

EECS 473 Final Exam

Fall 2015

Name: _____ 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.	/13
2.	/7
3.	/15
4.	/10
5.	/10
6.	/10
Design	/35
Total	/100

NOTES:

1. Closed book and Closed notes
2. There are **13** pages total for the exam as well as a handout. The last 2 pages of the exam can be removed and used as reference for the last problem.
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 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. Fill in the blank or circle the best answer. [13 points, -2 per wrong or blank answer, minimum 0]

a. Channel encoding can best be described as a way to

- help insure that data arrives correctly over the channel.
- reduce redundancy of data over the channel.
- add noise to a channel and thus reduce "dithering".
- reduce noise-power in a channel.

b. Say you are sending a signal using the band from 2.45GHz to 2.5 GHz and at the receiver the power of the signal is three times that of the ambient noise. According to the Shannon–Hartley theorem, the highest achievable data rate is about 25 / 100 / 250 / 1000 / 2500 Mbits/sec

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

c. If a transmitting antenna is rated at 20dBi in a certain direction, the power sent in that direction is about 100 times greater that it would be if an isotropic antenna were used.

d. PCB manufacturers use a number of different terms to describe the distances. The most common are "thou", "mm" and "mil". A thou is 0.001 inches / 0.001 meters / 0.001 cm / 0.001 feet.
A mil is 0.001 inches / 0.001 meters / 0.001 cm / 0.001 feet.

e. Binary modulation / Channel encoding / Keying / Hamming channel is a family of modulations where we allow only a predetermined set of values rather than an arbitrary modulation.

f. dBi is the gain of an antenna compared with the hypothetical isotropic antenna. An isotropic antenna is an antenna which

- requires a 50Ω matching impedance.
- uniformly distributes energy in all directions.
- distributes energy in a toroid.
- must have the same length on the transmitter and receiver.

g. Which of the following is a technique we might use on a real-time systems that we would almost never use in a desktop computer? The technique and its reason for use must both be correct.

- Running code with optimization turned off to enable more consistent run times.
- Declaring variables "volatile" to improve processing speed.
- Turning off the data cache to enable more consistent run times.
- Running code with optimization turned off to enable memory-mapped I/O.

2. Answer the following questions about power converters.

[7 points, -2 per blank or wrong answer, minimum 0]

a. Figure #1 is a

linear / buck / boost / buck-boost converter.

b. What is the purpose of the “bypass capacitor” in figure 1?

- **Storing charge to enable Vout to be greater than Vin**
- **Keeping the noise from the pass transistor switching isolated from the output voltage.**
- **To cancel out the phase shift introduced by the decoupling capacitor**
- **Reducing the power loss of the feedback resistors.**

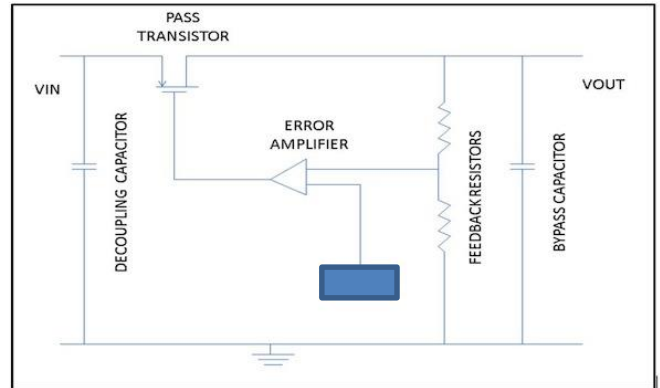


Figure 1: A voltage converter

c. A(n) **LDO / buck / boost / buck-boost** converter is a type of linear regulator which can generate a Vout less than Vin.

d. We've claimed that for extremely low-power devices, generally powered by a battery, we often prefer to use an LDO rather than a buck converter. Why is that?

- **An LDO typically requires larger capacitors and thus draws more power.**
- **When dealing with large voltage drops, an LDO is typically more efficient.**
- **LDOs are more resistant to LiPo battery noise and low-power applications typically use LiPo batteries.**
- **The quiescent current of an LDO is typically considerably lower than a buck converter.**

3. Short answer [15 points]

- a. Consider a 2.4GHz radio/receiver pair where the transmitter broadcasts at 18 dBm and uses an antenna with a gain of 40x over an isotropic antenna. The receiver has a sensitivity of -80 dBm and is an isotropic antenna. What is the expected range? Show your work [6]

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

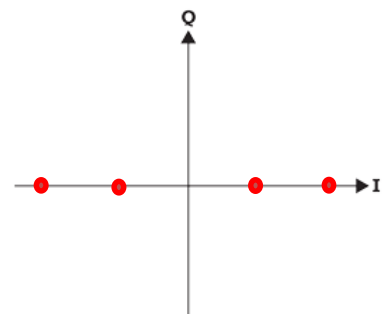
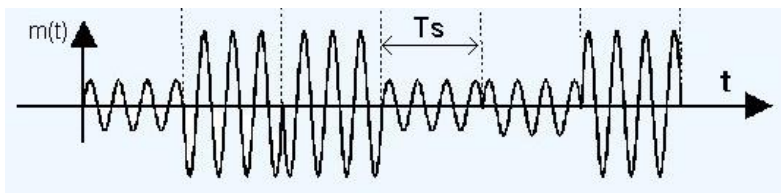
$$r = \frac{10^{(18+16+0+80)/20}}{41.88 \times 2.4 \times 10^3} = 4.986 \text{ km}$$

Note: a lot of folks forgot this formula assumes freq in MHz.

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

- b. What are the pros and cons of using a desktop OS (Linux, Windows etc.) in an embedded application? Give an example where you would certainly want to use such an OS and where you certainly would not want to. [5]
- Pros: can run big, off-the-shelf applications (OpenCV, Word, whatever). Can run very complex systems. **Example:** If you are doing computer vision, you are probably better off using OpenCV than writing your own vision libraries or porting them.
 - Cons: Real-time responses can be hard if not impossible. Getting I/O to work at all can be very frustrating as you have to deeply understand how the OS does I/O. **Example:** a hard-real-time system like an airbag would be a horrible place for a desktop OS.

- c. Sketch the constellation associated with the following modulation scheme (assume all modulation options are shown). [4]



4. Fixed point [10 points]

Throughout this problem, when asked for a specific numeric value, give your answers as 8-bit binary numbers.

- a. What are the pros and cons of using fixed point rather than floating point in an embedded application? [2]

Fixed point is lower-power and processors without floating point tend to be cheaper. However fixed point has a much more limited range-of-representation and it can be more difficult to work with. Plus (and most high-level languages) understand floating point but not fixed point.

- b. Say, in a rather non-typical application, you wish to add an 8-bit Q7 number to an 8-bit Q6 number and store the result as an 8-bit Q5 number. All numbers are two's complement values.

- Could the result overflow? If yes, give an example, writing the numbers as 8-bit binary numbers. If no, explain why not. [2]

No. The range of an 8-bit Q7 is -1 to almost 1. The range of an 8-bit Q6 is -2 to almost 2. The range of an 8-bit Q5 number is -4 to almost 4. The sum of the first two can't be out of range of the third.

- Write a C function which adds an 8-bit Q7 number and an 8-bit Q6 number returning the value as the nearest an 8-bit Q5 number (if between two values you may round to either). You may assume there is no overflow. You also should 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. [6]

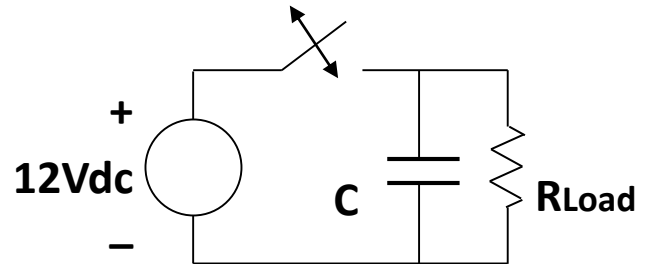
```
char Qadd(char Q7, char Q6)
{
    return (Q7<<1+Q6+2)>>2;
}
```

5. Switching power supplies [10 points]

a. Below is a simplistic idea of what a buck converter might be. If the switch were open half the time, one should get an output voltage of 6V (half of V_{in}). However, real implementations will burn out the switch.

- Why is that? [2]

You are connecting a 12V supply to directly to a capacitor at a lower voltage. The switch (and wire) will need to drop the difference in voltage. That's a lot current.

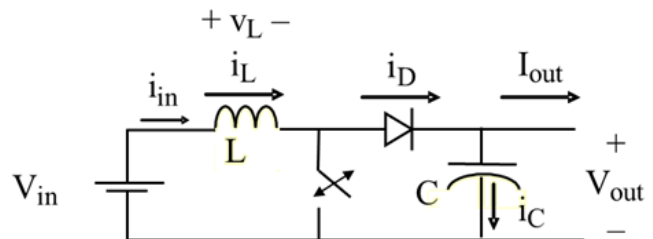


- How can we fix that problem? Either modify the above circuit or draw a new one. [4]

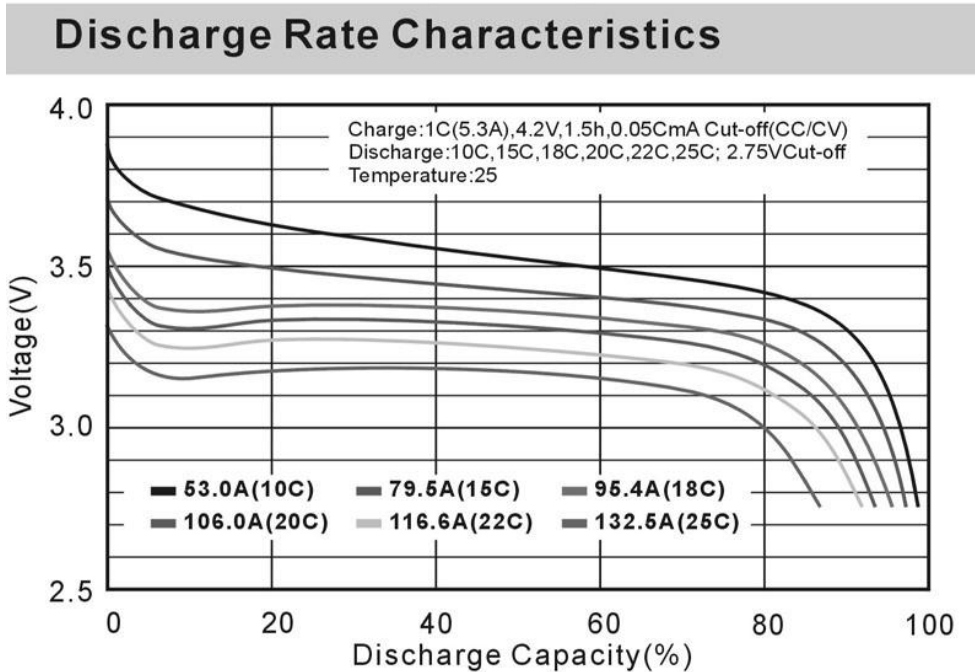
Add an inductor between the switch and the line going to the cap. That will limit the current. (A better answer would note that that inductor also creates problems and you'll want a diode, but that wasn't required for full points).

b. What is the purpose of the diode in the circuit below? What would happen if it were removed? [4]

When the switch is closed prevents the positive side of the cap from being connected to ground.



6. Consider the following (apparently real) LIPO battery discharge characteristics from “Fullymax”. (The actual figure on their website is also in black-and-white). **[10 points]**



- a) What is the capacity (in mAh) of this battery? **[3]**

5300mAh

- b) If you draw 100A from this battery, about how long will it last before the voltage drops below 3.0V? Show your work. **[4]**

Looks like at 100A you are at around 90% discharge before it drops below 3.0V. So $(5.3Ah/100A) \cdot 0.9 = 2.86$ minutes

- c) When drawing heavily from this battery, it has one or more properties that would make “state-of-charge” analysis (figuring out how charged the battery is) a bit tricky. What issue(s) is/are there? **[3]**

The voltage “droop” when the battery is still highly charged will make it hard to figure out if the battery is in the 0-20% range or ~80% range.

Design problem: Detecting a person in a room [35 points]

(You will want to read the whole problem before starting)

You are given the task of modifying an existing system that will identify and locate a person in a room. The system uses a Raspberry Pi (RPI) running FreeRTOS and has a working distance sensor mounted on a servo to scan the room. Naturally the distance sensor cannot distinguish between people and other objects. The developer wants to mount a narrow beam Passive infrared (PIR) sensor next to the distance sensor to detect human heat. Additionally, the developer wishes to use an external device to generate the pulse-width modulation (PWM) needed to control the servo. An I2C two-channel PWM chip is provided.

You are tasked with doing the following:

- Connecting the various components, adding passive devices as needed.
- Providing initialization details
- Writing a function which turns the servo to a specific angle
- Writing a function which reads the PIR sensor (via the PWM chip)
- Creating a main program and task that will scan the room by sweeping the sensors in a 90 degree arc (the servo will be mounted in the corner of the room). If you detect a person, you are to convert angle and distance into coordinates and send those coordinates via a wireless module.

Processor, Sensor and Actuator Characteristics

- **RPI:** RTOS tick is every 1ms.
- **PIR Sensor:** Powered by a 5V input, it provides a logic 1 when heat is detected and otherwise is a zero.
- **Servo:** Its position is zero degrees when the high pulse is 1 ms, 180 degrees when it is 2.8ms and the angle changes linearly with the pulse width. The frequency is to be 50Hz.
- **Distance Sensor:** Is an I2C sensor connected to the RPI GPIO 0 and 1 with an I2C 7-bit address of 0x60.
- **PWM device:** The External I2C PWM Device is a PCA9530. It is a 2 Channel Programmable PWM via I2C. The channels can be used for GPIO or PWM outputs. Excerpts from the data sheet are provided as a supplement.

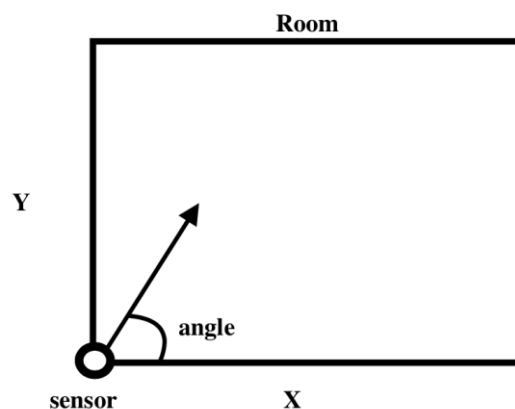
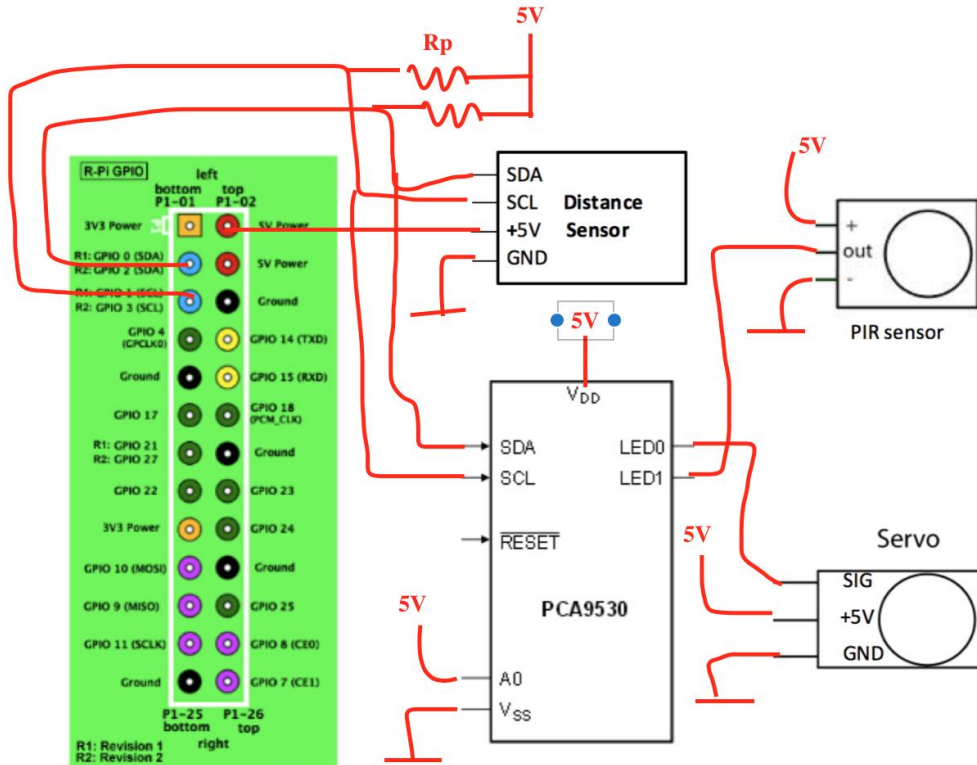


Figure 2: The sensor in the room.

Part I: Connections [8]

Connect the PCA9530 PWM device to the RPI. It is possible to use one of the PWM channels as a GPIO input. We want to keep all the RPI GPIO in reserve, so you are to use PWM channel 0 for the servo and PWM channel 1 for the PIR detection. You also need to share the I2C line with distance sensor. Assume that the PCA9530 reset is tied to a system reset. You may assume the RPI's 5V output generates enough current for all devices. You may add passive devices as needed.



Part II: PCA9530 Register Initialization [6]

Fill in the table to initialize the registers. Use DC for don't care. The servo should start at position zero. Values other than LSO should be in decimal. LSO should be in hex.

Register Name	Initial Value
INPUT	DC
PCSO	2
PWM0	13
PCS1	DC
PWM1	DC
LSO	0xF2 (hex)

Part III: Servo Control API [9]

In this part of the design problem, you are to write a function which causes the servo to turn to a specified angle (where 0 degrees is along the X-axis of Figure 1). The prototype for your function is as follows.

```
bool set_servo_angle(unsigned int angleInDegrees);
```

The function sets the servo angle between 0 and 180 degrees. If the angle specified is out of range, the function should return false (0), otherwise the servo should turn to the appropriate angle and return true (1).

The following I2C function is available.

```
void i2c_transaction(uint8_t daddr, uint8_t raddr, char* data)
```

The function performs either a read or write based on the address (as normal for I2C devices).

- **daddr**: The 8-bit read/write device address
- **raddr**: The address of the register to be read or written
- **data**: A pointer to a local buffer

```
#define DUTY_CYCLE_0 0.05 // 1ms / 20ms  
#define DUTY_CYCLE_180 0.14 // 2.8ms / 20ms  
#define DUTY_CYCLE_STEP 0.0005 // (0.14ms - 0.05ms) / 180  
#define PCA_ADDR_W 0xC2 // 0b1100,0010  
#define PCA_ADDR_R 0xC3 // 0b1100,0011
```

```
bool set_servo_angle(unsigned int angleInDegrees)  
{  
  
    double dutyCycle = DUTY_CYCLE_0;  
  
    char data;  
  
    if (angle > 180) return false;  
  
    dutyCycle = DUTY_CYCLE_0 + angle * DUTY_CYCLE_STEP;  
  
    data = (char)(dutyCycle * 256);  
  
    i2c_transaction(PCA_ADDR_W, PWM0_REG_ADDR, data);  
  
    return true;  
  
}
```

Part IV: PWM Chip GPIO Read [3]

In this part of the design problem, you are to write a function which gets the PIR value (1 or 0) from the PCA9530. The function prototype is as follows.

```
int read_PWM_GPIO(void)
```

The function should return the value of the PIR sensor. You may use the I2C function listed above.

```
#define INPUT_REG_ADDR 0x00 // table 3

int read_PWM_GPIO(void)
{
    char value;

    i2c_transaction(PCA_ADDR_R, INPUT_REG_ADDR, value);

    return (value & 2) >> 1;
}
```

Part V: Write the Sweep and Detect Task [9]

Finally, you are to complete the main program and write a FreeRTOS task which will cause the servo to scan the room once every 4.5 seconds. Every half a second you should turn the servo 10 degrees. Once it reaches one of the boundaries, the servo should change direction and sweep back at the same rate (0°, 10°, 20°, ..., 80°, 90°, 80°, ..., 10°, 0°, 10°, etc.) You are to complete the main program on this page and write the task on the next page.

You may use the functions you wrote in parts III and IV. In addition, you may assume that following functions exist:

- `int read_distance_sensor(void)`
The distance is returned in cm. If nothing is detected, a -1 is returned .
- `void send_coordinate(int x, int y)`
Calling this function will cause a message to be sent over the wireless module indicating that there is a person at location x, y.

```
int main (void){
//The following function will initialize the pwm device
// to the values you selected in Part II. You do not need to
// implement this function.
init_pwm(void);

// create task 1
xTaskCreate(task1, "task1", 1000, NULL, 1, NULL);

vTaskStartScheduler();
enable_interrupts();

while(1);

return 0;
}
```

Please write the task associated with part 5 here.

```
void task1() {
    int servoAngle = 0;
    int distance = 0;
    int x = 0;
    int y = 0;
    bool ifPosDirection = true;
    // lastwake time and frequency
    TickType_t xLastWakeTime;
    // 4.5 seconds -> 9 * 10 degrees -> turn 10 degrees every 0.5 second
    // xLastWakeTime is 500 clock tick
    const TickType_t xFrequency = 500;
    xLastWakeTime = xTaskGetTickCount();
    // main loop
    for (;;) {
        vTaskDelayUntil(&xLastWakeTime, xFrequency);
        if (read_PWM_GPIO()) {
            distance = read_distance_sensor();
            x = distance * cos(servoAngle);
            y = distance * sin(servoAngle);
            wireless_send(x, y);
        }
        if (ifPosDirection) {
            if (servoAngle < 90) servoAngle +=10;
            else {
                ifPosDirection = false;
                servoAngle -= 10;
            }
        } else {
            if (servoAngle > 0) servoAngle -= 10;
            else {
                ifPosDirection = true;
                servoAngle += 10;
            }
        }
        set_servo_angle(servoAngle);
    }
}
```