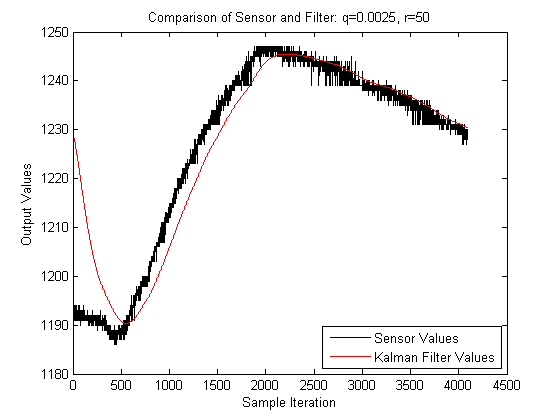


Figure 9: Kalman Filter Parameters q = 0.0025, r = 50

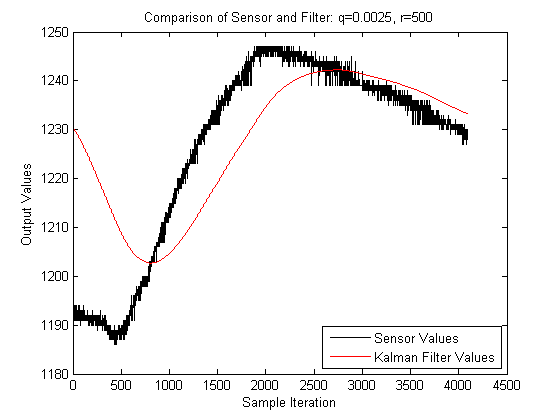


Figure 10: Kalman Filter Parameters q = 0.0025, r = 500