Alarm Clock

Custom Project Final Report

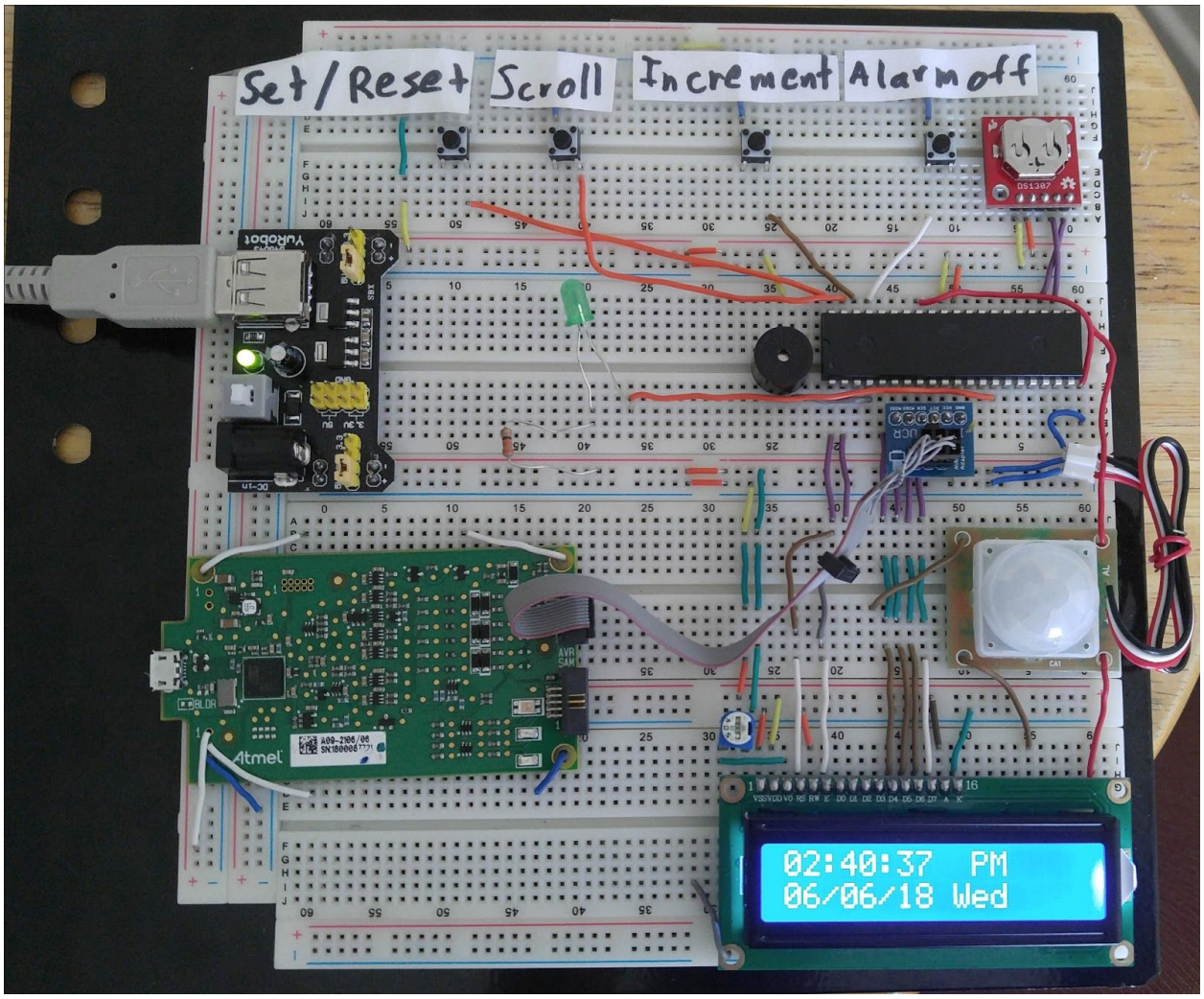
Spring 2018

Jose Peralta

Table of Contents

Introduction

This is an alarm clock with an addition for heavy sleepers. The display is an LCD screen. The RTC module keeps track of the current date and time. The screen displays the time, date, and extra settings as the user sets the alarm. 4 buttons control the alarm settings. A PIR motion sensor will act as an “annoying” snooze button. In order to turn off the alarm, the user must wave their hand over the sensor for 5 seconds while pressing the “Alarm Off” button.



User Guide

- Setting the Alarm:

    - Press Set/Reset. “Set Alarm Hour” displays. Press Scroll. Press or press and hold Increment to select desired alarm hour. Once reached desired hour, press scroll again.
    - “Set Alarm Minute” displays. Press Scroll. Press or press and hold Increment to select desired alarm minute. Once reached desired minute, press scroll again.
    - “Set AM or PM” displays. Press Scroll. Press Increment to select AM or PM. Press Scroll. “Alarm Set” displays.
- Turn off an active alarm: For 5 seconds, wave your hand back and forth above the motion sensor while pressing and holding Alarm Off at the same time.
- Cancel an upcoming alarm: If calendar is displayed, press Set/Reset twice. “Alarm Off” will display for 1 second, then go back to calendar
- Reset the Alarm: When setting alarm, press Set/Reset at any time to reset alarm settings. “Alarm Off” will display for 1 second, then go back to calendar

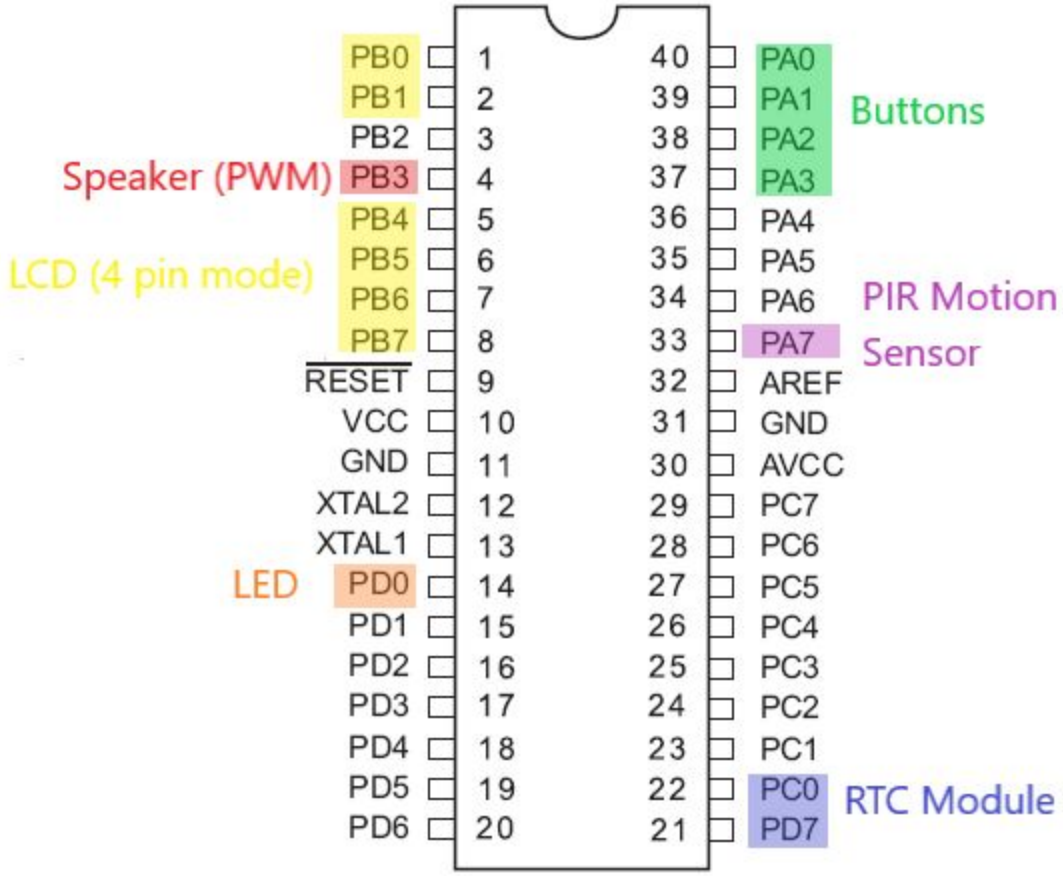
Hardware

Parts List

The hardware that was used in this design is listed below. The equipment that was not taught in this course has been bolded.

- ATMega1284 microcontroller
- LED
- 330 Ohm Resistor
- Speaker
- Buttons
- **LCD Screen** in **4 pin mode**
- **PIR Motion Sensor (JST)**
- **RTC DS1307 Module**

Pinout



Software

- AVR Studio 7
- EEPROM on Atmega1284

Source Files

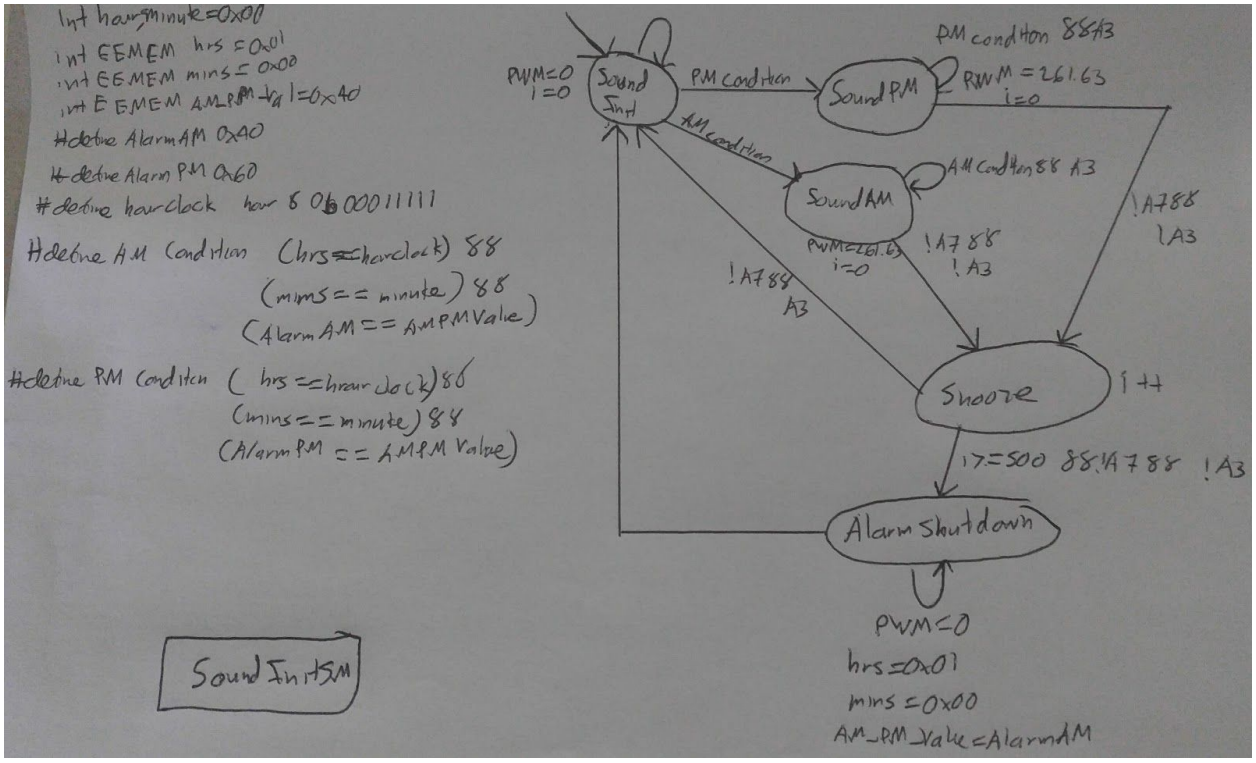
- Main.c - implementation of alarm settings and speaker settings
- LCD_definitions.c - implementation of LCD functions
- RTC_definitions.c -implementation of RTC functions
- LCD_Declarations.h - header file for LCD functions
- RTCDeclarations.h - header file for RTC functions
- Timer.h - header file for Timer usage
- Bit.h - header file for GetBit, SetBit usage
- Pwm.h - header file for PWM
- RTCLCDMerged.eep - EEPROM settings

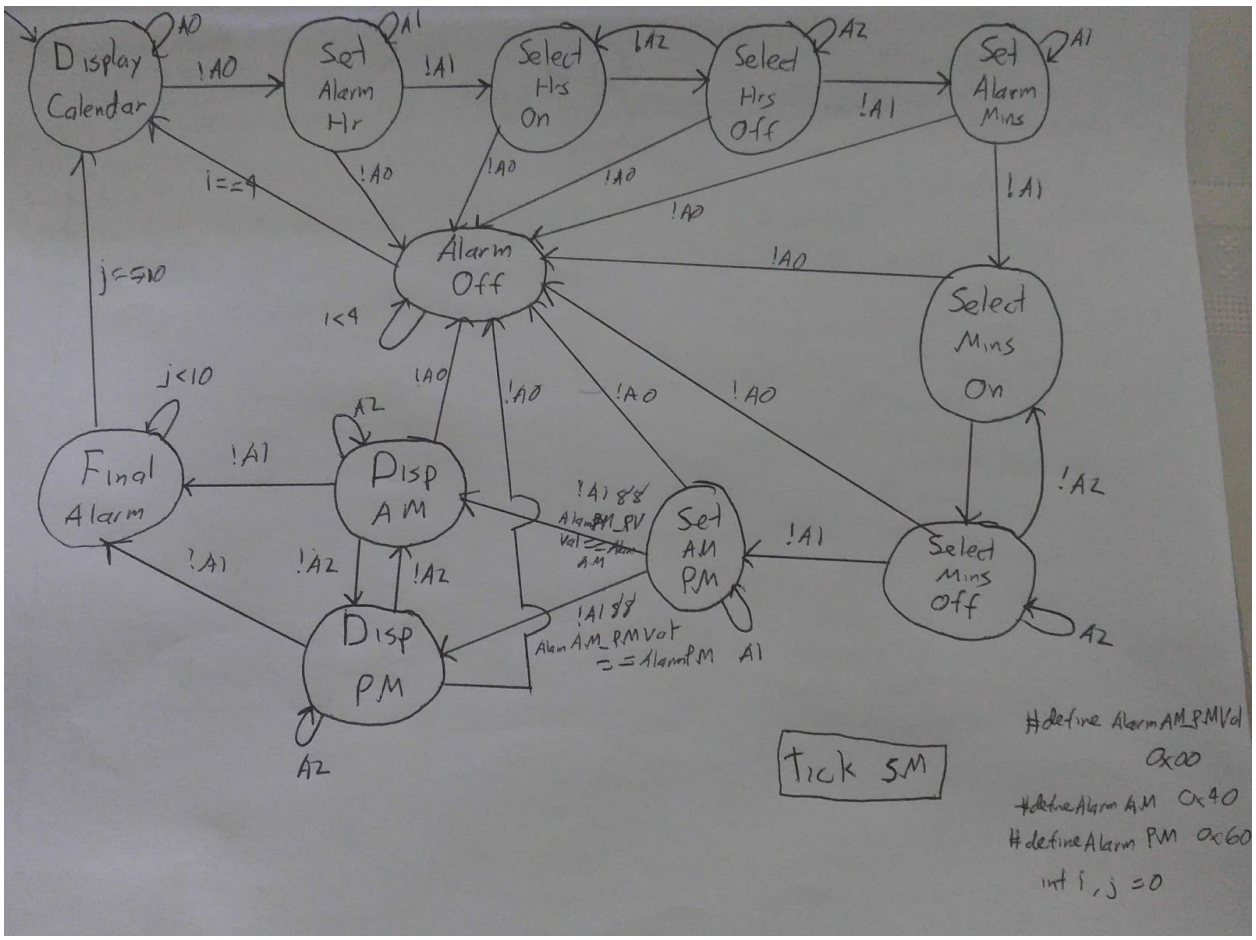
- All the source files are here:
    - C files https://drive.google.com/open?id=1doO3lW1BaqqWAo_yx9UdurTmCoe7KuCD
    - H files https://drive.google.com/open?id=1mDYki1DJrop2AURaM27OXOZo7kcJN-Jm

int hour_minute=0x00
int EEMEM hrs=0x01
int EEMEM mins=0x00
int EEMEM AM_PM_val=0x40
#define AlarmAM 0x40
#define Alarm PM 0x60
#define hourclock hour & 0b00011111

#define AM Condition (hrs==hourclock) && (mins==minute) && (AlarmAM==AMPMValue)

#define PM condition (hrs==hourclock) && (mins==minute) && (AlarmPM==AMPMValue)





#define AlarmAM_PMVal 0x00
#define Alarm AM 0x40
#define Alarm PM 0x60
int i, j = 0

Complexities

These are new topics that were not taught in this course.

- Using the RTC module via I2C
- Using EEPROM to save the alarm time
- Using LCD screen in 4 pin mode
- Reading input from PIR motion sensor

Youtube Link

https://youtu.be/KmKkiifMTRU

Known Bugs and Shortcomings

- Sometimes, upon multiple times cutting and resetting the power, the buttons start to behave strangely, sometimes they lag and sometimes a quick button press scrolls through settings too fast. This may be a due to a flaw in the timer setting. Will investigate the timer period and how it is used/changed.

Future work

The next feature will be adding a song to the alarm instead of just a regular buzzing noise. Will investigate how to produce different frequencies using PWM in order to make a complex song.