EECS 498-006 Practice Final Exam *Answers*

Fall 2011

Name: _____ unique name: _____

Sign the honor code:

I have neither given nor received aid on this exam nor observed anyone else doing so.

_____

Scores:

Section #	Points
I	/20
II	/40
III	/20
IV	/10
V	/10
Total	**/100**

NOTES:

1. Closed book and Closed notes
2. There are **10** pages total. Count them to be sure you have them all.
3. Calculators are allowed, but no PDAs, Portables, Cell phones, etc. Using a calculator to store notes is not allowed.
4. You have about 120 minutes for the exam.
5. **Be sure to show work and explain what you've done when asked to do so. That will be very significant in the grading of this exam.**

Answers and comments are in red. Note, this would likely be too long of an exam for a "real" exam, but it's a pretty good bit of practice!

I. Multiple choice/fill in the blank. 20 points. -2 per wrong or blank answer. Minimum zero.

1. A number of people and companies make hardware expansion modules that plug into the Arduino board. These expansions might be for wireless networking, Ethernet or motor controllers. They are generally called ***shields / ports / modules / cans***. (Slides 40-41/46, "Lecture 1: Class introduction, a bit on AVR and Arduino")

2. Say we have designed a four-layer board and we've made one of the two inner layers ground and the other power. Those two layers form a capacitor. You would expect that capacitor to have a ***greater / lesser / similar*** capacitance to capacitors you would typically have on a PCB to keep the power-ground noise as low as possible. You would also expect that capacitor so formed by the two planes would have a parasitic resistance to be ***greater than / less than / similar*** to a standard capacitor. Those two facts, when combined, mean that the capacitor formed by the power-ground planes will generally be best at reducing ***high-frequency noise / low-frequency noise / mid-range noise / all noise equally well independent of frequency***. (Slides 22/27, "Hardware/Software codesign issues, PCB, topic talks")

3. On a PCB, a "ground island" is _ ***section of the ground plane on a PCB that is separated from the rest of the ground plane*** __ while a via is a__ ***small, circular, conductive hole that allows signals in different layers to connect to each other.*** _ (via is in Pages 7-8/12 of "Lab 3: PCB design with Eagle: Eagle tutorial"; ground island is a bit questionable as a fair question)

4. A typical non-rechargeable (alkaline) AA battery can store about **20mAh / 200mAh / 2Ah / 20Ah** of charge. You would expect that the total amount of charge you can draw would ***go up / go down / stay the same*** as you increased the current drain. (typically 2.7Ah actually, Page 10/15 of "Battery" report).

5. ~~When creating a kernel module for Linux, you must always ***specify the module license /allocate memory using kmalloc / use kprint to log your operations***.~~
   Horrible question, sorry. See end of this exam for an explaination as to why.[i]

6. The primary advantage of a low drop out (LDO) power supply is that it is ***very power efficient / fairly simple / able to output a voltage higher than the input voltage***.
   Question should have been about "the advantage when compared to switching power supplies". See Power supplies talk.

7. One reason that a capacitive touch screen might be hard to use in the winter is that _***Rarely work with with gloves***_. (probably a bad/unfair question, not covered in the written part of the talk)

8. What does Shannon's limit describe?
   a. ***That computation itself always requires a bit of power and so there is a lower bound on how little power a given operation can take.***
   b. ***That the ratio of a countable infinity over an uncountable infinity is zero.***
   c. ***That over a given bandwidth-limited channel there is a maximum rate at which information can move.***
   (from slides 16-17/32 of "Wireless" lecture)

II. Short answer—40 points

1. Perhaps the first issue to resolve when designing an embedded system is figuring out what type of device you will use for your central processing unit. For each of the following scenarios, indicate if the task is best addressed by using an **FPGA**, having an **ASIC** built, or using a **microcontroller** and explain why. **[8 points, 4 each]**

   a. You are to make wireless sensors to monitor all the bridges in San Francisco. Each of the 1000 or so bridges will require an average of 10 sensors (some only 2, some 100). These wireless sensor nodes will generally be turned off unless a vibration sensor causes the device to wake up, in which case it will take a reading simple reading of the vibration sensor and then power back down. The devices have small solar cells which keep the batteries charged.

   This task is best addressed with a microcontroller. Microcontrollers are the cheapest of the three options above, and they have all of the features required - analog-digital converters or I2C/SPI for sensor interfacing and sleep modes - already built in to the system. Thus, we do not need the flexibility afforded by FPGAs, which are also typically more expensive and time-consuming to design on. While power is a concern, it is not a heavy enough constraint to consider designing an ASIC, since the solar panels will probably provide enough power for the intermittent wake-ups and readings

   b. You are to make a device which controls a scanning tunneling electron microscope (STEM) for a physics faculty member. The microscope generates data 100KB of data per second from a wide variety of non-standard sensors (including analog sensors, pulse-width based sensors, and a six or so I2C devices). The output consists of three SPI streams: two to a motor and one to a servo, as well as a USB connection to a PC where the data is sent.

   Since the project requires connecting many high-speed peripherals - more than most microcontrollers could handle - the task is best addressed with an FPGA. Since ASICs, which are typically reserved for projects with large markets that need decent performance and are highly power constrained (Slide 27/28 of "Lecture 2"), are prohibitively expensive for a one-off device (at least $10,000 from MOSIS - Slide 13/28) and we are not highly power constrained, an ASIC is not an ideal device for this scenario.

2. Why do CAN and USB both use bit stuffing but RS-232 and I2C buses don't? **[5 points]**

   CAN uses bit stuffing to prevent clock drift errors because the devices that use this protocol do not explicitly share a clock signal over the bus (Slides 21-26/41 in "CAN Presentation"). USB also uses bit-stuffing to mitigate clock drift (USB slides). From 373 we know that the I2C shares a clock signal over their bus wires (through SCL), so bit stuffing is not necessary to prevent clock drift errors. RS-232 does not require bit-stuffing because receivers are able to synchronize their clocks to the start and stop bit of each frame (that part is difficult, but you should know serial communication from lab so should be able to figure it out…)

3. Fixed point, rather than floating point, is often used in low-end microprocessors to handle fractional numbers. Say we were using a machine where integers were 16 bits and longs are 32 bits. And further, say we were using the signed integers to hold fixed-point values where the binary point was right after the most-significant digit. **[6 points, 3 each]**
   a. What is the range of representation for this fixed point value (be exact)?
      Question is a bit vague if 16, 32-bit or both are desired.
      16-bit is -1 to $1-(2^{-15})$. 32-bit is -1 to $1-(2^{-31})$.

   b. The following C function is used to multiply two 16-bit fixed-point numbers:

```
Int16 multiply(Int16 a, Int16 b)
{
      Int32 tmp=a*b;
      return((Int16) (tmp+0x4000)>>15);
}
```

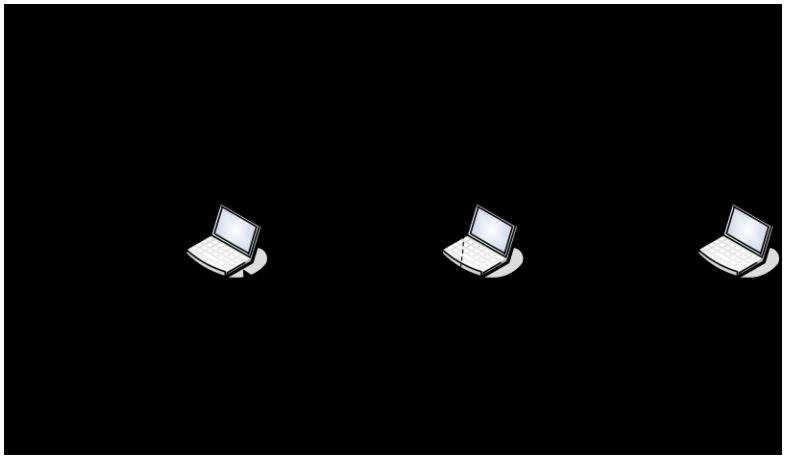
      Explain the purpose of the addition of 0x4000. Be sure to explain why that exact number is used.
      This was explained in class ("Fixed point" lecture). We are trying to round rather than truncate. Because the result is going from a 32-bit Q30 to a 16-bit Q15 and we wish to round, we need to add half of the range of representation of a Q15. That would be 2^{-16}. Since we are in Q30, the value we'd need to add is 2^{14}, which is 0x4000. (It's might be easier to think about this in terms of adding to the value that would be "1/2" as a Q15).

4. Serial buses often use differential signaling. What is the benefit of using differential signaling? Why does it have that benefit? **[5 points]**

   (The following answer is **much** more detailed than is expected; material was covered in the context of USB and CAN).

   The benefit of using differential signaling is its (1) tolerance to ground offsets, (2) noise immunity with low-voltage electronics, and (3) resistance to electromagnetic interference (1) is due to the receiver reading the difference between the two voltages without respect to ground. (2) gives twice the noise immunity as a single-ended system if one assumes that the noise on one wire is uncorrelated with the noise on the other wire, therefore requiring twice as much noise to cause an error in transmission. (3) only applies when the differential signals are transmitted over balanced lines; interference induced in one wire will be induced equally in the other wire, so the differential between the shifted voltage should remain roughly the same.

5. One problem with wireless communication is the “hidden terminal problem.” Illustrated below is a situation where two devices might want to communicate, but A and C are hidden from each other while B has no hidden terminals.



   a. Explain why this could cause problems for A’s communication with B. **[3 points]**
   Since nodes A and C are outside of each other's range (they are hidden from each other), the two might fail to detect a simultaneous transmission, resulting in a collision at B. Discussed in class, some found on slide 27/30 in "Wireless".

   b. Provide a scheme to address this problem. Be detailed and clear. **[5 points]**
   RTS-CTS (page 31 of “wireless”) can be used to deal with this problem. A station that wishes to send one or more packets must acquire the floor before transmitting a packet train to the receiver B. The floor is acquired by an RTS-CTS exchange in which the station sends an RTS Request-to-Send frame to the destination node and the destination node replies with a CTS Clear-to-Send frame. After the station has finished transmitting its data to the destination node, the destination node sends an ACK frame to signal the end of the transmission with that node and open the floor to other transmitting nodes.

6. 6. Define the term “frequency division multiplexing”. Draw a picture which illustrates the idea.
   [4 points]
   Frequency division multiplexing is a form of signal multiplexing which involves assigning non-overlapping frequency ranges to different signals (Page 3/19 in "LTE" report).

   Looking for a picture like the one on slide 17/41 in "LTE" presentation.

7. What is multi-factor authentication? List one advantage and one disadvantage of multi-factor authentication **[4 points]**
    a. Definition:
    Multi-factor authentication is a combination of the two or three different types of authentication: things that you have, things that make up who you are, and things that you know (Page 8/10 in "Authentication" report).

    b. Advantage:
    The main advantage of multi-factor authentication is that a failure of a single factor doesn't let a malicious user gain access (Page 8/10 in "Authentication" report).

    c. Disadvantage:
    Multi-factor authentication is more difficult for the user (more things to lose/forget and more likelyhood of false negative), but is still susceptible to man-in-the-middle attacks, which could compromise and steal all of the authentication factors (Page 9/10 in "Authentication" report).

8. What is the definition of a “hard” deadline for a real-time system? A soft deadline? Give an example of both. **[4 points]**
    A hard deadline is a deadline that must be met. Missing a hard deadline results in system failure. (Slide 5/28 in "RTOS") An example of a hard deadline is the power control for a cutting tool, which could destroy the work piece if the RTOS doesn't turn off the machine in time

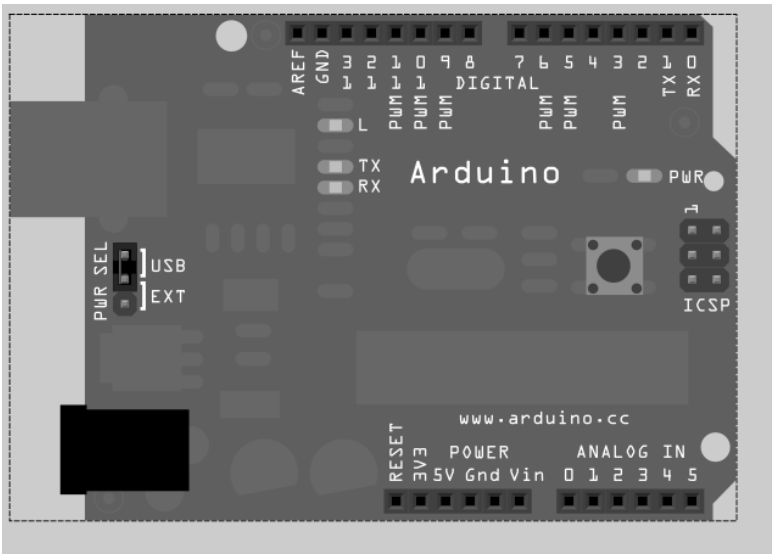
    If you have a soft deadline, the usefulness of a result degrades after its deadline, thereby degrading the system's quality of service. (Slide 5/28 in "RTOS"). An example of a soft deadline is the direction notification in a GPS navigation system, as even somewhat late notices about a turn can still be useful. (Page 1/17 in "RTOS Report").

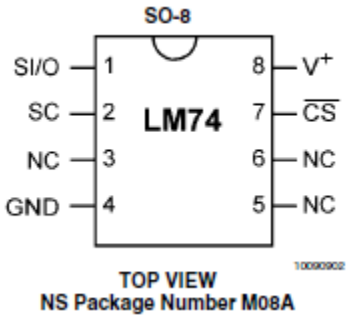
III. Working with parts—20 points

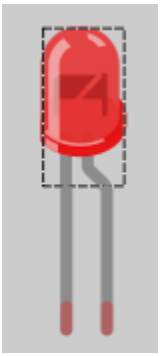
Consider the LM74 temperature sensor (on a real exam the data sheet would be attached. It's at http://www.national.com/ds/LM/LM74.pdf). For this question, you will show how to have an Arduino light single LED light if and only if the LM74 sensor is reading a value above 25 degrees centigrade.

A simple Arduino sketch, to help you with syntax and the like, is on the bottom on this page.

1. Indicate, by drawing wires, how you will connect the three







components. **[3 points]**

Sample sketch:

```
int ledPin = 13;                // LED connected to digital pin 13

void setup()                    // run once, when the sketch starts
{
  pinMode(ledPin, OUTPUT);      // sets the digital pin as output
}
void loop()                     // run over and over again
{
  digitalWrite(ledPin, HIGH);   // sets the LED on
  delay(1000);                  // waits for a second
  digitalWrite(ledPin, LOW);    // sets the LED off
  delay(1000);                  // waits for a second
}
```

* Since we may not use the SPI library functions, we can choose arbitrary digital pins on the Arduino for SC, SI, CS, and the LED. Therefore, let CS be digital pin 2, SI be digital pin 3, SC be digital pin 4, and LED be digital pin 5.
* NC, which stands for "not connected", can be left unconnected.
* V+ must be connected to the 5V pin and GND to the Gnd pin.
* We will make our LED active low by connecting the cathode of the LED to the digital pin on the Arduino. This has the beneficial property that current is sourced directly from VCC, not the microcontroller. (Further discussion here: http://www.w9xt.com/page_microdesign_pt4_drive_led.html). While the Arduino can source up to 40mA per pin, it can only source a total of 200mA for all pins. (Page 313/440 in the "Atmega328" datasheet)
    * We ***should*** also include a resistor between the anode of the LED and 5V to limit the current. (Say 220 ohms so that (5-1.7)/220 = 15mA is the limit).

We wouldn't take off points for A) not making it active low or B) not including the resistor, though you would certainly have challenges without the resistor...

2. Write the Arduino sketch that will accomplish the desired task. It may not use library functions other than those shown in the sample sketch. **[17 points]**

- **Note the following:**
  - The device requires a positive clock polarity; data is clocked out on the falling edge of the serial clock and clocked in on the rising edge of the clock (Pages 5-6,8/16 of "LM74").
  - The device provides a two's complement 13-bit ADC value (Page 9/16 of "LM74").
  - The devices marks 25°C at 0,0001,1001,0000 (Page 7/16 of "LM74").
  - Integers are 16 bits on Arduino ("int" http://www.arduino.cc/en/Reference/Int).
  - A complete transmit/receive communication will consist of 32 serial clocks. The first 16 clocks comprise the transmit phase of communication, while the second 16 clocks are the receive phase (Page 9,11/16 of "LM74")
  - One complete temperature conversion period will have to pass before the LM74 temperature register will contain the new temperature data. Until then, it will contain "stale" temperature (the data that was in the register before going into shutdown mode) (Page 9/16 of "LM74")
  - After completion of the first full temperature conversion, the register will contain tempreature measurement data in bits D15 (the temperature data MSB) through D3 (the temperature data LSB). Bit D2 will be fixed high; bits D1 and D0 are undefined. (Page 9/16)
  - The LM74 power-up default condition is continuous conversion mode. (Page 8/16)

Code is on the next page.

```
#define CS   2
#define SI   3
#define SCLK 4
#define LED 5

void setup() {
        // Set I/O directionality
        pinMode(SI, INPUT);
        pinMode(SCLK, OUTPUT);
        pinMode(CS, OUTPUT);
        pinMode(LED, OUTPUT);

        // Set pins initial state
        digitalWrite(SI, LOW);
        digitalWrite(SCLK, LOW);
        digitalWrite(CS, HIGH);
        digitalWrite(LED, HIGH);
}

unsigned char spi_read() {
        // Read 8 bits from SPI
        int i;
        unsigned char b = 0;
        for (i = 7; i >= 0; i++) {
                // Read value after falling edge of CS or SCLK (Page 11/16)
                // The microsecond delays should create a roughly 500KHz clock with period 2us. In reality, this
will be much slower due to the digitalWrites but could be achieved with direct PORT manipulation. SCLK period
minimum is .16us (Page 5/16)
                delayMicroseconds(1);
                b |= digitalRead(SI)<<i;
                digitalWrite(SCLK, HIGH);
                delayMicroseconds(1);
                digitalWrite(SCLK, LOW);
        }
        return b;
}

float get_temp() {
        // Return floating point temperature from LM74
        int spi_val = 0;

        // Initiate transaction by setting chip select low
        digitalWrite(CS, LOW);
        spi_val |= spi_read() << 8;
        spi_val |= spi_read();
        digitalWrite(CS, HIGH);

        // Remove the high-bit and TRI-STATE bits
        // Right arithmetic shift will preserve the sign bit
        spi_val = spi_val >> 3;

        // Linearize the transfer function with LSB equal to 0.0625°. Loses accuracy beyond +-25°C.
        // A more complete implementation would follow the nonlinear transfer function on (Page 7/16).
        return (float)spi_val * .0625;
}


void loop() {
        // Discard the first stale temperature (Page 9/16 of "LM74")
        float temp;
        temp = get_temp();

        // Loop forever, collecting temperatures and toggling the LED
        while (1) {
                temp = get_temp();
                // Toggle the LED depending on the temperature reading
                if (temp > 25.0) {
                        digitalWrite(LED, LOW); // Turn LED on
                }
                else {
                        digitalWrite(LED, HIGH); // Turn LED off
                }
        }
}
```

IV. Power – 10 points

1. You are designing a sensor network node that is to run off of two AA batteries for as long as possible. You are trying to chose between two processors "SlowOne" and "FastOne" The sensor node needs to take data once every 10 seconds. Each time data is to be taken the following must occur:
   - In parallel the processor and sensor must power up. The sensor takes 0.1 ms to power up.
   - The processor then asks the sensor to gather data. This request requires the execution of 100 instructions.
   - The sensor then takes 1.0 ms to gather the data.
   - Once the sensor is done gathering data the processor then executes 1900 instructions.
   - Finally both the processor and sensor are powered down (this takes no time).

   The processors have the following characteristics and you may assume the rest of the system (power supply etc.) consumes no significant power.

	SlowOne	**FastOne**
MIPS	1	2
Power when active*	1mW	1.2mW
Power when sleeping	10nW	20nW
Time to wake up	0.15ms	0.10ms

* "active" includes anytime the processor is on including wakeup.

a. Which processor choice will cause the system to consume less average power? Clearly show your work. **[8 points]**

The **SlowOne** processor is active .15ms + (2000 Instructions/1 MIPS)*1000) +1ms =3.15ms during the sensor data phase and sleeps 10,000-3.15=9996.85ms for the rest of the 10 second frame.

Therefore, the average power consumption of the device is
(1mW*3.15ms + 1e-5mW *9996.85ms)/(10000ms) = 325nW average power.

The **FastOne** processor is active (.1ms + 2000 Instructions/2MIPS*1000) +1ms =2.1ms during the sensor data phase and sleeps 10,000-2.1=9997.9ms for the rest of the 10 second frame.

Therefore, the average power consumption of the device is
(1.2mW*2.1ms + 2e-5mW*9997.9ms)/(10000ms) = 272nW average power.

* Therefore, the **FastOne** processor will cause the system to consume less average power.

b. Given the above scenario what could you do to further reduce average power consumption while still gathering the same data? **[2 points]**

Most obvious thing would be to go back to sleep during the 1ms data is being taken. It would drop power consumption by a fair bit.

V. Essay – 10 points

Drawing on class material and your own experiences with your class project, discuss fundamental principles and difficulties of hardware/software co-design.

No answer provided.

[i] Part I, Q5. (Thanks to Jeremy for the detailed notes!)

Specifying the module license is only necessary if you plan to redistribute the Linux kernel with your module compiled into the kernel. Modules that do not include the MODULE_LICENSE macro are presumed not free; they will load just fine into the kernel, but they will display a warning message that the kernel has been tainted with a proprietary driver. ("How does MODULE_LICENSE work in the Linux kernel?" http://askville.amazon.com/MODULE_LICENSE-work-Linux-kernel/AnswerViewer.do?requestId=2639289)

(Page 9/79 of "The Linux Kernel Module Programming Guide", http://tldp.org/LDP/lkmpg/2.6/lkmpg.pdf)

(Slide 5/32 of "Linux Device Drivers") and (Page 3/10 in "Lab 4: Linux Interrupts") mention module licensing…

kmalloc is necessary for dynamic memory allocation in the kernel. Automatic memory allocation on the stack through lexically-scoped variables works the same way as it does in a userspace program and is handled at compile-time. One cannot use malloc for dynamic memory allocation in a kernel module, so in that sense, kmalloc is necessary for allocating memory.

I meant printk, not kprint. printk sends messages to the log buffer (the /proc/kmsg file); klogd then reads these messages and typically sends them to the syslogd daemon, which appends the messages to the end of /var/log/messages. One cannot use printf to log messages in Linux kernel modules, so in that sense, printk is necessary for logging your operations in Linux. ("Chapter 18: Debugging" http://www.makelinux.co.il/books/lkd2/ch18lev1sec3)