Embedded System
Experiment 3

Submitted By:

Pratyush Jaiswal
18EE30021

AIM:

Implement Serial Communication and then receive the input values and output the filtered values.

Procedures:

- First the filter coefficients are generated using the fdatool (the filter from the experiment 2 is used here).
- The serial communication is implemented using the datasheet of the Amtega328p
- Then the filter code from the experiment 2 is used for filtering the inputs received serially.
- After every filtered value, the data is transmitted serially to the serial monitor.
- Since all the outputted values are in ascii values, they are changed to decimal and then used to make the graphs using the filter code (using the same as in experiment 2).

Atmel Code for serial communication and filtering:

```
;
; Serial_com.asm
;
; Created: 19-02-2021 15:23:43
; Author : Pratyush Jaiswal
; Roll   : 18EE30021
;

; Replace with your application code

.EQU F_CPU=16000000
.EQU BR=9600
.EQU R_X=(F_CPU/(16*BR)-1)

LDI R16,HIGH(RAMEND)    ; initialize high
OUT SPH,R16                   ; byte of SP
LDI R16,LOW(RAMEND)           ; initialize low
OUT SPL,R16                   ; byte of SP
```

```
USART_Init:
            ; Set baud rate
            LDI R16, 0
            sts UBRR0H, R16
            LDI R16, 103
            sts UBRR0L, R16

            ; Enable receiver and transmitter
            LDI R16, (1<<RXEN0) | (1<<TXEN0)
            sts UCSR0B, R16

            ; Set frame format: 8data
            LDI R16,(1<<USBS0)| (1<<UCSZ01) | (1<<UCSZ00)   ; 8-bit data;
no parity,
            sts UCSR0C, R16
      ; 2 stop bits

filter_init:
      .EQU H0 = 3
      .EQU H1 = 13
      .EQU H2 = 29      ; the 7 constants calculated through the fdatool
provided by MATLAB
      .EQU H3 = 37
      .EQU H4 = 29
      .EQU H5 = 13
      .EQU H6 = 3

      ; temporary registers for storing the results
      #define RC R19
      #define AC1 R17
      #define AC0 R18

      ; for the window of 7 values
      LDI R21, 0
      LDI R27, 0
      LDI R26, 0
      LDI R25, 0
      LDI R24, 0                  ; temp variables required for updating
      LDI R23, 0
      LDI R22, 0
main:
      RCALL USART_RECEIVE                ; Receiving the value and
storing into R16
      RCALL START                        ; calling start function for
filtering the current input
      RCALL USART_TRANSMIT          ; transmitting the current filtered
value
      RJMP main                          ; repeating for next values


START:
      LDI AC1, 0
      LDI AC0, 0
      ;h[0]*x[7]
      LDI RC, H0
      MOV R22, R16            ; Copying the current received value into
R22 for multplying to the zeroth coefficient
      MULS RC, R22            ; signed MULtiplication stored in R0 R1
which is updated to AC0 and AC1 along with carry for AC1
      ADD AC0, R0
      ADC AC1, R1

      ;h[0]*x[7]+h[1]*x[6]
      LDI RC, H1
      MULS RC, R23
      ADD AC0, R0
      ADC AC1, R1

      ;h[0]*x[7]+h[1]*x[6]+h[2]*x[5]
      LDI RC, H2
      MULS RC, R24
      ADD AC0, R0
      ADC AC1, R1

      ;h[0]*x[7]+h[1]*x[6]+h[2]*x[5]+h[3]*x[4]
      LDI RC, H3
      MULS RC, R25
      ADD AC0, R0
      ADC AC1, R1

      ;h[0]*x[7]+h[1]*x[6]+h[2]*x[5]+h[3]*x[4]+h[4]*x[3]
      LDI RC, H4
      MULS RC, R26
      ADD AC0, R0
      ADC AC1, R1

      ;h[0]*x[7]+h[1]*x[6]+h[2]*x[5]+h[3]*x[4]+h[4]*x[3]+h[5]*x[2]
      LDI RC, H5
      MULS RC, R27
      ADD AC0, R0
      ADC AC1, R1

      ;h[0]*x[7]+h[1]*x[6]+h[2]*x[5]+h[3]*x[4]+h[4]*x[3]+h[5]*x[2]+h[6]*x[
1]f
      LDI RC, H6
      MULS RC, R21
      ADD AC0, R0
      ADC AC1, R1

      LSL AC0                 ; AC1:AC0 is a 16 bit answer of two 8 bit
numbers where first bit is just the sign bit, in 16 bit only 1 sign bit.
LSL to extract MSB of AC0 in C
      ROL AC1                 ; ROL removes the sign bit and shifts all
bits joining the msb of ac0 at lsb of ac1
```

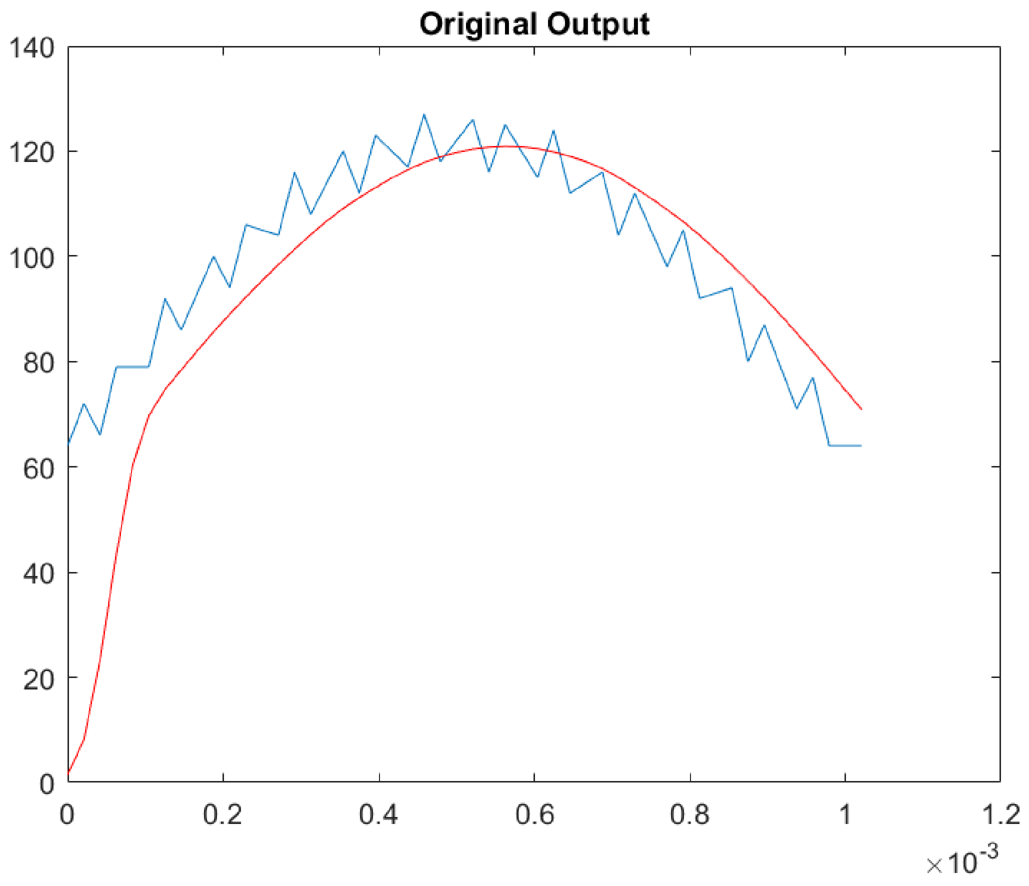
```
      MOV R21, R27
      MOV R27, R26
      MOV R26, R25      ; moving window as we want the new 7 values
      MOV R25, R24
      MOV R24, R23
      MOV R23, R22

      MOV R16, AC1      ; Copying the current result to R16 for
transmitting
      RET

USART_Transmit:
            ; Wait for empty transmit buffer
            LDS R20, UCSR0A
            SBRS R20, UDRE0
            RJMP USART_Transmit
            ; Put data (r16) into buffer, sends the data
            sts UDR0, R16
            RET

USART_RECEIVE:
            ; Wait for data to be received
            LDS R20, UCSR0A
            SBRS R20, RXC0
            RJMP USART_RECEIVE
            ; Get and return received data from buffer
            LDS R16, UDR0
            RET
```

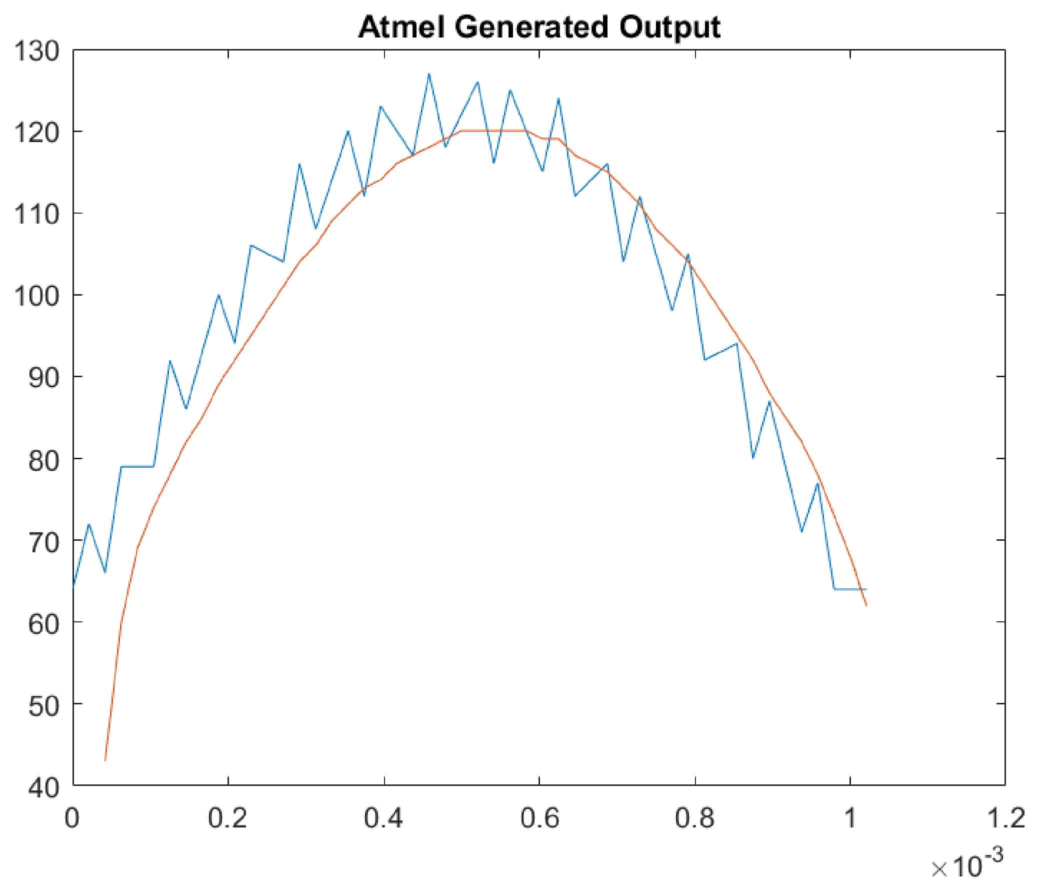
Matlab Code for generating the graphs:

```
clear;
fs = 48000;
T = 0.01;
ts = 1/fs;
t = 0:ts:T;
f1 = 500;
f2 = 18000;
% Input Signal
x = 1*sin(2*pi*f1*t)+0.1*sin(2*pi*f2*t);
newx=normalize(x,'range',[0,1]);
newx2 = ceil(127*newx);
figure(1);
plot(t(1:50),newx2(1:50));
hold
% Filter Output with fixed point filter arithmetic coefficients
quant_coeffs =
[0.0234375000000000,0.101562500000000,0.226562500000000,0.289062500000000,
0.226562500000000,0.101562500000000,0.0234375000000000];
y = filter(quant_coeffs,1,newx2);
plot(t(1:50),y(1:50),"r");
title("Original Output");
% Filtered Output from the Serial Communication with Filter Code
```

```
output= [43, 60, 69, 74, 78, 82, 85, 89, 92, 95, 98, 101, 104, 106, 109,
111, 113, 114, 116, 117, 118, 119, 120, 120, 120, 120, 120, 119, 119, 117,
116, 115, 113, 111, 108, 106, 104, 101, 98, 95, 92, 88, 85, 82, 78, 73,
68, 62];
figure(2);
plot(t(1:50),newx2(1:50));
hold
plot(t(3:50),output);
title("Atmel Generated Output");
```

Grpahs:




Atmel Generated Output
130
120
110
100
90
80
70
60
50
40
0
0.2
0.4
0.6
0.8
1
1.2
×10^-3