EECS 473 Final Exam Answer Key

Fall 2023

Name: ______________________Key____________________________ unique name: _______Key__________

Sign the honor code:

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

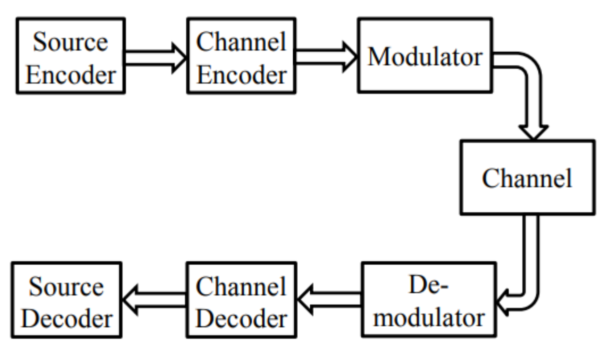
____

NOTES:

1. Closed book and Closed notes
2. There are **12** pages total for the exam. There is also a references handout.
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. Be sure you answer each question on the appropriate page.

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	dBm	mW	dBm	mW
-3	0.5	9	8	21	126
-2	0.6	10	10	22	158
-1	0.8	11	13	23	200
0	1.0	12	16	24	250
1	1.3	13	20	25	316
2	1.6	14	25	26	398
3	2.0	15	32	27	500
4	2.5	16	40	28	630
5	3.2	17	50	29	800
6	4	18	63	30	1000
7	5	19	79	33	2000
8	6	20	100	36	4000



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

$$r = \frac{10^{(p_t + g_t + g_r - p_r)/20}}{41.88 \times f}$$

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

1) Matching (place a letter) **[3 points, -2 per wrong or blank answer, minimum 0]**

- According to ______C__________, as the rate of discharge of a battery increases, its available capacity decreases.
- _______A_________ describes the maximum rate of error-free data that can theoretically be transferred over the channel if the link is subject to random data transmission errors, for a particular noise level.

A. Shannon's limit
B. McGraw's conjecture
C. Peukert's law
D. Smith's maximum
E. Feiger's rate

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

A. Say you are sending a signal using the band from 2.4 GHz to 2.45 GHz and the signal/noise ratio at the receiver is 18dB. The highest achievable data rate is *about* ***100 / 300 / 900 / 3000 / 9000 / 10000*** Mbits/sec

B. On a PCB, the label for a resistor is usually found on the ***silk screen / via / solder mask / metal /hidden*** layer.

C. One advantage of a linear regulator compared to a buck converter is that the linear regulator is generally ***simpler to design and get working / uses an inductor rather than a capacitor /***
is more power efficient when you need to drop a voltage a fair bit /
is also capable of increasing the voltage rather than just reducing it.

3) True/False: *Circle the true statements*. **[8 points, -2 per wrong or blank answer, minimum 0]**

A. Source decoding generally involves checking error correction bits and doing error correction/detection as needed.

B. 64 QAM modulation involves varying both frequency and phase of the signal.

C. You could reasonably expect a battery being drained at a constant rate of 0.05C to last at least 18 hours but couldn't expect that same battery being drained at 5C to last for at least 15 minutes.

D. The largest value (closest to positive infinity) that can be represented by a signed 4-bit Q2 number is 1.75.

E. Paying a consultant to modify a program licensed under the GPL is generally legal.

F. Busybox is a utility found on embedded Linux platforms. It is a single executable that can perform the basic functions of many common Linux programs.

G. In our taxonomy a "firm" deadline means the desired information still has (potentially reduced) utility after the deadline passes.

H. 8-PSK sends 3 bits of information in a single symbol.

4) Short answer **[16 points]**

A. For the read function member of the file_operations struct, describe the meaning of the return value (be sure to include 0). An example function header is found below. **[4]**

```
ssize_t memory_read(struct file *filp, char *buf,size_t count,
                    loff_t *f_pos);
```

The return value indicates to the user program how many bytes have been read. A return value of 0 indicates end of file.

B. Say you have one task that has a period of 4ms and an execution time of 2ms. Find a period and execution time of a second task such that the two can be scheduled by EDF but cannot be scheduled by RMS. Show the EDF schedule and the RMS schedule. You can assume there is no overhead of any type. Your answers do not need to be integers (though there are integer solutions). **[6]**

One example: Task B has period 10ms and execution time of 5ms
With RMS, Task B will miss its deadline at time = 10ms.
With EDF, Task B will gain priority after the 2nd instance of Task A, allowing it to finish on time.

C. Add 1.75 and -2.125 as 6-bit Q3 numbers doing binary addition. Clearly show your work. **[6]**

-2.125 as 6-bit Q3 2's complement: 2.125 = 010.001 => complement + 1 => -2.125 = 101.111

1.75	001.110	001110
+ -2.125	101.111	101111
= -0.375	111.101	111101

(Technically there should be no decimal point ".", but no points were taken off for this)

5) Say you have a code where you have 3 data elements (d1 to d3) followed by 3 parity bits (p1 to p3) where the parity bits are defined as follows (the function P() returns the even one's parity as we did in class, so e.g., P(1,1,1) =1 and P(0,1,1)=0).

- p1=P(d2, d3)
- p2=P(d1, d2)

List all possible functions that could be used for p3 that would allow for the correction of any one-bit error. Briefly justify your answer. **[6 points]**

p3 = P(d1, d3) and p3 = P(d1, d2, d3)

Each data bit must be covered by two or more parity bits and each data bit must be covered differently. So d1 and d3 must be covered by p3, while d2 need not be but can be.

If d1 is flipped in transmission, p2 an p3 will be wrong. If d3 is flipped, p1 and p3 will be wrong. If d2 is flipped, p1 and p2 will be wrong for the first solution and p1, p2, and p3 will be wrong for the second solution.

If any parity bit goes bad, it only affects that parity bit.

6) Say we have a 7.2V battery with 2.5 Ah capacity to drive a device that needs 5V and has a constant current draw of 100mA. You are to assume the battery maintains a 7.2V output until it is drained. **[8 points]**

A. How long would the battery last if we use an ideal LDO? Clearly show your work. **[4]**

2500mAh / 100mA = 25 hours

B. If you used an ideal buck converter, at what "C rate" would you be drawing from the battery? Clearly show your work. **[4]**

P = 5V * 100mA = 500mW

Current in = 500mw / 7.2V = 69.44mA

2500mAh / 69.44mA = 36.00 hours

Since 1C = 1 hour, the C rate = 1/36 = 0.0278

7) We generally think of sourcing encoding and channel encoding as very different things. **[7 points]**

A. Define those terms in your own words. **[4]**

Source encoding:

Source encoding is about the compression of the message to reduce message length.

Channel encoding:

Channel codes systematically add redundancy to enable error correction and thus enable high-rate transmissions because we can correct errors that might occur due to sending at such a high speed.

B. One of them generally adds bits to the data and another one removes them. Explain how doing each of those things is useful/needed. **[3]**

Source encoding removes bits and this is helpful as it makes the data that needs to be sent smaller while keeping the overall data intact. Channel encoding adds bits to allow the receiver to error correct in the event data is lost during the transfer.

8) Consider a 6.0GHz radio/receiver pair where the transmitter broadcasts at 100mW and uses an antenna with a gain of 2.4dBi. The receiver has a sensitivity of -90 dBm and uses the same antenna as the transmitter. What is the expected range? Show your work. Recall there is useful information on the cover page. **[6 points]**

10 ^ ((20 + 2.4 + 2.4 + 90) / 20) / (41.88 * 6000 MHz) = 2.18 km

Design Problem: A fire engine toy [37 Points]

Please read all the entire problem BEFORE starting

The Elf Toy Company is building a new fire-engine toy for the season. You are tasked with designing a prototype for the electronics and software needed to operate the toy. The toy will be mechanically powered, but has an **electric emergency brake, lights, and a siren**. It should have the following features: The **lights must flash** and the **buzzer siren must sound** 10 times per second with a duty cycle of 50% (so on for 50ms, off for 50ms, etc.). For safety, the **emergency brake shall activate until the next power cycle** if the toy falls or gets too close to an obstacle. Assume the TICK length is 1ms.

You will have the following components available to build the toy:

- 1 Arduino UNO;
- 1 Ultrasonic Sensor
- 1 LIS3DHH Accelerometer
- 1 bi-directional level shifter.
    - Connect the lower voltage source to LV, the higher voltage source to HV. The device connects HV1 to LV1, HV2 to LV2, etc.
    - Arduino interprets **3.3V as HIGH**, so 3.3V inputs to it **DO NOT** need shifting
- 1 3.3 V regulator
- 1 LED Emergency Light
- 1 Siren Buzzer
- 1 Emergency Brake Module
    - This device has a single **enable (EN) pin** which actives the brake when **HIGH.**
- Resistors/Capacitors/Inductors as needed

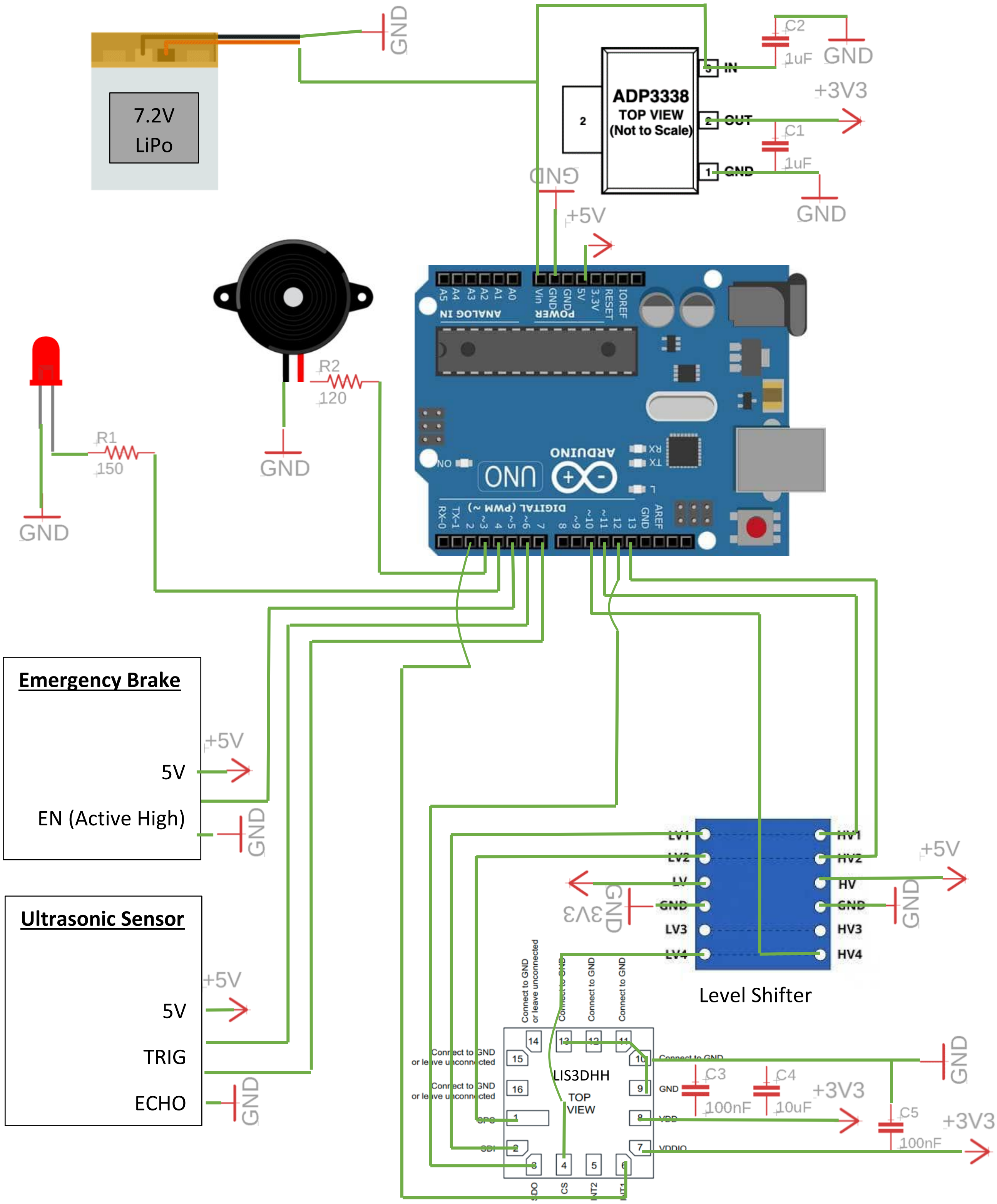
Your system should have the following characteristics:

- You must use an external 3.3V regulator to power the LIS3DHH sensor
- Your buzzer may be connected to a GPIO pin, provided a 120Ω current-limiting resistor is used.
- Your LED requires a 150Ω current-limiting resistor.
- The buzzer AND light should flash 10 times per second with a duty cycle of 50% and have the **HIGHEST** priority (**task1**).
- Your Arduino will monitor the acceleration via SPI with deferred interrupt processing.
    - If the Z-axis acceleration is <0.5g, this indicates the vehicle has fallen and the emergency brake should be activated.
    - Please use **4MHz** for your SPI clock frequency
    - Use the **FIR** filter on the chip and **235 Hz** bandwidth
    - The deferred-interrupt reads should be **MEDIUM** priority (**task2**).
- Another task will monitor distance.
    - You **DO NOT** need to implement this task function
    - However, you must declare this task and it will run at the **LOWEST** priority (**task3**)
- Use the following pin mappings for your devices
    - LIS3DHH Accelerometer: **CS – 10, MOSI – 11, MISO – 12, SCK – 13, INT1 – 2**
    - Other GPIO Outputs: **Siren – 3, LED – 4, Emergency Brake EN – 5**
    - Ultrasonic Sensor: **TRIG – 6 (Output), ECHO – 7 (Input)**
        - You must **INITIALIZE** these pins in setup.
        - No reads/writes to these pins outside the provided task are necessary.

Assume a FreeRTOS "tick" is 1ms. No pdMS_TO_TICKS() is necessary.

Part A: Wiring [9 Points]

Please complete the circuit here. For 3.3V, 5V and GND you are **highly encouraged to use labels** to keep the diagram neat. Draw passives (resistors, capacitors, inductors) where needed. Be sure to include values for all passive components you use.



Part B: Interface [4 Points]

Please implement the following SPI read/write functions:

```
uint8_t sensorSPITransaction(uint8_t cmd, uint8_t data)
{
      // This core function is intended for read and write

      uint8_t resp;
      SPISettings settings(4000000, MSBFIRST, SPI_MODE3);

      SPI.beginTransaction(settings);
      SPI.transfer(cmd);
      resp = SPI.transfer(data);
      SPI.endTransaction();


      return resp;
}
```

```
void sensorSPIWrite(uint8_t cmd, uint8_t data)
{
      // Optional: Wrapper to help with write

      sensorSPITransaction(cmd & 0x7F, data);
}
```

```
uint8_t sensorSPIRead(uint8_t cmd)
{
      // Optional: Wrapper to help with read

      return sensorSPITransaction(cmd | 0x80, 0);
}
```

Part C: Short Answer [4 points]

1. The emergency brake shall activate when the Z-axis acceleration is less than 0.5G (this indicates a fall). Please calculate the corresponding 16-bit (signed) integer value. (**Hint:** Page 7 of the datasheet defines the resolution in mg/digit)

$$0.5G * \frac{1000mG}{G} * \frac{1\ digit}{0.076\ mg} = 6579$$

2. What does the most-significant bit of every SPI command indicate for this device? Please specify the correct case for each value. (This answer will help you write you wrapper functions if you choose to do so)

R/W
Read = 1, Write = 0

3. Which mode (all are listed in the datasheet makes most sense for this application. Please assume that the Arduino has enough CPU time to read the acceleration before the next value is ready (ie. you do not need buffering).

Bypass Mode – avoids buffer altogether

4. Remember that the acceleration readings are interrupt-driven. However, the reads will be handled using deferred interrupts to ensure the siren/light timing is consistent. Please briefly explain how to use deferred interrupts with FreeRTOS.

In FreeRTOS, deferred interrupts are handled using semaphores. The ISR gives a semaphore which the handler task will wait on. Once available the handler task will take the semaphore, complete the read and then block again, waiting for the semaphore.

Part D: Setup [6 points]

```
#define SIREN_PIN 3
#define LED_PIN 4
#define EMER_BRAKE_PIN 5
#define TRIG_PIN 6
#define ECHO_PIN 7
#define LIGHT_SIREN_FREQ 50

#define CTRL_REG1 0x20
#define INT1_CTRL 0x21
#define CTRL_REG4 0x23

#define OUTZ_LSB  0x2C
#define OUTZ_MSB  0x2D

// Semaphore for Interrupt-Based SPI
SemaphoreHandle_t xAccelSem;
vSemaphoreCreateBinary(xAccelSem);

void setup()
{
      // Configure interrupt for accelerometer
      attachInterrupt(2, ISR, RISING);
      SPI.begin();

      // Configure GPIO pins
      pinMode(LED_PIN, OUTPUT);
      pinMode(SIREN_PIN , OUTPUT);
      pinMode(EMER_BRAKE_PIN, OUTPUT);
      pinMode(TRIG_PIN, OUTPUT);
      pinMode(ECHO_PIN, INPUT);

      // Configure SPI Accelerometer
      sensorSPIWrite(CTRL_REG1, 0b11000000);
      sensorSPIWrite(INT1_CTRL, 0b10000000);
      sensorSPIWrite(CTRL_REG4, 0b01000000);

      // Define and begin FreeRTOS tasks
      xTaskCreate(task1, “Accelerometer”, 200, NULL, 2, NULL);
      xTaskCreate(task2, “LightSiren”, 200, NULL, 3, NULL);
      xTaskCreate(task3, “Ultrasound”, 200, NULL, 1, NULL);

      vTaskStartScheduler();


}
```

Part E: Interrupt [5 points]

```
void ISR(void)
{
	xSemaphoreGiveFromISR(xAccelSem, NULL);



}
```

,

Accelerometer Task [5 points]

```
# define HALF_G 6579
void task1(void *pvParameters)
{
	// Accelerometer Task
	int16_t accel;
	while(1) {
	  if(xSemaphoreTake(xAccelSem, portMAX_DELAY)) {

		// do SPI read, likely to bytes
		accel = SensorSPIRead(OUTZ_MSB) << 8;
		accel |= SensorSPIRead(OUTZ_LSB);

		// if Z-axis is too low, then brake
		if(accel < HALF_G) {
			digitalWrite(EMER_BRAKE_PIN, HIGH);
		}

	}


}
```

Part F: Lights [5 points]

```
void task2(void *pvParameters)
{
      // Lights and Siren
      const portTickType xFrequency = LIGHT_SIREN_FREQ;
      xLastWakeTime = xTaskGetTickCount();
      uint8_t state = 0;

      while(1) {

            // flip lights and siren states
            digitalWrite(SIREN_PIN, state);
            digitalWrite(LED_PIN, state);
            state = !state;

            // block for one half period
      vTaskDelayUntil(&xLastWakeTime, xFrequency);

      }
}
```

```
void task3 (void *pvParameters)
{
      // Ultrasonic Task

      REMINDER: This is implemented for you!
      You ONLY need to initialize this task
}
```