EE 472 Lab 1
Introducing the Lab Environment

Jonathan Ellington
Patrick Ma
Jarrett Gaddy

Contents

List of Tables

List of Figures

1 ABSTRACT

In this lab the students are to take on the role of an embedded system design team. They will design an exciting new medical instrument to monitor various patient metrics. When the device finds metrics are out of the acceptable range, the user will be notified, thus saving them from potential health risks. The students must first layout the design for their system using various design tools, then they must implement the system in software. Finally the students must test their system to make sure that it is ready to start saving lives.

2 INTRODUCTION

Brief introduction and overview of the purpose of the lab and of the methods and tools used.

The students are to design an embedded system on the Texas Instruments Stellaris EKI-LM3S8962 and EE 472 embedded design testboard. The design must implement a medical monitoring device. This device must monitor a patient's temperature, heart rate, and blood pressure, as well as its own battery state. The design must indicate when a monitored value is outside of a specified range by flashing an LED on the test board. When a value deviates even further from the valid range an alarm will sound. This alarm will sound until the values return to the valid range or the user acknowledges the alarm with a button. The values of each measurement will also be printed to the oled screen.

The design will be tested to verify proper behavior on alarm and warning notifications. In addition the implementation will be tested by measuring the amount of time that each of the 5 program tasks running the instrument take to execute. These tasks are mini prograams that each handle a part of the instruments purpose.

3 DISCUSSION OF THE LAB

3.1 Design Specification

3.1.1 Specification Overview

The entire system must satisfy several lofty objectives. The final product must be portable, lightweight, and internet enabled. The system must also make measurements of vital bodily functions, perform simple computations, provide datalogging functionality, and indicate when measured vitals exceed given ranges, or the user fails to comply with a prescribed logging regimen.
At the present time, only two subsystems must be produced: the display and alarm portions. Additionally, the system must demonstrate the ability to store basic measurements.

The initial functional requirements for the system are:

- Provide continuous sensor monitoring capability
- Produce a visual display of the sensor values
- Accept variety of input data types
- Provide visual indication of warning states
- Provide an audible indicator of alarm states

3.1.2 Detailed Specifications

For this project, these requirements have been further specified as follows:

The system must have the following inputs:

- Alarm acknowledgment capability using a pushbutton
- Sensor measurement input capability consisting of:
  - Body temperature measurement
  - Pulse rate measurement
  - Systolic blood pressure measurement
  - Diastolic blood pressure measurement

The system must have the following outputs:

- Visual display of the following data in human-readable formats:
  - Body temperature
  - Pulse rate
  - Systolic blood pressure
  - Diastolic blood pressure
  - Battery status
- Visually indicate warning state with a flashing LED
- Visually indicate a low battery state with an LED
- Audibly indicate an alarm state using a speaker

The initization values, normal measurement ranges, displayed units, and warning and alarm behaviors for each vital measurement are given in Table **??**. The sensors must be sampled every five seconds.

Measurement	Units	Initial Value	Min. Value	Max. Value	Warning Flash Period
Body Temperature	C	75	36.1C	37.8C	1 sec
Systolic BP	mm Hg	80	-	120 mmHg	0.5 sec
Diastolic BP	mm Hg	80	-	80mmHg	0.5 sec
Pulse Rate	BPM	50	60 BPM	100 BPM	2 sec
Remaining Battery	%	200	40 %	-	Constant

A measurement enters a warning state when its value falls outside the stated normal range by 5%. An alarm state occurs when any measurement falls outside its stated normal range by 10%.

Additionally, the system must be implemented using the Stellaris EKI-LM3S8962 ARM Cortex-M3 microcomputer board, The software for the system must be written in C using the IAR Systems Embedded Workbench/Assembler IDE.

3.1.3 Identified Use Cases

Taking the functional requirements listed above, several use cases were developed. A Use case diagram of these scenarios is given in Figure **??**. Each use case is expanded and explained below.

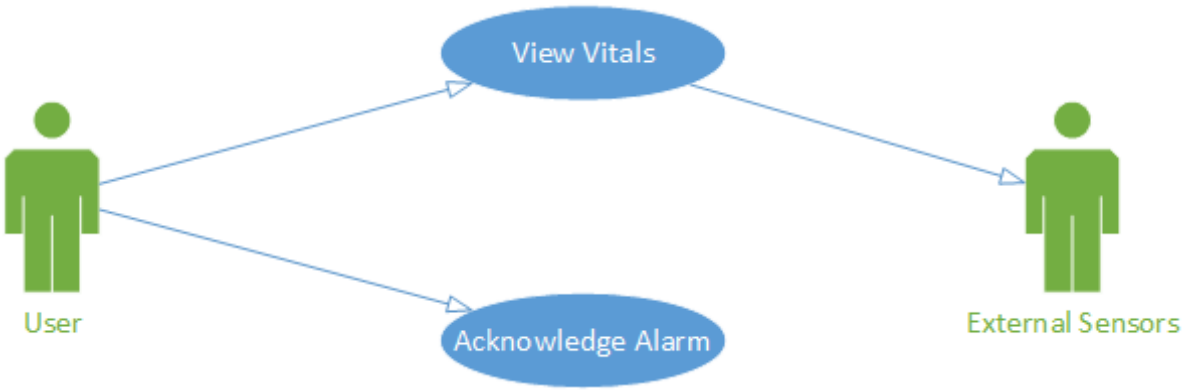

Figure 1: Use case diagram

Use Case #1: View Vital Measurements
In the first use case, the user views the basic measurements picked up by the sensors connected to the device.
During normal operation, once the device is turned on by the user, the system records the value output by each sensor. This raw value is linearized and converted into a human-readable form. Finally, this value is displayed onscreen.

Three exceptional conditions were identified for this use case:

One or more of the expected sensors is not connected: If this occurs, the measurements taken by the device may be erratic. At the present moment, no action will be taken in such events. Later revisions may address the issue

A measured value is outside 5% of the specified normal range: In this case, a warning signal will flash as an indication of the warning condition

A measured value falls outside 10% of a specified "normal" range: In this case, an audible alarm will sound to indicate the alarm condition

Use Case #2: Acknowledge Alarm
In the second case, the system is in an alarm state. The user acknowledges the alarm condition by pressing a button.
Upon pressing the button, the system silences the audible alarm. Any visual warnings continue to flash during the silenced period. If a specified amount time passes and the sensor reading(s) continue to maintain an alarmed state, the audible alarm will recommence.

No exceptional conditions were identified for this use case.

3.2 Software Implementation Jon

3.2.1 Top level design Jon

The design process began by identifying the use cases and actors involved with the system. In the specified system, the user interacts with the system in one of two ways:

1. The user can view their vitals

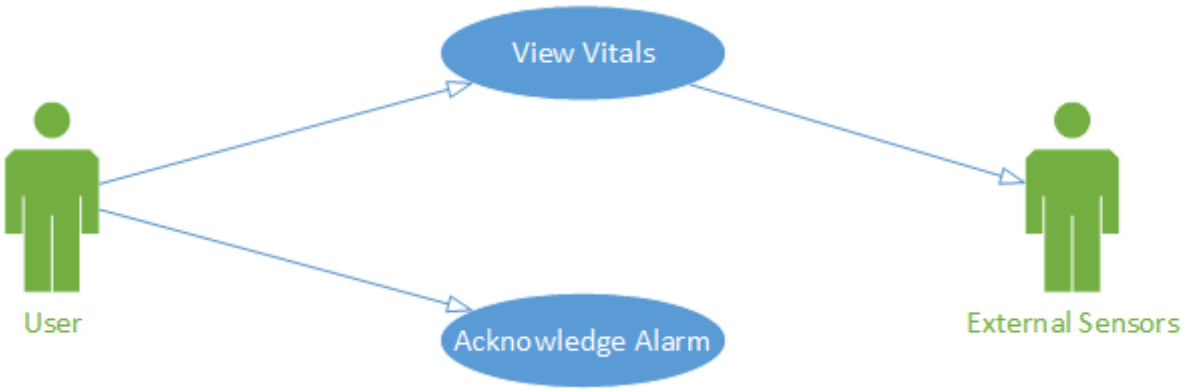

Figure 2: Use-Case Diagram

2. The user will acknowledge an alarm condition to silence the alarm

In order for a user to view their vitals, the system will have to interact with some external sensors. Specifically, the system will interact with blood pressure, temperature, and pulse rate sensors. A graphical depiction of this is shown in Figure 2.

After understanding how the user would interact with the device, the system was functionally decomposed into high-level blocks as shown in Figure 3. The main system control is located in the CPU, which controls all data flow into and out of the peripheral devices. The OLED displays the user's current vitals including blood pressure (systolic and diastolic), temperature, and pulse rate. In the future external sensors will be added, but for now the values are simulated using the CPU. The CPU also controls three LEDs colored green, yellow, and red. These LEDs are used to inform the user on the current state of their vitals as well as the state of the device. Under normal circumstances, the green LED will be lit. If the users' vitals fall outside of a specified range, the red LED will flash at a specified rate, dependant on which vital is out of range. If the battery is low, the yellow LED will be illuminated.

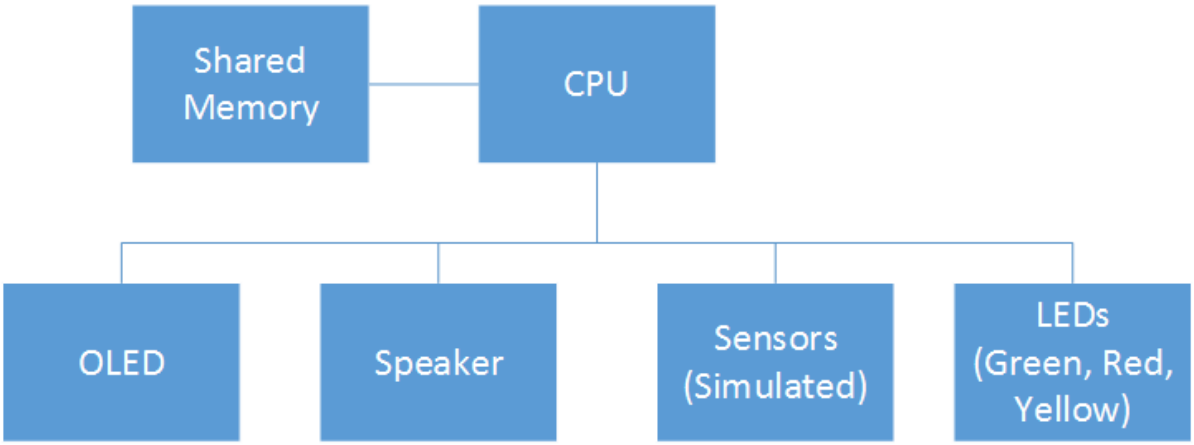

Figure 3: Functional Decomposition

Next, the system architecture was developed (Figure 4). At a high level the system works on two main concepts, the scheduler and tasks. Tasks embody some sort of work being done, and the scheduler is in charge of determining the speed and order in which the tasks execute. The system has several tasks, each with their own specific job. For modularity reasons, each task should have the same public interface and the scheduler should be able to run each task

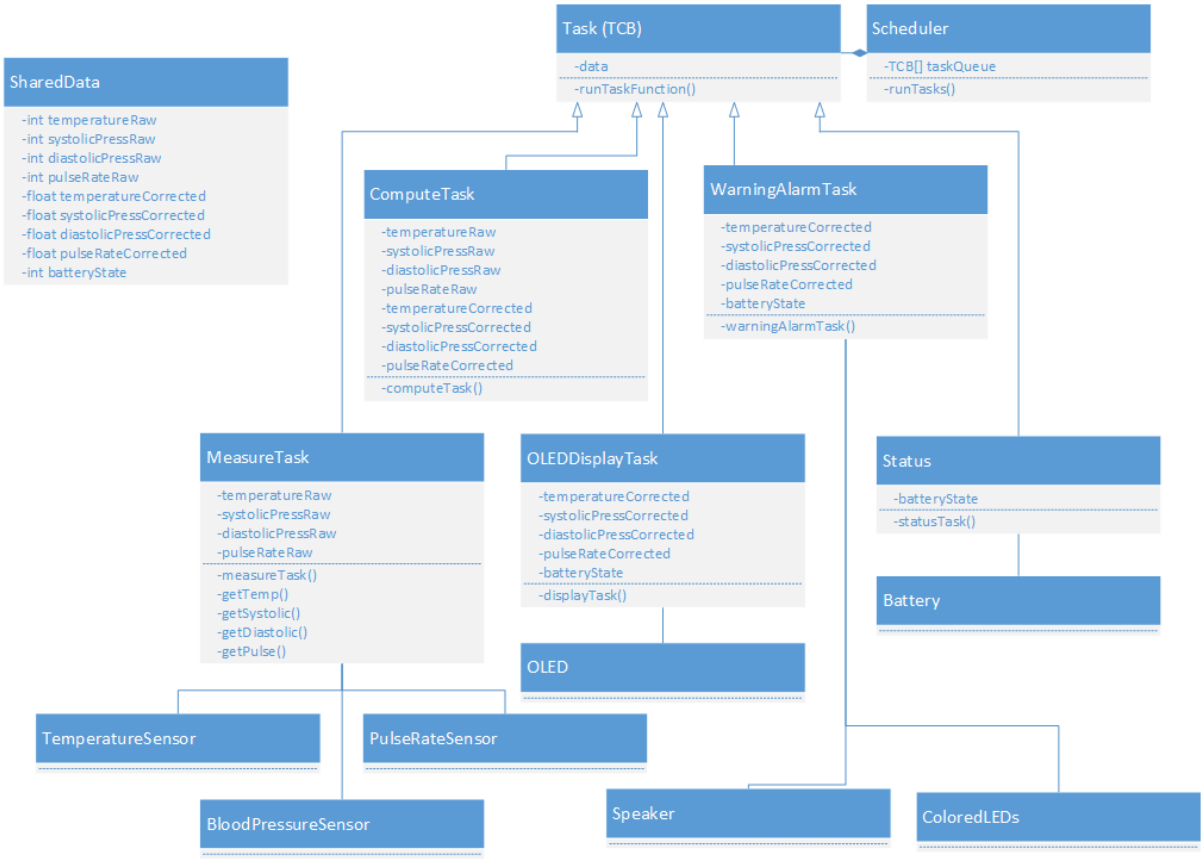

Figure 4: System Architecture Diagram

regardless of that specific tasks job or implementation. Thus the task concept is abstracted into a Task Control Block (TCB), and the scheduler maintains a queue of TCBs to run. The TCB abstraction is shown in Figure 4 using inheritance, and the fact that the scheduler has a queue of TCBs is shown with composition. The core functionality of the system was divided into the following five main tasks:

- **Measure Task** - In charge of interacting with the blood pressure, temperature, and pulse sensors (simulated)
- **Compute Task** - Converts sensor data into human readable format
- **Display Task** - Displays the measurements on the Stellaris OLED
- **Warning/Alarm Task** - Interacts with the red, yellow, and green LEDs, as well as the speaker to annunciate warning and alarm information
- **Status Task** - Receives battery information from the device

Each of these tasks interact using the shared data shown in Figure 4.

After developing the system architecture, the design needed to be translated into the C programming language. The design manifested in a multifile program consisting of the following source files:

- **globals.c/globals.h** - Used to define the Shared Data used among the tasks
- **schedule.c/schedule.h** - Defines the scheduler interface and it's implementation

- **timebase.h**

3.2.2 low level design Jarret

implementation details here. Tasks, scheduler, etc. Control diagram goes here, activity diagram, etc.

4 PRESENTATION, DISCUSSION, AND ANALYSIS OF THE RESULTS

Based upon the execution of your design, present your results. Explain them and what was expected, and draw any conclusions (for example, did this prove your design worked).

In addition to a detailed discussion and analysis of your project and your results, you must include all the answers to all questions raised in the lab.

4.1 results

The project was completed and demonstrated on January 29, 2014.

Demonstration of the system to the interested parties showed that the system met the requrements initially presented at the onset of the lab project. Testing of the system prior to demonstration also verified that the system met the specifications listed in Section 3.1.
Additionally, during the demonstration, to show that several warning features worked as expected, source code was temporarily modified to speed up the progression of the system through various warning states. These changes were simple to execute and caused the desired effect on the system without causing unwanted abberrant behavior. Reverting the source code was similarly intuitive. During the actual coding and implementation of the design, remarks were made several times about the ease of execution during that phase of the project. After the initial high level Design phase, very few changes were made, or required, to the functional design or system architecture.

Using an oscilloscope, the runtimes of each task were empirically determined. The results are given in Table 5.

Task	Runtime (μs)
Measure	23.4
Compute	55.4
Display	22900.0
Warning	27.4
Status	5.6

Figure 5: Empirically determined task runtimes

Answers to the last three questions in the list of items to include in the project report:
You don't find the stealth submarine. That's why they are so expensive; at that cost, you take great pains to never lose one.
A helium balloon always rises. It just rises upside-down.
If you really managed to lose the stealth submersible, you first have to tell the government, which will deny it has any stealth submersibles, then you have to comb the seven seas until your comb hits the sub.

4.2 discussion of results Jon

4.3 Analysis of any Errors

The project was completed without any residual errors or unsolved problems. See the following section for analysis of issues encountered while working on the project itself.

4.4 Analysis of problems and issues encountered and what efforts were made to identify the root cause of any problems Jarrett

State any problems you encountered while working on the project. If your project did not work or worked only partially, provide an analysis of why and what efforts were made to identify the root cause of any problems.

Some points to bring up: did not enable the GPIO bank (caused OLED display to not work), could not get switch to work (solved by understanding that switch required pull up). P or J can talk about design solutions that did not work. On the whole, we had problems with going too deep, too quickly.

5 TEST PLAN

To ensure that this project meets the specifications listed in section 3.1, the following parts of the system must be tested:

- Vitals are measured and updated every five seconds
- System properly displays corrected measurements and units properly
- System enters, indicates, and exits the proper warning state for blood pressure, temperature, pulse, and battery
- System enters and exits the alarm state correctly
- Alarm is silenced upon button push
- Alarm recommences sound after silencing if system remains in alarm state longer than slilence period

Additional tests to determine the runtime of each specific task are also required.

5.1 Test Specification Jon

Annotated description of what is to be tested and the test limits. This specification quantifies inputs, outputs, and constraints on the system. That is, it provides specific values for each.

Note, this does not specify test implementation...this is what to do, not how to do it.

5.2 Test Cases Jarrett

Annotated description of how your system is to be tested against the test limits Note, this does specify test implementation...this is not what to do, this is how to do it based upon the test specification.

6 SUMMARY AND CONCLUSION

6.1 Final Summary

The students began creating their medical instrument through a rigorous design process at different levels of detail. The students then continued work by implementing their design in C code for the ARM Cortex A3 microprocessor. Next the code was tested and debugged using the IAR workbench debugging tool, as well as visual queues programmed into the design. Finally, after verifying that the system worked as specified, it was presented to the instructor.

6.2 Project Conclusions

This project contained 3 major phases, the design, implementation, and testing steps. The students were immediately introduced to using the unified modeling language(UML) to design embedded systems. This is the first time many students will have used UML for system design which caused some confusion and difficulty. In the end through the use of the UML guidelines for design, the students were able to implement their system in code for the Texas Instruments Stellaris EKI-LM3S8962 much more quickly and with far fewer errors than if they had spent less time in the design phase of this project.

Effective design tools allowed the students to quickly implement their embedded system in C code for an ARM Cortex A3 processor, and move onto the testing phase of the project quickly. Unfortunately, while testing the students encountered a number of problems in using the pwm and general purpose input and output signals. After consulting the documentation for the Stellaris kit and solving their input/output problems, they began testing thri design using visual and audio queues, the IAR embedded workbench debugger, and a few specifically programmed debug features. After the results of the testing verefied the design to be working correctly, the students proceeded to present their medical instrument to their instructor.

A BREAKDOWN OF LAB PERSON-HOURS (ESTIMATED)

Person	Design Hrs	Code Hrs	Test/Debug Hrs	Documentation Hrs
Patrick	15	7	2	9
Jarret	x	x	x	x
Jonathan	x	x	x	x

By initializing/signing above, I attest that I did in fact work the estimated number of hours stated. I also attest, under penalty of shame, that the work produced during the lab and contained herein is actually my own (as far as I know to be true). If special considerations or dispensations are due others or myself, I have indicated them below.

B SOURCE CODE

Source code for this project is provided below.

B.1 The first part

B.2 The second part