EECS 473 Final Exam

Fall 2021

Name: _____ unique name: _____

Sign the honor code:

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

_____

NOTES:

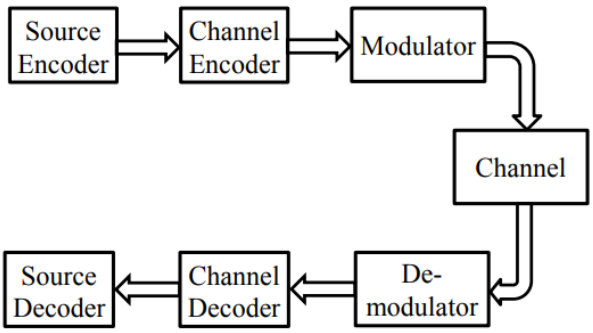
1. Closed book and Closed notes
2. **Do not write anything you want graded on the back pages of the exam.**
3. There are **15** pages total for the exam. There is also a references handout.
4. Calculators are allowed, but no PDAs, Portables, Cell phones, etc. Using a calculator to store notes is not allowed nor is a calculator with any type of wireless capability.
5. You have about 120 minutes for the exam.
6. Be sure you answer each question on the appropriate page.
7. When doing calculations, you must show your work to get credit.

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.

dBm	mW
-3	0.5
-2	0.6
-1	0.8
0	1.0
1	1.3
2	1.6
3	2.0
4	2.5
5	3.2
6	4
7	5
8	6

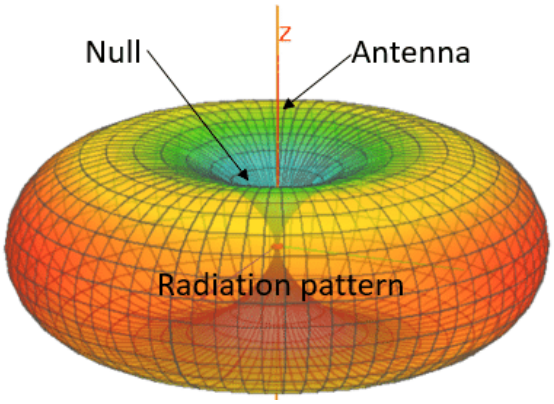
dBm	mW
9	8
10	10
11	13
12	16
13	20
14	25
15	32
16	40
17	50
18	63
19	79
20	100

dBm	mW
21	126
22	158
23	200
24	250
25	316
26	398
27	500
28	630
29	800
30	1000
33	2000
36	4000



$$C = B\log_2\left(1+\frac{S}{N}\right) \qquad r = \frac{10^{(p_t + g_t + g_r - p_r)/20}}{41.88 \times f} \qquad \sum_{i=1}^{n} U \le n(2^{1/n}-1)$$

1. Multiple choice. Circle the correct answer or fill in the blank.
[12 points, -2 per wrong or blank, minimum 0]



a) The figure to the right is the radiation pattern of a(n) ***isotropic / dipole / dish*** antenna.

b) What does Shannon's limit describe?
- ***That you can only move so much current through a fixed-diameter wire.***
- ***That communication over a noisy, bandwidth limited, channel has an absolute limit on the amount of information that can be communicated in a given amount of time.***
- ***The faster you discharge a battery, the less total useful energy that can be extracted.***
- ***That given enough time, all software will be licensed under a viral license such as the GPL***

c) On a PCB running low-power traces, most low-cost PCB manufacturers typically require a clearance between traces of *about* ***1 / 5 / 25 / 100*** thous.

d) Using RM scheduling, you can schedule your tasks for certain, no matter how many tasks you have, if your CPU utilization is **100% / 82% / 88% / 75% / 70%** or lower (pick the largest that is true).

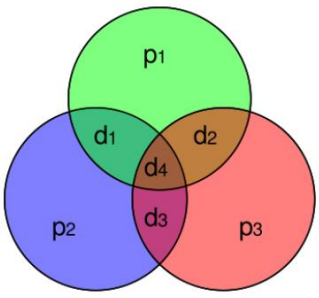
e) The range of representation of a signed 4-bit Q3 number is from _____ to _____

f) In the context of communication theory, a "source encoder"
- ***compresses a message***
- ***adds guard bits to a message***
- ***adds error correction to a message***
- ***modulates the signal***

g) The Peukert Effect describes the phenomenon that as we draw more current from a battery
- ***the battery gets hotter.***
- ***the actual capacity of the battery drops.***
- ***the battery is more and more likely to be unable to be recharged.***
- ***the cathode of the battery is more likely to get coated by the electrolyte***

h) Assuming we are encoding our data as d[1:4]p[1:3], when using the scheme shown XXXX, we would treat the packet 0001001 as having

sent what 4 bits of data? _____



2. Consider the Shannon-Hartley theorem as stated here: **[10 points]**

$$C = B \log_2 \left(1 + \frac{S}{N}\right)$$

   a) Circle each of the following that are true. **[6, -2 per wrong answer min 0]**
      - **C** represents the *channel capacity,* that is the maximum rate that of bits of information plus error correction that can be reliably sent.
      - **B** represents the *bandwidth* available. So if the data were sent using only the frequencies between 2.35 GHz and 2.4 GHz, the bandwidth would be 50 MHz.
      - **S** represents the *signal* measured in units of power.
      - **N** represents the *noise* measured in units of voltage.
      - This formula relies on some assumptions. One such assumption is that all noise is Gaussian.
      - This formula implies that with no noise of any type, the channel capacity would be infinite.

   b) If the SNR is -20 dB and the bandwidth available is 1 MHz, what is the channel capacity? **[4]**

3. Say we have a 7.2V 2.0Ah battery. We are using it through a regulator which outputs 5V to a processor that uses 20mW. Nothing other than the processor is driven by the regulator or battery. You must clearly show your work to get credit. **[10 points]**

   a) How long would you expect the battery to last if the regulator were an otherwise ideal LDO that has a quiescent current of 1mA? **[5]**

   b) How long would you expect the battery to last if the regulator were an otherwise ideal buck converter with a quiescent current of 2mA? **[5]**

4. Write a fixed-point function which takes two 8-bit Q4 numbers and returns the product as a 16-bit Q4 rounding to the nearest value (ties may round either way) if needed. You may assume there is no overflow. Also assume that ints are 32-bits, shorts are 16 bits and chars are 8 bits. We have provided the start of the function for you. **[7 points]**

```
short Qmult(char a, char b)
{
```

5. Short answer **[7 points]**:
    a) If you have a file named /dev/gpio23 in your Linux directory, provide a one-line Linux command which you could reasonably expect will cause pin 23 to go high. **[3]**

    b) When designing a 2-layer PCB, it is common ground-fill one side and power-fill the other. Describe what that means and *briefly* explain why we do it. **[4]**

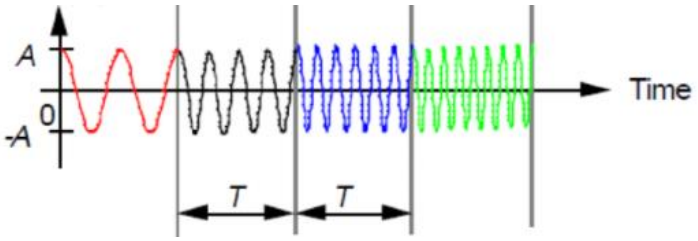
6. Wireless communication **[10 points]**

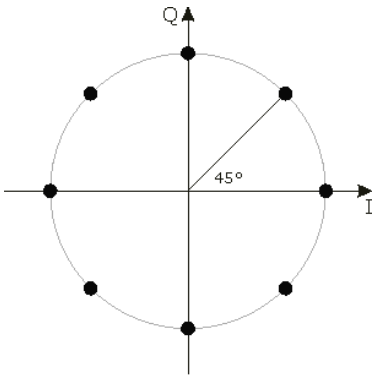
a) One concern when sending data wirelessly is that a packet may get lost (dropped). Assuming packet drops are rare and the data is of a type that requesting a resend of the data makes sense (so not live voice perhaps), how could the receiver know a packet has been dropped? You may not assume the data is being sent at a constant rate. **[4]**

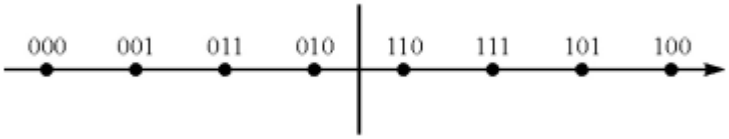
b) In free space, you would expect that signal centered at exactly 128MHz would go how much farther than one centered at 1MHz? Circle one. **[2]**

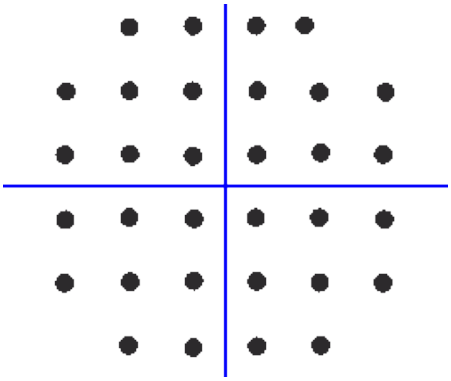
- About the same distance
- About 1/7 as far
- About 1/128 as far
- About 7 times as far
- About 128 times as far
- About 1/11.3 as far

c) Label each figure as one following: nFSK, nPSK, nQAM, or nASK where you are to supply the n (so 4FSK or 8QAM) for example. If more than one answer applies, you may select any. **[4]**

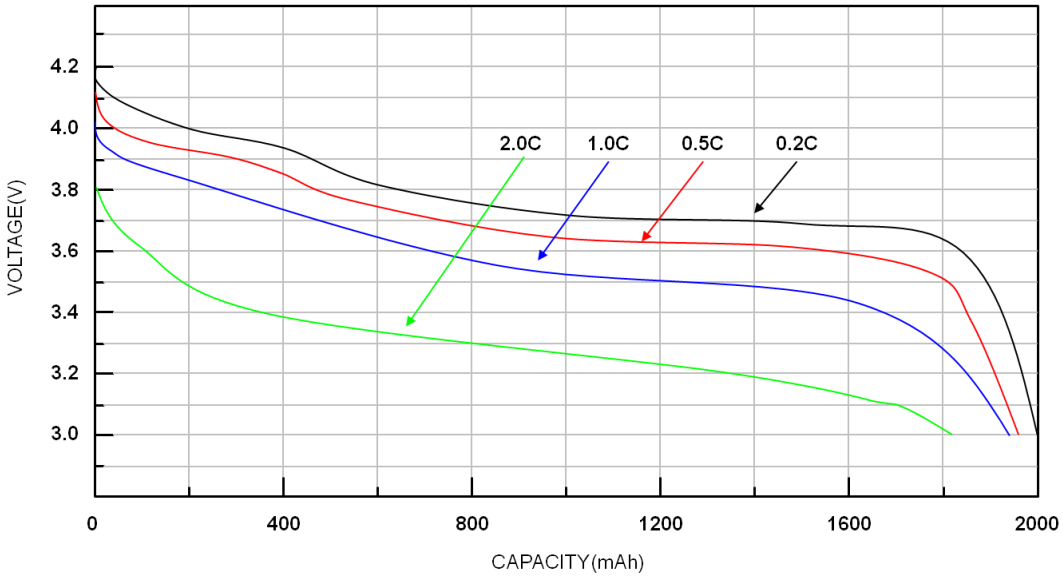








7. Consider the following LIPO battery discharge characteristics. **[5 points]**



If you draw 4A from this battery, about how long will it last before the voltage drops below 3.2V? Show your work.

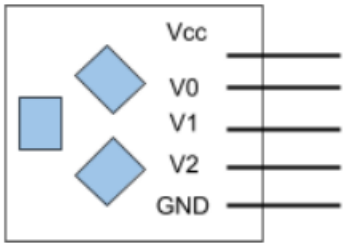
Watch your Attitude! [39 points]

<u>You should read the entire question before starting</u>.

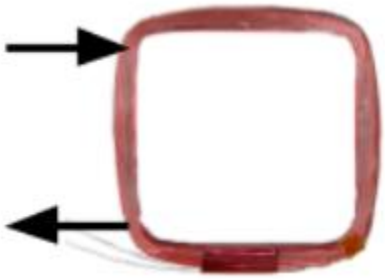
It turns out you did well enough with the satellite Electrical Power System (from the midterm), that you've now been tasked with interfacing with the various satellite elements in the **Attitude Determination and Control System (ADACS)**. The ADACS is responsible for using sensors to determine the attitude (i.e. rotational state: angular position and angular velocity) of the satellite, then using actuators to control the attitude so that it matches what is needed for mission parameters (e.g. pointing a very sensitive and very expensive camera at the earth rather than the blinding sun).

You have been tasked with designing an early prototype to interface with the sensors and actuators of a satellite ADACS. You will NOT have to develop the attitude determination algorithm or the attitude control algorithm, as that will be the responsibility of another part of the team. The system has the following components:

1. 1x Arduino Uno Board
2. 1x Sun Sensor custom-built by the electrical hardware team. This device contains three photodiodes with transimpedance amplifiers so that there are three analog voltage signals between 0 and 5V that correspond to how much light is hitting each photodiode. These analog voltages can be used to calculate a "sun vector" needed to determine attitude.



3. 1x LIS3MDL Magnetometer: This sensor measures the earth's magnetic field vector, which is used as the second vector necessary to determine attitude. You are to communicate with the Magnetometer using I2C.
4. 2x Magnetorquers placed in 2 perpendicular axes: These are coils of wire that form electromagnets, which can be controlled with PWM signals **(just like a motor)**. These electromagnets interact with the earth's magnetic field to cause a torque that rotates the satellite. They should be driven with +10V when active. The arrows in the diagram point in the direction of "forward" current.



5. 1x LD293D H-Bridge Driver: This H-bridge driver can be used to control the direction of current flow in the magnetorquers so that the magnetic field can be generated in both directions.
6. 1x 10V Power Rail + GND coming from the EPS
7. Any other passives or connectors you may need.

You will be implementing 1 interrupt service routine, and 2 of 3 tasks in FreeRTOS. Assume you have a special Arduino where the FreeRTOS Tick is set to 1ms

- **ISR:** `drdy_isr()` should be run every time there is new data from the magnetometer. It should read the current time and the raw sensor data and store them in global variables:

  ```
  unsigned long sense_time;  //read with the millis() function call
  uint8_t mag_bytes[6];           //raw magnetometer data
  uint16_t sun_data[3];           //raw sun sensor data
  ```

  After storing the values, the ISR should release task 1.

- **TASK 1:** `task1_sensors` will run at medium priority. The task should perform appropriate calculations on the magnetometer and sun sensor data to convert the raw data for the three dimensions of each into milligauss and millivolts respectively, and then call a special function:

  ```
  void send_data(unsigned long sense_time, float mag_mgauss[3],
                                           float sun_mvolts[3]);
  ```

  Which will enqueue the data for the control algorithm task.

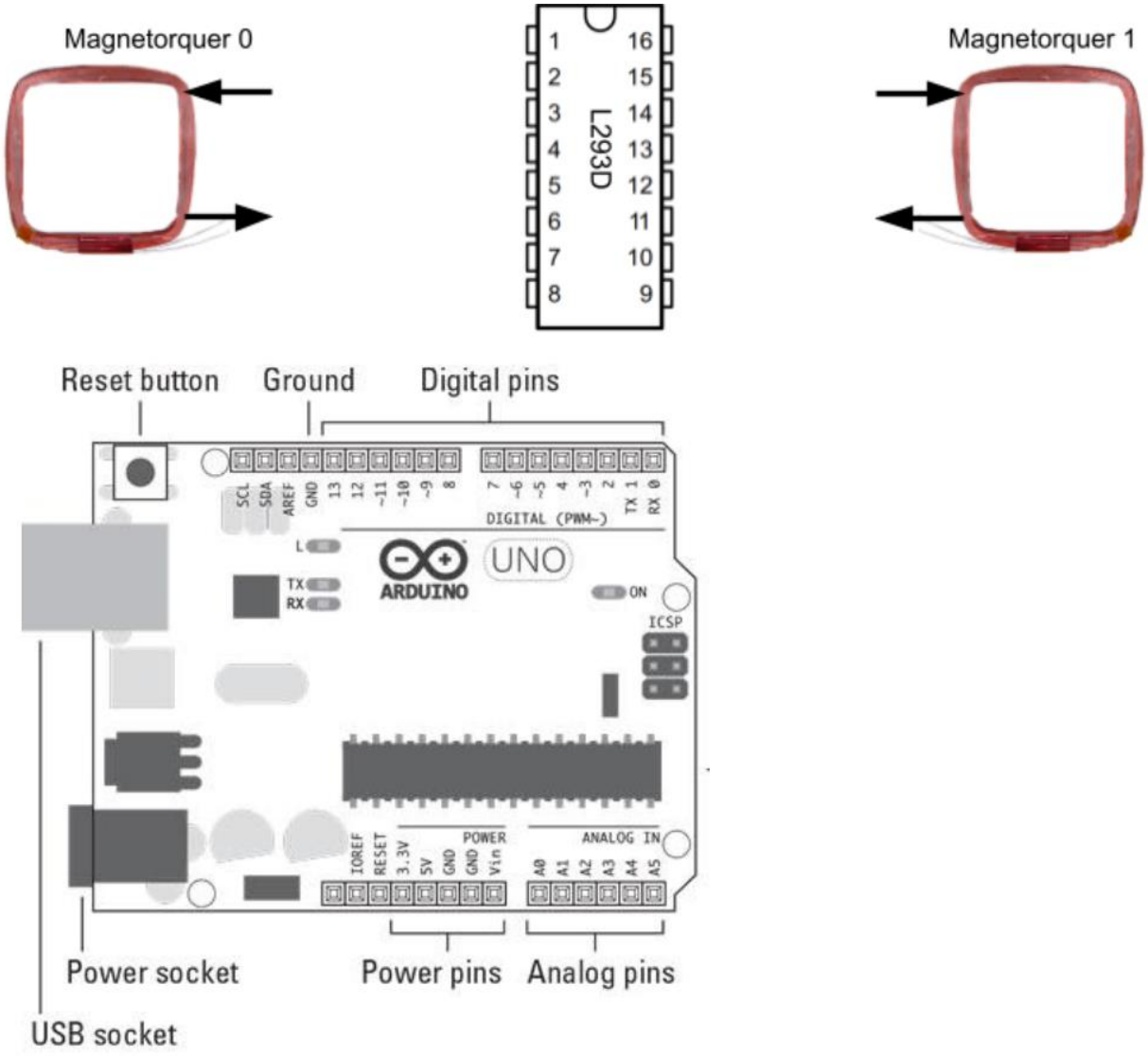
- **TASK 2:** `task2_actuators` will run at high priority every 10ms. The task should read the global variables:

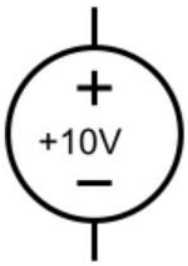
  ```
  float magtor_duty_cycles[2]; //float from 0.0 to 100.0 (percent)
  bool magtor_dirs[2]; //0 - Forward, 1 - Reverse
  ```

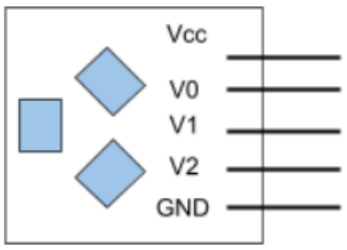
  and send out appropriate signals that will control the magnetic fields generated by the magnetorquers. The global variables will be set by the control algorithm task.

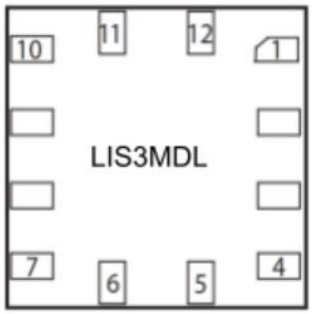
- **TASK 3:** `task3_control` will be running at lowest priority, but **YOU WILL NOT BE IMPLEMENTING THIS TASK.** You only need to initialize the task.

Part 1: Circuit diagram [8 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. *Use labels where applicable to make your connections easier to understand.*









Part 2: Helper Functions and Magnetometer Initialization [7 points]
We have provided a "write" helper function and started a "read" helper function. Finish the read function and initialize the magnetometer such that:

1. All 3 axes are in Ultra-High-Performance mode with an Output Data Rate of 80 Hz.
2. The full-scale range is +-4 gauss
3. The device is in continuous conversion mode

```
#define I2C_ADDR _____________          //fill this in

void lis3mdl_write(uint8_t reg_addr, unit8_t value) {

      Wire.beginTransmission(I2C_ADDR);
      Wire.write(reg_addr);   //send register address first
      Wire.write(value);      // Write Byte
      Wire.endTransmission(); //Sends everything out
}

void lis3mdl_read(uint8_t reg_addr, uint8_t *bytes, uint8_t count) {

      Wire.beginTransmission(I2C_ADDR);









}


int lis3mdl_init() {









}
```

Part 3: Setup Function [8 points]
Write a function to initialize the I2C Connection, the magnetometer, set up the interrupt drdy_isr based on the signal from the LIS3MDL, set up any necessary semaphores, and initialize the tasks.

```
#include <I2C.h>
#include <Arduino_FreeRTOS.h>
#include <semphr.h>
#include <time.h>

unsigned long sense_time;     //read with the millis() function call
uint8_t mag_bytes[6];         //raw magnetometer data
uint16_t sun_data[3];         //raw sun sensor data

float magtor_duty_cycles[3] = {0};  //float from 0.0 to 100.0 (percent)
bool magtor_dirs[3] = {0};          //0 - Forward, 1 - Reverse

void send_data(unsigned long sense_time, float *mag_mgauss,
                                                    float *sun_mvolts);

void setup(void){
```

Part 4: ISR & Task 1 [8 points]
Implement the ISR and task 1 so that the current time and sensor data is read, converted to appropriate values, and sent to the control algorithm every time data is ready.

```
void drdy_isr() {









}


void task1_sensors(void *pParam){
```

Part 5: Task 2 [8 points]
Implement task 2 to use the magnetometers as calculated by the control algorithm in Task 3.

```
void task2_actuators (void *pParam){

}
```