School of Electrical Engineering and Computer Science

CptS466: Embedded Systems

Fall 2021

Project 6 (P6)

Code & Report Due: 12/03/2021 @ 11:59pm (Canvas)

Demo Due: 12/03/2021 in the class

1. Preparation

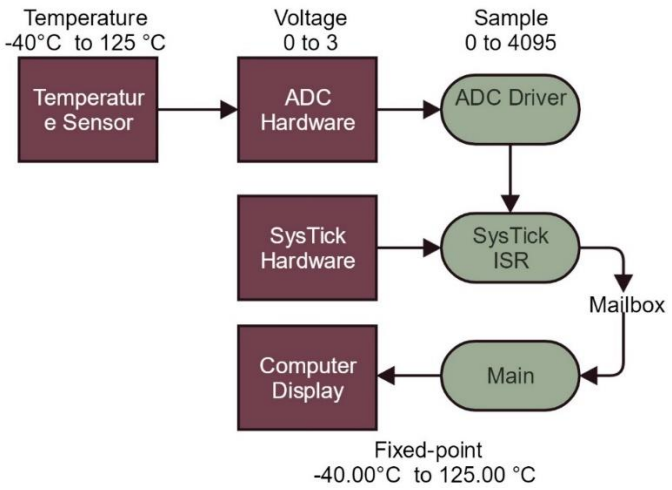
You will need the LaunchPad, Bluetooth module, TMP36-analog temperature sensor, LED, audio jack, breadboard, headphone or speaker, some resistors, and some wires. In addition, you will need a terminal application such as CoolTerm to perform serial port communication on a PC.

2. Project Description

In this project, you will design a smart home safety system. Your software will use the 12-bit ADC built into the microcontroller. The ADC module will be programmed to sample the sensor at 40 Hz using SysTick interrupts. You will write a C function that converts the ADC samples into temperature, with units of 0.01°C. The data stream will be passed from the ISR into the main program using a mailbox, and the main program will output the data to a computer. In this project, the TMP36 temperature sensor serves as our analog sensor. The default ADC channel is AIN1, which is on PE2. The TM4C123 ADC will convert voltage into a 12-bit digital number (0 to 4095). This ADC is a successive approximation device with a conversion time on the order of several microseconds. The datasheet for the TMP46 temperature sensor can be found under files/Resources in Canvas. The datasheet will help you develop the function to read temperature values.

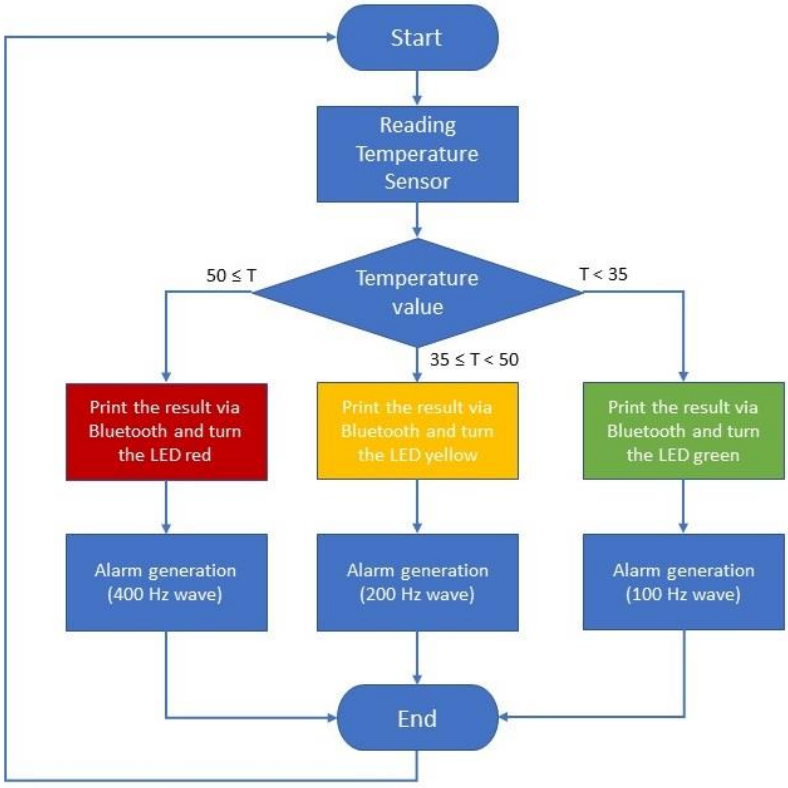
The following Figure shows a possible data flow graph of the system. Dividing the system into modules allows for concurrent development and eases the reuse of the code. Each software module needs to have its own code/routine and a corresponding header file. The code should contain the actual implementation, and the header file needs to have the prototypes for public functions. For instance, you may include the SysTick initialization, SysTick ISR, and the main program in the main.c file. The ADC module will

consist of the ADC.c and ADC.h files. The UART files also can be used similar to your previous project on serial communication.



You need to make the temperature resolution and accuracy as good as possible using the 12-bit ADC. The temperature resolution is the smallest change in temperature that your system can reliably detect. In other words, if the resolution is 0.01°C, and the temperature needs to change from 1.00°C to 1.01°C, then your device would be able to recognize the change. The resolution will depend on the amount of electrical noise, the number of ADC bits, and the resolution of the output display software. Accuracy is defined as the absolute difference between the actual temperature and the value measured by your device. Accuracy is dependent on the same parameters like resolution, but it is also dependent on the reproducibility of the sensor. Long-term drift, mechanical vibrations can also affect the accuracy.

After reading the temperature sensor outputs, you will add additional functionalities, both in hardware and software. The smart home safety system includes sensors and actuators. Your software will manage a temperature sensor using ADC (Analog to Digital Conversion). The system will produce an alarm by generating a sound using DAC (Digital to Analog Conversion) for different situations. The system continuously reads the temperature sensor to ensure the safety of the home. The flowchart below shows different processes of the smart home safety system. For instance, if the temperature is above 50° of Celsius, we will print out the temperature value on the serial terminal and turn the multicolor LED ‘Red’. We also generate a 400 Hz sine wave for the alarm purpose.



Here are some additional requirements.

- You will use the default settings of the Bluetooth module. The baud rate is 9600, 8-bit word length, no parity bits, one stop bit, FIFOs enabled. The password of the module is 1234 if you have not changed it.
- Use 80 MHz as the system clock.
- The temperature sensor data will be displayed via Bluetooth module on your computer only when the sensor has a new value.
- You will need to design a 4-bit DAC and then generate a sine wave accordingly. The safety condition includes generating three sine waves with three different frequencies (i.e., 100 Hz, 200 Hz, and 400 Hz).

3. Experiment

The temperature threshold values shown in the above flowchart are example values used in a real-world situation. You may change these values depending on the experimental data collection conditions that are available to you. After your design is complete and tested, conduct these experiments: Collect the sensor data (samples collected from the temperature sensor) in three different environments of your choice with potentially different temperature levels (e.g., outdoor, indoor, building, room, etc.). Each data collection

situation is expected to activate one of the alarm types (with different frequencies). You can decide to collect data in more than three environments and conditions and set appropriate threshold values that trigger the alarms. Make sure that your system has generated three different alarms as a result of your experiments. For each data collection scenario, collect the data for approximately 1 minute and save the collected data in three different files (in plain text format).

4. What to submit?

Submit a .zip file with the following content:

- Your well commented C source code implements the application described above.
- A report that documents your development process, explanation of the experiments conducted, discussion of your observations, and a discussion of your design decisions.
- Data files that contain your experimental data.

Show a demo of your application in the class on the due date. During your demo, discuss with the instructor what threshold values you chose for safety monitoring. During the demo, you should show how your system generates waveforms of different frequencies.

5. Report

In a separate written document (in Word or PDF), compile sections A through C as follows:

A: System Design. Briefly, in one paragraph, describe your high-level design. Next, discuss any specific observations and any difficulties you ran into while designing or testing the system. Finally, discuss various parameters of your ADC (precision, resolution, range, etc.).

B: *Experiment.* Discuss how you conducted the experiments, what environments, and any challenges associated with designing a realistic experiment.

C: Results. Plot the data from your experiments on a graph with all three environments included in the same graph. You can use any software for this purpose (e.g., Excel, MATLAB, Python, etc.).

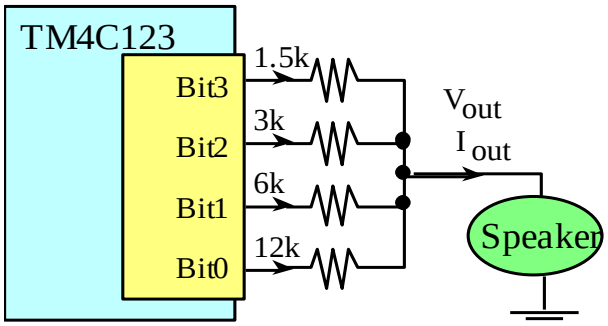
6. Grading

Assume that the whole assignment is worth 200 points.

- 70 pts for well commented and correct C source code satisfying the project requirements.
- 70 pts for report.
- 60 pts for demo.

7. Appendix A – Example System Schematic

A 4-bit DAC can be designed with the below circuit. Note that to make a 3k resistor, you can connect two 1.5k resistors in series or three 1k resistors in series.



You can interface the temperature sensor based on the below schematic. PE2 is ADC channel 1, which is a potential pin that you can use.