Assignment 1: Digital Circuits

You may work together with a partner. However, all source code, answers, and figures must be your own. Copying another's work is a violation of the academic honesty policy.

Part 1 (5 pts)

Draw a rough circuit containing a Beaglebone, and LED, and a resistor designed to illuminate the LED when a GPIO pin is configured as an output and set to the HIGH state (output V_{cc}). What is the logic level voltage outputted by the BeagleBone?

Refer to the following product page to find the forward voltage bias of a typical surface-mount green LED. Calculate the required resistance of a series current-limiting resistor to ensure no more than 3ma is pulled from the GPIO pin.

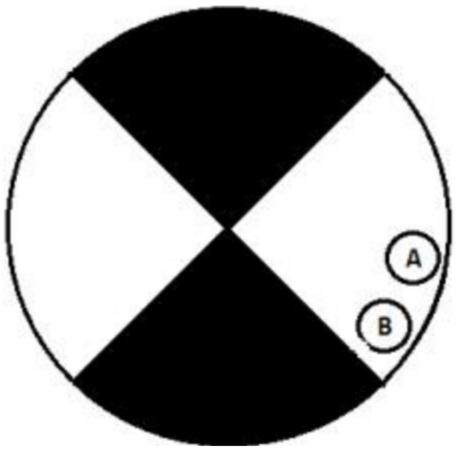
http://www.digikey.com/product-detail/en/LG%20Q971-KN-1/475-1409-1-ND/1802597

Part 2 (5 pts)

Using the fact that the forward voltage bias of these LEDs is greater than $V_{cc}/2$, draw a second circuit with two LEDs and one resistor such that both LEDs are turned off by putting the GPIO pin into a high-impedance state and either LED can be turned on when the GPIO pin is set to HIGH or LOW.

Part 3 (5 pts)

Given the encoder wheel below. Draw the square waveforms you would expect to come out of sensors A and B through one full counterclockwise rotation of the wheel. Assume the sensor reads HIGH (V_{cc}) when the sensor is in a white region. Start your waveform at time zero as the wheel is currently positioned. What is the resolution of this encoder?



Part 4 (5 pts)

Imagine you are programming a microprocessor which has been tasked with counting the above encoder. It is configured to wake up on a hardware GPIO interrupt every time a signal transitions from one logic state to another, this is called edge-detection. It is also configured to execute one of the following two interrupt service routines (ISR) each time a pin sees an edge. Fill in your own pseudo-code that increments the global variable ‘position’ up or down depending on the direction of rotation. You may call the made-up functions ‘get_pin(A)’ and ‘get_pin(B)’ which will return HIGH or LOW (1 or 0) reflecting the state of the pins immediately after the interrupt.

```
#define HIGH 1

#define LOW 0

int position;


void pin_A_ISR(){

        // Interrupt Service Routine for Pin A


}

void pin_B_ISR(){

        // Interrupt Service Routine for Pin B


}
```

This is most easily done with if/else logic. 3 Bonus points will be awarded if each ISR contains only a single line of C code and contains no if/else statements.