EECS 473 Midterm Exam -- KEY

Fall 2022

Name: _____**KEY**_____ unique name: _____

Sign the honor code:

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

_____

NOTES:

1. Closed book and Closed notes
2. There are **13** pages total for the exam as well as handouts which you will need for the last question.
3. Calculators are allowed, but no PDAs, Portables, Cell phones, etc. Using a calculator to store notes is not allowed nor is a calculator with any type of wireless capability.
4. You have about 120 minutes for the exam.
5. Though the last question is worth significantly less than half the points, we expect it will take at least half of the exam time.

Be sure to show work and explain what you've done when asked to do so. That may be very significant in the grading of this exam.

1) **<u>Circle</u> the letter in front of all the true statements.** [9 points, -2 per wrong circle/lack of a circle, minimum 0]

a) It is very likely that a fully-charged 100Ah lead-acid battery will run out of energy before 10 hours have passed when drained at a rate of 5A.

b) Compared to the PCB power/ground plane, a bypass capacitor typically has a higher capacitance, ESR, and ESL.

c) Alkaline batteries are a common type of primary-cell batteries while lithium-polymer batteries are a common type of secondary-cell batteries.

d) On a PCB, a 20 mil-wide trace has more resistance than a 10mm-wide trace of the same length.

e) When placing multiple capacitors across the same power and ground pins of a given device, you want to put the capacitors as close to the pins as possible and put the smaller-value capacitors closer to those pins than the larger-value ones.

f) In general, LiPo batteries are better suited for a task than Alkaline batteries when the batteries are likely to sit unused for months at a time.

g) The primary purpose of the silkscreen layer on a PCB is to prevent solder from adhering to unnecessary parts of the PCB when components are mounted on it.

h) One advantage of Rate Monotonic scheduling over Earliest Deadline First scheduling is that RMS doesn't require preemption.

i) Deferred interrupts are generally used so that low-priority tasks can be done outside of the ISR.

j) Jitter is variability in response (or release) time for a given operation and it is generally associated with delays in interrupt response times.

2) **Multiple choice—Circle the best answer.** [8 points, -2.5 per wrong/blank answer, min 0]

a) Which of the following is an advantage of RM scheduling over EDF scheduling?

- ***RMS can schedule certain sets of periodic tasks EDF cannot.***
- ***RMS doesn't require dynamic priorities for periodic tasks, while EDF generally does.***
- ***When RMS fails to schedule, it will always fail to schedule the task with the largest CPU needs, while EDF could fail to schedule other tasks.***
- ***RMS handles deferred interrupts properly (as long as priority inheritance is enabled) while EDF often does not.***

b) Which of the following statements about linear regulators is FALSE?

- ***A 5V output linear regulator with a 10V input would be expected to waste less power than that same regulator with a 12V input.***
- ***An LDO is a type of linear regulator.***
- ***A linear regulator is generally expected to have a less-noisy output than a switching regulator.***
- ***A real linear regulator with a 10V input and 5V output would be expected to waste no more than 50% of the input power.***

c) The term "rat's nest" is best understood to refer to what with respect to PCBs?

- ***The layout when most/all connections are still "air wires".***
- ***The schematic once the connections have been made.***
- ***The (potentially excessive) use of "neck downs" to reduce PCB trace resistance.***
- ***The use of an excessive number of vias on a PCB.***

d) Which of the following statements about the GPL is FALSE?

- ***Things which use the GPL are open source and in the public domain.***
- ***The GPL is sometimes referred to as being a "viral license" because if you use code licensed under the GPL you likely need to license that code under the GPL.***
- ***Linux and gcc are both licensed under the GPL.***
- ***Among other things, GPLv3 addresses issues involving software patents.***

3) Scheduling [10 points]

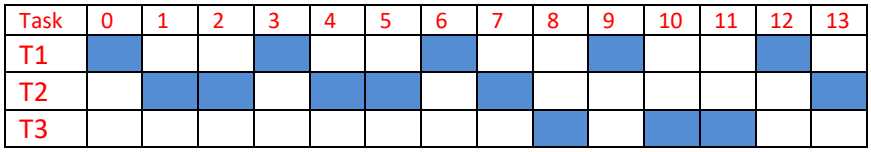
Say you have the following groups of tasks. For each group find the CPU utilization and identify which groups are RM and which are EDF schedulable. Indicate if you needed to do the critical instant analysis. *If needed, **clearly** show that analysis.* The following equation may prove useful.

$$\sum_{i=1}^{n} U \leq n(2^{1/n}-1)$$

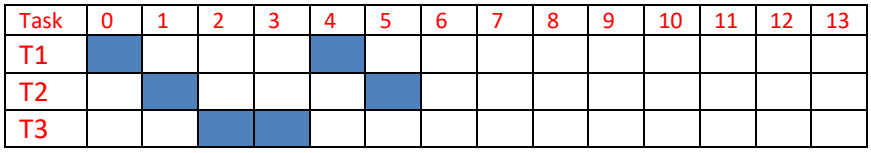
Group	T1 Execution Time	T1 Period	T2 Execution Time	T2 Period	T3 Execution Time	T3 Period	% Utilization
A	1	3	2	4	1	5	103%
B	1	3	5	12	3	14	96%
C	3	6	3	10	--	--	80%
D	1	4	1	5	2	6	78%

Group	EDF Schedulable?	RM Schedulable?	Did you need to examine the critical instance?
A	No	No	No
B	Yes	Yes	Yes
C	Yes	Yes	No
D	Yes	Yes	Yes

B:



T3 makes its deadline.



T3 makes its deadline.

4) **Linux device drivers** [8 points]

Consider the following code found as the write function member of the file_operations struct for a Linux kernel module. It is associated with the device file "/dev/txx2" (so a write of the file /dev/txx2 will result in this function being called). Assume that everything is set up appropriately beforehand. Ignore the fact that copy_from_user's return value is being ignored (it's "just" a warning…).

```
char s[10] = "ABCDEFGHI";  // Global
ssize_t memory_write( struct file *filp,
      char *buf, size_t count, loff_t *f_pos) {
      char *tmp;

      int loc = count>5?5:count
      tmp=buf+count-loc;
      copy_from_user(s,tmp,loc);
      printk("<1> %s\n",s); //prints the string s to the log file
      return loc;
}
```

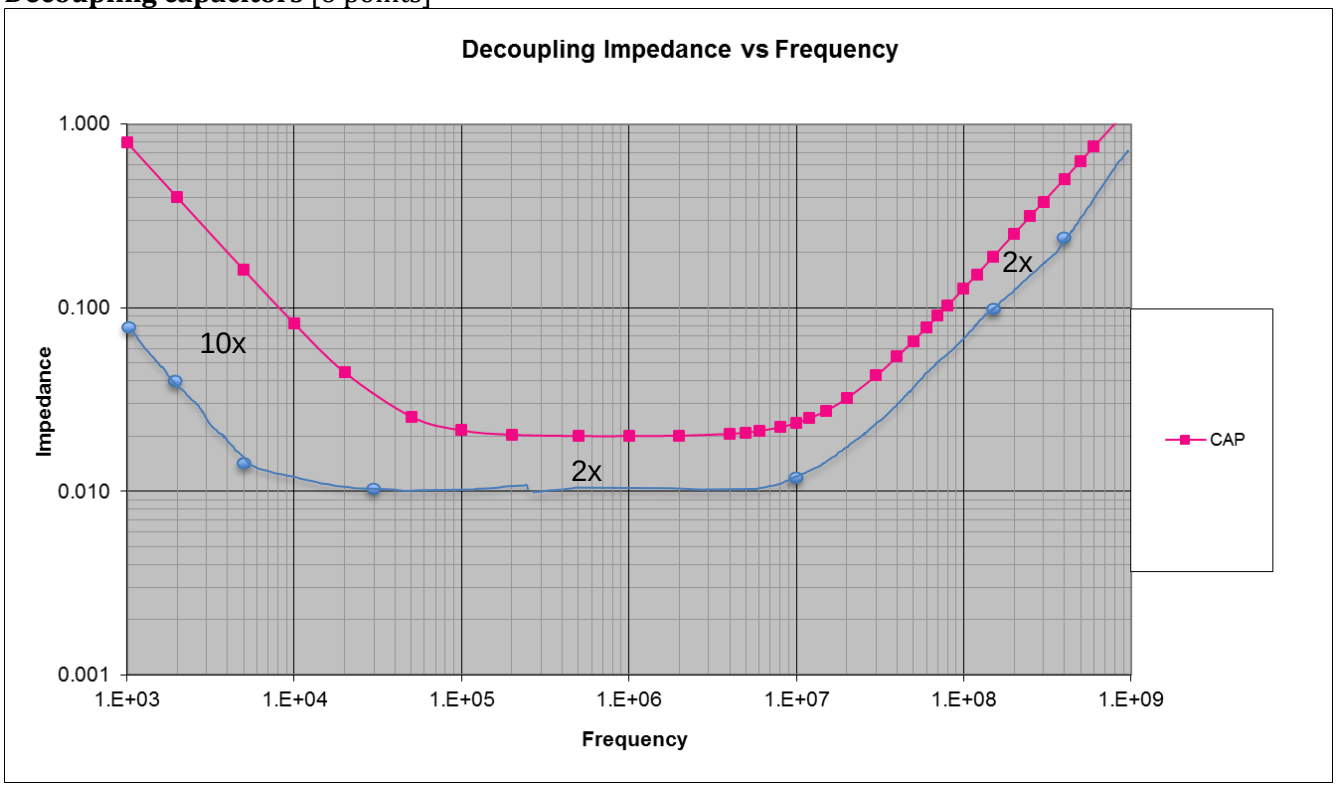
a) Say that the first thing someone does after loading the module is to write the string "WXYZ" to /dev/txx2. What will appear in the log file? **[3]**

<1> WXYZEFGHI

b) Say that the first thing someone does after loading the module is to write the string "LMNOPQRS" to /dev/txx2. What will appear in the log file? **[5]**

<1> OPQRSFGHI
<1> QRSRSFGHI

5) **Decoupling capacitors** [6 points]

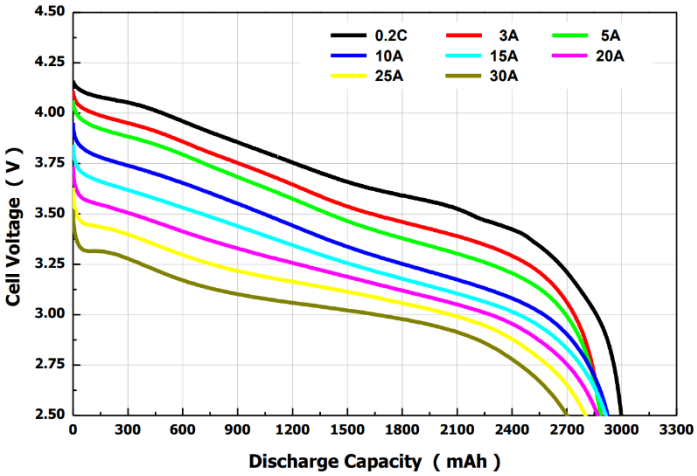


The above graph shows the frequency vs. impedance for a given capacitor. Redraw the graph showing the same information but if we instead put used 2 new capacitors (in parallel) which each had the same ESR and ESL but five times the capacitance.

Lines aren't great. Dots are there to help the drawing. The line should be shifted down by 10x in the first part and 2x in the later parts.

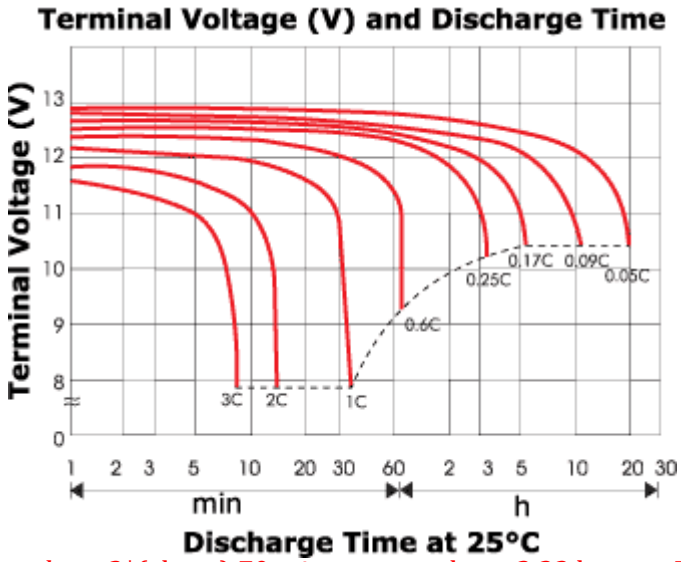
6) **Batteries** [10 points]

Consider the following battery discharge curves from a real battery specifications. ***Clearly justify your answers.***



a) *About* how long would you expect the battery on the left to be able to drive 12 Amps while maintaining at least 3.25V output? **[4]**

12A @3.25 is around 1600 or 1700mAh effective capacity. So 1.7Ah/12A=0.14 hours or about 8.5 minutes.



b) *About* how long could a 750mAh battery of the chemistry to the left drive a 50Ω load that requires at least 11 Volts? **[6]**

V=IR; I=V/R; 11V/50Ω=0.22A. 220mA/750mA gives a draw rate of C=0.29. Per the chart .25C would last ~3 hours @11V. So this will be no more than .25/.29*3 hours or about 2.5h, probably a bit less (because the draw rate is greater than .25C). We could aslo look at .6C and conclude that it's going to last at *least* 2*(about) 70 minutes or at least 2.33 hours. Thus, answers between 2.3 and 2.6 hours are pretty reasonable. If we had to pick a number, 2.45 or so is probably where we'd guess.

7) **Linear regulators** [5 points]
You have a linear regulator with an 8V input, a 6V output, and a quiescent current of 15mA. If the load being driven by the regulator is a constant 50 Ohms, how much power is wasted by the regulator? You must *clearly* show your work.

Current_load =6V/50Ω=0.12A.
Power_used=6V*.12A=720mW. (Or $6V^2/50\Omega$ if you prefer).
Power_input=8V*(.12A+.015A)=1080mW.
Power_wasted=1080mW-720mW=**360mW**

8) **Bit banging** [7 points]
We had to use "bit banging" to control the LCD in Lab 2.
a) What is "bit banging" and what is the alternative? **[5]**

Bit banging is when you use software to directly set and read GPIO pins to communicate to a device. The alternative is to use a dedicated hardware interface that supports whatever protocol the device uses.

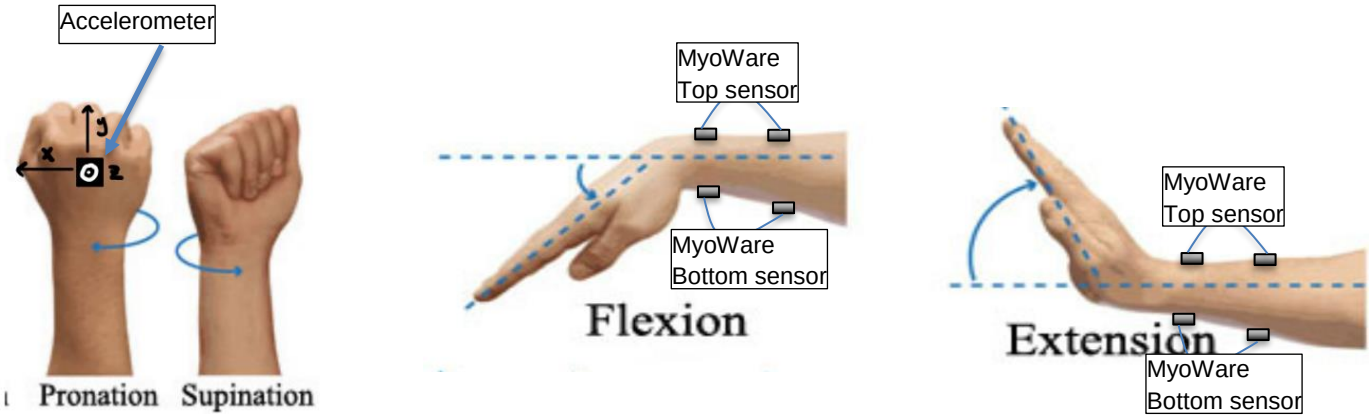
b) Do you think the Arduino implementation of the LCD interface on the Uno uses bit banging? Very briefly justify your answer. **[2]**

Yes, there is no dedicated hardware on the Atmel chip used by the Arduino to control that supports the LCD's protocol.

Comments: This was fairly hard to grade. A lot of people seemed to think that if you were calling a library that meant you weren't bit banging. Of course the library might well be using that. Also, a lot of people appeared to think that only serial buses (I2C, SPI, UART, etc.) were possible to handle in any way other than bit banging. And that of course isn't true.

9) **Design Problem** [37 Points]

Starting in the Winter of 2023, the Department of Biomedical Engineering is interested in developing a robotic wrist joint that is able to sense wrist position and flex of a human wrist and mirror those characteristics onto a simple robotic wrist joint. Their plan is to use one low-power High-G 3-axis accelerometer to sense the wrist rotation as well as two MyoWare muscle sensors placed on either side of the forearm to sense and determine wrist flex. A layout of this device is shown below.



The department has requested that you make a prototype of this device. This means assembling the hardware as well as writing the associated firmware. The device has the following components.

1. One Arduino Uno board
2. One Lithium Ion battery with an output voltage of 5V
3. One **LM3940** 5V to 3.3V LDO
4. One **H3LIS331**, a low-power High-G 3-axis accelerometer. This sensor will be placed on the back of the patient's hand as shown above and will be communicated to by using I2C.
5. Two **MyoWare 3** lead electromyography sensors that will be placed on either side of the patient's forearm as shown above. The connection to their forearm has been done for you.
6. Two **SG90** hobby servo motors. One of these servos will be used for the rotation of the wrist joint, the other will be used for the flex of the joint.
7. Any other passives you may require

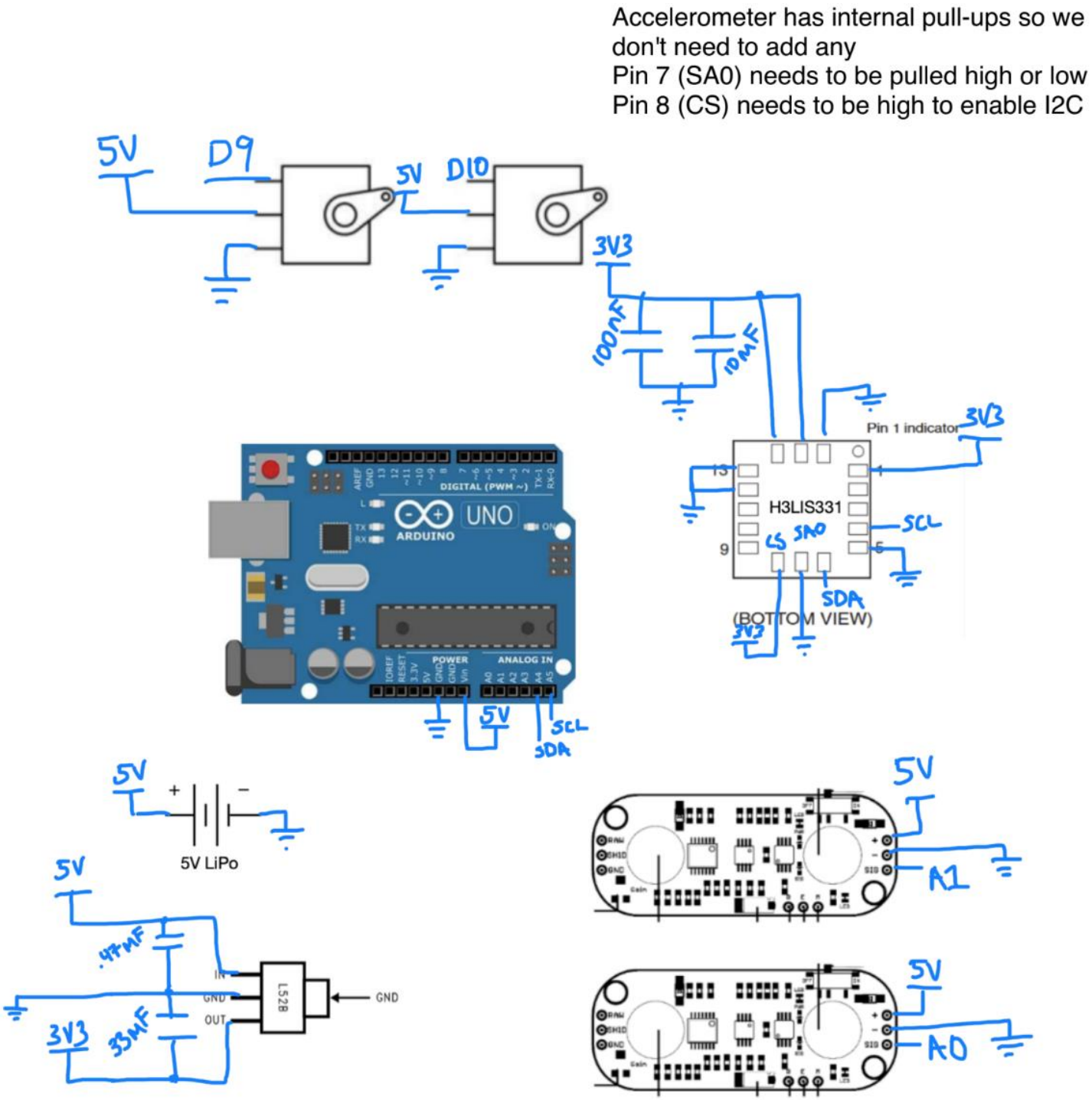
The system should have the following characteristics:

1. The battery and LDO should provide power to the necessary parts of the system.
2. The Arduino should monitor the accelerometer to measure the rotation of the human wrist and map that rotation to the equivalent on the wrist-joint servo motor.
    - Only the accelerometer is to be used to control the wrist-joint servo.
3. The Arduino should monitor the muscle potential across both sides of the forearm and map that rotation to the equivalent on wrist-flexion servo motor.
    - Only the MyoWare sensors will be used to control the wrist flexion servo. Implement an algorithm that uses both to control the flexion servo.

Biomedical knowledge and assumptions:

- The top sensor will detect extension, while the bottom sensor will detect flexion.
    - You can assume that there is a linear relationship between the sensor output and the angle, where 90 degrees of flex creates a maximum output voltage from the sensor.
- The pronation and supination will never be greater than 90 degrees from level.

Part A: Wiring [10 points]
Provide the connections between the different components of the system. You should also provide power and GND to all components. Add resistors and capacitors as needed. You may use labels to make connections.



Part B: Interface Design [6 points]
Write two functions that will let you write and read registers on the H3LIS331. They are to use the i2c protocol.

```
//I2C device address is 0x18 or 0x19 depending on pin setting

void h3_write(uint8_t addr, unit16_t value) {
     Wire.beginTransmission(0x18); //device address, can also be 0x19
     Wire.write(addr);             // send subaddress first
     Wire.write(value & 0xFF);     // write low byte
     Wire.write((value >> 8) & 0xFF); //write high byte
     Wire.endTransmission();









}

int h3_read(uint8_t addr) {
     Wire.beginTransmission(0x18);
     Wire.write(addr);
     Wire.endTransmission();

     //dual byte return
     Wire.requestfrom(ox18,2);
     while (Wire.available()<=2);{};
     int result = Wire.read();
     result = result<<8;
     result |= Wire.read();
     return result;









}
```

Part C: Setup [6 points]
Write a setup function that will initialize any inputs, outputs, and sensors you may need. Define any variables that you may need.

```
// Choose 2 analog pins for top and bottom Myoware sensors from Analog pins  A0-A5
#include <Servo.h>

Servo flex_servo, rot_servo;

void setup() {
      // Control register address 0x20 PM2(bit7)-PM0(bit5)) bits must be non-zero to take the
      // device out of power down mode.
      // Also, either the z(bit2) or x(bit 0) axis must enabled
      Wire.begin();
      h3_write(0x20, 0b00100001)    // or 0x21, 0x61 …

      //Initialize the servo pins when using the servo library
      flex_servo.attach(9);   // One is 9, the other 10.
      rot_servo.attach(10);

}
```

Part D: Loop [15 points]
Write the loop function for the system. This function should monitor the accelerometer as well as the muscle potential values from the MyoWare sensors and map those readings to the servo motors. Define any variables or helper functions you may need.

```
#define TOP_MYO A0
#define BOT_MYO A1
#define FLEXION_SERVO 3
#define ROTATION_SERVO 5
#define OUT_X_L 0x28


void loop() {
      uint16_t top_angle = (analogRead(TOP_MYO) * 90) / 1024;
      uint16_t bot_angle = (analogRead(BOT_MYO) * 90) / 1024;

      // sensor scaling depends on how control register is set up for
      // full scale. Assumptions that were stated were usually marked
      // correct. This solution is assuming acceleration is full scale
      // when in line with gravitational acceleration which is not
      // necessarily what is possible with this sensor. Please submit a
      // regrade request if you believe your scaling is correct.

      // Also only assuming wrist_rot servo is only moving between when
      // x is parallel and perpendicular to gravity (90°)

      int16_t accel_angle = ((int32_t)h3_read(OUT_X_L) * 90) / 65536;

      if (top_angle > bot_angle) {
            flex_servo.write(90 + top_angle);
      }
      else {
            flex_servo.write(90 - bot_angle);
      }

      rot_servo.write(accel_angle);

      delay(20);
}
```