James Bocinsky

04/19/17

39058198

Final Project Report

Introduction

The goal of this module is to create a functioning alarm clock using the MCP7940N RTC. The alarm clock will use light to wake up the user. To communicate with this device, an I2C controller had to be developed with the PIC microcontroller. This allowed me to set the time and retrieve the time from the RTC. With an LCD, two potentiometers, and three buttons the end user is able to set the time and set an alarm to a certain hour, minute, and second. When the alarm was triggered, a light would slowly brighten until a button was pressed.

Design

The hardware design aspects required a power regulation circuit, the PIC, a LCD, a RTC, three push buttons, a DAC, and two potentiometers. The specific part numbers for these parts are referenced in the bill of materials. If any specifications are required to differentiate part numbers, they will be used in this design section. The power regulation could convert any voltage from 6 to 24 volts down to 5 volts using an LDO and two capacitors; a .47uF on the input, and a 50uF on the output. All of the components required a 5V power supply and the capacitors were added as they were specified in the LDO manual. The LCD was used to display the time and the different states of the system. The different states were setting the time, setting the alarm, displaying the time, and alarm going off. Upon start up, the set time state would occur and the user would have to insert the time. The time was adjusted by a potentiometer and based on what time unit the user was selecting, the value was limited to a certain range on the LCD. For instance the month was limited from 1 to 12. This was handled in software by dividing the analog value by a certain value corresponding to the range of values that was valid for that time field. When the user had the correct time unit, the user could press

the select button to move on to the next time unit. The button was implemented in software via an interrupt that would change a state variable to move on to the next reading. Once all the time units were selected on the LCD, the PIC would send the stored values to the RTC and start the clock. The state would change to displaying the time by polling the RTC and the clock display would then be updated on the LCD whenever the second unit that was set in software was changed.

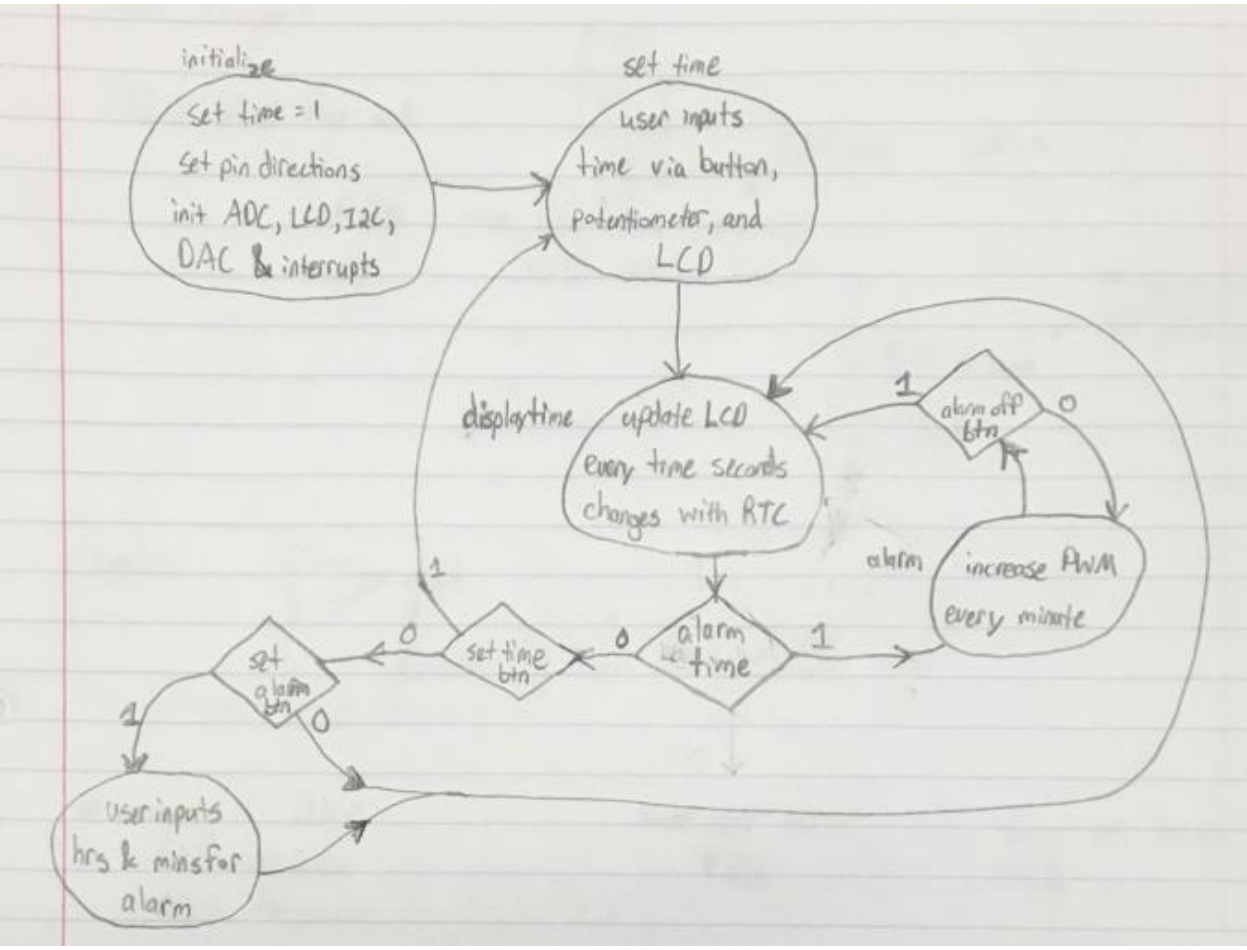
One of the potentiometers was used to control the brightness on the LCD. To implement this the analog value was read from the potentiometer by the PIC. Then in each fourth segment of the analog value, the LCD brightness would change in brightness by a fourth. So if the analog value was in the first fourth of values 0 to 255, then the analog output was 0. If the analog value was in the third fourth of values 512 to 768, then the analog output was 0xAA.The analog value was interpreted by the PIC and the corresponding brightness was sent to the LCD brightness via an external DAC.

When the user wanted to input an alarm, he would push the set alarm button, controlled by an interrupt, and then be asked to input the alarm time similarly to how the user set the normal time. The user can also continue to use the set alarm time every day. Although, if the user does not want to use the same alarm on the next day, he would need to push the set alarm button and twist the potentiometer all the way to the left. The user would be greeted with a message asking if he would like to turn off the alarm. If selected, the set alarm state would display alarm off and go back to displaying the time. When the user wanted to set a different time, he could push the set time button to set the time. The state then goes into the set time state which was discussed earlier in this design section.

When the alarm time would match the actual time minus 15 minutes, the LED controlled by the PIC would start lighting up. The bright LED was designed to start lighting up 15 minutes prior to the alarm time to slowly wake the user up from sleep using PWM. Every subsequent minute after this first minute, the light PWM, controlled by bit banging and delays, would change in duty cycle. This PWM signal sent by the PIC was connected to a logic level power mosfet that could handle the amount of current and frequency required to operate the

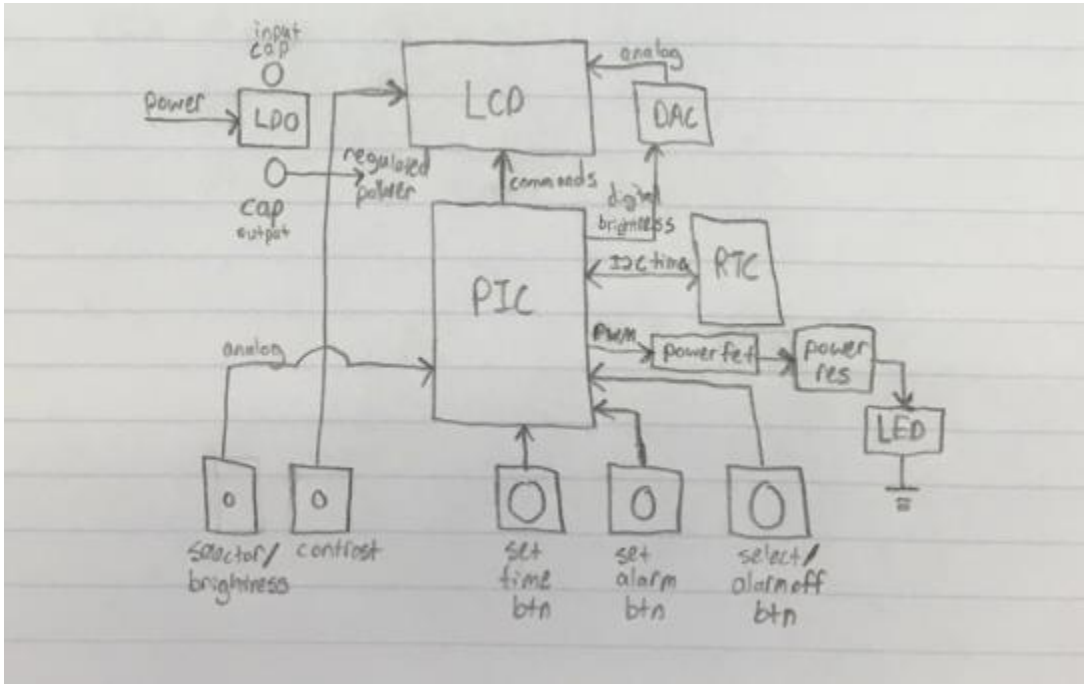
LED. Once the alarm time matched the actual time, the PWM controlling the LED would become 100% duty cycle and remain on until the user pushed the alarm on button.

Software Flow:

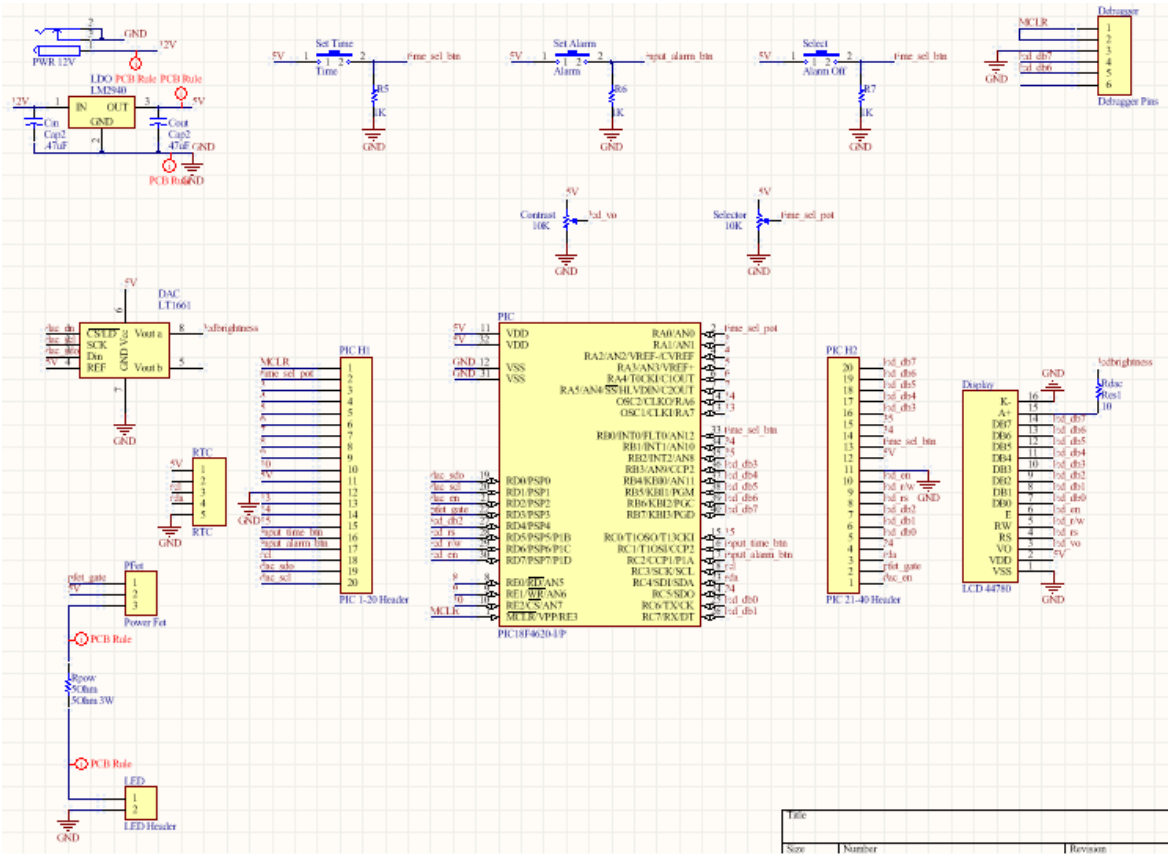


The overall software flow was as follows. Initialize the state variables and input/output pin direction. Initialize the ADC, LCD, I2C, DAC and interrupts. Change the state for the user to set the time. Change the state to show the time. When a button is pressed an interrupt isr would change the state variable and put the user in the setting alarm state or in the setting time state depending on which button was pressed. Once the alarm or time was set, the display time state would continue and the time would be displayed on the LCD, updating the clock every second until the alarm went off. At this point the LED would slowly light up for 15 minutes until it was at full brightness. The LED would remain at full brightness until the alarm off button was pressed and then the time would continue.

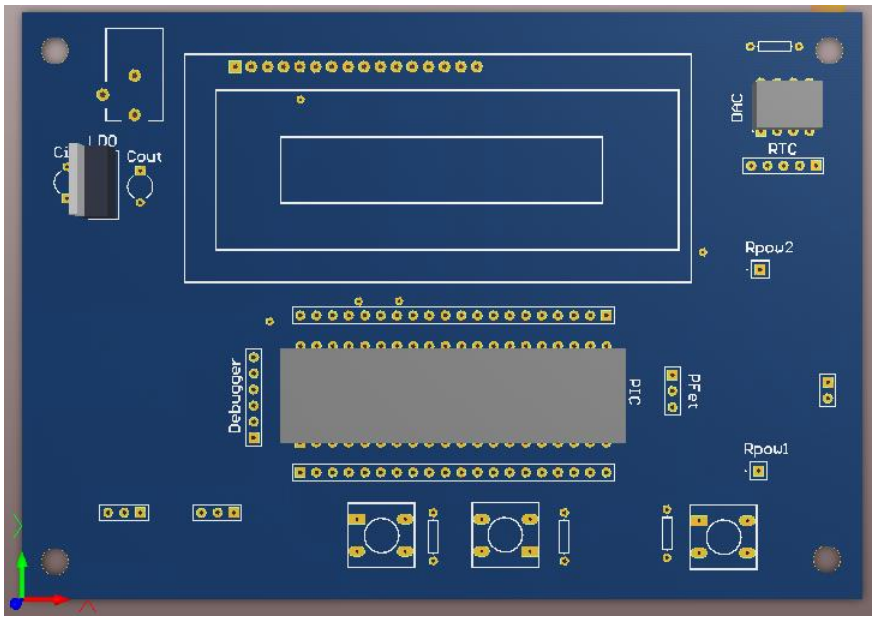
Block Diagram:

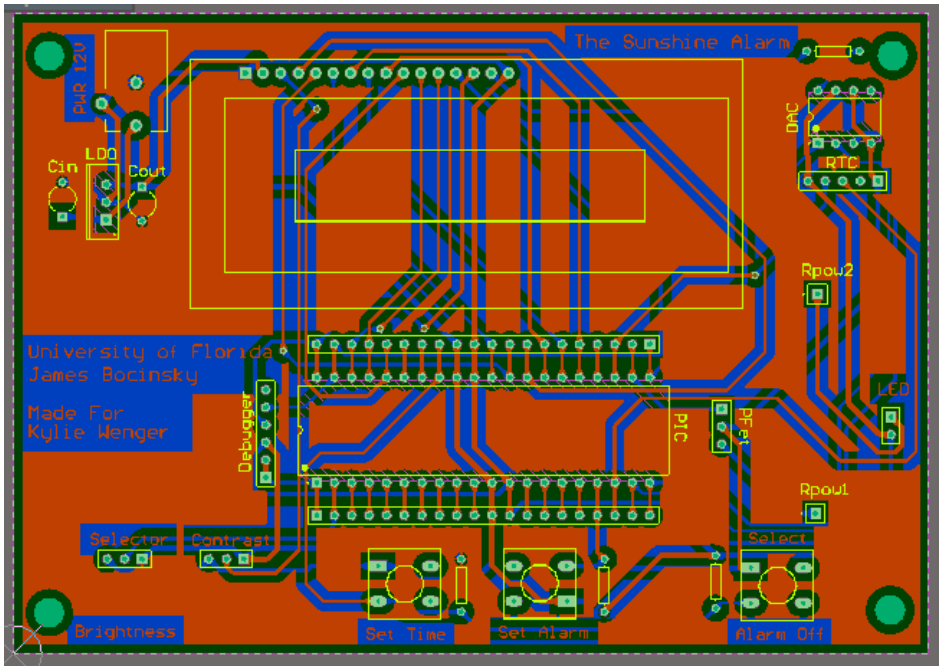


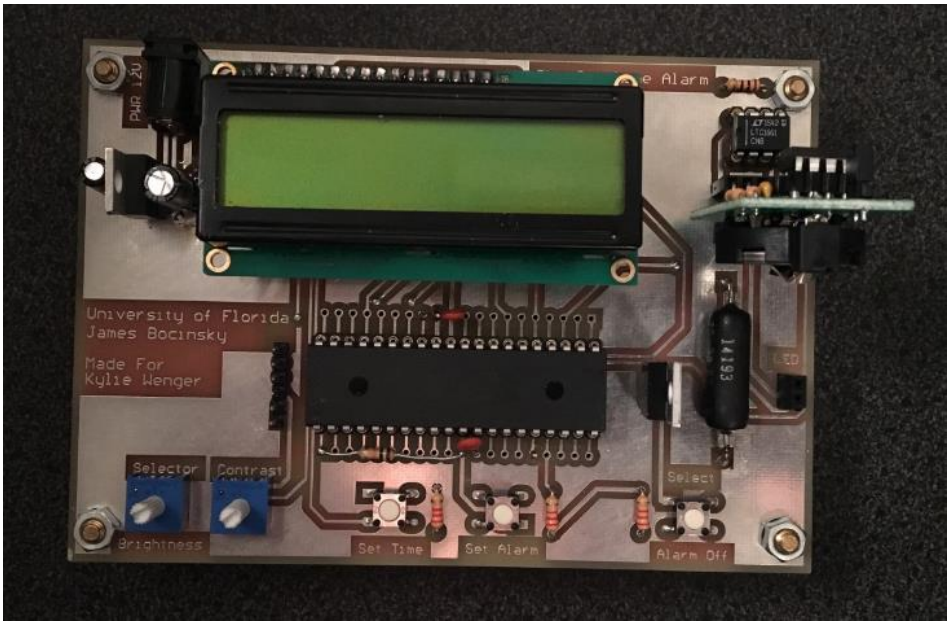
Schematic:



PCB:


LDO
Cout
DAC
RTC
Rpow2
Debugger
PIC
PFet
Rpow1


The Sunshine Alarm
PWR 12V
Cin
LDO
Cout
DAC
RTC
Rpow2
University of Florida
James Bocinsky
Made For
Kylie Wenger
Debugger
PIC
PFet
LDO
Rpow1
Selector
Contrast
Select
Brightness
Set Time
Set Alarm
Alarm Off


PWR 12V
e Alarm
University of Florida
James Bocinsky
Made For
Kylie Wenger
14493
LED
Selector
Contrast
Brightness
Set Time
Set Alarm
Select
Alarm Off

Bill of Materials:

Part	Part Number	Source	Amount	Cost ($)
PCB	V 1.0	Bay Area Circuits	1	85.00
10K Potentiometers	3362P-103	Digi-Key	2	1.02
Pic Microcontroller	18f4620	Arrow	1	5.83
LCD	CFAH1602Z	Crystalfontz	1	13.86
Push Buttons	COM-00097	Mouser Electronics	3	1.05
Resistor Pack	28-12966	MCM Electronics	1	4.89
3W Power Resistor	5Ohm 3W	UF ECE	1	0.50
Bright LED	10DGL-DZ-3W	Chanzon	1	0.70
LDO Power Regulator	LM2940	eBay	1	0.26
Capacitor (.1uF)	D103Z25Z5VF63L6R	Mouser Electronics	2	0.20
Capacitor (.47uF)	D103Z25Z5VF63L6R	Mouser Electronics	1	0.25
Capacitor (47uF)	UHE1E470MDD	Mouser Electronics	1	0.32
DAC	LTC1661	Arrow	1	3.34
RTC	MCP7940N	UF ECE	1	1.10
Logic Level Power Mosfet	FQP30N06L	Mouser	1	1.11
Total Cost	N/A	N/A	N/A	119.43

Conclusion

The final results of creating the alarm clock were within spec and worked wonderfully. I plan to use the alarm myself with a wall DC power supply. It took a great deal of effort and time to be able to get the I2C to communicate with the RTC. This took a long time and was the most amount of struggle with the module. Another issue was a mislabeled net in my schematic, two button outputs has the same net so when the PCB was created this net had to be shaved down and disconnected. The button outputs then had to be routed to the correct pins on the PIC. Setting the alarm, time, and LED brightness was more so a matter of getting it done, than figuring it out. With this said, the project worked very well and I was proud of the system.