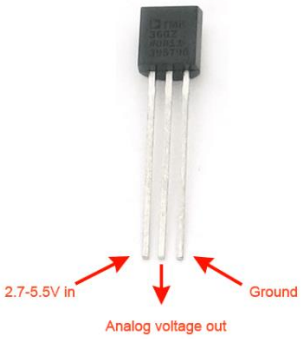
EE4144 Quiz 1 Name: _____

NYU Tandon School of Engineering ID: _____

1. Suppose we are using the Atmel328P microcontroller connected to a 1 MHz clock (unlike the Arduino Uno which uses a 16 MHz clock). Our goal is to provide an indication of temperature by using a blinking LED (using a PWM output). We are using the TMP36 temperature sensor shown below:



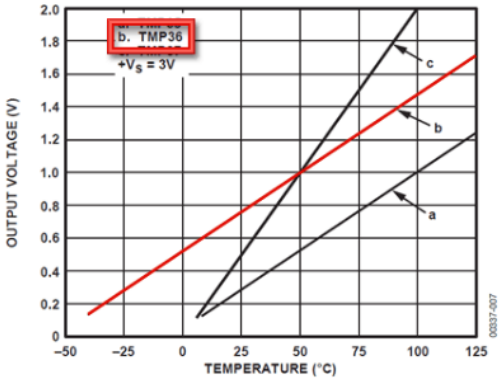


Figure 6. Output Voltage vs. Temperature

Temperature is calculated as follows (as per the data sheet):

Centigrade temperature = [(analog voltage in mV) - 500] / 10

We would like to achieve the following characteristics:

- If the temperature is 0°C or less, an LED blinks at a rate of 2 Hz. That is 0.25s "On", 0.25s "Off" (50% duty cycle)
- If the temperature is 50°C or greater, the LED appears to stay "On" permanently. Blinking "On" and "Off" so quickly that it appears "On")
- The blinking rate changes proportionally to the temperature. For example 25°C would blink the LED "On" for 0.125s, "Off" for 0.125s.
- The duty cycle is fixed at 50% for all temperatures. Just the blinking rate is changing.

a. Sketch a schematic of the hardware setup, showing the connections between the pins, TMP36 etc. Don't forget the LED needs a current limiting resistor in series. You can use a 330Ω resistor.
b. Write the C code to set up the analog input registers for the TMP36. (ADMUX, SDCSRA, ADCSRB, DIDR0)
c. Write the C code to set up the PWM(timer) configuration registers (TCCR0A, TCCR0B) to satisfy the timing requirements.
d. Lastly, write the C code that updates the output compare registers (OCR0A, OCR0B) based on the ADC data (ADCH, ADCL) from the analog input, to satisfy the blinking rate requirement.

2. Suppose you are writing your own high levels functions to read and write the digital pins.
   a. Write 9 “#define” macros for PINB,C,D, PORTB,C,D and DDRB,C,D, so that you can use them in part b.
   b. Write the C code to set up the PIN,PORT,and DDR registers for the following functions. Be sure to consider all possible input cases,ranges, etc.

   WriteGPIOhigh(int port, unsigned char mask) //port=0 means PORTB, 1=PORTC, 2=PORTD
   //mask = 8 bit mask indicating which pins to write high. Ex. 0001 0010= Write bit 1 and 4 high

   WriteGPIOlow(int port, unsigned char mask) //port=0 means PORTB, 1=PORTC, 2=PORTD
   //mask = 8 bit mask indicating which pins to write low. Ex. 0010 0001= Write bit 0 and 5 low

   EnableGPIOpullup(int port, unsigned char mask) //port=0 means PORTB, 1=PORTC, 2=PORTD
   //mask = 8 bit mask indicating which pins to enable pullup.

   unsigned char readGPIOport(int port) //port=0 means PORTB, 1=PORTC, 2=PORTD
   //returns all 8 pins from port “port” by way of an unsigned char

   Are the above functions sufficient to completely implement GPIO functionality? If not propose modifications or additions necessary to completely implement full GPIO functionality.