Michael Tran
Allen Breyer
CS 362

Weather Monitoring Alarm Clock Final Report

Description:

For our project we plan to design an alarm clock that is able to provide useful functions for an individual. Functions are not limited to but including date, time, customizable alarm, measuring humidity, and measuring temperature of the room. The DHT11 sensor can measure both humidity and temperature and it is what we are going to be using for our project. We will have our two Arduinos communicate to one another via serial communication so that we can split the work of our design. One Arduino will be receiving data from the DHT11 sensor for humidity and temperature to display on its LCD. It updates every second. In addition, this Arduino will be in charge of simulating an alarm clock buzzer when the set alarm time matches both the current time and date. The second Arduino will be displaying the time/date. There will be various buttons on the second Arduino to control the time/date and setting the alarm. When the current time and date has passed the set alarm time and date, the second Arduino will signal the first Arduino to start buzzing.

On the second Arduino, we have five buttons. Two buttons are designed to change the data fields -- one going back, and the other forward. It will start at the month data field. The other data fields consists of day, year, hour, minute, and second. The data fields can also loop around, for example, you can go from month to second. Two more buttons are used to increment and decrement the data fields. If we are changing the data field of month, we can increment to 12 or decrement to 1. We are able to loop around so incrementing past 12 would bring the value back to 1. These four buttons will not work until we are in “edit” or “alarm” mode. The last button is used for changing modes on the clock. We have three modes: TIME, EDIT, and ALARM. The TIME mode will be displaying the current time and date, the EDIT mode will stop the ticking of the current time/date and let the user be able to configure the date/time using the four buttons previously mentioned above. Then the ALARM mode will do the same as EDIT, however, the current date/time will still be ticking since we are only changing the alarm.

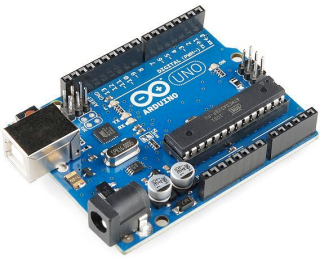
Possible additional features we wanted to add were weather based off a weather api, nightlight feature, snooze button, quote of the day, online calendar checker so that our date will be more accurate (i.e. there is no 31st in February).

Materials:

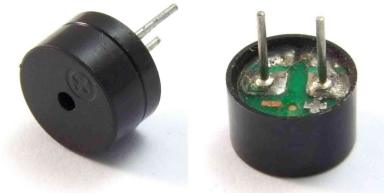
- 

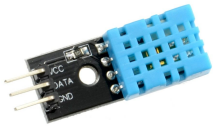
16x2 LCD Display x2

- 
Pushbuttons x 5

- 

Arduino Uno x2

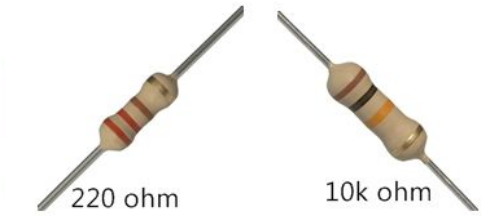
- 
Passive Buzzer x 1

- 

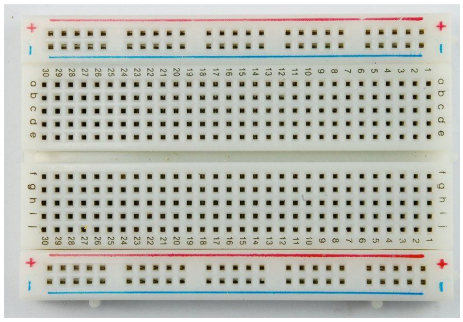
DHT11 x 1

- 
Jumper Wires (Many)

- 
Potentiometer

- 

Resistors (220, 10k)

- 
Breadboard x 2

- 

9V Battery

- 
Arduino USB

References:

Pushbutton

- Lab 2 (Input and Output)
- This lab provided us the ability to have debounce when reading buttons because we wanted to make sure the when the user pressed the button, it only registered as one press.

Date/Time

- Lab 6 Time library
- Github : https://github.com/PaulStoffregen/Time
- Using the library : http://www.pjrc.com/teensy/td_libs_Time.html
- Used for keeping track of the date and time

Alarm

- Lab 5 (Buzzer) and Lab 6
- https://www.arduino.cc/reference/en/language/functions/advanced-io/tone/
- Used to play a tone when alarm should be ringing (the alarm time has been reached)

Humidity/Temperature

- http://www.circuitbasics.com/how-to-set-up-the-dht11-humidity-sensor-on-an-arduino/
- Plan on setting up the dht11 humidity sensor and setting it up as analog input for our arduinos to monitor the humidity and temperature in the air.

Serial Communication

- Lab 6 and Lab 7
- http://arduino.cc/en/Reference/Serial#.UwYyzfldV8E
- http://arduino.cc/en/Serial/Available#.UwYy2PldV8E
- http://arduino.cc/en/Serial/ReadBytesUntil#.UwYy6_ldV8E
- Plan on connecting two Arduinos to share information between each other so that one Arduino can be handling the alarm and time portion whereas the other Arduino can be handling the temperature and buzzer. We wanted to break up the work for the Arduinos so it will be easier to debug.

Schedule:

10/25/2018

- Focus on getting Lab 6 and 7 working, will come in handy for project
- Order the parts we may not have, but need for this project

  *** Lab 6 and 7 completed. DHT11 humidity and temperature sensor part ordered**

11/8/2018

- Work on creating an alarm clock that buzzes the passive buzzer when certain time has passed. Can be hard coded to have a 10 second alarm for testing.
- Attempt to gather humidity and temperature using the DHT11 (should be in by now) and having that display on LCD
- Expand the alarm clock so that we could control the internal timer that sets off the alarm using buttons.

  *** DHT11 sensor delivered on 11/6/2018**

  *** Successful in testing the DHT11 sensor for humidity and temperature**

11/9/2018

- Finalize 4 page write-up and submit it
- Catch up on any designs in case we are behind. ~~If not behind, try adding additional features.~~

11/15/2018

- Make sure circuit works with all features we planned on having: date, time, customizable alarm, measuring humidity, and measuring temperature of the room.

11/22/2018

- Start on final report
- Debug the circuit to ensure all features are working correctly
- ~~If there is extra time, we can enhance the project with small additional features.~~

11/29/2018

- Create project video
- Finish final report
- Get the project checked in by a TA

12/6/2018

- Check in project with Professor/TA before last office hours on 12/7 if necessary.

What we have done (11/9/18):

- Tested DHT11 sensor to detect the humidity and temperature
- Got a buzzer to buzz for a couple seconds and then stop. Then repeat. This is to simulate the alarm clock.
- Have a simple buzz when time has passed the internal alarm clock (hard coded alarm clock time)
- Displaying current time/date on the lcd
- Able to turn on/off the alarm clock

What did not work (11/9/18):

- Prototype 1 Schematic was very cluttered and the code became cluttered (could not wire the additional buttons we wanted to add).
- We tried having the alarm clock time customizable through buttons, but it hasn't worked out yet. We have a problem with the date/time that we set using the buttons not saving.
- We have a problem with the buzzer still buzzing when the time matches the alarm clock time even though we turned off the alarm beforehand.
- There is an issue of the buzzer buzzing for 1 second when the current time matches the alarm clock time and is silent for a couple of seconds. Then it proceeds with our alarm buzzing the way we wanted it to.
- Displaying on the lcd for the date and time is hard to see, we suspect that this is because the lcd is updating many times and we are cleaning the lcd each iteration before we show the date/time.

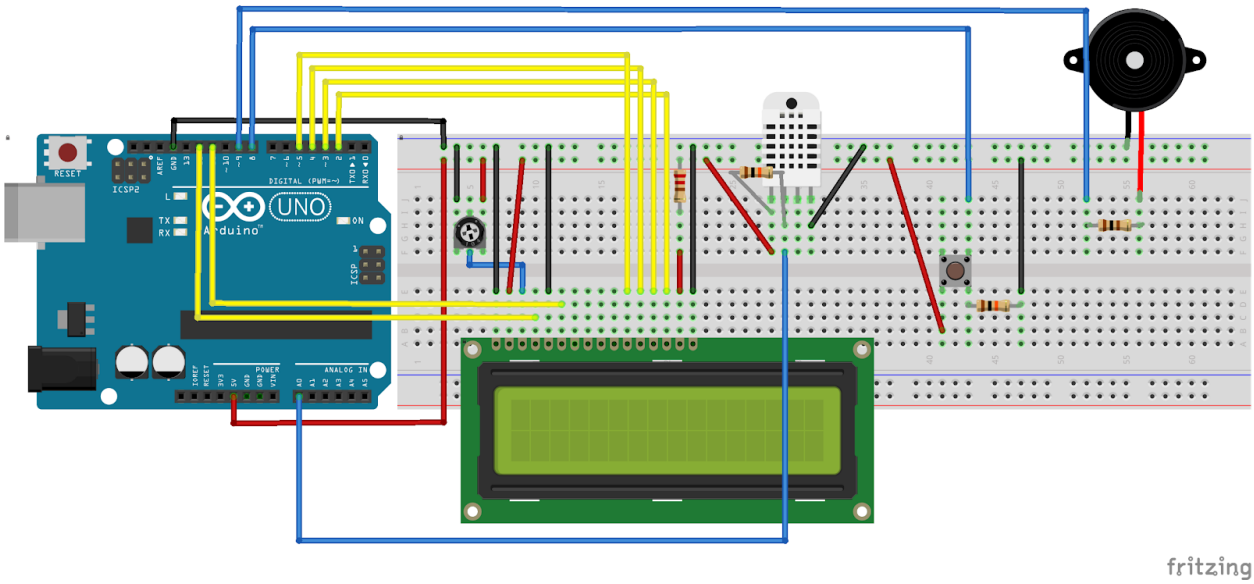
What we have done (11/29/18):

- Designed a circuit holding all the materials listed above
- Able to display the time and date
- Able to change the time and date, fixed the issue of date/time not saving when we exited out of the alarm mode
- Able to set an alarm time and date
- Alarm will buzz when the current time matches the alarm date and time
- Able to display temperature and humidity of the room
- Fixed the issue on lcd updating too fast by changing it every couple milliseconds by storing now() and updating when our stored time is different from now()

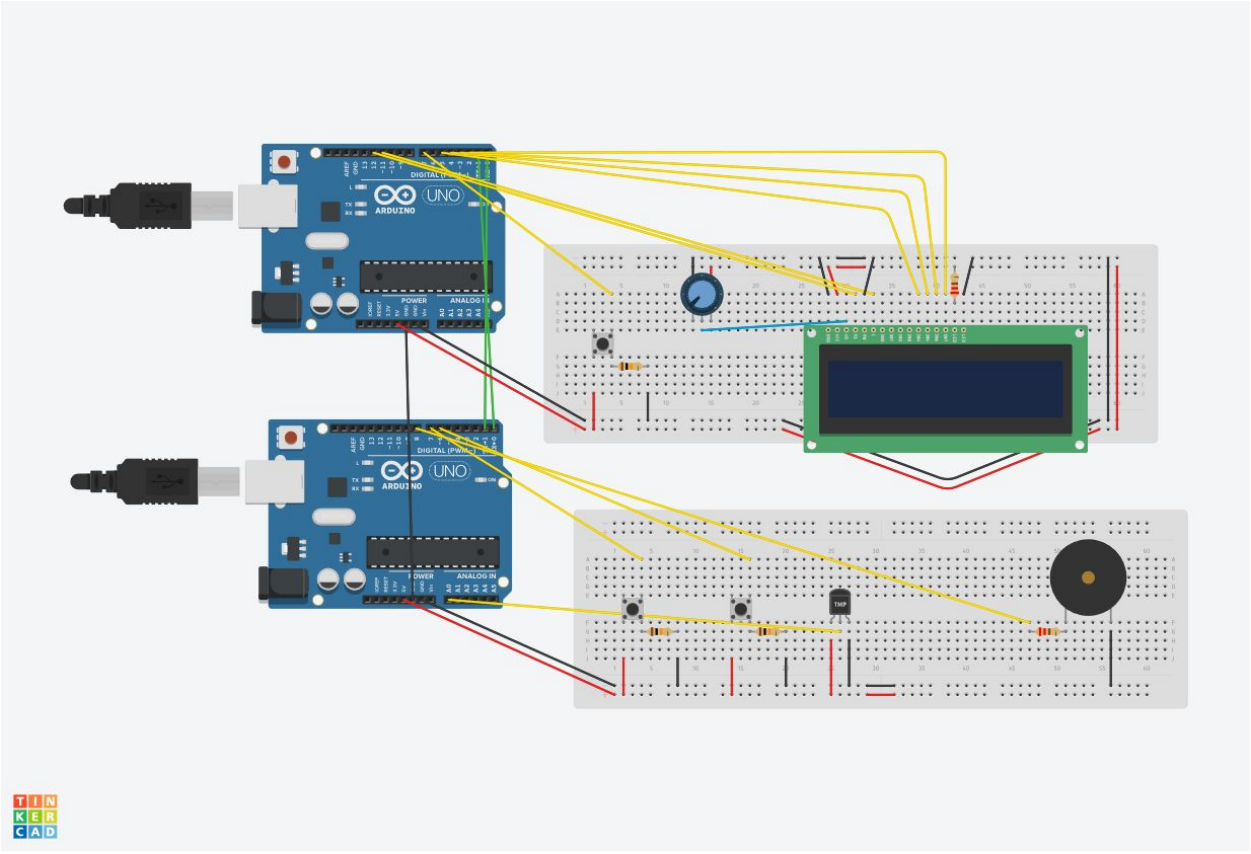
What did not work (11/29/18):

- We were unable to update the humidity and temperature normally when the buzzer was buzzing. This was because we used delay() to simulate the alarm sound, which caused us to not update the humidity and temperature at the rate we had it.
- We were unable to add a snooze button and some of the additional features that we wanted to because our Arduinos were close to out of pin. We required a bigger Arduino with more pins so that we could enhance the alarm clock.

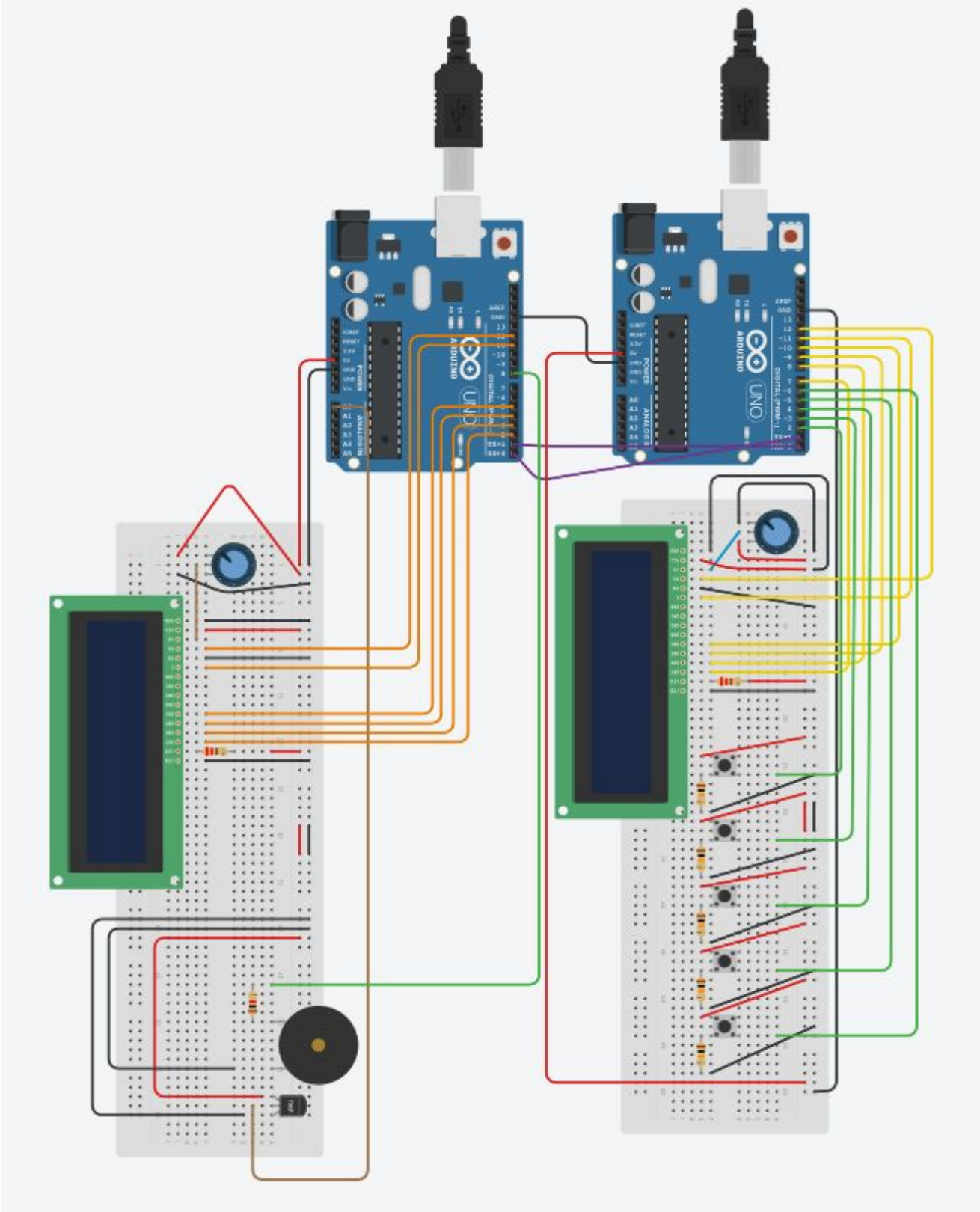
Prototype 1 Schematic



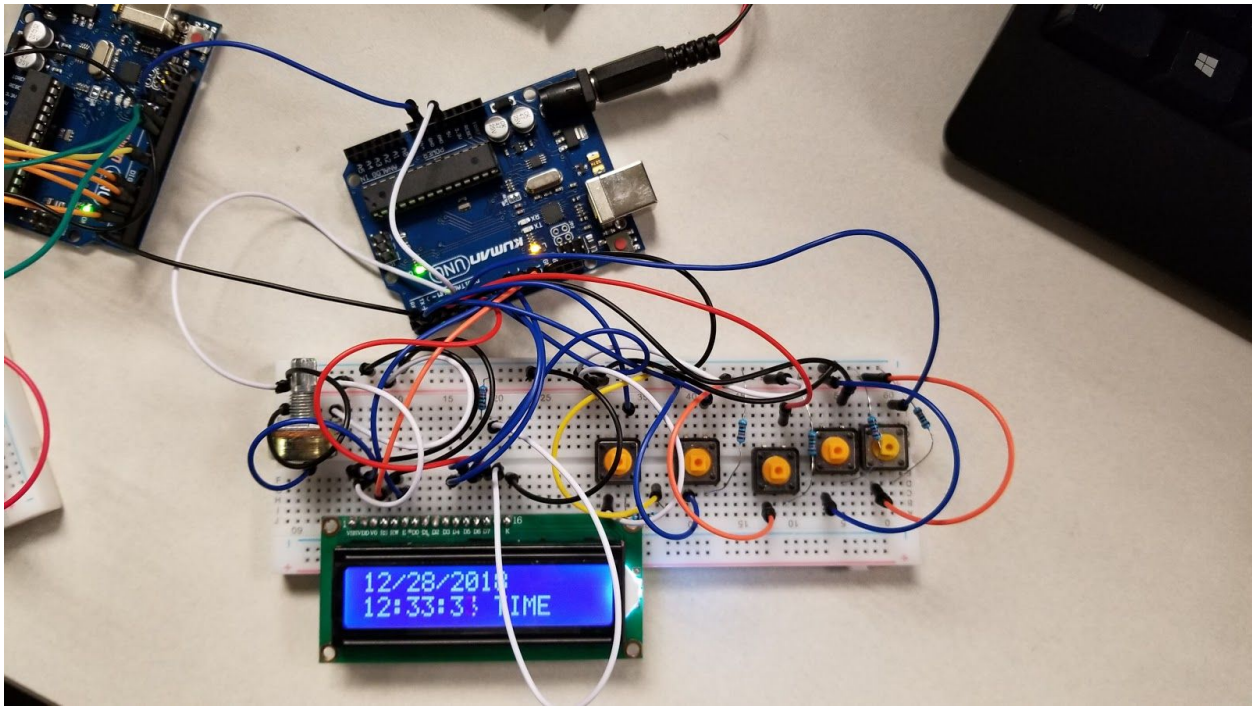
Prototype 2 Schematic



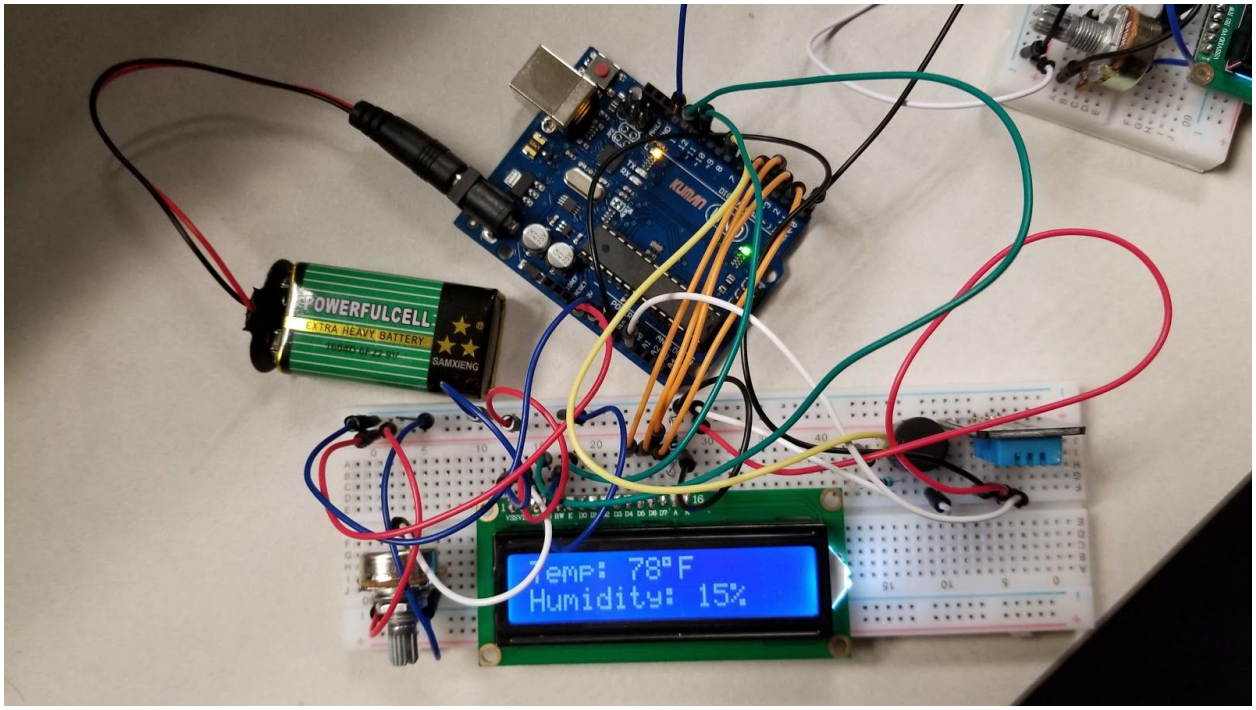
Final Schematic:



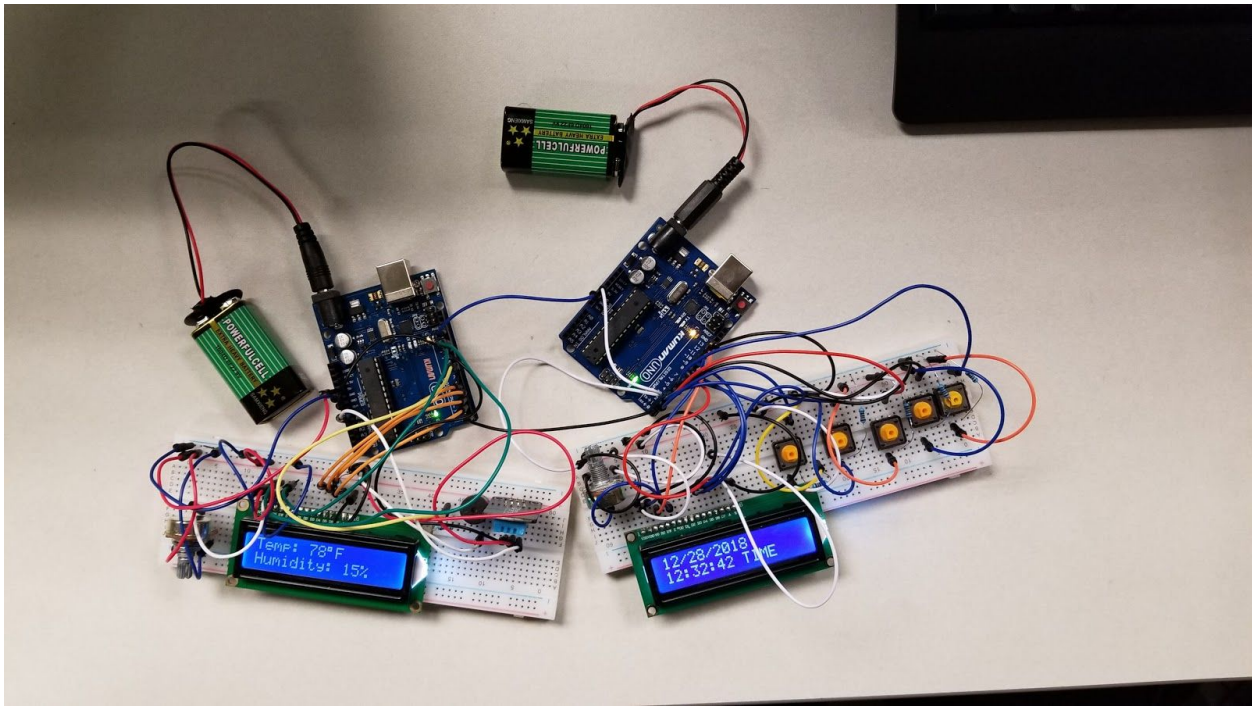
Arduino 1 (Time/Date):



Arduino 2 (Temperature/Humidity/Buzzer):



Final Circuit:



Problems encountered:

- We did encounter a few problems when building our project. First, we wanted to use the interrupts for the buttons to implement our functions for setting the date and time for the alarm clock. One issue with the interrupts was the button catching noise. The button would register multiple interrupts when the button was being pressed only once. To solve this issue, we used a form of debounce which would read the button, and then wait 50ms before it could read again.

- Another problem was when we found out that only 2 pins could be used as interrupts, but we needed up to five interrupts for our buttons. Instead of using interrupts, we used polling, which checks the buttons until the state of the button has changed causing us to unnecessarily waste CPU cycles.

- Another issue we had was simulating the buzzer from an alarm clock without affecting the other inputs/outputs. We wanted to have the buzzer make noise on and off for several seconds. We had to use delays to simulate the alarm for our project which affected the display of the LCD. When the alarm was buzzing, the LCD would not update normally because of the delay.

- At one point, we accidently touched one of the 5V jumper wires to one of our pushbuttons (we aren't quite sure which part of the pushbutton), but that caused our usb port to not work anymore. It would not supply power to any usb cable. We fixed it by restarting the computer, but that caused us a lot of confusion and time spent figuring out how to fix. We suspect that it may have short circuited the usb port like how you can short circuit outlets in traditional homes. To fix the outlet, you would usually flip the switches in the circuit breaker panel.

Future Implementations:

- For future implementations, we would like to extend the functionality so that we can have online connection to a weather api and online calendar/time. This is so that we can determine the weather based on where the Arduino is or the user inputting the zipcode of the city. For the online calendar/time, we thought it would be nice to have the time/date sync with the real time zones like how our phones sync up with the date and time. This way, we avoid having to change our clocks during daylight savings and also easier to change times.

Code for Arduino 1 (Date/time):

```
/*
    Michael Tran (mtt2) - 658520892
    Final Project
*/

// include the library code:
#include <LiquidCrystal.h>
#include <TimeLib.h>

// initialize the library by associating any needed LCD interface pin
// with the arduino pin number it is connected to
const int rs = 12, en = 11, d4 = 10, d5 = 9, d6 = 8, d7 = 7;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

// Our desired quote to scroll through
String quote = "Don't cry because it's over, smile because it happened. -Dr.
Seuss";

// Amount of columns for the LCD
int columns = 16;
// Amount of rows for the LCD
int rows = 2;

// Our String that will be size columns when setup is done.
// This helps us clean the quote line (replace the screen with (# of columns)
spaces).
String cleanLine = "";

// (buttons pins)
```

```
int leftPin = 2;
int rightPin = 3;
int decrementPin = 4;
int incrementPin = 5;
int editPin = 6;

// past states of the buttons
int pastLeftPinState      = 0;
int pastRightPinState     = 0;
int pastIncrementPinState = 0;
int pastDecrementPinState = 0;
int pastEditPinState      = 0;

// Variables for Date/Time
int currMonth = 0;
int currDay = 0;
int currYear = 0;
int currHour = 0;
int currMin = 0;
int currSec = 0;

// Variables for alarm clock
int alarmMonth = 0;
int alarmDay = 0;
int alarmYear = 0;
int alarmHour = 0;
int alarmMin = 0;
int alarmSec = 0;

// Current time
int currTime = 0;

// Boolean value to determine when
// editing date/time
int mode = 0;

// const variables for our different states
// when editing our date/time on alarm clock
int currState = 0;
const int STATE_MONTH = 0;
const int STATE_DAY = 1;
const int STATE_YEAR = 2;
const int STATE_HOUR = 3;
const int STATE_MINUTE = 4;
const int STATE_SECOND = 5;

// Different data fields for date/time
String stateName[]{
  "MONTH",
  "DAY",
  "YEAR",
  "HOUR",
  "MINUTE",
  "SECOND"
```

```
};

// const variables for our different modes
// current time, editing, and alarm clock
const int MODE_TIME = 0;
const int MODE_EDIT = 1;
const int MODE_ALARM = 2;

// Different modes
String modeName[]{
  "TIME",
  "EDIT",
  "ALARM",
};

void setup() {

  Serial.begin(9600);

  // set up the LCD's number of columns and rows:
  lcd.begin(columns, rows);

  // Keep adding to the cleanLine string empty spaces so that its length size
columns
  for(int i = 0; i < columns; i++){
    cleanLine = cleanLine + " ";
  }

  // Initialize the variables to current time.
  currMonth = month();
  currDay = day();
  currYear = year();
  currHour = hour();
  currMin = minute();
  currSec = second();

}

/* This function cleans the line at col and row of LCD */
void lcdCleanLine(int col, int row){
  lcd.setCursor(col, row);
  lcd.print(cleanLine);
}

/* This function cleans the line and prints current date */
void lcdPrintDate(){
    lcdCleanLine(0, 0);
    lcd.setCursor(0, 0);
    lcd.print(currMonth);
    lcd.print("/");
    lcd.print(currDay);
    lcd.print("/");
    lcd.print(currYear);
    lcd.print(" ");
```

```
    if(mode != MODE_TIME)
      lcd.print(stateName[currState]);
}

/* This function cleans the line and prints current time */
void lcdPrintTime(){
    lcdCleanLine(0, 1);
    lcd.setCursor(0, 1);
    lcd.print(currHour);
    lcd.print(":");
    lcd.print(currMin);
    lcd.print(":");
    lcd.print(currSec);
    lcd.print(" ");

    lcd.print(modeName[mode]);
}

/* This function changes the modes on our alarm clock.
 *  The three modes are: TIME, EDIT, ALARM.
 *  TIME  - Display current time/date
 *  EDIT  - Change current time/date
 *  ALARM - Set alarm for time/date
 */
void editMode(){
    switch(mode){
      case MODE_TIME:
        mode = MODE_EDIT;
        break;
      case MODE_EDIT:
        currState = 0;
        mode = MODE_ALARM;
        setTime(currHour, currMin, currSec, currDay, currMonth, currYear);
        break;
      case MODE_ALARM:
        mode = MODE_TIME;
        currState = 0;

        // Alarm variables set
        alarmMonth = currMonth;
        alarmDay = currDay;
        alarmYear = currYear;
        alarmHour = currHour;
        alarmMin = currMin;
        alarmSec = currSec;

        // Update current time that has been ticking
        // since we paused
        currMonth = month();
        currDay = day();
        currYear = year();
        currHour = hour();
        currMin = minute();
```

```
        currSec = second();
        break;
      default:
        break;
    }
    lcdPrintDate();
    lcdPrintTime();
}

/* Function for moving data field to the "left",
 * changing year to day, day to month, etc
 */
void moveLeft(){
  if(mode != MODE_TIME){
    Serial.println("moving left");
    switch(currState){
      case STATE_MONTH:
        currState = STATE_SECOND;
        break;
      case STATE_DAY:
        currState = STATE_MONTH;
        break;
      case STATE_YEAR:
        currState = STATE_DAY;
        break;
      case STATE_HOUR:
        currState = STATE_YEAR;
        break;
      case STATE_MINUTE:
        currState = STATE_HOUR;
        break;
      case STATE_SECOND:
        currState = STATE_MINUTE;
        break;
      default:
        break;
    }
    lcdPrintDate();
  }
}

/* Function for moving data field to the "right",
 * changing month to day, day to year, etc
 */
void moveRight(){
  if(mode != MODE_TIME){
    switch(currState){
      case STATE_MONTH:
        currState = STATE_DAY;
        break;
      case STATE_DAY:
        currState = STATE_YEAR;
        break;
      case STATE_YEAR:
```

```
        currState = STATE_HOUR;
        break;
      case STATE_HOUR:
        currState = STATE_MINUTE;
        break;
      case STATE_MINUTE:
        currState = STATE_SECOND;
        break;
      case STATE_SECOND:
        currState = STATE_MONTH;
        break;
      default:
        break;
    }
    lcdPrintDate();
  }
}

/* Function for incrementing (Increment button) */
void increment(){
    if(mode != MODE_TIME){
      switch(currState){
      case STATE_MONTH:
        currMonth = (currMonth + 1 ) % 13;
        if(currMonth == 0)
          currMonth = 1;
        break;
      case STATE_DAY:
        currDay = (currDay + 1) % 32;
        if(currDay == 0)
          currDay = 1;
        break;
      case STATE_YEAR:
        currYear = (currYear + 1) % 10000;
        if(currYear == 0)
          currYear = 1000;
        break;
      case STATE_HOUR:
        currHour = (currHour + 1) % 24;
        break;
      case STATE_MINUTE:
        currMin = (currMin + 1) % 60;
        break;
      case STATE_SECOND:
        currSec = (currSec + 1) % 60;
        break;
      default:
        break;
    }
  }
  lcdPrintDate();
  lcdPrintTime();
}
```

```
/* Function for decrementing (Decrement button) */
void decrement(){
    if(mode != MODE_TIME){
      switch(currState){
      case STATE_MONTH:
        currMonth = (currMonth - 1 );
        if(currMonth == 0)
          currMonth = 12;
        break;
      case STATE_DAY:
        currDay = (currDay - 1);
        if(currDay == 0)
          currDay = 31;
        break;
      case STATE_YEAR:
        currYear = (currYear - 1);
        if(currYear == 1000)
          currYear = 10000;
        break;
      case STATE_HOUR:
        currHour = (currHour - 1);
        if(currHour == -1)
          currHour = 23;
        break;
      case STATE_MINUTE:
        currMin = (currMin - 1);
        if(currMin == -1)
          currMin = 59;
        break;
      case STATE_SECOND:
        currSec = (currSec - 1);
        if(currSec == -1)
          currSec = 59;
        break;
      default:
        break;
    }
  }

  lcdPrintDate();
  lcdPrintTime();
}

/* Function that detects the whether the button has been pressed
 *  and uses debounce to detect only one press. This applies to
 *  all five buttons
 */
void buttonPressed(){

  int leftPinState      = digitalRead(leftPin);
  int rightPinState     = digitalRead(rightPin);
  int incrementPinState = digitalRead(incrementPin);
  int decrementPinState = digitalRead(decrementPin);
  int editPinState      = digitalRead(editPin);
```

```
  delay(20);

  // Button changed for leftPin
  if(leftPinState == digitalRead(leftPin)){
    if(pastLeftPinState != leftPinState){
      if(pastLeftPinState == HIGH)
        moveLeft();
    }
    pastLeftPinState = leftPinState;
  }

  // Button changed for rightPin
  if(rightPinState == digitalRead(rightPin)){
    if(pastRightPinState != rightPinState){
      if(pastRightPinState == HIGH)
        moveRight();
    }
    pastRightPinState = rightPinState;
  }

  // Button changed for incrementPin
  if(incrementPinState == digitalRead(incrementPin)){
    if(pastIncrementPinState != incrementPinState){
      if(pastIncrementPinState == HIGH)
        increment();
    }
    pastIncrementPinState = incrementPinState;
  }

  // Button changed for decrementPin
  if(decrementPinState == digitalRead(decrementPin)){
    if(pastDecrementPinState != decrementPinState){
      if(pastDecrementPinState == HIGH)
        decrement();
    }
    pastDecrementPinState = decrementPinState;
  }

  // Button changed for editPin
  if(editPinState == digitalRead(editPin)){
    if(pastEditPinState != editPinState){
      Serial.println("Hello");
      if(editPinState == HIGH)
        editMode();
    }
    pastEditPinState = editPinState;
  }
}

/* Function returns true if alarm matches current time. False otherwise */
bool checkAlarm(){
  if(currMonth == alarmMonth &&
     currDay   == alarmDay   &&
```

```
      currYear  == alarmYear  &&
      currHour  == alarmHour  &&
      currMin   == alarmMin   &&
      currSec   == alarmSec      )
      buzzAlarm();
}

/* Function to activate the alarm (Signal other Arduino) */
void buzzAlarm(){
  // Activate alarm on other arduino
  Serial.write('y');
  Serial.print("Sent alarm!");
}

void loop() {

  // Check whether a button has been pressed
  buttonPressed();

  // Check whether to send the alarm signal
  checkAlarm();

  // Printing current time every now and then, mode is TIME
  if(currTime != now() && mode == MODE_TIME){
    currMonth = month();
    currDay = day();
    currYear = year();
    currHour = hour();
    currMin = minute();
    currSec = second();

    currTime = now();

    lcdPrintDate();
    lcdPrintTime();
  }
}

/* Function for debugging Alarm/Current time */
void debug(){
  // Print Alarm variables
  Serial.print("Alarm:");
  Serial.print(alarmMonth);
  Serial.print(" ");
  Serial.print(alarmDay);
  Serial.print(" ");
  Serial.print(alarmYear);
  Serial.println(" ");
  Serial.print(alarmHour);
  Serial.print(" ");
  Serial.print(alarmMin);
  Serial.print(" ");
  Serial.print(alarmSec);
  Serial.println();
```

```
  // Print Current time variables
  Serial.print(currMonth);
  Serial.print(" ");
  Serial.print(currDay);
  Serial.print(" ");
  Serial.print(currYear);
  Serial.println(" ");
  Serial.print(currHour);
  Serial.print(" ");
  Serial.print(currMin);
  Serial.print(" ");
  Serial.print(currSec);
  Serial.println();
}
```

Arduino 2 (Temperature/Humidity/Buzzer):

```
/*
 * Allen Breyer III - 654748297
 * Final Lab - Weather-Monitoring Alarm Clock
 */

/* READ THIS!!!
 *  For the program to compile properly in regards to the DHT11 sensor, you
need the correct libraries installed!
 *  Install the following libraries from the "Manage Libraries" manager:
 *  - Adafruit AM2320 sensor library
 *  - Adafruit Circuit Playground
 *  - Adafruit Unified Sensor
 *  - DHT sensor library
 */

#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

#include "DHT.h"
#include <TimeLib.h>

// DHT sensor using analog pin A0
#define DHTPIN A0
#define DHTTYPE DHT11

// used for temperature
float tempF;
// used for humidity
float humidity;

// Buzzer to arduino pin 8
```

```
const int BUZZER = 8;
int counter = 0;
int currTime = 0;

// Initialize DHT sensor
DHT dht(DHTPIN, DHTTYPE);

void setup()
{
  // Set up the serial communication
  Serial.begin(9600);
  // Sert up the buzzer
  pinMode(BUZZER, OUTPUT);
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  dht.begin();
}

void loop()
{
  // If it's not the current time
  if (currTime != now())
  {
    // Read temperature as Fahrenheit (isFahrenheit = true)
    tempF = dht.readTemperature(true);
    humidity = dht.readHumidity();
    counter = 0;

    // Error checking: Check if any reads failed and exit early (to try
again).
    if (isnan(humidity))
    {
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("Failed to read");
      lcd.setCursor(0,1);
      lcd.print("from DHT sensor!");
      return;
    }

    //delay(100);
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Temp: ");
    lcd.print((int) tempF);
    lcd.setCursor(8,0);
    // degrees symbol
    lcd.print((char)223);
```

```
    lcd.setCursor(9,0);
    lcd.print("F");
    lcd.setCursor(0,1);
    lcd.print("Humidity: ");
    lcd.print((int) humidity);
    lcd.setCursor(12,1);
    lcd.print("%");
    // update the current time
    currTime = now();
  }

  // If a signal is being sent through serial communication
  if(Serial.available() > 0)
  {
    char input = Serial.read();
    Serial.println(input);
    // If the correct signal is read, sound the buzzer alarm
    if (input == 'y')
    {
      tone(BUZZER, 1000);
      delay(1000);
      noTone(BUZZER);
    }
  }
}
```