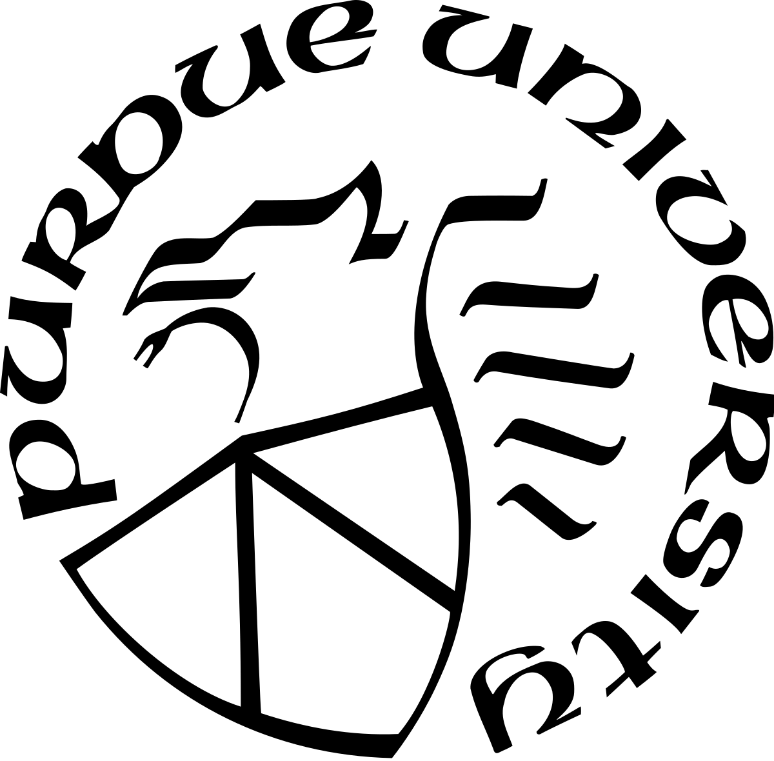
**KITCHEN TIMER**

By

**Peter Daum & Austin Schlegel**

ECE 471 Project Report



School of Electrical and Computer Engineering

Indianapolis, Indiana

April 2019

Table of Contents

[2 List of figures 4](#_Toc6861514)

[3 Introduction 5](#_Toc6861515)

[3.1 Project Proposal 5](#_Toc6861516)

[3.2 Peripherals used 5](#_Toc6861517)

[3.2.1 LCD screen 5](#_Toc6861518)

[3.2.2 Timer 0 5](#_Toc6861519)

[3.2.3 Potentiometer (ADC) 5](#_Toc6861520)

[3.2.4 Temperature Sensor (ADC) 6](#_Toc6861521)

[3.2.5 Speaker (PWM) 6](#_Toc6861522)

[4 Requirements 6](#_Toc6861523)

[5 Detailed Design 6](#_Toc6861524)

[5.1 Hardware Design 6](#_Toc6861525)

[5.1.1 LCD 6](#_Toc6861526)

[5.1.2 Potentiometer 6](#_Toc6861527)

[5.1.3 Temperature Sensor 7](#_Toc6861528)

[5.1.4 Pushbutton 7](#_Toc6861529)

[5.1.5 LEDs 7](#_Toc6861530)

[5.1.6 Speaker 7](#_Toc6861531)

[5.2 Software Design 8](#_Toc6861532)

[5.2.1 C program file structure 8](#_Toc6861533)

[5.2.2 LCD 8](#_Toc6861534)

[5.2.3 Potentiometer 10](#_Toc6861535)

[5.2.4 Speaker 11](#_Toc6861536)

[5.2.5 Temperature sensor 12](#_Toc6861537)

[5.2.6 Timer0 12](#_Toc6861538)

[5.2.7 Menu 13](#_Toc6861539)

[5.2.8 Main 13](#_Toc6861540)

[6 Implementation Details 14](#_Toc6861541)

[6.1 List of Parts and Tools 14](#_Toc6861542)

[6.1.1 PIC18F4331 14](#_Toc6861543)

[6.1.2 1602A 16x2 LCD 14](#_Toc6861544)

[6.1.3 LEDs (Red, Green, Blue) 14](#_Toc6861545)

[6.1.4 LM35 Temperature Sensor 14](#_Toc6861546)

[6.1.5 10k Potentiometer 14](#_Toc6861547)

[6.1.6 Speaker 14](#_Toc6861548)

[6.1.7 Push Button 14](#_Toc6861549)

[6.1.8 Elegoo Power MB V2 14](#_Toc6861550)

[6.1.9 9V Battery 14](#_Toc6861551)

[6.2 Wiring Diagram 15](#_Toc6861552)

[6.3 Layout Diagram 16](#_Toc6861553)

[6.4 Software Tools 16](#_Toc6861554)

[6.4.1 MPLABX IDE 16](#_Toc6861555)

[6.4.2 XC8 Compiler 16](#_Toc6861556)

[7 Experiment Results 17](#_Toc6861557)

[7.1 Kitchen Timer System Results 17](#_Toc6861558)

[8 Discussion 17](#_Toc6861559)

[8.1 Software 17](#_Toc6861560)

[8.2 Hardware 18](#_Toc6861561)

[9 Conclusion 19](#_Toc6861562)

[10 References 19](#_Toc6861563)

[10.1 PIC18F4331 Datasheet 19](#_Toc6861564)

[10.2 1602A LCD Datasheet 19](#_Toc6861565)

[10.3 LM35 Temperature Sensor Datasheet 19](#_Toc6861566)

[11 Appendices 19](#_Toc6861567)

[11.1 Specifications 19](#_Toc6861568)

[11.2 Complete Software Code 20](#_Toc6861569)

[11.3 Manual 20](#_Toc6861570)

[11.3.1 Power On 20](#_Toc6861571)

[11.3.2 User Interface 20](#_Toc6861572)

[11.3.3 Setting a time 20](#_Toc6861573)

[11.4 Video Link 21](#_Toc6861574)

# List of figures

[Figure 1 File Structure 11](#_Toc6860998)

[Figure 2 Circuit Diagram 18](#_Toc6860999)

[Figure 3 Layout Diagram 19](#_Toc6861000)

[Figure 4 LCD Screen Result 20](#_Toc6861001)

# Introduction

## Project Proposal

For our project we propose to create a kitchen timer device using the PIC18F4331 microcontroller. We will use an LCD display to show the time and temperature. The user menu will have the options to select timer, pause timer, resume timer, and a bonus feature. The user interface will be controlled by a potentiometer and a push button. The time displayed is controlled by the timer peripheral.

After selecting a timer, the countdown begins. The green LED will show the timer status by blinking when the timer is running or off when the timer is paused or off. Once the time is up an alarm will sound and the red LED will flash. There temperature will be collected using an analog temperature sensor.

## Peripherals used

### LCD screen

The LCD screen will be used to display the user interface. The user interface consists of three menu options, current temperature, and time left on the timer.

### Timer 0

The PIC18F4331’s timer 0 module is used to control the applications timer functionality. The time counted in HH:MM:SS is displayed to the LCD screen.

### Potentiometer (ADC)

A variable 10K Ohm potentiometer is used to scroll through the menu and also to scroll to select time for timer. The potentiometer is analog input to the PIC.

### Temperature Sensor (ADC)

The current temperature is collected via temperature sensor. The temperature sensor is analog input to the PIC and displayed to the LCD screen.

### Speaker (PWM)

The speaker is controlled by the PIC’s pulse width modulation port and plays a tune when the timer counts down to 0.

# Requirements

* Project must use a PIC microcontroller
* The hardware must be built using the PIC microcontroller, premade circuit will not be accepted
* In addition to general purpose I/O, the project must use four peripheral components

# Detailed Design

## Hardware Design

### LCD

The LCD uses 8-bit data, therefor 8 wires had to be connected to D0-D7. The output for data uses all of port d of the PIC. RS is connected to pin RB0 and EN is connected to pin RB1. To control the contrast V0 is connected to a 10k Ohm resistor which is connected to ground. This makes for optimal contrast.

### Potentiometer

The potentiometer is connected to pin AN0 and configured for ADC.

### Temperature Sensor

The temperature sensor is connected to pin AN1 and configured for ADC.

### Pushbutton

The pushbutton is pull down, so the value will read at pin RC0 will be 1 unless the pushbutton is pressed. The connection to ground has a resistor value of 330 Ohms.

### LEDs

Red, Blue, and Green LEDs are connected to RB2, RB3, and RB4 in that order. To control the brightness of the LEDs they are connected to a resistor, with value 330 Ohms, connected to ground.

### Speaker

The speaker is connected to pin CCP1 which can be configured for PWM.

## Software Design

### C program file structure

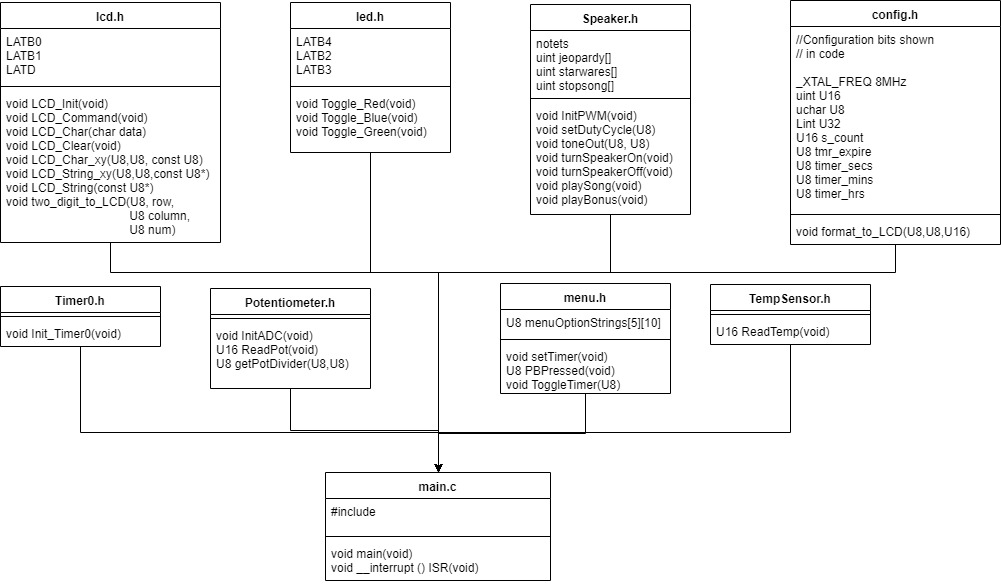


Figure File Structure

### LCD

The software implementation of the LCD uses lcd.c and lcd.h. The function used for the LCD are LCD\_Init, LCD\_Command, LCD\_Char, LCD\_Clear, LCD\_Char\_xy, LCD\_String\_xy, LCD\_String, and two\_digit\_to\_lcd. Here is what each function does and the parameters required:

#### LCD\_Init

Params in: void

Params out: void

Description: The LCD\_Init function initializes the LCD screen for use. It first delays 15 ms to wait for the LCD power on delay. It then send 0x38 to the LCD which specifies that each character on the LCD is a 5x7 matrix of pixels, this also initializes the LCD to be 16 columns by 2 rows. Then the screen is cleared by sending 0x01 to the LCD. Then display on cursor is turned off by sending 0x0C to the LCD. Then the cursor is incremented by sending 0x06 to the LCD.

#### LCD\_Command

Params in: U8 cmd

Params out: void

Description: The LCD\_Command function is used to send commands or data to the LCD. We used 8-bit data for the LCD by using all 8 of the data connection to the LCD circuit. Ldata, which is defined in lcd.h as LATD (Port D) of the PIC, uses all of port d to send data to the LCD. RS (register select) is high when data input and low when instruction input. Since we are sending data (ldata) RS is set to high. EN (enable signal) is then set to high. No operation, then delay 10 ms to send the signal.

#### LCD\_Clear

Params in: void

Params out: void

Description: LCD\_Clear calls LCD\_Command to send 0x01 to the LCD, clearing it.

#### LCD\_Char

Params in: U8 data

Params out: void

Description: LCD\_Char displays sends one U8 char to the LCD.

#### LCD\_Char\_xy

Params in: U8 row, U8 pos, const U8 msg

Params out: void

Description: LCD\_Char\_xy prints a given character to the LCD at the specified row and column.

#### LCD\_String\_xy

Params in: U8 row, U8 pos, const U8\* msg

Params out: void

Description: LCD\_String\_xy prints a given string to the LCD at the specified row and column.

#### LCD\_String

Params in: const U8\* msg

Params out: void

Description: LCD\_String takes a message string and prints to character by character to the LCD.

#### Two\_digit\_to\_lcd

Params in: U8 row, U8 pos, const U8 num

Params out: void

Description: Two\_digit\_to\_lcd prints two digits to the LCD.

### Potentiometer

The potentiometer is used as the scroll wheel for the menu. It scrolls through 0-4 options.

#### InitADC

Params in: void

Params out: void

Description: Initializes analog to digital conversion for potentiometer and Temperature sensor. The ADC is configured as single channel, single shot, FIFO enabled, left justified, Fosc/2, AN0 and AN1 as input, group A and B, AN0 and AN1 as analog.

#### ReadPot

Params in: void

Params out: U16

Description: ReadPot reads the pot value by taking group A and waiting until the conversion is done. The result is in ADRESH and returned.

#### getPotDivider

Params in: U8 maxPotValue, U8 numMenuOptions

Params out: U8

Description: getPotDivisor divides the potentiometer max value of 255 into the provided numMenuOptions.

### Speaker

#### InitPWM

Params in: void

Params out: void

Description: Initializes PWM configuration to use CCP2 and timer2. Pin CCP2 is what the PWN is outputted to.

#### setDutyCycle

Params in: U8 duty\_cycle

Params out: void

Description: Sets the duty cycle for PWM for the speaker.

#### toneOut

Params in: U8 tone, U16 delay

Params out: void

Description: Produces a duty cycle and plays tone for delay time.

#### turnSpeakerOn

Params in: void

Params out: void

Description: Turns the speaker on by setting a duty cycle of 50.

#### turnSpeakerOff

Params in: void

Params out: void

Description: Turns the speaker of by clearing the duty cycle.

#### playSong

Params in: void

Params out: void

Description: Plays the start wars theme song with the given note frequencies. Loops for 26 times (the length of the song) generating the PWM period each time.

#### playBonus

Params in: void

Params out: void

Description: Same as playSong, except this plays the jeopardy theme song (atleasts attempts it) and flashes all the LEDs.

### Temperature sensor

#### ReadTemp

Params in: void

Params out: U16

Description: Takes input from group B and waits for that to be converted. After the conversion the result is stored in result and converted from Celsius to Fahrenheit. The result is returned.

### Timer0

#### Init\_timer0

Params in: void

Params out: void

Description: Initialized the PIC’s timer 0 module with the correct initial value (to give the timer a 1ms overflow interval), 16-bit mode, and 32-bit pre-scalar. Global interrupts are then enabled as well as timer 0 interrupt enable.

### Menu

#### setTimer

Params in: void

Params out: void

Description: Uses the read potentiometer value to select a time for HH:MM:SS. Hours has a max value of 99, minutes has a max value of 59, and seconds has a max value of 59. The push button is used to confirm the selection for each time unit.

#### toggleTimer

Params in: U8 sel

Params out: void

Description: sets timer 0 either to be on or off based on sel.

#### PBPressed

Params in: void

Params out: U8

Description: Return the inverse of the value read from the Push button pin. It returns the inverse because the push button is a pull down button and reads 1 when not pressed and 0 when pressed.

### Main

#### Main

Params in: void

Params out: void

Description: The main loop for the program.

#### Format\_to\_LCD

Params in: U8 row, U8 column, U16 num

Params out: void

Description: Takes a U8 or U16 and displays it to the LCD screen.

#### ISR

Params in: void

Params out: void

Description: This interrupt is triggered by timer0 every second. It counts down seconds, minutes, and hours. When all of the time units reach 0 the timer is disabled, the speaker plays a tone, and the red LED is toggled.

# Implementation Details

## List of Parts and Tools

### PIC18F4331

### 1602A 16x2 LCD

### LEDs (Red, Green, Blue)

### LM35 Temperature Sensor

### 10k Potentiometer

### Speaker

### Push Button

### Elegoo Power MB V2

### 9V Battery

## Wiring Diagram

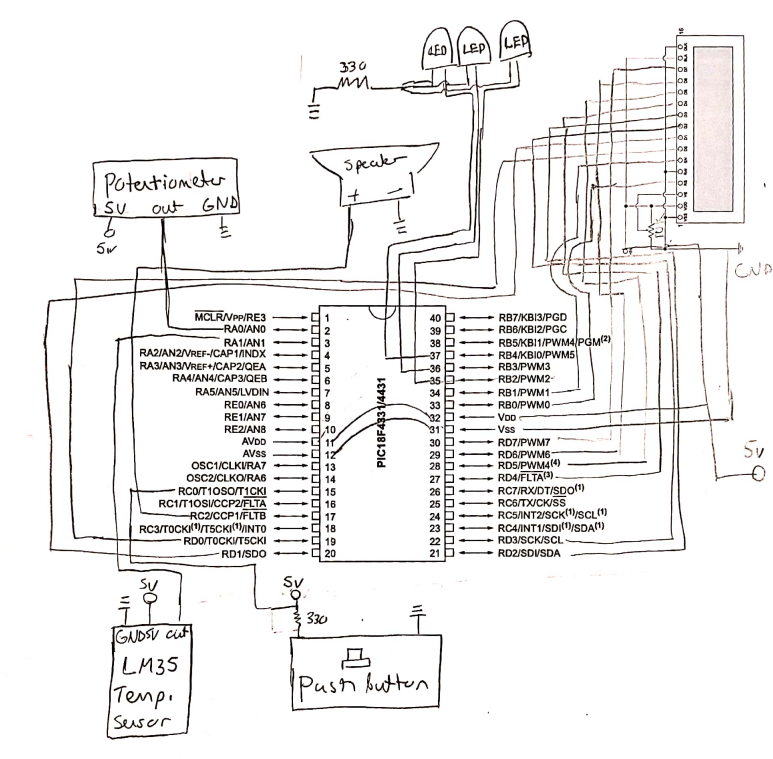
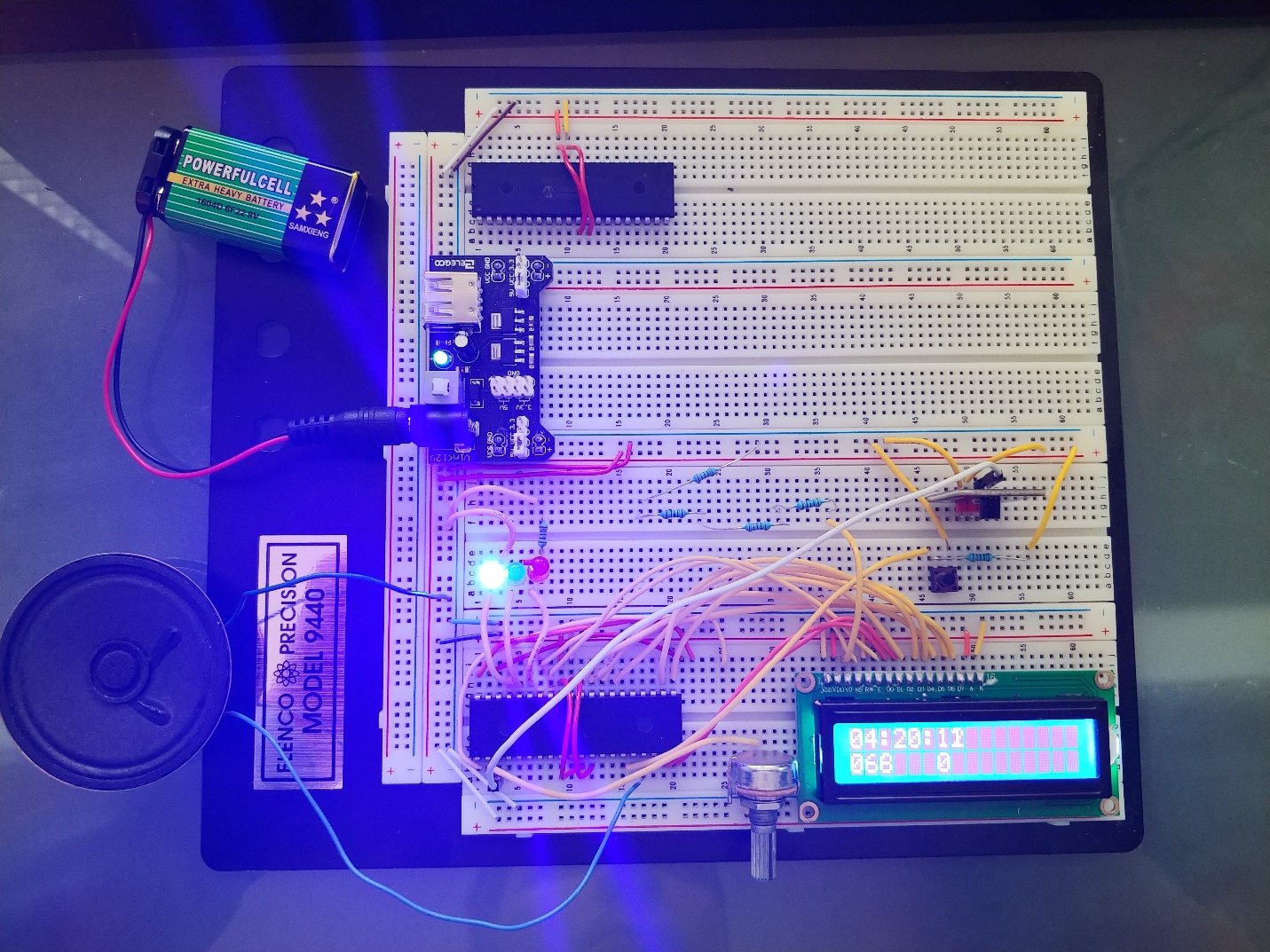


Figure Circuit Diagram

## Layout Diagram



4.1.7

4.1.9

4.1.8

4.1.6

4.1.5

4.1.4

4.1.3

4.1.1

4.1.2

Figure Layout Diagram

The layout diagram is marked with part numbers from section 4.1.

## Software Tools

### MPLABX IDE

For Development we used MPLABX IDE. This is the IDE provided by Microchip for development with PIC microcontrollers.

### XC8 Compiler

This is the compiler provided by Microchip for 8-bit PIC microcontrollers.

# Experiment Results

## Kitchen Timer System Results

The final result of this project is that the kitchen timer embedded system will count down time using timer0 module and sound an alarm using the speaker and PWM when done. The system will also measure temperature and has a scrollable menu with options to set time, pause time, resume time, and bonus. The program prints information to the LCD screen like follows:

HH:MM:SS

TTT Menu Option

Figure LCD Screen Result

# Discussion

## Software

The files for this project are all organized based on the module. The software implementations for each module required the use of module datasheets. The datasheet for the LCD specifies what each command does and the amount of time to delay after sending a signal. In order to setup the registers properly for ADC and timer we used the PIC18F4331 datasheet. There was nowhere in the PIC’s datasheet that showed the syntax for an interrupt, so we had to look that up on Microchip’s website.

The timer module uses timer 0 of the PIC18F4331. We chose to use timer 0 because it has the largest amount of programmable pre-scalar values and a 16-bit operation. We set up the timer, so it overflowed every second and stopped when every time unit was depleted. During development we first created a timer that would count up every second. We achieved this by incrementing a second count variable whenever the timer would overflow. We had some issues with getting the timer to be exactly 1 second, so we changed the start value of the timer register until it was the closest to 1 second. We tested this by comparing our timer with a timer on or phones, the timer ends up being off by around 1 second after 1 minute.

Initially we tried using the DHT11 temperature/humidity sensor. This sensor outputted a digital signal that required very precise delay to check if the input is high or low. The delay resolution that this required was in microseconds, which proved very difficult to implement. We could not use the built-in delay function that xc.h provided because our interrupt timer would throw the delay value off. So, we had a solution was to create a timer that interrupted every 10 milliseconds which we could use to make a delay. That proved to be very challenging to implement and due to time constraints, we decided to ditch that sensor and go with an analog sensor.

The analog sensor was much easier to implement. The hardest part of implementing the analog temperature sensor was converting 2 channels of analog. We accomplished this by only doing single shot conversion and selecting the proper channel when desired.

## Hardware

Since neither of us have a lot of experience with developing hardware this part of the project was challenging. We faced issues with bad wires, bad breadboards, improper wiring, and sometimes things just didn’t work for reasons we couldn’t explain. When things didn’t work, we tried debugging the wiring and if that didn’t work, we re-done the wiring.

The LCD proved to be the most challenging hardware module. The wiring for the LCD required a total of 16 wires since we decided to go with 8-bit data. The LCD at one point stopped working entirely due to an unknown error. After a few hours of trying to debug the error we decided to try a new breadboard, which fixed the problem. We also had problems when starting the project with wires that did not work.

# Conclusion

For this project we learned how to use the PIC’s timer module, ADC, and digital I/O. After implementing all our peripherals the kitchen timer worked flawlessly and without any known bugs. Completing this project made us more comfortable with developing an embedded system for a PIC microcontroller. Learning how to read and understand microcontroller datasheets makes it possible to work with many other microcontrollers as well and develop other embedded systems.

# References

## PIC18F4331 Datasheet

<http://ww1.microchip.com/downloads/en/devicedoc/39616b.pdf>

## 1602A LCD Datasheet

<https://www.openhacks.com/uploadsproductos/eone-1602a1.pdf>

## LM35 Temperature Sensor Datasheet

<http://www.ti.com/lit/ds/symlink/lm35.pdf>

# Appendices

## Specifications

Params is short for parameters.

All commands sent to the LCD are specified in the LCD datasheet.

## Complete Software Code

Git repository link: <https://github.com/ajschleg/Kitchen_Timer_Project>

## Manual

### Power On

To power on the system press the switch on the Power MB V2 module. A tune will play on startup.

### User Interface

Once the system boots up the timer will show:

00:00:00

064 4

Or something similar. The time left on the timer is on the 1 row shown above the temperature and menu option which are on the second row. To scroll through the menu use the potentiometer, there are 0-4 options. The options for the UI are as follows:

0 – Set time

1 – Pause timer

2 – Resume timer

3 – Unimplemented

4 – Bonus

Once you are on the desired option press the push button located above the LCD.

### Setting a time

To set a time select option 0. After option 0 is selected the potentiometer can be used to scroll through the desired time. First select second by setting to a value between 0 and 59 then pressing the push button. Then select minutes and hours.

## Video Link

<https://youtu.be/glxxr_63HbA>