Jonathan W. Valvano

(6) Question 1. First, multiply by a power of 2 and then divide by that same amount. Shifting is much faster than multiplication and division.

z = (x - 3*y + z)/8

This cannot overflow, because the sum of the coefficients in this equation is 5, a 16-bit number times 5 will be at most a 19-bit number, which of course will always fit in a 32-bit temporary register. A 19-bit number shifted right three times will be at most a 16-bit number, so the result will fit back into the 16-bit variable. We must enable and disable interrupts to remove the read-modify-write critical section on x,y,z. It did mention other software was accessing x,y,z. You could also use StartCritical and EndCritical

```
short Filter(void){
  DisableInterrupts();
  z = (x - 3*y + z)>>3;
  EnableInterrupts();
  return z;
}
```

(6) Question 2. Consider the following SysTick ISR.

Part a) LR contains 0xFFFFFFF9 during the execution of the ISR, signifying an ISR is running.

Part b) R0,R1,R2,R3,R12,LR,PC, and PSW are pushed during the invocation of the ISR.

Part c) PD0 will go high every 50μs. PD0 will go low after executing about 6 to 10 assembly instructions. If you estimate each instruction takes 1 or 2 cycles, then the fall of PD0 will be less than 1μs after the rise.

(2) Question 3. The first point of the IEEE Code of Ethics is

to **accept responsibility** consistent with the **safety, health and welfare** of the public;

(4) Question 4. State two differences between ceramic and tantalum capacitors?

1) Ceramic capacitors are nonpolarized and tantalum capacitors are polarized.
2) Ceramic capacitors have lower leakage current than tantalum capacitors.
3) Ceramic capacitors can have lower tolerance than tantalum capacitors.
4) Ceramic capacitors work at high frequencies and tantalum capacitors work at lower frequencies.

(4) Question 5. Diameter is the distance across which the communication occurs. The Bee in ZigBee refers to the fact that this communication will hop across nearby nodes to reach its final destination. In order to increase diameter, a channel will receive weak inputs and rebroadcast the signal at higher amplitude, which is called a repeater. An analog communication channel loses SNR when repeated. However, a digital communication channel does not lose SNR when repeated. This means a digital signal can be received with no error, all bits are intact, then retransmitted at a higher amplitude. A digital communication channel has infinite diameter, while an analog channel will have finite diameter.

(4) Question 6. According to the Shannon Channel Capacity

$C = W \log_2 (1 + SNR)$ in bits/sec = 1 MHz $\log_2 (1001) \approx$ 10 Mbits/sec = 10^7 bits/sec

(4) Question 7. The slew rate of a digital signal defines the maximum frequency component in the wave (f=1/τ). Using wave theory we can estimate the wavelength, $\lambda = v/f$. If the wavelength is less than 4 times the length of the wire, then the channel behaves like a transmission line (reflections occur at impedance changes).

(10) Question 8.

Part a) Because ADC is sampled at 100 Hz, the sampled data contains frequency components from 0 to 50 Hz, according to the Nyquist Theorem.

Part b) Because the ADC clock is 125kHz, each ADC sample will take about 8 μs to convert. Because the UART FIFO is 16 elements deep, the UART busy-wait output will never wait. There are no other major software delays so the ISR should finish in about 10 μs.

(4) Question 9. We add **`volatile`** to tell the compiler that the value of this variable can change outside the direct consequence of any software currently being executed. We need it for global variables shared between threads and for I/O ports.

(10) Problem 10. Begin with what you must remember; the solution will require 12 states

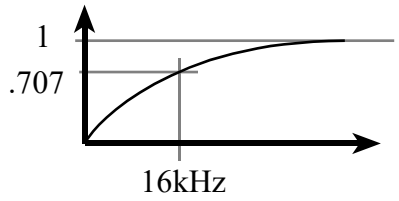
N even, M even, no switch
N even, M even, waiting for A switch to release
N even, M even, waiting for B switch to release
N odd, M even, no switch
N odd, M even, waiting for A switch to release
N odd, M even, waiting for B switch to release
N even, M odd, no switch
N even, M odd, waiting for A switch to release
N even, M odd, waiting for B switch to release
N odd, M odd, no switch
N odd, M odd, waiting for A switch to release
N odd, M odd, waiting for B switch to release

(10) Question 11. In the Laplace domain the impedance of the capacitor is Z=1/(Cs). s=jω, or s=j2πf. According to Ohm's Law, the output is a resistor divider.

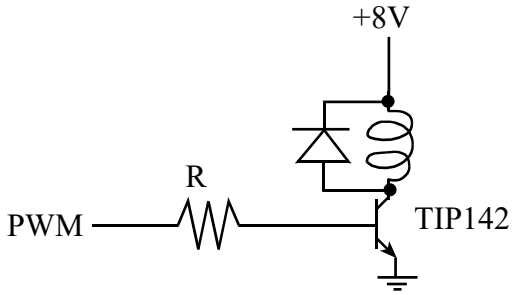
$Y(s) = R/(R+1/(Cs))*X(s)$

So $H(s) = R/(R+1/(Cs)) = RCs/(1+RCs)$

The frequency response is a high pass filter with a cutoff at R*C*2π*f equal to 1. f_c= 100kHz/2π ≈ 16kHz



(10) Question 12. This is a very poorly written datasheet; but it is all there was for the TIP142. The motor requires 1A and, the current gain is 1000, so we need 1mA of I_B. The data sheet doesn't give $V_{BE(on)}$ at I_C=1A, so we guess somewhere around 1V. V_{OH} near 3V means about 2V across the resistor. To get I_B an of 1mA, we need the resistor be less than or equal to 2kΩ. For a safety factor of 2 to 4, choose R to be 470Ω to 1kΩ. The diode is required to remove the back-EMF generated during the PWM switching.



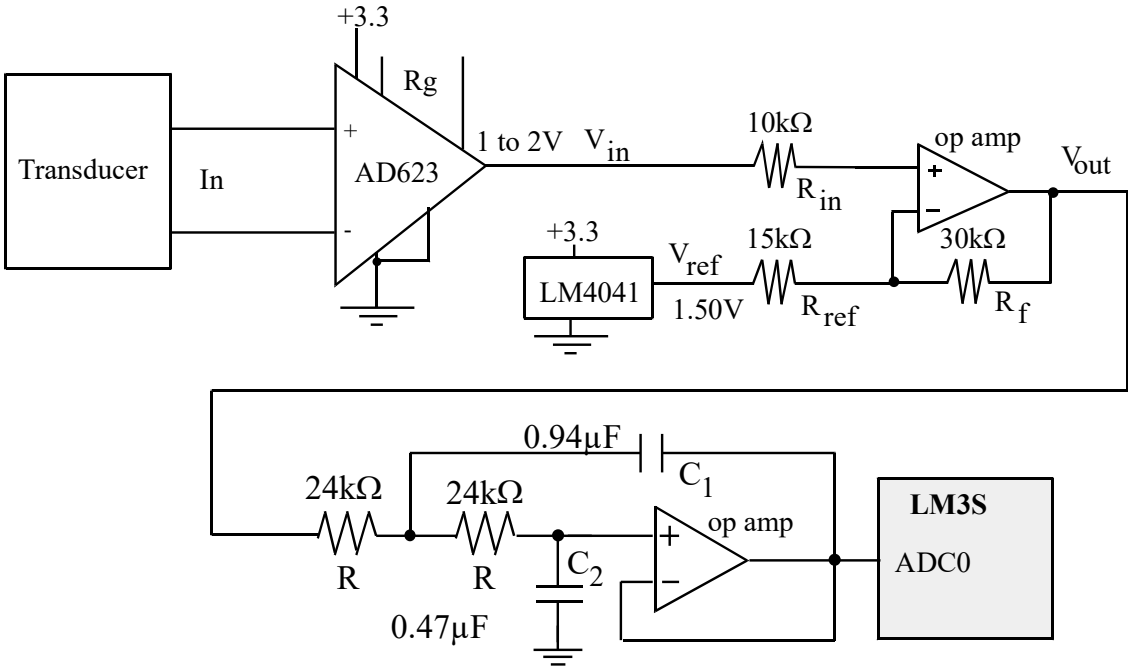
(10) Question 13. We could use an instrumentation amp because the transducer output is differential. The AD623 can be used because it has a gain equation of 1+100kΩ/Rg.

The desired transfer function is	$V_{out}=3V_{in}$ -3.
Use a reference of V_{ref}=1.5 V	$V_{out}=3V_{in}$ -2V_{ref}.
No ground gain needed	
Choose feedback resistor	R_f= 30kΩ
V_{in} gain is +3	R_1 = 10kΩ
V_{ref} gain is -2	R_{ref} = 15kΩ

We choose R_f/R_{in} to be 3, and R_f/R_{ref} to be 2.

I used the LPF equations from lpf.xls, which is the same as book section 8.3.2. Two poles were used because of the large noise at 60 Hz

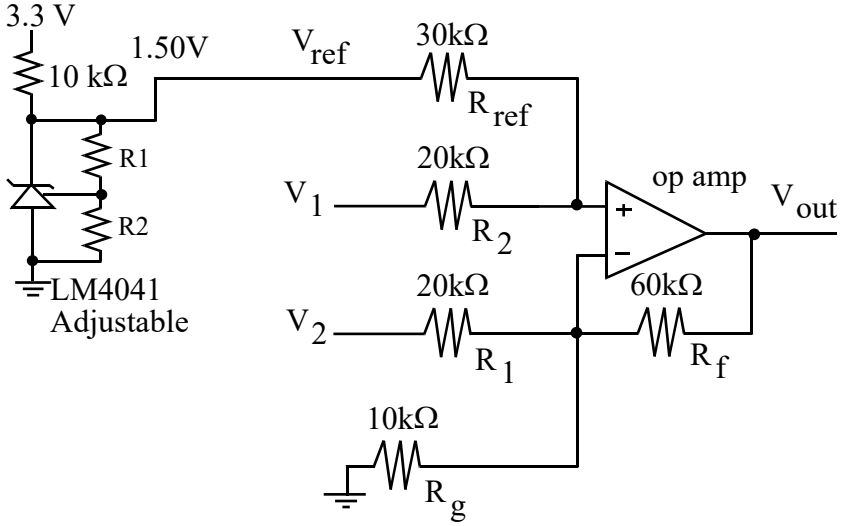
initial starting point		
fc (Hz)	1	
R (kohm)	10	
C1 (µF)	141.4	
C2 (µF)	70.7	
first design step is to select the cutoff		
fc (Hz)	10	fill this in
RA (kohm)	10	same as initial R
C1A (µF)	2.25045	is 141.4/(2•π•fc)
C2A (µF)	1.12523	is 70.7/(2•π•fc) or 0.5•C1A
second design step is to choose convenient Capacitor values		
fc (Hz)	10	same as previous fc
RB (kohm)	23.941	new value to match exact fc
C1B (µF)	0.94	fill this in
C2B (µF)	0.47	is 0.5•C1B
third design step is to choose a convenient resistor value		
fc (Hz)	9.9754	new cutoff based on these convenient values
RC (kohm)	24.000	fill this value in
C1C (µF)	0.94	same as C1B
C2C (µF)	0.47	same as C2B



Another answer (worse Z_{in} but satisfies other specifications) is to write one equation

The desired transfer function is	$V_{out}=3V_1-3V_2-3.$
Use a reference of V_{ref}=1.5 V	$V_{out}=3V_1-3V_2-2V_{ref}.$
Add ground gain to make sum of gains equal to one:	$V_{out}=3V_1-3V_2-2V_{ref}+3V_g.$
Common multiple of 2 and 3 is 6	$R_f=60k\Omega$
V_1 gain is +3	$R_1=20k\Omega$
V_2 gain is -3	$R_2=20k\Omega$
V_{ref} gain is -2	$R_{ref}=30k\Omega$
V_g gain is +3	$R_g=20k\Omega$

Since it says there is a large 60 Hz noise, it would be better to use a 2-pole filter (same as above)



There was no Question 14

(10) Question 15. Add semaphore synchronization. **`Count`** is the semaphore

```
unsigned long volatile Count=0;
```

```
void Timer0B_Handler(void){
  TIMER0_ICR_R = 0x0100; // ack
  DisableInterrupts();
  while(Count){
    PD0 ^= 0x01;
    Count--;
  }
  EnableInterrupts();
}
```

```
void Trigger(void){
// add code here

  DisableInterrupts();
  Count++;
  EnableInterrupts();
// other stuff
}
```

This solution uses two counters and does not have any critical sections. Technically this solution is not a semaphore. A semaphore is defined as a counter that is incremented in one thread and decremented in another thread. However, since it performs the same operation without disabling interrupts, it is superior. The following solution is not critical in the same way the **`FIFO_Put`** and **`FIFO_Get`** implementations in the book are not critical. In this solution like the book, one thread increments and the other thread just tests for equality.

```
unsigned long volatile PD0Count=0;  // incremented on each PD0 toggle
unsigned long volatile TriggerCount=0; // incremented in Trigger
```

```
void Timer0B_Handler(void){
  TIMER0_ICR_R = 0x0100; // ack
  while(TriggerCount != PD0Count){
    PD0 ^= 0x01;
    PD0Count++;
  }
}
```

```
void Trigger(void){
// add code here
  TriggerCount++;
// other stuff
}
```

(6) Problem 16. $V_{out} = V_0+2V_1+4V_2$