

P442 Alarm Clock Profile Report

Authors: Ching Ng, Brian Rak

Customer: Bryce Himebaugh

Overview

This report outlines the high level design for the P442 Alarm Clock (from now on referred to as “clock”, or “the clock”) for customer Bryce Himebaugh. This report will touch on design considerations, background, high level technical details, assumptions, and finally the deliverables.

The objective of the report is to inform the reader of design considerations in creating Alarm Clocks from scratch. In addition, hardware and software requirements will be outlined and specified. Finally design and functional requirements will be specified and estimated.

Background

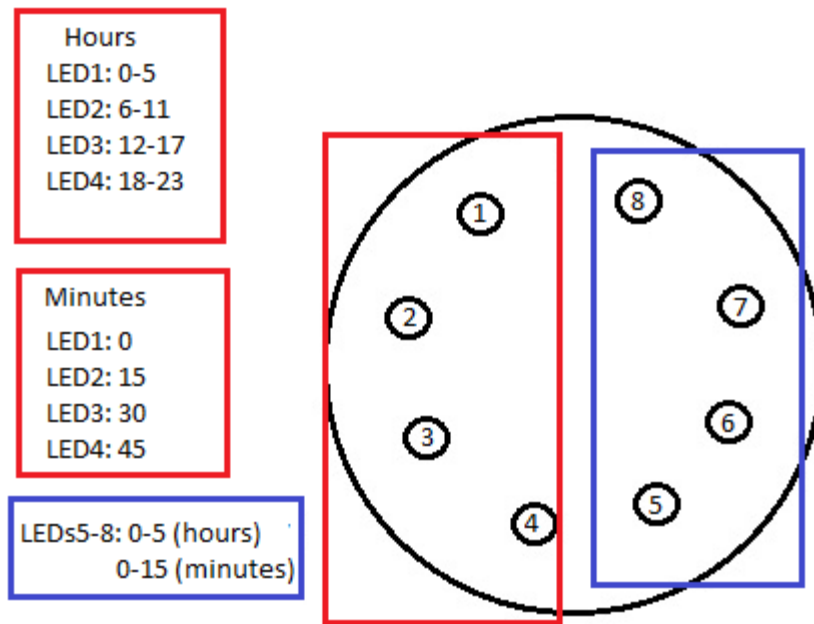
The requirements of the clock are as follows:

Design Requirements
Must use real time clock on the STM32
Must use LEDs as the time display
Must use a General Purpose Input Output (GPIO) Buzzer
Must be possible to set alarm directly from the board
Must be possible to set the time/date from a workstation when the board is connected via USB
Make a reasonable effort to prioritize battery life
May use additional GPIO buttons
Must log: button presses, movement (gyro), time set, alarm set, alarm activation

To accomplish these objectives the authors will utilize pair programming technique to minimize errors and further facilitate the debugging process. The clock will be written in the C programming language and compiled using the arm-none-eabi compiler. No operating system will be used on the board itself, but the Red Hat Enterprise Linux and command line tools will be used for development and downloading to the clock. Also, Python or Bash (TBD), will be used to create the command line application to set the clock.

The clock that will be used has the capacity to store seconds, hours, days of the week, days of the month, month, and year. The clock does not provide an alarm by itself, but instead must be programmed and wired correctly for use.

Below is a (crude) implementation of the clock.



The clock uses an interesting and concise numbering system. The left side is represented in quadrants. Hours are broken up in quadrants of count 6, and minutes are broken up into quadrants of count 15 since they are in base 24 and base 60 respectively. The lights light up according to their respective quadrant in the legend above.

The right hand side of the clock is represented in binary up to 0xf. However, 0xe is the last value needed since 0xf will start a new quadrant in the quadrant category. Minutes and hours can be toggled or displayed concurrently. In the concurrent display mode, hours will be dimmed. Both approaches will be used since the user may not be able to tell the exact time due to overlapping of LEDs.

Examples of Time (Concurrent time mode, Discrete Time Mode displays fully lit lights):

20:43

LED 4 is dimmed (18) + LED 7 is dimmed (2) : LED 3 is lit (30) + LED 8, 6, 5 are lit (13)

2:10

LED 1 is dimmed (0) + LED 7 is dimmed (2) : LED1 is lit (0) + LED 5, 7 are lit (10)

Assumptions

We will use the following set of assumptions:

- Working STM32f30x board will be provided
- The arm-none-eabi compiler and debugger will be provided
- Appropriate GPIO Libraries and STM32f30x Libraries will be provided
- The customer, Bryce Himebaugh, will be open for consultation and clarification during normal business hours

- Peripherals, batteries, quartz clock and other necessary hardware not listed here will be provided
- The user is able to read basic binary format 0-f 0-1111 to read the display on the clock
- The user is able to hear the buzzer on the alarm clock
- The user will not expose the clock to hazardous environments or moisture
- The clock will not endure any *unreasonable* physical force on it
- The ARM board will provide a working serial port communicating at 9600 baud
- Customer is capable of using a Command Line Interface (CLI) to set the clock
- Customer is able to operate basic commands in a Linux environment to operate the clock
- Customer has Python installed
- The device will have continuous power via battery or USB
- The device's 9v battery will last no more than 6 months of continuous operation

Weight and size will be approximately a quarter pound and approximately the dimensions of the STM board. Wires will protrude from the board and it will be up to the customer to provide the casing for the board if desired.

There will be a cost that is no greater than the components of the board. The cost is based on market price of the STM32f30x board (basic model), STM Real Time Clock and Crystal, 1 Button, Less than 10 Male to Male wires. 1 9v battery, and 1 Universal Asynchronous Receiver and Transmitter (UART) plugin. There is no cost for the programmers due to lack of expertise in the field. Credit in the form of grades should be provided.

There will be a relatively small total development time of 1 week. The main development time of the drivers is estimated at 2 days. The application should be developed in two days. The serial connection script written in either bash or Python will be estimated at less than one day. Testing will last approximately 1 day. The customer will have the opportunity to test the application and clock on the last day of testing.

Setting the alarm

The users will be using two buttons to set the time for the alarm. The first button, BUTTON1, will be the user button and the second button, BUTTON2, will be the extra button that we add on breadboard. User can switch from the time display mode to set hour mode by holding down the blue user button. The alarm will begin with the LEDs blinking twice to alert the user they have entered the alarm set mode. The user should then set the hours with BUTTON2. The user should toggle between hours and minutes with BUTTON1. When finished, the user can hold down the user button to exit the alarm set mode and return to display mode.

Event Logger

An event logger will be included to log system events. The events logged are enumerated in the design requirements. The events are encoded as constants as below:

Event	Description
ERROR	An unknown event occurred
BUTTON_PRESS	The User button on the STM board was pressed
BUTTON2_PRESS	The button on the bread board was pressed
GYRO_MOVE	The Gyro moved over the threshold of 10 dgps
START_SET_TIME	The User has initiated the set time sequence
END_SET_TIME	The user has exited the set time sequence
SOUND_ALARM	The alarm has been triggered by the system
QUIET_ALARM	The alarm has been turned off by the user
START_SET_ALARM	The user has initiated the set alarm sequence
END_SET_ALARM	The user has exited the set alarm sequence

The current time (<TIME>) will be queried from the board's real time clock and recorded as:

<hour (24)>:<minute (60)>:<second (60)>,<month (12)>,<day (31)>,<year (9999)>

Therefore each entry in the log will take the form <EVENT>,<TIME>\n in CSV format.

Note: that the gyro values (of floating point type) will be logged as a separate line directly following the GYRO_MOVE tag. The format of the log is x: <float>, y: <float>, z: <float>\n

To avoid writing to the disk frequently, there will be a temporary buffer to store the data. The data will be stored in RAM and may be lost in the event of a power outage. After a certain fixed interval, defaulted at 10 seconds, the log will be dumped to the SD card with the attributes listed above. The file will be appended to the file "log.txt". If no file is present, it will be created before logging. If no SD card is present, the data will not be dumped to the file.

Objectives / Deliverables

The following Deliverables will be set:

Objective	Deliverable
Battery Life	Minimum battery life of 6 month based on usage assumptions.
Battery Availability	Design uses commercial 9v battery, available at any retail store selling batteries
Alarm Set	Alarm is able to be set via the user buttons
Clock Visibility	Clock is able to be viewed in two modes, concurrent and discrete using 8 LEDs
Clock Reset	Clock will able to be set and reset programmatically from a remote workstation tethered via USB
Clock Set	Clock will able to be set by user
Code Visibility	Project will have limited open source with the customer, closed source to the public

