EE 472 Lab 1
Introducing the Lab Environment

Jonathan Ellington
Patrick Ma
Jarrett Gaddy

Contents

List of Tables

List of Figures

1 ABSTRACT

The abstract should provide a brief overview of the report. It should provide a summary of the main specific points for the introduction, the main tests and experiments, the results, and the conclusions. It is called an abstract because you can literally ”abstract” sentences from the other sections.

Once again, this is not a narrative of your experiences as you executed the design. The abstract should mirror (albeit in a very condensed way) the content of your report.

2 INTRODUCTION

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

3 DISCUSSION OF THE LAB

This section should include the following:

3.1 Design Specification Patrick

In this subsection you will textually describe your client’s requirements. What does he or she need in the project you are developing. If you are incorporating extra features or capabilities, please describe them clearly in this section.i

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:

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

The system must have the following outputs:

- Display of measured vitals and battery status
- Visual signals for three battery states
- Visual signal of low battery state
- Audio signal of alarm state

The system must have the following inputs:

- Alarm acknowledgement capability
- Sensor measurement input capability

Overall summary description of the module - 2-3 paragraphs maximum (explanation of use cases goes here)

Specification of the public interface to the module

- Inputs
- Outputs
- Side effects

Psuedo English description of algorithms, functions, or procedures

Timing constraints

Error handling

Animal	Description	Price ($)
Gnat	per gram	13.65
	each	0.01
Gnu	stuffed	92.50
Emu	stuffed	33.33
Armadillo	frozen	8.99

Table 1: Example table.

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
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 1.

After understanding how the user would interact with the device, the system was functionally decomposed into high-level blocks as shown in Figure 2. 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

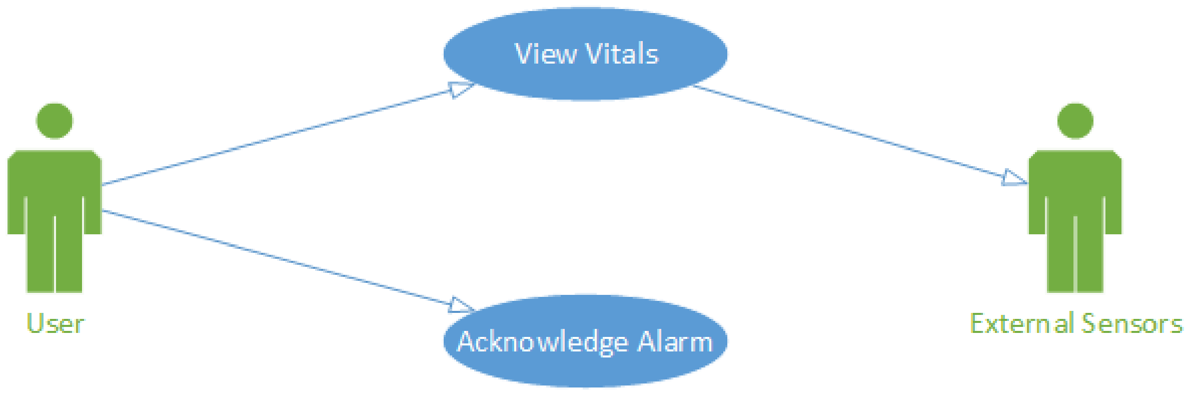

Figure 1: Use-Case Diagram

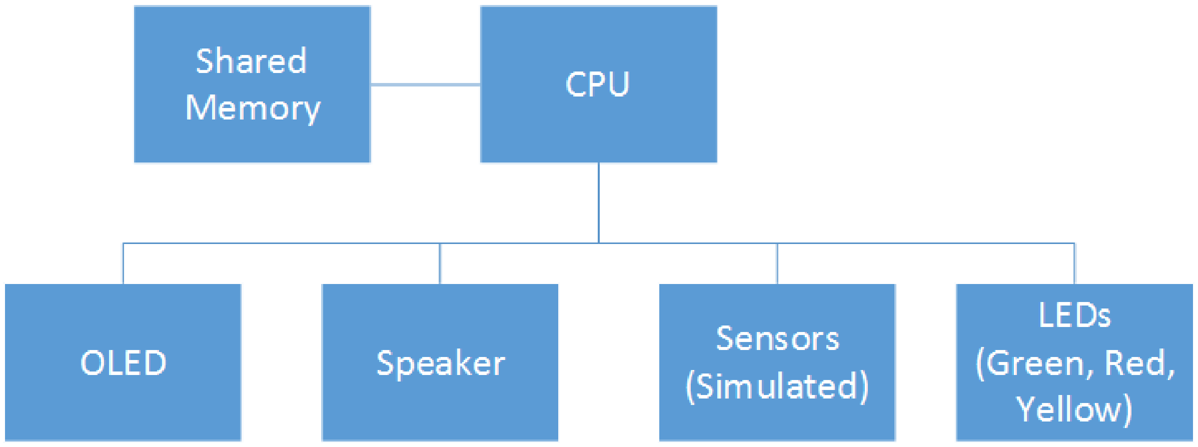

Figure 2: Functional Decomposition

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.

Next, the system architecture was developed (Figure 3). 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 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 3 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

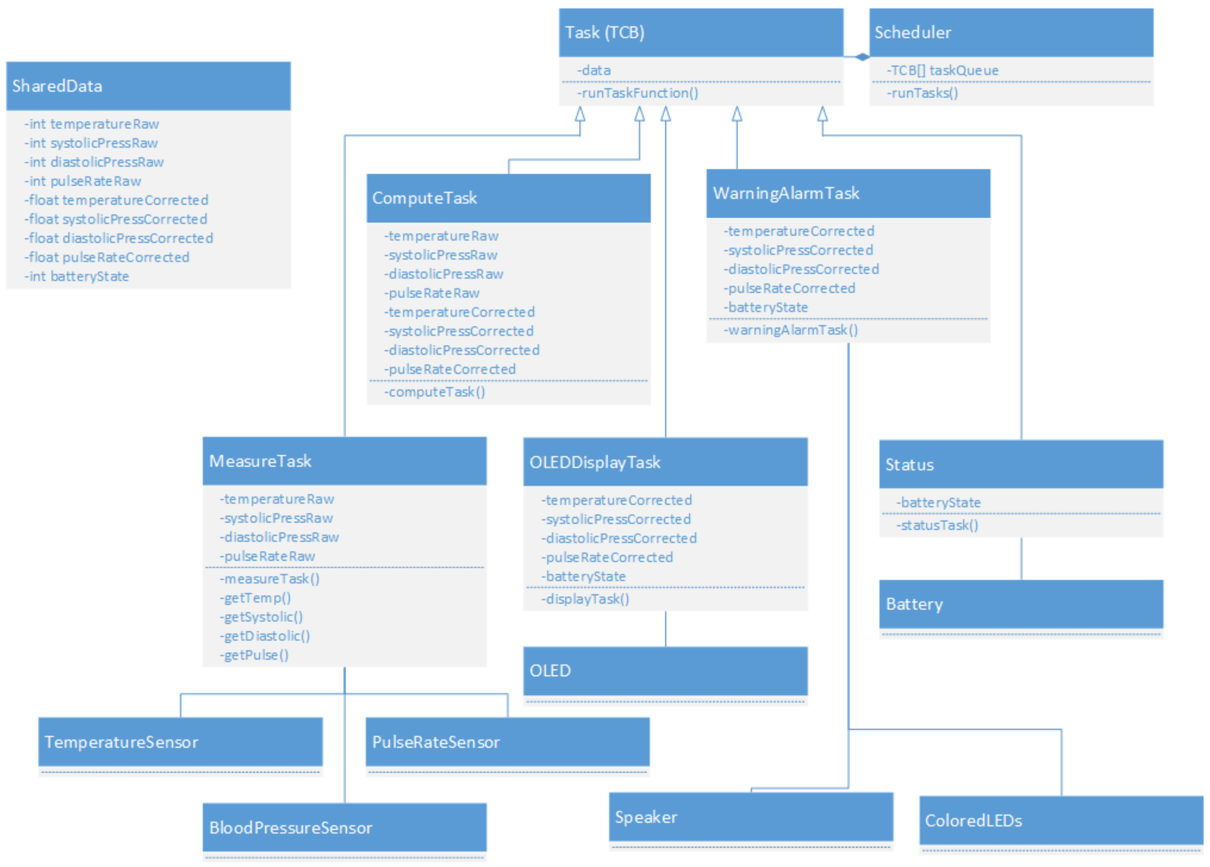

Figure 3: System Architecture Diagram

- **Status Task** - Receives battery information from the device

Each of these tasks interact using the shared data shown in Figure 3.

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, as well as the TCB structure
- **timebase.h** - Defines the timebase used for the scheduler and tasks

Each task also has it's corresponding ".c" and ".h" file (for example, measure.c and measure.h).

The TCB structure that the scheduler uses must work for all tasks, and must not contain any task-specific information. Instead, the TCB consists of only a void pointer to the tasks data, and a pointer to a function that returns void and takes a void pointer, as follows:

```
struct TCB {
  void *taskDataPtr;
  void (*taskRunFn)(void *);
}
```

Leaving out the type information allows the scheduler to pass the task's data (*taskDataPtr) into the task's run function completely unaware of the kind of data the task uses or how the task works.

For increased modularity, the data structure used by each task was not put in the task's header file. Instead, the structure was declared within the task implementation file, and instantiated using a task initialization function. In the header file, a void pointer pointing to the initialized structure is exposed with global scope, as well as the task's run and initialization functions.

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 patrick

4.2 discussion of results Jon

4.3 Analysis of any Errors Patric

This one is obvious. Do this section as appropriate. If it improves the flow, it does not need to be a separate section and may be included in the presentation, discussion, and analysis of the results. However, it will still be graded separately and must be present.

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 PATRICK

Overall summary of what needs to be tested to ensure that your design meets the original requirements, 2-3 paragraphs maximum unless specified otherwise

5.1 Test Specification

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

You should know these sections very well, no need to explain. Note, however, that they are two different sections. The summary is just that, a summary of your project. It should loosely mirror the abstract with a bit more detail. The conclusion concludes the report, potentially adds information that is often outside the main thrust of the report, and may offer suggestions or recommendations about the project.

6.1 Final Summary

6.2 Project Conclusions

A SOURCE CODE

Source code for this project is provided below.

A.1 The first part

A.2 The second part