ECE-231 Lab Assignment #5

Assigned Thursday 3/14/24
Due: 11:59 pm Saturday 3/30/24

Moodle References:

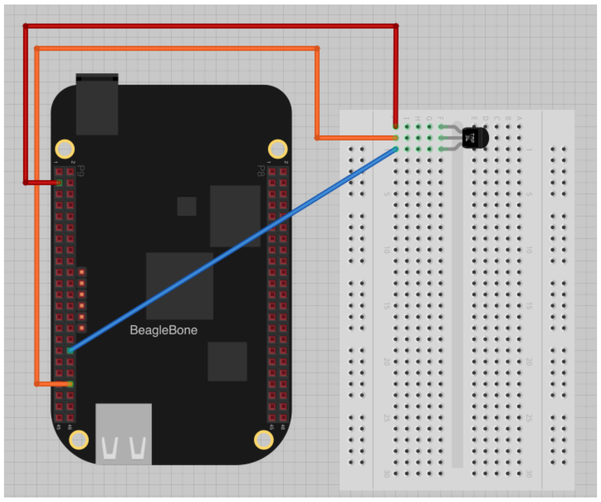
- Lecture 8 and 9 cover Interrupt concepts and programming
- Lecture 10 and 11 covers Thread concepts and programming
- Lecture 12 covers Sensing

A. Just like in Lab4, create a shell script file named “pwm_gen.sh”

1. Write code that configures the pin **P9_16** as a **pwm pin** and starts a PWM of period **0.1** second and **50%** duty cycle
2. Configure the pin **P8_8** as a GPIO **input pin**
3. Enable **rising edge** interrupt on pin **P8_8**

B. Just like in Lab4, take a jumper wire and connect pin P9_16 to P8_8 on your beaglebone

C. Interface the temperature sensor (TMP36), provided in your kit, to pin P9_40 as shown below:



Note: Make sure that you get the TMP36 the right way around.
● The blue lead is connected from the GNDA_ADC (P9_34) connection to the GND pin of the TMP36 temperature sensor
● The red lead is connected from pin 3 (P9_3) of the other connector (3.3V) to the positive supply pin of the TMP36
● The orange lead connects to pin P9_40 (AIN1). Note that only certain pins can be used as analog inputs.

D. Develop a C program that meets the following requirements:

4. Declare a global **buffer array** of size 10 as a **struct sensor_type** as follows:

```
struct sensor_type {
     struct timespec timestamp;
     double celcius_temperature;
}
```

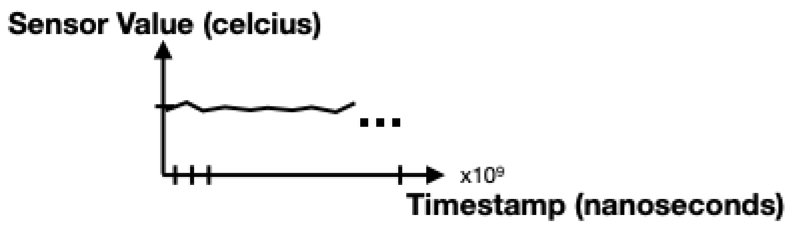
   Declare the buffer array with struct sensor_type as follows:

```
struct sensor_type buffer[10];
```

   To access variables inside the struct sensor_type, use:

```
buffer.timestamp.tv_sec
buffer.timestamp.tv_nsec
buffer.celcius_temperature
```

5. Create a new thread function named **inputThread(..)**, and inside this thread,
   a. Use Linux **epoll** to configure interrupt settings
   b. Use **epoll_wait(..)** to wait for the interrupt
   c. When a rising edge interrupt occurs on pin P8_8,
      i. take a timestamp using **clock_gettine(CLOCK_MONOTONIC,..)**, and store the struct timespec timestamp in the declared **buffer array**
      ii. read the temperature sensor data from **P9_40 (AIN1)** in celcius, and store this celsius value in the declared **buffer array**
   d. Store **10** data points in the buffer array using the steps above
   e. Make appropriate use of Mutex inside the thread
6. Create a second thread function named **outputThread(..)**, and inside this thread,
   a. Create a new file named “**firstname_lastname_sensordata.txt**”
   b. read the data from the buffer array, and write these data points in the newly created txt file using **fprintf(..)**. Also print these data points to the command line using **printf(..)**
   c. Make sure you read the data in the same order as it was written to in the buffer, via appropriate use of Mutex inside the thread.
7. Gather 50 data points using the steps above, and print every data point to the terminal as well as save it to your file. You should have 50 data points in a duration of 5seconds
8. Plot a graph of the sensor values and timestamps saved in your file. The graph should look like this.
   Use any graph plotting tool of your choice: python, excel etc.



E. Write a makefile and use it to compile your C program

F. Run your shell script, then run the C program binary, and observe the terminal output

Notes:

- This lab leverages part of your Lab2 and Lab4
- You have to use your beaglebone for this assignment
- **This is an individual assignment: you must write your own code and do not share**
- Read the instructions carefully multiple times to understand the program requirements and to produce the desired outcome
- The lecture material supporting this assignment has already been covered in the class
- The TAs will support you during lab hours

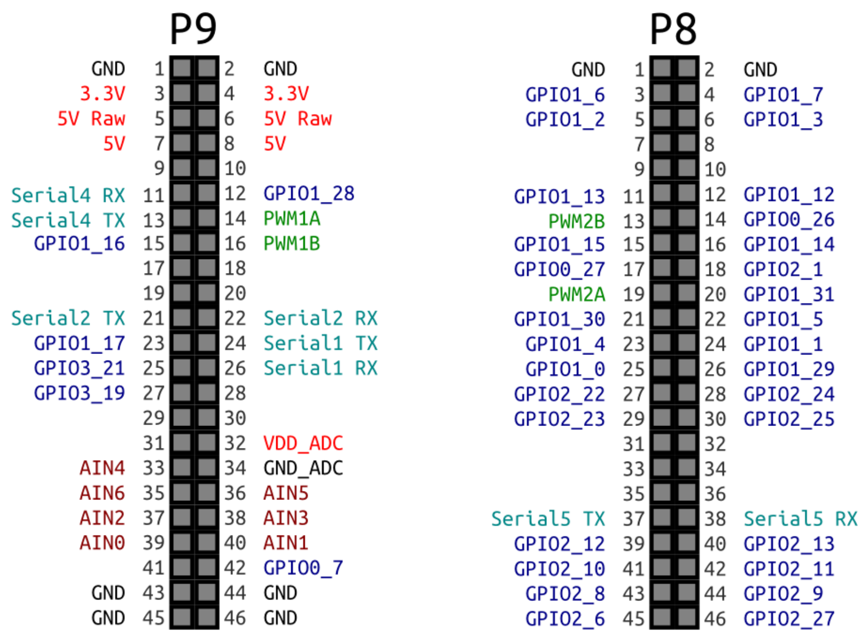
What to turn in:

- By the deadline, upload to Moodle the following list of files:
    - C source code file
    - Makefile
    - firstname_lastname_sensordata.txt file
    - Plotted graph of data
    - Video recording that shows your beaglebone with jumper wire and sensor connections, and the terminal output printing the sensor data and timestamp

Reference Material: Temperature Sensing using Analog pins:

The Beaglebone Black provides seven analog ports labeled AIN0 through AIN6. These are analog-to-digital converter (ADC) inputs: ADC uses 12 bits with a sample rate of 200 KSPS (Kilo Samples Per Second). There is an interface associated with each AIN input. The “in_voltage6_raw” corresponds to AIN6, “in_voltage5_raw” corresponds to AIN5 and so on. Refer to lecture 12 for more details.

It is important to make sure that the sensor operates within the appropriate ranges of temperature and voltage as follows:
1. The temperature range for the TMP36 is -50^{o}C to 280^{o}C
2. The analog inputs of the Beaglebone Black operate at 0 to 1.8V
3. The TMP36 has a theoretical maximum output of 3.3V
4. The TMP36 outputs 1.8V at 130^{o}C (366^{o}F)
5. There might be a potential problem to the Beaglebone Back if the voltage exceeds 1.8 V, so do not operate in spaces where the temperature is above 130^{o}C.



P9

GND	1	2	GND
3.3V	3	4	3.3V
5V Raw	5	6	5V Raw
5V	7	8	5V
	9	10	
Serial4 RX	11	12	GPIO1_28
Serial4 TX	13	14	PWM1A
GPIO1_16	15	16	PWM1B
	17	18	
	19	20	
Serial2 TX	21	22	Serial2 RX
GPIO1_17	23	24	Serial1 TX
GPIO3_21	25	26	Serial1 RX
GPIO3_19	27	28	
	29	30	
	31	32	VDD_ADC
AIN4	33	34	GND_ADC
AIN6	35	36	AIN5
AIN2	37	38	AIN3
AIN0	39	40	AIN1
	41	42	GPIO0_7
GND	43	44	GND
GND	45	46	GND

P8

GND	1	2	GND
GPIO1_6	3	4	GPIO1_7
GPIO1_2	5	6	GPIO1_3
	7	8	
	9	10	
GPIO1_13	11	12	GPIO1_12
PWM2B	13	14	GPIO0_26
GPIO1_15	15	16	GPIO1_14
GPIO0_27	17	18	GPIO2_1
PWM2A	19	20	GPIO1_31
GPIO1_30	21	22	GPIO1_5
GPIO1_4	23	24	GPIO1_1
GPIO1_0	25	26	GPIO1_29
GPIO2_22	27	28	GPIO2_24
GPIO2_23	29	30	GPIO2_25
	31	32	
	33	34	
	35	36	
Serial5 TX	37	38	Serial5 RX
GPIO2_12	39	40	GPIO2_13
GPIO2_10	41	42	GPIO2_11
GPIO2_8	43	44	GPIO2_9
GPIO2_6	45	46	GPIO2_27