EECS 498 Midterm Exam--KEY

Fall 2012

Name: _____KEY_____ unique name: _____

Sign the honor code:

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

_____

Scores:

Problem #	Points
1	/15
2	/10
3	/25
4	/15
5	/35
Total	**/100**

NOTES:

1. Closed book and Closed notes
2. There are **9** 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 80 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.**

1) Short answer **[15 points]**
   a) Briefly explain what busybox is and why embedded Linux distributions tend to use it. **[5]**

Busybox is a single executable that is capable of handling the basic functionality of a wide range of standard Linux tools. This includes sh, ls, nm, etc. Softlinks are used so that "ls" actually points to busybox and it uses argv[0] to figure out what program it is emulating.

Being a single executable it saves space and is also easy to maintain.

   b) Loadable Kernel Modules
      - Why might one wish to write a kernel module for an embedded system rather than writing code in user space? **[5]**

In general writing KMs is harder than writing in userspace. The only main reason to do this is because you need access to something that the kernel doesn't export or shouldn't export (many memory-mapped devices). Dealing with multiple processing trying to use the same resource might also be a good reason.

While this question didn't ask this, in general you should avoid writing KMs. If you make a mistake you can crash the whole system and you can easily introduce security bugs.

      - What does the "loadable" part of LKM mean? Why is this useful? **[5]**

Loadable means that the module can be added to the kernel while the kernel is running. This is commonly used because you might suddenly need a new bit of kernel functionality (say a new device was plugged into the machine) and you want to be able to support it. You can't have all devices supported all the time—it would take too much memory.

2) Consider the following scenarios. Indicate which of the following device you would recommend to be used to address the problem? Briefly explain you answer. Assume these are your only choices. **[10 points]**

(i) Off-the-shelf Arduino (16KB Flash, 1KB SRAM, 512 bytes EEPROM, 10 MIPS) with the same form factor as used in labs 1 and 2. $30.
(ii) Off-the-shelf MSP430 board (64KB Flash, 256KB SRAM, 10 MIPS) with the same form factor as the LaunchPad used in lab 2. $10
(iii) Custom PCB with a MSP430 chip (64KB Flash, 256KB SRAM, 10 MIPS)
(iv) Custom PCB with a MSP430 chip (1MB Flash, 1MB SRAM, 20 MIPS)
(v) Custom ASIC (arbitrary specifications)

a) A faculty member wants a device for a science demo he will be taking around to high schools that raises and lowers a conductor's baton based upon the volume of noise in the room. He has a total budget of $300 including labor and parts for the embedded system (the servo and baton aren't a part of this budget). **[5]**

(i) is the best answer. His needs are minimal, cost (and thus development time) is really limited, and the functionality he wants is supported by Arduino. Plus their might be a shield for the audio (if needed). (ii) is also a reasonable answer due to cost, but you'd expect development costs to eclipse the cost savings in the board.

b) A local company has an idea for a child's toy. It has 10 LEDs, 8 buttons and a speaker. It needs to play music on the speaker and flash certain lights when buttons are pressed in a given order. (If other music/lights have already been triggered it will stop the old music/lights and start the new). The music can fit in 128KB and the rest of the program/data required seems fairly small (say 8KB at most). They have a target sales number of 100,000 units with a retail price of $30. The part must be small and light (needs to fit in a 1 inch cubic volume). Beyond that cost is the first priority and power consumption is important to them. **[5]**

Lots of I/O, size requirements and memory requirements probably eliminate i and ii. iii) won't work due to the need for 128Kbytes of non-volatile memory for the music (otherwise the first time power is lost the device will break). v) is almost certainly too expensive. The development cost is probably large (you're going to need a CPU, who's going to create it?) and the cost to spin the chip is probably way too much of the $3M total this thing will bring in (retail remember). That leaves us with iv. Getting it to fit into a 1" cube could be tough, but should be doable.

3) Consider an embedded application which consists of 3 tasks named A, B, and C. Each task is CPU bound (that is instruction execution is the only reason tasks take time) and periodic. Each task's deadline is when the next instance of the task is ready to start. These tasks have the properties and requirements shown in the table below. You are to assume there is *no overhead* of any type (including scheduling overhead) and that this machine runs any given instruction in exactly the same amount of time. **[25 points]**

Task	Maximum number instructions executed by a single instance of the task	How often the task needs to run
A	6 Million	90ms
B	14 Million	120ms
C	4 Million	300ms

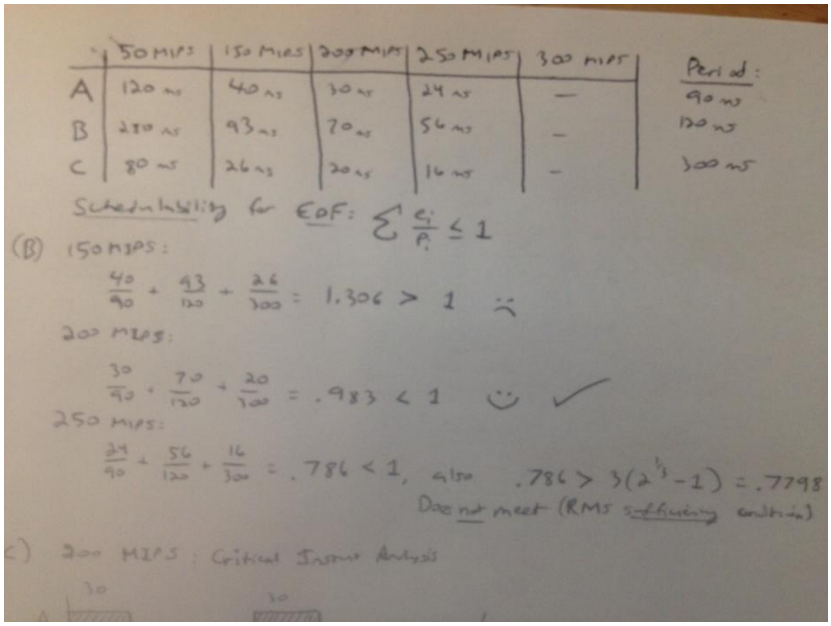
a) Which task do you give the highest priority under Rate Monotonic scheduling? The lowest? **[2]**

- Highest ___A_____
- Lowest ___C_____

b) You are choosing between 4 different processors each with a different MIPS rating (listed below). Which of those 4 has the lowest MIPS rating that could schedule these tasks using ***EDF***? You must *clearly* explain your work to get any credit. **[6]**

(1) 150 MIPS
(2) 200 MIPS
(3) 250 MIPS
(4) 300 MIPS

Quick math says the processor utilization is around 98% at 200 MIPS. So given there is no other overhead, EDF should be able to schedule this. (In reality I'd worry about that overhead!)

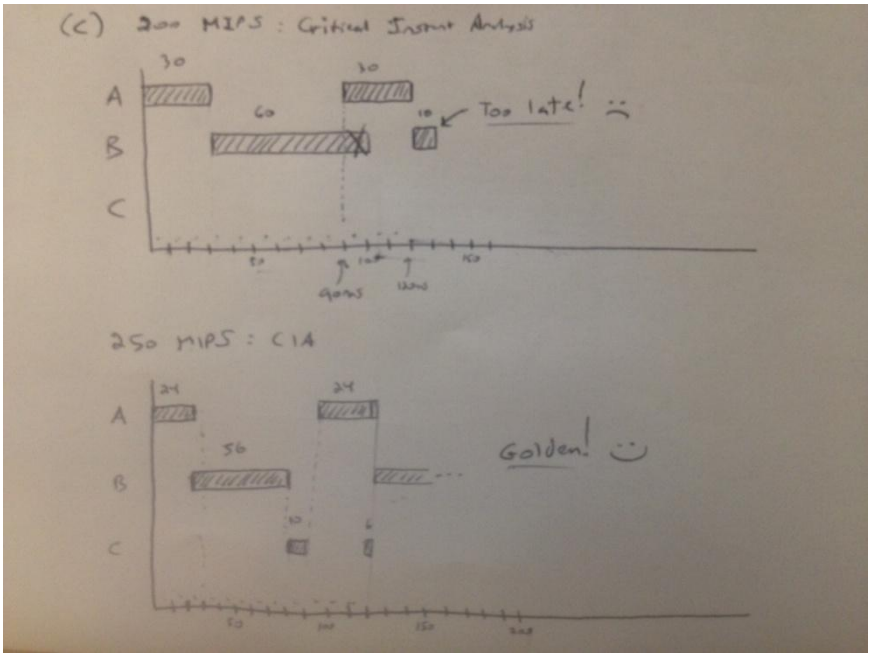


<Continued on next page>

Task	Maximum number instructions executed by a single instance of the task	How often the task needs to run
A	6 Million	90ms
B	14 Million	120ms
C	4 Million	300ms

c) You are choosing between 4 different processors each with a different MIPS rating (listed below). Which of those 4 has the lowest MIPS rating that could schedule these tasks using **RM scheduling**? You must clearly explain your work to get any credit. **[10]**

(1) 150 MIPS
(2) 200 MIPS
(3) 250 MIPS
(4) 300 MIPS



Two steps here. First we need to show that it doesn't work for 200 and then that it does for 250. At 200 MIPS A will take 30ms, B 70ms, and C 20ms. Doing the critical instance analysis we can see that B will only get to run for 60ms before it is interrupted by the second instance of A. By the time A finishes (@90ms) B will have missed it's deadline.

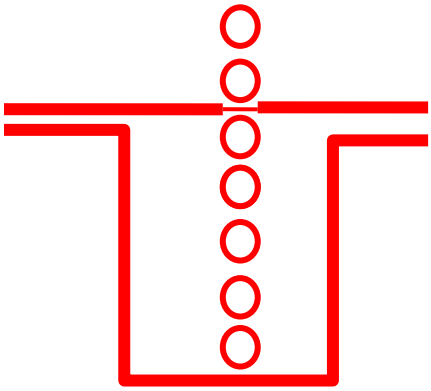
For 250MIPS we end up with A taking 24ms, B taking 56ms and C taking 16ms. The first instance of B will now finish at time 80ms, well before A starts. C will get 10ms of runtime between 80ms and 90ms. A will then run until 114ms. C will get 6ms (running until 120ms) of CPU time, letting C finish. Thus, it is schedulable as all the initial tasks have completed and met their deadlines.

d) Using the FreeRTOS system, write the task creation functions you would call to schedule this application? (Parts of the FreeRTOS documentation are attached). The task functions take no arguments and are named A(), B(), and C(). We are **just** asking you to create the tasks, not write the rest of the main or the task functions themselves! **[7]**

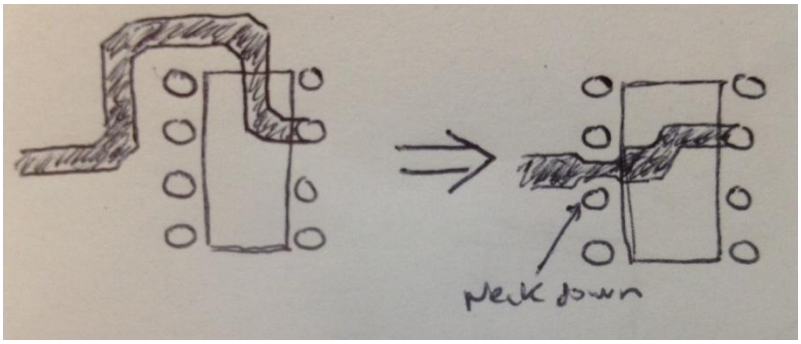
```
xTaskCreate(A, "A", STACK_SIZE, NULL, 3, NULL); // Highest priority
xTaskCreate(B, "B", STACK_SIZE, NULL, 2, NULL);
xTaskCreate(C, "C", STACK_SIZE, NULL, 1, NULL);
```

4) Say you've done a PCB design at work. There is one power trace that is highly sensitive to additional resistance (it is the power line for an FPGA that has strict voltage requirements). Your boss says that "Your trace is too long. You'll get a lower resistance if you neck down the trace and shorten it." **[15 points]**

a) Sketch and label a diagram that illustrates the situation (including why your trace was so long to begin with) and how your boss wants you to fix the problem **[5]**



The circles represent thru-hole slots or vias. And rather than routing around them (lower line) she wants us to route through them. Another (better? Picture is on the right).



b) Say that the original trace was 5 cm long and 30mil wide and that after your follow her suggestion, the new trace is 2 cm long and 30 mil wide for all but 5 mm of those 2 cm where it is 6 mil wide. Would you expect the new or old trace would have a lower resistance? Show and explain your work. **[5]**

Original scheme as resistance proportional to 5cm/30mil. Second scheme is proportional to 1.5cm/30mil + .5cm/6mil. The new scheme has the lower resistance.

c) Assuming her solution does reduce the resistance and still meets the PCB design rules; identify the biggest problem that should none-the-less concern you with this change. **[5]**

The narrower wire will be able to carry less current. In the worst case it could act like a fuse and melt! It would take a fair bit of current to do that, but given it is a power line…

5) Consider the TMP37 temperature sensor (much of the datasheet is supplied). For this question, you will A) answer a few questions about the device and then B) show how to have an Arduino light single LED light if and only if the sensor is reading a value above 25 degrees centigrade. **Use 5V for the device's power.** A simple Arduino sketch and definitions of some Arduino functions are also attached as the last page of this exam. Please feel free to rip them out of the exam if you desire. **[35 points]**

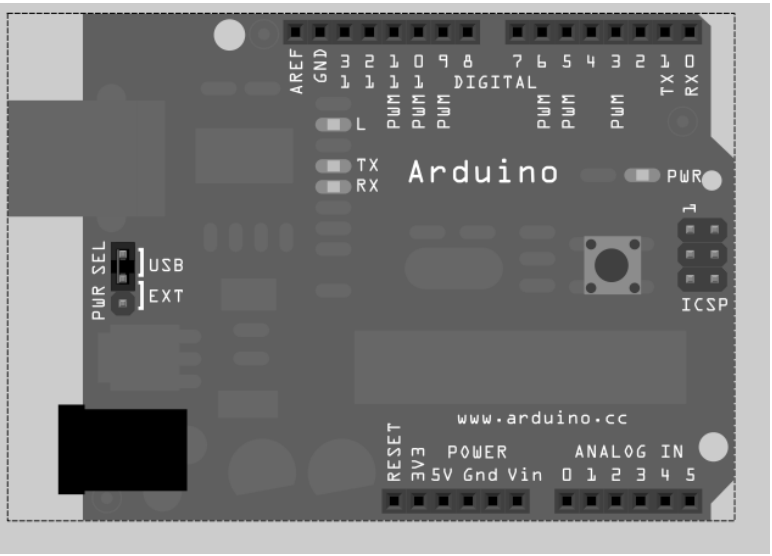
a) Assuming Vout is connected directly to the Arduino (which has a large input resistance), how much *power* will this device typically use at room temperature? **[3]**

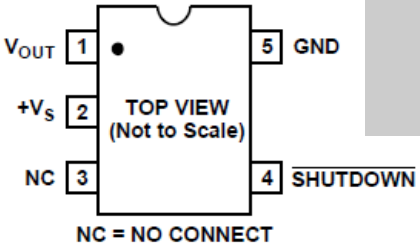
Vout = 5V, Average/typical current=25uA (Figure 12: Supply Current vs. Supply Voltage)
P = V * I = 5V * 25uA = 125uW

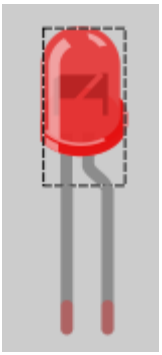
b) Say you wish to use less power, what would you expect the best average power draw you could achieve if you only wanted to take one sample per second? Assume you are still at room temperature. Also assume while the device is powering up or powering down it draws power as if it were not in low-power mode. Show and explain your work. **[5]**

Extreme cases:
Always sleeping = 5V * .01uA = 50nW
Always on = 5V * 25uA = 125uW

It is unclear exactly what the minimum % of time the device could be left on per second.
Here's one answer assuming 300us is the min.
((1s – (275 + 25 us))*(5V)*(.01uA) + (275+25us)*(5V)*(25uA)) / 1s = 87.5 nW

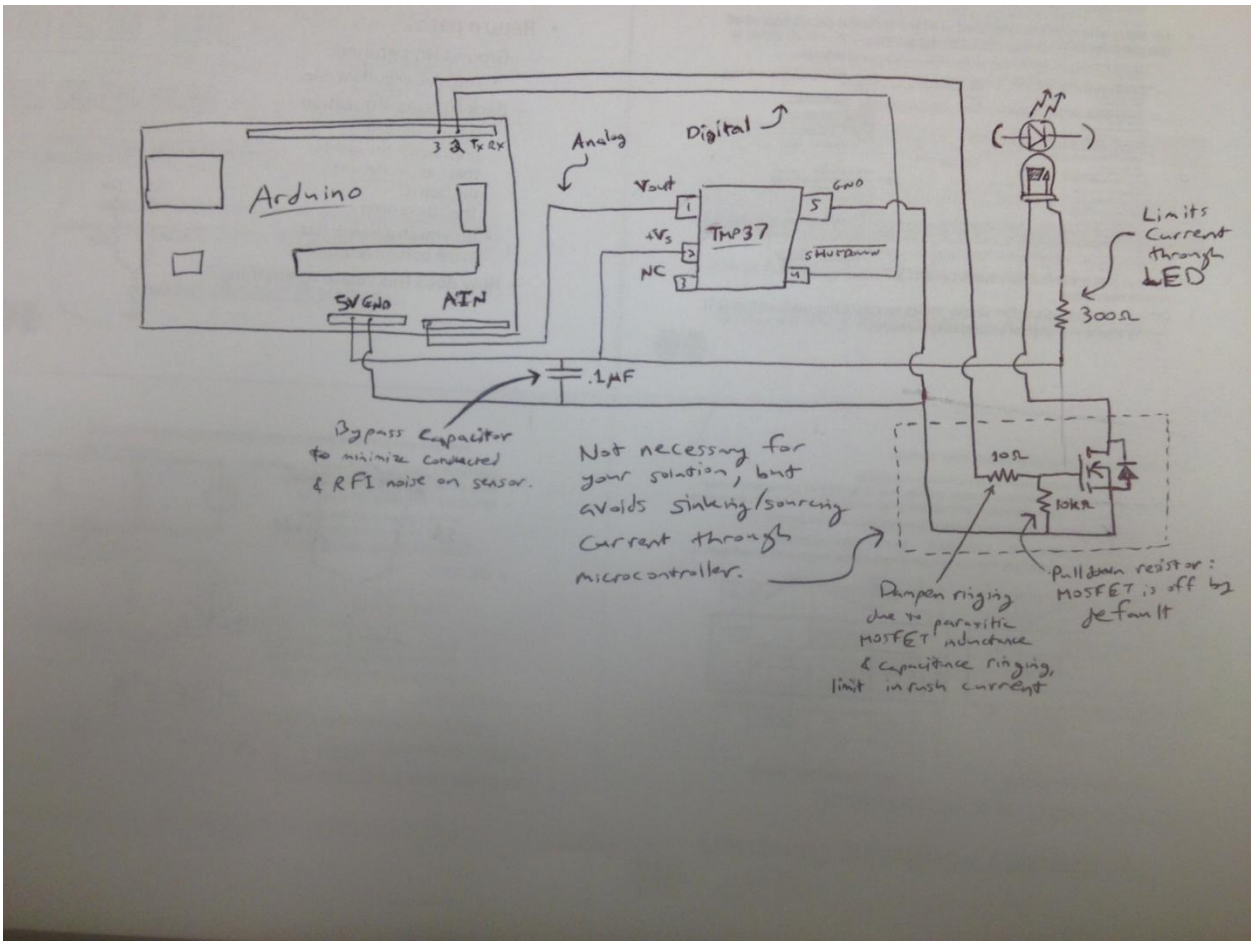






c) Indicate, by drawing wires, how you will connect the three components and draw any other components needed. Assume you will want to shutdown the device when it's not in use. **[5]**

This answer is a lot more complex than we were looking for. We mainly wanted just the pin connections and, ideally, the bypass cap. The resistor with the LED is a heck of a good idea also.



d) Write an Arduino sketch that lights the LED if the temperature sensor reads a value above 25 degrees centigrade. If you can't quite do that due to quantization[1] issues it is required that the light be on when the temperature if 25 degrees or above and it is desired that the light not come on at any lower temperature than necessary to meet that requirement. You should sample the sensor about once a second and should power down the sensor when it is not being used. Assume your hardware is connected as you've shown above. **[22]**

```
void setup() {
      pinMode(4, OUTPUT); // ~SHUTDOWN pin
      pinMode(3, OUTPUT);   // LED pin
}

void loop() {
      delay(999);           // Delay for ~1 second
      digitalWrite(4,1);    // Wake up TMP37
      delayMicroseconds(50);// Figure 15 shows Vout response time at
                            // 25C to be 25-50us
      if (analogRead(A0) > 103) } { // If temperature is above 25C:
                                    // (.5 * 1024 / 5 = 103)
            digitalWrite(3,1); // Turn LED on
      }
      else {
            digitalWrite(3,0); // Turn LED off
      }
      digitalWrite(4,0); // Shutdown TMP37
}
```

[1] That is, you may have some rounding issues that make detecting exactly 25 degrees exactly. If that occurs, you must insure that when you get to 25 degrees the LED comes on, even if that means you may turn the light on at temperatures below 25 degees.

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
}
```

analogReference(type)

Description
Configures the reference voltage used for analog input (i.e. the value used as the top of the input range). The options are:

- DEFAULT: the default analog reference of 5 volts (on 5V Arduino boards) or 3.3 volts (on 3.3V Arduino boards)
- INTERNAL: an built-in reference, equal to 1.1 volts on the ATmega168 or ATmega328 and 2.56 volts on the ATmega8 (*not available on the Arduino Mega*)
- EXTERNAL: the voltage applied to the AREF pin (**0 to 5V only**) is used as the reference.

Parameters
type: which type of reference to use (DEFAULT, INTERNAL, INTERNAL1V1, INTERNAL2V56, or EXTERNAL).

Returns
None.

analogRead(pin)

Description
Reads the value from the specified analog pin. The Arduino board contains a 6 channel (8 channels on the Mini and Nano, 16 on the Mega), 10-bit analog to digital converter. This means that it will map input voltages between 0 and 5 volts into integer values between 0 and 1023. This yields a resolution between readings of: 5 volts / 1024 units or, .0049 volts (4.9 mV) per unit. The input range and resolution can be changed using analogReference().

Syntax
analogRead(pin)

Parameters
pin: the number of the analog input pin to read from (0 to 5 on most boards, 0 to 7 on the Mini and Nano, 0 to 15 on the Mega)

Returns
int (0 to 1023)