PG5500 - Embedded Systems

Assignment 2 - Eivind Vegsundvåg

Introduction

The solution has the following features:

- Clock and alarm display over ST7735 clone
- Variable resistor volume knob
- Control stick
- Alarm reset button
- Very annoying sound, according to everyone ever.

A video of the solution in action can be seen here:
https://www.youtube.com/watch?v=ebcWG8KeeIo
The solution is an object oriented C++ solution where applicable.

Prerequisites

RTCLib(https://github.com/adafruit/RTClib)
Adafruit GFX Library(https://github.com/adafruit/Adafruit-GFX-Library)
Adafruit ST7735 Library(https://github.com/adafruit/Adafruit-ST7735-Library)
Adafruit Fritzing Library(https://github.com/adafruit/Fritzing-Library)

Wiring

The wiring is available as a Fritzing sketch. Note that many components were not readily available as Fritzing components, and that the pin wirings therefore seem utterly wrong in the sketch due to similar, but different components being used.

Components

ST7735 TFT Display

The ST7735 display is controlled by the AlarmDisplay service. This service supports the following functions:

- Full draw of time components
- Draw of single time components:
    - Day in month
    - Day in week
    - Month by name
    - Year
    - Hour

    - Minute
    - Seconds
    - Decorations(currently only time separators(:)
- Full draw of alarm components

Alarm components are only drawn in full due to major changes happening in that area on every change(background, numbers, unsetting etc).
Time components only draw the required area. Y positions are largely hard coded, but X positions are calculated by “Center left”, “Center” and “Center right” with offsets based on content length.

RTC1307 Clock

The RTC1307 is controlled by the TimeService service. It doesn’t do much - it simply wraps around RTCLib for abstraction.

Speaker

This simple speaker is controlled by the AlarmService. The AlarmService checks if the supplied current time is applicable for playing an alarm, and resets the alarm when the reset button is pressed. The alarm plays for up to five minutes. Given how annoying the sound is, that should be plenty of time to wake up.

Analog Stick

The control stick is controlled by the ControllerService. This service is capable of reading the functions of an analog stick, mapping movement on the Y axis to alarm adjustment, the X axis to alarm selection(hour/minute), and simple clicks on the stick.

General program flow

The Arduino application sets up the various components during the setup phase. Afterwards, it loops through them to take input and applies it to the correct services. The abstraction layers allows me to prevent some actions such as the alarm playing while setting the alarm. The loop cycle repeats 10 times per second.

Due to some issues with debugging during the start of development, I converted all the components in the Arduino application to pointers. This prevents eager instantiation of components before the Serial interface is ready, meaning the application should be quite simple to debug.

Unsolved issues

1. I initially intended to synchronize the time with time servers once per hour due to the inaccuracy of my RTC component. However, I could not get anything less than

garbage after the fifth status header of a HTTP request. I believe this is due to excessive memory usage combined with a limited serial buffer.

2. As is obvious in the video, the screen redraws the entire alarm area every cycle. This is due to me not quite figuring out how to reliably switch the backgrounds of the selected alarm components on and off. I had some solutions for this, but they largely resembled spaghetti code.